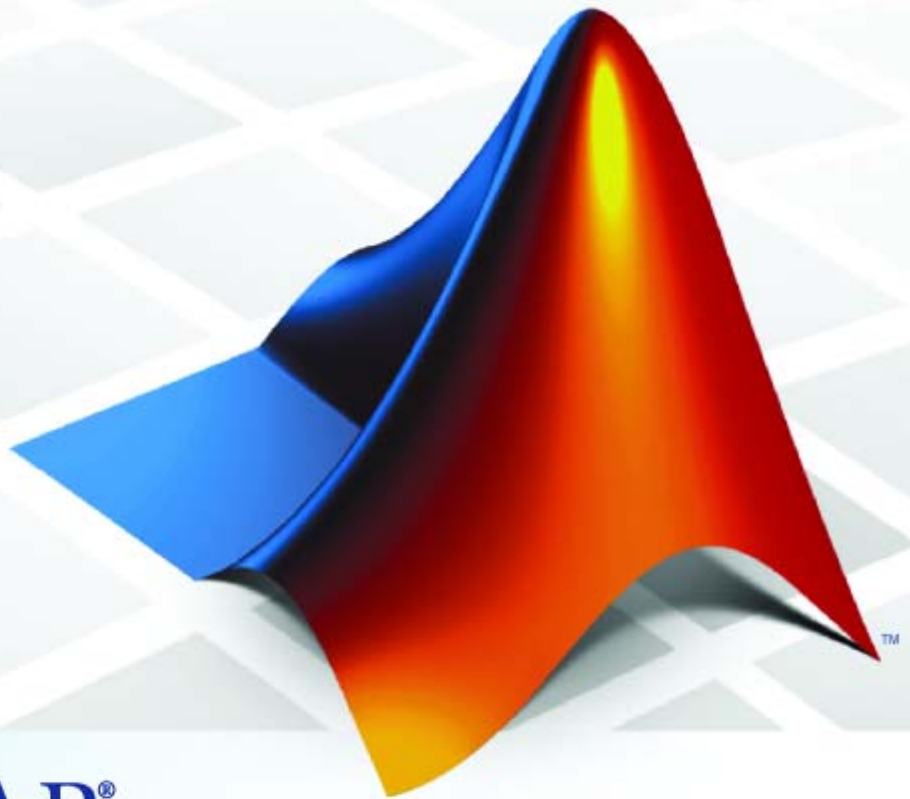


Video and Image Processing Blockset™ 3

Reference



MATLAB®
& **SIMULINK®**

How to Contact The MathWorks



www.mathworks.com Web
comp.soft-sys.matlab Newsgroup
www.mathworks.com/contact_TS.html Technical Support



suggest@mathworks.com Product enhancement suggestions
bugs@mathworks.com Bug reports
doc@mathworks.com Documentation error reports
service@mathworks.com Order status, license renewals, passcodes
info@mathworks.com Sales, pricing, and general information



508-647-7000 (Phone)



508-647-7001 (Fax)



The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

Video and Image Processing Blockset™ Reference

© COPYRIGHT 2004 –2010 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

The MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

March 2007	Online only	New for Version 2.3 (Release 2007a)
September 2007	Online only	Revised for Version 2.4 (Release 2007b)
March 2008	Online only	Revised for Version 2.5 (Release 2008a)
October 2008	Online only	Revised for Version 2.6 (Release 2008b)
March 2009	Online only	Revised for Version 2.7 (Release 2009a)
September 2009	Online only	Revised for Version 2.8 (Release 2009b)
March 2010	Online only	Revised for Version 3.0 (Release 2010a)

Block Reference

1

Analysis & Enhancement	1-2
Conversions	1-2
Filtering	1-3
Geometric Transformations	1-3
Morphological Operations	1-4
Sinks	1-4
Sources	1-5
Statistics	1-5
Text & Graphics	1-7
Transforms	1-7
Utilities	1-8

Blocks — Alphabetical List

2

System Object Reference

3

Analysis & Enhancement	3-2
Conversions	3-2
Filtering	3-3
Geometric Transformations	3-3
Morphological Operations	3-4
Sinks	3-4
Sources	3-5
Statistics	3-5
Text & Graphics	3-6
Transforms	3-6
Utilities	3-7

Alphabetical List

4

Function Reference

5

Video and Image Processing Functions 5-2

Block Reference

Analysis & Enhancement (p. 1-2)	Analyze or enhance images or video
Conversions (p. 1-2)	Perform conversion operations such as color space conversion
Filtering (p. 1-3)	Filter images or video
Geometric Transformations (p. 1-3)	Manipulate size, shape, and orientation of images or video
Morphological Operations (p. 1-4)	Perform morphological operations such as erosion and dilation
Sinks (p. 1-4)	Export or display images or video
Sources (p. 1-5)	Import images or video into Simulink
Statistics (p. 1-5)	Perform statistical operations on images or video
Text & Graphics (p. 1-7)	Annotate images or video
Transforms (p. 1-7)	Perform transform operations such as 2-D FFT and 2-D DCT
Utilities (p. 1-8)	Perform processing operations such as image padding and block processing

Analysis & Enhancement

Block Matching	Estimate motion between images or video frames
Contrast Adjustment	Adjust image contrast by linearly scaling pixel values
Corner Detection	Calculate corner metric matrix and find corners in images
Deinterlacing	Remove motion artifacts by deinterlacing input video signal
Edge Detection	Find edges of objects in images using Sobel, Prewitt, Roberts, or Canny method
Histogram Equalization	Enhance contrast of images using histogram equalization
Median Filter	Perform 2-D median filtering
Optical Flow	Estimate object velocities
Template Matching	Locate a template in an image
Trace Boundaries	Trace object boundaries in binary images

Conversions

Autothreshold	Convert intensity image to binary image
Chroma Resampling	Downsample or upsample chrominance components of images
Color Space Conversion	Convert color information between color spaces
Demosaic	Demosaic Bayer's format images

Gamma Correction	Apply or remove gamma correction from images or video streams
Image Complement	Compute complement of pixel values in binary, intensity, or RGB images
Image Data Type Conversion	Convert and scale input image to specified output data type

Filtering

2-D Convolution	Compute 2-D discrete convolution of two input matrices
2-D FIR Filter	Perform 2-D FIR filtering on input matrix
Kalman Filter	Predict or estimate states of dynamic systems
Median Filter	Perform 2-D median filtering

Geometric Transformations

Apply Geometric Transformation	Apply projective or affine transformation to an image
Estimate Geometric Transformation	Estimate geometric transformation from matching point pairs
Projective Transformation	Transform quadrilateral into another quadrilateral
Resize	Enlarge or shrink image sizes
Rotate	Rotate image by specified angle

Shear	Shift rows or columns of image by linearly varying offset
Translate	Translate image in 2-D plane using displacement vector

Morphological Operations

Bottom-hat	Perform bottom-hat filtering on intensity or binary images
Closing	Perform morphological closing on binary or intensity images
Dilation	Find local maxima in binary or intensity images
Erosion	Find local minima in binary or intensity images
Label	Label connected components in binary images
Opening	Perform morphological opening on binary or intensity images
Top-hat	Perform top-hat filtering on intensity or binary images

Sinks

Frame Rate Display	Calculate average update rate of input signal
To Multimedia File	Write video frames and audio samples to multimedia file
To Video Display	Display video data

Video To Workspace	Export video signal to MATLAB® workspace
Video Viewer	Display binary, intensity, or RGB images or video streams
Write AVI File (Obsolete)	Write video frames to uncompressed AVI file
Write Binary File	Write binary video data to files

Sources

From Multimedia File	Read video frames and audio samples from compressed multimedia file
Image From File	Import image from image file
Image From Workspace	Import image from MATLAB workspace
Read Binary File	Read binary video data from files
Video From Workspace	Import video signal from MATLAB workspace

Statistics

2-D Autocorrelation	Compute 2-D autocorrelation of input matrix
2-D Correlation	Compute 2-D cross-correlation of two input matrices
2-D Histogram (Obsolete)	Generate histogram of each input matrix
2-D Mean (Obsolete)	Find mean value of each input matrix

2-D Median (Obsolete)	Find median value of each input matrix
2-D Standard Deviation (Obsolete)	Find standard deviation of each input matrix
2-D Variance (Obsolete)	Compute variance of each input matrix
Blob Analysis	Compute statistics for labeled regions
Find Local Maxima	Find local maxima in matrices
Histogram	Generate histogram of each input matrix
Maximum	Find maximum values in input or sequence of inputs
Mean	Find mean value of each input matrix
Median	Find median value of each input matrix
Minimum	Find minimum values in input or sequence of inputs
PSNR	Compute peak signal-to-noise ratio (PSNR) between images
Standard Deviation	Find standard deviation of each input matrix
Variance	Compute variance of input or sequence of inputs

Text & Graphics

Compositing	Combine pixel values of two images, overlay one image over another, or highlight selected pixels
Draw Markers	Draw markers by embedding predefined shapes on output image
Draw Shapes	Draw rectangles, lines, polygons, or circles on images
Insert Text	Draw text on image or video stream.

Transforms

2-D DCT	Compute 2-D discrete cosine transform (DCT)
2-D FFT	Compute 2-D FFT of input
2-D IDCT	Compute 2-D inverse discrete cosine transform (IDCT)
2-D IFFT	Compute 2-D IFFT of input
Gaussian Pyramid	Perform Gaussian pyramid decomposition
Hough Lines	Find Cartesian coordinates of lines described by rho and theta pairs
Hough Transform	Find lines in images

Utilities

Block Processing

Repeat user-specified operation on submatrices of input matrix

Image Pad

Pad signal along its rows, columns, or both

Variable Selector

Specify subset of rows or columns from input

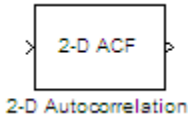
Blocks — Alphabetical List

2-D Autocorrelation

Purpose Compute 2-D autocorrelation of input matrix

Library Statistics

Description The 2-D Autocorrelation block computes the two-dimensional autocorrelation of the input matrix. Assume that input matrix A has dimensions (Ma, Na). The equation for the two-dimensional discrete autocorrelation is



$$C(i, j) = \sum_{m=0}^{(Ma-1)} \sum_{n=0}^{(Na-1)} A(m, n) \cdot \text{conj}(A(m+i, n+j))$$

where $0 \leq i < 2Ma - 1$ and $0 \leq j < 2Na - 1$.

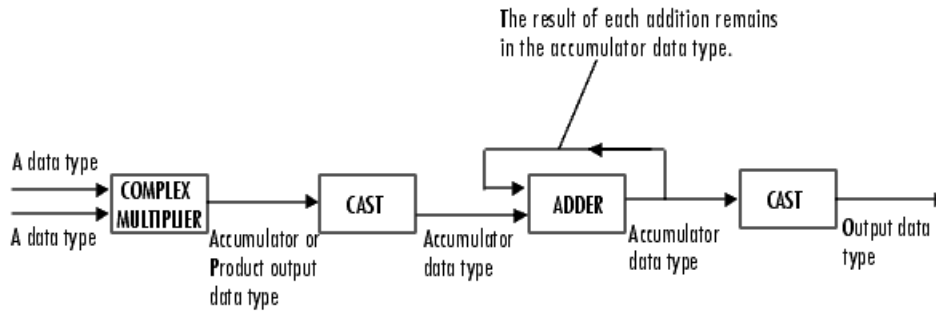
The output of this block has dimensions $(2Ma - 1, 2Na - 1)$.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values or a scalar, vector, or matrix that represents one plane of the RGB video stream	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	Yes
Output	Autocorrelation of the input matrix	Same as Input port	Yes

If the data type of the input is floating point, the output of the block has the same data type.

Fixed-Point Data Types

The following diagram shows the data types used in the 2-D Autocorrelation block for fixed-point signals.



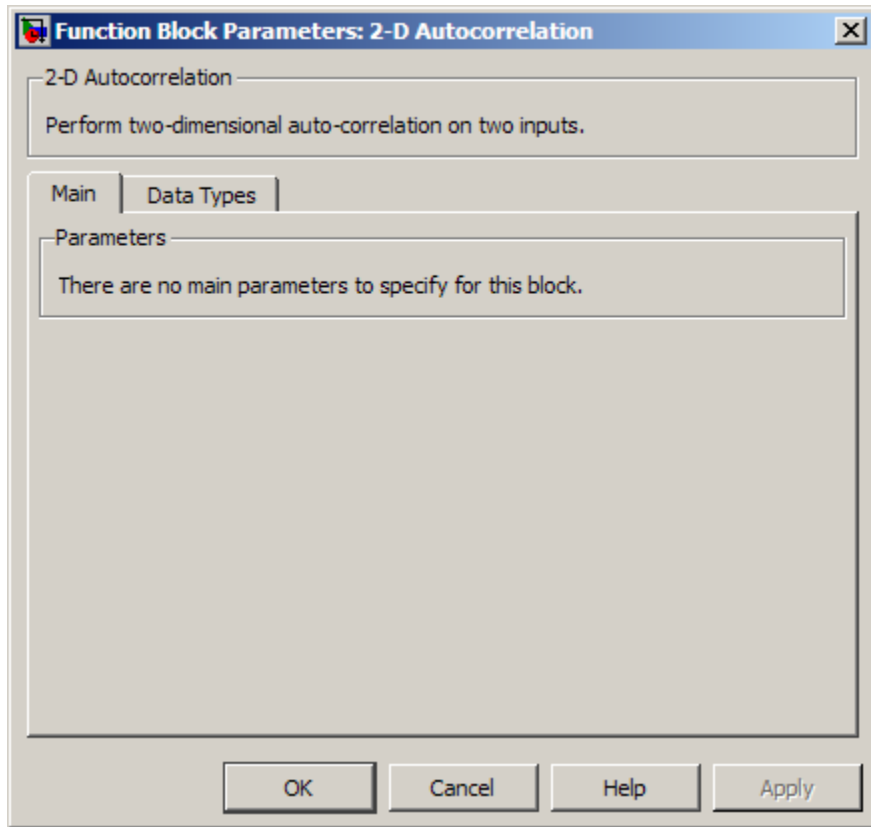
You can set the product output, accumulator, and output data types in the block mask as discussed in “Dialog Box” on page 2-4.

The output of the multiplier is in the product output data type if at least one of the inputs to the multiplier is real. If both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, refer to “Multiplication Data Types” in the Signal Processing Blockset™ documentation.

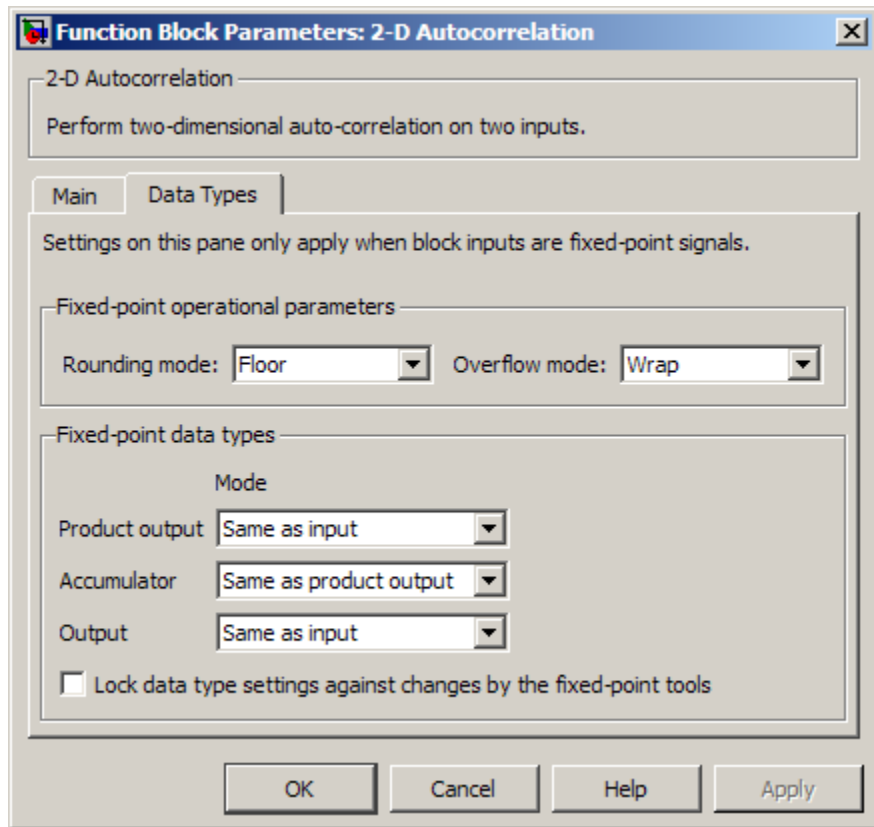
2-D Autocorrelation

Dialog Box

The **Main** pane of the 2-D Autocorrelation dialog box appears as shown in the following figure.



The **Data Types** pane of the 2-D Autocorrelation dialog box appears as shown in the following figure.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how to designate the product output word and fraction lengths. Refer to "Fixed-Point Data Types" on page 2-2 and "Multiplication Data Types" in the Signal Processing

2-D Autocorrelation

Blockset documentation for illustrations depicting the use of the product output data type in this block:

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset™ software is 0.

Accumulator

Use this parameter to specify how to designate the accumulator word and fraction lengths. Refer to “Fixed-Point Data Types” on page 2-2 and “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block. The accumulator data type is only used when both inputs to the multiplier are complex.

- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset software is 0.

Output

Choose how to specify the output word length and fraction length.

- When you select `Same as input`, these characteristics match those of the input to the block.

- When you select **Binary point scaling**, you can enter the word length and the fraction length of the output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the Video and Image Processing Blockset software is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink® documentation.

See Also

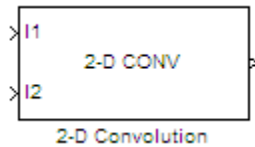
2-D Correlation	Video and Image Processing Blockset software
Histogram	Video and Image Processing Blockset software
Mean	Video and Image Processing Blockset software
Median	Video and Image Processing Blockset software
Standard Deviation	Video and Image Processing Blockset software
Variance	Video and Image Processing Blockset software
Maximum	Signal Processing Blockset software
Minimum	Signal Processing Blockset software

2-D Convolution

Purpose Compute 2-D discrete convolution of two input matrices

Library Filtering
vipfilter

Description



The 2-D Convolution block computes the two-dimensional convolution of two input matrices. Assume that matrix A has dimensions (M_a, N_a) and matrix B has dimensions (M_b, N_b) . When the block calculates the full output size, the equation for the 2-D discrete convolution is

$$C(i, j) = \sum_{m=0}^{(M_a-1)} \sum_{n=0}^{(N_a-1)} A(m, n) * B(i-m, j-n)$$

where $0 \leq i < M_a + M_b - 1$ and $0 \leq j < N_a + N_b - 1$.

Port	Input/Output	Supported Data Types	Complex Values Supported
I1	Matrix of intensity values or a matrix that represents one plane of the RGB video stream	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer	Yes
I2	Matrix of intensity values or a matrix that	Same as I1 port	Yes

Port	Input/Output	Supported Data Types	Complex Values Supported
	represents one plane of the RGB video stream		
Output	Convolution of the input matrices	Same as I1 port	Yes

If the data type of the input is floating point, the output of the block has the same data type.

The dimensions of the output are dictated by the **Output size** parameter. Assume that the input at port I1 has dimensions (Ma, Na) and the input at port I2 has dimensions (Mb, Nb). If, for the **Output size** parameter, you choose **Full**, the output is the full two-dimensional convolution with dimensions (Ma+Mb-1, Na+Nb-1). If, for the **Output size** parameter, you choose **Same as input port I1**, the output is the central part of the convolution with the same dimensions as the input at port I1. If, for the **Output size** parameter, you choose **Valid**, the output is only those parts of the convolution that are computed without the zero-padded edges of any input. This output has dimensions (Ma-Mb+1, Na-Nb+1). However, if $\text{all}(\text{size}(I1) < \text{size}(I2))$, the block errors out.

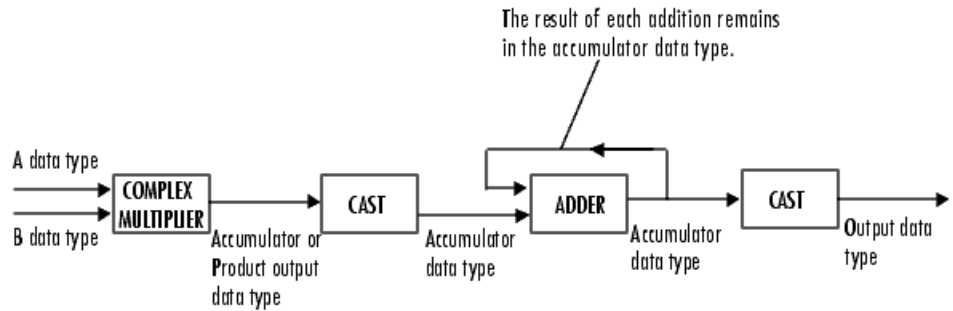
If you select the **Output normalized convolution** check box, the block's output is divided by $\sqrt{\text{sum}(\text{dot}(I1p, I1p)) * \text{sum}(\text{dot}(I2, I2))}$, where I1p is the portion of the I1 matrix that aligns with the I2 matrix. See "Example 2" on page 2-12 for more information.

Note When you select the **Output normalized convolution** check box, the block input cannot be fixed point.

Fixed-Point Data Types

The following diagram shows the data types used in the 2-D Convolution block for fixed-point signals.

2-D Convolution



You can set the product output, accumulator, and output data types in the block mask as discussed in “Dialog Box” on page 2-15.

The output of the multiplier is in the product output data type if at least one of the inputs to the multiplier is real. If both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, refer to “Multiplication Data Types” in the Signal Processing Blockset documentation.

Examples

Example 1

Suppose $I1$, the first input matrix, has dimensions (4,3) and $I2$, the second input matrix, has dimensions (2,2). If, for the **Output size** parameter, you choose **Full**, the block uses the following equations to determine the number of rows and columns of the output matrix:

$$C_{\text{full}_{\text{rows}}} = I1_{\text{rows}} + I2_{\text{rows}} - 1 = 5$$

$$C_{\text{full}_{\text{columns}}} = I1_{\text{columns}} + I2_{\text{columns}} - 1 = 4$$

The resulting matrix is

$$C_{\text{full}} = \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \\ c_{40} & c_{41} & c_{42} & c_{43} \end{bmatrix}$$

If, for the **Output size** parameter, you choose **Same** as input port I1, the output is the central part of C_{full} with the same dimensions as the input at port I1, (4,3). However, since a 4-by-3 matrix cannot be extracted from the exact center of C_{full} , the block leaves more rows and columns on the top and left side of the C_{full} matrix and outputs:

$$C_{\text{same}} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \\ c_{41} & c_{42} & c_{43} \end{bmatrix}$$

If, for the **Output size** parameter, you choose **Valid**, the block uses the following equations to determine the number of rows and columns of the output matrix:

$$C_{\text{valid}_{\text{rows}}} = I1_{\text{rows}} - I2_{\text{rows}} + 1 = 3$$

2-D Convolution

$$C_{\text{valid_columns}} = I1_{\text{columns}} - I2_{\text{columns}} + 1 = 2$$

In this case, it is always possible to extract the exact center of C_{full} .
Therefore, the block outputs

$$C_{\text{full}} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix}$$

Example 2

In convolution, the value of an output element is computed as a weighted sum of neighboring elements.

For example, suppose the first input matrix represents an image and is defined as

$$I1 = \begin{bmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{bmatrix}$$

The second input matrix also represents an image and is defined as

$$I2 = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

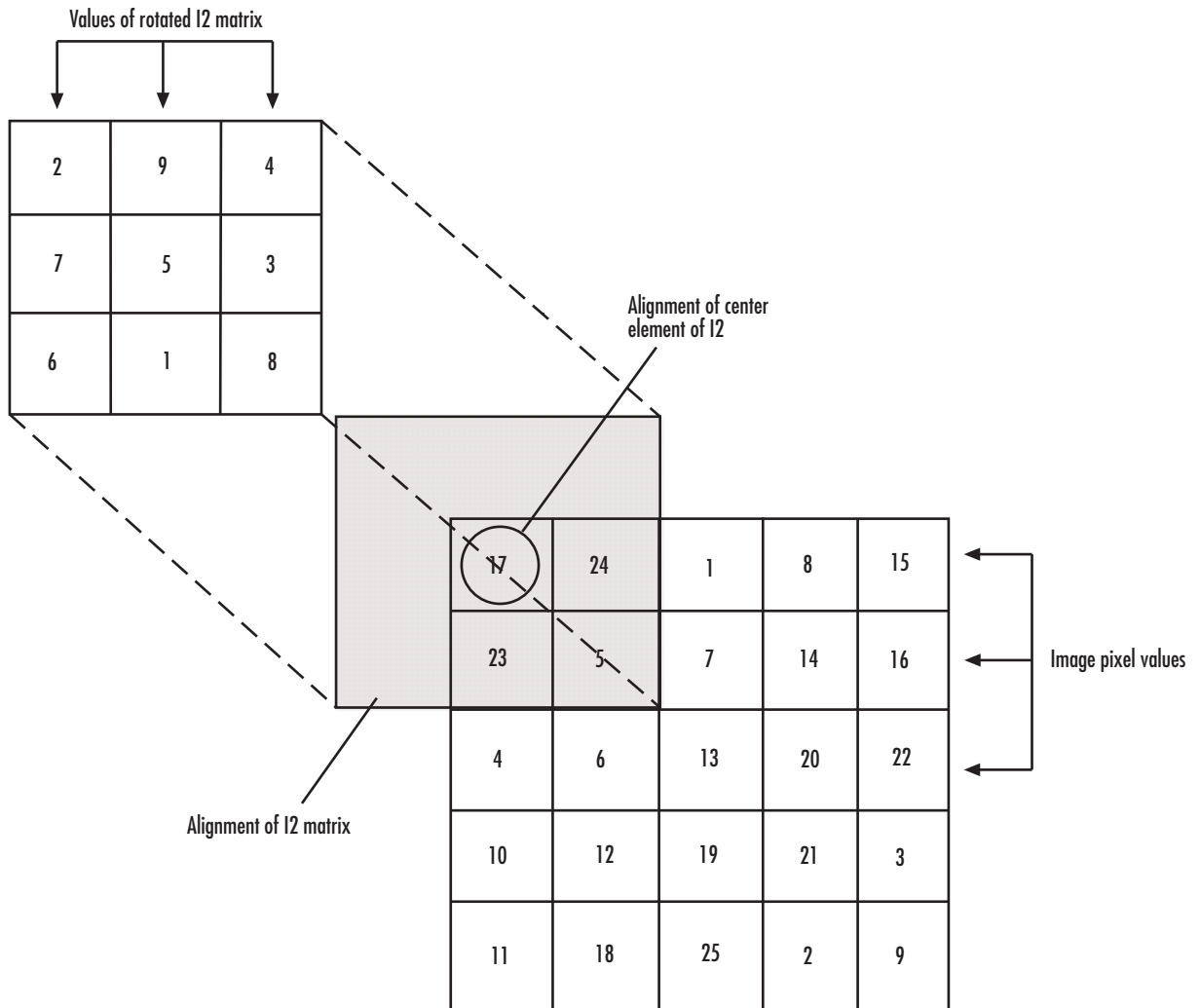
The following figure shows how to compute the (1,1) output element (zero-based indexing) using these steps:

- 1** Rotate the second input matrix, I2, 180 degrees about its center element.
- 2** Slide the center element of I2 so that it lies on top of the (0,0) element of I1.
- 3** Multiply each element of the rotated I2 matrix by the element of I1 underneath.
- 4** Sum the individual products from step 3.

Hence the (1,1) output element is

$$0 \cdot 2 + 0 \cdot 9 + 0 \cdot 4 + 0 \cdot 7 + 17 \cdot 5 + 24 \cdot 3 + 0 \cdot 6 + 23 \cdot 1 + 5 \cdot 8 = 220 .$$

2-D Convolution

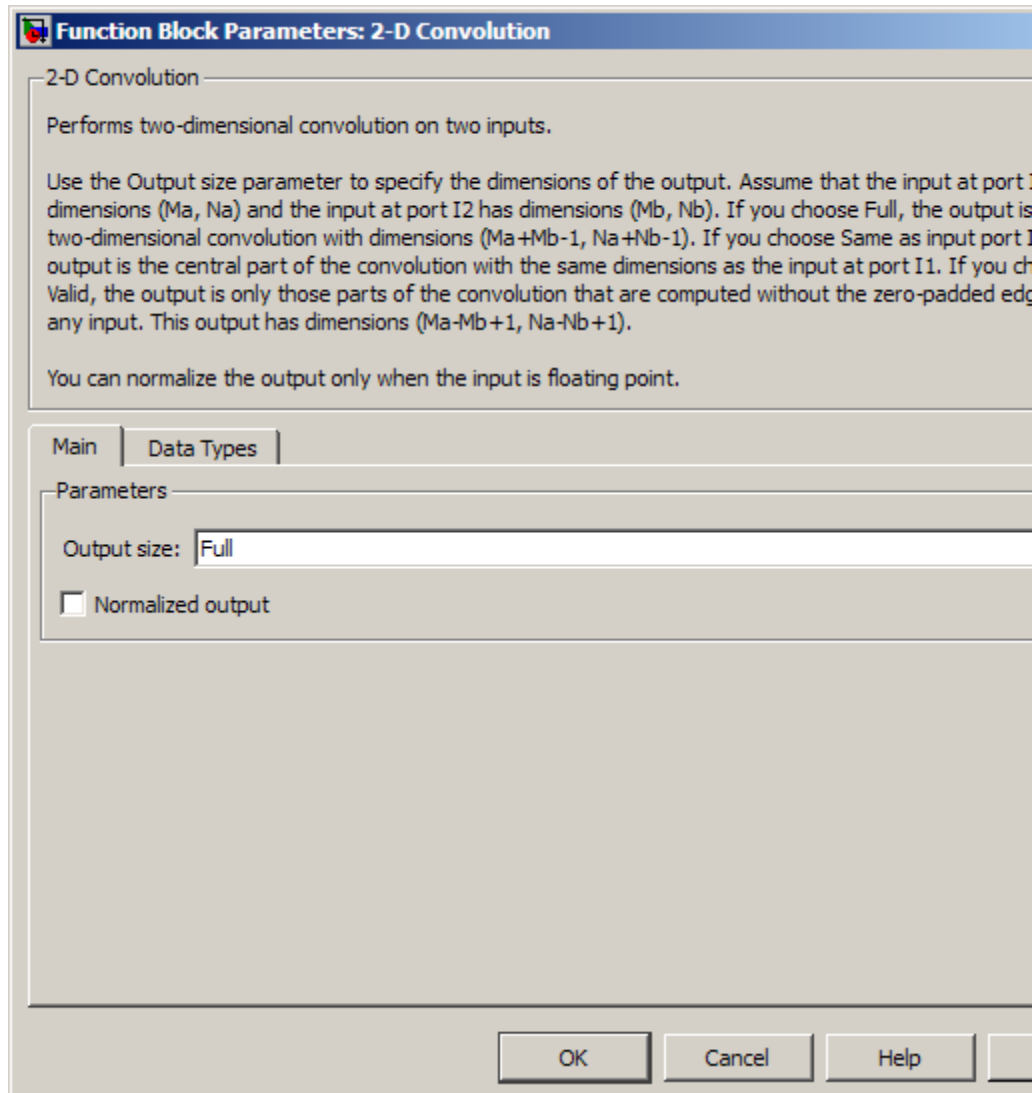


Computing the (1,1) Output of Convolution

The normalized convolution of the (1,1) output element is $220/\sqrt{\text{sum}(\text{dot}(I1p, I1p)) * \text{sum}(\text{dot}(I2, I2))} = 0.3459$, where $I1p = [0 \ 0 \ 0; 0 \ 17 \ 24; 0 \ 23 \ 5]$.

Dialog Box

The **Main** pane of the 2-D Convolution dialog box appears as shown in the following figure.



2-D Convolution

Output size

This parameter controls the size of the output scalar, vector, or matrix produced as a result of the convolution between the two inputs. If you choose `Full`, the output has dimensions (M_a+M_b-1, N_a+N_b-1) . If you choose `Same as input port I1`, the output has the same dimensions as the input at port I1. If you choose `Valid`, output has dimensions (M_a-M_b+1, N_a-N_b+1) .

Output normalized convolution

If you select this check box, the block's output is normalized.

The **Data Types** pane of the 2-D Convolution dialog box appears as shown in the following figure.

Function Block Parameters: 2-D Convolution

2-D Convolution

Performs two-dimensional convolution on two inputs.

Use the Output size parameter to specify the dimensions of the output. Assume that the input at port I1 has dimensions (M_a, N_a) and the input at port I2 has dimensions (M_b, N_b) . If you choose Full, the output is the two-dimensional convolution with dimensions (M_a+M_b-1, N_a+N_b-1) . If you choose Same as input port I1, the output is the central part of the convolution with the same dimensions as the input at port I1. If you choose Valid, the output is only those parts of the convolution that are computed without the zero-padded edges of any input. This output has dimensions (M_a-M_b+1, N_a-N_b+1) .

You can normalize the output only when the input is floating point.

Main | Data Types

Settings on this pane only apply when block inputs are fixed-point signals.

Fixed-point operational parameters

Rounding mode: Floor Overflow mode: Wrap

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Product output	Binary point scaling	Inherit	32	10
Accumulator	Same as product output			
Output	Binary point scaling	Same as input	32	12

Lock data type settings against changes by the fixed-point tools

OK Cancel Help

2-D Convolution

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how to designate the product output word and fraction lengths. Refer to “Fixed-Point Data Types” on page 2-9 and “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the product output data type in this block:

- When you select `Same as first input`, these characteristics match those of the first input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset software is 0.

The Product Output inherits its sign according to the inputs. If either or both input **I1** and **I2** are signed, the Product Output will be signed. Otherwise, the Product Output is unsigned. The following table shows all cases.

Sign of Input I1	Sign of Input I2	Sign of Product Output
unsigned	unsigned	unsigned
unsigned	signed	signed
signed	unsigned	signed
signed	signed	signed

Accumulator

Use this parameter to specify how to designate the accumulator word and fraction lengths. Refer to “Fixed-Point Data Types” on page 2-9 and “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block. The accumulator data type is only used when both inputs to the multiplier are complex:

- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as first input`, these characteristics match those of the first input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset software is 0.

Output

Choose how to specify the word length and fraction length of the output of the block:

- When you select `Same as first input`, these characteristics match those of the first input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the Video and Image Processing Blockset software is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more

2-D Convolution

information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

See Also

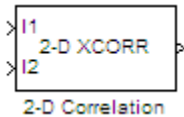
2-D FIR Filter

Video and Image Processing Blockset software

Purpose Compute 2-D cross-correlation of two input matrices

Library Statistics
vipstatistics

Description



The 2-D Correlation block computes the two-dimensional cross-correlation of two input matrices. Assume that matrix A has dimensions (M_a, N_a) and matrix B has dimensions (M_b, N_b) . When the block calculates the full output size, the equation for the two-dimensional discrete cross-correlation is

$$C(i, j) = \sum_{m=0}^{(M_a-1)} \sum_{n=0}^{(N_a-1)} A(m, n) \cdot \text{conj}(B(m+i, n+j))$$

where $0 \leq i < M_a + M_b - 1$ and $0 \leq j < N_a + N_b - 1$.

Port	Input/Output	Supported Data Types	Complex Values Supported
I1	Vector or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	Yes
I2	Scalar, vector, or matrix of intensity values or a scalar, vector, or matrix that represents one plane of the RGB video stream	Same as I1 port	Yes
Output	Convolution of the input matrices	Same as I1 port	Yes

2-D Correlation

If the data type of the input is floating point, the output of the block is the same data type.

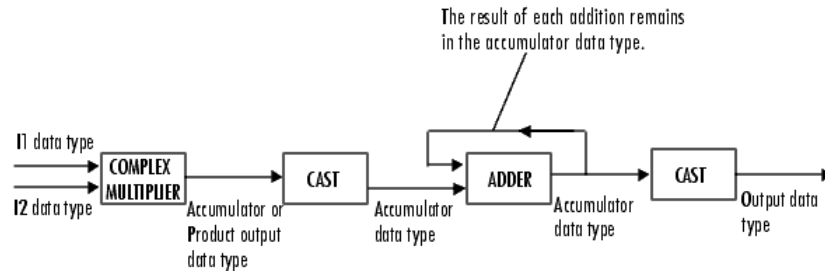
The dimensions of the output are dictated by the **Output size** parameter and the sizes of the inputs at ports I1 and I2. For example, assume that the input at port I1 has dimensions (Ma, Na) and the input at port I2 has dimensions (Mb, Nb). If, for the **Output size** parameter, you choose **Full**, the output is the full two-dimensional cross-correlation with dimensions (Ma+Mb-1, Na+Nb-1). If, for the **Output size** parameter, you choose **Same as input port I1**, the output is the central part of the cross-correlation with the same dimensions as the input at port I1. If, for the **Output size** parameter, you choose **Valid**, the output is only those parts of the cross-correlation that are computed without the zero-padded edges of any input. This output has dimensions (Ma-Mb+1, Na-Nb+1). However, if `all(size(I1) < size(I2))`, the block errors out.

If you select the **Normalized output** check box, the block's output is divided by $\sqrt{\text{sum}(\text{dot}(I1p, I1p)) * \text{sum}(\text{dot}(I2, I2))}$, where *I1p* is the portion of the I1 matrix that aligns with the I2 matrix. See “Example 2” on page 2-25 for more information.

Note When you select the **Normalized output** check box, the block input cannot be fixed point.

Fixed-Point Data Types

The following diagram shows the data types used in the 2-D Correlation block for fixed-point signals.



You can set the product output, accumulator, and output data types in the block mask as discussed in “Dialog Box” on page 2-29.

The output of the multiplier is in the product output data type if at least one of the inputs to the multiplier is real. If both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, refer to “Multiplication Data Types” in the Signal Processing Blockset documentation.

Examples

Example 1

Suppose $I1$, the first input matrix, has dimensions (4,3). $I2$, the second input matrix, has dimensions (2,2). If, for the **Output size** parameter, you choose **Full**, the block uses the following equations to determine the number of rows and columns of the output matrix:

$$C_{\text{full_rows}} = I1_{\text{rows}} + I2_{\text{rows}} - 1 = 4 + 2 - 1 = 5$$

$$C_{\text{full_columns}} = I1_{\text{columns}} + I2_{\text{columns}} - 1 = 3 + 2 - 1 = 4$$

The resulting matrix is

2-D Correlation

$$C_{\text{full}} = \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \\ c_{40} & c_{41} & c_{42} & c_{43} \end{bmatrix}$$

If, for the **Output size** parameter, you choose Same as input port I1, the output is the central part of C_{full} with the same dimensions as the input at port I1, (4,3). However, since a 4-by-3 matrix cannot be extracted from the exact center of C_{full} , the block leaves more rows and columns on the top and left side of the C_{full} matrix and outputs:

$$C_{\text{same}} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \\ c_{41} & c_{42} & c_{43} \end{bmatrix}$$

If, for the **Output size** parameter, you choose Valid, the block uses the following equations to determine the number of rows and columns of the output matrix:

$$C_{\text{valid}_{\text{rows}}} = I1_{\text{rows}} - I2_{\text{rows}} + 1 = 3$$

$$C_{\text{valid}_{\text{columns}}} = I1_{\text{columns}} - I2_{\text{columns}} + 1 = 2$$

In this case, it is always possible to extract the exact center of C_{full} . Therefore, the block outputs

$$C_{full} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix}$$

Example 2

In cross-correlation, the value of an output element is computed as a weighted sum of neighboring elements.

For example, suppose the first input matrix represents an image and is defined as

$$I1 = \begin{bmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{bmatrix}$$

The second input matrix also represents an image and is defined as

$$I2 = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

The following figure shows how to compute the (2,4) output element (zero-based indexing) using these steps:

- 1 Slide the center element of I2 so that lies on top of the (1,3) element of I1.

2-D Correlation

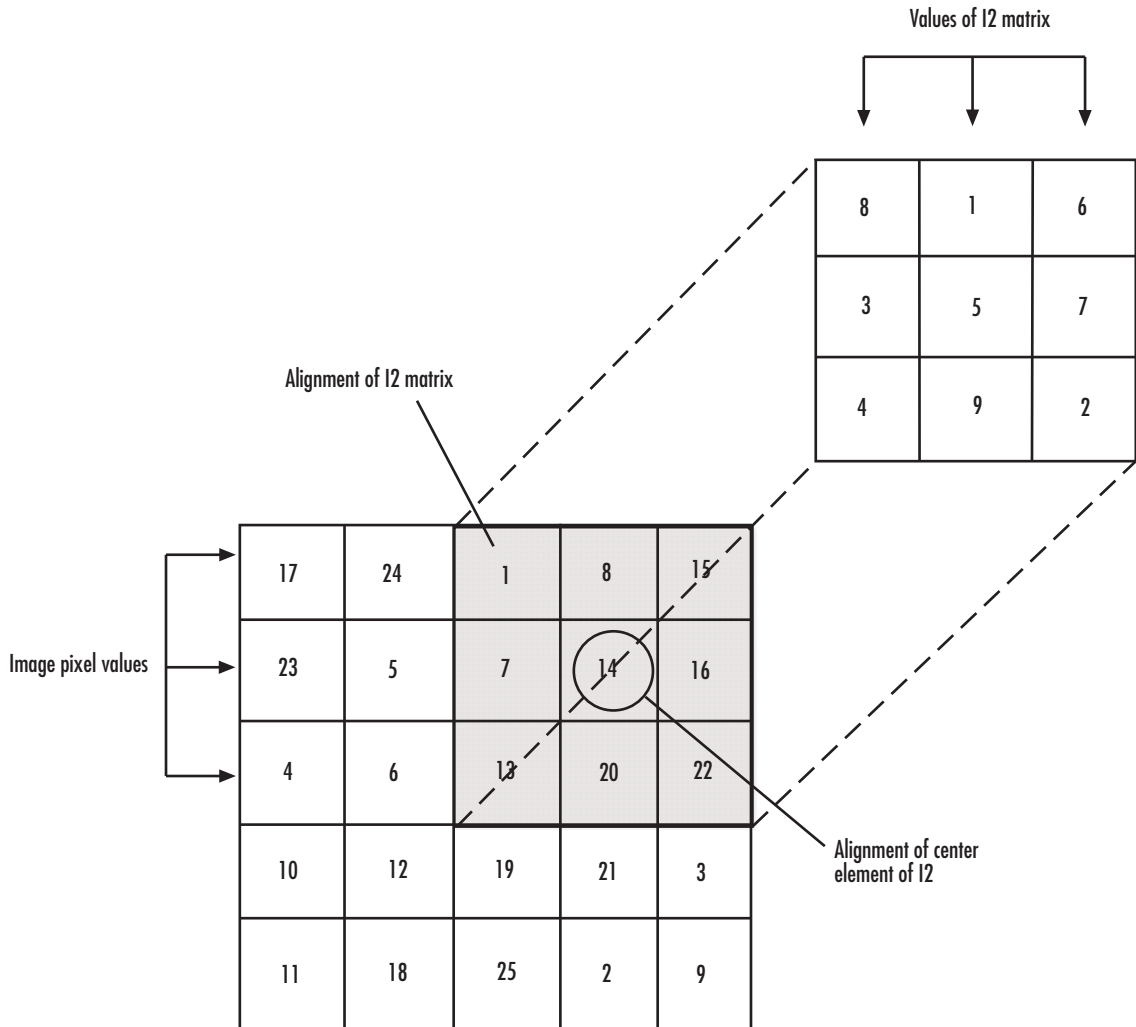
2 Multiply each weight in I2 by the element of I1 underneath.

3 Sum the individual products from step 2.

The (2,4) output element from the cross-correlation is

$$1 \cdot 8 + 8 \cdot 1 + 15 \cdot 6 + 7 \cdot 3 + 14 \cdot 5 + 16 \cdot 7 + 13 \cdot 4 + 20 \cdot 9 + 22 \cdot 2 = 585 .$$

2-D Correlation



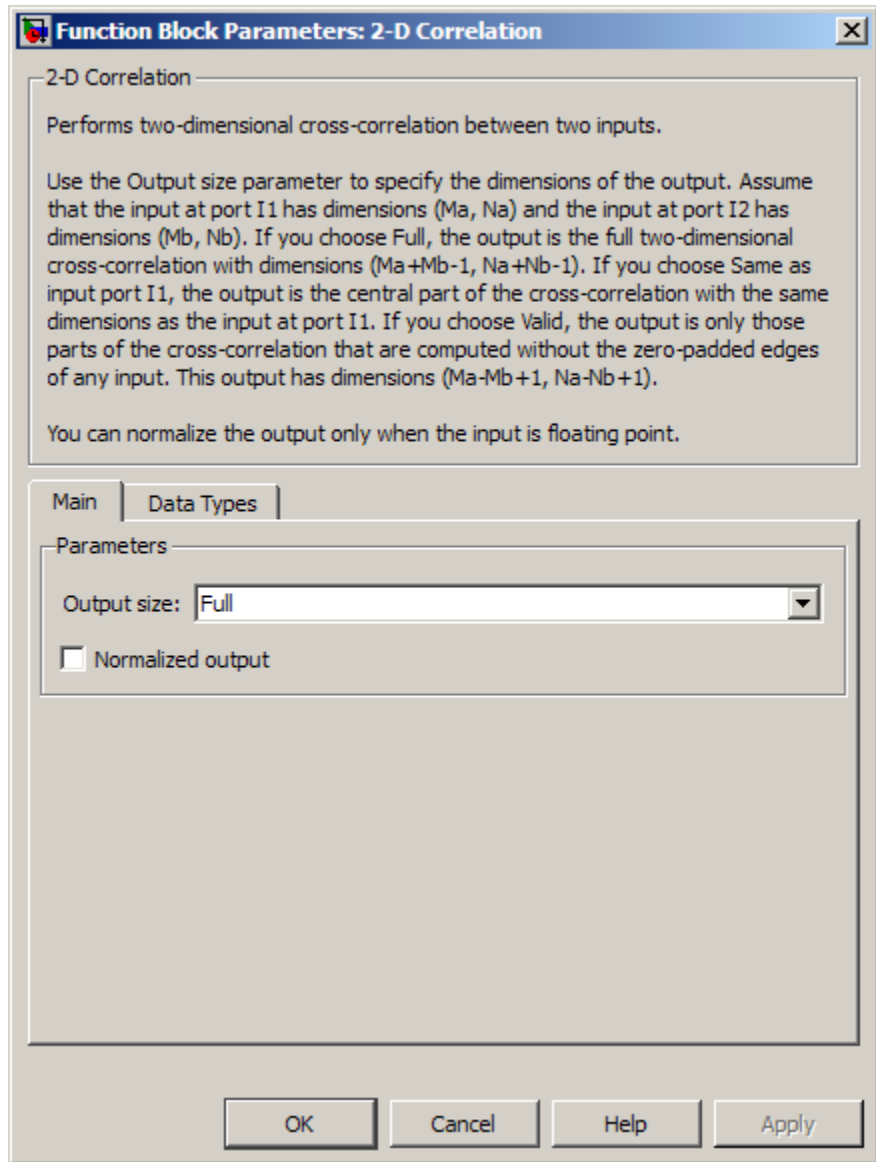
Computing the (2,4) Output of Cross-Correlation

2-D Correlation

The normalized cross-correlation of the (2,4) output element is $585/\sqrt{\text{sum}(\text{dot}(I1p, I1p)) * \text{sum}(\text{dot}(I2, I2))} = 0.8070$, where $I1p = [1 \ 8 \ 15; 7 \ 14 \ 16; 13 \ 20 \ 22]$.

Dialog Box

The **Main** pane of the 2-D Correlation dialog box appears as shown in the following figure.



2-D Correlation

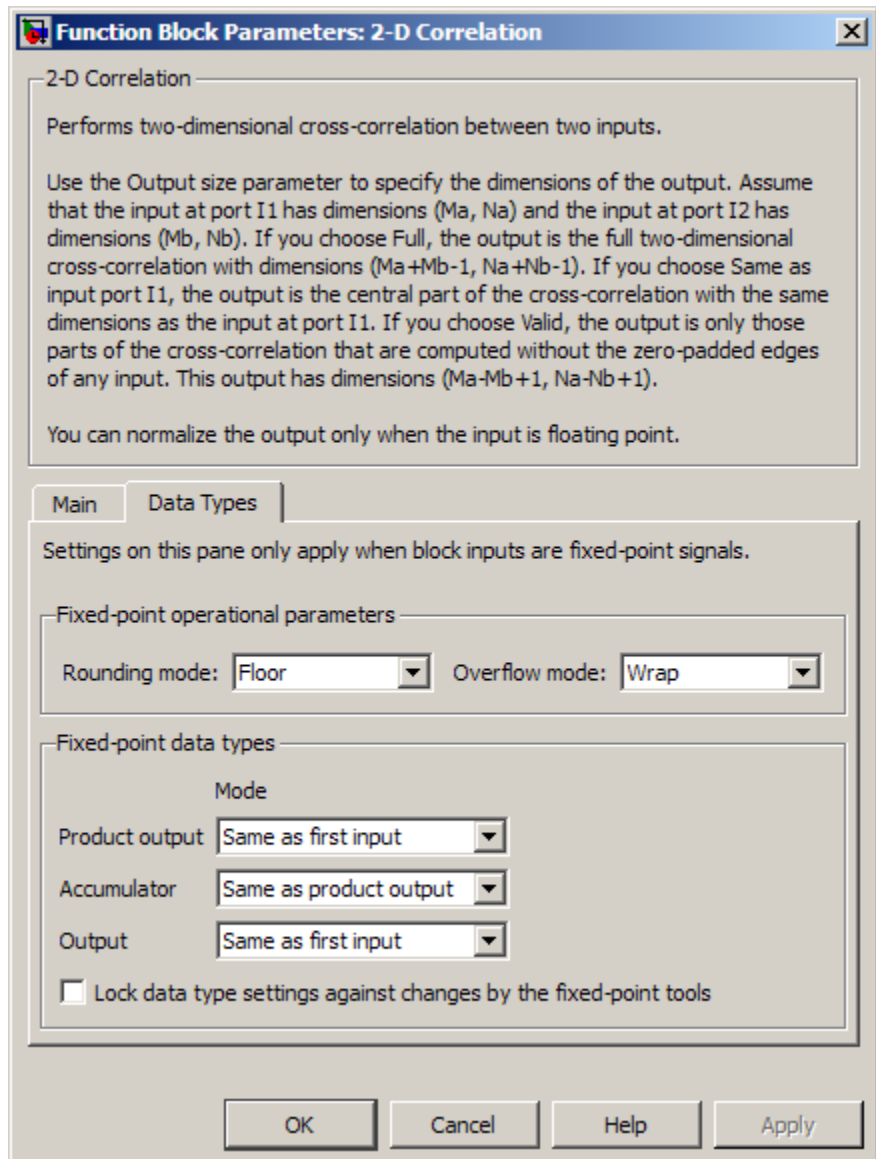
Output size

This parameter controls the size of the output scalar, vector, or matrix produced as a result of the cross-correlation between the two inputs. If you choose `Full`, the output has dimensions (M_a+M_b-1, N_a+N_b-1) . If you choose `Same as input port I1`, the output has the same dimensions as the input at port I1. If you choose `Valid`, output has dimensions (M_a-M_b+1, N_a-N_b+1) .

Normalized output

If you select this check box, the block's output is normalized.

The **Data Types** pane of the 2-D Correlation dialog box appears as shown in the following figure.



2-D Correlation

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how to designate the product output word and fraction lengths. Refer to “Fixed-Point Data Types” on page 2-22 and “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the product output data type in this block:

- When you select `Same as first input`, these characteristics match those of the first input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset software is 0.

The Product Output inherits its sign according to the inputs. If either or both input **I1** and **I2** are signed, the Product Output will be signed. Otherwise, the Product Output is unsigned. The table below show all cases.

Sign of Input I1	Sign of Input I2	Sign of Product Output
unsigned	unsigned	unsigned
unsigned	signed	signed
signed	unsigned	signed
signed	signed	signed

Accumulator

Use this parameter to specify how to designate the accumulator word and fraction lengths. Refer to “Fixed-Point Data Types” on page 2-22 and “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block. The accumulator data type is only used when both inputs to the multiplier are complex:

- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as first input`, these characteristics match those of the first input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset software is 0.

Output

Choose how to specify the word length and fraction length of the output of the block:

- When you select `Same as first input`, these characteristics match those of the first input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the Video and Image Processing Blockset software is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more

2-D Correlation

information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

See Also

2-D Autocorrelation	Video and Image Processing Blockset software
Histogram	Video and Image Processing Blockset software
Mean	Video and Image Processing Blockset software
Median	Video and Image Processing Blockset software
Standard Deviation	Video and Image Processing Blockset software
Variance	Video and Image Processing Blockset software
Maximum	Signal Processing Blockset software
Minimum	Signal Processing Blockset software

Purpose Compute 2-D discrete cosine transform (DCT)

Library Transforms
viptransforms

Description The 2-D DCT block calculates the two-dimensional discrete cosine transform of the input signal. The equation for the two-dimensional DCT is



$$F(m,n) = \frac{2}{\sqrt{MN}} C(m)C(n) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cos \frac{(2x+1)m\pi}{2M} \cos \frac{(2y+1)n\pi}{2N}$$

where $C(m), C(n) = 1/\sqrt{2}$ for $m, n = 0$ and $C(m), C(n) = 1$ otherwise.

The number of rows and columns of the input signal must be powers of two. The output of this block has dimensions the same dimensions as the input.

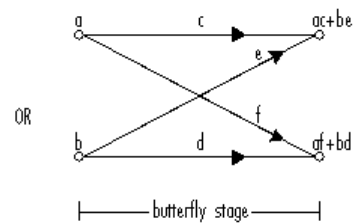
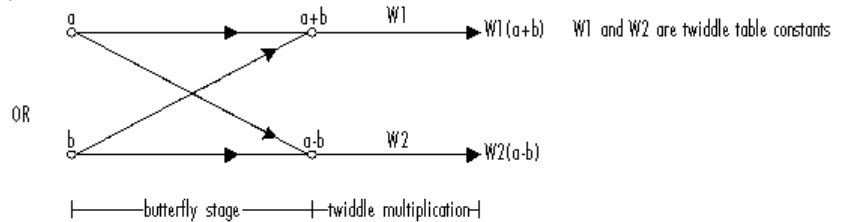
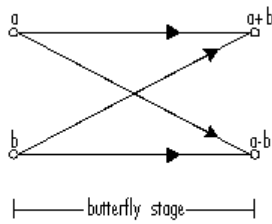
Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	No
Output	2-D DCT of the input	Same as Input port	No

If the data type of the input signal is floating point, the output of the block is the same data type.

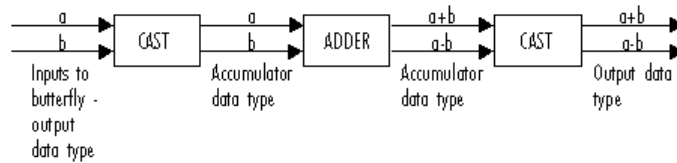
Use the **Sine and cosine computation** parameter to specify how the block computes the sine and cosine terms in the DCT algorithm. If you select `Trigonometric fcn`, the block computes the sine and cosine values during the simulation. If you select `Table lookup`, the block computes and stores the trigonometric values before the simulation starts. In this case, the block requires extra memory.

Fixed-Point Data Types

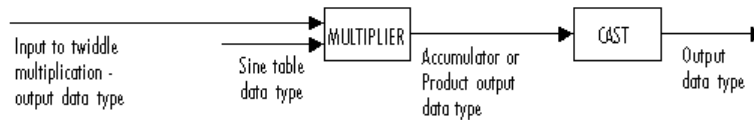
The following diagram shows the data types used in the 2-D DCT block for fixed-point signals. Inputs are first cast to the output data type and stored in the output buffer. Each butterfly stage processes signals in the accumulator data type, with the final output of the butterfly being cast back into the output data type.



Butterfly Stage Data Types



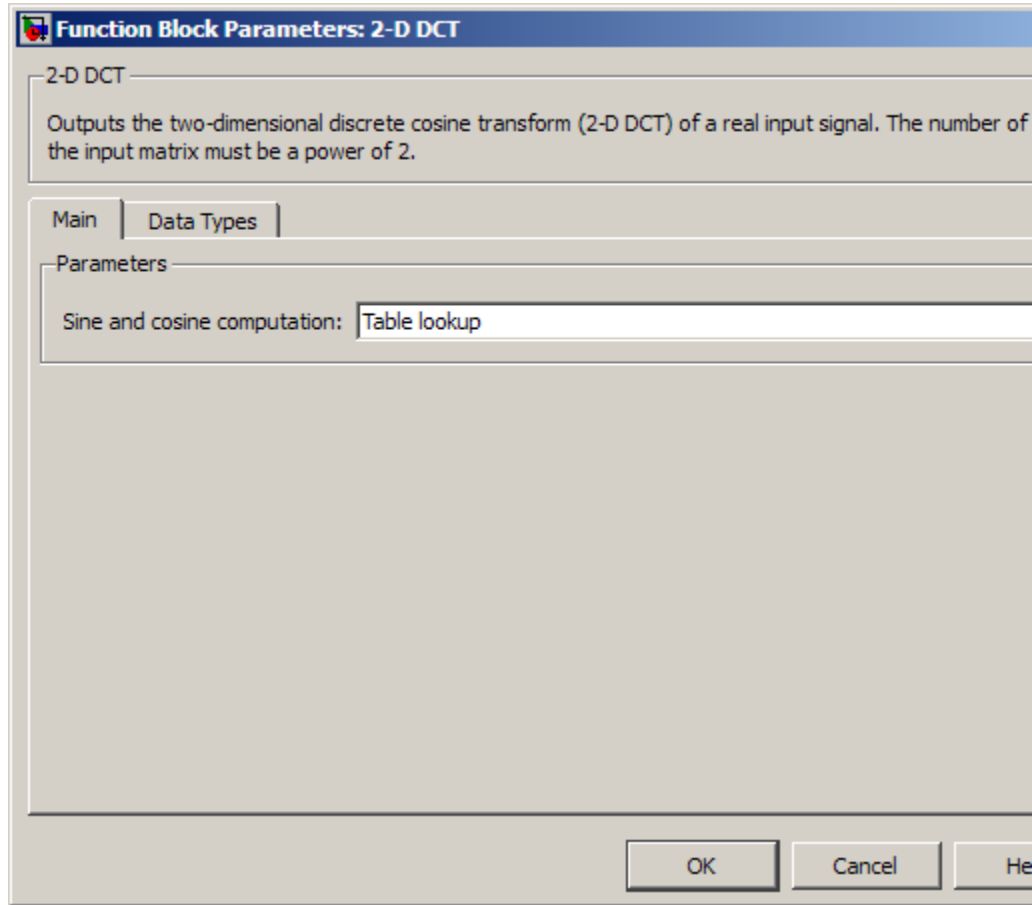
Twiddle Multiplication Data Types



The output of the multiplier is in the product output data type when at least one of the inputs to the multiplier is real. When both inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, refer to “Multiplication Data Types” in the Signal Processing Blockset documentation. You can set the sine table, product output, accumulator, and output data types in the block mask as discussed in the next section.

Dialog Box

The **Main** pane of the 2-D DCT dialog box appears as shown in the following figure.



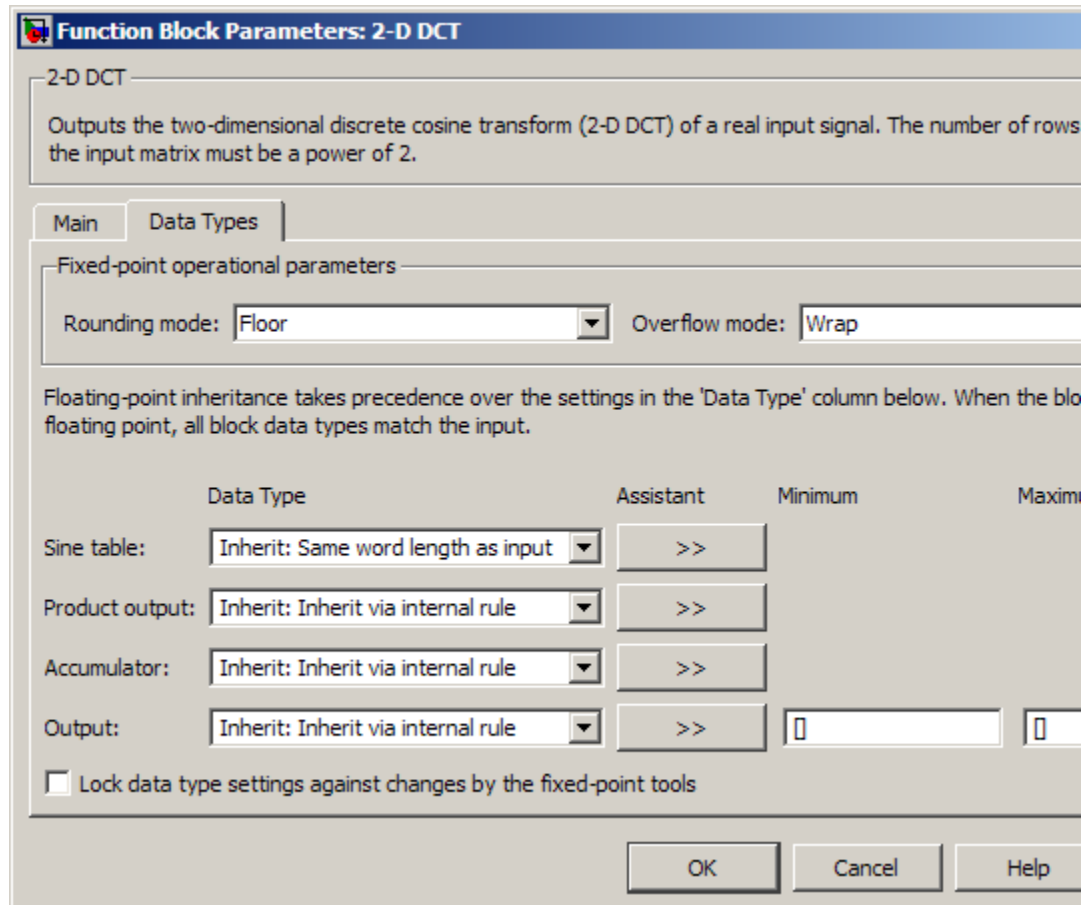
Sine and cosine computation

Specify how the block computes the sine and cosine terms in the DCT algorithm. If you select **Trigonometric fcn**, the block computes the sine and cosine values during the simulation. If you select **Table lookup**, the block computes and stores the

2-D DCT

trigonometric values before the simulation starts. In this case, the block requires extra memory.

The **Data Types** pane of the 2-D DCT dialog box appears as shown in the following figure.



Rounding mode

Select the rounding mode for fixed-point operations. The sine table values do not obey this parameter; they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations. The sine table values do not obey this parameter; instead, they are always saturated.

Sine table data type

Choose how you specify the word length of the values of the sine table. The fraction length of the sine table values always equals the word length minus one. You can set this parameter to:


- A rule that inherits a data type, for example, `Inherit: Same word length as input`
- An expression that evaluates to a valid data type, for example, `fixdt(1,16)`

The sine table values do not obey the **Rounding mode** and **Overflow mode** parameters; instead, they are always saturated and rounded to Nearest.

Product output data type

Specify the product output data type. See “Fixed-Point Data Types” on page 2-36 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`
- An expression that evaluates to a valid data type, for example, `fixdt(1,16,0)`


Click the **Show data type assistant** button  to display the **Data Type Assistant**, which helps you set the **Product output data type** parameter.

See “Using the Data Type Assistant” in *Simulink User’s Guide* for more information.

Accumulator data type

Specify the accumulator data type. See “Fixed-Point Data Types” on page 2-36 for illustrations depicting the use of the accumulator data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`
- An expression that evaluates to a valid data type, for example, `fixdt(1,16,0)`

Click the **Show data type assistant** button  to display the **Data Type Assistant**, which helps you set the **Accumulator data type** parameter.

See “Using the Data Type Assistant” in *Simulink User’s Guide* for more information.

Output data type

Specify the output data type. See “Fixed-Point Data Types” on page 2-36 for illustrations depicting the use of the output data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`.


When you select `Inherit: Inherit via internal rule`, the block calculates the output word length and fraction length automatically. The internal rule first calculates an ideal output word length and fraction length using the following equations:

$$WL_{ideal\ output} = WL_{input} + \text{floor}(\log_2(DCT\ length - 1)) + 1$$

$$FL_{ideal\ output} = FL_{input}$$

Using these ideal results, the internal rule then selects word lengths and fraction lengths that are appropriate for your hardware. For more information, see “Inherit via Internal Rule”.

- An expression that evaluates to a valid data type, for example, `fixdt(1,16,0)`

Click the **Show data type assistant** button  to display the **Data Type Assistant**, which helps you set the **Output data type** parameter.

See “Specifying Block Output Data Types” in *Simulink User’s Guide* for more information.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool. For more information, see `fxpdtlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxpdtlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

References

- [1] Chen, W.H, C.H. Smith, and S.C. Fralick, “A fast computational algorithm for the discrete cosine transform,” *IEEE Trans. Commun.*, vol. COM-25, pp. 1004-1009. 1977.
- [2] Wang, Z. “Fast algorithms for the discrete W transform and for the discrete Fourier transform,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 803-816, Aug. 1984.

2-D DCT

See Also

2-D IDCT	Video and Image Processing Blockset software
2-D FFT	Video and Image Processing Blockset software
2-D IFFT	Video and Image Processing Blockset software

Purpose Compute 2-D FFT of input

Library Transforms
viptransforms

Description



The 2-D FFT block computes the fast Fourier transform (FFT) of a two-dimensional M -by- N input matrix in two steps. First it computes the one-dimensional FFT along one dimension (row or column). Then it computes the FFT of the output of the first step along the other dimension (column or row). The dimensions of the input matrix, M and N , must be powers of two. To work with other input sizes, use the Pad block to pad or truncate these dimensions to powers of two.

The output of the 2-D FFT block is equivalent to the MATLAB `fft2` function:

```
y = fft2(A) % Equivalent MATLAB code
```

Computing the FFT of each dimension of the input matrix is equivalent to calculating the two-dimensional discrete Fourier transform (DFT), which is defined by the following equation:

$$F(m,n) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j\frac{2\pi mx}{M}} e^{-j\frac{2\pi ny}{N}}$$

where $0 \leq m \leq M-1$ and $0 \leq n \leq N-1$.

2-D FFT

Port	Description	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer	Yes
Output	2-D FFT of the input	Same as input port	Yes

If the input signal data type is floating point, the data type of the output signal uses the same floating-point data type. Otherwise, the output can be any fixed-point data type.

Optimizing the Table of Trigonometric Values

The block computes all the possible trigonometric values of the twiddle factor

$$e^{-j\frac{2\pi k}{K}}$$

where K is the greater value of either M or N and $k = 0, \dots, K - 1$. The block stores these values in a table and retrieves them during simulation. You can optimize the table of trigonometric values for memory or speed using the **Optimize table for** parameter. This parameter varies the number of table entries as summarized in the following table.

Optimize Table for Parameter Setting	Number of Table Entries for N-Point FFT
Speed	3N/4-floating point N - fixed point
Memory	N/4 -floating point Not supported for fixed point

Ordering Output Column Entries

Use the **Output in bit-reversed order** parameter to specify the ordering of the column elements of the output as either linear or bit-reversed. If you select the **Output in bit-reversed order** check box, the row and column elements are output in bit-reversed order. Thus, the m th row element appears at the k th position, where k is the bit reversed value of m . Also, the n th column element appears at the l th position, where l is the bit reversed value of n . If you clear the **Output in bit-reversed order** check box, the channel elements are output in linear order.

Note The 2-D FFT block calculates its output in bit-reversed order. Linearly ordering the 2-D FFT block output requires an extra bit-reversal operation. Thus, in many situations, you can increase the speed of the 2-D FFT block by selecting the **Output in bit-reversed order** check box.

For more information ordering of the output, see “Bit-Reversed Order” on page 2-50. The 2-D FFT block bit-reverses the order of both the columns and the rows.

Algorithms Used for FFT Computation

Which algorithms the block uses depends on whether the block input is floating-point or fixed-point, real or complex. The choice of algorithms is also affected by whether you want the output in linear or bit-reversed order. Based on these specifications, the block can use any of the following algorithms:

- Bit-reversal operation
- Double-signal algorithm
- Half-length algorithm
- Radix-2 decimation-in-time (DIT) algorithm
- Radix-2 decimation-in-frequency (DIF) algorithm

Floating-Point Signals

Complexity of Input	Output Ordering	Algorithms Used for FFT Computation
Complex	Linear	Bit-reversal operation and radix-2 DIT
Complex	Bit-reversed	Radix-2 DIF

Floating-Point Signals (Continued)

Complexity of Input	Output Ordering	Algorithms Used for FFT Computation
Real	Linear	Bit-reversal operation and radix-2 DIT in conjunction with the half-length and double-signal algorithms
Real	Bit-reversed	Radix-2 DIF in conjunction with the half-length and double-signal algorithms

Fixed-Point Signals

Complexity of Input	Output Ordering	Algorithms Used for FFT Computation
Real or complex	Linear	Bit-reversal operation and radix-2 DIT
Real or complex	Bit-reversed	Radix-2 DIF

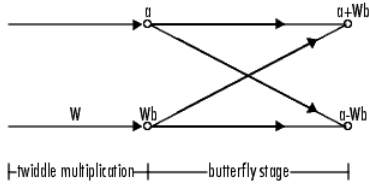
Fixed-Point Data Types

The following diagrams show the data types used in the 2-D FFT block for fixed-point signals. You can set the sine table, accumulator, product output, and output data types displayed in the diagrams in the 2-D FFT dialog box as discussed in “Dialog Box” on page 2-53.

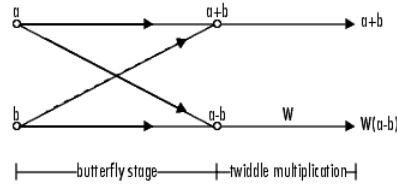
The block first casts inputs to the output data type and stores them in the output buffer. Each butterfly stage then processes signals in the accumulator data type, with the final output of the butterfly being cast back into the output data type. The block multiplies twiddle factor values before each butterfly stage in a decimation-in-time FFT and after each butterfly stage in a decimation-in-frequency FFT.

2-D FFT

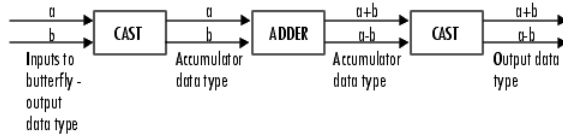
Decimation-in-time FFT



Decimation-in-frequency FFT



Butterfly stage data types



Twiddle multiplication data types



The output of the multiplier appears in the accumulator data type because both of the inputs to the multiplier are complex. For details on the complex multiplication performed, refer to “Multiplication Data Types” in the Signal Processing Blockset documentation.

Example

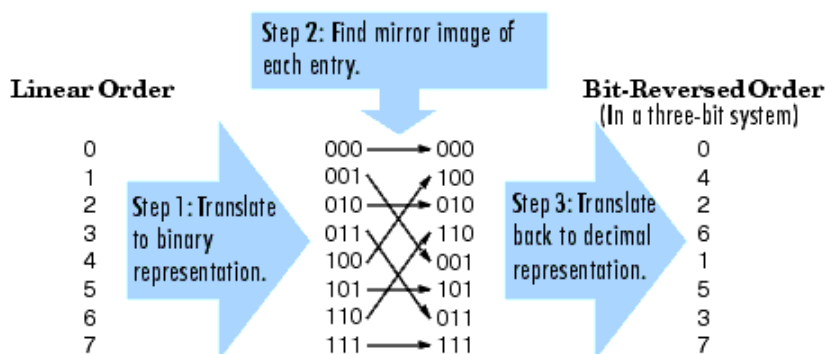
Bit-Reversed Order

Two numbers are bit-reversed values of each other when the binary representation of one is the mirror image of the binary representation of the other. For example, in a three-bit system, one and four are bit-reversed values of each other because the three-bit binary representation of one, 001, is the mirror image of the three-bit binary

representation of four, 100. The following diagram shows the row indices in linear order. To put them in bit-reversed order

- 1 Translate the indices into their binary representation with the minimum number of bits. In this example, the minimum number of bits is three because the binary representation of 7 is 111.
- 2 Find the mirror image of each binary entry, and write it beside the original binary representation.
- 3 Translate the indices back to their decimal representation.

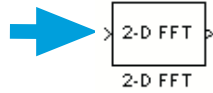
The row indices now appear in bit-reversed order.



If, on the 2-D FFT block parameters dialog box, you select the **Output in bit-reversed order** check box, the block bit-reverses the order of both the columns and the rows. The next diagram illustrates the linear and bit-reversed outputs of the 2-D FFT block. The output values are the same, but they appear in different order.

2-D FFT

Input to FFT block
(must be linear order)

$$\begin{bmatrix} 7 & 6 & 7 & 1 & 3 & 6 & 2 & 3 \\ 1 & 3 & 7 & 8 & 7 & 0 & 1 & 6 \\ 4 & 4 & 3 & 1 & 3 & 5 & 1 & 6 \\ 3 & 6 & 7 & 4 & 3 & 3 & 5 & 4 \\ 7 & 7 & 0 & 2 & 6 & 6 & 2 & 3 \\ 6 & 5 & 2 & 1 & 4 & 4 & 4 & 7 \\ 3 & 1 & 6 & 0 & 1 & 5 & 1 & 6 \\ 0 & 3 & 0 & 5 & 5 & 3 & 5 & 5 \end{bmatrix}$$


Output in linear order

Output in bit-reversed order

Output in linear order

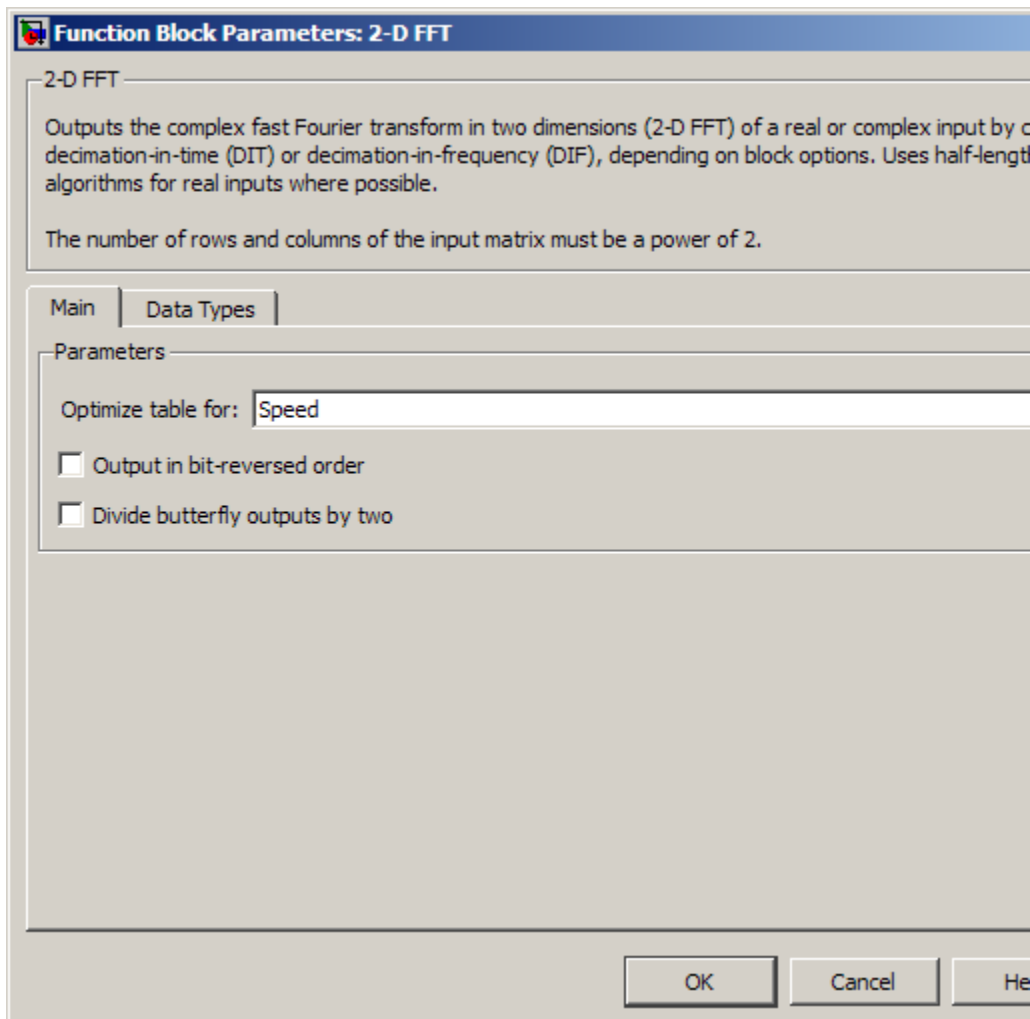
	0	1	2	3	4	5	6	7
Linearly ordered row and column indices	↓	↓	↓	↓	↓	↓	↓	↓
0 →	245	13.9 - 0.4i	10 - 5i	-15.9 + 21.6i	-13	-15.9 - 21.6i	10 + 5i	13.9
1 →	-4.3 - 10.3i	-27.6 - 6.6i	-5.6 + 13.1i	-3.4 + 8.7i	1.1 - i	-2.6i	-11.5 - 11i	6.2 + 13i
2 →	18 - 5i	-4.3 - 10.4i	19 - 24i	12.4 - 11.4i	6 - 3i	-5.7 + 16.4i	5 + 4i	5.5 + 1.4i
3 →	8.4 - 2.4i	-0.6 + 2.7i	-4.5 + 1.1i	17.6 + 9.4i	11 - 9i	-2.2 - 13i	-18.4 + 25.1i	34 + 0.5i
4 →	-9	16.3 + 5.9i	14 - 31i	17.7 + 23.9i	1	17.7 - 23.9i	14 + 31i	16.3 - 5.9i
5 →	8.4 + 2.4i	3.4 - 5.4i	-18.4 - 25.1i	-2.2 + 13.1i	11 + 9i	17.6 - 9.4i	-4.5 - 1.1i	-1 - 2.7i
6 →	18 + 5i	5.5 - 1.4i	5 - 4i	-5.7 - 16.4i	6 + 3i	12.5 + 11.3i	19 + 24i	-4.3 + 10.4i
7 →	-4.4 + 10.3i	6.2 - 13i	-11.5 + 11i	2.6i	1.1 + i	-3.4 - 8.7i	-5.6 - 13.1i	-27.6 + 6.6i

Output in bit-reversed order

	0	1	2	3	4	5	6	7
Bit-reversed row and column indices	↓	↓	↓	↓	↓	↓	↓	↓
0 →	245	-13	10 - 5i	10 + 5i	13.9 - 0.4i	-15.9 - 21.6i	-15.9 + 21.6i	13.9
1 →	-9	1	14 - 31i	14 + 31i	16.3 + 5.9i	17.7 - 23.9i	17.7 + 23.9i	16.3 - 5.9i
2 →	18 - 5i	6 - 3i	19 - 24i	5 + 4i	-4.3 - 10.4i	-5.7 + 16.4i	12.4 - 11.4i	5.5 + 1.4i
3 →	18 + 5i	6 + 3i	5 - 4i	19 + 24i	5.5 - 1.4i	12.5 + 11.3i	-5.7 - 16.4i	34 + 0.5i
4 →	-4.3 - 10.3i	1.1 - i	-5.6 + 13.1i	-11.5 - 11i	-27.6 - 6.6i	-2.6i	-3.4 + 8.7i	6.2 + 13i
5 →	8.4 + 2.4i	11 + 9i	-18.4 - 25.1i	-4.5 - 1.1i	3.4 - 5.4i	17.6 - 9.4i	-2.2 + 13i	-1 - 2.7i
6 →	8.4 - 2.4i	11 - 9i	-4.5 + 1.1i	-18.4 + 25.1i	-0.6 + 2.7i	-2.2 - 13i	17.6 + 9.4i	34 + 0.5i
7 →	-4.4 + 10.3i	1.1 + i	-11.5 + 11i	-5.6 - 13.1i	6.2 - 13i	-3.4 - 8.7i	2.6i	-27.6 + 6.6i

Dialog Box

The **Main** pane of the 2-D FFT dialog box appears as shown in the following figure.



Optimize table for

Optimize the table of twiddle factor values for Speed or Memory. This parameter must be set to Speed for fixed-point signals.

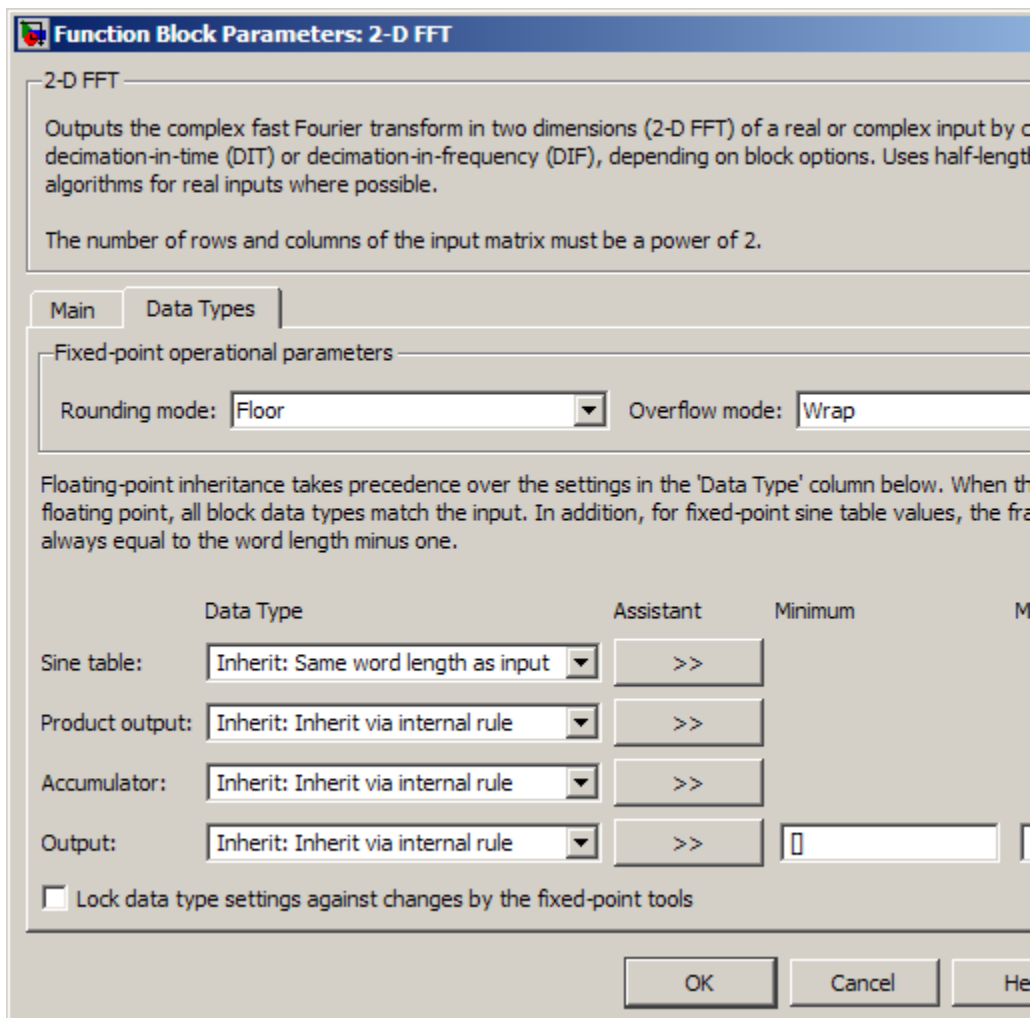
Output in bit-reversed order

Designate the order of the output channel elements relative to the ordering of the input elements. When selected, the output channel elements appear in bit-reversed order relative to the input ordering. Otherwise, the output column elements display in linear order relative to the input ordering. Linearly ordering the output requires extra data sorting manipulation. For more information, see “Bit-Reversed Order” on page 2-50.

Divide butterfly outputs by two

When you select this parameter, the output of each butterfly of the FFT is divided by two.

The **Data Types** pane of the 2-D FFT dialog box appears as shown in the following figure.



Rounding mode

Select the rounding mode for fixed-point operations. The sine table values do not obey this parameter; instead, they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations. The sine table values do not obey this parameter; instead, they are always saturated.

Sine table data type

Choose how you specify the word length of the values of the sine table. The fraction length of the sine table values always equals the word length minus one. You can set this parameter to:


- A rule that inherits a data type, for example, `Inherit: Same word length as input`
- An expression that evaluates to a valid data type, for example, `fixdt(1,16)`

The sine table values do not obey the **Rounding mode** and **Overflow mode** parameters; instead, they are always saturated and rounded to Nearest.

Product output data type

Specify the product output data type. See “Fixed-Point Data Types” on page 2-49 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`
- An expression that evaluates to a valid data type, for example, `fixdt(1,16,0)`


Click the **Show data type assistant** button  to display the **Data Type Assistant**, which helps you set the **Product output data type** parameter.

See “Using the Data Type Assistant” in *Simulink User’s Guide* for more information.

Accumulator data type

Specify the accumulator data type. See “Fixed-Point Data Types” on page 2-49 for illustrations depicting the use of the accumulator data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`
- An expression that evaluates to a valid data type, for example, `fixdt(1,16,0)`

Click the **Show data type assistant** button  to display the **Data Type Assistant**, which helps you set the **Accumulator data type** parameter.

See “Using the Data Type Assistant” in *Simulink User’s Guide* for more information.

Output data type

Specify the output data type. See “Fixed-Point Data Types” on page 2-49 for illustrations depicting the use of the output data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule.`

When you select `Inherit: Inherit via internal rule`, the block calculates the output word length and fraction length automatically. The internal rule first calculates an ideal output word length and fraction length using the following equations:


$$WL_{ideal\ output} = WL_{input} + \text{floor}(\log_2(\text{FFT length} - 1)) + 1$$

$$FL_{ideal\ output} = FL_{input}$$

Using these ideal results, the internal rule then selects word lengths and fraction lengths that are appropriate for your

hardware. For more information, see “Inherit via Internal Rule”.

- An expression that evaluates to a valid data type, for example, `fixdt(1,16,0)`

Click the **Show data type assistant** button  to display the **Data Type Assistant**, which helps you set the **Output data type** parameter.

See “Specifying Block Output Data Types” in *Simulink User’s Guide* for more information.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

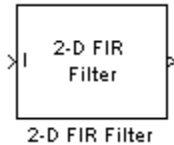
See Also

2-D DCT	Video and Image Processing Blockset software
2-D IDCT	Video and Image Processing Blockset software
2-D IFFT	Video and Image Processing Blockset software
FFT	Signal Processing Blockset software
IFFT	Signal Processing Blockset software
Pad	Signal Processing Blockset software
bitrevorder	Signal Processing Toolbox software
fft	MATLAB
ifft	MATLAB

Purpose Perform 2-D FIR filtering on input matrix

Library Filtering

Description The 2-D Finite Impulse Response (FIR) filter block filters the input matrix I using the coefficient matrix H or the coefficient vectors HH and HV.



Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	Yes
H	Matrix of filter coefficients	Same as I port.	Yes
HH	Vector of filter coefficients	Same as I port. The input to ports HH and HV must be the same data type.	Yes
HV	Vector of filter coefficients	Same as I port. The input to ports HH and HV must be the same data type.	Yes
PVal	Scalar value that represents the constant pad value	Input must have the same data type as the input to I port.	Yes
Output	Scalar, vector, or matrix of filtered values	Same as I port.	Yes

2-D FIR Filter

If the input has a floating-point data type, then the output uses the same data type. Otherwise, the output can be any fixed-point data type.

Select the **Separable filter coefficients** check box if your filter coefficients are separable. Using separable filter coefficients reduces the amount of calculations the block must perform to compute the output. For example, suppose your input image is M-by-N and your filter coefficient matrix is x-by-y. For a nonseparable filter with the **Output size** parameter set to `Same as input port I`, it would take

$$x \cdot y \cdot M \cdot N$$

multiply-accumulate (MAC) operations for the block to calculate the output. For a separable filter, it would only take

$$(x + y) \cdot M \cdot N$$

MAC operations. If you do not know whether or not your filter coefficients are separable, use the `isfilterseparable` function.

Here is an example of the function syntax, `[S, HCOL, HROW] = isfilterseparable(H)`. The `isfilterseparable` function takes the filter kernel, H, and returns S, HCOL and HROW. Here, S is a Boolean variable that is 1 if the filter is separable and 0 if it is not. HCOL is a vector of vertical filter coefficients, and HROW is a vector of horizontal filter coefficients.

Use the **Coefficient source** parameter to specify how to define your filter coefficients. If you select the **Separable filter coefficients** check box and then select a **Coefficient source** of `Specify via dialog`, the **Vertical coefficients (across height)** and **Horizontal coefficients (across width)** parameters appear in the dialog box. You can use these parameters to enter vectors of vertical and horizontal filter coefficients, respectively.

You can also use the variables HCOL and HROW, the output of the `isfilterseparable` function, for these parameters. If you select the **Separable filter coefficients** check box and then select a **Coefficient source** of `Input port`, ports HV and HH appear on the block. Use these ports to specify vectors of vertical and horizontal filter coefficients.

If you clear the **Separable filter coefficients** check box and select a **Coefficient source** of *Specify via dialog*, the **Coefficients** parameter appears in the dialog box. Use this parameter to enter your matrix of filter coefficients.

If you clear the **Separable filter coefficients** check box and select a **Coefficient source** of *Input port*, port **H** appears on the block. Use this port to specify your filter coefficient matrix.

The block outputs the result of the filtering operation at the **Output** port. The **Output size** parameter and the sizes of the inputs at ports **I** and **H** dictate the dimensions of the output. For example, assume that the input at port **I** has dimensions (M_i, N_i) and the input at port **H** has dimensions (M_h, N_h) . If you select an **Output size** of *Full*, the output has dimensions (M_i+M_h-1, N_i+N_h-1) . If you select an **Output size** of *Same as input port I*, the output has the same dimensions as the input at port **I**. If you select an **Output size** of *Valid*, the block filters the input image only where the coefficient matrix fits entirely within it, so no padding is required. The output has dimensions (M_i-M_h+1, N_i-N_h+1) . However, if $\text{all}(\text{size}(\mathbf{I}) < \text{size}(\mathbf{H}))$, the block errors out.

Use the **Padding options** parameter to specify how to pad the boundary of your input matrix. To pad your matrix with a constant value, select *Constant*. To pad your input matrix by repeating its border values, select *Replicate*. To pad your input matrix with its mirror image, select *Symmetric*. To pad your input matrix using a circular repetition of its elements, select *Circular*. For more information on padding, see the *Image Pad* block reference page.

If, for the **Padding options** parameter, you select *Constant*, the **Pad value source** parameter appears in the dialog box. If you select *Specify via dialog*, the **Pad value** parameter appears in the dialog box. Use this parameter to enter the constant value with which to pad your matrix. If you select **Pad value source** of *Input port*, the **PVal** port appears on the block. Use this port to specify the constant value with which to pad your matrix. The pad value must be real if the input image is real. You will get an error message if the pad value is complex when the input image is real.

2-D FIR Filter

Use the **Filtering based on** parameter to specify the algorithm by which the block filters the input matrix. If you select **Convolution** and set the **Output size** parameter to **Full**, the block filters your input using the following algorithm

$$C(i, j) = \sum_{m=0}^{(Ma-1)} \sum_{n=0}^{(Na-1)} A(m, n) * H(i - m, j - n)$$

where $0 \leq i < Ma + Mh - 1$ and $0 \leq j < Na + Nh - 1$. If you select **Correlation** and set the **Output size** parameter to **Full**, the block filters your input using the following algorithm

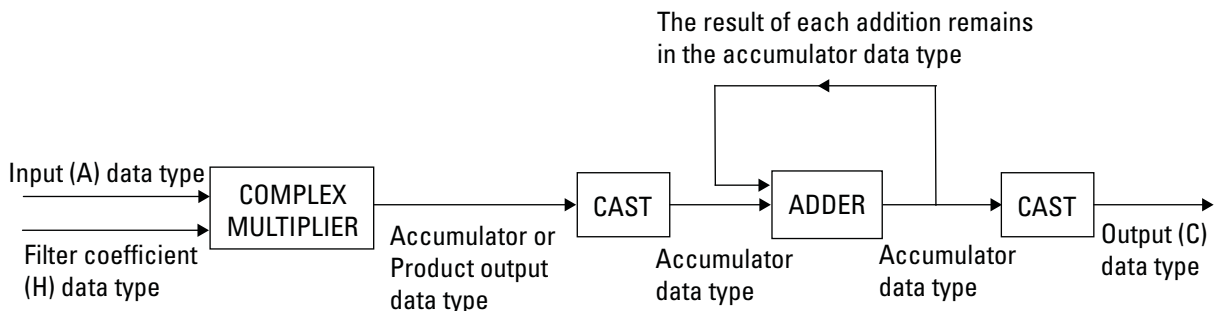
$$C(i, j) = \sum_{m=0}^{(Ma-1)} \sum_{n=0}^{(Na-1)} A(m, n) \cdot \text{conj}(H(m + i, n + j))$$

where $0 \leq i < Ma + Mh - 1$ and $0 \leq j < Na + Nh - 1$.

The `imfilter` function from the Image Processing Toolbox™ product similarly performs N-D filtering of multidimensional images.

Fixed-Point Data Types

The following diagram shows the data types used in the 2-D FIR Filter block for fixed-point signals.



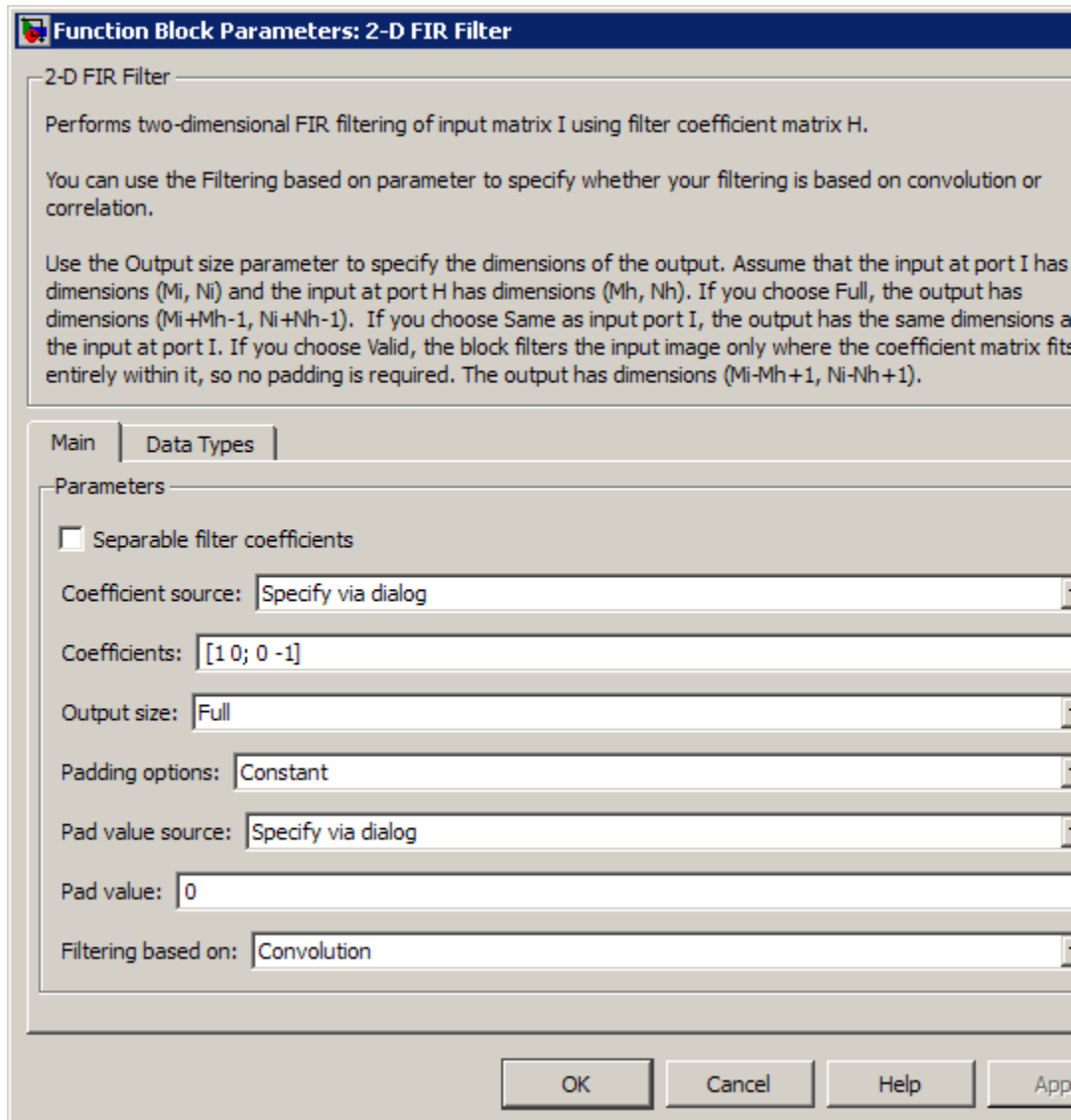
You can set the coefficient, product output, accumulator, and output data types in the block mask as discussed in “Dialog Box” on page 2-64.

The output of the multiplier is in the product output data type if at least one of the inputs to the multiplier is real. If both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, refer to “Multiplication Data Types” in the Signal Processing Blockset software documentation.

2-D FIR Filter

Dialog Box

The **Main** pane of the 2-D FIR Filter dialog box appears as shown in the following figure.



Separable filter coefficients

Select this check box if your filter coefficients are separable. Using separable filter coefficients reduces the amount of calculations the block must perform to compute the output.

Coefficient source

Specify how to define your filter coefficients. Select **Specify via dialog** to enter your coefficients in the block parameters dialog box. Select **Input port** to specify your filter coefficient matrix using port H or ports HH and HV.

Coefficients

Enter your real or complex-valued filter coefficient matrix. This parameter appears if you clear the **Separable filter coefficients** check box and then select a **Coefficient source** of **Specify via dialog**. Tunable.

Vertical coefficients (across height)

Enter the vector of vertical filter coefficients for your separable filter. This parameter appears if you select the **Separable filter coefficients** check box and then select a **Coefficient source** of **Specify via dialog**.

Horizontal coefficients (across width)

Enter the vector of horizontal filter coefficients for your separable filter. This parameter appears if you select the **Separable filter coefficients** check box and then select a **Coefficient source** of **Specify via dialog**.

Output size

This parameter controls the size of the filtered output. If you choose **Full**, the output has dimensions (M_a+M_h-1, N_a+N_h-1) . If you choose **Same as input port I**, the output has the same dimensions as the input at port I. If you choose **Valid**, output has dimensions (M_a-M_h+1, N_a-N_h+1) .

Padding options

Specify how to pad the boundary of your input matrix. Select **Constant** to pad your matrix with a constant value. Select **Replicate** to pad your input matrix by repeating its border

2-D FIR Filter

values. Select **Symmetric** to pad your input matrix with its mirror image. Select **Circular** to pad your input matrix using a circular repetition of its elements. This parameter appears if you select an **Output size** of **Full** or **Same** as input port I.

Pad value source

Use this parameter to specify how to define your constant boundary value. Select **Specify via dialog** to enter your value in the block parameters dialog box. Select **Input port** to specify your constant value using the PVal port. This parameter appears if you select a **Padding options** of **Constant**.

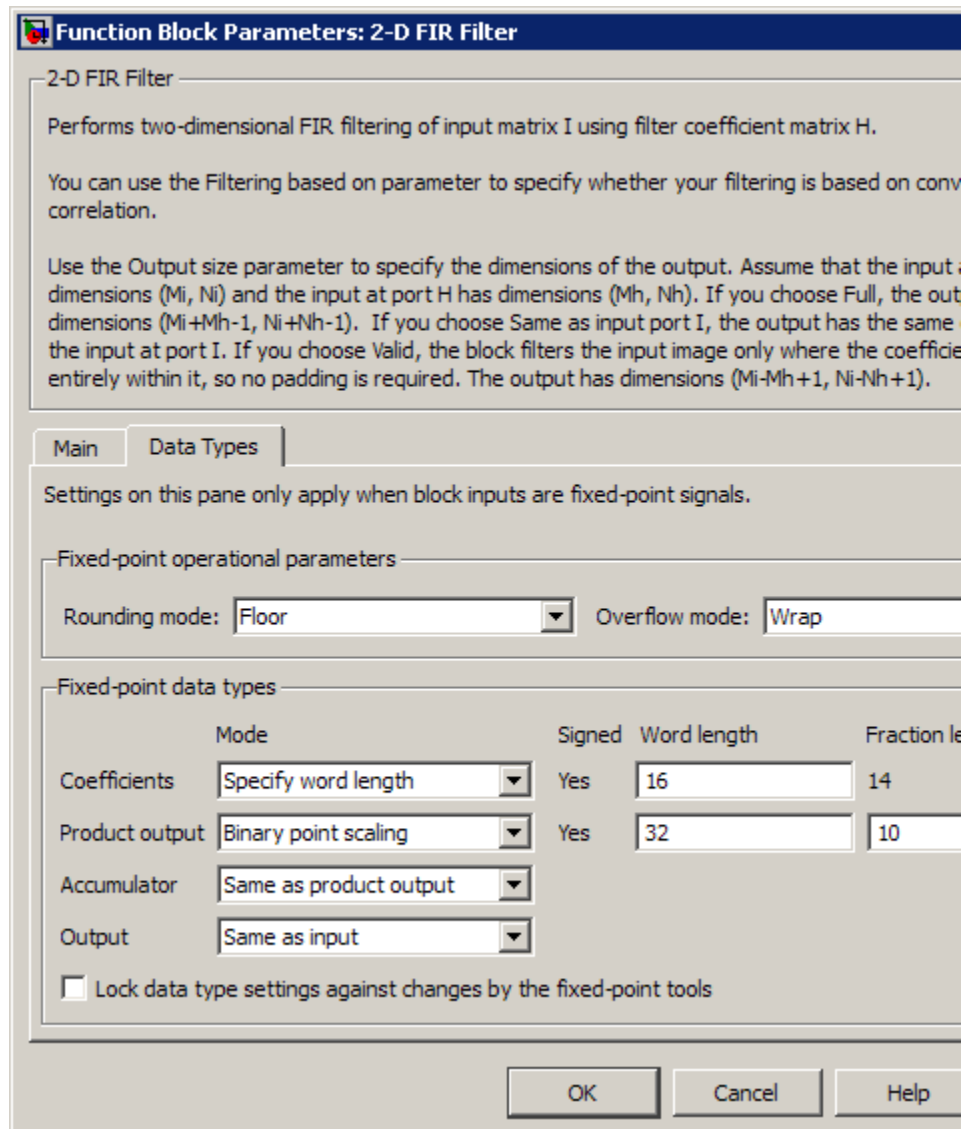
Pad value

Enter the constant value with which to pad your matrix. This parameter is visible if, for the **Pad value source** parameter, you select **Specify via dialog**. Tunable. The pad value must be real if the input image is real. You will get an error message if the pad value is complex when the input image is real.

Filtering based on

Specify the algorithm by which the block filters the input matrix. You can select **Convolution** or **Correlation**.

The **Data Types** pane of the 2-D FIR Filter dialog box appears as shown in the following figure.



2-D FIR Filter

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Coefficients

Choose how to specify the word length and the fraction length of the filter coefficients.

- When you select **Same word length as input**, the word length of the filter coefficients match that of the input to the block. In this mode, the block automatically sets the fraction length of the coefficients to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select **Specify word length**, you can enter the word length of the coefficients, in bits. In this mode, the block automatically sets the fraction length of the coefficients to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the coefficients, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the coefficients. All signals in the Video and Image Processing Blockset software have a bias of 0.

The filter coefficients do not obey the **Rounding mode** and the **Overflow mode** parameters; instead, they always saturated and rounded to Nearest.

Product output

Use this parameter to specify how to designate the product output word and fraction lengths. Refer to “Fixed-Point Data Types” on page 2-62 and “Multiplication Data Types” in the Signal

Processing Blockset documentation for illustrations depicting the use of the product output data type in this block:

- When you select **Same as input**, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. All signals in the Video and Image Processing Blockset software have a bias of 0.

If you set the **Coefficient source** (on the **Main** tab) to **Input port** the Product Output will inherit its sign according to the inputs. If either or both input **I1** and **I2** are signed, the Product Output will be signed. Otherwise, the Product Output is unsigned. The following table shows all cases.

Sign of Input I1	Sign of Input I2	Sign of Product Output
unsigned	unsigned	unsigned
unsigned	signed	signed
signed	unsigned	signed
signed	signed	signed

Accumulator

Use this parameter to specify how to designate the accumulator word and fraction lengths. Refer to “Fixed-Point Data Types” on page 2-62 and “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block. The accumulator data type is only used when both inputs to the multiplier are complex:

2-D FIR Filter

- When you select **Same as product output**, these characteristics match those of the product output.
- When you select **Same as input**, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. All signals in the Video and Image Processing Blockset software have a bias of 0.

Output

Choose how to specify the word length and fraction length of the output of the block:

- When you select **Same as input**, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the output. All signals in the Video and Image Processing Blockset software have a bias of 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

See Also

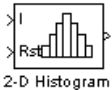
`imfilter`

Image Processing Toolbox

Purpose Generate histogram of each input matrix

Library Statistics

Description



Note The 2-D Histogram block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset software. Use the replacement block Histogram.

The 2-D Histogram block computes the frequency distribution of the elements in each input matrix or in a sequence of inputs over a period of time. Use the **Running histogram** check box to select between the block's basic and running operation.

The output of the 2-D Histogram block is different than the output of the `imhist` function in the Image Processing Toolbox. For intensity images, the `imhist` function defines the p th bin boundaries as

$$\frac{A(p-1.5)}{(N-1)} \leq x < \frac{A(p-0.5)}{(N-1)}$$

where A is maximum value of the data type, N is the number of bins in the histogram, and p starts from 1. The 2-D Histogram block defines bin boundaries as

$$\frac{A(p-1)}{N} < x \leq \frac{Ap}{N}$$

where A corresponds to the **Maximum value of input** parameter and the **Minimum value of input** parameter is assumed to be 0.

2-D Histogram (Obsolete)

Port	Input/Output	Supported Data Types	Complex Values Supported
Input / I	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	Yes
Rst	Signal that triggers a reset event	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Boolean• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No
Output	Sample-based 1-by-N vector that represents the frequency distribution of each M-by-N input matrix or the frequency distributions in a series of M-by-N inputs	Same as Input port	No

Length-M 1-D vector inputs are treated as M-by-1 column vectors.

The block sorts the elements of each input matrix into the number of discrete bins, n , specified by the **Number of bins** parameter. Complex inputs are sorted by their magnitude squared values.

The histogram value for a given bin represents the frequency of occurrence of the input values bracketed by that bin. You specify the upper boundary of the highest-valued bin in the **Maximum value of input** parameter, B_M , and the lower boundary of the lowest-valued

bin in the **Minimum value of input** parameter, B_m . The bins have equal width of

$$\Delta = \frac{B_M - B_m}{n}$$

where n is the number of bins. The centers are located at

$$B_m + \left(k + \frac{1}{2}\right)\Delta \quad k = 0, 1, 2, \dots, n - 1$$

Input values that fall on the border between two bins are sorted into the lower-valued bin; that is, each bin includes its upper boundary. For example, a bin of width 4 centered on the value 5 contains the input value 7, but not the input value 3. Input values greater than the **Maximum value of input** parameter or less than **Minimum value of input** parameter are sorted into the highest-valued or lowest-valued bin, respectively. The values you enter for the **Maximum value of input** and **Minimum value of input** parameters must be real-valued scalar values.

Basic Operation

If you clear the **Running histogram** check box, the block computes the frequency distribution of each M-by-N input matrix and outputs a sample-based 1-by-N vector.

For example, if your input is $\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$ and you set the block parameters as follows:

- **Minimum value of input** = 0
- **Maximum value of input** = 4
- **Number of bins** = 4

2-D Histogram (Obsolete)

The block outputs [3 3 3 0].

If you select the **Normalized** check box, the block scales each element of the output so that $\text{sum}(v)$ is 1, where v is the output vector.

Running Operation

If you select the **Running histogram** check box, the block computes the frequency distributions in a series of M-by-N inputs.

For example, if your first input is $\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$, your second and current input is $\begin{bmatrix} 2 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \end{bmatrix}$, and you set the block parameters as follows:

- **Minimum value of input** = 0
- **Maximum value of input** = 4
- **Number of bins** = 4

The block outputs [3 6 6 3]. For the next input, the block computes the frequency distribution for the first three inputs, and so on.

Resetting the Running Histogram

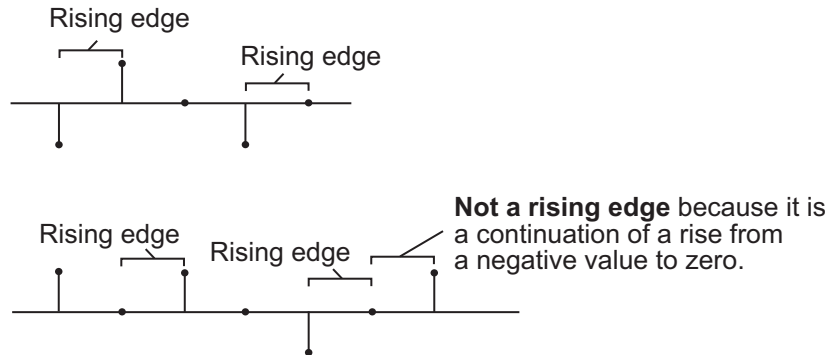
The block resets the running histogram whenever a reset event is detected at the optional Rst port. The reset signal and the input data signal must be the same rate.

To enable the Rst port, select the **Reset port** parameter. You specify the reset event in the **Trigger type** parameter, and can be one of the following:

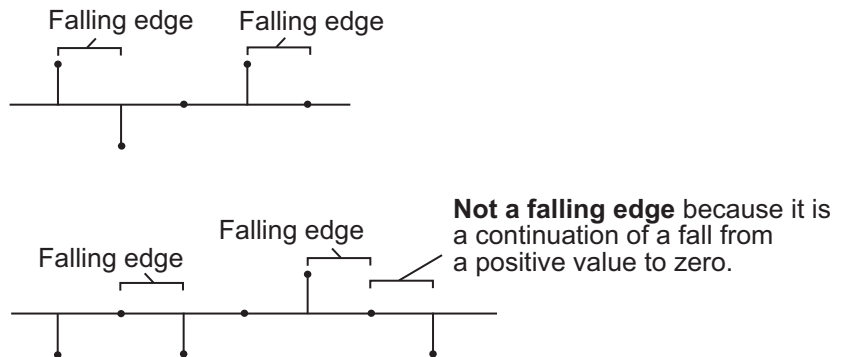
- **Rising edge** — Triggers a reset operation when the Rst input does one of the following:
 - Rises from a negative value to a positive value or 0

2-D Histogram (Obsolete)

- Rises from 0 to a positive value, where the rise is not a continuation of a rise from a negative value to 0 (see the following figure)



- Falling edge — Triggers a reset operation when the Rst input does one of the following:
 - Falls from a positive value to a negative value or 0
 - Falls from zero to a negative value, where the fall is not a continuation of a fall from a positive value to 0 (see the following figure)



- Either edge -- Triggers a reset operation when the Rst input is a Rising edge or Falling edge (as described previously)

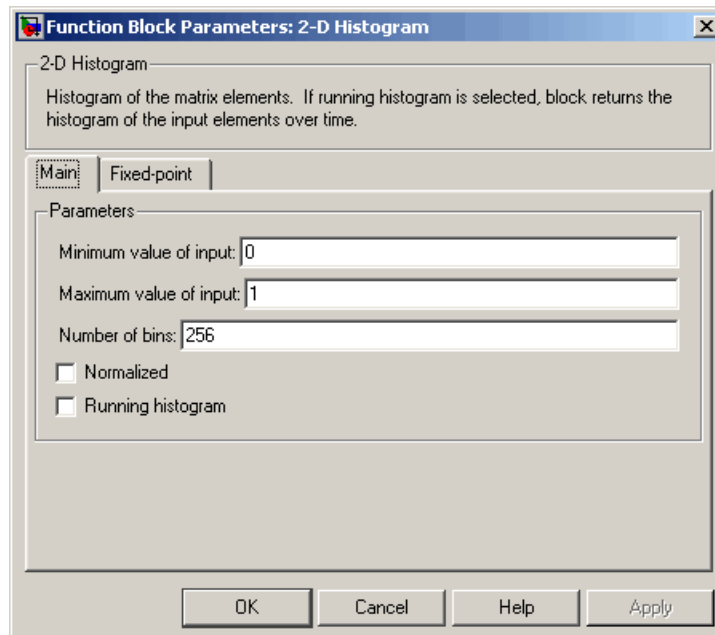
2-D Histogram (Obsolete)

- Non-zero sample -- Triggers a reset operation at each sample time that the Rst input is not 0

Note When running simulations in the Simulink MultiTasking mode, sample-based reset signals have a one-sample latency, and frame-based reset signals have one frame of latency. Thus, there is a one-sample or one-frame delay between the time the block detects a reset event, and when it applies the reset. For more information on latency and the Simulink tasking modes, see “Configuration Parameters Dialog Box” in the Simulink documentation.

Dialog Box

The **Main** pane of the 2-D Histogram dialog box:



Minimum value of input

Enter a real-valued scalar value for the lower boundary, B_m , of the lowest-valued bin. Tunable.

Maximum value of input

Enter a real-valued scalar value for the upper boundary, B_M , of the highest-valued bin. Tunable.

Number of bins

Enter the number of bins, n , in the histogram.

Normalized

If you select this check box, the block normalizes the output vector (1-norm). Tunable.

Use of this parameter is not supported for fixed-point signals.

Running histogram

Select this check box to enable the block's running histogram operation.

Reset port

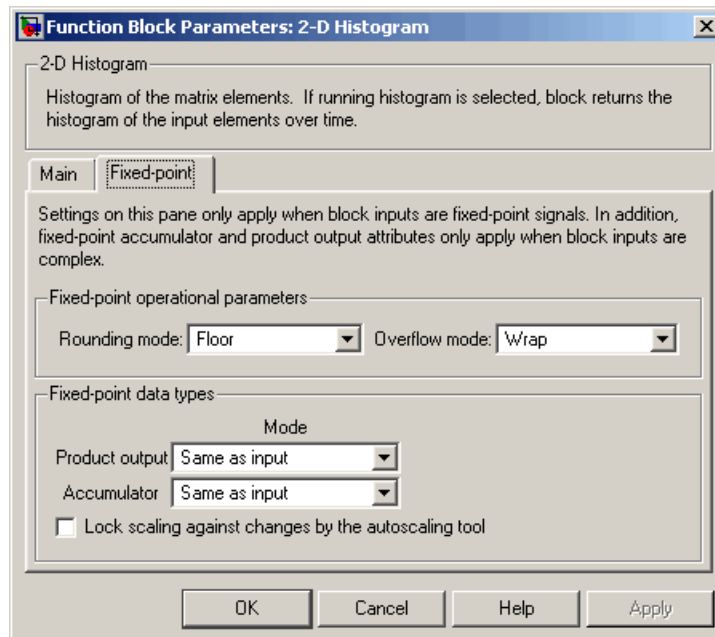
Enables the Rst input port when selected. The reset signal and the input data signal must be the same rate. This parameter is visible if you select the **Running histogram** check box.

Trigger type

The type of event that resets the running histogram. For more information, see "Resetting the Running Histogram" on page 2-74. This parameter is enabled only when you set the **Reset port** parameter.

The **Fixed-point** pane of the 2-D Histogram dialog box:

2-D Histogram (Obsolete)



Note The fixed-point parameters are only used for fixed-point complex inputs, which are sorted by squared magnitude.

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how to designate the product output word and fraction lengths:

- When you select **Same as input**, these characteristics match those of the input to the block.

- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of 0.

Accumulator

Use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block:

- When you select **Same as product output**, these characteristics match those of the product output.
- When you select **Same as input**, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of 0.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

See Also

2-D Autocorrelation	Video and Image Processing Blockset software
2-D Correlation	Video and Image Processing Blockset software

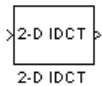
2-D Histogram (Obsolete)

Mean	Video and Image Processing Blockset software
Median	Video and Image Processing Blockset software
Standard Deviation	Video and Image Processing Blockset software
Variance	Video and Image Processing Blockset software
Histogram	Signal Processing Blockset software
Maximum	Signal Processing Blockset software
Minimum	Signal Processing Blockset software
hist	MATLAB application
imhist	Image Processing Toolbox software

Purpose Compute 2-D inverse discrete cosine transform (IDCT)

Library Transforms
viptransforms

Description



The 2-D IDCT block calculates the two-dimensional inverse discrete cosine transform of the input signal. The equation for the two-dimensional IDCT is

$$f(x,y) = \frac{2}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} C(m)C(n)F(m,n) \cos \frac{(2x+1)m\pi}{2M} \cos \frac{(2y+1)n\pi}{2N}$$

where $F(m,n)$ is the DCT of the signal $f(x,y)$ and $C(m), C(n) = \frac{1}{\sqrt{2}}$ for $m, n = 0$ and $C(m), C(n) = 1$ otherwise.

The number of rows and columns of the input signal must be powers of two. The output of this block has dimensions the same dimensions as the input.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	No

2-D IDCT

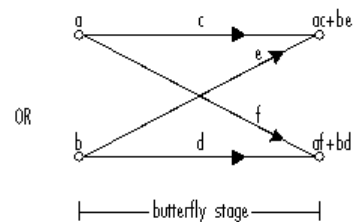
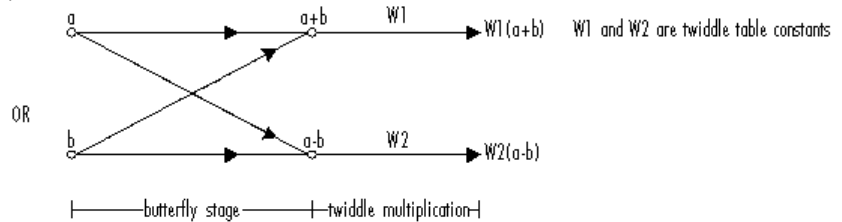
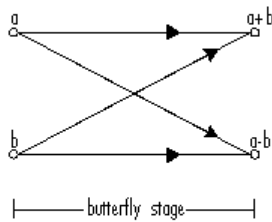
Port	Input/Output	Supported Data Types	Complex Values Supported
Output	2-D IDCT of the input	Same as Input port	No

If the data type of the input signal is floating point, the output of the block is the same data type.

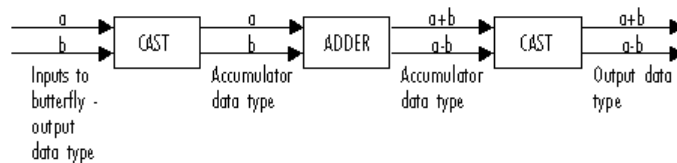
Use the **Sine and cosine computation** parameter to specify how the block computes the sine and cosine terms in the IDCT algorithm. If you select **Trigonometric fcn**, the block computes the sine and cosine values during the simulation. If you select **Table lookup**, the block computes and stores the trigonometric values before the simulation starts. In this case, the block requires extra memory.

Fixed-Point Data Types

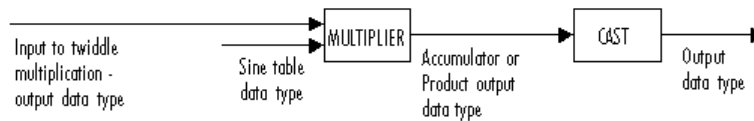
The following diagram shows the data types used in the 2-D IDCT block for fixed-point signals. Inputs are first cast to the output data type and stored in the output buffer. Each butterfly stage processes signals in the accumulator data type, with the final output of the butterfly being cast back into the output data type.



Butterfly Stage Data Types



Twiddle Multiplication Data Types

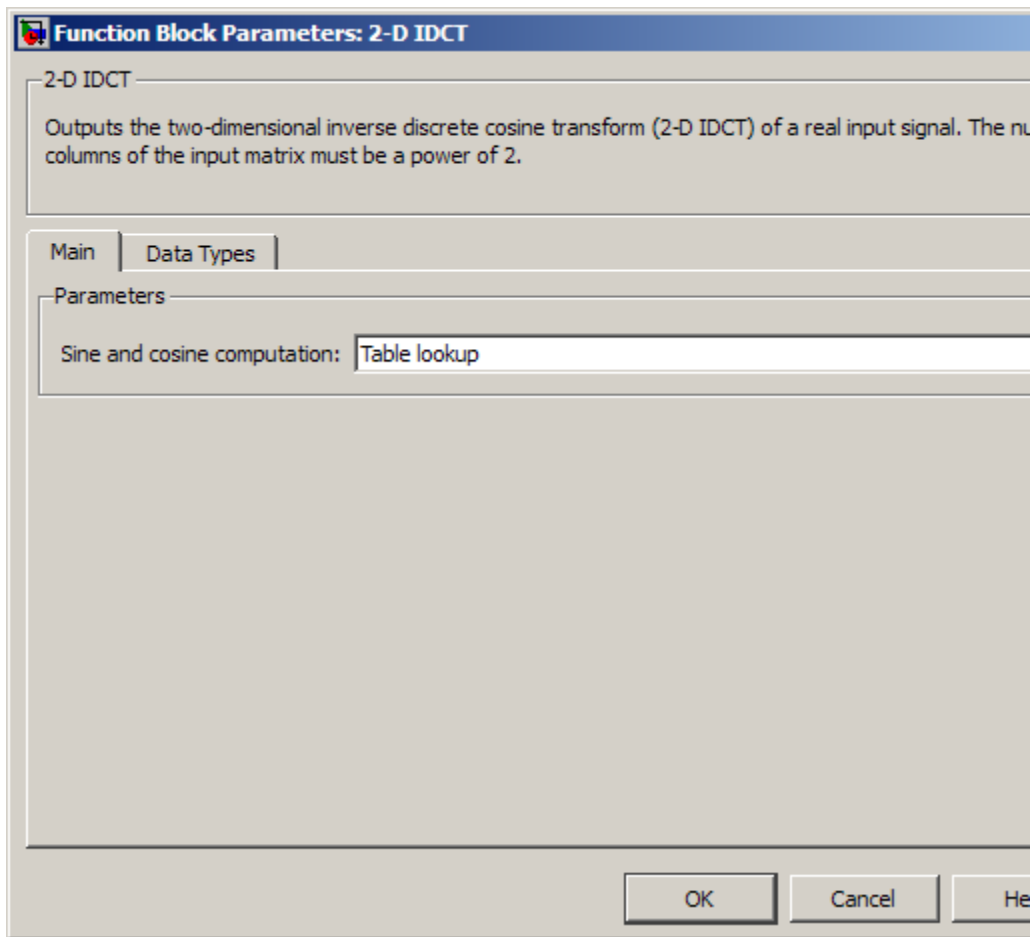


2-D IDCT

The output of the multiplier is in the product output data type when at least one of the inputs to the multiplier is real. When both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, refer to “Multiplication Data Types” in the Signal Processing Blockset documentation. You can set the sine table, product output, accumulator, and output data types in the block mask as discussed in the next section.

Dialog Box

The **Main** pane of the 2-D IDCT dialog box appears as shown in the following figure.



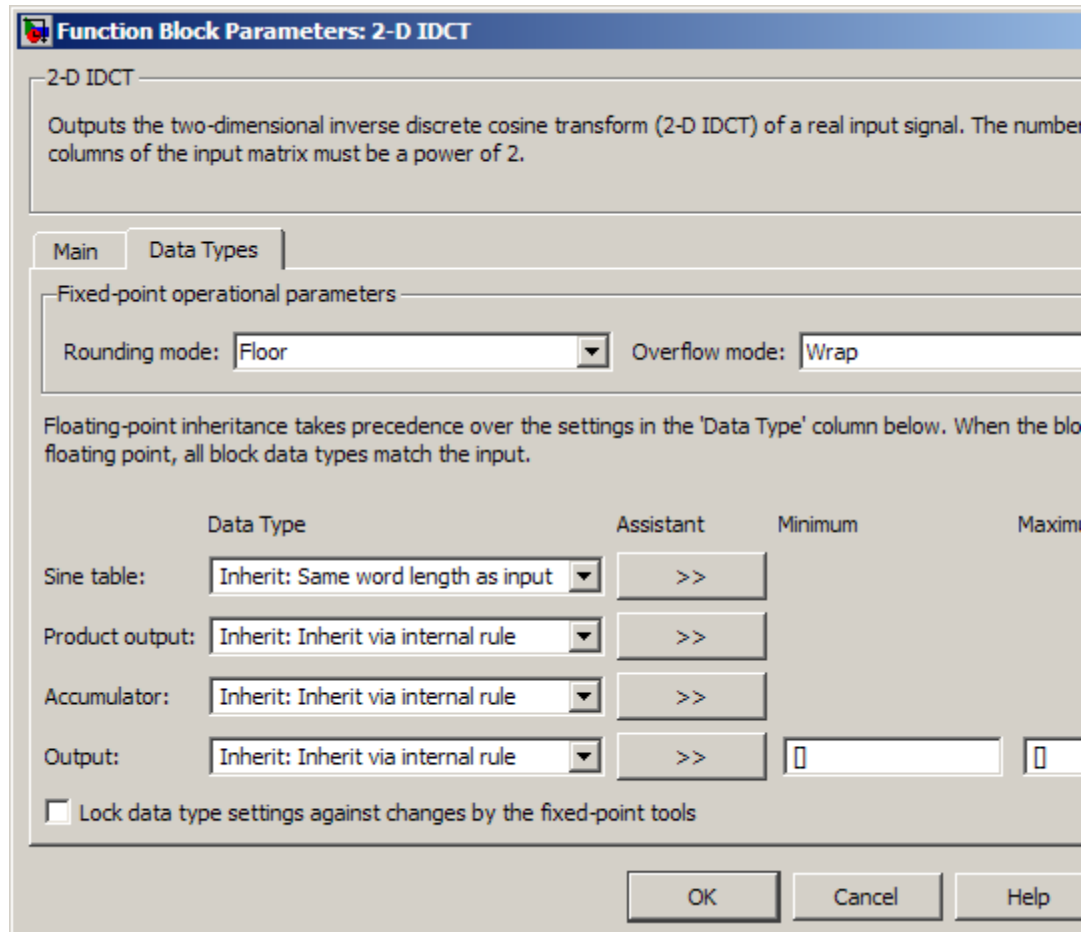
Sine and cosine computation

Specify how the block computes the sine and cosine terms in the IDCT algorithm. If you select `Trigonometric fcn`, the block computes the sine and cosine values during the simulation. If

2-D IDCT

you select **Table lookup**, the block computes and stores the trigonometric values before the simulation starts. In this case, the block requires extra memory.

The **Data Types** pane of the 2-D IDCT dialog box appears as shown in the following figure.



Rounding mode

Select the rounding mode for fixed-point operations. The sine table values do not obey this parameter; they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations. The sine table values do not obey this parameter; instead, they are always saturated.

Sine table data type

Choose how you specify the word length of the values of the sine table. The fraction length of the sine table values always equals the word length minus one. You can set this parameter to:


- A rule that inherits a data type, for example, `Inherit: Same word length as input`
- An expression that evaluates to a valid data type, for example, `fixdt(1,16)`

The sine table values do not obey the **Rounding mode** and **Overflow mode** parameters; instead, they are always saturated and rounded to Nearest.

Product output data type

Specify the product output data type. See “Fixed-Point Data Types” on page 2-82 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`
- An expression that evaluates to a valid data type, for example, `fixdt(1,16,0)`

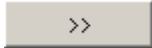
Click the **Show data type assistant** button  to display the **Data Type Assistant**, which helps you set the **Product output data type** parameter.

See “Using the Data Type Assistant” in *Simulink User’s Guide* for more information.

Accumulator data type

Specify the accumulator data type. See “Fixed-Point Data Types” on page 2-82 for illustrations depicting the use of the accumulator data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`
- An expression that evaluates to a valid data type, for example, `fixdt(1,16,0)`

Click the **Show data type assistant** button  to display the **Data Type Assistant**, which helps you set the **Accumulator data type** parameter.

See “Using the Data Type Assistant” in *Simulink User’s Guide* for more information.

Output data type

Specify the output data type. See “Fixed-Point Data Types” on page 2-82 for illustrations depicting the use of the output data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`.


When you select `Inherit: Inherit via internal rule`, the block calculates the output word length and fraction length automatically. The internal rule first calculates an ideal output word length and fraction length using the following equations:

$$WL_{ideal\ output} = WL_{input} + \text{floor}(\log_2(DCT\ length - 1)) + 1$$

$$FL_{ideal\ output} = FL_{input}$$

Using these ideal results, the internal rule then selects word lengths and fraction lengths that are appropriate for your hardware. For more information, see “Inherit via Internal Rule”.

- An expression that evaluates to a valid data type, for example, `fixdt(1,16,0)`

Click the **Show data type assistant** button  to display the **Data Type Assistant**, which helps you set the **Output data type** parameter.

See “Specifying Block Output Data Types” in *Simulink User’s Guide* for more information.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool. For more information, see `fxpdtlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxpdtlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

References

- [1] Chen, W.H, C.H. Smith, and S.C. Fralick, “A fast computational algorithm for the discrete cosine transform,” *IEEE Trans. Commun.*, vol. COM-25, pp. 1004-1009. 1977.
- [2] Wang, Z. “Fast algorithms for the discrete W transform and for the discrete Fourier transform,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 803-816, Aug. 1984.

2-D IDCT

See Also

2-D DCT	Video and Image Processing Blockset software
2-D FFT	Video and Image Processing Blockset software
2-D IFFT	Video and Image Processing Blockset software

Purpose Compute 2-D IFFT of input

Library Transforms
viptransforms

Description



The 2-D IFFT block computes the inverse fast Fourier transform (IFFT) of an M -by- N input matrix in two steps. First, it computes the one-dimensional IFFT along one dimension (row or column). Next, it computes the IFFT of the output of the first step along the other dimension (column or row). The dimensions of the input matrix, M and N , must be powers of two. To work with other input sizes, use the Pad block to pad or truncate these dimensions to powers of two.

The output of the IFFT block is equivalent to the MATLAB `ifft2` function:

```
y = ifft2(A)    % Equivalent MATLAB code
```

Computing the IFFT of each dimension of the input matrix is equivalent to calculating the two-dimensional inverse discrete Fourier transform (IDFT), which is defined by the following equation:

$$f(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F(m,n) e^{j\frac{2\pi mx}{M}} e^{j\frac{2\pi ny}{N}}$$

where $0 \leq x \leq M - 1$ and $0 \leq y \leq N - 1$.

The output of this block has the same dimensions as the input.

2-D IFFT

Port	Description	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer	Yes
Output	2-D IFFT of the input	Same as Input port	Yes

If the input signal has a floating-point data type, the data type of the output signal uses the same floating-point data type. Otherwise, the output can be any fixed-point data type.

Optimizing the Table of Trigonometric Values

The block computes all the possible trigonometric values of the twiddle factor

$$e^{j\frac{2\pi k}{K}}$$

where K is the greater value of either M or N and $k = 0, \dots, K - 1$. The block stores these values in a table and retrieves them during simulation. You can optimize the table of trigonometric values for memory consumption or speed using the **Optimize table for** parameter. This parameter varies the number of table entries as summarized in the following table.

Optimize Table for Parameter Setting	Number of Table Entries for N-Point IFFT
Speed	$3N/4$ —floating point N — fixed point
Memory	$N/4$ — floating point Not supported for fixed point

Input Order

You must select the **Input is in bit-reversed order** check box to designate whether the input column elements should appear in linear or bit-reversed order. If you select the **Input is in bit-reversed order** check box, the block assumes the input is in bit-reversed order. If you clear the **Input is in bit-reversed order** check box, block assumes the input is in linear order.

For more information ordering of the output, see “Bit-Reversed Order” on page 2-50. The 2-D FFT block bit-reverses the order of both the columns and the rows..

Conjugate Symmetric Input

The FFT block yields conjugate symmetric output when its input is real valued. Taking the IFFT of a conjugate symmetric input matrix produces real-valued output. Therefore, if the input to the block is both floating point and conjugate symmetric and you select the **Input is conjugate symmetric** check box, the block produces real-valued outputs. Selecting this check box optimizes the block’s computation method.

If the IFFT block input is conjugate symmetric and you do not select the **Input is conjugate symmetric** check box, the IFFT block outputs a complex-valued signal with small imaginary parts. The block output is invalid if you select this check box and the input is not conjugate symmetric.

Note The **Input is conjugate symmetric** parameter cannot be used for fixed-point signals.

Scaled Output

The **Divide output by product of FFT length in each input dimension** check box defaults to selected. The block computes scaled and unscaled versions of the IFFT. If you select this option, the block computes the scaled version of the IFFT.

The unscaled IFFT is defined by the following equation:

$$f(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F(m,n) e^{j\frac{2\pi mx}{M}} e^{j\frac{2\pi ny}{N}}$$

where $0 \leq x \leq M-1$ and $0 \leq y \leq N-1$.

The scaled version of the IFFT multiplies the above unscaled version

by $\frac{1}{MN}$.

Algorithms Used for IFFT Computation

Depending on whether the block input is floating point or fixed point, real or complex valued, and conjugate symmetric, the block uses one or more of the following algorithms as summarized in the following tables:

- Butterfly operation
- Double-signal algorithm
- Half-length algorithm
- Radix-2 decimation-in-time (DIT) algorithm
- Radix-2 decimation-in-frequency (DIF) algorithm

Algorithms for Floating-Point Signals

Input Complexity	Other Parameter Settings	Algorithms Used for IFFT Computation
Real or complex	<input type="checkbox"/> Input is in bit-reversed order <input type="checkbox"/> Input is conjugate symmetric	Butterfly operation and radix-2 DIT
Real or complex	<input checked="" type="checkbox"/> Input is in bit-reversed order <input type="checkbox"/> Input is conjugate symmetric	Radix-2 DIF
Real or complex	<input type="checkbox"/> Input is in bit-reversed order <input checked="" type="checkbox"/> Input is conjugate symmetric	Butterfly operation and radix-2 DIT in conjunction with the half-length and double-signal algorithms
Real or complex	<input checked="" type="checkbox"/> Input is in bit-reversed order <input checked="" type="checkbox"/> Input is conjugate symmetric	Radix-2 DIF in conjunction with the half-length and double-signal algorithms

Algorithms for Fixed-Point Signals

Input Complexity	Other Parameter Settings	Algorithms Used for IFFT Computation
Real or complex	<input type="checkbox"/> Input is in bit-reversed order <input type="checkbox"/> Input is conjugate symmetric	Butterfly operation and radix-2 DIT
Real or complex	<input checked="" type="checkbox"/> Input is in bit-reversed order <input type="checkbox"/> Input is conjugate symmetric	Radix-2 DIF

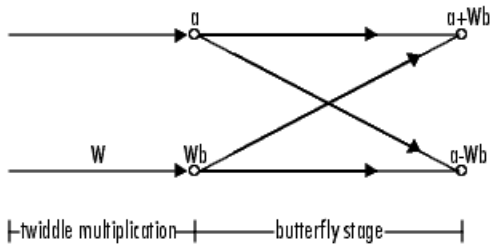
Note The **Input is conjugate symmetric** parameter cannot be used for fixed-point signals.

Fixed-Point Data Types

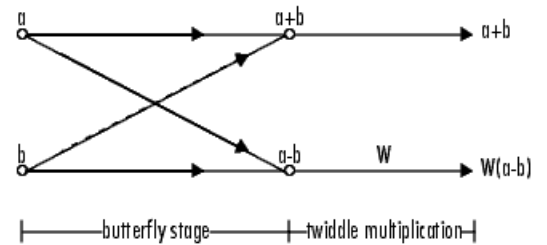
The following diagrams show the data types used in the IFFT block for fixed-point signals. You can set the sine table, accumulator, product output, and output data types displayed in the diagrams in the IFFT dialog box as discussed in “Dialog Box” on page 2-98.

Inputs to the IFFT block are first cast to the output data type and stored in the output buffer. Each butterfly stage then processes signals in the accumulator data type, with the final output of the butterfly being cast back into the output data type. The block multiplies in a twiddle factor before each butterfly stage in a decimation-in-time IFFT and after each butterfly stage in a decimation-in-frequency IFFT.

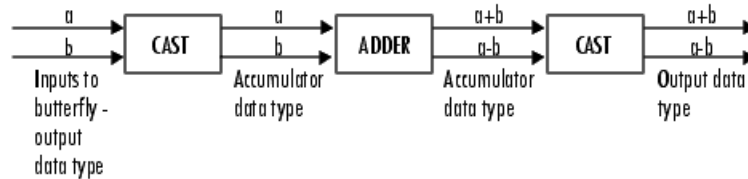
Decimation-in-time IFFT



Decimation-in-frequency IFFT



Butterfly stage data types



Twiddle multiplication data types

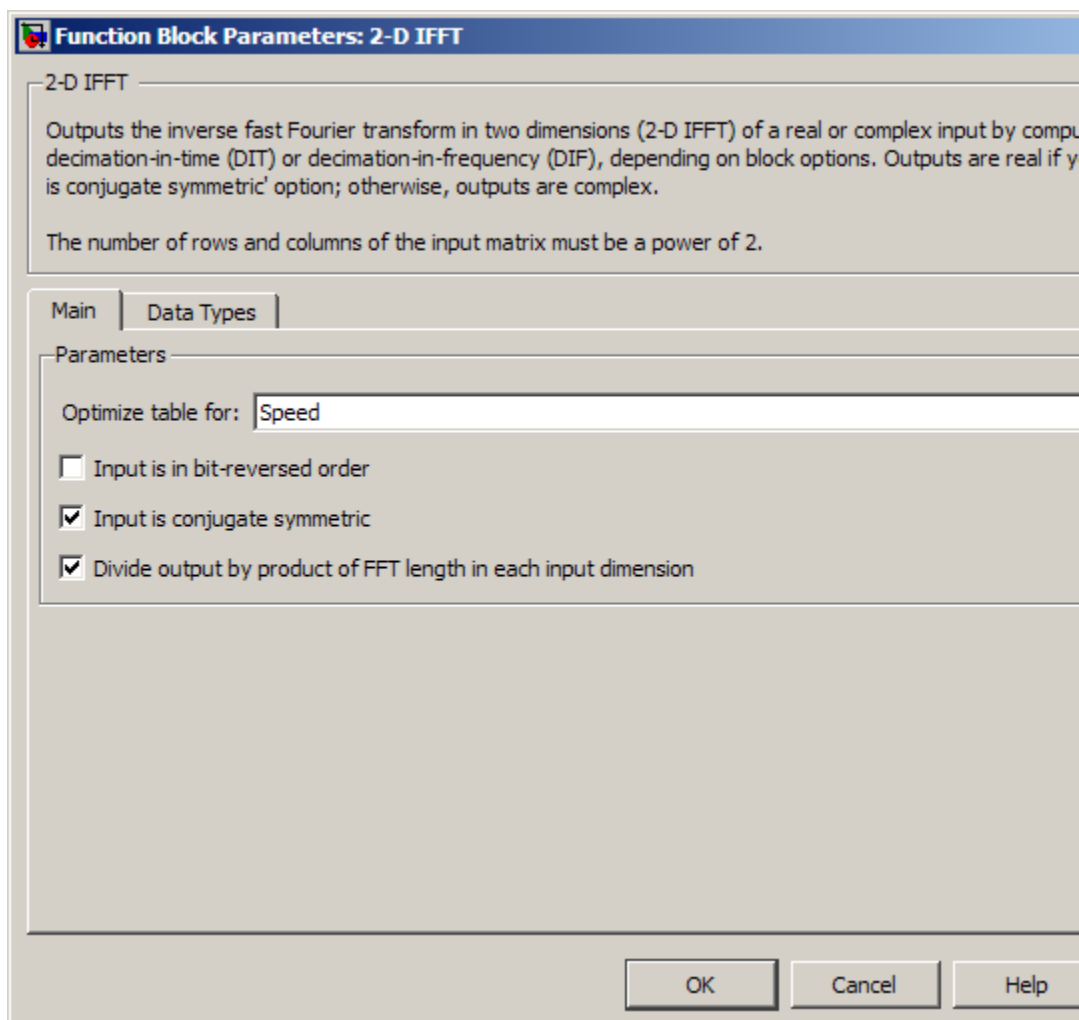


The multiplier output appears in the accumulator data type because both of the inputs to the multiplier are complex. For details on the complex multiplication performed, refer to “Multiplication Data Types” in the Signal Processing Blockset documentation.

2-D IFFT

Dialog Box

The **Main** pane of the 2-D IFFT dialog box appears as shown in the following figure.



Optimize table for

Optimize the table of trigonometric values for **Speed** or **Memory**. This parameter must be set to **Speed** for fixed-point signals.

Input is in bit-reversed order

Designate the order of the input channel elements. Select this check box when the input should appear in reversed order, and clear it when the input should appear in linear order. in linear order. The block yields invalid outputs when you do not set this parameter correctly. See “Input Order” on page 2-93.

Input is conjugate symmetric

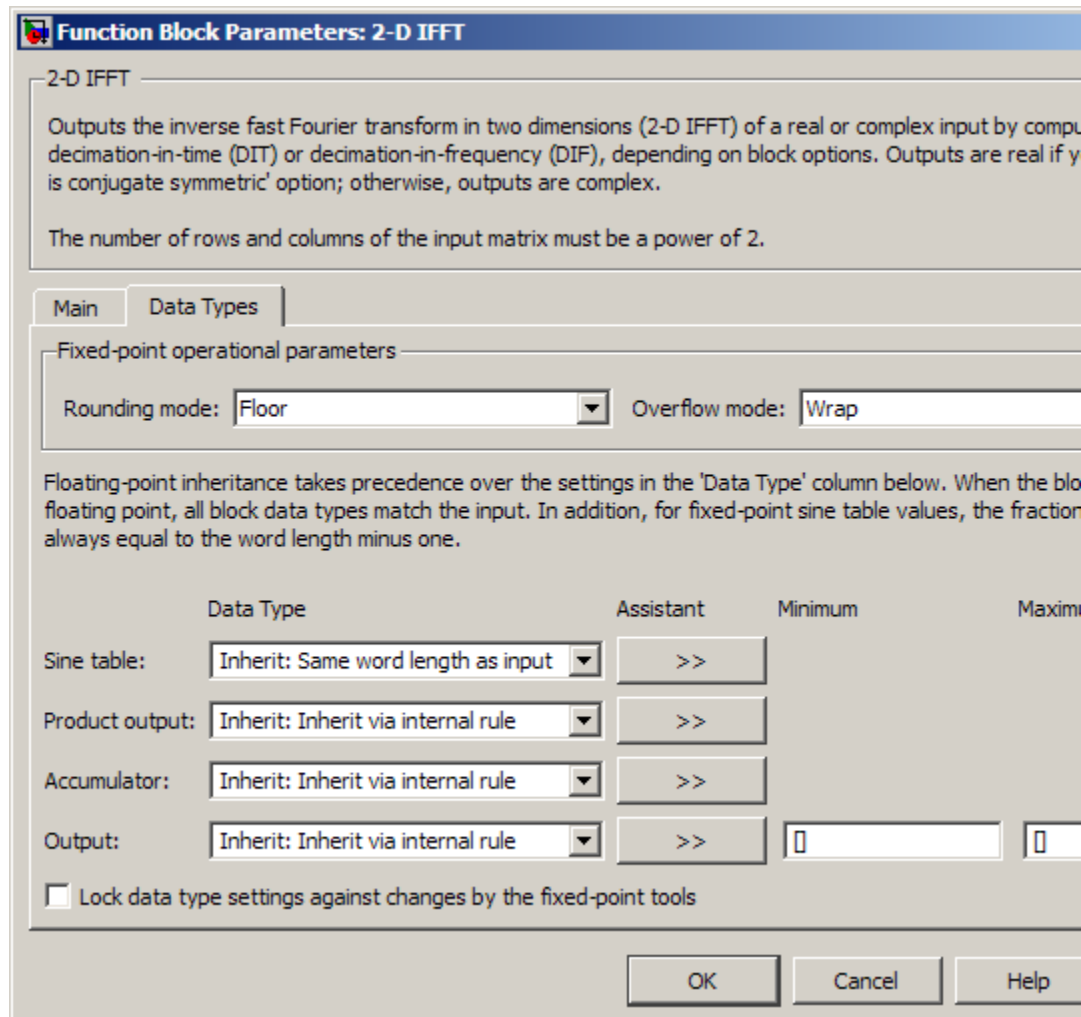
Select when the input to the block is both floating point and conjugate symmetric, and you want real-valued outputs. The block output is invalid when you set this parameter when the input is not conjugate symmetric. You cannot use this parameter for fixed-point signals.

Divide output by product of FFT length in each input dimension

Select this check box to compute the scaled IFFT.

The **Data Types** pane of the 2-D IFFT dialog box appears as shown in the following figure.

2-D IFFT



Rounding mode

Select the rounding mode for fixed-point operations. The sine table values do not obey this parameter; instead, they always round to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations. The sine table values do not obey this parameter; instead, they are always saturated.

Sine table data type

Choose how you specify the word length of the values of the sine table. The fraction length of the sine table values always equals the word length minus one. You can set this parameter to:


- A rule that inherits a data type, for example, `Inherit: Same word length as input`
- An expression that evaluates to a valid data type, for example, `fixdt(1,16)`

The sine table values do not obey the **Rounding mode** and **Overflow mode** parameters; instead, they are always saturated and rounded to Nearest.

Product output data type

Specify the product output data type. See “Fixed-Point Data Types” on page 2-96 and “Multiplication Data Types” for illustrations depicting the use of the product output data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`
- An expression that evaluates to a valid data type, for example, `fixdt(1,16,0)`


Click the **Show data type assistant** button  to display the **Data Type Assistant**, which helps you set the **Product output data type** parameter.

See “Using the Data Type Assistant” in *Simulink User’s Guide* for more information.

Accumulator data type

Specify the accumulator data type. See “Fixed-Point Data Types” on page 2-96 for illustrations depicting the use of the accumulator data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`
- An expression that evaluates to a valid data type, for example, `fixdt(1,16,0)`

Click the **Show data type assistant** button  to display the **Data Type Assistant**, which helps you set the **Accumulator data type** parameter.

See “Using the Data Type Assistant” in *Simulink User’s Guide* for more information.

Output data type

Specify the output data type. See “Fixed-Point Data Types” on page 2-96 for illustrations depicting the use of the output data type in this block. You can set this parameter to:

- A rule that inherits a data type, for example, `Inherit: Inherit via internal rule`.

When you select `Inherit: Inherit via internal rule`, the block calculates the output word length and fraction length automatically. The internal rule first calculates an ideal output word length and fraction length using the following equations:


$$WL_{ideal\ output} = WL_{input} + floor(\log_2(FFT\ length - 1)) + 1$$

$$FL_{ideal\ output} = FL_{input}$$

Using these ideal results, the internal rule then selects word lengths and fraction lengths that are appropriate for your

hardware. For more information, see “Inherit via Internal Rule”.

- An expression that evaluates to a valid data type, for example, `fixdt(1,16,0)`

Click the **Show data type assistant** button  to display the **Data Type Assistant**, which helps you set the **Output data type** parameter.

See “Specifying Block Output Data Types” in *Simulink User’s Guide* for more information.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

See Also

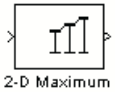
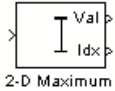
2-D DCT	Video and Image Processing Blockset software
2-D FFT	Video and Image Processing Blockset software
2-D IDCT	Video and Image Processing Blockset software
FFT	Signal Processing Blockset software
IFFT	Signal Processing Blockset software
Pad	Signal Processing Blockset software
bitrevorder	Signal Processing Toolbox software
fft	MATLAB
ifft	MATLAB

2-D Maximum (Obsolete)

Purpose Find maximum values in an input or sequence of inputs

Library vipobslib

Description



Note The 2-D Maximum block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset software. Use the replacement block Maximum.

The 2-D Maximum block identifies the value and/or position of the largest element in each column of the input, or tracks the maximum values in a sequence of inputs over a period of time. The **Mode** parameter specifies the block's mode of operation and can be set to Value, Index, Value and Index, or Running.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point -- Signed and unsigned real fixed point, and signed complex fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	Yes
Rst	Scalar value	Boolean	No

2-D Maximum (Obsolete)

Port	Input/Output	Supported Data Types	Complex Values Supported
Val	Maximum value in each M-by-N input matrix	Same as Input port	Yes
Idx	Two-element vector of the form [row index column index] that represents the zero-based location of the maximum value	Same as Input port	No

Length- M 1-D vector inputs are treated as M -by-1 column vectors.

Value Mode

When **Mode** is set to **Value**, the block computes the maximum value in each column of the M -by- N input matrix u independently at each sample time.

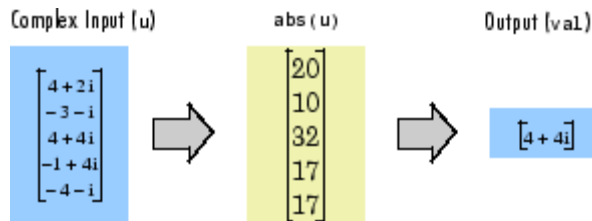
```
val = max(u)    % Equivalent MATLAB code
```

For convenience, length- M 1-D vector inputs and *sample-based* length- M row vector inputs are both treated as M -by-1 column vectors.

The output at each sample time, **val**, is a 1-by- N vector containing the maximum value of each column in u .

For complex inputs, the block selects the value in each column that has the maximum magnitude squared as shown in the following figure. For complex value $u = a + bi$, the magnitude squared is $a^2 + b^2$.

2-D Maximum (Obsolete)



Index Mode

When **Mode** is set to Index, the block computes the maximum value in each column of the M -by- N input matrix u ,

```
[val, idx] = max(u) % Equivalent MATLAB code
```

and outputs the sample-based 1-by- N index vector, idx . Each value in idx is an integer in the range $[1 M]$ indexing the maximum value in the corresponding column of u . When inputs to the block are double-precision values, the index values are double-precision values. Otherwise, the index values are 32-bit unsigned integer values.

As in Value mode, length- M 1-D vector inputs and *sample-based* length- M row vector inputs are both treated as M -by-1 column vectors.

When a maximum value occurs more than once in a particular column of u , the computed index corresponds to the first occurrence. For example, when the input is the column vector $[3 \ 2 \ 1 \ 2 \ 3]'$, the computed index of the maximum value is 1 rather than 5.

Value and Index Mode

When **Mode** is set to Value and Index, the block outputs both the vector of maxima, val , and the vector of indices, idx .

Running Mode

When **Mode** is set to Running, the block tracks the maximum value of each channel in a *time-sequence* of M -by- N inputs. For sample-based inputs, the output is a sample-based M -by- N matrix with each element y_{ij} containing the maximum value observed in element u_{ij} for all inputs since the last reset. For frame-based inputs, the output is a frame-based

M -by- N matrix with each element y_{ij} containing the maximum value observed in the j th column of all inputs since the last reset, up to and including element u_{ij} of the current input.

As in the other modes, length- M 1-D vector inputs and *sample-based* length- M row vector inputs are both treated as M -by-1 column vectors.

Resetting the Running Maximum

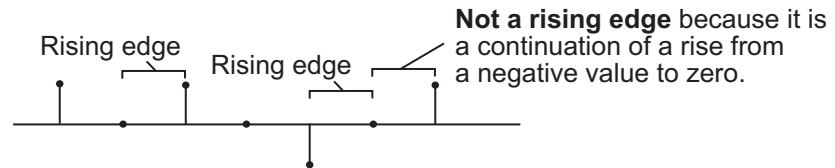
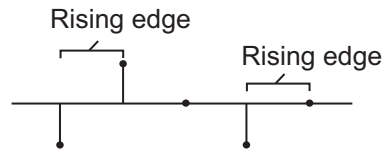
The block resets the running maximum whenever a reset event is detected at the optional Rst port. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input.

For sample-based inputs, a reset event causes the running maximum for each channel to be initialized to the value in the corresponding channel of the current input. For frame-based inputs, a reset event causes the running maximum for each channel to be initialized to the earliest value in each channel of the current input.

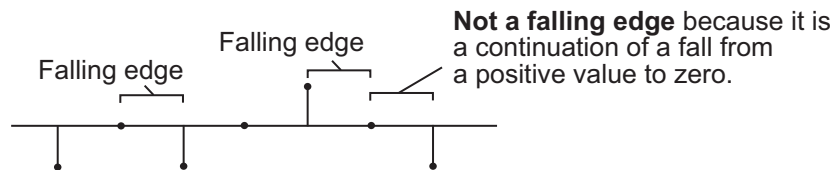
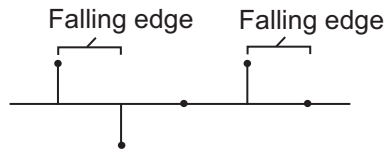
You specify the reset event in the **Reset port** menu:

- None — Disables the Rst port.
- Rising edge — Triggers a reset operation when the Rst input does one of the following:
 - Rises from a negative value to a positive value or 0
 - Rises from 0 to a positive value, where the rise is not a continuation of a rise from a negative value to 0 (see the following figure)

2-D Maximum (Obsolete)



- Falling edge — Triggers a reset operation when the Rst input does one of the following:
 - Falls from a positive value to a negative value or 0
 - Falls from 0 to a negative value, where the fall is not a continuation of a fall from a positive value to 0 (see the following figure)



- Either edge — Triggers a reset operation when the Rst input is a Rising edge or Falling edge (as described previously)
- Non-zero sample — Triggers a reset operation at each sample time that the Rst input is not 0

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see “Configuration Parameters Dialog Box” in the Simulink documentation.

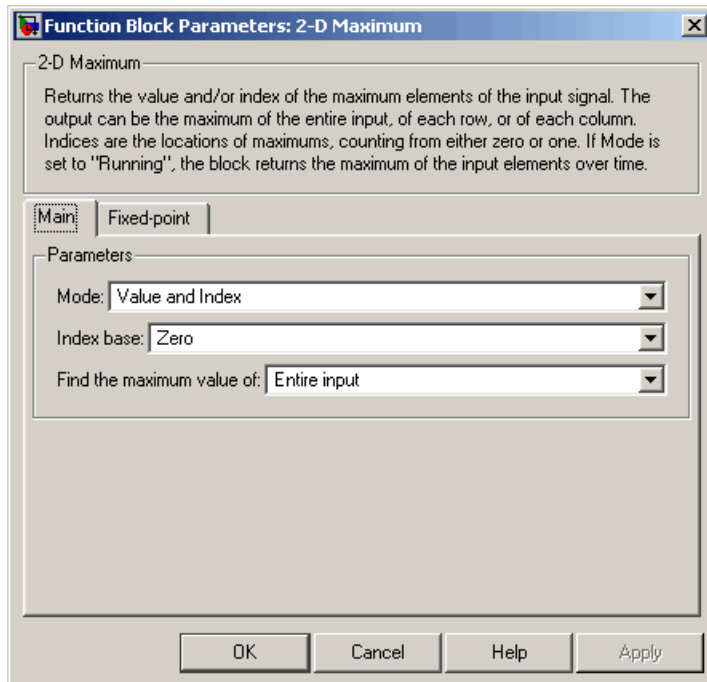
Fixed-Point Data Types

The parameters on the **Fixed-point** pane of the dialog box are only used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before a comparison is made, as described in “Value Mode” on page 2-105. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ignored for other types of inputs.

2-D Maximum (Obsolete)

Dialog Box

The **Main** pane of the 2-D Maximum dialog box appears as shown in the following figure.



Mode

Specify the block's mode of operation:

- **Value** — Output the maximum value of each input matrix
- **Index** — Output the zero-based index location of the maximum value
- **Value and Index** — Output both the value and the index location
- **Running** — Track the maximum value of the input sequence over time

Index base

Specify whether the index is zero based or one based.

Find the maximum value of

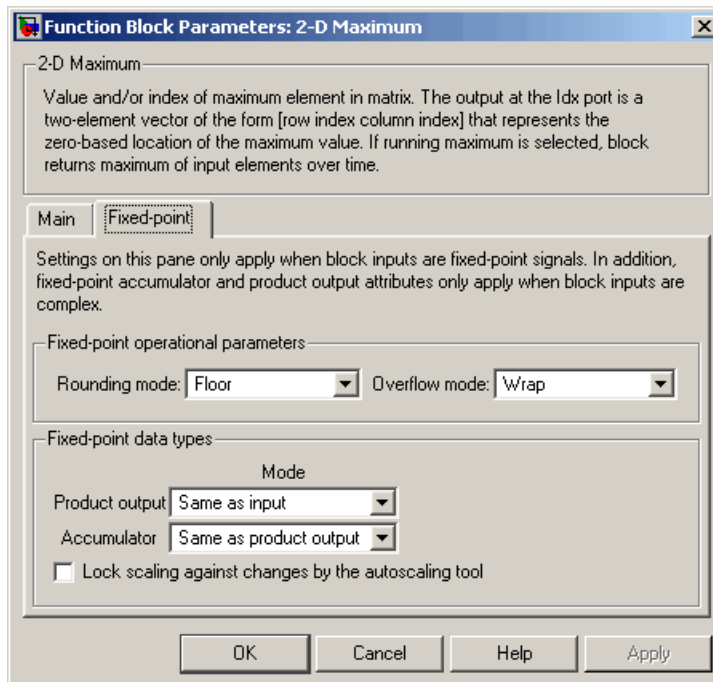
Specify whether the block should find the maximum of the entire input or of each row or column.

Reset port

Specify the reset event detected at the Rst input port when you select **Running** for the **Mode** parameter. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input. This parameter is only visible if, for the **Mode** parameter, you select **Running**.

The **Fixed-point** pane of the 2-D Maximum dialog box appears as shown in the following figure.

2-D Maximum (Obsolete)



Note The parameters on the **Fixed-point** pane are only used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before a comparison is made, as described in “Value Mode” on page 2-105. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ignored for other types of inputs.

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how to designate the product output word and fraction lengths resulting from a complex-complex multiplication in the block. Refer to “Multiplication Data Types” in the Signal Processing Blockset documentation for more information:

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of 0.

Accumulator

Use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block. Refer to “Multiplication Data Types” in the Signal Processing Blockset documentation for more information:

- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of 0.

2-D Maximum (Obsolete)

Lock scaling against changes by the autoscaling tool

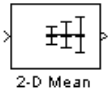
Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

2-D Mean (Obsolete)

Purpose Find mean value of each input matrix

Library Statistics

Description



Note The 2-D Mean block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset software. Use the replacement block Mean.

The 2-D Mean block computes the mean of each input matrix or the mean value in a sequence of inputs over time. It can also compute the mean over a particular region of interest (ROI). Use the **Running mean** check box to choose between the block's basic and running operation.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer	Yes
Rst	Scalar value	Boolean	No

2-D Mean (Obsolete)

Port	Input/Output	Supported Data Types	Complex Values Supported
ROI	<ul style="list-style-type: none"> • Rectangle — [r c height width] • Lines — [r1 c1 r2 c2], where r1 and c1 are the row and column coordinates of the beginning of the line and r2 and c2 are the row and column coordinates of the end of the line. • Binary mask — Binary image matrix that enables you to specify which pixels to highlight. 	Rectangles and lines — <ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer Binary mask — <ul style="list-style-type: none"> • Boolean 	No
Label	Matrix where pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on.	<ul style="list-style-type: none"> • 8-, 16-, and 32-bit unsigned integer 	No
Label Numbers	Vector containing the label numbers for the regions for which the block will compute the statistics.	<ul style="list-style-type: none"> • 8-, 16-, and 32-bit unsigned integer 	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Output/Out	<p>Without ROI processing — Mean of each M-by-N input matrix or the mean for each element of a series of M-by-N inputs.</p> <p>With ROI processing — Vector of separate statistical values for each ROI or a scalar value that represents the statistical value for all specified ROIs.</p>	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	Yes
Flag	Boolean value that indicates whether the ROI is within the image bounds or the label number is within the label matrix.	Boolean	No

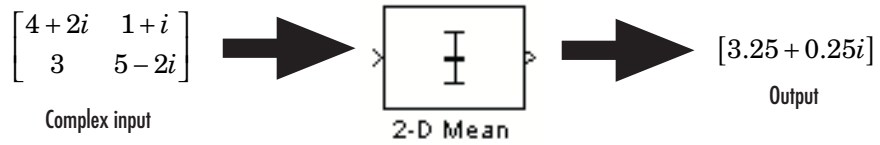
Length-M 1-D vector inputs are treated as M-by-1 column vectors.

Basic Operation

When you clear the **Running mean** check box, the block computes the mean of each M-by-N input matrix and outputs it from the block. The equivalent MATLAB code is `mean(u(:))`, where `u` is the input matrix.

The mean of a complex input is computed independently for the real and imaginary components, as shown in the following figure.

2-D Mean (Obsolete)



Running Operation

When you select the **Running mean** check box, the block computes the mean for each element of a series of M-by-N inputs.

For example, suppose A is the first input to the block and B is the second and current input to the block, where

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

and

$$B = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}$$

The block computes the mean corresponding to each element,

$$\begin{bmatrix} \text{mean}([1,5]) & \text{mean}([3,7]) \\ \text{mean}([2,6]) & \text{mean}([4,8]) \end{bmatrix}$$

and outputs

$$\begin{bmatrix} 3 & 5 \\ 4 & 6 \end{bmatrix}$$

For the next input, the block computes the mean for each element of the first three inputs, and so on.

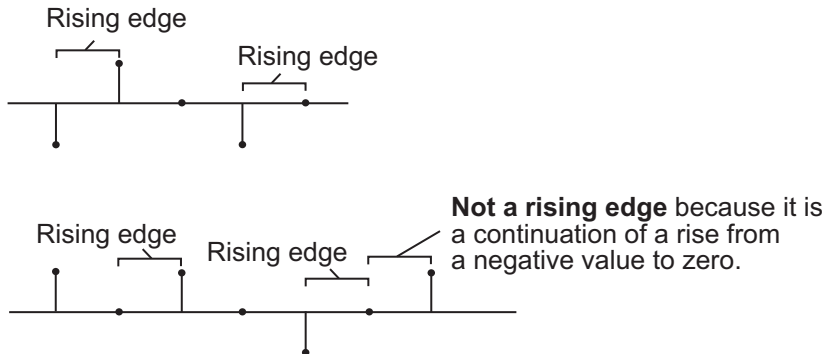
Resetting the Running Mean

The block resets the running mean whenever a reset event is detected at the optional Rst port. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input.

When the block is reset, the running mean associated with each element is initialized to the value in the corresponding location of the current input.

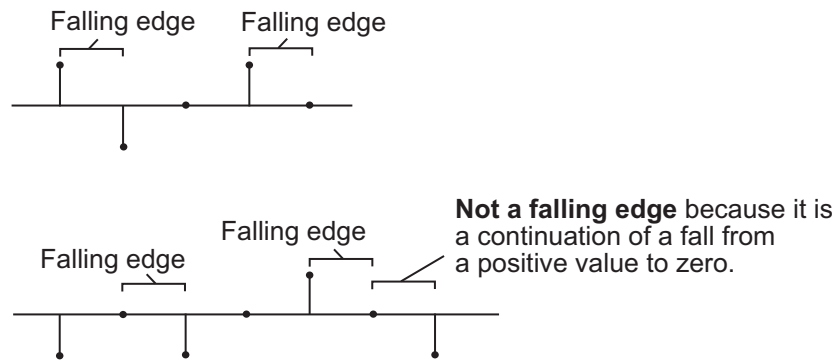
You specify the reset event using the **Reset port** parameter:

- None — Disables the Rst port
- Rising edge — Triggers a reset operation when the Rst input does one of the following:
 - Rises from a negative value to a positive value or 0
 - Rises from 0 to a positive value, where the rise is not a continuation of a rise from a negative value to 0 (see the following figure)



- Falling edge — Triggers a reset operation when the Rst input does one of the following:
 - Falls from a positive value to a negative value or 0
 - Falls from 0 to a negative value, where the fall is not a continuation of a fall from a positive value to 0 (see the following figure)

2-D Mean (Obsolete)



- Either edge — Triggers a reset operation when the Rst input is a Rising edge or Falling edge (as described previously)
- Non-zero sample — Triggers a reset operation at each sample time that the Rst input is not 0

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see “Configuration Parameters Dialog Box” in the Simulink documentation.

ROI Processing

To calculate the statistical value within a particular region of each image, select the **Enable ROI processing** check box. This option is not available when the block is in running mode.

Use the **ROI type** parameter to specify whether the ROI is a rectangle, line, label matrix, or binary mask. A binary mask is a binary image that enables you to specify which pixels to highlight, or select. In a label matrix, pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on. When the **ROI type** parameter is set to Label matrix, the

Label and Label Numbers ports appear on the block. Use the Label Numbers port to specify the objects in the label matrix for which the block calculates statistics. The input to this port must be a vector of scalar values that correspond to the labeled regions in the label matrix. For more information about the format of the input to the ROI port when the ROI is a rectangle or a line, see the Draw Shapes block reference page.

For rectangular ROIs, use the **ROI portion to process** parameter to specify whether to calculate the statistical value for the entire ROI or just the ROI perimeter.

Use the **Output** parameter to specify the block output. The block can output separate statistical values for each ROI or the statistical value for all specified ROIs. This parameter is not available if, for the **ROI type** parameter, you select Binary mask.

If, for the **ROI type** parameter you select Rectangles or Lines, the **Output flag indicating if ROI is within image bounds** check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

Output = Individual statistics for each ROI

Flag Port Output	Description
0	ROI is completely outside the input image.
1	ROI is completely or partially inside the input image.

2-D Mean (Obsolete)

Output = Single statistic for all ROIs

Flag Port Output	Description
0	All ROIs are completely outside the input image.
1	At least one ROI is completely or partially inside the input image.

If the ROI is partially outside the image, the block only computes the statistical values for the portion of the ROI that is within the image.

If, for the **ROI type** parameter you select `Label matrix`, the **Output flag indicating if input label numbers are valid** check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

Output = Individual statistics for each ROI

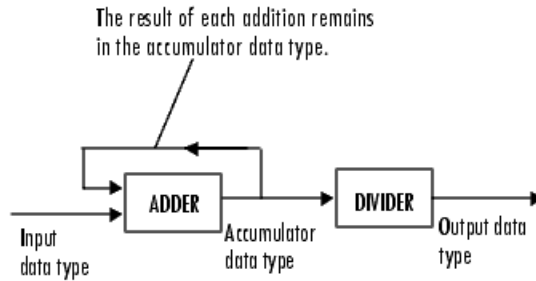
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

Output = Single statistic for all ROIs

Flag Port Output	Description
0	None of the label numbers are in the label matrix.
1	At least one of the label numbers is in the label matrix.

Fixed-Point Data Types

The following diagram shows the data types used in the 2-D Mean block for fixed-point signals.

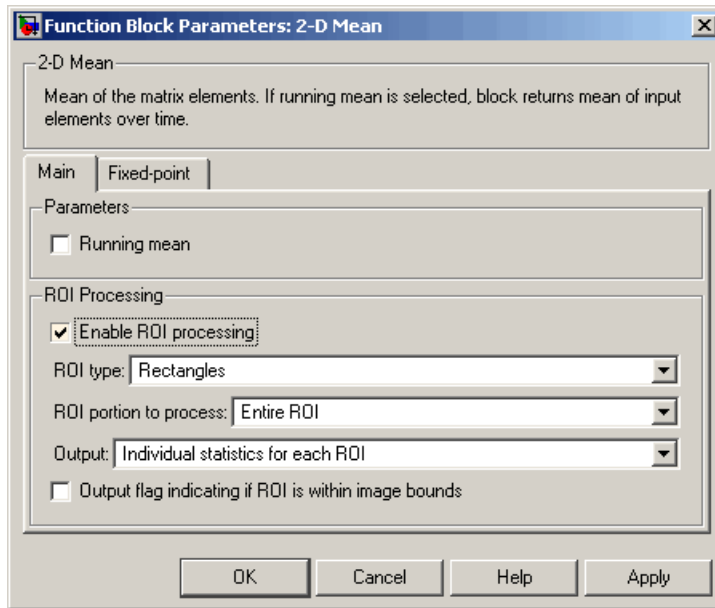


You can set the accumulator and output data types in the dialog box.

2-D Mean (Obsolete)

Dialog Box

The **Main** pane of the 2-D Mean dialog box appears as shown in the following figure.



Running mean

Select this check box to enable the block's running operation.

Reset port

Determines the reset event that causes the block to reset the running mean. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input. This parameter is visible only when you select the **Running mean** check box.

Enable ROI processing

Select this check box to calculate the statistical value within a particular region of each image. This parameter is not available when the block is in running mode.

ROI type

Specify the type of ROI you want to use. Your choices are Rectangles, Lines, Label matrix, or Binary mask.

ROI portion to process

Specify whether you want to calculate the statistical value for the entire ROI or just the ROI perimeter. This parameter is only visible if, for the **ROI type** parameter, you specify Rectangles.

Output

Specify the block output. The block can output a vector of separate statistical values for each ROI or a scalar value that represents the statistical value for all the specified ROIs. This parameter is not available if, for the **ROI type** parameter, you select Binary mask.

Output flag indicating if ROI is within image bounds

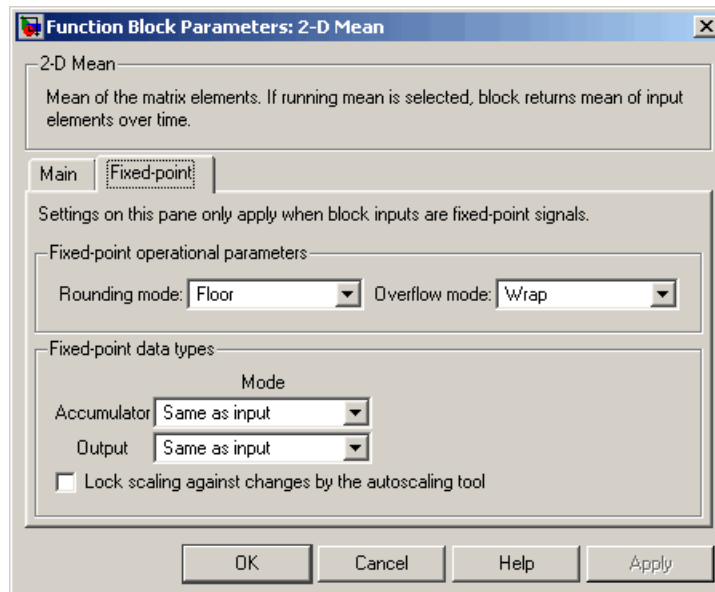
If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in “ROI Processing” on page 2-120. This parameter is visible if, for the **ROI type** parameter, you select Rectangles or Lines.

Output flag indicating if label numbers are valid

If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in “ROI Processing” on page 2-120. This parameter is visible if, for the **ROI type** parameter, you select Label matrix.

The **Fixed-point** pane of the 2-D Mean dialog box appears as shown in the following figure.

2-D Mean (Obsolete)



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Accumulator

Use this parameter to specify the accumulator word and fraction lengths:

- When you select **Same as input**, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of 0.

Output

Choose how to specify the output word length and fraction length:

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of 0.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

See Also

2-D Autocorrelation	Video and Image Processing Blockset software
2-D Correlation	Video and Image Processing Blockset software
Histogram	Video and Image Processing Blockset software
Median	Video and Image Processing Blockset software
Standard Deviation	Video and Image Processing Blockset software
Variance	Video and Image Processing Blockset software
Maximum	Signal Processing Blockset software
Mean	Signal Processing Blockset software

2-D Mean (Obsolete)

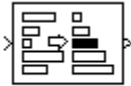
Minimum
mean

Signal Processing Blockset software
MATLAB software

Purpose Find median value of each input matrix

Library Statistics

Description



2-D Median

Note The 2-D Median block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset software. Use the replacement block Median.

The 2-D Median block outputs the median value of the M-by-N input matrix.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, 32-, and 128-bit signed integer• 8-, 16-, 32-, and 128-bit unsigned integer	Yes
Output	Median value of each M-by-N input matrix	Same as Input port	Yes

Length-M 1-D vector inputs are treated as M-by-1 column vectors.

When M is odd, the block sorts the column elements by value, and outputs the central row of the sorted matrix.

2-D Median (Obsolete)

```
s = sort(u(:));  
y = s((M+1)/2)
```

When M is even, the block sorts the column elements by value, and outputs the average of the two central rows in the sorted matrix.

```
s = sort(u(:));  
y = mean([s(M/2), s(M/2+1)])
```

Complex inputs are sorted by *magnitude squared*. For complex value $u = a + bi$, the magnitude squared is $a^2 + b^2$.

Fixed-Point Data Types

For fixed-point inputs, you can specify accumulator, product output, and output data types as discussed in “Dialog Box” on page 2-131. Not all these fixed-point parameters are applicable for all types of fixed-point inputs. The following table shows when each kind of data type and scaling is used.

	Output Data Type	Accumulator Data Type	Product Output Data Type
Even M	X	X	
Odd M	X		
Odd M and complex	X	X	X
Even M and complex	X	X	X

The accumulator and output data types and scalings are used for fixed-point signals when M is even. The result of the sum performed while calculating the average of the two central rows of the input matrix is stored in the accumulator data type and scaling. The total result of the average is then put into the output data type and scaling.

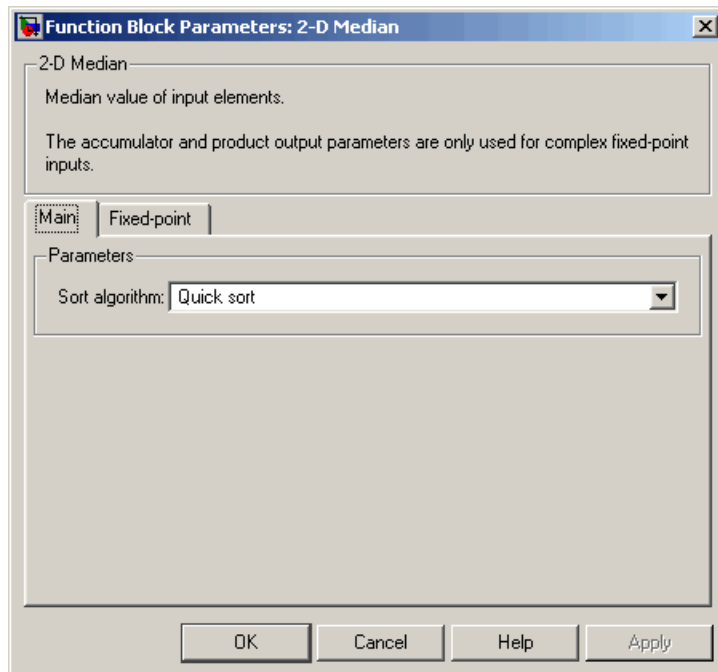
The accumulator and product output parameters are used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before the input elements are sorted.

The results of the squares of the real and imaginary parts are placed into the product output data type and scaling. The result of the sum of the squares is placed into the accumulator data type and scaling.

For fixed-point inputs that are both complex and have even M, the data types are used in all of the ways described. Therefore, in such cases the accumulator type is used in two different ways.

Dialog Box

The **Main** pane of the 2-D Median dialog box appears as shown in the following figure.

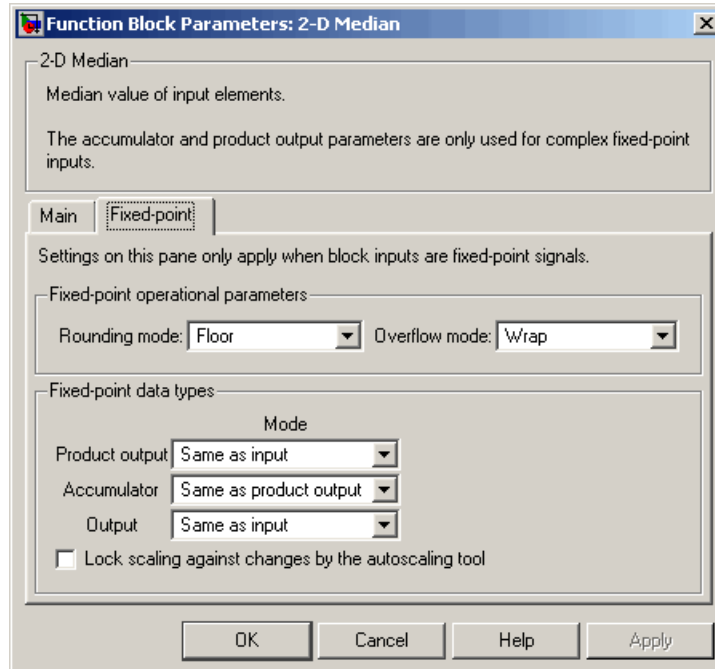


Sort algorithm

Specify whether the elements of the input are sorted using a Quick sort or an Insertion sort algorithm.

2-D Median (Obsolete)

The **Fixed-point** pane of the 2-D Median dialog box appears as shown in the following figure.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Note The product output, accumulator, and output parameters are only used in certain cases. Refer to “Fixed-Point Data Types” on page 2-130 for more information.

Product output

Use this parameter to specify how to designate the product output word and fraction lengths:

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of 0.

Accumulator

Use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block:

- When you select `Same as product output`, these characteristics match those of the product output
- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of 0.

Output

Choose how to specify the output word length and fraction length:

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.

2-D Median (Obsolete)

- When you select **Slope** and **bias scaling**, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of 0.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

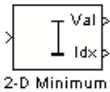
See Also

2-D Autocorrelation	Video and Image Processing Blockset software
2-D Correlation	Video and Image Processing Blockset software
Histogram	Video and Image Processing Blockset software
Mean	Video and Image Processing Blockset software
Standard Deviation	Video and Image Processing Blockset software
Variance	Video and Image Processing Blockset software
Maximum	Signal Processing Blockset software
Median	Signal Processing Blockset software
Minimum	Signal Processing Blockset software
median	MATLAB software

Purpose Find minimum values in an input or sequence of inputs

Library vipobslib

Description



Note The 2-D Minimum block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset software. Use the replacement block Minimum.

The 2-D Minimum block identifies the value and/or position of the smallest element in each column of the input, or tracks the minimum values in a sequence of inputs over a period of time. The **Mode** parameter specifies the block's mode of operation, and can be set to Value, Index, Value and Index, or Running.

The Minimum block supports real and complex floating-point and fixed-point inputs. Fixed-point real inputs can be either signed or unsigned, while fixed-point complex inputs must be signed. The data type of the minimum values output by the block match the data type of the input. The index values output by the block are double when the input is double, and uint32 otherwise.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	Yes
Rst	Scalar value	Boolean	No

2-D Minimum (Obsolete)

Port	Input/Output	Supported Data Types	Complex Values Supported
Val	Minimum value in each M-by-N input matrix	Same as Input port	Yes
Idx	Two-element vector of the form [row index column index] that represents the zero-based location of the minimum value	Same as Input port	No

Length- M 1-D vector inputs are treated as M -by-1 column vectors.

Value Mode

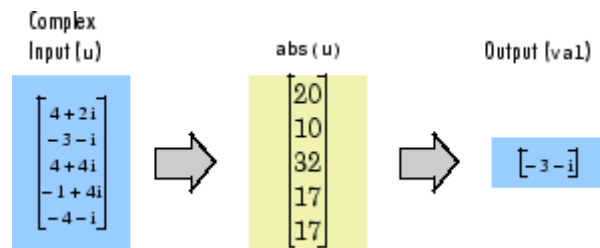
When **Mode** is set to **Value**, the block computes the minimum value in each column of the M -by- N input matrix u independently at each sample time.

```
val = min(u)    % Equivalent MATLAB code
```

For convenience, length- M 1-D vector inputs and *sample-based* length- M row vector inputs are both treated as M -by-1 column vectors.

The output at each sample time, val , is a 1-by- N vector containing the minimum value of each column in u .

For complex inputs, the block selects the value in each matrix that has the minimum magnitude squared as shown in the following figure. For complex value $u = a + bi$, the magnitude squared is $a^2 + b^2$.



Index Mode

When **Mode** is set to **Index**, the block computes the minimum value in each column of the M -by- N input matrix u ,

```
[val,idx] = min(u)      % Equivalent MATLAB code
```

and outputs the sample-based 1-by- N index vector, idx . Each value in idx is an integer in the range $[1M]$ indexing the minimum value in the corresponding column of u . When inputs to the block are double-precision values, the index values are double-precision values. Otherwise, the index values are 32-bit unsigned integer values.

As in **Value** mode, length- M 1-D vector inputs and *sample-based* length- M row vector inputs are both treated as M -by-1 column vectors.

When a minimum value occurs more than once in a particular column of u , the computed index corresponds to the first occurrence. For example, when the input is the column vector $[-1 \ 2 \ 3 \ 2 \ -1]'$, the computed index of the minimum value is 1 rather than 5.

Value and Index Mode

When **Mode** is set to **Value** and **Index**, the block outputs both the vector of minima, val , and the vector of indices, idx .

Running Mode

When **Mode** is set to **Running**, the block tracks the minimum value of each channel in a *time-sequence* of M -by- N inputs. For sample-based inputs, the output is a sample-based M -by- N matrix with each element y_{ij} containing the minimum value observed in element u_{ij} for all inputs since the last reset. For frame-based inputs, the output is a frame-based M -by- N matrix with each element y_{ij} containing the minimum value observed in the j th column of all inputs since the last reset, up to and including element u_{ij} of the current input.

As in the other modes, length- M 1-D vector inputs and *sample-based* length- M row vector inputs are both treated as M -by-1 column vectors.

2-D Minimum (Obsolete)

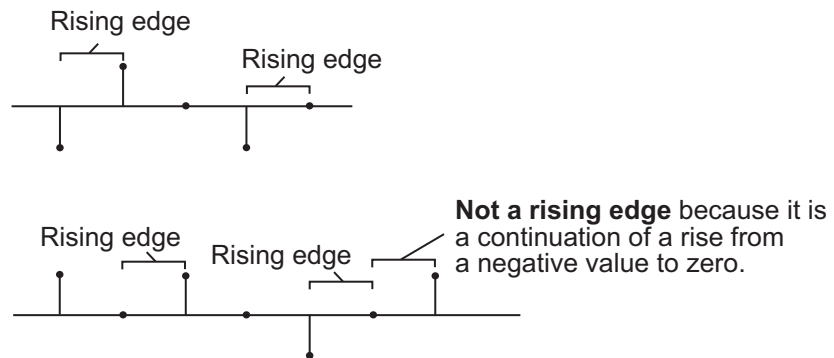
Resetting the Running Minimum

The block resets the running minimum whenever a reset event is detected at the optional Rst port. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input.

When the block is reset for sample-based inputs, the running minimum for each channel is initialized to the value in the corresponding channel of the current input. For frame-based inputs, the running minimum for each channel is initialized to the earliest value in each channel of the current input.

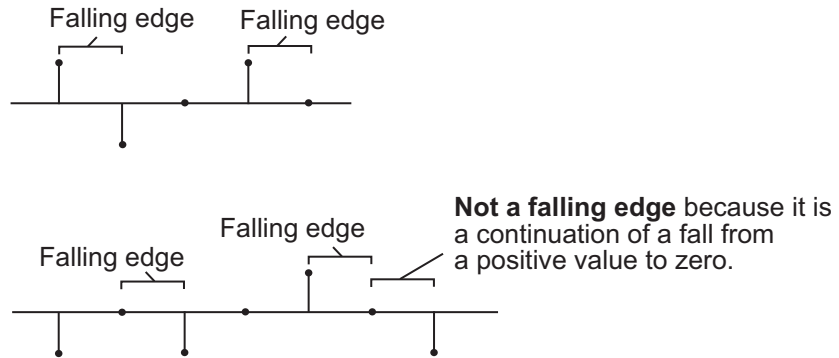
You specify the reset event by the **Reset port** parameter:

- None disables the Rst port.
- **Rising edge** — Triggers a reset operation when the Rst input does one of the following:
 - Rises from a negative value to a positive value or 0
 - Rises from 0 to a positive value, where the rise is not a continuation of a rise from a negative value to 0 (see the following figure)



- **Falling edge** — Triggers a reset operation when the Rst input does one of the following:
 - Falls from a positive value to a negative value or 0

- Falls from 0 to a negative value, where the fall is not a continuation of a fall from a positive value to 0 (see the following figure)



- Either edge — Triggers a reset operation when the Rst input is a Rising edge or Falling edge (as described previously)
- Non-zero sample — Triggers a reset operation at each sample time that the Rst input is not 0

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see “Configuration Parameters Dialog Box” in the Simulink documentation.

Fixed-Point Data Types

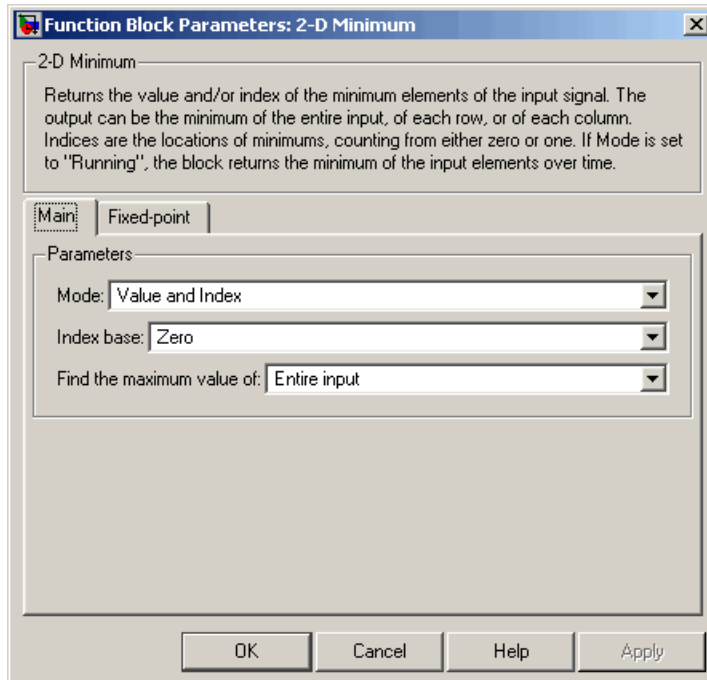
The parameters on the **Fixed-point** pane of the dialog box are only used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before a comparison is made, as described in “Value Mode” on page 2-136. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into

2-D Minimum (Obsolete)

the accumulator data type. These parameters are ignored for other types of inputs.

Dialog Box

The **Main** pane of the 2-D Minimum dialog box appears as shown in the following figure.



Mode

Specify the block's mode of operation:

- **Value** — Output the minimum value of each input matrix
- **Index** — Output the zero-based index location of the minimum value
- **Value and Index** — Output both the value and the index location

- **Running** — Track the minimum value of the input sequence over time

Index base

Specify whether the index is zero based or one based.

Find the maximum value of

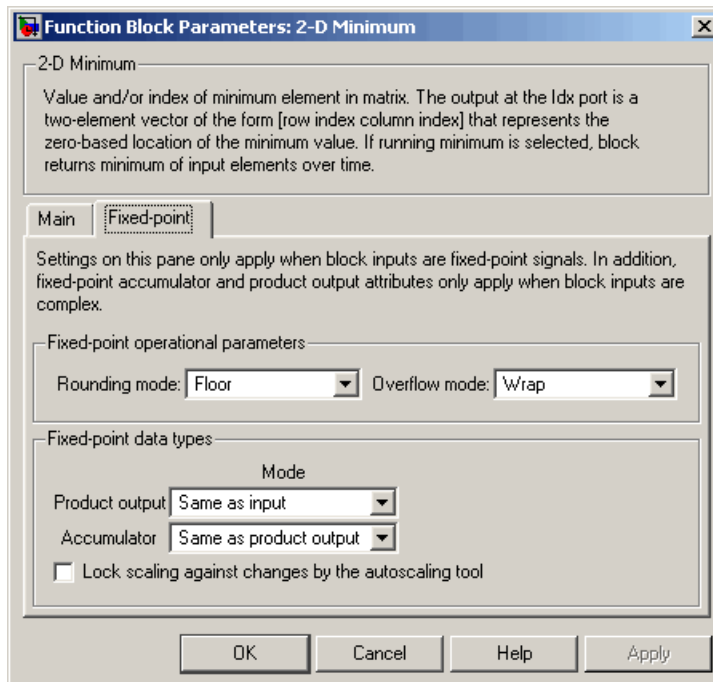
Specify whether the block should find the maximum of the entire input or of each row or column.

Reset port

Specify the reset event detected at the Rst input port when you select **Running** for the **Mode** parameter. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input. This parameter is only visible if, for the **Mode** parameter, you select **Running**.

The **Fixed-point** pane of the 2-D Minimum dialog box appears as shown in the following figure.

2-D Minimum (Obsolete)



Note The parameters on the **Fixed-point** pane are only used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before a comparison is made, as described in “Value Mode” on page 2-136. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ignored for other types of inputs.

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how to designate the product output word and fraction lengths resulting from a complex-complex multiplication in the block. Refer to “Multiplication Data Types” in the Signal Processing Blockset documentation for more information:

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of 0.

Accumulator

Use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block. Refer to “Multiplication Data Types” in the Signal Processing Blockset documentation for more information:

- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of 0.

2-D Minimum (Obsolete)

Lock scaling against changes by the autoscaling tool

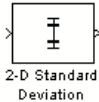
Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

2-D Standard Deviation (Obsolete)

Purpose Find standard deviation of each input matrix

Library Statistics

Description



Note The 2-D Standard Deviation block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset software. Use the replacement block Standard Deviation.

The 2-D Standard Deviation block computes the standard deviation of each M-by-N input matrix or of a sequence of inputs over time. Use the **Running standard deviation** check box to select between the block's basic and running operation. This block's functionality is different from the Signal Processing Blockset Standard Deviation block, which computes the standard deviation of each column in the input.

2-D Standard Deviation (Obsolete)

Port	Input/Output	Supported Data Types	Complex Values Supported
Input / I	Vector or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point 	Yes
Rst	Signal that triggers a reset event	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
ROI	<ul style="list-style-type: none"> • Rectangle — [r c height width] • Lines — [r1 c1 r2 c2], where r1 and c1 are the row and column coordinates of the beginning of the line and r2 and c2 are the row and column coordinates of the end of the line. • Binary mask — Binary image matrix that enables you to specify which pixels to highlight. 	Rectangles and lines — <ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer Binary mask — <ul style="list-style-type: none"> • Boolean 	No

2-D Standard Deviation (Obsolete)

Port	Input/Output	Supported Data Types	Complex Values Supported
Label	Matrix where pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on.	<ul style="list-style-type: none"> 8-, 16-, and 32-bit unsigned integer 	No
Label Numbers	Vector containing the label numbers for the regions for which the block will compute the statistics.	<ul style="list-style-type: none"> 8-, 16-, and 32-bit unsigned integer 	No
Output / Out	<p>Without ROI processing — Standard deviation of each M-by-N input matrix, or the standard deviation of a sequence of M-by-N inputs.</p> <p>With ROI processing — Vector of separate statistical values for each ROI or a scalar value that represents the statistical value for all specified ROIs.</p>	Same as Input port	Yes
Flag	Boolean value that indicates whether the ROI is within the image bounds or the label number is within the label matrix.	Boolean	No

2-D Standard Deviation (Obsolete)

Length-M 1-D vector inputs are treated as M-by-1 column vectors.

Basic Operation

If you clear the **Running standard deviation** check box, the block outputs the standard deviation of each M-by-N input matrix.

For purely real or purely imaginary inputs, the standard deviation is the square root of the variance and is given by the following equation:

$$y = \sigma = \sqrt{\frac{\sum_{i=1}^M \sum_{j=1}^N (u_{ij} - \mu)^2}{M \times N - 1}}$$

where μ is the mean of the input matrix u . For complex inputs, the block outputs the total standard deviation of the input matrix, which is the square root of the total variance.

$$\sigma = \sqrt{\sigma_{\text{Re}}^2 + \sigma_{\text{Im}}^2}$$

The total standard deviation is not equal to the sum of the real and imaginary standard deviations.

Running Operation

If you select the **Running standard deviation** check box, the block computes the standard deviation of a sequence of M-by-N inputs.

For example, suppose A is the first input to the block and B is the second and current input to the block, where

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

and

$$B = \begin{bmatrix} 5 & 6 \\ 7 & 3 \end{bmatrix}$$

2-D Standard Deviation (Obsolete)

The block computes the standard deviation as

$$\begin{bmatrix} \text{std}([1,5]) & \text{std}([3,6]) \\ \text{std}([2,7]) & \text{std}([4,3]) \end{bmatrix}$$

and outputs

$$\begin{bmatrix} 2.8284 & 2.1213 \\ 3.5355 & 0.7071 \end{bmatrix}$$

For the next input, the block computes the standard deviation for each element of the first three inputs, and so on.

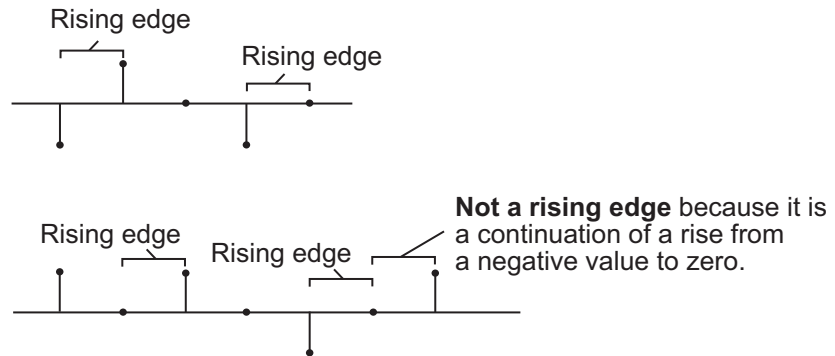
Resetting the Running Standard Deviation

The block resets the running standard deviation whenever a reset event is detected at the optional Rst port. The reset signal rate must be a positive integer multiple of the rate of the data signal input.

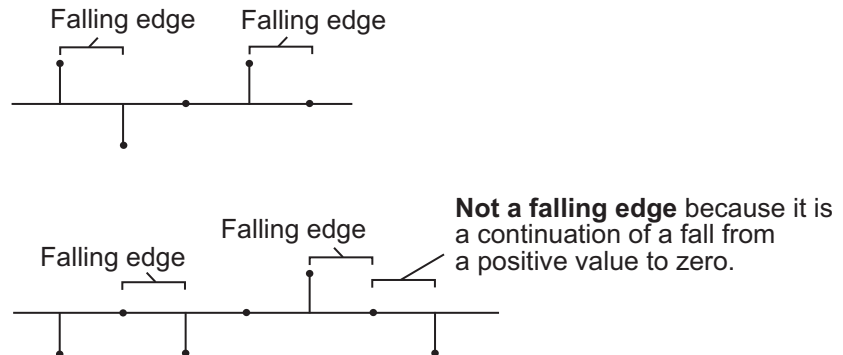
You specify the reset event using the **Reset port** parameter:

- **None** — Disables the Rst port
- **Rising edge** — Triggers a reset operation when the Rst input does one of the following:
 - Rises from a negative value to a positive value or 0
 - Rises from 0 to a positive value, where the rise is not a continuation of a rise from a negative value to 0 (see the following figure)

2-D Standard Deviation (Obsolete)



- Falling edge — Triggers a reset operation when the Rst input does one of the following:
 - Falls from a positive value to a negative value or 0
 - Falls from 0 to a negative value, where the fall is not a continuation of a fall from a positive value to 0 (see the following figure)



- Either edge — Triggers a reset operation when the Rst input is a Rising edge or Falling edge (as described previously).
- Non-zero sample — Triggers a reset operation at each sample time that the Rst input is not 0.

2-D Standard Deviation (Obsolete)

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see “Configuration Parameters Dialog Box” in the Simulink documentation.

ROI Processing

To calculate the statistical value within a particular region of each image, select the **Enable ROI processing** check box. This option is not available when the block is in running mode.

Use the **ROI type** parameter to specify whether the ROI is a rectangle, line, label matrix, or binary mask. A binary mask is a binary image that enables you to specify which pixels to highlight, or select. In a label matrix, pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on. When the **ROI type** parameter is set to `Label matrix`, the Label and Label Numbers ports appear on the block. Use the Label Numbers port to specify the objects in the label matrix for which the block calculates statistics. The input to this port must be a vector of scalar values that correspond to the labeled regions in the label matrix. For more information about the format of the input to the ROI port when the ROI is a rectangle or a line, see the Draw Shapes block reference page.

For rectangular ROIs, use the **ROI portion to process** parameter to specify whether to calculate the statistical value for the entire ROI or just the ROI perimeter.

Use the **Output** parameter to specify the block output. The block can output separate statistical values for each ROI or the statistical value for all specified ROIs. This parameter is not available if, for the **ROI type** parameter, you select `Binary mask`.

If, for the **ROI type** parameter you select `Rectangles` or `Lines`, the **Output flag indicating if ROI is within image bounds** check box

2-D Standard Deviation (Obsolete)

appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

Output = Individual statistics for each ROI

Flag Port Output	Description
0	ROI is completely outside the input image.
1	ROI is completely or partially inside the input image.

Output = Single statistic for all ROIs

Flag Port Output	Description
0	All ROIs are completely outside the input image.
1	At least one ROI is completely or partially inside the input image.

If the ROI is partially outside the image, the block only computes the statistical values for the portion of the ROI that is within the image.

If, for the **ROI type** parameter you select `Label matrix`, the **Output flag indicating if input label numbers are valid** check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

2-D Standard Deviation (Obsolete)

Output = Individual statistics for each ROI

Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

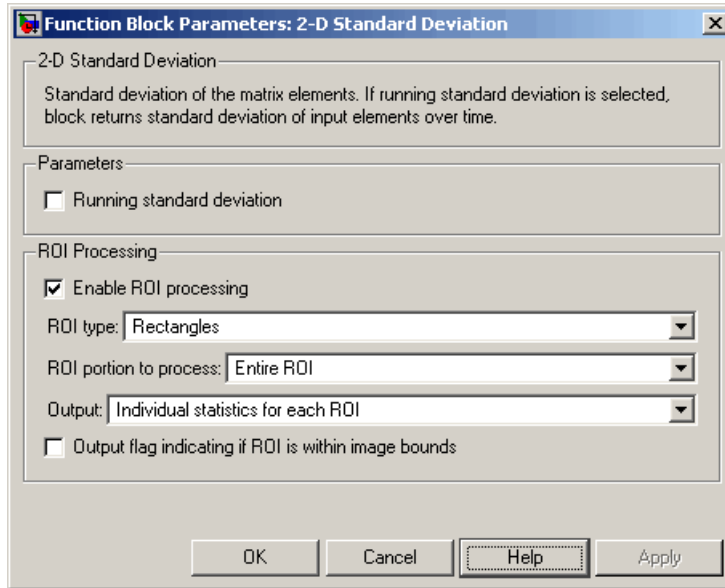
Output = Single statistic for all ROIs

Flag Port Output	Description
0	None of the label numbers are in the label matrix.
1	At least one of the label numbers is in the label matrix.

2-D Standard Deviation (Obsolete)

Dialog Box

The 2-D Standard Deviation dialog box appears as shown in the following figure.



Running standard deviation

Select this check box to enable the block's running operation.

Reset port

Determines the reset event that causes the block to reset the running standard deviation. The reset signal rate must be a positive integer multiple of the rate of the data signal input. This parameter is available if you select the **Running standard deviation** check box.

Enable ROI processing

Select this check box to calculate the statistical value within a particular region of each image. This parameter is not available when the block is in running mode.

2-D Standard Deviation (Obsolete)

ROI type

Specify the type of ROI to use. Your choices are Rectangles, Lines, Label matrix, or Binary mask.

ROI portion to process

Specify whether you want to calculate the statistical value for the entire ROI or just the ROI perimeter. This parameter is only visible if, for the **ROI type** parameter, you specify Rectangles.

Output

Specify the block output. The block can output a vector of separate statistical values for each ROI or a scalar value that represents the statistical value for all the specified ROIs. This parameter is not available if, for the **ROI type** parameter, you select Binary mask.

Output flag indicating if ROI is within image bounds

If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in “ROI Processing” on page 2-151. This parameter is visible if, for the **ROI type** parameter, you select Rectangles or Lines.

Output flag indicating if label numbers are valid

If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in “ROI Processing” on page 2-151. This parameter is visible if, for the **ROI type** parameter, you select Label matrix.

See Also

2-D Autocorrelation	Video and Image Processing Blockset software
2-D Correlation	Video and Image Processing Blockset software
Histogram	Video and Image Processing Blockset software
Mean	Video and Image Processing Blockset software

2-D Standard Deviation (Obsolete)

Median	Video and Image Processing Blockset software
Variance	Video and Image Processing Blockset software
Maximum	Signal Processing Blockset software
Minimum	Signal Processing Blockset software
Standard Deviation	Signal Processing Blockset software
std	MATLAB software

2-D Variance (Obsolete)

Purpose Compute variance of each input matrix

Library Statistics

Description



Note The 2-D Variance block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset software. Use the replacement block Variance.

The 2-D Variance block computes the variance of each M-by-N input matrix or of a sequence of inputs over time. Use the **Running variance** check box to choose between the block's basic and running operation.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input / I	Vector or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	Yes
Rst	Signal that triggers a reset event	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No

2-D Variance (Obsolete)

Port	Input/Output	Supported Data Types	Complex Values Supported
ROI	<ul style="list-style-type: none"> • Rectangle — [r c height width] • Lines — [r1 c1 r2 c2], where r1 and c1 are the row and column coordinates of the beginning of the line and r2 and c2 are the row and column coordinates of the end of the line. • Binary mask — Binary image matrix that enables you to specify which pixels to highlight. 	<ul style="list-style-type: none"> • Rectangles and lines <ul style="list-style-type: none"> ▪ Double-precision floating point ▪ Single-precision floating point ▪ Boolean ▪ 8-, 16-, and 32-bit signed integer ▪ 8-, 16-, and 32-bit unsigned integer • Binary mask <ul style="list-style-type: none"> ▪ Boolean 	No
Label	Matrix where pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on.	<ul style="list-style-type: none"> • 8-, 16-, and 32-bit unsigned integer 	No
Label Numbers	Vector containing the label numbers for the regions for which the block will compute the statistics.	<ul style="list-style-type: none"> • 8-, 16-, and 32-bit unsigned integer 	No

2-D Variance (Obsolete)

Port	Input/Output	Supported Data Types	Complex Values Supported
Output/Out	Without ROI processing — Variance of each M-by-N input matrix or the variance for each element in a sequence of M-by-N inputs. With ROI processing — Vector of separate statistical values for each ROI or a scalar value that represents the statistical value for all specified ROIs.	Same as Input port	Yes
Flag	Boolean value that indicates whether the ROI is within the image bounds or the label number is within the label matrix.	Boolean	No

Length-M 1-D vector inputs are treated as M-by-1 column vectors.

Basic Operation

If you clear the **Running variance** check box, the block outputs the variance of each M-by-N input matrix. A scalar input generates a zero-valued output.

For purely real or purely imaginary inputs, the variance of a M-by-N matrix is the square of the standard deviation:

2-D Variance (Obsolete)

$$y = \sigma^2 = \frac{\sum_{i=1}^M \sum_{j=1}^N |u_{ij}|^2 - \frac{\left| \sum_{i=1}^M \sum_{j=1}^N u_{ij} \right|^2}{M * N}}{M * N - 1}$$

For complex inputs, the variance is given by the following equation:

$$\sigma^2 = \sigma_{\text{Re}}^2 + \sigma_{\text{Im}}^2$$

When the input values are double-precision floating point, the equivalent MATLAB code is `var(u(:))`, where `u` is the input.

Running Operation

If you select the **Running variance** check box, the block computes the variance of a sequence of M-by-N inputs.

For example, suppose `A` is the first input to the block and `B` is the second and current input to the block, where

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

and

$$B = \begin{bmatrix} 5 & 6 \\ 7 & 3 \end{bmatrix}$$

The block computes the variance,

$$\begin{bmatrix} \text{var}([1,5]) & \text{var}([3,6]) \\ \text{var}([2,7]) & \text{var}([4,3]) \end{bmatrix}$$

and outputs

$$\begin{bmatrix} 8 & 4.5 \\ 12.5 & 0.5 \end{bmatrix}$$

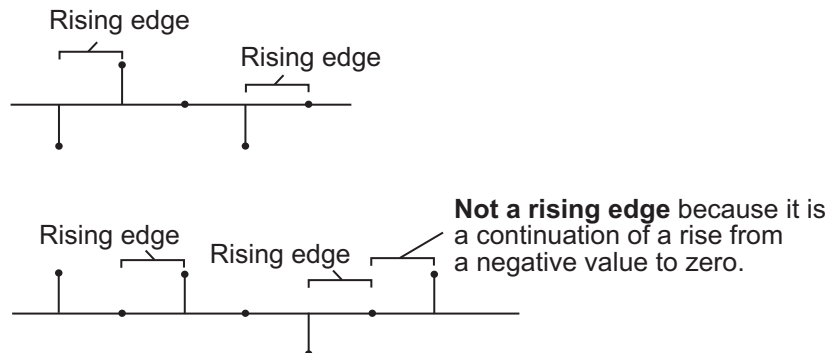
For the next input, the block computes the variance for each element of the first three inputs, and so on.

Resetting the Running Variance

The block resets the running variance whenever a reset event is detected at the optional Rst port. The reset signal rate must be a positive integer multiple of the rate of the data signal input.

You specify the reset event using the **Reset port** parameter:

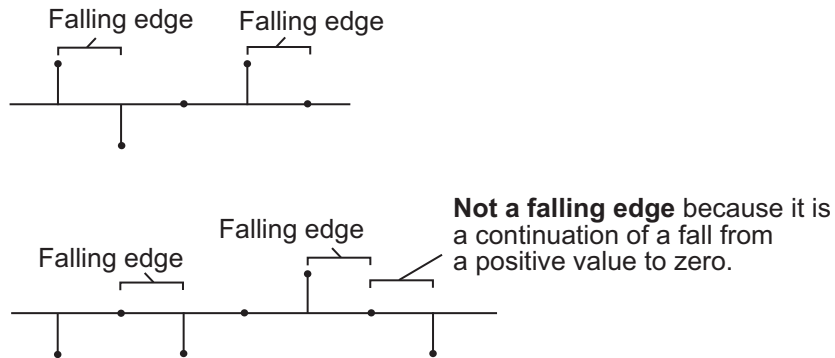
- **None** — Disables the Rst port
- **Rising edge** — Triggers a reset operation when the Rst input does one of the following:
 - Rises from a negative value to a positive value or 0
 - Rises from 0 to a positive value, where the rise is not a continuation of a rise from a negative value to 0 (see the following figure)



- **Falling edge** — Triggers a reset operation when the Rst input does one of the following:

2-D Variance (Obsolete)

- Falls from a positive value to a negative value or 0
- Falls from 0 to a negative value, where the fall is not a continuation of a fall from a positive value to 0 (see the following figure)



- Either edge — Triggers a reset operation when the Rst input is a Rising edge or Falling edge (as described previously)
- Non-zero sample — Triggers a reset operation at each sample time that the Rst input is not 0

Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see “Configuration Parameters Dialog Box” in the Simulink documentation.

ROI Processing

To calculate the statistical value within a particular region of each image, select the **Enable ROI processing** check box. This option is not available when the block is in running mode.

Use the **ROI type** parameter to specify whether the ROI is a binary mask, label matrix, rectangle, or line.

- A binary mask is a binary image that enables you to specify which pixels to highlight, or select.
- In a label matrix, pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on. When the **ROI type** parameter is set to **Label matrix**, the Label and Label Numbers ports appear on the block. Use the Label Numbers port to specify the objects in the label matrix for which the block calculates statistics. The input to this port must be a vector of scalar values that correspond to the labeled regions in the label matrix.
- For more information about the format of the input to the ROI port when the ROI is a rectangle or a line, see the Draw Shapes reference page.

Note For rectangular ROIs, use the **ROI portion to process** parameter to specify whether to calculate the statistical value for the entire ROI or just the ROI perimeter.

Use the **Output** parameter to specify the block output. The block can output separate statistical values for each ROI or the statistical value for all specified ROIs. This parameter is not available if, for the **ROI type** parameter, you select **Binary mask**.

If, for the **ROI type** parameter you select **Rectangles** or **Lines**, the **Output flag indicating if ROI is within image bounds** check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

2-D Variance (Obsolete)

Output = Individual Statistics for Each ROI

Flag Port Output	Description
0	ROI is completely outside the input image.
1	ROI is completely or partially inside the input image.

Output = Single Statistic for All ROIs

Flag Port Output	Description
0	All ROIs are completely outside the input image.
1	At least one ROI is completely or partially inside the input image.

If the ROI is partially outside the image, the block only computes the statistical values for the portion of the ROI that is within the image.

If, for the **ROI type** parameter you select `Label matrix`, the **Output flag indicating if input label numbers are valid** check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output based on the block parameters.

Output = Individual Statistics for Each ROI

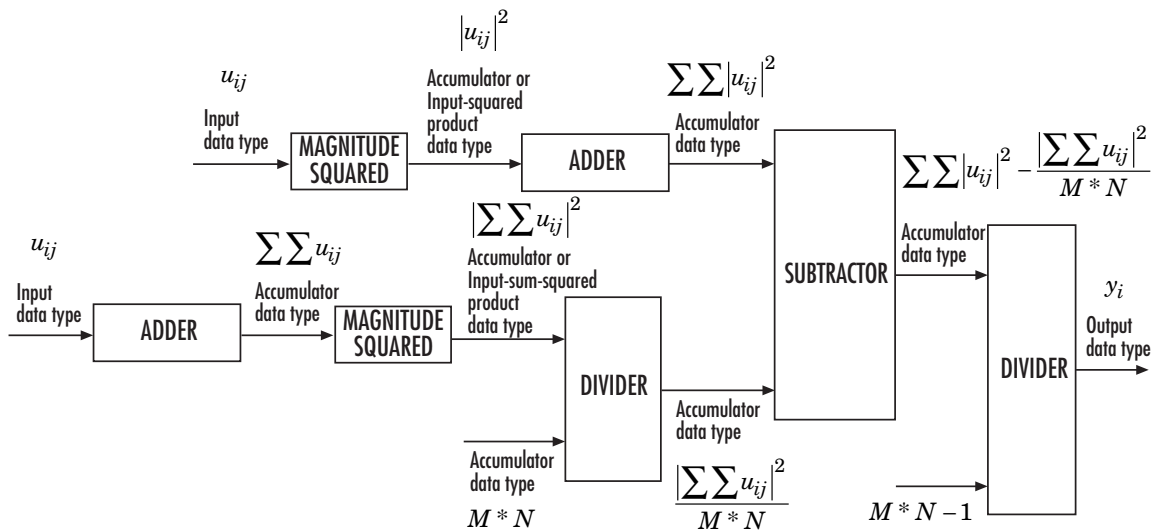
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

Output = Single Statistic for All ROIs

Flag Port Output	Description
0	None of the label numbers are in the label matrix.
1	At least one of the label numbers is in the label matrix.

Fixed-Point Data Types

The following diagram shows the data types used in the 2-D Variance block for fixed-point signals.

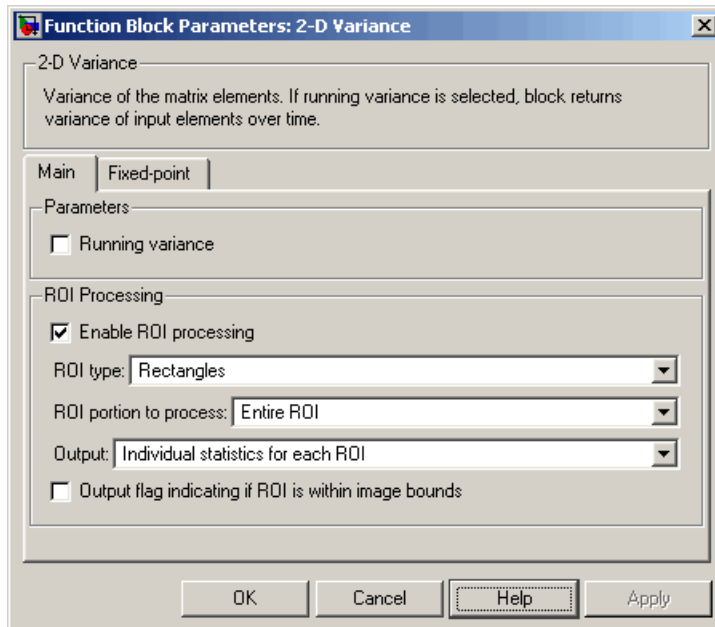


The results of the magnitude squared calculations in the preceding diagram are in the product output data type. You can set the accumulator, product output, and output data types in the dialog box.

2-D Variance (Obsolete)

Dialog Box

The **Main** pane of the 2-D Variance dialog box appears as shown in the following figure.



Running variance

Select this check box to enable the block's running operation.

Reset port

Determines the reset event that causes the block to reset the running variance. The reset signal rate must be a positive integer multiple of the rate of the data signal input. This parameter is visible only if you select the **Running variance** check box.

Enable ROI processing

Select this check box to calculate the statistical value within a particular region of each image. This parameter is not available when the block is in running mode.

ROI type

Specify the type of ROI to use. Your choices are Rectangles, Lines, Label matrix, or Binary mask.

ROI portion to process

Specify whether to calculate the statistical value for the entire ROI or just the ROI perimeter. This parameter is only visible if, for the **ROI type** parameter, you specify Rectangles.

Output

Specify the block output. The block can output a vector of separate statistical values for each ROI or a scalar value that represents the statistical value for all the specified ROIs. This parameter is not available if, for the **ROI type** parameter, you select Binary mask.

Output flag indicating if ROI is within image bounds

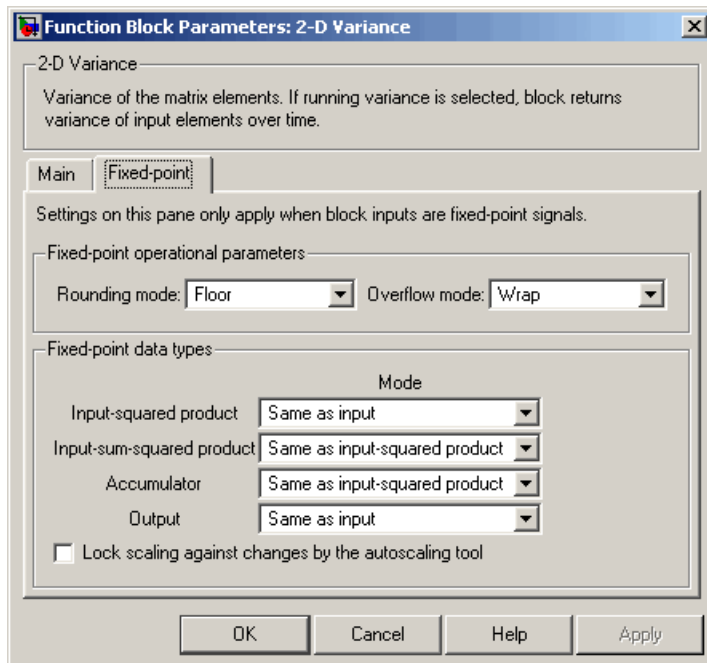
If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in “ROI Processing” on page 2-162. This parameter is visible if, for the **ROI type** parameter, you select Rectangles or Lines.

Output flag indicating if label numbers are valid

If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in “ROI Processing” on page 2-162. This parameter is visible if, for the **ROI type** parameter, you select Label matrix.

The **Fixed-point** pane of the 2-D Variance dialog box appears as shown in the following figure:

2-D Variance (Obsolete)



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Note Refer to “Fixed-Point Data Types” on page 2-165 for more information on how the product output, accumulator, and output data types are used in this block.

Input-squared product

Use this parameter to specify how to designate the input-squared product word and fraction lengths:

- When you select **Same as input**, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the input-squared product, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the input-squared product. This block requires power-of-two slope and a bias of 0.

Input-sum-squared product

Use this parameter to specify how to designate the input-sum-squared product word and fraction lengths:

- When you select **Same as input-squared product**, these characteristics match those of the input-squared product.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the input-sum-squared product, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the input-sum-squared product. This block requires power-of-two slope and a bias of 0.

Note To compute the required fixed-point settings for this parameter, pick your brightest image and sum all the pixel values across the image. Then, square the value. Specify a word length and fraction length so that the resulting value fits within the **Input-sum-squared product** data type without overflow. This specification might require picking a large scaling factor (LSB weight 2^L , $L > 1$), or, in other words, setting a negative fraction length.

2-D Variance (Obsolete)

Accumulator

Use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block:

- When you select `Same as input-squared product`, these characteristics match those of the input-squared product.
- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of 0.

Output

Choose how to specify the output word length and fraction length:

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of 0.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

See Also

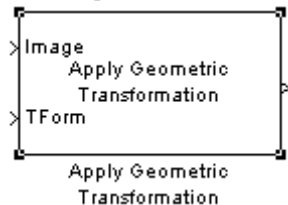
2-D Autocorrelation	Video and Image Processing Blockset software
2-D Correlation	Video and Image Processing Blockset software
Histogram	Video and Image Processing Blockset software
Mean	Video and Image Processing Blockset software
Median	Video and Image Processing Blockset software
Standard Deviation	Video and Image Processing Blockset software
Maximum	Signal Processing Blockset software
Minimum	Signal Processing Blockset software
Variance	Signal Processing Blockset software
var	MATLAB software

Apply Geometric Transformation

Purpose Apply projective or affine transformation to an image

Library Geometric Transformations

Description



Use the Apply Geometric Transformation block to apply projective or affine transform to an image. You can use this block to transform the entire image or portions of the image with either polygon or rectangle Regions of Interest (ROIs).

Port	Description
Image	M -by- N or M -by- N -by- P input matrix. M : Number of rows in the image. N : Number of columns in the image. P : Number of color planes in the image.
TForm	When you set the Transformation matrix source parameter to Input port, the TForm input port accepts: <ul style="list-style-type: none">• 2-by-3 matrix (affine transform) or 6-by-Q matrix (multiple affine transforms)• 3-by-3 matrix (projective transform) or 9-by-Q matrix (multiple projective transforms) Q : Number of transformations.
ROI	When you set the ROI source parameter to Input port, the ROI input port accepts: <ul style="list-style-type: none">• 4-element vector rectangle ROI

Apply Geometric Transformation

Port	Description
	<ul style="list-style-type: none">• $2L$-element vector polygon ROI• 4-by-R matrix for multiple rectangle ROIs• $2L$-by-R matrix for multiple polygon ROIs <p>R: Number of Region of Interests (ROIs).</p> <p>L ($L \geq 3$): Number of vertices in a polygon ROI.</p>

Transformations

The size of the transformation matrix will dictate the transformation type. See the table above for details.

Affine Transformation

For affine transformation, the value of the pixel located at $[\hat{x}, \hat{y}]^T$ in the output image, is determined by the value of the pixel located at $[x, y]^T$ in the input image. The relationship between the input and the output point locations is defined by the following equations:

$$\begin{cases} \hat{x} = xh_1 + yh_2 + h_3 \\ \hat{y} = xh_4 + yh_5 + h_6 \end{cases}$$

where h_1, h_2, \dots, h_6 are transformation coefficients.

If you use one transformation, the transformation coefficients must be arranged as a 2 -by- 3 matrix as in:

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \end{bmatrix}$$

or in a 6 -by- 1 vector as in $H = [h_1 \ h_2 \ h_3 \ h_4 \ h_5 \ h_6]^T$.

If you use more than one transformation, the transformation coefficients must be arranged as a 6 -by- Q matrix, where each column has the

Apply Geometric Transformation

format of $H = [h_1 \ h_2 \ h_3 \ h_4 \ h_5 \ h_6]^T$, and Q is the number of transformations as in:

$$H = \begin{bmatrix} h_{11} & h_{21} & \dots & h_{Q1} \\ h_{12} & h_{22} & \dots & h_{Q2} \\ \cdot & \cdot & \dots & \cdot \\ h_{16} & h_{26} & \dots & h_{Q6} \end{bmatrix}$$

Projective Transformation

For projective transformation, the relationship between the input and the output points is defined by the following equations:

$$\begin{cases} \hat{x} = \frac{xh_1 + yh_2 + h_3}{xh_7 + yh_8 + h_9} \\ \hat{y} = \frac{xh_4 + yh_5 + h_6}{xh_7 + yh_8 + h_9} \end{cases}$$

where h_1, h_2, \dots, h_9 are transformation coefficients.

If you use one transformation, the transformation coefficients must be arranged as a 3-by-3 matrix as in:

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

or in a 9-by-1 vector as in, $H = [h_1 \ h_2 \ h_3 \ h_4 \ h_5 \ h_6 \ h_7 \ h_8 \ h_9]^T$.

If you use more than one transformation, the transformation coefficients must be arranged as a 9-by- Q matrix, where each column has the

format of $H = [h_1 \ h_2 \ h_3 \ h_4 \ h_5 \ h_6 \ h_7 \ h_8 \ h_9]^T$, and Q is the number of transformations. For example,

$$H = \begin{bmatrix} h_{11} & h_{21} & \dots & h_{Q1} \\ h_{12} & h_{22} & \dots & h_{Q2} \\ \cdot & \cdot & \dots & \cdot \\ h_{19} & h_{29} & \dots & h_{Q9} \end{bmatrix}$$

Region of Interest

The Apply Geometric Transformation block can apply transformations to the entire image, or a portion (or portions) of an image specified by ROIs.

Region of Interest (ROI) Types

Two types of ROIs are supported. ROIs can be one or more rectangles or polygons.

A **rectangle** ROI is specified by its top-left corner and its size. If you specify one ROI, it must be a *4-element* vector of format $[r, c, h, w]^T$. If you specify more than one ROI, it must be a *4-by-R* matrix such that each column is $[r, c, h, w]^T$.

A **polygon** ROI is specified by the vertices of the polygon in clockwise or counter-clockwise order, with at least three or more vertices. If you specify one ROI of *L* vertices, it must be a *2L-element* vector of format $[r1, c1, r2, c2, \dots, rL, cL]^T$. If you specify more than one ROI, it must be a *2L-by-R* matrix, where now *L* is the maximum number of vertices in the ROIs. For ROI with vertices fewer than *L*, its last vertex can be repeated to form a vector of *2L*. See “Defining Shapes to Draw” on page 2-337 for details.

ROI Processing

The transformations will be applied on the whole image, or on specified multiple ROIs. The table below outlines how transformation matrices are handled with an entire image and with single and multiple ROIs.

Apply Geometric Transformation

Number of Transformation Matrices	Region of Interest
One transformation matrix	You can apply the transformation on the entire image, single ROI or multiple ROIs.
Multiple transformation matrices	<ul style="list-style-type: none">• You can apply multiple transformation matrices on one ROI or on the entire image. The transformations are done in the order they are entered in the TForm.• You can apply multiple transformation matrices on multiple ROIs. Each transformation matrix is applied to one ROI. The first transformation matrix specified is applied to the first ROI specified. The second transformation matrix is applied to the second ROI specified, and so on. The number of transformation matrices must be equal to the number of ROIs.

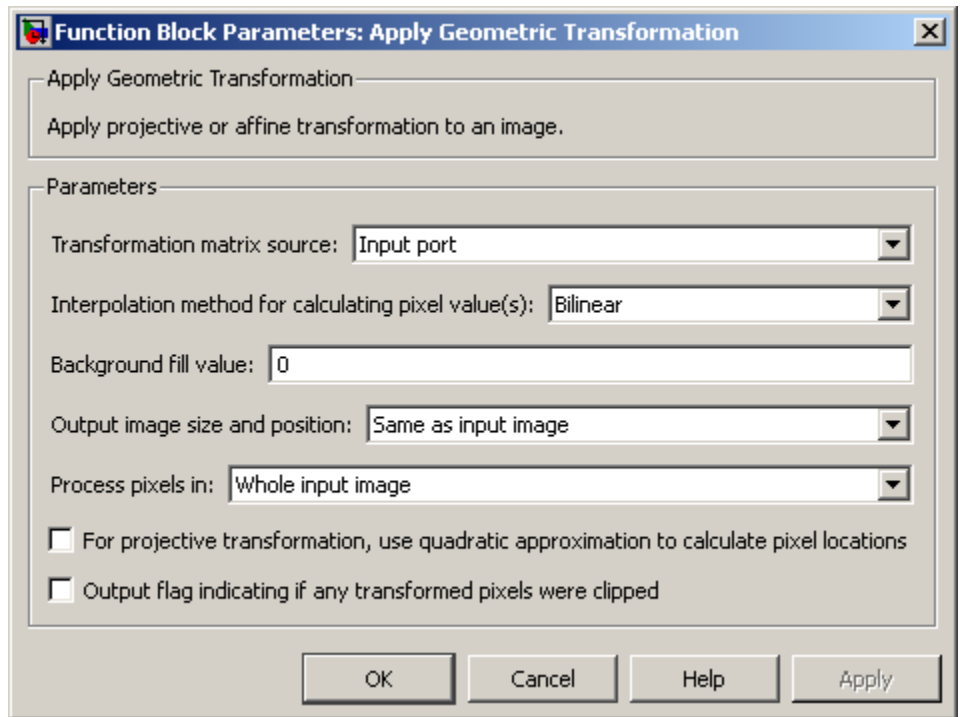
Quadratic Approximation Mode for Projective Transformation

Projective Transformation block provides an approximation mode which reduces the number of pixels requiring division calculations [1]. The accuracy of the approximation to determine pixel locations is specified by the user in the **Error tolerance** parameter.

Dialog Box

The **Main** pane of the Apply Geometric Transformation dialog box appears as shown in the following figure.

Apply Geometric Transformation



Transformation matrix source

Specify input matrix source, either Specified via dialog, or Input port. If you select Specify via dialog, you can enter the transformation matrix parameters in the parameter that appear with this selection.

Transformation matrix

Specify a 2-by-3, 3-by-3, 6-by-Q, or a 9-by-Q matrix. This option appears when you set Transformation matrix source to Specified via dialog.

Interpolation method for calculating pixel value(s)

Specify interpolation method, either Nearest neighbor, Bilinear, or Bicubic interpolation to calculate output pixel values. See Geometric

Apply Geometric Transformation

Transformation Interpolation Methods for an overview of these methods.

Background fill value

Specify the value of the pixels that are outside of the input image. Use either a scalar value or P-element vector.

Output image size and position

Specify the output image size to be either Same as input image, or Specify via dialog. If you select to Specify via dialog, you can specify the bounding box in the size and location parameters that appear with this selection.

Size [height width]

Specify the height and width for the output image size as [height width]. You can specify this parameter, along with the **Location of the upper left corner [row column]** parameter when you set the **Output image size and position** parameter is set to Specify via dialog.

Location of the upper left corner [row col]

Specify the row and column location for the upper left corner of the output image. You can specify this parameter, along with the **Size [height width]** parameter, when you set the **Output image size and position** parameter to Specify via dialog.

Process pixels in

Specify the region to process pixels in. Specify Whole input image, Rectangle ROI, or Polygon ROI. If you select Rectangle ROI, or Polygon ROI the **ROI source** parameter becomes available.

ROI source

Specify the source for the region of interest (ROI), either Specify via dialog or Input port. This parameter is available when you set the **Process pixels in** parameter to either Rectangle ROI, or Polygon ROI.

Location and size of rectangle ROI [row col height width]

Specify a 4-element vector or 4-by- R matrix, (where R represents the number of ROIs). This parameter is available when the **Process pixels in** parameter is set to Rectangle ROI.

Apply Geometric Transformation

Vertices of polygon ROI [r1 c1 r2 c2 ... rn cn]

Specify a $2L$ -element vector or $2L$ -by- R matrix, (where L is the number of vertices in a polygon and R represents the number of ROIs).

This parameter becomes available when you set **Process pixels in** parameter to Polygon ROI.

Output flag indicating if any part of ROI is outside input image

Select the **Output flag indicating if any part of ROI is outside input image** check box to enable this output port on the Apply Geometric Transformation block.

For projective transformation, use quadratic approximation to calculate pixel locations

Specify whether to use an exact computation or an approximation for the projective transformation. If you select this option, you can enter an error tolerance in the **Error tolerance (in pixels)** parameter.

Error tolerance (in pixels)

Specify the maximum error tolerance in pixels. This parameter becomes available when you select **For projective transformation, use quadratic approximation to calculate pixel locations** check box.

Output flag indicating if any transformed pixels were clipped

Select the **Output flag indicating if any transformed pixels were clipped** check box to enable this output port on the Apply Geometric Transformation block. Clipping occurs when any of the transformed pixels fall outside of the output image.

Examples

“Apply a Projective Transformation to an Image” on page 2-180	A simple model using the Apply Geometric Transformation block.
“Create an Image of a Cube using Three Regions of Interest” on page 2-180	A model which uses the Apply Geometric Transformation block to create an image of a cube using ROIs.

Apply Geometric Transformation

Apply a Projective Transformation to an Image

The Simple projective transformation model `doc_vipApplyGeo_proj`, uses the Apply Geometric Transformation block, two Constant blocks and two Video Viewer blocks to illustrate a basic model. The transformation matrix determines a projective transformation and is applied to the entire input image. The input image is a checker board. The steps taken to run this model were:

- 1 Add two Constant blocks for the input image and the transformation matrix. Set the **Constant value** parameters for the constant blocks as follows:
 - for the input image, "checker_board", and
 - for the transformation matrix, $[0.06 \ -0.15 \ 15; \ 0.04 \ 0.1 \ 2; \ -0.01 \ 0 \ 1]$
- 2 Add two Video Viewer blocks, connecting one directly to the input image output port, and the other one to the Apply Geometric Transformation output port.

Create an Image of a Cube using Three Regions of Interest

This example shows how to apply affine transformation on multiple ROIs of an image. It also sets the background color of the output image to a solid color purple. The input image, transformation matrix, and ROI vertices are provided to the Apply Geometric Transformation block via constant blocks. Video viewers are used to view the original image and the output image created. Open this model by typing `doc_vipApplyGeo_roi` at the MATLAB command prompt. The steps taken to run this model was:

- 1 Change the **Process pixels in** parameter to Polygon ROI.
- 2 Change the **Background fill value** to $[0.5 \ 0.5 \ 0.75]$
- 3 Add three Constant blocks for the input image, transformation matrix, and ROI vertices. Set the **Constant value** parameters for the three blocks as follows:

Apply Geometric Transformation

- For the input image, "checker_board"
 - For the transformation matrix, [1 0 -15 0 1 15; 0.4082 0 15 -0.4082 1.0204 35; 1 -0.4082 5.4082 0 0.4082 44.5918] ', and
 - For the polygon ROI, [50 0 50 49 99 49 99 0; 0 0 0 49 49 49 49 0; 50 50 50 99 99 99 99 50] '.
- 4 Add two Video Viewer blocks, connecting one directly to the Constant block containing the input image. The other, to the Apply Geometric Transformation output port.

References

[1] George Wolberg, "Digital Image Warping", IEEE Computer Society Press, 3rd edition, 1994.

Richard Hartley and Andrew Zisserman, "Multiple View Geometry in Computer Vision", Cambridge University Press, 2nd edition, 2003.

Supported Data Types

Port	Supported Data Types
Image	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point
TForm	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point
ROI	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• 8-, 16-, and 32-bit signed integers• 8-, 16-, and 32-bit unsigned integers
Output	Same as input

Apply Geometric Transformation

Port	Supported Data Types
Err_roi	Boolean
Err_clip	Boolean

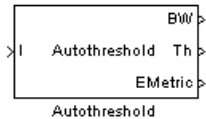
See Also

imtransform	Image Processing Toolbox
Estimate Geometric Transformation	Video and Image Processing Blockset
Trace Boundaries	Video and Image Processing Blockset
Blob Analysis	Video and Image Processing Blockset
Video and Image Processing Demos	Video and Image Processing Blockset

Purpose Convert intensity image to binary image

Library Conversions
vipconversions

Description The Autothreshold block converts an intensity image to a binary image using a threshold value computed using Otsu's method.



This block computes this threshold value by splitting the histogram of the input image such that the variance of each pixel group is minimized.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
BW	Scalar, vector, or matrix that represents a binary image	Boolean	No
Th	Threshold value	Same as I port	No
EMetric	Effectiveness metric	Same as I port	No

Use the **Thresholding operator** parameter to specify the condition the block places on the input values. If you select > and the input value is greater than the threshold value, the block outputs 1 at the BW port; otherwise, it outputs 0. If you select <= and the input value is less

Autothreshold

than or equal to the threshold value, the block outputs 1; otherwise, it outputs 0.

Select the **Output threshold** check box to output the calculated threshold values at the Th port.

Select the **Output effectiveness metric** check box to output values that represent the effectiveness of the thresholding at the EMetric port. This metric ranges from 0 to 1. If every pixel has the same value, the effectiveness metric is 0. If the image has two pixel values or the histogram of the image pixels is symmetric, the effectiveness metric is 1.

If you clear the **Specify data range** check box, the block assumes that floating-point input values range from 0 to 1. To specify a different data range, select this check box. The **Minimum value of input** and **Maximum value of input** parameters appear in the dialog box. Use these parameters to enter the minimum and maximum values of your input signal.

Use the **When data range is exceeded** parameter to specify the block's behavior when the input values are outside the expected range. The following options are available:

- **Ignore** — Proceed with the computation and do not issue a warning message. If you choose this option, the block performs the most efficient computation. However, if the input values exceed the expected range, the block produces incorrect results.
- **Saturate** — Change any input values outside the range to the minimum or maximum value of the range and proceed with the computation.
- **Warn and saturate** — Display a warning message in the MATLAB Command Window, saturate values, and proceed with the computation.
- **Error** — Display an error dialog box and terminate the simulation.

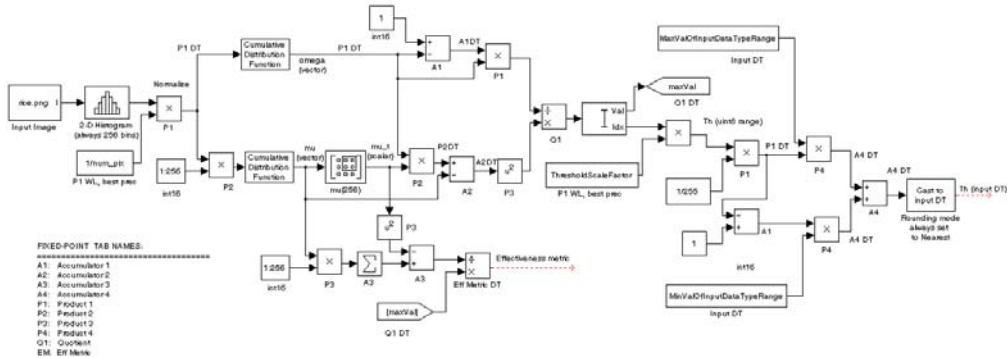
If you clear the **Scale threshold** check box, the block uses the threshold value computed by Otsu's method to convert intensity images

into binary images. If you select the **Scale threshold** check box, the **Threshold scaling factor** appears in the dialog box. Enter a scalar value. The block multiplies this scalar value with the threshold value computed by Otsu's method and uses the result as the new threshold value.

Fixed-Point Data Types

The following diagram shows the data types used in the Autothreshold block for fixed-point signals. You can use the default fixed-point parameters if your input has a word length less than or equal to 16.

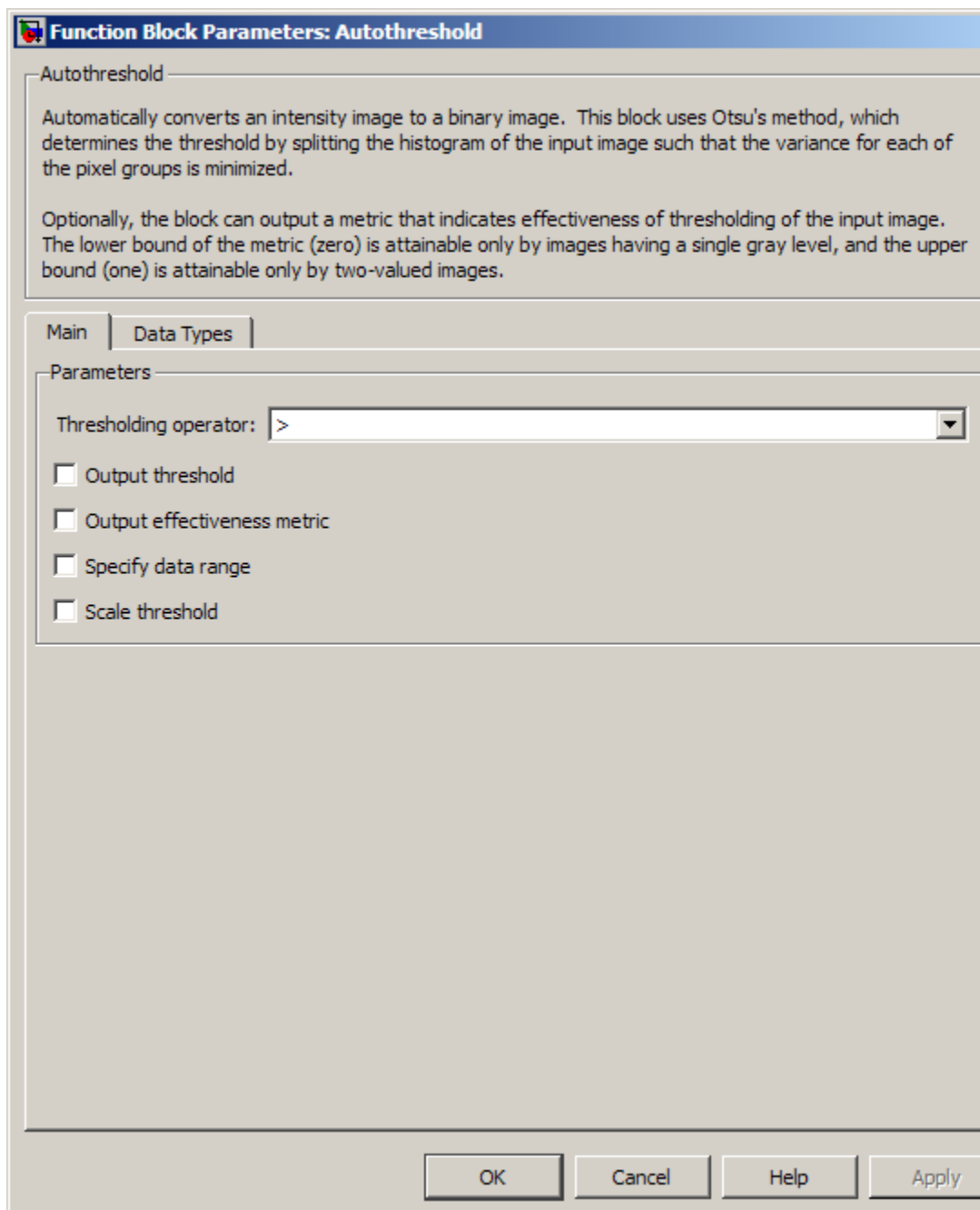
In this diagram, DT means data type. You can set the product, accumulator, quotient, and effectiveness metric data types in the block mask.



Autothreshold

Dialog Box

The **Main** pane of the Autothreshold dialog box appears as shown in the following figure.



Thresholding operator

Specify the condition the block places on the input matrix values. If you select > or <=, the block outputs 0 or 1 depending on whether the input matrix values are above, below, or equal to the threshold value.

Output threshold

Select this check box to output the calculated threshold values at the Th port.

Output effectiveness metric

Select this check box to output values that represent the effectiveness of the thresholding at the EMetric port.

Specify data range

If you clear this check box, the block assumes that floating-point input values range from 0 to 1. To specify a different data range, select this check box.

Minimum value of input

Enter the minimum value of your input data. This parameter is visible if you select the **Specify data range** check box.

Maximum value of input

Enter the maximum value of your input data. This parameter is visible if you select the **Specify data range** check box.

When data range is exceeded

Specify the block's behavior when the input values are outside the expected range. Your options are Ignore, Saturate, Warn and saturate, or Error. This parameter is visible if you select the **Specify data range** check box.

Scale threshold

Select this check box to scale the threshold value computed by Otsu's method.

Threshold scaling factor

Enter a scalar value. The block multiplies this scalar value with the threshold value computed by Otsu's method and uses the

Autothreshold

result as the new threshold value. This parameter is visible if you select the **Scale threshold** check box.

The **Data Types pane** of the Autothreshold dialog box appears as follows. You can use the default fixed-point parameters if your input has a word length less than or equal to 16.

Function Block Parameters: Autothreshold

Autothreshold

Automatically converts an intensity image to a binary image. This block uses Otsu's method, which determines the threshold by splitting the histogram of the input image such that the variance for each of the pixel groups is minimized.

Optionally, the block can output a metric that indicates effectiveness of thresholding of the input image. The lower bound of the metric (zero) is attainable only by images having a single gray level, and the upper bound (one) is attainable only by two-valued images.

Main | Data Types

Settings on this pane only apply when block inputs are fixed-point signals.

Fixed-point operational parameters

Rounding mode: Overflow mode:

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Product 1	<input type="text" value="Specify word length"/>	Yes	<input type="text" value="32"/>	30
Accumulator 1	<input type="text" value="Same as Product 1"/>			
Product 2	<input type="text" value="Specify word length"/>	Yes	<input type="text" value="32"/>	22
Accumulator 2	<input type="text" value="Same as Product 2"/>			
Product 3	<input type="text" value="Specify word length"/>	Yes	<input type="text" value="32"/>	14
Product 4	<input type="text" value="Binary point scaling"/>	Same as input	<input type="text" value="32"/>	<input type="text" value="15"/>
Accumulator 4	<input type="text" value="Same as Product 4"/>			
Quotient	<input type="text" value="Specify word length"/>	Yes	<input type="text" value="32"/>	16

Lock data type settings against changes by the fixed-point tools

Rounding mode

Select the rounding mode for fixed-point operations. This parameter does not apply to the Cast to input DT step shown in “Fixed-Point Data Types” on page 2-185. For this step, **Rounding mode** is always set to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations.

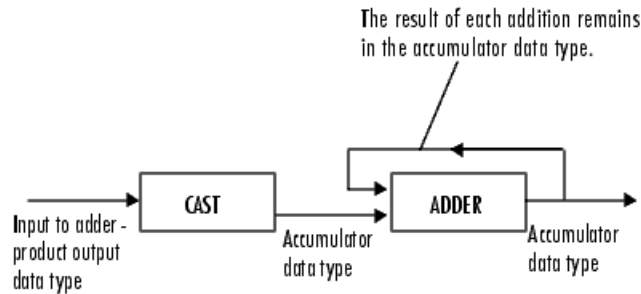
Product 1, 2, 3, 4



As shown previously, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate the product output word and fraction lengths.

- When you select **Specify word length**, you can enter the word length of the product values in bits. The block sets the fraction length to give you the best precision.
- When you select **Same as input**, the characteristics match those of the input to the block. This choice is only available for the **Product 4** parameter.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output in bits.
- When you select **Slope and bias scaling**, you can enter the word length in bits and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset software is 0.

Accumulator 1, 2, 3, 4



As shown previously, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate the accumulator word and fraction lengths.

- When you select **Same as Product**, these characteristics match those of the product output.
- When you select **Specify word length**, you can enter the word length of the accumulator values in bits. The block sets the fraction length to give you the best precision. This choice is not available for the **Accumulator 4** parameter because it is dependent on the input data type.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator in bits.
- When you select **Slope and bias scaling**, you can enter the word length in bits and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset software is 0.

The **Accumulator 3** parameter is only visible if, on the **Main** pane, you select the **Output effectiveness metric** check box.

Quotient

Choose how to specify the word length and fraction length of the quotient data type:

- When you select **Specify word length**, you can enter the word length of the quotient values in bits. The block sets the fraction length to give you the best precision.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the quotient, in bits.
- When you select **Slope and bias scaling**, you can enter the word length in bits and the slope of the quotient. The bias of all signals in the Video and Image Processing Blockset software is 0.

Eff Metric

Choose how to specify the word length and fraction length of the effectiveness metric data type. This parameter is only visible if, on the **Main** tab, you select the **Output effectiveness metric** check box.

- When you select **Specify word length**, you can enter the word length of the effectiveness metric values, in bits. The block sets the fraction length to give you the best precision.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the effectiveness metric in bits.
- When you select **Slope and bias scaling**, you can enter the word length in bits and the slope of the effectiveness metric. The bias of all signals in the Video and Image Processing Blockset software is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

See Also

Compare To Constant

Simulink

Relational Operator

Simulink

graythresh

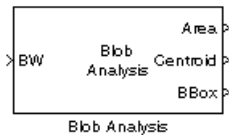
Image Processing Toolbox

Blob Analysis

Purpose Compute statistics for labeled regions

Library Statistics

Description Use the Blob Analysis block to calculate statistics for labeled regions in a binary image. The block returns quantities such as the Centroid, label matrix, and blob count.



The Blob Analysis block supports variable size signals at the input and output.

For information on pixel and spatial coordinate system definitions, see “Coordinate Systems” in the *Video and Image Processing Blockset User’s Guide*.

Use the Variable Selector block to select certain blobs based on their statistics. For more information about this block, see the Variable Selector block reference page in the Signal Processing Blockset documentation.

Port	Input/Output	Supported Data Types	Complex Values Supported
BW	Vector or matrix that represents a binary image	Boolean	No
Area	Vector that represents the number of pixels in labeled regions	32-bit signed integer	No
Centroid	2-by-N matrix of centroid coordinates, where N is the number of blobs	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point	No

Port	Input/Output	Supported Data Types	Complex Values Supported
BBox	4-by-N matrix of bounding box coordinates, where N is the number of blobs	32-bit signed integer	No
MajorAxis	Vector that represents the lengths of major axes of ellipses	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point 	No
MinorAxis	Vector that represents the lengths of minor axes of ellipses	Same as MajorAxis port	No
Orientation	Vector that represents the angles between the major axes of the ellipses and the <i>x</i> -axis.	Same as MajorAxis port	No
Eccentricity	Vector that represents the eccentricities of the ellipses	Same as MajorAxis port	No
Diameter ^2	Vector that represents the equivalent diameters squared	Same as Centroid port	No
Extent	Vector that represents the results of dividing the areas of the blobs by the area of their bounding boxes	Same as Centroid port	No

Blob Analysis

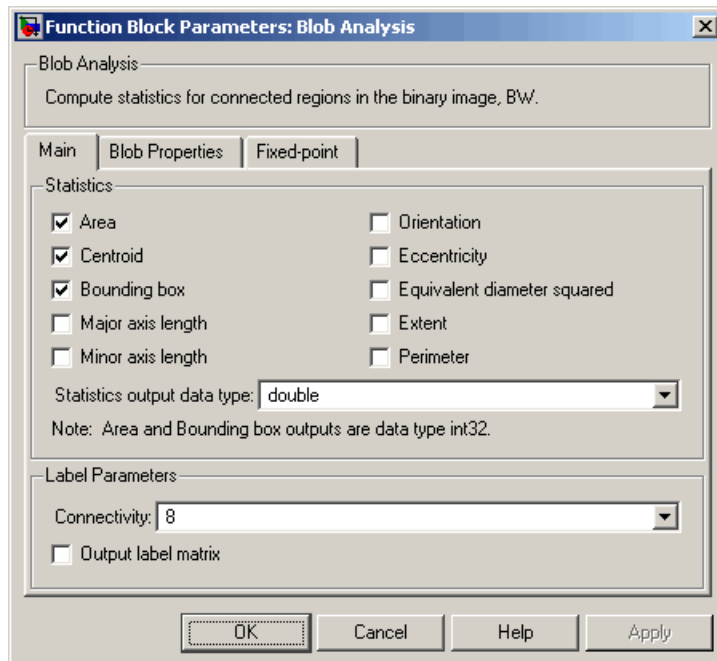
Port	Input/Output	Supported Data Types	Complex Values Supported
Perimeter	Vector containing an estimate of the perimeter length, in pixels, for each blob	Same as Centroid port	No
Label	Label matrix	8-, 16-, or 32-bit unsigned integer	No
Count	Scalar value that represents the actual number of labeled regions in each image	Same as Label port	No

Dialog Box

Use the check boxes to specify the statistics values you want the block to output. The following table summarizes the Blob Analysis block behavior based on which check box you select. For a full description of each of these statistics, see the `regionprops` function reference page in the Image Processing Toolbox documentation.

Main tab

The following table describes parameters that appear in the Main tab of Blob Analysis.



Parameter	Value
Area	Select this check box to output a vector that represents the number of pixels in labeled regions
Centroid	<p>Select this check box to output a $2\text{-by-}N$ matrix. The columns represent the coordinates of the centroid of each region, where N is the number of blobs.</p> <p><i>Example:</i> Suppose there are two blobs, where the row and column coordinates of their centroids are $r1$, $c1$ and $r2$, $c2$, respectively. The block outputs</p> $\begin{bmatrix} r1 & r2 \\ c1 & c2 \end{bmatrix}$ <p>at the Centroid port.</p>

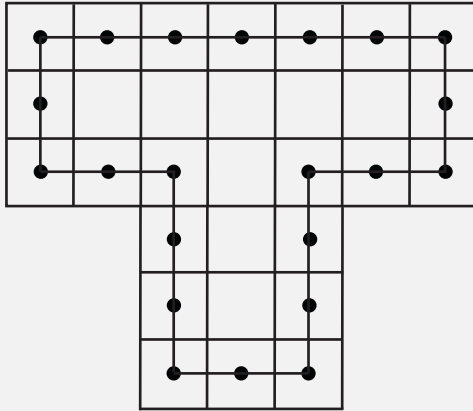
Blob Analysis

Parameter	Value
Bounding box	<p>Select this check box to output a $4\text{-by-}N$ matrix. The columns represent the coordinates of each bounding box, where N is the number of blobs.</p> <p><i>Example:</i> Suppose there are two blobs, where r and c define the row and column location of the upper-left corner of the bounding box. Where w and h define the width and height of the bounding box. The block outputs</p> $\begin{bmatrix} r1 & r2 \\ c1 & c2 \\ h1 & h2 \\ w1 & w2 \end{bmatrix}$ <p>at the BBox port.</p>
Major axis length	<p>Select this check box to output a vector with the following characteristics:</p> <ul style="list-style-type: none"> • Represents the lengths of the major axes of ellipses • Has the same normalized second central moments as the labeled regions
Minor axis length	<p>Select this check box to output a vector with the following characteristics:</p> <ul style="list-style-type: none"> • Represents the lengths of the minor axes of ellipses • Has the same normalized second central moments as the labeled regions
Orientation	<p>Select this check box to output a vector that represents the angles between the major axes of the ellipses and the x-axis. The angle values are in radians and range between:</p> $-\frac{\pi}{2} \text{ and } \frac{\pi}{2}$

Parameter	Value
Eccentricity	Select this check box to output a vector that represents the eccentricities of ellipses that have the same second moments as the region
Equivalent diameter squared	Select this check box to output a vector that represents the equivalent diameters squared
Extent	Select this check box to output a vector that represents the results of dividing the areas of the blobs by the area of their bounding boxes
Perimeter	Select this check box to output an N -by-1 vector of the perimeter lengths, in pixels, of each blob, where N is the number of blobs.
Statistics output data type	Use this parameter to specify the data type of the outputs at the Centroid , MajorAxis , MinorAxis , Orientation , Eccentricity , Equivalent diameter squared , and Extent ports. If you select Fixed-point , the block cannot calculate the major axis, minor axis, orientation, or eccentricity and the associated check boxes become unavailable.
Connectivity	<p>Use this parameter to define which pixels connect to each other. If you want to connect pixels located on the top, bottom, left, and right, select 4. If you want to connect pixels to the other pixels on the top, bottom, left, right, and diagonally, select 8. For more information about this parameter, see the Label block reference page. The Connectivity parameter also affects how the block calculates the perimeter of a blob. For example:</p> <p>The following figure illustrates how the block calculates the perimeter when you set the Connectivity parameter to 4.</p>

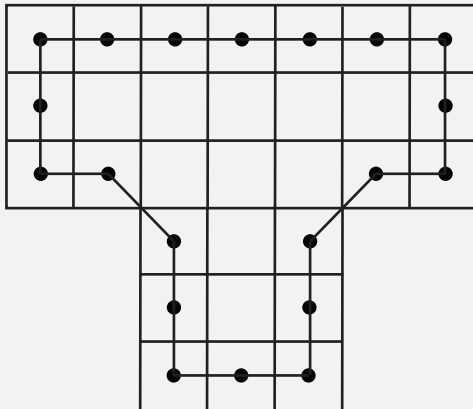
Blob Analysis

Parameter	Value
-----------	-------



The block calculates the distance between the center of each pixel (marked by the black dots) and estimates the perimeter to be 22.

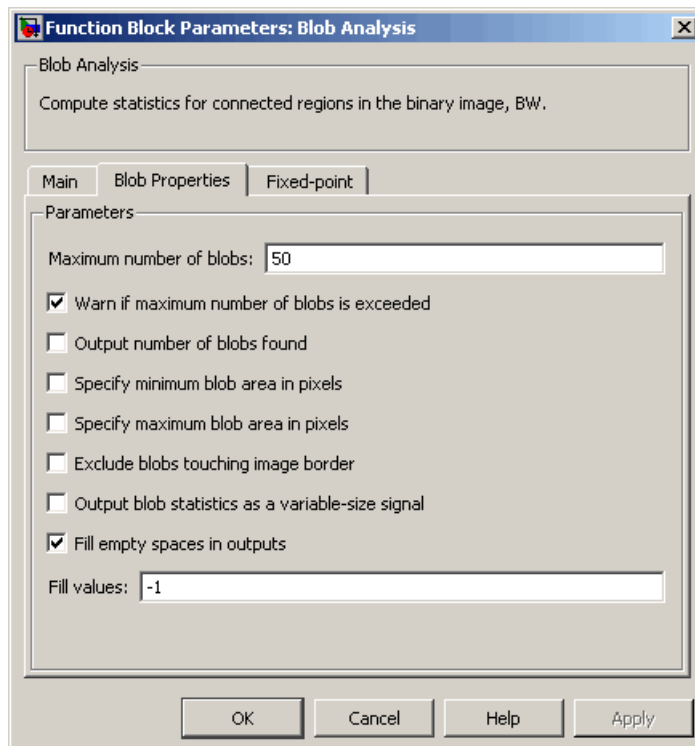
The next figure illustrates how the block calculates the perimeter of a blob when you set the **Connectivity** parameter to 8.



Parameter	Value
	The block takes a different path around the blob and estimates the perimeter to be $18 + 2\sqrt{2}$.
Output label matrix	Select this check box, to output the label matrix at the Label port. The pixels equal to 0 represent the background. The pixels equal to 1 represent the first object. The pixels equal to 2 represent the second object, and so on.

Blob Properties tab

The following table describes the parameters of the Blob Properties tab.



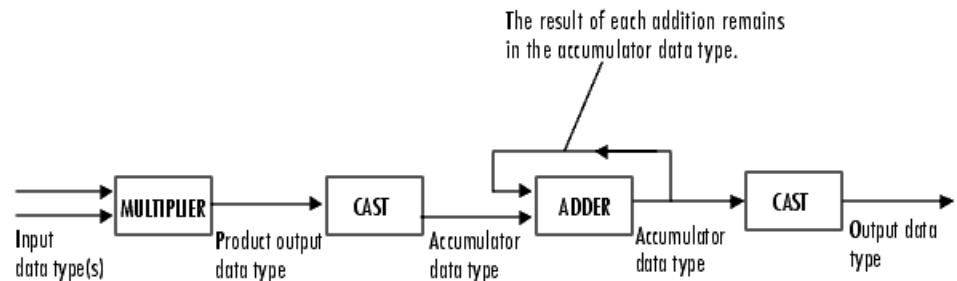
Blob Analysis

Parameter	Value
Maximum number of blobs	Specify the maximum number of labeled regions in each input image. The block uses this value to preallocate vectors and matrices to ensure that they are long enough to hold the statistical values.
Warn if maximum number of blobs is exceeded	Select this check box to output a warning when the number of blobs in an image is greater than the value of Maximum number of blobs parameter.
Output number of blobs found	Select this check box to output a scalar value that represents the actual number of connected regions in each image at the Count port.
Specify maximum blob area in pixels	Select this check box to enter the minimum blob area in the edit box that appears beside the check box. The blob gets a label if the number of pixels meets the minimum size specified. The maximum allowable value is the maximum of <code>unit32</code> data type. This parameter is tunable.
Exclude blobs touching image border	Select this check box to exclude a labeled blob that contains at least one border pixel.
Output blob statistics as a variable-size signal	Select this check box to output blob statistics as a variable-size signal. Selecting this check box means that you do not need to specify fill values.

Parameter	Value
Fill empty spaces in outputs outputs	<p>Select this check box to fill empty spaces in the statistical vectors with the values you specify in the Fill values parameter.</p> <p>The Fill empty spaces in outputs check box does not appear when you select the Output blob statistics as a variable-size signal check box.</p>
Fill values	<p>If you enter a scalar value, the block fills all the empty spaces in the statistical vectors with this value. If you enter a vector, it must have the same length as the number of selected statistics. The block uses each vector element to fill a different statistics vector. If the empty spaces do not affect your computation, you can deselect the Fill empty spaces in outputs check box. As a best practice, leave this check box selected.</p> <p>The Fill values parameter is not visible when you select the Output blob statistics as a variable-size signal check box.</p>

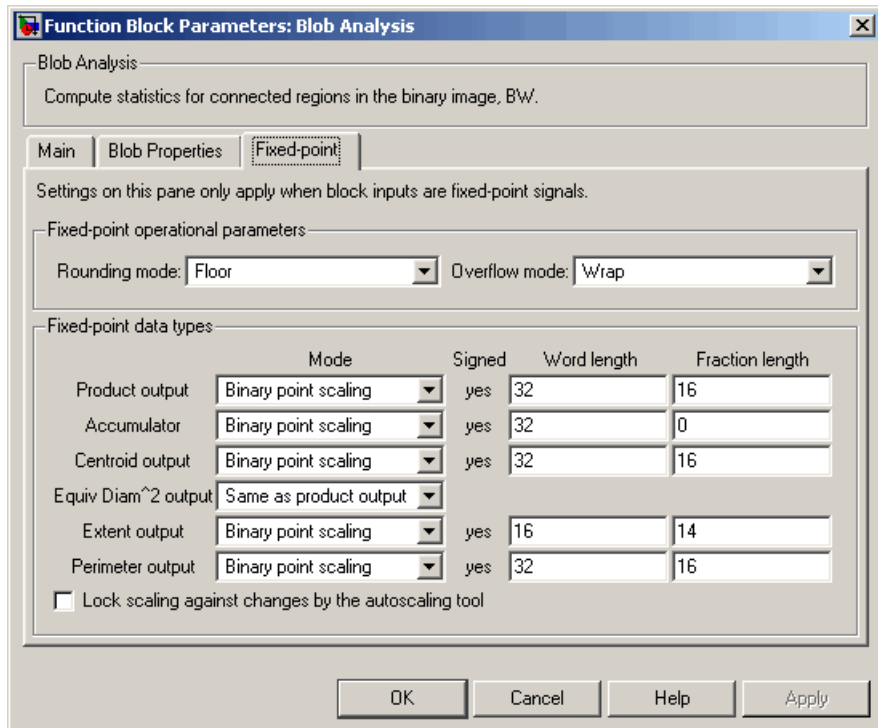
Fixed-Point Data Types

The following diagram shows the data types used in the Blob Analysis block for fixed-point signals.



The parameters on the **Fixed-point** tab apply only when you set the **Statistics output data type** parameter to Specify via Fixed-point tab.

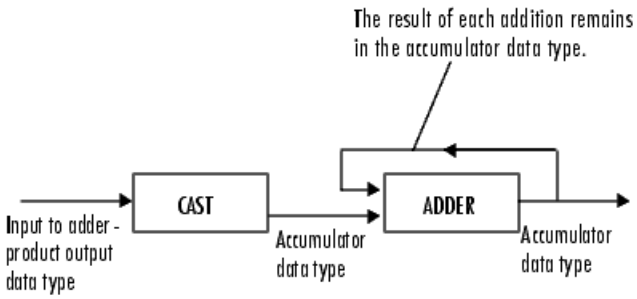
Blob Analysis



Parameter	Value
Rounding mode	Select the rounding mode Floor, Ceiling, Nearest or Zero for fixed-point operations.
Overflow mode	Select the overflow mode, Wrap or Saturate for fixed-point operations.

Parameter	Value
Product output	<p>When you select Binary point scaling, you can enter the Word length and the Fraction length of the product output, in bits.</p> <p>When you select Slope and bias scaling, you can enter the Word length in bits, and the Slope of the product output. All signals in the Video and Image Processing Blockset software have a bias of 0.</p> <div data-bbox="442 534 891 638" data-label="Diagram"> <pre> graph LR A[Accumulator data type] --> M[MULTIPLIER] B[4/pi data type] --> M M --> C[Product output data type] </pre> </div> <p>The block places the output of the multiplier into the Product output data type and scaling. The computation of the equivalent diameter squared uses the product output data type. During this computation, the block multiplies the blob area (stored in the accumulator) by the $4/\pi$ factor. This factor has a word length that equals the value of Equivalent diameter squared output data type Word length. The value of the Fraction length equals its word length minus two. Use this parameter to specify how to designate this product output word and fraction lengths.</p>
Accumulator	<p>When you select Same as product output the characteristics match the characteristics of the product output.</p> <p>When you select Binary point scaling, you can enter the Word length and the Fraction length of the accumulator, in bits.</p> <p>When you select Slope and bias scaling, you can enter the Word length, in bits, and the Slope of the Accumulator. All signals in the Video and Image Processing Blockset software have a bias of 0.</p>

Blob Analysis

Parameter	Value
	 <p data-bbox="397 760 1276 885">Inputs to the Accumulator get cast to the accumulator data type. Each element of the input gets added to the output of the adder, which remains in the accumulator data type. Use this parameter to specify how to designate this accumulator word and fraction lengths.</p>
Centroid output	<p data-bbox="397 904 1253 963">Choose how to specify the Word length and Fraction length of the output at the Centroid port:</p> <ul data-bbox="402 998 1276 1258" style="list-style-type: none"> <li data-bbox="402 998 1276 1067">• When you select Same as accumulator, these characteristics match the characteristics of the accumulator. <li data-bbox="402 1076 1276 1145">• When you select Binary point scaling, you can enter the Word length and Fraction length of the output, in bits. <li data-bbox="402 1154 1276 1258">• When you select Slope and bias scaling, you can enter the Word length, in bits, and the Slope of the output. All signals in the Video and Image Processing Blockset software have a bias of 0.

Parameter	Value
Equiv Diam² output	<p>Choose how to specify the Word length and Fraction length of the output at the Diameter² port:</p> <ul style="list-style-type: none"> • When you select Same as accumulator, these characteristics match the characteristics of the Accumulator. • When you select Same as product output, these characteristics match the characteristics of the Product output. • When you select Binary point scaling, you can enter the Word length and Fraction length of the output, in bits. • When you select Slope and bias scaling, you can enter the Word length, in bits, and the Slope of the output. All signals in the Video and Image Processing Blockset software have a bias of 0.
Extent output	<p>Choose how to specify the Word length and Fraction length of the output at the Extent port:</p> <ul style="list-style-type: none"> • When you select Same as accumulator, these characteristics match the characteristics of the accumulator. • When you select Binary point scaling, you can enter the Word length and Fraction length of the output, in bits. • When you select Slope and bias scaling, you can enter the Word length, in bits, and the Slope of the output. All signals in the Video and Image Processing Blockset software have a bias of 0.
Fill in empty spaces outputs	<p>Select this check box to fill empty spaces in the statistical vectors with the value you specify in the Fill values parameter.</p> <p>The Fill empty spaces in outputs check box is not visible when you select the Output blob statistics as a variable-size signal check box.</p>

Blob Analysis

Parameter	Value
Perimeter output	<p>Choose how to specify the Word length and Fraction length of the output at the Perimeter port:</p> <ul style="list-style-type: none">• When you select Same as accumulator, these characteristics match the characteristics of the accumulator.•• When you select Binary point scaling, you can enter the Word length and Fraction length of the output, in bits.• When you select Slope and bias scaling, you can enter the word length, in bits, and the Slope of the output. All signals in the Video and Image Processing Blockset software have a bias of 0.
Lock scaling against changes by the autoscaling tool	<p>Select this parameter to prevent the autoscaling tool in the Fixed-Point Tool overriding any fixed-point scaling you specify in this block mask. For more information, see <code>fxptdlg</code>, a reference page on the Fixed-Point Tool in the Simulink documentation.</p>

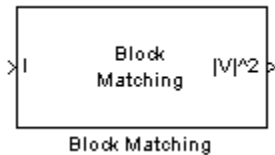
See Also

Label	Video and Image Processing Blockset
Variable Selector	Signal Processing Blockset
<code>regionprops</code>	Image Processing Toolbox

Purpose Estimate motion between images or video frames

Library Analysis & Enhancement
vipanalysis

Description



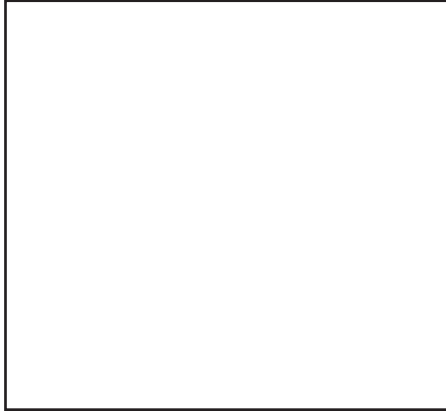
The Block Matching block estimates motion between two images or two video frames using “blocks” of pixels. The Block Matching block matches the block of pixels in frame k to a block of pixels in frame $k+1$ by moving the block of pixels over a search region.

Suppose the input to the block is frame k . The Block Matching block performs the following steps:

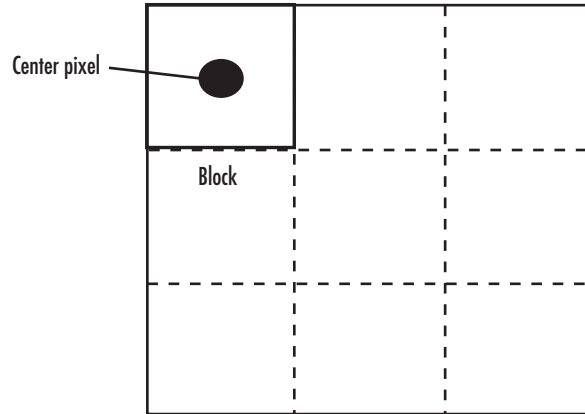
- 1 The block subdivides this frame using the values you enter for the **Block size [height width]** and **Overlap [r c]** parameters. In the following example, the **Overlap [r c]** parameter is [0 0].
- 2 For each subdivision or block in frame $k+1$, the Block Matching block establishes a search region based on the value you enter for the **Maximum displacement [r c]** parameter.
- 3 The block searches for the new block location using either the Exhaustive or Three-step search method.

Block Matching

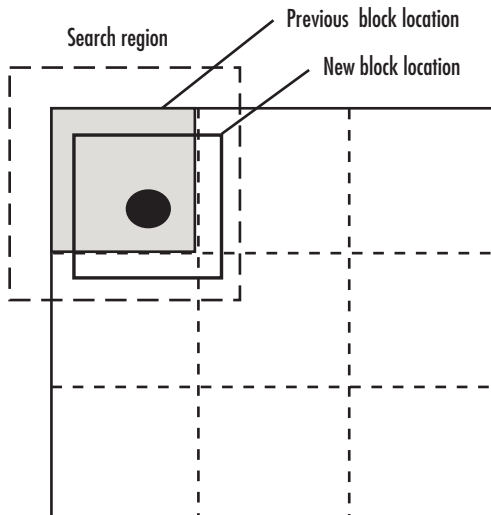
Input image = frame k



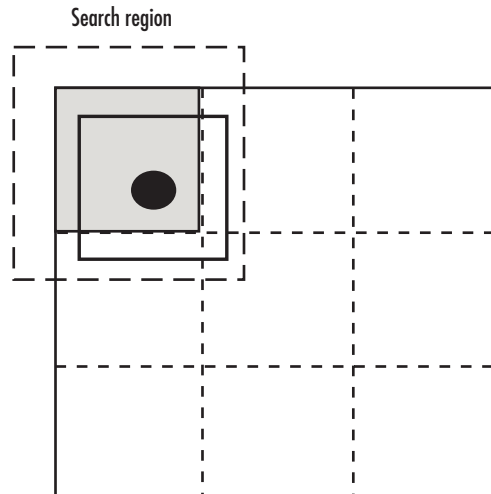
STEP 1: Subdivide the image in frame k.



STEP 2: Establish the search region in frame k+1.



STEP 3: Search for the new block location in frame k+1.



Port	Output	Supported Data Types	Complex Values Supported
I/I1	Scalar, vector, or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
I2	Scalar, vector, or matrix of intensity values	Same as I port	No
$ V ^2$	Matrix of velocity magnitudes	Same as I port	No
V	Matrix of velocity components in complex form	Same as I port	Yes

Use the **Estimate motion between** parameter to specify whether to estimate the motion between two images or two video frames. If you select **Current frame** and **N-th frame back**, the **N** parameter appears in the dialog box. Enter a scalar value that represents the number of frames between the reference frame and the current frame.

Use the **Search method** parameter to specify how the block locates the block of pixels in frame $k+1$ that best matches the block of pixels in frame k .

- If you select **Exhaustive**, the block selects the location of the block of pixels in frame $k+1$ by moving the block over the search region 1 pixel at a time. This process is computationally expensive.
- If you select **Three-step**, the block searches for the block of pixels in frame $k+1$ that best matches the block of pixels in frame k using a steadily decreasing step size. The block begins with a step size

Block Matching

approximately equal to half the maximum search range. In each step, the block compares the central point of the search region to eight search points located on the boundaries of the region and moves the central point to the search point whose values is the closest to that of the central point. The block then reduces the step size by half, and begins the process again. This option is less computationally expensive, though it might not find the optimal solution.

Use the **Block matching criteria** parameter to specify how the block measures the similarity of the block of pixels in frame k to the block of pixels in frame $k+1$. If you select **Mean square error (MSE)**, the Block Matching block estimates the displacement of the center pixel of the block as the (d_1, d_2) values that minimize the following MSE equation:

$$MSE(d_1, d_2) = \frac{1}{N_1 \times N_2} \sum_{(n_1, n_2) \in B} [s(n_1, n_2, k) - s(n_1 + d_1, n_2 + d_2, k + 1)]^2$$

In the previous equation, B is an $N_1 \times N_2$ block of pixels, and $s(x, y, k)$ denotes a pixel location at (x, y) in frame k . If you select **Mean absolute difference (MAD)**, the Block Matching block estimates the displacement of the center pixel of the block as the (d_1, d_2) values that minimize the following MAD equation:

$$MAD(d_1, d_2) = \frac{1}{N_1 \times N_2} \sum_{(n_1, n_2) \in B} |s(n_1, n_2, k) - s(n_1 + d_1, n_2 + d_2, k + 1)|$$

Use the **Block size [height width]** and **Overlap [r c]** parameters to specify how the block subdivides the input image. For a graphical description of these parameters, see the first figure in this reference page. If the **Overlap [r c]** parameter is not $[0 \ 0]$, the blocks would overlap each other by the number of pixels you specify.

Use the **Maximum displacement [r c]** parameter to specify the maximum number of pixels any center pixel in a block of pixels might

move from image to image or frame to frame. The block uses this value to determine the size of the search region.

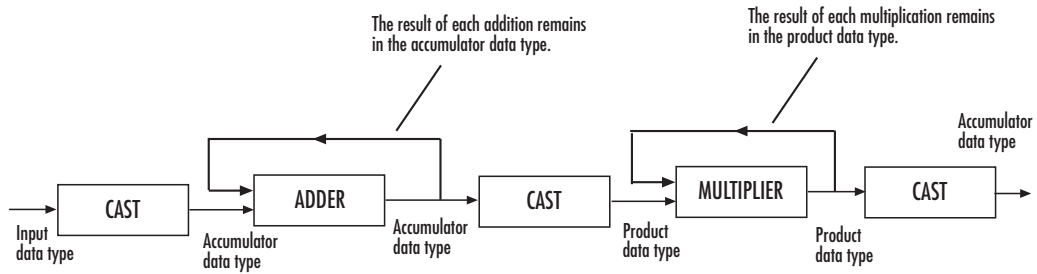
Use the **Velocity output** parameter to specify the block's output. If you select **Magnitude-squared**, the block outputs the optical flow matrix where each element is of the form u^2+v^2 . If you select **Horizontal and vertical components in complex form**, the block outputs the optical flow matrix where each element is of the form $u + jv$. The real part of each value is the horizontal velocity component and the imaginary part of each value is the vertical velocity component.

Fixed-Point Data Types

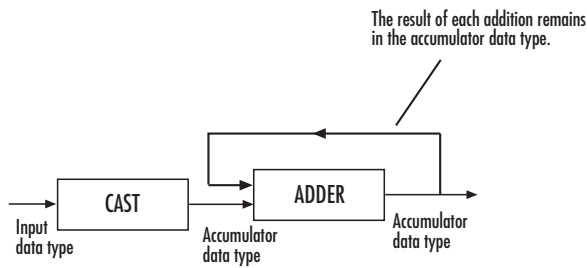
The following diagram shows the data types used in the Block Matching block for fixed-point signals.

Block Matching

MSE Block Matching



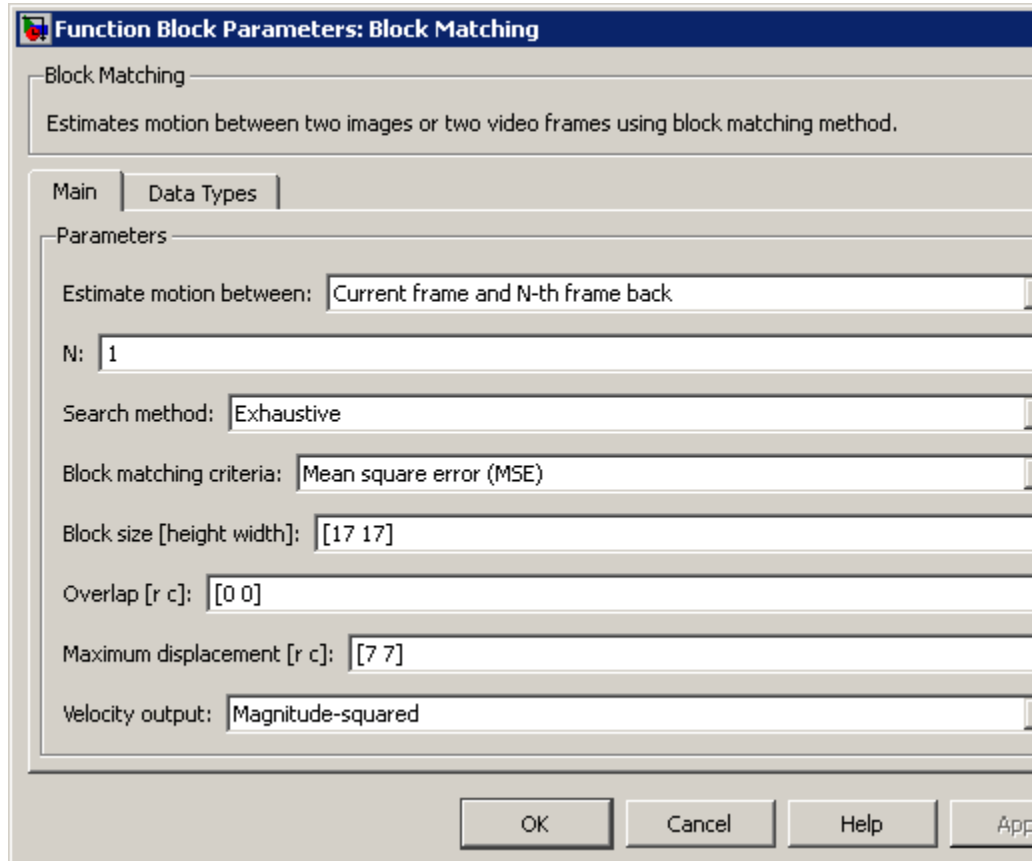
MAD Block Matching



You can set the accumulator and output data types in the block mask as discussed in the next section.

Dialog Box

The **Main** pane of the Block Matching dialog box appears as shown in the following figure.



Estimate motion between

Select **Two images** to estimate the motion between two images.
Select **Current frame and N-th frame back** to estimate the motion between two video frames that are N frames apart.

Block Matching

N

Enter a scalar value that represents the number of frames between the reference frame and the current frame. This parameter is only visible if, for the **Estimate motion between** parameter, you select **Current frame and N-th frame back**.

Search method

Specify how the block searches for the block of pixels in the next image or frame. Your choices are **Exhaustive** or **Three-step**.

Block matching criteria

Specify how the block measures the similarity of the block of pixels in frame k to the block of pixels in frame $k+1$. Your choices are **Mean square error (MSE)** or **Mean absolute difference (MAD)**.

Block size [height width]

Specify the size of the block of pixels.

Overlap [r c]

Specify the overlap (in pixels) of two subdivisions of the input image.

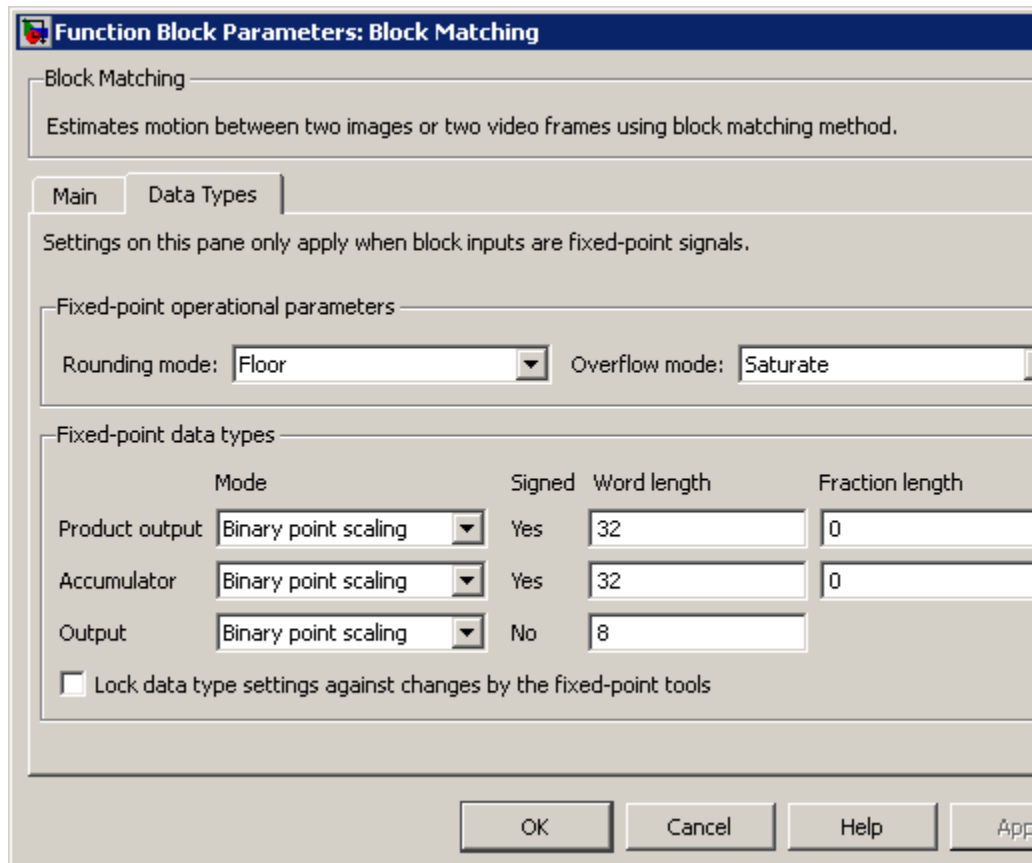
Maximum displacement [r c]

Specify the maximum number of pixels any center pixel in a block of pixels might move from image to image or frame to frame. The block uses this value to determine the size of the search region.

Velocity output

If you select **Magnitude-squared**, the block outputs the optical flow matrix where each element is of the form $u^2 + v^2$. If you select **Horizontal and vertical components in complex form**, the block outputs the optical flow matrix where each element is of the form $u + jv$.

The **Data Types** pane of the Block Matching dialog box appears as shown in the following figure.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Block Matching

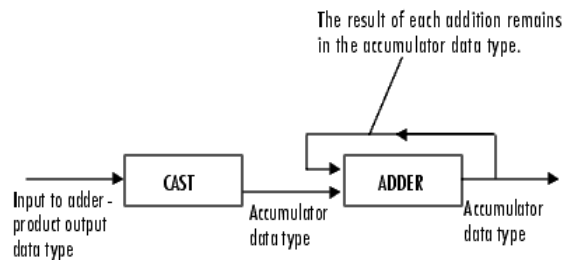
Product output



As shown previously, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate the product output word and fraction lengths.

- When you select **Same as input**, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset software is 0.

Accumulator



As depicted previously, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the

accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset software is 0.

Output

Choose how to specify the word length and fraction length of the output of the block:

- When you select **Binary point scaling**, you can enter the word length of the output, in bits. The fractional length is always 0.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, of the output. The bias of all signals in the Video and Image Processing Blockset software is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

See Also

Optical Flow

Video and Image Processing Blockset software

Block Processing

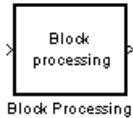
Purpose

Repeat user-specified operation on submatrices of input matrix

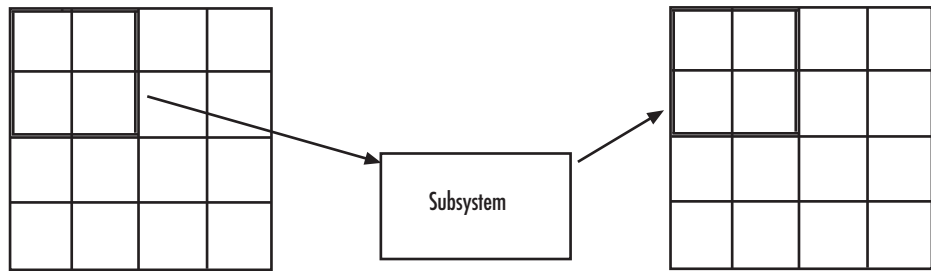
Library

Utilities

Description



The Block Processing block extracts submatrices of a user-specified size from each input matrix. It sends each submatrix to a subsystem for processing, and then reassembles each subsystem output into the output matrix.



Note Because you modify the Block Processing block's subsystem, the link between this block and the block library is broken when you click-and-drag a Block Processing block into your model. As a result, this block will not be automatically updated if you upgrade to a newer version of the Video and Image Processing Blockset software. If you right-click on the block and select **Look under mask**, you can delete blocks from this subsystem without triggering a warning. Lastly, if you search for library blocks in a model, this block will not be part of the results.

The blocks inside the subsystem dictate the frame status of the input and output signals, whether single channel or multichannel signals are supported, and which data types are supported by this block.

Use the **Number of inputs** and **Number of outputs** parameters to specify the number of input and output ports on the Block Processing block.

Use the **Block size** parameter to specify the size of each submatrix in cell array format. Each vector in the cell array corresponds to one input; the block uses the vectors in the order you enter them. If you have one input port, enter one vector. If you have more than one input port, you can enter one vector that is used for all inputs or you can specify a different vector for each input. For example, if you want each submatrix to be 2-by-3, enter `{[2 3]}`.

Use the **Overlap** parameter to specify the overlap of each submatrix in cell array format. Each vector in the cell array corresponds to the overlap of one input; the block uses the vectors in the order they are specified. If you enter one vector, each overlap is the same size. For example, if you want each 3-by-3 submatrix to overlap by 1 row and 2 columns, enter `{[1 2]}`.

The **Traverse order** parameter determines how the block extracts submatrices from the input matrix. If you select **Row-wise**, the block extracts submatrices by moving across the rows. If you select **Column-wise**, the block extracts submatrices by moving down the columns.

Click the **Open Subsystem** button to open the block's subsystem. Click-and-drag blocks into this subsystem to define the processing operation(s) the block performs on the submatrices. The input to this subsystem are the submatrices whose size is determined by the **Block size** parameter.

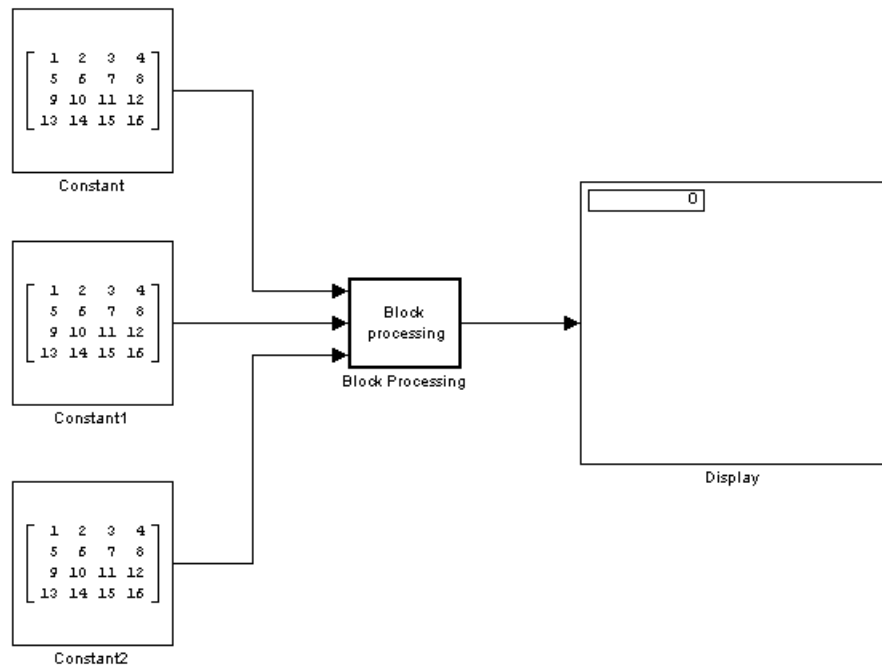
Note When you place an Assignment block inside a Block Processing block's subsystem, the Assignment block behaves as though it is inside a For Iterator block. For a description of this behavior, see the "Iterated Assignment" section of the Assignment block reference page. To achieve the normal behavior of the Assignment block, use an Overwrite Values block inside the Block Processing block's subsystem.

Block Processing

Example

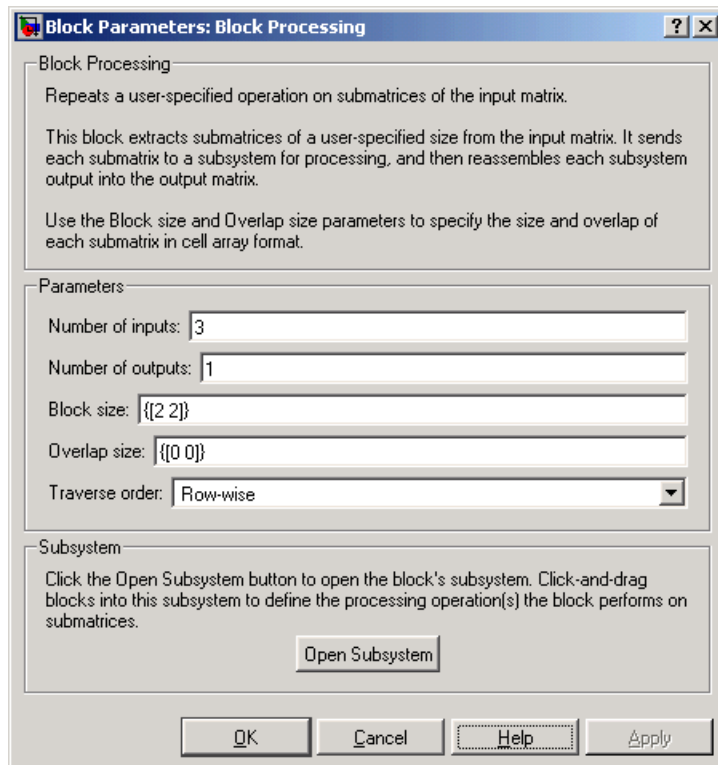
Example 1 -- Multiple Inputs

In this example, you multiply each element of three input matrices by two and add the results using the Block Processing block. Suppose you have the following model:



1 Use the Block Processing block to perform the multiplication and addition on submatrices of the three input matrices. Set the block parameters as shown in the following figure.

- **Number of inputs** = 3
- **Number of outputs** = 1
- **Block size** = $\{[2 \ 2]\}$

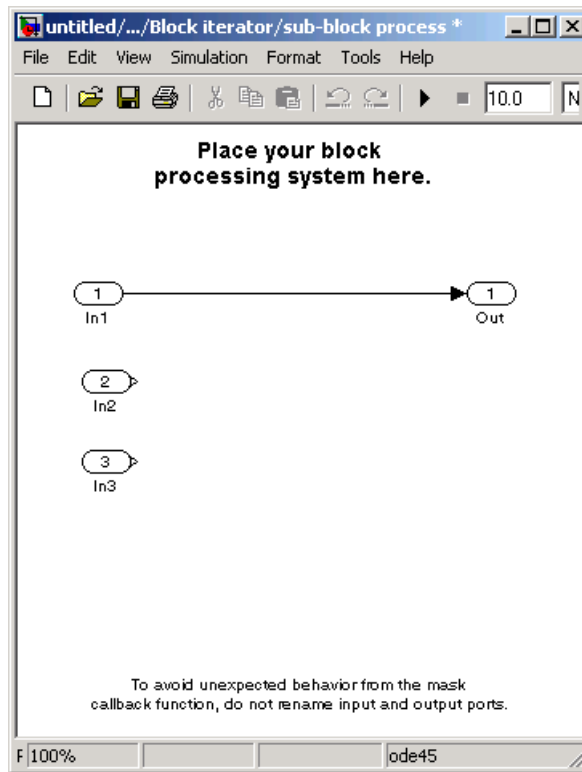


For each iteration, the block sends a 2-by-2 submatrix from each input matrix to the Block Processing blocks' subsystem to be processed. The block calculates its total number of iterations using the dimensions of the matrix connected to the top input port. In this case, the first input is a 4-by-4 matrix. Since the block can extract four 2-by-2 submatrices from this input matrix, the block iterates four times.

2 Click **Open Subsystem**.

The block's subsystem opens.

Block Processing

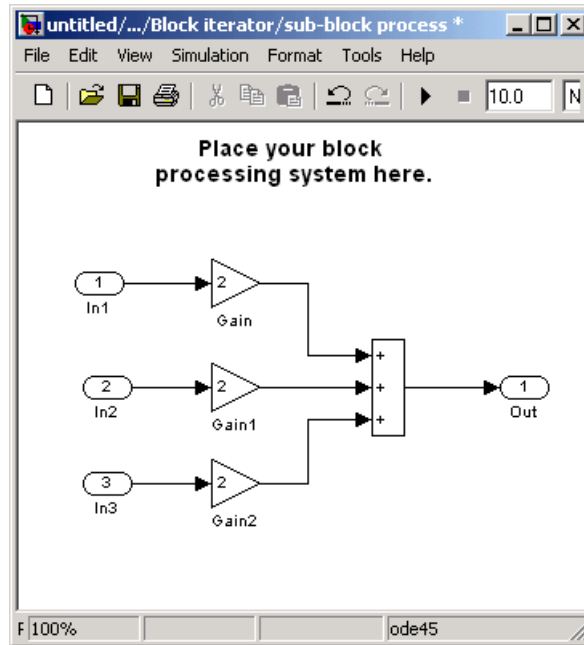


- 3 Click and drag the following blocks into the subsystem:

Block	Library	Quantity
Gain	Simulink / Math Operations	3
Sum	Simulink / Math Operations	1

- 4 Use the Gain blocks to multiply the elements of each submatrix by two. Set the **Gain** parameter to 2.
- 5 Use the Sum block to add the values. Set the **Icon shape** parameter to rectangular and the **List of signs** parameter to +++.

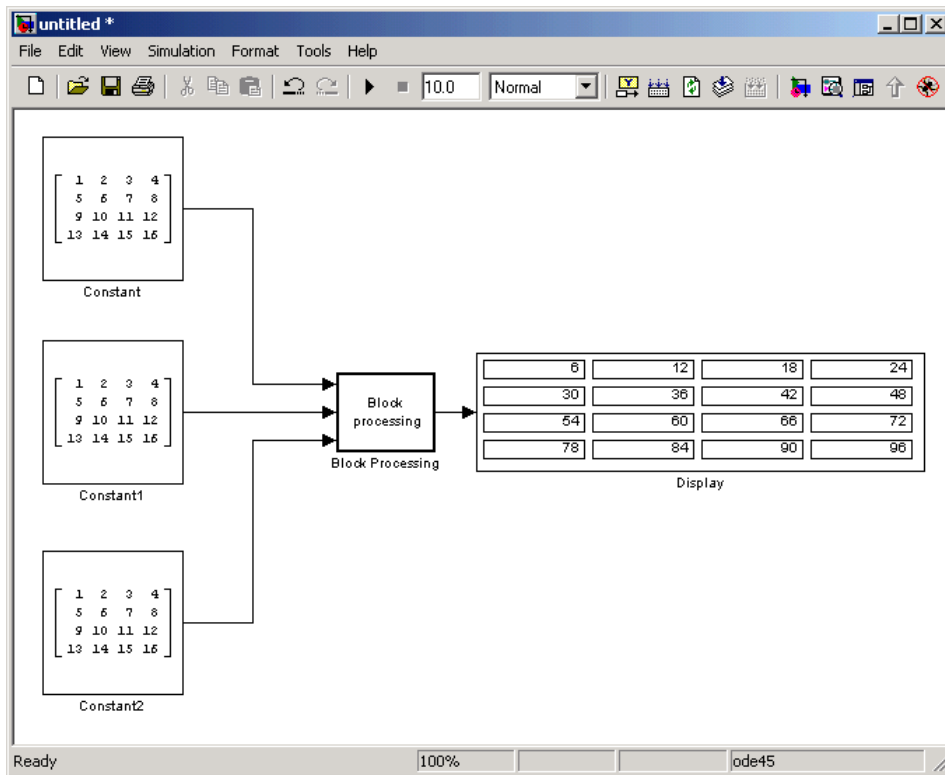
6 Connect the blocks as shown in the following figure.



7 Close the subsystem and Click **OK**.

8 Run the model.

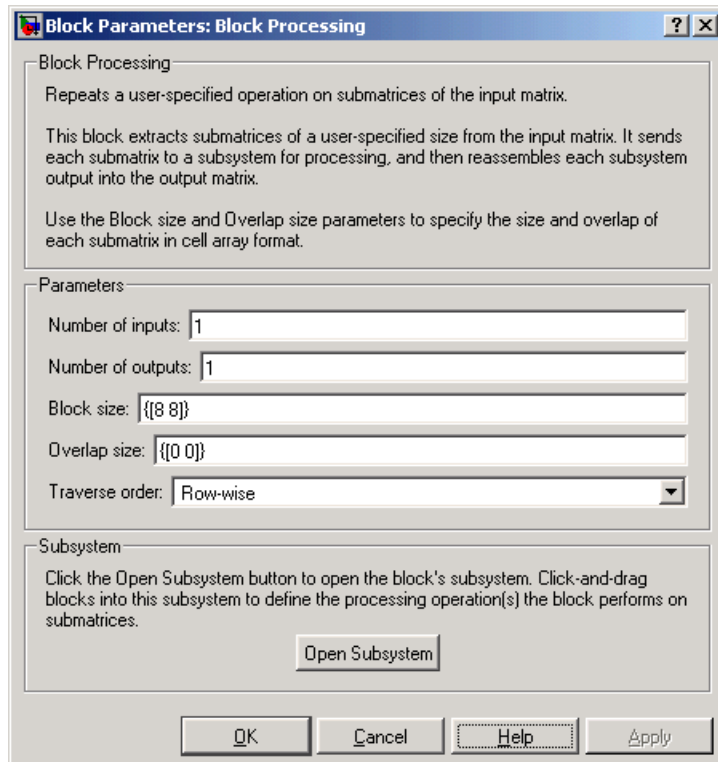
Block Processing



The Block Processing block operates on the submatrices and assembles the results into an output matrix that is displayed using the Display block.

Dialog Box

The Block Processing dialog box appears as shown in the following figure.



Number of inputs

Enter the number of input ports on the Block Processing block.

Number of outputs

Enter the number of output ports on the Block Processing block.

Block size

Specify the size of each submatrix in cell array format. Each vector in the cell array corresponds to one input.

Block Processing

Overlap

Specify the overlap of each submatrix in cell array format. Each vector in the cell array corresponds to the overlap of one input.

Traverse order

Determines how the block extracts submatrices from the input matrix. If you select **Row-wise**, the block extracts submatrices by moving across the rows. If you select **Column-wise**, the block extracts submatrices by moving down the columns.

Open Subsystem

Click this button to open the block's subsystem. Click-and-drag blocks into this subsystem to define the processing operation(s) the block performs on the submatrices.

See Also

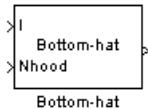
For Iterator
blockproc

Simulink
Image Processing Toolbox

Purpose Perform bottom-hat filtering on intensity or binary images

Library Morphological Operations

Description Use the Bottom-hat block to perform bottom-hat filtering on an intensity or binary image using a predefined neighborhood or structuring element. Bottom-hat filtering is the equivalent of subtracting the input image from the result of performing a morphological closing operation on the input image. This block uses flat structuring elements only.



Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
Nhood	Matrix or vector of ones and zeros that represents the neighborhood values	Boolean	No
Output	Scalar, vector, or matrix that represents the filtered image	Same as I port	No

If your input image is a binary image, for the **Input image type** parameter, select **Binary**. If your input image is an intensity image, select **Intensity**.

Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select **Specify via dialog**, the **Neighborhood or structuring**

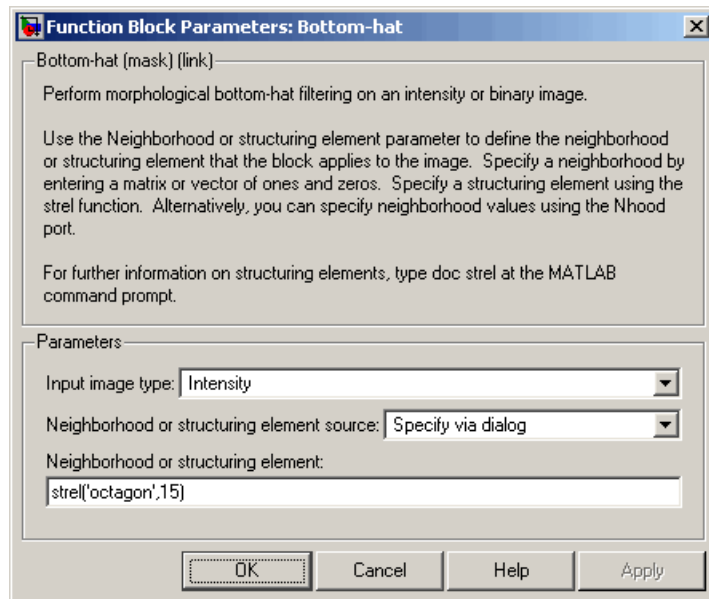
Bottom-hat

element parameter appears in the dialog box. If you select Input port, the Nhood port appears on the block. Use this port to enter your neighborhood values as a matrix or vector of 1s and 0s. You can only specify a structuring element using the dialog box.

Use the **Neighborhood or structuring element** parameter to define the region the block moves throughout the image. Specify a neighborhood by entering a matrix or vector of 1s and 0s. Specify a structuring element with the `strel` function from the Image Processing Toolbox. If the structuring element is decomposable into smaller elements, the block executes at higher speeds due to the use of a more efficient algorithm.

Dialog Box

The Bottom-hat dialog box appears as shown in the following figure.



Input image type

If your input image is a binary image, select `Binary`. If your input image is an intensity image, select `Intensity`.

Neighborhood or structuring element source

Specify how to enter your neighborhood or structuring element values. Select `Specify via dialog` to enter the values in the dialog box. Select `Input port` to use the `Nhood` port to specify the neighborhood values. You can only specify a structuring element using the dialog box.

Neighborhood or structuring element

If you are specifying a neighborhood, this parameter must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the `strel` function from the Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or structuring element source** parameter, you select `Specify via dialog`.

See Also

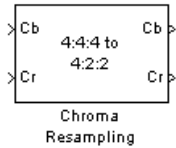
<code>Closing</code>	Video and Image Processing Blockset software
<code>Dilation</code>	Video and Image Processing Blockset software
<code>Erosion</code>	Video and Image Processing Blockset software
<code>Label</code>	Video and Image Processing Blockset software
<code>Opening</code>	Video and Image Processing Blockset software
<code>Top-hat</code>	Video and Image Processing Blockset software
<code>imbothat</code>	Image Processing Toolbox software
<code>strel</code>	Image Processing Toolbox software

Chroma Resampling

Purpose Downsample or upsample chrominance components of images

Library Conversions

Description The Chroma Resampling block downsamples or upsamples chrominance components of pixels to reduce the bandwidth required for transmission or storage of a signal.

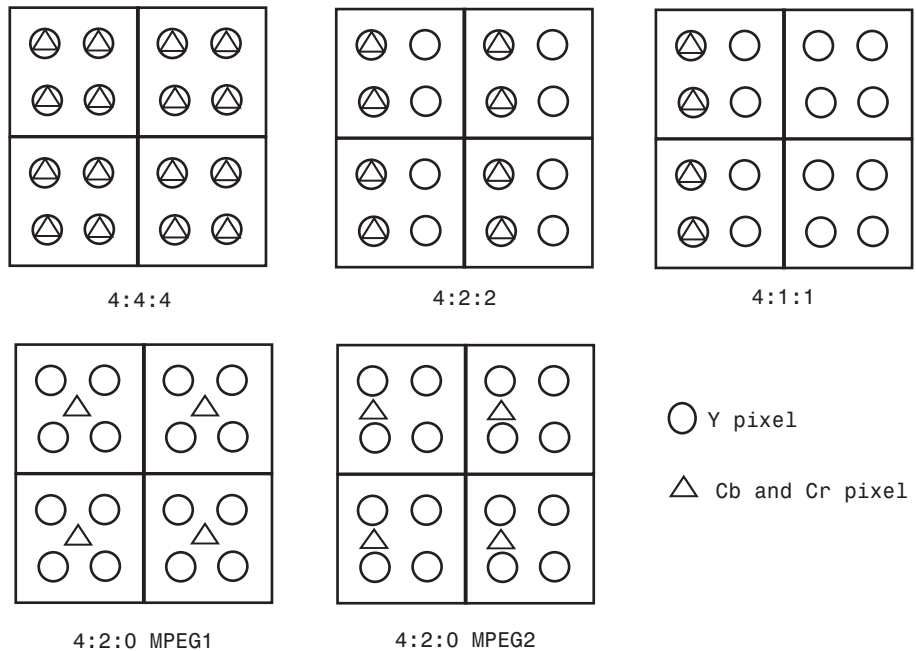


Port	Input/Output	Supported Data Types	Complex Values Supported
Cb	Matrix that represents one chrominance component of an image	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• 8-bit unsigned integer	No
Cr	Matrix that represents one chrominance component of an image	Same as Cb port	No

The data type of the output signals is the same as the data type of the input signals.

Chroma Resampling Formats

The Chroma Resampling block supports the formats shown in the following diagram.



Downsampling

If, for the **Resampling** parameter, you select 4:4:4 to 4:2:2, 4:4:4 to 4:2:0 (MPEG1), 4:4:4 to 4:2:0 (MPEG2), 4:4:4 to 4:1:1, 4:2:2 to 4:2:0 (MPEG1), or 4:2:2 to 4:2:0 (MPEG2), the block performs a downsampling operation. When the block downsamples from one format to another, it can bandlimit the input signal by applying a lowpass filter to prevent aliasing.

If, for the **Antialiasing filter** parameter, you select Default, the block uses a built-in lowpass filter to prevent aliasing.

If, for the **Resampling** parameter, you select 4:4:4 to 4:2:2, 4:4:4 to 4:2:0 (MPEG1), 4:4:4 to 4:2:0 (MPEG2), or 4:4:4 to 4:1:1 and, for the **Antialiasing filter** parameter, you select

Chroma Resampling

User-defined, the **Horizontal filter coefficients** parameter appears on the dialog box. Enter the filter coefficients to apply to your input.

If, for the **Resampling** parameter, you select 4:4:4 to 4:2:0 (MPEG1), 4:4:4 to 4:2:0 (MPEG2), 4:2:2 to 4:2:0 (MPEG1), or 4:2:2 to 4:2:0 (MPEG2) and, for the **Antialiasing filter** parameter, you select User-defined. **Vertical filter coefficients** parameters appear on the dialog box. Enter an even number of filter coefficients to apply to your input signal.

If, for the **Antialiasing filter** parameter, you select None, the block does not filter the input signal.

Upsampling

If, for the **Resampling** parameter, you select 4:2:2 to 4:4:4, 4:2:0 (MPEG1) to 4:2:2, 4:2:0 (MPEG1) to 4:4:4, 4:2:0 (MPEG2) to 4:2:2, 4:2:0 (MPEG2) to 4:4:4, or 4:1:1 to 4:4:4, the block performs an upsampling operation.

When the block upsamples from one format to another, it uses interpolation to approximate the missing chrominance values. If, for the **Interpolation** parameter, you select Linear, the block uses linear interpolation to calculate the missing values. If, for the **Interpolation** parameter, you select Pixel replication, the block replicates the chrominance values of the neighboring pixels to create the upsampled image.

Row-Major Data Format

The MATLAB environment and the Video and Image Processing Blockset software use column-major data organization. However, the Chroma Resampling block gives you the option to process data that is stored in row-major format. When you select the **Input image is transposed (data order is row major)** check box, the block assumes that the input buffer contains contiguous data elements from the first row first, then data elements from the second row second, and so on through the last row. Use this functionality only when you meet all the following criteria:

- You are developing algorithms to run on an embedded target that uses the row-major format.
- You want to limit the additional processing required to take the transpose of signals at the interfaces of the row-major and column-major systems.

When you use the row-major functionality, you must consider the following issues:

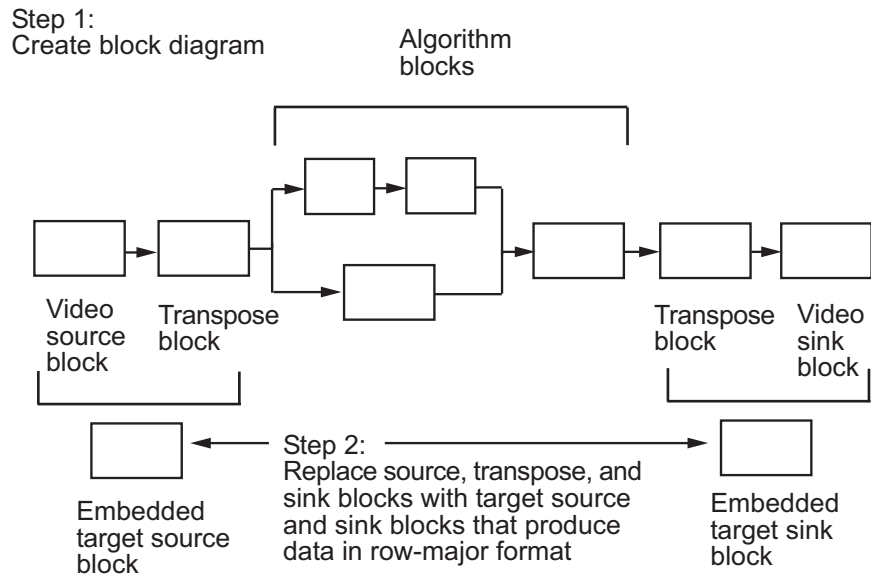
- When you select this check box, the signal dimensions of the Chroma Resampling block's input are swapped.
- All the Video and Image Processing Blockset blocks can be used to process data that is in the row-major format, but you need to know the image dimensions when you develop your algorithms.

For example, if you use the 2-D FIR Filter block, you need to verify that your filter coefficients are transposed. If you are using the Rotate block, you need to use negative rotation angles, etc.

- Only three blocks have the **Input image is transposed (data order is row major)** check box. They are the Chroma Resampling, Deinterlacing, and Insert Text blocks. You need to select this check box to enable row-major functionality in these blocks. All other blocks must be properly configured to process data in row-major format.

Use the following two-step workflow to develop algorithms in row-major format to run on an embedded target.

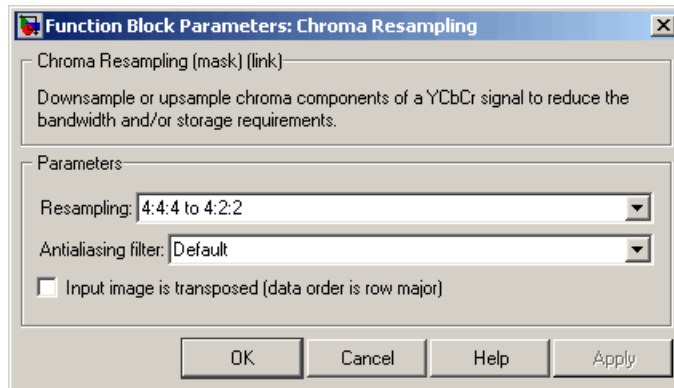
Chroma Resampling



See the DM642 EVM Video ADC and DM642 EVM Video DAC reference pages in the *Target Support Package™ User's Guide* for more information about data order in embedded targets.

Dialog Box

The Chroma Resampling dialog box appears as shown in the following figure.



Resampling

Specify the resampling format.

Antialiasing filter

Specify the lowpass filter that the block uses to prevent aliasing. If you select **Default**, the block uses a built-in lowpass filter. If you select **User-defined**, the **Horizontal filter coefficients** and/or **Vertical filter coefficients** parameters appear on the dialog box. If you select **None**, the block does not filter the input signal. This parameter is visible when you are downsampling the chrominance values.

Horizontal filter coefficients

Enter the filter coefficients to apply to your input signal. This parameter is visible if, for the **Resampling** parameter, you select **4:4:4 to 4:2:2**, **4:4:4 to 4:2:0 (MPEG1)**, **4:4:4 to 4:2:0 (MPEG2)**, or **4:4:4 to 4:1:1** and, for the **Antialiasing filter** parameter, you select **User-defined**.

Vertical filter coefficients

Enter the filter coefficients to apply to your input signal. This parameter is visible if, for the **Resampling** parameter, you

Chroma Resampling

select 4:4:4 to 4:2:0 (MPEG1), 4:4:4 to 4:2:0 (MPEG2), 4:2:2 to 4:2:0 (MPEG1), or 4:2:2 to 4:2:0 (MPEG2) and, for the **Antialiasing filter** parameter, you select User-defined.

Interpolation

Specify the interpolation method that the block uses to approximate the missing chrominance values. If you select **Linear**, the block uses linear interpolation to calculate the missing values. If you select **Pixel replication**, the block replicates the chrominance values of the neighboring pixels to create the upsampled image. This parameter is visible when you are upsampling the chrominance values. This parameter is visible if the **Resampling** parameter is set to 4:2:2 to 4:4:4 , 4:2:0 (MPEG1) to 4:4:4 , 4:2:0 (MPEG2) to 4:4:4 , 4:1:1 to 4:4:4 , 4:2:0 (MPEG1) to 4:2:2 , or 4:2:0 (MPEG2) to 4:2:2 .

Input image is transposed (data order is row major)

When you select this check box, the block assumes that the input buffer contains data elements from the first row first, then data elements from the second row second, and so on through the last row.

References

- [1] Haskell, Barry G., Atul Puri, and Arun N. Netravali. *Digital Video: An Introduction to MPEG-2*. New York: Chapman & Hall, 1996.
- [2] Recommendation ITU-R BT.601-5, Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide Screen 16:9 Aspect Ratios.
- [3] Wang, Yao, Jorn Ostermann, Ya-Qin Zhang. *Video Processing and Communications*. Upper Saddle River, NJ: Prentice Hall, 2002.

See Also

Autothreshold

Video and Image Processing Blockset software

Color Space Conversion

Video and Image Processing Blockset software

Image Complement

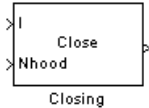
Video and Image Processing Blockset software

Closing

Purpose Perform morphological closing on binary or intensity images

Library Morphological Operations

Description The Closing block performs a dilation operation followed by an erosion operation using a predefined neighborhood or structuring element. This block uses flat structuring elements only.



Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• Boolean• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No
Nhood	Matrix or vector of ones and zeros that represents the neighborhood values	Boolean	No
Output	Vector or matrix of intensity values that represents the closed image	Same as I port	No

The output signal has the same data type as the input to the I port.

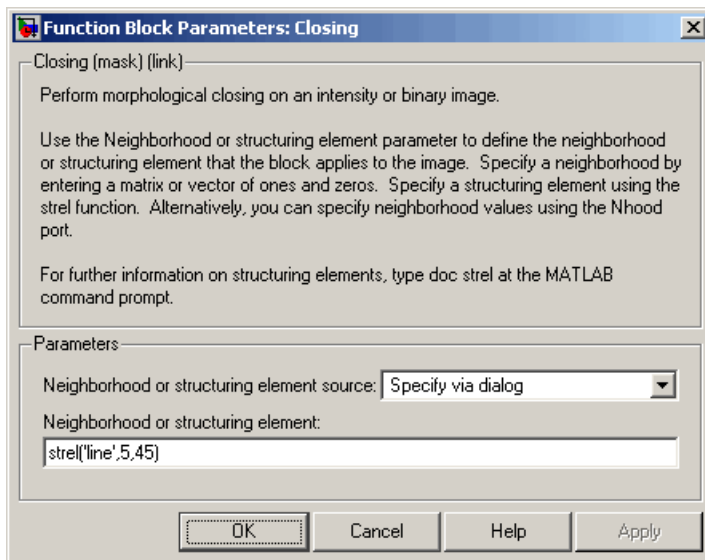
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values.

If you select `Specify via dialog`, the **Neighborhood or structuring element** parameter appears in the dialog box. If you select `Input port`, the `Nhood` port appears on the block. Use this port to enter your neighborhood values as a matrix or vector of 1s and 0s. You can only specify a structuring element using the dialog box.

Use the **Neighborhood or structuring element** parameter to define the region the block moves throughout the image. Specify a neighborhood by entering a matrix or vector of 1s and 0s. Specify a structuring element with the `strel` function from the Image Processing Toolbox. If the structuring element is decomposable into smaller elements, the block executes at higher speeds due to the use of a more efficient algorithm.

Dialog Box

The Closing dialog box appears as shown in the following figure.



Neighborhood or structuring element source

Specify how to enter your neighborhood or structuring element values. Select `Specify via dialog` to enter the values in the

dialog box. Select `Input port` to use the `Nhood` port to specify the neighborhood values. You can only specify a structuring element using the dialog box.

Neighborhood or structuring element

If you are specifying a neighborhood, this parameter must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the `strel` function from the Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or structuring element source** parameter, you select `Specify via dialog`.

References

[1] Soille, Pierre. *Morphological Image Analysis*. 2nd ed. New York: Springer, 2003.

See Also

Bottom-hat	Video and Image Processing Blockset software
Dilation	Video and Image Processing Blockset software
Erosion	Video and Image Processing Blockset software
Label	Video and Image Processing Blockset software
Opening	Video and Image Processing Blockset software
Top-hat	Video and Image Processing Blockset software
<code>imclose</code>	Image Processing Toolbox software
<code>strel</code>	Image Processing Toolbox software

Purpose Convert color information between color spaces

Library Conversions

Description



The Color Space Conversion block converts color information between color spaces. Use the **Conversion** parameter to specify the color spaces you are converting between. Your choices are R'G'B' to Y'CbCr, Y'CbCr to R'G'B', R'G'B' to intensity, R'G'B' to HSV, HSV to R'G'B', sR'G'B' to XYZ, XYZ to sR'G'B', sR'G'B' to L*a*b*, and L*a*b* to sR'G'B'.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input / Output	M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • 8-bit unsigned integer 	No
R'	Matrix that represents one plane of the input RGB video stream	Same as the Input port	No
G'	Matrix that represents one plane of the input RGB video stream	Same as the Input port	No
B'	Matrix that represents one plane of the input RGB video stream	Same as the Input port	No
Y'	Matrix that represents the luma portion of an image	Same as the Input port	No
Cb	Matrix that represents one chrominance component of an image	Same as the Input port	No

Color Space Conversion

Port	Input/Output	Supported Data Types	Complex Values Supported
Cr	Matrix that represents one chrominance component of an image	Same as the Input port	No
I'	Matrix of intensity values	Same as the Input port	No
H	Matrix that represents the hue component of an image	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point 	No
S	Matrix that represents represent the saturation component of an image	Same as the H port	No
V	Matrix that represents the value (brightness) component of an image	Same as the H port	No
X	Matrix that represents the X component of an image	Same as the H port	No
Y	Matrix that represents the Y component of an image	Same as the H port	No
Z	Matrix that represents the Z component of an image	Same as the H port	No
L*	Matrix that represents the luminance portion of an image	Same as the H port	No
a*	Matrix that represents the a* component of an image	Same as the H port	No
b*	Matrix that represents the b* component of an image	Same as the H port	No

The data type of the output signal is the same as the data type of the input signal.

Use the **Image signal** parameter to specify how to input and output a color video signal. If you select **One multidimensional signal**, the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

Note The prime notation indicates that the signals are gamma corrected.

Conversion Between R'G'B' and Y'CbCr Color Spaces

The R'G'B' to Y'CbCr conversion and the Y'CbCr to R'G'B' conversion are defined by the following equations:

$$\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + A \times \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = B \times \left(\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \right)$$

The values in the A and B matrices are based on your choices for the **Use conversion specified by** and **Scanning standard** parameters. The following table summarizes the possible values:

Color Space Conversion

Matrix	Use conversion specified by = Rec. 601 (SDTV)	Use conversion specified by = Rec. 709 (HDTV)	
		Scanning standard = 1125/60/2:1	Scanning standard = 1250/50/2:1
A	$\begin{bmatrix} 0.25678824 & 0.50412941 & 0.09790588 \\ -0.1482229 & -0.29099279 & 0.43921569 \\ 0.43921569 & -0.36778831 & -0.07142737 \end{bmatrix}$	$\begin{bmatrix} 0.18258588 & 0.61423059 & 0.06200706 \\ -0.10064373 & -0.33857195 & 0.43921569 \\ 0.43921569 & -0.39894216 & -0.04027352 \end{bmatrix}$	$\begin{bmatrix} 0.25678824 & 0.50412941 & 0.09790588 \\ -0.1482229 & -0.29099279 & 0.43921569 \\ 0.43921569 & -0.36778831 & -0.07142737 \end{bmatrix}$
B	$\begin{bmatrix} 1.1643836 & 0 & 1.5960268 \\ 1.1643836 & -0.39176229 & -0.81296765 \\ 0.16438356 & 2.0172321 & 0 \end{bmatrix}$	$\begin{bmatrix} 1.16438356 & 0 & 1.79274107 \\ 1.16438356 & -0.21324861 & -0.53290933 \\ 1.16438356 & 2.11240179 & 0 \end{bmatrix}$	$\begin{bmatrix} 1.1643836 & 0 & 1.5960268 \\ 1.1643836 & -0.39176229 & -0.81296765 \\ 0.16438356 & 2.0172321 & 0 \end{bmatrix}$

Conversion R'B'G' to Intensity

The conversion from the R'B'G' color space to intensity is defined by the following equation:

$$\text{intensity} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

Conversion Between R'G'B' and HSV Color Spaces

The R'G'B' to HSV conversion is defined by the following equations. In these equations, *MAX* and *MIN* represent the maximum and minimum values of each R'G'B' triplet, respectively. *H*, *S*, and *V* vary from 0 to 1, where 1 represents the greatest saturation and value.

$$H = \begin{cases} \left(\frac{G' - B'}{MAX - MIN} \right) / 6, & \text{if } R' = MAX \\ \left(2 + \frac{B' - R'}{MAX - MIN} \right) / 6, & \text{if } G' = MAX \\ \left(4 + \frac{R' - G'}{MAX - MIN} \right) / 6, & \text{if } B' = MAX \end{cases}$$

$$S = \frac{MAX - MIN}{MAX}$$

$$V = MAX$$

The HSV to R'G'B' conversion is defined by the following equations:

$$H_i = \lfloor 6H \rfloor$$

$$f = 6H - H_i$$

$$p = 1 - S$$

$$q = 1 - fS$$

$$t = 1 - (1 - f)S$$

$$\text{if } H_i = 0, \quad R_{tmp} = 1, \quad G_{tmp} = t, \quad B_{tmp} = p$$

$$\text{if } H_i = 1, \quad R_{tmp} = q, \quad G_{tmp} = 1, \quad B_{tmp} = p$$

$$\text{if } H_i = 2, \quad R_{tmp} = p, \quad G_{tmp} = 1, \quad B_{tmp} = t$$

$$\text{if } H_i = 3, \quad R_{tmp} = p, \quad G_{tmp} = q, \quad B_{tmp} = 1$$

$$\text{if } H_i = 4, \quad R_{tmp} = t, \quad G_{tmp} = p, \quad B_{tmp} = 1$$

$$\text{if } H_i = 5, \quad R_{tmp} = 1, \quad G_{tmp} = p, \quad B_{tmp} = q$$

$$u = V / \max(R_{tmp}, G_{tmp}, B_{tmp})$$

$$R' = uR_{tmp}$$

$$G' = uG_{tmp}$$

$$B' = uB_{tmp}$$

For more information about the HSV color space, see “HSV Color Space” in the Image Processing Toolbox documentation.

Color Space Conversion

Conversion Between sR'G'B' and XYZ Color Spaces

The sR'G'B' to XYZ conversion is a two-step process. First, the block converts the gamma-corrected sR'G'B' values to linear sRGB values using the following equations:

$$\begin{aligned} &\text{If } R'_{sRGB}, G'_{sRGB}, B'_{sRGB} \leq 0.03928 \\ &R_{sRGB} = R'_{sRGB} / 12.92 \\ &G_{sRGB} = G'_{sRGB} / 12.92 \\ &B_{sRGB} = B'_{sRGB} / 12.92 \\ &\text{otherwise, if } R'_{sRGB}, G'_{sRGB}, B'_{sRGB} > 0.03928 \\ &R_{sRGB} = \left[\frac{(R'_{sRGB} + 0.055)}{1.055} \right]^{2.4} \\ &G_{sRGB} = \left[\frac{(G'_{sRGB} + 0.055)}{1.055} \right]^{2.4} \\ &B_{sRGB} = \left[\frac{(B'_{sRGB} + 0.055)}{1.055} \right]^{2.4} \end{aligned}$$

Then the block converts the sRGB values to XYZ values using the following equation:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.41239079926596 & 0.35758433938388 & 0.18048078840183 \\ 0.21263900587151 & 0.71516867876776 & 0.07219231536073 \\ 0.01933081871559 & 0.11919477979463 & 0.95053215224966 \end{bmatrix} \times \begin{bmatrix} R_{sRGB} \\ G_{sRGB} \\ B_{sRGB} \end{bmatrix}$$

The XYZ to sR'G'B' conversion is also a two-step process. First, the block converts the XYZ values to linear sRGB values using the following equation:

$$\begin{bmatrix} R_{sRGB} \\ G_{sRGB} \\ B_{sRGB} \end{bmatrix} = \begin{bmatrix} 0.41239079926596 & 0.35758433938388 & 0.18048078840183 \\ 0.21263900587151 & 0.71516867876776 & 0.07219231536073 \\ 0.01933081871559 & 0.11919477979463 & 0.95053215224966 \end{bmatrix}^{-1} \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Then the block applies gamma correction to obtain the sR'G'B' values. This process is described by the following equations:

$$\text{If } R_{sRGB}, G_{sRGB}, B_{sRGB} \leq 0.00304$$

$$R'_{sRGB} = 12.92R_{sRGB}$$

$$G'_{sRGB} = 12.92G_{sRGB}$$

$$B'_{sRGB} = 12.92B_{sRGB}$$

$$\text{otherwise, if } R_{sRGB}, G_{sRGB}, B_{sRGB} > 0.00304$$

$$R'_{sRGB} = 1.055R_{sRGB}^{(1.0/2.4)} - 0.055$$

$$G'_{sRGB} = 1.055G_{sRGB}^{(1.0/2.4)} - 0.055$$

$$B'_{sRGB} = 1.055B_{sRGB}^{(1.0/2.4)} - 0.055$$

Note Video and Image Processing Blockset software uses a D65 white point, which is specified in Recommendation ITU-R BT.709, for this conversion. In contrast, the Image Processing Toolbox conversion is based on ICC profiles, and it uses a D65 to D50 Bradford adaptation transformation to the D50 white point. If you are using these two products and comparing results, you must account for this difference.

Conversion Between sR'G'B' and L*a*b* Color Spaces

The Color Space Conversion block converts sR'G'B' values to L*a*b* values in two steps. First it converts sR'G'B' to XYZ values using the equations described in “Conversion Between sR'G'B' and XYZ Color Spaces” on page 2-248. Then it uses the following equations to transform the XYZ values to L*a*b* values. Here, X_n , Y_n , and Z_n are the tristimulus values of the reference white point you specify using the **White point** parameter:

Color Space Conversion

$$L^* = 116(Y/Y_n)^{1/3} - 16, \text{ for } Y/Y_n > 0.008856$$

$$L^* = 903.3Y/Y_n, \quad \text{otherwise}$$

$$a^* = 500(f(X/X_n) - f(Y/Y_n))$$

$$b^* = 200(f(Y/Y_n) - f(Z/Z_n)),$$

$$\text{where } f(t) = t^{1/3}, \text{ for } t > 0.008856$$

$$f(t) = 7.787t + 16/166, \quad \text{otherwise}$$

The block converts L*a*b* values to sR'G'B' values in two steps as well. The block transforms the L*a*b* values to XYZ values using these equations:

$$\text{For } Y/Y_n > 0.008856$$

$$X = X_n(P + a^*/500)^3$$

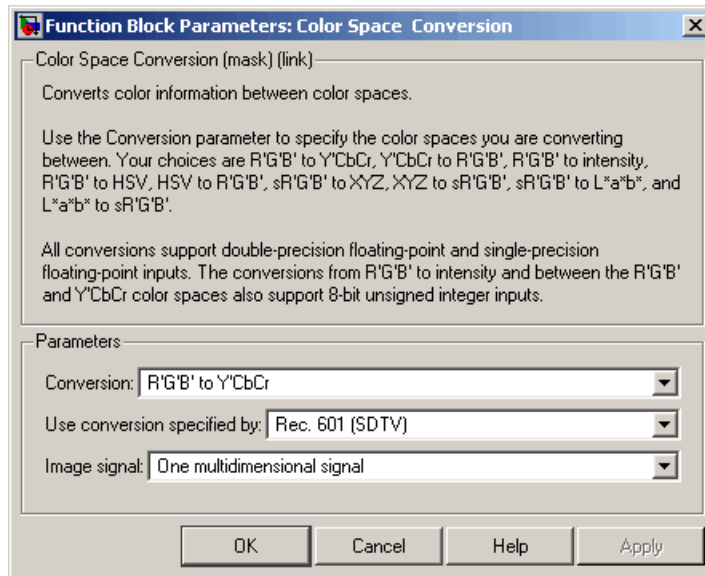
$$Y = Y_n P^3$$

$$Z = Z_n(P - b^*/200)^3,$$

$$\text{where } P = (L^* + 16)/116$$

Dialog Box

The Color Space Conversion dialog box appears as shown in the following figure.



Conversion

Specify the color spaces you are converting between. Your choices are R'G'B' to Y'CbCr, Y'CbCr to R'G'B', R'G'B' to intensity, R'G'B' to HSV, HSV to R'G'B', sR'G'B' to XYZ, XYZ to sR'G'B', sR'G'B' to L*a*b*, and L*a*b* to sR'G'B'.

Use conversion specified by

Specify the standard to use to convert your values between the R'G'B' and Y'CbCr color spaces. Your choices are Rec. 601 (SDTV) or Rec. 709 (HDTV). This parameter is only available if, for the **Conversion** parameter, you select R'G'B' to Y'CbCr or Y'CbCr to R'G'B'.

Scanning standard

Specify the scanning standard to use to convert your values between the R'G'B' and Y'CbCr color spaces. Your choices are

Color Space Conversion

1125/60/2:1 or 1250/50/2:1. This parameter is only available if, for the **Use conversion specified by** parameter, you select Rec. 709 (HDTV).

White point

Specify the reference white point. This parameter is visible if, for the **Conversion** parameter, you select sR'G'B' to L*a*b* or L*a*b* to sR'G'B'.

Image signal

Specify how to input and output a color video signal. If you select **One multidimensional signal**, the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

References

- [1] Poynton, Charles A. *A Technical Introduction to Digital Video*. New York: John Wiley & Sons, 1996.
- [2] Recommendation ITU-R BT.601-5, Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide Screen 16:9 Aspect Ratios.
- [3] Recommendation ITU-R BT.709-5. Parameter values for the HDTV standards for production and international programme exchange.
- [4] Stokes, Michael, Matthew Anderson, Srinivasan Chandrasekar, and Ricardo Motta, "A Standard Default Color Space for the Internet - sRGB." November 5, 1996.
- [5] Berns, Roy S. *Principles of Color Technology, 3rd ed.* New York: John Wiley & Sons, 2000.

See Also

Chroma Resampling	Video and Image Processing Blockset software
rgb2hsv	MATLAB software

hsv2rgb
rgb2ycbcr
ycbcr2rgb
rgb2gray
makecform
applycform

MATLAB software
Image Processing Toolbox software
Image Processing Toolbox software
Image Processing Toolbox software
Image Processing Toolbox software
Image Processing Toolbox software

Compositing

Purpose Combine pixel values of two images, overlay one image over another, or highlight selected pixels

Library Text & Graphics
viptextngfix

Description



You can use the Compositing block to combine two images, where each pixel of the output image is a linear combination of the pixels in each input image. This process is defined by the following equation:

$$O(i, j) = (1 - X) * I1(i, j) + X * I2(i, j)$$

The opacity factor, X , where $0 \leq X \leq 1$, defines the amount by which to scale each pixel value before combining them.

You can use the Compositing block to overlay a Image 2 over Image 1. The masking factor and the location determine which Image 1 pixels are overwritten. The masking factor(s) can be 0 or 1, where 0 corresponds to not overwriting pixels and 1 corresponds to overwriting pixels.

You can use the Compositing block to highlight selected pixels in the input image. Use a binary image, input at the Mask port, to specify which pixels to highlight.

Note This block supports intensity and color images on its ports.

Port	Input/Output	Supported Data Types	Complex Values Supported
Image 1	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
Image 2	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	Same as Image 1 port	No
Factor	Scalar or matrix of opacity or masking factor	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No

Compositing

Port	Input/Output	Supported Data Types	Complex Values Supported
Mask	Binary image that specifies which pixels to highlight	Same as Factor port When the Operation parameter is set to Highlight selected pixel , the input to the Mask port must be a Boolean data type.	No
Location	Two-element vector that specifies the position of the upper-left corner of the image input at port I2	<ul style="list-style-type: none">• Double-precision floating point. (Only supported if the input to the Image 1 and Image 2 ports is a floating-point data type.)• Single-precision floating point. (Only supported if the input to the Image 1 and Image 2 ports is a floating-point data type.)• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No
Output	Vector or matrix of intensity or color values	Same as Image 1 port	No

Use the **Operation** parameter to specify the operation you want the block to perform. If you choose **Blend**, the block linearly combines the pixels of one image with another image. If you choose **Binary mask**, the block overwrites the pixel values of one image with the pixel values of another image. If you choose **Highlight selected pixel**, the block uses the binary image input at the Mask port to determine which pixels are set to the maximum value supported by their data

type. For example, for every 1 in the binary image, the block sets the corresponding pixel in input image to the maximum value supported by its data type. For every 0 in the binary image, the block leaves the pixel value alone.

If, for the **Operation** parameter, you choose **Blend**, the **Opacity factor(s) source** parameter appears on the dialog box. Use this parameter to indicate where to specify the opacity factor(s).

- If you choose **Specify via dialog**, the **Opacity factor(s)** parameter appears on the dialog box. Use this parameter to define the amount by which the block scales each I2 pixel value before combining them with the Image 1 pixel values. You can enter a scalar value used for all pixels or a matrix of values that is the same size as Image 2.
- If you choose **Input port**, the **Factor** port appears on the block. The input to this port must be a scalar or matrix of values as described for the **Opacity factor(s)** parameter. If the input to the Image 1 and Image 2 ports is floating point, the input to this port must be the same floating-point data type.

If, for the **Operation** parameter, you choose **Binary mask**, the **Mask source** parameter appears on the dialog box. Use this parameter to indicate where to specify the masking factor(s).

- If you choose **Specify via dialog**, the **Mask** parameter appears on the dialog box. Use this parameter and the location of the I2 image to define which pixels are overwritten. You can enter 0 or 1, which is used for all pixels in I2, or a matrix of 0s and 1s that defines the factor for each I2 pixel.
- If you choose **Input port**, the **Factor** port appears on the block. The input to this port must be a 0 or 1 whose data type is Boolean or a matrix of 0s or 1s whose data type is Boolean as described for the **Mask** parameter.

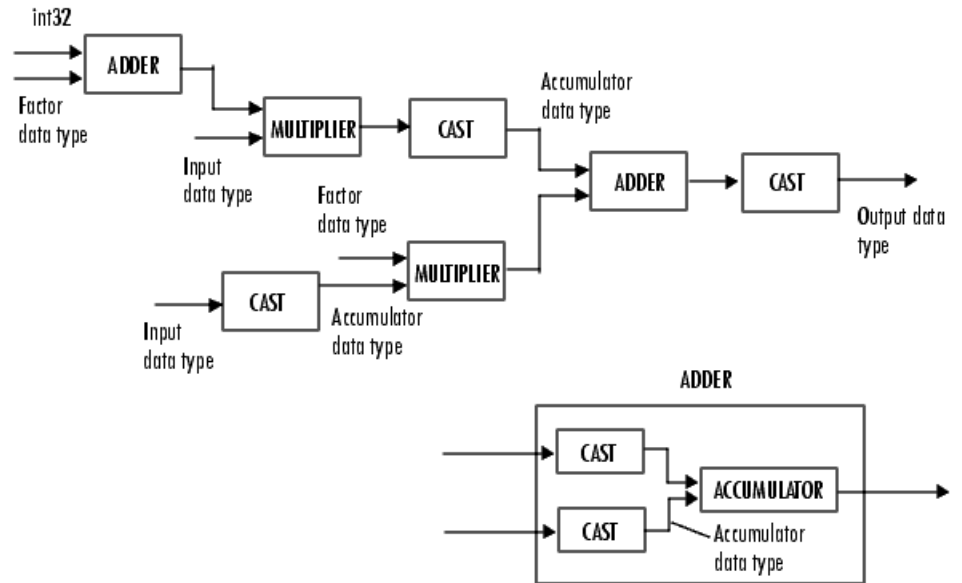
Use the **Location source** parameter to specify where to enter the zero-based location of the upper-left corner of the image input at port I2.

- If you choose **Specify** via dialog, the **Location [row column]** parameter appears on the dialog box. Enter a two-element vector that specifies the row and column position of the upper-left corner of the image input at port Image 2 relative to the upper-left corner of the image input at port Image 1. Positive values move the image down and to the right; negative values move the image up and to the left. If the first element is greater than the number of rows in the Image 1 matrix, the value is clipped to the total number of rows. If the second element is greater than the number of columns in the Image 1 matrix, the value is clipped to the total number of columns.
- If you choose **Input port**, the **Location port** appears on the block. The input to this port must be a two-element vector as described for the **Location [row column]** parameter.

If, for the **Operation** parameter, you choose **Highlight selected pixels**, the **Location source** parameter appears on the dialog box. This parameter is described above.

Fixed-Point Data Types

The following diagram shows the data types used in the Compositing block for fixed-point signals. It is only applicable when the **Operation** parameter is set to **Blend**.

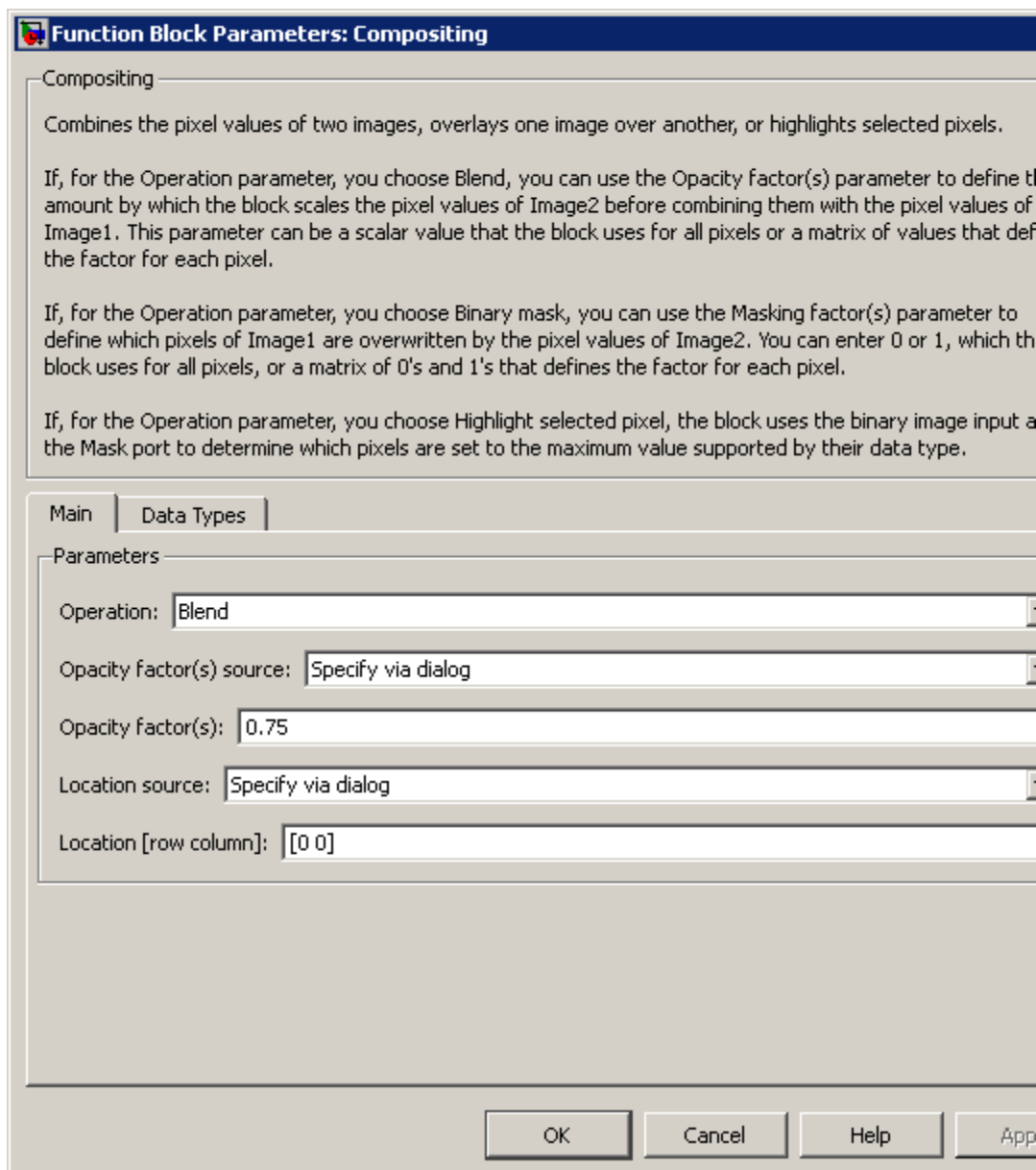


You can set the product output, accumulator, and output data types in the block mask as discussed in the next section.

Compositing

Dialog Box

The **Main** pane of the Compositing dialog box appears as shown in the following figure.



Operation

Specify the operation you want the block to perform. If you choose **Blend**, the block linearly combines the pixels of one image with another image. If you choose **Binary mask**, the block overwrites the pixel values of one image with the pixel values of another image. If you choose **Highlight selected pixel**, the block uses the binary image input at the **Mask** port to determine which pixels are set to the maximum value supported by their data type.

Opacity factor(s) source

Indicate where to specify the opacity factor(s). Your choices are **Specify via dialog** and **Input port**. This parameter is visible if, for the **Operation** parameter you choose **Blend**.

Opacity factor(s)

Define the amount by which the block scales each pixel value before combining them. You can enter a scalar value used for all pixels or a matrix of values that defines the factor for each pixel. This parameter is visible if, for the **Opacity factor(s) source** parameter you choose **Specify via dialog**. Tunable.

Mask source

Indicate where to specify the masking factor(s). Your choices are **Specify via dialog** and **Input port**. This parameter is visible if, for the **Operation** parameter you choose **Binary mask**.

Mask

Define which pixels are overwritten. You can enter 0 or 1, which is used for all pixels, or a matrix of 0s and 1s that defines the factor for each pixel. This parameter is visible if, for the **Mask source** parameter you choose **Specify via dialog**. Tunable.

Location source

Use this parameter to specify where to enter the location of the upper-left corner of the image input at port **I2**. Your choices are **Specify via dialog** and **Input port**.

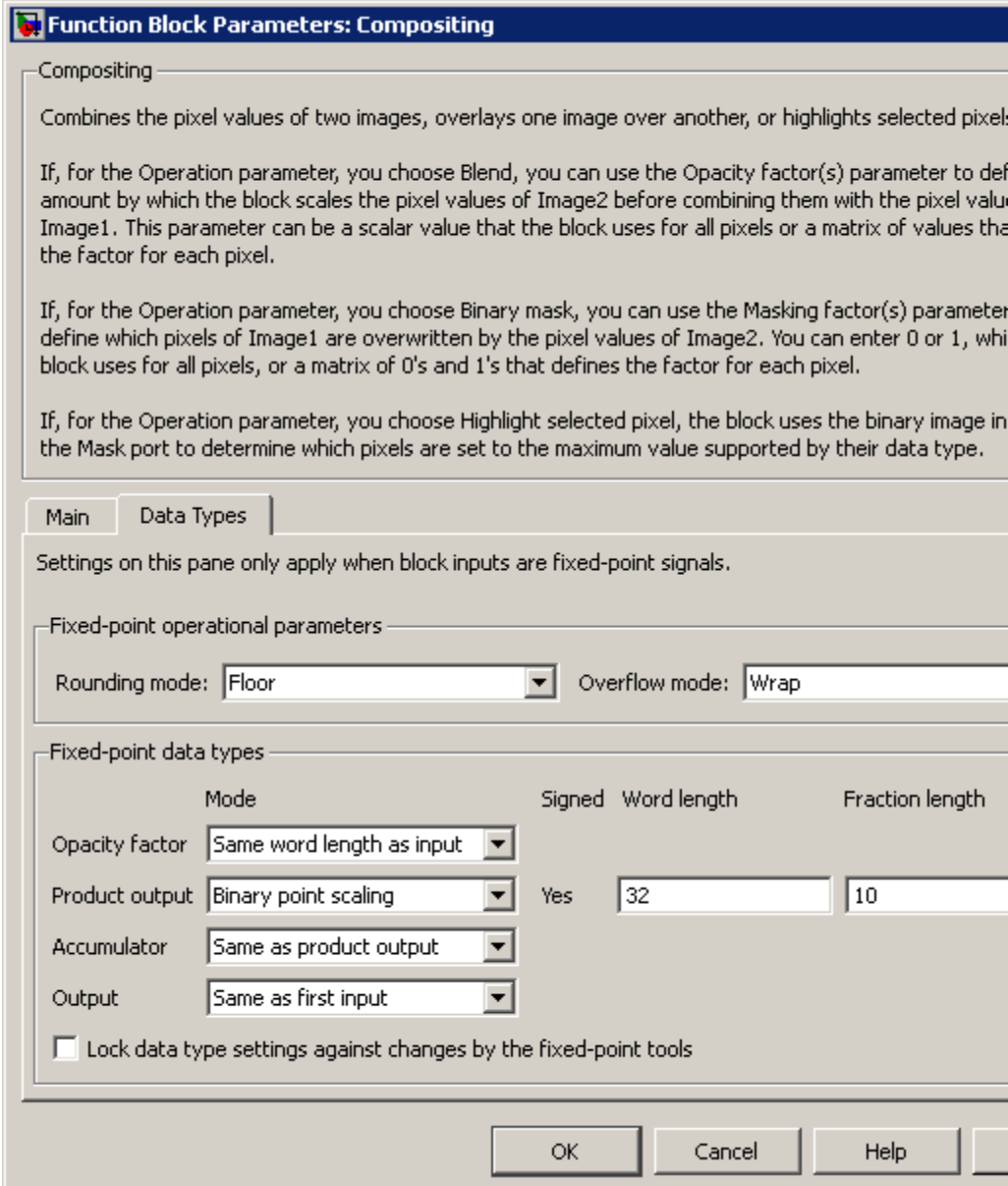
Location [row column]

Enter a two-element vector that specifies the row and column position of the upper-left corner of the image input at port **Image**

Compositing

2 relative to the upper-left corner of the image input at port Image 1. This parameter is visible if, for the **Location source** parameter you choose **Specify via dialog**. Tunable.

The **Data Types** pane of the Compositing dialog box appears as follows. These parameters are applicable only when the **Operation** parameter is set to **Blend**.

The image shows a software dialog box titled "Function Block Parameters: Compositing". It has a blue header bar with a small icon on the left. The main area is divided into sections. The top section, "Compositing", contains three paragraphs of text explaining the "Operation" parameter options: Blend, Binary mask, and Highlight selected pixel. Below this is a tabbed interface with "Main" and "Data Types" tabs. The "Data Types" tab is active and contains a note about fixed-point signals, a section for "Fixed-point operational parameters" with dropdowns for "Rounding mode" (set to "Floor") and "Overflow mode" (set to "Wrap"), and a section for "Fixed-point data types" with a table of settings for "Opacity factor", "Product output", "Accumulator", and "Output". At the bottom of the "Data Types" section is a checkbox for "Lock data type settings against changes by the fixed-point tools". The dialog box ends with "OK", "Cancel", and "Help" buttons.

Function Block Parameters: Compositing

Compositing

Combines the pixel values of two images, overlays one image over another, or highlights selected pixels.

If, for the Operation parameter, you choose Blend, you can use the Opacity factor(s) parameter to define the amount by which the block scales the pixel values of Image2 before combining them with the pixel values of Image1. This parameter can be a scalar value that the block uses for all pixels or a matrix of values that the block uses for each pixel.

If, for the Operation parameter, you choose Binary mask, you can use the Masking factor(s) parameter to define which pixels of Image1 are overwritten by the pixel values of Image2. You can enter 0 or 1, which the block uses for all pixels, or a matrix of 0's and 1's that defines the factor for each pixel.

If, for the Operation parameter, you choose Highlight selected pixel, the block uses the binary image in the Mask port to determine which pixels are set to the maximum value supported by their data type.

Main Data Types

Settings on this pane only apply when block inputs are fixed-point signals.

Fixed-point operational parameters

Rounding mode: Floor Overflow mode: Wrap

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Opacity factor	Same word length as input			
Product output	Binary point scaling	Yes	32	10
Accumulator	Same as product output			
Output	Same as first input			

Lock data type settings against changes by the fixed-point tools

OK Cancel Help

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Opacity factor

Choose how to specify the word length and fraction length of the opacity factor:

- When you select **Same word length as input**, these characteristics match those of the input to the block.
- When you select **Specify word length**, enter the word length of the opacity factor.
- When you select **Binary point scaling**, you can enter the word length of the opacity factor, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, of the opacity factor. The bias of all signals in the Video and Image Processing Blockset software is 0.

Product output

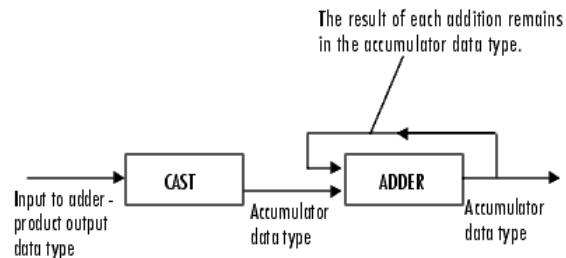


As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths.

- When you select **Same as first input**, these characteristics match those of the input to the block.

- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset software is 0.

Accumulator



As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

- When you select **Same as product output**, these characteristics match those of the product output.
- When you select **Same as first input**, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset software is 0.

Output

Choose how to specify the word length and fraction length of the output of the block:

- When you select `Same as first input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the Video and Image Processing Blockset software is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

See Also

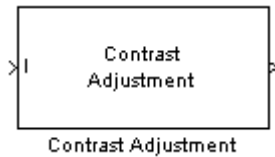
Insert Text

Video and Image Processing Blockset software

Purpose Adjust image contrast by linearly scaling pixel values

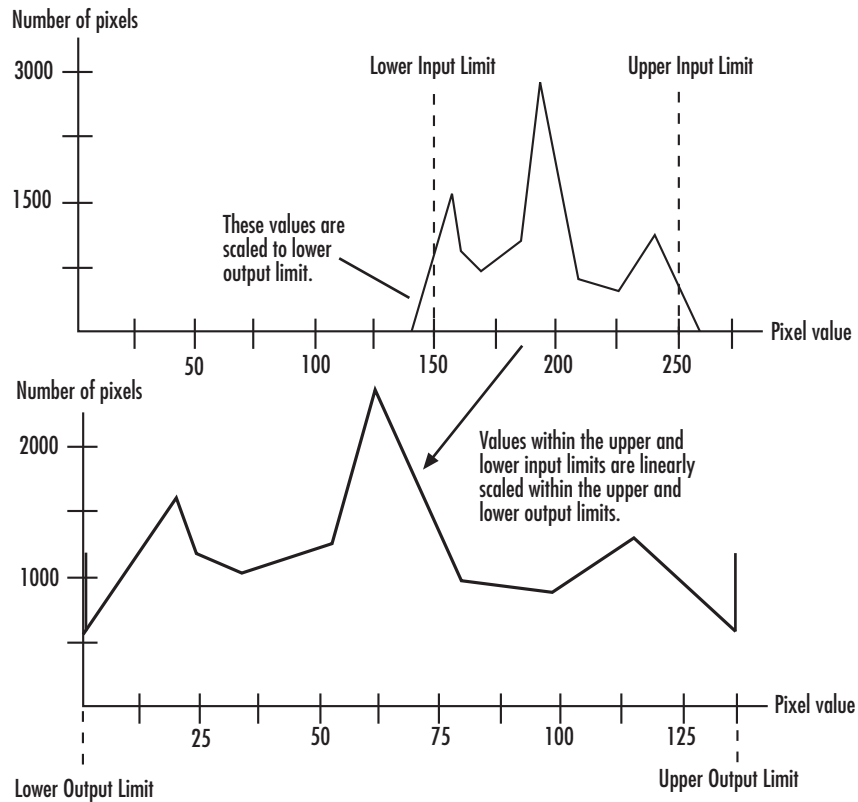
Library Analysis & Enhancement
vipanalysis

Description



The Contrast Adjustment block adjusts the contrast of an image by linearly scaling the pixel values between upper and lower limits. Pixel values that are above or below this range are saturated to the upper or lower limit value, respectively.

Contrast Adjustment



Mathematically, the contrast adjustment operation is described by the following equation, where the input limits are $[low_in\ high_in]$ and the output limits are $[low_out\ high_out]$:

$$Output = \left\{ \begin{array}{l} low_out, \quad Input \leq low_in \\ low_out + (Input - low_in) \frac{high_out - low_out}{high_in - low_in}, \quad low_in < Input < high_in \\ high_out, \quad Input \geq high_in \end{array} \right\}$$

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
Output	Scalar, vector, or matrix of intensity values or a scalar, vector, or matrix that represents one plane of the RGB video stream	Same as I port	No

Specifying upper and lower limits

Use the **Adjust pixel values from** and **Adjust pixel values to** parameters to specify the upper and lower input and output limits. All options are described below.

Input limits

Use the **Adjust pixel values from** parameter to specify the upper and lower input limits.

If you select **Full input data range [min max]**, uses the minimum input value as the lower input limit and the maximum input value as the upper input limit.

If you select **User-defined**, the **Range [low high]** parameter associated with this option appears. Enter a two-element vector of scalar values, where the first element corresponds to the lower input limit and the second element corresponds to the upper input limit.

If you select **Range determined by saturating outlier pixels**, the **Percentage of pixels to saturate [low high] (in %)**, **Specify number of histogram bins (used to calculate the range when**

Contrast Adjustment

outliers are eliminated), and **Number of histogram bins** parameters appear on the block. The block uses these parameter values to calculate the input limits in this three-step process:

- 1** Find the minimum and maximum input values, [*min_in max_in*].
- 2** Scale the pixel values from [**min_in max_in**] to [0 *num_bins*-1], where *num_bins* is the scalar value you specify in the **Number of histogram bins** parameter. This parameter always displays the value used by the block. Then the block calculates the histogram of the scaled input. For additional information about histograms, see the Histogram block reference page.
- 3** Find the lower input limit such that the percentage of pixels with values smaller than the lower limit is at most the value of the first element of the **Percentage of pixels to saturate [low high] (in %)** parameter. Similarly, find the upper input limit such that the percentage of pixels with values greater than the upper limit is at least the value of the second element of the parameter.

Output limits

Use the **Adjust pixel values to** parameter to specify the upper and lower output limits.

If you select `Full data type range`, the block uses the minimum value of the input data type as the lower output limit and the maximum value of the input data type as the upper out

If you select `User-defined range`, the **Range [low high]** parameter appears on the block. Enter a two-element vector of scalar values, where the first element corresponds to the lower output limit and the second element corresponds to the upper output limit.

For INF, -INF and NAN Input Values

If any input pixel value is either `INF` or `-INF`, the Contrast Adjustment block will change the pixel value according to how the parameters are set. The following table shows how the block handles these pixel values.

If Adjust pixel values from parameter is set to...	Contrast Adjustment block will:
Full data range [min,max]	Set the entire output image to the lower limit of the Adjust pixel values to parameter setting.
Range determined by saturating outlier pixels	
User defined range	Lower and higher limits of the Adjust pixel values to parameter set to -INF and INF, respectively.

If any input pixel has a NAN value, the block maps the pixels with valid numerical values according to the user-specified method. It maps the NAN pixels to the lower limit of the **Adjust pixels values to** parameter.

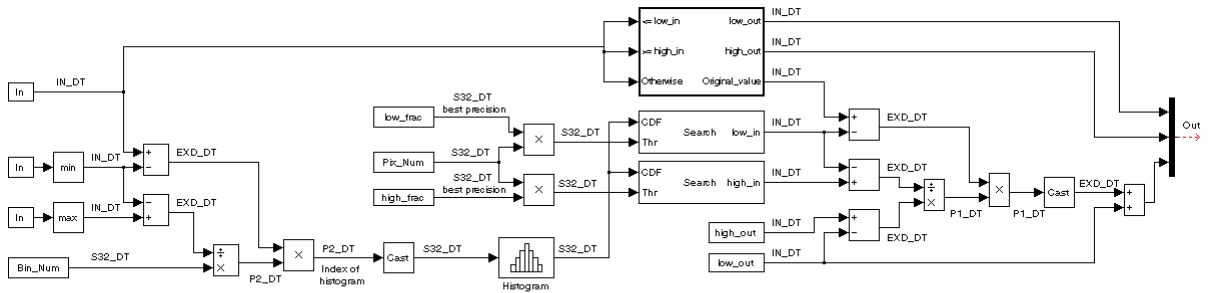
Examples

See “Adjusting the Contrast in Intensity Images” in the *Video and Image Processing Blockset User’s Guide*.

Fixed-Point Data Types

The following diagram shows the data types used in the Contrast Adjustment block for fixed-point signals:

Contrast Adjustment

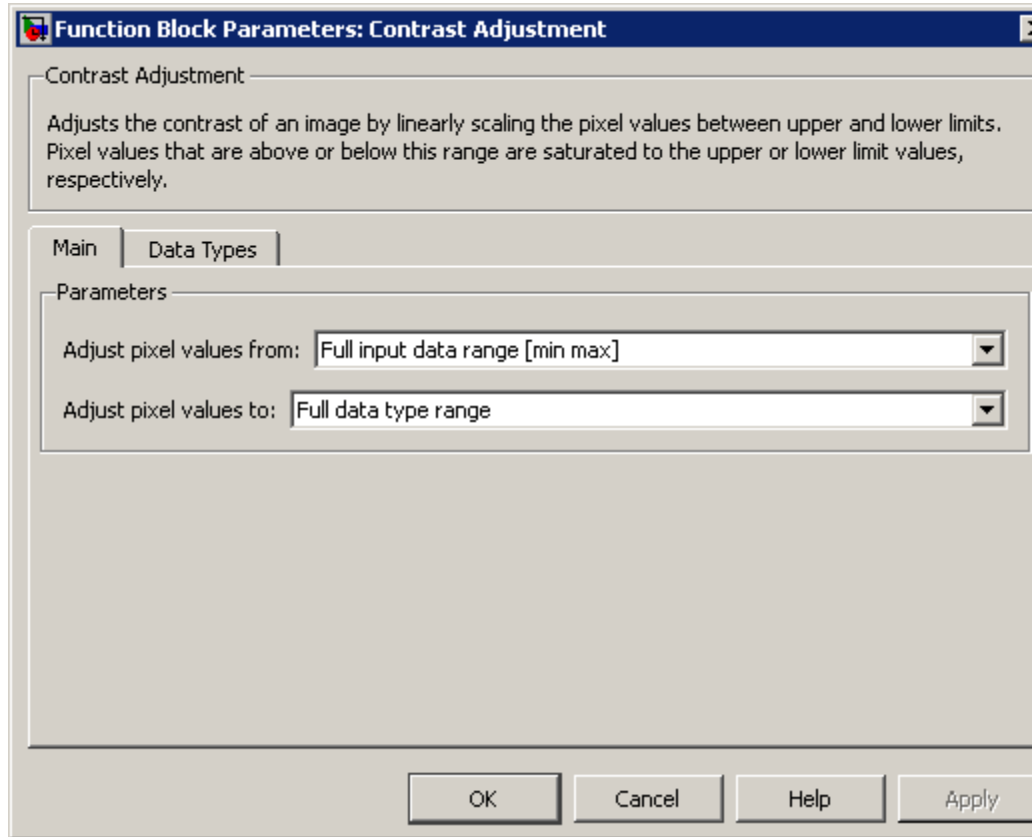


FIXED-POINT TAB NAMES:
 =====
 IN_DT: Datatype of the input image
 EXD_DT: IN_DT with extra bit for sign
 P1_DT: Datatype of product 1
 P2_DT: Datatype of product 2

SUB-SYSTEMS:
 =====
 Search:
 Calculate the pixel value that corresponds
 to Thr in the cumulative histogram (CDF).

Dialog Box

The Contrast Adjustment dialog box appears as shown in the following figure.



Adjust pixel values from

Specify how to enter the upper and lower input limits. Your choices are Full input data range [min max], User-defined, and Range determined by saturating outlier pixels.

Contrast Adjustment

Range [low high]

Enter a two-element vector of scalar values. The first element corresponds to the lower input limit, and the second element corresponds to the upper input limit. This parameter is visible if, for the **Adjust pixel values from** parameter, you select User-defined.

Percentage of pixels to saturate [low high] (in %)

Enter a two-element vector. The block calculates the lower input limit such that the percentage of pixels with values smaller than the lower limit is at most the value of the first element. It calculates the upper input limit similarly. This parameter is visible if, for the **Adjust pixel values from** parameter, you select Range determined by saturating outlier pixels.

Specify number of histogram bins (used to calculate the range when outliers are eliminated)

Select this check box to change the number of histogram bins. This parameter is editable if, for the **Adjust pixel values from** parameter, you select Range determined by saturating outlier pixels.

Number of histogram bins

Enter the number of histogram bins to use to calculate the scaled input values. This parameter is available if you select the **Specify number of histogram bins (used to calculate the range when outliers are eliminated)** check box.

Adjust pixel values to

Specify the upper and lower output limits. If you select Full data type range, the block uses the minimum value of the input data type as the lower output limit and the maximum value of the input data type as the upper output limit. If you select User-defined range, the **Range [low high]** parameter appears on the block.

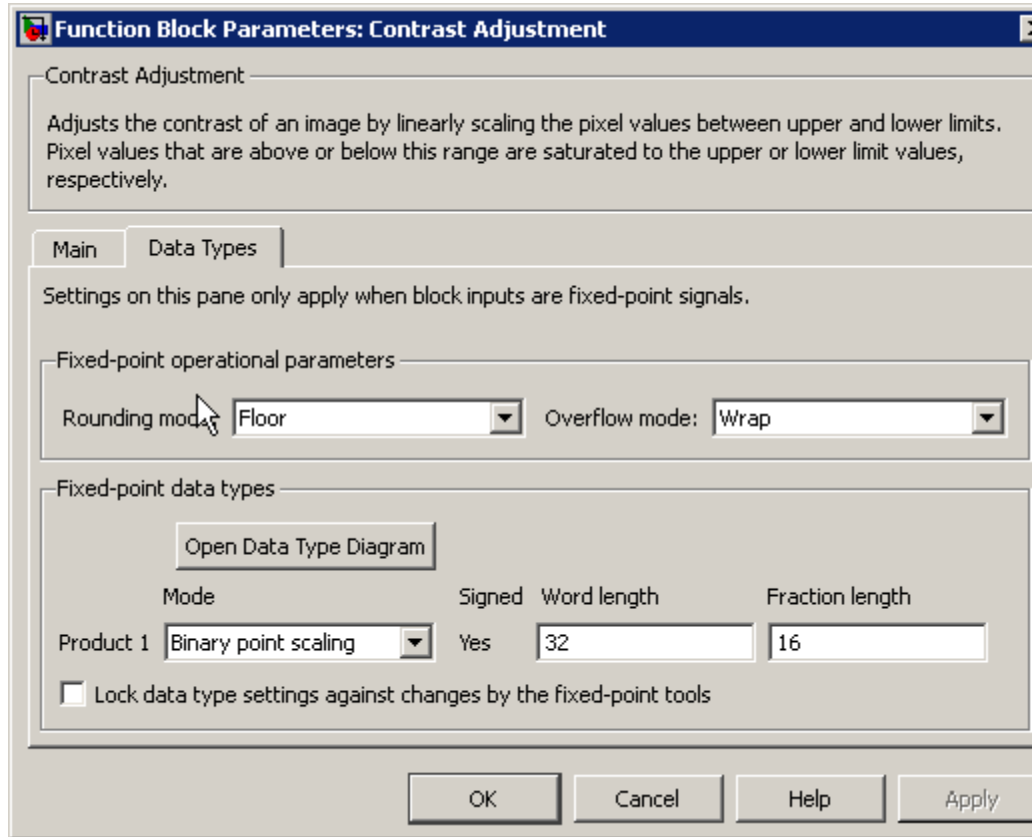
Range [low high]

Enter a two-element vector of scalar values. The first element corresponds to the lower output limit and the second element corresponds to the upper output limit. This parameter is

Contrast Adjustment

visible if, for the **Adjust pixel values to** parameter, you select **User-defined range**

The **Data Types** pane of the Contrast Adjustment dialog box appears as shown in the following figure.



Rounding mode

Select the rounding mode for fixed-point operations.

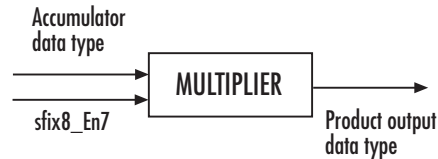
Contrast Adjustment

Overflow mode

Select the overflow mode for fixed-point operations.

Product 1

The product output type when the block calculates the ratio between the input data range and the number of histogram bins.



As shown in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.

When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset software is 0.

Product 2

The product output type when the block calculates the bin location of each input value.



As shown in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.

When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset software is 0.

This parameter is visible if, for the **Adjust pixel values from** parameter, you select **Range determined by saturating outlier pixels**.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

See Also

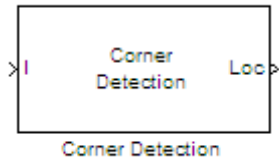
Histogram	Video and Image Processing Blockset software
Histogram Equalization	Video and Image Processing Blockset software

Corner Detection

Purpose Calculate corner metric matrix and find corners in images

Library Analysis & Enhancement
vipanalysis

Description



The Corner Detection block finds corners in an image using the Harris corner detection, minimum eigenvalue, or local intensity comparison method. The block finds the corners in the image based on the pixels that have the largest corner metric values.

For the most accurate results, use the “Minimum Eigenvalue Method” on page 2-279. For the fastest computation, use the “Local Intensity Comparison” on page 2-280. For the trade-off between accuracy and computation, use the “Harris Corner Detection Method” on page 2-279.

Input/Output	Description
I	Matrix of intensity values
Loc	2-by-N matrix that represents the locations of the corners where N is the maximum number of corners
Count	Scalar value that represents the number of detected corners
Metric	Matrix of corner metric values that is the same size as the input image

Minimum Eigenvalue Method

This method is more computationally expensive than the Harris corner detection algorithm because it directly calculates the eigenvalues of the sum of the squared difference matrix, M .

The sum of the squared difference matrix, M , is defined as follows:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

The previous equation is based on the following values:

$$A = (I_x)^2 \otimes w$$

$$B = (I_y)^2 \otimes w$$

$$C = (I_x I_y)^2 \otimes w$$

where I_x and I_y are the gradients of the input image, I , in the x and y direction, respectively. The \otimes symbol denotes a convolution operation.

Use the **Coefficients for separable smoothing filter** parameter to define a vector of filter coefficients. The block multiplies this vector of coefficients by its transpose to create a matrix of filter coefficients, w .

The block calculates the smaller eigenvalue of the sum of the squared difference matrix. This minimum eigenvalue corresponds to the corner metric matrix.

Harris Corner Detection Method

The Harris corner detection method avoids the explicit computation of the eigenvalues of the sum of squared differences matrix by solving for the following corner metric matrix, R :

$$R = AB - C^2 - k(A + B)^2$$

A , B , C are defined in the previous section, “Minimum Eigenvalue Method” on page 2-279.

Corner Detection

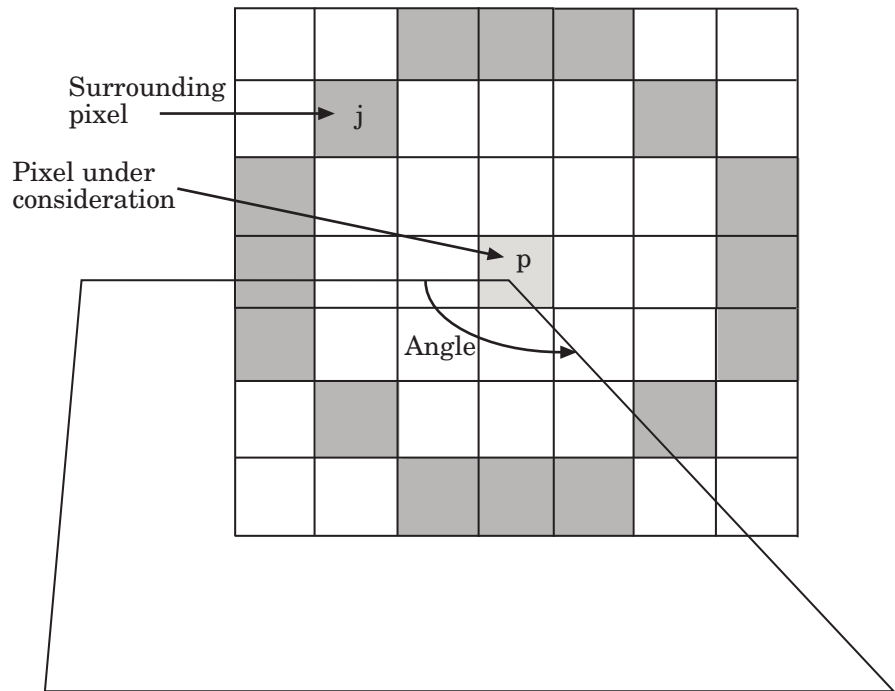
The variable k corresponds to the sensitivity factor. You can specify its value using the **Sensitivity factor ($0 < k < 0.25$)** parameter. The smaller the value of k , the more likely it is that the algorithm can detect sharp corners.

Use the **Coefficients for separable smoothing filter** parameter to define a vector of filter coefficients. The block multiplies this vector of coefficients by its transpose to create a matrix of filter coefficients, w .

Local Intensity Comparison

This method determines that a pixel is a possible corner if it has either, N contiguous valid bright surrounding pixels, or N contiguous dark surrounding pixels. Specifying the value of N is discussed later in this section. The next section explains how the block finds these surrounding pixels.

Suppose that p is the pixel under consideration and j is one of the pixels surrounding p . The locations of the other surrounding pixels are denoted by the shaded areas in the following figure.



I_p and I_j are the intensities of pixels p and j , respectively. Pixel j is a valid bright surrounding pixel if $I_j - I_p \geq T$. Similarly, pixel j is a valid dark surrounding pixel if $I_p - I_j \geq T$. In these equations, T is the value you specified for the **Intensity comparison threshold** parameter.

The block repeats this process to determine whether the block has N contiguous valid surrounding pixels. The value of N is related to the value you specify for the **Maximum angle to be considered a corner (in degrees)**, as shown in the following table.

Corner Detection

Number of Valid Surrounding Pixels, N	Angle (degrees)
15	22.5
14	45
13	67.5
12	90
11	112.5
10	135
9	157.5

After the block determines that a pixel is a possible corner, it computes its corner metric using the following equation:

$$R = \max \left(\sum_{j: I_j \geq I_p + T} |I_p - I_j| - T, \sum_{j: I_j \leq I_p - T} |I_p - I_j| - T \right)$$

Block Output

Use the **Output** parameter to determine whether the block outputs the corner location, corner location and metric matrix, or the metric matrix. The block outputs the corner locations in a 2-by-N matrix where each row stores the row and column locations of the corners and N is the maximum number of corners. The block outputs the corner metric values in a matrix that is the same size as the input image.

If you set the **Output** parameter to **Corner location** or **Corner location and metric matrix**, the **Maximum number of corners**, **Minimum metric value that indicates a corner**, and **Neighborhood size (suppress region around detected corners)** parameters appear on the block.

To determine the final corner values, the block follows this process:

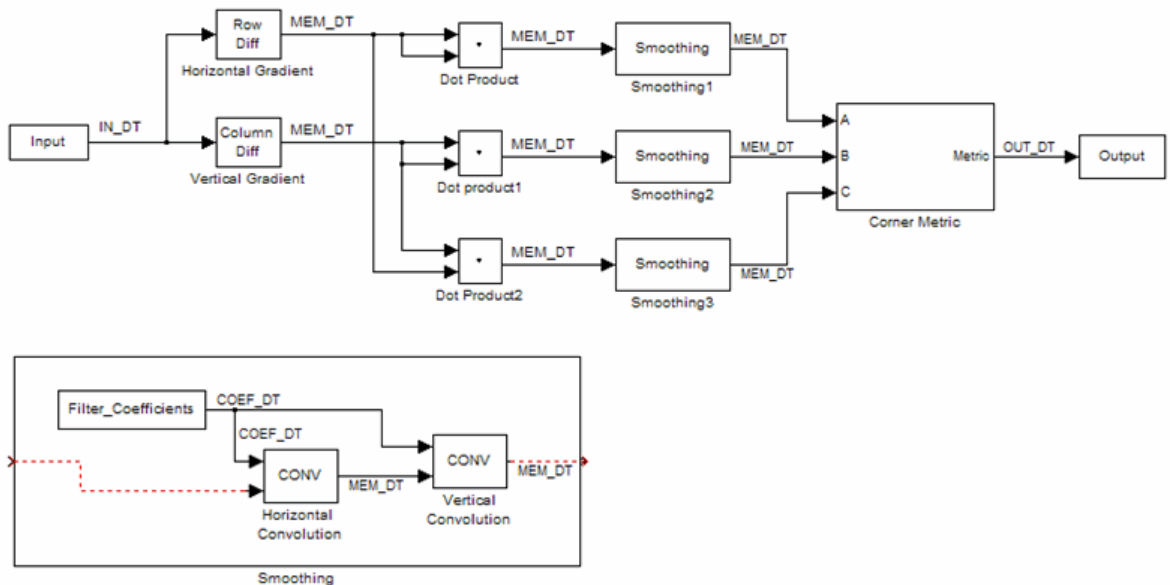
- 1 Find the pixel with the largest corner metric value.

- 2 Verify that the metric value is greater than or equal to the value you specified for the **Minimum metric value that indicates a corner** parameter.
- 3 Suppress the region around the corner value by the size defined in the **Neighborhood size (suppress region around detected corners)** parameter.

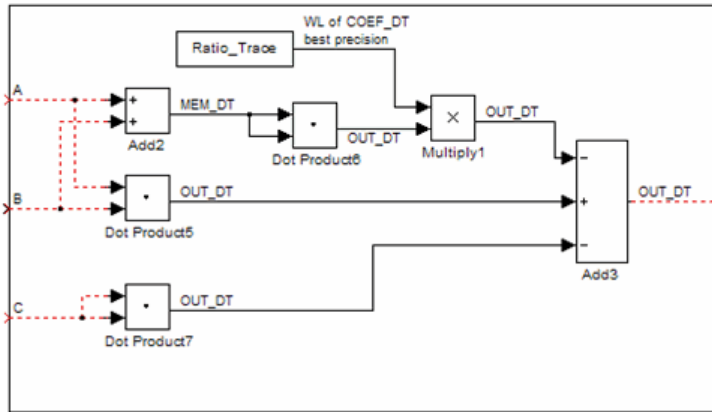
The block repeats this process until it finds all the corners in the image or it finds the number of corners you specified in the **Maximum number of corners** parameter.

Fixed-Point Data Types

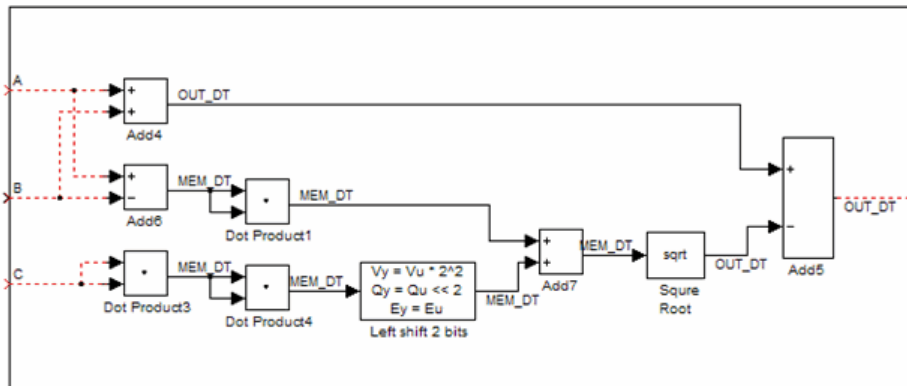
The following diagram shows the data types used in the Corner Detection block for fixed-point signals. These diagrams apply to the Harris corner detection and minimum eigenvalue methods only.



Corner Detection



Corner Metric by Harris Algorithm



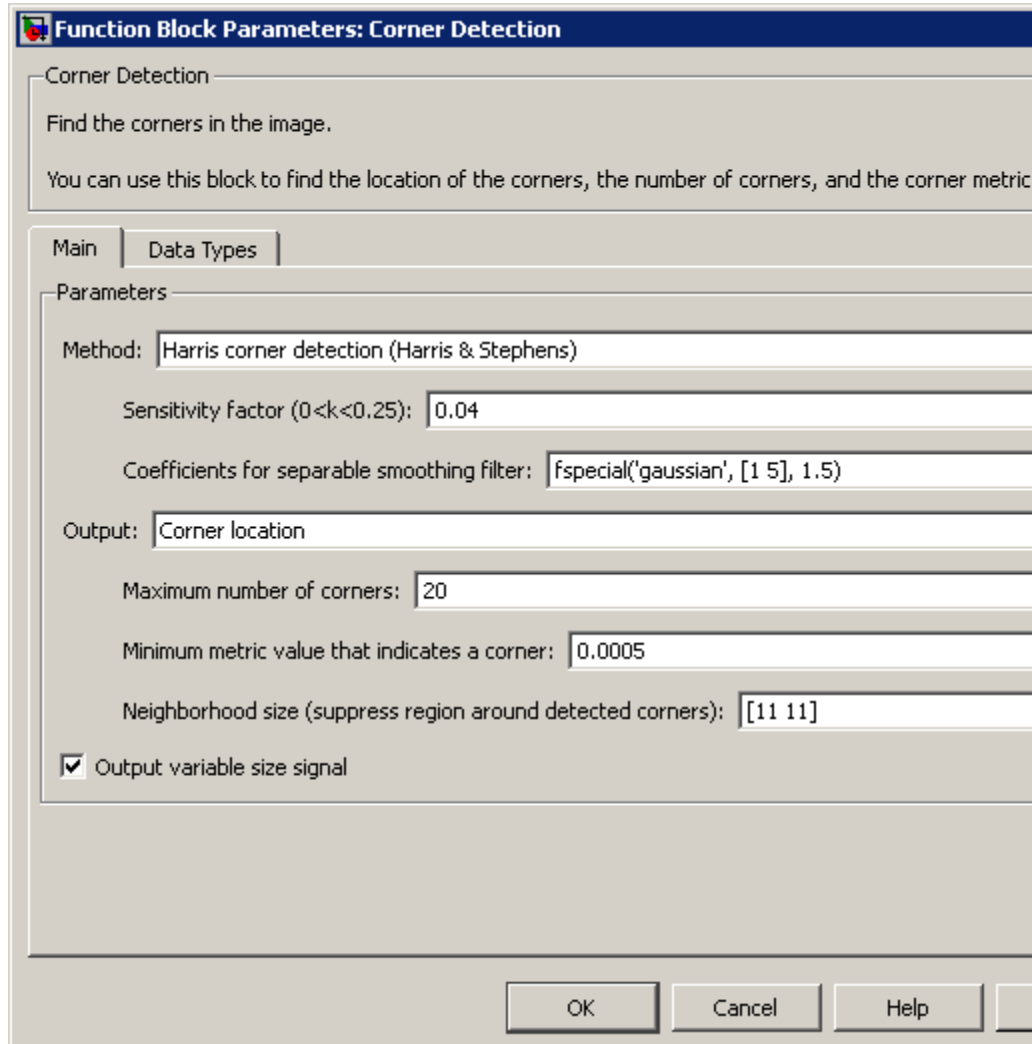
Corner Metric by Minimum Eigenvalue Algorithm

The following table summarizes the variables used in the previous diagrams.

Variable Name	Definition
IN_DT	Input data type
MEM_DT	Memory data type
OUT_DT	Metric output data type
COEF_DT	Coefficients data type

Dialog Box

The Corner Detection dialog box appears as shown in the following figure.



Corner Detection

Method

Specify the method to use to find the corner values. Your choices are Harris corner detection (Harris & Stephens), Minimum eigenvalue (Shi & Tomasi), and Local intensity comparison (Rosen & Drummond).

Sensitivity factor ($0 < k < 0.25$)

Specify the sensitivity factor, k . The smaller the value of k the more likely the algorithm is to detect sharp corners. This parameter is visible if you set the **Method** parameter to Harris corner detection (Harris & Stephens). This parameter is tunable.

Coefficients for separable smoothing filter

Specify a vector of filter coefficients for the smoothing filter. This parameter is visible if you set the **Method** parameter to Harris corner detection (Harris & Stephens) or Minimum eigenvalue (Shi & Tomasi).

Intensity comparison threshold

Specify the threshold value used to find valid surrounding pixels. This parameter is visible if you set the **Method** parameter to Local intensity comparison (Rosen & Drummond). This parameter is tunable.

Maximum angle to be considered a corner (in degrees)

Specify the maximum corner angle. This parameter is visible if you set the **Method** parameter to Local intensity comparison (Rosen & Drummond). This parameter is tunable for Simulation only.

Output

Specify the block output. Your choices are Corner location, Corner location and metric matrix, and Metric matrix.

Maximum number of corners

Enter the maximum number of corners you want the block to find. This parameter is visible if you set the **Output** parameter to Corner location or Corner location and metric matrix.

Minimum metric value that indicates a corner

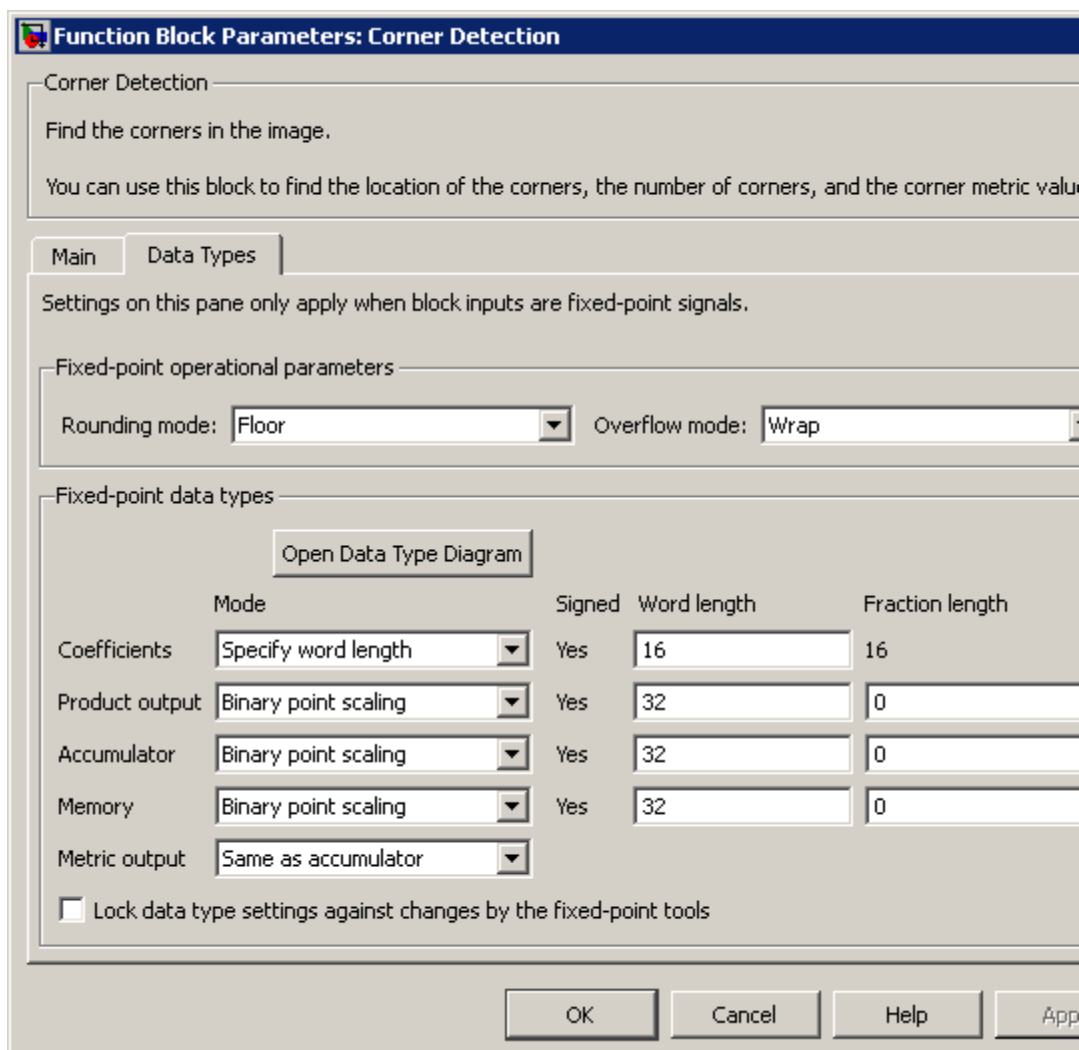
Specify the minimum corner metric value. This parameter is visible if you set the **Output** parameter to `Corner location` or `Corner location and metric matrix`. This parameter is tunable.

Neighborhood size (suppress region around detected corners)

Specify the size of the neighborhood around the corner metric value over which the block zeros out the values. Enter a two-element vector of positive odd integers, `[r c]`. Here, `r` is the number of rows in the neighborhood and `c` is the number of columns. This parameter is visible if you set the **Output** parameter to `Corner location` or `Corner location and metric matrix`.

The **Data Types** pane of the Corner Detection dialog box appears as shown in the following figure.

Corner Detection



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Coefficients

Choose how to specify the word length and the fraction length of the coefficients:

- When you select **Same word length as input**, the word length of the coefficients match that of the input to the block. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select **Specify word length**, you can enter the word length of the coefficients, in bits. The block automatically sets the fraction length to give you the best precision.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the coefficients, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the coefficients. The bias of all signals in the Video and Image Processing Blockset software is 0.

Product output

As shown in the following figure, the output of the multiplier is placed into the product output data type and scaling.



Use this parameter to specify how to designate the product output word and fraction lengths.

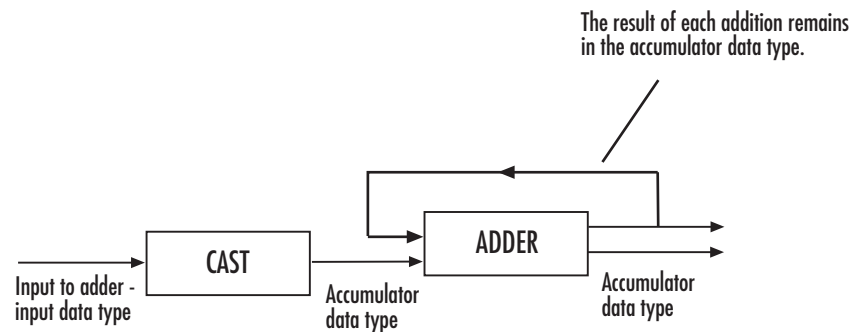
- When you select **Same as input**, these characteristics match those of the input to the block.

Corner Detection

- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset software is 0.

Accumulator

As shown in the following figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it.



Use this parameter to specify how to designate this accumulator word and fraction lengths:

- When you select **Same as input**, these characteristics match those of the input.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset software is 0.

Memory

Choose how to specify the memory word length and fraction length:

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of 0.

Metric output

Choose how to specify the metric output word length and fraction length:

- When you select `Same as accumulator`, these characteristics match those of the accumulator.
- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

References

- [1] C. Harris and M. Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*. August 1988, pp. 147-151.

Corner Detection

[2] J. Shi and C. Tomasi. “Good Features to Track.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. June 1994, pp. 593–600.

[3] E. Rosten and T. Drummond. “Fusing Points and Lines for High Performance Tracking.” *Proceedings of the IEEE International Conference on Computer Vision* Vol. 2 (October 2005): pp. 1508–1511.

Supported Data Types

Port	Supported Data Types
I	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer
Loc	32-bit unsigned integer
Count	32-bit unsigned integer
Metric	Same as I port

See Also

Find Local Maxima	Video and Image Processing Blockset software
Estimate Geometric Transformation	Video and Image Processing Blockset software

Purpose Remove motion artifacts by deinterlacing input video signal

Library Analysis & Enhancement

Description The Deinterlacing block takes the input signal, which is the combination of the top and bottom fields of the interlaced video, and converts it into deinterlaced video using line repetition, linear interpolation, or vertical temporal median filtering.



Note This block supports intensity and color images on its ports.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Combination of top and bottom fields of interlaced video	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
Output	Frames of deinterlaced video	Same as Input port	No

Use the **Deinterlacing method** parameter to specify how the block deinterlaces the video.

The following figure illustrates the block's behavior if you select Line repetition.

Deinterlacing

Line Repetition

Original Interlaced Video

Top Field			Bottom Field		
Row 1	A	B	C	Row 1	
Row 2				Row 2	D E F
Row 3	G	H	I	Row 3	
Row 4				Row 4	J K L
Row 5	M	N	O	Row 5	
Row 6				Row 6	P Q R

Block Input			Block Output - Deinterlaced Video		
Row 1	A	B	C	Row 1	A B C
Row 2	D	E	F	Row 2	A B C
Row 3	G	H	I	Row 3	G H I
Row 4	J	K	L	Row 4	G H I
Row 5	M	N	O	Row 5	M N O
Row 6	P	Q	R	Row 6	M N O

The following figure illustrates the block's behavior if you select Linear interpolation.

Linear Interpolation

Original Interlaced Video

Top Field				Bottom Field			
Row 1	A	B	C	Row 1			
Row 2				Row 2	D	E	F
Row 3	G	H	I	Row 3			
Row 4				Row 4	J	K	L
Row 5	M	N	O	Row 5			
Row 6				Row 6	P	Q	R

Block Input				Block Output - Deinterlaced Video			
Row 1	A	B	C	Row 1	A	B	C
Row 2	D	E	F	Row 2	$(A+G)/2$	$(B+H)/2$	$(C+I)/2$
Row 3	G	H	I	Row 3	G	H	I
Row 4	J	K	L	Row 4	$(G+M)/2$	$(H+N)/2$	$(I+O)/2$
Row 5	M	N	O	Row 5	M	N	O
Row 6	P	Q	R	Row 6	M	N	O

The following figure illustrates the block's behavior if you select Vertical temporal median filtering.

Deinterlacing

Vertical Temporal Median Filtering

Original Interlaced Video

Top Field			Bottom Field		
Row 1	A	B	C	Row 1	
Row 2				Row 2	D E F
Row 3	G	H	I	Row 3	
Row 4				Row 4	J K L
Row 5	M	N	O	Row 5	
Row 6				Row 6	P Q R

Block Input

Block Output - Deinterlaced Video

Row 1	A	B	C	Row 1	A	B	C
Row 2	D	E	F	Row 2	median([A,D,G])	median([B,E,H])	median([C,F,I])
Row 3	G	H	I	Row 3	G	H	I
Row 4	J	K	L	Row 4	median([G,J,M])	median([H,K,N])	median([L,O])
Row 5	M	N	O	Row 5	M	N	O
Row 6	P	Q	R	Row 6	M	N	O

Row-Major Data Format

The MATLAB environment and the Video and Image Processing Blockset software use column-major data organization. However, the Deinterlacing block gives you the option to process data that is stored in

row-major format. When you select the **Input image is transposed (data order is row major)** check box, the block assumes that the input buffer contains contiguous data elements from the first row first, then data elements from the second row second, and so on through the last row. Use this functionality only when you meet all the following criteria:

- You are developing algorithms to run on an embedded target that uses the row-major format.
- You want to limit the additional processing required to take the transpose of signals at the interfaces of the row-major and column-major systems.

When you use the row-major functionality, you must consider the following issues:

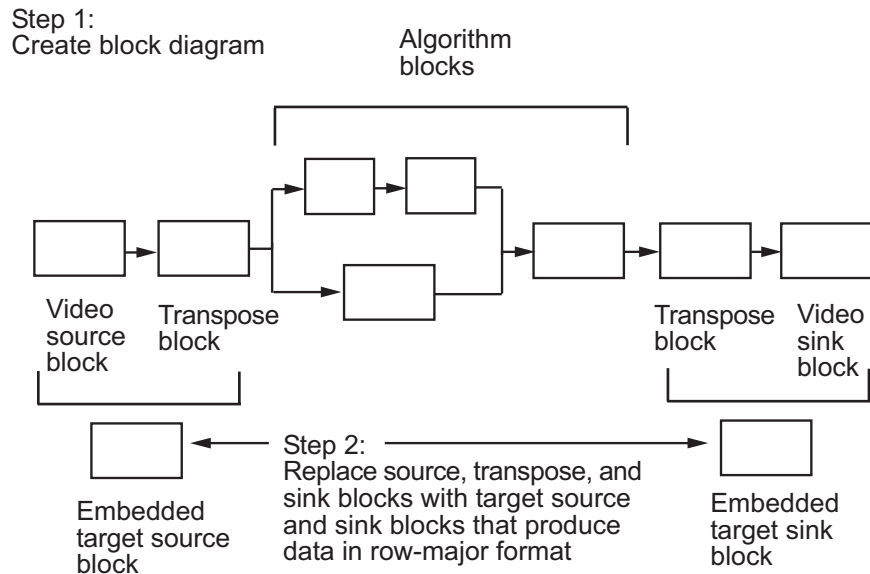
- When you select this check box, the first two signal dimensions of the Deinterlacing block's input are swapped.
- All the Video and Image Processing Blockset blocks can be used to process data that is in the row-major format, but you need to know the image dimensions when you develop your algorithms.

For example, if you use the 2-D FIR Filter block, you need to verify that your filter coefficients are transposed. If you are using the Rotate block, you need to use negative rotation angles, etc.

- Only three blocks have the **Input image is transposed (data order is row major)** check box. They are the Chroma Resampling, Deinterlacing, and Insert Text blocks. You need to select this check box to enable row-major functionality in these blocks. All other blocks must be properly configured to process data in row-major format.

Use the following two-step workflow to develop algorithms in row-major format to run on an embedded target.

Deinterlacing



See the DM642 EVM Video ADC and DM642 EVM Video DAC reference pages in the *Target Support Package User's Guide* for more information about data order in embedded targets.

Example

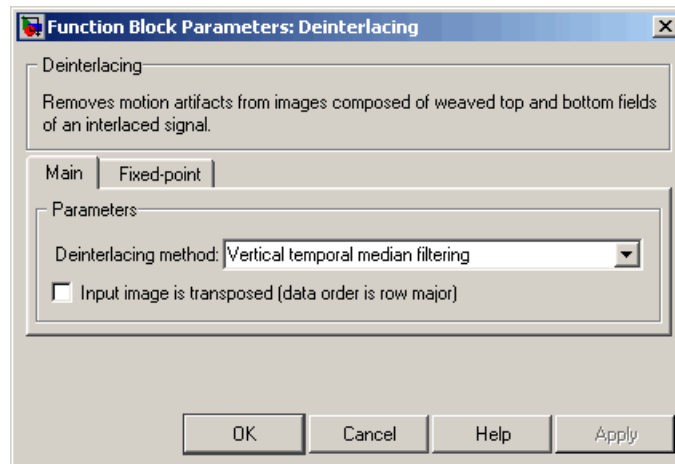
The following example shows you how to use the Deinterlacing block to remove motion artifacts from an image.

- 1 Open the example model by typing

```
doc_deinterlace
```

at the MATLAB command prompt.

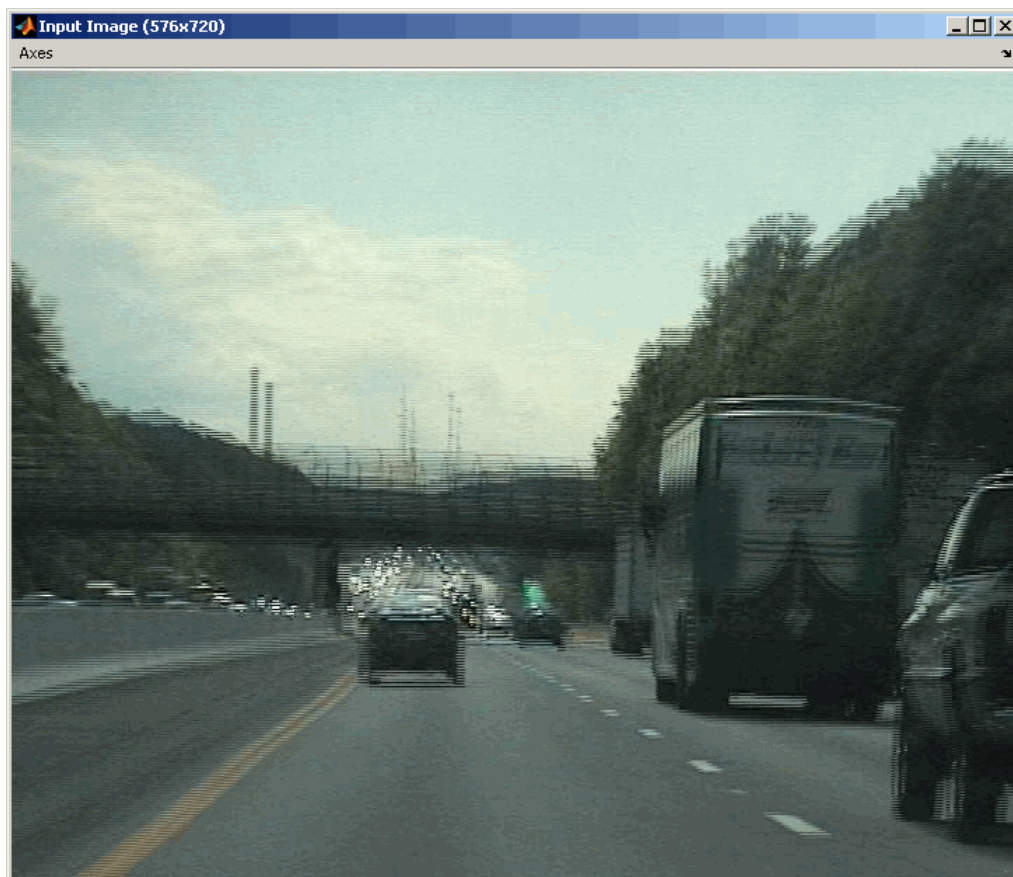
- 2 Double-click the Deinterlacing block. The model uses this block to remove the motion artifacts from the input image. The **Deinterlacing method** parameter is set to Vertical temporal median filtering.



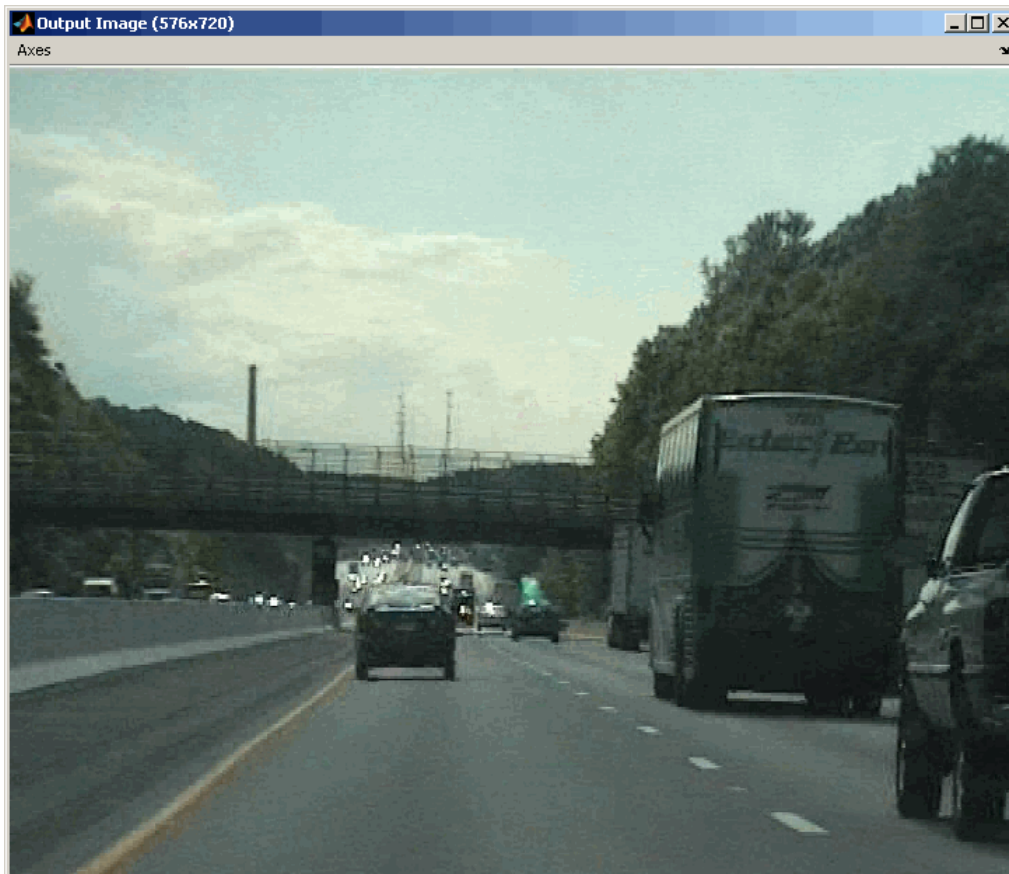
3 Run the model.

The original image that contains the motion artifacts appears in the Input Image window.

Deinterlacing



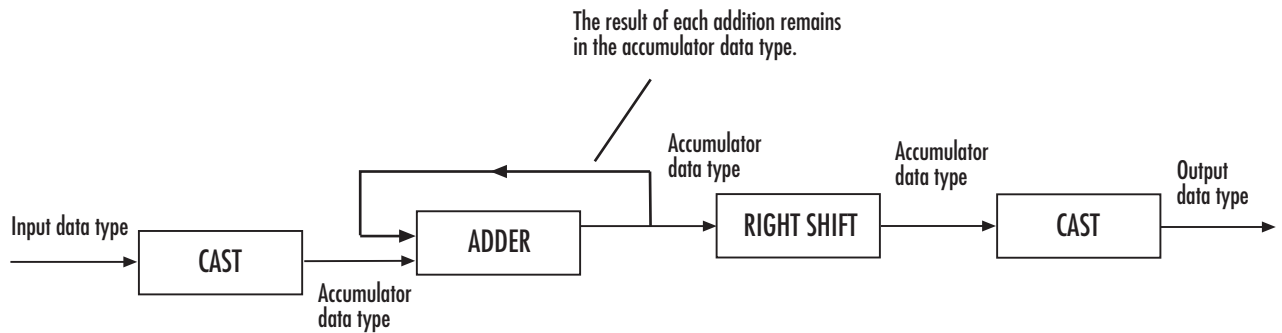
The clearer output image appears in the Output Image window.



Fixed-Point Data Types

The following diagram shows the data types used in the Deinterlacing block for fixed-point signals.

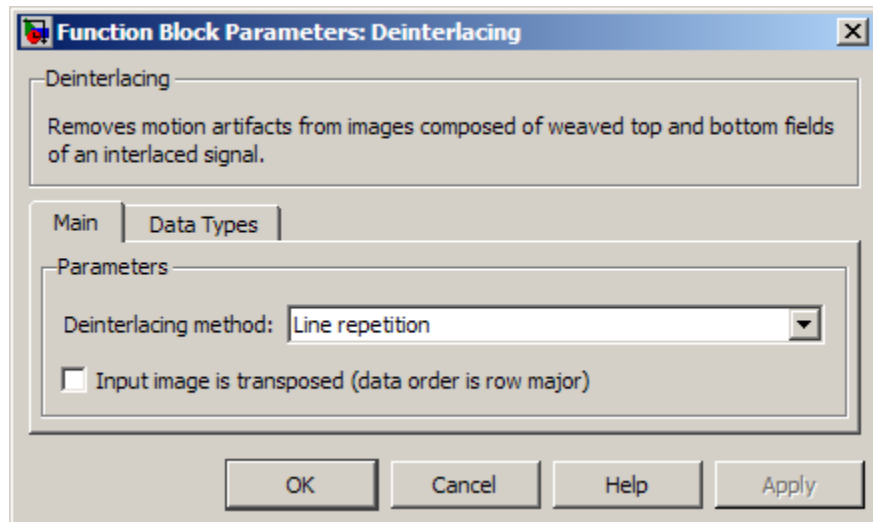
Deinterlacing



You can set the product output, accumulator, and output data types in the block mask as discussed in the next section.

Dialog Box

The **Main** pane of the Deinterlacing dialog box appears as shown in the following figure.



Deinterlacing method

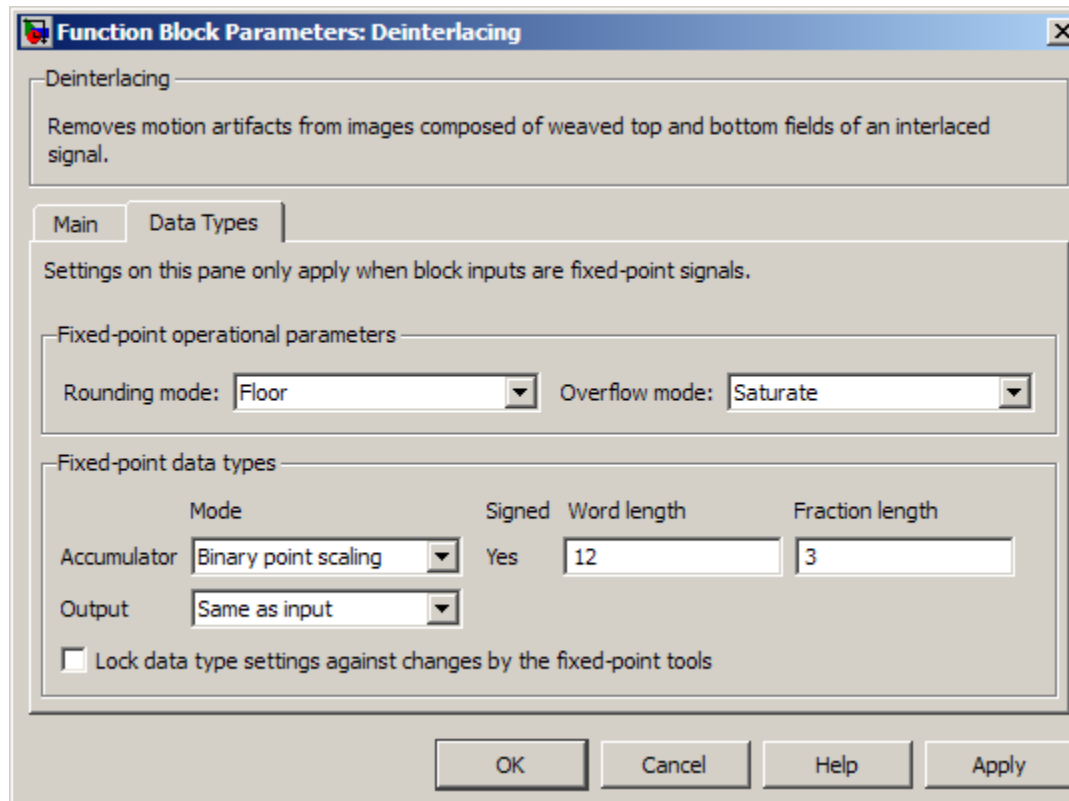
Specify how the block deinterlaces the video. Your choices are Line repetition, Linear interpolation, or Vertical temporal median filtering.

Input image is transposed (data order is row major)

When you select this check box, the block assumes that the input buffer contains data elements from the first row first, then data elements from the second row second, and so on through the last row.

The **Data Types** pane of the Deinterlacing dialog box appears as shown in the following figure.

Deinterlacing



Note The parameters on the **Data Types** pane are only available if, for the **Deinterlacing method**, you select **Linear interpolation**.

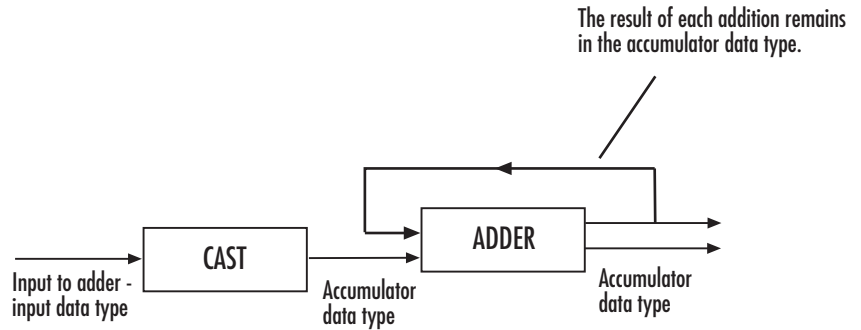
Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Accumulator



As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths:

- When you select **Same** as product output, these characteristics match those of the product output.
- When you select **Same** as input, these characteristics match those of the input.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Output

Choose how to specify the output word length and fraction length:

- When you select **Same** as input, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the output, in bits.

Deinterlacing

- When you select **Slope** and **bias scaling**, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of 0.

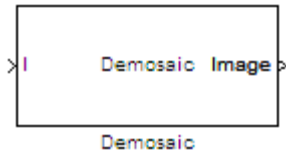
Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

Purpose Demosaic Bayer's format images

Library Conversions
vipconversions

Description



The following figure illustrates a 4-by-4 image in Bayer's format with each pixel labeled R, G, or B.

B	G	B	G
G	R	G	R
B	G	B	G
G	R	G	R

The Demosaic block takes in images in Bayer's format and outputs RGB images. The block performs this operation using a gradient-corrected linear interpolation algorithm or a bilinear interpolation algorithm.

Demosaic

Port	Input/Output	Supported Data Types	Complex Values Supported
I	<p>Matrix of intensity values</p> <ul style="list-style-type: none"> • If, for the Interpolation algorithm parameter, you select Bilinear, the number of rows and columns must be greater than or equal to 3. • If, for the Interpolation algorithm parameter, you select Gradient-corrected linear, the number of rows and columns must be greater than or equal to 5. 	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
R, G, B	<p>Matrix that represents one plane of the input RGB video stream. Outputs from the R, G, or B ports have the same data type.</p>	Same as I port	No
Image	<p>M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes.</p>	Same as I port	No

Use the **Interpolation algorithm** parameter to specify the algorithm the block uses to calculate the missing color information. If you select **Bilinear**, the block spatially averages neighboring pixels to calculate the color information. If you select **Gradient-corrected linear**, the block uses a Weiner approach to minimize the mean-squared error in

the interpolation. This method performs well on the edges of objects in the image. For more information, see [1].

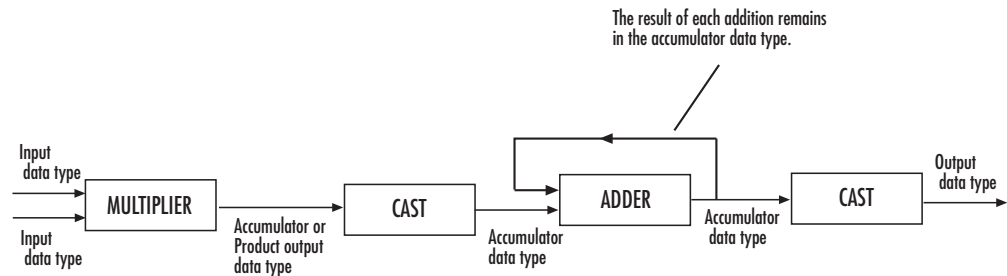
Use the **Sensor alignment** parameter to specify the alignment of the input image. Select the sequence of R, G and B pixels that correspond to the 2-by-2 block of pixels in the top-left corner of the image. You specify the sequence in left-to-right, top-to-bottom order. For example, for the image at the beginning of this reference page, you would select **BGGR**.

Both methods use symmetric padding at the image boundaries. For more information, see the Image Pad block reference page.

Use the **Output image signal** parameter to specify how to output a color video signal. If you select **One multidimensional signal**, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port outputs one M-by-N plane of an RGB video stream.

Fixed-Point Data Types

The following diagram shows the data types used in the Demosaic block for fixed-point signals.

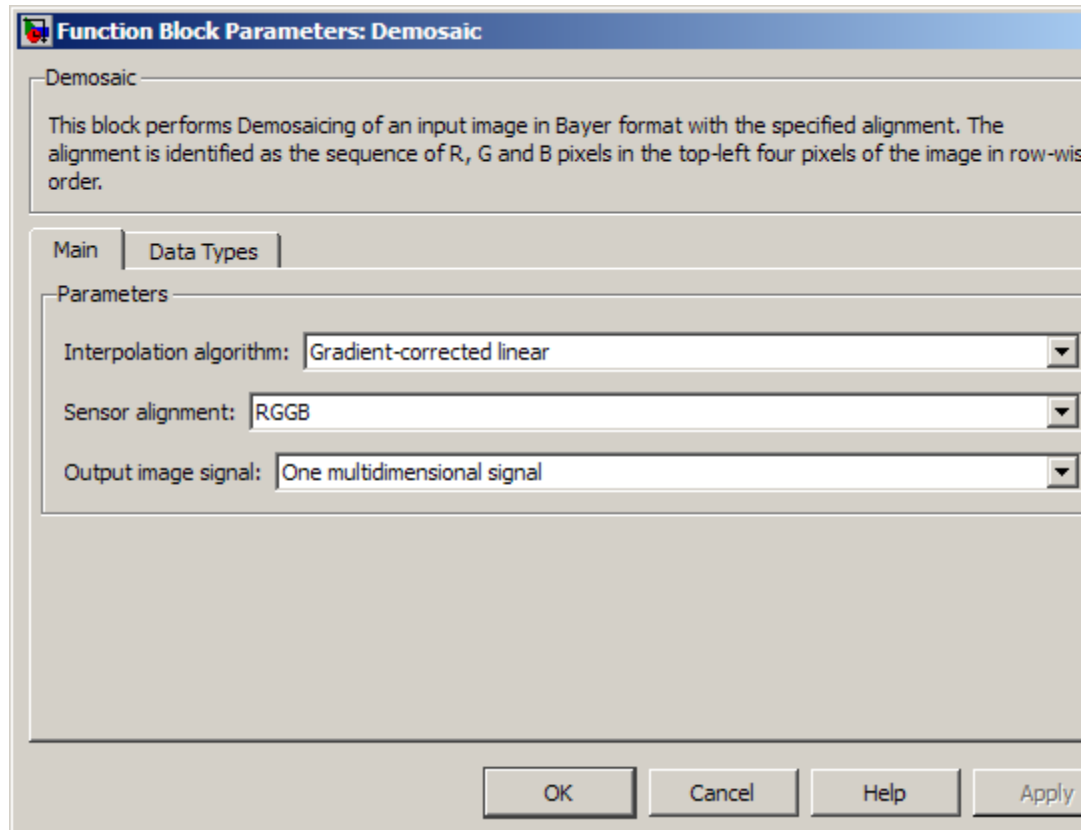


You can set the product output and accumulator data types in the block mask as discussed in the next section.

Demosaic

Dialog Box

The **Main** pane of the Demosaic dialog box appears as shown in the following figure.



Interpolation algorithm

Specify the algorithm the block uses to calculate the missing color information. Your choices are Bilinear or Gradient-corrected linear.

Sensor alignment

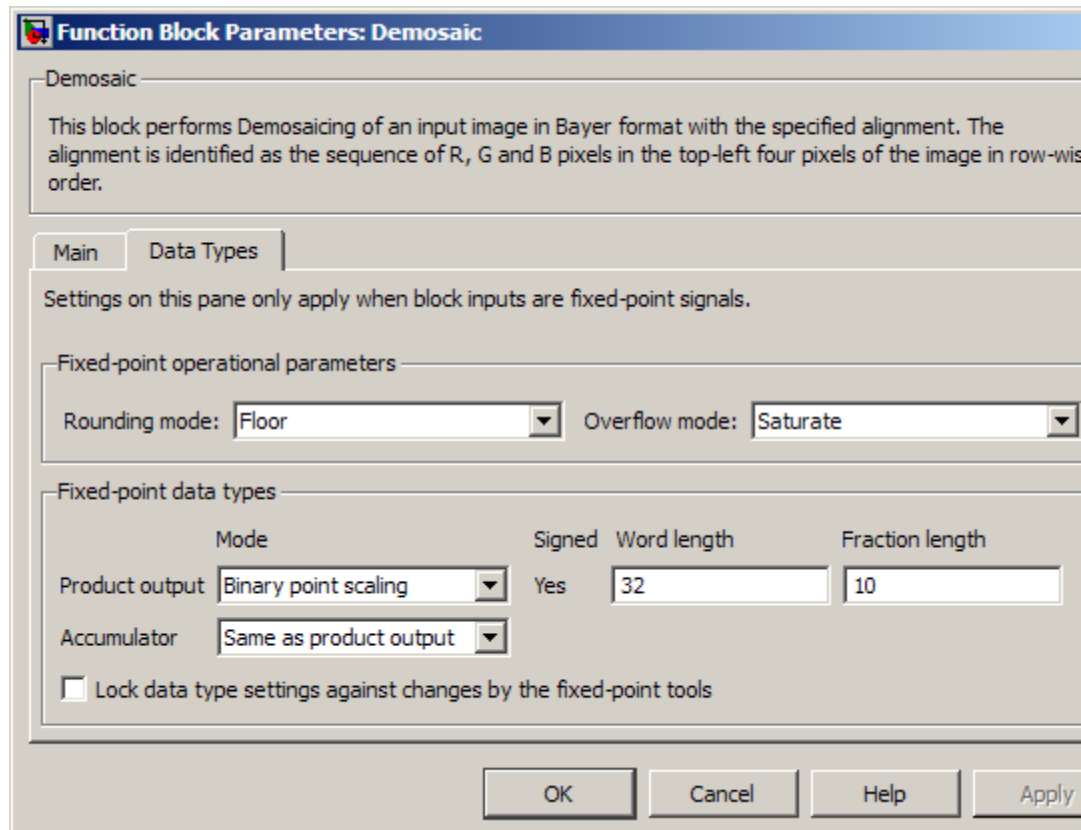
Select the sequence of R, G and B pixels that correspond to the 2-by-2 block of pixels in the top left corner of the image. You specify the sequence in left-to-right, top-to-bottom order.

Output image signal

Specify how to output a color video signal. If you select **One multidimensional signal**, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port outputs one M-by-N plane of an RGB video stream.

The **Data Types** pane of the Demosaic dialog box appears as shown in the following figure.

Demosaic



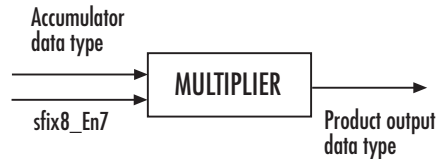
Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output



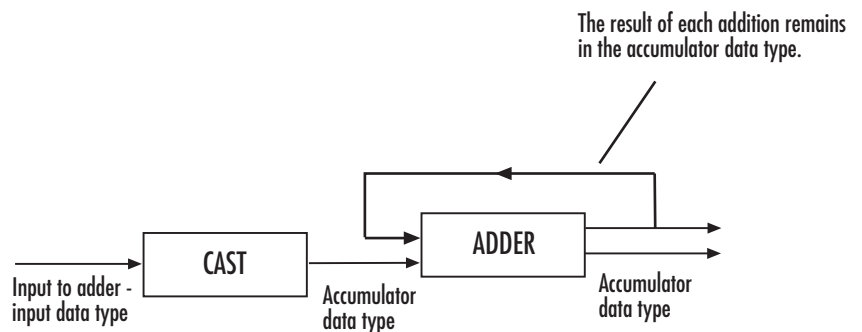
As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

When you select `Same` as `input`, these characteristics match those of the input to the block.

When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.

When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Accumulator



As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder

remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths:

- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as input`, these characteristics match those of the input.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

References

- [1] Malvar, Henrique S., Li-wei He, and Ross Cutler, "High-Quality Linear Interpolation for Demosaicing of Bayer-Patterned Color Images," *Microsoft Research*, One Microsoft Way, Redmond, WA 98052
- [2] Gunturk, Bahadir K., John Glotzbach, Yucel Altunbasak, Ronald W. Schafer, and Russel M. Mersereau, "Demosaicking: Color Filter Array Interpolation," *IEEE Signal Processing Magazine*, Vol. 22, Number 1, January 2005.

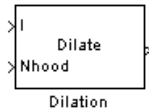
Purpose

Find local maxima in binary or intensity images

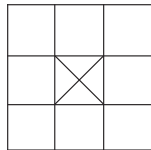
Library

Morphological Operations

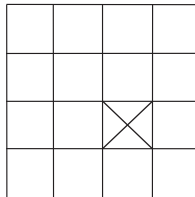
Description



The Dilation block rotates the neighborhood or structuring element 180 degrees. Then it slides the neighborhood or structuring element over an image, finds the local maxima, and creates the output matrix from these maximum values. If the neighborhood or structuring element has a center element, the block places the maxima there, as illustrated in the following figure.



If the neighborhood or structuring element does not have an exact center, the block has a bias toward the lower-right corner, as a result of the rotation. The block places the maxima there, as illustrated in the following figure.



This block uses flat structuring elements only.

Dilation

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• Boolean• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No
Nhood	Matrix or vector of ones and zeros that represents the neighborhood values	Boolean	No
Output	Vector or matrix of intensity values that represents the dilated image	Same as I port	No

The output signal has the same data type as the input to the I port.

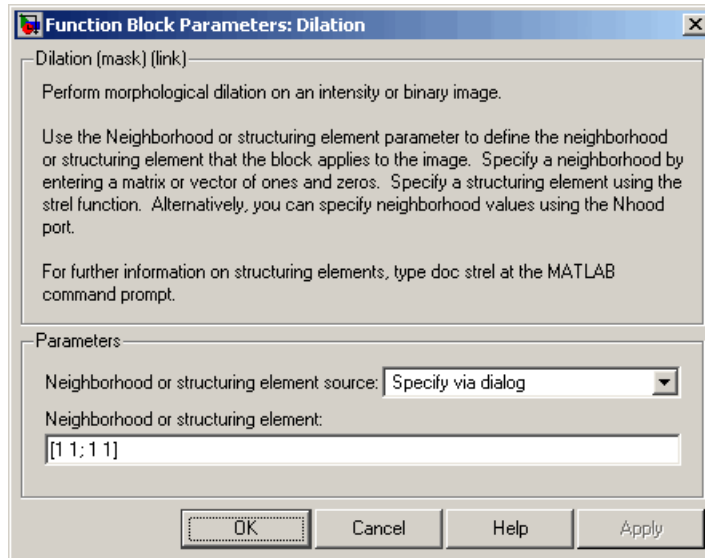
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select **Specify via dialog**, the **Neighborhood or structuring element** parameter appears in the dialog box. If you select **Input port**, the Nhood port appears on the block. Use this port to enter your neighborhood values as a matrix or vector of 1s and 0s. You can only specify a structuring element using the dialog box.

Use the **Neighborhood or structuring element** parameter to define the neighborhood or structuring element that the block applies to the image. Specify a neighborhood by entering a matrix or vector of 1s and 0s. Specify a structuring element with the `strel` function from the Image Processing Toolbox. If the structuring element is decomposable into smaller elements, the block executes at higher speeds due to the

use of a more efficient algorithm. If you enter an array of STREL objects, the block applies each object to the entire matrix in turn.

Dialog Box

The Dilation dialog box appears as shown in the following figure.



Neighborhood or structuring element source

Specify how to enter your neighborhood or structuring element values. Select **Specify via dialog** to enter the values in the dialog box. Select **Input port** to use the Nhood port to specify the neighborhood values. You can only specify a structuring element using the dialog box.

Neighborhood or structuring element

If you are specifying a neighborhood, this parameter must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the `strel` function from the Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or structuring element source** parameter, you select **Specify via dialog**.

Dilation

References

[1] Soille, Pierre. *Morphological Image Analysis. 2nd ed.* New York: Springer, 2003.

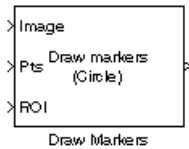
See Also

Bottom-hat	Video and Image Processing Blockset software
Closing	Video and Image Processing Blockset software
Erosion	Video and Image Processing Blockset software
Label	Video and Image Processing Blockset software
Opening	Video and Image Processing Blockset software
Top-hat	Video and Image Processing Blockset software
<code>imdilate</code>	Image Processing Toolbox software
<code>strel</code>	Image Processing Toolbox software

Purpose Draw markers by embedding predefined shapes on output image

Library Text & Graphics

Description The Draw Markers block can draw multiple circles, x-marks, plus signs, stars, or squares on images by overwriting pixel values. Overwriting the pixel values embeds the shapes.



This block uses Bresenham's circle drawing algorithm to draw circles and Bresenham's line drawing algorithm to draw all other markers.

Port	Input/Output	Supported Data Types	Complex Values Supported
Image	M -by- N matrix of intensity values or an M -by- N -by- P color values where P is the number of color planes	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
R, G, B	Scalar, vector, or matrix that represents one plane of the input RGB video stream. Inputs to the R, G, and B ports must have the same dimensions and data type.	Same as Image port	No

Draw Markers

Port	Input/Output	Supported Data Types	Complex Values Supported
Pts	<p>2-by-N matrix of row and column pairs,</p> $\begin{bmatrix} r_1 & r_2 & \cdots & r_N \\ c_1 & c_2 & \cdots & c_N \end{bmatrix}$ <p>where N is the total number of markers and each row and column pair defines the center of a marker.</p>	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer <p>If the input to the Image port is an integer, fixed point, or boolean data type, the input to the Pts port must also be an integer data type.</p>	No
ROI	<p>Four-element vector of integers that define a rectangular area in which to draw the markers. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the area. The second two elements represent the height and width of the area.</p>	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
Clr	<p>P-element vector or P-by-N matrix where P is the number of color planes</p>	Same as Image port	No
Output	<p>Scalar, vector, or matrix of pixel values that contain the marker(s)</p>	Same as Image port	No

The output signal is the same size and data type as the inputs to the Image, R, G, and B ports.

Use the **Marker shape** parameter to specify one of the following types of markers:

- Circle
- X-mark
- Plus
- Star
- Square

Use the **Marker size** parameter to define the size of the marker, in pixels. Enter a scalar value, M , that defines a $(2M+1)$ -by- $(2M+1)$ pixel square into which the marker fits. M must be greater than or equal to 1.

If, for the **Marker shape** parameter, you select:

- Circle, X-mark, or Star

and you then select the,

- **Use antialiasing** check box

the block performs a smoothing algorithm. The Draw Markers block uses an algorithm similar to the `poly2mask` function to determine which subpixels to draw.

Use the **Draw markers in** parameter to define one of the following types of areas in which to draw the markers.

- Entire image, enables you to draw markers in the entire image.
- Specify region of interest via port, the ROI port appears on the block. Enter a four-element vector of integer values, `[r c height width]`, where `r` and `c` are the row and column coordinates of the upper-left corner of the area, and `height` and `width` represent

Draw Markers

the height (in rows) and width (in columns) of the area. If you specify values that are outside the image, the block clips the values to the image boundaries.

Use the **Image signal** parameter to specify one of the following ways to input and output a color video signal.

- One multidimensional signal, the block accepts an *M-by-N-by-P* color video signal, where *P* is the number of color planes, at one port.
- Separate color signals, additional ports appear on the block. Each port accepts one *M-by-N* plane of an RGB video stream.

Selecting Marker Fill and Border Colors

You can set the marker fill or border color via the input port or via the input dialog. Use the color input or color parameter to determine the appearance of the rectangle(s), line(s), polygon(s), or circle(s).

- “Fill Color” on page 2-322
- “Border Color” on page 2-323
- “Color Values” on page 2-323
- “Opacity Factor” on page 2-323

Fill Color

If you select the **Filled** check box, the **Fill color source**, **Fill color** and **Opacity factor (between 0 and 1)** parameters appear in the dialog box. Use the **Fill color source** parameter to specify either **Input port** or **Specify via dialog** for the color source. If **Specify via dialog** is selected, you can specify either **Black**, **White**, or **User-specified value** for the **Fill color** parameter for the shading inside the shape. The **Color value(s)** parameter is applicable when the **User-specified value** is selected. Use the **Opacity factor (between 0 and 1)** parameter to specify the opacity of the shading inside the shape, where 0 is transparent and 1 is opaque.

Border Color

If the **Filled** check box is not selected, the **Border color source**, and **Border color** parameters are available. Use the **Border color source** parameter to specify either **Input port** or **Specify via dialog** for the color source. If **Specify via dialog** is selected, you can specify either **Black**, **White**, or **User-specified** value for the **Border color** parameter. If the color is user specified, the **Color value(s)** parameter is used to enter the color.

Color Values

The following table describes what to enter for the **Color Value(s)** parameter based on the block input and the number of markers you are drawing. This parameter is applicable when **User-specified** value is selected for the border color source.

Block Input	Color Value(s) for Drawing One Marker or Multiple Markers with the Same Color	Color Value(s) for Drawing Multiple Markers with Unique Color
Intensity image	Scalar intensity value	R -element vector where R is the number of markers
Color image	P -element vector where P is the number of color planes	P -by- R matrix where P is the number of color planes and R is the number of markers

For each value in the parameter, enter a number between the minimum and maximum values that can be represented by the data type of the input image. If you enter a value outside this range, the block produces an error message.

Opacity Factor

The following table describes what to enter for the **Opacity factor(s) (between 0 and 1)** parameter based on the block input and the number

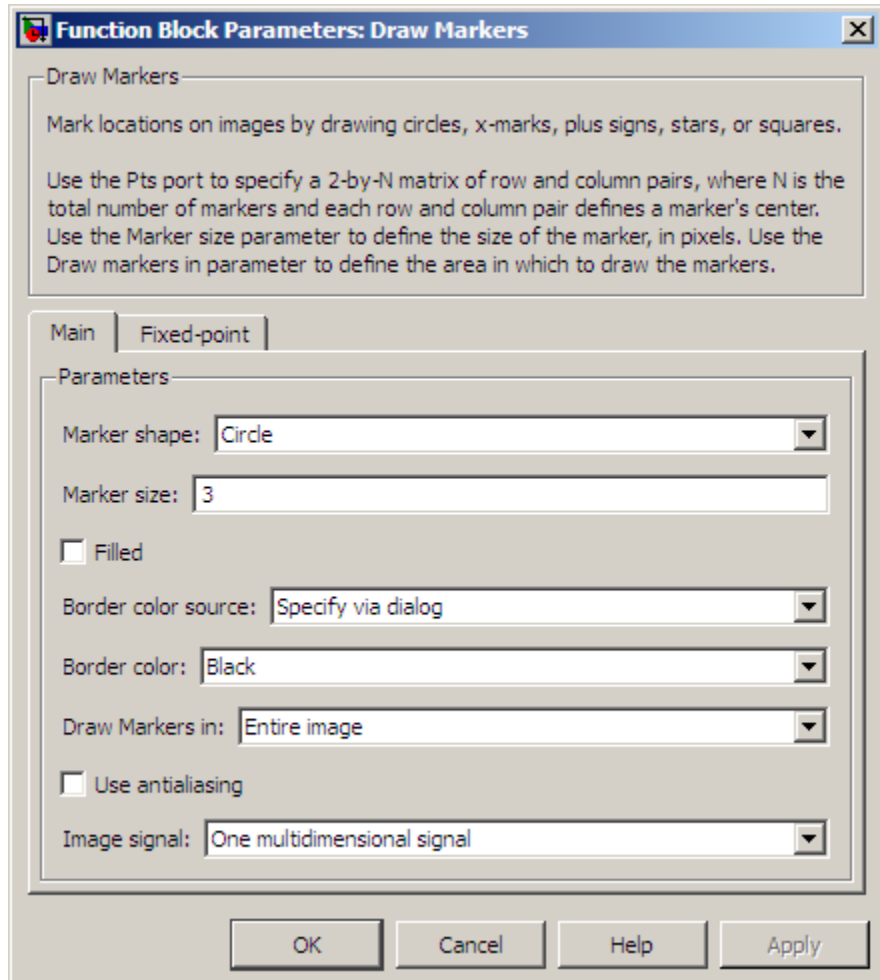
Draw Markers

of markers you are drawing. This parameter is applicable when the Filled check box is selected.

Opacity Factor value for Drawing One Marker or Multiple Markers with the Same Color	Opacity Factor value for Drawing Multiple Marker with Unique Color
Scalar intensity value	R -element vector where R is the number of markers

Dialog Box

The Draw Markers dialog box appears as shown in the following figure.



Marker shape

Specify the type of marker(s) to draw. Your choices are Circle, X-mark, Plus, Star, or Square.

Draw Markers

Marker size

Enter a scalar value that represents the size of the marker, in pixels.

Filled

Select this check box to fill the marker with an intensity value or a color. This parameter is visible if, for the **Marker shape** parameter, you choose **Circle** or **Square**.

Fill color source

Specify source for fill color value to either **Specify via dialog** or **Input port**. This parameter is visible if you select the **Filled** check box.

Fill color

If you select **Black**, the marker is black. If you select **White**, the marker is white. If you select **User-specified value**, the **Color value(s)** parameter appears in the dialog box. This parameter is visible if you select the **Filled** check box.

Border color source

Specify source for the border color value to either **Specify via dialog** or **Input port**. Border color options are visible when the fill shapes options are not selected. This parameter is visible if you select the **Filled** check box.

Border color

Specify the appearance of the shape's border. If you select **Black**, the border is black. If you select **White**, the border is white. If you select **User-specified value**, the **Color value(s)** parameter appears in the dialog box. This parameter is visible if you clear the **Fill shapes** check box.

Color value(s)

Specify an intensity or color value for the marker's border or fill. This parameter is visible if, for the **Border color** or **Fill color** parameter, you select **User-specified value**. Tunable.

Opacity factor (between 0 and 1)

Specify the opacity of the shading inside the marker, where 0 is transparent and 1 is opaque. This parameter is visible if you select the **Filled** check box. This parameter is tunable.

Draw markers in

Define the area in which to draw the markers. If you select **Entire image**, you can draw markers in the entire image. If you select **Specify region of interest via port**, the ROI port appears on the block. Enter a four-element vector, [r c height width], where r and c are the row and column coordinates of the upper-left corner of the area, and height and width represent the height (in rows) and width (in columns) of the area.

Use antialiasing

Perform a smoothing algorithm on the marker. This parameter is visible if, for the **Marker shape** parameter, you select **Circle**, **X-mark**, or **Star**.

Image signal

Specify how to input and output a color video signal. If you select **One multidimensional signal**, the block accepts an *M-by-N-by-P* color video signal, where *P* is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port accepts one *M-by-N* plane of an RGB video stream.

See Also

Draw Shapes

Video and Image Processing
Blockset software

Insert Text

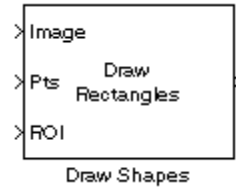
Video and Image Processing
Blockset software

Draw Shape (Obsolete)

Purpose Draw rectangle around region of interest (ROI)

Library vipobslib

Description



Note The Draw Shape block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset blocks. Use the replacement block Draw Shapes.

The Draw Shape block draws a rectangle around a user-defined ROI by overwriting pixel values. As a result, the rectangle is embedded on the output image.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• Boolean	No
ROI	Four-element vector of integers. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the ROI. The second two elements represent the height and width of the ROI.	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No

Draw Shape (Obsolete)

Port	Input/Output	Supported Data Types	Complex Values Supported
Output	Scalar, vector, or matrix of pixel values that contains the region of interest	Same as I port	No

The output signal is the same size and data type as the input to the I port.

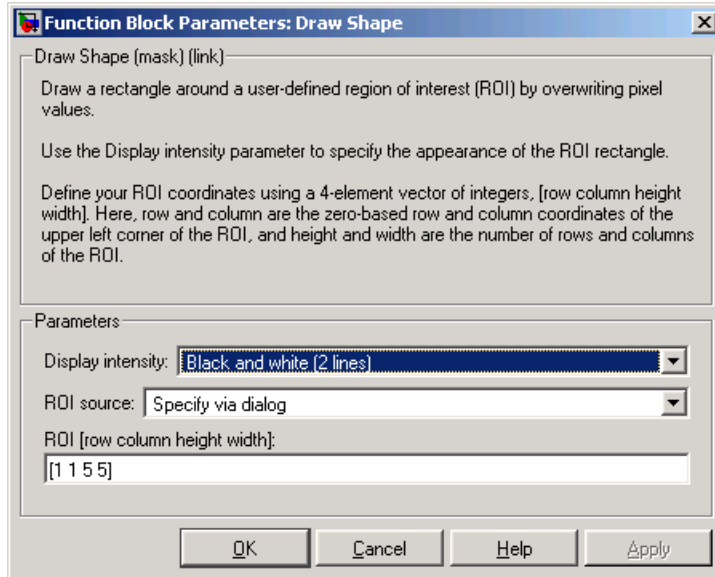
Use the **Display intensity** parameter to determine the appearance of the ROI rectangle. If you select **Black** or **White**, the rectangle is black or white, respectively. If you select **Black and white (2 lines)**, the rectangle is created by a black line on the outside and a white line on the inside. If you select **User-specified intensity**, the **Intensity value (0 to 1)** parameter appears in the dialog box. Enter a scalar intensity value from 0 to 1, where 0 corresponds to black and 1 corresponds to white.

Use the **ROI source** parameter to determine how to enter your ROI coordinates. If you select **Specify via dialog**, the **ROI [row column height width]** parameter appears on the dialog box. Enter a four-element vector of integers. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the ROI. The second two elements represent the height and width of the ROI. If you select **Input port**, the ROI port appears on the dialog box. The input to this port must be a four-element of integers as previously defined.

Draw Shape (Obsolete)

Dialog Box

The Draw Shape dialog box appears as shown in the following figure.



Display intensity

Specify the appearance of the ROI rectangle. If you select **Black** or **White**, the rectangle is black or white. If you select **Black and white (2 lines)**, the rectangle is created by a black line on the outside and a white line on the inside. If you select **User-specified intensity**, the **Intensity value (0 to 1)** parameter appears in the dialog box.

Intensity value

Enter a scalar intensity value from 0 to 1, where 0 corresponds to black and 1 corresponds to white. This parameter is visible if, for the **Display intensity** parameter, you select **User-specified intensity**. Tunable.

ROI source

Specify how to enter your ROI coordinates. If you select **Specify via dialog**, the **ROI [row column height width]** parameter

appears on the dialog box. If you select `Input port`, the ROI port appears on the dialog box. The input to this port must be a four-element vector of integers. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the ROI. The second two elements represent the height and width of the ROI.

ROI [row column height width]

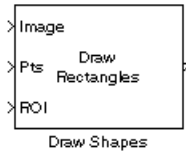
Enter a four-element vector of integers. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the ROI. The second two elements represent the height and width of the ROI. Tunable.

Draw Shapes

Purpose Draw rectangles, lines, polygons, or circles on images

Library Text & Graphics

Description The Draw Shapes block draws multiple rectangles, lines, polygons, or circles on images by overwriting pixel values. As a result, the shapes are embedded on the output image.



This block uses Bresenham's line drawing algorithm to draw lines, polygons, and rectangles. It uses Bresenham's circle drawing algorithm to draw circles.

Port	Input/Output	Supported Data Types	Complex Values Supported
Image	M -by- N matrix of intensity values or an M -by- N -by- P color values where P is the number of color planes	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
R, G, B	Scalar, vector, or matrix that represents one plane of the input RGB video stream. Inputs to the R, G, and B ports must have the same dimensions and data type.	Same as Image port	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Pts	<p>Use integer values to define zero-based shape coordinates. If you enter noninteger values, the block rounds them to the nearest integer.</p> <p>For more information about how to specify shape coordinates for different shapes, see “Defining Shapes to Draw” on page 2-337.</p>	<ul style="list-style-type: none"> • Double-precision floating point (only supported if the input to the I or R, G, and B ports is floating point) • Single-precision floating point (only supported if the input to the I or R, G, and B ports is floating point) • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
ROI	<p>4-element vector of integers that defines a rectangular area in which to draw the shapes. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the area. The second two elements represent the height and width of the area.</p>	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
Clr	<p>P-element vector or P-by-N matrix, where P is the number of color planes</p>	Same as Image port	No
Output	<p>Scalar, vector, or matrix of pixel values that contain the shape(s)</p>	Same as Image port	No

The output signal is the same size and data type as the inputs to the Image, R, G, and B ports.

Draw Shapes

Use the **Shape** parameter to specify one of the following types type of shape(s) to draw.

- Rectangles
- Lines
- Polygons
- Circles

Use the **Draw shapes in** parameter to define one of the following types of area in which to draw the shapes.

- Entire image, enables you to draw shapes in the entire image.
- Specify region of interest via port, the ROI port appears on the block. Enter a four-element vector of integer values, [r c height width], where r and c are the row and column coordinates of the upper-left corner of the area, and height and width represent the height (in rows) and width (in columns) of the area.

Note If you specify values that are outside the image, the block sets the values to the image boundaries.

If, for the **Shape** parameter, you select:

- Lines, Polygons, or Circles

and you then select the,

- Use **antialiasing** check box,

the block performs a smoothing algorithm. The Draw Shapes block uses an algorithm similar to the poly2mask function to determine which subpixels to draw.

Use the **Image signal** parameter to specify one of the following ways to input and output a color video signal.

- One multidimensional signal, the block accepts an M -by- N -by- P color video signal, where P is the number of color planes, at one port.
- Separate color signals, additional ports appear on the block. Each port accepts one M -by- N plane of an RGB video stream.

Selecting Shape Fill and Border Colors

You can set the shape fill or border color via the input port or via the input dialog. Use the color input or color parameter to determine the appearance of the rectangle(s), line(s), polygon(s), or circle(s).

- Fill Color on page 335
- Border Color on page 336
- Color Values on page 336
-

Fill Color

If you select the **Fill shapes** check box, the **Fill color source**, **Fill color** and **Opacity factor (between 0 and 1)** parameters appear in the dialog box. Use the **Fill color source** parameter to specify either Input port or Specify via dialog for the color source. If Specify via dialog is selected, you can specify either Black, White, or User-specified value for the **Fill color** parameter for the shading inside the shape. The **Color value(s)** parameter is applicable when the User-specified value is selected. Use the **Opacity factor (between 0 and 1)** parameter to specify the opacity of the shading inside the shape, where 0 is transparent and 1 is opaque.

Draw Shapes

Note If you are generating code and you select the **Fill shapes** check box, the word length of the block input(s) cannot exceed 16 bits.

Border Color

If the **Fill shapes** check box is not selected, the **Border color source**, and **Border color** parameters are available. Use the **Border color source** parameter to specify either Input port or Specify via dialog for the color source. If Specify via dialog is selected, you can specify either Black, White, or User-specified value for the **Border color** parameter. If the color is user specified, the **Color value(s)** parameter is used to enter the color.

Color Values

The following table describes what to enter for the **Color Value(s)** parameter based on the block input and the number of shapes you are drawing.

Block Input	Color Value(s) for Drawing One Shape or Multiple Shapes with the Same Color	Color Value(s) for Drawing Multiple Shapes with Unique Color
Intensity image	Scalar intensity value	R -element vector where R is the number of shapes
Color image	P -element vector where P is the number of color planes	P -by- R matrix where P is the number of color planes and R is the number of shapes

For each value in the **Color Value(s)** parameter, enter a number between the minimum and maximum values that can be represented by the data type of the input image. If you enter a value outside this range, the block produces an error message.

Opacity Factor

The following table describes what to enter for the **Opacity factor(s)** (between 0 and 1) parameter based on the block input and the number of shapes you are drawing. This parameter is applicable when the Filled check box is selected.

Opacity Factor value for Drawing One Shape or Multiple Shapes with the Same Color	Opacity Factor value for Drawing Multiple Shapes with Unique Color
Scalar intensity value	R -element vector where R is the number of shapes

Defining Shapes to Draw


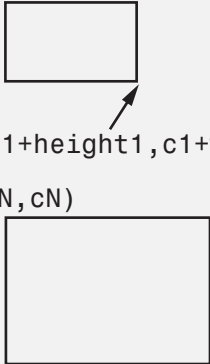
This section explains how to use the **Shape** parameter and the Pts port to draw the following shapes:

- Drawing Rectangles on page 337
- Drawing Lines and Polylines on page 338
- Drawing Polygons on page 341
- Drawing Circles on page 342

Drawing Rectangles

The Draw Shapes block lets you draw one or more rectangles. Set the **Shape** parameter to **Rectangles**, and then follow the instructions in the table to specify the input to the Pts port to obtain the desired number of rectangles.

Draw Shapes

Shape	Input to the Pts Port	Drawn Shape
Single Rectangle	<p>Four-element row or column vector [r c height width] where</p> <ul style="list-style-type: none"> r and c are the zero-based row and column coordinates of the upper-left corner of the rectangle. height and width are the height, in pixels, and width, in pixels, of the rectangle. Here, height and width must be greater than 0. 	<p>(r,c)</p>  <p>(r+height,c+width)</p>
N Rectangles	<p>4-by-N matrix</p> $\begin{bmatrix} r_1 & \cdots & r_N \\ c_1 & \cdots & c_N \\ height_1 & \cdots & height_N \\ width_1 & \cdots & width_N \end{bmatrix}$ <p>where each column of the matrix corresponds to a different rectangle and is of the same form as the vector for a single rectangle.</p>	<p>N=2:</p> <p>(r1,c1)</p>  <p>(r1+height1,c1+width1)</p> <p>(rN,cN)</p> <p>(rN+heightN,cN+widthN)</p>

For an example of how to use the Draw Shapes block to draw a rectangle, see “Tracking an Object Using Correlation”.

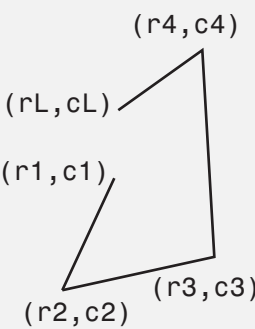
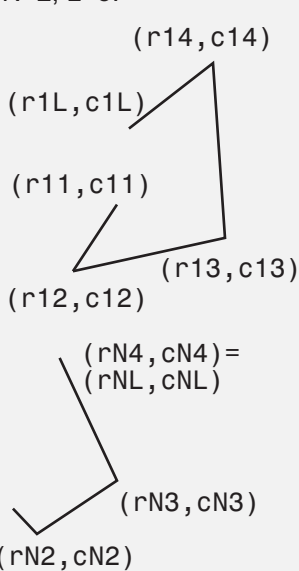
Drawing Lines and Polylines

The Draw Shapes block lets you draw either a single line, or one or more polylines, where each polyline is a series of connected line segments.

Set the **Shape** parameter to **Lines**, and then follow the instructions in the table to specify the input to the Pts port to obtain the desired shape.

Shape	Input to the Pts Port	Drawn Shape
Single Line	<p>Four-element row or column vector [r1 c1 r2 c2] where</p> <ul style="list-style-type: none"> r1 and c1 are the row and column coordinates of the beginning of the line. r2 and c2 are the row and column coordinates of the end of the line. 	<p>(r1, c1)</p> <p>(r2, c2)</p>
N Lines	<p>4-by-N matrix</p> $\begin{bmatrix} r_{11} & \cdots & r_{1N} \\ c_{11} & \cdots & c_{1N} \\ r_{21} & \cdots & r_{2N} \\ c_{21} & \cdots & c_{2N} \end{bmatrix}$ <p>where each column of the matrix corresponds to a different line and is of the same form as the vector for a single line.</p>	<p>N=2:</p> <p>(r11, c11)</p> <p>(r21, c21)</p> <p>(r1N, c1N)</p> <p>(r2N, c2N)</p>

Draw Shapes

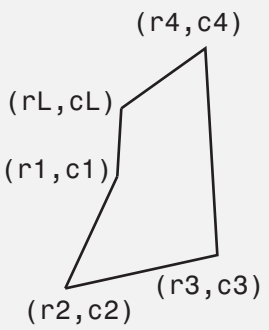
Shape	Input to the Pts Port	Drawn Shape
Single Polyline with L-1 Segments	<p>Vector of size $2L$ [r_1 c_1 r_2 c_2 ... r_L c_L] where</p> <ul style="list-style-type: none"> r_1 and c_1 are the row and column coordinates of the beginning of the first line segment. r_2 and c_2 are the row and column coordinates of the end of the first line segment and the beginning of the second line segment r_L and c_L are the row and column coordinates of the end of the $L-1^{\text{th}}$ line segment. <p>The block produces an error message if the number of rows is less than two or is not a multiple of two.</p>	<p>L=5:</p> 
N Polylines with the largest number of line segments in any line being L-1	<p>$2L$-by-N matrix</p> $\begin{bmatrix} r_{11} & \cdots & r_{N1} \\ c_{11} & \cdots & c_{N1} \\ r_{12} & \cdots & r_{N2} \\ c_{12} & \cdots & c_{N2} \\ \vdots & \ddots & \vdots \\ r_{1L} & \cdots & r_{NL} \\ c_{1L} & \cdots & c_{NL} \end{bmatrix}$ <p>where each column of the matrix corresponds to a different polyline and is of the same form as the vector for a single polyline. If some polylines are shorter than others, repeat the ending coordinates to fill the polyline matrix.</p> <p>The block produces an error message if the number of rows is less than two or is not a multiple of two.</p>	<p>N=2, L=5:</p> 

If you select the **Use antialiasing** check box, the block applies an edge smoothing algorithm.

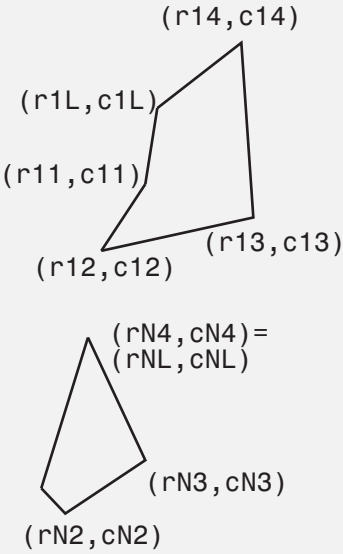
For examples of how to use the Draw Shapes block to draw a line, see “Finding Lines in Images” and “Measuring an Angle Between Lines”.

Drawing Polygons

The Draw Shapes block lets you draw one or more polygons. Set the **Shape** parameter to Polygons, and then follow the instructions in the table to specify the input to the Pts port to obtain the desired number of polygons.

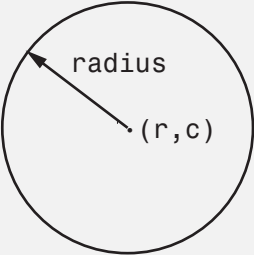
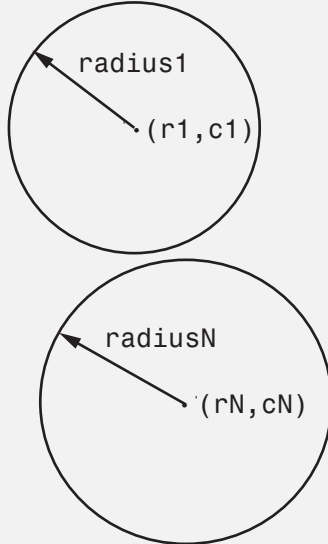
Shape	Input to the Pts Port	Drawn Shape
Single Polygon with L line segments	<p>Row or column vector of size 2L [r1 c1 r2 c2 ... rL cL] where</p> <ul style="list-style-type: none"> r1 and c1 are the row and column coordinates of the beginning of the first line segment. r2 and c2 are the row and column coordinates of the end of the first line segment and the beginning of the second line segment rL and cL are the row and column coordinates of the end of the L-1th line segment and the beginning of the Lth line segment. <p>The block connects [r1 c1] to [rL cL] to complete the polygon. The block produces an error if the number of rows is negative or not a multiple of two.</p>	<p>L=5:</p> 

Draw Shapes

Shape	Input to the Pts Port	Drawn Shape
<p>N Polygons with the largest number of line segments in any line being L</p>	<p>2L-by-N matrix</p> $\begin{bmatrix} r_{11} & \cdots & r_{N1} \\ c_{11} & \cdots & c_{N1} \\ r_{12} & \cdots & r_{N2} \\ c_{12} & \cdots & c_{N2} \\ \vdots & \ddots & \vdots \\ r_{1L} & \cdots & r_{NL} \\ c_{1L} & \cdots & c_{L} \end{bmatrix}$ <p>where each column of the matrix corresponds to a different polygon and is of the same form as the vector for a single polygon. If some polygons are shorter than others, repeat the ending coordinates to fill the polygon matrix.</p> <p>The block produces an error message if the number of rows is less than two or is not a multiple of two.</p>	<p>N=2, L=5:</p> 

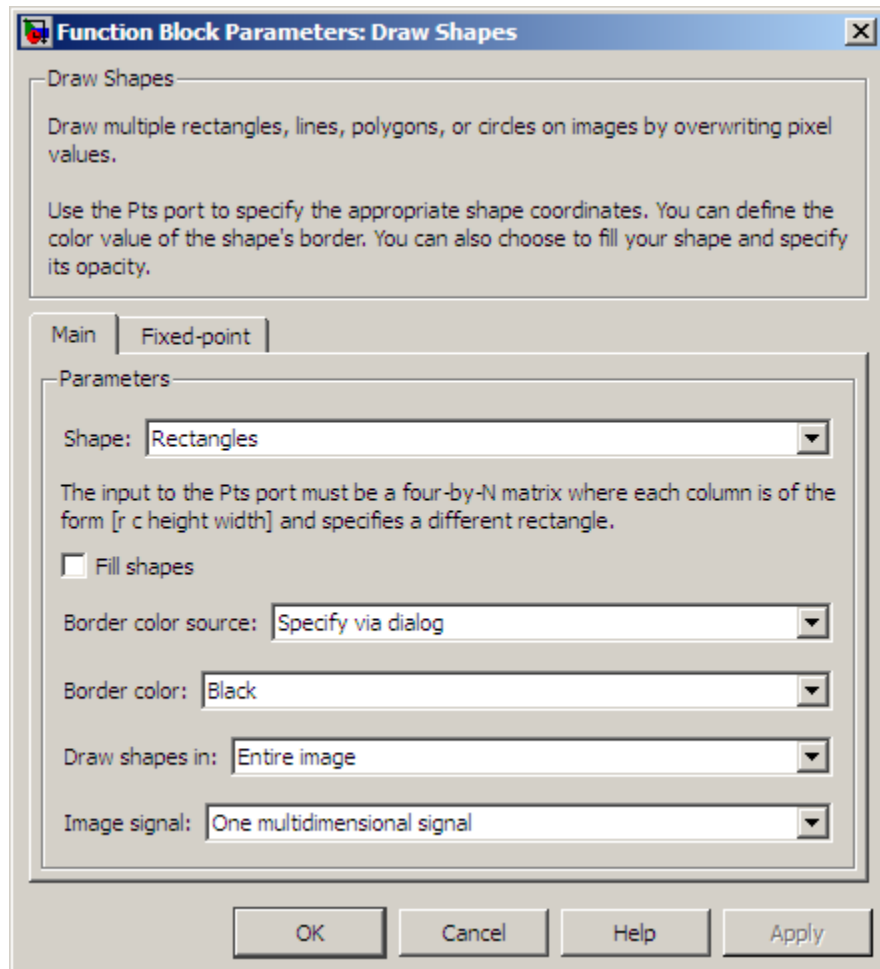
Drawing Circles

The Draw Shapes block lets you draw one or more circles. Set the **Shape** parameter to **Circles**, and then follow the instructions in the table to specify the input to the Pts port to obtain the desired number of circles.

Shape	Input to the Pts Port	Drawn Shape
Single Circle	<p>Three-element row or column vector $[r \ c \ radius]$ where</p> <ul style="list-style-type: none"> r and c are the row and column coordinates of the center of the circle. $radius$ is the radius of the circle, which must be greater than 0. 	 <p>A circle is drawn with its center at coordinates (r, c). An arrow points from the center to the circumference, labeled "radius".</p>
N Circles	<p>3-by-N matrix</p> $\begin{bmatrix} r_1 & \cdots & r_N \\ c_1 & \cdots & c_N \\ radius_1 & \cdots & radius_N \end{bmatrix}$ <p>where each column of the matrix corresponds to a different circle and is of the same form as the vector for a single circle.</p>	<p>$N=2$:</p>  <p>Two circles are drawn, one above the other. The top circle has center (r_1, c_1) and radius $radius_1$. The bottom circle has center (r_N, c_N) and radius $radius_N$. Each circle has an arrow pointing from its center to its circumference, labeled with its respective radius.</p>

Draw Shapes

Dialog Box



Shape

Specify the type of shape(s) to draw. Your choices are Rectangles, Lines, Polygons, or Circles.

Fill shapes

Fill the shape with an intensity value or a color.

Fill color source

Specify source for fill color value to either `Specify via dialog` or `Input port`. This parameter is visible if you select the **Fill shapes** check box.

Fill color

If you select `Black`, the border is black. If you select `White`, the border is white. If you select `User-specified value`, the **Color value(s)** parameter appears in the dialog box. This parameter is visible if you select the **Fill shapes** check box.

Border color source

Specify source for the border color value to either `Specify via dialog` or `Input port`. Border color options are visible when the fill shapes options are not selected. This parameter is visible if you select the **Fill shapes** check box.

Border color

Specify the appearance of the shape's border. If you select `Black`, the border is black. If you select `White`, the border is white. If you select `User-specified value`, the **Color value(s)** parameter appears in the dialog box. This parameter is visible if you clear the **Fill shapes** check box.

Color value(s)

Specify an intensity or color value for the shape's border or fill. This parameter is visible if, for the **Border color** or **Fill color** parameter, you select `User-specified value`. This parameter is tunable.

Opacity factor (between 0 and 1)

Specify the opacity of the shading inside the shape, where 0 is transparent and 1 is opaque. This parameter is visible if you select the **Fill shapes** check box.

Draw shapes in

Define the area in which to draw the shapes. If you select `Entire image`, you can draw shapes in the entire image. If you select `Specify region of interest via port`, the ROI port appears on the block. Enter a four-element vector, `[r c height`

Draw Shapes

width], where *r* and *c* are the row and column coordinates of the upper-left corner of the area, and *height* and *width* represent the height (in rows) and width (in columns) of the area.

Use antialiasing

Perform a smoothing algorithm on the line, polygon, or circle.

This parameter is visible if, for the **Shape** parameter, you select **Lines**, **Polygons**, or **Circles**.

Image signal

Specify how to input and output a color video signal. If you select **One multidimensional signal**, the block accepts an *M-by-N-by-P* color video signal, where *P* is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port accepts one *M-by-N* plane of an RGB video stream.

See Also

Draw Markers

Video and Image Processing Blockset software

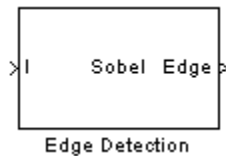
Insert Text

Video and Image Processing Blockset software

Purpose Find edges of objects in images using Sobel, Prewitt, Roberts, or Canny method

Library Analysis & Enhancement
vipanalysis

Description



If, for the **Method** parameter, you select Sobel, Prewitt, or Roberts, the Edge Detection block finds the edges in an input image by approximating the gradient magnitude of the image. The block convolves the input matrix with the Sobel, Prewitt, or Roberts kernel. The block outputs two gradient components of the image, which are the result of this convolution operation. Alternatively, the block can perform a thresholding operation on the gradient magnitudes and output a binary image, which is a matrix of Boolean values. If a pixel value is 1, it is an edge.

If, for the **Method** parameter, you select Canny, the Edge Detection block finds edges by looking for the local maxima of the gradient of the input image. It calculates the gradient using the derivative of the Gaussian filter. The Canny method uses two thresholds to detect strong and weak edges. It includes the weak edges in the output only if they are connected to strong edges. As a result, the method is more robust to noise, and more likely to detect true weak edges.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point (not supported for the Canny method) • 8-, 16-, 32-bit signed integer (not supported for the Canny method) 	No

Edge Detection

Port	Input/Output	Supported Data Types	Complex Values Supported
		<ul style="list-style-type: none">8-, 16-, 32-bit unsigned integer (not supported for the Canny method)	
Th	Matrix of intensity values	Same as I port	No
Edge	Matrix that represents a binary image	Boolean	No
Gv	Matrix of gradient values in the vertical direction	Same as I port	No
Gh	Matrix of gradient values in the horizontal direction	Same as I port	No
G45	Matrix of gradient values	Same as I port	No
G135	Matrix of gradient values	Same as I port	No

The output of the Gv, Gh, G45, and G135 ports is the same data type as the input to the I port. The input to the Th port must be the same data type as the input to the I port.

Use the **Method** parameter to specify which algorithm to use to find edges. You can select `Sobel`, `Prewitt`, `Roberts`, or `Canny` to find edges using the Sobel, Prewitt, Roberts, or Canny method.

Sobel, Prewitt, and Roberts Methods

Use the **Output type** parameter to select the format of the output. If you select `Binary image`, the block outputs a Boolean matrix at the Edge port. The nonzero elements of this matrix correspond to the edge pixels and the zero elements correspond to the background pixels. If you select `Gradient components` and, for the **Method** parameter, you

select **Sobel** or **Prewitt**, the block outputs the gradient components that correspond to the horizontal and vertical edge responses at the **Gh** and **Gv** ports, respectively. If you select **Gradient components** and, for the **Method** parameter, you select **Roberts**, the block outputs the gradient components that correspond to the 45 and 135 degree edge responses at the **G45** and **G135** ports, respectively. If you select **Binary image and gradient components**, the block outputs both the binary image and the gradient components of the image.

Select the **User-defined threshold** check box to define a threshold values or values. If you clear this check box, the block computes the threshold for you.

Use the **Threshold source** parameter to specify how to enter your threshold value. If you select **Specify via dialog**, the **Threshold** parameter appears in the dialog box. Enter a threshold value that is within the range of your input data. If you choose **Input port**, use input port **Th** to specify a threshold value. This value must have the same data type as the input data. Gradient magnitudes above the threshold value correspond to edges.

The Edge Detection block computes the automatic threshold using the mean of the gradient magnitude squared image. However, you can adjust this threshold using the **Threshold scale factor (used to automatically calculate threshold value)** parameter. The block multiplies the value you enter with the automatic threshold value to determine a new threshold value.

Select the **Edge thinning** check box to reduce the thickness of the edges in your output image. This option requires additional processing time and memory resources.

Note This block is most efficient in terms of memory usage and processing time when you clear the **Edge thinning** check box and use the **Threshold** parameter to specify a threshold value.

Edge Detection

Canny Method

Select the **User-defined threshold** check box to define the low and high threshold values. If you clear this check box, the block computes the threshold values for you.

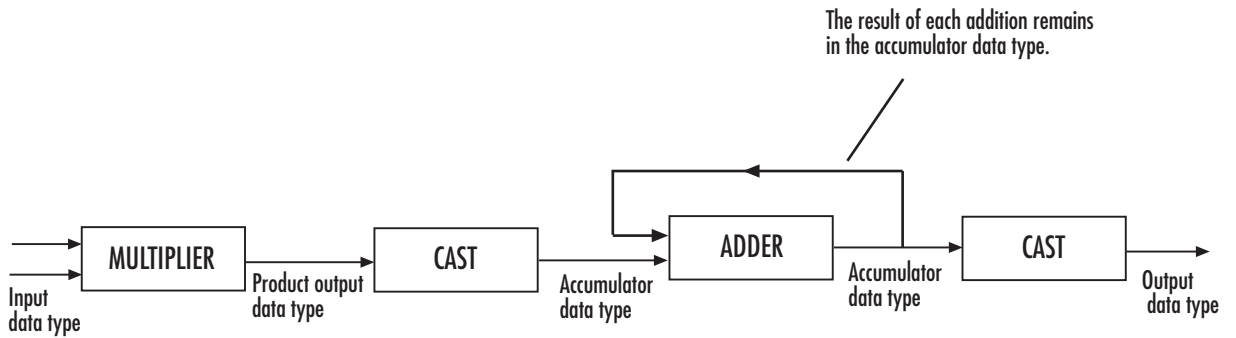
Use the **Threshold source** parameter to specify how to enter your threshold values. If you select **Specify via dialog**, the **Threshold [low high]** parameter appears in the dialog box. Enter the threshold values. If a pixel's magnitude in the gradient image, which is formed by convolving the input image with the derivative of the Gaussian filter, exceeds the high threshold, then the pixel corresponds to a strong edge. Any pixel connected to a strong edge and having a magnitude greater than the low threshold corresponds to a weak edge. If, for the **Threshold source** parameter, you choose **Input port**, use input port **Th** to specify a two-element vector of threshold values. These values must have the same data type as the input data.

The Edge Detection block computes the automatic threshold values using an approximation of the number of weak and nonedge image pixels. Enter this approximation for the **Approximate percentage of weak edge and nonedge pixels (used to automatically calculate threshold values)** parameter.

Use the **Standard deviation of Gaussian filter** parameter to define the Gaussian filter whose derivative is convolved with the input image.

Fixed-Point Data Types

The following diagram shows the data types used in the Edge Detection block for fixed-point signals.

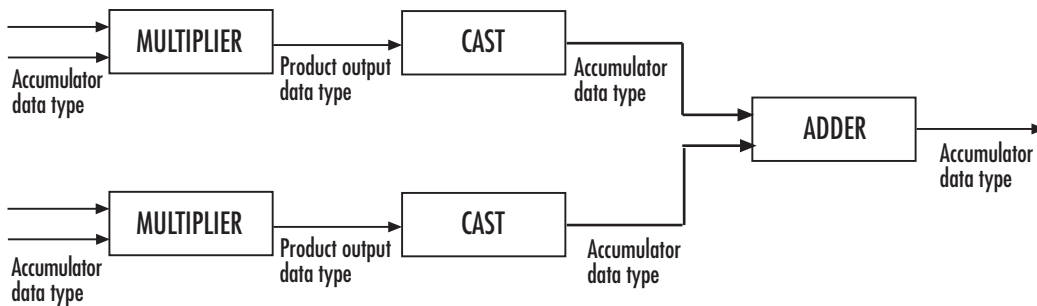


The block squares the threshold and compares it to the sum of the squared gradients to avoid using square roots.

Threshold:



Gradients:

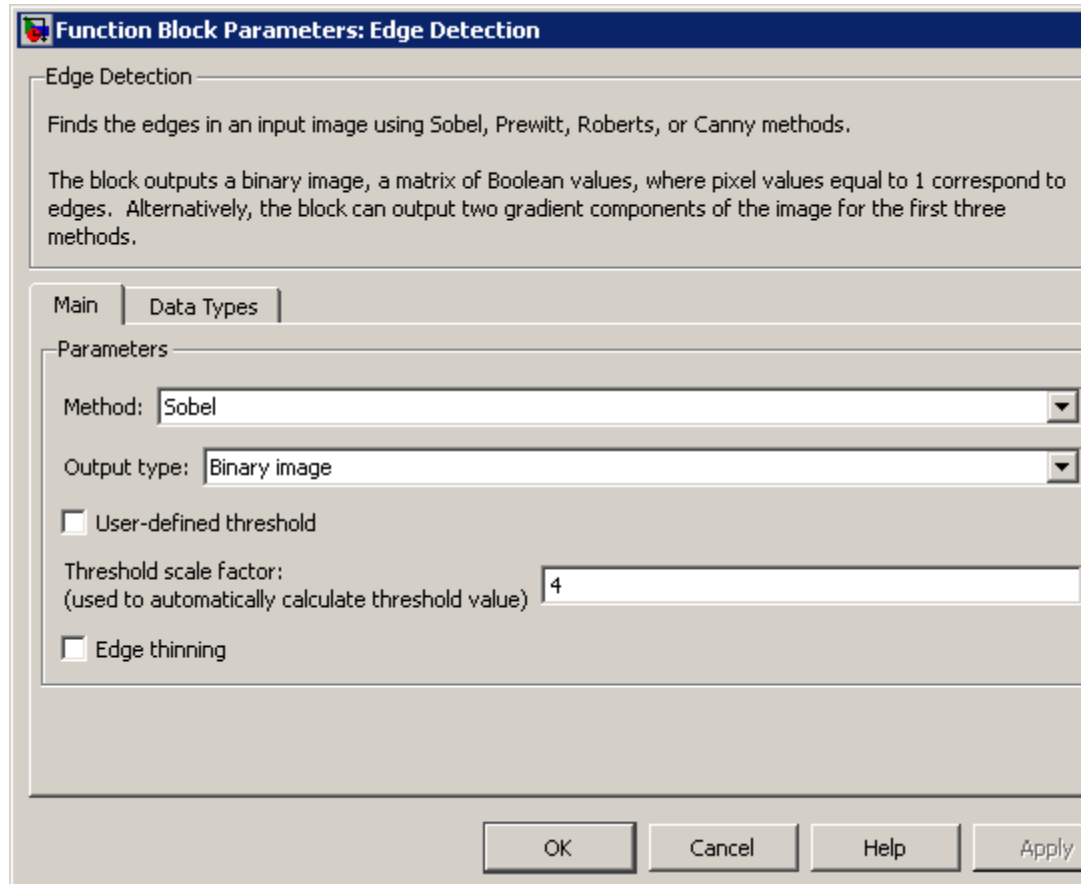


Edge Detection

You can set the product output and accumulator data types in the block mask as discussed in the next section.

Dialog Box

The **Main** pane of the Edge Detection dialog box appears as shown in the following figure.



Method

Select the method by which to perform edge detection. Your choices are Sobel, Prewitt, Roberts, or Canny.

Output type

Select the desired form of the output. If you select **Binary image**, the block outputs a matrix that is filled with ones, which correspond to edges, and zeros, which correspond to the background. If you select **Gradient components** and, for the **Method** parameter, you select Sobel or Prewitt, the block outputs the gradient components that correspond to the horizontal and vertical edge responses. If you select **Gradient components** and, for the **Method** parameter, you select Roberts, the block outputs the gradient components that correspond to the 45 and 135 degree edge responses. If you select **Binary image and gradient components**, the block outputs both the binary image and the gradient components of the image. This parameter is visible if, for the **Method** parameter, you select Sobel, Prewitt, or Roberts.

User-defined threshold

If you select this check box, you can enter a desired threshold value. If you clear this check box, the block computes the threshold for you. This parameter is visible if, for the **Method** parameter, you select Sobel, Prewitt, or Roberts, and, for the **Output type** parameter, you select **Binary image** or **Binary image and gradient components**. This parameter is also visible if, for the **Method** parameter, you select Canny.

Threshold source

If you select **Specify via dialog**, enter your threshold value in the dialog box. If you choose **Input port**, use the **Th** input port to specify a threshold value that is the same data type as the input data. This parameter is visible if you select the **User-defined threshold** check box.

Threshold

Enter a threshold value that is within the range of your input data. This parameter is visible if, for the **Method** parameter, you

Edge Detection

select Sobel, Prewitt, or Roberts, you select the **User-defined threshold** check box, and, for **Threshold source** parameter, you select Specify via dialog. .

Threshold [low high]

Enter the low and high threshold values that define the weak and strong edges. This parameter is visible if, for the **Method** parameter, you select Canny. Then you select the **User-defined threshold** check box, and, for **Threshold source** parameter, you select Specify via dialog. Tunable.

Threshold scale factor (used to automatically calculate threshold value)

Enter a multiplier that is used to adjust the calculation of the automatic threshold. This parameter is visible if, for the **Method** parameter, you select Sobel, Prewitt, or Roberts, and you clear the **User-defined threshold** check box. Tunable.

Edge thinning

Select this check box if you want the block to perform edge thinning. This option requires additional processing time and memory resources. This parameter is visible if, for the **Method** parameter, you select Sobel, Prewitt, or Roberts, and for the **Output type** parameter, you select Binary image or Binary image and gradient components.

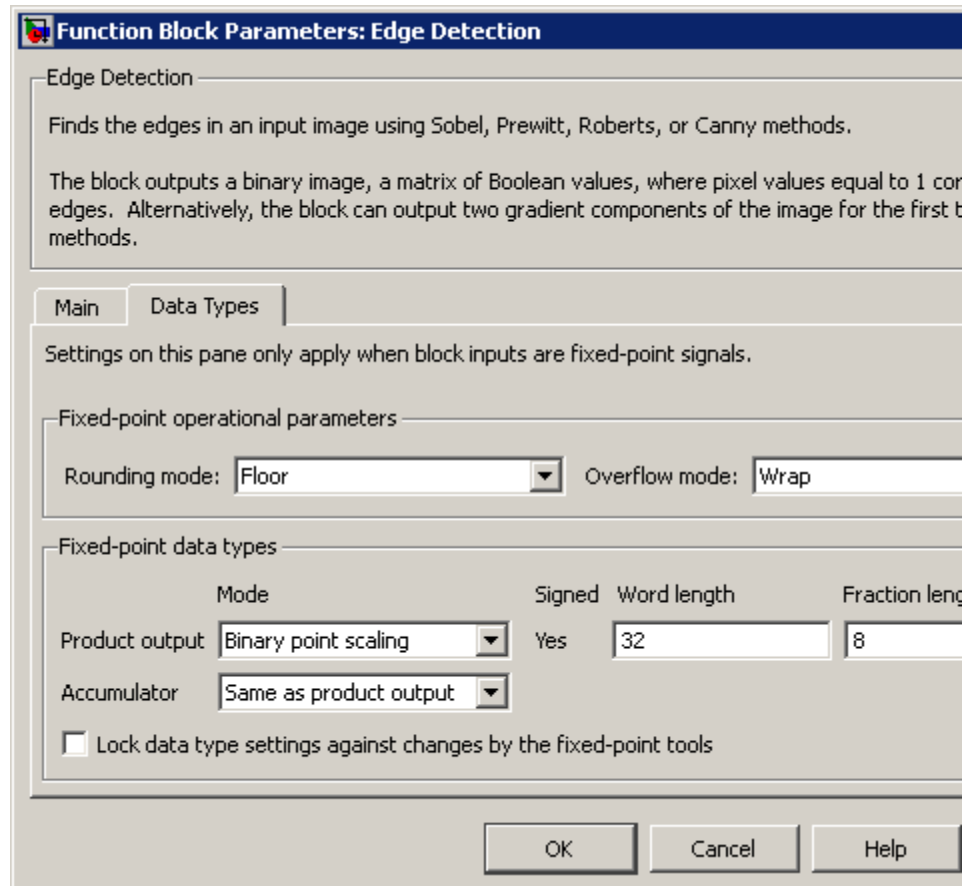
Approximate percentage of weak edge and nonedge pixels (used to automatically calculate threshold values)

Enter the approximate percentage of weak edge and nonedge image pixels. The block computes the automatic threshold values using this approximation. This parameter is visible if, for the **Method** parameter, you select Canny. Tunable.

Standard deviation of Gaussian filter

Enter the standard deviation of the Gaussian filter whose derivative is convolved with the input image. This parameter is visible if, for the **Method** parameter, you select Canny.

The **Data Types** pane of the Edge Detection dialog box appears as shown in the following figure.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Edge Detection

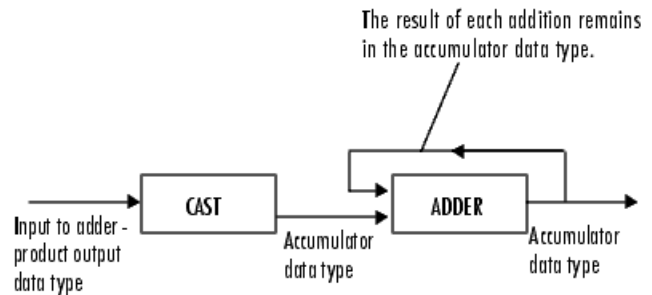
Product output



Here, the internal coefficients are the Sobel, Prewitt, or Roberts masks. As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths.

- When you select **Same as first input**, these characteristics match those of the first input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Accumulator



As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

- When you select **Same as product output**, these characteristics match those of the product output.
- When you select **Same as first input**, these characteristics match those of the first input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Gradients

Choose how to specify the word length and fraction length of the outputs of the Gv and Gh ports. This parameter is visible if, for the **Output type** parameter, you choose **Gradient components** or **Binary image and gradient components**:

Edge Detection

- When you select `Same as accumulator`, these characteristics match those of the accumulator.
- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as first input`, these characteristics match those of the first input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

References

[1] Gonzales, Rafael C. and Richard E. Woods. *Digital Image Processing, 2nd ed.* Englewood Cliffs, NJ: Prentice-Hall, 2002.

[2] Pratt, William K. *Digital Image Processing, 2nd ed.* New York: John Wiley & Sons, 1991.

See Also

`edge`

Image Processing Toolbox

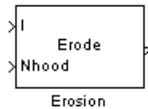
Purpose

Find local minima in binary or intensity images

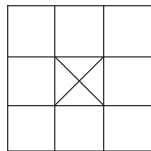
Library

Morphological Operations

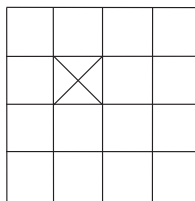
Description



The Erosion block slides the neighborhood or structuring element over an image, finds the local minima, and creates the output matrix from these minimum values. If the neighborhood or structuring element has a center element, the block places the minima there, as illustrated in the following figure.



If the neighborhood or structuring element does not have an exact center, the block has a bias toward the upper-left corner and places the minima there, as illustrated in the following figure.



This block uses flat structuring elements only.

Erosion

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• Boolean• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No
Nhood	Matrix or vector of 1s and 0s that represents the neighborhood values	Boolean	No
Output	Vector or matrix of intensity values that represents the eroded image	Same as I port	No

The output signal is the same data type as the input to the I port.

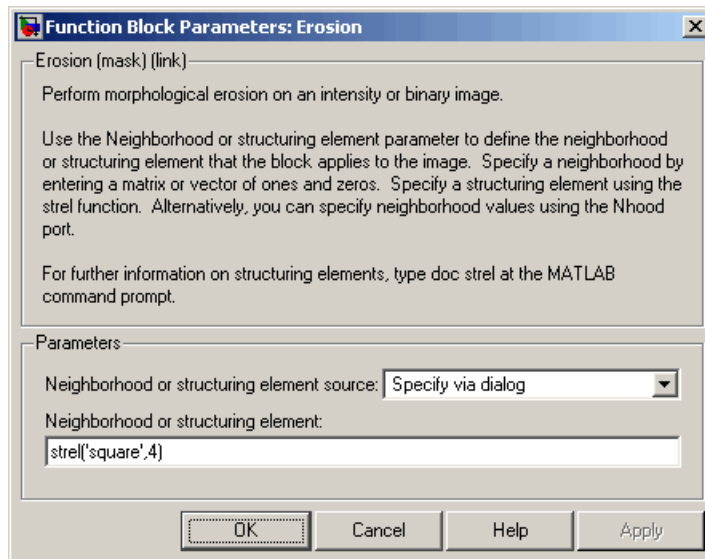
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select **Specify via dialog**, the **Neighborhood or structuring element** parameter appears in the dialog box. If you select **Input port**, the Nhood port appears on the block. Use this port to enter your neighborhood values as a matrix or vector of 1s and 0s. You can only specify a structuring element using the dialog box.

Use the **Neighborhood or structuring element** parameter to define the neighborhood or structuring element that the block applies to the image. Specify a neighborhood by entering a matrix or vector of 1s and 0s. Specify a structuring element with the `strel` function from the Image Processing Toolbox. If the structuring element is decomposable into smaller elements, the block executes at higher speeds due to the

use of a more efficient algorithm. If you enter an array of STREL objects, the block applies each object to the entire matrix in turn.

Dialog Box

The Erosion dialog box appears as shown in the following figure.



Neighborhood or structuring element source

Specify how to enter your neighborhood or structuring element values. Select **Specify via dialog** to enter the values in the dialog box. Select **Input port** to use the Nhood port to specify the neighborhood values. You can only specify a structuring element using the dialog box.

Neighborhood or structuring element

If you are specifying a neighborhood, this parameter must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the `strel` function from the Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or structuring element source** parameter, you select **Specify via dialog**.

References

[1] Soille, Pierre. *Morphological Image Analysis. 2nd ed.* New York: Springer, 2003.

See Also

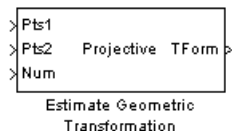
Bottom-hat	Video and Image Processing Blockset software
Closing	Video and Image Processing Blockset software
Dilation	Video and Image Processing Blockset software
Label	Video and Image Processing Blockset software
Opening	Video and Image Processing Blockset software
Top-hat	Video and Image Processing Blockset software
imerode	Image Processing Toolbox software
strel	Image Processing Toolbox software

Estimate Geometric Transformation

Purpose Estimate geometric transformation from matching point pairs

Library Geometric Transformations

Description



Use the Estimate Geometric Transformation block to find the transformation matrix which maps the greatest number of point pairs between two images. A *point pair* refers to a point in the input image and its related point on the image created using the transformation matrix. You can select to use the RANdom SAMple Consensus (RANSAC) or the Least Median Squares algorithm to exclude outliers and to calculate the transformation matrix. You can also use all input points to calculate the transformation matrix.

Port	Input/Output	Supported Data Types	Complex Values Supported
Pts1/Pts2	2xN Matrix, (where N is the maximum number of points) coordinates of the input points	<ul style="list-style-type: none"> • Double • Single • 8, 16, 32-bit signed integer • 8, 16, 32-bit unsigned integer 	No
Num	Scalar value that represents the number of valid points in Pts1 and Pts 2	<ul style="list-style-type: none"> • 8, 16, 32-bit signed integer • 8, 16, 32-bit unsigned integer 	No

Estimate Geometric Transformation

Port	Input/Output	Supported Data Types	Complex Values Supported
TForm	2x3 or 3x3, the transformation matrix	<ul style="list-style-type: none"> • Double • Single 	No
Inlier	1xN, indicates which points have been used to calculate TForm	Boolean	No

Ports Pts1 and Pts2 are the points on two images that have the same data type. When Pts 1 and Pts 2 are single or double, the output transformation matrix will also have single or double data type. When Pts1 and Pts2 images are built-in integers, the option is available to set the transformation matrix data type to either Single or Double. The TForm output provides the transformation matrix. The Inlier output port provides the Inlier points on which the transformation matrix is based. This output appears when you select the **Output Boolean signal indication which point pairs are inliers** checkbox.

RANSAC and Least Median Squares Algorithms

The RANSAC algorithm relies on a distance threshold. A pair of points, p_i^a (image a , Pts1) and p_i^b (image b , Pts 2) is an inlier only when the distance between p_i^b and the projection of p_i^a based on the transformation matrix falls within the specified threshold. The distance metric used in the RANSAC algorithm is as follows:

$$d = \sum_{i=1}^{Num} \min(D(p_i^b, \Psi(p_i^a : H)), t)$$

The Least Median Squares algorithm assumes at least 50% of the point pairs can be mapped by a transformation matrix. The algorithm does not need to explicitly specify the distance threshold. Instead, it uses the

Estimate Geometric Transformation

median distance between all input point pairs. The distance metric used in the Least Median of Squares algorithm is as follows:

$$d = \text{median}(D(p_1^b, \psi(p_1^a : H)), D(p_2^b, \psi(p_2^a : H)), \dots, D(p_{Num}^b, \psi(p_N^a : H)))$$

For both equations:

p_i^a is a point in image a (Pts1)

p_i^b is a point in image b (Pts2)

$\psi(p_i^a : H)$ is the projection of a point on image a based on transformation matrix H

$D(p_i^b, p_j^b)$ is the distance between two point pairs on image b

t is the threshold

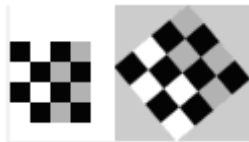
Num is the number of points

The smaller the distance metric, the better the transformation matrix and therefore the more accurate the projection image.

Transformations

The Estimate Geometric Transformation block supports Nonreflective similarity, affine, and projective transformation types, which are described in this section.

Nonreflective similarity transformation supports translation, rotation, and isotropic scaling. It has four degrees of freedom and requires two pairs of points.



Estimate Geometric Transformation

The transformation matrix is: $H = \begin{bmatrix} h_1 & h_2 & h_3 \\ -h_2 & h_1 & h_4 \end{bmatrix}$

The projection of a point $[x \ y]^T$ by H is: $[\hat{x} \ \hat{y}]^T = H[x \ y \ 1]^T$

affine transformation supports nonisotropic scaling in addition to all transformations that the nonreflective similarity transformation supports. It has six degrees of freedom that can be determined from three pairs of noncollinear points.



The transformation matrix is: $H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \end{bmatrix}$

The projection of a point $[x \ y]^T$ by H is: $[\hat{x} \ \hat{y}]^T = H[x \ y \ 1]^T$

Projective transformation supports tilting in addition to all transformations that the affine transformation supports.



The transformation matrix is: $h = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$

The projection of a point $[x \ y]^T$ by H is represented by homogeneous coordinates as: $[\hat{u} \ \hat{v} \ \hat{w}]^T = H[x \ y \ 1]^T$

Distance Measurement

For computational simplicity and efficiency, this block uses algebraic

distance. The algebraic distance for a pair of points, $\begin{bmatrix} x^a & y^a \end{bmatrix}^T$ on

image a , and $\begin{bmatrix} x^b & y^b \end{bmatrix}^T$ on image b , according to transformation H , is defined as follows;

For projective transformation:

$$D(p_i^b, \psi(p_i^a : H)) = ((u^a - w^a x^b)^2 + (v^a - w^a y^b)^2)^{\frac{1}{2}}, \text{ where}$$

$$\begin{bmatrix} \hat{u}^a & \hat{v}^a & \hat{w}^a \end{bmatrix}^T = H \begin{bmatrix} x^a & y^a & 1 \end{bmatrix}^T$$

For Nonreflective similarity or affine transformation:

$$D(p_i^b, \psi(p_i^a : H)) = ((x^a - x^b)^2 + (y^a - y^b)^2)^{\frac{1}{2}},$$

$$\text{where } \begin{bmatrix} \hat{x}^a & \hat{y}^a \end{bmatrix}^T = H \begin{bmatrix} x^a & y^a & 1 \end{bmatrix}^T$$

Algorithm

The block performs a comparison and repeats it M number of times between successive transformation matrices. If you select the **Find and exclude outliers** option, the RANSAC and Least Median Squares (LMS) algorithms become available. These algorithms calculate and compare a distance metric. The transformation matrix that produces the smaller distance metric becomes the new transformation matrix that the next comparison uses. A final transformation matrix is resolved when either:

- M number of random samplings is performed
- The RANSAC algorithm, when enough number of inlier point pairs can be mapped, (dynamically updating M)

Estimate Geometric Transformation

The Estimate Geometric Transformation algorithm follows these steps:

1

A transformation matrix H is initialized to zeros

2

Set count = 0 (Randomly sampling).

3

While count < M, where M is total number of random samplings to perform, perform the following;

a

Increment the count; count = count + 1.

b

Randomly select pair of points from images a and b , (2 pairs for Nonreflective similarity, 3 pairs for affine, or 4 pairs for projective).

c

Calculate a transformation matrix H , from the selected points.

d

If H has a distance metric less than that of H , then replace H with H .

(Optional for RANSAC algorithm only)

i.

Update M dynamically.

ii.

Exit out of sampling loop if enough number of point pairs can be mapped by H .

4

Use all point pairs in images a and b that can be mapped by H to calculate a refined transformation matrix H

5

Iterative Refinement, (Optional for RANSAC and LMS algorithms)

a

Denote all point pairs that can be mapped by H as inliers.

b

Use inlier point pairs to calculate a transformation matrix H .

c

If H has a distance metric less than that of H , then replace H with H , otherwise exit the loop.

Number of Random Samplings

The number of random samplings can be specified by the user for the RANSAC and Least Median Squares algorithms. You can use an additional option with the RANSAC algorithm, which calculates this number based on an accuracy requirement. The **Desired Confidence** level drives the accuracy.

The calculated number of random samplings, M used with the RANSAC algorithm, is as follows:

$$M = \frac{\log(1-p)}{\log(1-q^s)}$$

where

- p is the probability of independent point pairs belonging to the largest group that can be mapped by the same transformation. The probability is dynamically calculated based on the number of inliers found versus the total number of points. As the probability increases, the number of samplings, M , decreases.

Estimate Geometric Transformation

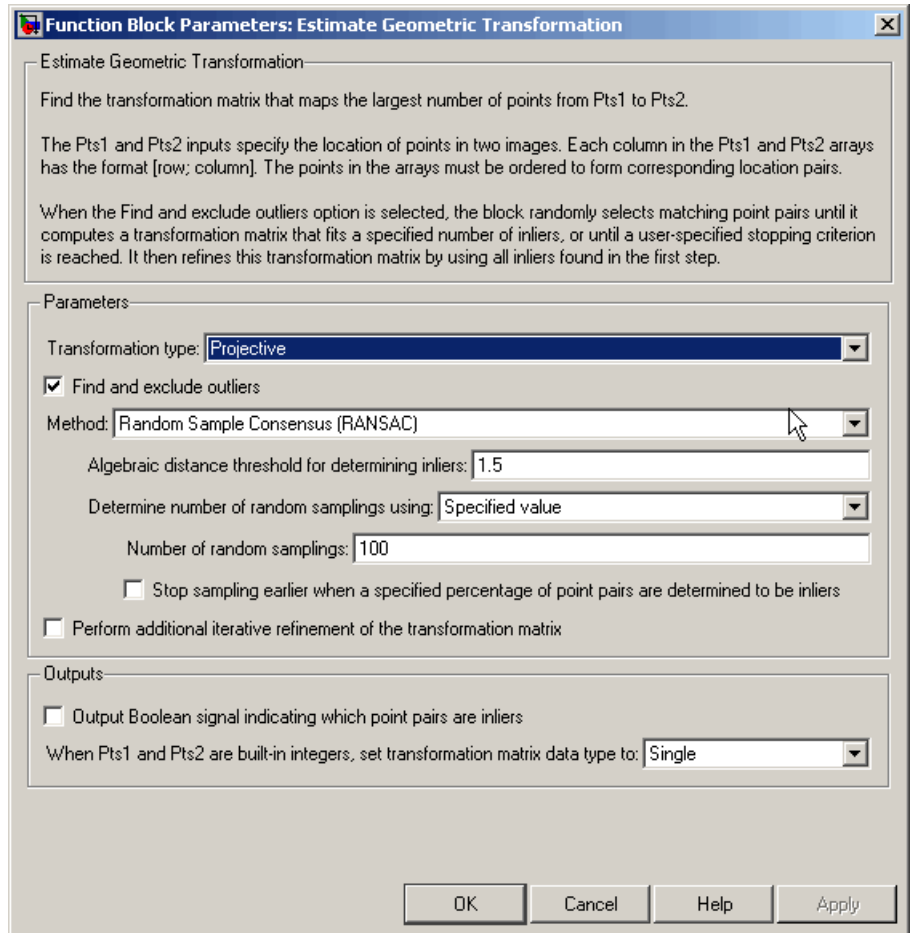
- q is the probability of finding the largest group that can be mapped by the same transformation.
- s is equal to the value 2, 3, or 4 for Nonreflective similarity, affine, and projective transformation, respectively.

Iterative Refinement of Transformation Matrix

The transformation matrix calculated from all inliers can be used to calculate a refined transformation matrix. The refined transformation matrix is then used to find a new set of inliers. This procedure can be repeated until the transformation matrix cannot be further improved. This iterative refinement is optional.

Estimate Geometric Transformation

Dialog Box



Transformation Type

Specify transformation type, either Nonreflective similarity, affine, or projective transformation. If you select projective transformation, you can also specify a scalar algebraic distance

Estimate Geometric Transformation

threshold for determining inliers. If you select either **affine** or **projective** transformation, you can specify the distance threshold for determining inliers in pixels. See “Transformations” on page 2-365 for a more detailed discussion.

Find and exclude outliers

When selected, the block finds and excludes outliers from the input points and uses only the inlier points to calculate the transformation matrix. When this option is not selected, all input points are used to calculate the transformation matrix.

Method

Select either the **RANdom SAMple Consensus (RANSAC)** or the **Least Median of Squares** algorithm to find outliers. See “RANSAC and Least Median Squares Algorithms” on page 2-364 for a more detailed discussion.

Algebraic distance threshold for determining inliers

Specify a scalar threshold value for determining inliers. The threshold controls the upper limit used to find the algebraic distance in the RANSAC algorithm. This parameter appears when the **Method** parameter is **Random Sample Consensus (RANSAC)** and the **Transformation type** parameter is **projective**.

Distance threshold for determining inliers (in pixels)

Specify the upper limit distance a point can differ from the projection location of its associating point. This parameter appears when the **Method** parameter is set to **Random Sample Consensus (RANSAC)** and the value of the **Transformation type** parameter is set to **Nonreflective similarity** or **affine**.

Determine number of random samplings using

Select **Specified value** to enter a positive integer value for number of random samplings, or select **Desired confidence** to set the number of random samplings as a percentage and a maximum number. This parameter appears when you select **Find and exclude outliers** parameter, and the value of the **Method** parameter is **Random Sample Consensus (RANSAC)**.

Number of random samplings

Specify the number of random samplings for the algorithm to perform. This parameter appears when the value of the **Determine number of random samplings using** parameter is Specified value.

Desired confidence (in %)

Specify a percent by entering a number between 0 and 100. The Desired confidence is the probability to find the largest group of points that can be mapped by a transformation matrix. This parameter is visible when the **Determine number of random samplings using** is Desired confidence.

Maximum number of random samplings

Specify an integer number for the maximum number of random samplings. This parameter appears when the **Method** parameter is set to Random Sample Consensus (RANSAC) and the value of **Determine number of random samplings using** parameter is Desired confidence.

Stop sampling earlier when a specified percentage of point pairs are determined to be inlier

Specify to stop random sampling when a percentage of input points have been found as inliers. This parameter appears when the **Method** parameter is Random Sample Consensus (RANSAC).

Perform additional iterative refinement of the transformation matrix

Specify whether to perform refinement on the transformation matrix. This parameter appears when you select **Find and exclude outliers** parameter.

Output Boolean signal indicating which point pairs are inliers

Select this option to output the inlier point pairs that were used to calculate the transformation matrix. This parameter appears when you select **Find and exclude outliers** parameter. This parameter is not used when the data type of points is signed or double.

Estimate Geometric Transformation

When **Pts1** and **Pts2** are built-in integers, set transformation matrix data type to

Specify transformation matrix data type as **Single** or **Double** when the input points are built-in integers. This parameter is not used when the data type of points is signed or double.

Parameter Name	Default Value	Visibility	Tunability
Transformation type	projective	Always	No
Find and exclude outliers	Checked	Always	No
Method	RANSAC	Visible when Find and exclude outliers parameter is selected	No
Algebraic distance threshold for determining inliers	1.5	Visible when Method parameter is Random Sample Consensus (RANSAC) and the Transformatinon type parameter is projective	Yes
Distance threshold for determining inliers (in pixels)	1.5	Visible when Method parameter is Random Sample Consensus (RANSAC) and the Transformatinon type parameter is Nonreflective similarity or affine	Yes
Determine number of random samplings	Specified value	Visible when Find and exclude outliers parameter is selected	No

Estimate Geometric Transformation

Parameter Name	Default Value	Visibility	Tunability
Number of random samplings	100	Visible when Determine number of random samplings using parameter is Specified value	Yes
Maximum number of random samplings	200	Visible when the Method parameter is set to Random Sample Consensus (RANSAC) and Determine number of random samplings using parameter is Desired confidence	Yes
Desired confidence (in%)	99	Visible when the Determine number of random samplings using is Desired confidence	Yes
Stop sampling earlier when a specified percentage of point pairs are determined to be inliers	Unchecked	Visible when the Method parameter is Random Sample Consensus (RANSAC)	No
Inlier percentage	75	Visible when Stop sampling earlier when a specified percentage of point pairs are determined to be inliers parameter is checked	Yes

Estimate Geometric Transformation

Parameter Name	Default Value	Visibility	Tunability
Perform additional iterative refinement of the transformation matrix	Unchecked	Visible when Find and exclude outliers parameter is selected	No
Output Boolean signal indicating which input points are inliers	Unchecked	Visible when Find and exclude outliers parameter is selected	No
When Pts1 and Pts2 are built-in integers, set transformation matrix data type to	Single	Always	No

Examples

Calculate transformation matrix from largest group of point pairs

Examples of input data and application of the Estimate Geometric Transformation block appear in the following figures. Figures (a) and (b) show the point pairs. The points are denoted by stars or circles, and the numbers following them show how they are paired. Some point pairs can be mapped by the same transformation matrix. Other point pairs require a different transformation matrix. One matrix exists that maps the largest number of point pairs, the block calculates and returns this matrix. The block finds the point pairs in the largest group and uses them to calculate the transformation matrix. The point pairs connected by the magenta lines are the largest group.

The transformation matrix can then be used to stitch the images as shown in Figure (e).

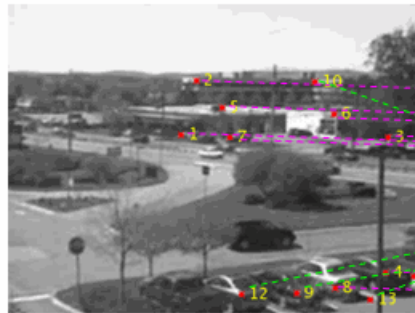
Estimate Geometric Transformation



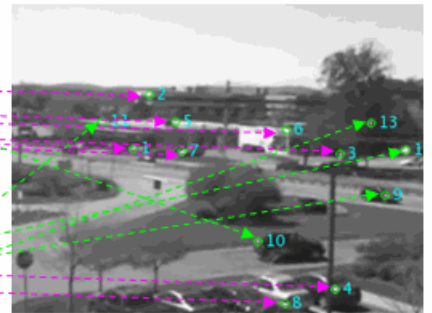
(a)



(b)



(c)



(d)



(e)

Estimate Geometric Transformation

Video Mosaicking

To see an example of the Estimate Geometric Transformation block used in a model with other blocks, see the **Video Mosaicking Demo**, [vipmosaicking](#).

References

R. Hartley and A. Ziserman, "Multiple View Geometry in Computer Vision," Second edition, Cambridge University Press, 2003

See Also

[cp2tform](#)

Image Processing Toolbox software

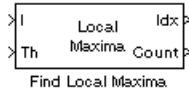
[vipmosaicking](#)

Video and Image Processing Blockset demo

Purpose Find local maxima in matrices

Library Statistics

Description



The Find Local Maxima block finds the local maxima in the input matrix. It does this by comparing the maximum value in the matrix to a user-specified threshold. If the maximum value is greater than or equal to this threshold, the block considers the value a valid local maximum. Then, it sets all the matrix values in the neighborhood, an area around and including the maximum value, to 0. This step ensures that this maximum is not included in subsequent searches. The size of the neighborhood must be appropriate for the data set. That is, it must eliminate enough of the values around the maximum so that false peaks are not discovered. The block repeats this entire process until either it finds all the valid maxima or it finds the number of local maxima equal to the **Maximum number of local maxima (N)** parameter value, whichever comes first.

The block outputs the zero-based row and column coordinates of the maxima at the Idx port and the number of valid local maxima found at the Count port.

Port	Input/Output	Supported Data Types	Complex Values Supported
I/ Hough	Matrix in which you want to find the maxima	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	No
Th	Scalar value that represents the value the maxima should meet or exceed	Same as I/Hough port	No

Find Local Maxima

Port	Input/Output	Supported Data Types	Complex Values Supported
Idx	Vector or matrix that represents the zero-based coordinates of the maxima. The first row represents the row coordinates and the second row represents the column coordinates.	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• 8-, 16-, and 32-bit unsigned integer	No
Count	Scalar value that represents the number of maxima that meet or exceed the threshold value	Same as Idx port	No

The inputs to the I/Hough and Th ports must be the same data type.

Use the **Maximum number of local maxima (N)** parameter to specify the maximum number of maxima to find.

Use the **Neighborhood size** parameter to specify the size of the neighborhood around the maxima over which the block zeros out the values. Enter a two-element vector of positive odd integers, [r c]. Here, r is the number of rows in the neighborhood and c is the number of columns.

Use the **Source of threshold value** parameter to specify how to enter the threshold value. If you select **Input port**, the Th port appears on the block. If you select **Specify via dialog**, the **Threshold** parameter appears in the dialog box. Enter a scalar value that represents the value all maxima should meet or exceed.

If the input to this block is a Hough matrix output from the Hough Transform block, select the **Input is Hough matrix spanning full theta range** check box. If you select this check box, the block assumes that the Hough port input is antisymmetric about the rho axis and

theta ranges from $-\pi/2$ to $\pi/2$ radians. If the block finds a local maxima near the boundary such that the neighborhood lies outside the Hough matrix, the block finds only one local maximum, and it ignores the corresponding antisymmetric maximum.

Use the **Index output data type** parameter to specify the data type of the Idx port output. Your choices are double, single, uint8, uint16, or uint32.

Use the **Count output data type** parameter to specify the data type of the Count port output. Your choices are double, single, uint8, uint16, or uint32.

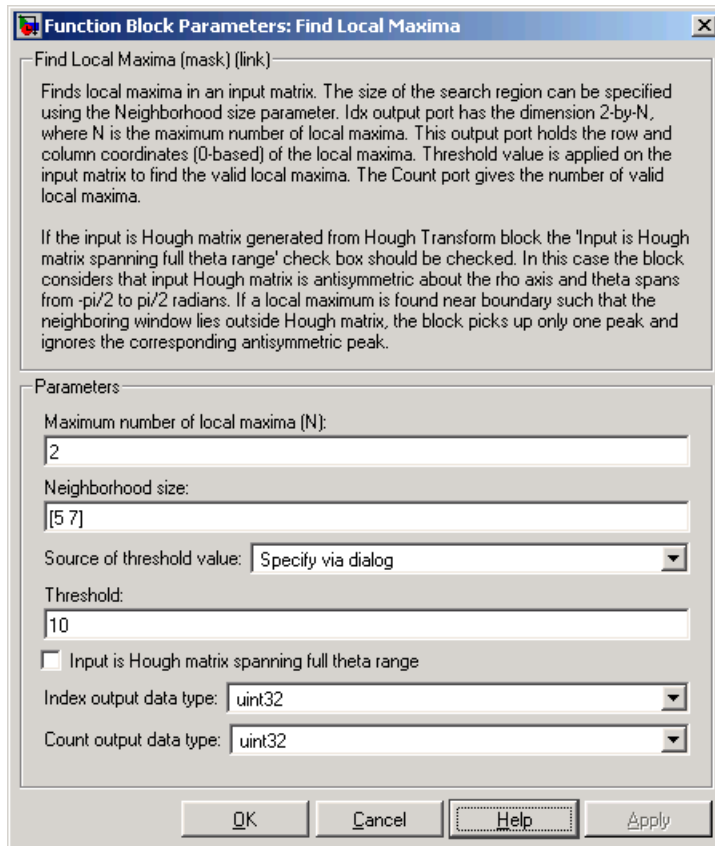
Examples

See “Finding Lines in Images” and “Measuring an Angle Between Lines” in the *Video and Image Processing Blockset User’s Guide*.

Find Local Maxima

Dialog Box

The Find Local Maxima dialog box appears as shown in the following figure.



Maximum number of local maxima (N)

Specify the maximum number of maxima you want the block to find.

Neighborhood size

Specify the size of the neighborhood around the maxima over which the block zeros out the values. Enter a two-element vector of positive odd integers, [r c].

Source of threshold value

Specify how to enter the threshold value. If you select **Input port**, the **Th** port appears on the block. If you select **Specify via dialog**, the **Threshold** parameter appears in the dialog box.

Threshold

Enter a scalar value that represents the value all maxima should meet or exceed. This parameter is visible if, for the **Source of threshold value** parameter, you choose **Specify via dialog**.

Input is Hough matrix spanning full theta range

If you select this check box, the block assumes that the Hough port input is antisymmetric about the rho axis and theta ranges from $-\pi/2$ to $\pi/2$ radians.

Index output data type

Specify the data type of the Peaks port output. Your choices are double, single, uint8, uint16, or uint32.

Count output data types

Specify the data type of the Count port output. Your choices are double, single, uint8, uint16, or uint32.

See Also

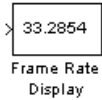
Hough Lines	Video and Image Processing Blockset software
Hough Transform	Video and Image Processing Blockset software
houghpeaks	Image Processing Toolbox software

Frame Rate Display

Purpose Calculate average update rate of input signal

Library Sinks

Description The Frame Rate Display block calculates and displays the average update rate of the input signal. This rate is in relation to the wall clock time. For example, if the block displays 30, the model is updating the input signal 30 times every second. You can use this block to check the video frame rate of your simulation. During code generation, Real-Time Workshop[®] does not generate code for this block.



Note This block supports intensity and color images on its port.

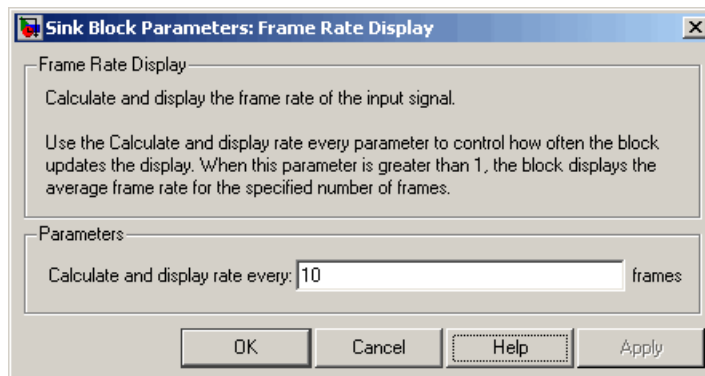
Port	Input	Supported Data Types	Complex Values Supported
Input	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• Boolean• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No

Use the **Calculate and display rate every** parameter to control how often the block updates the display. When this parameter is greater than 1, the block displays the average update rate for the specified number of video frames. For example, if you enter 10, the block calculates the amount of time it takes for the model to pass 10 video frames to the block. It divides this time by 10 and displays this average video frame rate on the block.

Note If you do not connect the Frame Rate Display block to a signal line, the block displays the base (fastest) rate of the Simulink model.

Dialog Box

The Frame Rate Display dialog box appears as shown in the following figure.



Calculate and display rate every

Use this parameter to control how often the block updates the display.

See Also

To Multimedia File	Signal Processing Blockset software
To Video Display	Video and Image Processing Blockset software
Video To Workspace	Video and Image Processing Blockset software
Video Viewer	Video and Image Processing Blockset software

From Multimedia File

Purpose Read video frames and audio samples from compressed multimedia file

Library Sources

Description The From Multimedia File block is a Signal Processing Blockset block. For more information, see the From Multimedia File block reference page in the Signal Processing Blockset software documentation.

Purpose

Apply or remove gamma correction from images or video streams

Library

Conversions

Description



Use the Gamma Correction block to apply or remove gamma correction from an image or video stream. For input signals normalized between 0 and 1, the block performs gamma correction as defined by the following equations. For integers and fixed-point data types, these equations are generalized by applying scaling and offset values specific to the data type:

$$S_{LS} = \frac{1}{\frac{\gamma}{B_P^{(\frac{1}{\gamma}-1)}} - \gamma B_P + B_P}$$

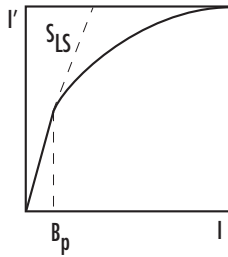
$$F_S = \frac{\gamma S_{LS}}{B_P^{(\frac{1}{\gamma}-1)}}$$

$$C_O = F_S B_P^{\frac{1}{\gamma}} - S_{LS} B_P$$

$$I' = \left\{ \begin{array}{ll} S_{LS} I, & I \leq B_P \\ F_S I^{\frac{1}{\gamma}} - C_O, & I > B_P \end{array} \right\}$$

S_{LS} is the slope of the straight line segment. B_P is the break point of the straight line segment, which corresponds to the **Break point** parameter. F_S is the slope matching factor, which matches the slope of the linear segment to the slope of the power function segment. C_O is the segment offset, which ensures that the linear segment and the power function segments connect. Some of these parameters are illustrated by the following diagram.

Gamma Correction



For normalized input signals, the block removes gamma correction, which linearizes the input video stream, as defined by the following equation:

$$I = \begin{cases} I'/S_{LS}, & I' \leq B_P \\ \left(\frac{I' + C_O}{F_S} \right)^\gamma, & I' > B_P \end{cases}$$

Typical gamma values range from 1 to 3. Most monitor gamma values range from 1.8 to 2.2. Check with the manufacturer of your hardware to obtain the exact gamma value. Gamma function parameters for some common standards are shown in the following table:

Standard	Slope	Break Point	Gamma
CIE L*	9.033	0.008856	3
Recommendation ITU-R BT.709-3, Parameter Values for the HDTV Standards for Production and International Programme Exchange	4.5	0.018	20/9
sRGB	12.92	0.00304	2.4

Note This block supports intensity and color images on its ports.

The properties of the input and output ports are summarized in the following table:

Port	Input/Output	Supported Data Types	Complex Values Supported
I	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point (up to 16-bit word length) • 8- and 16-bit signed integer • 8- and 16-bit unsigned integer 	No
I'	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	Same as I port	No

Use the **Operation** parameter to specify the block's operation. If you want to perform gamma correction, select **Gamma**. If you want to linearize the input signal, select **De-gamma**.

If, for the **Operation** parameter, you select **Gamma**, use the **Gamma** parameter to enter the desired gamma value of the output video stream. This value must be greater than or equal to 1. If, for the **Operation** parameter, you select **De-gamma**, use the **Gamma** parameter to enter the gamma value of the input video stream.

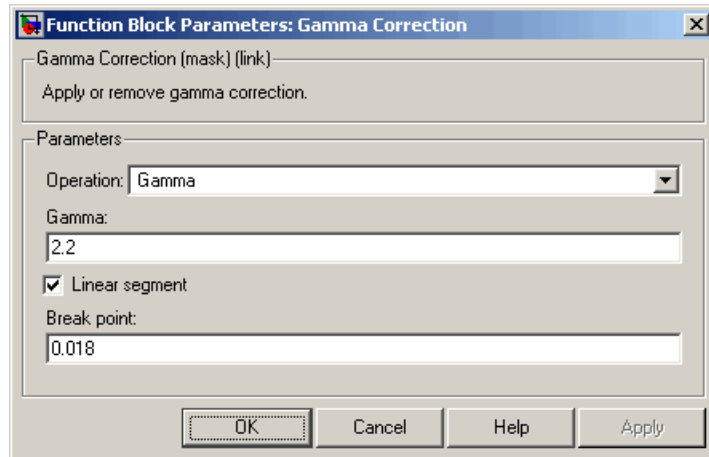
Select the **Linear segment** check box if you want the gamma curve to have a linear portion near black. If you select this check box, the **Break point** parameter appears on the dialog box. Enter a scalar value that

Gamma Correction

indicates the *I*-axis value of the end of the linear segment. The break point is shown in the first diagram of this block reference page.

Dialog Box

The Gamma Correction dialog box appears as shown in the following figure.



Operation

Specify the block's operation. Your choices are Gamma or De-gamma.

Gamma

If, for the **Operation** parameter, you select Gamma, enter the desired gamma value of the output video stream. This value must be greater than or equal to 1. If, for the **Operation** parameter, you select De-gamma, enter the gamma value of the input video stream.

Linear segment

Select this check box if you want the gamma curve to have a linear portion near the origin.

Break point

Enter a scalar value that indicates the I -axis value of the end of the linear segment. This parameter is visible if you select the **Linear segment** check box.

References

[1] Poynton, Charles. *Digital Video and HDTV Algorithms and Interfaces*. San Francisco, CA: Morgan Kaufman Publishers, 2003.

See Also

Color Space Conversion	Video and Image Processing Blockset software
imadjust	Image Processing Toolbox software

Gaussian Pyramid

Purpose Perform Gaussian pyramid decomposition

Library Transforms
viptransforms

Description The Gaussian Pyramid block computes Gaussian pyramid reduction or expansion to resize an image. The image reduction process involves lowpass filtering and downsampling the image pixels. The image expansion process involves upsampling the image pixels and lowpass filtering. You can also use this block to build a Laplacian pyramid. For more information, see “Examples” on page 2-394.



Note This block supports intensity and color images on its ports.

Port	Output	Supported Data Types	Complex Values Supported
Input	<p>In Reduce mode, the input can be an M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes.</p> <p>In Expand mode, the input can be a scalar, vector, or M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes.</p>	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	No
Output	In Reduce mode, the output can be a scalar,	Same as Input port	No

Port	Output	Supported Data Types	Complex Values Supported
	<p>vector, or matrix that represents one level of a Gaussian pyramid.</p> <p>In Expand mode, the output can be a matrix that represents one level of a Gaussian pyramid.</p>		

Use the **Operation** parameter to specify whether to reduce or expand the input image. If you select **Reduce**, the block applies a lowpass filter and then downsamples the input image. If you select **Expand**, the block upsamples and then applies a lowpass filter to the input image.

Use the **Pyramid level** parameter to specify the number of times the block upsamples or downsamples each dimension of the image by a factor of 2. For example, suppose you have a 4-by-4 input image. You set the **Operation** parameter to **Reduce** and the **Pyramid level** to 1. The block filters and downsamples the image and outputs a 2-by-2 pixel output image. If you have an M-by-N input image and you set the **Operation** parameter to **Reduce**, you can calculate the dimensions of the output image using the following equation:

$$\text{ceil}\left(\frac{M}{2}\right) \text{ by } \text{ceil}\left(\frac{N}{2}\right)$$

You must repeat this calculation for each successive pyramid level. If you have an M-by-N input image and you set the **Operation** parameter to **Expand**, you can calculate the dimensions of the output image using the following equation:

$$\left[(M-1)2^l + 1\right] \text{ by } \left[(N-1)2^l + 1\right]$$

Gaussian Pyramid

In the previous equation, l is the scalar value from 1 to ∞ that you enter for the **Pyramid level** parameter.

Use the **Coefficient source** parameter to specify the coefficients of the lowpass filter. If you select **Default separable filter** [$1/4 - a/2$ $1/4$ a $1/4$ $1/4 - a/2$], use the **a** parameter to define the coefficients in the vector of separable filter coefficients. If you select **Specify via dialog**, use the **Coefficient for separable filter** parameter to enter a vector of separable filter coefficients.

Examples

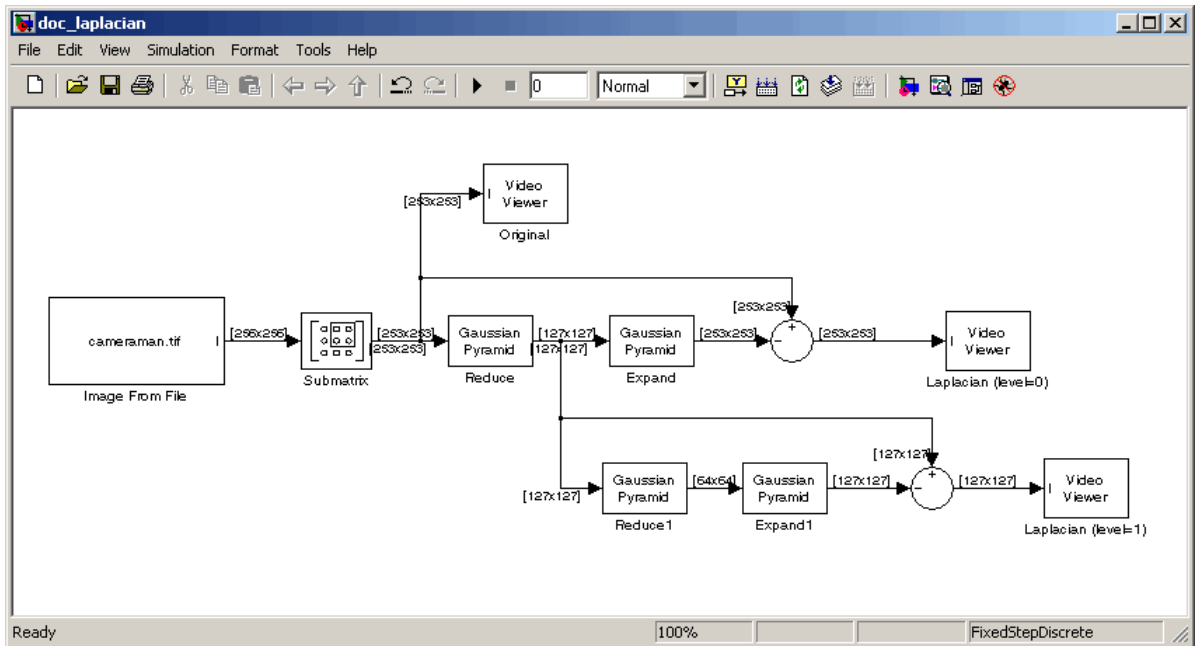
The following example model shows how to construct a Laplacian pyramid:

- 1 Open this model by typing

```
doc_laplacian
```

at the MATLAB command prompt.

Gaussian Pyramid



2 Run the model to see the following results.

Gaussian Pyramid

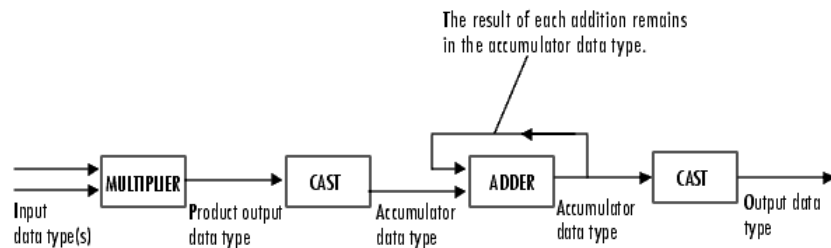




You can construct a Laplacian pyramid if the dimensions of the input image, R-by-C, satisfy $R = M_R 2^N + 1$ and $C = M_C 2^N + 1$, where M_R , M_C , and N are integers. In this example, you have an input matrix that is 256-by-256. If you set M_R and M_C equal to 63 and N equal to 2, you find that the input image needs to be 253-by-253. So you use a Submatrix block to crop the dimensions of the input image to 253-by-253.

Fixed-Point Data Types

The following diagram shows the data types used in the Gaussian Pyramid block for fixed-point signals:

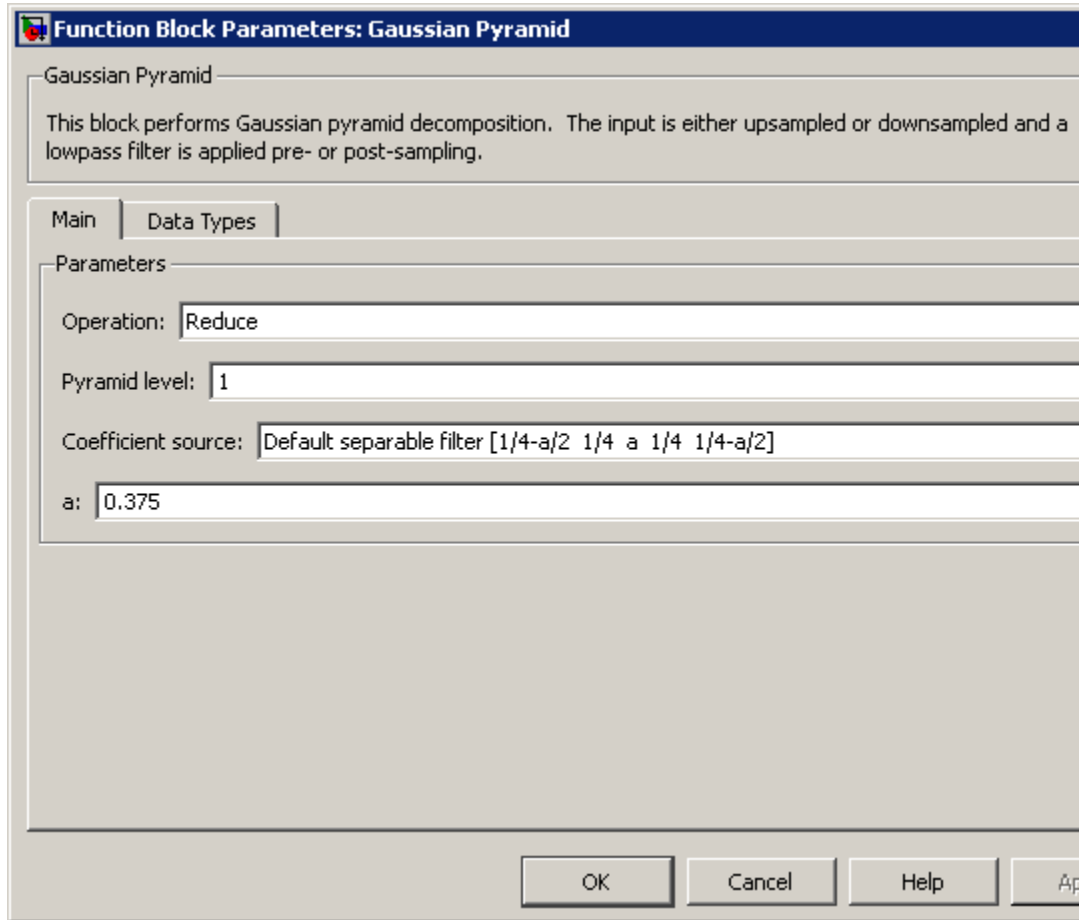


You can set the coefficients table, product output, accumulator, and output data types in the block mask.

Gaussian Pyramid

Dialog Box

The **Main** pane of the Gaussian Pyramid dialog box appears as shown in the following figure.



Operation

Specify whether you want to reduce or expand the input image.

Pyramid level

Specify the number of times the block upsamples or downsamples each dimension of the image by a factor of 2.

Coefficient source

Determine how to specify the coefficients of the lowpass filter. Your choices are **Default separable filter** [$1/4 - a/2$ $1/4$ a $1/4$ $1/4 - a/2$] or **Specify via dialog**.

a

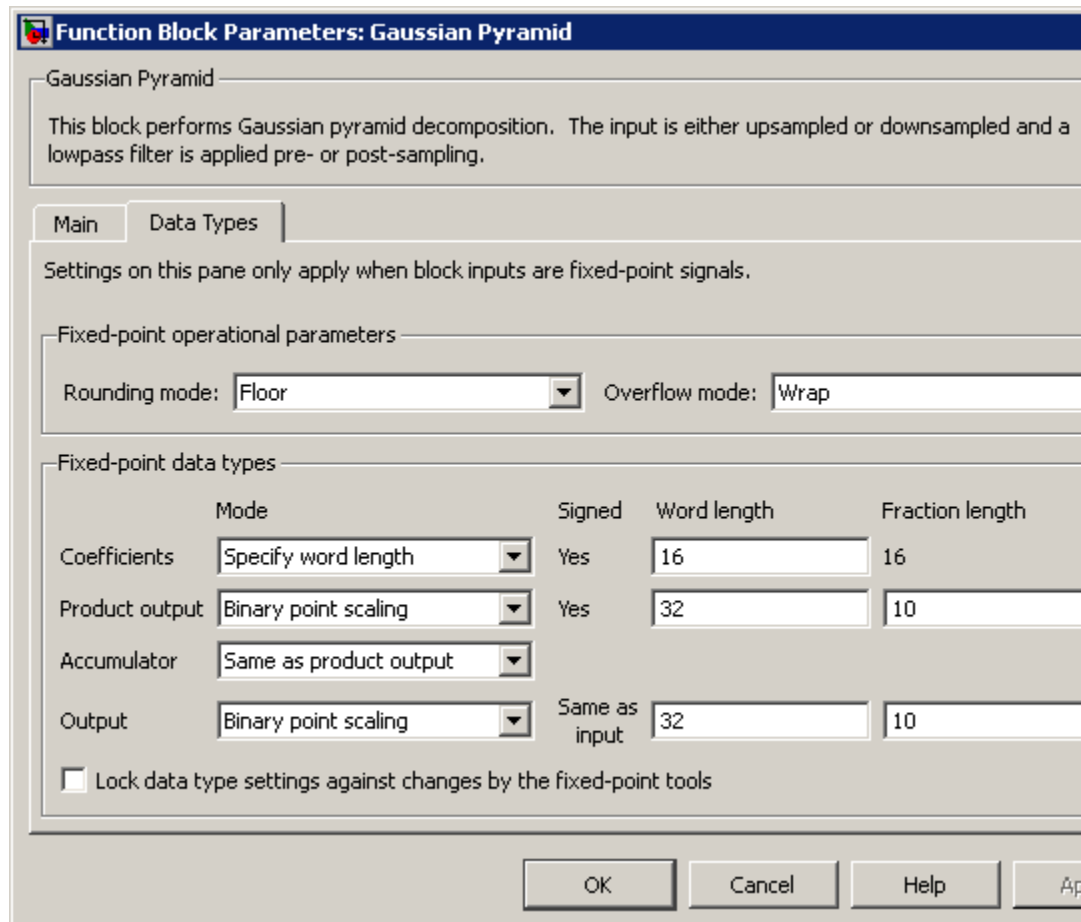
Enter a scalar value that defines the coefficients in the default separable filter [$1/4 - a/2$ $1/4$ a $1/4$ $1/4 - a/2$]. This parameter is visible if, for the **Coefficient source** parameter, you select **Default separable filter** [$1/4 - a/2$ $1/4$ a $1/4$ $1/4 - a/2$].

Coefficients for separable filter

Enter a vector of separable filter coefficients. This parameter is visible if, for the **Coefficient source** parameter, you select **Specify via dialog**.

The **Data Types** pane of the Gaussian Pyramid dialog box appears as shown in the following figure.

Gaussian Pyramid



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Coefficients

Choose how to specify the word length and the fraction length of the coefficients:

- When you select **Same word length as input**, the word length of the coefficients match that of the input to the block. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select **Specify word length**, you can enter the word length of the coefficients, in bits. The block automatically sets the fraction length to give you the best precision.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the coefficients, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the coefficients. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Product output



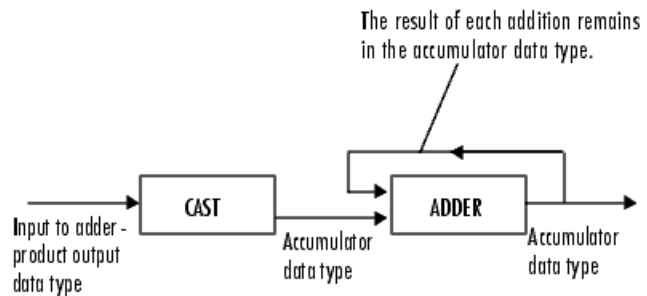
As shown in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate the product output word and fraction lengths.

- When you select **Same as input**, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.

Gaussian Pyramid

- When you select **Slope** and **bias** scaling, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Accumulator



As shown in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate the accumulator word and fraction lengths.

- When you select **Same** as product output, these characteristics match those of the product output.
- When you select **Same** as input, these characteristics match those of the input to the block.
- When you select **Binary point** scaling, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope** and **bias** scaling, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Output

Choose how to specify the word length and fraction length of the output of the block:

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

See Also

Resize

Video and Image Processing Blockset software

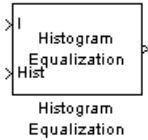
Histogram

Purpose	Generate histogram of each input matrix
Library	Statistics
Description	The Histogram block is a Signal Processing Blockset block. For more information, see the Histogram block reference page in the Signal Processing Blockset software documentation.

Purpose Enhance contrast of images using histogram equalization

Library Analysis & Enhancement

Description The Histogram Equalization block enhances the contrast of images by transforming the values in an intensity image so that the histogram of the output image approximately matches a specified histogram.



Port	Input/Output	Supported Data Types	Complex Values Supported
I	Matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	No
Hist	Vector of integer values that represents the desired intensity values in each bin	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	No
Output	Matrix of intensity values	Same as I port	No

If the data type of input to the I port is floating point, the input to Hist port must be the same data type. The output signal has the same data type as the input signal.

Use the **Target histogram** parameter to designate the histogram you want the output image to have.

Histogram Equalization

If you select **Uniform**, the block transforms the input image so that the histogram of the output image is approximately flat. Use the **Number of bins** parameter to enter the number of equally spaced bins you want the uniform histogram to have.

If you select **User-defined**, the **Histogram source** and **Histogram** parameters appear on the dialog box. Use the **Histogram source** parameter to select how to specify your histogram. If, for the **Histogram source** parameter, you select **Specify via dialog**, you can use the **Histogram** parameter to enter the desired histogram of the output image. The histogram should be a vector of integer values that represents the desired intensity values in each bin. The block transforms the input image so that the histogram of the output image is approximately the specified histogram.

If, for the **Histogram source** parameter, you select **Input port**, the **Hist port** appears on the block. Use this port to specify your desired histogram.

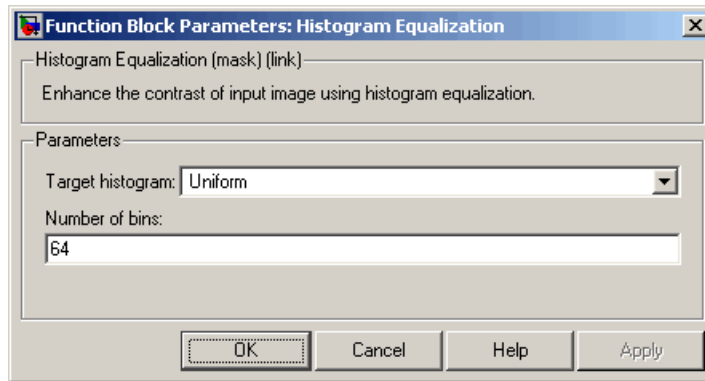
Note The vector input to the **Hist port** must be normalized such that the sum of the values in all the bins is equal to the number of pixels in the input image. The block does not error if the histogram is not normalized.

Examples

See “Adjusting the Contrast in Intensity Images” and “Adjusting the Contrast in Color Images” in the *Video and Image Processing Blockset User’s Guide*.

Dialog Box

The Histogram Equalization dialog box appears as shown in the following figure.



Target histogram

Designate the histogram you want the output image to have.

If you select **Uniform**, the block transforms the input image so that the histogram of the output image is approximately flat. If you select **User-defined**, you can specify the histogram of your output image.

Number of bins

Enter the number of equally spaced bins you want the uniform histogram to have. This parameter is visible if, for the **Target histogram** parameter, you select **Uniform**.

Histogram source

Select how to specify your histogram. Your choices are **Specify via dialog** and **Input port**. This parameter is visible if, for the **Target histogram** parameter, you select **User-defined**.

Histogram

Enter the desired histogram of the output image. This parameter is visible if, for the **Target histogram** parameter, you select **User-defined**.

Histogram Equalization

See Also

`imadjust`

Image Processing Toolbox

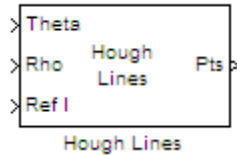
`histeq`

Image Processing Toolbox

Purpose Find Cartesian coordinates of lines described by rho and theta pairs

Library Transforms

Description

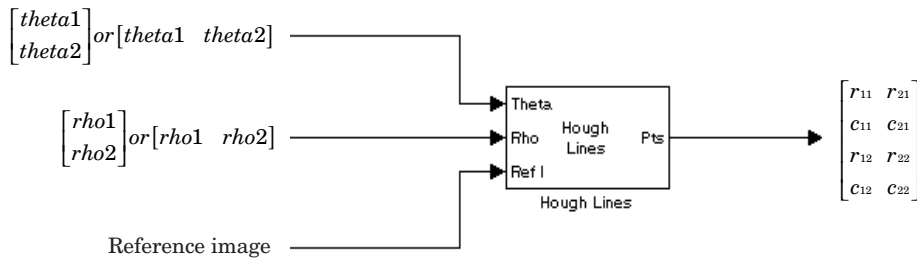


The Hough Lines block finds the points of intersection between the reference image boundary lines and the line specified by a (rho, theta) pair. The block outputs the zero-based row and column positions of the intersections. The boundary lines indicate the left and right vertical boundaries and the top and bottom horizontal boundaries of the reference image.

If the line specified by the (rho, theta) pair does not intersect two border lines in the reference image, the block outputs the values, $[(-1, -1), (-1, -1)]$. This output intersection value allows the next block in your model to ignore the points. Generally, the Hough Lines block precedes a block that draws a point or shape at the intersection.

The following figure shows the input and output coordinates for the Hough Lines block.

Hough Lines



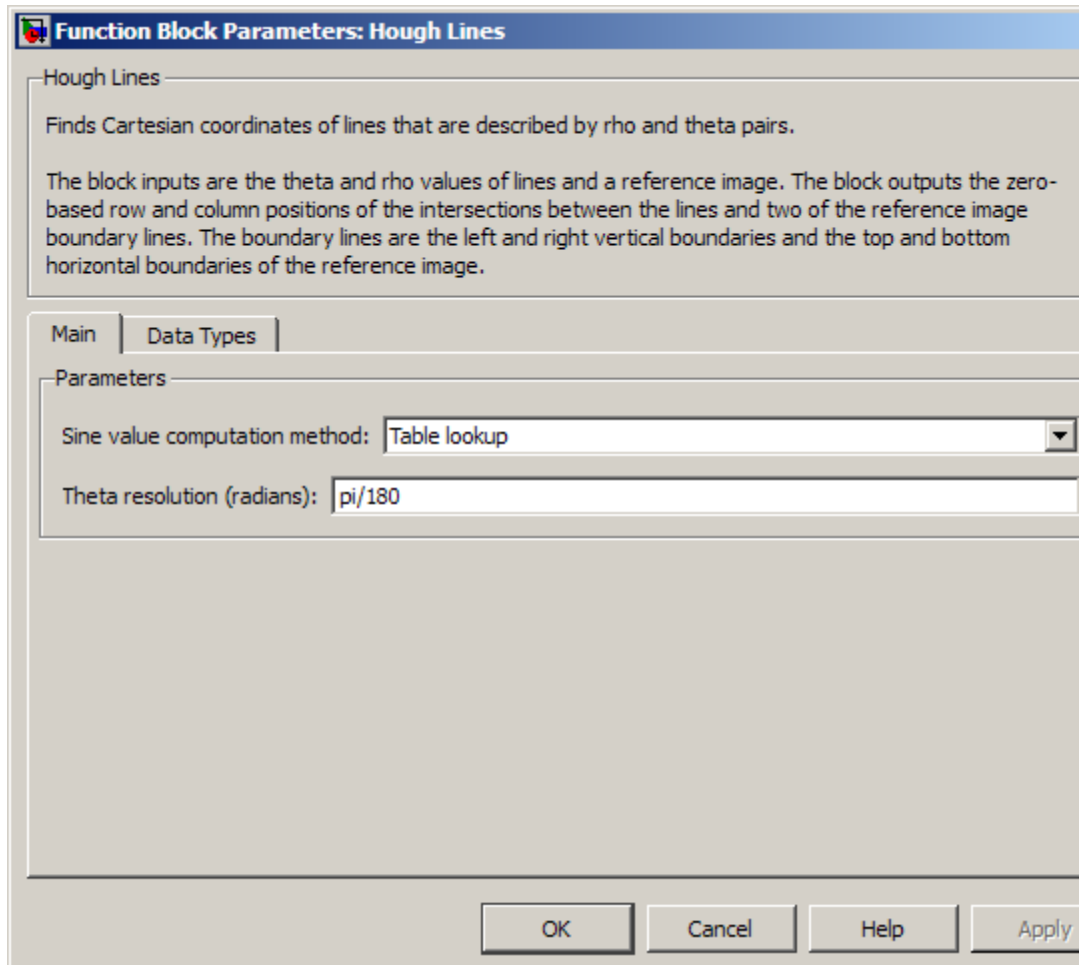
Port	Input/Output	Supported Data Types	Complex Values Supported
Theta	Vector of theta values that represent input lines	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point (signed, word length less than or equal to 32) • 8-, 16-, and 32-bit signed integer 	No
Rho	Vector of rho values that represent input lines	Same as Theta port	No
Ref I	Matrix that represents a binary or intensity image or matrix that represents one plane of an RGB image	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed-point (signed and unsigned) • Custom data types • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Pts	4-by-N matrix of intersection values, where N is the number of input lines	<ul style="list-style-type: none">• 32-bit signed integer	No

Hough Lines

Dialog Box

The **Main** pane of the Hough Lines dialog box appears as shown in the following figure.



Sine value computation method

If you select Trigonometric function, the block computes sine and cosine values to calculate the intersections of the lines during

the simulation. If you select `Table lookup`, the block computes and stores the trigonometric values to calculate the intersections of the lines before the simulation starts. In this case, the block requires extra memory.

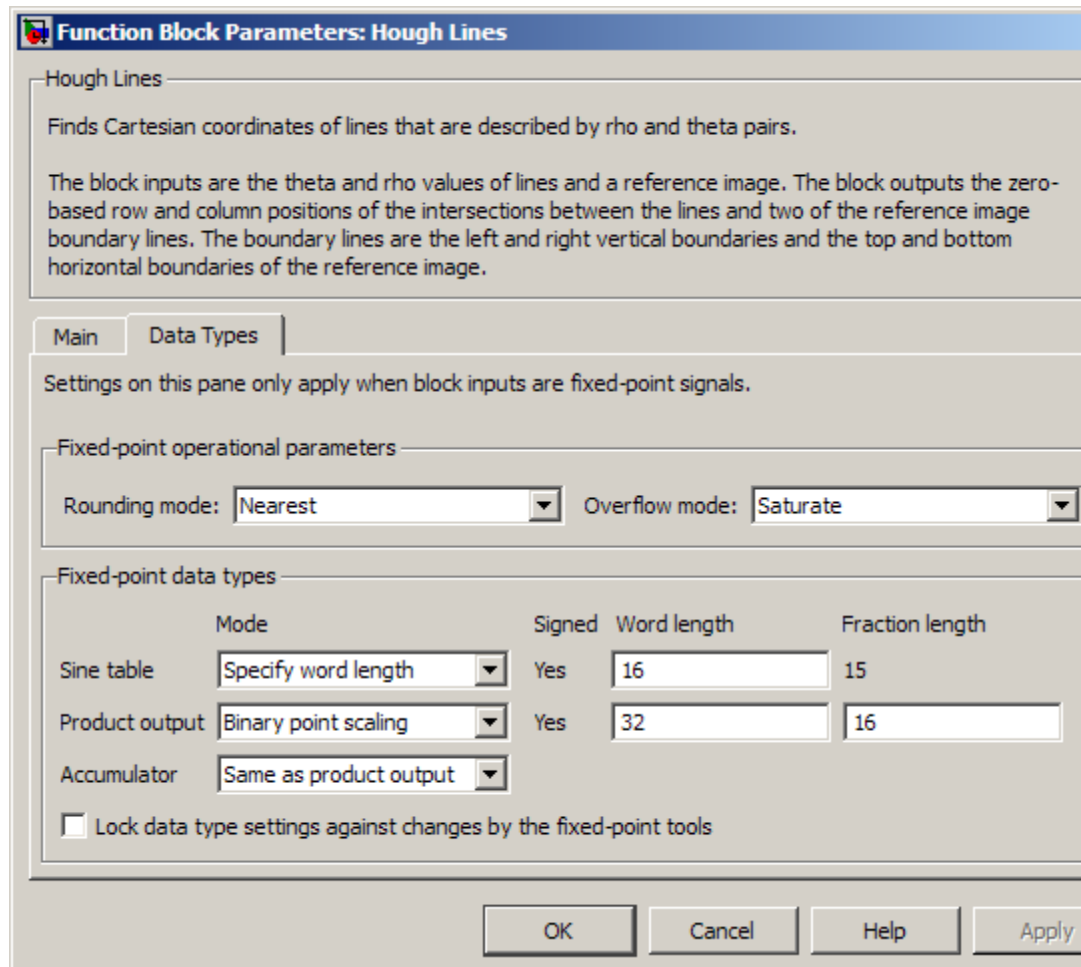
For floating-point inputs, set the **Sine value computation method** parameter to `Trigonometric function`. For fixed-point inputs, set the parameter to `Table lookup`.

Theta resolution (radians)

Use this parameter to specify the spacing of the theta-axis. This parameter appears in the dialog box only if, for the **Sine value computation method** parameter, you select `Table lookup`. parameter appears in the dialog box.

The **Data Types** pane of the Hough Lines dialog box appears as shown in the following figure.

Hough Lines



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Sine table

Choose how to specify the word length of the values of the sine table. The fraction length of the sine table values always equals the word length minus one:

When you select `Specify word length`, you can enter the word length of the sine table.

The sine table values do not obey the **Rounding mode** and **Overflow mode** parameters; they saturate and round to Nearest.

Product output

Use this parameter to specify how to designate this product output word and fraction lengths:

When you select `Same as first input`, the characteristics match the characteristics of the first input to the block.

When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.

When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the product output. All signals in the Video and Image Processing Blockset blocks have a bias of 0.

See “Multiplication Data Types” for illustrations depicting the use of the product output.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths.

When you select `Same as product output` the characteristics match the characteristics of the product output.

When you select `Binary point scaling`, you can enter the **Word length** and the **Fraction length** of the accumulator, in bits.

Hough Lines

When you select **Slope** and **bias scaling**, you can enter the **Word length**, in bits, and the **Slope** of the **Accumulator**. All signals in the Video and Image Processing Blockset software have a bias of 0.

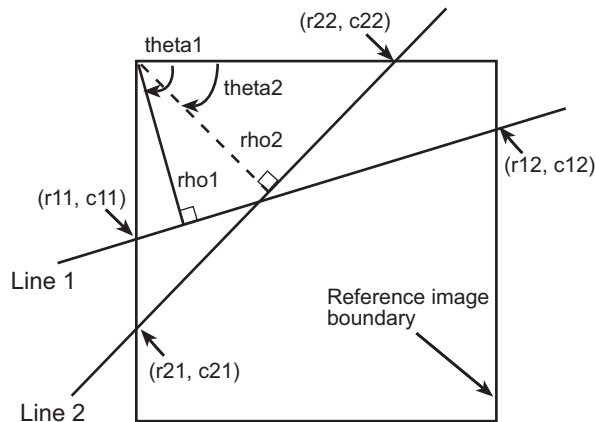
See “Multiplication Data Types” for illustrations depicting the use of the accumulator data type in this block.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

Examples

The following figure shows Line 1 intersecting the boundaries of the reference image at $[(r11, c11) (r12, c12)]$ and Line 2 intersecting the boundaries at $[(r21, c21) (r22, c22)]$



See “Finding Lines in Images” and “Measuring an Angle Between Lines” in the *Video and Image Processing Blockset User Guide*.

See Also

[Find Local Maxima](#)

[Video and Image Processing Blockset](#)

[Hough Transform](#)

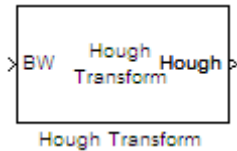
[Video and Image Processing Blockset](#)

Hough Transform

Purpose Find lines in images

Library Transforms
viptransforms

Description

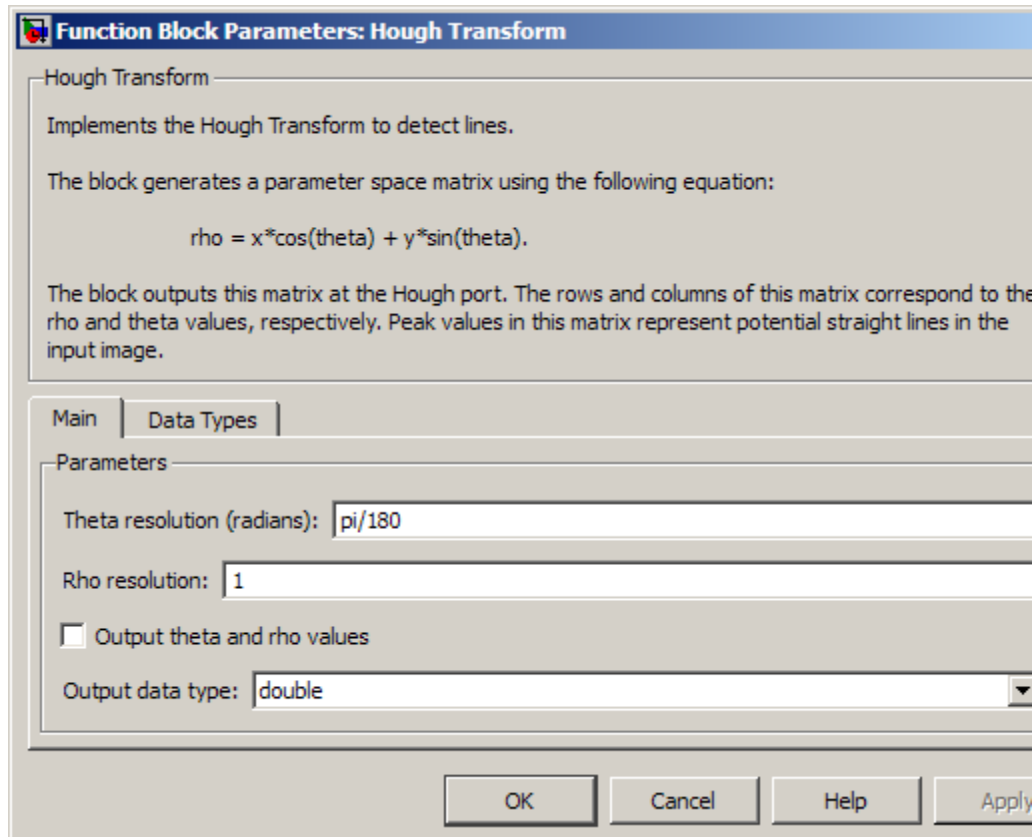


Use the Hough Transform block to find lines in an image. The block outputs the Hough space matrix and, optionally, the *rho*-axis and *theta*-axis vectors. Peak values in the matrix represent potential lines in the input image. Generally, the Hough Transform block precedes the Hough Lines block which uses the output of this block to find lines in an image. You can instead use a custom algorithm to locate peaks in the Hough space matrix in order to identify potential lines.

Port	Input/Output	Supported Data Types	Supported Complex Values
BW	Matrix that represents a binary image	Boolean	No
Hough	Parameter space matrix	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point (unsigned, fraction length equal to 0) • 8-, 16-, 32-bit unsigned integer 	No
Theta	Vector of theta values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point (signed) • 8-, 16-, 32-bit signed integer 	No
Rho	Vector of rho values	Same as Theta port	No

Dialog Boxes

The **Main** pane of the Hough Transform dialog box appears as shown in the following figure.



Theta resolution (radians)

Specify the spacing of the Hough transform bins along the *theta*-axis.

Rho resolution (pixels)

Specify the spacing of the Hough transform bins along the *rho*-axis.

Hough Transform

Output theta and rho values

If you select this check box, the **Theta** and **Rho** ports appear on the block. The block outputs theta and rho-axis vector values at these ports.

Output data type

Specify the data type of your output signal.

The **Data Types** pane of the Hough Transform block dialog appears as shown in the following figure. The Data Types pane will not show fixed-point parameters when **Output data type** parameter is set to double or single.

Function Block Parameters: Hough Transform

Hough Transform

Implements the Hough Transform to detect lines.

The block generates a parameter space matrix using the following equation:

$$\rho = x \cdot \cos(\theta) + y \cdot \sin(\theta).$$

The block outputs this matrix at the Hough port. The rows and columns of this matrix correspond to the ρ and θ values, respectively. Peak values in this matrix represent potential straight lines in the input image.

Main | Data Types

Settings on this pane only apply when block inputs are fixed-point signals.

Fixed-point operational parameters

Rounding mode: Overflow mode:

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Sine Table	<input type="text" value="Binary point scaling"/>	Yes	<input type="text" value="16"/>	<input type="text" value="14"/>
Rho	<input type="text" value="Binary point scaling"/>	Yes	<input type="text" value="32"/>	<input type="text" value="16"/>
Product output	<input type="text" value="Binary point scaling"/>	Yes	<input type="text" value="32"/>	<input type="text" value="20"/>
Accumulator	<input type="text" value="Binary point scaling"/>	Yes	<input type="text" value="32"/>	<input type="text" value="20"/>
Hough output	<input type="text" value="Binary point scaling"/>	No	<input type="text" value="16"/>	<input type="text" value="0"/>
Theta output	<input type="text" value="Binary point scaling"/>	Yes	<input type="text" value="32"/>	<input type="text" value="16"/>

Lock data type settings against changes by the fixed-point tools

OK Cancel Help

Hough Transform

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Sine table

Choose how to specify the word length of the values of the sine table:

- When you select **Binary point scaling**, you can enter the word length of the sine table values, in bits.
- When you select **Slope and bias scaling**, you can enter the word length of the sine table values, in bits.

The sine table values do not obey the **Rounding mode** and **Overflow mode** parameters; they always saturate and round to Nearest.

Rho

Choose how to specify the word length and the fraction length of the rho values:

- When you select **Binary point scaling**, you can enter the word length and the fraction length of the rho values, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the rho values. All signals in Video and Image Processing Blockset blocks have a bias of 0.

Product output

. Use this parameter to specify how to designate the product output word and fraction lengths:

- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. All

signals in Video and Image Processing Blockset blocks have a bias of 0.

See “Multiplication Data Types” for illustrations depicting the use of the product output.

Accumulator

Use this parameter to specify how to designate this accumulator word and fraction lengths:

- When you select **Same** as product output, these characteristics match the characteristics of the product output.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. All signals in Video and Image Processing Blockset blocks have a bias of 0.

See “Multiplication Data Types” for illustrations depicting the use of the accumulator data type in this block.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

Hough output

Choose how to specify the word length and fraction length of the Hough output of the block:

- When you select **Binary point scaling**, you can enter the word length of the Hough output, in bits. The fraction length always has a value of 0.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, of the Hough output. The slope always

Hough Transform

has a value of 0. All signals in Video and Image Processing Blockset blocks have a bias of 0.

Theta output

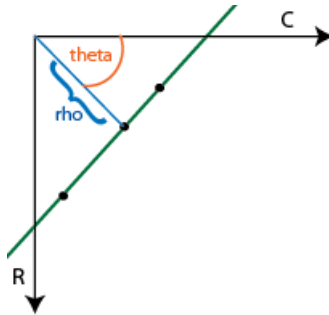
Choose how to specify the word length and fraction length of the theta output of the block:

- When you select **Binary point scaling**, you can enter the word length and the fraction length of the theta output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the theta output. All signals in Video and Image Processing Blockset blocks have a bias of 0.

Algorithm

The Hough Transform block implements the Standard Hough Transform (SHT). The SHT uses the parametric representation of a line:

$$rho = x * \cos(theta) + y * \sin(theta)$$



The variable *rho* indicates the perpendicular distance from the origin to the line.

The variable *theta* indicates the angle of inclination of the normal line

from the x-axis. The range of *theta* is $-\frac{\pi}{2} \leq \theta < +\frac{\pi}{2}$ with a step-size determined by the **Theta resolution (radians)** parameter. The SHT

measures the angle of the line clockwise with respect to the positive x-axis.

The Hough Transform block creates an accumulator matrix. The $(rho, theta)$ pair represent the location of a cell in the accumulator matrix. Every valid (logical true) pixel of the input binary image represented by (R, C) produces a rho value for all theta values. The block quantizes the rho values to the nearest number in the rho vector. The rho vector depends on the size of the input image and the user-specified rho resolution. The block increments a counter (initially set to zero) in those accumulator array cells represented by $(rho, theta)$ pairs found for each pixel. This process validates the point (R, C) to be on the line defined by $(rho, theta)$. The block repeats this process for each logical true pixel in the image. The **Hough** block outputs the resulting accumulator matrix.

Examples

See “Finding Lines in Images” and “Measuring an Angle Between Lines” in the *Video and Image Processing Blockset User Guide*.

See Also

Find Local Maxima	Video and Image Processing Blockset
Hough Lines	Video and Image Processing Blockset
hough	Image Processing Toolbox
houghlines	Image Processing Toolbox
houghpeaks	Image Processing Toolbox

Image Complement

Purpose Compute complement of pixel values in binary, intensity, or RGB images

Library Conversions

Description The Image Complement block computes the complement of a binary, intensity, or RGB image. For binary images, the block replaces pixel values equal to 0 with 1 and pixel values equal to 1 with 0. For an intensity or RGB image, the block subtracts each pixel value from the maximum value that can be represented by the input data type and outputs the difference.



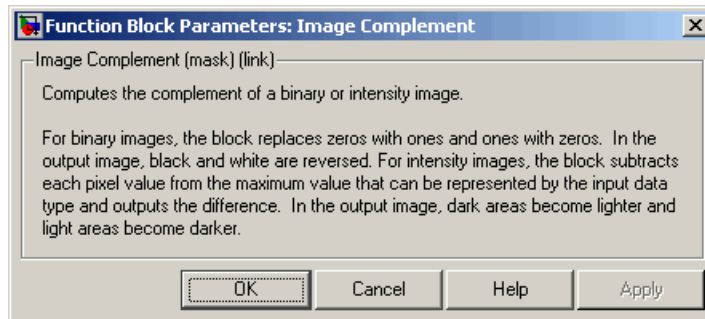
For example, suppose the input pixel values are given by $x(i)$ and the output pixel values are given by $y(i)$. If the data type of the input is double or single precision floating-point, the block outputs $y(i) = 1.0 - x(i)$. If the input is an 8-bit unsigned integer, the block outputs $y(i) = 255 - x(i)$.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Boolean• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer	No
Output	Complement of a binary, intensity, or RGB image	Same as Input port	No

The dimensions, data type, complexity, and frame status of the input and output signals are the same.

Dialog Box

The Image Complement dialog box appears as shown in the following figure.



See Also

Autothreshold	Video and Image Processing Blockset software
Chroma Resampling	Video and Image Processing Blockset software
Color Space Conversion	Video and Image Processing Blockset software
imcomplement	Image Processing Toolbox software

Image Data Type Conversion

Purpose Convert and scale input image to specified output data type

Library Conversions

Description



The Image Data Type Conversion block changes the data type of the input to the user-specified data type and scales the values to the new data type's dynamic range. To convert between data types without scaling, use the Simulink Data Type Conversion block.

When converting between floating-point data types, the block casts the input into the output data type and clips values outside the range to 0 or 1. When converting to the Boolean data type, the block maps 0 values to 0 and all other values to one. When converting to or between all other data types, the block casts the input into the output data type and scales the data type values into the dynamic range of the output data type. For double- and single-precision floating-point data types, the dynamic range is between 0 and 1. For fixed-point data types, the dynamic range is between the minimum and maximum values that can be represented by the data type.

Note This block supports intensity and color images on its ports.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point (word length less than or equal to 16)• Boolean• 8-, 16-bit signed integer• 8-, 16-bit unsigned integer	No

Image Data Type Conversion

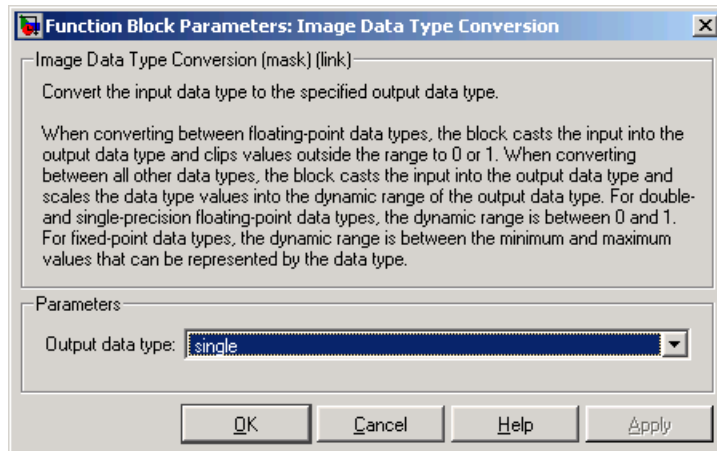
Port	Input/Output	Supported Data Types	Complex Values Supported
Output	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	Same as Input port	No

The dimensions, complexity, and frame status of the input and output signals are the same.

Use the **Output data type** parameter to specify the data type of your output signal values.

Dialog Box

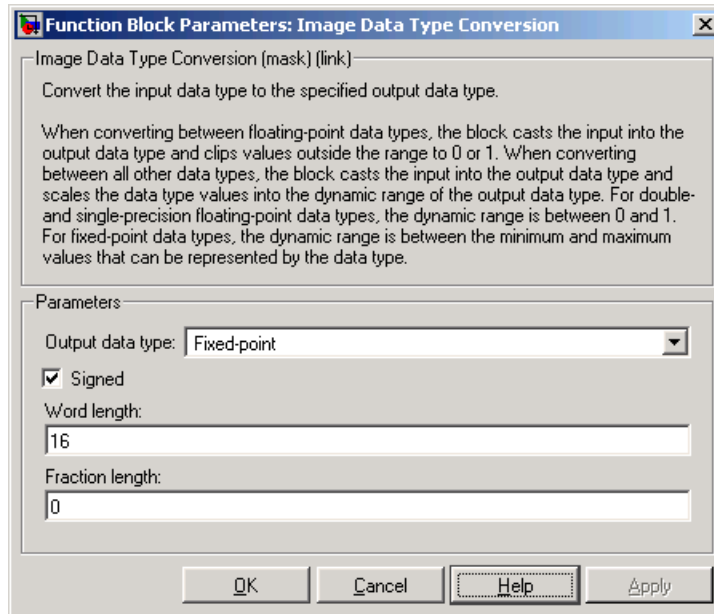
The Image Data Type Conversion dialog box appears as shown in the following figure.



Output data type

Use this parameter to specify the data type of your output signal.

Image Data Type Conversion



Signed

Select this check box if you want the output fixed-point data to be signed. This parameter is visible if, for the **Output data type** parameter, you choose **Fixed-point**.

Word length

Use this parameter to specify the word length of your fixed-point output. This parameter is visible if, for the **Output data type** parameter, you choose **Fixed-point**.

Fraction length

Use this parameter to specify the fraction length of your fixed-point output. This parameter is visible if, for the **Output data type** parameter, you choose **Fixed-point**.

See Also

Autothreshold

Video and Image Processing Blockset
software

Image From File

Purpose Import image from image file

Library Sources

Description



Use the Image From File block to import an image from a supported image file. For a list of supported file formats, see the `imread` function reference page in the MATLAB documentation. If the image is a M-by-N array, the block outputs a binary or intensity image, where M and N are the number of rows and columns in the image. If the image is a M-by-N-by-P array, the block outputs a color image, where M and N are the number of rows and columns in each color plane, P.

Port	Output	Supported Data Types	Complex Values Supported
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	Yes
R, G, B	Scalar, vector, or matrix that represents one plane of the input RGB video stream. Outputs from the R, G, or B ports have the same dimensions.	Same as I port	Yes

For the Video and Image Processing Blockset blocks to display video data properly, double- and single-precision floating-point pixel values must be between 0 and 1. If the input pixel values have a different data type than the one you select using the **Output data type** parameter,

the block scales the pixel values, adds an offset to the pixel values so that they are within the dynamic range of their new data type, or both.

Use the **File name** parameter to specify the name of the graphics file that contains the image to import into the Simulink modeling and simulation software. If the file is not on the MATLAB path, use the **Browse** button to locate the file. This parameter supports URL paths.

Use the **Sample time** parameter to set the sample period of the output signal.

Use the **Image signal** parameter to specify how the block outputs a color video signal. If you select **One multidimensional signal**, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port outputs one M-by-N plane of an RGB video stream.

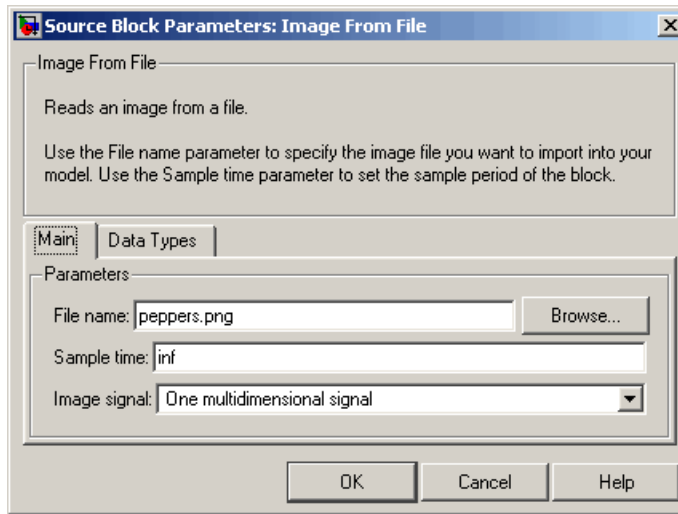
Use the **Output port labels** parameter to label your output ports. Use the spacer character, |, as the delimiter. This parameter is visible if you set the **Image signal** parameter to **Separate color signals**.

On the **Data Types** pane, use the **Output data type** parameter to specify the data type of your output signal.

Image From File

Dialog Box

The **Main** pane of the Image From File dialog box appears as shown in the following figure.



File name

Specify the name of the graphics file that contains the image to import into the Simulink environment.

Sample time

Enter the sample period of the output signal.

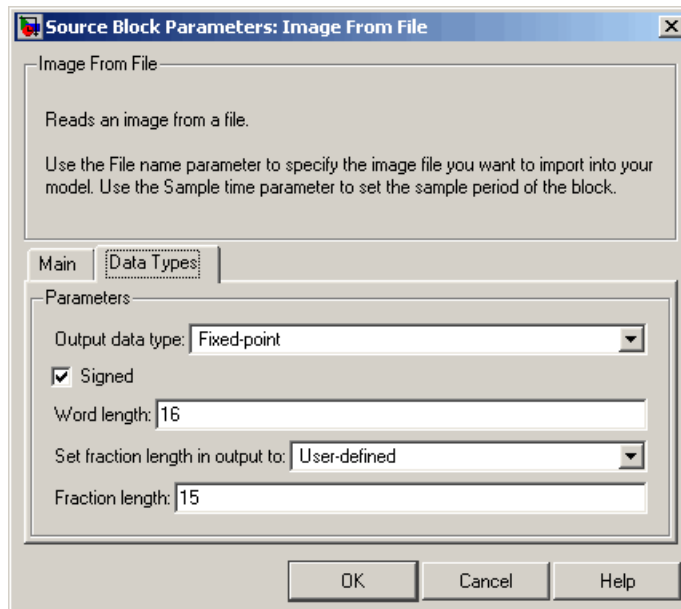
Image signal

Specify how the block outputs a color video signal. If you select **One multidimensional signal**, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port outputs one M-by-N plane of an RGB video stream.

Output port labels

Enter the labels for your output ports using the spacer character, |, as the delimiter. This parameter is visible if you set the **Image signal** parameter to Separate color signals.

The **Data Types** pane of the Image From File dialog box appears as shown in the following figure.



Output data type

Specify the data type of your output signal.

Signed

Select to output a signed fixed-point signal. Otherwise, the signal will be unsigned. This parameter is only visible if, from the **Output data type** list, you select Fixed-point.

Word length

Specify the word length, in bits, of the fixed-point output data type. This parameter is only visible if, from the **Output data type** list, you select **Fixed-point**.

Set fraction length in output to

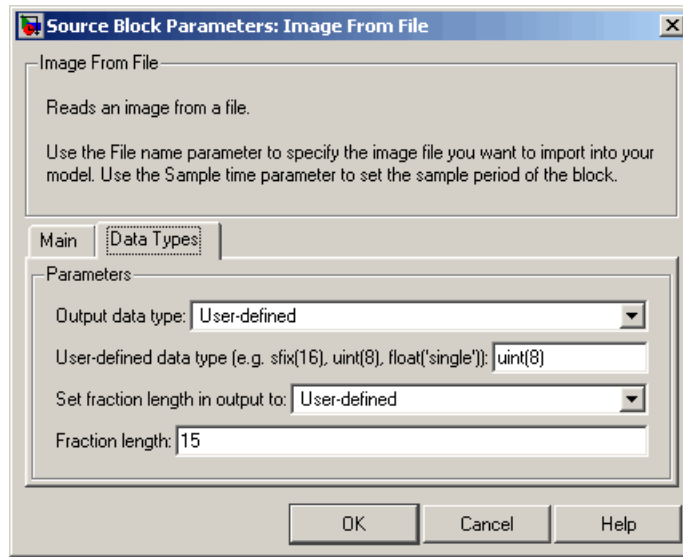
Specify the scaling of the fixed-point output by either of the following two methods:

- Choose **Best precision** to have the output scaling automatically set such that the output signal has the best possible precision.
- Choose **User-defined** to specify the output scaling in the **Fraction length** parameter.

This parameter is only visible if, from the **Output data type** list, you select **Fixed-point** or when you select **User-defined**.

Fraction length

For fixed-point output data types, specify the number of fractional bits, or bits to the right of the binary point. This parameter is only visible when you select **Fixed-point** or **User-defined** for the **Output data type** parameter and **User-defined** for the **Set fraction length in output to** parameter.



User-defined data type

Specify any built-in or fixed-point data type. You can specify fixed-point data types using the `sfixed`, `ufixed`, `sint`, `uint`, `sfrac`, and `ufrac` functions from the Simulink Fixed Point library. This parameter is only visible when you select `User-defined` for the **Output data type** parameter.

See Also

From Multimedia File	Video and Image Processing Blockset software
Image From Workspace	Video and Image Processing Blockset software
To Video Display	Video and Image Processing Blockset software
Video From Workspace	Video and Image Processing Blockset software
Video Viewer	Video and Image Processing Blockset software

Image From File

`im2double`

Image Processing Toolbox software

`im2uint8`

Image Processing Toolbox software

`imread`

MATLAB

Purpose Import image from MATLAB workspace

Library Sources

Description Use the Image From Workspace block to import an image from the MATLAB workspace. If the image is a M-by-N workspace array, the block outputs a binary or intensity image, where M and N are the number of rows and columns in the image. If the image is a M-by-N-by-P workspace array, the block outputs a color image, where M and N are the number of rows and columns in each color plane, P.



Port	Output	Supported Data Types	Complex Values Supported
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	No
R, G, B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B ports have the same dimensions.	Same as I port	No

For the Video and Image Processing Blockset blocks to display video data properly, double- and single-precision floating-point pixel values must be between 0 and 1. If the input pixel values have a different data type than the one you select using the **Output data type** parameter, the block scales the pixel values, adds an offset to the pixel values so that they are within the dynamic range of their new data type, or both.

Image From Workspace

Use the **Value** parameter to specify the MATLAB workspace variable that contains the image you want to import into Simulink environment.

Use the **Sample time** parameter to set the sample period of the output signal.

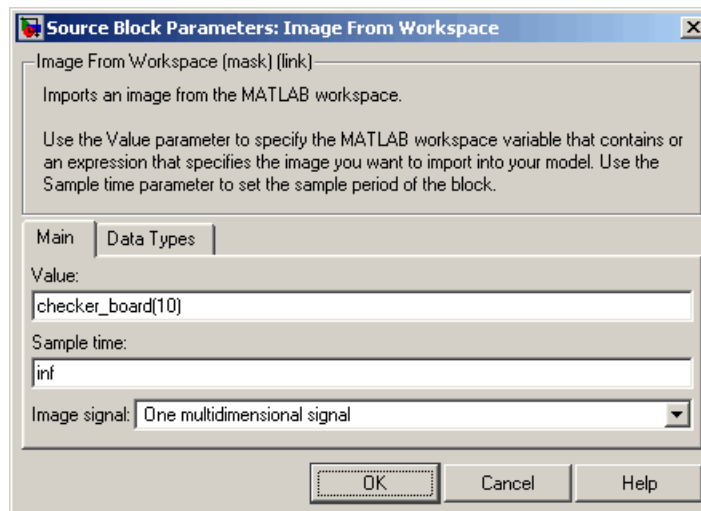
Use the **Image signal** parameter to specify how the block outputs a color video signal. If you select **One multidimensional signal**, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port outputs one M-by-N plane of an RGB video stream.

Use the **Output port labels** parameter to label your output ports. Use the spacer character, |, as the delimiter. This parameter is visible if you set the **Image signal** parameter to **Separate color signals**.

On the **Data Types** pane, use the **Output data type** parameter to specify the data type of your output signal.

Dialog Box

The **Main** pane of the Image From Workspace dialog box appears as shown in the following figure.



Value

Specify the MATLAB workspace variable that you want to import into Simulink environment.

Sample time

Enter the sample period of the output signal.

Image signal

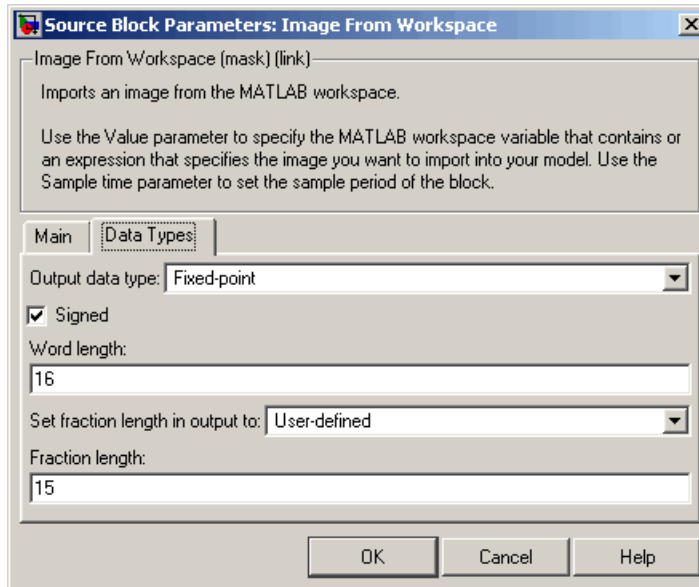
Specify how the block outputs a color video signal. If you select **One multidimensional signal**, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port outputs one M-by-N plane of an RGB video stream.

Output port labels

Enter the labels for your output ports using the spacer character, |, as the delimiter. This parameter is visible if you set the **Image signal** parameter to **Separate color signals**.

The **Data Types** pane of the Image From Workspace dialog box appears as shown in the following figure.

Image From Workspace



Output data type

Specify the data type of your output signal.

Signed

Select to output a signed fixed-point signal. Otherwise, the signal is unsigned. This parameter is only visible if, from the **Output data type** list, you select Fixed-point.

Word length

Specify the word length, in bits, of the fixed-point output data type. This parameter is only visible if, from the **Output data type** list, you select Fixed-point.

Set fraction length in output to

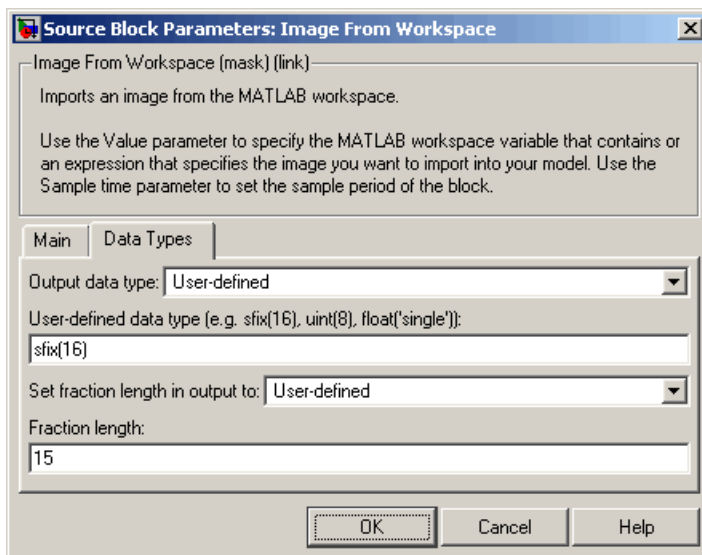
Specify the scaling of the fixed-point output by either of the following two methods:

- Choose **Best** precision to have the output scaling automatically set such that the output signal has the best possible precision.
- Choose **User-defined** to specify the output scaling in the **Fraction length** parameter.

This parameter is only visible if, from the **Output data type** list, you select **Fixed-point** or when you select **User-defined**.

Fraction length

For fixed-point output data types, specify the number of fractional bits, or bits to the right of the binary point. This parameter is only visible when you select **Fixed-point** or **User-defined** for the **Output data type** parameter and **User-defined** for the **Set fraction length in output to** parameter.



User-defined data type

Specify any built-in or fixed-point data type. You can specify fixed-point data types using the `sfix`, `ufix`, `sint`, `uint`, `sfrac`,

Image From Workspace

and `ufrac` functions from the Simulink Fixed Point library. This parameter is only visible when you select `User-defined` for the **Output data type** parameter.

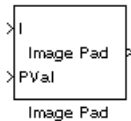
See Also

From Multimedia File	Video and Image Processing Blockset software
To Video Display	Video and Image Processing Blockset software
Video From Workspace	Video and Image Processing Blockset software
Video Viewer	Video and Image Processing Blockset software
<code>im2double</code>	Image Processing Toolbox software
<code>im2uint8</code>	Image Processing Toolbox software

Purpose Pad signal along its rows, columns, or both

Library Utilities

Description The Image Pad block expands or crops the dimensions of a signal by padding or truncating its rows, columns, or both.



Port	Input/Output	Supported Data Types	Complex Values Supported
Image / I	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	Yes
PVal	Scalar value that represents the constant pad value	Same as I port	Yes
Output	Padded scalar, vector, or matrix	Same as I port	Yes

The data type of the input signal is the data type of the output signal.

Use the **Method** parameter to specify how you pad the input signal.

- **Constant** — Pad with a constant value

Image Pad

- **Replicate** — Pad by repeating its border values
- **Symmetric** — Pad with its mirror image
- **Circular** — Pad using a circular repetition of its elements

If you set the **Method** parameter to **Constant**, the **Pad value source** parameter appears on the dialog box.

- **Input port** — The PVal port appears on the block. Use this port to specify the constant value with which to pad your signal
- **Specify via dialog** — The **Pad value** parameter appears in the dialog box. Enter the constant value with which to pad your signal.

Setting the Specify Parameter to Pad size

If you set the **Specify** parameter to **Pad size**, you can enter the size of the padding in the horizontal and vertical directions.

The **Pad rows at** parameter controls the padding at the left, right or both sides of the input signal.

- **Left** — The block adds additional columns on the left side.
- **Right** — The block adds additional columns on the right side.
- **Both left and right** — The block adds additional columns to the left and right side.
- **No padding** — The block does not change the number of columns.

Use the **Pad size along rows** parameter to specify the size of the padding in the horizontal direction. Enter a scalar value, and the block adds this number of columns to the left, right, or both sides of your input signal. If you set the **Pad rows at** parameter to **Both left and right**, you can enter a two element vector. The left element controls the number of columns the block adds to the left side of the signal; the right element controls the number of columns the block adds to the right side of the signal.

The **Pad columns at** parameter controls the padding at the top and bottom of the input signal.

- **Top** — The block adds additional rows to the top.
- **Bottom** — The block adds additional rows to the bottom.
- **Both top and bottom** — The block adds additional rows to the top and bottom.
- **No padding** — The block does not change the number of rows.

Use the **Pad size along columns** parameter to specify the size of the padding in the vertical direction. Enter a scalar value, and the block adds this number of rows to the top, bottom, or both of your input signal. If you set the **Pad columns at** parameter to **Both top and bottom**, you can enter a two element vector. The left element controls the number of rows the block adds to the top of the signal; the right element controls the number of rows the block adds to the bottom of the signal.

Setting the Specify Parameter to Output size

If, for the **Specify** parameter, you select **Output size**, you can enter the total number of output columns and rows. This setting enables you to pad or truncate the input signal. See the previous section for descriptions of the **Pad rows at** and **Pad columns at** parameters. If you are using the Image Pad block to truncate the input signal, these parameters control where the signal is truncated.

If **Pad rows at** parameter is set to **Both left and right**, the block splits the padding or truncation evenly. If an even split is not possible, the block adds or removes elements from the end of the rows. The block behaves similarly if the **Pad columns at** parameter is set to **Both top and bottom**.

Use the **Output row mode** parameter to describe how to pad the input signal.

- **User-specified** — Use the **Row size** parameter to specify the total number of rows.

Image Pad

- **Next power of two** — The block pads the input signal along the rows until the length of the rows is equal to a power of two. When the length of the input signal's rows is equal to a power of two, the block does not pad the input signal's rows.

Use the **Output column mode** parameter to describe how to pad the input signal.

- **User-specified** — Use the **Column size** parameter to specify the total number of columns.
- **Next power of two** — The block pads the input signal along the columns until the length of the columns is equal to a power of two. When the length of the input signal's columns is equal to a power of two, the block does not pad the input signal's columns.

The following options are available for the **Action when truncation occurs** parameter:

- **None** — Select this option when you do not want to be notified that the input signal is truncated.
- **Warning** — Select this option when you want to receive a warning in the MATLAB Command Window when the input signal is truncated.
- **Error** — Select this option when you want an error dialog box displayed and the simulation terminated when the input signal is truncated.

Examples

The following four examples demonstrate the four different padding methods:

- “Example 1” on page 2-449 — Demonstrates the block's behavior when the **Method** parameter is set to **Constant**.
- “Example 2” on page 2-450— Demonstrates the block's behavior when the **Method** parameter is set to **Replicate**.

- “Example 3” on page 2-451— Demonstrates the block’s behavior when the **Method** parameter is set to **Symmetric**.
- “Example 4” on page 2-452— Demonstrates the block’s behavior when the **Method** parameter is set to **Circular**.

Example 1

Suppose you want to pad the rows of your input signal with three initial values equal to 0 and your input signal is defined as follows:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}$$

Set the Image Pad block parameters as follows:

- **Method** = Constant
- **Pad value source** = Specify via dialog
- **Pad value** = 0
- **Specify** = Output size
- **Pad rows at** = Left
- **Output row mode** = User-specified
- **Row size** = 6
- **Pad columns at** = No padding

The Image Pad block outputs the following signal:

Image Pad

$$\begin{bmatrix} 0 & 0 & 0 & a_{00} & a_{01} & a_{02} \\ 0 & 0 & 0 & a_{10} & a_{11} & a_{12} \\ 0 & 0 & 0 & a_{20} & a_{21} & a_{22} \end{bmatrix}$$

Example 2

Suppose you want to pad your input signal with its border values, and your input signal is defined as follows:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}$$

Set the Image Pad block parameters as follows:

- **Method** = Replicate
- **Specify** = Pad size
- **Pad rows at** = Both left and right
- **Pad size along rows** = 2
- **Pad columns at** = Both top and bottom
- **Pad size along columns** = [1 3]

The Image Pad block outputs the following signal:

$$\begin{bmatrix}
 a_{00} & a_{00} & a_{00} & a_{01} & a_{02} & a_{02} & a_{02} \\
 a_{00} & a_{00} & a_{00} & a_{01} & a_{02} & a_{02} & a_{02} \\
 a_{10} & a_{10} & a_{10} & a_{11} & a_{12} & a_{12} & a_{12} \\
 a_{20} & a_{20} & a_{20} & a_{21} & a_{22} & a_{22} & a_{22} \\
 a_{20} & a_{20} & a_{20} & a_{21} & a_{22} & a_{22} & a_{22} \\
 a_{20} & a_{20} & a_{20} & a_{21} & a_{22} & a_{22} & a_{22} \\
 a_{20} & a_{20} & a_{20} & a_{21} & a_{22} & a_{22} & a_{22}
 \end{bmatrix}$$

Input matrix

The border values of the input signal are replicated on the top, bottom, left, and right of the input signal so that the output is a 7-by-7 matrix. The values in the corners of this output matrix are determined by replicating the border values of the matrices on the top, bottom, left and right side of the original input signal.

Example 3

Suppose you want to pad your input signal using its mirror image, and your input signal is defined as follows:

$$\begin{bmatrix}
 a_{00} & a_{01} & a_{02} \\
 a_{10} & a_{11} & a_{12} \\
 a_{20} & a_{21} & a_{22}
 \end{bmatrix}$$

Set the Image Pad block parameters as follows:

- **Method** = Symmetric
- **Specify** = Pad size
- **Pad rows at** = Both left and right

Image Pad

- Pad size along rows = [5 6]
- Pad columns at = Both top and bottom
- Pad size along columns = 2

The Image Pad block outputs the following signal:

$$\begin{bmatrix}
 a_{11} & a_{12} & a_{12} & a_{11} & a_{10} & a_{10} & a_{11} & a_{12} & a_{12} & a_{11} & a_{10} & a_{10} & a_{11} & a_{12} \\
 a_{01} & a_{02} & a_{02} & a_{01} & a_{00} & a_{00} & a_{01} & a_{02} & a_{02} & a_{01} & a_{00} & a_{00} & a_{01} & a_{02} \\
 a_{01} & a_{02} & a_{02} & a_{01} & a_{00} & a_{00} & a_{01} & a_{02} & a_{02} & a_{01} & a_{00} & a_{00} & a_{01} & a_{02} \\
 a_{11} & a_{12} & a_{12} & a_{11} & a_{01} & a_{10} & a_{11} & a_{12} & a_{12} & a_{11} & a_{10} & a_{10} & a_{11} & a_{12} \\
 a_{21} & a_{22} & a_{22} & a_{21} & a_{20} & a_{20} & a_{21} & a_{22} & a_{22} & a_{21} & a_{20} & a_{20} & a_{21} & a_{22} \\
 a_{21} & a_{22} & a_{22} & a_{21} & a_{20} & a_{20} & a_{21} & a_{22} & a_{22} & a_{21} & a_{20} & a_{20} & a_{21} & a_{22} \\
 a_{11} & a_{12} & a_{12} & a_{11} & a_{01} & a_{01} & a_{11} & a_{12} & a_{12} & a_{11} & a_{10} & a_{10} & a_{11} & a_{12}
 \end{bmatrix}$$

Input matrix

The block flips the original input matrix and each matrix it creates about their top, bottom, left, and right sides to populate the 7-by-13 output signal. For example, in the preceding figure, you can see how the block flips the input matrix about its right side to create the matrix directly to its right.

Example 4

Suppose you want to pad your input signal using a circular repetition of its values. Your input signal is defined as follows:

$$\begin{bmatrix}
 a_{00} & a_{01} & a_{02} \\
 a_{10} & a_{11} & a_{12} \\
 a_{20} & a_{21} & a_{22}
 \end{bmatrix}$$

Set the Image Pad block parameters as follows:

- **Method** = Circular
- **Specify** = Output size
- **Pad rows at** = Both left and right
- **Output row mode** = User-specified
- **Row size** = 9
- **Pad columns at** = Both top and bottom
- **Output column mode** = User-specified
- **Column size** = 9

The Image Pad block outputs the following signal:

$$\begin{array}{c}
 \left[\begin{array}{ccc|ccc|ccc}
 a_{00} & a_{01} & a_{02} & a_{00} & a_{01} & a_{02} & a_{00} & a_{01} & a_{02} \\
 a_{10} & a_{11} & a_{12} & a_{10} & a_{11} & a_{12} & a_{10} & a_{11} & a_{12} \\
 a_{20} & a_{21} & a_{22} & a_{20} & a_{21} & a_{22} & a_{20} & a_{21} & a_{22} \\
 \hline
 a_{00} & a_{01} & a_{02} & a_{00} & a_{01} & a_{02} & a_{00} & a_{01} & a_{02} \\
 a_{10} & a_{11} & a_{12} & a_{10} & a_{11} & a_{12} & a_{10} & a_{11} & a_{12} \\
 a_{20} & a_{21} & a_{22} & a_{20} & a_{21} & a_{22} & a_{20} & a_{21} & a_{22} \\
 \hline
 a_{00} & a_{01} & a_{02} & a_{00} & a_{01} & a_{02} & a_{00} & a_{01} & a_{02} \\
 a_{10} & a_{11} & a_{12} & a_{10} & a_{11} & a_{12} & a_{10} & a_{11} & a_{12} \\
 a_{20} & a_{21} & a_{22} & a_{20} & a_{21} & a_{22} & a_{20} & a_{21} & a_{22}
 \end{array} \right]
 \end{array}$$

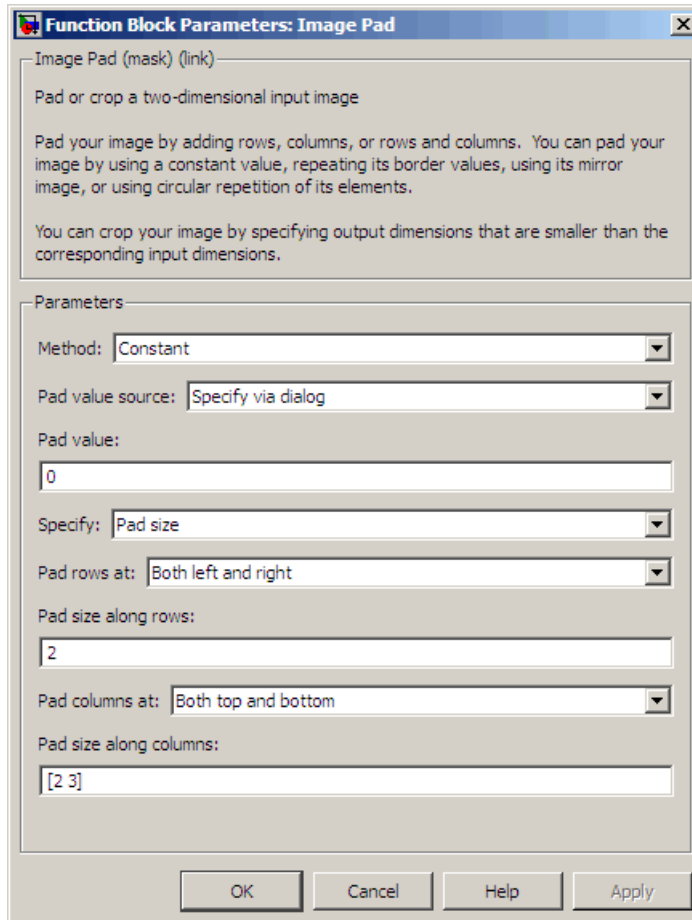
Input matrix

The block repeats the values of the input signal in a circular pattern to populate the 9-by-9 output matrix.

Image Pad

Dialog Box

The Image Pad dialog box appears as shown in the following figure.



Method

Specify how you want the block to pad your signal.

Pad value source

If you select **Input port**, the **PVal** port appears on the block. Use this port to specify the constant value with which to pad your signal. If you select **Specify via dialog**, the **Pad value** parameter becomes available. This parameter is visible if, for the **Method** parameter, you select **Constant**.

Pad value

Enter the constant value with which to pad your signal. This parameter is visible if, for the **Pad value source** parameter, you select **Specify via dialog**. This parameter is tunable.

Specify

If you select **Pad size**, you can enter the size of the padding in the horizontal and vertical directions. If you select **Output size**, you can enter the total number of output columns and rows.

Pad rows at

Select **Left** to add additional columns to the left side of the signal. Select **Right** to add additional columns to the right side of the signal. Select **Both left and right** to add additional columns to the left and right side of the signal. If you select **No padding**, the block does not change the number of columns of the input signal.

Pad size along rows

This parameter controls how many columns are added to the right and/or left side of your input signal. Enter a scalar value, and the block adds this number of columns to the left, right, or both sides of your signal. If, for the **Pad rows at** parameter you selected **Both left and right**, enter a two-element vector. The left element controls the number of columns the block adds to the left side of the signal and the right element controls how many columns the block adds to the right side of the signal. This parameter is visible if, for the **Specify** parameter, you select **Pad size**.

Output row mode

Describe how to pad the input signal. If you select **User-specified**, the **Row size** parameter appears on the block

dialog box. If you select `Next power of two`, the block pads the input signal along the rows until the length of the rows is equal to a power of two. This parameter is visible if, for the **Specify** parameter, you select `Output size`.

Row size

Enter a scalar value that represents the total number of output columns. This parameter is visible if you set the **Output row mode** parameter to `User-specified`.

Pad columns at

Select `Top` to add additional rows at the top of the input signal. Select `Bottom` to add additional rows at the bottom of the signal. Select `Both top and bottom` to add additional rows at the top and bottom of the signal. If you select `No padding`, the block does not change the number of rows of the input signal.

Pad size along columns

This parameter controls how many rows are added to the top, bottom, or both of your input signal. Enter a scalar value and the block adds this number of columns to the top, bottom, or both of your signal. If, for the **Pad columns at** parameter you selected `Both top and bottom`, enter a two-element vector. The left element controls the number of rows the block adds to the top of the signal and the right element controls how many rows the block adds to the bottom of the signal. This parameter is visible if you set the **Specify** parameter to `Pad size`.

Output column mode

Describe how to pad the input signal. If you select `User-specified`, the **Column size** parameter appears on the block dialog box. If you select `Next power of two`, the block pads the input signal along the columns until the length of the columns is equal to a power of two. This parameter is visible if, for the **Specify** parameter, you select `Output size`.

Column size

Enter a scalar value that represents the total number of output columns. This parameter is visible if you set the **Output column mode** parameter to **User-specified**.

Action when truncation occurs

Choose **None** when you do not want to be notified that the input signal is truncated. Select **Warning** to display a warning when the input signal is truncated. Choose **Error** when you want an error dialog box displayed and the simulation terminated when the input signal is truncated.

Insert Text

Purpose Draw text on image or video stream.

Library Text & Graphics

Description



The Insert Text block draws formatted text or numbers on an image or video stream. The block uses the FreeType 2.3.5 library, an open-source font engine, to produce stylized text bitmaps. To learn more about the FreeType Project, visit <http://www.freetype.org/>. The Insert Text block does not support character sets other than ASCII.

The Insert Text block lets you draw one or more instances of one or more strings, including:

- A single instance of one text string
- Multiple instances of one text string
- Multiple instances of text, with a different text string at each location

Input/Output	Description
Image	M -by- N matrix of intensity values or an M -by- N -by- P color video signal where P is the number of color planes.
R, G, B	Matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B ports have the same dimensions and data type.
Select	Zero-based index value that indicates which text string to display.
Variables	Vector or matrix whose values are used to replace ANSI C printf-style format specifications.
Color	Intensity input — Scalar value used for all strings or 1-by- N vector of intensity values whose length is equal to the number of strings. Color input — Three-element vector that specifies one color for all strings or a 3-by- N matrix of color values, where N is the number of strings.

Input/Output	Description
Location	2-by- N matrix, where N is the number of text strings to insert, that specifies the top-left corner of the text string bounding boxes.
Opacity	Scalar value that is used for all strings or vector of opacity values whose length is equal to the number of strings.

Use the **Text** parameter to specify the text string to be drawn on the image or video frame. This parameter can be a single text string, such as 'Figure1', a cell array of strings, such as {'Figure1', 'Figure2'}, or an ANSI C printf-style format specifications, such as %s.

If, for the **Text** parameter, you enter a cell array of strings, the Insert Text block does not display all of the strings simultaneously. Instead, the **Select** port appears on the block to let you indicate which text string to display. The input to this port must be a scalar value, where 0 indicates the first string. If the input is less than 0 or greater than one less than the number of strings in the cell array, the block does not draw text on the image or video frame.

If, for the **Text** parameter, you enter ANSI C printf-style format specifications, such as %d, %f, or %s, the **Variables** port appears on the block. The block replaces the format specifications in the **Text** parameter with each element of the input vector in turn. Use the %s option to specify a set of text strings for the block to display simultaneously at different locations. For example, using a Constant block, enter [uint8('Text1') 0 uint8('Text2')] for the **Constant value** parameter. The following table summarizes the supported conversion specifications.

Insert Text

Text Parameter Supported Conversion Specifications

Supported specifications	Support for multiple instances of the same specification?	Support for mixed specifications?
%d, %i, %u, %c, %f, %o, %x, %X, %e, %E, %g, and %G	Yes	No
%s	No	No

Use the **Location source** parameter to indicate where to specify the text location:

- **Specify via dialog** — the **Location [row column]** parameter appears on the dialog box.
- **Input port** — the Location port appears on the block.

The following table describes how to format the location of the text strings depending on the number of strings you specify to insert. You can specify more than one location regardless of how many text strings you specify, but the only way to get a different text string at each location is to use the %s option for the **Text** parameter to specify a set of text strings. You can enter negative values or values that exceed the dimensions of the input image or video frame, but the text might not be visible.

Location Parameter Text String Insertion

Parameter	One Instance of One Text String	Multiple Instances of the Same Text String	Multiple Instances of Unique Text Strings
Location [row column] parameter setting or the input to the Location port	Two-element vector of the form [<i>row column</i>] that indicates the top-left corner of the text bounding box.	2-by- <i>N</i> matrix, where <i>N</i> is the number of locations at which to display the text string. Each column indicates the row and column coordinate of the top-left corner of the text bounding box for the string, e.g., <code>[[row1 column1]' [row2 column2]']</code>	2-by- <i>N</i> matrix, where <i>N</i> is the number of text strings. Each column indicates the row and column coordinate of the top-left corner of the text bounding box for the corresponding string, e.g., <code>[[row1 column1]' [row2 column2]']</code> .

Use the **Color value source** parameter to indicate where to specify the text color:

- Specify via dialog — the **Color value** parameter appears on the dialog box.
- Input port — the Color port appears on the block.

The following table describes how to format the color of the text strings depending on the block input and the number of strings you want to insert. If the input image is a floating-point data type, the color values must be between 0 and 1. If the input image is an 8-bit unsigned integer data type, the color values must range between 0 and 255.

Text String Color Values

Block Input	One Text String	Multiple Text Strings
Intensity image	Color value parameter or the input to the Color port = Scalar intensity value	Color value parameter or the input to the Color port = Vector of intensity values whose length is equal to the number of strings
Color image	Color value parameter or the input to the Color port = RGB triplet that specifies the color of the text	Color value parameter or the input to the Color port = 3-by- N matrix of color values, where N is the number of strings

Use the **Opacity source** parameter to indicate where to specify the text's opacity:

- **Specify via dialog** — the **Opacity** parameter appears on the dialog box.
- **Input port** — the Opacity port appears on the block.

The following table describes how to format the opacity of the text strings depending on the number of strings you want to insert.

Text String Opacity Values

Parameter	One Text String	Multiple Text Strings
Opacity parameter setting or the input to the Opacity port	Scalar value between 0 and 1, where 0 is translucent and 1 is opaque	Vector whose length is equal to the number of strings

Use the **Image signal** parameter to specify how to input and output a color video signal:

- **One multidimensional signal** — the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port.
- **Separate color signals** — additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

Use the **Font face** parameter to specify the font of your text. The block populates this list with the TrueType fonts installed on your system. On Windows, the block searches the system registry for font files. On UNIX, the block searches the X Server's font path for font files.

Use the **Font size (points)** parameter to specify the font size.

If you select the **Anti-aliased** check box, the block smooths the edges of the text, which can be computationally expensive. If you want your model to run faster, clear this check box.

Row-Major Data Format

MATLAB and the Video and Image Processing Blockset blocks use column-major data organization. However, the Insert Text block gives you the option to process data that is stored in row-major format. When you select the **Input image is transposed (data order is row major)** check box, the block assumes that the input buffer contains contiguous data elements from the first row first, then data elements from the second row second, and so on through the last row. Use this functionality only when you meet all the following criteria:

Insert Text

- You are developing algorithms to run on an embedded target that uses the row-major format.
- You want to limit the additional processing required to take the transpose of signals at the interfaces of the row-major and column-major systems.

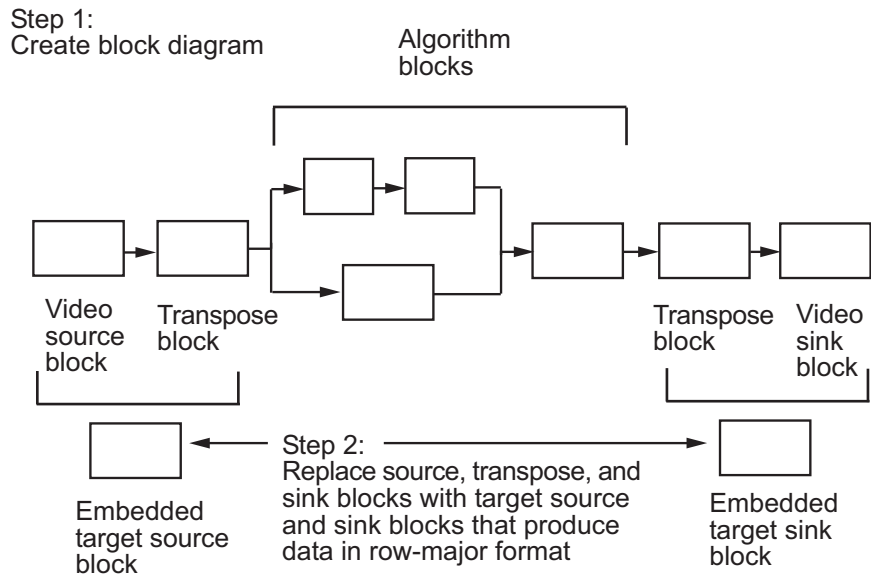
When you use the row-major functionality, you must consider the following issues:

- When you select this check box, the first two signal dimensions of the Insert Text block's input are swapped.
- All Video and Image Processing Blockset software blocks can be used to process data that is in the row-major format, but you need to know the image dimensions when you develop your algorithms.

For example, if you use the 2-D FIR Filter block, you need to verify that your filter coefficients are transposed. If you are using the Rotate block, you need to use negative rotation angles, etc.

- Only three blocks have the **Input image is transposed (data order is row major)** check box. They are the Chroma Resampling, Deinterlacing, and Insert Text blocks. You need to select this check box to enable row-major functionality in these blocks. All other blocks must be properly configured to process data in row-major format.

Use the following two-step workflow to develop algorithms in row-major format to run on an embedded target.



See the DM642 EVM Video ADC and DM642 EVM Video DAC reference pages in the *Target Support Package User's Guide* for more information about data order in embedded targets.

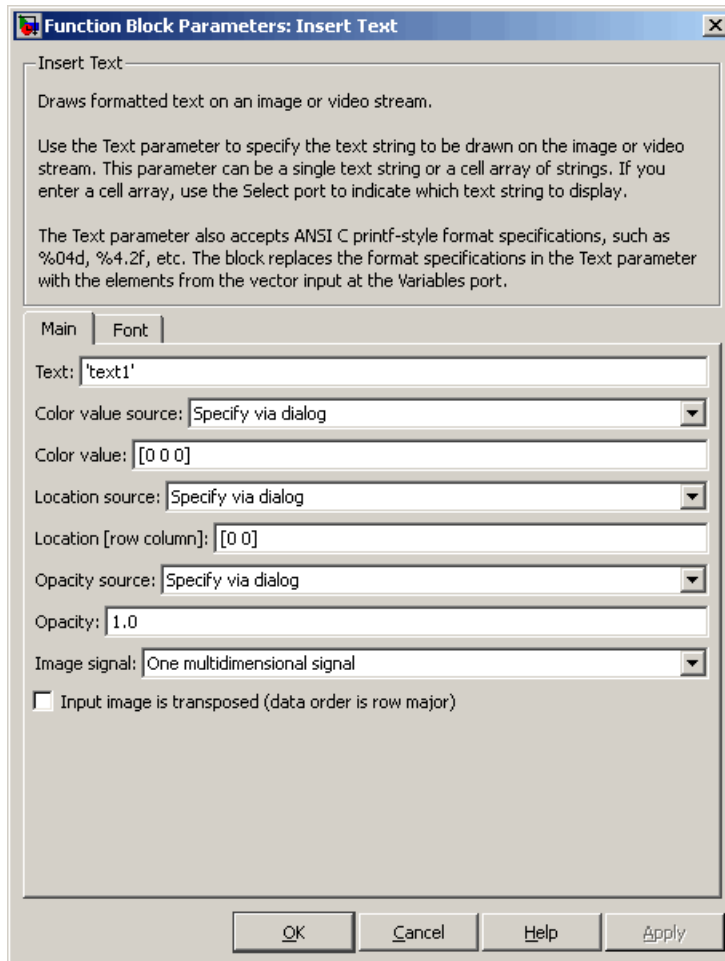
Examples

See “Annotating AVI Files with Video Frame Numbers” and “Annotating AVI Files at Two Separate Locations” in the *Video and Image Processing Blockset User's Guide*. Many of the Video and Image Processing Blockset Demos make use of the Insert Text block.

Insert Text

Dialog Box

The **Main** pane of the Insert Text dialog box appears as shown in the following figure.



Text

Specify the text string to be drawn on the image or video stream. This parameter can be a single text string or a cell array of strings.

To create a **Select** port enter a cell array of strings. To create a **Variables** port, enter ANSI C printf-style format specifications, such as %d, %f, or %s.

Color value source

Indicate where you want to specify the text color. Your choices are Specify via dialog or Input port.

Color value

Specify the intensity or color of the text. This parameter is visible if, for the **Color source** parameter, you select Specify via dialog. Tunable.

Location source

Indicate where you want to specify the text location. Your choices are Specify via dialog or Input port.

Location [row column]

Specify the text location. This parameter is visible if, for the **Location source** parameter, you select Specify via dialog. Tunable.

Opacity source

Indicate where you want to specify the text's opaqueness. Your choices are Specify via dialog or Input port.

Opacity

Specify the opacity of the text. This parameter is visible if, for the **Opacity source** parameter, you select Specify via dialog. Tunable.

Image signal

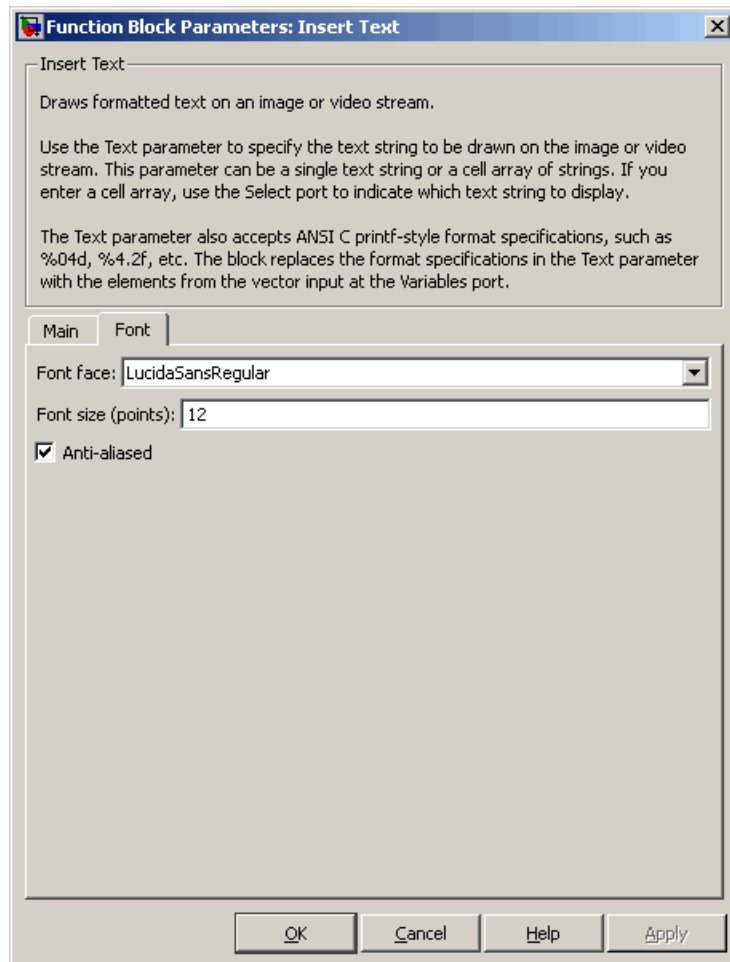
Specify how to input and output a color video signal. If you select **One multidimensional signal**, the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

Insert Text

Input image is transposed (data order is row major)

When you select this check box, the block assumes that the input buffer contains data elements from the first row first, then data elements from the second row second, and so on through the last row.

The **Font** pane of the Insert Text dialog box appears as shown in the following figure.



Font face

Specify the font of your text. The block populates this list with the fonts installed on your system.

Font size (points)

Specify the font size.

Insert Text

Anti-aliased

Select this check box if you want the block to smooth the edges of the text.

Supported Data Types

Port	Supported Data Types
Input/Image	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point(signed, word length less than or equal to 32.)• Boolean• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer
R, G, B	Same as Input port
Select	<ul style="list-style-type: none">• Double-precision floating point. (This data type is only supported if the input to the I or R, G, and B ports is a floating-point data type.)• Single-precision floating point. (This data type is only supported if the input to the I or R, G, and B ports is a floating-point data type.)• Boolean• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer

Port	Supported Data Types
Variables	<p>The data types supported by this port depend on the conversion specification you are using in the Text parameter.</p> <p>%d, %i, and %u:</p> <ul style="list-style-type: none"> • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer <p>%c and %s:</p> <ul style="list-style-type: none"> • 8-bit unsigned integer <p>%f:</p> <ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point <p>%o, %x, %X, %e, %E, %g, and %G:</p> <ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer
Color	<p>Same as Input port (The input to this port must be the same data type as the input to the Input port.)</p>

Insert Text

Port	Supported Data Types
Location	<ul style="list-style-type: none">• Double-precision floating point. (This data type is only supported if the input to the I or R, G, and B ports is a floating-point data type.)• Single-precision floating point. (This data type is only supported if the input to the I or R, G, and B ports is a floating-point data type.)• Boolean• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer
Opacity	<ul style="list-style-type: none">• Double-precision floating point. (This data type is only supported if the input to the Input or R, G, and B ports is a double-precision floating-point data type.)• Single-precision floating point. (This data type is only supported if the input to the I or R, G, and B ports is a single-precision floating-point data type.)• <code>ufix8_En7</code> (This data type is only supported if the input to the I or R, G, and B ports is a fixed-point data type.)

See Also

Draw Shapes

Video and Image Processing Blockset
software

Purpose	Predict or estimate states of dynamic systems
Library	Filtering
Description	The Kalman Filter block is a Signal Processing Blockset block. For more information, see the Kalman Filter block reference page in the Signal Processing Blockset software documentation.

Label

Purpose Label connected components in binary images

Library Morphological Operations

Description

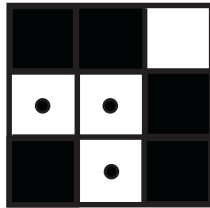


The Label block labels the objects in a binary image, BW, where the background is represented by pixels equal to 0 (black) and objects are represented by pixels equal to 1 (white). At the Label port, the block outputs a label matrix that is the same size as the input matrix. In the label matrix, pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on. At the Count port, the block outputs a scalar value that represents the number of labeled objects.

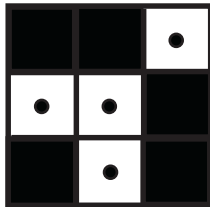
Port	Input/Output	Supported Data Types	Complex Values Supported
BW	Vector or matrix that represents a binary image	Boolean	No
Label	Label matrix	<ul style="list-style-type: none">8-, 16-, and 32-bit unsigned integer	No
Count	Scalar that represents the number of labeled objects	Same as Label port	No

Use the **Connectivity** parameter to define which pixels are connected to each other. If you want a pixel to be connected to the other pixels located on the top, bottom, left, and right, select 4. If you want a pixel to be connected to the other pixels on the top, bottom, left, right, and diagonally, select 8.

Consider the following 3-by-3 image. If, for the **Connectivity** parameter, you select 4, the block considers the white pixels marked by black circles to be connected.



If, for the **Connectivity** parameter, you select 8, the block considers the white pixels marked by black circles to be connected.



Use the **Output** parameter to determine the block's output. If you select `Label matrix` and `number of labels`, ports `Label` and `Count` appear on the block. The block outputs the label matrix at the `Label` port and the number of labeled objects at the `Count` port. If you select `Label matrix`, the `Label` port appears on the block. If you select `Number of labels`, the `Count` port appears on the block.

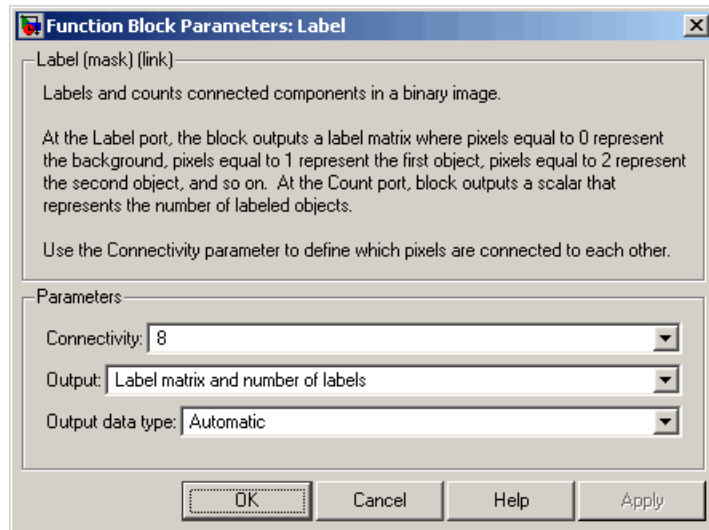
Use the **Output data type** parameter to set the data type of the outputs at the `Label` and `Count` ports. If you select `Automatic`, the block calculates the maximum number of objects that can fit inside the image based on the image size and the connectivity you specified. Based on this calculation, it determines the minimum output data type size that guarantees unique region labels and sets the output data type appropriately. If you select `uint32`, `uint16`, or `uint8`, the data type of the output is 32-, 16-, or 8-bit unsigned integers, respectively. If you select `uint16`, or `uint8`, the **If label exceeds data type size, mark remaining regions using** parameter appears in the dialog box. If the number of found objects exceeds the maximum number that can be represented by the output data type, use this parameter to specify the

Label

block's behavior. If you select **Maximum value of the output data type**, the remaining regions are labeled with the maximum value of the output data type. If you select **Zero**, the remaining regions are labeled with zeroes.

Dialog Box

The Label dialog box appears as shown in the following figure.



Connectivity

Specify which pixels are connected to each other. If you want a pixel to be connected to the pixels on the top, bottom, left, and right, select 4. If you want a pixel to be connected to the pixels on the top, bottom, left, right, and diagonally, select 8.

Output

Determine the block's output. If you select **Label matrix and number of labels**, the Label and Count ports appear on the block. The block outputs the label matrix at the Label port and the number of labeled objects at the Count port. If you select

Label matrix, the Label port appears on the block. If you select Number of labels, the Count port appears on the block.

Output data type

Set the data type of the outputs at the Label and Count ports. If you select Automatic, the block determines the appropriate data type for the output. If you select uint32, uint16, or uint8, the data type of the output is 32-, 16-, or 8-bit unsigned integers, respectively.

If label exceeds data type size, mark remaining regions using

Use this parameter to specify the block's behavior if the number of found objects exceeds the maximum number that can be represented by the output data type. If you select Maximum value of the output data type, the remaining regions are labeled with the maximum value of the output data type. If you select Zero, the remaining regions are labeled with zeroes. This parameter is visible if, for the **Output data type** parameter, you choose uint16 or uint8.

See Also

Bottom-hat	Video and Image Processing Blockset software
Closing	Video and Image Processing Blockset software
Dilation	Video and Image Processing Blockset software
Erosion	Video and Image Processing Blockset software
Opening	Video and Image Processing Blockset software
Top-hat	Video and Image Processing Blockset software

Label

`bwlabel`

Image Processing Toolbox software

`bwlabeln`

Image Processing Toolbox software

Purpose Find maximum values in input or sequence of inputs

Library Statistics

Description The Maximum block is a Signal Processing Blockset block. For more information, see the Maximum block reference page in the Signal Processing Blockset software documentation.

Mean

Purpose	Find mean value of each input matrix
Library	Statistics
Description	The Mean block is a Signal Processing Blockset block. For more information, see the Mean block reference page in the Signal Processing Blockset software documentation.

Purpose Find median value of each input matrix

Library Statistics

Description The Median block is a Signal Processing Blockset block. For more information, see the Median block reference page in the Signal Processing Blockset software documentation.

Median Filter

Purpose

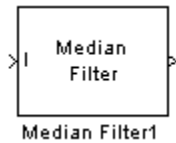
Perform 2-D median filtering

Library

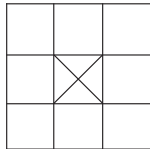
Filtering / Analysis & Enhancement

`vipfilter`

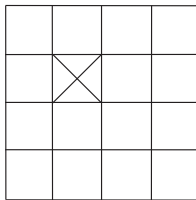
Description



The Median Filter block replaces the central value of an M -by- N neighborhood with its median value. If the neighborhood has a center element, the block places the median value there, as illustrated in the following figure.



The block has a bias toward the upper-left corner when the neighborhood does not have an exact center. See the median value placement in the following figure.



The block pads the edge of the input image, which sometimes causes the pixels within $[M/2 N/2]$ of the edges to appear distorted. The median value is less sensitive than the mean to extreme values. As a result, the Median Filter block can remove salt-and-pepper noise from an image without significantly reducing the sharpness of the image.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	No
Val	Scalar value that represents the constant pad value	Same as I port	No
Output	Matrix of intensity values	Same as I port	No

If the data type of the input signal is floating point, the output has the same data type. The data types of the signals input to the I and Val ports must be the same.

Fixed-Point Data Types

The information in this section is applicable only when the dimensions of the neighborhood are even.

For fixed-point inputs, you can specify accumulator and output data types as discussed in “Dialog Box” on page 2-485. Not all these fixed-point parameters apply to all types of fixed-point inputs. The following table shows the output and accumulator data type used for each fixed-point input.

Fixed-Point Input	Output Data Type	Accumulator Data Type
Even M	X	X
Odd M	X	

Median Filter

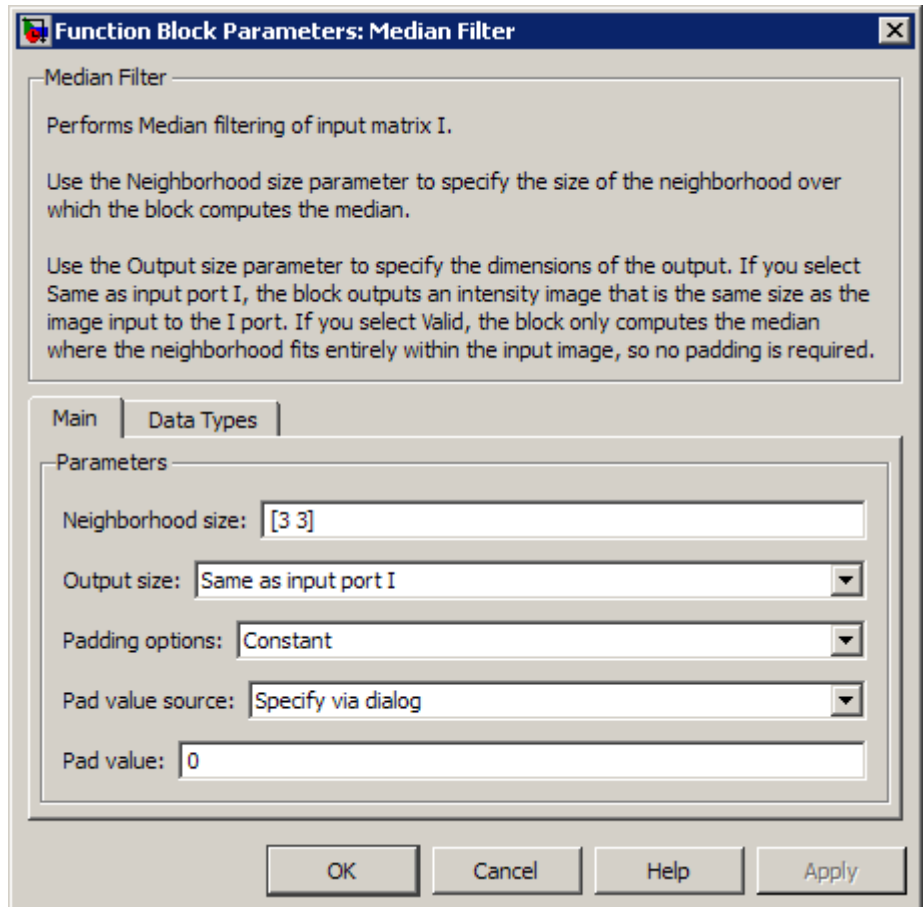
Fixed-Point Input	Output Data Type	Accumulator Data Type
Odd M and complex	X	X
Even M and complex	X	X

When M is even, fixed-point signals use the accumulator and output data types. The accumulator data type store the result of the sum performed while calculating the average of the two central rows of the input matrix. The output data type stores the total result of the average.

Complex fixed-point inputs use the accumulator parameters. The calculation for the sum of the squares of the real and imaginary parts of the input occur, before sorting input elements. The accumulator data type stores the result of the sum of the squares.

Dialog Box

The **Main** pane of the Median Filter dialog box appears as shown in the following figure.



Neighborhood size

Specify the size of the neighborhood over which the block computes the median.

Median Filter

- Enter a scalar value that represents the number of rows and columns in a square matrix.
- Enter a vector that represents the number of rows and columns in a rectangular matrix.

Output size

This parameter controls the size of the output matrix.

- If you choose `Same` as `input port I`, the output has the same dimensions as the input to port I. The **Padding options** parameter appears in the dialog box. Use the **Padding options** parameter to specify how to pad the boundary of your input matrix.
- If you select `Valid`, the block only computes the median where the neighborhood fits entirely within the input image, with no need for padding. The dimensions of the output image are, $\text{output rows} = \text{input rows} - \text{neighborhood rows} + 1$, and $\text{output columns} = \text{input columns} - \text{neighborhood columns} + 1$.

Padding options

Specify how to pad the boundary of your input matrix.

- Select `Constant` to pad your matrix with a constant value. The **Pad value source** parameter appears in the dialog box
- Select `Replicate` to pad your input matrix by repeating its border values.
- Select `Symmetric` to pad your input matrix with its mirror image.
- Select `Circular` to pad your input matrix using a circular repetition of its elements. This parameter appears if, for the **Output size** parameter, you select `Same as input port I`.

For more information on padding, see the Image Pad block reference page.

Pad value source

Use this parameter to specify how to define your constant boundary value.

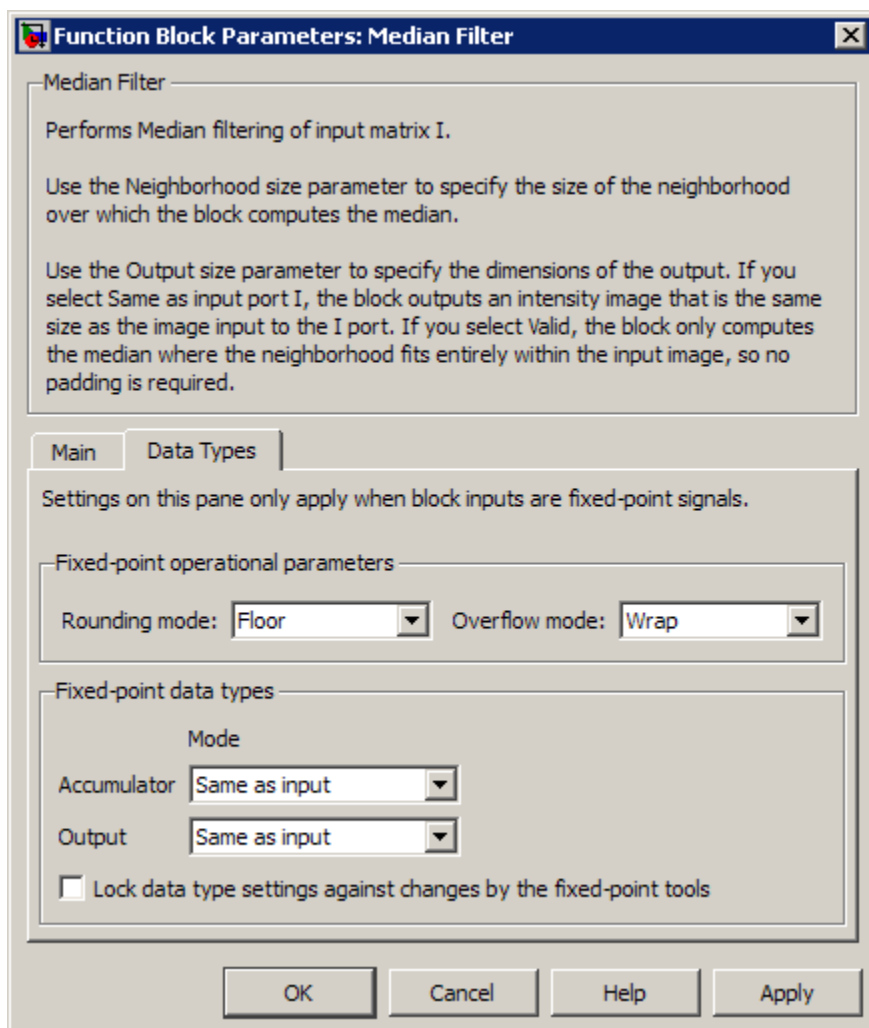
- Select **Specify via dialog** to enter your value in the block parameters dialog box. The **Pad value** parameter appears in the dialog box.
- Select **Input port** to specify your constant value using the **Val** port. This parameter appears if, for the **Padding options** parameter, you select **Constant**.

Pad value

Enter the constant value with which to pad your matrix. This parameter appears if, for the **Pad value source** parameter, you select **Specify via dialog**. Tunable.

The **Data Types** pane of the Median Filter dialog box appears as follows. The parameters on this dialog box becomes visible only when the dimensions of the neighborhood are even.

Median Filter



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Note Only certain cases require the use of the accumulator and output parameters. Refer to “Fixed-Point Data Types” on page 2-483 for more information.

Accumulator

Use this parameter to specify the accumulator word and fraction lengths resulting from a complex-complex multiplication in the block:

- When you select **Same as input**, these characteristics match the related input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of 0.

Output

Choose how to specify the output word length and fraction length:

- When you select **Same as input**, these characteristics match the related input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the output. This block requires power-of-two slope and a bias of 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more

Median Filter

information, see `fxptd1g`, a reference page on the Fixed-Point Tool in the Simulink documentation.

References

[1] Gonzales, Rafael C. and Richard E. Woods. *Digital Image Processing. 2nd ed.* Englewood Cliffs, NJ: Prentice-Hall, 2002.

See Also

2-D Convolution

Video and Image Processing Blockset

2-D FIR Filter

Video and Image Processing Blockset

`medfilt2`

Image Processing Toolbox

Purpose Find minimum values in input or sequence of inputs

Library Statistics

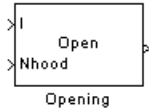
Description The Minimum block is a Signal Processing Blockset block. For more information, see the Minimum block reference page in the Signal Processing Blockset software documentation.

Opening

Purpose Perform morphological opening on binary or intensity images

Library Morphological Operations

Description The Opening block performs an erosion operation followed by a dilation operation using a predefined neighborhood or structuring element. This block uses flat structuring elements only.



Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• Boolean• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No
Nhood	Matrix or vector of ones and zeros that represents the neighborhood values	Boolean	No
Output	Scalar, vector, or matrix of intensity values that represents the opened image	Same as I port	No

The output signal has the same data type as the input to the I port.

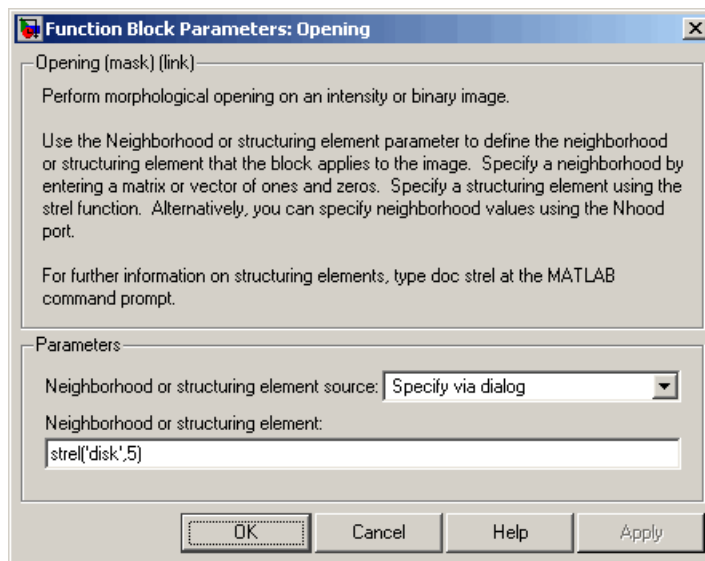
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select Specify via dialog, the **Neighborhood or structuring**

element parameter appears in the dialog box. If you select Input port, the Nhood port appears on the block. Use this port to enter your neighborhood values as a matrix or vector of 1s and 0s. You can only specify a structuring element using the dialog box.

Use the **Neighborhood or structuring element** parameter to define the region the block moves throughout the image. Specify a neighborhood by entering a matrix or vector of 1s and 0s. Specify a structuring element with the `strel` function from the Image Processing Toolbox. If the structuring element is decomposable into smaller elements, the block executes at higher speeds due to the use of a more efficient algorithm.

Dialog Box

The Opening dialog box appears as shown in the following figure.



Neighborhood or structuring element source

Specify how to enter your neighborhood or structuring element values. Select **Specify via dialog** to enter the values in the dialog box. Select **Input port** to use the Nhood port to specify the

neighborhood values. You can only specify a structuring element using the dialog box.

Neighborhood or structuring element

If you are specifying a neighborhood, this parameter must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the `strel` function from the Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or structuring element source** parameter, you select Specify via dialog.

References

[1] Soille, Pierre. *Morphological Image Analysis. 2nd ed.* New York: Springer, 2003.

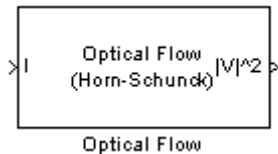
See Also

Bottom-hat	Video and Image Processing Blockset software
Closing	Video and Image Processing Blockset software
Dilation	Video and Image Processing Blockset software
Erosion	Video and Image Processing Blockset software
Label	Video and Image Processing Blockset software
Top-hat	Video and Image Processing Blockset software
<code>imopen</code>	Image Processing Toolbox software
<code>strel</code>	Image Processing Toolbox software

Purpose Estimate object velocities

Library Analysis & Enhancement
vipanalysis

Description



The Optical Flow block estimates the direction and speed of object motion from one image to another or from one video frame to another using either the Horn-Schunck or the Lucas-Kanade method.

Port	Output	Supported Data Types	Complex Values Supported
I/I1	Scalar, vector, or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point (supported when the Method parameter is set to Lucas-Kanade) 	No
I2	Scalar, vector, or matrix of intensity values	Same as I port	No
V ^2	Matrix of velocity magnitudes	Same as I port	No
V	Matrix of velocity components in complex form	Same as I port	Yes

To compute the optical flow between two images, you must solve the following optical flow constraint equation:

$$I_x u + I_y v + I_t = 0$$

In this equation, the following values are represented:

- I_x , I_y and I_t are the spatiotemporal image brightness derivatives
- u is the horizontal optical flow
- v is the vertical optical flow

Because this equation is underconstrained, there are several methods to solve for u and v :

- Horn-Schunck Method
- Lucas-Kanade Method

See the following two sections for descriptions of these methods

Horn-Schunck Method

By assuming that the optical flow is smooth over the entire image, the Horn-Schunck method computes an estimate of the velocity field,

$[u \ v]^T$, that minimizes this equation:

$$E = \iint (I_x u + I_y v + I_t)^2 dx dy + \alpha \iint \left\{ \left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 \right\} dx dy$$

In this equation, $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial y}$ are the spatial derivatives of the optical velocity component u , and α scales the global smoothness term. The Horn-Schunck method minimizes the previous equation to obtain the velocity field, $[u \ v]$, for each pixel in the image, which is given by the following equations:

$$u_{x,y}^{k+1} = \bar{u}_{x,y}^{-k} - \frac{I_x [I_x \bar{u}_{x,y}^k + I_y \bar{v}_{x,y}^k + I_t]}{\alpha^2 + I_x^2 + I_y^2}$$

$$v_{x,y}^{k+1} = \bar{v}_{x,y}^{-k} - \frac{I_y [I_x \bar{u}_{x,y}^k + I_y \bar{v}_{x,y}^k + I_t]}{\alpha^2 + I_x^2 + I_y^2}$$

In this equation, $\begin{bmatrix} u_{x,y}^k & v_{x,y}^k \end{bmatrix}$ is the velocity estimate for the pixel at (x,y) , and $\begin{bmatrix} \bar{u}_{x,y}^{-k} & \bar{v}_{x,y}^{-k} \end{bmatrix}$ is the neighborhood average of $\begin{bmatrix} u_{x,y}^k & v_{x,y}^k \end{bmatrix}$. For $k=0$, the initial velocity is 0.

If you set the **Method** parameter to Horn-Schunck, the block solves for u and v as follows:

- 1 Compute I_x and I_y using the Sobel convolution kernel:

$\begin{bmatrix} -1 & -2 & -1; & 0 & 0 & 0; & 1 & 2 & 1 \end{bmatrix}$, and its transposed form for each pixel in the first image.

- 2 Compute I_t between images 1 and 2 using the $\begin{bmatrix} -1 & 1 \end{bmatrix}$ kernel.

- 3 Assume the previous velocity to be 0, and compute the average velocity for each pixel using $\begin{bmatrix} 0 & 1 & 0; & 1 & 0 & 1; & 0 & 1 & 0 \end{bmatrix}$ as a convolution kernel.

- 4 Iteratively solve for u and v .

Use the **Compute optical flow between** parameter to specify whether to compute the optical flow between two images or two video frames. If you select **Current frame and N-th frame back**, the **N** parameter appears in the dialog box. Enter a scalar value that represents the number of frames between the reference frame and the current frame.

Use the **Velocity output** parameter to specify the block's output. If you select **Magnitude-squared**, the block outputs the optical flow matrix

where each element is of the form $u^2 + v^2$. If you select **Horizontal** and **vertical** components in **complex** form, the block outputs the optical flow matrix where each element is of the form $u + jv$. The horizontal velocity component represents the real part of each value and the vertical velocity component represents the imaginary part of each value.

The smoothness factor, α , is a positive constant. If the relative motion between the two images or video frames is large, enter a large positive scalar value for the **Smoothness factor**. If the relative motion is small, enter a small positive scalar value. You must experiment to find the smoothness factor that best suits your application.

The Optical Flow block uses an iterative process to calculate the optical flow between two images or two video frames. Use the **Stop iterative solution** parameter to control when the iterative process stops. If you want it to stop when the velocity difference is below a certain threshold value, select **When velocity difference falls below threshold**. Then, use the **Velocity difference threshold** parameter to specify a threshold value. If you want the iterative process to stop after a certain number of iterations, choose **When maximum number of iterations is reached**. Then use the **Maximum number of iterations** parameter to specify the maximum number of iterations you want the block to perform. If you select **Whichever comes first**, you must enter values for both the **Velocity difference threshold** and **Maximum number of iterations** parameters.

The block stops iterating as soon as one of these conditions is satisfied.

Lucas-Kanade Method

To solve the optical flow constraint equation for u and v , the Lucas-Kanade method divides the original image into smaller sections and assumes a constant velocity in each section. Then, it performs a weighted least-square fit of the optical flow constraint equation to

a constant model for $[u \ v]^T$ in each section, Ω , by minimizing the following equation:

$$\sum_{x \in \Omega} W^2 [I_x u + I_y v + I_t]^2$$

Here, W is a window function that emphasizes the constraints at the center of each section. The solution to the minimization problem is given by the following equation:

$$\begin{bmatrix} \sum W^2 I_x^2 & \sum W^2 I_x I_y \\ \sum W^2 I_y I_x & \sum W^2 I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum W^2 I_x I_t \\ \sum W^2 I_y I_t \end{bmatrix}$$

If you set the **Method** parameter to **Lucas-Kanade**, the block computes I_t using a difference filter or a derivative of a Gaussian filter.

The two following sections explain how I_x , I_y , I_t , and then u and v are computed.

Difference Filter

If you set the **Temporal gradient filter** parameter to **Difference filter** $[-1 \ 1]$, the block solves for u and v as follows:

- 1 Compute I_x and I_y using the kernel $[-1 \ 8 \ 0 \ -8 \ 1]/12$ and its transposed form.

If you are working with fixed-point data types, the kernel values are signed fixed-point values with word length equal to 16 and fraction length equal to 15.

- 2 Compute I_t between images 1 and 2 using the $[-1 \ 1]$ kernel.
- 3 Smooth the gradient components, I_x , I_y , and I_t , using a separable and isotropic 5-by-5 element kernel whose effective 1-D coefficients are $[1 \ 4 \ 6 \ 4 \ 1]/16$. If you are working with fixed-point data types, the kernel values are unsigned fixed-point values with word length equal to 8 and fraction length equal to 7.

- 4 Solve the 2-by-2 linear equations for each pixel using the following method:

- If $A = \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} \sum W^2 I_x^2 & \sum W^2 I_x I_y \\ \sum W^2 I_y I_x & \sum W^2 I_y^2 \end{bmatrix}$

Then the eigenvalues of A are $\lambda_i = \frac{a+c}{2} \pm \frac{\sqrt{4b^2 + (a-c)^2}}{2}; i = 1, 2$

In the fixed-point diagrams, $P = \frac{a+c}{2}, Q = \frac{\sqrt{4b^2 + (a-c)^2}}{2}$

- When the block finds the eigenvalues, it compares them to the threshold, τ , that corresponds to the value you enter for the **Threshold for noise reduction** parameter. The results fall into one of the following cases:

Case 1: $\lambda_1 \geq \tau$ and $\lambda_2 \geq \tau$

A is nonsingular, so the block solves the system of equations using Cramer's rule.

Case 2: $\lambda_1 \geq \tau$ and $\lambda_2 < \tau$

A is singular (noninvertible), so the block normalizes the gradient flow to calculate u and v .

Case 3: $\lambda_1 < \tau$ and $\lambda_2 < \tau$

The optical flow, u and v , is 0.

The **Compute optical flow between, N**, and **Velocity output** parameters are described in "Horn-Schunck Method" on page 2-496.

Use the **Threshold for noise reduction** parameter to eliminate the effect of small movements between frames. The higher the number, the less small movements impact the optical flow calculation. Experiment with this parameter to find the value that best suits your application.

Derivative of Gaussian

If you set the **Temporal gradient filter** parameter to Derivative of Gaussian, the block solves for u and v using the following steps. You can see the flow chart for this process at the end of this section:

- 1 Compute I_x and I_y using the following steps:
 - a Use a Gaussian filter to perform temporal filtering. Specify the temporal filter characteristics such as the standard deviation and number of filter coefficients using the **Number of frames to buffer for temporal smoothing** parameter.
 - b Use a Gaussian filter and the derivative of a Gaussian filter to smooth the image using spatial filtering. Specify the standard deviation and length of the image smoothing filter using the **Standard deviation for image smoothing filter** parameter.
- 2 Compute I_t between images 1 and 2 using the following steps:
 - a Use the derivative of a Gaussian filter to perform temporal filtering. Specify the temporal filter characteristics such as the standard deviation and number of filter coefficients using the **Number of frames to buffer for temporal smoothing** parameter.
 - b Use the filter described in step 1b to perform spatial filtering on the output of the temporal filter.
- 3 Smooth the gradient components, I_x , I_y , and I_t , using a gradient smoothing filter. Use the **Standard deviation for gradient smoothing filter** parameter to specify the standard deviation and the number of filter coefficients for the gradient smoothing filter.
- 4 Solve the 2-by-2 linear equations for each pixel using the following method:

- If $A = \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} \sum W^2 I_x^2 & \sum W^2 I_x I_y \\ \sum W^2 I_y I_x & \sum W^2 I_y^2 \end{bmatrix}$

Then the eigenvalues of A are $\lambda_i = \frac{a+c}{2} \pm \frac{\sqrt{4b^2 + (a-c)^2}}{2}; i = 1, 2$

- When the block finds the eigenvalues, it compares them to the threshold, τ , that corresponds to the value you enter for the **Threshold for noise reduction** parameter. The results fall into one of the following cases:

Case 1: $\lambda_1 \geq \tau$ and $\lambda_2 \geq \tau$

A is nonsingular, so the block solves the system of equations using Cramer's rule.

Case 2: $\lambda_1 \geq \tau$ and $\lambda_2 < \tau$

A is singular (noninvertible), so the block normalizes the gradient flow to calculate u and v .

Case 3: $\lambda_1 < \tau$ and $\lambda_2 < \tau$

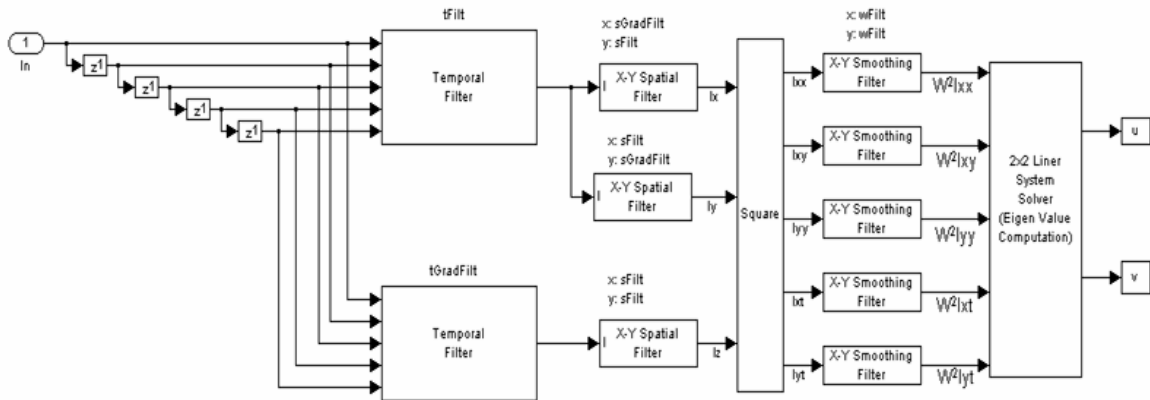
The optical flow, u and v , is 0.

Select the **Discard normal flow estimates when constraint equation is ill-conditioned** check box if you want the block to set the motion vector to zero when the optical flow constraint equation is ill-conditioned. The block calculates these motion vectors on a pixel-by-pixel basis.

Select the **Output image corresponding to motion vectors (accounts for block delay)** check box if you want the block to output the image that corresponds to the motion vector being output by the block.

The **Velocity output** parameter is described in "Horn-Schunck Method" on page 2-496.

Use the **Threshold for noise reduction** parameter to eliminate the effect of small movements between frames. The higher the number, the less small movements impact the optical flow calculation. Experiment with this parameter to find the value that best suits your application.



tFilt = Coefficients of Gaussian Filter
 tGradFilt = Coefficients of the Derivative of a Gaussian Filter

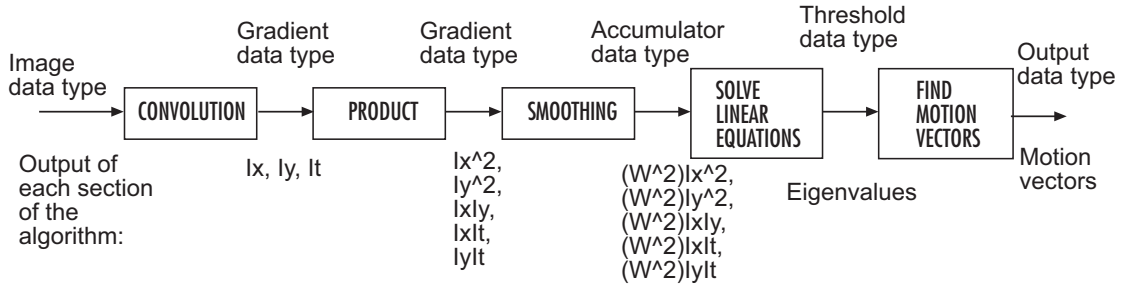
sFilt = Coefficients of Gaussian Filter
 sGradFilt = Coefficients of the Derivative of a Gaussian Filter

Fixed-Point Data Type Diagram

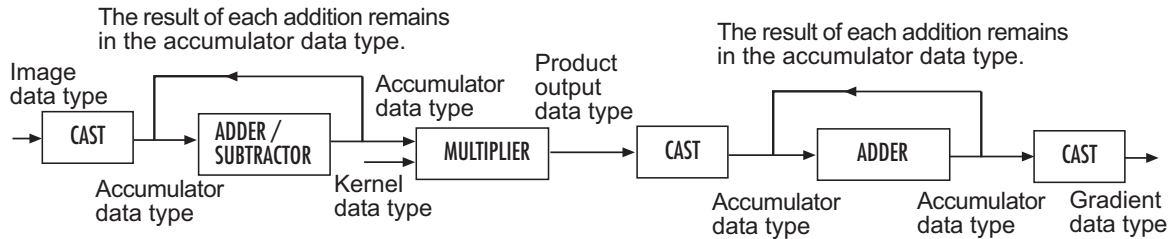
The following diagrams shows the data types used in the Optical Flow block for fixed-point signals. The block supports fixed-point data types only when the **Method** parameter is set to Lucas-Kanade.

Optical Flow

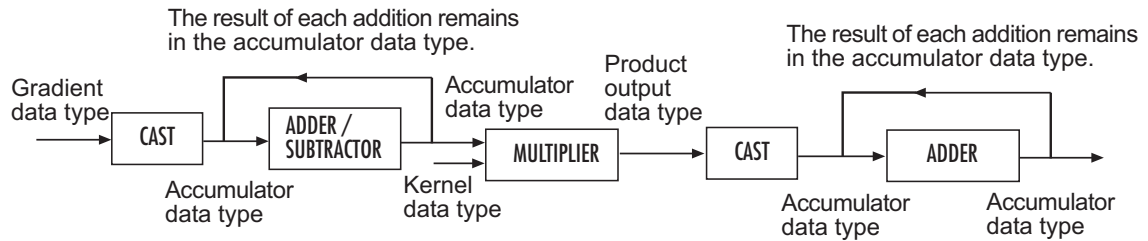
Data type diagram for Optical Flow block's overall algorithm



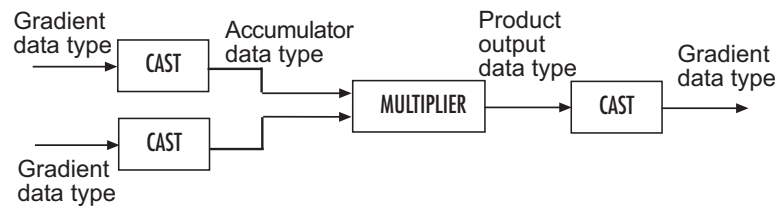
Data type diagram for convolution algorithm



Data type diagram for smoothing algorithm



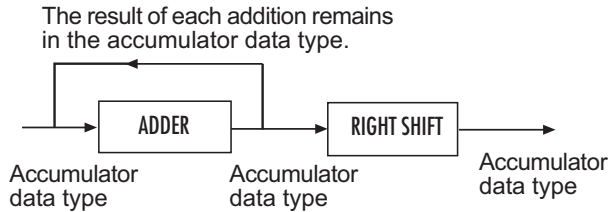
Data type diagram for product algorithm



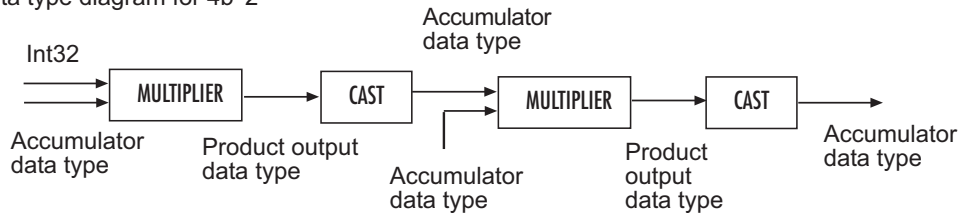
Optical Flow

Solving linear equations to compute eigenvalues
(see Step 4 in the Lucas-Kanade Method section for the eigenvalue equations)

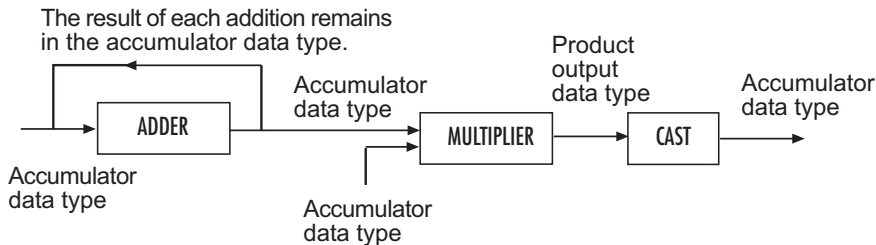
Data type diagram for P



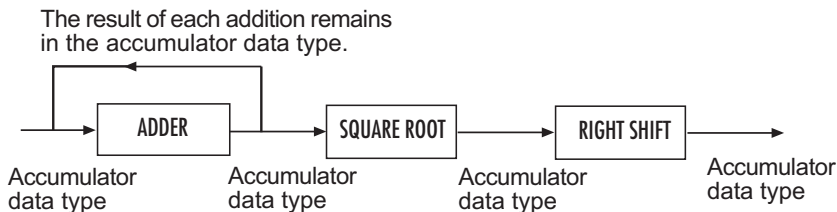
Data type diagram for $4b^2$



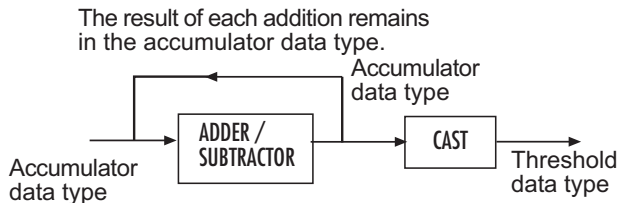
Data type diagram for $(a-c)^2$



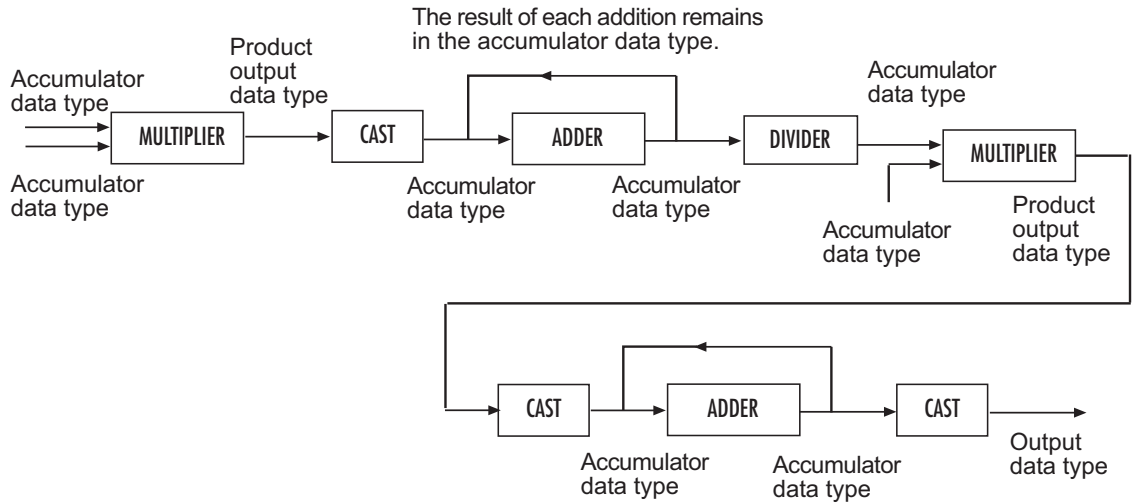
Data type diagram for Q



Data type diagram for eigenvalues



Data type diagram for finding the motion vectors algorithm

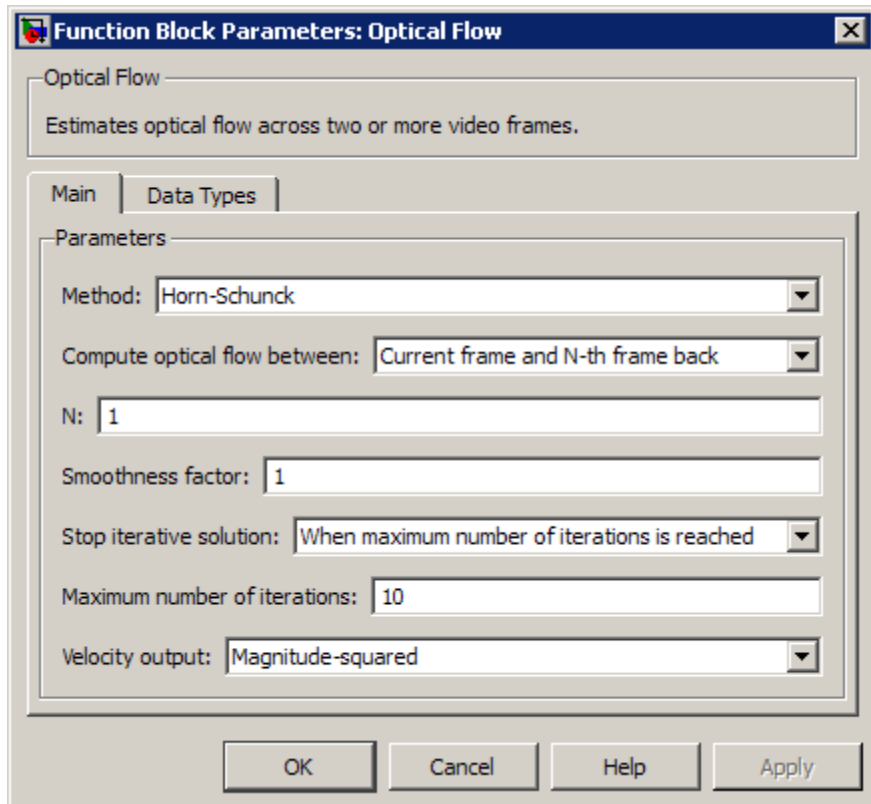


You can set the product output, accumulator, gradients, threshold, and output data types in the block mask.

Optical Flow

Dialog Box

The **Main** pane of the Optical Flow dialog box appears as shown in the following figure.



Method

Select the method the block uses to calculate the optical flow. Your choices are Horn-Schunck or Lucas-Kanade.

Compute optical flow between

Select **Two images** to compute the optical flow between two images. Select **Current frame and N-th frame back** to compute the optical flow between two video frames that are N frames apart.

This parameter is visible if you set the **Method** parameter to Horn-Schunck or you set the **Method** parameter to Lucas-Kanade and the **Temporal gradient filter** to Difference filter [-1 1].

N

Enter a scalar value that represents the number of frames between the reference frame and the current frame. This parameter becomes available if you set the **Compute optical flow between** parameter, you select Current frame and N-th frame back.

Smoothness factor

If the relative motion between the two images or video frames is large, enter a large positive scalar value. If the relative motion is small, enter a small positive scalar value. This parameter becomes available if you set the **Method** parameter to Horn-Schunck.

Stop iterative solution

Use this parameter to control when the block's iterative solution process stops. If you want it to stop when the velocity difference is below a certain threshold value, select **When velocity difference falls below threshold**. If you want it to stop after a certain number of iterations, choose **When maximum number of iterations is reached**. You can also select **Whichever comes first**. This parameter becomes available if you set the **Method** parameter to Horn-Schunck.

Maximum number of iterations

Enter a scalar value that represents the maximum number of iterations you want the block to perform. This parameter is only visible if, for the **Stop iterative solution** parameter, you select **When maximum number of iterations is reached** or **Whichever comes first**. This parameter becomes available if you set the **Method** parameter to Horn-Schunck.

Velocity difference threshold

Enter a scalar threshold value. This parameter is only visible if, for the **Stop iterative solution** parameter, you select **When**

velocity difference falls below threshold or Whichever comes first. This parameter becomes available if you set the **Method** parameter to Horn-Schunck.

Velocity output

If you select Magnitude-squared, the block outputs the optical flow matrix where each element is of the form $u^2 + v^2$. If you select Horizontal and vertical components in complex form, the block outputs the optical flow matrix where each element is of the form $u + jv$.

Temporal gradient filter

Specify whether the block solves for u and v using a difference filter or a derivative of a Gaussian filter. This parameter becomes available if you set the **Method** parameter to Lucas-Kanade.

Number of frames to buffer for temporal smoothing

Use this parameter to specify the temporal filter characteristics such as the standard deviation and number of filter coefficients. This parameter becomes available if you set the **Temporal gradient filter** parameter to Derivative of Gaussian.

Standard deviation for image smoothing filter

Specify the standard deviation for the image smoothing filter. This parameter becomes available if you set the **Temporal gradient filter** parameter to Derivative of Gaussian.

Standard deviation for gradient smoothing filter

Specify the standard deviation for the gradient smoothing filter. This parameter becomes available if you set the **Temporal gradient filter** parameter to Derivative of Gaussian.

Discard normal flow estimates when constraint equation is ill-conditioned

Select this check box if you want the block to set the motion vector to zero when the optical flow constraint equation is ill-conditioned. This parameter becomes available if you set the **Temporal gradient filter** parameter to Derivative of Gaussian.

Output image corresponding to motion vectors (accounts for block delay)

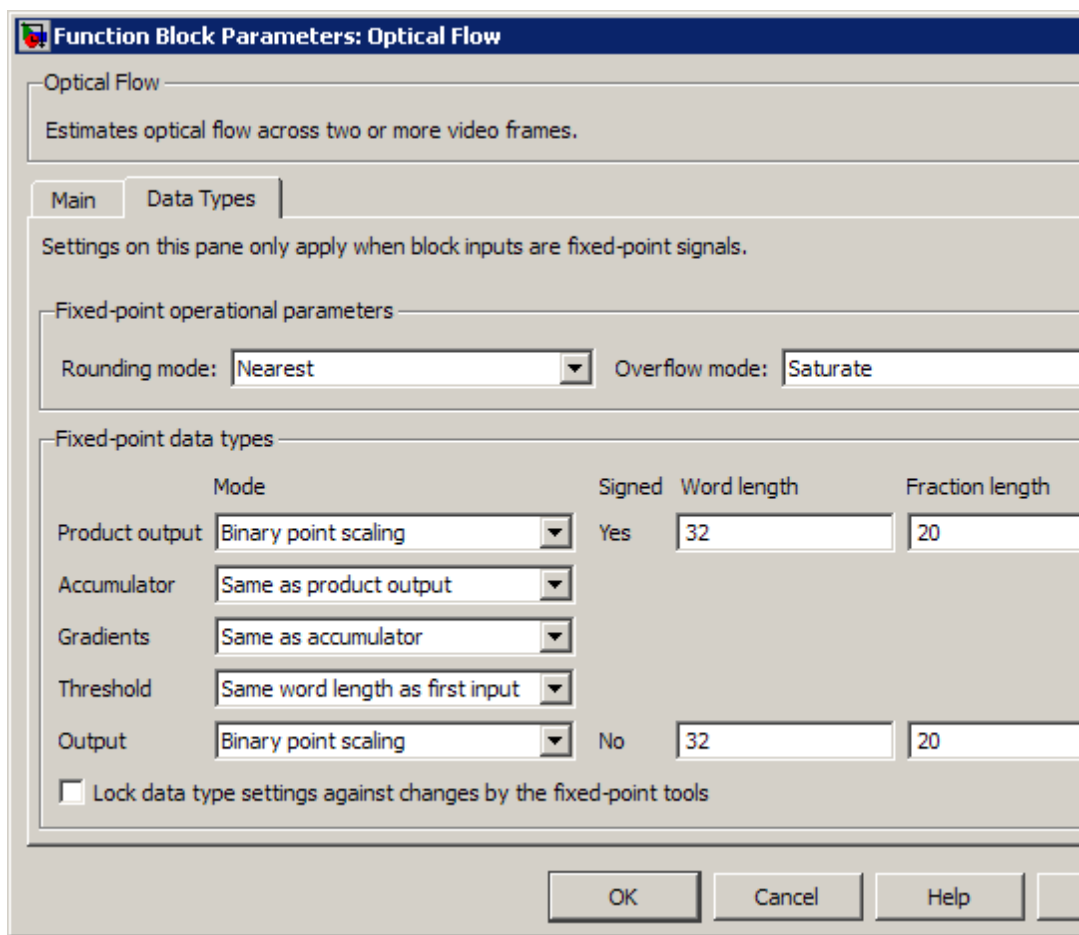
Select this check box if you want the block to output the image that corresponds to the motion vector being output by the block. This parameter becomes available if you set the **Temporal gradient filter** parameter to Derivative of Gaussian.

Threshold for noise reduction

Enter a scalar value that determines the motion threshold between each image or video frame. The higher the number, the less small movements impact the optical flow calculation. This parameter becomes available if you set the **Method** parameter to Lucas-Kanade.

The **Data Types** pane of the Optical Flow dialog box appears as shown in the following figure. The parameters on this dialog box becomes visible only when the Lucas-Kanade method is selected.

Optical Flow



Rounding mode

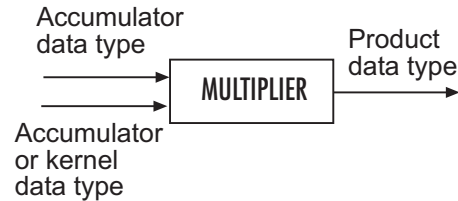
Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

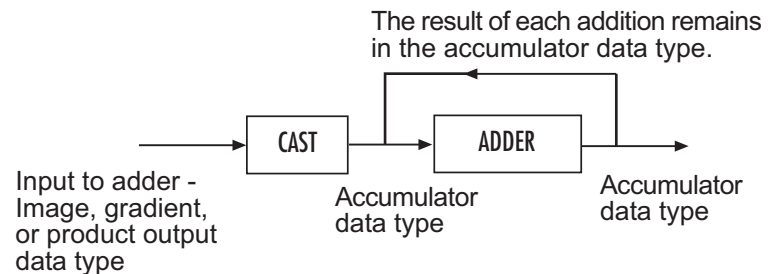
Use this parameter to specify how to designate the product output word and fraction lengths.



- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output in bits.
- When you select **Slope and bias scaling**, you can enter the word length in bits and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Accumulator

Use this parameter to specify how to designate this accumulator word and fraction lengths.



- When you select **Same as product output**, these characteristics match those of the product output.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator in bits.

- When you select **Slope and bias scaling**, you can enter the word length in bits and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Gradients

Choose how to specify the word length and fraction length of the gradients data type:

- When you select **Same as accumulator**, these characteristics match those of the accumulator.
- When you select **Same as product output**, these characteristics match those of the product output.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the quotient, in bits.
- When you select **Slope and bias scaling**, you can enter the word length in bits and the slope of the quotient. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Threshold

Choose how to specify the word length and fraction length of the threshold data type:

- When you select **Same word length as first input**, the threshold word length matches that of the first input.
- When you select **Specify word length**, enter the word length of the threshold data type.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the threshold, in bits.
- When you select **Slope and bias scaling**, you can enter the word length in bits and the slope of the threshold. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Output

Choose how to specify the word length and fraction length of the output data type:

- When you select **Binary point scaling**, you can enter the word length and the fraction length of the output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length in bits and the slope of the output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

References

[1] Barron, J.L., D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt. *Performance of optical flow techniques*. CVPR, 1992.

See Also

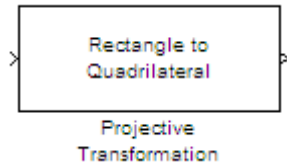
Block Matching	Video and Image Processing Blockset software
Gaussian Pyramid	Video and Image Processing Blockset software

Projective Transformation

Purpose Transform quadrilateral into another quadrilateral

Library Geometric Transformations
vipgeotforms

Description



The Projective Transformation block transforms rectangles into quadrilaterals, quadrilaterals into rectangles, and quadrilaterals into other quadrilaterals.

Port	Output	Supported Data Types	Complex Values Supported
Input / Output	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• Boolean• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer	No
R, G, B (input and output)	Scalar, vector, or matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B ports have the same dimensions and data type.	Same as I port	No

Projective Transformation

Port	Output	Supported Data Types	Complex Values Supported
InPts	Eight-element vector, [r1 c1 r2 c2 ... r4 c4], of scalar values that represents the row and column coordinates of the four corners of the input quadrilateral	<ul style="list-style-type: none"> • Double-precision floating point. (This data type is only supported if the input to the I or R, G, and B ports is a floating-point data type.) • Single-precision floating point. (This data type is only supported if the input to the I or R, G, and B ports is a floating-point data type.) • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer <p>The block rounds the values input at this port and converts them to 32-bit signed integers.</p>	No
OutPts	Eight-element vector, [r1 c1 r2 c2 ... r4 c4], of scalar values that represents the row and column coordinates of the four corners of the output quadrilateral	Same as InPts port	No
InROI	Four-element vector, [r c height width], that defines the row and column coordinates of the top-left corner of a rectangular ROI as well as its height and width	Same as InPts port	No

Projective Transformation

Port	Output	Supported Data Types	Complex Values Supported
OutSize	Four-element vector, [r c height width], that represents the row and column coordinates of the upper-left corner of the rectangular output image as well as its height and width	Same as InPts port	No
Valid	Boolean value that indicates whether or not three quadrilateral vertices are collinear	Boolean	No
InPtsValid	Boolean value that indicates whether or not three input quadrilateral vertices are collinear	Boolean	No
OutPtsValid	Boolean value that indicates whether or not three output quadrilateral vertices are collinear	Boolean	No

Use the **Image signal** parameter to specify how to input and output a color video signal. If you select **One multidimensional signal**, the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

The following sections summarize the behavior of the Projective Transformation block in its three modes.

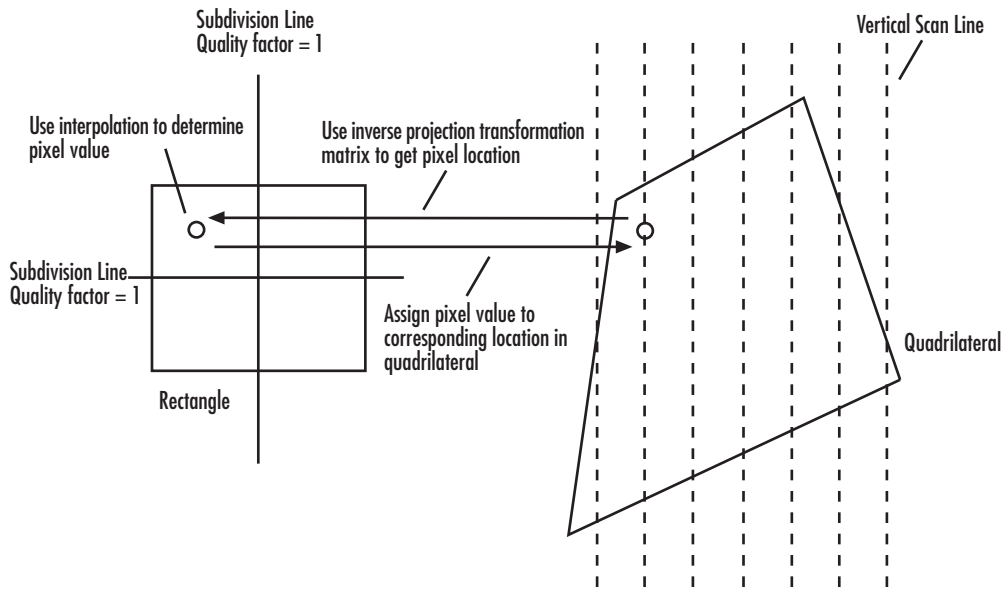
Rectangle to Quadrilateral Mode

Use the **Inverse mapping method** parameter to specify the algorithm the block uses to implement the projective transformation. If you choose **Exact solution**, the block divides the output shape using vertical scan lines. For each pixel location on a scan line, it uses an inverse projection transformation matrix to find the corresponding pixel location in the input image. When this pixel location is not located directly on a pixel in the input image, the block uses 2-D interpolation to calculate the pixel value. Then it assigns this pixel value to the corresponding location in the output image.

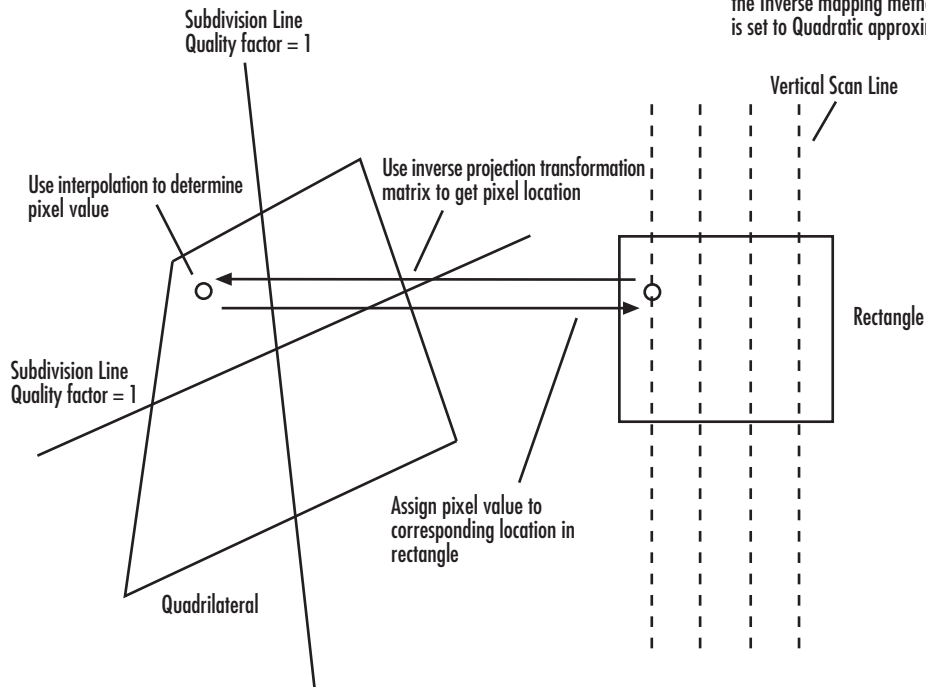
If you choose **Quadratic approximation**, the block divides the input shape using subdivision lines and the output shape using vertical scan lines. For the first pixel location on a scan line, the block uses an inverse projection transformation matrix to find the corresponding pixel location in the input image. If this pixel location is not located directly on a pixel in the input image, the block uses 2-D interpolation to calculate the pixel value. Then it assigns this pixel value to the corresponding location in the output image. The block calculates the remaining pixel locations using x and y offsets that it computes from the inverse projection transformation matrix. Then it repeats the interpolation process to find all the pixel values in the output image.

The following figures summarize two operations of the Projective Transformation block.

Projective Transformation



*The subdivision lines are only used when the Inverse mapping method parameter is set to Quadratic approximation.



Use the **Quality factor (number of subdivisions)** parameter to specify the number of pairs of horizontal and vertical lines (subdivision lines) the block uses to subdivide the output shape. Enter a scalar integer value that is greater than or equal to 0 and less than or equal to the height or width of the input image, whichever is smaller. The larger the quality factor, the closer the approximate solution is to the exact solution. Experiment with this parameter to find the value that best suits your application.

Use the **Background fill value(s)** parameter to specify the background of the output image. If the block outputs an intensity image, enter a scalar value. If the block outputs an RGB image, enter a scalar value that the block uses as each of the R, G, and B values or a three-element vector that specifies an RGB triplet.

Use the **Interpolation method** parameter to specify which 2-D interpolation method the block uses to calculate the pixel values in the output image. If you select **Nearest neighbor**, the block uses the value of the nearest pixel for the new pixel intensity. If you select **Bilinear**, the new pixel value is the weighted average of the four nearest pixel intensities. If you select **Bicubic**, the new pixel value is the weighted average of the 16 nearest pixel intensities.

Input image parameters — Use the **Rectangular ROI** parameter to define the portion of the input image that the block transforms into a quadrilateral. Your choices are **Full image** or **User-defined**. If you select **User-defined**, the **Rectangular ROI source** parameter appears in the dialog box. Use this parameter to specify whether you want to define the ROI using the block dialog box or an input port. If you select **Specify via dialog**, use the **ROI [r,c,height,width]** parameter to enter the row and column coordinates of the upper-left corner of the ROI as well as its height and width. If you select **Input port**, the **If ROI is invalid** parameter appears in the dialog box. Use it to specify the block's behavior if the four-element vector input to the **InROI** port contains values that are outside the input image. Your choices are **Clip**, **Clip and warn**, or **Error**. If you select **Clip**, the block changes the row, column, height, or width values so the ROI fits entirely within the input image.

Projective Transformation

Output image parameters — Use the **Quadrilateral vertices source** parameter to specify how to define the quadrilateral vertices. If you select Specify via dialog, the **Quadrilateral vertices [r1,c1,...,r4,c4]** and **Size** parameters appear in the dialog box. For the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter, enter an eight-element vector of values that represents the row and column coordinates of the four corners of the quadrilateral. Use the **Size** parameter to specify the size of the output image. If you select Full, the output image size is determined by the values you enter for the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter. That is, the block output is big enough so you see the entire output quadrilateral. If you select User-defined, use the **Location and size [r,c,height,width]** parameter to define the row and column coordinates of the upper-left corner of the output image as well as its height and width. If, for the **Quadrilateral vertices source** parameter, you select Input port, the OutPts port appears on the block. The input to this port must be an eight-element vector of scalar values that represent the row and column coordinates of the four corners of the output quadrilateral. Use the **Location and size [r,c,height,width]** parameter to define the row and column coordinates as well as the height and width of the block's output image, which can differ from the size of the output quadrilateral.

If you select the **Output validity of quadrilateral vertices (three points cannot be collinear)** check box, the Valid port appears on the block. If the quadrilateral vertices are not collinear, the block outputs 1 at this port. Otherwise it outputs 0, and the block does not compute an output image.

Quadrilateral to Rectangle Mode

The **Inverse mapping method**, **Quality factor (number of subdivisions)**, **Background fill value(s)**, and **Interpolation method** parameters are described in “Rectangle to Quadrilateral Mode” on page 2-519.

Input image parameters — Use the **Quadrilateral vertices source** parameter to specify how to define the input quadrilateral vertices. If you select Specify via dialog, the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter appears in the dialog box. Enter

an eight-element vector of values that represent the row and column coordinates of the four corners of the quadrilateral. If you select **Input port**, the **InPts** port appears on the block. The input to this port must be an eight-element vector of scalar values that represent the row and column coordinates of the four corners of the input quadrilateral. Use the **If vertices are outside input image** parameter to specify the block's behavior if the input to the **InPts** port contains vertices outside the input image. Your choices are **Clip**, **Clip and warn**, or **Error**. If you select **Clip**, the block changes the row or column values of the vertices so that the quadrilateral fits entirely within the input image. If you select the **Output validity of quadrilateral vertices (three points cannot be collinear)** check box, the **Valid** port appears on the block. If the quadrilateral vertices are not collinear, the block outputs 1 at this port. Otherwise it outputs 0, and the block does not compute an output image.

Output image parameters — Use the **Rectangle size source** parameter to specify how to define the output rectangle size. If you select **Specify via dialog**, the **Rectangle location and size [r,c,height,width]** and **Size** parameters appear in the dialog box. For the **Rectangle location and size [r,c,height,width]** parameter, enter scalar values that represent the row and column coordinates as well as the height and width of the output rectangle. Use the **Size** parameter to specify the size of the block's output image, which can differ from the size of the output rectangle. If you select **Full**, the block output size is determined by the values you enter for the **Rectangle location and size [r,c,height,width]** parameter. That is, the block output is big enough so you see the entire output rectangle. If you select **User-defined**, use the **Location and size [r,c,height,width]** parameter to define the row and column coordinates as well as the height and width of the output image. If, for the **Rectangle size source** parameter, you select **Input port**, the **OutSize** port appears on the block. The input to this port must be a four-element vector of scalar values that represent the row and column coordinates of the upper-left corner of the output rectangle as well as its height and width. Use the **Location and size [r,c,height,width]** parameter to define

Projective Transformation

the row and column coordinates as well as the height and width of the block's output image.

Note If you set the **Inverse mapping method** parameter to Quadratic approximation and the **Quality factor (number of subdivisions)** parameter to a value greater than 0, the subquadrilaterals formed by the subdivision lines might have three collinear vertices. In this case, the block does not compute an output image.

Quadrilateral to Quadrilateral Mode

The **Inverse mapping method**, **Quality factor (number of subdivisions)**, **Background fill value(s)**, and **Interpolation method** parameters are described in “Rectangle to Quadrilateral Mode” on page 2-519.

Input image parameters — Use the **Quadrilateral vertices source** parameter to specify how to define the input quadrilateral vertices. If you select Specify via dialog, the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter appears in the dialog box. Enter an eight-element vector of values that represent the row and column coordinates of the four corners of the input quadrilateral. If you select Input port, the InPts port appears on the block. The input to this port must be an eight-element vector of scalar values that represent the row and column coordinates of the four corners of the input quadrilateral. Use the **If vertices are outside input image** parameter to specify the block's behavior if the input to the InPts port contains vertices outside the input image. Your choices are Clip, Clip and warn, or Error. If you select Clip, the block changes the row or column values of the vertices so that the quadrilateral fits entirely within the input image. If you select the **Output validity of quadrilateral vertices (three points cannot be collinear)** check box, the InPtsValid port appears on the block. If the quadrilateral vertices are not collinear, the block outputs 1 at this port. Otherwise it outputs 0, and the block does not compute an output image.

Output image parameters — Use the **Quadrilateral vertices source** parameter to specify how to define the output quadrilateral vertices. If you select **Specify via dialog**, the **Quadrilateral vertices [r1,c1,...,r4,c4]** and **Size** parameters appear in the dialog box. For the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter, enter an eight-element vector of values that represent the row and column coordinates of the four corners of the output quadrilateral. If, for the **Size** parameter, you select **Full**, the block output image size is determined by the values you enter for the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter. If you select **User-defined**, use the **Location and size [r,c,height,width]** parameter to define the row and column coordinates as well as the height and width of the output image, which can differ from the size of the output quadrilateral. If, for the **Quadrilateral vertices source**, you select **Input port**, the **OutPts** port appears on the block. The input to this port must be an eight-element vector of scalar values that represent the row and column coordinates of the four corners of the output quadrilateral. Use the **Location and size [r,c,height,width]** parameter to define the row and column coordinates as well as the height and width of the block's output image. If you select the **Output validity of quadrilateral vertices (three points cannot be collinear)** check box, the **OutPtsValid** port appears on the block. If the quadrilateral vertices are not collinear, the block outputs 1 at this port. Otherwise it outputs 0, and the block does not compute an output image.

Note If you set the **Inverse mapping method** parameter to **Quadratic approximation** and the **Quality factor (number of subdivisions)** parameter to a value greater than 0, the subquadrilaterals formed by the subdivision lines might have three collinear vertices. In this case, the block does not compute an output image.

Projective Transformation

Example

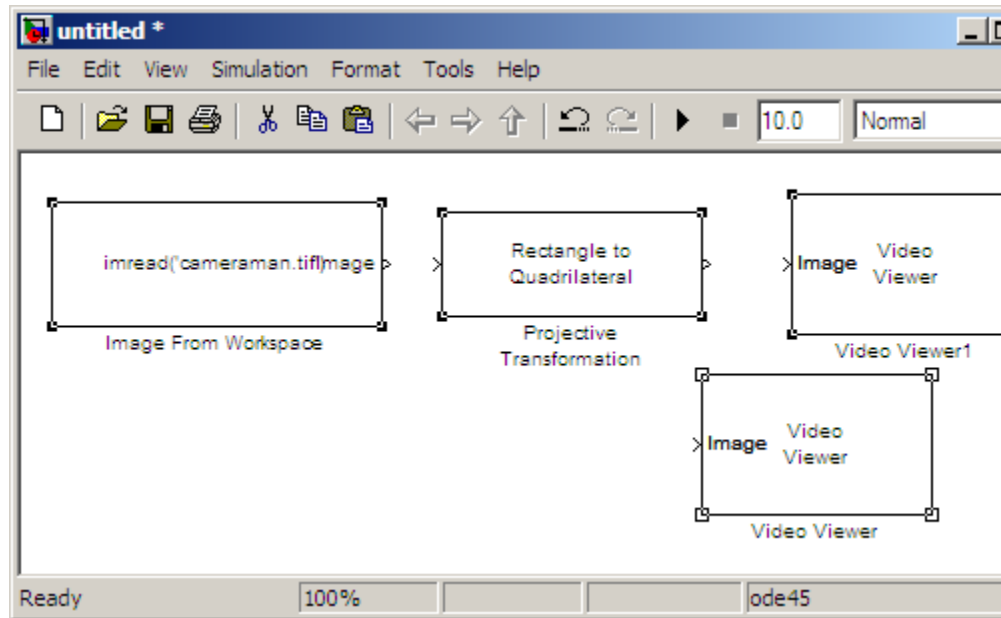
The following example shows you how to convert a rectangular image into a quadrilateral. It also shows you how to change the sizes of the input and output images.

- 1 Create a new Simulink model.
- 2 Click-and-drag the following blocks into your model.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset software / Sources	1
Projective Transformation	Video and Image Processing Blockset software / Geometric Transformations	1
Video Viewer	Video and Image Processing Blockset software / Sinks	2

- 3 Place the blocks so your model looks similar to the following figure.

Projective Transformation

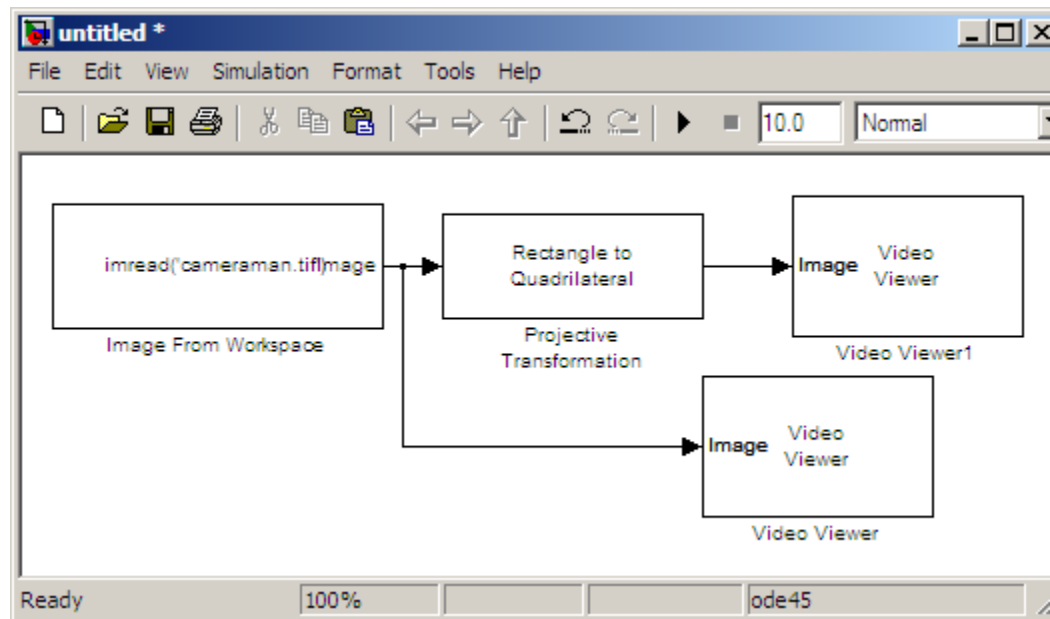


- 4 Use the Image From Workspace block to import an image into your model.
 - Set the **Value** parameter to `imread('cameraman.tif')`.
- 5 Use the Projective Transformation block to transform your rectangular image into a quadrilateral.
 - Set the **Quadrilateral vertices** `[r1,c1,...,r4,c4]` parameter to `[3 40 70 360 285 35 25 5]`.

Projective Transformation

Note The order in which you enter the quadrilateral vertices in the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter affects the appearance of the output image. The block assumes that the first row and column pair correspond to the new location of the upper-left corner of the image. The second row and column pair correspond to the new location of the upper-right corner, and so on in a clockwise direction around the image.

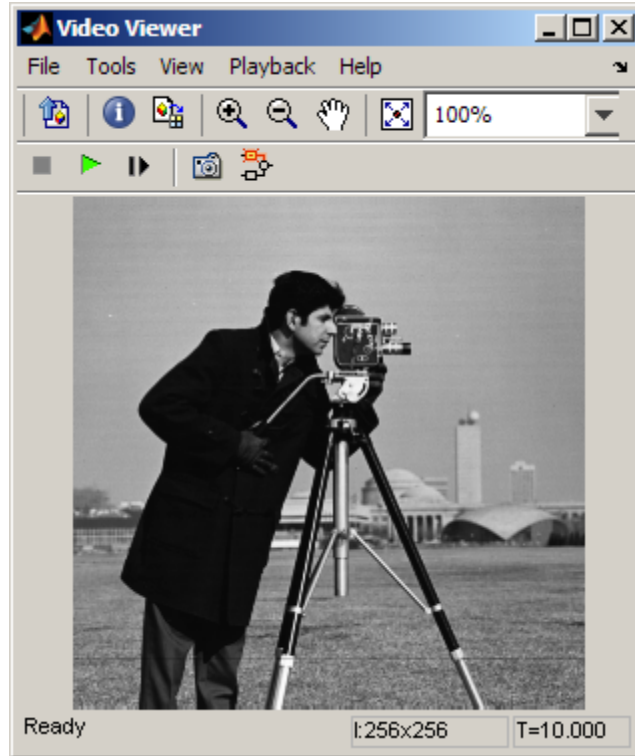
- 6 Use the Video Viewer and Video Viewer1 blocks to view the rectangular and quadrilateral images, respectively. Use the default parameters.
- 7 Connect the blocks so that your model resembles the following figure.



- 8 Run the model.

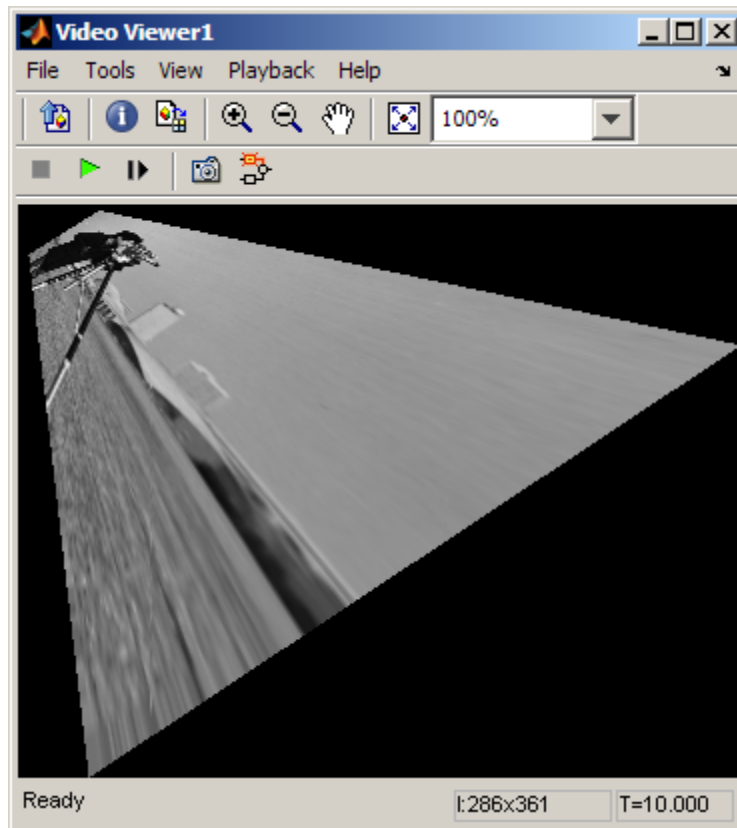
Projective Transformation

The original rectangular image appears in the Video Viewer1 window.

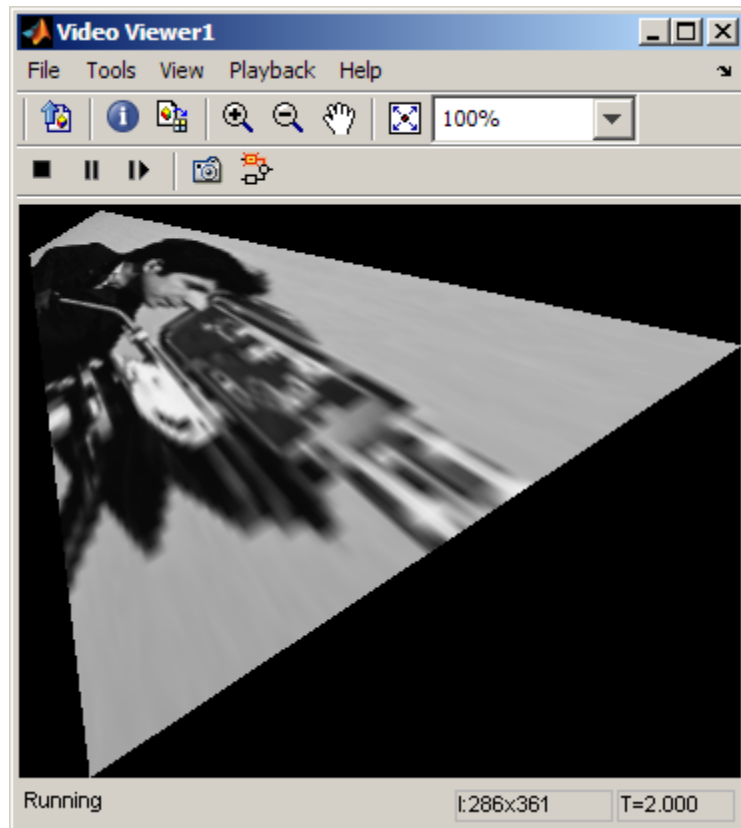


The quadrilateral image appears in the Video Viewer window.

Projective Transformation



- 9 You can change the dimensions of the input image using the parameters in the **Input image parameters** section of the Projective Transformation dialog box. Set the block parameters as follows:
 - **Rectangular ROI** = User-defined
 - **ROI [r,c,height,width]** = [30 20 100 140]
- 10 Run your model. Because you cropped your input image, the quadrilateral image is now a close-up of the man's face and camera.



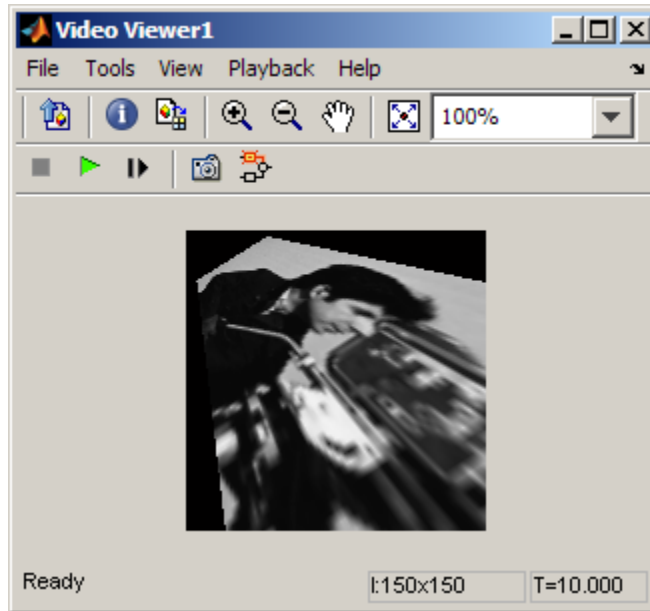
11 You can resize of the output image using the parameters in the **Output image parameters** section of the Projective Transformation dialog box. Set the block parameters as follows:

- **Size** = User-defined
- **Location and size [r,c,height,width]** = [0 0 150 150]

The **Location and size [r,c,height,width]** parameter defines the row and column coordinates as well as the height and width of the output image.

Projective Transformation

- 12 Run your model. The Projective Transformation block outputs a portion of the quadrilateral image, so you can no longer see all of the quadrilateral corners.

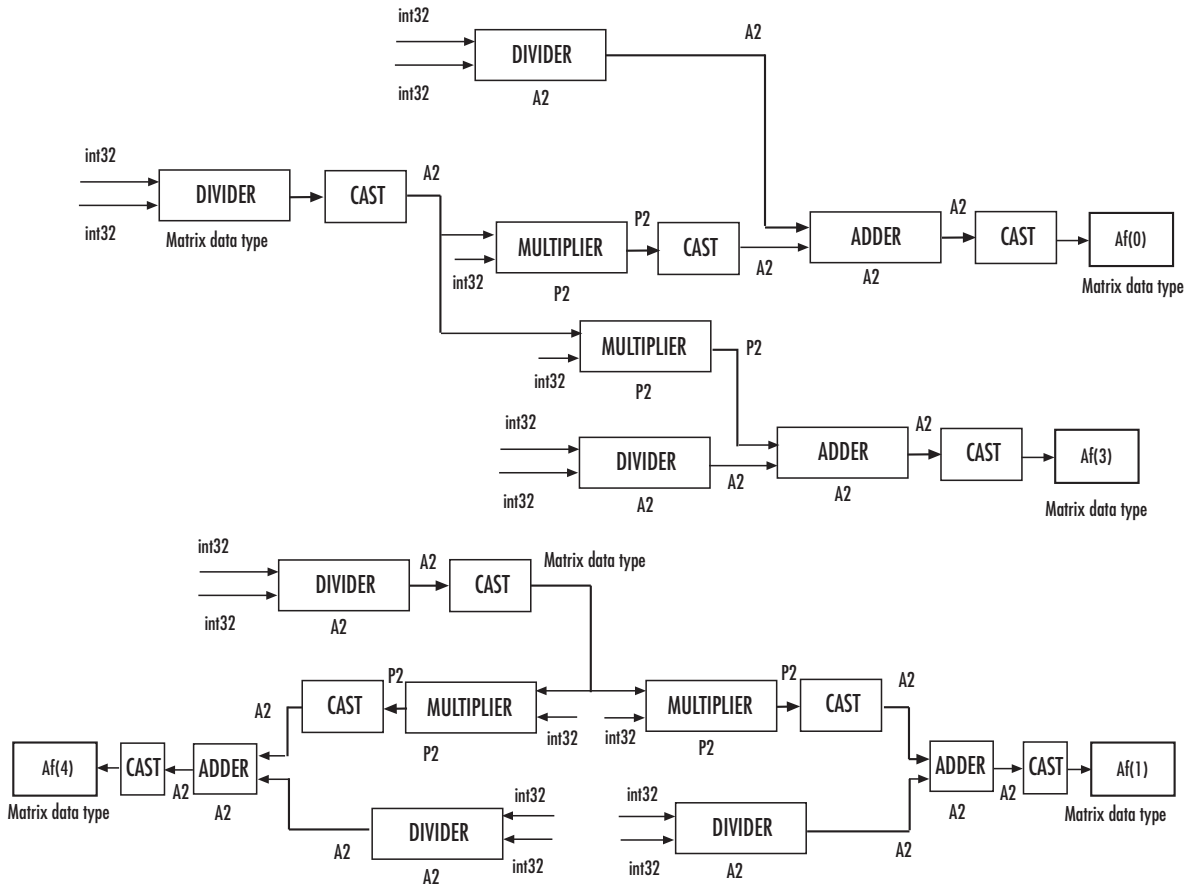


Fixed-Point Data Types

The following diagram shows the data types used in the Projective Transformation block for fixed-point signals:

Projective Transformation

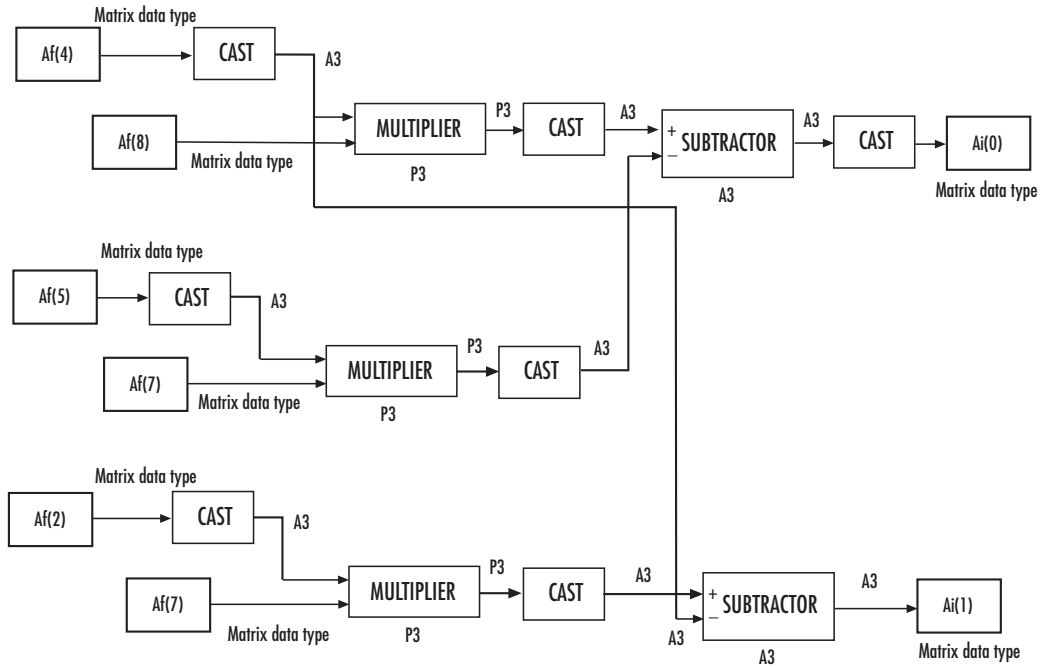
Calculate Forward Projective Transformation Matrix (Af)



The block uses a similar computation to calculate Af(2), Af(4), Af(5), Af(6), Af(7), and Af(8).

Projective Transformation

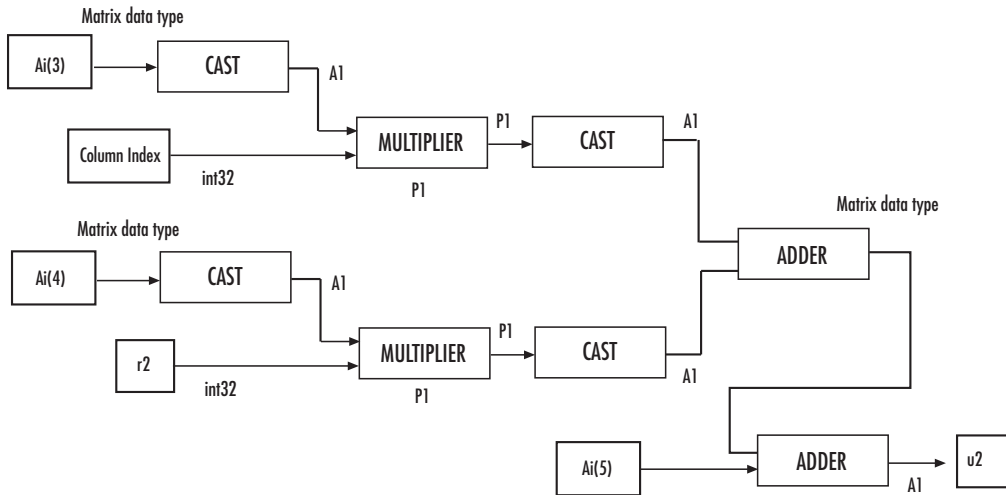
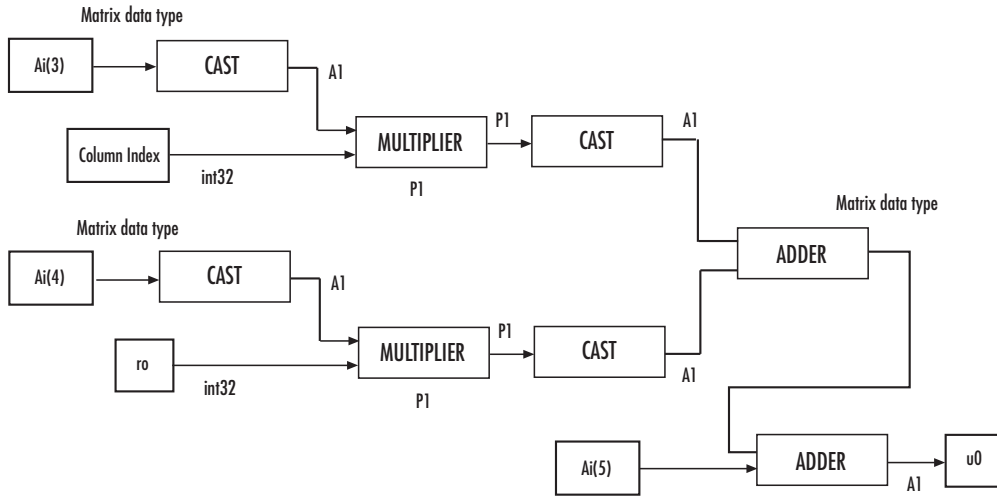
Calculate Inverse Projective Transformation Matrix (A_i)



The block uses a similar computation to calculate $A_i(2)$, $A_i(3)$, $A_i(4)$, $A_i(5)$, $A_i(6)$, $A_i(7)$, and $A_i(8)$.

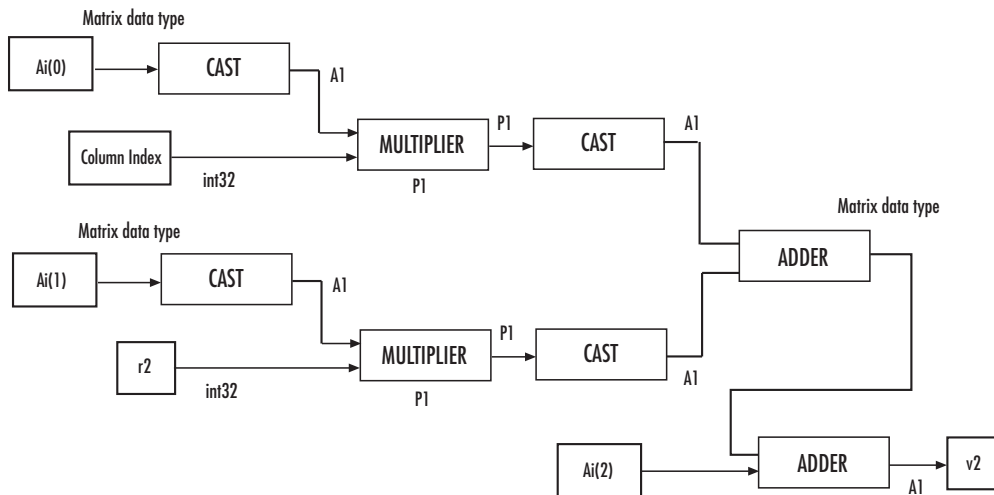
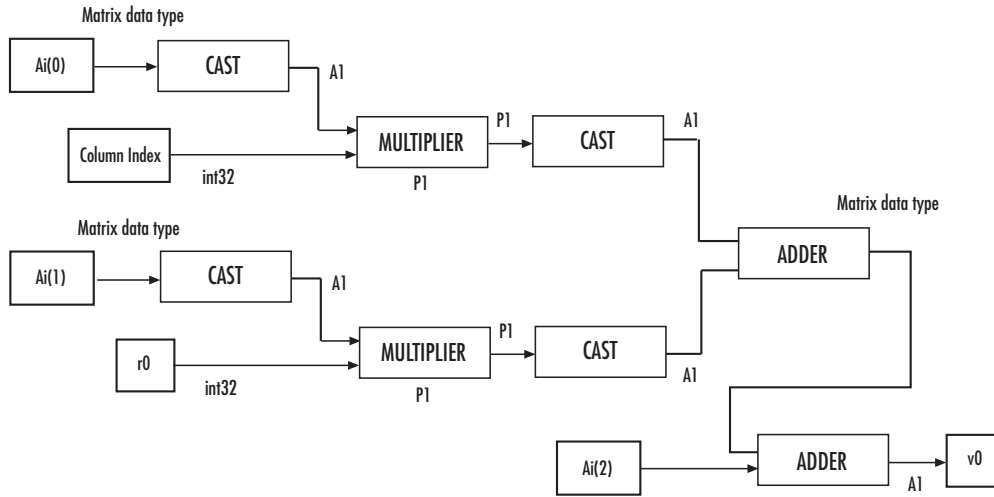
Projective Transformation

Compute Output Pixel



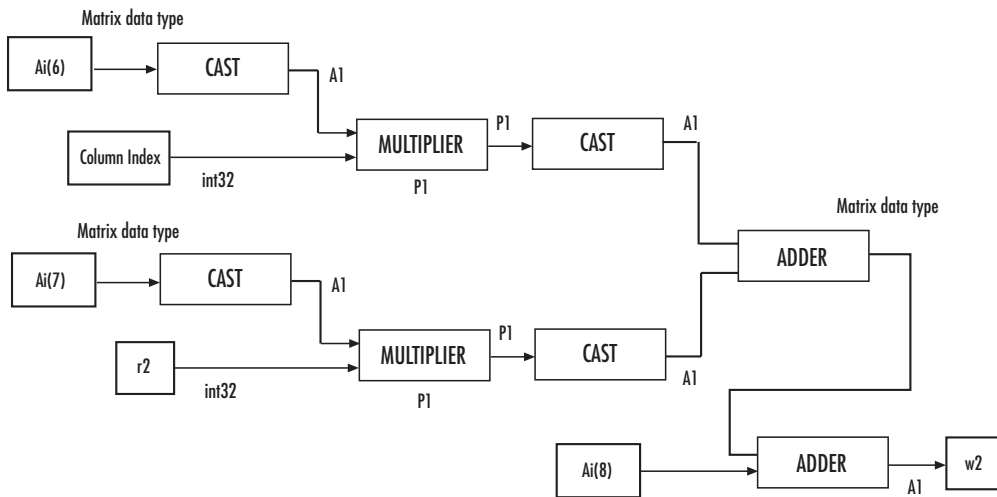
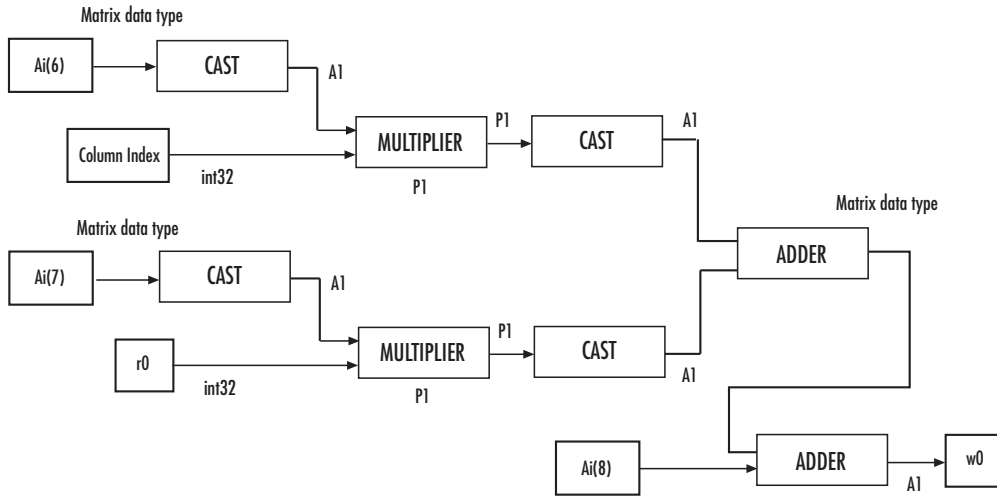
Projective Transformation

Compute Output Pixel (continued)



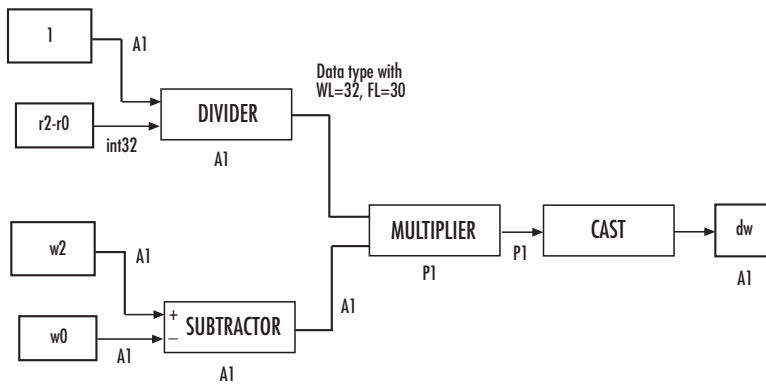
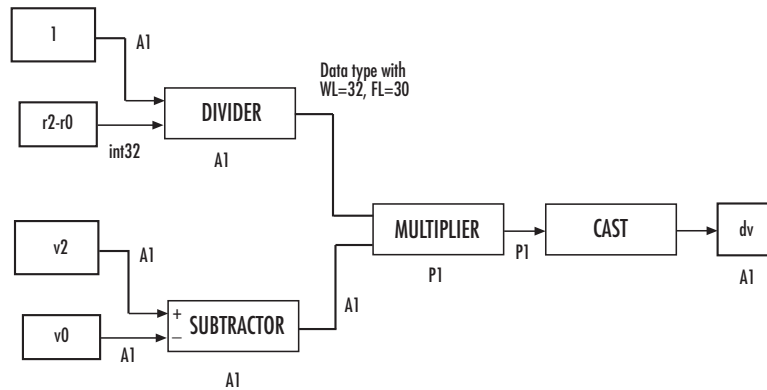
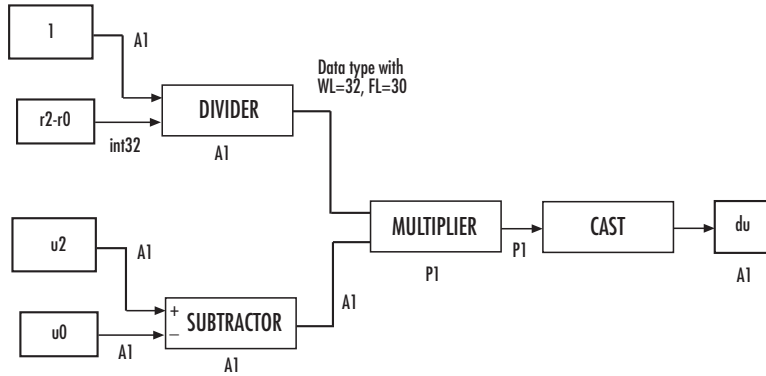
Projective Transformation

Compute Output Pixel (continued)



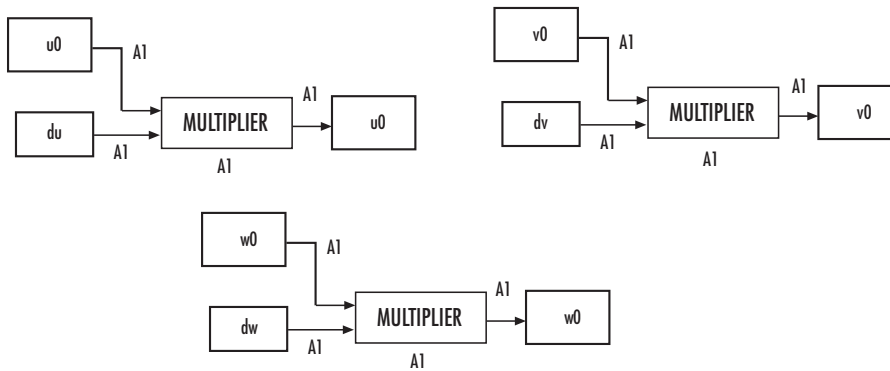
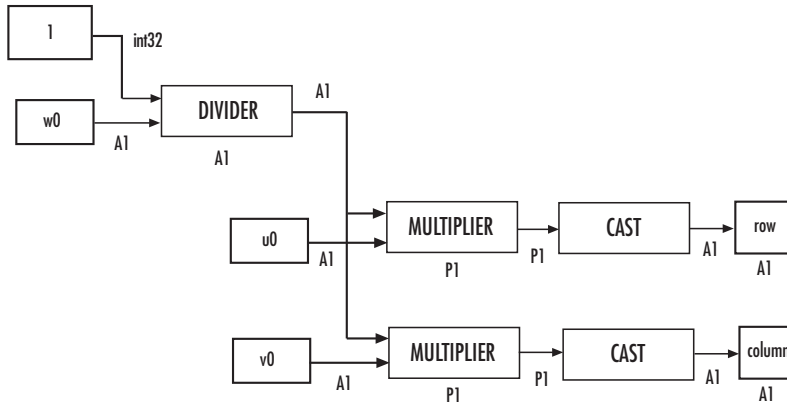
Projective Transformation

Calculate du , dv , dw

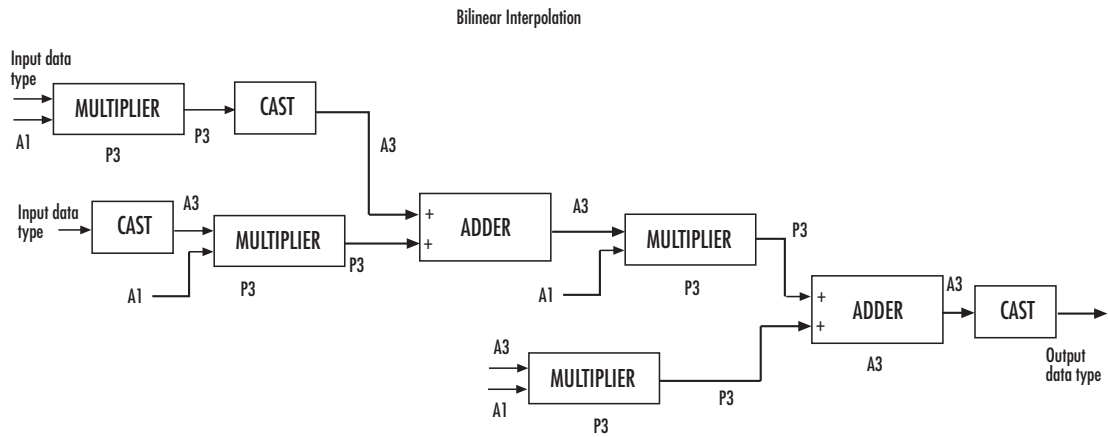


Projective Transformation

Calculate row and column indices of the input image



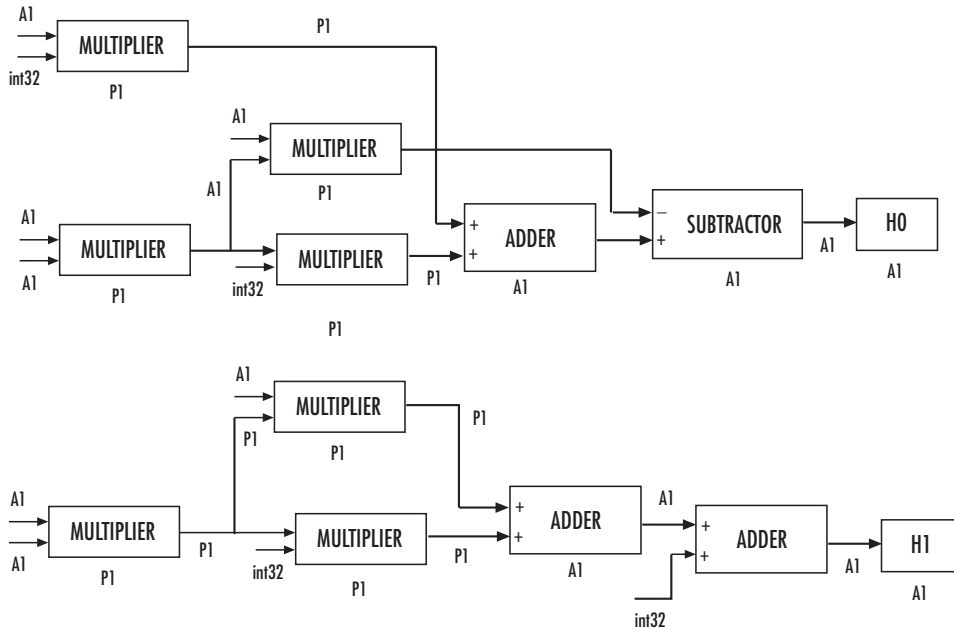
Projective Transformation



Projective Transformation

Bicubic Interpolation

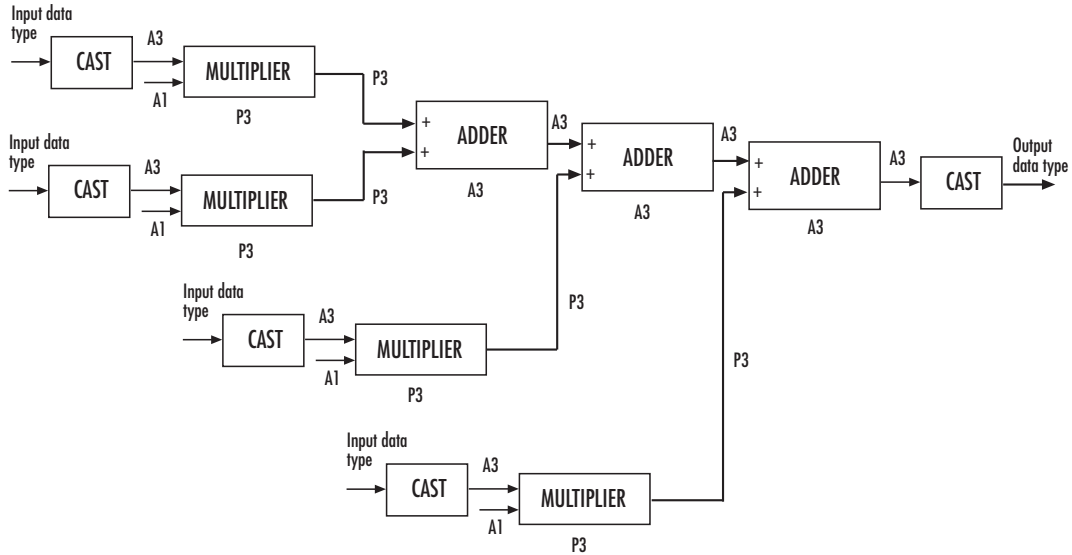
Calculate the bicubic interpolation coefficients, H0, H1, H2, and H3.



The block uses a similar computation to calculate H2 and H3.

Projective Transformation

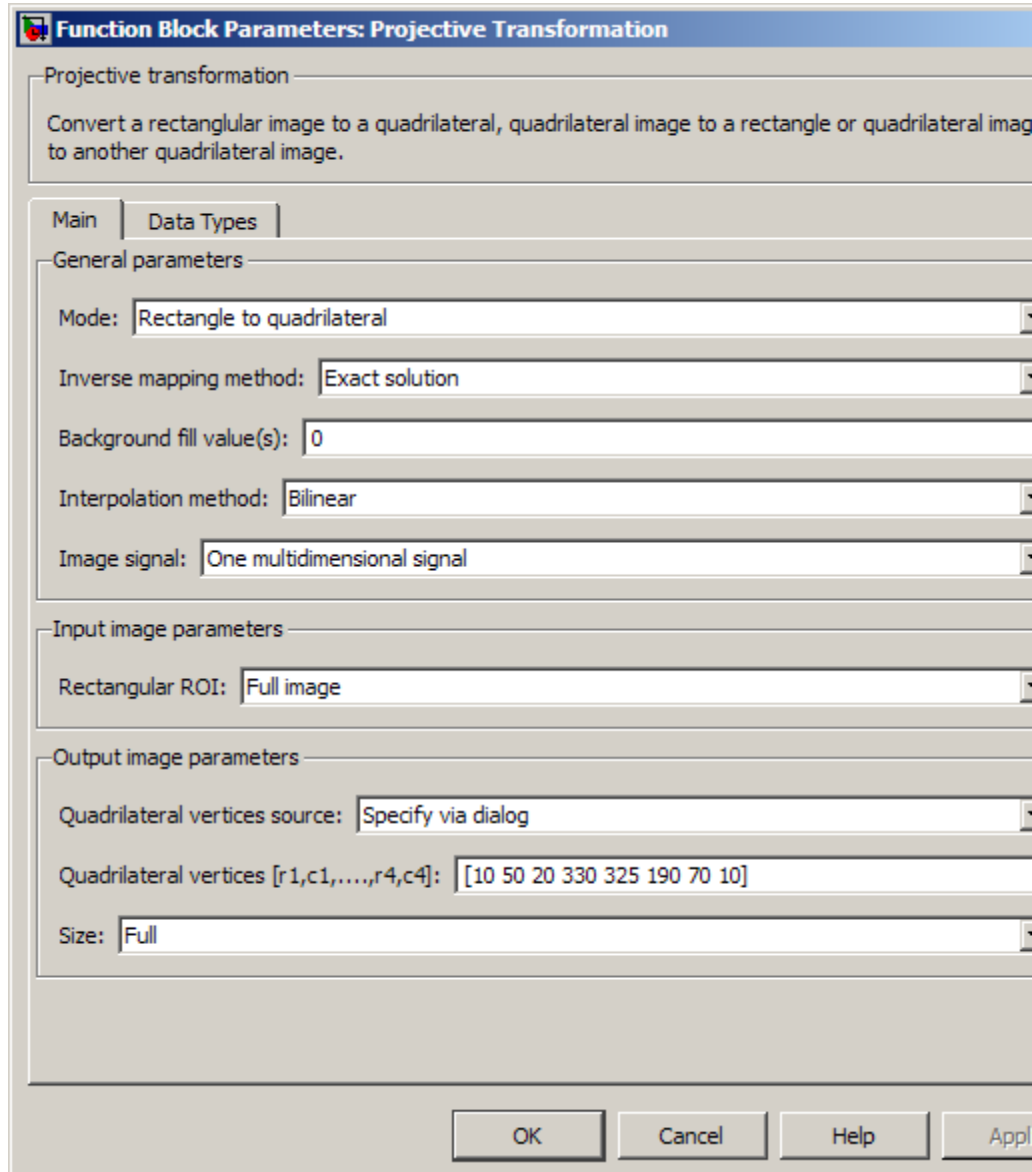
Bicubic Interpolation (continued)



You can set the product, accumulator, matrix, and output data types in the block mask as discussed next.

Dialog Box

The **Main** pane of the Projective Transformation dialog box appears as shown in the following figure.



Projective Transformation

Mode

Select the shape you want to convert. Your choices are Rectangle to quadrilateral, Quadrilateral to rectangle, or Quadrilateral to quadrilateral.

Inverse mapping method

Specify the algorithm the block uses to implement the projective transformation. Your choices are Exact solution or Quadratic approximation.

Quality factor (number of subdivisions)

Enter a scalar integer value greater than or equal to 0 and less than or equal to the height or width of the input image, whichever is smaller. The larger the quality factor, the closer the approximate solution is to the exact solution. This parameter is visible if, for the **Inverse mapping method** parameter, you select Quadratic approximation. Tunable in some modes.

Background fill value(s)

Set the background of the output image. If the block outputs an intensity image, enter a scalar value. If the block outputs an RGB image, enter a scalar value or a three-element vector that specifies an RGB triplet. Tunable in some modes.

Interpolation method

Specify how the block calculates the pixel intensities in the output image. If you select Nearest neighbor, the block uses the value of the nearest pixel for the new pixel intensity. If you select Bilinear, the new pixel value is the weighted average of the four nearest pixel intensities. If you select Bicubic, the new pixel value is the weighted average of the 16 nearest pixel intensities.

Image signal

Specify how to input and output a color video signal. If you select One multidimensional signal, the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select Separate color signals, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

Rectangular ROI

Define the portion of the input image that the block transforms into a quadrilateral. Your choices are **Full image** or **User-defined**.

Rectangular ROI source

Specify whether you want to define the ROI using the Projective Transformation dialog box or an input port. This parameter is visible if, for the **Rectangular ROI** parameter, you select **User-defined**.

ROI [r,c,height,width]

Enter the row and column coordinates of the upper-left corner as well as the height and width of the ROI. This parameter is visible if, for the **Rectangular ROI source** parameter, you select **Specify via dialog**. Tunable in some modes.

If ROI is invalid

Specify the block's behavior if the four-element vector input to the InROI port contains values that are outside the input image. Your choices are **Clip**, **Clip and warn**, or **Error**. During code generation with Real-Time Workshop, this parameter is automatically set to **Clip**. This parameter is visible if, for the **Rectangular ROI source** parameter, you select **Input port**.

Quadrilateral vertices source

Specify how to define the quadrilateral vertices. Your choices are **Specify via dialog** or **Input port**.

Quadrilateral vertices [r1,c1,...,r4,c4]

Enter an eight-element vector of values that represent the row and column coordinates of the four corners of the quadrilateral. This parameter is visible if, for the **Quadrilateral vertices source** parameter, you select **Specify via dialog**.

Size

Specify the size of the output image. If you select **Full**, the block output size is determined by the values you enter for the **Quadrilateral vertices [r1,c1,...,r4,c4]** or **Rectangle location and size [r,c,height,width]** parameter. If you select

Projective Transformation

User-defined, the **Location and size [r,c,height,width]** parameter appears in the dialog box. This parameter is visible if, for the **Quadrilateral vertices source** parameter or **Rectangle size source** parameter, you select Specify via dialog.

Location and size [r,c,height,width]

Define the row and column coordinates as well as the height and width of the output image. This parameter is visible if, for the **Quadrilateral vertices source** or **Rectangle size source** parameter, you select Input port or if, for the **Size** parameter, you select User-defined.

If vertices are outside input image

Specify the block's behavior if the input to the InPts port is invalid. Your choices are Clip, Clip and warn, or Error. During code generation with Real-Time Workshop, this parameter is automatically set to Clip. This parameter is visible if, for the **Mode** parameter, you select Quadrilateral to rectangle or Quadrilateral to quadrilateral and, for the **Quadrilateral vertices source** parameter, you select Input port.

Output validity of quadrilateral vertices (three points cannot be collinear)

Select this check box if you want the block to output 0 at the Valid port if three quadrilateral vertices are collinear. Otherwise, the block outputs 1 at this port.

Rectangle size source

Specify how to define the rectangle size. Your choices are Specify via dialog and Input port.

Rectangle location and size [r,c,height,width]

Enter scalar values that represent the row and column coordinates as well as the height and width of the output rectangle. This parameter is visible if, for the **Rectangle size source** parameter, you select Specify via dialog.

The **Data Types** pane of the Projective Transformation dialog box appears as follows.

Projective Transformation

Function Block Parameters: Projective Transformation

Projective transformation
Convert a rectangular image to a quadrilateral, quadrilateral image to a rectangle or quadrilateral image to another quadrilateral image.

Main | Data Types

Settings on this pane only apply when block inputs are fixed-point signals.

Fixed-point operational parameters

Rounding mode: Overflow mode:

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Product 1	<input type="text" value="Binary point scaling"/>	Yes	<input type="text" value="32"/>	<input type="text" value="18"/>
Accumulator 1	<input type="text" value="Same as Product 1"/>			
Product 2	<input type="text" value="Binary point scaling"/>	Yes	<input type="text" value="32"/>	<input type="text" value="20"/>
Accumulator 2	<input type="text" value="Same as Product 2"/>			
Product 3	<input type="text" value="Binary point scaling"/>	Yes	<input type="text" value="32"/>	<input type="text" value="15"/>
Accumulator 3	<input type="text" value="Same as Product 3"/>			
Matrix	<input type="text" value="Binary point scaling"/>	Yes	<input type="text" value="32"/>	<input type="text" value="20"/>
Output	<input type="text" value="Same as first input"/>			

Lock data type settings against changes by the fixed-point tools

Projective Transformation

Rounding mode

Select the rounding mode for fixed-point operations. For Boolean input, the **Product 3** and **Accumulator 3 Rounding mode** parameter is always set to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations.

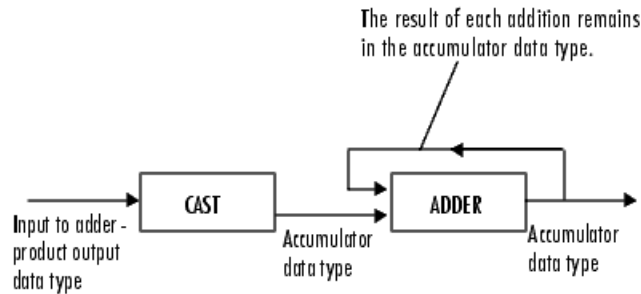
Product 1, 2, 3



As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate the product output word and fraction lengths.

- When you select **Same as input**, the characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output in bits.
- When you select **Slope and bias scaling**, you can enter the word length in bits and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Accumulator 1, 2, 3, 4



As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

- When you select **Same** as Product 1, 2, 3, these characteristics match those of the product output.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator in bits.
- When you select **Slope and bias scaling**, you can enter the word length in bits and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Matrix

Choose how to specify the word length and fraction length of the matrix data type:

- When you select **Binary point scaling**, you can enter the word length and the fraction length of the quotient, in bits.
- When you select **Slope and bias scaling**, you can enter the word length in bits and the slope of the quotient. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Projective Transformation

Output

Choose how to specify the word length and fraction length of the output data type:

- When you select `Same as first input`, these characteristics match those of the first input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the effectiveness metric in bits.
- When you select `Slope and bias scaling`, you can enter the word length in bits and the slope of the effectiveness metric. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

References

[1] Wolberg, George. *Digital Image Warping*. Washington: IEEE Computer Society Press, 1990.

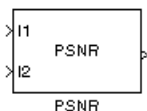
See Also

Resize	Video and Image Processing Blockset software
Rotate	Video and Image Processing Blockset software
Shear	Video and Image Processing Blockset software
Translate	Video and Image Processing Blockset software

Purpose Compute peak signal-to-noise ratio (PSNR) between images

Library Statistics

Description



The PSNR block computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is often used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image.

The *Mean Square Error (MSE)* and the *Peak Signal to Noise Ratio (PSNR)* are the two error metrics used to compare image compression quality. The MSE represents the cumulative squared error between the compressed and the original image, whereas PSNR represents a measure of the peak error. The lower the value of MSE, the lower the error.

To compute the PSNR, the block first calculates the mean-squared error using the following equation:

$$MSE = \frac{\sum_{M,N} [I_1(m,n) - I_2(m,n)]^2}{M * N}$$

In the previous equation, M and N are the number of rows and columns in the input images, respectively. Then the block computes the PSNR using the following equation:

$$PSNR = 10 \log_{10} \left(\frac{R^2}{MSE} \right)$$

In the previous equation, R is the maximum fluctuation in the input image data type. For example, if the input image has a double-precision floating-point data type, then R is 1. If it has an 8-bit unsigned integer data type, R is 255, etc.

Recommendation for Computing PSNR for Color Images

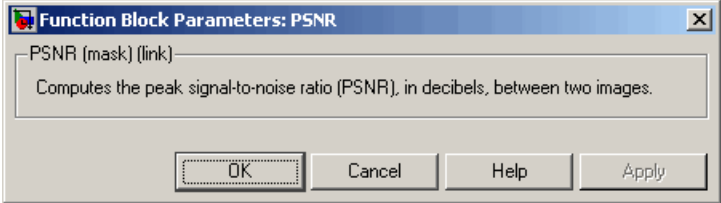
Different approaches exist for computing the PSNR of a color image. Because the human eye is most sensitive to luma information, compute the PSNR for color images by converting the image to a color space that separates the intensity (luma) channel, such as YCbCr. The Y (luma), in YCbCr represents a weighted average of R, G, and B. G is given the most weight, again because the human eye perceives it most easily. With this consideration, compute the PSNR only on the luma channel.

Ports

Port	Output	Supported Data Types	Complex Values Supported
I1	Scalar, vector, or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No
I2	Scalar, vector, or matrix of intensity values	Same as I1 port	No
Output	Scalar value that represents the PSNR	<ul style="list-style-type: none">• Double-precision floating point <p>For fixed-point or integer input, the block output is double-precision floating point. Otherwise, the block input and output are the same data type.</p>	No

**Dialog
Box**

The PSNR dialog box appears as shown in the following figure.



Read AVI File (Obsolete)

Purpose Read uncompressed video frames from AVI file

Library vipobslib

Description



Note The Read AVI File block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset blocks. Use the replacement block From Multimedia File.

The Read AVI File block reads video frames from an uncompressed AVI file and import them into a Simulink model. You can view the video frames using a To Video Display block or Video Viewer block. This block does not support audio samples. Also, this block is supported for simulation only. It produces an error during Real-Time Workshop code generation.

The output ports of the Read AVI File block change according the content of the AVI file. If the file contains RGB video frames, the R, G, and B ports appear on the block. If the file contains intensity video frames, the I port appears on the block.

Port	Output	Supported Data Types	Complex Values Supported
I	Scalar, vector, or matrix of intensity values	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • 8-, 16- 32-bit signed integer • 8-, 16- 32-bit unsigned integer 	No
R, G, B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Outputs from the R, G,	Same as I port	No

Read AVI File (Obsolete)

Port	Output	Supported Data Types	Complex Values Supported
	or B ports have the same dimensions.		
EOF	Scalar value	Boolean	No

Use the **File name** parameter to specify the name of the AVI file from which to read. If the location of this file is on your MATLAB path, enter the filename. If the location of this file is not on your MATLAB path, use the **Browse** button to specify the full path to the file as well as the filename.

If `filename.avi` has a colormap associated with it, the AVI file must satisfy the following conditions or the block produces an error:

- The colormap must be empty or have 256 values.
- The data must represent an intensity image.
- The pixel values must be 8-bit.

Use the **Video output data type** parameter to set the data type of the values output from the block. You can choose `double`, `single`, `int8`, `uint8`, `int16`, `uint16`, `int32`, `uint32`, and `Inherit from file`. If you choose `double` or `single`, the block scales the input pixels values and outputs values between 0 and 1. If you choose `int8`, `uint8`, `int16`, `uint16`, `int32`, or `uint32`, the blocks scales the input pixel values and outputs values between the minimum and maximum values supported by the chosen data type. If you choose `Inherit from file`, the block does not scale the input pixel values.

Use the **Number of times to play file** parameter to enter the number of times to play the file. The number you enter must be a positive integer or `inf`, to play the file until you stop the simulation.

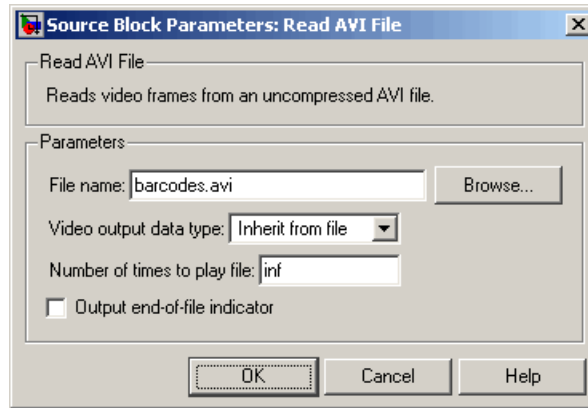
Use the **Output end-of-file indicator** parameter to determine when the last video frame in the AVI file is output from the block. When you

Read AVI File (Obsolete)

select this check box, a Boolean output port labeled EOF appears on the block. The output from the EOF port is 1 when the last video frame is output from the block. Otherwise, the output from the EOF port is 0.

Dialog Box

The Read AVI File dialog box appears as shown in the following figure.



File name

Specify the name of the AVI file from which to read.

Video output data type

Set the data type of the video data output from the block.

Number of times to play file

Enter a positive integer or `inf` to represent the number of times to play the file.

Output end-of-file indicator

Use this check box to determine whether the output is the last video frame in the AVI file.

See Also

From Multimedia File	Video and Image Processing Blockset software
Image From File	Video and Image Processing Blockset software
Image From Workspace	Video and Image Processing Blockset software
To Multimedia File	Signal Processing Blockset software
To Video Display	Video and Image Processing Blockset software
Video From Workspace	Video and Image Processing Blockset software
Video Viewer	Video and Image Processing Blockset software

Read Binary File

Purpose Read binary video data from files

Library Sources

Description The Read Binary File block reads video data from a binary file and imports it into a Simulink model.



This block takes user specified parameters that describe the format of the video data. These parameters together with the raw binary file, which stores only raw pixel values, creates the video data signal for a Simulink model. The video data read by this block must be stored in row major format.

Note This block supports code generation only for platforms that have file I/O available. You cannot use this block to do code generation with RTWin (Real-Time Windows Target™).

Port	Output	Supported Data Types	Complex Values Supported
Output	Scalar, vector, or matrix of integer values	<ul style="list-style-type: none">• 8-, 16- 32-bit signed integer• 8-, 16- 32-bit unsigned integer	No
EOF	Scalar value	Boolean	No

Four Character Code Video Formats

Four Character Codes (FOURCC) identify video formats. For more information about these codes, see <http://www.fourcc.org>.

Use the **Four character code** parameter to identify the binary file format. Then, use the **Rows** and **Cols** parameters to define the size of the output matrix. These dimensions should match the matrix dimensions of the data inside the file.

Custom Video Formats

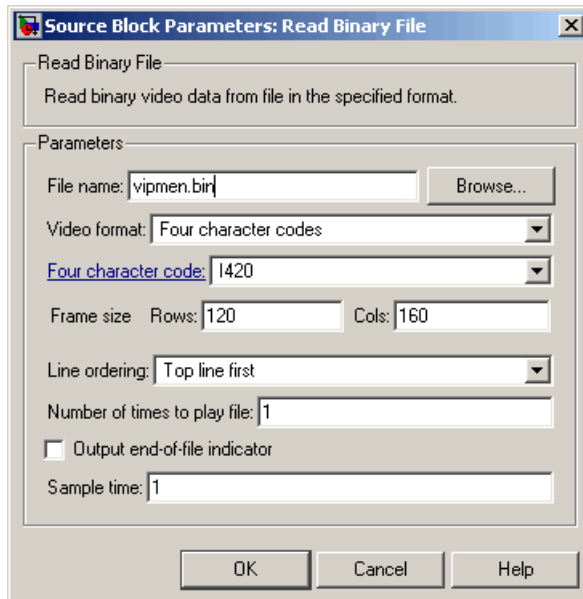
If your binary file contains data that is not in FOURCC format, you can configure the Read Binary File block to understand a custom format:

- Use the **Bit stream format** parameter to specify whether your data is planar or packed. If your data is packed, use the **Rows** and **Cols** parameters to define the size of the output matrix.
- Use the **Number of output components** parameter to specify the number of components in the binary file. This number corresponds to the number of block output ports.
- Use the **Component**, **Bits**, **Rows**, and **Cols** parameters to specify the component name, bit size, and size of the output matrices, respectively. The block uses the **Component** parameter to label the output ports.
- Use the **Component order in binary file** parameter to specify how the components are arranged within the file.
- Select the **Interlaced video** check box if the binary file contains interlaced video data.
- Select the **Input file has signed data** check box if the binary file contains signed integers.
- Use the **Byte order in binary file** to indicate whether your binary file has little endian or big endian byte ordering.

Read Binary File

Dialog Box

The Read Binary File dialog box appears as shown in the following figure.



File name

Specify the name of the binary file to read. If the location of this file is on your MATLAB path, enter the filename. If the location of this file is not on your MATLAB path, use the **Browse** button to specify the full path to the file as well as the filename.

Video format

Specify the format of the binary video data. Your choices are **Four character codes** or **Custom**. See “Four Character Code Video Formats” on page 2-558 or “Custom Video Formats” on page 2-559 for more details.

Four character code

From the drop-down list, select the binary file format.

Frame size: Rows, Cols

Define the size of the output matrix. These dimensions should match the matrix dimensions of the data inside the file.

Line ordering

Specify how the block fills the output matrix. If you select `Top line first`, the block first fills the first row of the output matrix with the contents of the binary file. It then fills the other rows in increasing order. If you select `Bottom line first`, the block first fills the last row of the output matrix. It then fills the other rows in decreasing order.

Number of times to play file

Specify the number of times to play the file. The number you enter must be a positive integer or `inf`, to play the file until you stop the simulation.

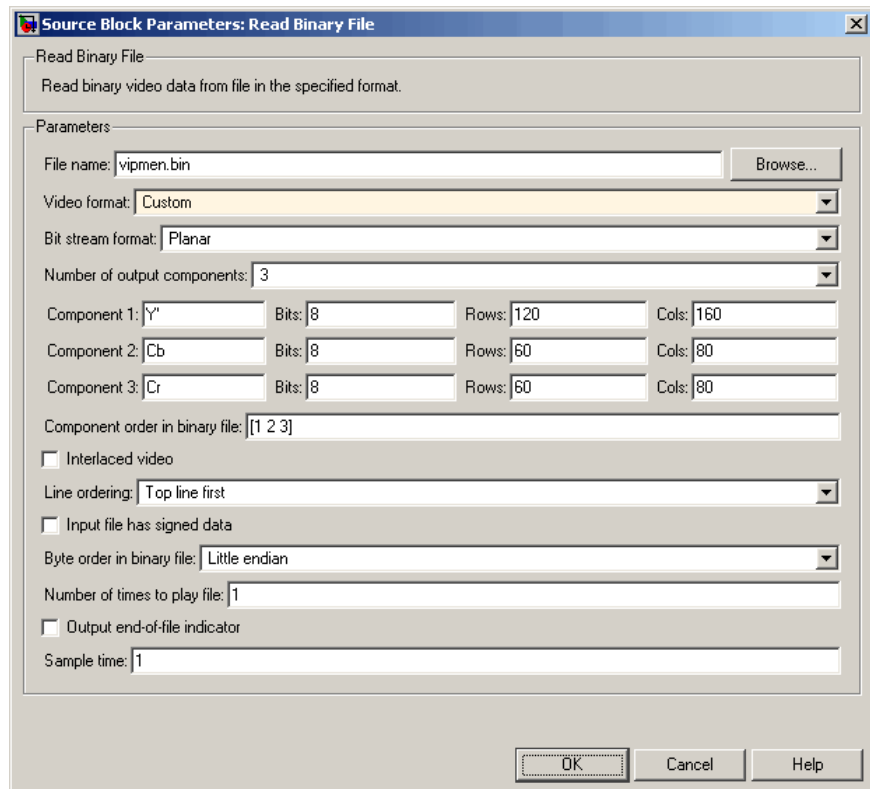
Output end-of-file indicator

Specifies the output is the last video frame in the binary file. When you select this check box, a Boolean output port labeled `EOF` appears on the block. The output from the `EOF` port is 1 when the last video frame in the binary file is output from the block. Otherwise, the output from the `EOF` port is 0.

Sample time

Specify the sample period of the output signal.

Read Binary File



Bit stream format

Specify whether your data is planar or packed.

Frame size: Rows, Cols

Define the size of the output matrix. This parameter appears when you select a **Bit stream format** parameter of Packed.

Number of output components

Specify the number of components in the binary file.

Component, Bits, Rows, Cols

Specify the component name, bit size, and size of the output matrices, respectively.

Component order in binary file

Specify the order in which the components appear in the binary file.

Interlaced video

Select this check box if the binary file contains interlaced video data.

Input file has signed data

Select this check box if the binary file contains signed integers.

Byte order in binary file

Use this parameter to indicate whether your binary file has little endian or big endian byte ordering.

See Also

From Multimedia File	Video and Image Processing Blockset
Write Binary File	Video and Image Processing Blockset

Resize

Purpose Enlarge or shrink image sizes

Library Geometric Transformations
vipgeotforms

Description The Resize block enlarges or shrinks an image by resizing the image along one dimension (row or column). Then, it resizes the image along the other dimension (column or row).



Note This block supports intensity and color images on its ports.

Port	Input/Output	Supported Data Types	Complex Values Supported
Image / Input	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer	No
ROI	Four-element vector that defines the ROI	<ul style="list-style-type: none">• Double-precision floating point (only supported if the input to the Input port is floating point)• Single-precision floating point (only supported if the input to the Input port is floating point)• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Output	Resized image	Same as Input port	No
Flag	Boolean value that indicates whether the ROI is within the image bounds	Boolean	No

If the data type of the input signal is floating point, the output has the same data type.

Use the **Specify** parameter to designate the parameters to use to resize your image. Your choices are Output size as a percentage of input size, Number of output columns and preserve aspect ratio, Number of output rows and preserve aspect ratio, Number of output rows and columns.

If, for the **Specify** parameter, you select Output size as a percentage of input size, the **Resize factor in %** parameter appears in the dialog box. Enter a scalar percentage value that is applied to both rows and columns. You must enter a scalar value that is greater than 0. For a $0 < \text{resize factor} < 100$, the block shrinks the image. For $\text{resize factor} = 100$, the block does not change the image. For $\text{resize factor} > 100$, the block enlarges the image. The dimensions of the output matrix depend on the **Resize factor in %** parameter and are given by the following equations:

```
number_output_rows =
round(number_input_rows*resize_factor/100);

number_output_cols =
round(number_input_cols*resize_factor/100);
```

Alternatively, you can enter a two-element vector, where the first element is the percentage by which to resize the rows and the second element is the percentage by which to resize the columns.

If, for the **Specify** parameter, you select Number of output columns and preserve aspect ratio, the **Number of output columns**

Resize

parameter appears in the dialog box. Enter a scalar value that represents the number of columns you want the output image to have. The block calculates the number of output rows so that the output image has the same aspect ratio as the input image.

If, for the **Specify** parameter, you select `Number of output rows and preserve aspect ratio`, the **Number of output rows** parameter appears in the dialog box. Enter a scalar value that represents the number of rows you want the output image to have. The block calculates the number of output columns so that the output image has the same aspect ratio as the input image.

If, for the **Specify** parameter, you select `Number of output rows and columns`, the **Number of output rows and columns** parameter appears in the dialog box. Enter a two-element vector, where the first element is the number of rows in the output image and the second element is the number of columns. In this case, the aspect ratio of the image can change.

Use the **Interpolation method** parameter to specify which interpolation method the block uses to resize the image. If you select `Nearest neighbor`, the block uses one nearby pixel to interpolate the pixel value. This selection is the most computationally efficient, but it is the least accurate. If you select `Bilinear`, the block uses four nearby pixels to interpolate the pixel value. If you select `Bicubic` or `Lanczos2`, the block uses 16 nearby pixels to interpolate the pixel value. If you select `Lanczos3`, the block uses 36 surrounding pixels to interpolate the pixel value.

The Resize block performs optimally when the **Interpolation method** parameter is set to `Nearest neighbor` and one of the following conditions is met:

- The **Resize factor in %** parameter is a multiple of 100.
- Dividing 100 by the **Resize factor in %** parameter value results in an integer value.

Shrinking an image can introduce high frequency components into the image and aliasing might occur. If you select the **Perform antialiasing when resize factor is between 0 and 100** check box, the block performs low pass filtering on the input image before shrinking it.

ROI Processing

To resize a particular region of each image, select the **Enable ROI processing** check box. This option is available under these conditions:

- **Specify** = Number of output rows and columns
- **Interpolation method** = Nearest neighbor, Bilinear, or Bicubic
- Clear the **Perform antialiasing when resize factor is between 0 and 100** check box.

If you select the **Enable ROI processing** check box, the ROI port appears on the block. Use this port to define a region of interest (ROI) in the input matrix, I, that you want to resize. The input to this port must be a four-element vector, [row column height width]. The first two elements define the upper-left corner of the ROI, and the second two elements define the height and width of the ROI.

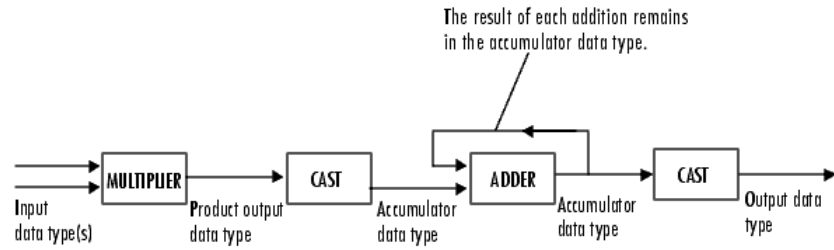
If you select the **Enable ROI processing** check box, the **Output flag indicating if any part of ROI is outside image bounds** check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output.

Flag Port Output	Description
0	ROI is completely inside the input image.
1	ROI is completely or partially outside the input image.

Fixed-Point Data Types

The following diagram shows the data types used in the Resize block for fixed-point signals.

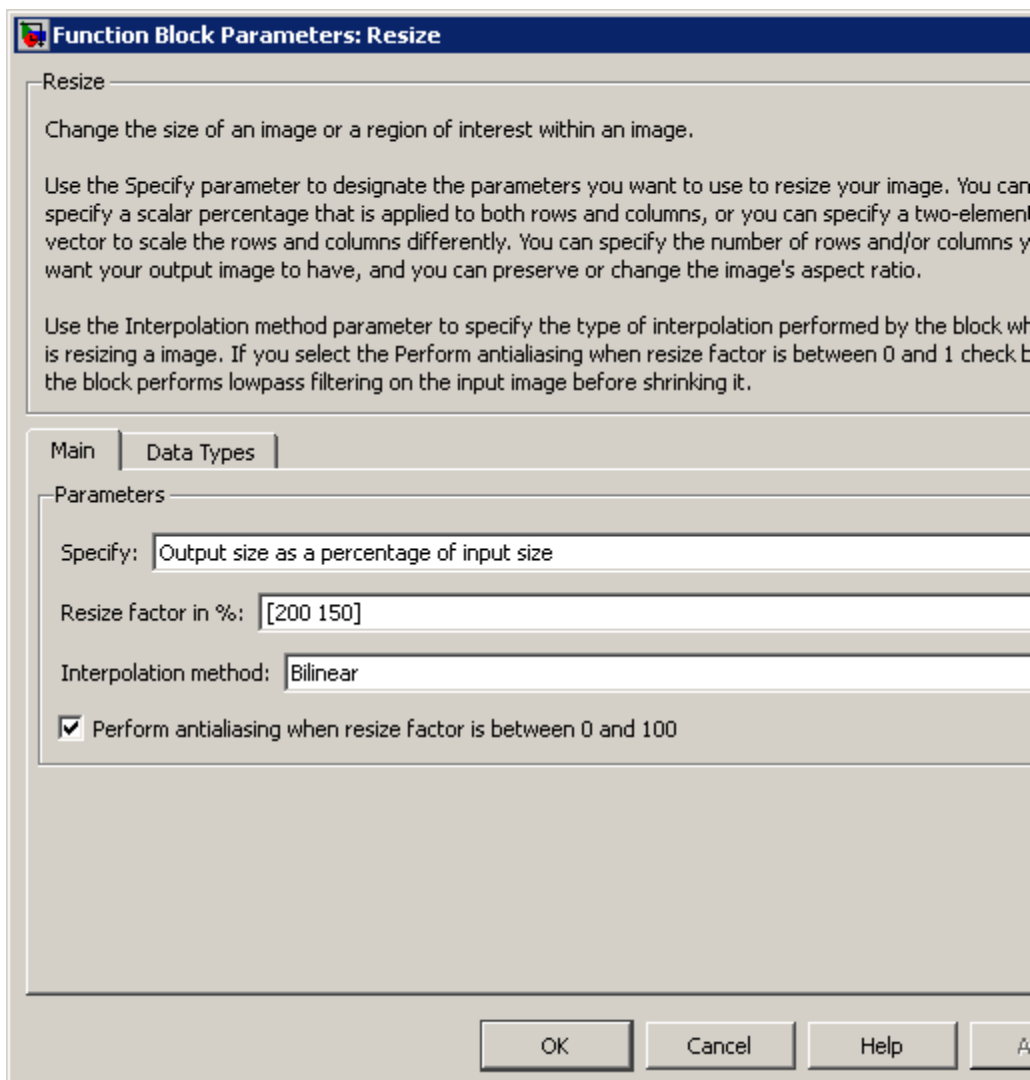
Resize



You can set the interpolation weights table, product output, accumulator, and output data types in the block mask.

Dialog Box

The **Main** pane of the Resize dialog box appears as shown in the following figure:



Resize

Specify

Specify which aspects of the image to resize. Your choices are Output size as a percentage of input size, Number of output columns and preserve aspect ratio, Number of output rows and preserve aspect ratio, Number of output rows and columns.

Resize factor in %

Enter a scalar percentage value that is applied to both rows and columns or a two-element vector, where the first element is the percentage by which to resize the rows and the second element is the percentage by which to resize the columns. This parameter is visible if, for the **Specify** parameter, you select Output size as a percentage of input size.

Number of output columns

Enter a scalar value that represents the number of columns you want the output image to have. This parameter is visible if, for the **Specify** parameter, you select Number of output columns and preserve aspect ratio.

Number of output rows

Enter a scalar value that represents the number of rows you want the output image to have. This parameter is visible if, for the **Specify** parameter, you select Number of output rows and preserve aspect ratio.

Number of output rows and columns

Enter a two-element vector, where the first element is the number of rows in the output image and the second element is the number of columns. This parameter is visible if, for the **Specify** parameter, you select Number of output rows and columns.

Interpolation method

Determine which interpolation method the block uses to resize the image. If you select **Nearest neighbor**, the block uses one nearby pixel to interpolate the pixel value. If you select **Bilinear**, the block uses two nearby pixels to interpolate the pixel value. If you select **Bicubic** or **Lanczos2**, the block uses four nearby pixels to

interpolate the pixel value. If you select Lanczos3, the block uses six surrounding pixels to interpolate the pixel value.

Perform antialiasing when resize factor is between 0 and 100

If you select this check box, the block performs low-pass filtering on the input image before shrinking it to prevent aliasing.

Enable ROI processing

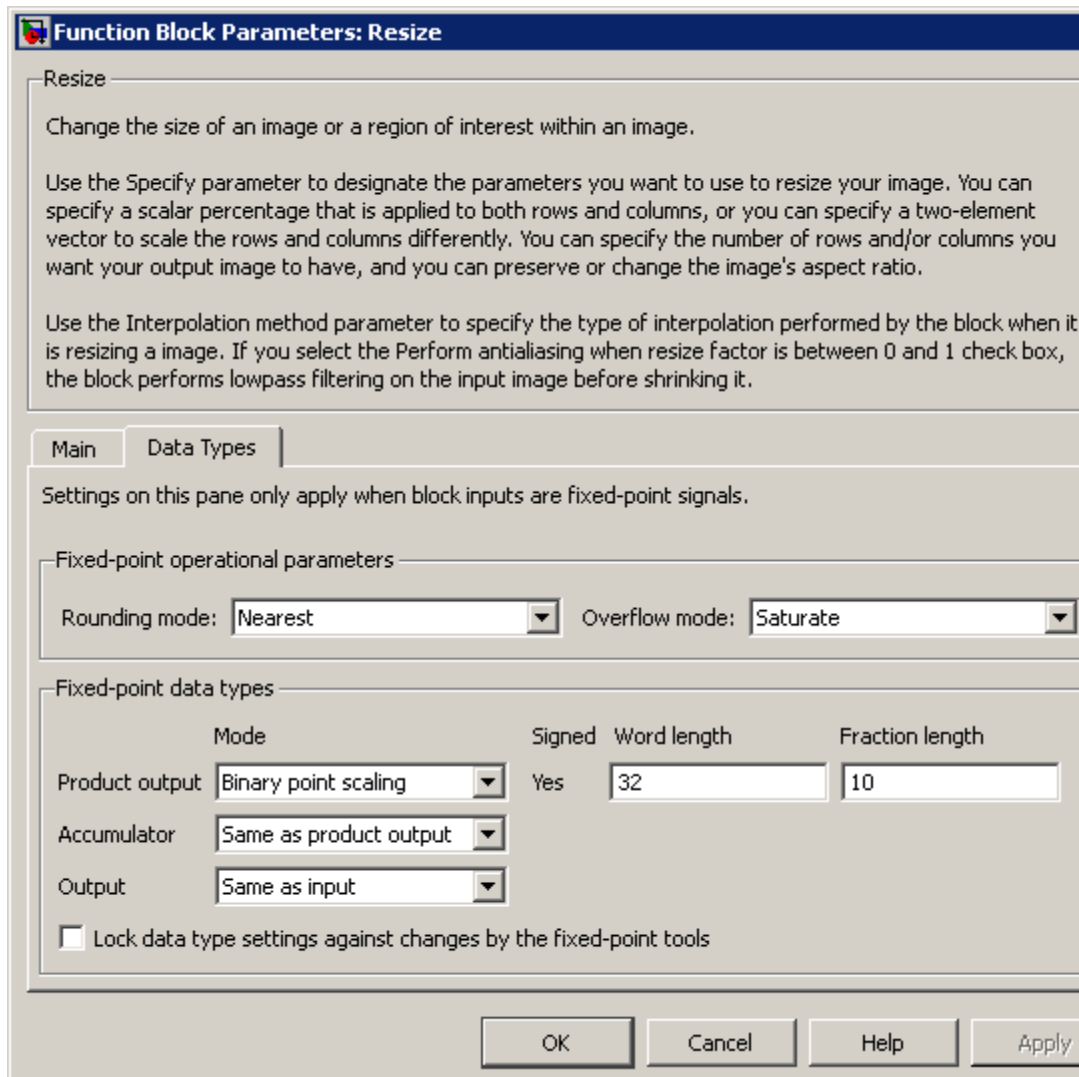
Select this check box to resize a particular region of each image. This parameter is available when the **Specify** parameter is set to Number of output rows and columns, the **Interpolation method** parameter is set to Nearest neighbor, Bilinear, or Bicubic, and the **Perform antialiasing when resize factor is between 0 and 100** check box is not selected.

Output flag indicating if any part of ROI is outside image bounds

If you select this check box, the Flag port appears on the block. The block outputs 1 at this port if the ROI is completely or partially outside the input image. Otherwise, it outputs 0.

The **Data Types** pane of the Resize dialog box appears as shown in the following figure.

Resize



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Interpolation weights table

Choose how to specify the word length of the values of the interpolation weights table. The fraction length of the interpolation weights table values is always equal to the word length minus one:

- When you select **Same as input**, the word length of the interpolation weights table values match that of the input to the block.
- When you select **Binary point scaling**, you can enter the word length of the interpolation weights table values, in bits.
- When you select **Slope and bias scaling**, you can enter the word length of the interpolation weights table values, in bits.

Product output



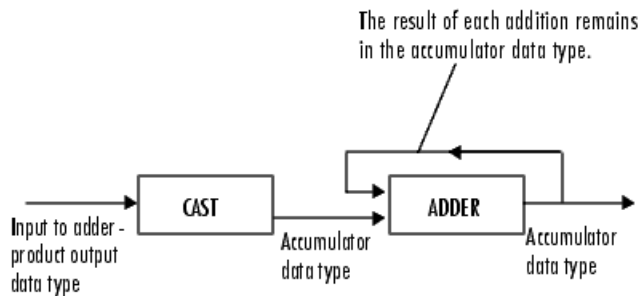
As depicted in the preceding diagram, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths.

- When you select **Same as input**, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.

Resize

- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Accumulator



As depicted in the preceding diagram, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

- When you select **Same as product output**, these characteristics match those of the product output.
- When you select **Same as input**, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Output

Choose how to specify the word length and fraction length of the output of the block:

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

References

- [1] Ward, Joseph and David R. Cok. "Resampling Algorithms for Image Resizing and Rotation", *Proc. SPIE Digital Image Processing Applications*, vol. 1075, pp. 260-269, 1989.
- [2] Wolberg, George. *Digital Image Warping*. Washington: IEEE Computer Society Press, 1990.

See Also

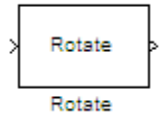
Rotate	Video and Image Processing Blockset software
Shear	Video and Image Processing Blockset software
Translate	Video and Image Processing Blockset software
<code>imresize</code>	Image Processing Toolbox software

Rotate

Purpose Rotate image by specified angle

Library Geometric Transformations

Description Use the Rotate block to rotate an image by an angle specified in radians.



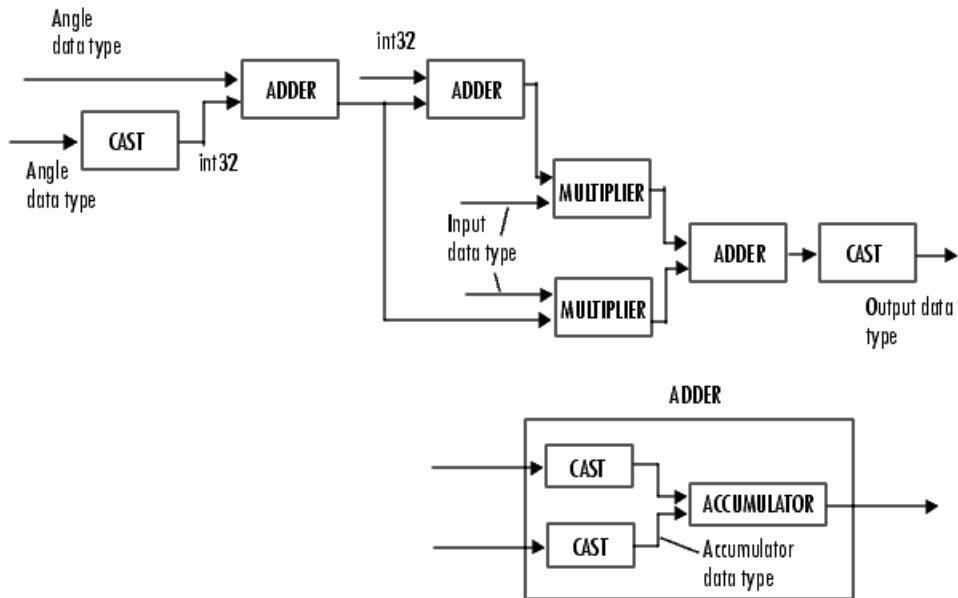
Note This block supports intensity and color images on its ports.

Port	Description
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes
Angle	Rotation angle
Output	Rotated matrix

The Rotate block uses the 3-pass shear rotation algorithm to compute its values, which is different than the algorithm used by the `imrotate` function in the Image Processing Toolbox.

Fixed-Point Data Types

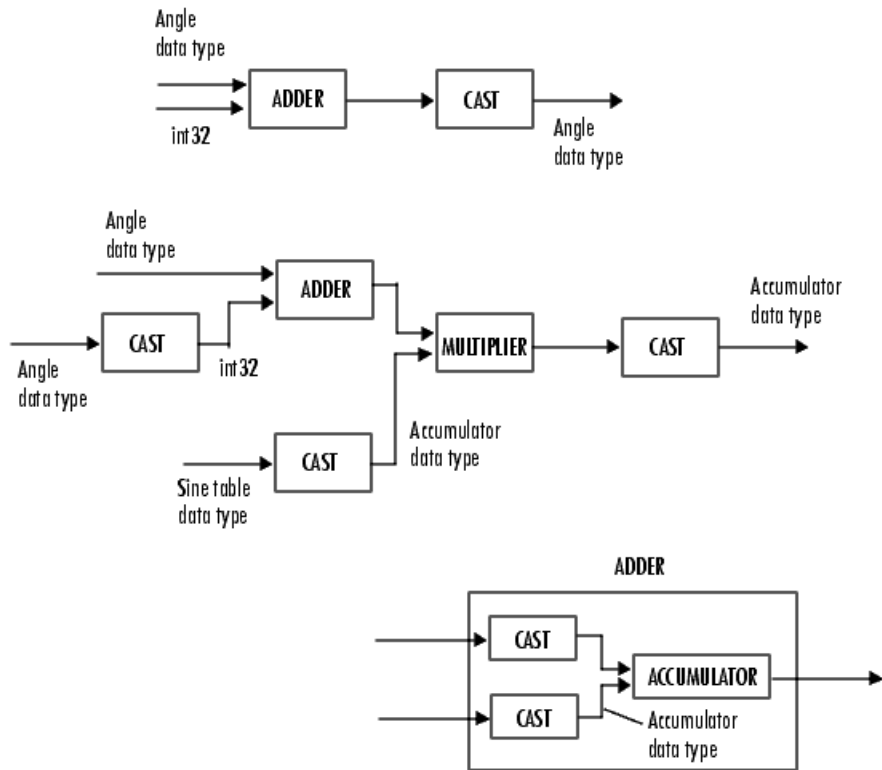
The following diagram shows the data types used in the Rotate block for bilinear interpolation of fixed-point signals.



You can set the angle values, product output, accumulator, and output data types in the block mask.

The Rotate block requires additional data types. The Sine table value has the same word length as the angle data type and a fraction length that is equal to its word length minus one. The following diagram shows how these data types are used inside the block.

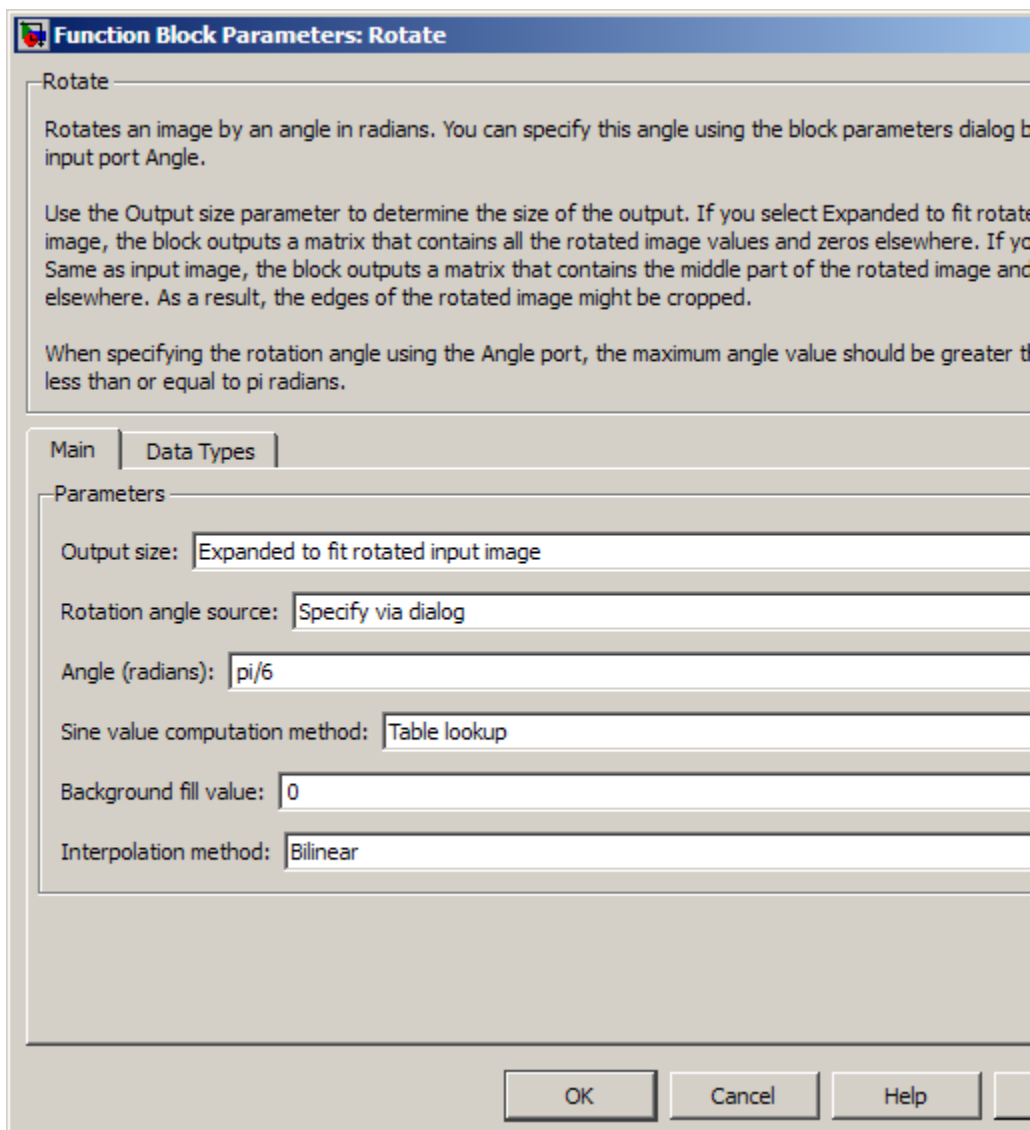
Rotate



Note If overflow occurs, the rotated image might appear distorted.

Dialog Box

The **Main** pane of the Rotate dialog box appears as shown in the following figure.



Output size

Specify the size of the rotated matrix. If you select `Expanded to fit rotated input image`, the block outputs a matrix that contains all the rotated image values. If you select `Same as input image`, the block outputs a matrix that contains the middle part of the rotated image. As a result, the edges of the rotated image might be cropped. Use the **Background fill value** parameter to specify the pixel values outside the image.

Rotation angle source

Specify how to enter your rotation angle. If you select `Specify via dialog`, the **Angle (radians)** parameter appears in the dialog box.

If you select `Input port`, the Angle port appears on the block. The block uses the input to this port at each time step as your rotation angle. The input to the Angle port must be the same data type as the input to the **I** port.

Angle (radians)

Enter a real, scalar value for your rotation angle. This parameter is visible if, for the **Rotation angle source** parameter, you select `Specify via dialog`.

When the rotation angle is a multiple of $\pi/2$, the block uses a more efficient algorithm. If the angle value you enter for the **Angle (radians)** parameter is within 0.00001 radians of a multiple of $\pi/2$, the block rounds the angle value to the multiple of $\pi/2$ before performing the rotation.

Maximum angle (enter pi radians to accommodate all positive and negative angles)

Enter the maximum angle by which to rotate the input image. Enter a scalar value, between 0 and π radians. The block determines which angle, $0 \leq \text{angle} \leq \text{max angle}$, requires the largest output matrix and sets the dimensions of the output port accordingly.

This parameter is visible if you set the **Output size** parameter, to **Expanded to fit rotated input image**, and the **Rotation angle source** parameter to **Input port**.

Display rotated image in

Specify how the image is rotated. If you select **Center**, the image is rotated about its center point. If you select **Top-left corner**, the block rotates the image so that two corners of the rotated input image are always in contact with the top and left sides of the output image.

This parameter is visible if, for the **Output size** parameter, you select **Expanded to fit rotated input image**, and, for the **Rotation angle source** parameter, you select **Input port**.

Sine value computation method

Specify the value computation method. If you select **Trigonometric function**, the block computes sine and cosine values it needs to calculate the rotation of your image during the simulation. If you select **Table lookup**, the block computes and stores the trigonometric values it needs to calculate the rotation of your image before the simulation starts. In this case, the block requires extra memory.

Background fill value

Specify a value for the pixels that are outside the image.

Interpolation method

Specify which interpolation method the block uses to rotate the image. If you select **Nearest neighbor**, the block uses the value of one nearby pixel for the new pixel value. If you select **Bilinear**, the new pixel value is the weighted average of the four nearest pixel values. If you select **Bicubic**, the new pixel value is the weighted average of the sixteen nearest pixel values.

The number of pixels the block considers affects the complexity of the computation. Therefore, the **Nearest-neighbor** interpolation is the most computationally efficient. However, because the accuracy of the method is proportional to the number of pixels

Rotate

considered, the **Bicubic** method is the most accurate. For more information, see “Geometric Transformation Interpolation Methods” in the *Video and Image Processing Blockset User’s Guide*.

The **Data Types** pane of the Rotate dialog box appears as shown in the following figure.

Function Block Parameters: Rotate

Rotate

Rotates an image by an angle in radians. You can specify this angle using the block parameters dialog box input port Angle.

Use the Output size parameter to determine the size of the output. If you select Expanded to fit rotated image, the block outputs a matrix that contains all the rotated image values and zeros elsewhere. If you select Same as input image, the block outputs a matrix that contains the middle part of the rotated image and zeros elsewhere. As a result, the edges of the rotated image might be cropped.

When specifying the rotation angle using the Angle port, the maximum angle value should be greater than or less than or equal to pi radians.

Main | Data Types

Settings on this pane only apply when block inputs are fixed-point signals.

Fixed-point operational parameters

Rounding mode: Overflow mode:

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Angle values	<input type="text" value="Same word length as input"/>			
Product output	<input type="text" value="Binary point scaling"/>	Yes	<input type="text" value="32"/>	<input type="text" value="10"/>
Accumulator	<input type="text" value="Same as product output"/>			
Output	<input type="text" value="Same as first input"/>			

Lock data type settings against changes by the fixed-point tools

OK Cancel Help

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Angle values

Choose how to specify the word length and the fraction length of the angle values.

- When you select **Same word length as input**, the word length of the angle values match that of the input to the block. In this mode, the fraction length of the angle values is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the angle values.
- When you select **Specify word length**, you can enter the word length of the angle values, in bits. The block automatically sets the fraction length to give you the best precision.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the angle values, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the angle values. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

This parameter is only visible if, for the **Rotation angle source** parameter, you select **Specify via dialog**.

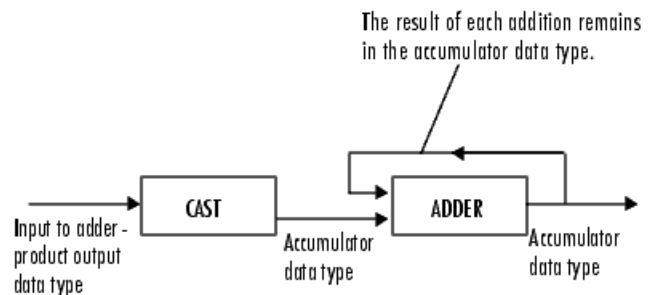
Product output



As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths.

- When you select **Same** as first input, these characteristics match those of the input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Accumulator



As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

- When you select **Same as product output**, these characteristics match those of the product output.
- When you select **Same as first input**, these characteristics match those of the first input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Output

Choose how to specify the word length and fraction length of the output of the block:

- When you select **Same as first input**, these characteristics match those of the first input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

Supported Data Types

Port	Supported Data Types
Image	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer
Angle	Same as Image port
Output	Same as Image port

If the data type of the input signal is floating point, the output signal is the same data type as the input signal.

References

[1] Wolberg, George. *Digital Image Warping*. Washington: IEEE Computer Society Press, 1990.

See Also

Resize	Video and Image Processing Blockset software
Translate	Video and Image Processing Blockset software
Shear	Video and Image Processing Blockset software
imrotate	Image Processing Toolbox software

SAD (Obsolete)

Purpose Perform 2-D sum of absolute differences (SAD)

Library vipobslib

Description



Note The SAD block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset software. Use the replacement block Template Matching.

The SAD block finds the similarity between two input images by performing the sum of absolute differences. The greater the similarity between the two matrices, the smaller the SAD values that result. Assume that input matrix I has dimensions (M_i , N_i) and the input matrix Template has dimensions (M_t , N_t). The equation for the two-dimensional discrete SAD is

$$C(j,k) = \sum_{m=0}^{(M_t-1)} \sum_{n=0}^{(N_t-1)} \text{abs}(I(m+j, n+k) - T(m,n))$$

where

$$0 \leq j < M_i - M_t + 1$$

and

$$0 \leq k < N_i - N_t + 1$$

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• Boolean• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No
Template	Matrix of intensity values	Same as I port	No
ROI	Four-element vector that defines the ROI	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No
Val	Matrix of SAD values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No
Idx	Vector that represents the zero-based index location of the minimum SAD value	<ul style="list-style-type: none">• 32-bit signed integers	No

SAD (Obsolete)

Port	Input/Output	Supported Data Types	Complex Values Supported
NVals	N-by-N matrix of SAD values centered around the minimum SAD value	Same as Val port	No
NValid	Boolean 0 or 1 that represents whether or not the block went beyond the dimensions of the SAD value matrix to construct an N-by-N matrix around the minimum SAD value	Boolean	No

The data type of the two input signals must be the same. The output signal is the same data type as the input signals.

The dimensions of the output at the Val port are determined by the sizes of the inputs at ports I and Template. If the input at port I has dimensions (Mi, Ni) and the input at the Template port dimensions (Mt, Nt), then the output has dimensions (Mi-Mt+1, Ni-Nt+1).

Use the **Output** parameter to determine the output of the block. If you select **SAD values**, the block outputs the SAD values at the Val port. If you select **Minimum SAD value index**, the block outputs the zero-based index location of the minimum SAD value at the Idx port.

If, for the **Output** parameter, you select **Minimum SAD value index**, the **Search method** parameter appears in the dialog box. If you select **Exhaustive**, the block searches the two input matrices for the minimum difference pixel-by-pixel. This process is described by the previous equation and is computationally expensive.

If, for the **Search method** parameter, you select **Three-step**, the block searches the two input matrices for the minimum difference using a steadily decreasing step size. The block begins with a step size approximately equal to half the maximum search range. In each step,

the block compares the central point of the search region to eight search points located on the boundaries of the region and moves the central point to the search point whose values is the closest to that of the central point. The block then decrements the step by one, and begins the process again. The search terminates with a match within one pixel.

If, for the **Output** parameter, you select **Minimum SAD value index**, the **Use ROI for input I** check box appears in the dialog box. If you select this check box, the ROI port appears on the block. Use this port to define a region of interest (ROI) in the input matrix, I, over which you want to compute the SAD. The input to this port must be a four-element vector, [row column height width]. The first two elements define the upper-left corner of the ROI, and the second two elements define the height and width of the ROI.

Use the **Invalid ROI** parameter to specify the block's behavior if you enter a ROI that is outside the bounds of the input matrix, I. The options are

- **Ignore** -- Proceed with the computation and do not issue an alert. The output is not valid.
- **Warn** -- Display a warning message in the MATLAB Command Window, and continue the simulation. The output is not valid.
- **Error** -- Display an error dialog box and terminate the simulation.

If, for the **Output** parameter, you select **Minimum SAD value index**, the **Output NxN matrix of SAD values around minimum** check box appears on the dialog box. If you select this check box, the NVals and NValid ports appear on the block. The block outputs an N-by-N matrix of SAD values centered around the minimum SAD value at the NVals port. Use the **Size (N) of square matrix** parameter to determine the size of this matrix. The value you enter must be a real-valued, odd integer that is greater than or equal to 1.

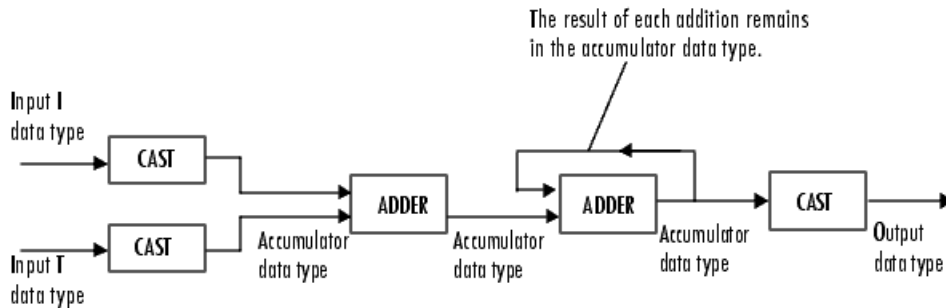
If the block must go beyond the dimensions of the SAD value matrix to construct an N-by-N matrix around the minimum SAD value, the values outside the SAD value matrix are 0. In this case, the block

SAD (Obsolete)

outputs a Boolean 0 at the NValid port. If the block does not go beyond the dimensions of the SAD value matrix to construct an N-by-N matrix around the minimum SAD value, the block outputs a Boolean 1 at the NValid port.

Fixed-Point Data Types

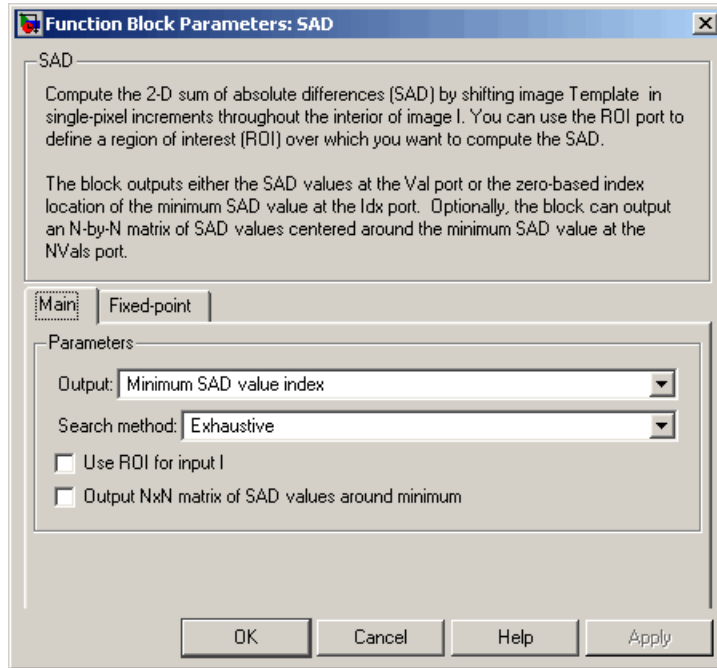
The following diagram shows the data types used in the SAD block for fixed-point signals.



You can set the accumulator, and output data types in the block mask as discussed in the next section.

Dialog Box

The **Main** pane of the SAD dialog box appears as shown in the following figure.



Output

Specify the output of the block. Your choices are **SAD values** or **Minimum SAD value index**. If you select **Minimum SAD value index**, the block outputs the zero-based index location of the minimum SAD value.

Search method

Specify how the block searches for the minimum difference between the two input matrices. If you select **Exhaustive**, the block searches for the minimum difference pixel-by-pixel. If you select **Three-step**, the block searches for the minimum difference

SAD (Obsolete)

using a steadily decreasing step size. This parameter is visible if, for the **Output** parameter, you select Minimum SAD value index.

Use ROI for input I

If you select this check box, the ROI port appears on the block. Use this port to define a region of interest (ROI) in the input matrix, I, over which you want to compute the SAD. This parameter is visible if, for the **Output** parameter, you select Minimum SAD value index.

Invalid ROI

Specify the block's behavior if you enter a ROI that is outside the bounds of the input matrix, I. The options are Ignore, Warn, or Error.

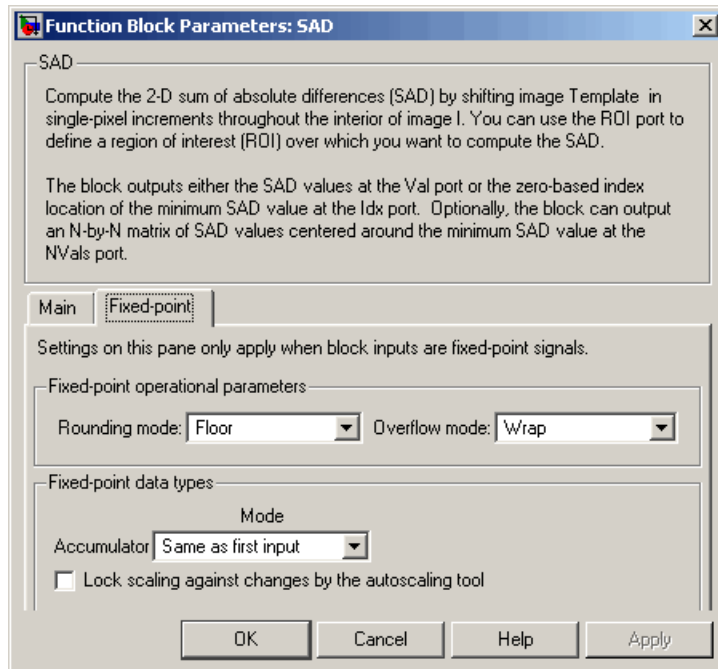
Output NxN matrix of SAD values around minimum

If you select this check box, the NVals and NValid ports appear on the block. The block outputs an N-by-N matrix of SAD values centered around the minimum SAD value at the NVals port. If the block must go beyond the dimensions of the SAD value matrix to construct the N-by-N output matrix, the block outputs a Boolean 0 at the NValid port. Otherwise, the block outputs a Boolean 1 at the NValid port. This parameter is visible if, for the **Output** parameter, you select Minimum SAD value index.

Size (N) of square matrix

Enter an odd number that determines the size of the N-by-N matrix of SAD values. This parameter is visible if you select the **Output NxN matrix of SAD values around minimum** check box.

The **Fixed-point** pane of the SAD dialog box appears as shown in the following figure.



Rounding mode

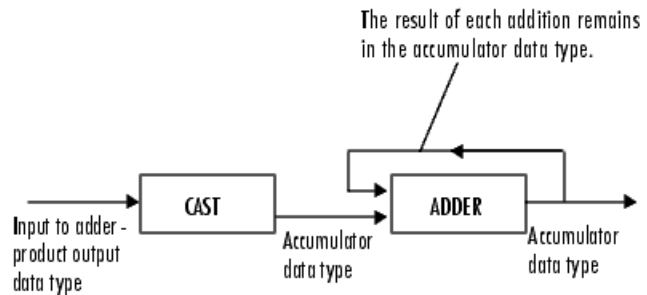
Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

SAD (Obsolete)

Accumulator



As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

- When you select **Same as first input**, these characteristics match those of the input to the block. When you have a Boolean input, you cannot select this choice.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Output

Choose how to specify the word length and fraction length of the output of the block:

- When you select **Same as first input**, these characteristics match those of the first input to the block. When you have a Boolean input, you cannot select this choice.

- When you select **Binary point scaling**, you can enter the word length and the fraction length of the output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

This parameter is not visible if, for the **Output** parameter you select **Minimum SAD value index**, and you clear the **Output NxN matrix of SAD values around minimum** check box.

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

References

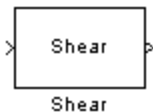
- [1] Koga, T., et al. *Motion-compensated interframe coding for video conferencing*. In Nat. Telecommun. Conf., Nov. 1981, G5.3.1-5, New Orleans, LA.
- [2] Wang, Yao, Jorn Ostermann, Ya-Qin Zhang. *Video Processing and Communications*. Upper Saddle River, NJ: Prentice Hall, 2002.

Shear

Purpose Shift rows or columns of image by linearly varying offset

Library Geometric Transformations
vipgeotforms

Description The Shear block shifts the rows or columns of an image by a gradually increasing distance left or right or up or down.



Note This block supports intensity and color images on its ports.

Port	Input/Output	Supported Data Types	Complex Values Supported
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer	No
S	Two-element vector that represents the number of pixels by which you want to shift your first and last rows or columns	Same as I port	No
Output	Shifted image	Same as I port	No

If the data type of the input to the I port is floating point, the input to the S port of this block must be the same data type. Also, the block output is the same data type.

Use the **Shear direction** parameter to specify whether you want to shift the rows or columns. If you select **Horizontal**, the first row

has an offset equal to the first element of the **Row/column shear values [first last]** vector. The following rows have an offset that linearly increases up to the value you enter for the last element of the **Row/column shear values [first last]** vector. If you select **Vertical**, the first column has an offset equal to the first element of the **Row/column shear values [first last]** vector. The following columns have an offset that linearly increases up to the value you enter for the last element of the **Row/column shear values [first last]** vector.

Use the **Output size after shear** parameter to specify the size of the sheared image. If you select **Full**, the block outputs a matrix that contains the entire sheared image. If you select **Same as input image**, the block outputs a matrix that is the same size as the input image and contains the top-left portion of the sheared image. Use the **Background fill value** parameter to specify the pixel values outside the image.

Use the **Shear values source** parameter to specify how to enter your shear parameters. If you select **Specify via dialog**, the **Row/column shear values [first last]** parameter appears in the dialog box. Use this parameter to enter a two-element vector that represents the number of pixels by which you want to shift your first and last rows or columns. For example, if for the **Shear direction** parameter you select **Horizontal** and, for the **Row/column shear values [first last]** parameter, you enter **[50 150]**, the block moves the top-left corner 50 pixels to the right and the bottom left corner of the input image 150 pixels to the right. If you want to move either corner to the left, enter negative values. If for the **Shear direction** parameter you select **Vertical** and, for the **Row/column shear values [first last]** parameter, you enter **[-10 50]**, the block moves the top-left corner 10 pixels up and the top right corner 50 pixels down. If you want to move either corner down, enter positive values.

Use the **Interpolation method** parameter to specify which interpolation method the block uses to shear the image. If you select **Nearest neighbor**, the block uses the value of the nearest pixel for the new pixel value. If you select **Bilinear**, the new pixel value is the weighted average of the two nearest pixel values. If you select **Bicubic**,

the new pixel value is the weighted average of the four nearest pixel values.

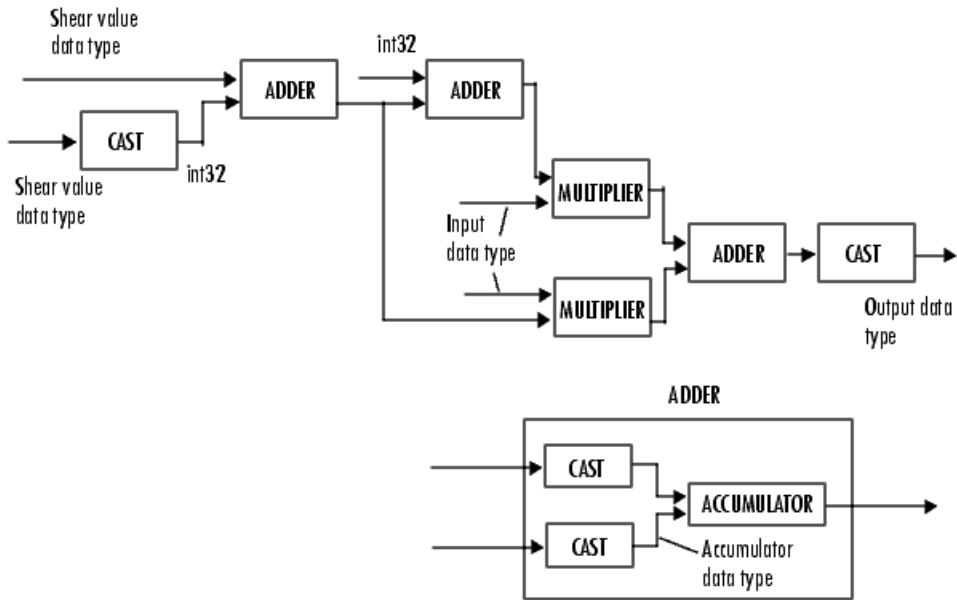
The number of pixels the block considers affects the complexity of the computation. Therefore, the nearest-neighbor interpolation is the most computationally efficient. However, because the accuracy of the method is proportional to the number of pixels considered, the bicubic method is the most accurate. For more information, see “Geometric Transformation Interpolation Methods” in the *Video and Image Processing Blockset User’s Guide*.

If, for the **Shear values source** parameter, you select `Input port`, the `S` port appears on the block. At each time step, the input to the `S` port must be a two-element vector that represents the number of pixels by which to shift your first and last rows or columns.

If, for the **Output size after shear** parameter, you select `Full`, and for the **Shear values source** parameter, you select `Input port`, the **Maximum shear value** parameter appears in the dialog box. Use this parameter to enter a real, scalar value that represents the maximum number of pixels by which to shear your image. The block uses this parameter to determine the size of the output matrix. If any input to the `S` port is greater than the absolute value of the **Maximum shear value** parameter, the block saturates to the maximum value.

Fixed-Point Data Types

The following diagram shows the data types used in the Shear block for bilinear interpolation of fixed-point signals.

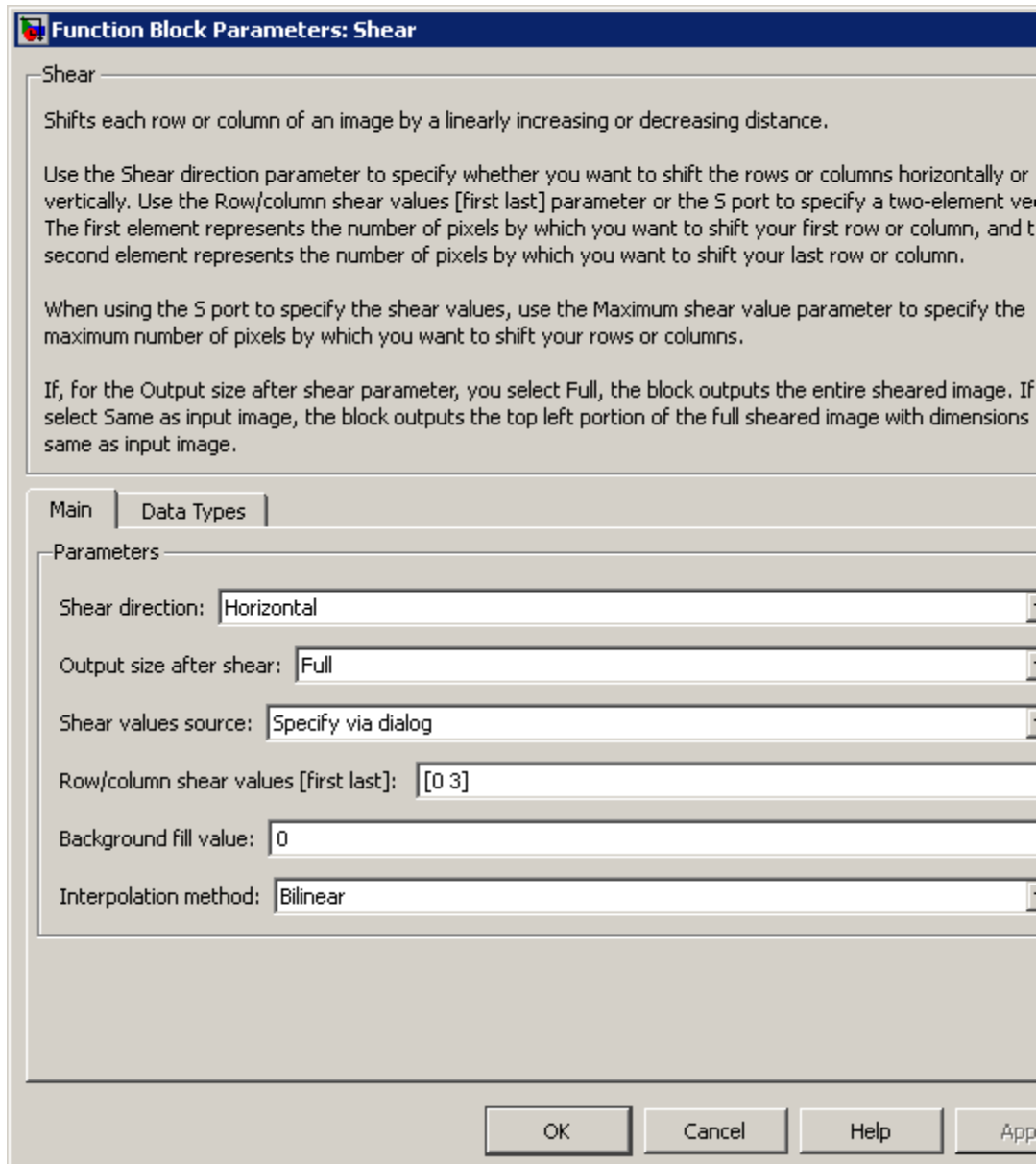


You can set the product output, accumulator, and output data types in the block mask.

Shear

Dialog Box

The **Main** pane of the Shear dialog box appears as shown in the following figure.



Shear direction

Specify whether you want to shift the rows or columns of the input image. Select **Horizontal** to linearly increase the offset of the rows. Select **Vertical** to steadily increase the offset of the columns.

Output size after shear

Specify the size of the sheared image. If you select **Full**, the block outputs a matrix that contains the sheared image values. If you select **Same as input image**, the block outputs a matrix that is the same size as the input image and contains a portion of the sheared image.

Shear values source

Specify how to enter your shear parameters. If you select **Specify via dialog**, the **Row/column shear values [first last]** parameter appears in the dialog box. If you select **Input port**, port **S** appears on the block. The block uses the input to this port at each time step as your shear value.

Row/column shear values [first last]

Enter a two-element vector that represents the number of pixels by which to shift your first and last rows or columns. This parameter is visible if, for the **Shear values source** parameter, you select **Specify via dialog**.

Maximum shear value

Enter a real, scalar value that represents the maximum number of pixels by which to shear your image. This parameter is visible if, for the **Output size after shear** parameter, you select **Full** and, for the **Shear values source** parameter, you select **Input port**.

Background fill value

Specify a value for the pixels that are outside the image.

Interpolation method

Specify which interpolation method the block uses to shear the image. If you select **Nearest neighbor**, the block uses the value of one nearby pixel for the new pixel value. If you select **Bilinear**, the new pixel value is the weighted average of the two nearest

Shear

pixel values. If you select **Bicubic**, the new pixel value is the weighted average of the four nearest pixel values.

The **Data Types** pane of the Shear dialog box appears as shown in the following figure.

Function Block Parameters: Shear

Shear

Shifts each row or column of an image by a linearly increasing or decreasing distance.

Use the Shear direction parameter to specify whether you want to shift the rows or columns horizontal vertically. Use the Row/column shear values [first last] parameter or the S port to specify a two-element. The first element represents the number of pixels by which you want to shift your first row or column, the second element represents the number of pixels by which you want to shift your last row or column.

When using the S port to specify the shear values, use the Maximum shear value parameter to specify maximum number of pixels by which you want to shift your rows or columns.

If, for the Output size after shear parameter, you select Full, the block outputs the entire sheared image. If you select Same as input image, the block outputs the top left portion of the full sheared image with dimensions same as input image.

Main | **Data Types**

Settings on this pane only apply when block inputs are fixed-point signals.

Fixed-point operational parameters

Rounding mode: Overflow mode:

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Shear values	<input type="text" value="Same word length as input"/>			
Product output	<input type="text" value="Binary point scaling"/>	Yes	<input type="text" value="32"/>	<input type="text" value="10"/>
Accumulator	<input type="text" value="Same as product output"/>			
Output	<input type="text" value="Same as first input"/>			

Lock data type settings against changes by the fixed-point tools

OK Cancel Help

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Shear values

Choose how to specify the word length and the fraction length of the shear values.

- When you select **Same word length as input**, the word length of the shear values match that of the input to the block. In this mode, the fraction length of the shear values is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the shear values.
- When you select **Specify word length**, you can enter the word length of the shear values, in bits. The block automatically sets the fraction length to give you the best precision.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the shear values, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the shear values. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

This parameter is visible if, for the **Shear values source** parameter, you select **Specify via dialog**.

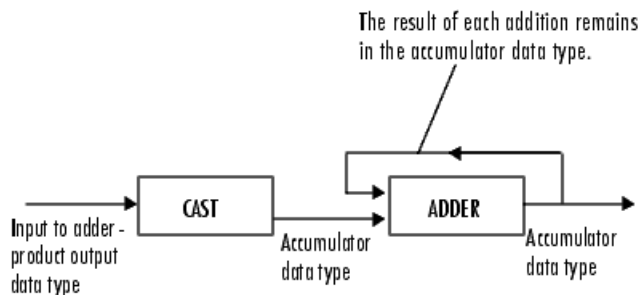
Product output



As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths.

- When you select **Same** as first input, these characteristics match those of the first input to the block at the I port.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Accumulator



As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

- When you select **Same as product output**, these characteristics match those of the product output.
- When you select **Same as first input**, these characteristics match those of the first input to the block at the I port.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Output

Choose how to specify the word length and fraction length of the output of the block:

- When you select **Same as first input**, these characteristics match those of the first input to the block at the I port.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

References

[1] Wolberg, George. *Digital Image Warping*. Washington: IEEE Computer Society Press, 1990.

See Also

Resize	Video and Image Processing Blockset software
Rotate	Video and Image Processing Blockset software
Translate	Video and Image Processing Blockset software

Standard Deviation

Purpose Find standard deviation of each input matrix

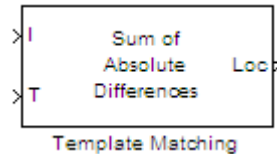
Library Statistics

Description The Standard Deviation block is a Signal Processing Blockset block. For more information, see the Standard Deviation block reference page in the Signal Processing Blockset software documentation.

Purpose Locate a template in an image

Library Analysis & Enhancement

Description



The Template Matching block finds the best match of a template within an input image. The block computes match metric values by shifting a template over a region of interest or the entire image, and then finds the best match location.

Algorithm

The match metrics use a difference equation with general form:

$$d_p(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

l_n^p denotes the metric space (R^n, d_p) for R^n $n > 1$.

- **Sum of Absolute Differences (SAD)**

This metric is also known as the *Taxicab* or *Manhattan Distance* metric. It sums the absolute values of the differences between pixels in the original image and the corresponding pixels in the template image. This metric is the l^1 norm of the difference image. The lowest SAD score estimates the best position of template within the search image. The general SAD distance metric becomes:

$$d_1(I_j, T) = \sum_{i=1}^n |I_{i,j} - T_i|$$

- **Sum of Squared Differences (SSD)**

This metric is also known as the *Euclidean Distance* metric. It sums

Template Matching

the square of the absolute differences between pixels in the original image and the corresponding pixels in the template image. This metric is the square of the l^2 norm of the difference image. The general SSD distance metric becomes:

$$d_2(I_j, T) = \sum_{i=1}^n |I_{i,j} - T_i|^2$$

- **Maximum Absolute Difference (MaxAD)**

This metric is also known as the *Uniform Distance* metric. It sums the maximum of absolute values of the differences between pixels in the original image and the corresponding pixels in the template image. This distance metric provides the l^∞ norm of the difference image. The general MaxAD distance metric becomes:

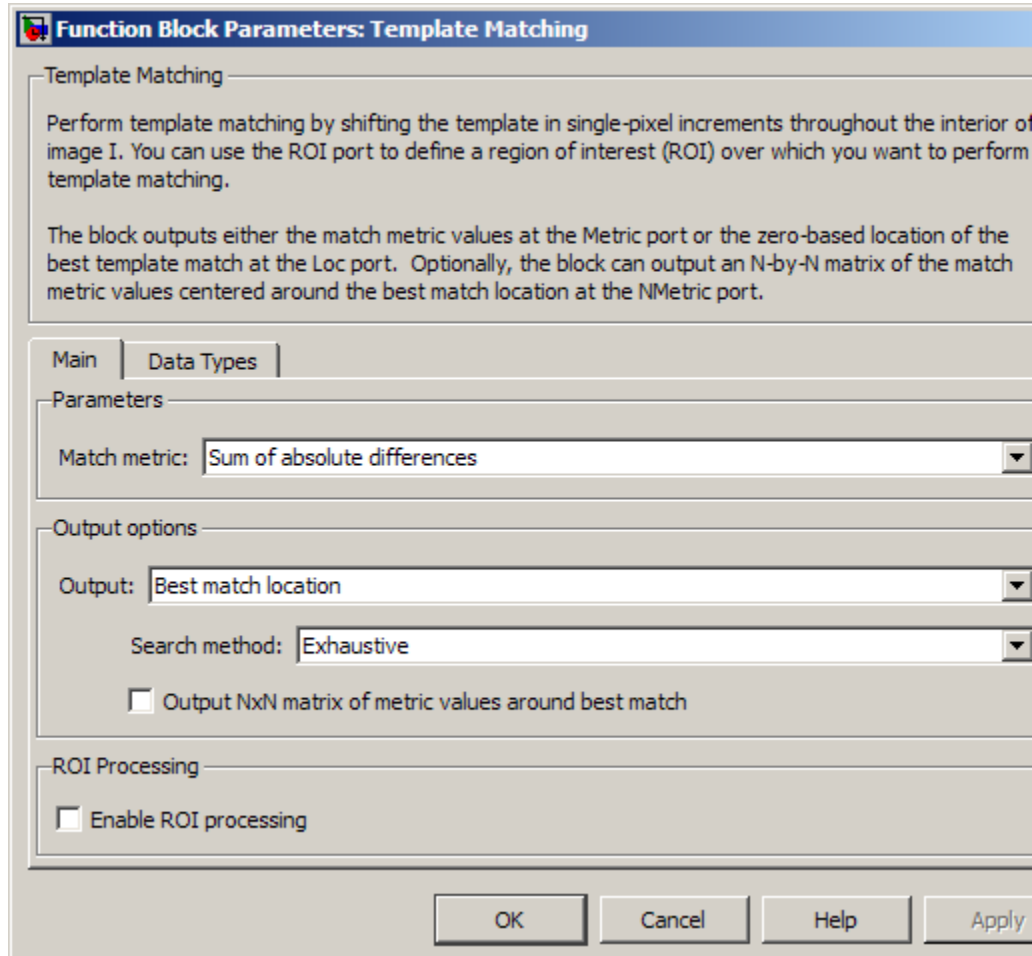
$$d_\infty(I_j, T) = \lim_{x \rightarrow \infty} \sum_{i=1}^n |I_{i,j} - T_i|^x$$

which simplifies to:

$$d_\infty(I_j, T) = \max_i^n |I_{i,j} - T_i|^x$$

Main Dialog Box

The **Main** pane of the Template Matching block appears as shown in the following figure.



Match metric

Select one of three types of match metrics:

Template Matching

- Sum of absolute differences (SAD)
- Sum of squared differences (SSD)
- Maximum absolute difference (MaxAD)

Output

Select one of two output types:

- **Metric matrix**
Select this option to output the match metric matrix. This option adds the **Metric** output port to the block.
- **Best match location**
Select this option to output the location index of the best match. This option adds the **Loc** output port to the block. When you select **Best match location**, the **Search method**, **Output NxN matrix of metric values around best match**, and **Enable ROI processing** parameter options appear.

Search method

This option appears when you select **Best match location** for the **Output** parameter. Select one of two search methods.

- Exhaustive
- Three-step

Output NxN matrix of metric values around best match

This option appears when you select **Best match location** for the **Output** parameter. Select the check box to output a matrix of metric values centered around the best match. When you do so, the block adds the **NMetric** and **NValid** output ports.

N

This option appears when you select the **Output NxN matrix of metric values around best match** check box. Enter an integer number that determines the size of the N -by- N output matrix centered around the best match location index. N must be an odd number.

Enable ROI processing

This option appears when you select **Best match location** for the **Output** parameter. Select the check box for the Template Matching block to perform region of interest processing. When you do so, the block adds the **ROI** input port and the **Output flag indicating if ROI is valid** check box appears.

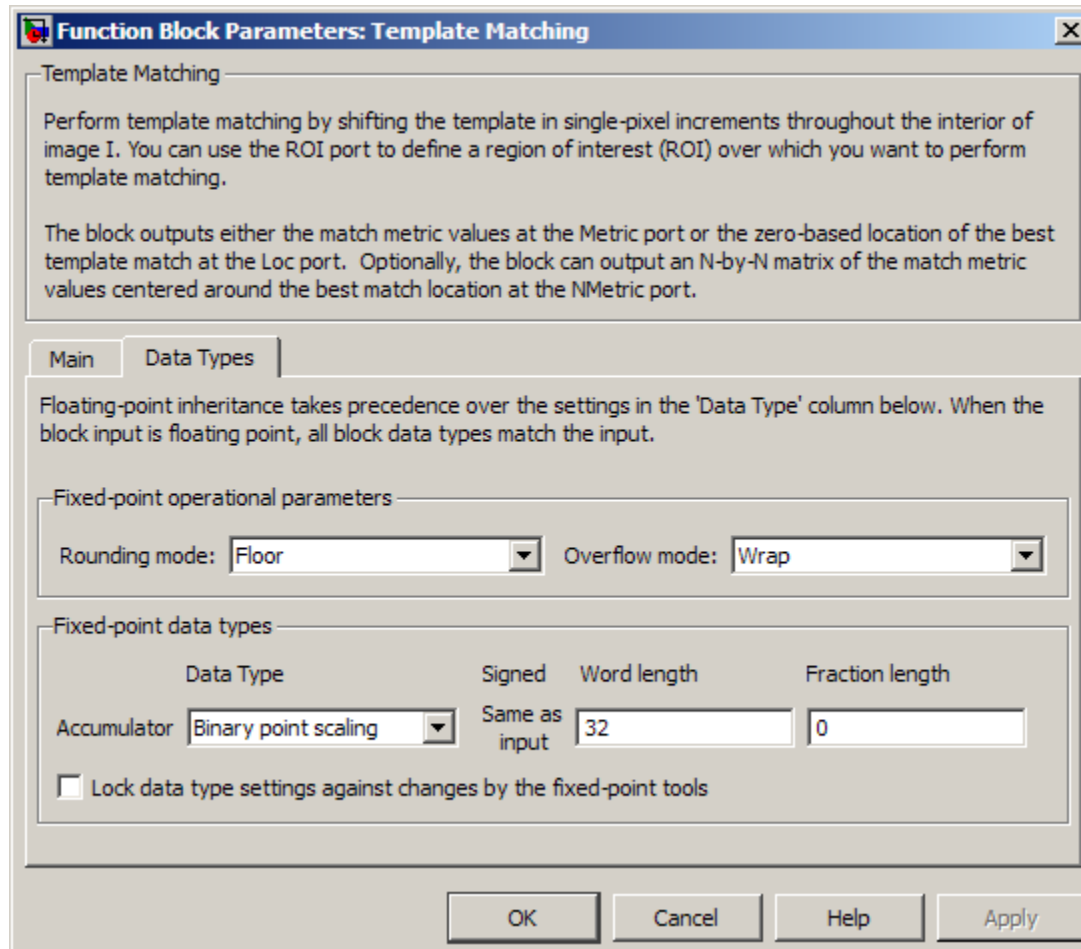
Output flag indicating if ROI is valid

This option appears when you select the **Enable ROI processing** check box. Select the check box for the Template Matching block to indicate whether the ROI is within the valid region of the image boundary. When you do so, the block adds the **ROIValid** output port.

Template Matching

Data Types Dialog Box

The **Data Types** pane of the Template Matching block dialog box appears as shown in the following figure.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

- Wrap
- Saturate

Product output

- Use this parameter to specify how to designate the product output word and fraction lengths. Refer to “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the product output data type in this block:
 - When you select `Same as input`, these characteristics match those of the input to the block.
 - When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.
 - When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to specify how you would like to designate the accumulator word and fraction lengths.

- When you select `Same as product output` the characteristics match the characteristics of the product output. See “Multiplication Data Types” for illustrations depicting the use of the accumulator data type in this block:

When you select `Binary point scaling`, you can enter the **Word length** and the **Fraction length** of the accumulator, in bits.

When you select `Slope and bias scaling`, you can enter the **Word length**, in bits, and the **Slope** of the **Accumulator**. All signals in the Video and Image Processing Blockset software have a bias of 0.

Template Matching

The block casts inputs to the **Accumulator** to the accumulator data type. It adds each element of the input to the output of the adder, which remains in the accumulator data type. Use this parameter to specify how to designate this accumulator word and fraction lengths.

Output

Choose how to specify the **Word length**, **Fraction length** and **Slope** of the **Template Matching** output:

- When you select **Same as first input**, these characteristics match the characteristics of the accumulator.
- When you select **Binary point scaling**, you can enter the **Word length** and **Fraction length** of the output, in bits.
- When you select **Slope and bias scaling**, you can enter the **Word length**, in bits, and the **Slope** of the output. All signals in the Video and Image Processing Blockset software have a bias of 0.

The **Output** parameter on the Data Types pane appears when you select **Metric matrix** or if you select **Best match location** and the **Output NxN matrix of metric values around best match** check box is selected.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

Supported Data Types

Port	Supported Data Types
I (Input Image)	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point (signed, unsigned or both)• Boolean• 8-, 16-, and 32-bit signed integers• 8-, 16-, and 32-bit unsigned integers
T (Template)	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point (signed, unsigned or both)• Boolean• 8-bit unsigned integers

Template Matching

Port	Supported Data Types
ROI (Region of Interest)	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point (signed, unsigned or both)• Boolean• 8-bit unsigned integers
Metric (Match Metric Values)	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point (signed, unsigned or both)• Boolean• 32-bit unsigned integers
Loc (Best match location [x,y])	<ul style="list-style-type: none">• 32-bit unsigned integers
NMetric (Metric values in Neighborhood of best match)	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point (signed, unsigned or both)• Boolean• 8-bit unsigned integers

Port	Supported Data Types
NValid (Neighborhood valid)	<ul style="list-style-type: none">• Boolean
ROIValid (ROI valid)	<ul style="list-style-type: none">• Boolean

Reference

[1] Koga T., et. Al. Motion-compensated interframe coding for video conferencing. In National Telecommunications Conference. Nov. 1981, G5.3.1–5, New Orleans, LA.

[2] Zakai M., “General distance criteria” *IEEE Transaction on Information Theory*, pp. 94–95, January 1964.

[3] Yu, J., J. Amores, N. Sebe, Q. Tian, "A New Study on Distance Metrics as Similarity Measurement" IEEE International Conference on Multimedia and Expo, 2006 .

See Also

“Template Matching”

Video Stabilization

Video and Image Processing Blockset Demos

To Multimedia File

Purpose Write video frames and audio samples to multimedia file

Library Sinks

Description The To Multimedia File block is a Signal Processing Blockset block. For more information, see the To Multimedia File block reference page in the Signal Processing Blockset software documentation.

Purpose Display video data

Library Sinks

Description



The To Video Display block sends video data to your computer screen.

Note This block supports code generation and is only available on Windows platforms that have file I/O available. This excludes RTWin (Real-Time Windows Target). This block performs best on platforms with DirectX Version 9.0 or later and Windows Media Version 9.0 or later.

Input	Description
Image	M-by-N matrix of intensity values or an M-by-N-by-3 color video signal
R, G, B	Matrix that represents one plane of the RGB video stream. Inputs to the R, G, or B ports must have the same dimensions and data type.

For the block to display video data properly, double- and single-precision floating-point pixel values must be from 0 to 1. For any other data type, the pixel values must be between the minimum and maximum values supported by their data type.

Select **Full-screen** to display your video stream in a full-screen window. Use the Esc key to exit or to return to other applications, hold down

To Video Display

the **Alt** key and press the **Tab** key. The display window will revert to previous size if other blocks open additional display windows.

Window size and position

Size and position of the display window is saved when the model is saved. Any changes while working with the model should be saved again in order that these preferences are maintained when the model runs.

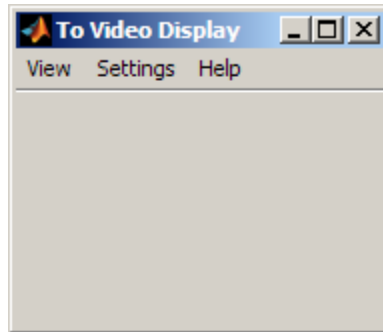
Host or external monitor visibility

When running a model that contains a To Video Display block, the output of the block might be visible on the host monitor but not the external monitor or vice versa. There are two ways to work around this problem:

- 1** Replace the To Video Display block with a Video Viewer block.
or
- 2** Disable the DirectDraw Acceleration, Direct3D Acceleration, and AGP Texture Acceleration on your system.
 - a** **Start > Run.**
 - b** For the **Open** parameter, type dxdiag. Click **OK**. The DirectX Diagnostic Tool opens.
 - c** On the **Display** tab, click the **Disable** buttons that are next to DirectDraw Acceleration, Direct3D Acceleration, and AGP Texture Acceleration.
 - d** Click **Exit**.

Menu Options

The To Video Display appears as shown in the following figure.

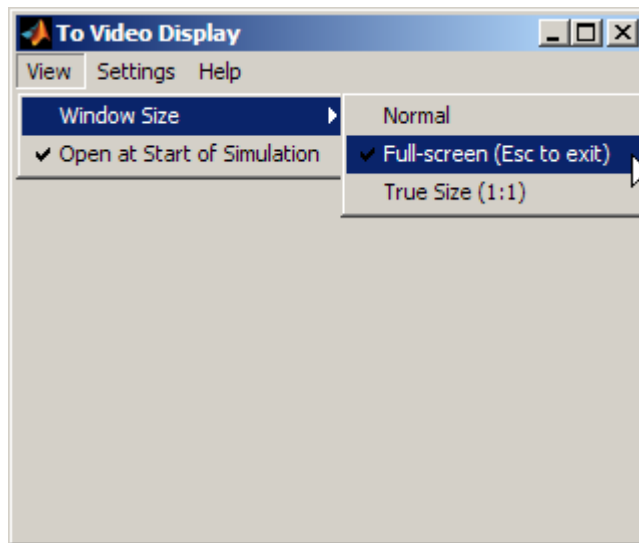


Rapid Accelerator mode

If your model is set to run in Rapid Accelerator mode, the menu options will not be available. In particular, if **Open at Start of Simulation** is unchecked, the block will not be included during the run, and therefore the video display will not be visible. For Rapid Accelerator mode, menu preferences are saved only when the model is compiled. To change any of the menu options, change the model to run in Normal mode, and re-save it. You can then run in Rapid Accelerator mode with the new preferences.

To Video Display

Full-screen



Select Full-screen mode to display your video stream in a full screen window.

Open at Start of Simulation

Select **Open at Start of Simulation** from the **View** menu for the display window to appear while running the model. If this is not selected, double click the block to display the window.

Image signal

Select the **Image signal** from the **Settings** menu to specify how the block accepts a color video signal. If you select **One multidimensional signal**, the block accepts an M-by-N-by-3 color video signal at one port. If you select **Separate color signals**, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

Preferences

Select **Preferences** from the **Settings** menu to view or modify **Video and Image Processing Blockset Preferences**

Help

The Help menu provides direct links to the To Video Display reference help, Video and Image Processing Blockset reference help, and version information.

Supported Data Types

Port	Supported Data Types
Image	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Boolean• 8-, 16, and 32-bit signed integer• 8-, 16, and 32-bit unsigned integer
R, G, B	Same as Image port

See Also

Frame Rate Display	Video and Image Processing Blockset software
From Multimedia File	Video and Image Processing Blockset software
To Multimedia File	Signal Processing Blockset software
Video To Workspace	Video and Image Processing Blockset software
Video Viewer	Video and Image Processing Blockset software

Top-hat

Purpose Perform top-hat filtering on intensity or binary images

Library Morphological Operations

Description The Top-hat block performs top-hat filtering on an intensity or binary image using a predefined neighborhood or structuring element. Top-hat filtering is the equivalent of subtracting the result of performing a morphological opening operation on the input image from the input image itself. This block uses flat structuring elements only.



Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• Boolean• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer	No
Nhood	Matrix or vector of 1s and 0s that represents the neighborhood values	Boolean	No
Output	Scalar, vector, or matrix that represents the filtered image	Same as I port	No

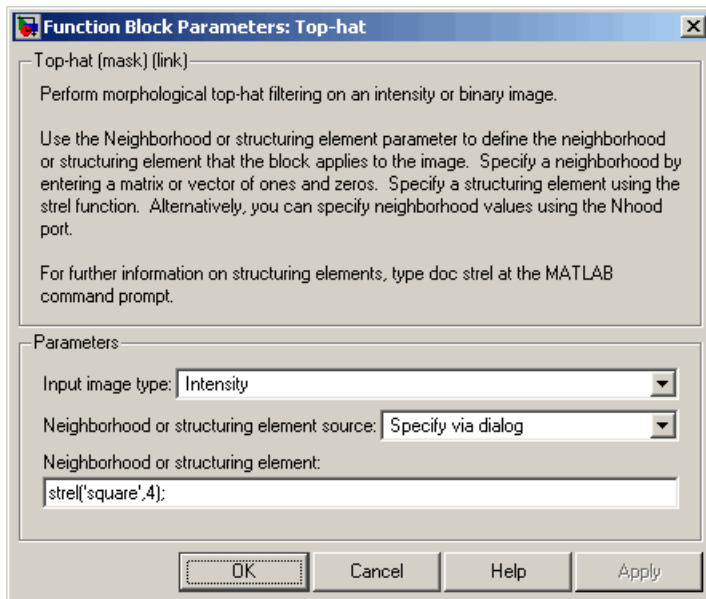
If your input image is a binary image, for the **Input image type** parameter, select **Binary**. If your input image is an intensity image, select **Intensity**.

Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select **Specify via dialog**, the **Neighborhood or structuring element** parameter appears in the dialog box. If you select **Input port**, the **Nhood** port appears on the block. Use this port to enter your neighborhood values as a matrix or vector of 1s and 0s. Choose your structuring element so that it matches the shapes you want to remove from your image. You can only specify a it using the dialog box.

Use the **Neighborhood or structuring element** parameter to define the region the block moves throughout the image. Specify a neighborhood by entering a matrix or vector of 1s and 0s. Specify a structuring element with the `strel` function from the Image Processing Toolbox. If the structuring element is decomposable into smaller elements, the block executes at higher speeds due to the use of a more efficient algorithm.

Dialog Box

The Top-hat dialog box appears as shown in the following figure.



Input image type

If your input image is a binary image, select **Binary**. If your input image is an intensity image, select **Intensity**.

Neighborhood or structuring element source

Specify how to enter your neighborhood or structuring element values. Select **Specify via dialog** to enter the values in the dialog box. Select **Input port** to use the Nhood port to specify the neighborhood values. You can only specify a structuring element using the dialog box.

Neighborhood or structuring element

If you are specifying a neighborhood, this parameter must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the `strel` function from the Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or**

structuring element source parameter, you select Specify via dialog.

See Also

Bottom-hat	Video and Image Processing Blockset software
Closing	Video and Image Processing Blockset software
Dilation	Video and Image Processing Blockset software
Erosion	Video and Image Processing Blockset software
Label	Video and Image Processing Blockset software
Opening	Video and Image Processing Blockset software
imtophat	Image Processing Toolbox software
strel	Image Processing Toolbox software

Trace Boundaries

Purpose Trace object boundaries in binary images

Library Analysis & Enhancement

Description



The Trace Boundaries block traces object boundaries in binary images, where nonzero pixels represent objects and 0 pixels represent the background.

Port	Input/Output	Supported Data Types	Complex Values Supported
BW	Vector or matrix that represents a binary image	Boolean	No
Start Pts	2-by-N matrix where each column represents the zero-based row and column coordinates of the boundary starting point, and N is the total number of starting points: $\begin{bmatrix} r_1 & r_2 & \cdots & r_N \\ c_1 & c_2 & \cdots & c_N \end{bmatrix}$	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No

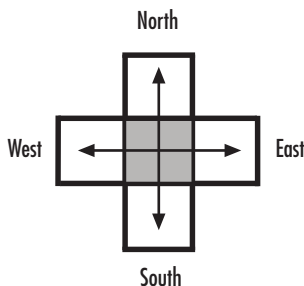
Port	Input/Output	Supported Data Types	Complex Values Supported
Pts	<p>2M-by-N matrix where each column contains the zero-based row and column coordinates of the boundary pixels, M is the maximum number of boundary pixels, and N is the total number of starting points:</p> $\begin{bmatrix} r_{11} & \cdots & r_{N1} \\ c_{11} & \cdots & c_{N1} \\ r_{12} & \cdots & r_{N2} \\ c_{12} & \cdots & c_{N2} \\ \vdots & \ddots & \vdots \\ r_{1M} & \cdots & r_{NM} \\ c_{1M} & \cdots & c_{NM} \end{bmatrix}$	Same as Start Pts port	No
Count	<p>1-by-N vector where each element represents the actual number of boundary pixels found for the corresponding starting point, where N is the number of starting points.</p>	32-bit unsigned integer	No

Use the **Connectivity** parameter to define which pixels are connected to each other. If you want a pixel to be connected to the other pixels located on the top, bottom, left, and right, select 4. If you want a pixel to be connected to the other pixels on the top, bottom, left, right, and

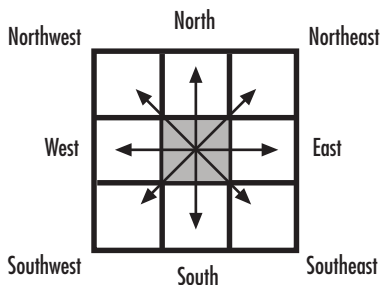
Trace Boundaries

diagonally, select **8**. For more information about this parameter, see the Label block reference page.

Use the **Initial search direction** parameter to specify the first direction in which to look to find the next boundary pixel that is connected to the starting pixel. If, for the **Connectivity** parameter, you select **4**, the following figure illustrates the four possible initial search directions:



If, for the **Connectivity** parameter, you select **8**, the following figure illustrates the eight possible initial search directions:



Use the **Trace direction** parameter to specify the direction in which to trace the boundary. Your choices are **Clockwise** or **Counterclockwise**.

Use the **Maximum number of boundary pixels** parameter to specify the maximum number of boundary pixels for each starting point. The block uses this value to preallocate the number of rows of the Pts port output matrix so that it can hold all the boundary pixel location values.

To output the actual number of boundary pixels for each starting point, select the **Output number of boundary pixels found** check box. The Count port appears on the block. The block outputs a 1-by-N vector at this port where each element represents the actual number of boundary pixels found for each starting point. Here, N is the number of starting points.

Because you specify the number of rows of the Pts port output matrix using the **Maximum number of boundary pixels** parameter, use the **Action to take for empty output points** parameter to specify what happens to the empty elements in this vector when the number of boundary pixels is less than the maximum.

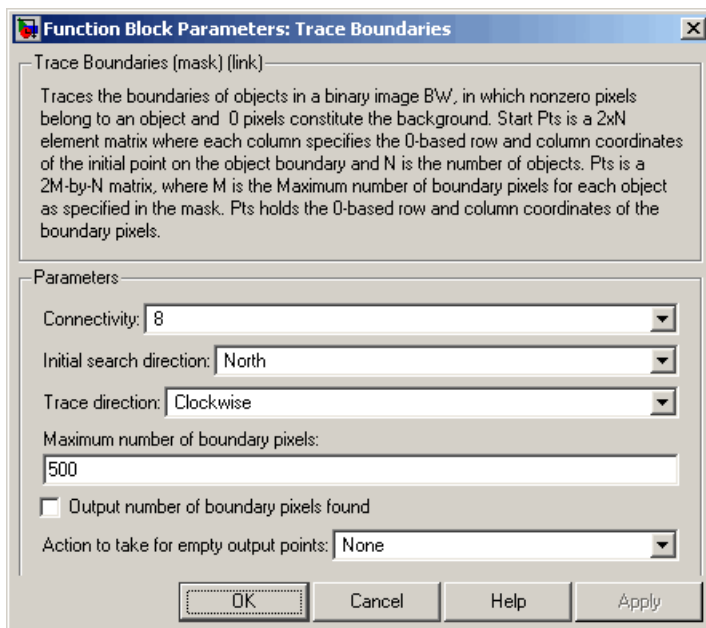
- If you select **None**, the block takes no action. So, any element that does not contain a boundary pixel location will not have a meaningful value.
- If you select **Fill with last point found**, the block fills the remaining elements with the position of the last boundary pixel.
- If you select **Fill with user-defined values**, the **Fill values** parameter appears on the block.

For the **Fill values** parameter, enter a scalar value or two-element vector that you want the block to use to fill in the empty elements.

Trace Boundaries

Dialog Box

The Trace Boundaries dialog box appears as shown in the following figure.



Connectivity

Specify which pixels are connected to each other. If you want a pixel to be connected to the pixels on the top, bottom, left, and right, select 4. If you want a pixel to be connected to the pixels on the top, bottom, left, right, and diagonally, select 8.

Initial search direction

Specify the first direction in which to look to find the next boundary pixel that is connected to the starting pixel.

Trace direction

Specify the direction in which to trace the boundary. Your choices are Clockwise or Counterclockwise.

Maximum number of boundary pixels

Specify the maximum number of boundary pixels. The block uses this value to preallocate the number of rows of the Pts port output matrix so that it can hold all the boundary pixel location values.

Output number of boundary pixels found

If you select this check box, the block outputs a vector at the Count port where each element represents the actual number of boundary pixels found for each starting point.

Action to take for empty output points

Specify how to fill the empty spaces in the Pts port output matrix. If you select **None**, the block takes no action. So, any element that does not contain a boundary pixel location will not have a meaningful value. If you select **Fill with last point found**, the block fills the remaining elements with the position of the last boundary pixel. If you select **Fill with user-defined values**, the **Fill values** parameter appears on the block.

Fill values

Enter a scalar value or two-element vector that you want the block to use to fill in the remaining empty elements. This parameter is visible if, for the **Action to take for empty output points** parameter, you select **Fill with user-defined values**.

See Also

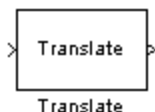
Edge Detection	Video and Image Processing Blockset software
Label	Video and Image Processing Blockset software
bwboundaries	Image Processing Toolbox software
bwtraceboundary	Image Processing Toolbox software

Translate

Purpose Translate image in 2-D plane using displacement vector

Library Geometric Transformations
vipgeotforms

Description Use the Translate block to move an image in a two-dimensional plane using a displacement vector, a two-element vector that represents the number of pixels by which you want to translate your image. The block outputs the image produced as the result of the translation.



Note This block supports intensity and color images on its ports.

Port	Input/Output	Supported Data Types	Complex Values Supported
Image / Input	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer	No
Offset	Vector of values that represent the number of pixels by which to translate the image	Same as I port	No
Output	Translated image	Same as I port	No

The input to the Offset port must be the same data type as the input to the Image port. The output is the same data type as the input to the Image port.

Use the **Output size after translation** parameter to specify the size of the translated image. If you select **Full**, the block outputs a matrix that contains the entire translated image. If you select **Same as input image**, the block outputs a matrix that is the same size as the input image and contains a portion of the translated image. Use the **Background fill value** parameter to specify the pixel values outside the image.

Use the **Translation values source** parameter to specify how to enter your displacement vector. If you select **Specify via dialog**, the **Offset** parameter appears in the dialog box. Use it to enter your displacement vector, a two-element vector, $[r \ c]$, of real, integer values that represent the number of pixels by which you want to translate your image. The r value represents how many pixels up or down to shift your image. The c value represents how many pixels left or right to shift your image. The axis origin is the top-left corner of your image. For example, if you enter $[2.5 \ 3.2]$, the block moves the image 2.5 pixels downward and 3.2 pixels to the right of its original location. When the displacement vector contains fractional values, the block uses interpolation to compute the output.

Use the **Interpolation method** parameter to specify which interpolation method the block uses to translate the image. If you translate your image in either the horizontal or vertical direction and you select **Nearest neighbor**, the block uses the value of the nearest pixel for the new pixel value. If you translate your image in either the horizontal or vertical direction and you select **Bilinear**, the new pixel value is the weighted average of the four nearest pixel values. If you translate your image in either the horizontal or vertical direction and you select **Bicubic**, the new pixel value is the weighted average of the sixteen nearest pixel values.

The number of pixels the block considers affects the complexity of the computation. Therefore, the nearest-neighbor interpolation is the most computationally efficient. However, because the accuracy of the method is roughly proportional to the number of pixels considered, the bicubic method is the most accurate. For more information, see

Translate

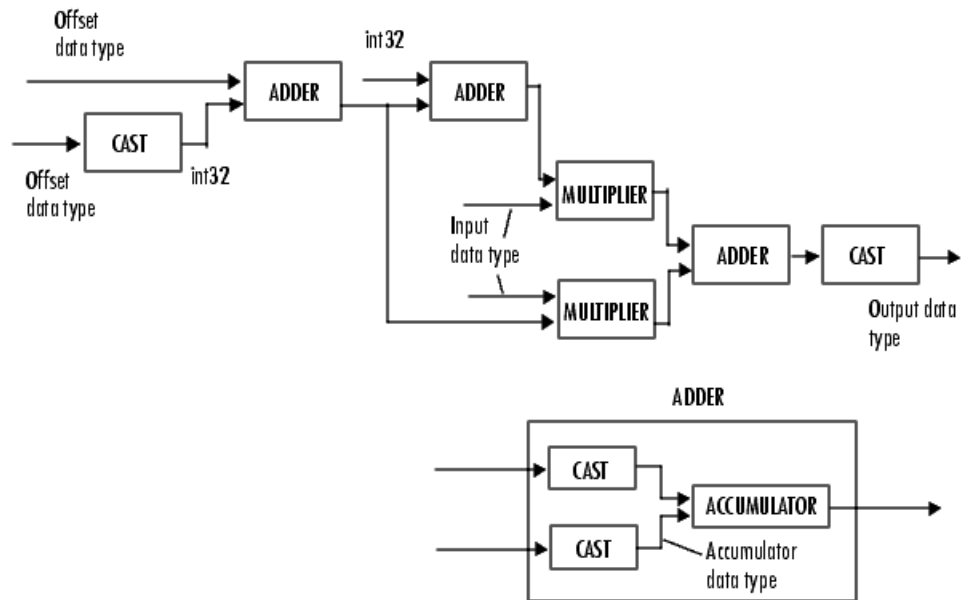
“Geometric Transformation Interpolation Methods” in the *Video and Image Processing Blockset User’s Guide*.

If, for the **Output size after translation** parameter, you select Full, and for the **Translation values source** parameter, you select Input port, the **Maximum offset** parameter appears in the dialog box. Use the **Maximum offset** parameter to enter a two-element vector of real, scalar values that represent the maximum number of pixels by which you want to translate your image. The block uses this parameter to determine the size of the output matrix. If the input to the Offset port is greater than the **Maximum offset** parameter values, the block saturates to the maximum values.

If, for the **Translation values source** parameter, you select Input port, the Offset port appears on the block. At each time step, the input to the Offset port must be a vector of real, scalar values that represent the number of pixels by which to translate your image.

Fixed-Point Data Types

The following diagram shows the data types used in the Translate block for bilinear interpolation of fixed-point signals.

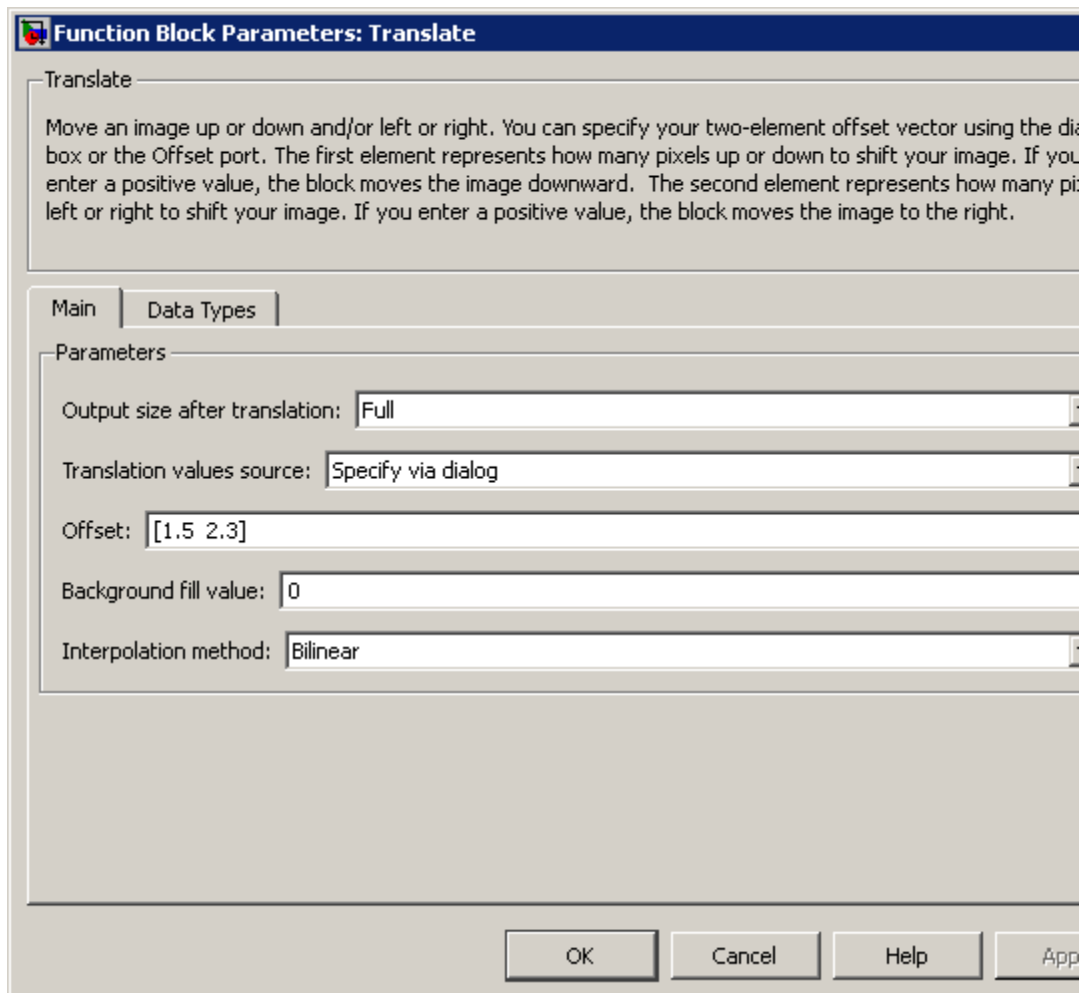


You can set the product output, accumulator, and output data types in the block mask as discussed in the next section.

Translate

Dialog Box

The **Main** pane of the Translate dialog box appears as shown in the following figure.



Output size after translation

If you select **Full**, the block outputs a matrix that contains the translated image values. If you select **Same as input image**, the block outputs a matrix that is the same size as the input image and contains a portion of the translated image.

Translation values source

Specify how to enter your translation parameters. If you select **Specify via dialog**, the **Offset** parameter appears in the dialog box. If you select **Input port**, port O appears on the block. The block uses the input to this port at each time step as your translation values.

Offset

Enter a vector of real, scalar values that represent the number of pixels by which to translate your image.

Background fill value

Specify a value for the pixels that are outside the image.

Interpolation method

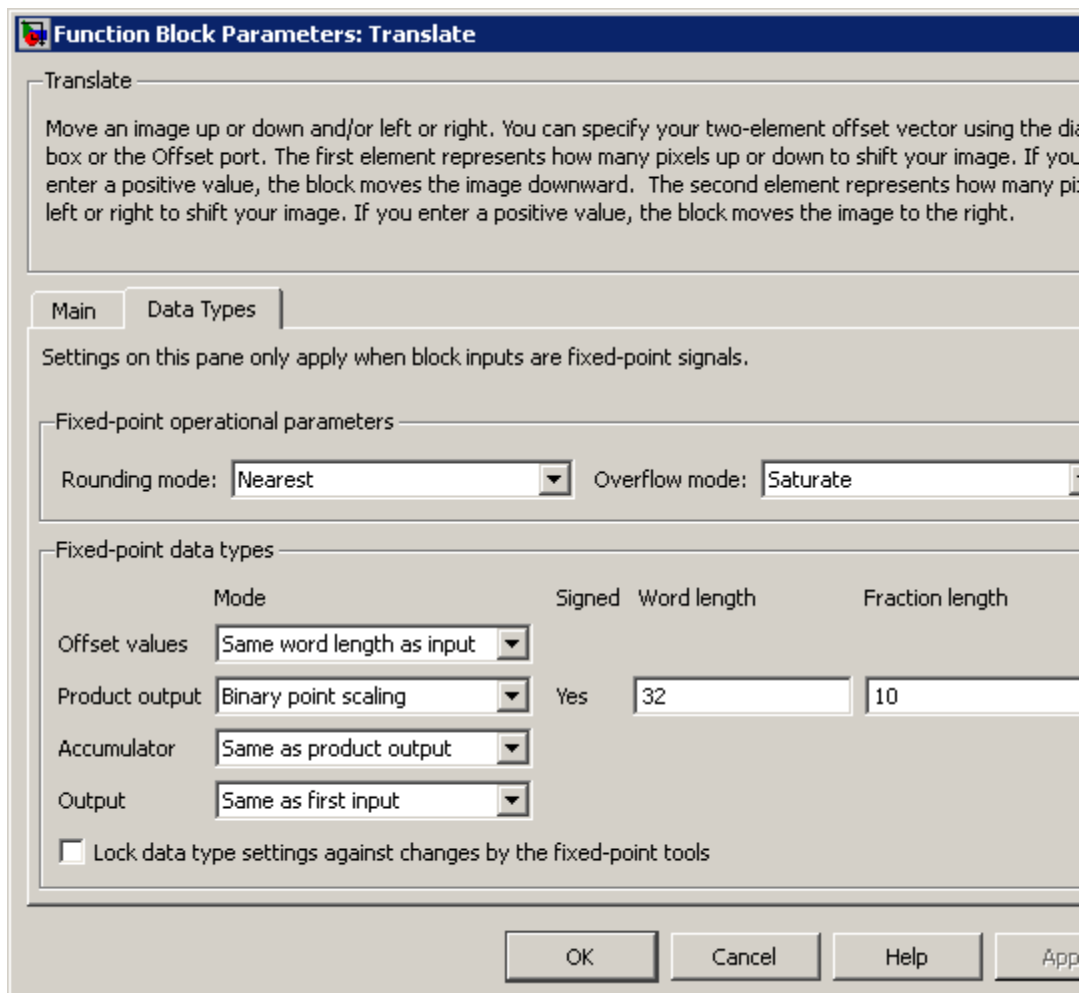
Specify which interpolation method the block uses to translate the image. If you select **Nearest neighbor**, the block uses the value of one nearby pixel for the new pixel value. If you select **Bilinear**, the new pixel value is the weighted average of the four nearest pixel values. If you select **Bicubic**, the new pixel value is the weighted average of the sixteen nearest pixel values.

Maximum offset

Enter a vector of real, scalar values that represent the maximum number of pixels by which you want to translate your image. This parameter must have the same data type as the input to the **Offset** port. This parameter is visible if, for the **Output size after translation** parameter, you select **Full** and, for the **Translation values source** parameter, you select **Input port**.

The **Data Types** pane of the Translate dialog box appears as shown in the following figure.

Translate



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Offset values

Choose how to specify the word length and the fraction length of the offset values.

- When you select **Same word length as input**, the word length of the offset values match that of the input to the block. In this mode, the fraction length of the offset values is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the offset values.
- When you select **Specify word length**, you can enter the word length of the offset values, in bits. The block automatically sets the fraction length to give you the best precision.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the offset values, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the offset values. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

This parameter is visible if, for the **Translation values source** parameter, you select **Specify** via dialog.

Product output

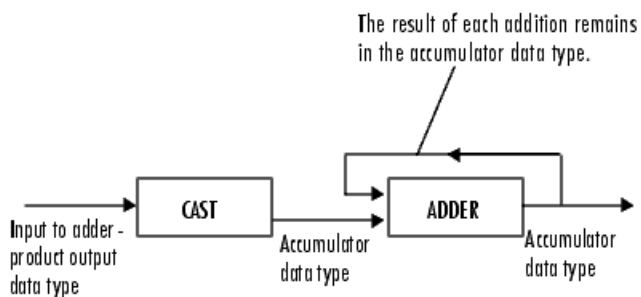


As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this

parameter to specify how to designate this product output word and fraction lengths.

- When you select **Same as first input**, these characteristics match those of the first input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the product output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Accumulator



As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

- When you select **Same as product output**, these characteristics match those of the product output.
- When you select **Same as first input**, these characteristics match those of the first input to the block.

- When you select **Binary point scaling**, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Output

Choose how to specify the word length and fraction length of the output of the block:

- When you select **Same as first input**, these characteristics match those of the first input to the block.
- When you select **Binary point scaling**, you can enter the word length and the fraction length of the output, in bits.
- When you select **Slope and bias scaling**, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Lock data type settings against change by the fixed-point tools

Select this parameter to prevent the fixed-point tools from overriding the data types you specify on the block mask. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

References

- [1] Wolberg, George. *Digital Image Warping*. Washington: IEEE Computer Society Press, 1990.

See Also

Resize	Video and Image Processing Blockset software
Rotate	Video and Image Processing Blockset software
Shear	Video and Image Processing Blockset software

Variable Selector

Purpose Specify subset of rows or columns from input

Library Utilities

Description The Variable Selector block is a Signal Processing Blockset block. For more information, see the Variable Selector block reference page in the Signal Processing Blockset software documentation.

Purpose Compute variance of input or sequence of inputs

Library Statistics

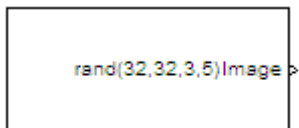
Description The Variance block is a Signal Processing Blockset block. For more information, see the Variance block reference page in the Signal Processing Blockset software documentation.

Video From Workspace

Purpose Import video signal from MATLAB workspace

Library Sources
vipsrcs

Description



Video From Workspace

The Video From Workspace block imports a video signal from the MATLAB workspace. If the video signal is a M-by-N-by-T workspace array, the block outputs an intensity video signal, where M and N are the number of rows and columns in a single video frame, and T is the number of frames in the video signal. If the video signal is a M-by-N-by-C-by-T workspace array, the block outputs a color video signal, where M and N are the number of rows and columns in a single video frame, C is the number of color channels, and T is the number of frames in the video stream. In addition to the video signals previously described, this block supports fi objects and variables that are in the structure format returned by the MATLAB `mmreader` function.

Note If you generate code from a model that contains this block, Real-Time Workshop takes a long time to compile the code because it puts all of the video data into the `.c` file. Before you generate code, you should convert your video data to a format supported by the From Multimedia File block or the Read Binary File block.

Port	Output	Supported Data Types	Complex Values Supported
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	No
R, G, B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B ports have the same dimensions.	Same as I port	No

For the Video and Image Processing Blockset blocks to display video data properly, double- and single-precision floating-point pixel values must be from 0 to 1. This block does not scale pixel values.

Use the **Signal** parameter to specify the MATLAB workspace variable from which to read. For example, to read an AVI file, use the following syntax:

```
mov = mmreader('filename.avi')
```

The `mmreader` function reads the AVI file into the MATLAB movie structure `mov`. The MATLAB movie structure might be obsoleted in the future. For more information, see `mmreader` class in the MATLAB documentation.

If `filename.avi` has a colormap associated with it, the AVI file must satisfy the following conditions or the block produces an error:

Video From Workspace

- The colormap must be empty or have 256 values.
- The data must represent an intensity image.
- The data type of the image values must be `uint8`.

Use the **Sample time** parameter to set the sample period of the output signal.

When the block has output all of the available signal samples, it can start again at the beginning of the signal, repeat the final value, or generate 0s until the end of the simulation. The **Form output after final value by** parameter controls this behavior:

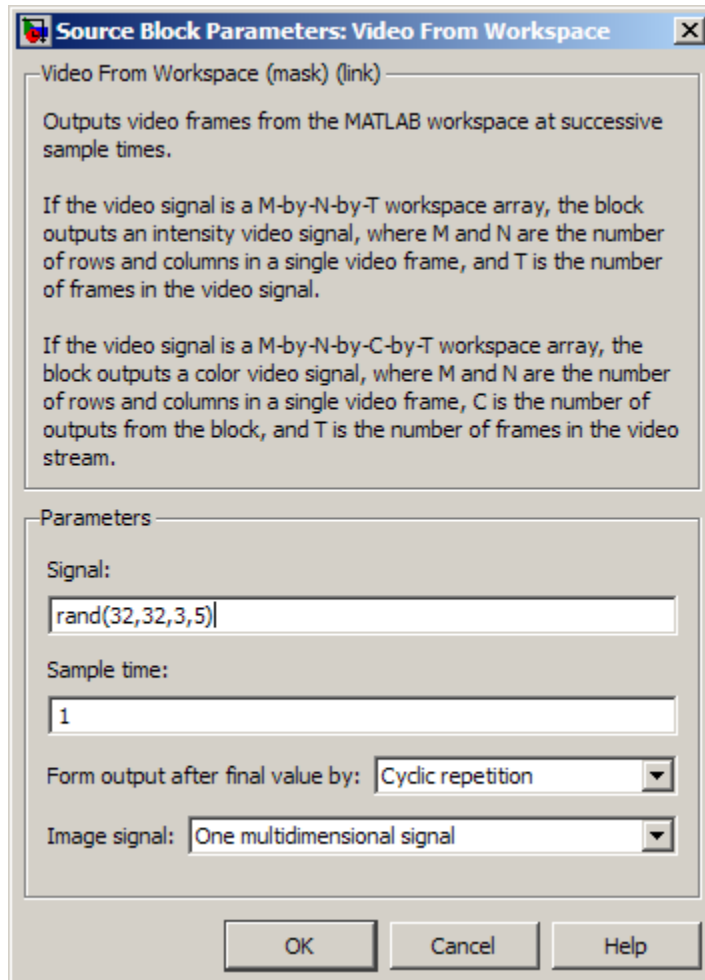
- When you specify `Setting To Zero`, the block generates zero-valued outputs for the duration of the simulation after generating the last frame of the signal.
- When you specify `Holding Final Value`, the block repeats the final frame for the duration of the simulation after generating the last frame of the signal.
- When you specify `Cyclic Repetition`, the block repeats the signal from the beginning after it reaches the last frame in the signal.

Use the **Image signal** parameter to specify how the block outputs a color video signal. If you select `One multidimensional signal`, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select `Separate color signals`, additional ports appear on the block. Each port outputs one M-by-N plane of an RGB video stream.

Use the **Output port labels** parameter to label your output ports. Use the spacer character, `|`, as the delimiter. This parameter is available when the **Image signal** parameter is set to `Separate color signals`.

Dialog Box

The Video From Workspace dialog box appears as shown in the following figure.



Video From Workspace

Signal

Specify the MATLAB workspace variable that contains the video signal, or use the `mmreader` function to specify an AVI filename.

Sample time

Enter the sample period of the output.

Form output after final value by

Specify the output of the block after all of the specified signal samples have been generated. The block can output zeros for the duration of the simulation (`Setting to zero`), repeat the final value (`Holding Final Value`) or repeat the entire signal from the beginning (`Cyclic Repetition`).

Image signal

Specify how the block outputs a color video signal. If you select `One multidimensional signal`, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select `Separate color signals`, additional ports appear on the block. Each port outputs one M-by-N plane of an RGB video stream.

Output port labels

Enter the labels for your output ports using the spacer character, `|`, as the delimiter. This parameter is available when the **Image signal** parameter is set to `Separate color signals`.

See Also

From Multimedia File	Video and Image Processing Blockset software
Image From Workspace	Video and Image Processing Blockset software
Read Binary File To Video Display	Video and Image Processing Blockset software
Video Viewer	Video and Image Processing Blockset software

Purpose Export video signal to MATLAB workspace

Library Sinks

Description



The Video To Workspace block exports a video signal to the MATLAB workspace. If the video signal is represented by intensity values, it appears in the workspace as a three-dimensional M-by-N-by-T array, where M and N are the number of rows and columns in a single video frame, and T is the number of frames in the video signal. If it is a color video signal, it appears in the workspace as a four-dimensional M-by-N-by-C-by-T array, where M and N are the number of rows and columns in a single video frame, C is the number of inputs to the block, and T is the number of frames in the video stream. During code generation, Real-Time Workshop does not generate code for this block.

Note This block supports intensity and color images on its ports.

Port	Input	Supported Data Types	Complex Values Supported
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer 	No
R, G, B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Outputs	Same as I port	No

Video To Workspace

Port	Input	Supported Data Types	Complex Values Supported
	from the R, G, or B ports have the same dimensions.		

Use the **Variable name** parameter to specify the MATLAB workspace variable to which to write the video signal.

Use the **Number of inputs** parameter to determine the number of inputs to the block. If the video signal is represented by intensity values, enter 1. If it is a color (R, G, B) video signal, enter 3.

Use the **Limit data points to last** parameter to determine the number of video frames, T, you want to export to the MATLAB workspace.

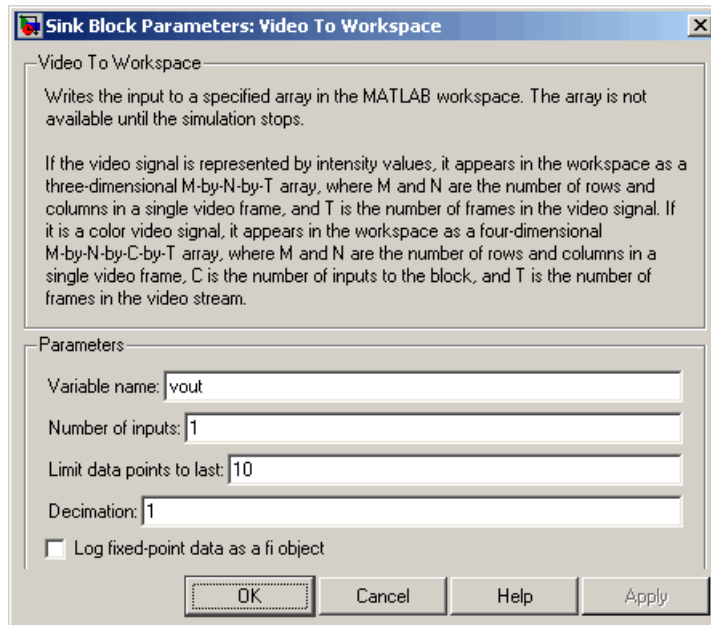
If you want to downsample your video signal, use the **Decimation** parameter to enter your decimation factor.

If your video signal is fixed point and you select the **Log fixed-point data as a fi object** check box, the block creates a fi object in the MATLAB workspace.

Use the **Input port labels** parameter to label your input ports. Use the spacer character, |, as the delimiter. This parameter is available if the **Number of inputs** parameter is greater than 1.

Dialog Box

The Video To Workspace dialog box appears as shown in the following figure.



Variable name

Specify the MATLAB workspace variable to which to write the video signal.

Number of inputs

Enter the number of inputs to the block. If the video signal is black and white, enter 1. If it is a color (R, G, B) video signal, enter 3.

Limit data points to last

Enter the number of video frames to export to the MATLAB workspace.

Decimation

Enter your decimation factor.

Video To Workspace

Log fixed-point data as a fi object

If your video signal is fixed point and you select this check box, the block creates a fi object in the MATLAB workspace. For more information of fi objects, see the Fixed-Point Toolbox documentation.

Input port labels

Enter the labels for your input ports using the spacer character, |, as the delimiter. This parameter is available if the **Number of inputs** parameter is greater than 1.

See Also

Signal To Workspace	Signal Processing Blockset software
To Multimedia File	Signal Processing Blockset software
To Video Display	Video and Image Processing Blockset software
Video Viewer	Video and Image Processing Blockset software

Purpose

Display binary, intensity, or RGB images or video streams

Library

Sinks

Description



The Video Viewer block enables you to view a binary, intensity, or RGB image or a video stream. The block provides simulation controls for play, pause, and step while running the model. The block also provides pixel region analysis tools. During code generation, Real-Time Workshop software does not generate code for this block.

Note The To Video Display block supports code generation.

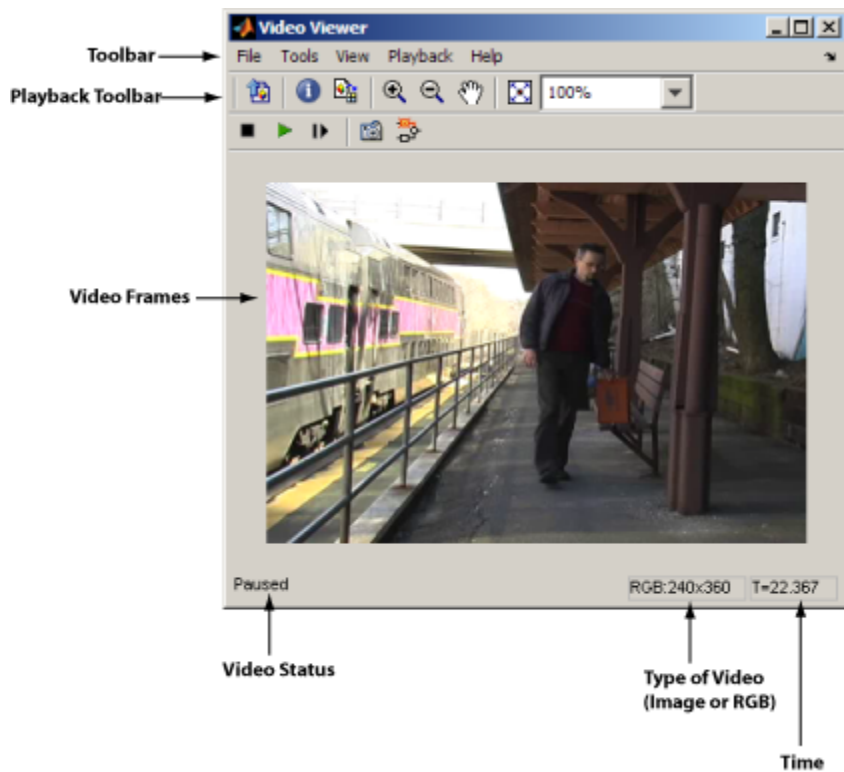
See the following table for descriptions of both input types.

Input	Description
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes.
R/G/B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Inputs to the R, G, or B ports must have the same dimensions and data type.

Select **File > Image Signal** to set the input to either **Image** or **RGB**.


Video Viewer

Dialogs








Toolbar

Toolbar




GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
	File > Export to Image Tool	Ctrl+E	Send the current video frame to the Image Tool. For more information, see "Using the Image Tool to Explore Images"

Toolbar (Continued)

GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
			in the Image Processing Toolbox documentation.
<p>Note The Image Tool can only know that the frame is an intensity image if the colormap of the frame is grayscale (<code>gray(256)</code>). Otherwise, the Image Tool assumes the frame is an indexed image and disables the Adjust Contrast button.</p>			
	Tools > Video Information	V	View information about the video data source.
	Tools > Pixel Region	N/A	Open the Pixel Region tool. For more information about this tool, see the Image Processing Toolbox documentation.
	Tools > Zoom In	N/A	Zoom in on the video display.
	Tools > Zoom Out	N/A	Zoom out of the video display.
	Tools > Pan	N/A	Move the image displayed in the GUI.





Video Viewer

Toolbar (Continued)



GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
	Tools > Maintain Fit to Window	N/A	Scale video to fit GUI size automatically. Toggle the button on or off.
	N/A 	N/A	Enlarge or shrink the video frame. This option becomes available if you do not select the Maintain Fit to Window .

Playback Toolbar

Playback Toolbar

GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
	Playback > Stop	S	Stop the video.
	Playback > Play	P, Space bar	Play the video.
	Playback > Pause	P, Space bar	Pause the video. This button appears only when the video is playing.
	Playback > Step Forward	Right arrow, Page Down	Step forward one frame.

Playback Toolbar (Continued)

GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
	Playback > Simulink Snapshot	N/A	Click this button to freeze the display in the viewer window.
	Playback > Highlight Simulink Signal	Ctrl+L	In the model window, highlight the Simulink signal the viewer is displaying.

Setting Viewer Configuration

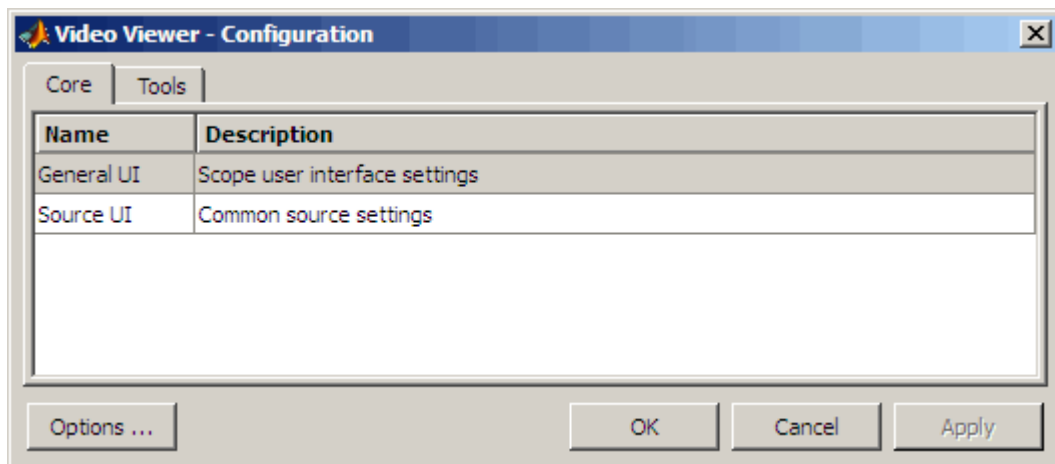
The Video Viewer Configuration preferences enables you to change the behavior and appearance of the graphic user interface (GUI) as well as the behavior of the playback shortcut keys.

- To open the Configuration dialog box, select **File > Configuration Set > Edit**.
- To save the configuration settings for future use, select **File > Configuration Set > Save as**.

Core Pane

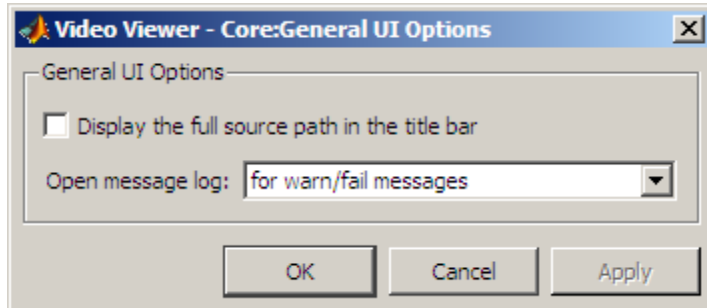
The Core pane in the Viewer Configuration dialog box controls the GUI's general settings.

Video Viewer



General UI

Click **General UI**, and click the **Options** button to open the General UI Options dialog box.



If you select the **Display the full source path in the title bar** check box, the GUI displays the model name and full Simulink path to the video data source in the title bar. Otherwise, it displays a shortened name.

Use the **Open message log:** parameter to control when the Message log window opens. You can use this window to debug issues with video

playback. Your choices are for any new messages, for warn/fail messages, only for fail messages, or manually.

Tools Pane

The Tools pane in the Viewer Configuration dialog box contains the tools that appear on the Video Viewer GUI. Select the **Enabled** check box next to the tool name to specify which tools to include on the GUI.

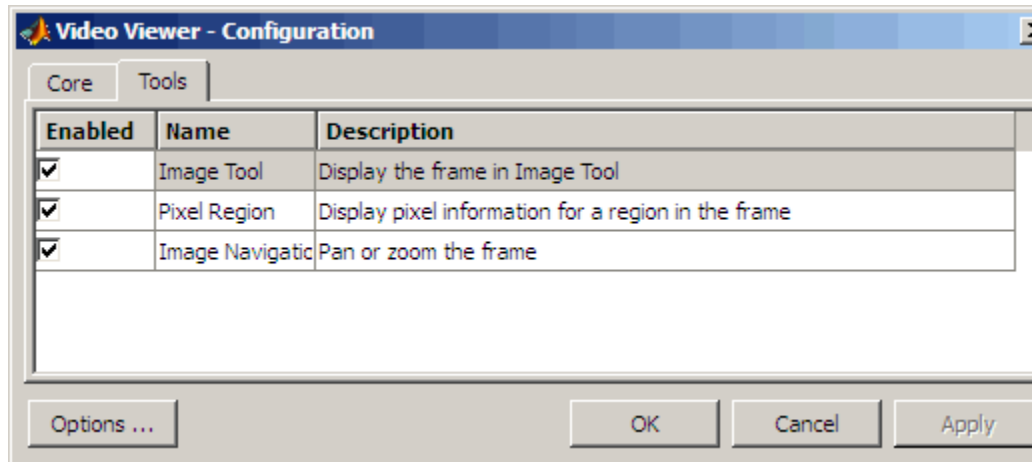
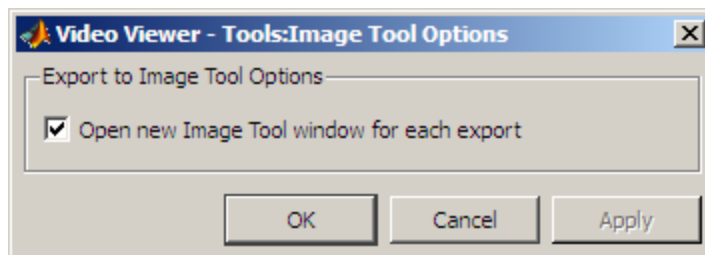


Image Tool

Click **Image Tool**, and then click the **Options** button to open the Image Tool Options dialog box.



Video Viewer

Select the **Open new Image Tool window for export** check box if you want to send each video frame to a different session of Image Tool.

Pixel Region

Select the **Pixel Region** check box to display and enable the pixel region GUI button. For more information on working with pixel regions see Getting Information about the Pixels in an Image.

Image Navigation Tools


Select the **Image Navigation Tools** check box to enable the pan-and-zoom GUI button.

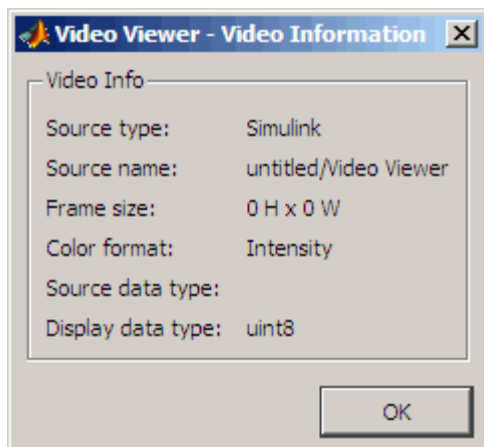
Instrumentation Set

Select the **Instrumentation Set** check box to enable the option to load and save viewer settings. The option appears in the **File** menu.

Video Information

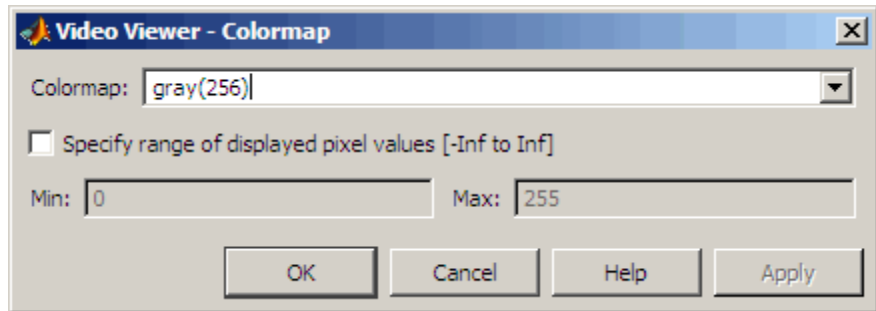
The Video Information dialog box lets you view basic information about the video. To open this dialog box, you can select **Tools > Video**

Information , click the information button  , or press the **V** key.



Colormap for Intensity Video

The Colormap dialog box lets you change the colormap of an intensity video. You cannot access the parameters on this dialog box when the GUI displays an RGB video signal. To open this dialog box for an intensity signal, select **Tools > Colormap** or press **C**.



Use the **Colormap** parameter to specify the colormap to apply to the intensity video.

If you know that the pixel values do not use the entire data type range, you can select the **Specify range of displayed pixel values** check box and enter the range for your data. The dialog box automatically displays the range based on the data type of the pixel values.

Status Bar

A status bar appear along the bottom of the Video Viewer. It displays information pertaining to the video status (running, paused or ready), type of video (Intensity or RGB) and video time.

Message Log

The Message Log dialog provides a system level record of configurations and extensions used. You can filter what messages to display by **Type** and **Category**, view the records, and display record details.

The **Type** parameter allows you to select either All, Info, Warn, or Fail message logs. The **Category** parameter allows you to select either Configuration or Extension message summaries. The Configuration

Video Viewer

messages indicate when a new configuration file is loaded. The Extension messages indicate a component is registered. For example, you might see a Simulink message, which indicates the component is registered and available for configuration.

Saving the Settings of Multiple Video Viewer GUIs

The Video Viewer GUI enables you to save and load the settings of multiple GUI instances. Thus, you only need to configure the Video Viewer GUIs that are associated with your model once. To save the GUI settings, select **File > Instrumentation Sets > Save Set**. To open the preconfigured GUIs, select **File > Instrumentation Sets > Load Set**.

Supported Data Types

Port	Supported Data Types
Image	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Boolean• 8-, 16-, and 32-bit signed integer• 8-, 16-, and 32-bit unsigned integer
R/G/B	Same as Image port

See Also

From Multimedia File	Video and Image Processing Blockset software
mplay	Video and Image Processing Blockset software
To Multimedia File	Signal Processing Blockset software

To Video Display

Video and Image Processing Blockset
software

Video To Workspace

Video and Image Processing Blockset
software

implay

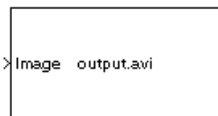
Image Processing Toolbox

Write AVI File (Obsolete)

Purpose Write video frames to uncompressed AVI file

Library Sinks

Description



Write AVI File

Note The Write AVI File block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset blocks. Use the replacement block To Multimedia File.

The Write AVI File block writes video frames to an uncompressed AVI file from a Simulink model. If the data type of the input pixel values is anything other than 8-bit unsigned integers, the block scales the values. Then, it writes values between the minimum and maximum values supported by the 8-bit unsigned integer data type to the AVI file. This block does not support audio samples. During code generation, Real-Time Workshop does not generate code for this block.

Port	Input	Supported Data Types	Complex Values Supported
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • Boolean • 8-, 16- 32-bit signed integer • 8-, 16- 32-bit unsigned integer 	No
R, G, B	Matrix that represents one plane of the RGB video stream. Inputs to the R, G, or B ports must have the same dimensions.	Same as I port	No

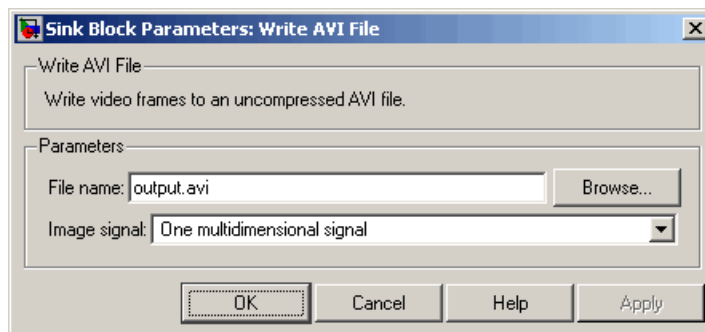
Write AVI File (Obsolete)

Use the **File name** parameter to specify the name of the AVI file to which to write. The block creates the AVI file in your current directory. To specify a different directory, use the **Browse** button; then enter the filename.

Use the **Image signal** parameter to specify how the block accepts a color video signal. If you select **One multidimensional signal**, the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

Dialog Box

The Write AVI File dialog box appears as shown in the following figure.



File name

Specify the name of the AVI file to which to write.

Image signal

Specify how the block accepts a color video signal. If you select **One multidimensional signal**, the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

Write AVI File (Obsolete)

See Also

To Multimedia File	Signal Processing Blockset software
To Video Display	Video and Image Processing Blockset software
Video To Workspace	Video and Image Processing Blockset software
Video Viewer	Video and Image Processing Blockset software

Purpose Write binary video data to files

Library Sinks

Description The Write Binary File block takes video data from a Simulink model and exports it to a binary file.



This block produces a raw binary file with no header information. It has no encoded information providing the data type, frame rate or dimensionality. The video data for this block appears in row major format.

Note This block supports code generation only for platforms that have file I/O available. You cannot use this block to do code generation with RTWin (Real-Time Windows Target).

Port	Input	Supported Data Types	Complex Values Supported
Input	Matrix that represents the luma (Y') and chroma (Cb and Cr) components of a video stream	<ul style="list-style-type: none"> 8-, 16- 32-bit signed integer 8-, 16- 32-bit unsigned integer 	No

Four Character Code Video Formats

Four Character Codes (FOURCC) identify video formats. For more information about these codes, see <http://www.fourcc.org>.

Use the **Four character code** parameter to identify the video format.

Custom Video Formats

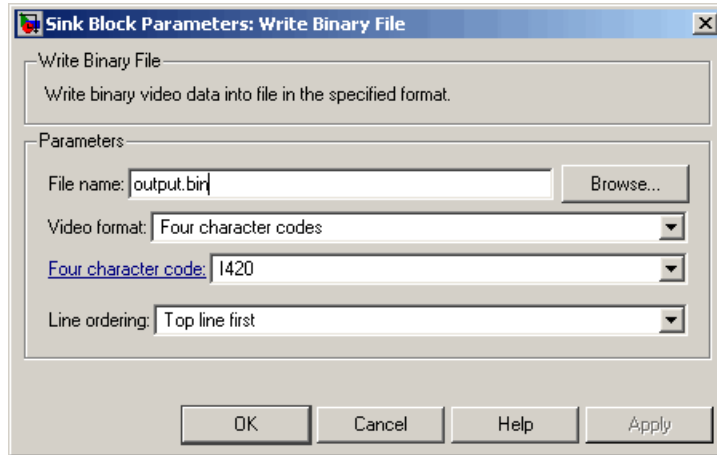
You can use the Write Binary File block to create a binary file that contains video data in a custom format.

Write Binary File

- Use the **Bit stream format** parameter to specify whether you want your data in planar or packed format.
- Use the **Number of input components** parameter to specify the number of components in the video stream. This number corresponds to the number of block input ports.
- Select the **Inherit size of components from input data type** check box if you want each component to have the same number of bits as the input data type. If you clear this check box, you must specify the number of bits for each component.
- Use the **Component** parameters to specify the component names.
- Use the **Component order in binary file** parameter to specify how to arrange the components in the binary file.
- Select the **Interlaced video** check box if the video stream represents interlaced video data.
- Select the **Write signed data to output file** check box if your input data is signed.
- Use the **Byte order in binary file** parameter to specify whether the byte ordering in the output binary file is little endian or big endian.

Dialog Box

The Write Binary File dialog box appears as shown in the following figure.



File name

Specify the name of the binary file. If the location of this file is on your MATLAB path, enter the filename. If the location of this file is not on your MATLAB path, use the **Browse** button to specify the full path to the file including the filename.

Video format

Specify the format of the binary video data as either **Four character codes** or **Custom**. See “Four Character Code Video Formats” on page 2-673 or “Custom Video Formats” on page 2-673 for more details.

Four character code

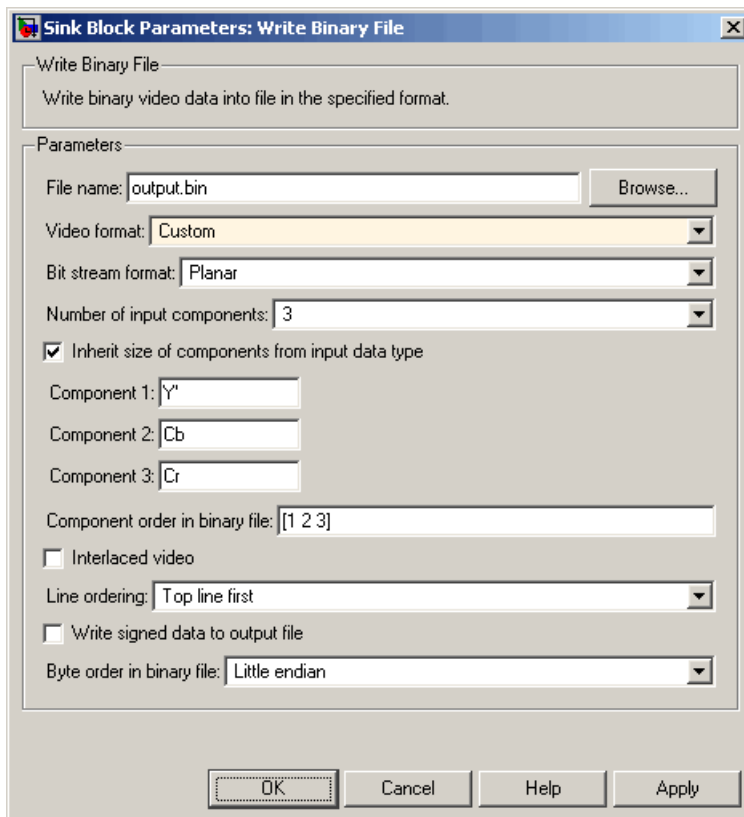
From the list, select the binary file format.

Line ordering

Specify how the block fills the binary file. If you select **Top line first**, the block first fills the binary file with the first row of the video frame. It then fills the file with the other rows in increasing order. If you select **Bottom line first**, the block first fills the

Write Binary File

binary file with the last row of the video frame. It then fills the file with the other rows in decreasing order.



Bit stream format

Specify whether you want your data in planar or packed format.

Number of input components

Specify the number of components in the video stream. This number corresponds to the number of block input ports.

Inherit size of components from input data type

Select this check box if you want each component to have the same number of bits as the input data type. If you clear this check box, you must specify the number of bits for each component.

Component

Specify the component names.

Component order in binary file

Specify how to arrange the components in the binary file.

Interlaced video

Select this check box if the video stream represents interlaced video data.

Write signed data to output file

Select this check box if your input data is signed.

Byte order in binary file

Use this parameter to specify whether the byte ordering in the output binary file is little endian or big endian.

See Also

Read Binary File Video and Image Processing Blockset
To Multimedia File Signal Processing Blockset

Write Binary File

System Object Reference

- “Analysis & Enhancement” on page 3-2
- “Conversions” on page 3-2
- “Filtering” on page 3-3
- “Geometric Transformations” on page 3-3
- “Morphological Operations” on page 3-4
- “Sinks” on page 3-4
- “Sources” on page 3-5
- “Statistics” on page 3-5
- “Text & Graphics” on page 3-6
- “Transforms” on page 3-6
- “Utilities” on page 3-7

Analysis & Enhancement

video.BlockMatcher	Estimate motion between images or video frames
video.BoundaryTracer	Trace object boundaries in binary images
video.ContrastAdjuster	Adjust image contrast by linear scaling
video.CornerDetector	Corner metric matrix and corner detector
video.Deinterlacer	Remove motion artifacts by deinterlacing input video signal
video.EdgeDetector	Find edges of objects in images
video.HistogramEqualizer	Enhance contrast of images using histogram equalization
video.OpticalFlow	Estimate object velocities
video.TemplateMatcher	Perform template matching by shifting template over image

Conversions

video.Autothresholder	Convert intensity image to binary image
video.ChromaResampler	Downsample or upsample chrominance components of images
video.ColorSpaceConverter	Convert color information between color spaces
video.DemosaicInterpolator	Demosaic Bayer's format images
video.GammaCorrector	Apply or remove gamma correction from images or video streams

<code>video.ImageComplementer</code>	Compute complement of pixel values in binary, intensity, or RGB images
<code>video.ImageDataTypeConverter</code>	Convert and scale input image to specified output data type

Filtering

<code>isfilterseparable</code>	Determine whether filter coefficients are separable
<code>video.Convolver2D</code>	Compute 2-D discrete convolution of two input matrices
<code>video.ImageFilter</code>	Perform 2-D FIR filtering of input matrix
<code>video.MedianFilter2D</code>	2D median filtering

Geometric Transformations

<code>video.GeometricRotator</code>	Rotate image by specified angle
<code>video.GeometricScaler</code>	Enlarge or shrink image sizes
<code>video.GeometricTransformer</code>	Apply projective or affine transformation to an image
<code>video.GeometricTransformEstimator</code>	Estimate geometric transformation from matching point pairs
<code>video.GeometricTranslator</code>	Translate image in two-dimensional plane using displacement vector

Morphological Operations

<code>video.ConnectedComponentLabeler</code>	Label and count the connected regions in a binary image
<code>video.MorphologicalBottomHat</code>	Bottom-hat filtering on image
<code>video.MorphologicalClose</code>	Perform morphological closing on image
<code>video.MorphologicalDilate</code>	Perform morphological dilation on an image
<code>video.MorphologicalErode</code>	Perform morphological erosion on an image
<code>video.MorphologicalOpen</code>	Perform morphological opening on an image
<code>video.MorphologicalTopHat</code>	Top-hat filtering on image

Sinks

<code>mplay</code>	View video from MATLAB workspace, multimedia file, or Simulink model
<code>video.BinaryFileWriter</code>	Write binary video data to files
<code>video.DeployableVideoPlayer</code>	Send video data to computer screen
<code>video.MultimediaFileWriter</code>	Write video frames and audio samples to multimedia file
<code>video.VideoPlayer</code>	Play video or display image sequences

Sources

video.BinaryFileReader	Read video data from binary files
video.MultimediaFileReader	Read video and/or audio samples from multimedia file

Statistics

video.Autocorrelator2D	Compute 2-D autocorrelation of input matrix
video.BlobAnalysis	Compute statistics for connected regions in a binary image
video.Crosscorrelator2D	Compute 2-D cross-correlation of two input matrices
video.Histogram2D	Generate histogram of each input matrix
video.LocalMaximaFinder	Find local maxima in matrices
video.Maximum	Find maximum values in input or sequence of inputs
video.Mean	Find mean value of input or sequence of inputs
video.Median	Find median values in an input.
video.Minimum	Find minimum values in input or sequence of inputs
video.PSNR	Compute peak signal-to-noise ratio (PSNR) between images
video.StandardDeviation	Find standard deviation of input or sequence of inputs
video.Variance	Find variance values in an input or sequence of inputs

Text & Graphics

video.AlphaBlender	Combine images, overlay images, or highlight selected pixels
video.MarkerInserter	Draw markers on output image
video.ShapeInserter	Draw rectangles, lines, polygons, or circles on images
video.TextInserter	Draw text on image or video stream

Transforms

video.DCT2D	Compute 2-D discrete cosine transform
video.FFT2D	Two-dimensional discrete Fourier transform
video.HoughLines	Find Cartesian coordinates of lines that are described by rho and theta pairs
video.HoughTransform	Find lines in images via Hough transform
video.IDCT2D	Compute 2-D inverse discrete cosine transform
video.IFFT2D	Two-dimensional inverse discrete Fourier transform
video.Pyramid	Perform Gaussian pyramid decomposition

Utilities

video.ImagePadder

Pad or crop input image along its rows, columns, or both

Alphabetical List

isfilterseparable

Purpose Determine whether filter coefficients are separable

Syntax `S = isfilterseparable(H)`
`[S, HCOL, HROW] = isfilterseparable(H)`

Description `S = isfilterseparable(H)` takes in the filter kernel `H` and returns 1 (true) when the filter is separable, and 0 (false) otherwise.

`[S, HCOL, HROW] = isfilterseparable(H)` uses the filter kernel, `H`, to return its vertical coefficients `HCOL` and horizontal coefficients `HROW` when the filter is separable. Otherwise, `HCOL` and `HROW` are empty.

Definition **Separable two dimensional filters**

Separable two-dimensional filters reflect the outer product of two vectors. Separable filters help reduce the number of calculations required.

A two-dimensional convolution calculation requires (*width*height*) number of multiplications for each output pixel. The general case equation for a two-dimensional convolution is:

$$Y(m,n) = \sum_k \sum_l H(k,l)U(m-k,n-l)$$

If the filter H is separable then,

$$H(k,l) = H_{row}(k) * H_{col}(l)$$

Shifting the filter instead of the image, the two-dimensional equation becomes:

$$Y(m,n) = \sum_k H_{row}(m-k) \sum_l H_{col}(n-l)U(m,n)$$

This calculation requires only (width + height) number of multiplications for each pixel.

Input Arguments

H

H numeric or logical, 2-D, and nonsparse.

Output Arguments

HCOL

HCOL is the same data type as input H when H is either single or double floating point. Otherwise, HCOL becomes double floating point. If S is true, HCOL is a vector of vertical filter coefficients. Otherwise, HCOL is empty.

HROW

HROW is the same data type as input H when H is either single or double floating point. Otherwise, HROW becomes double floating point. If S is true, HROW is a vector of horizontal filter coefficients. Otherwise, HROW is empty.

S

S is a logical variable that is true, when the filter is separable, and false, when it is not.

Examples

Determine if the Gaussian filter created using the `fspecial` function is separable.

```
% Create a gaussian filter
two_dimensional_filter = fspecial('gauss');
% Test with isfilterseparable
[iseparable, hcol, hrow] = ...
isfilterseparable(two_dimensional_filter)
```

When you run this example, notice that `hcol*hrow` equals the `two_dimensional_filter`. This result is expected for a Gaussian filter.

Algorithm

The `isfilterseparable` function uses the singular value decomposition `svd` function to determine the rank of the matrix.

See Also

2-D FIR Filter | `svd` | `rank`

isfilterseparable

Related Links

- [MATLAB Central — Separable Convolution](#)

Purpose	View video from MATLAB workspace, multimedia file, or Simulink model
Syntax	<pre>mplay mplay('filename.avi') mplay('filename.avi',FPS) mplay(A) mplay(A,FPS) mplay({line_handles}) mplay({'block',PORT})</pre>
Description	<p><code>mplay</code> opens an MPlay GUI that allows you to view video from files or the MATLAB workspace. The MPlay GUI does not play audio.</p> <p><code>mplay('filename.avi')</code> connects the MPlay GUI to the specified AVI file. You can also view video signals in Simulink models.</p> <p><code>mplay('filename.avi',FPS)</code> plays the specified frame rate in frames per second, (FPS). The FPS value equals that of the frame rate specified in the file.</p> <p><code>mplay(A)</code> connects the MPlay GUI to the variable in the MATLAB workspace, A.</p> <p><code>mplay(A,FPS)</code> plays the specified frame rate in frames per second, FPS. The FPS value defaults to 20.</p> <p><code>mplay({line_handles})</code> connects the MPlay GUI to one or three Simulink signal lines to display, where all signals must originate from the same block.</p> <p><code>mplay({'block',PORT})</code> connects the MPlay GUI to the output signal of the specified block, 'block', on output port port. All ports on the specified block are selected if PORT is omitted.</p>
Inputs	<p>A</p> <p>A is a variable in the MATLAB workspace, which must have one of the following formats:</p>

- MATLAB movie structure
- Intensity video. This input is an M -by- N -by- T , or M -by- N -by-1-by- T array, where the size of each frame is M -by- N and there are T image frames.
- RGB video array. This input is an M -by- N -by-3-by- T array, where the size of each RGB image is M -by- N -by-3 and there are T image frames.

For performance considerations, the video input A data type converts to uint8 as follows:

Supported Data Types	Converted to uint8
double	✓
single	✓
int8	✓
uint8	
int16	✓
uint16	
int32	✓
uint32	✓
Boolean	✓
Fixed point	✓

block

The **block** is a full path to a specified Simulink blockset block. To get the full block path name of the currently selected Simulink block, issue the command `mplay({gcb,1})` on the MATLAB command line.

filename.avi

`Filename.avi` is a specified AVI file.

FPS

FPS stands for frames per second. You can specify the frame rate in frames per second.

line_handles

`line_handles` are Simulink signal lines. To get the handles to the Simulink signals, `line_handles`, issue the command `mplay({gs1})` on the MATLAB command line.

port

The `port` refers to a Simulink block output port number.

Examples

Play a video created in MATLAB workspace.

```
fig=figure; % create a video
set(gca,'xlim',[-80 80],'ylim',[-80 80],'NextPlot', ...
'replace','Visible','off');
x = -pi:.1:pi;
radius = 0:length(x);
video = []; % initialize video variable
for i=length(x):-1:1
    patch(sin(x)*radius(i),cos(x)*radius(i), ...
[abs(cos(x(i))) 0 0]);
    F = getframe(gca);
    video = cat(4,video,F.cdata); % video is MxNx3xT
end
close(fig);
mplay(video); % display a video
```

Alternatives

Access this GUI by selecting **Tools > MPlay Video Viewer**.

See Also

Video Viewer | To Video Display

Tutorials

- Video and Image Processing Blockset demos

How To

- “Using the MPlay GUI”

video.AlphaBlender class

Purpose	Combine images, overlay images, or highlight selected pixels
Description	The AlphaBlender object combines two images, overlays one image over another, or highlights selected pixels.
Construction	<p>H = video.AlphaBlender returns an alpha blending System object, H, that combines the pixel values of two images, using an opacity factor of 0.75.</p> <p>H = video.AlphaBlender(<i>'PropertyName'</i>, <i>PropertyValue</i>, ...) returns an alpha blending object, H, with each specified property set to the specified value.</p>
Properties	<p>Operation</p> <p>Operation to perform</p> <p>Specify the operation that the object performs as Blend, Binary mask, or Highlight selected pixels. If this property is set to Blend, the object linearly combines the pixels of one image with another image. If this property is set to Binary mask, the object overwrites the pixel values of one image with the pixel values of another image. If this property is set to Highlight selected pixel, the object uses the binary image input, MASK, to determine which pixels are set to the maximum value supported by their data type.</p> <p>OpacitySource</p> <p>Source of opacity factor</p> <p>Specify how to determine the opacity factor(s) as Property or Input port. This property applies when you set the Operation property to Blend. The default value of this property is Property.</p> <p>Opacity</p> <p>Amount by which the object scales each pixel value before combining them</p>

Specify the amount by which the object scales each pixel value before combining them as a scalar value used for all pixels, or a matrix of values that defines the factor for each pixel. This property applies when you set the `OpacitySource` property to `Property`. This property is tunable. The default value of this property is 0.75.

MaskSource

Source of binary mask

Specify how to determine the masking factor(s) as `Property` or `Input port`. This property applies when you set the `Operation` property to `Binary mask`. The default value of this property is `Property`.

Mask

Which pixels are overwritten

Specify which pixels are overwritten as a binary scalar 0 or 1 used for all pixels, or a matrix of 0s and 1s that defines the factor for each pixel. This property applies when you set the `MaskSource` property to `Property`. This property is tunable. The default value of this property is 1.

LocationSource

Source of location of the upper-left corner of second input image

Specify how to enter location of the upper-left corner of second input image as `Property` or `Input port`. The default value of this property is `Property`.

Location

Location [row column] of upper-left corner of second input image relative to first input image

Specify the row and column position of upper-left corner of the second input image relative to upper-left corner of first input image as a two-element vector. This property applies when you

video.AlphaBlender class

set the `LocationSource` property to `Property`. The default value of this property is `[0 0]`. This property is tunable.

See `Coordinate Systems` for a discussion on pixel coordinates and spatial coordinates, which are the two main coordinate systems used in the Video and Image Processing Blockset software.

Fixed-Point Properties

`RoundingMethod`

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value of this property is `Floor`.

`OverflowAction`

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. The default value of this property is `Wrap`.

`OpacityDataType`

Opacity word and fraction lengths

Specify the opacity factor fixed-point data type as `Same word length as input` or `Custom`. The default value of this property is `Same word length as input`.

`CustomOpacityDataType`

Opacity word and fraction lengths

Specify the opacity factor fixed-point type as an unscaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OpacityDataType` property to `Custom`. The default value of this property is `numericType([], 16)`.

`ProductDataType`

Product word and fraction lengths

Specify the product fixed-point data type as `Same as first input` or `Custom`. The default value of this property is `Custom`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `ProductDataType` property to `Custom`. The default value of this property is `numerictype([],32,10)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as product`, `Same as first input`, or `Custom`. The default value of this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numerictype([],32,10)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Same as first input` or `Custom`. The default value of this property is `Same as first input`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set

video.AlphaBlender class

the `OutputDataType` property to `Custom`. The default value of this property is `numerictype([],32,10)`.

Methods

<code>clone</code>	Create alpha blender object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Blend images, overlay images, or highlight selected pixels

Examples

Blend two images

```
i1 = im2single(imread('blobs.png'));
i2 = im2single(imread('circles.png'));
halphablend = video.AlphaBlender;
ib = step(halphablend,i1,i2);
imshow(ib)
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Compositing block reference page. The object properties correspond to the block parameters.

See Also

`video.TextInserter`

Purpose	Create alpha blender object with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates an AlphaBlender System object <code>C</code> , with the same property values as <code>H</code> . The <code>clone</code> method creates a new unlocked object.

video.AlphaBlender.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.AlphaBlender.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the AlphaBlender System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose

Blend images, overlay images, or highlight selected pixels

Syntax

```
Y = step(H,I1,I2)
Y = step(H,I1,I2,OPACITY)
Y = step(H,I1,I2,MASK)
Y = step(H,I1,MASK)
Y = step(H,I1,I2,...,LOCATION)
```

Description

`Y = step(H,I1,I2)` performs the alpha blending operation on images I1 and I2.

`Y = step(H,I1,I2,OPACITY)` uses `OPACITY` input to combine pixel values of I1 and I2 when you set the `Operation` property to `Blend` and the `OpacitySource` property to `Input port`.

`Y = step(H,I1,I2,MASK)` uses `MASK` input to overlay I2 over I1 when you set the `Operation` property to `Binary mask` and the `MaskSource` property to `Input port`.

`Y = step(H,I1,MASK)` uses `MASK` input to determine which pixels in I1 are set to the maximum value supported by their data type when you set the `Operation` property to `Highlight selected pixels` and the `MaskSource` property to `Input port`.

`Y = step(H,I1,I2,...,LOCATION)` uses `LOCATION` input to specify the upper-left corner position of I2 when you set the `LocationSource` property to `Input port`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.Autocorrelator2D class

Purpose Compute 2-D autocorrelation of input matrix

Description The Autocorrelator2D object computes 2-D autocorrelation of input matrix.

Construction `H = video.Autocorrelator2D` returns a System object, H, that performs two-dimensional auto-correlation of an input matrix.

`H = video.Autocorrelator2D('PropertyName',PropertyValue,...)` returns a 2-D autocorrelation System object, H, with each specified property set to the specified value.

Properties **Fixed-Point Properties**

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as product`, `Same as input`, or `Custom`. The default value for this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType([],32,30)`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Custom`. The default value of this property is `numericType([],16,15)`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `ProductDataType` property to `Custom`. The default value of this property is `numericType([],32,30)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Same as input` or `Custom`. The default value for this property is `Same as input`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. The default value for this property is `Wrap`.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as `Same as input`, `Custom`. The default value for this property is `Same as input`.

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value for this property is `Floor`.

Methods

<code>clone</code>	Create 2-D autocorrelator object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method

video.Autocorrelator2D class

<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Compute cross-correlation of two input matrices

Examples

Compute the 2D autocorrelation of a matrix.

```
hac2d = video.Autocorrelator2D;  
x = [1 2;2 1];  
Y = step(hac2d, x)
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the 2-D Autocorrelation block reference page. The object properties correspond to the block parameters.

See Also

`video.Crosscorrelator2D` | `signalblks.Autocorrelator`

Purpose

Create 2-D autocorrelator object with same property values

Syntax

`C = clone(H)`

Description

`C = clone(H)` creates an `Autocorrelator2D` System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.Autocorrelator2D.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.Autocorrelator2D.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.Autocorrelator2D.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the Autocorrelator2D System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Compute cross-correlation of two input matrices

Syntax $Y = \text{step}(H, X)$

Description $Y = \text{step}(H, X)$ returns the 2-D autocorrelation, Y , of input matrix X .
Calling `step` on an unlocked `System` object will lock that object.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the `System` object issues a warning and re-initializes.

video.Autothresher class

Purpose	Convert intensity image to binary image
Description	Convert intensity images to binary images. Autothresholding uses Otsu's method, which determines the threshold by splitting the histogram of the input image such that the variance for each of the pixel groups is minimized.
Construction	<p>H = video.Autothresher returns a System object, H, that automatically converts an intensity image to a binary image.</p> <p>H = video.Autothresher('PropertyName',PropertyValue,...) returns an autothreshold object, H, with each specified property set to the specified value.</p>
Properties	<p>Operator</p> <p>Threshold operator on input matrix values</p> <p>Specify the condition the System object places on the input matrix values as > or <=. If this property is set to > and the input value is greater than the threshold value, the System object outputs 1; otherwise, it outputs 0. If this property is set to <= and the input value is less than or equal to the threshold value, the System object outputs 1; otherwise, it outputs 0. The default value of this property is >.</p> <p>ThresholdOutputPort</p> <p>Enable threshold output</p> <p>Set this property to true to enable the output of the calculated threshold values. The default value of this property is false.</p> <p>EffectivenessOutputPort</p> <p>Enable threshold effectiveness output</p> <p>Set this property to true to enable the output of the effectiveness of the thresholding. This effectiveness metric ranges from 0 to 1. If every pixel has the same value, the effectiveness metric is 0. If the image has two pixel values or the histogram of the image</p>

pixels is symmetric, the effectiveness metric is 1. The default value of this property is false.

InputRangeSource

Source of input data range

Specify the input data range as `Auto` or `Property`. If this property is set to `Auto`, then the `System` object assumes that the input range is between 0 and 1, inclusive, for floating point data types. For all other data types, the input range is the full range of the data type. To specify a different input data range, set this property to `Property`. The default value of this property is `Auto`.

InputRange

Input data range

Specify the input data range as a two element numeric row vector. First element of the input data range vector represents the minimum input value while the second element represents the maximum value. This property applies when you set the `InputRangeSource` property to `Property`.

InputRangeViolationAction

Behavior when input values are out of range

Specify the `System` object's behavior when the input values are outside the expected data range as `Ignore`, or `Saturate`. This property applies when you set the `InputRangeSource` property to `Property`. The default value for this property is `Saturate`.

ThresholdScaleFactor

Threshold scale factor

Specify the threshold scale factor as a numeric scalar greater than 0. The `System` object multiplies this scalar value with the threshold value computed by Otsu's method and uses the result as the new threshold value. The default value of this property is 1, i.e. no threshold scaling. This property is tunable.

video.Autothresher class

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value for this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. The default value of this property is `Wrap`.

Product1DataType

Product-1 word and fraction lengths

This is a constant property with value `Custom`.

CustomProduct1DataType

Product-1 word and fraction lengths

Specify the product-1 fixed-point type as a signed numeric type object with a `Signedness` of `Auto`. The default value of this property is `numerictype([],32)`.

Accumulator1DataType

Accumulator-1 word and fraction lengths

Specify the accumulator-1 fixed-point data type as `Same as product 1`, `Custom`. The default value of this property is `Same as product 1`.

CustomAccumulator1DataType

Accumulator-1 word and fraction lengths

Specify the accumulator-1 fixed-point type as a signed numeric type object with a `Signedness` of `Auto`. This

property applies when you set the `Accumulator1DataType` property to `Custom`. The default value of this property is `numerictype([],32)`.

`Product2DataType`

Product-2 word and fraction lengths

This is a constant property with value `Custom`.

`CustomProduct2DataType`

Product-2 word and fraction lengths

Specify the product-2 fixed-point type as a signed `numerictype` object with a `Signedness` of `Auto`. The default value of this property is `numerictype([],32)`.

`Accumulator2DataType`

Accumulator-2 word and fraction lengths

Specify the accumulator-2 fixed-point data type as `Same` as product 2, `Custom`. The default value of this property is `Same` as product 2.

`CustomAccumulator2DataType`

Accumulator-2 word and fraction lengths

Specify the accumulator-2 fixed-point type as a signed `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `Accumulator2DataType` property to `Custom`. The default value of this property is `numerictype([],32)`.

`Product3DataType`

Product-3 word and fraction lengths

This is a constant property with value `Custom`.

`CustomProduct3DataType`

Product-3 word and fraction lengths

video.Autothresher class

Specify the product-3 fixed-point type as a signed `numericType` object with a `Signedness` of `Auto`. The default value of this property is `numericType([],32)`.

Accumulator3DataType

Accumulator-3 word and fraction lengths

Specify the accumulator-3 fixed-point data type as `Same as product 3`, `Custom`. This property applies when you set the `EffectivenessOutputPort` property to `true`. The default value of this property is `Same as product 3`.

CustomAccumulator3DataType

Accumulator-3 word and fraction lengths

Specify the accumulator-3 fixed-point type as a signed `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `EffectivenessOutputPort` property to `true`, and when you set the `Accumulator3DataType` property to `Custom`. The default value of this property is `numericType([],32)`.

Product4DataType

Product-4 word and fraction lengths

Specify the product-4 fixed-point data type as `Same as input`, or `Custom`. The default value for this property is `Custom`.

CustomProduct4DataType

Product-4 word and fraction lengths

Specify the product-4 fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `Product4DataType` property to `Custom`. The default value of this property is `numericType([],32,15)`.

Accumulator4DataType

Accumulator-4 word and fraction lengths

Specify the accumulator-4 fixed-point data type as Same as product 4, Custom. The default value of this property is Same as product 4.

CustomAccumulator4DataType

Accumulator-4 word and fraction lengths

Specify the accumulator-4 fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `Accumulator4DataType` property to `Custom`. The default value of this property is `numericType([],16,4)`.

QuotientDataType

Quotient word and fraction lengths

Specify the quotient fixed-point data type as `Custom`.

CustomQuotientDataType

Quotient word and fraction lengths

Specify the quotient fixed-point type as a signed `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `QuotientDataType` property to `Custom`. The default value of this property is `numericType([],32)`.

EffectivenessDataType

Effectiveness metric word and fraction lengths

This is a constant property with value `Custom`. This property applies when you set the `EffectivenessOutputPort` property to `true`.

CustomEffectivenessDataType

Effectiveness metric word and fraction lengths

Specify the effectiveness metric fixed-point type as a signed `numericType` object with a `Signedness` of `Auto`. This property

video.Autothresher class

applies when you set the `EffectivenessOutputPort` property to `true`. The default value of this property is `numerictype([],16)`.

Methods

<code>clone</code>	Create autothresher object with same property values
<code>getNumInputs</code>	Number of expected inputs to <code>step</code> method
<code>getNumOutputs</code>	Number of outputs from <code>step</code> method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Convert input intensity image to a binary image

Examples

Convert an image of peppers to binary.

```
img = im2single(rgb2gray(imread('peppers.png')));  
imshow(img);  
hautoth = video.Autothresher;  
bin = step(hautoth,img);  
pause(2);  
figure;imshow(bin);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the [Autothreshold](#) block reference page. The object properties correspond to the block parameters, except for:

- You can only specify a value of `Ignore` or `Saturate` for the `InputRangeViolationAction` property of the `System` object. The object does not support the `Error` and `Warn` and `Saturate` options that the corresponding **When data range is exceeded** block parameter offers.

See Also `video.ColorSpaceConverter`

video.Autothresher.clone

Purpose Create autothresher object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a `Autothresher System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.Autothresher.getNumInputs

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.Autothresher.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the Autothresher System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.Autothresher.step

Purpose Convert input intensity image to a binary image

Syntax
BW = step(H,I)
[BW,TH] = step(H,I)
[... ,EMETRIC] = step(H,I)

Description BW = step(H,I) converts input intensity image, I, to a binary image, BW.

[BW,TH] = step(H,I) also returns the threshold, TH, when the ThresholdOutputPort property is true.

[... ,EMETRIC] = step(H,I) also returns EMETRIC, a metric indicating the effectiveness of thresholding the input image when the EffectivenessOutputPort property is true. The lower bound of the metric (zero) is attainable only by images having a single gray level, and the upper bound (one) is attainable only by two-valued images.

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Read video data from binary files
Description	The BinaryFileReader object reads video data from binary files.
Construction	<p>H = video.BinaryFileReader returns a System object, H, that reads binary video data from the specified file in I420 Four Character Code (FOURCC) video format.</p> <p>H = video.BinaryFileReader('PropertyName',PropertyValue,...) returns a binary file reader System object, H, with each specified property set to the specified value.</p> <p>H = video.BinaryFileReader(FILE,'PropertyName',PropertyValue,...) returns a binary file reader System object, H, with the Filename property set to FILE and other specified properties set to the specified values.</p>
Properties	<p>Filename</p> <p>Name of binary file to read from</p> <p>Specify the name of the binary file as a string. The full path for the file needs to be specified only if the file is not on the MATLAB path. The default value of this property is vipmen.bin.</p> <p>VideoFormat</p> <p>Format of binary video data</p> <p>Specify the format of the binary video data as Four character codes, or Custom. The default value of this property is Four character codes.</p> <p>FourCharCode</p> <p>Four Character Code video format</p> <p>Specify the binary file format from the available list of Four Character Code video formats. For more information on Four Character Codes, see http://www.fourcc.org. This property applies</p>

video.BinaryFileReader class

when you set the VideoFormat property to Four character codes.

BitstreamFormat

Format of data as planar or packed

Specify the data format as Planar or Packed. This property applies when you set the VideoFormat property to Custom. The default value of this property is Planar.

OutputSize

Size of output matrix

Specify the size of the output matrix. This property applies when you set the BitstreamFormat property to Packed.

VideoComponentCount

Number of video components in video stream

Specify the number of video components in the video stream as 1, 2, 3 or 4. This number corresponds to the number of video component outputs. This property applies when you set the VideoFormat property to Custom. The default value of this property is 3.

VideoComponentBits

Bit size of video components

Specify the bit sizes of video components as an integer valued vector of length N , where N is the value of the VideoComponentCount property. This property applies when you set the VideoFormat property to Custom. The default value of this property is [8 8 8].

VideoComponentSizes

Size of output matrix

Specify the size of the output matrix. This property must be set to an N -by-2 array, where N is the value of the VideoComponentCount

property. Each row of the matrix corresponds to the size of that video component, with the first element denoting the number of rows and the second element denoting the number of columns. This property applies when you set the `VideoFormat` property to `Custom` and the `BitstreamFormat` property to `Planar`. The default value of this property is [120 160; 60 80; 60 80].

VideoComponentOrder

How to arrange video components in binary file

Specify how to arrange the components in the binary file. This property must be set to a vector of length N , where N is the value of the `VideoComponentCount` property. This property applies when you set the `VideoFormat` property to `Custom`. The default value of this property is [1 2 3].

InterlacedVideo

Whether data stream represents interlaced video

Set this property to `true` if the video stream represents interlaced video data. This property applies when you set the `VideoFormat` property to `Custom`. The default value of this property is `false`.

LineOrder

How to fill binary file

Specify how to fill the binary file as `Top line first`, or `Bottom line first`. If this property is set to `Top line first`, the `System` object first fills the binary file with the first row of the video frame. If it is set to `Bottom line first`, the `System` object first fills the binary file with the last row of the video frame. The default value of this property is `Top line first`.

SignedData

Whether input data is signed

Set this property to `true` if the input data is signed. This property applies when you set the `VideoFormat` property to `Custom`. The default value of this property is `false`.

video.BinaryFileReader class

ByteOrder

Byte ordering as little endian or big endian

Specify the byte ordering in the output binary file as `Little endian`, `Big endian`. This property applies when you set the `VideoFormat` property to `Custom`. The default value of this property is `Little endian`.

PlayCount

Number of times to play the file

Specify the number of times to play the file as a positive integer or `inf`. The default value of this property is 1.

Methods

<code>clone</code>	Create binary file reader object with same property values
<code>close</code>	Release resources for the <code>System</code> object
<code>getNumInputs</code>	Number of expected inputs to <code>step</code> method
<code>getNumOutputs</code>	Number of outputs from <code>step</code> method
<code>isDone</code>	End-of-file status (logical)
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>reset</code>	Reset to beginning of file
<code>step</code>	Read video components from a binary file

Examples

Read in a binary video file and play it back on the screen

```
hbfr = video.BinaryFileReader('ecolicells.bin');
```



```
hbfr.VideoFormat = 'Custom';
hbfr.VideoComponentCount = 1;
hbfr.VideoComponentBits = 16;
hbfr.VideoComponentSizes = [442 538];
hbfr.VideoComponentOrder = 1;

hvp = video.VideoPlayer;
while ~isDone(hbfr)
    y = step(hbfr);
    step(hvp,y);
end
close(hbfr); % close the input file
close(hvp); % close the video display
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Write Binary File block reference page. The object properties correspond to the block parameters.

See Also

[video.MultimediaFileReader](#) | [video.BinaryFileWriter](#)

video.BinaryFileReader.clone

Purpose Create binary file reader object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a `BinaryFileReader System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object with uninitialized states.

Purpose	Release resources for the BinaryFileReader System object
Syntax	<code>close(H)</code>
Description	<code>close(H)</code> releases system resources (such as memory, file handles or hardware connections).

video.BinaryFileReader.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.BinaryFileReader.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.BinaryFileReader.isDone

Purpose	End-of-file status (logical)
Syntax	TF = isDone(H)
Description	TF = isDone(H) returns true if the BinaryFileReader System object, H , has reached the end of the binary file. If PlayCount property is set to a value greater than 1 , this method will return true every time the end is reached.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the BinaryFileReader System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.BinaryFileReader.reset

Purpose	Reset to beginning of file
Syntax	<code>reset(H)</code>
Description	<code>reset(H)</code> System object H to the beginning of the specified file.

Purpose Read video components from a binary file

Syntax

```
[Y,Cb,Cr] = step(H)
Y = step(H)
[Y,Cb] = step(H)
[Y,Cb,Cr] = step(H)
[Y,Cb,Cr,Alpha] = step(H)
[... , EOF] = step(H)
```

Description [Y,Cb,Cr] = step(H) reads the luma, Y and chroma, Cb and Cr components of a video stream from the specified binary file when you set the VideoFormat property to 'Four character codes'.

Y = step(H) reads the video component Y from the binary file when you set the VideoFormat property to Custom and the VideoComponentCount property to 1.

[Y,Cb] = step(H) reads video the components Y and Cb from the binary file when you set the VideoFormat property to Custom and the VideoComponentCount property to 2.

[Y,Cb,Cr] = step(H) reads the video components Y, Cb and Cr when you set the VideoFormat property to Custom, and the VideoComponentCount property to 3.

[Y,Cb,Cr,Alpha] = step(H) reads the video components Y, Cb, Cr and Alpha when you set the VideoFormat property to Custom and the VideoComponentCount property to 4.

[... , EOF] = step(H) also returns the end-of-file indicator, EOF . EOF is set to true each time the output contains the last video frame in the file.

video.BinaryFileReader.step

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Write binary video data to files
Description	The BinaryFileWriter object writes binary video data to files.
Construction	<p>H = video.BinaryFileWriter returns a System object, H, that writes binary video data to an output file, output.bin in the I420 Four Character Code format.</p> <p>H = video.BinaryFileWriter('PropertyName',PropertyValue, ...) returns a binary file writer System object, H, with each specified property set to the specified value.</p> <p>H = video.BinaryFileWriter(FILE,'PropertyName',PropertyValue, ...) returns a binary file writer System object, H, with the Filename property set to FILE and other specified properties set to the specified values.</p>
Properties	<p>Filename</p> <p>Name of binary file to write to</p> <p>Specify the name of the binary file as a string. The default value of this property is the file output.bin.</p> <p>VideoFormat</p> <p>Format of binary video data</p> <p>Specify the format of the binary video data as Four character codes, or Custom. The default value of this property is Four character codes.</p> <p>FourCharCode</p> <p>Four Character Code video format</p> <p>Specify the binary file format from the available list of Four Character Code video formats. For more information on Four Character Codes, see http://www.fourcc.org. This property applies</p>

video.BinaryFileWriter class

when you set the VideoFormat property to Four character codes.

BitstreamFormat

Format of data as planar or packed

Specify the data format as Planar or Packed. This property applies when you set the VideoFormat property to Custom. The default value of this property is Planar.

VideoComponentCount

Number of video components in video stream

Specify the number of video components in the video stream as 1, 2, 3 or 4. This number corresponds to the number of video component outputs. This property applies when you set the VideoFormat property to Custom. The default value of this property is 3.

VideoComponentBitsSource

How to specify the size of video components

Indicate how to specify the size of video components as Auto or Property. If this property is set to Auto, each component will have the same number of bits as the input data type. Otherwise, the number of bits for each video component is specified using the VideoComponentBits property. This property applies when you set the VideoFormat property to Custom. The default value of this property is Auto.

VideoComponentBits

Bit size of video components

Specify the bit size of video components using a vector of length N, where N is the value of the VideoComponentCount property. This property applies when you set the VideoComponentBitsSource property to Property. The default value of this property is [8 8 8].

VideoComponentOrder

How to arrange video components in binary file

Specify how to arrange the components in the binary file. This property must be set to a vector of length N , where N is the value of the `VideoComponentCount` property. This property applies when you set the `VideoFormat` property to `Custom`. The default value of this property is `[1 2 3]`.

InterlacedVideo

Whether data stream represents interlaced video

Set this property to true if the video stream represents interlaced video data. This property applies when you set the `VideoFormat` property to `Custom`. The default value of this property is `false`.

LineOrder

How to fill binary file

Specify how to fill the binary file as `Top line first`, or `Bottom line first`. If this property is set to `Top line first`, the object first fills the binary file with the first row of the video frame. Otherwise, the object first fills the binary file with the last row of the video frame. The default value of this property is `Top line first`.

SignedData

Whether input data is signed

Set this property to true if the input data is signed. This property applies when you set the `VideoFormat` property to `Custom`. The default value of this property is `false`.

ByteOrder

Byte ordering as little endian or big endian

Specify the byte ordering in the output binary file as `Little endian`, or `Big endian`. This property applies when you set

video.BinaryFileWriter class

the VideoFormat property to Custom. The default value of this property is Little endian.

Methods

clone	Create binary file writer object with same property values
close	Close binary data file
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Write video frame to output file

Examples

Write video to a binary video file

```
filename = fullfile(tempdir,'output.bin');
hbfr = video.BinaryFileReader;
hbfw = video.BinaryFileWriter(filename);
while ~isDone(hbfr)
    [y,cb,cr] = step(hbfr);
    step(hbfw,y,cb,cr);
end
close(hbfr); % close the input file
close(hbfw); % close the output file
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Write Binary File block reference page. The object properties correspond to the block parameters.

See Also

[video.MultimediaFileWriter](#) | [video.BinaryFileReader](#)

Purpose	Create binary file writer object with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates a <code>BinaryFileWriter</code> System object <code>C</code> , with the same property values as <code>H</code> . The <code>clone</code> method creates a new unlocked object.

video.BinaryFileWriter.close

Purpose	Close binary data file
Syntax	<code>close(H)</code>
Description	<code>close(H)</code> closes the binary data file.

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.BinaryFileWriter.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the BinaryFileWriter System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.BinaryFileWriter.step

Purpose Write video frame to output file

Syntax

```
step(H,Y,Cb,Cr)
step(H,Y)
step(H,Y,Cb)
step(H,Y,Cb,Cr)
step(H,Y,Cb,Cr,Alpha)
```

Description `step(H,Y,Cb,Cr)` writes one frame of video to the specified output file. Y , Cb, Cr represent the luma (Y) and chroma (Cb and Cr) components of a video stream. This option applies when you set the `VideoFormat` property to Four character codes.

`step(H,Y)` writes video component Y to the output file when the `VideoFormat` property is set to Custom and the `VideoComponentCount` property is set to 1.

`step(H,Y,Cb)` writes video components Y and Cb to the output file when the `VideoFormat` property is Custom and the `VideoComponentCount` property is set to 2.

`step(H,Y,Cb,Cr)` writes video components Y , Cb and Cr to the output file when the `VideoFormat` property is set to Custom and the `VideoComponentCount` property is set to3.

`step(H,Y,Cb,Cr,Alpha)` writes video components Y , Cb, Cr and Alpha to the output file when the `VideoFormat` property is set to Custom, and the `VideoComponentCount` property is set to 4.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Compute statistics for connected regions in a binary image
Description	The BlobAnalysis object computes statistics for connected regions in a binary image.
Construction	<p>H = video.BlobAnalysis returns a System object, H, that computes area, centroid, and bounding box for connected regions in a binary image.</p> <p>H = video.BlobAnalysis('PropertyName',PropertyValue,...) returns a System object, H, with each specified property set to the specified value.</p>
Properties	<p>AreaOutputPort</p> <p>Return blob area</p> <p>Setting this property to true outputs the area of the blobs. The default value for this property is true.</p> <p>CentroidOutputPort</p> <p>Return coordinates of blob centroids</p> <p>Set this property to true to output the coordinates of the centroid of the blobs. The default value for this property is true.</p> <p>BoundingBoxOutputPort</p> <p>Return coordinates of bounding boxes</p> <p>Set this property to true to output the coordinates of the bounding boxes. The default value for this property is true.</p> <p>MajorAxisLengthOutputPort</p> <p>Return vector whose values represent lengths of ellipses' major axes</p> <p>Set this property to true to output a vector whose values represent the lengths of the major axes of the ellipses that have the same normalized second central moments as the labeled regions. This</p>

video.BlobAnalysis class

property applies when you set the `OutputDataType` property to `double` or `single`. The default value for this property is `false`.

MinorAxisLengthOutputPort

Return vector whose values represent lengths of ellipses' minor axes

Set this property to `true` to output a vector whose values represent the lengths of the minor axes of the ellipses that have the same normalized second central moments as the labeled regions. This property is available when the `OutputDataType` property is `double` or `single`. The default value of this property is `false`.

OrientationOutputPort

Return vector whose values represent angles between ellipses' major axes and x-axis

Set this property to `true` to output a vector whose values represent the angles between the major axes of the ellipses and the x-axis. This property applies when you set the `OutputDataType` property to `double` or `single`. The default value of this property is `false`.

EccentricityOutputPort

Return vector whose values represent ellipses' eccentricities

Set this property to `true` to output a vector whose values represent the eccentricities of the ellipses that have the same second moments as the region. This property applies when you set the `OutputDataType` property to `double` or `single`. The default value for this property is `false`.

EquivalentDiameterSquaredOutputPort

Return vector whose values represent equivalent diameters squared

Set this property to `true` to output a vector whose values represent the equivalent diameters squared. The default value for this property is `false`.

ExtentOutputPort

Return vector whose values represent results of dividing blob areas by bounding box areas

Set this property to `true` to output a vector whose values represent the results of dividing the areas of the blobs by the area of their bounding boxes. The default value for this property is `false`.

PerimeterOutputPort

Return vector whose values represent estimates of blob perimeter lengths

Set this property to `true` to output a vector whose values represent estimates of the perimeter lengths, in pixels, of each blob. The default value of this property is `false`.

OutputDataType

Output data type of statistics

Specify the data type of the output statistics as `double`, `single`, or `Fixed point`. Area and bounding box outputs are always an `int32` data type. Major axis length, Minor axis length, Orientation and Eccentricity do not apply when you set this property to `Fixed point`. The default value of this property is `double`.

Connectivity

Which pixels are connected to each other

Specify connectivity of pixels as 4 or 8.

LabelMatrixOutputPort

Return label matrix

video.BlobAnalysis class

Set this property to `true` to output the label matrix. The default value for this property is `false`.

MaximumCount

Maximum number of labeled regions in each input image

Specify the maximum number of blobs in the input image as a positive scalar integer. The default value for this property is 50.

NumBlobsOutputPort

Return scalar value that represents actual number of labeled regions in each image

Set this property to `true` to output a scalar value that represents the actual number of labeled regions in each image. The default value of this property is `false`.

MinimumBlobAreaSource

Source of minimum blob area

Specify how the `BlobAnalysis` object determines the minimum blob area using `Auto` or `Property`. In this case, the minimum blob area is 0. The default value for this property is `Auto`.

MinimumBlobArea

Minimum blob area in pixels

Specify the minimum blob area in pixels. This property applies when you set the `MinimumBlobAreaSource` to `Property`. The default value for this property is 0.

MaximumBlobAreaSource

Source of maximum blob area

Specify how the `BlobAnalysis` object determines the maximum blob area using `Auto` or `Property`. In this case, the maximum blob area is `intmax('uint32')`. The default value for this property is `Auto`.

MaximumBlobArea

Maximum blob area in pixels

Specify the maximum blob area in pixels. This property applies when you set the `MaximumBlobAreaSource` property to `Property`. The default value for this property is `intmax('uint32')`.

`ExcludeBorderBlobs`

Exclude blobs that contain at least one border pixel

Set this property to `true` if you do not want to label blobs that contain at least one border pixel. The default value for this property is `false`.

`FillEmptySpaces`

Fill empty spaces in statistical vectors with specified values

Use this property to fill the empty spaces in the statistical vectors with the value(s) you specify in the `FillValues` property. The default value for this property is `true`.

`FillValues`

Value(s) used to fill all empty spaces in statistical vectors

Specify a scalar value that is used to fill all the empty spaces in the statistical vectors or a vector, where the object uses each vector element to fill a different statistics vector. This property applies when you set the `FillEmptySpaces` property to `true`. The default value for this property is `-1`.

Fixed-Point Properties

`RoundingMethod`

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. This property applies when you set the `OutputDataType` property to `Fixed point`. The default value of this property is `Floor`.

video.BlobAnalysis class

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. This property applies when you set the `OutputDataType` property to `Fixed point`.

ProductDataType

Product word and fraction lengths

This property is constant and is set to `Custom`. This property applies when you set the `OutputDataType` property to `Fixed point` and the `EquivalentDiameterSquaredOutputPort` property to `true`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Fixed point` and the `EquivalentDiameterSquaredOutputPort` property to `true`. The default value of this property is `numericType([],32,16)`.

AccumulatorDataType

Accumulator word and fraction lengths

This property is constant and is set to `Custom`. This property applies when you set the `OutputDataType` property to `Fixed point`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Fixed point`. The default value of this property is `numericType([],32,0)`.

CentroidDataType

Centroid word and fraction lengths

Specify the centroid output's fixed-point data type as Same as accumulator, Custom. This property applies when you set the OutputDataType property to Fixed point and the CentroidOutputPort property to true. The default value of this property is Custom.

CustomCentroidDataType

Centroid word and fraction lengths

Specify the centroid output's fixed-point type as a scaled numeric type object with a Signedness of Auto. This property applies when you set the OutputDataType property to Fixed point and the CentroidDataType property to Custom and when the CentroidOutputPort property to true. The default value of this property is numeric type ([], 32, 16).

EquivalentDiameterSquaredDataType

Equivalent diameter squared word and fraction lengths

Specify the equivalent diameters squared output's fixed-point data type as Same as accumulator, Same as product, Custom. This property applies when you set the OutputDataType property to Fixed point and the EquivalentDiameterSquaredOutputPort property to true. The default value for this property is Same as product.

CustomEquivalentDiameterSquaredDataType

Equivalent diameter squared word and fraction lengths

Specify the equivalent diameters squared output's fixed-point type as a scaled numeric type object with a Signedness of Auto. This property applies when you set the OutputDataType property to Fixed point and the EquivalentDiameterSquaredDataType property to Custom and when the EquivalentDiameterSquaredOutputPort

video.BlobAnalysis class

property to true. The default value of this property is `numerictype([],32,16)`.

ExtentDataType

Extent word and fraction lengths

Specify the extent output's fixed-point data type as `Same` as accumulator or `Custom`. This property applies when you set the `OutputDataType` property to `Fixed point` and the `ExtentOutputPort` property to true.

CustomExtentDataType

Extent word and fraction lengths

Specify the extent output's fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Fixed point`, the `ExtentDataType` property to `Custom` and the `ExtentOutputPort` property to true. The default value of this property is `numerictype([],16,14)`.

PerimeterDataType

Perimeter word and fraction lengths

Specify the perimeter output's fixed-point data type as `Same` as accumulator or `Custom`. This property applies when you set the `OutputDataType` property to `Fixed point` and the `PerimeterOutputPort` property to true. The default value of this property is `Custom`.

CustomPerimeterDataType

Perimeter word and fraction lengths

Specify the perimeter output's fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Fixed point`, the `PerimeterDataType` property to `Custom` and the `PerimeterOutputPort` property to true. The default value of this property is `numerictype([],32,16)`.

Methods

clone	Create blob analysis object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Compute and returns statistics of input binary image

Examples

Find the centroid of a blob.

```
hblob = video.BlobAnalysis;  
hblob.AreaOutputPort = false;  
hblob.BoundingBoxOutputPort = false;  
hblob.NumBlobsOutputPort = true;  
img = logical([0 0 0 0 0 0; ...  
0 1 1 1 1 0; ...  
0 1 1 1 1 0; ...  
0 1 1 1 1 0; ...  
0 0 0 0 0 0]);  
[centroid, numBlobs] = step(hblob, img);  
centroid = centroid(:,numBlobs)
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Blob Analysis block reference page. The object properties correspond to the block parameters, except for:

- The **Warn if maximum number of blobs is exceeded** block parameter does not have a corresponding object property. The object does not issue a warning.

video.BlobAnalysis class

- The **Output blob statistics as a variable-size signal** block parameter does not have a corresponding object property.

See Also

`video.Autothresher` | `video.ConnectedComponentLabeler`

Purpose

Create blob analysis object with same property values

Syntax

`C = clone(H)`

Description

`C = clone(H)` creates a `BlobAnalysis System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.BlobAnalysis.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.BlobAnalysis.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.BlobAnalysis.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the BlobAnalysis System objects.

The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.

Purpose Compute and returns statistics of input binary image

Syntax

```
AREA = step(H,BW)
[ ...,CENTROID] = step(H,BW)
[ ...,BBOX] = step(H,BW)
[ ...,MAJORAXIS] = step(H,BW)
[ ...,MINORAXIS] = step(H,BW)
[ ...,ORIENTATION] = step(H,BW)
[ ...,ECCENTRICITY] = step(H,BW)
[ ...,EQDIASQ] = step(H,BW)
[ ...,EXTENT] = step(H,BW)
[ ...,PERIMETER] = step(H,BW)
[ ...,LABEL] = step(H,BW)
[ ...,NUMBLOBS] = step(H,BW)
[ AREA,CENTROID,BBOX] = step(H,BW)
```

Description

`AREA = step(H,BW)` computes the AREA of the blobs found in input binary image BW when the AreaOutputPort property is set to true.

`[...,CENTROID] = step(H,BW)` computes the CENTROID of the blobs found in input binary image BW when the CentroidOutputPort property is set to true.

`[...,BBOX] = step(H,BW)` computes the bounding box BBOX of the blobs found in input binary image BW when the BoundingBoxOutputPort property is set to true.

`[...,MAJORAXIS] = step(H,BW)` computes the major axis length MAJORAXIS of the blobs found in input binary image BW when the MajorAxisLengthOutputPort property is set to true.

`[...,MINORAXIS] = step(H,BW)` computes the minor axis length MINORAXIS of the blobs found in input binary image BW when the MinorAxisLengthOutputPort property is set to true.

`[...,ORIENTATION] = step(H,BW)` computes the ORIENTATION of the blobs found in input binary image BW when the OrientationOutputPort property is set to true.

video.BlobAnalysis.step

[...,ECCENTRICITY] = step(H,BW) computes the ECCENTRICITY of the blobs found in input binary image BW when the EccentricityOutputPort property is set to true.

[...,EQDIASQ] = step(H,BW) computes the equivalent diameter squared EQDIASQ of the blobs found in input binary image BW when the EquivalentDiameterSquaredOutputPort property is set to true.

[...,EXTENT] = step(H,BW) computes the EXTENT of the blobs found in input binary image BW when the ExtentOutputPort property is set to true.

[...,PERIMETER] = step(H,BW) computes the PERIMETER of the blobs found in input binary image BW when the PerimeterOutputPort property is set to true.

[...,LABEL] = step(H,BW) returns a label matrix LABEL of the blobs found in input binary image BW when the LabelMatrixOutputPort property is set to true.

[...,NUMBLOBS] = step(H,BW) returns the number of blobs found in the input binary image BW when the NumBlobsOutputPort property is set to true.

[AREA,CENTROID,BBOX] = step(H,BW) returns the area, centroid and the bounding box of the blobs, when the AreaOutputPort, CentroidOutputPort and BoundingBoxOutputPort properties are set to true. You can use this to calculate multiple statistics.

The step method computes and returns statistics of the input binary image depending on the property values specified. The different options can be used simultaneously. The order of the returned values when there are multiple outputs are in the order they are described.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.BlockMatcher class

Purpose	Estimate motion between images or video frames
Description	The BlockMatcher object estimates motion between images or video frames.
Construction	<p>H = video.BlockMatcher returns a System object, H, that estimates motion between two images or two video frames using a block matching method by moving a block of pixels over a search region.</p> <p>H = video.BlockMatcher('PropertyName',PropertyValue,...) returns a block matcher System object, H, with each specified property set to the specified value.</p>
Properties	<p>ReferenceFrameSource</p> <p>Source of reference frame</p> <p>Specify the source of the reference frame as Input port or Property. When you set the ReferenceFrameSource property to Input port a reference frame input must be specified to the step method of the block matcher object. The default value of this property is Property.</p> <p>ReferenceFrameDelay</p> <p>Number of frames between reference and current frames</p> <p>Specify the number of frames between the reference frame and the current frame as a scalar integer value greater than or equal to zero. This property applies when you set the ReferenceFrameSource property to Property. The default value of this property is 1.</p> <p>SearchMethod</p> <p>Search method for best match</p> <p>Specify how to locate the block of pixels in frame $k+1$ that best matches the block of pixels in frame k. You can specify the search method as Exhaustive or Three-step. If this property is set to Exhaustive, the block matcher object selects the location of</p>

the block of pixels in frame $k+1$ by moving the block over the search region one pixel at a time. This process is computationally expensive.

If this property is set to `Three-step`, the block matcher object searches for the block of pixels in frame $k+1$ that best matches the block of pixels in frame k using a steadily decreasing step size. The object begins with a step size approximately equal to half the maximum search range. In each step, the object compares the central point of the search region to eight search points located on the boundaries of the region and moves the central point to the search point whose values is the closest to that of the central point. The object then reduces the step size by half, and begins the process again. This option is less computationally expensive, though it might not find the optimal solution. The default value of this property is `Exhaustive`.

BlockSize

Size of block in pixels

Specify the size of the block in pixels. The default value of this property is `[17 17]`.

Overlap

Overlap of two subdivisions of input image in pixels

Specify the overlap (in pixels) of two subdivisions of the input image. The default value of this property is `[0 0]`.

MaximumDisplacement

Maximum displacement to search in pixels

Specify the maximum number of pixels that any center pixel in a block of pixels might move, from image to image or from frame to frame. The block matcher object uses this property to determine the size of the search region. The default value of this property is `[7 7]`.

MatchCriteria

video.BlockMatcher class

Match criteria between blocks

Specify how the System object measures the similarity of the block of pixels between two frames or images as Mean square error (MSE), or Mean absolute difference (MAD). The default value for this property is Mean square error (MSE).

OutputValue

Desired form of motion output

Specify the desired form of motion output as Magnitude-squared, or Horizontal and vertical components in complex form. The default value for this property is Magnitude-squared.

Fixed-Point Properties

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as Same as input, or Custom. This property applies when you set the MatchCriteria property to Mean square error (MSE). The default value of this property is Custom.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled numeric type object with a Signedness of Auto. This property applies when you set the MatchCriteria property to Mean square error (MSE) and the ProductDataType property to Custom. The default value of this property is numeric type ([], 32, 0).

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as Custom.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. The default value of this property is `numericType([],32,0)`.

`OutputDataType`

Output word and fraction lengths

Specify the output fixed-point data type as `Custom`.

`CustomOutputDataType`

Output word and fraction lengths

Specify the output fixed-point type as an unscaled `numericType` object with a `Signedness` of `Auto`. The `numericType` object should be unsigned if the `OutputValue` property is `Magnitude-squared` and, signed if it is `Horizontal` and `vertical` components in complex form. The default value of this property is `numericType([],8)`.

`RoundingMethod`

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value of this property is `Floor`.

`OverflowAction`

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. The default value of this property is `Saturate`

video.BlockMatcher class

Methods

clone	Create block matcher object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Compute motion of input image

Examples

Estimate motion

```
img1 = im2double(rgb2gray(imread('onion.png')));
htran = video.GeometricTranslator('Offset', [5 5], ...
    'OutputSize', 'Same as input image');
hbm = video.BlockMatcher( ...
    'ReferenceFrameSource', 'Input port', 'BlockSize', [35 35]);
hbm.OutputValue = ...
    'Horizontal and vertical components in complex form';
hds = video.ShapeInserter('Shape', 'Lines', ...
    'BorderColor', 'Custom', ...
    'CustomBorderColor', 255);
img2 = step(htran, img1);
tmp = step(hbm, img1, img2);
% compute motion for the two images
[Y X] = meshgrid(1:35:size(img1, 2), 1:35:size(img1, 1));
lines = ...
    [X(:)';Y(:)'; X(:)'+imag(tmp(:))';Y(:)'+real(tmp(:))'];
% draw lines to indicate motion
out = step(hds, img1, lines);
imshow(uint8(out)); % show the motion
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Block Matching block reference page. The object properties correspond to the block parameters.

See Also

`video.OpticalFlow`

video.BlockMatcher.clone

Purpose Create block matcher object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a BlockMatcher System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.BlockMatcher.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the BlockMatcher System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.BlockMatcher.step

Purpose Compute motion of input image

Syntax
 $V = \text{step}(H, I)$
 $C = \text{step}(H, I)$
 $Y = \text{step}(H, I, IREF)$

Description $V = \text{step}(H, I)$ computes the motion of input image I from one video frame to another, and returns V as a matrix of velocity magnitudes.

$C = \text{step}(H, I)$ computes the motion of input image I from one video frame to another, and returns C as a complex matrix of horizontal and vertical components, when you set the `OutputValue` property to `Horizontal` and vertical components in complex form.

$Y = \text{step}(H, I, IREF)$ computes the motion between input image I and reference image $IREF$ when the `ReferenceFrameSource` property is `Input` port.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Trace object boundaries in binary images
Description	The boundary tracer object traces object boundaries in binary images.
Construction	<p>H = video.BoundaryTracer returns a System object, H, that traces the boundaries of objects in a binary image in which nonzero pixels belong to an object and zero-valued pixels constitute the background.</p> <p>H = video.BoundaryTracer('PropertyName',PropertyValue,...) returns an object, H, with each specified property set to the specified value.</p>
Properties	<p>Connectivity</p> <p>Which pixels are connected to each other</p> <p>Specify which pixels are connected to each other as 4 or 8. Set this property to 4 to connect a pixel to the pixels on the top, bottom, left, and right. Set this property to 8 to connect a pixel to the pixels on the top, bottom, left, right, and diagonally. The default value of this property is 8.</p> <p>InitialSearchDirection</p> <p>First search direction to find next boundary pixel</p> <p>Specify the first direction in which to look to find the next boundary pixel that is connected to the starting pixel. This property can be set to one of North, Northeast, East, Southeast, South, Southwest, West, Northwest when the Connectivity property is set to 8 and can be set to one of North, East, South, West when the Connectivity property is set to 4. The default value of this property is North.</p> <p>TraceDirection</p> <p>Direction in which to trace the boundary</p> <p>Specify the direction in which to trace the boundary as Clockwise or Counterclockwise. The default value of this property is Clockwise.</p>

video.BoundaryTracer class

MaximumPixelCount

Maximum number of boundary pixels

Specify the maximum number of boundary pixels as a scalar integer greater than 1. The object uses this value to preallocate the number of rows of the output matrix Y so that it can hold all the boundary pixel location values. The default value of this property is 500.

PixelCountOutputPort

Enable output of actual number of boundary pixels

Set this property to true to output a vector where each element represents the actual number of boundary pixels found for each starting point. The default value of this property is false.

NoBoundaryAction

How to fill empty spaces in output matrix

Specify how to fill the empty spaces in the output matrix Y as None, Fill with last point found, or Fill with user-defined values. If you set this property to None, the object takes no action. So, any element that does not contain a boundary pixel location will not have a meaningful value. If you set this property to Fill with last point found, the object fills the remaining elements with the position of the last boundary pixel. If you set this property to Fill with user-defined values, you must specify the values in the FillValues property. The default value of this property is None.

FillValues

Value to fill in remaining empty elements in output matrix

Set this property to a scalar value or two-element vector to fill in the remaining empty elements in the output matrix Y. This property applies when you set the NoBoundaryAction property to Fill with user-defined values. The default value of this property is [0 0].

Methods

clone	Create boundary tracer object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Trace boundaries of objects in binary image

Examples

Trace boundaries around objects in an image

```
hautoth = video.Autothresher;
hboundtrace = video.BoundaryTracer;
% Read in the image
x = imread('coins.png');
% Use autothresholding to binarize the image.
bw = step(hautoth,x);
% Derive the start points
[row, col]= find(bw,1);
startpts = [row-2;col];
% Determine the boundaries
y = step(hboundtrace,bw,startpts);
y(y == 0) = [];
% Display the results
figure, imshow(bw);
hold('on');
plot(y(2:2:end,:)+1,y(1:2:end, :)+1,'r','Linewidth',2);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Trace Boundaries block reference page. The object properties correspond to the block parameters.

video.BoundaryTracer class

See Also

`video.EdgeDetector` | `video.ConnectedComponentLabeler` |
`video.Autothresher`

Purpose

Create boundary tracer object with same property values

Syntax

`C = clone(H)`

Description

`C = clone(H)` creates a `BoundaryTracer System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.BoundaryTracer.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.BoundaryTracer.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.BoundaryTracer.isLocked

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the BoundaryTracer System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

Purpose Trace boundaries of objects in binary image

Syntax $Y = \text{step}(H, X, \text{STARTPTS})$
 $[Y, \text{CNT}] = \text{step}(H, \dots)$

Description $Y = \text{step}(H, X, \text{STARTPTS})$ traces the boundaries of objects in the binary image X . The starting points for searching the boundary points is specified by the second 2-by- N element input matrix STARTPTS . Each column specifies the zero-based row and column coordinates of the initial point on the object boundary, and N represents the number of objects. The output Y is a $2M$ -by- N matrix, where each column contains the zero-based row and column coordinates of the boundary pixels. M represents the maximum number of boundary pixels for each object, as specified by the `MaximumPixelCount` property.

$[Y, \text{CNT}] = \text{step}(H, \dots)$ outputs CNT , a 1-by- N vector which indicates the number of boundary points for every boundary specified by the starting points. This applies when you set the `PixelCountOutputPort` property to `true`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.ChromaResampler class

Purpose	Downsample or upsample chrominance components of images
Description	The ChromaResampler object downsamples or upsample chrominance components of images.
Construction	<p>H = video.ChromaResampler returns a chroma resampling System object, H, that downsamples or upsamples chroma components of a YCbCr signal to reduce the bandwidth and/or storage requirements.</p> <p>H = video.ChromaResampler('PropertyName',PropertyValue,...) returns a chroma resampling System object, H, with each specified property set to the specified value.</p>
Properties	<p>Resampling</p> <p>Resampling format</p> <p>Specify the resampling format as follows:</p> <ul style="list-style-type: none">• To downsample the chrominance components of images, set this property to one of the following:<ul style="list-style-type: none">[4:4:4 to 4:2:2][4:4:4 to 4:2:0 (MPEG1)][4:4:4 to 4:2:0 (MPEG2)][4:4:4 to 4:1:1][4:2:2 to 4:2:0 (MPEG1)][4:2:2 to 4:2:0 (MPEG2)]• To upsample the chrominance components of images, set this property to one of the following:<ul style="list-style-type: none">[4:2:2 to 4:4:4][4:2:0 (MPEG1) to 4:4:4][4:2:0 (MPEG2) to 4:4:4][4:1:1 to 4:4:4][4:2:0 (MPEG1) to 4:2:2][4:2:0 (MPEG2) to 4:2:2]

The default value of this property is [4:4:4 to 4:2:2]

InterpolationFilter

Method used to approximate missing values

Specify the interpolation method used to approximate the missing chrominance values as `Pixel replication`, `Linear`. If this property is set to `Linear`, the System object uses linear interpolation to calculate the missing values. If this property is set to `Pixel replication`, the System object replicates the chrominance values of the neighboring pixels to create the upsampled image. This property applies when you upsample the chrominance values. The default value of this property is `Linear`.

AntialiasingFilterSource

Lowpass filter used to prevent aliasing

Specify the lowpass filter used to prevent aliasing as `Auto`, `Property`, or `None`. If this property is set to `Auto`, the System object uses a built-in lowpass filter. If this property is set to `Property`, the coefficients of the filters are specified by the properties `HorizontalFilterCoefficients` and/or `VerticalFilterCoefficients`. If this property is set to `None`, the System object does not filter the input signal. This property applies when you downsample the chrominance values. The default value of this property is `Auto`.

HorizontalFilterCoefficients

Horizontal filter coefficients

Specify the filter coefficients to apply to the input signal. This property applies when you set the `Resampling` property to one of: [4:4:4 to 4:2:2], [4:4:4 to 4:2:0 (MPEG1)], [4:4:4 to 4:2:0 (MPEG2)], or [4:4:4 to 4:1:1] and the `AntialiasingFilterSource` property to `Property`. The default value of this property is [0.2 0.6 0.2].

VerticalFilterCoefficients

video.ChromaResampler class

Vertical filter coefficients

Specify the filter coefficients to apply to the input signal. This property applies when you set the Resampling property to one of [4:4:4 to 4:2:0 (MPEG1)], [4:4:4 to 4:2:0 (MPEG2)], [4:2:2 to 4:2:0 (MPEG1)], [4:2:2 to 4:2:0 (MPEG2)] and the AntialiasingFilterSource property to Property. The default value of this property is [0.5 0.5].

TransposedInput

Input is row-major format

Set this property to true when the input contains data elements from the first row first, then data elements from the second row second, and so on through the last row. Otherwise, the System object assumes that the input data is stored in column-major format. The default value of this property is false.

Methods

clone	Create chroma resampling object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Resample input chrominance components

Examples

Resample the chrominance components of an image.

```
H = video.ChromaResampler;  
hcsc = video.ColorSpaceConverter;  
x = imread('peppers.png');
```

```
x1 = step(hcsc, x);  
[Cb, Cr] = step(H, x1(:,:,2), x1(:,:,3));
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Chroma Resampling block reference page. The object properties correspond to the block parameters.

See Also

`video.ColorSpaceConverter`

video.ChromaResampler.clone

Purpose Create chroma resampling object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a ChromaResampler System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.ChromaResampler.getNumInputs

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.ChromaResampler.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the ChromaResampler System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.ChromaResampler.step

Purpose Resample input chrominance components

Syntax `[Cb1,Cr1] = step(H,Cb,Cr)`

Description `[Cb1,Cr1] = step(H,Cb,Cr)` resamples the input chrominance components `Cb` and `Cr` and returns `Cb1` and `Cr1` as the resampled outputs.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.ColorSpaceConverter class

Purpose	Convert color information between color spaces
Description	The ColorSpaceConverter object converts color information between color spaces.
Construction	<p>H = video.ColorSpaceConverter returns a System object, H, that converts color information from RGB to YCbCr using conversion standard Rec. 601 (SDTV).</p> <p>H = video.ColorSpaceConverter('PropertyName',PropertyValue,...) returns a color space conversion object, H, with each specified property set to the specified value.</p>
Properties	<p>Conversion</p> <p>Color space input/output conversion</p> <p>Specify the color spaces to convert between as the following:</p> <ul style="list-style-type: none">[RGB to YCbCr][YCbCr to RGB][RGB to intensity][RGB to HSV][HSV to RGB][sRGB to XYZ][XYZ to sRGB][sRGB to L*a*b*][L*a*b* to sRGB] <p>Note that the R, G, B and Y (luma) signal components in the above color space conversions are gamma corrected. The default value for this property is [RGB to YCbCr].</p> <p>WhitePoint</p> <p>Reference white point</p> <p>Specify the reference white point as D50, D55, or D65. This property applies when you set the Conversion property to [sRGB</p>

video.ColorSpaceConverter class

to L*a*b*] or [L*a*b* to sRGB]. The default value for this property is D65.

ConversionStandard

Standard for RGB to YCbCr conversion

Specify the standard used to convert the values between the RGB and YCbCr color spaces as Rec. 601 (SDTV), or Rec. 709 (HDTV). This property applies when you set the Conversion property to [RGB to YCbCr] or [YCbCr to RGB]. The default value for this property is Rec. 601 (SDTV).

ScanningStandard

Scanning standard for RGB to YCbCr conversion

Specify the scanning standard used to convert the values between the RGB and YCbCr color spaces as [1125/60/2:1], or [1250/50/2:1]. This property applies when you set the ConversionStandard property to Rec. 709 (HDTV). The default value for this property is [1125/60/2:1].

Methods

clone	Create color space converter object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Convert color space of input image

Examples

Convert an image from an RGB to an intensity color space.

```
i1 = imread('pears.png');
imshow(i1);
hcsc = video.ColorSpaceConverter;
hcsc.Conversion = 'RGB to intensity';
i2 = step(hcsc, i1);
pause(2);
imshow(i2);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Color Space Conversion block reference page. The object properties correspond to the block parameters, except for:

- The **Image signal** block parameter allows you to specify whether the block accepts the color video signal as **One multidimensional signal** or **Separate color signals**. The object does not have a property that corresponds to the **Image signal** block parameter. You must always provide the input image to the `step` method of the object as a single multidimensional signal.

See Also

`video.Autothresher`

video.ColorSpaceConverter.clone

Purpose	Create color space converter object with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates a ColorSpaceConverter System object <code>C</code> , with the same property values as <code>H</code> . The <code>clone</code> method creates a new unlocked object.

video.ColorSpaceConverter.getNumInputs

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.ColorSpaceConverter.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the ColorSpaceConverter System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.ColorSpaceConverter.step

Purpose Convert color space of input image

Syntax $C2 = \text{step}(H, C1)$

Description $C2 = \text{step}(H, C1)$ converts a multidimensional input image $C1$ to a multidimensional output image $C2$. $C1$ and $C2$ are images in different color spaces.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.ConnectedComponentLabeler class

Purpose

Label and count the connected regions in a binary image

Description

The `ConnectedComponentLabeler` object labels and counts the connected regions in a binary image. The `System` object can output a label matrix, where pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on. The object can also output a scalar that represents the number of labeled objects.

Construction

`H = video.ConnectedComponentLabeler` returns a `System` object, `H`, that labels and counts connected regions in a binary image.

`H = video.ConnectedComponentLabeler('PropertyName',PropertyValue,...)` returns a label `System` object, `H`, with each property set to the specified value.

Properties

`Connectivity`

Which pixels are connected to each other

Specify which pixels are connected to each other as either 4 or 8. If a pixel should be connected to the pixels on the top, bottom, left, and right, set this property to 4. If a pixel should be connected to the pixels on the top, bottom, left, right, and diagonally, set this property to 8. The default value of this property is 8.

`LabelMatrixOutputPort`

Enable output of label matrix

Set to `true` to output the label matrix. Both the `LabelMatrixOutputPort` and `LabelCountOutputPort` properties cannot be set to `false` at the same time. The default value of this property is `true`.

`LabelCountOutputPort`

Enable output of number of labels

video.ConnectedComponentLabeler class

Set to true to output the number of labels. Both the `LabelMatrixOutputPort` and `LabelCountOutputPort` properties cannot be set to false at the same time. The default value of this property is true.

OutputDataType

Output data type

Set the data type of the output to one of `Automatic`, `uint32`, `uint16`, `uint8`. If this property is set to `Automatic`, the System object determines the appropriate data type for the output. If it is set to `uint32`, `uint16`, or `uint8`, the data type of the output is 32-, 16-, or 8-bit unsigned integers, respectively. The default value for this property is `Automatic`.

OverflowAction

Behavior if number of found objects exceeds data type size of output

Specify the System object's behavior if the number of found objects exceeds the maximum number that can be represented by the output data type as `Use maximum value of the output data type`, or `Use zero`. If this property is set to `Use maximum value of the output data type`, the remaining regions are labeled with the maximum value of the output data type. If this property is set to `Use zero`, the remaining regions are labeled with zeroes. This property applies when you set the `OutputDataType` property to `uint16` or `uint8`. The default value for this property is `Use maximum value of the output data type`.

Methods

<code>clone</code>	Create connected component labeler object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method

video.ConnectedComponentLabeler class

getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Label and count connected regions in input

Examples

Label connected regions in an image.

```
img = logical([0 0 0 0 0 0 0 0 0 0 0 0 0; ...
0 1 1 1 1 0 0 0 0 0 0 1 0; ...
0 1 1 1 1 1 0 0 0 0 1 1 0; ...
0 1 1 1 1 1 0 0 0 1 1 1 0; ...
0 1 1 1 1 0 0 0 1 1 1 1 0; ...
0 0 0 0 0 0 0 1 1 1 1 1 0; ...
0 0 0 0 0 0 0 0 0 0 0 0 0])
hlabel = video.ConnectedComponentLabeler;
hlabel.LabelMatrixOutputPort = true;
hlabel.LabelCountOutputPort = false;
labeled = step(hlabel, img)
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Label block reference page. The object properties correspond to the block parameters, except for:

- The LabelCountOutputPort and LabelMatrixOutputPort object properties correspond to the **Output** block parameter.

See Also

video.AutoThresholder | video.BlobAnalysis

video.ConnectedComponentLabeler.clone

Purpose Create connected component labeler object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a `ConnectedComponentLabeler` System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.ConnectedComponentLabeler.getNumInputs

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.ConnectedComponentLabeler.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.ConnectedComponentLabeler.isLocked

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the ConnectedComponentLabeler System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.ConnectedComponentLabeler.step

Purpose Label and count connected regions in input

Syntax
L = step(H,BW)
COUNT = step(H,BW)
[L,COUNT] = step(H,BW)

Description L = step(H,BW) outputs the matrix, L for input binary image BW when the LabelMatrixOutputPort property is true.

COUNT = step(H,BW) outputs the number of distinct, connected regions found in input binary image BW when you set the LabelMatrixOutputPort property to false and LabelCountOutputPort property to true.

[L,COUNT] = step(H,BW) outputs both the L matrix and number of distinct, connected regions, COUNT when you set both the LabelMatrixOutputPort property and LabelCountOutputPort to true.

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Adjust image contrast by linear scaling
Description	The ContrastAdjuster object adjusts image contrast by linearly scaling pixel values between upper and lower limits. Pixel values that are above or below this range are saturated to the upper or lower limit values.
Construction	<p>H = video.ContrastAdjuster returns a contrast adjustment object, H, that adjusts the contrast of an image by linearly scaling the pixel values between the maximum and minimum values of the input data.</p> <p>H = video.ContrastAdjuster('PropertyName', PropertyValue, ...) returns a contrast adjustment object, H, with each property set to the specified value.</p>
Properties	<p>InputRange</p> <p>How to specify lower and upper input limits</p> <p>Specify how to determine the lower and upper input limits as Full input data range [min max], Custom, or Range determined by saturating outlier pixels. The default value for this property is Full input data range [min max].</p> <p>CustomInputRange</p> <p>Lower and upper input limits</p> <p>Specify the lower and upper input limits as a two-element vector of real numbers, where the first element corresponds to the lower input limit, and the second element corresponds to the upper input limit. This property applies only when you set the InputRange property to Custom. This property is tunable.</p> <p>PixelSaturationPercentage</p> <p>Percentage of pixels to consider outliers</p> <p>Specify the percentage of pixels to consider outliers, as a two-element vector. The contrast adjustment object calculates the lower input limit such that the percentage of pixels with values smaller than the lower limit is at most the value of the</p>

video.ContrastAdjuster class

first element. Similarly, the object calculates the upper input limit such that the percentage of pixels with values greater than the upper limit is at least the value of the second element. This property only applies when you set the `InputRange` property to `Range` determined by saturating outlier pixels. The default value of this property is `[1 1]`.

HistogramNumBins

Number of histogram bins

Specify the number of histogram bins used to calculate the scaled input values. The default value of this property is `256`.

OutputRangeSource

How to specify lower and upper output limits

Specify how to determine the lower and upper output limits as `Auto` or `Property`. If you set the value of this property to `Auto`, the object uses the minimum value of the input data type as the lower output limit and the maximum value of the input data type as the upper output limit. The default value for this property is `Auto`.

OutputRange

Lower and upper output limits

Specify the lower and upper output limits as a two-element vector of real numbers, where the first element corresponds to the lower output limit and the second element corresponds to the upper output limit. This property only applies when you set the `OutputRangeSource` property to `Property`. This property is tunable.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value for this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. The default value of this property is `Wrap`

ProductInputDataType

Product input word and fraction lengths

Specify the product input fixed-point data type as `Custom`.

CustomProductInputDataType

Product input word and fraction lengths

Specify the product input fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. The default value of this property is `numericType([],32,16)`.

ProductHistogramDataType

Product histogram word and fraction lengths

Specify the product histogram fixed-point data type as `Custom`. This property only applies when you set the `InputRange` property to `Range` determined by saturating outlier pixels.

CustomProductHistogramDataType

Product histogram word and fraction lengths

Specify the product histogram fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property only applies when you set the `InputRange` property to `Range` determined by saturating outlier pixels. The default value of this property is `numericType([],32,16)`.

video.ContrastAdjuster class

Methods

<code>clone</code>	Create contrast adjuster with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Adjust contrast in input image

Examples

Use contrast adjuster to enhance image quality:

```
hcontadj = video.ContrastAdjuster;  
x = imread('pout.tif');  
y = step(hcontadj, x);  
imshow(x); title('Original Image');  
figure, imshow(y);  
title('Enhanced image after contrast adjustment');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Contrast Adjustment block reference page. The object properties correspond to the block parameters.

See Also

`video.HistogramEqualizer`

Purpose Create contrast adjuster with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an instance of the current contrast adjuster object with the same property values. The `clone` method creates a new unlocked object

video.ContrastAdjuster.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.ContrastAdjuster.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.ContrastAdjuster.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the ContrastAdjuster System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Adjust contrast in input image

Syntax $Y = \text{step}(H, X)$

Description $Y = \text{step}(H, X)$ performs contrast adjustment of input X and returns the adjusted image Y .

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.Convolver2D class

Purpose	Compute 2-D discrete convolution of two input matrices
Description	The Convolver2D object computes 2-D discrete convolution of two input matrices.
Construction	<p><code>H = video.Convolver2D</code> returns a System object, H, that performs two-dimensional convolution on two inputs.</p> <p><code>H = video.Convolver2D('PropertyName',PropertyValue,...)</code> returns a 2-D convolution System object, H, with each specified property set to the specified value.</p>
Properties	<p>OutputSize</p> <p>Specify dimensions of output</p> <p>This property controls the size of the output scalar, vector, or matrix produced as a result of the convolution between the two inputs. This property can be set to one of <code>Full</code>, <code>Same as first input</code>, or <code>Valid</code>. If this property is set to <code>Full</code>, the output is the full two-dimensional convolution with $(Ma+Mb-1, Na+Nb-1)$. If you set this property to <code>Same as first input</code>, the output is the central part of the convolution with the same dimensions as the first input. If you set this property to <code>Valid</code>, the output consists of only those parts of the convolution that are computed without the zero-padded edges of any input. This output has dimensions $(Ma-Mb+1, Na-Nb+1)$. (Ma, Na) is the size of the first input matrix and (Mb, Nb) is the size of the second input matrix. The default value of this property is <code>Full</code>.</p> <p>Normalize</p> <p>Whether to normalize the output</p> <p>Set to true to normalize the output. The default value of this property is <code>false</code>.</p>

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value of this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. The default value of this property is `Wrap`.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as `Same as first input`, or `Custom`. The default value of this property is `Custom`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `ProductDataType` property to `Custom`. The default value of this property is `numericType([], 32, 10)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as product`, `Same as first input`, `Custom`. The default value of this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

video.Convolver2D class

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType([],32,10)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Same` as first input, or `Custom`. The default value of this property is `Custom`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Custom`. The default value of this property is `numericType([],32,12)`.

Methods

<code>clone</code>	Create 2-D convolver object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Compute 2-D convolution of input matrices

Examples

Compute the 2D convolution of two matrices.

```
hconv2d = video.Convolver2D;  
x1 = [1 2;2 1];
```

```
x2 = [1 -1;-1 1];  
y = step(hconv2d, x1, x2)
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the 2-D Convolution block reference page. The object properties correspond to the block parameters.

See Also

[video.Crosscorrelator2D](#) | [video.AutoCorrelator2D](#)

video.Convolver2D.clone

Purpose	Create 2-D convolver object with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates a Convolver2D System object C, with the same property values as H. The <code>clone</code> method creates a new unlocked object.

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.Convolver2D.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the Convolver2D System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.Convolver2D.step

Purpose Compute 2-D convolution of input matrices

Syntax `Y = step(HCONV2D,X1,X2)`

Description `Y = step(HCONV2D,X1,X2)` computes 2-D convolution of input matrices `X1` and `X2`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Corner metric matrix and corner detector
Description	The CornerDetector object calculates a corner metric matrix and detects corners in images.
Construction	<p>H = video.CornerDetector returns a corner detector object, H, that finds corners in an image based on pixels with the largest corner metric values.</p> <p>H = video.CornerDetector('PropertyName',PropertyValue, ...) returns a corner detector object, H, with each property set to the specified value.</p>
Properties	<p>Method</p> <p>Method to find corner values</p> <p>Specify the method to find the corner values as Harris corner detection (Harris & Stephens), Minimum eigenvalue (Shi & Tomasi), or Local intensity comparison (Rosen & Drummond). The default value for this property is Harris corner detection (Harris & Stephens).</p> <p>Sensitivity</p> <p>Sensitivity factor to detect sharp corners</p> <p>Specify the sensitivity factor, k, used in the Harris corner detection algorithm as a real number such that $0 < k < 1$. The smaller the value of k, the more likely the algorithm detects sharp corners. This property only applies when you set the Method property to Harris corner detection (Harris & Stephens). The default value of this property is 0.04. This property is tunable.</p> <p>SmoothingFilterCoefficients</p> <p>Smooth filter coefficients</p> <p>Specify the filter coefficients for the separable smoothing filter as a real-valued vector. For more information, see fspecial. This property applies only when you set the Method property to either</p>

video.CornerDetector class

Harris corner detection (Harris & Stephens), or Minimum eigenvalue (Shi & Tomasi). The default value of this property is the output of `fspecial('gaussian', 1 5, 1.5)`.

IntensityThreshold

Intensity comparison threshold

Specify the intensity threshold value used to find valid bright or dark surrounding pixels as a positive real number. This property applies only when you set the `Method` property to `Local intensity comparison` (Rosen & Drummond). The default value of this property is 0.1. This property is tunable.

MaximumAngleThreshold

Maximum valid corner angle in degrees

Specify the maximum angle in degrees considered a corner as 22.5, 45.0, 67.5, 90.0, 112.5, 135.0, or 157.5. This property only applies when you set the `Method` property to `Local intensity comparison` (Rosen & Drummond). The default value of this property is 157.5. This property is tunable.

CornerLocationOutputPort

Enables output of the corner location

Set this property to `true` to output the corner location. This property and the `MetricMatrixOutputPort` property cannot both be set to `false`. The default value for this property is `true`.

MetricMatrixOutputPort

Enables output of the corner metric matrix

Set this property to `true` to output the corner metric matrix. This property and the `CornerLocationOutputPort` property cannot both be set to `false`. The default value for this property is `false`.

MaximumCornerCount

Maximum number of corners to detect

Specify the maximum number of corners to detect as a positive integer. This property only applies when you set the `CornerLocationOutputPort` property to true. The default value of this property is 20.

CornerThreshold

Minimum metric value that indicates a corner

Specify the minimum metric value that indicates a corner as a positive real number. This property applies only when you set the `CornerLocationOutputPort` property to true. The default value of this property is 0.0005. This property is tunable.

NeighborhoodSize

Size of suppressed region around detected corner

Specify the size of the neighborhood around the corner metric value over which the object zeros out the values. The neighborhood size is a two element vector of positive odd integers, $[r \ c]$. Here r is the number of rows in the neighborhood and c is the number of columns. This property only applies when you set the `CornerLocationOutputPort` property to true. The default value of this property is $[11 \ 11]$.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value of this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap`, or `Saturate`. The default value of this property is `Wrap`.

video.CornerDetector class

CoefficientsDataType

Coefficients word and fraction lengths

Specify the coefficients fixed-point data type as `Same` word length as `input`, or `Custom`. This property applies only when the `Method` property is not `Local intensity comparison` (Rosen & Drummond). The default value of this property is `Custom`.

CustomCoefficientsDataType

Coefficients word and fraction lengths

Specify the coefficients fixed-point type as a signed `numericType` object with a `Signedness` of `Auto`. This property applies only when the `Method` property is not `Local intensity comparison` (Rosen & Drummond) and the `CoefficientsDataType` property is `Custom`. The default value of this property is `numericType([],16)`.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as `Same` as `input`, or `Custom`. This property is applicable when the `Method` property is not `Local intensity comparison` (Rosen & Drummond). The default value of this property is `Custom`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when the `Method` property is not `Local intensity comparison` (Rosen & Drummond) and the `ProductDataType` property is `Custom`. The default value of this property is `numericType([],32,0)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same` as `input`, or `Custom`. The default value of this property is `Custom`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object. This property applies only when the `AccumulatorDataType` property is `Custom`. The default value of this property is `numericType([],32,0)`.

MemoryDataType

Memory word and fraction lengths

Specify the memory fixed-point data type as `Same as input`, `Custom`. This property applies when the `Method` property is not `Local intensity comparison (Rosen & Drummond)`. The default value of this property is `Custom`.

CustomMemoryDataType

Memory word and fraction lengths

Specify the memory fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies only when the `Method` property is not `Local intensity comparison (Rosen & Drummond)` and the `MemoryDataType` property is `Custom`. The default value of this property is `numericType([],32,0)`.

MetricOutputDataType

Metric output word and fraction lengths

Specify the metric output fixed-point data type as `Same as accumulator`, `Same as input`, or `Custom`. The default value of this property is `Same as accumulator`.

CustomMetricOutputDataType

Metric output word and fraction lengths

Specify the metric output fixed-point type as a signed, scaled `numericType` object with a `Signedness` of `Auto`. This property applies only when the `MetricOutputDataType` property is `Custom`. The default value of this property is `numericType([],32,0)`.

video.CornerDetector class

Methods

clone	Create corner detector with same property values
getNumInputs	Return number of expected inputs to step method
getNumOutputs	Return number of outputs of step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Detect corners in input image

Examples

Detect corners in an input image:

```
x = im2single(imread('circbw.tif'));
hcornerdet = video.CornerDetector( ...
    'Method', ...,
    'Local intensity comparison (Rosen & Drummond)', ...
    'MaximumAngleThreshold', '135.0', ...
    'MaximumCornerCount', 200);
[pts, cnt] = step(hcornerdet, x);
hdrawmarkers = video.MarkerInserter( ...
    'Shape', 'Circle', ...
    'BorderColor', 'Custom', ...
    'CustomBorderColor', [1 0 0]);
y = step(hdrawmarkers, cat(3,x,x,x), pts);
imshow(y); title ('Corners detected in a binary image');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the [Corner Detection](#) block reference page. The object properties correspond to the block parameters, except for:

- The `CornerLocationOutput` and `MetricMatrixOutputPort` are logical properties for the object. These properties correspond to the **Output** block parameter.

See Also

`video.LocalMaximaFinder` | `video.EdgeDetector` |
`video.MarkerInserter`

video.CornerDetector.clone

Purpose	Create corner detector with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates a corner detector object, <code>C</code> , with the same property values. The <code>clone</code> method creates a new unlocked object.

Purpose

Return number of expected inputs to step method

Syntax

`getNumInputs(H)`

Description

`getNumInputs(H)` returns the number of expected inputs to the `step` method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.CornerDetector.getNumOutputs

Purpose Return number of outputs of step method

Syntax `getNumOutputs(H)`

Description `getNumOutputs(H)` returns the number of outputs from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	<code>isLocked(H)</code>
Description	<p><code>isLocked(H)</code> returns the locked state of the corner detector.</p> <p>The <code>isLocked</code> method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the <code>step</code> method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the <code>isLocked</code> method returns a <code>true</code> value.</p>

video.CornerDetector.step

Purpose Detect corners in input image

Syntax
[LOC, CNT] = step(H,I)
METRIC = step(H,I)
[LOC,CNT,METRIC] = step(H,I)

Description [LOC, CNT] = step(H,I) finds the corners in input image I. LOC is a 2-by-*N* matrix that represents the locations of the corners, where *N* is the maximum number of corners in image I specified by the MaximumCornerCount property. CNT represents the number of detected corners in the image.

METRIC = step(H,I) returns a matrix with corner metric values, METRIC, when the MetricMatrixOutputPort property is true. The size of metric matrix is the same as that of the input image.

[LOC,CNT,METRIC] = step(H,I) returns the locations of the corners in LOC, the number of detected corners in CNT, and the corner metric matrix in METRIC, when both the CornerLocationOutputPort and MetricMatrixOutputPort properties are true.

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Compute 2-D cross-correlation of two input matrices
Description	The Crosscorrelator2D object computes 2-D cross-correlation of two input matrices.
Construction	<p>H = video.Crosscorrelator2D returns a System object, H, that performs two-dimensional cross-correlation between two inputs.</p> <p>H = video.Crosscorrelator2D('PropertyName',PropertyValue,...) returns a 2-D cross correlation System object, H, with each specified property set to the specified value.</p>
Properties	<p>OutputSize</p> <p>Specify dimensions of output</p> <p>This property controls the size of the output scalar, vector, or matrix produced as a result of the cross-correlation between the two inputs. This property can be set to one of <code>Full</code>, <code>Same as first input</code>, <code>Valid</code>. If this property is set to <code>Full</code>, the output is the full two-dimensional cross-correlation with dimensions $(Ma+Mb-1, Na+Nb-1)$. if this property is set to <code>same as first input</code>, the output is the central part of the cross-correlation with the same dimensions as the first input. if this property is set to <code>valid</code>, the output is only those parts of the cross-correlation that are computed without the zero-padded edges of any input. this output has dimensions $(Ma-Mb+1, Na-Nb+1)$. (Ma, Na) is the size of the first input matrix and (Mb, Nb) is the size of the second input matrix. The default value for this property is <code>Full</code>.</p> <p>Normalize</p> <p>Normalize output</p> <p>Set this property to <code>true</code> to normalize the output. If you set this property to <code>true</code>, the object divides the output by</p> $\sqrt{\sum(I_{1p} \cdot I_{1p}) \times \sum(I_2 \cdot I_2)},$ where I_{1p} is the portion of the input

video.Crosscorrelator2D class

matrix, I_1 that aligns with the input matrix, I_2 . This property must be set to `false` for fixed-point inputs. The default value of this property is `false`.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value of this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. The default value of this property is `Wrap`.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as `Same as first input`, `Custom`. The default value of this property is `Same as first input`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `ProductDataType` property to `Custom`. The default value of this property is `numericType([], 32, 30)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as product`, `Same as first input`, `Custom`. The default value for this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType([], 32, 30)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Same as first input`, `Custom`. The default value of this property is `Same as first input`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Custom`. The default value of this property is `numericType([], 16, 15)`.

Methods

<code>clone</code>	Create 2-D cross correlator object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method

video.Crosscorrelator2D class

isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Compute 2-D correlation of input matrices

Examples

Compute the 2-D correlation of two matrices.

```
hcorr2d = video.Crosscorrelator2D;  
x1 = [1 2;2 1];  
x2 = [1 -1;-1 1];  
y = step(hcorr2d,x1,x2);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the 2-D Correlation block reference page. The object properties correspond to the block parameters.

See Also

[video.Autocorrelator2D](#) | [signalblks.Crosscorrelator](#)

Purpose Create 2-D cross correlator object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a `Crosscorrelator2D` System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.Crosscorrelator2D.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, `N` to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.Crosscorrelator2D.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.Crosscorrelator2D.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the Crosscorrelator2D System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Compute 2-D correlation of input matrices

Syntax $Y = \text{step}(H, X1, X2)$

Description $Y = \text{step}(H, X1, X2)$ computes 2D correlation of input matrices $X1$ and $X2$.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.DCT2D class

Purpose	Compute 2-D discrete cosine transform
Description	The DCT2D object computes a 2-D discrete cosine transform. The number of rows and columns of the input matrix must be a power of 2.
Construction	<p>H = video.DCT2D returns a discrete cosine transform System object, H, used to compute the two-dimensional discrete cosine transform (2-D DCT) of a real input signal.</p> <p>H = video.DCT2D('PropertyName',PropertyValue,...) returns a discrete cosine transform System object, H, with each specified property set to the specified value.</p>
Properties	<p>SineComputation</p> <p>Specify how the System object computes sines and cosines as Trigonometric function, or Table lookup. This property must be set to Table lookup for fixed-point inputs.</p> <p>Fixed-Point Properties</p> <p>RoundingMethod</p> <p>Rounding method for fixed-point operations</p> <p>Specify the rounding method as Ceiling, Convergent, Floor, Nearest, Round, Simplest, or Zero. This property applies when you set the SineComputation to Table lookup. The default value of this property is Floor.</p> <p>OverflowAction</p> <p>Overflow action for fixed-point operations</p> <p>Specify the overflow action as Wrap or Saturate. This property applies when you set the SineComputation to Table lookup. The default value of this property is Wrap.</p> <p>SineTableDataType</p> <p>Sine table word-length designation</p>

Specify the sine table fixed-point data type as `Same word length as input`, or `Custom`. This property applies when you set the `SineComputation` to `Table lookup`. The default value of this property is `Same word length as input`.

CustomSineTableDataType

Sine table word length

Specify the sine table fixed-point type as a signed, unscaled `numericType` object. This property applies when you set the `SineComputation` to `Table lookup` and you set the `SineTableDataType` property to `Custom`. The default value of this property is `numericType(true,16)`.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as `Internal rule`, `Same as first input`, or `Custom`. This property applies when you set the `SineComputation` to `Table lookup`. The default value of this property is `Custom`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a signed, scaled `numericType` object. This property applies when you set the `SineComputation` to `Table lookup`, and the `ProductDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Internal rule`, `Same as input`, `Same as product`, `Same as first input`, or `Custom`. This property applies when you set the `SineComputation` property to `Table lookup`. The default value of this property is `Internal rule`.

video.DCT2D class

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a signed, scaled `numericType` object. This property applies when you set the `SineComputation` to `Table lookup`, and `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Internal rule`, `Same as first input`, or `Custom`. This property applies when you set the `SineComputation` to `Table lookup`. The default value of this property is `Custom`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a signed, scaled `numericType` object. This property applies when you set the `SineComputation` to `Table lookup`, and the `OutputDataType` property to `Custom`. The default value of this property is `numericType(true,16,15)`.

Methods

<code>clone</code>	Create 2-D discrete cosine transform object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method

isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Compute 2-D discrete cosine transform on input

Examples

Use 2-D discrete cosine transform to analyze the energy content in an image. Set the DCT coefficients lower than a threshold of 0, and then reconstruct the image using the 2-D inverse discrete cosine transform object.

```
hdct2d = video.DCT2D;  
I = double(imread('cameraman.tif'));  
J = step(hdct2d, I);  
imshow(log(abs(J)),[]), colormap(jet(64)), colorbar  
  
hidct2d = video.IDCT2D;  
J(abs(J) < 10) = 0;  
It = step(hidct2d, J);  
figure, imshow(I, [0 255]), title('Original image')  
figure, imshow(It,[0 255]), title('Reconstructed image')
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the 2-D DCT block reference page. The object properties correspond to the block parameters.

See Also

[video.IDCT2D](#) | [signalblks.DCT](#) | [signalblks.IDCT](#)

video.DCT2D.clone

Purpose	Create 2-D discrete cosine transform object with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates a DCT2D System object <code>C</code> , with the same property values as <code>H</code> . The <code>clone</code> method creates a new unlocked object.

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.DCT2D.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the DCT2D System object.</p> <p>The <code>isLocked</code> method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the <code>step</code> method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the <code>isLocked</code> method returns a true value.</p>

video.DCT2D.step

Purpose	Compute 2-D discrete cosine transform on input
Syntax	$Y = \text{step}(H, X)$
Description	$Y = \text{step}(H, X)$ computes the 2-D DCT Y of input X .

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Remove motion artifacts by deinterlacing input video signal
Description	The <code>Deinterlacer</code> object removes motion artifacts by deinterlacing input video signal.
Construction	<p><code>H = video.Deinterlacer</code> returns a deinterlacing System object, <code>H</code>, that removes motion artifacts from images composed of weaved top and bottom fields of an interlaced signal.</p> <p><code>H = video.Deinterlacer('PropertyName',PropertyValue,...)</code> returns a deinterlacing System object, <code>H</code>, with each specified property set to the specified value.</p>
Properties	<p>Method</p> <p>Method used to deinterlace input video</p> <p>Specify how the object deinterlaces the input video as <code>Line repetition</code>, <code>Linear interpolation</code>, <code>Vertical temporal median filtering</code>. The default value for this property is <code>Line repetition</code>.</p> <p>TransposedInput</p> <p>Indicate if input data is in row-major order</p> <p>Set this property to <code>true</code> to assume that the input buffer contains data elements from the first row first, then data elements from the second row second, and so on through the last row. The default value of this property is <code>false</code>.</p> <p>Fixed-Point Properties</p> <p>RoundingMethod</p> <p>Rounding method for fixed-point operations</p> <p>Specify the rounding method as <code>Ceiling</code>, <code>Convergent</code>, <code>Floor</code>, <code>Nearest</code>, <code>Round</code>, <code>Simplest</code>, or <code>Zero</code>. This property applies when</p>

video.Deinterlacer class

you set the `Method` property to `Linear Interpolation`. The default value of this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. This property applies when you set the `Method` property to `Linear Interpolation`. The default value of this property is `Wrap`

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as input`, `Custom`. This property applies when you set the `Method` property to `Linear Interpolation`. The default value for this property is `Custom`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property is applicable when the `AccumulatorDataType` property is `Custom`. This property applies when you set the `Method` property to `Linear Interpolation`. The default value of this property is `numericType([],12,3)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Same as input`, `Custom`. This property applies when you set the `Method` property to `Linear Interpolation`. The default value of this property is `Same as input`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property is applicable when the `OutputDataType` property is `Custom`. This property applies when you set the `Method` property to `Linear Interpolation`. The default value of this property is `numericType([],8,0)`.

Methods

<code>clone</code>	Create deinterlacer object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Deinterlace input video signal

Examples

Use deinterlacing to remove motion artifacts from an input image.

```
hdint = video.Deinterlacer;  
x = imread('vipinterlace.png');  
y = step(hdint, x);  
imshow(x); title('Original Image');  
figure, imshow(y); title('Image after deinterlacing');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Deinterlacing block reference page. The object properties correspond to the block parameters.

See Also

`video.CornerDetector`

video.Deinterlacer.clone

Purpose Create deinterlacer object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a Deinterlacer System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.Deinterlacer.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the Deinterlacer System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.Deinterlacer.step

Purpose Deinterlace input video signal

Syntax $Y = \text{step}(H, X)$

Description $Y = \text{step}(H, X)$ deinterlaces input X according to the algorithm set in the Method property.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.DemosaicInterpolator class

Purpose

Demosaic Bayer's format images

Description

The `DemosaicInterpolator` object demosaics Bayer's format images. The alignment is identified as the sequence of R, G and B pixels in the top-left four pixels of the image in row-wise order.

Construction

`H = video.DemosaicInterpolator` returns a System object, H, that performs demosaic interpolation on an input image in Bayer format with the specified alignment.

H =

`video.DemosaicInterpolator('PropertyName',PropertyValue,...)` returns a System object, H, with each specified property set to the specified value.

Properties

Method

Interpolation algorithm

Specify the algorithm the object uses to calculate the missing color information as `Bilinear` or `Gradient-corrected linear`. The default value for this property is `Gradient-corrected linear`.

`SensorAlignment`

Alignment of the input image

Specify the sequence of R, G and B pixels that correspond to the 2-by-2 block of pixels in the top left corner of the image. It can be set to one of `RGGB`, `GRBG`, `GBRG`, or `BGGR`. The sequence should be specified in left-to-right, top-to-bottom order. The default value for this property is `RGGB`.

Fixed-Point Properties

`RoundingMethod`

Rounding method for fixed-point operations

video.DemosaicInterpolator class

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value for this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. The default value for this property is `Saturate`.

ProductDataType

Product output word and fraction lengths

Specify the product output fixed-point data type as `Same as input`, `Custom`. The default value for this property is `Custom`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `ProductDataType` property to `Custom`. The default value of this property is `numericType([],32,10)`.

AccumulatorDataType

Data type of the accumulator

Specify the accumulator fixed-point data type as `Same as product`, `Same as input`, or `Custom`. The default value for this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType([],32,10)`.

video.DemosaicInterpolator class

Methods

clone	Create demosaic interpolator object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Perform demosaic operation on input

Examples

Demosaic a Bayer pattern encoded-image photographed by a camera with a sensor alignment of 'BGGR'.

```
x = imread('mandi.tif');
hdemosaic = ...
    video.DemosaicInterpolator('SensorAlignment', 'BGGR');
y = step(hdemosaic, x);
imshow(x,'InitialMagnification',20);
title('Original Image');
figure, imshow(y,'InitialMagnification',20);
title('RGB image after demosaic');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Demosaic block reference page. The object properties correspond to the block parameters, , except for:

- The **Output image signal** block parameter allows you to specify whether the block outputs the image as **One multidimensional signal** or **Separate color signals**. The object does not have a property that corresponds to the **Output image signal** block

video.DemosaicInterpolator class

parameter. The object always outputs the image as an M -by- N -by- P color video signal.

See Also

`video.GammaCorrector`

Purpose Create demosaic interpolator object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a DemosaicInterpolator System object C, with the same property values as H. The `clone` method creates a new unlocked object.

video.DemosaicInterpolator.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.DemosaicInterpolator.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The getNumOutputs method returns a positive integer representing the number of outputs from the step method. This value will change if any properties that turn inputs on or off are changed.

video.DemosaicInterpolator.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the DemosaicInterpolator System object.

The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.

Purpose Perform demosaic operation on input

Syntax $Y = \text{step}(H, X)$

Description $Y = \text{step}(H, X)$ performs the demosaic operation on the input X to produce an M -by- N -by- P color video signal where P is the number of color planes.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.DeployableVideoPlayer class

Purpose Send video data to computer screen

Description The DeployableVideoPlayer object sends video data to computer screen.

Construction H = video.DeployableVideoPlayer returns a deployable video player System object, H, that sends video data to a computer screen.

H =
video.DeployableVideoPlayer('PropertyName',PropertyValue,...)
returns a deployable video player System object, H, with each specified property set to the specified value.

H =
video.DeployableVideoPlayer(FRAMERATE,'PropertyName',PropertyValue,...)
returns a deployable video player System object, H, with the FrameRate property set to FRAMERATE and other specified properties set to the specified values.

Properties

WindowLocation

Location of top left corner of video window

Specify the location of top left corner of video player window as a two-element vector where the first and second elements represent the vertical and horizontal positions respectively. [0 0] represents the top left corner of the window. The default value of this property is [0 0].

WindowCaption

Caption that is displayed on video

Specify the caption to display on the video player window as any string. The default value of this property is Deployable Video Player.

WindowSize

Size of video display window

video.DeployableVideoPlayer class

Specify the video player window size as Full-screen, or True size (1:1). When this property is set to Full-screen, use the Esc key to exit out of full-screen mode. The default value for this property is True size (1:1).

FrameRate

Frame rate of video data

Specify the rate of video data in frames per second as a positive integer-valued scalar. The default value of this property is 30.

Methods

clone	Create deployable video player object with same property values
close	Release resources for the System object
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Send multidimensional video to computer screen

Examples

Play back a video on the screen.

```
hmfr = video.MultimediaFileReader;  
hdvp = video.DeployableVideoPlayer;  
while ~isDone(hmfr)  
    frame = step(hmfr);  
    step(hdvp, frame);  
end  
close(hmfr);
```

video.DeployableVideoPlayer class

```
close(hdvp);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the To Video Display block reference page. The object properties correspond to the block parameters, except for:

- The **Window Size** block parameter includes an additional option for Normal window size. You can only set the corresponding WindowSize object property to Full-screen or True size (1:1).
- The **Image Signal** block parameter allows you to specify whether the block accepts the color video signal as One Multidimensional Signal or Separate Color Signals. The object does not have a property that corresponds to the **Image Signal** block parameter. You must always provide the input image to the step method of the object as a single multidimensional signal.
- The To Video Display block opens the video player at the start of simulation.

See Also

signalblks.MultimediaFileWriter

Purpose Create deployable video player object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a `DeployableVideoPlayer` System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.DeployableVideoPlayer.close

Purpose Release resources for the DeployableVideoPlayer System object

Syntax `close(h)`

Description `close(h)` releases system resources (such as memory, file handles or hardware connections).

video.DeployableVideoPlayer.getNumInputs

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.DeployableVideoPlayer.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.DeployableVideoPlayer.isLocked

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the DeployableVideoPlayer System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.DeployableVideoPlayer.step

Purpose Send multidimensional video to computer screen

Syntax `step(H,I)`

Description `step(H,I)` sends one frame of a multidimensional video, I, to the computer screen.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose

Find edges of objects in images

Description

The EdgeDetector object finds edges of objects in images. For Sobel, Prewitt and Roberts algorithms, the object finds edges in an input image by approximating the gradient magnitude of the image. The gradient is obtained as a result of convolving the image with the Sobel, Prewitt or Roberts kernel. For Canny algorithm, the object finds edges by looking for the local maxima of the gradient of the input image. It calculates the gradient using the derivative of a Gaussian filter. This algorithm is more robust to noise and more likely to detect true weak edges.

Construction

`H = video.EdgeDetector` returns an edge detection System object, H, that finds edges in an input image using Sobel, Prewitt, Roberts, or Canny algorithm.

`H = video.EdgeDetector('PropertyName',PropertyValue,...)` returns an edge detection object, H, with each specified property set to the specified value.

Properties

Method

Edge detection algorithm

Specify the edge detection algorithm as `Sobel`, `Prewitt`, `Roberts`, or `Canny`. The default value for this property is `Sobel`.

BinaryImageOutputPort

Output the binary image

Set this property to `true` to output the binary image after edge detection. When this property is set to `true`, the object will output a boolean matrix. The nonzero elements of this matrix correspond to the edge pixels and the zero elements correspond to the background pixels. This property applies when you set the Method property to `Sobel`, `Prewitt` or `Roberts`. The default value for this property is `true`.

GradientComponentOutputPorts

video.EdgeDetector class

Output the gradient components

Set this property to `true` to output the gradient components after edge detection. When you set this property to `true`, and the `Method` property to `Sobel` or `Prewitt`, this `System` object outputs the gradient components that correspond to the horizontal and vertical edge responses. When you set the `Method` property to `Roberts`, the `System` object outputs the gradient components that correspond to the 45 and 135 degree edge responses. Both `BinaryImageOutputPort` and `GradientComponentOutputPorts` properties cannot be `false` at the same time. The default value for this property is `false`.

ThresholdSource

Source of threshold value

Specify how to determine threshold as `Auto`, `Property`, `Input port`. This property applies when you set the `Method` property to `Canny`. This property also applies when you set the `Method` property to `Sobel`, `Prewitt` or `Roberts` and the `BinaryImageOutputPort` property to `true`. The default value for this property is `Auto`.

Threshold

Threshold value(s)

Specify the threshold value as a scalar of MATLAB built-in numeric data type that is within the range of the input data when you set the `Method` property to `Sobel`, `Prewitt` or `Roberts`. Specify the threshold as a two-element vector of low and high values that define the weak and strong edges when you set the `Method` property to `Canny`. The default value is `[0.25 0.6]` when you set the `Method` property to `Canny`. Otherwise, the default value is `20`. This property is accessible when the `ThresholdSource` property is `Property`. This property is tunable.

ThresholdScaleFactor

Multiplier to adjust value of automatic threshold

Specify multiplier that is used to adjust calculation of automatic threshold as a scalar MATLAB built-in numeric data type. This property applies when you set the Method property to Sobel, Prewitt or Roberts and the ThresholdSource property to Auto. The default value for this property is 4. This property is tunable.

EdgeThinning

Enable performing edge thinning

Indicate whether edge thinning should be performed. Choosing to perform edge thinning requires additional processing time and resources. This property applies when you set the Method property to Sobel, Prewitt or Roberts and the BinaryImageOutputPort property to true. The default value of this property is false.

NonEdgePixelsPercentage

Approximate percentage of weak and non-edge pixels

Specify the approximate percentage of weak edge and non-edge image pixels as a scalar between 0 and 100. This property applies when set the Method property to Canny and the ThresholdSource to Auto. The default value for this property is 70. This property is tunable.

GaussianFilterStandardDeviation

Standard deviation of Gaussian

filter Specify the standard deviation of the Gaussian filter whose derivative is convolved with the input image. This property can be set to any positive scalar. This property applies when you set the Method property to Canny. The default value for this property is 1.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

video.EdgeDetector class

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. This property applies when you do not set the `Method` property to `Canny`. The default value for this property is `Custom`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. This property applies when you do not set the `Method` property to `Canny`. The default value for this property is `Wrap`.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as `Same as first input`, `Custom`. This property applies when you do not set the `Method` property to `Canny`. The default value for this property is `Custom`

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you do not set the `Method` property to `Canny`. This property applies when you set the `ProductDataType` property to `Custom`. The default value of this property is `numerictype([],32,8)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as first input`, `Same as product`, `Custom`. This property applies when you do not set the `Method` property to `Canny`. The default value for this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you do not set the `Method` property to `Canny`. This property applies when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType([],32,8)`.

GradientDataType

Gradient word and fraction lengths

Specify the gradient components fixed-point data type as `Same as accumulator`, `Same as first input`, `Same as product`, or `Custom`. This property applies when you do not set the `Method` property to `Canny` and you set the `GradientComponentPorts` property to `true`. The default value of this property is `Same as first input`.

CustomGradientDataType

Gradient word and fraction lengths

Specify the gradient components fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property is accessible when the `Method` property is not `Canny`. This property is applicable when the `GradientDataType` property is `Custom`. The default value of this property is `numericType([],16,4)`.

Methods

<code>clone</code>	Create edge detector object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method

video.EdgeDetector class

isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Operate on inputs to calculate outputs

Examples

Find the edges in an image

```
hedge = video.EdgeDetector;
hcsc = video.ColorSpaceConverter(...
'Conversion', 'RGB to intensity');
hidtypeconv = ...
    video.ImageDataTypeConverter('OutputDataType', 'single');
img = step(hcsc, imread('peppers.png'));
img1 = step(hidtypeconv, img);
edges = step(hedge, img1);
imshow(edges);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Edge Detection block reference page. The object properties correspond to the block parameters.

See Also

`video.TemplateMatcher`

Purpose

Create edge detector object with same property values

Syntax

`C = clone(H)`

Description

`C = clone(H)` creates a `EdgeDetector System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.EdgeDetector.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.EdgeDetector.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the EdgeDetector System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Operate on inputs to calculate outputs

Syntax

```
EDGES = step(H,IMG)
[GV,GH] = step(H,IMG)
[EDGES,GV,GH] = step(H,IMG)
```

Description EDGES = step(H,IMG) finds the edges, EDGES , in input IMG using the specified algorithm when the BinaryImageOutputPort property is true. EDGES is a boolean matrix with non-zero elements representing edge pixels and zero elements representing background pixels.

[GV,GH] = step(H,IMG) finds the two gradient components, GV and GH , of the input IMG when you set the Method property to Sobel, Prewitt or Roberts, the GradientComponentOutputPorts property to true and the BinaryImageOutputPort property to false. If you set the Method property to Sobel or Prewitt, then GV is a matrix of gradient values in the vertical direction and GH is a matrix of gradient values in the horizontal direction. If you set the Method property to Roberts, then GV represents the gradient component at 45 degree edge response, and GH represents the gradient component at 135 degree edge response.

[EDGES,GV,GH] = step(H,IMG) finds the edges, EDGES , and the two gradient components, GV and GH , of the input IMG when you set the Method property to Sobel, Prewitt or Roberts and both the BinaryImageOutputPort and GradientComponentOutputPorts properties are true.

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.FFT2D class

Purpose Two-dimensional discrete Fourier transform

Description The `video.FFT2D` object computes the 2D discrete Fourier transform (DFT) of a two-dimensional input matrix. Both the row and column dimension of the input matrix must be powers of two. The object uses one or more of the following fast Fourier transform (FFT) algorithms depending on the complexity of the input and whether the output is in linear or bit-reversed order:

- Double-signal algorithm
- Half-length algorithm
- Radix-2 decimation-in-time (DIT) algorithm
- Radix-2 decimation-in-frequency (DIF) algorithm

Construction `H = video.FFT2D` returns a 2D FFT object, `H`, that computes the fast Fourier transform of a two-dimensional input.

`H = video.FFT2D('PropertyName', PropertyValue, ...)` returns a 2D FFT object, `H`, with each property set to the specified value.

Properties

`TableOptimization`

Optimization of the trigonometric values table

Select the optimization of the trigonometric values table to be `Speed` or `Memory`. You must set this property to `Speed` for fixed-point inputs. The default value of this property is `Speed`.

`BitReversedOutput`

Output in bit-reversed order relative to input

Designates the order of output channel elements relative to the order of input elements. Set this property to `true` to output the frequency indices in bit-reversed order. The default value of this property is `false`, which corresponds to a linear ordering of frequency indices.

Normalize

Divide butterfly outputs by two

Set this property to `true` to divide each butterfly of the FFT by 2. The default value of this property is `false` and no scaling occurs.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. This property applies only when you set the `TableOptimization` property to `Speed`. The default value of this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. This property applies only when the `TableOptimization` property is `Speed`. The default value of this property is `Wrap`.

SineTableDataType

Sine table word and fraction lengths

Specify the sine table data type as `Same word length as input`, or `Custom`. This property applies only when the `TableOptimization` property is `Speed`. The default value of this property is `Same word length as input`.

CustomSineTableDataType

Sine table word and fraction lengths

Specify the sine table fixed-point type as an `unscaled numeric type` object with a `Signedness` of `Auto`. This property applies only when the `TableOptimization` property is `Speed` and the

video.FFT2D class

SineTableDataType property is Custom. The default value of this property is `numerictype([],16)`.

ProductDataType

Product word and fraction lengths

Specify the product data type as `Internal` rule, `Same as input`, or `Custom`. This property applies only when the `TableOptimization` property is `Speed`. The default value of this property is `Internal` rule.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies only when the `TableOptimization` property is `Speed` and the `ProductDataType` property is `Custom`. The default value of this property is `numerictype([],32,30)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator data type as `Internal` rule, `Same as input`, `Same as product`, or `Custom`. This property applies only when the `TableOptimization` property is `Speed`. The default value of this property is `Internal` rule.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies only when the `TableOptimization` property is `Speed` and the `AccumulatorDataType` property is `Custom`. The default value of this property is `numerictype([],32,30)`.

OutputDataType

Output word and fraction lengths

Specify the output data type as `Internal` rule, `Same` as input, or `Custom`. This property applies only when the `TableOptimization` property is `Speed`. The default value of this property is `Internal` rule.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies only when the `TableOptimization` property is `Speed` and the `OutputDataType` property is `Custom`. The default value of this property is `numericType([], 16, 15)`.

Methods

<code>clone</code>	Create FFT2D object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Compute 2D discrete Fourier transform of input

Examples

Use 2D FFT to view the frequency components of an image:

```
hfft2d = video.FFT2D;  
hcsc = video.ColorSpaceConverter(...  
'Conversion', 'RGB to intensity');  
hgs = video.GeometricScaler(...  
'SizeMethod', 'Number of output rows and columns', ...  
'Size', [512 512]);
```

video.FFT2D class

```
x = imread('saturn.png');
x1 = step(hgs,x);
ygs = step(hcsc, x1);
y = step(hfft2d, ygs);
y1 = fftshift(double(y));
imshow(log(max(abs(y1), 1e-6)),[]);
colormap(jet(64));
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the 2-D FFT block reference page. The object properties correspond to the block parameters.

See Also

[video.IFFT2D](#) | [video.DCT2D](#) | [video.IDCT2D](#)

Purpose	Create FFT2D object with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates an instance of the current FFT2D object with the same property values. The <code>clone</code> method creates a new unlocked object.

video.FFT2D.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.FFT2D.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the FFT2D System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Compute 2D discrete Fourier transform of input

Syntax $Y = \text{step}(H, X)$

Description $Y = \text{step}(H, X)$ computes the 2D discrete Fourier transform (DFT), Y , of an M -by- N input matrix X , where M and N are integer powers of two.

video.GammaCorrector class

Purpose Apply or remove gamma correction from images or video streams

Description The GammaCorrector object applies gamma correction to input images or video streams.

Construction `H = video.GammaCorrector` returns a System object, HGAMMACORR, that applies or removes gamma correction from images or video streams.

`H = video.GammaCorrector('PropertyName',PropertyValue,...)` returns a gamma corrector System object, H, with each specified property set to the specified value.

`H = video.GammaCorrector(GAMMA,'PropertyName',PropertyValue,...)` returns a gamma corrector System object, H, with the Gamma property set to GAMMA and other specified properties set to the specified values.

Properties

Correction

Specify gamma correction or linearization

Specify the object's operation as Gamma or De-gamma. The default value of this property is Gamma

Gamma

Gamma value of output or input

If you set the Correction property to Gamma, this property gives the desired gamma value of the output video stream. If you set the Correction property to De-gamma, this property indicates the gamma value of the input video stream. This property must be a numeric scalar value greater than or equal to 1. The default value of this property is 2.2.

LinearSegment

Enable gamma curve to have linear portion near origin

Set this property to true to make the gamma curve have a linear portion near the origin. The default value of this property is true.

BreakPoint

I-axis value of the end of gamma correction linear segment

Specify the I-axis value of the end of the gamma correction linear segment as a scalar numeric value between 0 and 1. This property applies when you set the `LinearSegment` property to `true`. The default value of this property is `0.018`.

Methods

<code>clone</code>	Create gamma corrector object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Apply or remove gamma correction from input

Examples

Improve image contrast.

```
hgamma = ...
    video.GammaCorrector(2.0, 'Correction', 'De-gamma');
x = imread('pears.png');
y = step(hgamma, x);
imshow(x); title('Original Image');
figure, imshow(y);
title('Enhanced Image after De-gamma Correction');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Gamma Correction block reference page. The object properties correspond to the block parameters.

video.GammaCorrector class

See Also

`video.HistogramEqualizer` | `video.ContrastAdjuster`

Purpose

Create gamma corrector object with same property values

Syntax

```
C = clone(H)
```

Description

`C = clone(H)` creates a `GammaCorrector System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.GammaCorrector.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.GammaCorrector.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.GammaCorrector.isLocked

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the GammaCorrector System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

Purpose

Apply or remove gamma correction from input

Syntax

$Y = \text{step}(H, X)$

Description

$Y = \text{step}(H, X)$ applies or removes gamma correction from input X and returns the gamma corrected or linearized output Y .

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.GeometricRotator class

Purpose Rotate image by specified angle

Description The GeometricRotator object rotates image by specified angle.

Construction `H = video.GeometricRotator` returns a geometric rotator System

object, H, that rotates an image by $\frac{\pi}{6}$ radians.

`H = video.GeometricRotator('PropertyName',PropertyValue...)` returns a geometric rotator object, H, with each specified property set to the specified value.

Properties OutputSize

Output size as full or same as input image size

Specify the size of output image as `Expanded to fit rotated input image`, or `Same as input image`. If this property is set to `Expanded to fit rotated input image`, the object outputs a matrix that contains all the rotated image values. If it is set to `Same as input image`, the object outputs a matrix that contains the middle part of the rotated image. The default value for this property is `Expanded to fit rotated input image`.

AngleSource

Source of angle

Specify how to specify the rotation angle as `Property` or `Input port`. The default value of this property is `Property`.

Angle

Rotation angle value (radians)

Set this property to a real, scalar value for the rotation angle (radians). This property applies when you set the `AngleSource` property to `Property`. The default value of this property is `pi/6`.

MaximumAngle

Maximum angle by which to rotate image

Specify the maximum angle by which to rotate the input image as a numeric scalar value greater than 0. This property applies when you set the `AngleSource` property to `Input port`. The default value of this property is `pi`.

RotatedImageLocation

How the image is rotated

Specify how the image is rotated as `Top-left corner`, or `Center`. If this property is set to `Center`, the image is rotated about its center point. If it is set to `Top-left corner`, the object rotates the image so that two corners of the input image are always in contact with the top and left side of the output image. This property applies when you set the `OutputSize` property to `Expanded to fit rotated input image`, and, the `AngleSource` property to `Input port`. The default value for this property is `Center`

SineComputation

How to calculate the rotation

Specify how to calculate the rotation as `Trigonometric function`, `Table lookup`. If this property is set to `Trigonometric function`, the object computes sine and cosine values it needs to calculate the rotation of the input image. If it is set to `Table lookup`, the object computes and stores the trigonometric values it needs to calculate the rotation of the input image in a table and uses the table for each step call. In this case, the object requires extra memory. The default value for this property is `Table lookup`.

BackgroundFillValue

Value of pixels outside image

Specify the value of pixels that are outside the image as a numeric scalar value or a numeric vector of same length as the third dimension of input image. The default value of this property is `0`. This property is tunable.

video.GeometricRotator class

InterpolationMethod

Interpolation method used to rotate image

Specify the interpolation method used to rotate the image as `Nearest neighbor`, `Bilinear`, or `Bicubic`. If this property is set to `Nearest neighbor`, the object uses the value of one nearby pixel for the new pixel value. If it is set to `Bilinear`, the new pixel value is the weighted average of the four nearest pixel values. If it is set to `Bicubic`, the new pixel value is the weighted average of the sixteen nearest pixel values. The default value for this property is `Bilinear`.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. This property applies when you set the `SineComputation` property to `Table lookup`. The default value for this property is `Nearest`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. This property applies when you set the `SineComputation` property to `Table lookup`. The default value for this property is `Saturate`.

AngleDataType

Angle word and fraction lengths

Specify the angle fixed-point data type as `Same word length as input`, or `Custom`. This property applies when you set the `SineComputation` property to `Table lookup`, and the `AngleSource` property to `Property`. The default value for this property is `Same word length as input`.

CustomAngleDataType

Angle word and fraction lengths

Specify the angle fixed-point type as a signed `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `SineComputation` property to `Table lookup`, the `AngleSource` property is `Property` and the `AngleDataType` property to `Custom`. The default value of this property is `numerictype([],32,10)`.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as `Same as first input`, or `Custom`. This property applies when you set the `SineComputation` property to `Table lookup`. The default value for this property is `Custom`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `SineComputation` property to `Table lookup`, and the `ProductDataType` property to `Custom`. The default value of this property is `numerictype([],32,10)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as product`, or `Same as first input`, `Custom`. This property applies when you set the `SineComputation` property to `Table lookup`. The default value for this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when

video.GeometricRotator class

you set the `SineComputation` property to `Table lookup`, and the `AccumulatorDataType` property to `Custom`. The default value of this property is `numerictype([],32,10)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Same as first input`, `Custom`. This property applies when you set the `SineComputation` property to `Table lookup`. The default value for this property is `Same as first input`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `SineComputation` property to `Table lookup`, and the `OutputDataType` property to `Custom`. The default value of this property is `numerictype([],32,10)`.

Methods

<code>clone</code>	Create geometric rotator object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Return a rotated image

Examples

Rotate an image 90 degrees ($\pi/2$ radians).

```
hrotate1 = video.GeometricRotator;
```

```
hrotate1.Angle = pi / 2;  
img1 = im2double(rgb2gray(imread('peppers.png')));  
% rotimg1 contains img1 rotated  
rotimg1 = step(hrotate1,img1);  
imshow(rotimg1);
```

Rotate an image, with the rotation angle provided as an input. By setting the AngleSource property to Input port, the rotation angle is passed as an input.

```
hrotate2 = video.GeometricRotator;  
hrotate2.AngleSource = 'Input port';  
img2 = im2double(rgb2gray(imread('onion.png')));  
% rotimg2 contains img2 rotated  
rotimg2 = step(hrotate2,img2,pi/4);  
imshow(rotimg2);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Rotate block reference page. The object properties correspond to the block parameters.

See Also

[video.GeometricTranslator](#) | [video.GeometricScaler](#)

video.GeometricRotator.clone

Purpose Create geometric rotator object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a GeometricRotator System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.GeometricRotator.getNumInputs

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.GeometricRotator.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the GeometricRotator System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.GeometricRotator.step

Purpose Return a rotated image

Syntax `Y = step(H,IMG)`
`Y = step(H,IMG,ANGLE)`

Description `Y = step(H,IMG)` returns a rotated image `Y` , with the rotation angle specified by the `Angle` property.
`Y = step(H,IMG,ANGLE)` uses input `ANGLE` as the angle to rotate the input `IMG` when the `AngleSource` property is set to `Input port`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Enlarge or shrink image sizes
Description	The GeometricScaler object enlarges or shrinks image sizes.
Construction	<p>H = video.GeometricScaler returns a System object, H, that changes the size of an image or a region of interest within an image.</p> <p>H = video.GeometricScaler('PropertyName',PropertyValue,...) returns a geometric scaler object, H, with each specified property set to the specified value.</p>
Properties	<p>SizeMethod</p> <p>Aspects of image to resize</p> <p>Specify which aspects of the input image to resize as Output size as a percentage of input size, Number of output columns and preserve aspect ratio, Number of output rows and preserve aspect ratio, Number of output rows and columns. The default value for this property is Output size as a percentage of input size.</p> <p>ResizeFactor</p> <p>Percentage by which to resize rows and columns</p> <p>Set this property to a scalar percentage value that is applied to both rows and columns or a two-element vector, where the first element is the percentage by which to resize the rows and the second element is the percentage by which to resize the columns. This property applies when you set the SizeMethod property to Output size as a percentage of input size. The default value of this property is [200 150].</p> <p>NumOutputColumns</p> <p>Number of columns in output image</p> <p>Specify the number of columns of the output image as a positive integer scalar value. This property applies when you set the</p>

video.GeometricScaler class

SizeMethod property to Number of output columns and preserve aspect ratio. The default value of this property is 25.

NumOutputRows

Number of rows in output image

Specify the number of rows of the output image as a positive integer scalar value. This property applies when you set the SizeMethod property to Number of output rows and preserve aspect ratio. The default value of this property is 25.

Size

Dimensions of output image

Set this property to a two-element vector, where the first element is the number of rows in the output image and the second element is the number of columns. This property applies when you set the SizeMethod property to Number of output rows and columns. The default value of this property is [25 35].

InterpolationMethod

Interpolation method used to resize the image

Specify the interpolation method to resize the image as Nearest neighbor, Bilinear, Bicubic, Lanczos2, Lanczos3. If this property is set to Nearest neighbor, the object uses one nearby pixel to interpolate the pixel value. If it is set to Bilinear, the object uses four nearby pixels to interpolate the pixel value. If it is set to Bicubic or Lanczos2, the object uses sixteen nearby pixels to interpolate the pixel value. If it is set to Lanczos3, the object uses thirty six surrounding pixels to interpolate the pixel value. The default value for this property is Bilinear.

Antialiasing

Enable low-pass filtering when shrinking image

Set this property to true to perform low-pass filtering on the input image before shrinking it, to prevent aliasing when ResizeFactor is between 0 and 100 percent.

ROIProcessing

Enable region-of-interest processing

Indicate whether to resize a particular region of each input image. This property applies when you set the `SizeMethod` property to `Number of output rows and columns`, the `InterpolationMethod` parameter to `Nearest neighbor`, `Bilinear`, or `Bicubic`, and the `Antialiasing` property to `false`. The default value is `false`.

ROIValidityOutputPort

Enable indication that ROI is outside input image

Indicate whether to return the validity of the specified ROI being completely inside image. This property applies when you set the `ROIProcessing` property to `true`. The default value is `false`.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value for this property is `Nearest`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. The default value for this property is `Saturate`.

InterpolationWeightsDataType

Interpolation weights word and fraction lengths

Specify the interpolation weights fixed-point data type as `Same word length as input`, `Custom`. The default value for this

video.GeometricScaler class

property is Same word length as input. This property is applicable under any of the following conditions:

- If you set the InterpolationMethod property to Bicubic, Lanczos2, or Lanczos3.
- If you set the SizeMethod property to any value other than Output size as a percentage of input size.
- If you set:

The SizeMethod property to Output size as a percentage of input size.

The InterpolationMethod property to Bilinear or Nearest neighbor.

Any of the elements of the ResizeFactor is less than 100, implying shrinking the input image.

The Antialiasing property to true

CustomInterpolationWeightsDataType

Interpolation word and

fraction lengths Specify the interpolation weights fixed-point type as an unscaled numeric type object with a Signedness of Auto. This property applies when you set the InterpolationWeightsDataType property to Custom. The default value of this property is numeric type([], 32).

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as Same as input, or Custom. The default value for this property is Custom.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled numeric type object with a Signedness of Auto. This property applies when you

set the `ProductDataType` property to `Custom`. The default value of this property is `numerictype([],32,10)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as product`, `Same as input`, `Custom`. The default value for this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numerictype([],32,10)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Same as input`, or `Custom`. The default value for this property is `Same as input`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Custom`. The default value of this property is `numerictype([],32,10)`.

Methods

<code>clone</code>	Create geometric scaler object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method

video.GeometricScaler class

getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Resize an image

Examples

Enlarge an image. Display the original and the enlarged images.

```
x=imread('cameraman.tif');
hgs=video.GeometricScaler;
hgs.SizeMethod = ...
    'Output size as a percentage of input size';
hgs.InterpolationMethod='Bilinear';
y = step(hgs,x);
imshow(x); title('Original Image');
figure,imshow(y);title('Resized Image');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the [Resize](#) block reference page. The object properties correspond to the block parameters.

See Also

[video.Pyramid](#) | [video.GeometricRotator](#) | [video.GeometricTranslator](#)

Purpose

Create geometric scaler object with same property values

Syntax

```
C = clone(H)
```

Description

`C = clone(H)` creates a `GeometricScaler System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.GeometricScaler.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.GeometricScaler.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the GeometricScaler System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Resize an image

Syntax
`Y = step(H,X)`
`Y = step(H,X,ROI)`
`[Y,FLAG] = step(H,X,ROI)`

Description `Y = step(H,X)` returns a resized image, `Y`, of input image `X`.

`Y = step(H,X,ROI)` resizes a particular region of the image `X` defined by the `ROI` input. This option applies when you set the `SizeMethod` property to `Number of output rows and columns`, the `Antialiasing` property to `false`, the `InterpolationMethod` property to `Bilinear`, `Bicubic` or `Nearest neighbor`, and the `ROIProcessing` property to `true`.

`[Y,FLAG] = step(H,X,ROI)` also returns `FLAG` which indicates whether the given region of interest is within the image bounds. This applies when you set the `SizeMethod` property to `Number of output rows and columns`, the `Antialiasing` property to `false`, the `InterpolationMethod` property to `Bilinear`, `Bicubic` or `Nearest neighbor` and, the `ROIProcessing` and the `ROIValidityOutputPort` properties to `true`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the `System` object issues a warning and re-initializes.

video.GeometricTransformer class

Purpose	Apply projective or affine transformation to an image
Description	The GeometricTransformer object applies a projective or affine transformation to an image.
Construction	<p>H = video.GeometricTransformer returns a geometric transformation System object, H, which applies a projective or affine transformation to an image.</p> <p>H = video.GeometricTransformer('PropertyName', PropertyValue, ...) returns a geometric transformation object, H, with each specified property set to the specified value.</p>
Properties	<p>TransformMatrixSource Method to specify transformation matrix Specify as Property or Input port. The default value for this property is Input port.</p> <p>TransformMatrix Transformation matrix Specify the applied transformation matrix as a 2-by-3 or 6-by-Q affine transformation matrix or a 3-by-3 or a 9-by-Q projective transformation matrix. Q is the number of transformations. This property applies when you set the TransformMatrixSource property to Property. The default value for this property is [1 0 0; 0 1 0; 0 0 1].</p> <p>InterpolationMethod Interpolation method Specify as Nearest neighbor, Bilinear, or Bicubic for calculating the output pixel value. The default value for this property is Bilinear.</p> <p>BackgroundFillValue</p>

Background fill value

Specify the value of the pixels that are outside of the input image. The value can be either scalar or a P -element vector, where P is the number of color planes. The default value for this property is 0.

OutputImagePositionSource

Method to specify output image location and size

Specify the value of this property as `Auto` or `Property`. If this property is set to `Auto`, the output image location and size are the same values as the input image. The default value for this property is `Auto`.

OutputImagePosition

Output image position vector

Specify the location and size of output image, as a four-element double vector in pixels, of the form, [left top height width]. This property applies when you set the `OutputImagePositionSource` property to `Property`. The default value for this property is [0 0 512 512].

ROIInputPort

Enable the region of interest input port

Set this property to `true` to enable the input of the region of interest. When set to `false`, the whole input image is processed. The default value for this property is `false`.

ROIShape

Region of interest shape

Specify `ROIShape` as `Rectangle ROI`, or `Polygon ROI`. This property applies when you set the `ROIInputPort` property to `true`. The default value for this property is `Rectangle ROI`.

ROIValidityOutputPort

video.GeometricTransformer class

Enable output of ROI flag

Set this property to `true` to enable the output of an ROI flag indicating when any part of the ROI is outside the input image. This property applies when you set the `ROIInputPort` property to `true`. The default value for this property is `false`.

`ProjectiveTransformMethod`

Projective transformation method

Method to compute the projective transformation. Specify as `Compute exact values`, or `Use quadratic approximation`. The default value for this property is `Compute exact values`.

`ErrorTolerance`

Error tolerance (in pixels)

Specify the maximum error tolerance in pixels for the projective transformation. This property applies when you set the `ProjectiveTransformMethod` property to `Use quadratic approximation`. The default value for this property is `1`.

`ClippingStatusOutputPort`

Enable clipping status flag output

Set this property to `true` to enable the output of a flag indicating if any part of the output image is outside the input image. The default value for this property is `false`.

Methods

<code>clone</code>	Create geometric transformer object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method

isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Apply geometric transform to input image

Examples

Apply a horizontal shear to an intensity image.

```
htrans1 = video.GeometricTransformer(...
    'TransformMatrixSource', 'Property', ...
    'TransformMatrix',[1 0 0; .5 1 0; 0 0 1],...
    'OutputImagePositionSource', 'Property',...
    'OutputImagePosition', [0 0 400 750]);
img1 = im2single(rgb2gray(imread('peppers.png')));
transimg1 = step(htrans1,img1);
imshow(transimg1);
```

Apply a transform with multiple polygon ROIs.

```
htrans2 = video.GeometricTransformer;
img2 = checker_board(20,10);
tfMat=[1      0      -15*2      0      1      15*2; ...
       0.4082 0      15*2      -0.4082 1.0204 35*2; ...
       1      -0.4082 5.4082*2 0      0.4082 44.5918*2]';

polyROI = [50  0 50 49 99 49 99 0; ...
           0  0 0 49 49 49 49 0; ...
           50 50 50 99 99 99 99 50]' * 2;

htrans2.BackgroundFillValue = [0.5 0.5 0.75];
htrans2.ROIInputPort = true;
htrans2.ROIShape = 'Polygon ROI';
transimg2 = step(htrans2,img2,tfMat,polyROI);
imshow(img2);
figure;imshow(transimg2);
```

video.GeometricTransformer class

Algorithm

This object implements the algorithm, inputs, and outputs described on the Apply Geometric Transformation block reference page. The object properties correspond to the block parameters.

See Also

`video.GeometricTransformEstimator`

Purpose Create geometric transformer object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a GeometricTransformer System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.GeometricTransformer.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.GeometricTransformer.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.GeometricTransformer.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the GeometricTransformer System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose

Apply geometric transform to input image

Syntax

```
Y = step(H,X,TFORM)
Y = step(H,X)
Y = step(H,X,ROI)
[Y, ROIFLAG]=(H,X,...)
[Y,CLIPFLAG]=(H,X,...)
[Y,ROIFLAG,CLIPFLAG] = step(H,X,TFORM,ROI)
```

Description

`Y = step(H,X,TFORM)` outputs the transformed image, `Y`, of the input image, `X`. `X` is either an M -by- N or an M -by- N -by- P matrix, where M is the number of rows, N is the number of columns and P is the number of color planes in the image. `TFORM` is the applied transformation matrix. `TFORM` can be a 2-by-3 or 6-by- Q affine transformation matrix, or a 3-by-3 or 9-by- Q projective transformation matrix, where Q is the number of transformations.

`Y = step(H,X)` outputs the transformed image, `Y`, of the input image, `X`, when you set the `TransformMatrixSource` property to `Property`.

`Y = step(H,X,ROI)` outputs the transformed image of the input image within the region of interest, `ROI`. When specifying a rectangular region of interest, `ROI` must be a 4-element vector or a 4-by- R matrix. When specifying a polygonal region of interest, `ROI` must be a $2L$ -element vector or a $2L$ -by- R matrix. R is the number of regions of interest, and L is the number of vertices in a polygon region of interest.

`[Y, ROIFLAG]=(H,X,...)` returns a boolean flag, `ROIFLAG`, indicating if any part of the region of interest is outside the input image, when you set the `ROIValidityOutputPort` property to `true`.

`[Y,CLIPFLAG]=(H,X,...)` returns a boolean flag, `CLIPFLAG`, indicating if any transformed pixels were clipped, when you set the `ClippingStatusOutputPort` property to `true`.

`[Y,ROIFLAG,CLIPFLAG] = step(H,X,TFORM,ROI)` outputs the transformed image, `Y`, of the input image, `X`, within the region of interest, `ROI`, and using the transformation matrix, `TFORM`. `ROIFLAG`, indicates if any part of the region of interest is outside the input

video.GeometricTransformer.step

image, and CLIPFLAG , indicates if any transformed pixels were clipped. This provides all operations simultaneously with all possible inputs. Properties must be set appropriately.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.GeometricTransformEstimator class

Purpose	Estimate geometric transformation from matching point pairs
Description	The GeometricTransformEstimator object estimates geometric transformation from matching point pairs.
Construction	<p>H = video.GeometricTransformEstimator returns a geometric transform estimation System object, H, that finds the transformation matrix that maps the largest number of points between two images.</p> <p>H = video.GeometricTransformEstimator('PropertyName', PropertyValue, ...) returns a geometric transform estimation object, H, with each specified property set to the specified value.</p>
Properties	<p>Transform</p> <p>Transformation type</p> <p>Specify transformation type as <code>Nonreflective similarity</code>, <code>Affine</code>, or <code>Projective</code> . The default value for this property is <code>Projective</code>.</p> <p>ExcludeOutliers</p> <p>Whether to exclude outliers from input points</p> <p>Set this property to <code>true</code> to find and exclude outliers from the input points and use only the inlier points to calculate the transformation matrix. When this property is <code>false</code>, all input points are used to calculate the transformation matrix. The default value of this property is <code>true</code>.</p> <p>Method</p> <p>Method to find outliers</p> <p>Specify the method to find outliers as <code>Random Sample Consensus (RANSAC)</code>, <code>Least Median of Squares</code>. The default value for this property is <code>Random Sample Consensus (RANSAC)</code>.</p> <p>AlgebraicDistanceThreshold</p>

video.GeometricTransformEstimator class

Algebraic distance threshold for determining inliers

Specify a scalar threshold value for determining inliers as a positive scalar value. The threshold controls the upper limit used to find the algebraic distance in the RANSAC Method. This property applies when you set the Transform property to Projective and the Method property to Random Sample Consensus (RANSAC). The default value of this property is 1.5. This property is tunable.

PixelDistanceThreshold

Distance threshold for determining inliers in pixels

Specify the upper limit of algebraic distance a point can differ from the projection location of its associating point as a positive scalar value. This property applies when you set the Transform property to Nonreflective similarity or to Affine, and the Method property to Random Sample Consensus (RANSAC). The default value of this property is 1.5. This property is tunable.

NumRandomSamplingsMethod

How to specify number of random samplings

Indicate how to specify number of random samplings as Specified value, or Desired confidence. Set this property to Desired confidence to specify the number of random samplings as a percentage and a maximum number. This property applies when you set the ExcludeOutliers property to true and the Method property to Random Sample Consensus (RANSAC). The default value for this property is Specified value.

NumRandomSamplings

Number of random samplings

Specify the number of random samplings for the method chosen to perform as a positive integer value. This property applies when you set the NumRandomSamplingsMethod property to Specified value. The default value of this property is 100. This property is tunable.

DesiredConfidence

Probability to find largest group of points

Specify the probability to find the largest group of points that can be mapped by a transformation matrix as a percentage. This property applies when you set the NumRandomSamplingsMethod property to Desired confidence. The default value of this property is 99. This property is tunable.

MaximumRandomSamples

Maximum number of random samplings

Specify the maximum number of random samplings as a positive integer value. This property applies when you set the NumRandomSamplingsMethod property to Desired confidence. The default value of this property is 200. This property is tunable.

InlierPercentageSource

Source of inlier percentage

Indicate how to specify the threshold to stop random sampling when a percentage of input point pairs have been found as inliers. This property can be set to one of Auto, or Property. If set to Auto then inlier threshold is disabled. This property applies when you set the Method property to Random Sample Consensus (RANSAC). The default value for this property is Auto.

InlierPercentage

Percentage of point pairs to be found to stop random sampling

Specify the percentage of point pairs that need to be determined as inliers to stop random sampling. This property applies when you set the InlierPercentageSource property to Property. The default value of this property is 75. This property is tunable.

RefineTransformMatrix

Whether to refine transformation matrix

video.GeometricTransformEstimator class

Set this property to `true` to perform additional iterative refinement on the transformation matrix. This property applies when you set the `ExcludeOutliers` property to `true`. The default value of this property is `false`.

InlierOutputPort

Enable output of the inlier point pairs

Set this property to `true` to output the inlier point pairs that were used to calculate the transformation matrix. This property applies when you set the `ExcludeOutliers` property to `true`. This property is not used when the data type of points is `signed` or `double`. The default value of this property is `false`.

TransformMatrixDataType

Data type of the transformation matrix

Specify transformation matrix data type as `single` or `double` when the input points are built-in integers. This property is not used when the data type of points is `single` or `double`. The default value for this property is `single`.

Methods

<code>clone</code>	Create geometric transform estimator object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method

video.GeometricTransformEstimator class

isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Calculate transformation matrix mapping largest number of valid points from input arrays

Examples

Compute nonreflective similarity transformation matrix that maps a pair of points between two images.

```
I = checkerboard; % checkerboard image
theta = 30/180*pi;
hrotate = video.GeometricRotator;
% default rotation angle of hrotate is 0.5236 radians
hgte = video.GeometricTransformEstimator(...
'Transform', 'Nonreflective similarity');
pts1=[14.0000 44.0000;70.0000 81.0000;44.7440 38.0656;...
44.7440 89.1072;72.8884 15.6455].';
% Corresponding points in the rotated image
m = [cos(theta) sin(theta);-sin(theta) cos(theta)];
pts2 = m*pts1;

% Rotate the image by 0.5236 radians (or 30 degrees)
Irot = step(hrotate, I);
% Verify that the estimated transformation matrix
% has same values as the matrix m
tform = step(hgte, pts1, pts2, uint8(5));
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Estimate Geometric Transformation block reference page. The object properties correspond to the block parameters.

See Also

video.GeometricTransformer

video.GeometricTransformEstimator.clone

Purpose Create geometric transform estimator object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a GeometricTransformEstimator System object C, with the same property values as H. The clone method creates a new unlocked object.

video.GeometricTransformEstimator.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.GeometricTransformEstimator.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.GeometricTransformEstimator.isLocked

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the GeometricTransformEstimator System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.GeometricTransformEstimator.step

Purpose Calculate transformation matrix mapping largest number of valid points from input arrays

Syntax `TFORM = step(H,PTS1,PTS2,NUM)`
`[TFORM,INLIERIND] = step(H,PTS1,PTS2,NUM)`

Description `TFORM = step(H,PTS1,PTS2,NUM)` calculates the transformation matrix, `TFORM`, that maps the largest number of valid points from the input arrays `PTS1` to `PTS2`. Both the input arguments, `PTS1` and `PTS2`, specify the location of points in two images, and each column in the two arrays has the format `[row; column]`. The points in the input arrays must be ordered to form corresponding location pairs. `NUM` is a scalar value that represents the number of valid points in `PTS1` and `PTS2`. Setting the `Transform` property to `Projective`, sets `TFORM` dimension to `3x3`, otherwise `TFORM` gets set to dimension of `2x3`.

`[TFORM,INLIERIND] = step(H,PTS1,PTS2,NUM)` performs the same operation as above, and also outputs the boolean vector, `INLIERIND`, indicating which points are the inliers. This option applies when you set the `InlierOutputPort` property to `true`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.GeometricTranslator class

Purpose	Translate image in two-dimensional plane using displacement vector
Description	The GeometricTranslator object translates images in two-dimensional plane using displacement vector.
Construction	<p>H = video.GeometricTranslator returns a System object, H, that moves an image up or down and/or left or right.</p> <p>H = video.GeometricTranslator('PropertyName',PropertyValue, ...) returns a geometric translator System object, H, with each specified property set to the specified value.</p>
Properties	<p>OutputSize</p> <p>Output size as full or same as input image size</p> <p>Specify the size of output image after translation as <code>Full</code> or <code>Same as input image</code>. If this property is set to <code>Full</code>, the object outputs a matrix that contains the translated image values. If it is set to <code>Same as input image</code>, the object outputs a matrix that is the same size as the input image and contains a portion of the translated image. The default value for this property is <code>Full</code>.</p> <p>OffsetSource</p> <p>Source of specifying offset values</p> <p>Specify how the translation parameters are provided as <code>Input port</code>, or <code>Property</code>. When the <code>OffsetSource</code> property is set to <code>Input port</code> a two-element offset vector must be provided to the System object's <code>step</code> method. The default value of this property is <code>Property</code>.</p> <p>Offset</p> <p>Translation values</p> <p>Specify the number of pixels to translate the image as a two-element offset vector. The first element of the vector represents a shift in the vertical direction and a positive value moves the image downward. The second element of the vector</p>

video.GeometricTranslator class

represents a shift in the horizontal direction and a positive value moves the image to the right. This property applies when you set the `OffsetSource` property to `Property`. The default value of this property is [1.5 2.3].

MaximumOffset

Maximum number of pixels by which to translate image

Specify the maximum number of pixels by which to translate the input image as a two-element real vector with elements greater than 0. This property must have the same data type as the `Offset` input. This property applies when you set the `OutputSize` property to `Full` and `OffsetSource` property to `Input port`. The system object uses this property to determine the size of the output matrix. If the `Offset` input is greater than this property value, the object saturates to the maximum value. The default value of this property is [8 10].

BackgroundFillValue

Value of pixels outside image

Specify the value of pixels that are outside the image as a numeric scalar value or a numeric vector of same length as the third dimension of input image. The default value of this property is 0.

InterpolationMethod

Interpolation method used to translate image

Specify the interpolation method used to translate the image as `Nearest neighbor`, `Bilinear`, `Bicubic`. If this property is set to `Nearest neighbor`, the System object uses the value of the nearest pixel for the new pixel value. If it is set to `Bilinear`, the new pixel value is the weighted average of the four nearest pixel values. If it is set to `Bicubic`, the new pixel value is the weighted average of the sixteen nearest pixel values. The default value for this property is `Bilinear`.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value for this property is `Nearest`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. The default value of this property is `Saturate`.

OffsetValuesDataType

Offset word and fraction lengths

Specify the offset fixed-point data type as `Same word length as input`, or `Custom`. The default value of this property is `Same word length as input`.

CustomOffsetValuesDataType

Offset word and fraction lengths

Specify the offset fixed-point type as a signed `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OffsetValuesDataType` property to `Custom`. The default value of this property is `numericType([], 16, 6)`.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as `Same as first input`, or `Custom`. The default value of this property is `Custom`.

CustomProductDataType

Product word and fraction lengths

video.GeometricTranslator class

Specify the product fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `ProductDataType` property to `Custom`. The default value of this property is `numericType([],32,10)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as product`, `Same as first input`, or `Custom`. The default value of this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType([],32,10)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Same as first input`, or `Custom`. The default value of this property is `Same as first input`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Custom`. The default value of this property is `numericType([],32,10)`.

video.GeometricTranslator class

Methods

clone	Create geometric translator object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Return a translated image

Examples

Translate an image

```
htranslate=video.GeometricTranslator;  
htranslate.OutputSize='Same as input image';  
htranslate.Offset=[30 30];  
I=im2single(imread('cameraman.tif'));  
Y = step(htranslate,I);  
imshow(Y);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Translate block reference page. The object properties correspond to the block parameters.

See Also

video.GeometricRotator

video.GeometricTranslator.clone

Purpose Create geometric translator object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an `GeometricTranslator System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.GeometricTranslator.getNumInputs

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.GeometricTranslator.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the GeometricTranslator System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.GeometricTranslator.step

Purpose Return a translated image

Syntax
`Y = step(H,I)`
`Y = step(H,I,Offset)`

Description `Y = step(H,I)` translates the input image `I`, and returns a translated image `Y`, with the offset specified by the `Offset` property.

`Y = step(H,I,Offset)` uses input `Offset` as the offset to translate the image `I` when the `OffsetSource` property is to `Input port`. `Offset` is a two-element offset vector that represents the number of pixels to translate the image. The first element of the vector represents a shift in the vertical direction and a positive value moves the image downward. The second element of the vector represents a shift in the horizontal direction and a positive value moves the image to the right.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Generate histogram of each input matrix
Description	The Histogram2D object generates histogram of each input matrix.
Construction	<p>H = video.Histogram2D returns a histogram System object, H, that computes the frequency distribution of the elements in each input matrix.</p> <p>H = video.Histogram2D('PropertyName',PropertyValue,...) returns a histogram object, H, with each specified property set to the specified value.</p> <p>H = video.Histogram2D(MIN,MAX,NUMBINS,'PropertyName',PropertyValue,...) returns a histogram System object, H, with the LowerLimit property set to MIN, UpperLimit property set to MAX, NumBins property set to NUMBINS and other specified properties set to the specified values.</p>
Properties	<p>LowerLimit Lower boundary Specify the lower boundary of the lowest-valued bin as a real-valued scalar value. NaN and Inf are not valid values for this property. The default value of this property is 0. This property is tunable.</p> <p>UpperLimit Upper boundary Specify the upper boundary of the highest-valued bin as a real-valued scalar value. NaN and Inf are not valid values for this property. The default value of this property is 1. This property is tunable.</p> <p>NumBins Number of bins in the histogram</p>

video.Histogram2D class

Specify the number of bins in the histogram. The default value of this property is 256.

Normalize

Enable output vector normalization

Specify whether the output vector, v , is normalized such that $\text{sum}(v) = 1$. Use of this property is not supported for fixed-point signals. The default value of this property is `false`.

RunningHistogram

Enable calculation over time

Set this property to `true` to enable computing the histogram of the input elements over time. Set this property to `false` to enable basic histogram operation. The default value of this property is `false`.

ResetInputPort

Enable resetting in running histogram mode

Set this property to `true` to enable resetting the running histogram. When the property is set to `true`, a reset input must be specified to the `step` method to reset the running histogram. This property applies when you set the `RunningHistogram` property to `true`. When you set this property to `false`, the object does not reset. The default value of this property is `false`.

ResetCondition

Condition for running histogram mode

Specify event to reset the running histogram as `Rising edge`, `Falling edge`, `Either edge`, or `Non-zero`. This property applies when you set the `ResetInputPort` property to `true`. The default value of this property is `Non-zero`.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value of this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. The default value of this property is `Wrap`.

ProductDataType

Data type of product

Specify the product data type as `Internal rule`, `Same as input`, or `Custom`. The default value of this property is `Internal rule`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a signed, scaled `numericType` object. This property applies only when you set the `ProductDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

AccumulatorDataType

Data type of the accumulator

Specify the accumulator data type as `Same as input`, `Same as product`, or `Custom`. The default value of this property is `Same as input`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a signed, scaled `numericType` object. This property applies only when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

video.Histogram2D class

Methods

<code>clone</code>	Create 2-D histogram object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>reset</code>	Reset histogram bin values to zero
<code>step</code>	Return histogram for input data

Examples

Compute histogram of a grayscale image.

```
img = im2single(rgb2gray(imread('peppers.png')));  
hhist2d = video.Histogram2D;  
y = step(hhist2d,img);  
bar((0:255)/256, y);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Histogram block reference page. The object properties correspond to the block parameters, except for:

- **Reset port** block parameter corresponds to both the `ResetCondition` and the `ResetInputPort` object properties.

See Also

`signalblks.Histogram`

Purpose

Create 2-D histogram object with same property values

Syntax

`C = clone(H)`

Description

`C = clone(H)` creates an Histogram2D System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object with uninitialized states.

video.Histogram2D.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.Histogram2D.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the Histogram2D System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose	Reset histogram bin values to zero
Syntax	<code>reset(H)</code>
Description	<code>reset(H)</code> sets the Histogram object bin values to zero when the <code>RunningHistogram</code> property is <code>true</code> .

video.Histogram2D.step

Purpose Return histogram for input data

Syntax $Y = \text{step}(H, X)$
 $Y = \text{step}(H, X, R)$

Description $Y = \text{step}(H, X)$ returns a histogram Y for the input data X . When you set the `RunningHistogram` property to `true`, Y corresponds to the histogram of the input elements over time.

$Y = \text{step}(H, X, R)$ computes the histogram of the input X elements over time, and optionally resets the object's state based on the value of R and the object's `ResetCondition` property. This applies when you set the `RunningHistogram` and `ResetInputPort` properties to `true`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.HistogramEqualizer class

Purpose	Enhance contrast of images using histogram equalization
Description	The HistogramEqualizer object enhances contrast of images using histogram equalization.
Construction	<p>H = video.HistogramEqualizer returns a System object, H, that enhances the contrast of input image using histogram equalization.</p> <p>H = video.HistogramEqualizer('PropertyName',PropertyValue,...) returns a histogram equalization object, H, with each specified property set to the specified value.</p>
Properties	<p>Histogram</p> <p>How to specify histogram</p> <p>Specify the desired histogram of the output image as Uniform, Input port, or Custom. The default value of this property is Uniform.</p> <p>CustomHistogram</p> <p>Desired histogram of output image</p> <p>Specify the desired histogram of output image as a numeric vector. This property applies when you set the Histogram property to Custom. The default value of this property is <code>ones(1,64)</code>.</p> <p>BinCount</p> <p>Number of bins for uniform histogram to have</p> <p>Specify the number of equally spaced bins the uniform histogram has as an integer scalar value greater than 1. This property applies when you set the Histogram property to Uniform. The default value of this property is 64.</p>

video.HistogramEqualizer class

Methods

clone	Create histogram equalizer object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Perform histogram equalization on input image

Examples

Enhance quality of an image.

```
hhisteq = video.HistogramEqualizer;
x = imread('tire.tif');
y = step(hhisteq, x);
imshow(x); title('Original Image');
figure, imshow(y);
title('Enhanced Image after histogram equalization');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Histogram Equalization block reference page. The object properties correspond to the block parameters, except for:

- The **Histogram** property for the object, corresponds to both the **Target Histogram** and the **Histogram Source** parameters for the block.

See Also

signalblks.Histogram

Purpose Create histogram equalizer object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an HistogramEqualizer System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.HistogramEqualizer.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.HistogramEqualizer.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.HistogramEqualizer.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the HistogramEqualizer System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose

Perform histogram equalization on input image

Syntax

`Y = step(H,X)`
`Y = step(H,X,HIST)`

Description

`Y = step(H,X)` performs histogram equalization on input image, `X`, and returns the enhanced image, `Y`.

`Y = step(H,X,HIST)` performs histogram equalization on input image, `X` using input histogram, `HIST`, and returns the enhanced image, `Y` when the `Histogram` property is `Input port`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.HoughLines class

Purpose Find Cartesian coordinates of lines that are described by rho and theta pairs

Description The HoughLines object finds Cartesian coordinates of lines that are described by rho and theta pairs. The object inputs are the theta and rho values of lines and a reference image. The object outputs the zero-based row and column positions of the intersections between the lines and two of the reference image boundary lines. The boundary lines are the left and right vertical boundaries and the top and bottom horizontal boundaries of the reference image.

Construction H = video.HoughLines returns a Hough lines System object, HHoughLines, that finds Cartesian coordinates of lines that are described by rho and theta pairs.

H = video.HoughLines('PropertyName',PropertyValue,...) returns a Hough lines object, HHoughLines, with each specified property set to the specified value.

Properties SineComputation
Method to calculate sine values used to find intersections of lines
Specify how to calculate sine values which are used to find intersection of lines as Trigonometric function, or Table lookup. If this property is set to Trigonometric function, the object computes sine and cosine values it needs to calculate the intersections of the lines. If it is set to Table lookup, the object computes and stores the trigonometric values it needs to calculate the intersections of the lines in a table and uses the table for each step call. In this case, the object requires extra memory. For floating-point inputs, this property must be set to Trigonometric function. For fixed-point inputs, the property must be set to Table lookup. The default value for this property is Table lookup.

ThetaResolution
Spacing of the theta-axis

Specify the spacing of the theta-axis. This property applies when you set the `SineComputation` property to `Table` lookup.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. This property applies when you set the `SineComputation` property to `Table` lookup. The default value for this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. This property applies when you set the `SineComputation` property to `Table` lookup. The default value for this property is `Wrap`.

SineTableDataType

Sine table word and fraction lengths

Specify the sine table fixed-point data type as a constant property always set to `Custom`. This property applies when you set the `SineComputation` property to `Table` lookup.

CustomSineTableDataType

Sine table word and fraction lengths

Specify the sine table fixed-point type as an unscaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `SineComputation` property to `Table` lookup, and the `SineTableDataType` property to `Custom`. The default value of this property is `numericType([], 16)`.

ProductDataType

Product word and fraction lengths

video.HoughLines class

Specify the product fixed-point data type as `Same as first input`, `Custom`. This property applies when you set the `SineComputation` property to `Table lookup`. The default value for this property is `Custom`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `SineComputation` property to `Table lookup`, and the `ProductDataType` property to `Custom`. The default value of this property is `numerictype([],32,16)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as product`, `Custom`. This property applies when you set the `SineComputation` property to `Table lookup`. The default value for this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `SineComputation` property to `Table lookup`, and the `AccumulatorDataType` property to `Custom`. The default value of this property is `numerictype([],32,16)`.

Methods

<code>clone</code>	Create hough lines object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method

getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Output intersection coordinates of a line described by a theta and rho pair and reference image boundary lines

Examples

Use Hough lines to detect the longest line in an image. Use the edge detection object to find edges in the intensity image. This `step` method outputs a binary image required by the `HoughTransform` object and improves the efficiency of the `HoughLines` object.

```
I = imread('circuit.tif');
hedge = video.EdgeDetector;
hhoughtrans = video.HoughTransform(pi/360, ...
    'ThetaRhoOutputPort', true);
hfindmax = video.LocalMaximaFinder(1, ...
    'HoughMatrixInput', true);
H = video.HoughLines('SineComputation', ...
    'Trigonometric function');

% Find the edges in the intensity image
BW = step(hedge, I);
% Run the edge output through the transform
[ht, theta, rho] = step(hhoughtrans, BW);
% Find the location of the max value in the Hough matrix.
idx = step(hfindmax, ht);
% Find the longest line.
linepts = step(H, theta(idx(2)), rho(idx(1)), I);

% View the image superimposed with the longest line.
imshow(I); hold on;
```

video.HoughLines class

```
line(linepts([2 4]), linepts([1 3]), 'color', [1 1 0]);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Hough Lines block reference page. The object properties correspond to the block parameters.

See Also

`video.HoughTransform` | `video.LocalMaximaFinder` |
`video.EdgeDetector`

Purpose

Create hough lines object with same property values

Syntax

`C = clone(H)`

Description

`C = clone(H)` creates a HoughLines System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.HoughLines.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.HoughLines.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the HoughLines System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Output intersection coordinates of a line described by a theta and rho pair and reference image boundary lines

Syntax `PTS = step(H, THETA, RHO, REFIMG)`

Description `PTS = step(H, THETA, RHO, REFIMG)` outputs PTS as the zero-based row and column positions of the intersections between the lines described by THETA and RHO and two of the reference image boundary lines.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.HoughTransform class

Purpose Find lines in images via Hough transform

Description The HoughTransform object finds lines in images via Hough transform. The Hough transform maps points in the Cartesian image space to curves in the Hough parameter space using the following equation:

$$\rho = x * \cos(\theta) + y * \sin(\theta)$$

Here, ρ denotes the distance from the origin to the line along a vector perpendicular to the line, and θ denotes the angle between the x -axis and this vector. This object computes the parameter space matrix, whose rows and columns correspond to the ρ and θ values respectively. Peak values in this matrix represent potential straight lines in the input image.

Construction `H = video.HoughTransform` returns a Hough transform System object, `H`, that implements the Hough transform to detect lines in images.

`H = video.HoughTransform('PropertyName', PropertyValue, ...)` returns a Hough transform object, `H`, with each specified property set to the specified value.

`H = video.HoughTransform(THETARES, RHORES, 'PropertyName', PropertyValue, ...)` returns a Hough transform object, `H`, with the `ThetaResolution` property set to `THETARES`, the `RhoResolution` property set to `RHORES`, and other specified properties set to the specified values.

Properties `ThetaResolution`

Theta resolution in radians

Specify the spacing of the Hough transform bins along the theta-axis in radians, as a scalar numeric value between 0 and $\pi/2$. The default value of this property is $\pi/180$.

`RhoResolution`

Rho resolution

Specify the spacing of the Hough transform bins along the rho-axis as a scalar numeric value greater than 0. The default value of this property is 1.

ThetaRhoOutputPort

Enable theta and rho outputs

Set this property to `true` for the object to output theta and rho values. The default value of this property is `false`.

OutputDataType

Data type of output

Specify the data type of the output signal as `double`, `single`, or `Fixed point`. The default value of this property is `double`.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. This property applies when you set the `OutputDataType` property to `Fixed point`. The default value for this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. This property applies when you set the `OutputDataType` property to `Fixed point`. The default value for this property is `Saturate`.

SineTableDataType

Sine table word and fraction lengths

This property is constant and is set to `Custom`. This property applies when you set the `OutputDataType` property to `Fixed point`.

video.HoughTransform class

CustomSineTableDataType

Sine table word and fraction lengths

Specify the sine table fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Fixed point`. The default value of this property is `numerictype([],16,14)`.

RhoDataType

Rho word and fraction lengths

This property is constant and is set to `Custom`. This property applies when you set the `OutputDataType` property to `Fixed point`.

CustomRhoDataType

Rho word and fraction lengths

Specify the rho fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Fixed point`. The default value of this property is `numerictype([],32,16)`.

ProductDataType

Product word and fraction lengths

This property is constant and is set to `Custom`. This property applies when you set the `OutputDataType` property to `Fixed point`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Fixed point`. The default value of this property is `numerictype([],32,30)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same` as product, `Custom`. This property applies when you set the `OutputDataType` property to `Fixed point`. The default value of this property is `Custom`.

`CustomAccumulatorDataType`

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Fixed point`. The default value of this property is `numericType([],32,30)`.

`HoughOutputDataType`

Hough output word and fraction lengths

This property is constant and is set to `Custom`. This property applies when you set the `OutputDataType` property to `Fixed point`.

`CustomHoughOutputDataType`

Hough output word and fraction lengths

Specify the hough output fixed-point data type as an unscaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Fixed point`. The default value of this property is `numericType(false,16)`.

`ThetaOutputDataType`

Theta output word and fraction lengths

This property is constant and is set to `Custom`. This property applies when you set the `OutputDataType` property to `Fixed point`.

`CustomThetaOutputDataType`

video.HoughTransform class

Theta output word and fraction lengths

Specify the theta output fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Fixed point`. The default value of this property is `numericType([],32,16)`.

Methods

<code>clone</code>	Create Hough transform object with same property values
<code>getNumInputs</code>	Number of expected inputs to <code>step</code> method
<code>getNumOutputs</code>	Number of outputs from <code>step</code> method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Output parameter space matrix for binary input image matrix

Examples

Compute the hough transform of an image. Use the hough transform to detect the longest line in the image.

```
I = imread('circuit.tif');

% Use the EdgeDetection System object to find edges in the
% intensity image. This step outputs a binary image
% required by the HoughTransform System object and
% improves the efficiency of the HoughLines System object.
hedge = video.EdgeDetector;
hhoughtrans = video.HoughTransform(pi/360, ...
    'ThetaRhoOutputPort', true);
hfindmax = video.LocalMaximaFinder(1, ...
    'HoughMatrixInput', true);
hhoughlines = video.HoughLines('SineComputation', ...
```

```
'Trigonometric function');

% Find the edges in the intensity image
BW = step(hedge, I);
% Run the edge output through the transform
[ht, theta, rho] = step(hhoughtrans, BW);
% Find the location of the max value in the Hough matrix.
idx = step(hfindmax, ht);
% Find the longest line.
linepts = step(hhoughlines, theta(idx(2)), rho(idx(1)), I);

% View the image superimposed with the longest line.
imshow(I); hold on;
line(linepts([2 4]), linepts([1 3]));
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Hough Transform block reference page. The object properties correspond to the block parameters.

See Also

[video.DCT2D](#) | [video.HoughLines](#) | [video.LocalMaximaFinder](#) | [video.EdgeDetector](#)

video.HoughTransform.clone

Purpose Create Hough transform object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a HoughTransform System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.HoughTransform.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.HoughTransform.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the HoughTransform System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.HoughTransform.step

Purpose Output parameter space matrix for binary input image matrix

Syntax `HT = step(H,BW)`
`[HT,THETA,RHO] = step(H,BW)`

Description `HT = step(H,BW)` outputs the parameter space matrix, `HT`, for the binary input image matrix `BW`.

`[HT,THETA,RHO] = step(H,BW)` also returns the theta and rho values, in vectors `THETA` and `RHO` respectively, when you set the `ThetaRhoOutputPort` property to `true`. `RHO` denotes the distance from the origin to the line along a vector perpendicular to the line, and `THETA` denotes the angle between the x-axis and this vector. This object computes the parameter space matrix, whose rows and columns correspond to the rho and theta values respectively. Peak values in this matrix represent potential straight lines in the input image.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Compute 2-D inverse discrete cosine transform
Description	The IDCT2D object computes 2-D inverse discrete cosine transform of the input signal. The number of rows and columns of the input matrix must be a power of 2.
Construction	<p>H = video.IDCT2D returns a System object, H, used to compute the two-dimensional inverse discrete cosine transform (2-D IDCT) of a real input signal.</p> <p>H = video.IDCT2D('PropertyName',PropertyValue,...) returns a 2-D inverse discrete cosine transform System object, H, with each specified property set to the specified value.</p>
Properties	<p>SineComputation</p> <p>Specify how the System object computes sines and cosines as Trigonometric function, or Table lookup. This property must be set to Table lookup for fixed-point inputs.</p> <p>Fixed-Point Properties</p> <p>RoundingMethod</p> <p>Rounding method for fixed-point operations</p> <p>Specify the rounding method as Ceiling, Convergent, Floor, Nearest, Round, Simplest, or Zero. This property applies when you set the SineComputation to Table lookup. The default value for this property is Floor.</p> <p>OverflowAction</p> <p>Overflow action for fixed-point operations</p> <p>Specify the overflow action as Wrap or Saturate. This property applies when you set the SineComputation to Table lookup. The default value for this property is Wrap.</p> <p>SineTableDataType</p>

video.IDCT2D class

Sine table word-length designation

Specify the sine table fixed-point data type as `Same word length as input`, or `Custom`. This property applies when you set the `SineComputation` to `Table lookup`. The default value for this property is `Same word length as input`.

CustomSineTableDataType

Sine table word length

Specify the sine table fixed-point type as a signed, unscaled `numericType` object. This property applies when you set the `SineComputation` to `Table lookup` and you set the `SineTableDataType` property to `Custom`. The default value of this property is `numericType(true, 16)`.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as `Internal rule`, `Same as first input`, or `Custom`. This property applies when you set the `SineComputation` to `Table lookup`. The default value for this property is `Custom`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a signed, scaled `numericType` object. This property applies when you set the `SineComputation` to `Table lookup`, and the `ProductDataType` property to `Custom`. The default value of this property is `numericType(true, 32, 30)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Internal rule`, `Same as input`, `Same as product`, `Same as first input`, or `Custom`. This property applies when you set the `SineComputation`

property to `Table lookup`. The default value of this property is `Internal rule`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a signed, scaled `numericType` object. This property applies when you set the `SineComputation` to `Table lookup`, and `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Internal rule`, `Same as first input`, or `Custom`. This property applies when you set the `SineComputation` to `Table lookup`. The default value for this property is `Custom`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a signed, scaled `numericType` object. This property applies when you set the `SineComputation` to `Table lookup`, and the `OutputDataType` property to `Custom`. The default value of this property is `numericType(true,16,15)`.

Methods

<code>clone</code>	Create 2-D inverse discrete cosine transform object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method

video.IDCT2D class

isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Compute 2-D inverse discrete cosine transform of the input

Examples

Use 2-D discrete cosine transform (DCT) to analyze the energy content in an image. Set the DCT coefficients lower than a threshold of 0. Reconstruct the image using 2-D inverse discrete cosine transform (IDCT).

```
hdct2d = video.DCT2D;
I = double(imread('cameraman.tif'));
J = step(hdct2d, I);
imshow(log(abs(J)),[], colormap(jet(64)), colorbar

hidct2d = video.IDCT2D;
J(abs(J) < 10) = 0;
It = step(hidct2d, J);
figure, imshow(I, [0 255]), title('Original image')
figure, imshow(It,[0 255]), title('Reconstructed image')
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the 2-D IDCT block reference page. The object properties correspond to the block parameters.

See Also

[video.DCT2D](#) | [signalblks.DCT](#) | [signalblks.IDCT](#)

Purpose	Create 2-D inverse discrete cosine transform object with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates a IDCT2D System object <code>C</code> , with the same property values as <code>H</code> . The <code>clone</code> method creates a new unlocked object.

video.IDCT2D.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.IDCT2D.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the IDCT2D System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Compute 2-D inverse discrete cosine transform of the input

Syntax $Y = \text{step}(H, X)$

Description $Y = \text{step}(H, X)$ computes the 2-D inverse discrete cosine transform, Y , of input X .

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.IFFT2D class

Purpose Two-dimensional inverse discrete Fourier transform

Description The `video.IFFT2D` object computes the inverse 2D discrete Fourier transform (IDFT) of a two-dimensional input matrix. Both the row and column dimensions of the input matrix must be powers of two. The object uses one or more of the following fast Fourier transform (FFT) algorithms depending on the complexity of the input and whether the output is in linear or bit-reversed order:

- Double-signal algorithm
- Half-length algorithm
- Radix-2 decimation-in-time (DIT) algorithm
- Radix-2 decimation-in-frequency (DIF) algorithm

Construction `H = video.IFFT2D` returns a 2D IFFT object, `H`, with the default property and value pair settings.

`H = video.IFFT2D('PropertyName',PropertyValue, ...)` returns a 2D IFFT object, `H`, with each property set to the specified value.

Properties

`TableOptimization`
Optimization of trigonometric values table
Select the optimization of the trigonometric values table as `Speed` or `Memory`. The property must be `Speed` for fixed-point inputs. The default value for this property is `Speed`.

`BitReversedInput`
Indicates whether input is in bit-reversed order
Set this property to `true` if the order of 2D FFT transformed input elements are in bit-reversed order. The default value of this property is `false`, which denotes linear ordering.

`ConjugateSymmetricInput`
Indicates whether input is conjugate symmetric

Set this property to `true` if the input is conjugate symmetric. The 2D DFT of a real valued signal is conjugate symmetric and setting this property to `true` optimizes the 2D IFFT computation method. Setting this property to `false` for conjugate symmetric inputs results in complex output values with nonzero imaginary parts. Setting this property to `true` for non conjugate symmetric inputs results in invalid outputs. This property must be `false` for fixed-point inputs. The default value of this property is `true`.

Normalize

Divide output by FFT length

Specify if the 2D IFFT output should be divided by the FFT length. The value of this property defaults to `true` and divides each element of the output by the product of the row and column dimensions of the input matrix.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. This property applies only when the `TableOptimization` property is `Speed`. The default value for this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. This property applies only when the `TableOptimization` property is `Speed`. The default value for this property is `Wrap`.

SineTableDataType

Sine table word and fraction lengths

video.IFFT2D class

Specify the sine table data type as `Same word length as input`, `Custom`. This property applies only when the `TableOptimization` property is `Speed`. The default value for this property is `Same word length as input`.

CustomSineTableDataType

Sine table word and fraction lengths

Specify the sine table fixed-point type as an unscaled `numericType` object with a `Signedness` of `Auto`. This property applies only when the `TableOptimization` property is `Speed` and the `SineTableDataType` property is `Custom`. The default value of this property is `numericType([],16)`.

ProductDataType

Product word and fraction lengths

Specify the product data type as `Internal rule`, `Same as input`, or `Custom`. This property applies only when the `TableOptimization` property is `Speed`. The default value for this property is `Internal rule`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies only when the `TableOptimization` property is `Speed` and the `ProductDataType` property is `Custom`. The default value of this property is `numericType([],32,30)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator data type as `Internal rule`, `Same as input`, `Same as product`, or `Custom`. This property applies only when the `TableOptimization` property is `Speed`. The default value for this property is `Internal rule`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies only when the `TableOptimization` property is `Speed` and the `AccumulatorDataType` property is `Custom`. The default value of this property is `numericType([],32,30)`.

OutputDataType

Output word and fraction lengths

Specify the output data type as `Internal`, `rule`, `Same` as input, or `Custom`. This property applies only when the `TableOptimization` property is `Speed`. The default value for this property is `Internal`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies only when the `TableOptimization` property is `Speed` and the `OutputDataType` property is `Custom`. The default value of this property is `numericType([],16,15)`.

Methods

<code>clone</code>	Create IFFT2D object with same property values
<code>getNumInputs</code>	Returns number of expected inputs to step method
<code>getNumOutputs</code>	Returns number of outputs of step method

video.IFFT2D class

isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Compute 2D inverse discrete Fourier transform

Examples

Use the 2D IFFT object to convert an intensity image.

```
hfft2d = video.FFT2D;  
hifft2d = video.IFFT2D;  
% Read in the image  
xorig = single(imread('cameraman.tif'));  
% Convert the image from the spatial  
% to frequency domain and back  
Y = step(hfft2d, xorig);  
xtran = step(hifft2d, Y);  
% Display the newly generated intensity image  
imshow(abs(xtran), []);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the 2-D IFFT block reference page. The object properties correspond to the Simulink block parameters.

See Also

[video.FFT2D](#) | [video.DCT2D](#) | [video.IDCT2D](#)

Purpose

Create IFFT2D object with same property values

Syntax

`C = clone(H)`

Description

`C = clone(H)` creates an instance of the current IFFT2D object with the same property values. The `clone` method creates a new unlocked object

video.IFFT2D.getNumInputs

Purpose Returns number of expected inputs to step method

Syntax `getNumInputs(H)`

Description `getNumInputs(H)` returns the number of expected inputs to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Returns number of outputs of step method

Syntax `getNumOutputs(H)`

Description `getNumOutputs(H)` returns the number of output arguments from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.IFFT2D.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax `isLocked(H)`

Description `isLocked(H)` returns the locked state of the IFFT2D object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Compute 2D inverse discrete Fourier transform

Syntax $Y = \text{step}(H, X)$

Description $Y = \text{step}(H, X)$ computes the 2D inverse discrete Fourier transform (IDFT), Y , of an M-by-N input matrix X , where M and N are integer powers of two.

video.ImageComplementer class

Purpose Compute complement of pixel values in binary, intensity, or RGB images

Description The ImageComplementer object computes the complement of pixel values in binary, intensity, or RGB images. For binary images, the object replaces pixel values equal to 0 with 1 and pixel values equal to 1 with 0. For an intensity or RGB image, the object subtracts each pixel value from the maximum value that can be represented by the input data type and outputs the difference.

Construction `H = video.ImageComplementer` returns an image complement System object, H, that computes the complement of a binary, intensity, or RGB image.

`H = video.ImageComplementer('PropertyName',PropertyValue,...)` returns an image complementer object, H, with each specified property set to the specified value.

Methods	<code>clone</code>	Create image complementer object with same property values
	<code>getNumInputs</code>	Number of expected inputs to step method
	<code>getNumOutputs</code>	Number of outputs from step method
	<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
	<code>step</code>	Compute complement of input image

Examples Compute the complement of an input image.

```
himgcomp = video.ImageComplementer;
```



```
hautoth = video.Autothresher;  
% Read in image  
I = imread('coins.png');  
% Convert the image to binary  
bw = step(hautoth, I);  
% Take the image complement  
Ic = step(himgcomp, bw);  
% Display the results  
figure;  
subplot(2,1,1), imshow(bw), title('Original Binary image')  
subplot(2,1,2), imshow(Ic), title('Complemented image')
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Image Complement block reference page. The object properties correspond to the block parameters.

See Also

`video.Autothresher`

video.ImageComplementer.clone

Purpose Create image complementer object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an ImageComplementer System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.ImageComplementer.getNumInputs

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.ImageComplementer.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the ImageComplementer System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.ImageComplementer.step

Purpose Compute complement of input image

Syntax $Y = \text{step}(H, X)$

Description $Y = \text{step}(H, X)$ computes the complement of an input image X .

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.ImageDataTypeConverter class

Purpose

Convert and scale input image to specified output data type

Description

The `ImageDataTypeConverter` object converts and scales an input image to a specified output data type. When converting between floating-point data types, the object casts the input into the output data type and clips values outside the range to 0 or 1. When converting between all other data types, the object casts the input into the output data type and scales the data type values into the dynamic range of the output data type. For double- and single-precision floating-point data types, the dynamic range is between 0 and 1. For fixed-point data types, the dynamic range is between the minimum and maximum values that can be represented by the data type.

Construction

`H = video.ImageDataTypeConverter` returns a System object, `H`, that converts the input image to a single precision data type.

`H = video.ImageDataTypeConverter('PropertyName',PropertyValue,...)` returns an image data type conversion object, `H`, with each specified property set to the specified value.

Properties

`OutputDataType`

Data type of output

Specify the data type of the output signal as `double`, `single`, `int8`, `uint8`, `int16`, `uint16`, `boolean`, or `Custom`. The default value for this property is `single`.

Fixed-Point Properties

`CustomOutputDataType`

Output word and fraction lengths

Specify the output fixed-point type as a signed or unsigned, `scalednumeric` object. This property applies when you set the `OutputDataType` property to `Custom`. The default value of this property is `numeric([],16,0)`.

video.ImageDataTypeConverter class

Methods

<code>clone</code>	Create image data type converter object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Convert data type of input image

Examples

Convert the image datatype from uint8 to single.

```
x = imread('pout.tif');
hidtypeconv = video.ImageDataTypeConverter;
y = step(hidtypeconv, x);
imshow(y);
whos y % Image has been converted from uint8 to single.
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Image Data Type Conversion block reference page. The object properties correspond to the block parameters.

See Also

`video.ColorSpaceConverter`

Purpose Create image data type converter object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a `ImageDataTypeConverter System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.ImageDataTypeConverter.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.ImageDataTypeConverter.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The getNumOutputs method returns a positive integer representing the number of outputs from the step method. This value will change if any properties that turn inputs on or off are changed.

video.ImageDataTypeConverter.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the ImageDataTypeConverter System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Convert data type of input image

Syntax $Y = \text{step}(H, X)$

Description $Y = \text{step}(H, X)$ converts the input image X to Y . The data type of Y is specified by the `OutputDataType` property.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.ImageFilter class

Purpose	Perform 2-D FIR filtering of input matrix
Description	The ImageFilter object performs 2-D FIR filtering of input matrix.
Construction	<p><code>H = video.ImageFilter</code> returns a System object, H, that performs two-dimensional FIR filtering of input matrix using the specified filter coefficient matrix.</p> <p><code>H = video.ImageFilter('PropertyName',PropertyValue,...)</code> returns an image filter System object, H, with each specified property set to the specified value.</p>
Properties	<p>SeparableCoefficients</p> <p>Set to true if filter coefficients are separable</p> <p>Using separable filter coefficients reduces the amount of calculations the object must perform to compute the output. The function <code>isfilterseparable</code> can be used to check filter separability. The default value of this property is <code>false</code>.</p> <p>CoefficientsSource</p> <p>Source of filter coefficients</p> <p>Indicate how to specify the filter coefficients as Property or Input port. The default value of this property is Property.</p> <p>Coefficients</p> <p>Filter coefficients</p> <p>Specify the filter coefficients as a real or complex-valued matrix. This property applies when you set the <code>SeparableCoefficients</code> property to <code>false</code> and the <code>CoefficientsSource</code> property to Property. The default value of this property is <code>[1 0; 0 -1]</code>.</p> <p>VerticalCoefficients</p> <p>Vertical filter coefficients for the separable filter</p>

Specify the vertical filter coefficients for the separable filter as a vector. This property applies when you set the `SeparableCoefficients` property to `true` and the `CoefficientsSource` property to `Property`. The default value of this property is `[4 0]`.

HorizontalCoefficients

Horizontal filter coefficients for the separable filter

Specify the horizontal filter coefficients for the separable filter as a vector. This property applies when you set the `SeparableCoefficients` property to `true` and the `CoefficientsSource` property to `Property`. The default value of this property is `[4 0]`.

OutputSize

Output size as `full`, `valid` or same as input image size

Specify how to control the size of the output as `Full`, `Same as first input`, or `Valid`. If you set this property to `Full`, the dimensions of the output image are as follows:

$$\begin{aligned}\text{output rows} &= \text{input rows} + \text{filter coefficient rows} + 1 \\ \text{output columns} &= \text{input columns} + \text{filter coefficient columns} + 1\end{aligned}$$

If you set this property to `Same as first input`, the output has the same dimensions as the input image.

If you set this property to `Valid`, the object filters the input image only where the coefficient matrix fits entirely within it, so no padding is required. In this case, the dimensions of the output image are as follows:

$$\begin{aligned}\text{output rows} &= \text{input rows} - \text{filter coefficient rows} + 1 \\ \text{output columns} &= \text{input columns} - \text{filter coefficient columns} + 1\end{aligned}$$

The default value for this property is `Full`.

PaddingMethod

How to pad boundary of input matrix

video.ImageFilter class

Specify how to pad the boundary of input matrix as `Constant`, `Replicate`, `Symmetric`, or `Circular`. Set this property to one of the following:

- `Constant` to pad the input matrix with a constant value
- `Replicate` to pad the input matrix by repeating its border values
- `Symmetric` to pad the input matrix with its mirror image
- `Circular` to pad the input matrix using a circular repetition of its elements

This property applies when you set the `OutputSize` property to `Full` or `Same` as first input.

The default value for this property is `Constant`.

`PaddingValueSource`

Source of padding value

Specify how to define the constant boundary value as `Property` or `Input port`. This property applies when you set the `PaddingMethod` property to `Constant`. The default value of this property is `Property`.

`PaddingValue`

Constant value with which to pad matrix

Specify a constant value with which to pad the input matrix. This property applies when you set the `PaddingMethod` property to `Constant` and the `PaddingValueSource` property to `Property`. The default value of this property is 0. This property is tunable.

`Method`

Method for filtering input matrix

Specify the method by which the object filters the input matrix as Convolution or Correlation. The default value of this property is Convolution.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as Ceiling, Convergent, Floor, Nearest, Round, Simplest, or Zero. The default value of this property is Floor.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as Wrap, or Saturate. The default value for this property is Wrap.

CoefficientsDataType

Coefficients word and fraction lengths

Specify the coefficients fixed-point data type as Same word length as input, or Custom. This property applies when you set the CoefficientsSource property to Property. The default value of this property is Custom.

CustomCoefficientsDataType

Coefficients word and fraction lengths

Specify the coefficients fixed-point type as a signed numeric type object with a Signedness of Auto. This property applies when you set the CoefficientsSource property to Property and the CoefficientsDataType property to Custom. The default value of this property is numeric type ([], 16).

ProductDataType

Product word and fraction lengths

video.ImageFilter class

Specify the product fixed-point data type as `Same as input`, or `Custom`. The default value of this property is `Custom`

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `ProductDataType` property to `Custom`. The default value of this property is `numericType([],32,10)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as product`, `Same as input`, or `Custom`. The default value of this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType([],32,10)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Same as input`, `Custom`. The default value for this property is `Same as input`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Custom`. The default value of this property is `numericType([],32,12)`.

Methods

<code>clone</code>	Create image filter object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Filter input image

Examples

Filter an image to enhance the edges of 45 degree

```
img = im2single(rgb2gray(imread('peppers.png')));  
hfir2d = video.ImageFilter;  
hfir2d.Coefficients = [1 0; 0 -.5];  
fImg = step(hfir2d, img);  
subplot(2,1,1);imshow(img);title('Original image')  
subplot(2,1,2);imshow(fImg);title('Filtered image')
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the 2-D FIR Filter block reference page. The object properties correspond to the block parameters.

See Also

`video.MedianFilter2D`

video.ImageFilter.clone

Purpose Create image filter object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an ImageFilter System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object with uninitialized states.

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.ImageFilter.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the ImageFilter System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.ImageFilter.step

Purpose Filter input image

Syntax
`Y = step(H,I)`
`Y = step(H,I,COEFFS)`
`Y = step(H,I,HV,HH)`
`Y = step(H,...,PVAL)`

Description `Y = step(H,I)` filters the input image `I` and returns the filtered image in `Y`.

`Y = step(H,I,COEFFS)` uses filter coefficients, `COEFFS`, to filter the input image when you set the `CoefficientsSource` property to `Input port` and the `SeparableCoefficients` property to `false`.

`Y = step(H,I,HV,HH)` uses vertical filter coefficients, `HV`, and horizontal coefficients, `HH`, to filter the input image when you set the `CoefficientsSource` property to `Input port` and the `SeparableCoefficients` property to `true`.

`Y = step(H,...,PVAL)` uses `PVAL` for the pad value when you set the `OutputSize` property to either `Full` or `Same` as first input, the `PaddingMethod` property to `Constant`, and the `PaddingValueSource` property to `Input port`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the `System` object issues a warning and re-initializes.

Purpose	Pad or crop input image along its rows, columns, or both
Description	The ImagePadder object pads or crop input image along its rows, columns, or both.
Construction	<p>HIMPAD = video.ImagePadder returns an image padder System object, HIMPAD, that performs two-dimensional padding and/or cropping of an input image.</p> <p>HIMPAD = video.ImagePadder('PropertyName',PropertyValue,...) returns an image padder object, HIMPAD, with each specified property set to the specified value.</p>
Properties	<p>Method</p> <p>How to pad input image</p> <p>Specify how to pad the input image as Constant, Replicate, Symmetric, or Circular. The default value for this property is Constant.</p> <p>PaddingValueSource</p> <p>How to specify pad value</p> <p>Indicate how to specify the pad value as either Property or Input port. This property applies when you set the Method property to Constant. The default value of this property is Property.</p> <p>PaddingValue</p> <p>Pad value</p> <p>Specify the constant scalar value with which to pad the image. This property applies when you set the Method property to Constant and the PaddingValueSource property to Property. The default value of this property is 0. This property is tunable.</p> <p>SizeMethod</p> <p>How to specify output image size</p>

video.ImagePadder class

Indicate how to pad the input image to obtain the output image by specifying `Pad size`, or `Output size`. When this property is `Pad size`, the size of the padding in the vertical and horizontal directions are specified. When this property is `Output size`, the total number of output rows and output columns are specified. The default value of this property is `Pad size`.

RowPaddingLocation

Location at which to add rows

Specify the direction in which to add rows to as `Top`, `Bottom`, `Both top and bottom`, or `None`. Set this property to `Top` to add additional rows to the top of the image, `Bottom` to add additional rows to the bottom of the image, `Both top and bottom` to add additional rows to the top and bottom of the image, and `None` to maintain the row size of the input image. The default value of this property is `Both top and bottom`.

NumPaddingRows

Number of rows to add

Specify the number of rows to be added to the top, bottom, or both sides of the input image as a scalar value. When the `RowPaddingLocation` property is `Both top and bottom`, this property can also be set to a two element vector, where the first element controls the number of rows the System object adds to the top of the image and the second element controls the number of rows the System object adds to the bottom of the image. This property applies when you set the `SizeMethod` property to `Pad size` and the `RowPaddingLocation` property is not set to `None`. The default value of this property is `[2 3]`.

NumOutputRowsSource

How to specify number of output rows

Indicate how to specify the number of output rows as `Property` or `Next power of two`. If this property is `Next power of two`, the System object adds rows to the input image until the number of

rows is equal to a power of two. This property applies when you set the `SizeMethod` property to `Output size`. The default value of this property is `Property`.

NumOutputRows

Total number of rows in output

Specify the total number of rows in the output as a scalar integer. If the specified number is smaller than the number of rows of the input image, then image is cropped. This property applies when you set the `SizeMethod` property to `Output size` and the `NumOutputRowsSource` property to `Property`. The default value of this property is 12.

ColumnPaddingLocation

Location at which to add columns

Specify the direction in which to add columns one of `Left`, `Right`, `Both left and right`, or `None`. Set this property to `Left` to add additional columns on the left side of the image, `Right` to add additional columns on the right side of the image, `Both left and right` to add additional columns on the left and right side of the image, and `None` to maintain the column length of the input image. The default value for this property is `Both left and right`.

NumPaddingColumns

Number of columns to add

Specify the number of columns to be added to the left, right, or both sides of the input image as a scalar value. When the `ColumnPaddingLocation` property is `Both left and right`, this property can also be set to a two element vector, where the first element controls the number of columns the System object adds to the left side of the image and the second element controls the number of columns the System object adds to the right side of the image. This property applies when you set the `SizeMethod` property to `Pad size` and the `NumPaddingColumns` property is not set to `None`. The default value of this property is 2.

video.ImagePadder class

NumOutputColumnsSource

How to specify number of output columns

Indicate how to specify the number of output columns as `Property` or `Next power of two`. If you set this property to `Next power of two`, the `System` object adds columns to the input until the number of columns is equal to a power of two. This property applies when you set the `SizeMethod` property to `Output size`. The default value of this property is `Property`.

NumOutputColumns

Total number of columns in output

Specify the total number of columns in the output as a scalar integer. If the specified number is smaller than the number of columns of the input image, then image is cropped. This property applies when you set the `SizeMethod` property to `Output size` and the `NumOutputColumnsSource` property to `Property`. The default value of this property is 10.

Methods

<code>clone</code>	Create image padder object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Perform two-dimensional padding or cropping of input

Examples

Pad two rows to the bottom, and three columns to the right of an image. Use the value of the last array element as the padding value.

```
himpad = video.ImagePadder('Method', 'Replicate', ...  
    'RowPaddingLocation', 'Bottom', ...  
    'NumPaddingRows', 2, ...  
    'ColumnPaddingLocation', 'Right', ...  
    'NumPaddingColumns', 3);  
x = [1 2;3 4];  
y = step(himpad,x);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Image Pad block reference page. The object properties correspond to the block parameters.

See Also

`video.GeometricScaler`

video.ImagePadder.clone

Purpose	Create image padder object with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates a ImagePadder System object C, with the same property values as H. The <code>clone</code> method creates a new unlocked object.

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.ImagePadder.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the ImagePadder System object.</p> <p>The <code>isLocked</code> method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the <code>step</code> method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the <code>isLocked</code> method returns a true value.</p>

video.ImagePadder.step

Purpose Perform two-dimensional padding or cropping of input

Syntax $Y = \text{step}(H, X)$
 $Y = \text{step}(H, X, \text{PAD})$

Description $Y = \text{step}(H, X)$ performs two-dimensional padding or cropping of input, X .
 $Y = \text{step}(H, X, \text{PAD})$ performs two-dimensional padding and/or cropping of input, X , using the input PAD as the pad value. This applies when you set the `Method` property to `Constant` and the `PaddingValueSource` property to `Input` port.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose

Find local maxima in matrices

Description

The LocalMaximaFinder object finds local maxima in matrices.

Construction

H = video.LocalMaximaFinder returns a local maxima finder System object, H, that finds local maxima in input matrices.

H =

video.LocalMaximaFinder('PropertyName',PropertyValue,...)

returns a local maxima finder object, H, with each specified property set to the specified value.

H =

video.LocalMaximaFinder(MAXNUM,NEIGHBORSIZE,'PropertyName',PropertyVal

returns a local maxima finder object, H, with the MaximumNumLocalMaxima property set to MAXNUM, NeighborhoodSize property set to NEIGHBORSIZE, and other specified properties set to the specified values.

Properties

MaximumNumLocalMaxima

Maximum number of maxima to find

Specify the maximum number of maxima to find as a positive scalar integer value. The default value of this property is 2.

NeighborhoodSize

Neighborhood size for zero-ing out values

Specify the size of the neighborhood around the maxima, over which the System object zeros out values, as a 2-element vector of positive odd integers. The default value of this property is [5 7].

ThresholdSource

Source of threshold

Specify how to enter the threshold value as Property, or Input port. The default value for this property is Property.

Threshold

video.LocalMaximaFinder class

Value that all maxima should match or exceed

Specify the threshold value as a scalar of MATLAB built-in numeric data type. This property applies when you set the ThresholdSource property to Property. The default value of this property is 10. This property is tunable.

HoughMatrixInput

Indicator of Hough Transform matrix input

Set this property to true if the input is antisymmetric about

the rho axis and the theta value ranges from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$ radians, which correspond to a Hough matrix. The default value of this property is false.

IndexDataType

Data type of index values

Specify the data type of index values as double, single, uint8, uint16, or uint32. The default value for this property is uint32.

CountDataType

Data type of value that represents number of maxima

Specify the data type of the value that represents the number of maxima as double, single, uint8, uint16, or uint32. The default value for this property is uint32.

Methods

clone	Create local maxima finder object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method

isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Find local maxima in input image

Examples

Find a local maxima in an input.

```
img = [0 0 0 0 0 0 0 0 0 0 0 0; ...  
0 0 0 1 1 2 3 2 1 1 0 0; ...  
0 0 0 1 2 3 4 3 2 1 0 0; ...  
0 0 0 1 3 5 7 5 3 1 0 0; ...  
0 0 0 1 2 3 4 3 2 1 0 0; ...  
0 0 0 1 1 2 3 2 1 1 0 0; ...  
0 0 0 0 0 0 0 0 0 0 0 0] / 7; % local max at row 3, col 6  
hfindmax=video.LocalMaximaFinder;  
hfindmax.NeighborhoodSize=[3 3];  
hfindmax.Threshold=.7;  
[index,count] = step(hfindmax, img);  
index(:,count)
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Find Local Maxima block reference page. The object properties correspond to the block parameters.

See Also

[video.HoughTransform](#) | [video.HoughLines](#) | [signalblks.Maximum](#)

video.LocalMaximaFinder.clone

Purpose Create local maxima finder object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an `LocalMaximaFinder` System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.LocalMaximaFinder.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.LocalMaximaFinder.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the LocalMaximaFinder System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.LocalMaximaFinder.step

Purpose Find local maxima in input image

Syntax [IDX,COUNT] = step(H,I)
[...] = step(H,I,THRESH)

Description [IDX,COUNT] = step(H,I) finds the local maxima in input image I , with the coordinates of the local maxima returned in IDX and the number of local maxima found in COUNT.

[...] = step(H,I,THRESH) finds the local maxima in input image I , using threshold value THRESH , when the ThresholdSource property is Input port.

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Draw markers on output image
Description	The MarkerInserter object draws predefined markers, such as circles, x-marks, plus signs, stars, or squares on an images by overwriting pixel values..
Construction	<p>H = video.MarkerInserter returns a marker inserter System object, H, that draws circles of size 3 on the input image.</p> <p>H = video.MarkerInserter('PropertyName',PropertyValue,...) returns a marker inserter object, H, with each specified property set to the specified value.</p>
Properties	<p>Shape</p> <p>Shape of marker(s) to draw</p> <p>Specify the type of marker(s) to draw as Circle, X-mark, Plus, Star, or Square. The default value of this property is Circle.</p> <p>Size</p> <p>Size of marker</p> <p>Specify the size of the marker, in pixels, as a scalar value greater than or equal to 1. The default value of this property is 3. This property is tunable.</p> <p>Fill</p> <p>Enable filling marker</p> <p>Set this property to true to fill the marker with an intensity value or a color. This property applies when you set the Shape property to Circle or Square. The default value of this property is false.</p> <p>BorderColorSource</p> <p>Source of border color</p> <p>Specify how the marker's border color is provided as Input port, Property. This property applies either when you set the Shape</p>

video.MarkerInserter class

property to X-mark, Plus, or Star, or when you set the Shape property to Circle or Square, and the Fill property to false. When you set BorderColorSource to Input port, a border color vector must be provided as an input to the System object's step method. The default value of this property is Property.

BorderColor

Border color of marker

Specify the border color of the marker as Black, White, or Custom. If this property is set to Custom, the CustomBorderColor property is used to specify the value. This property applies when the BorderColorSource property is enabled and set to Property. The default value of this property is Black.

CustomBorderColor

Intensity or color value for marker's border

Specify an intensity or color value for the marker's border. If the input is an intensity image, this property can be set to a scalar intensity value for one marker or R -element vector where R is the number of markers. If the input is a color image, this property can be set to a P -element vector where P is the number of color planes or a P -by- R matrix where P is the number of color planes and R is the number of markers. This property applies when you set the BorderColor property to Custom. This property is tunable when the Antialiasing property is false. The default value of this property is [200 120 50].

FillColorSource

Source of fill color

Specify how the marker's fill color is provided as Input port, or Property. This property applies when you set the Shape property to Circle or Square, and the Fill property to true. When this property is set to Input port, a fill color vector must be provided as an input to the System object's step method. The default value of this property is Property.

FillColor

Fill color of marker

Specify the color to fill the marker as `Black`, `White`, or `Custom`. If this property is set to `Custom`, the `CustomFillColor` property is used to specify the value. This property applies when the `FillColorSource` property is enabled and set to `Property`. The default value of this property is `Black`.

CustomFillColor

Intensity or color value for marker's interior

Specify an intensity or color value to fill the marker. If the input is an intensity image, this property can be set to a scalar intensity value for one marker or R -element vector where R is the number of markers. If the input is a color image, this property can be set to a P -element vector where P is the number of color planes or a P -by- R matrix where P is the number of color planes and R is the number of markers. This property applies when you set the `FillColor` property to `Custom`. This property is tunable when the `Antialiasing` property is `false`. The default value of this property is `[200 120 50]`.

Opacity

Opacity of shading inside marker

Specify the opacity of the shading inside the marker by a scalar value between 0 and 1, where 0 is transparent and 1 is opaque. This property applies when you set the `Fill` property to `true`. This property is tunable. The default value of this property is `0.6`.

ROIInputPort

Enable defining a region of interest to draw marker

Set this property to `true` to specify a region of interest (ROI) on the input image through an input to the `step` method. If the property is set to `false`, the object uses the entire image. The default value of this property is `false`.

video.MarkerInserter class

Antialiasing

Enable performing smoothing algorithm on marker

Set this property to `true` to perform a smoothing algorithm on the marker. This property applies when you do not set the `Shape` property to `Square` or `Plus`. The default value of this property is `false`.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. This property applies when you set the `Fill` property to `true` and/or the `Antialiasing` property to `true`. The default value of this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. This property applies when you set the `Fill` property to `true` and/or the `Antialiasing` property to `true`. The default value of this property is `Wrap`.

OpacityDataType

Opacity word length

Specify the opacity fixed-point data type as `Same word length as input` or `Custom`. This property applies when you set the `Fill` property to `true`. The default value for this property is `Custom`.

CustomOpacityDataType

Opacity word length

Specify the opacity fixed-point type as an `unscaled numeric type` object with a `Signedness` of `Auto`. This property applies when

you set the `Fill` property to `true` and the `OpacityDataType` property to `Custom`. The default value of this property is `numerictype([],16)`.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as `Same as first input` or `Custom`. This property applies when you set the `Fill` property to `true` and/or the `Antialiasing` property to `true`. The default value of this property is `Custom`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `Fill` property to `true` and/or the `Antialiasing` property to `true`, and the `ProductDataType` property to `Custom`. The default value of this property is `numerictype([],32,14)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as product`, `Same as first input`, or `Custom`. This property applies when you set the `Fill` property to `true` and/or the `Antialiasing` property to `true`. The default value of this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numerictype` object with a `Signedness` of `Auto`. This property applies when you set the `Fill` property to `true` and/or the `Antialiasing` property to `true`, and the `AccumulatorDataType` property to `Custom`. The default value of this property is `numerictype([],32,14)`;

video.MarkerInserter class

Methods

clone	Create marker inserter object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Draw specified marker on input image

Examples

Draw plus signs on an input image

```
hmarkerinserter = video.MarkerInserter;  
hmarkerinserter.Shape = 'Plus';  
I = im2double(imread('cameraman.tif'));  
Pts = [10 10; 20 20; 30 30]';  
y = step(hmarkerinserter, I, Pts);  
imshow(y);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Draw Markers block reference page. The object properties correspond to the block parameters, except for:

- The **Image signal** block parameter allows you to specify whether the block accepts the color video signal as **One multidimensional signal** or **Separate color signals**. The object does not have a property that corresponds to the **Image signal** block parameter. You must always provide the input image to the **step** method of the object as a single multidimensional signal.

See Also

video.ShapeInserter

Purpose	Create marker inserter object with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates an MarkerInserter System object <code>C</code> , with the same property values as <code>H</code> . The <code>clone</code> method creates a new unlocked object.

video.MarkerInserter.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.MarkerInserter.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.MarkerInserter.isLocked

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the MarkerInserter System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

Purpose Draw specified marker on input image

Syntax

```
Y = step(H,I,PTS)
Y = step(H,I,PTS,ROI)
Y = step(H,I,PTS,...,CLR)
```

Description `Y = step(H,I,PTS)` draws the specified marker on the input image `I` at the locations specified by `PTS`. The `PTS` input is a 2-by- N matrix of row and column pairs, where N is the total number of markers and each row and column pair defines a zero-based marker's center. The markers are embedded on the output image `Y`.

`Y = step(H,I,PTS,ROI)` draws the specified marker only in a rectangular area defined by the `ROI` when you set the `ROIInputPort` property to `true`. The `ROI` input is a four-element vector of integers, where the first two elements represent the zero-based row and column coordinates of the upper-left corner of the area and the second two elements represent the height and width of the area.

`Y = step(H,I,PTS,...,CLR)` uses the border or fill color `CLR` to draw the border or fill the specified marker, when you set the `BorderColorSource` property or the `FillColorSource` property to `Input port`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.Maximum class

Purpose	Find maximum values in input or sequence of inputs
Description	The Maximum object finds maximum values in an input or sequence of inputs.
Construction	<p>H = video.Maximum returns an object, H, that computes the value and index of the maximum elements in an input or a sequence of inputs.</p> <p>H = video.Maximum('PropertyName',PropertyValue,...) returns a maximum-finding object, H, with each specified property set to the specified value.</p>
Properties	<p>ValueOutputPort</p> <p>Output maximum value</p> <p>Set this property to true to output the maximum (when RunningMaximum is false) or the running maximum (when RunningMaximum is true). The default value of this property is true.</p> <p>RunningMaximum</p> <p>Calculate over single input or multiple inputs</p> <p>When you set this property to true, the object computes the maximum value over a sequence of inputs. When you set this property to false, the object computes the maximum value over the current input. The default value of this property is false.</p> <p>IndexOutputPort</p> <p>Output the index of the maximum value</p> <p>Set this property to true to output the index of the maximum value of the input. This property applies only when you set the RunningMaximum property to false. The default value of this property is true.</p> <p>ResetInputPort</p> <p>Additional input to enable resetting of running maximum</p>

Set this property to `true` to enable resetting of the running maximum. When you set this property to `true`, a reset input must be specified to the `step` method to reset the running maximum. This property applies only when you set the `RunningMaximum` property to `true`. The default value of this property is `false`.

ResetCondition

Condition that triggers resetting of running maximum

Specify the event that resets the running maximum as `Rising edge`, `Falling edge`, `Either edge`, or `Non-zero`. This property applies only when you set the `ResetInputPort` property to `true`. The default value of this property is `Non-zero`.

IndexBase

Numbering base for index of maximum value

Specify the numbering used when computing the index of the maximum value as starting from either `One` or `Zero`. This property applies only when you set the `IndexOutputPort` property to `true`. The default value of this property is `One`.

Dimension

Dimension to operate along

Specify how the maximum calculation is performed over the data as `All`, `Row`, `Column`, or `Custom`. This property applies only when you set the `RunningMaximum` property to `false`. The default value for this property is `Column`.

CustomDimension

Numerical dimension to calculate over

Specify the integer dimension of the input signal over which the object finds the maximum. The value of this property cannot exceed the number of dimensions in the input signal. This property only applies when you set the `Dimension` property to `Custom`. The default value of this property is `1`.

ROIProcessing

Enable region-of-interest processing

Set this property to `true` to enable calculation of the maximum value within a particular region of an image. This property applies when you set the `Dimension` property to `All` and the `RunningMaximum` property to `false`. The default value of this property is `false`.

ROIForm

Type of region of interest

Specify the type of region of interest as `Rectangles`, `Lines`, `Label matrix`, or `Binary mask`. This property applies only when you set the `ROIProcessing` property to `true`. The default value of this property is `Rectangles`.

ROIPortion

Calculate over entire ROI or just perimeter

Specify whether to calculate the maximum over the `Entire ROI` or the `ROI perimeter`. This property applies only when you set the `ROIForm` property to `Rectangles`. The default value of this property is `Entire ROI`.

ROIStatistics

Calculate statistics for each ROI or one for all ROIs

Specify whether to calculate `Individual statistics` for each ROI or a `Single statistic` for all ROIs. This property applies only when you set the `ROIForm` property to `Rectangles`, `Lines`, or `Label matrix`.

ValidityOutputPort

Output flag indicating if any part of ROI is outside input image

When you set the `ROIForm` property to `Lines` or `Rectangles`, set this property to `true` to return the validity of the specified ROI being completely inside of the image. When you set the `ROIForm`

property to `Label Matrix`, set this property to `true` to return the validity of the specified label numbers. The default value of this property is `false`.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value of this property is `Floor`.

OverflowAction

Action to take when integer input is out-of-range

Specify the overflow action as `Wrap` or `Saturate`. The default value of this property is `Wrap`.

ProductDataType

Data type of product

Specify the product fixed-point data type as `Same as input` or `Custom`. The default value of this property is `Same as input`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numericType` object. This property applies only when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

AccumulatorDataType

Data type of accumulator

Specify the accumulator fixed-point data type as `Same as product`, `Same as input`, or `Custom`. The default value of this property is `Same as product`.

video.Maximum class

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numerictype` object. This property applies only when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numerictype(true,32,30)`.

Methods

<code>clone</code>	Create maximum object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>reset</code>	Reset computation of running maximum
<code>step</code>	Compute maximum value

Examples

Determine the maximum value and its index in a grayscale image.

```
img = im2single(rgb2gray(imread('peppers.png')));  
hmax = video.Maximum;  
[m, ind] = step(hmax, img);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the `Maximum` block reference page. The object properties correspond to the block parameters, except for:

- Only the block supports the **Treat sample-based row input as column** parameter.

See Also `signalblks.Maximum` | `video.Mean` | `video.Minimum`

video.Maximum.clone

Purpose Create maximum object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a Maximum object C, with the same property values as H. The `clone` method creates a new unlocked object with uninitialized states.

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.Maximum.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the Maximum System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.Maximum.reset

Purpose	Reset computation of running maximum
Syntax	<code>reset(H)</code>
Description	<code>reset(H)</code> resets the computation of the running maximum for the <code>Maximum</code> object <code>H</code> .

Purpose

Compute maximum value

Syntax

```
[VAL,IND] = step(H,X)
VAL = step(H,X)
IND = step(H,X)
VAL = step(H,X,R)
[...] = step(H,I,ROI)
[...] = step(H,I,LABEL,LABELNUMBERS)
[... ,FLAG] = step(H,I,ROI)
[... ,FLAG] = step(H,I,LABEL,LABELNUMBERS)
```

Description

`[VAL,IND] = step(H,X)` returns the maximum value, VAL, and the index or position of the maximum value, IND, along a dimension of X specified by the value of the Dimension property.

`VAL = step(H,X)` returns the maximum value, VAL, of the input X. When the RunningMaximum property is true, VAL corresponds to the maximum value over a sequence of inputs.

`IND = step(H,X)` returns the zero- or one-based index IND of the maximum value. To enable this type of processing, set the IndexOutputPort property to true and the ValueOutputPort and RunningMaximum properties to false.

`VAL = step(H,X,R)` computes the maximum value, VAL, over a sequence of inputs, and resets the state of H based on the value of reset signal, R, and the ResetCondition property. To enable this type of processing, set the RunningMaximum property to true and the ResetInputPort property to true.

`[...] = step(H,I,ROI)` computes the maximum of an input image, I, within the given region of interest, ROI. To enable this type of processing, set the ROIProcessing property to true and the ROIForm property to Lines, Rectangles or Binary mask.

`[...] = step(H,I,LABEL,LABELNUMBERS)` computes the maximum of an input image, I, for a region whose labels are specified in the vector LABELNUMBERS. To enable this type of processing, set the ROIProcessing property to true and the ROIForm property to Label matrix.

video.Maximum.step

[..., FLAG] = step(H,I,ROI) also returns FLAG, indicating whether the given region of interest is within the image bounds. To enable this type of processing, set the ROIProcessing and ValidityOutputPort properties to true and the ROIForm property to Lines, Rectangles or Binary mask.

[..., FLAG] = step(H,I,LABEL,LABELNUMBERS) also returns FLAG, indicating whether the input label numbers are valid. To enable this type of processing, set the ROIProcessing and ValidityOutputPort properties to true and the ROIForm property to Label matrix.

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Find mean value of input or sequence of inputs
Description	The Mean object finds the mean of an input or sequence of inputs.
Construction	<p><code>H = video.Mean</code> returns an object, H, that computes the mean of an input or a sequence of inputs.</p> <p><code>H = video.Mean('PropertyName',PropertyValue,...)</code> returns a mean-finding object, H, with each specified property set to the specified value.</p>
Properties	<p>RunningMean</p> <p>Calculate over single input or multiple inputs</p> <p>When you set this property to <code>true</code>, the object calculates the mean over a sequence of inputs. When you set this property to <code>false</code>, the object computes the mean over the current input. The default value of this property is <code>false</code>.</p> <p>ResetInputPort</p> <p>Additional input to enable resetting of running mean</p> <p>Set this property to <code>true</code> to enable resetting of the running mean. When you set this property to <code>true</code>, a reset input must be specified to the <code>step</code> method to reset the running mean. This property applies only when you set the <code>RunningMean</code> property to <code>true</code>. The default value of this property is <code>false</code>.</p> <p>ResetCondition</p> <p>Condition that triggers resetting of running mean</p> <p>Specify the event that resets the running mean as <code>Rising edge</code>, <code>Falling edge</code>, <code>Either edge</code>, or <code>Non-zero</code>. This property applies only when you set the <code>ResetInputPort</code> property to <code>true</code>. The default value of this property is <code>Non-zero</code>.</p> <p>Dimension</p> <p>Dimension to operate along</p>

video.Mean class

Specify how the mean calculation is performed over the data as All, Row, Column, or Custom. This property applies only when you set the RunningMean property to false. The default value of this property is All.

CustomDimension

Numerical dimension to calculate over

Specify the integer dimension, indexed from one, of the input signal over which the object calculates the mean. The value of this property cannot exceed the number of dimensions in the input signal. This property only applies when you set the Dimension property to Custom. The default value of this property is 1.

ROIProcessing

Enable region-of-interest processing

Set this property to true to enable calculation of the mean within a particular region of an image. This property applies when you set the Dimension property to All and the RunningMean property to false. The default value of this property is false.

ROIForm

Type of region of interest

Specify the type of region of interest as Rectangles, Lines, Label matrix, or Binary mask. This property applies only when you set the ROIProcessing property to true. The default value of this property is Rectangles.

ROIPortion

Calculate over entire ROI or just perimeter

Specify whether to calculate the mean over the Entire ROI or the ROI perimeter. This property applies only when you set the ROIForm property to Rectangles. The default value of this property is Entire ROI.

ROIStatistics

Calculate statistics for each ROI or one for all ROIs

Specify whether to calculate `Individual` statistics for each ROI or a `Single` statistic for all ROIs. This property applies only when you set the `ROIForm` property to `Rectangles`, `Lines`, or `Label matrix`. The default value of this property is `Individual statistics for each ROI`.

`ValidityOutputPort`

Output flag indicating if any part of ROI is outside input image

When you set the `ROIForm` property to `Lines` or `Rectangles`, set this property to `true` to return the validity of the specified ROI being completely inside of the image. When you set the `ROIForm` property to `Label Matrix`, set this property to `true` to return the validity of the specified label numbers. The default value of this property is `false`.

Fixed-Point Properties

`RoundingMethod`

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value of this property is `Floor`.

`OverflowAction`

Action to take when integer input is out-of-range

Specify the overflow action as `Wrap` or `Saturate`. The default value of this property is `Wrap`.

`AccumulatorDataType`

Data type of accumulator

Specify the accumulator fixed-point data type as `Same as input`, or `Custom`. The default value of this property is `Same as input`.

video.Mean class

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object. This property applies only when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

OutputDataType

Data type of output

Specify the output fixed-point data type as `Same as accumulator`, `Same as input`, or `Custom`. The default value of this property is `Same as accumulator`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numericType` object. This property applies only when you set the `OutputDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

Methods

<code>clone</code>	Create mean object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Return the number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties

reset	Reset computation of running mean
step	Compute mean of input

Examples

Determine the mean of a grayscale image.

```
img = im2single(rgb2gray(imread('peppers.png')));  
hmean = video.Mean;  
m = step(hmean,img);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Mean block reference page. The object properties correspond to the block parameters, except for:

- Only the block supports the parameter **Treat sample-based row input as column**.

See Also

[signalblks.Mean](#) | [video.Maximum](#) | [video.Minimum](#)

video.Mean.clone

Purpose Create mean object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a Mean object `C`, with the same property values as `H`. The clone method creates a new unlocked object with uninitialized states.

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.Mean.getNumOutputs

Purpose Return the number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the Mean System object..</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.Mean.reset

Purpose Reset computation of running mean

Syntax `reset(H)`

Description `reset(H)` resets the computation of the running mean for the Mean object H.

Purpose	Compute mean of input
Syntax	<pre>Y = step(H,X) Y = step(H,X,R) Y = step(H,X,ROI) Y = step(H,X,LABEL,LABELNUMBERS) [Y,FLAG] = step(H,X,ROI) [Y,FLAG] = step(H,X,LABEL,LABELNUMBERS)</pre>
Description	<p><code>Y = step(H,X)</code> computes the mean of <code>X</code>. When you set the <code>RunningMean</code> property to <code>true</code>, <code>Y</code> corresponds to the mean of the input elements over time.</p> <p><code>Y = step(H,X,R)</code> computes the mean value, <code>Y</code>, of the input elements over time, and optionally resets the computation of the running mean based on the value of reset signal, <code>R</code>, and the value of the <code>ResetCondition</code> property. To enable this type of processing, set the <code>RunningMean</code> property to <code>true</code> and the <code>ResetInputPort</code> property to <code>true</code>.</p> <p><code>Y = step(H,X,ROI)</code> computes the mean of input image <code>X</code> within the given region of interest <code>ROI</code>. To enable this type of processing, set the <code>ROIProcessing</code> property to <code>true</code> and the <code>ROIForm</code> property to <code>Lines</code>, <code>Rectangles</code> or <code>Binary mask</code>.</p> <p><code>Y = step(H,X,LABEL,LABELNUMBERS)</code> computes the mean of the input image, <code>X</code>, for the region whose labels are specified in the vector <code>LABELNUMBERS</code>. The regions are defined and labeled in the matrix <code>LABEL</code>. To enable this type of processing, set the <code>ROIProcessing</code> property to <code>true</code> and the <code>ROIForm</code> property to <code>Label matrix</code>.</p> <p><code>[Y,FLAG] = step(H,X,ROI)</code> also returns <code>FLAG</code>, indicating whether the given region of interest <code>ROI</code>, is within the image bounds. To enable this type of processing, set the the <code>ROIProcessing</code> and <code>ValidityOutputPort</code> properties to <code>true</code> and the <code>ROIForm</code> property to <code>Lines</code>, <code>Rectangles</code> or <code>Binary mask</code>.</p> <p><code>[Y,FLAG] = step(H,X,LABEL,LABELNUMBERS)</code> also returns <code>FLAG</code> which indicates whether the input label numbers are valid. To enable this</p>

video.Mean.step

type of processing, set the ROIProcessing and ValidityOutputPort properties to true and the ROIForm property to Label matrix.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose Find median values in an input.

Description The Median object finds median values in an input.

Construction `H = video.Median` returns a System object, H, that computes the median of the input or a sequence of inputs.

`H = video.Median('PropertyName',PropertyValue,...)` returns a median System object, H, with each specified property set to the specified value.

Properties `SortMethod`

Sort method

Specify the sort method used for calculating the median as `Quick sort` or `Insertion sort`.

`Dimension`

Dimension with which to operate along

Specify how the calculation is performed over the data as `All`, `Row`, `Column`, or `Custom`. The default value of this property is `All`

`CustomDimension`

Numerical dimension over which to calculate

Specify the integer dimension of the input signal over which the object calculates the mean. The value of this property cannot exceed the number of dimensions in the input signal. This property only applies when you set the `Dimension` property to `Custom`. The default value of this property is 1.

Fixed-Point Properties

`RoundingMethod`

Rounding method for fixed-point operations

video.Median class

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value of this property is `Floor`.

OverflowAction

Action to take when integer input is out-of-range

Specify the overflow action as `Wrap` or `Saturate`. The default value of this property is `Wrap`.

ProductDataType

Product output word and fraction lengths

Specify the product output fixed-point data type as `Same as input` or `Custom`. The default value of this property is `Same as input`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numericType` object. This property applies when you set the `ProductDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

AccumulatorDataType

Data type of accumulator

Specify the accumulator fixed-point data type as `Same as input`, or `Custom`. The default value of this property is `Same as input`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object. This property applies only when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

OutputDataType

Data type of output

Specify the output fixed-point data type as `Same as accumulator`, `Same as input`, or `Custom`. The default value of this property is `Same as accumulator`.

`CustomOutputDataType`

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numericType` object. This property applies only when you set the `OutputDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

Methods

<code>clone</code>	Create median object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Compute median of input

Examples

Determine the median in a grayscale image.

```
img = im2single(rgb2gray(imread('peppers.png')));  
hmdn = video.Median;  
med = step(hmdn,img);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Median block reference page. The object properties correspond to the block parameters.

video.Median class

See Also

`signalblks.Median`

Purpose Create median object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an Median System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.Median.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.Median.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the Median System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Compute median of input

Syntax $Y = \text{step}(H, X)$

Description $Y = \text{step}(H, X)$ computes median of input X .

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.MedianFilter2D class

Purpose 2D median filtering

Description The MedianFilter2D object performs 2D median filtering.

Construction `H = video.MedianFilter2D` returns a 2D median filtering object, H, that performs two-dimensional median filtering of an input matrix.

`H = video.MedianFilter2D('PropertyName',PropertyValue, ...)` returns a median filter object, H, with each property set to the specified value.

`H = video.MedianFilter2D(SIZE,'PropertyName', PropertyValue, ...)` returns a median filter object, H, with the NeighborhoodSize property set to SIZE and other properties set to the specified values.

Properties NeighborhoodSize

Size of neighborhood to compute the median

Specify the size of the neighborhood over which the median filter computes the median. This property is either a positive integer that represents the number of rows and columns in a square matrix, or a two-element vector that represents the number of rows and columns in a rectangular matrix. The median filter does not support fixed-point properties when both neighborhood dimensions are odd. This property defaults to [3 3].

OutputSize

Output matrix size

Specify the output size as Same as input size or Valid. If the property value is Valid, the 2D median filter only computes the median where the neighborhood fits entirely within the input image and requires no padding. In this case, the dimensions of the output image are:

$$\text{output rows} = \text{input rows} - \text{neighborhood rows} + 1$$


```
output columns = input columns - neighborhood
columns + 1
```

Otherwise, the output has the same dimensions as the input image. The default value for this property is Same as input size.

PaddingMethod

Input matrix boundary padding method

Specifies how to pad the boundary of the input matrix as Constant, Replicate, Symmetric, or Circular. Set this property to Constant to pad the matrix with a constant value, Replicate to pad the input matrix by repeating its border values, Symmetric to pad the input matrix with its mirror image, and Circular to pad the input matrix using a circular repetition of its elements. This property applies only when the OutputSize property is Same as input size. The default value for this property is Constant.

PaddingValueSource

Source of constant boundary value

Specifies how to define the constant boundary value as Property or Input port. This property applies only when the PaddingMethod property is Constant. The default value for this property is Property.

PaddingValue

Constant padding value for input matrix

Specifies a constant value to pad the input matrix. This property applies only when the PaddingMethod property is Constant and the PaddingValueSource property is Property. The default value of this property is 0. This property is tunable.

Fixed-Point Properties

RoundingMethod

video.MedianFilter2D class

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. This property applies only when the `NeighborhoodSize` property corresponds to even neighborhood options. The default value for this property is `Floor`.

`OverflowAction`

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. This property is applied only when the `NeighborhoodSize` property corresponds to even neighborhood options. The default value for this property is `Wrap`.

`AccumulatorDataType`

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as input` or `Custom`. This property applies only when the `NeighborhoodSize` property corresponds to even neighborhood options. The default value for this property is `Same as input`.

`CustomAccumulatorDataType`

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies only when the `AccumulatorDataType` property is `Custom` and the `NeighborhoodSize` property corresponds to even neighborhood options. The default value of this property is `numericType([],32,30)`.

`OutputDataType`

Output word and fraction lengths

Specify the output fixed-point data type as `Same as input` or `Custom`. This property applies only when the `NeighborhoodSize`

property corresponds to even neighborhood options. The default value for this property is Same as input.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled numeric type object with a Signedness of Auto. This property applies only when the OutputDataType property is Custom and the NeighborhoodSize property corresponds to even neighborhood options. The default value of this property is numeric type ([], 16, 15).

Methods

clone	Create 2D median filter with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Perform median filtering on input image

Examples

Perform median filtering on an image with additive salt and pepper noise:

```
img = im2single(rgb2gray(imread('peppers.png')));  
img = imnoise(img, 'salt & pepper'); % add some noise  
imshow(img);  
hmedianfilt = video.MedianFilter2D([5 5]);  
filtered = step(hmedianfilt, img);  
pause(2);  
imshow(filtered);
```

video.MedianFilter2D class

Algorithm

This object implements the algorithm, inputs, and outputs described on the Median Filter block reference page. The object properties correspond to the block parameters.

See Also

`video.ImageFilter`

Purpose Create 2D median filter with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a `MedianFilter2D` System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.MedianFilter2D.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.MedianFilter2D.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.MedianFilter2D.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the MedianFilter2D System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Perform median filtering on input image

Syntax
`I2 = step(H, I1)`
`I2 = step(H, I1, PVAL)`

Description `I2 = step(H, I1)` performs median filtering on the input image `I1` and returns the filtered image `I2`.

`I2 = step(H, I1, PVAL)` performs median filtering on input image `I1`, using `PVAL` for the padding value. This option applies when you set the `OutputSize` property to `Same as input size`, the `PaddingChoice` property to `Constant`, and the `PaddingValueSource` property to `Input port`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.Minimum class

Purpose	Find minimum values in input or sequence of inputs
Description	The Minimum object finds minimum values in an input or sequence of inputs.
Construction	<p>H = video.Minimum returns an object, H, that computes the value and index of the minimum elements in an input or a sequence of inputs.</p> <p>H = video.Minimum('PropertyName',PropertyValue,...) returns a minimum-finding object, H, with each specified property set to the specified value.</p>
Properties	<p>ValueOutputPort</p> <p>Output minimum value</p> <p>Set this property to true to output the minimum (when RunningMinimum is false) or the running minimum (when RunningMinimum is true). The default value of this property is true.</p> <p>RunningMinimum</p> <p>Calculate over single input or multiple inputs</p> <p>When you set this property to true, the object computes the minimum value over a sequence of inputs. When you set this property to false, the object computes the minimum value over the current input. The default value of this property is false.</p> <p>IndexOutputPort</p> <p>Output the index of the minimum value</p> <p>Set this property to true to output the index of the minimum value of the input. This property applies only when you set the RunningMinimum property to false. The default value of this property is true.</p> <p>ResetInputPort</p> <p>Additional input to enable resetting of running minimum</p>

Set this property to true to enable resetting of the running minimum. When you set this property to true, a reset input must be specified to the `step` method to reset the running minimum. This property applies only when you set the `RunningMinimum` property to true. The default value of this property is false.

`ResetCondition`

Condition that triggers resetting of running minimum

Specify the event that resets the running minimum as `Rising edge`, `Falling edge`, `Either edge`, or `Non-zero`. This property applies only when you set the `ResetInputPort` property to true. The default value of this property is `Non-zero`.

`IndexBase`

Numbering base for index of minimum value

Specify the numbering used when computing the index of the minimum value as starting from either `One` or `Zero`. This property applies only when you set the `IndexOutputPort` property to true. The default value of this property is `One`.

`Dimension`

Dimension to operate along

Specify how the minimum calculation is performed over the data as `All`, `Row`, `Column`, or `Custom`. This property applies only when you set the `RunningMinimum` property to false. The default value of this property is `Column`.

`CustomDimension`

Numerical dimension to calculate over

Specify the integer dimension of the input signal over which the object finds the minimum. The value of this property cannot exceed the number of dimensions in the input signal. This property only applies when you set the `Dimension` property to `Custom`. The default value of this property is 1.

ROIProcessing

Enable region-of-interest processing

Set this property to `true` to enable calculation of the minimum value within a particular region of an image. This property applies when you set the `Dimension` property to `All` and the `RunningMinimum` property to `false`. The default value of this property is `false`.

ROIForm

Type of region of interest

Specify the type of region of interest as `Rectangles`, `Lines`, `Label matrix`, or `Binary mask`. This property applies only when you set the `ROIProcessing` property to `true`. The default value of this property is `Rectangles`.

ROIPortion

Calculate over entire ROI or just perimeter

Specify whether to calculate the minimum over the `Entire ROI` or the `ROI perimeter`. This property applies only when you set the `ROIForm` property to `Rectangles`. The default value of this property is `Entire ROI`.

ROIStatistics

Calculate statistics for each ROI or one for all ROIs

Specify whether to calculate `Individual statistics` for each ROI or a `Single statistic` for all ROIs. This property applies only when you set the `ROIForm` property to `Rectangles`, `Lines`, or `Label matrix`.

ValidityOutputPort

Output flag indicating if any part of ROI is outside input image

When you set the `ROIForm` property to `Lines` or `Rectangles`, set this property to `true` to return the validity of the specified ROI being completely inside of the image. When you set the `ROIForm`

property to `Label Matrix`, set this property to `true` to return the validity of the specified label numbers. The default value of this property is `false`.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value of this property is `Floor`.

OverflowAction

Action to take when integer input is out-of-range

Specify the overflow action as `Wrap` or `Saturate`. The default value of this property is `Wrap`.

ProductDataType

Data type of product

Specify the product fixed-point data type as `Same as input` or `Custom`. The default value of this property is `Same as input`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numericType` object. This property applies only when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

AccumulatorDataType

Data type of accumulator

Specify the accumulator fixed-point data type as `Same as product`, `Same as input`, or `Custom`. The default value of this property is `Same as product`.

video.Minimum class

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numerictype` object. This property applies only when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numerictype(true,32,30)`.

Methods

<code>clone</code>	Create minimum object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>reset</code>	Reset computation of running minimum
<code>step</code>	Compute minimum value

Examples

Determine the minimum value and its index in a grayscale image.

```
img = im2single(rgb2gray(imread('peppers.png')));  
hmax = video.Minimum;  
[m, ind] = step(hmax, img);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the `Minimum` block reference page. The object properties correspond to the block parameters, except for:

- Only the block supports the **Treat sample-based row input as column** parameter.

See Also [signalblks.Maximum](#) | [video.Maximum](#) | [video.Mean](#)

video.Minimum.clone

Purpose Create minimum object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a Minimum object C, with the same property values as H. The clone method creates a new unlocked object with uninitialized states.

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.Minimum.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the Minimum System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.Minimum.reset

Purpose	Reset computation of running minimum
Syntax	<code>reset(H)</code>
Description	<code>reset(H)</code> resets computation of the running minimum for the <code>Minimum</code> object <code>H</code> .

Purpose

Compute minimum value

Syntax

```
[VAL,IND] = step(H,X)
VAL = step(H,X)
IND = step(H,X)
VAL = step(H,X,R)
[...] = step(H,I,ROI)
[...] = step(H,I,LABEL,LABELNUMBERS)
[... ,FLAG] = step(H,I,ROI)
[... ,FLAG] = step(H,I,LABEL,LABELNUMBERS)
```

Description

`[VAL,IND] = step(H,X)` returns the minimum value, `VAL`, and the index or position of the minimum value, `IND`, along a dimension of `X` specified by the value of the `Dimension` property.

`VAL = step(H,X)` returns the minimum value, `VAL`, of the input `X`. When the `RunningMinimum` property is true, `VAL` corresponds to the minimum value over a sequence of inputs.

`IND = step(H,X)` returns the zero- or one-based index `IND` of the minimum value. To enable this type of processing, set the `IndexOutputPort` property to true and the `ValueOutputPort` and `RunningMinimum` properties to false.

`VAL = step(H,X,R)` computes the minimum value, `VAL`, over a sequence of inputs, and resets the state of `H` based on the value of reset signal, `R`, and the `ResetCondition` property. To enable this type of processing, set the `RunningMinimum` property to true and the `ResetInputPort` property to true.

`[...] = step(H,I,ROI)` computes the minimum of an input image, `I`, within the given region of interest, `ROI`. To enable this type of processing, set the `ROIProcessing` property to true and the `ROIForm` property to `Lines`, `Rectangles` or `Binary mask`.

`[...] = step(H,I,LABEL,LABELNUMBERS)` computes the minimum of an input image, `I`, for a region whose labels are specified in the vector `LABELNUMBERS`. To enable this type of processing, set the `ROIProcessing` property to true and the `ROIForm` property to `Label matrix`.

[..., FLAG] = step(H,I,ROI) also returns FLAG, indicating whether the given region of interest is within the image bounds. To enable this type of processing, set the ROIProcessing and ValidityOutputPort properties to true and the ROIForm property to Lines, Rectangles or Binary mask.

[..., FLAG] = step(H,I,LABEL,LABELNUMBERS) also returns FLAG, indicating whether the input label numbers are valid. To enable this type of processing, set the ROIProcessing and ValidityOutputPort properties to true and the ROIForm property to Label matrix.

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.MorphologicalBottomHat class

Purpose	Bottom-hat filtering on image
Description	The MorphologicalBottomHat object performs bottom-hat filtering on an intensity or binary image. Bottom-hat filtering is the equivalent of subtracting the input image from the result of performing a morphological closing operation on the input image. The bottom-hat filtering object uses flat structuring elements only.
Construction	<p>H = video.MorphologicalBottomHat returns a bottom-hat filtering object, H, that performs bottom-hat filtering on an intensity or binary image using a predefined neighborhood or structuring element.</p> <p>H = video.MorphologicalBottomHat(<i>PropertyName</i>, <i>PropertyValue</i>, ...) returns a bottom-hat filtering object, H, with each property set to the specified value.</p>
Properties	<p>ImageType</p> <p>Specify type of input image or video stream</p> <p>Specify the type of the input image as Intensity or Binary. The default value of this property is Intensity.</p> <p>NeighborhoodSource</p> <p>Source of neighborhood values</p> <p>Specify how to enter neighborhood or structuring element values as one of Property or Input port. If set to Property, use the Neighborhood property to specify the neighborhood or structuring element values. Otherwise, specify the neighborhood using an input to the step method. Note that you can specify structuring elements only by using the Neighborhood property. You can not specify structuring elements as inputs to the step method. The default value of this property is Property.</p> <p>Neighborhood</p> <p>Neighborhood or structuring element values</p>

video.MorphologicalBottomHat class

This property applies only when you set the `NeighborhoodSource` property to `Property`. If specifying a neighborhood, this property must be a matrix or vector of 1s and 0s. If specifying a structuring element, use the `strel` function. The default value of this property is `strel('octagon',15)`.

Methods

<code>clone</code>	Create morphological bottom hat filter with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Perform bottom-hat filtering on input image

Examples

Perform bottom-hat filtering on an image.

```
I = im2single(imread('blobs.png'));  
hbot = video.MorphologicalBottomHat('Neighborhood',...  
strel('disk', 5));  
J = step(hbot,I);  
figure;  
subplot(1,2,1),imshow(I); title('Original image');  
subplot(1,2,2),imshow(J);  
title('Bottom-hat filtered image');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Bottom-hat block reference page. The object properties correspond to the block parameters.

video.MorphologicalBottomHat class

See Also

[strel](#) | [video.MorphologicalTopHat](#) | [video.MorphologicalClose](#)

video.MorphologicalBottomHat.clone

Purpose Create morphological bottom hat filter with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an instance of the current morphological bottom hat object with the same property values. The `clone` method creates a new unlocked object.

video.MorphologicalBottomHat.getNumInputs

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.MorphologicalBottomHat.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.MorphologicalBottomHat.isLocked

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the MorphologicalBottomHat System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.MorphologicalBottomHat.step

Purpose Perform bottom-hat filtering on input image

Syntax `Y = step(H,I)`
`Y = step(H,I,NHOOD)`

Description `Y = step(H,I)` performs bottom-hat filtering on the input image, `I`, and returns the filtered image `Y`.

`Y = step(H,I,NHOOD)` performs bottom-hat filtering on the input image, `I` using `NHOOD` as the neighborhood when you set the `NeighborhoodSource` property to `Input` port. The object returns the filtered image in the output `Y`.

Purpose	Perform morphological closing on image
Description	The MorphologicalClose object performs morphological closing on an intensity or binary image. The MorphologicalClose System object performs a dilation operation followed by an erosion operation using a predefined neighborhood or structuring element. This System object uses flat structuring elements only.
Construction	<p>H = video.MorphologicalClose returns a System object, H, that performs morphological closing on an intensity or binary image.</p> <p>H = video.MorphologicalClose('PropertyName',PropertyValue,...) returns a morphological closing System object, H, with each specified property set to the specified value.</p>
Properties	<p>NeighborhoodSource</p> <p>Source of neighborhood values</p> <p>Specify how to enter neighborhood or structuring element values as Property or Input port. If set to Property, use the Neighborhood property to specify the neighborhood or structuring element values. Otherwise, specify the neighborhood using an input to the step method. Note that structuring elements can only be specified using Neighborhood property and they cannot be used as input to the step method. The default value for this property is Property.</p> <p>Neighborhood</p> <p>Neighborhood or structuring element values</p> <p>This property is applicable when the NeighborhoodSource property is set to Property. If you are specifying a neighborhood, this property must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the strel function. The default value of this property is strel('line',5,45).</p>

video.MorphologicalClose class

Methods

<code>clone</code>	Create morphological close object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Perform morphological closing on input image

Examples

Perform morphological closing on an image.

```
img = im2single(imread('blobs.png'));  
hclosing = video.MorphologicalClose;  
hclosing.Neighborhood = strel('disk', 10);  
closed = step(hclosing, img);  
figure;  
subplot(1,2,1),imshow(img); title('Original image');  
subplot(1,2,2),imshow(closed); title('Closed image');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Closing block reference page. The object properties correspond to the block parameters.

See Also

[video.MorphologicalOpen](#) | [video.ConnectedComponentLabeler](#) | [video.Autothresher](#) | [strel](#)

Purpose Create morphological close object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an `MorphologicalClose System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.MorphologicalClose.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.MorphologicalClose.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.MorphologicalClose.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the MorphologicalClose System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose

Perform morphological closing on input image

Syntax

IC = step(H,I)
IC = step(H,I,NHOOD)

Description

IC = step(H,I) performs morphological closing on input image I.

IC = step(H,I,NHOOD) performs morphological closing on input image I using input NHOOD as the neighborhood when the NeighborhoodSource property is set to Input port.

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.MorphologicalDilate class

Purpose Perform morphological dilation on an image

Description The MorphologicalDilate object performs morphological dilation on an image.

Construction `H = video.MorphologicalDilate` returns a System object, H, that performs morphological dilation on an intensity or binary image.

`H = video.MorphologicalDilate('PropertyName',PropertyValue,...)` returns a morphological dilation System object, H, with each specified property set to the specified value.

Properties NeighborhoodSource

Source of neighborhood values

Specify how to enter neighborhood or structuring element values as Property or Input port. If set to Property, use the Neighborhood property to specify the neighborhood or structuring element values. Otherwise, specify the neighborhood using an input to the step method. Note that structuring elements can only be specified using Neighborhood property and they cannot be used as input to the step method. The default value for this property is Property.

Neighborhood

Neighborhood or structuring element values

This property is applicable when the NeighborhoodSource property is set to Property. If you are specifying a neighborhood, this property must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the strel function. The default value of this property is `[1 1; 1 1]`.

Methods

clone	Create morphological dilate object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Operate on inputs to calculate outputs

Examples

Fuse fine discontinuities on images.

```
x = imread('peppers.png');
hcsc = video.ColorSpaceConverter;
hcsc.Conversion = 'RGB to intensity';
hautothresh = video.Autothresher;
hdilate = ...
    video.MorphologicalDilate('Neighborhood', ones(5,5));
x1 = step(hcsc, x);
x2 = step(hautothresh, x1);
y = step(hdilate, x2);
figure;
subplot(3,1,1),imshow(x); title('Original image');
subplot(3,1,2),imshow(x2); title('Thresholded Image');
subplot(3,1,3),imshow(y); title('Dilated Image');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Dilation block reference page. The object properties correspond to the block parameters.

video.MorphologicalDilate class

See Also

`video.MorphologicalErode` | `video.MorphologicalOpen` |
`video.MorphologicalClose` | `strel`

Purpose Create morphological dilate object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a MorphologicalDilate System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.MorphologicalDilate.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.MorphologicalDilate.getNumOutputs

Purpose

Number of outputs from step method

Syntax

`N = getNumOutputs(H)`

Description

`N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.MorphologicalDilate.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the MorphologicalDilate System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose

Operate on inputs to calculate outputs

Syntax

ID = step(H,I)
ID = step(H,I,NHOOD)

Description

ID = step(H,I) performs morphological dilation on input image I and returns the dilated image IE.

ID = step(H,I,NHOOD) uses input NHOOD as the neighborhood when the NeighborhoodSource property is set to Input port.

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.MorphologicalErode class

Purpose Perform morphological erosion on an image

Description The MorphologicalErode object performs morphological erosion on an image using a neighborhood specified by a square structuring element of width 4.

Construction H = video.MorphologicalErode returns a System object, H, that performs morphological erosion on an intensity or binary image.

H =
video.MorphologicalErode('PropertyName',PropertyValue,...)
returns a morphological erosion System object, H, with each specified property set to the specified value.

Properties NeighborhoodSource

Source of neighborhood values

Specify how to enter neighborhood or structuring element values as Property or Input port. If set to Property, use the Neighborhood property to specify the neighborhood or structuring element values. Otherwise, specify the neighborhood using an input to the step method. Note that structuring elements can only be specified using Neighborhood property and they cannot be used as input to the step method. The default value for this property is Property.

Neighborhood

Neighborhood or structuring element values

This property is applicable when the NeighborhoodSource property is set to Property. If you are specifying a neighborhood, this property must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the strel function. The default value of this property is strel('square',4).

Methods

clone	Create morphological erode object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Perform morphological erosion on input

Examples

Erode an input image.

```
x = imread('peppers.png');
hcsc = video.ColorSpaceConverter;
hcsc.Conversion = 'RGB to intensity';
hautothresh = video.Autothresher;
herode = ...
    video.MorphologicalErode('Neighborhood', ones(5,5));
x1 = step(hcsc, x); % convert input to intensity
x2 = step(hautothresh, x1); % convert input to binary
y = step(herode, x2); % Perform erosion on input
figure;
subplot(3,1,1),imshow(x); title('Original image');
subplot(3,1,2),imshow(x2); title('Thresholded Image');
subplot(3,1,3),imshow(y); title('Eroded Image');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Erosion block reference page. The object properties correspond to the block parameters.

video.MorphologicalErode class

See Also

`video.MorphologicalDilate` | `video.MorphologicalOpen` |
`video.MorphologicalClose` | `strel`

Purpose Create morphological erode object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an MorphologicalErode System object C, with the same property values as H. The `clone` method creates a new unlocked object.

video.MorphologicalErode.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.MorphologicalErode.getNumOutputs

Purpose

Number of outputs from step method

Syntax

`N = getNumOutputs(H)`

Description

`N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.MorphologicalErode.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the MorphologicalErode System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Perform morphological erosion on input

Syntax IE = step(H,I)
IE = step(H,I,NHOOD)

Description IE = step(H,I) performs morphological erosion on input image I and returns the eroded image IE.
IE = step(H,I,NHOOD) uses input NHOOD as the neighborhood when the NeighborhoodSource property is set to Input port.

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.MorphologicalOpen class

Purpose Perform morphological opening on an image

Description The MorphologicalOpen object performs morphological opening on an image. The MorphologicalOpen System object performs an erosion operation followed by a dilation operation using a predefined neighborhood or structuring element. This System object uses flat structuring elements only. For more information about structuring elements, see the `strel` function reference page in the Image Processing Toolbox documentation.

Construction `H = video.MorphologicalOpen` returns a System object, H, that performs morphological opening on an intensity or binary image.

`H = video.MorphologicalOpen('PropertyName',PropertyValue,...)` returns a morphological opening System object, H, with each specified property set to the specified value.

Properties **NeighborhoodSource**
Source of neighborhood values

Specify how to enter neighborhood or structuring element values as `Property` or `Input port`. If set to `Property`, use the `Neighborhood` property to specify the neighborhood or structuring element values. Otherwise, specify the neighborhood using an input to the `step` method. Note that structuring elements can only be specified using `Neighborhood` property and they cannot be used as input to the `step` method. The default value for this property is `Property`.

Neighborhood
Neighborhood or structuring element values

This property applies when you set the `NeighborhoodSource` property to `Property`. If you are specifying a neighborhood, this property must be a matrix or vector of 1s and 0s. If you are

specifying a structuring element, use the `strel` function. The default value of this property is `strel('disk',5)`.

Methods

<code>clone</code>	Create morphological open object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Perform morphological opening on input image

Examples

Perform opening on an image

```
img = im2single(imread('blobs.png'));
hopening = video.MorphologicalOpen;
hopening.Neighborhood = strel('disk', 5);
opened = step(hopening, img);
figure;
subplot(1,2,1),imshow(img); title('Original image');
subplot(1,2,2),imshow(opened); title('Opened image');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Opening block reference page. The object properties correspond to the block parameters.

See Also

[video.MorphologicalClose](#) | [video.ConnectedComponentLabeler](#) | [video.Autothresher](#) | [strel](#)

video.MorphologicalOpen.clone

Purpose Create morphological open object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an `MorphologicalOpen System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.MorphologicalOpen.getNumInputs

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.MorphologicalOpen.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the MorphologicalOpen System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.MorphologicalOpen.step

Purpose Perform morphological opening on input image

Syntax
IO = step(H,I)
IO = step(H,I,NHOOD)

Description IO = step(H,I) performs morphological opening on binary or intensity input image I.
IO = step(H,I,NHOOD) uses input NHOOD as the neighborhood when the NeighborhoodSource property is set to Input port.

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Top-hat filtering on image
Description	The MorphologicalTopHat object performs top-hat filtering on an intensity or binary image. Top-hat filtering is the equivalent of subtracting the result of performing a morphological opening operation on the input image from the input image itself. This top-hat filtering object uses flat structuring elements only.
Construction	<p>H = video.MorphologicalTopHat returns a top-hat filtering object, H, that performs top-hat filtering on an intensity or binary image using a predefined neighborhood or structuring element.</p> <p>H = video.MorphologicalTopHat(<i>PropertyName</i>, <i>PropertyValue</i>, ...) returns a top-hat filtering object, H, with each property set to the specified value.</p>
Properties	<p>ImageType</p> <p>Specify type of input image or video stream</p> <p>Specify the type of input image or video stream as Intensity or Binary. The default value of this property is Intensity.</p> <p>NeighborhoodSource</p> <p>Source of neighborhood values</p> <p>Specify how to enter neighborhood or structuring element values as Property or Input port. If set to Property, use the Neighborhood property to specify the neighborhood or structuring element values. Otherwise, specify the neighborhood using an input to the step method. Note that structuring elements can only be specified using the Neighborhood property. You cannot use the structuring elements as an input to the step method. The default value of this property is Property.</p> <p>Neighborhood</p> <p>Neighborhood or structuring element values</p>

video.MorphologicalTopHat class

This property applies only when you set the `NeighborhoodSource` property to `Property`. If specifying a neighborhood, this property must be a matrix or vector of 1s and 0s. If specifying a structuring element, use the `strel` function. The default value of this property is `strel('square',4)`.

Methods

<code>clone</code>	Create morphological top hat filter with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Perform top-hat filtering on input image

Examples

Perform top-hat filtering to correct uneven illumination.

```
I = im2single(imread('rice.png'));
htop = video.MorphologicalTopHat('Neighborhood', ...
    strel('disk', 12));
% To improve contrast of output image
hc = video.ContrastAdjuster; J = step(htop,I);
J = step(hc,J);
figure;
subplot(1,2,1),imshow(I); title('Original image');
subplot(1,2,2),imshow(J);
title('Top-hat filtered image');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Top-hat block reference page. The object properties correspond to the block parameters.

video.MorphologicalTopHat class

See Also

[strel](#) | [video.MorphologicalBottomHat](#) | [video.MorphologicalOpen](#)

video.MorphologicalTopHat.clone

Purpose	Create morphological top hat filter with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates an instance of the current morphological top hat object with the same property values. The <code>clone</code> method creates a new unlocked object

video.MorphologicalTopHat.getNumInputs

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.MorphologicalTopHat.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the MorphologicalTopHat System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.MorphologicalTopHat.step

Purpose Perform top-hat filtering on input image

Syntax
`Y = step(H,I)`
`Y = step(H,I,NHOOD)`

Description `Y = step(H,I)` top-hat filters the input image, `I`, and returns the filtered image `Y`.
`Y = step(H,I,NHOOD)` filters the input image, `I` using the input `NHOOD` as the neighborhood when you set the `NeighborhoodSource` property to `Input` port. The object returns the filtered image in the output `Y`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.MultimediaFileReader class

Purpose	Read video and/or audio samples from multimedia file
Description	The <code>MultimediaFileReader</code> object reads video and/or audio samples from a multimedia file.
Construction	<p><code>H = video.MultimediaFileReader</code> returns a multimedia file reader System object, <code>H</code>, to read video and/or audio from a multimedia file.</p> <p><code>H = video.MultimediaFileReader('PropertyName',PropertyValue,...)</code> returns a multimedia file reader System object, <code>H</code>, with each specified property set to the specified value.</p> <p><code>H = video.MultimediaFileReader(FILENAME,'PropertyName',PropertyValue,...)</code> returns a multimedia file reader System object, <code>H</code>, with <code>Filename</code> property set to <code>FILENAME</code> and other specified properties set to the specified values.</p>
Properties	<p>Filename</p> <p>Name of multimedia file from which to read</p> <p>Specify the name of the multimedia file as a string. The full path for the file needs to be specified only if the file is not on the MATLAB path. On UNIX[®] platforms, the System object only supports uncompressed AVI files. The default value of this property is <code>vipmen.avi</code>.</p> <p>PlayCount</p> <p>Number of times to play the file</p> <p>Specify a positive integer or <code>inf</code> to represent the number of times to play the file. The default value of this property is <code>inf</code>.</p> <p>AudioOutputPort</p> <p>Choose to output audio data</p>

video.MultimediaFileReader class

Use this property to control the audio output from the multimedia file reader. This property applies only when the multimedia file contains both audio and video streams. The default value of this property is `false`.

VideoOutputPort

Choose to output video data

Use this property to control the video output from the multimedia file reader object. This property only applies when the file contains both audio and video streams. The default value of this property is `false`.

SamplesPerAudioFrame

Number of samples in audio frame

Specify the number of samples in an audio frame as a positive scalar integer value. This property applies when the multimedia file contains only audio data. The default value of this property is 1024.

ImageColorSpace

Choose whether output is RGB, YCbCr, or intensity video

Specify whether you want the multimedia file reader object to output RGB, YCbCr 4:2:2 or intensity video frames. This property applies only when the multimedia file contains video. This property can be set to `RGB`, `Intensity`, or `YCbCr 4:2:2`. The default value of this property is `RGB`.

VideoOutputDataType

Data type of video data output

Set the data type of the video data output from the multimedia file reader object. This property applies if the multimedia file contains video. This property can be set to `double`, `single`, `int8`, `uint8`, `int16`, `uint16`, `int32`, or `Inherit`. The default value of this property is `single`.

AudioOutputDataType

Data type of audio samples output

Set the data type of the audio data output from the multimedia file reader object. This property applies only if the multimedia file contains audio. This property can be set to `double`, `single`, `int16`, or `uint8`. The default value of this property is `int16`.

Methods

<code>clone</code>	Create multimedia file reader object with same property values
<code>close</code>	Release resources for the multimedia file reader object
<code>getNumInputs</code>	Number of expected inputs to <code>step</code> method
<code>getNumOutputs</code>	Number of outputs from <code>step</code> method
<code>info</code>	Return information about the specified multimedia file
<code>isDone</code>	End-of-file status (logical)
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>reset</code>	Reset internal states of multimedia file reader to read from beginning of file
<code>step</code>	Output frame of multimedia signal

Examples

Read and play a video file.

```
hmfr = video.MultimediaFileReader;  
hp = video.VideoPlayer;
```

video.MultimediaFileReader class

```
while ~isDone(hmfr)
    videoFrame = step(hmfr);
    step(hp,videoFrame);
end
close(hp);
close(hmfr);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the From Multimedia File block reference page. The object properties correspond to the x block parameters, except for:

- The object has no corresponding property for the **Inherit sample time from file** block parameter. The object always inherits the sample time from the file.
- The **Multimedia outputs** block parameter corresponds to both the AudioOutputPort and the VideoOutputPort object properties.
- The **Image signal** block parameter allows you to specify whether the block outputs the image as `One multidimensional signal` or `Separate color signals`. The object does not have a property that corresponds to the **Image signal** block parameter. The object always outputs the image as an M -by- N -by- P color video signal.

See Also

video.MultimediaFileWriter

Purpose	Create multimedia file reader object with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates a <code>MultimediaFileReader System</code> object <code>C</code> , with the same property values as <code>H</code> . The clone method creates a new unlocked object with uninitialized states.

video.MultimediaFileReader.close

Purpose Release resources for the multimedia file reader object

Syntax `close(H)`

Description `close(H)` releases system resources (such as memory, file handles or hardware connections).

video.MultimediaFileReader.getNumInputs

Purpose Number of expected inputs to step method

Syntax N = getNumInputs(H)

Description N = getNumInputs(H) returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.MultimediaFileReader.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

Purpose

Return information about the specified multimedia file

Syntax

`S = info(H)`

Description

`S = info(H)` returns a MATLAB structure, `S`, with information about the multimedia file specified in the `Filename` property. The number of fields of `S` varies depending on the audio/video content of the file. The possible fields and values for the structure `S` are described below:

<code>Audio</code>	Logical value indicating if the file has audio content.
<code>Video</code>	Logical value indicating if the file has video content.
<code>AudioSampleRate</code>	Audio sampling rate of the multimedia file in Hz. This field applies when the file has audio content.
<code>AudioNumBits</code>	Number of bits used to encode the audio stream. This field applies when the file has audio content.
<code>AudioNumChannels</code>	Number of audio channels. This field applies when the file has audio content.
<code>FrameRate</code>	Frame rate of the video stream in frames per second. The value may vary from the actual frame rate of the recorded video, and takes into consideration any synchronization issues between audio and video streams when the file contains both audio and video content. This implies that video frames may be dropped if the audio stream leads the video stream by more than $1/(\text{actual video frames per second})$. This field applies when the file has video content.
<code>VideoSize</code>	Video size as a two-element numeric vector of the form: <code>[VideoWidthInPixels, VideoHeightInPixels]</code> This field applies when the file has video content.
<code>VideoFormat</code>	Video signal format. This field applies when the file has video content.

video.MultimediaFileReader.isDone

Purpose	End-of-file status (logical)
Syntax	TF = isDone(H)
Description	TF = isDone(H) returns a logical value, STATUS , indicating if the MultimediaFileReader System object, H , has reached the end of the multimedia file. STATUS remains true when you set the PlayCount property to a value greater than 1.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the MultimediaFileReader System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

video.MultimediaFileReader.reset

Purpose	Reset internal states of multimedia file reader to read from beginning of file
Syntax	<code>reset(H)</code>
Description	<code>reset(H)</code> resets the <code>MultimediaFileReader</code> object to read from the beginning of the file.

Purpose

Output frame of multimedia signal

Syntax

```
AUDIO = step(H)
I = step(H)
[I,AUDIO] = step(H)
[... ,EOF] = step(H)
[Y,CB,CR] = step(H)
[Y,CB,CR,AUDIO] = step(H)
```

Description

`AUDIO = step(H)` outputs one frame of audio samples, `AUDIO`. This behavior requires an input file which contains audio data and that you set the `AudioOutputPort` property to `true`.

`I = step(H)` outputs one frame of multidimensional video signal, `I`. This behavior requires an input file which contains video data and that you set the `VideoOutputPort` property to `true`.

`[I,AUDIO] = step(H)` outputs one frame of multidimensional video signal, `I`, and one frame of audio samples, `AUDIO`. This behavior requires an input file which contains audio and video data and that you set the `AudioOutputPort` and `VideoOutputPort` properties to `true`.

`[... ,EOF] = step(H)` returns the end-of-file indicator in `EOF`. The object sets `EOF` to `true` each time the output contains the last audio sample and/or video frame in the file.

`[Y,CB,CR] = step(H)` outputs one frame of YCbCr 4:2:2 video data in the color components `Y`, `CB`, and `CR`. This behavior applies when you set the `VideoOutputPort` property to `true`, the `ImageColorSpace` property to `YCbCr 4:2:2`, and an input file which contains video data.

`[Y,CB,CR,AUDIO] = step(H)` outputs one frame of YCbCr 4:2:2 video data in the color components `Y`, `CB`, and `CR`, and one frame of audio samples in `AUDIO`. This applies when you set the `AudioOutputPort` and `VideoOutputPort` properties to `true`, the `ImageColorSpace` property to `YCbCr 4:2:2`, and an input file which contains audio and video data.

video.MultimediaFileReader.step

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.MultimediaFileWriter class

Purpose

Write video frames and audio samples to multimedia file

Description

The `MultimediaFileWriter` object writes video and/or audio samples to a multimedia file.

Construction

`H = video.MultimediaFileWriter` returns a multimedia file writer System object, `H`, that writes uncompressed video frames to an AVI file, `output.avi`.

`H =`

`video.MultimediaFileWriter('PropertyName',PropertyValue,...)` returns a multimedia file writer System object, `H`, with each specified property set to the specified value.

`H =`

`video.MultimediaFileWriter(FILENAME,'PropertyName',PropertyValue,...)` returns a multimedia file writer System object, `H`, with `Filename` property set to `FILENAME` and other specified properties set to the specified values.

Properties

`Filename`

Multimedia output file name

Specify the name of the multimedia file as a string. The default value of this property is `output.avi`.

`FileFormat`

Format of output file

Specify the format of the file that is created. On Windows® platforms, this may be one of AVI, WAV, WMV, or WMA. On other platforms, this value is restricted to AVI. The default value for this property is AVI. These abbreviations correspond to the following file formats:

WAV: Microsoft® WAVE Files

WMV: Windows Media Video

WMA: Windows Media Audio

video.MultimediaFileWriter class

AVI: Audio-Video Interleave

AudioInputPort

Write audio data

Use this property to control whether the object writes audio samples to the multimedia file. When both this property and the `VideoInputPort` property are enabled, the video and audio input signals must have the same frame period. The frame size (or number of rows) of the audio signal might need to be adjusted so that the frame period of the video signal is the same as the frame period of the audio signal. To calculate the frame size, divide the frequency of the audio signal (in samples per second specified by the `SampleRate` property) by the frame rate of the video signal (in frames per second specified by the `FrameRate` property). The default value for this property is `false`.

The multimedia file object takes a column vector as an input. Every column is a separate channel and each row corresponds to a single audio sample.

VideoInputPort

Write video data

Use this property to control whether the object writes video frames to the multimedia file. When both this property and the `AudioInputPort` are enabled, the video and audio input signals must have the same frame period. The frame size (or number of rows) of the audio signal might need to be adjusted so that the frame period of the video signal is the same as the frame period of the audio signal. To calculate the frame size, divide the frequency of the audio signal (in samples per second specified by the `SampleRate` property) by the frame rate of the video signal (in frames per second specified by the `FrameRate` property). This default value for this property is `true`.

AudioCompressor

Compression algorithm for audio data

Specify the type of compression algorithm to implement for audio data. This compression reduces the size of the multimedia file. Choose `None` (uncompressed) to save uncompressed audio data to the multimedia file. The other options reflect the available audio compression algorithms installed on your system. This property applies when writing WAV or AVI files on Windows platforms.

VideoCompressor

Compression algorithm for video data

Specify the type of compression algorithm to use to compress the video data. This compression reduces the size of the multimedia file. Choose `None` (uncompressed) to save uncompressed video data to the multimedia file. The other options reflect the available video compression algorithms installed on your system. This property applies only when writing AVI files on Windows platforms.

SampleRate

Sampling rate of audio data stream

Specify the sampling rate of the input audio data as a positive numeric scalar. This property applies when you set the `AudioInputPort` property to `true`. The default value of this property is 44100.

FrameRate

Frame rate of video data stream

Specify the frame rate of the video data in frames per second as a positive numeric scalar. This property applies when you set the `VideoInputPort` property to `true`. The default value of this property is 30.

AudioDataType

Data type of the uncompressed audio

video.MultimediaFileWriter class

Specify the type of uncompressed audio data to write to the file. Note that this parameter applies only when writing uncompressed WAV files.

FileColorSpace

Color space for output file

Specify the color space of AVI files as RGB or YCbCr 4:2:2. This property applies when you set the FileFormat property to AVI and only on Windows platforms. The default value for this property is RGB.

Methods

clone	Create multimedia file writer object with same property values
close	Close multimedia file
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Write input multimedia data to file

Examples

Write a video to disk.

```
hmfr = video.MultimediaFileReader;
hmfw = video.MultimediaFileWriter('vipmen1.avi', ...
    'AudioInputPort',false, ...
    'VideoInputPort',true);
while ~isDone(hmfr)
    videoFrame = step(hmfr);
    step(hmfw,videoFrame);
end
```

```
end
close(hmfr);
close(hmfw);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the To Multimedia File block reference page. The object properties correspond to the block parameters, except for:

- The **Image signal** block parameter allows you to specify whether the block accepts the color video signal as `One multidimensional signal` or `Separate color signals`. The object does not have a property that corresponds to the **Image signal** block parameter. You must always provide the input image to the `step` method of the object as a single multidimensional signal.

See Also

`video.MultimediaFileReader`

video.MultimediaFileWriter.clone

Purpose Create multimedia file writer object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an `MultimediaFileWriter System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

Purpose	Close multimedia file
Syntax	<code>close(H)</code>
Description	<code>close(H)</code> closes the multimedia file in which the multimedia data was written.

video.MultimediaFileWriter.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.MultimediaFileWriter.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.MultimediaFileWriter.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the MultimediaFileWriter System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Write input multimedia data to file

Syntax

```
step(H,AUDIO)
step(H,I)
step(H,I,AUDIO)
step(H,Y,Cb,Cr)
step(H,Y,CB,CR)
step(H,Y,CB,CR,AUDIO)
```

Description

`step(H,AUDIO)` writes one frame of audio samples, `AUDIO`, to the output file when you set the `AudioInputPort` property to true. `AUDIO` is either a vector or an M -by-2 matrix for mono or stereo inputs, respectively.

`step(H,I)` writes one frame of video, `I`, to the output file when you set the `VideoInputPort` property to true. `I` can be an M -by- N -by-3 color video signal or an M -by- N intensity video signal.

`step(H,I,AUDIO)` writes one frame of video, `I`, and one frame of audio samples, `AUDIO`, to the output file when you set both the `AudioInputPort` and `VideoInputPort` properties to true.

`step(H,Y,Cb,Cr)` writes one frame of video with `Y`, `Cb`, `Cr` components to the file. This applies only when you set the `FileColorSpace` property to `YCbCr 4:2:2`.

`step(H,Y,CB,CR)` writes one frame of `YCbCr 4:2:2` data in the color components `Y`, `CB`, and `CR`, to the output file when you set the `VideoInputPort` property to true. The width of `CB` and `CR` must be half of the width of `Y`, and the value of the `FileColorSpace` property must be set to `YCbCr 4:2:2`.

`step(H,Y,CB,CR,AUDIO)` writes one frame of `YCbCr 4:2:2` data in the color components `Y`, `CB`, and `CR`, and one frame of audio samples, `AUDIO`, to the output file when you set both the `AudioInputPort` and `VideoInputPort` properties to true. The width of `CB` and `CR` must be half of the width of `Y`, and the value of the `FileColorSpace` property must be set to `YCbCr 4:2:2`.

video.MultimediaFileWriter.step

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Estimate object velocities
Description	The OpticalFlow System object estimates object velocities from one image or video frame to another.
Construction	<p>H = video.OpticalFlow returns an optical flow System object, H, that estimates the direction and speed of object motion from one image to another, or from one video frame to another.</p> <p>H = video.OpticalFlow('PropertyName',PropertyValue,...) returns an optical flow System object, H, with each specified property set to the specified value.</p>
Properties	<p>Method</p> <p>Algorithm to compute optical flow</p> <p>Specify the algorithm to compute the optical flow as Horn-Schunck, or Lucas-Kanade. The default value for this property is Horn-Schunck.</p> <p>ReferenceFrameSource</p> <p>Source of reference frame for optical flow calculation</p> <p>Specify computing optical flow between one of Property, or Input port. When this property is set to Property, you can use the ReferenceFrameDelay property to determine a previous frame with which to compare. When this property is set to Input port, an input image should be supplied for comparison.</p> <p>This property applies when you set the Method property to Horn-Schunck. This property also applies when you set the Method property to Lucas-Kanade and the TemporalGradientFilter property to Difference filter [-1 1]. The default value for this property is Property.</p> <p>ReferenceFrameDelay</p> <p>Number of frames between reference frame and current frame</p>

video.OpticalFlow class

Specify the number of frames between the reference and current frame as a positive scalar integer. This property applies when you set the ReferenceFrameSource property to Current frame and N-th frame back. The default value of this property is 1.

Smoothness

Expected smoothness of optical flow

Specify the smoothness factor as a positive scalar number. If the relative motion between the two images or video frames is large, specify a large positive scalar value. If the relative motion is small, specify a small positive scalar value. This property applies when you set the Method property to Horn-Schunck. The default value of this property is 1. This property is tunable.

IterationTerminationCondition

Condition to stop iterative solution computation

Specify when the optical flow iterative solution should stop as When maximum number of iterations is reached, When velocity difference falls below threshold, Whichever comes first . This property applies when you set the Method property to Horn-Schunck. The default value of this property is When maximum number of iterations is reached.

MaximumIterationCount

Maximum number of iterations to perform

Specify the maximum number of iterations to perform in the optical flow iterative solution computation as a positive scalar integer. This property applies when you set the Method property to Horn-Schunck and the IterationTerminationCondition property to either When maximum number of iterations is reached or Whichever comes first. The default value of this property is 10. This property is tunable.

VelocityDifferenceThreshold

Velocity difference threshold to stop computation

Specify the velocity difference threshold to stop the optical flow iterative solution computation as a positive scalar number. This property applies when you set the Method property to Horn-Schunck and the IterationTerminationCondition property to either When velocity difference falls below threshold or Whichever comes first. The default value of this property is eps. This property is tunable.

OutputValue

Form of velocity output

Specify the velocity output as Magnitude-squared or Horizontal and vertical components in complex form. The default value of this property is Magnitude-squared.

TemporalGradientFilter

Temporal gradient filter used by Lucas-Kanade algorithm

Specify the temporal gradient filter used by the Lucas-Kanade algorithm as Difference filter [-1 1], Derivative of Gaussian . This property applies when you set the Method property to Lucas-Kanade. The default value of this property is Difference filter [-1 1]

BufferedFramesCount

Number of frames to buffer for temporal smoothing

Specify the number of frames to buffer for temporal smoothing as an odd integer between 3 and 31, both inclusive. This property determines characteristics such as the standard deviation and the number of filter coefficients of the Gaussian filter used to perform temporal filtering. This property applies when you set the Method property to Lucas-Kanade and the TemporalGradientFilter property to Derivative of Gaussian. The default value of this property is 3.

ImageSmoothingFilterStandardDeviation

Standard deviation for image smoothing filter

video.OpticalFlow class

Specify the standard deviation for the Gaussian filter used to smooth the image using spatial filtering as a positive scalar number. This property applies when you set the `Method` property to `Lucas-Kanade` and the `TemporalGradientFilter` property to `Derivative of Gaussian`. The default value of this property is 1.5.

GradientSmoothingFilterStandardDeviation

Standard deviation for gradient smoothing filter

Specify the standard deviation for the filter used to smooth the spatiotemporal image gradient components as a positive scalar number. This property applies when you set the `Method` property to `Lucas-Kanade` and the `TemporalGradientFilter` property to `Derivative of Gaussian`. The default value of this property is 1.

DiscardIllConditionedEstimates

Discard normal flow estimates when constraint equation is ill-conditioned

Set this property to true if the motion vector should be set to 0 when the optical flow constraint equation is ill-conditioned. This property applies when you set the `Method` property to `Lucas-Kanade` and the `TemporalGradientFilter` property to `Derivative of Gaussian`. The default value of this property is false. This property is tunable.

MotionVectorImageOutputport

Return image corresponding to motion vectors

Set this property to true to output the image that corresponds to the motion vector being output by the `System` object. This property applies when you set the `Method` property to `Lucas-Kanade` and the `TemporalGradientFilter` property to `Derivative of Gaussian`. The default value of this property is false.

NoiseReductionThreshold

Threshold for noise reduction

Specify the motion threshold between each image or video frame as a positive scalar number. The higher the number, the less small movements impact the optical flow calculation. This property applies when you set the Method property to Lucas-Kanade. The default value of this property is 0.0039. This property is tunable.

Fixed-Point Properties

RoundingMethod

Rounding mode for fixed-point operations

Specify the rounding method as Ceiling, Convergent, Floor, Nearest, Round, Simplest, or Zero. This property applies when you set the Method property to Lucas-Kanade and the TemporalGradientFilter property to Difference filter [-1 1]. The default value of this property is Nearest.

OverflowAction

Overflow mode for fixed-point operations

Specify the overflow action as Wrap or Saturate. This property applies when you set the Method property to Lucas-Kanade and the TemporalGradientFilter property to Difference filter [-1 1]. The default value of this property is Saturate.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as Custom. This property applies when you set the Method property to Lucas-Kanade and the TemporalGradientFilter property to Difference filter [-1 1].

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a signed, scaled numeric type object. You can apply this property when

video.OpticalFlow class

you set the Method property to Lucas-Kanade and the TemporalGradientFilter property to Difference filter [-1 1]. This property applies when you set the ProductDataType property to Custom. The default value of this property is `numerictype(true,32,20)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as Same as product, or Custom . This property applies when you set the Method property to Lucas-Kanade and the TemporalGradientFilter property to Difference filter [-1 1]. The default value of this property is Same as product.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a signed, scaled numerictype object. You can apply this property when you set the Method property to Lucas-Kanade and the TemporalGradientFilter property to Difference filter [-1 1]. This property applies when you set the AccumulatorDataType property to Custom. The default value of this property is `numerictype(true,32,20)`.

GradientDataType

Gradients word and fraction lengths

Specify the gradient components fixed-point data type as Same as accumulator, Same as accumulator, Same as product, or Custom . This property applies when you set the Method property to Lucas-Kanade and the TemporalGradientFilter property to Difference filter [-1 1]. The default value of this property is Same as accumulator.

CustomGradientDataType

Gradients word and fraction lengths

Specify the gradient components fixed-point type as a signed, scaled `numericType` System object. You can apply this property when you set the `Method` property to `Lucas-Kanade` and the `TemporalGradientFilter` property to `Difference filter [-1 1]`. This property applies when you set the `GradientDataType` property to `Custom`. The default value of this property is `numericType(true,32,20)`.

ThresholdDataType

Threshold word and fraction lengths

Specify the threshold fixed-point data type as `Same word length as first input`, or `Custom`. This property applies when you set the `Method` property to `Lucas-Kanade` and the `TemporalGradientFilter` property to `Difference filter [-1 1]`. The default value of this property is `Same word length as first input`.

CustomThresholdDataType

Threshold word and fraction lengths

Specify the threshold fixed-point type as a signed `numericType` object with a `Signedness` of `Auto`. You can apply this property when you set the `Method` property to `Lucas-Kanade` and the `TemporalGradientFilter` property to `Difference filter [-1 1]`. This property applies when you set the `ThresholdMode` property to `Custom`. The default value of this property is `numericType([],16,12)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Custom`. This property applies when you set the `Method` property to `Lucas-Kanade` and the `TemporalGradientFilter` property to `Difference filter [-1 1]`.

CustomOutputDataType

video.OpticalFlow class

Output word and fraction lengths

Specify the product fixed-point type as a scaled numeric type object with a Signedness of Auto. The numeric type object should be unsigned if you set the OutputValue property to Magnitude-squared and signed if set to Horizontal and vertical components in complex form. You can apply this property when you set the Method property to Lucas-Kanade and the TemporalGradientFilter property to Difference filter [-1 1]. This property applies when you set the OutputDataType property to Custom. The default value of this property is numeric type (false,32,20).

Methods

clone	Create optical flow object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	returns the number of outputs of the step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Estimate direction and speed of object motion between video frames

Examples

Track cars in a video using optical flow.

```
hbfr = video.BinaryFileReader('Filename','viptraffic.bin');
hidtc = video.ImageDataTypeConverter;
hof = video.OpticalFlow('ReferenceFrameDelay', 1);
hof.OutputValue = ...
    'Horizontal and vertical components in complex form';
hsi = video.ShapeInserter('Shape','Lines', ...
```

```
        'BorderColor','Custom', ...
        'CustomBorderColor', 255);
hvp = video.VideoPlayer('WindowCaption', 'Motion Vector');
while ~isDone(hbfr)
    frame = step(hbfr);
    im = step(hidtc, frame); % convert to single precision
    of = step(hof, im);      % compute optical flow
    % generate coordinate points
    lines = videooptflowlines(of, 20);
    if ~isempty(lines)
        % draw lines to indicate flow
        out = step(hsi, im, lines);
        step(hvp, out);      % view in video player
    end
end
close(hvp);
close(hbfr);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Optical Flow Simulink block reference page. The object properties correspond to the block parameters.

See Also

video.Pyramid

video.OpticalFlow.clone

Purpose Create optical flow object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an `OpticalFlow System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, `N` to the step method.

For many System objects, this method is a no-op. Objects that have internal states will describe in their help what the reset method does for that object.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.OpticalFlow.getNumOutputs

Purpose returns the number of outputs of the step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The getNumOutputs method returns a positive integer representing the number of outputs from the step method. This value will change if any properties that turn inputs on or off are changed.

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	<code>isLocked(h)</code>
Description	<p><code>isLocked(h)</code> returns the locked status, TF of the <code>OpticalFlow</code> System object.</p> <p>The <code>isLocked</code> method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the <code>step</code> method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the <code>isLocked</code> method returns a true value.</p>

video.OpticalFlow.step

Purpose Estimate direction and speed of object motion between video frames

Syntax

```
VSQ = step(H,I)
V = step(H,I)
[...] = step(H,I1,I2)
[... , IMV] = step(H,I)
```

Description

`VSQ = step(H,I)` computes the optical flow of input image `I` from one video frame to another, and returns `VSQ` as a matrix of velocity magnitudes.

`V = step(H,I)` computes the optical flow of input image `I` from one video frame to another, and returns `V` as a complex matrix of horizontal and vertical components. This applies when you set the `OutputValue` property to `Horizontal` and `vertical` components in complex form.

`[...] = step(H,I1,I2)` computes the optical flow of the input image `I1`, using `I2` as a reference frame. This applies when you set the `ReferenceFrameSource` property to `Input port`.

`[... , IMV] = step(H,I)` outputs the delayed input image, `IMV`. The delay is equal to the latency introduced by the computation of the motion vectors. This property applies when you set the `Method` property to `Lucas-Kanade`, the `TemporalGradientFilter` property to `Derivative of Gaussian`, and the `MotionVectorImageOutputPort` property to `true`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the `System` object issues a warning and re-initializes.

Purpose Compute peak signal-to-noise ratio (PSNR) between images

Description The PSNR object computes the peak signal-to-noise ratio (PSNR) between images. This ratio is often used as a quality measurement between an original and a compressed image.

Construction `H = video.PSNR` returns a System object, H, that computes the peak signal-to-noise ratio (PSNR) in decibels between two images.

Methods	<code>clone</code>	Create peak signal to noise ratio object with same property values
	<code>getNumInputs</code>	Number of expected inputs to step method
	<code>getNumOutputs</code>	Number of outputs from step method
	<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
	<code>step</code>	Compute peak signal-to-noise ratio

Examples Compute the PSNR between an original image and its reconstructed image.

```
hdct2d = video.DCT2D;  
hidct2d = video.IDCT2D;  
hpsnr = video.PSNR;  
I = double(imread('cameraman.tif'));  
J = step(hdct2d, I);  
J(abs(J) < 10) = 0;  
It = step(hidct2d, J);  
psnr = step(hpsnr, I,It)  
imshow(I, [0 255]), title('Original image');
```

video.PSNR class

```
figure, imshow(It,[0 255]), title('Reconstructed image');
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the PSNR block reference page. The object properties correspond to the block parameters.

See Also

[video.DCT2D](#) | [video.IDCT2D](#)

Purpose Create peak signal to noise ratio object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a PSNR System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.PSNR.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.PSNR.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the PSNR System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Compute peak signal-to-noise ratio

Syntax $Y = \text{step}(H, X1, X2)$

Description $Y = \text{step}(H, X1, X2)$ computes the peak signal-to-noise ratio, Y , between images $X1$ and $X2$. The two images $X1$ and $X2$ must have the same size.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.Pyramid class

Purpose	Perform Gaussian pyramid decomposition
Description	The Pyramid object computes Gaussian pyramid reduction or expansion. The image reduction step involves lowpass filtering and downsampling the image pixels. The image expansion step involves upsampling the image pixels and lowpass filtering.
Construction	<p>H = video.Pyramid returns a System object, H, that computes a Gaussian pyramid reduction or expansion of an image.</p> <p>H = video.Pyramid('PropertyName',PropertyValue,...) returns a gaussian pyramid System object, H, with each specified property set to the specified value.</p>
Properties	<p>Operation</p> <p>Reduce or expand the input image</p> <p>Specify whether to reduce or expand the input image as Reduce or Expand. If this property is set to Reduce, the object applies a lowpass filter and then downsamples the input image. If this property is set to Expand, the object upsamples and then applies a lowpass filter to the input image. The default value for this property is Reduce.</p> <p>PyramidLevel</p> <p>Level of decomposition</p> <p>Specify the number of times the object upsamples or downsamples each dimension of the image by a factor of 2. The default value of this property is 1.</p> <p>SeparableFilter</p> <p>How to specify the coefficients of low pass filter</p> <p>Indicate how to specify the coefficients of the lowpass filter as Default or Custom. The default value for this property is Default.</p> <p>CoefficientA</p>

Coefficient 'a' of default separable filter

Specify the coefficients in the default separable filter $1/4 - a/2$ $1/4$ a $1/4$ $1/4 - a/2$ as a scalar value. This property applies when you set the `SeparableFilter` property to `Default`. The default value of this property is 0.375.

`CustomSeparableFilter`

Separable filter coefficients

Specify separable filter coefficients as a vector. This property applies when you set the `SeparableFilter` property to `Custom`. The default value of this property is [0.0625 0.25 0.375 0.25 0.0625].

Fixed-Point Properties

`RoundingMethod`

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value for this property is `Floor`.

`OverflowAction`

Overflow action for fixed-point operations

Specify the overflow action as `Wrap` or `Saturate`. The default value for this property is `Wrap`.

`SeparableFilterDataType`

`CustomSeparableFilter` word and fraction lengths

Specify the coefficients fixed-point data type as `Same` word length as input, `Custom`. The default value for this property is `Custom`.

`CustomSeparableFilterDataType`

`CustomSeparableFilter` word and fraction lengths

video.Pyramid class

Specify the coefficients fixed-point type as a signed `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `SeparableFilterDataType` property to `Custom`. The default value of this property is `numericType([],16,14)`.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as `Same as input`, or `Custom`. The default value for this property is `Custom`.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `ProductDataType` property to `Custom`. The default value of this property is `numericType([],32,10)`.

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as product`, `Same as input`, or `Custom`. The default value for this property is `Same as product`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a signed, scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType([],32,0)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as Same as input , or Custom. The default value for this property is Custom.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a signed, scaled numeric type object with a Signedness of Auto. This property applies when you set the OutputDataType property to Custom. The default value of this property is numeric type ([], 32, 10).

Methods

clone	Create pyramid object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Compute Gaussian pyramid decomposition of input

Examples

Resize image using gaussian pyramid decomposition

```
hgausspymd = video.Pyramid;  
hgausspymd.PyramidLevel = 2;  
x = imread('cameraman.tif');  
y = step(hgausspymd, x);  
figure, imshow(x); title(' Original Image');  
figure, imshow(mat2gray(double(y)));  
title('Decomposed Image');
```

video.Pyramid class

Algorithm

This object implements the algorithm, inputs, and outputs described on the Gaussian Pyramid block reference page. The object properties correspond to the block parameters.

See Also

`video.GeometricScaler`

Purpose

Create pyramid object with same property values

Syntax

`C = clone(H)`

Description

`C = clone(H)` creates a Pyramid System object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.Pyramid.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.Pyramid.isLocked

Purpose	Locked status (logical) for input attributes and non-tunable properties
Syntax	TF = isLocked(H)
Description	<p>TF = isLocked(H) returns the locked status, TF of the Pyramid System object.</p> <p>The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.</p>

Purpose Compute Gaussian pyramid decomposition of input

Syntax $Y = \text{step}(H, X)$

Description $Y = \text{step}(H, X)$ computes Y , the Gaussian pyramid decomposition of input X .

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.ShapeInserter class

Purpose	Draw rectangles, lines, polygons, or circles on images
Description	The ShapeInserter object draws rectangles, lines, polygons, or circles on images.
Construction	<p>H = video.ShapeInserter returns a System object, H, that draws multiple rectangles, lines, polygons, or circles on images by overwriting pixel values.</p> <p>H = video.ShapeInserter('PropertyName',PropertyValue,...) returns a shape inserter System object, H, with each specified property set to the specified value.</p>
Properties	<p>Shape</p> <p>Type of shape(s) to draw</p> <p>Specify the type of shape(s) to draw as Rectangles, Lines, Polygons, or Circles. The default value of this property is Rectangles.</p> <p>Fill</p> <p>Enable filling shape</p> <p>Set this property to true to fill the shape with an intensity value or a color. This property applies when the Shape property is not set to Lines. The default value of this property is false.</p> <p>BorderColorSource</p> <p>Source of border color</p> <p>Specify how the shape's border color is provided as Input port or Property. This property applies when you set theShape property to Lines or when the Shape property is not set to Lines and you set the Fill property to false. When BorderColorSource is set to Input port, a border color vector must be provided as an input to the System object's step method. The default value of this property is Property.</p>

BorderColor

Border color of shape

Specify the appearance of the shape's border as `Black`, `White`, or `Custom`. If this property is set to `Custom`, the `CustomBorderColor` property is used to specify the value. This property applies when the `BorderColorSource` property is enabled and set to `Property`. The default value of this property is `Black`.

CustomBorderColor

Intensity or color value for shape's border

Specify an intensity or color value for the shape's border. If the input is an intensity image, this property can be set to a scalar intensity value for one shape or R -element vector where R is the number of shapes. If the input is a color image, this property can be set to one of:

- A P -element vector where P is the number of color planes.

- A P -by- R matrix where P is the number of color planes and R is the number of shapes.

This property applies when you set the `BorderColor` property to `Custom`. This property is tunable when the `Antialiasing` property is `false`. The default value of this property is `[200 255 100]`.

FillColorSource

Source of fill color

Specify how the shape's fill color is provided as `Input port` or `Property`. This property applies when you set the `Fill` property to `true`, and you do not set the `Shape` property to `Lines`. When `FillColorSource` is set to `Input port`, a fill color vector must be provided as an input to the `System` object's `step` method. The default value of this property is `Property`.

FillColor

Fill color of shape

video.ShapeInserter class

Specify the intensity of the shading inside the shape as `Black`, `White`, or `Custom`. If this property is set to `Custom`, the `CustomFillColor` property is used to specify the value. This property applies when you enable the `FillColorSource` property and set it to `Property`. The default value of this property is `Black`.

`CustomFillColor`

Intensity or color value for shape's interior

Specify an intensity or color value for the shape's interior. If the input is an intensity image, this property can be set to a scalar intensity value for one shape or an R -element vector where R is the number of shapes. If the input is a color image, this property can be set to one of:

A P -element vector where P is the number of color planes.

A P -by- R matrix where P is the number of color planes and R is the number of shapes.

This property applies when you set the `FillColor` property to `Custom`. This property is tunable when the `Antialiasing` property is `false`. The default value of this property is `[200 255 100]`.

`Opacity`

Opacity of the shading inside shapes

Specify the opacity of the shading inside the shape by a scalar value between 0 and 1, where 0 is transparent and 1 is opaque. This property applies when you set the `Fill` property to `true`. This property is tunable. The default value of this property is `0.6`.

`ROIInputPort`

Enable defining area for drawing shapes via input

Set this property to `true` to define the area in which to draw the shapes via an input to the `step` method. The input is a four-element vector, `[r c height width]`, where r and c are the row and column coordinates of the upper-left corner of the area, and *height* and *width* represent the height (in rows) and width (in columns) of the area. If the property is `false` then the entire

image will be used as the area in which to draw. The default value of this property is false.

Antialiasing

Enable performing smoothing algorithm on shapes

Set this property to true to perform a smoothing algorithm on the line, polygon, or circle. This property applies when you set the Shape property to Lines, Polygons, or Circles. The default value of this property is false.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as Ceiling, Convergent, Floor, Nearest, Round, Simplest, or Zero. This property applies when you set the Fill property to true and/or the Antialiasing property to true. The default value of this property is Floor.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as Wrap , or Saturate. This property applies when you set the Fill property to true and/or the Antialiasing property to true. The default value of this property is Wrap.

OpacityDataType

Opacity word length

Specify the opacity fixed-point data type as Same word length as input, or Custom. This property applies when you set the Fill property to true. The default value of this property is Custom.

CustomOpacityDataType

Opacity word length

video.ShapeInserter class

Specify the opacity fixed-point type as a scaled numeric type object with a Signedness of Auto. This property applies when you set the Fill property to true and the OpacityDataType property to Custom. The default value of this property is numeric type([], 16).

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as Same as first input, or Custom. This property applies when you set the Fill property to true and/or the Antialiasing property to true. The default value of this property is Custom.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled numeric type object with a Signedness of Auto. This property applies when you set the Fill property to true and/or the Antialiasing property to true, and the ProductDataType property to Custom. The default value of this property is numeric type(true, 32, 14).

AccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as Same as product, Same as first input, or Custom. This property applies when you set the Fill property to true and/or the Antialiasing property to true. The default value of this property is Same as product.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled numeric type object with a Signedness of Auto. This property applies when you set the Fill property to true and/or the Antialiasing property

to true, and the AccumulatorDataType property to Custom. The default value of this property is numerictype([],32,14)

Methods

clone	Create shape inserter object with same property values
getNumInputs	Number of expected inputs to step method
getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Draw specified shape on image

Examples

Draw a rectangle on an input image.

```
hshapeins = video.ShapeInserter;  
I = im2double(imread('cameraman.tif'));  
Pts = [10 10 30 30];  
y = step(hshapeins, I, Pts);  
imshow(y);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Draw Shapes block reference page. The object properties correspond to the block parameters, except for:

- The **Image signal** block parameter allows you to specify whether the block accepts the color video signal as **One multidimensional signal** or **Separate color signals**. The object does not have a property that corresponds to the **Image signal** block parameter. You must always provide the input image to the **step** method of the object as a single multidimensional signal.

video.ShapeInserter class

See Also

`video.MarkerInserter`

Purpose	Create shape inserter object with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates a ShapeInserter System object <code>C</code> , with the same property values as <code>H</code> . The <code>clone</code> method creates a new unlocked object.

video.ShapeInserter.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.ShapeInserter.getNumOutputs

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.ShapeInserter.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the ShapeInserter System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Draw specified shape on image

Syntax
`Y = step(H,I,PTS)`
`Y = step(H,I,PTS,ROI)`
`Y = step(H,I,PTS,...,CLR)`

Description `Y = step(H,I,PTS)` draws the specified shape on image `I` at the coordinates specified by `PTS` .
`Y = step(H,I,PTS,ROI)` draws the specified shape only in a rectangular area defined by `ROI` when you set the `ROIInputPort` property to `true`.
`Y = step(H,I,PTS,...,CLR)` uses the border or fill color `CLR` to draw the border or fill the specified shape, when you set the `BorderColorSource` property to `true` or the `FillColorSource` property to `Input port`.

The shapes are embedded on the output image `Y`. When you set the `Shape` property to `Rectangles`, `PTS` must be a four-by- N matrix where each column is of the form `[r c height width]'` and specifies a different rectangle. The argument variables `r` and `c` are the zero-based row and column coordinates of the upper-left corner of the rectangle, and `height` and `width` are the height, in pixels, and width, in pixels, of the rectangle. Here, `height` and `width` must be greater than 0. When you set the `Shape` property to `Lines`, `PTS` must be a $2L$ -by- N matrix where each column specifies a different polyline. Each column must be of the form `[r1,c1,r2,c2...rL,cL]'`, which specifies the points to be connected in consecutive order. When you set the `Shape` property to `Polygons`, `PTS` must be a $2L$ -by- N matrix where each column specifies a different polygon. Each column must be of the form `[r1,c1,r2,c2...rL,cL]'`, which specifies the points to be connected in consecutive order. In this case `[r1,c1]` is also connected to `[rL,cL]`. When you set the `Shape` property to `Circles`, `PTS` must be a three-by- N matrix where each column is of the form `[r c radius]'` and specifies a different circle.

video.ShapeInserter.step

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose

Find standard deviation of input or sequence of inputs

Description

The StandardDeviation object finds standard deviation of input or sequence of inputs.

Construction

H = video.StandardDeviation returns a System object, H, that computes the standard deviation of the entire input.

H =

video.StandardDeviation('PropertyName',PropertyValue,...) returns a standard deviation System object, H, with each specified property set to the specified value.

Properties

RunningStandardDeviation

Enable calculation over time

Set this property to true to enable the calculation of standard deviation over time. The default value of this property is false.

ResetInputPort

Reset in running standard deviation mode

Set this property to true to enable resetting the running standard deviation. When the property is set to true, a reset input must be specified to the step method to reset the running standard deviation. This property applies when you set the RunningStandardDeviation property to true. The default value of this property is false.

ResetCondition

Reset condition for running standard deviation mode

Specify the event to reset the running standard deviation to Rising edge, Falling edge, Either edge, or Non-zero. This property applies when you set the ResetInputPort property to true. The default value for this property is Non-zero.

Dimension

video.StandardDeviation class

Numerical dimension to operate along

Specify how the standard deviation calculation is performed over the data as `All`, `Row`, `Column`, or `Custom`. The default value for this property is `All`.

`CustomDimension`

Numerical dimension to operate along

Specify the dimension (one-based value) of the input signal, over which the standard deviation is computed. The value of this property cannot exceed the number of dimensions in the input signal. This property applies when you set the `Dimension` property to `Custom`. The default value of this property is `1`.

`ROIProcessing`

Enable region of interest processing

Set this property to `true` to enable calculating the standard deviation within a particular region of each image. This property applies when you set the `Dimension` property to `All` and the `RunningStandardDeviation` property to `false`. The default value of this property is `false`.

`ROIForm`

Type of region of interest

Specify the type of region of interest to `Rectangles`, `Lines`, `Label matrix`, or `Binary mask`. This property applies when you set the `ROIProcessing` property to `true`. The default value for this property is `Rectangles`.

`ROIPortion`

Calculate over entire ROI or just perimeter

Specify the region over which to calculate the standard deviation to `Entire ROI`, or `ROI perimeter`. This property applies when you set the `ROIForm` property to `Rectangles`. The default value for this property is `Entire ROI`.

ROIStatistics

Statistics for each ROI, or one for all ROIs

Specify what statistics to calculate as `Individual` statistics for each ROI, or `Single` statistic for all ROIs. This property does not apply when you set the `ROIForm` property to `Binary mask`. The default value of this property is `Individual` statistics for each ROI

ValidityOutputPort

Produces an output with ROI validity status

Set this property to `true` to return the validity of the specified ROI being completely inside of the image. This applies when you set the `ROIForm` property to `Lines`, or `Rectangles`. Set this property to `true` to return the validity of the specified label numbers when you set the `ROIForm` property to `Label Matrix`. This property applies when you set the `ROIForm` property to anything except `Binary mask`. The default value of this property is `false`.

Methods

<code>clone</code>	Create standard deviation object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>reset</code>	Reset running standard deviation state
<code>step</code>	Compute standard deviation of input

video.StandardDeviation class

Examples

Determine the standard deviation in a grayscale image.

```
img = im2single(rgb2gray(imread('peppers.png')));  
hstd2d = video.StandardDeviation;  
std = step(hstd2d,img);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Standard Deviation block reference page. The object properties correspond to the block parameters.

See Also

signalblks.StandardDeviation

Purpose Create standard deviation object with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates a `StandardDeviation System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object with uninitialized states.

video.StandardDeviation.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.StandardDeviation.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.StandardDeviation.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the StandardDeviation System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Reset running standard deviation state

Syntax `reset(H)`

Description `reset(H)` resets the internal states of System object H, used for computing running standard deviation when the `RunningStandardDeviation` property is true.

video.StandardDeviation.step

Purpose Compute standard deviation of input

Syntax

```
Y = step(H,X)
Y = step(H,X,R)
Y = step(H,X,ROI)
Y = step(H,X,LABEL,LABELNUMBERS)
[Y, FLAG] = step(H,X,ROI)
[Y, FLAG] = step(H,X,LABEL,LABELNUMBERS)
```

Description

`Y = step(H,X)` computes the standard deviation of input `X` . It computes the standard deviation of the input elements over time, `Y` , when you set the `RunningStandardDeviation` property to true.

`Y = step(H,X,R)` computes the standard deviation of the input elements over time, `Y`, and optionally resets its state based on the value of reset signal `R` , the `ResetInputPort` property and the `ResetCondition` property. This option applies when you set the `RunningStandardDeviation` property to true and the `ResetInputPort` property to true.

`Y = step(H,X,ROI)` uses additional input `ROI` as the region of interest when you set the `ROIProcessing` property to true and the `ROIForm` property to `Lines`, `Rectangles` or `Binary mask`.

`Y = step(H,X,LABEL,LABELNUMBERS)` computes the standard deviation of input image `X` for region labels contained in vector `LABELNUMBERS` , with matrix `LABEL` marking pixels of different regions. This option applies when you set the `ROIProcessing` property to true and the `ROIForm` property to `Label matrix`.

`[Y, FLAG] = step(H,X,ROI)` also returns `FLAG` which indicates whether the given region of interest is within the image bounds when you set the `ValidityOutputPort` property to true.

`[Y, FLAG] = step(H,X,LABEL,LABELNUMBERS)` also returns `FLAG` which indicates whether the input label numbers are valid when you set the `ValidityOutputPort` property to true.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.TemplateMatcher class

Purpose	Perform template matching by shifting template over image
Description	The TemplateMatcher object performs template matching by shifting template over image.
Construction	<p>H = video.TemplateMatcher returns a template matcher System object, H, that performs template matching by shifting a template in single-pixel increments throughout the interior of an image.</p> <p>H = video.TemplateMatcher('PropertyName',PropertyValue,...) returns a template matcher object, H, with each specified property set to the specified value.</p>
Properties	<p>Metric</p> <p>Metric used for template matching</p> <p>Specify the metric to use for template matching as Sum of absolute differences, Sum of squared differences, or Maximum absolute difference. The default value for this property is Sum of absolute differences.</p> <p>OutputValue</p> <p>Type of output</p> <p>Specify the output that the object should return as Metric matrix, or Best match location. The default value of this property is Best match location.</p> <p>SearchMethod</p> <p>How to search for minimum difference between two inputs</p> <p>Specify how the object searches for the minimum difference between the two input matrices as Exhaustive or Three-step. If you set this property to Exhaustive, the object searches for the minimum difference pixel-by-pixel. If you set this property to Three-step, the object searches for the minimum difference using a steadily decreasing step size. The Three-step method is computationally less expensive than the Exhaustive method,</p>

though it might not find the optimal solution. This property applies when you set the `OutputValue` property to `Best match location`. The default value for this property is `Exhaustive`.

`BestMatchNeighborhoodOutputPort`

Enable metric values output

Set this property to `true` to return two outputs, `NMETRIC` and `NVALID`. The output `NMETRIC` denotes an N -by- N matrix of metric values around the best match, where N is the value of the `NeighborhoodSize` property. The output `NVALID` is a boolean indicating whether the object went beyond the metric matrix to construct output `NMETRIC`. This property applies when you set the `OutputValue` property to `Best match location`. The default value of this property is `false`.

`NeighborhoodSize`

Size of the metric values

Specify the size, N , of the N -by- N matrix of metric values as an odd number. For example, if the matrix size is 3-by-3 set this property to 3. This property applies when you set the `OutputValue` property to `Best match location` and the `BestMatchNeighborhoodOutputPort` property to `true`. The default value of this property is 3.

`ROIInputPort`

Enable ROI specification via input

Set this property to `true` to define the Region of Interest (ROI) over which to perform the template matching. If this property is set to `true`, the ROI is specified using an input to the `step` method. Otherwise the entire input image is used. The default value of this property is `false`.

`ROIValidityOutputPort`

Enable output of a flag indicating if any part of ROI is outside input image

video.TemplateMatcher class

When this boolean property is set to true, the object will return an ROI flag indicating, when false, that a part of the ROI is outside the input image. This property applies when you set the ROIInputPort property to true. The default value is false.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as Ceiling, Convergent, Floor, Nearest, Round, Simplest, or Zero. The default value for this property is Floor.

OverflowAction

Overflow action for fixed-point operations

Specify the overflow action as Wrap or Saturate. The default value for this property is Wrap.

ProductDataType

Product word and fraction lengths

Specify the product fixed-point data type as Same as first input, Custom. This property applies when you set the Metric property to Sum of squared differences. The default value for this property is Custom.

CustomProductDataType

Product word and fraction lengths

Specify the product fixed-point type as a scaled numeric type object with a Signedness of Auto. This property applies when you set the Metric property to Sum of squared differences, and the ProductDataType property to Custom. The default value of this property is numeric type ([], 32, 0).

AccumulatorDataType

video.TemplateMatcher class

Accumulator word and fraction lengths

Specify the accumulator fixed-point data type as `Same as first input`, or `Custom`. The default value for this property is `Custom`.

CustomAccumulatorDataType

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType([],32,0)`.

OutputDataType

Output word and fraction lengths

Specify the output fixed-point data type as `Same as first input`, `Custom`. This property applies when you set the `OutputValue` property to `Metric matrix`. This property applies when you set the `OutputValue` property to `Best match location`, and the `BestMatchNeighborhoodOutputPort` property to `true`. The default value for this property is `Same as first input`.

CustomOutputDataType

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numericType` object with a `Signedness` of `Auto`. This property applies when you set the `OutputDataType` property to `Custom`. The default value of this property is `numericType([],32,0)`.

Methods

<code>clone</code>	Create template matcher object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method

video.TemplateMatcher class

getNumOutputs	Number of outputs from step method
isLocked	Locked status (logical) for input attributes and non-tunable properties
step	Finds the best template match within an image

Examples

Find the location of a particular chip on an image of an electronic board

```
htm=video.TemplateMatcher;
hmi = video.MarkerInserter('Size', 10, ...
'Fill', true, 'FillColor', 'White', 'Opacity', 1);
I1=rgb2gray(imread('board.tif'));
I=I1(1:200,1:200); %Input image
T=I(20:75,90:135); %Use a second similar chip as template
Loc=step(htm,I,T); %Find the location of the first chip
% on the board
Im=step(hmi, I, Loc); %Mark the location on the image
%using a filled white circle
imshow(Im)
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Template Matching block reference page. The object properties correspond to the block parameters.

See Also

[video.OpticalFlow](#) | [video.MarkerInserter](#)

Purpose

Create template matcher object with same property values

Syntax

```
C = clone(H)
```

Description

`C = clone(H)` creates a `TemplateMatcher System` object `C`, with the same property values as `H`. The `clone` method creates a new unlocked object.

video.TemplateMatcher.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.TemplateMatcher.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.TemplateMatcher.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the TemplateMatcher System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose

Finds the best template match within an image

Syntax

```
LOC = step(H,I,T)
METRIC = step(H,I,T)
LOC = step(H,I,T,ROI)
[LOC,ROIINVALID] = step(H,I,T,ROI)
[LOC,NVALS,NVALID] = step(H,I,T)
[LOC,NVALS,NVALID,ROIINVALID] = step(H,I,T,ROI)
```

Description

`LOC = step(H,I,T)` computes the zero-based location, `LOC`, of the best template match relative to the top left corner of the image between the image matrix, `I`, and the template matrix, `T`. The object computes the location by shifting the template in single-pixel increments throughout the interior of the image.

`METRIC = step(H,I,T)` computes the match metric values for image, `I`, with `T` as the template, when you set the `OutputValue` property to `Metric matrix`.

`LOC = step(H,I,T,ROI)` computes the zero-based location of the best template match, `LOC`, in the specified region of interest, `ROI`, when you set the `OutputValue` property to `Best match location` and the `ROIInputPort` property to `true`. `ROI` must be a four element vector, `[row column height width]`, where the first two elements represent the zero-based row and column coordinates of the upper-left corner of the `ROI`, and the last two elements define the height and width of the `ROI`.

`[LOC,ROIINVALID] = step(H,I,T,ROI)` computes the zero-based location of the best template match, `LOC`, in the specified region of interest, `ROI`, and also returns a boolean flag in `ROIINVALID` indicating if the specified `ROI` is outside the bounds of the input image `I`. This option applies when you set the `OutputValue` property to `Best match location`, the `ROIInputPort` property to `true` and the `ROIValidityOutputPort` to `true`.

`[LOC,NVALS,NVALID] = step(H,I,T)` returns the best template match, `LOC`, the metric values around the best match, `NVALS`, and a boolean flag, `NVALID`. `NVALID` indicates, when `false`, that the neighborhood around the best match extended outside the borders of the metric

video.TemplateMatcher.step

value matrix when constructing NVALS. This syntax is possible when you set the `OutputValue` property to `Best match location` and the `BestMatchNeighborhoodOutputPort` property to `true`.

`[LOC,NVALS,NVALID,ROIVALID] = step(H,I,T,ROI)` returns the best template match, `LOC`, the metric values around the best match, `NVALS`, and two boolean flags, `NVALID` and `ROIVALID`. `NVALID` indicates, when `false`, that the neighborhood around the best match extended outside the borders of the metric value matrix when constructing `NVALS`. `ROIVALID` indicates, when `false`, that the specified `ROI` is outside the bounds of the input image `I`. This syntax is possible when you set the `OutputValue` property to `Best match location`, the `BestMatchNeighborhoodOutputPort` property to `true`, the `ROIInputPort` property to `true`, and the `ROIValidityOutputPort` property to `true`.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose

Draw text on image or video stream

Description

The TextInserter object draws text on image or video stream.

Construction

`H = video.TextInserter` returns a System object, H, that draws formatted text onto an image or video stream.

`H = video.TextInserter('PropertyName',PropertyValue,...)` returns a text inserter object, H, with each specified property set to the specified value.

`H = video.TextInserter(TEXT,'PropertyName',PropertyValue,...)` returns a text inserter object, H, with the Text property set to TEXT and other specified properties set to the specified values.

Properties

Text

Text string to draw on image or video stream

Specify the text string to be drawn on image or video stream as a single text string or a cell array of strings. The string(s) can include ANSI C printf-style format specifications, such as %d, %f, or %s.

ColorSource

Source of intensity or color of text

Specify the intensity or color value of the text as Property or Input port. The default value for this property is Property.

Color

Intensity or color of text

Specify the intensity or color of the text as a scalar integer value or a 3-element vector respectively. Alternatively, if the Text property is a cell array of *N* number of strings, specify a 1-by-*N* vector of intensity values or 3-by-*N* matrix of color values that correspond to each string. The default value of this property is

video.TextInserter class

[0 0 0]. This property applies when you set the `ColorSource` property to `Property`. This property is tunable.

LocationSource

Source of text location

Specify the location of the text as `Property` or `Input port`. The default value for this property is `Property`.

Location

Top-left corner of text bounding box

Specify the top-left corner of the text bounding box as a 2-element vector of integers, [*row column*]. The default value of this property is [0 0]. This property applies when you set the `LocationSource` property to `Property`. This property is tunable.

OpacitySource

Source of opacity of text

Specify the opacity of the text as `Property` or `Input port`. The default value of this property is `Property`.

Opacity

Opacity of text

Specify the opacity of the text as numeric scalar between 0 and 1. This property applies when you set the `OpacitySource` property to `Property`. The default value of this property is 1. This property is tunable.

TransposedInput

Specifies if input image data order is row major

Set this property to `true` to indicate that the input image data order is row major. The default value of this property is `false`.

Font

Font face of text

Specify the font of the text as the available fonts installed on the system.

FontSize

Font size in points

Specify the font size as any positive integer value. The default value of this property is 12.

Antialiasing

Perform smoothing algorithm on text edges

Set this property to `true` to smooth the edges of the text. The default value of this property is `true`.

Methods

<code>clone</code>	Create text inserter object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>step</code>	Draws the specified text onto input image

Examples

Draw a text string on a static image.

```
H = video.TextInserter('Peppers are good for you!');
H.Color = 1;
H.FontSize = 24;
H.Location = [315 100];
img1 = im2double(rgb2gray(imread('peppers.png')));
txtimg1 = step(H, img1);
```

video.TextInserter class

```
imshow(txtimg1);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the [Insert Text](#) block reference page. The object properties correspond to the block parameters, except for:

- The **Image signal** block parameter allows you to specify whether the block accepts the color video signal as `One multidimensional signal` or `Separate color signals`. The object does not have a property that corresponds to the **Image signal** block parameter. You must always provide the input image to the `step` method of the object as a single multidimensional signal.

See Also

[video.AlphaBlender](#) | [video.MarkerInserter](#) |
[video.ShapeInserter](#)

Purpose	Create text inserter object with same property values
Syntax	<code>C = clone(H)</code>
Description	<code>C = clone(H)</code> creates a <code>TextInserter System</code> object <code>C</code> , with the same property values as <code>H</code> . The <code>clone</code> method creates a new unlocked object.

video.TextInserter.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs, N from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.TextInserter.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the TextInserter System object.

The isLocked method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the isLocked method returns a true value.

Purpose

Draws the specified text onto input image

Syntax

```
Y = step(H, IMG)
Y = step(H, IMG, CELLIDX)
Y = step(H, IMG, VARS)
Y = step(H, IMG, COLOR)
Y = step(H, IMG, LOC)
Y = step(H, IMG, OPAC)
Y = step(H, IMG, CELLIDX, VARS, COLOR, LOC, OPAC)
```

Description

`Y = step(H, IMG)` draws the specified text onto input image `IMG` and returns the modified image `Y`. The image `IMG` can either be an M -by- N matrix of intensity values or an M -by- N -by- P array color video signal where P is the number of color planes.

`Y = step(H, IMG, CELLIDX)` draws the text string selection given in zero-based index value, `CELLIDX`, when the `Text` property is a cell array of strings. A `CELLIDX` value less than 0 or more than the length of cell array minus one, will cause no text to be drawn.

`Y = step(H, IMG, VARS)` uses the data in `VARS` for variable substitution, when the `Text` property contains ANSI C printf-style format specifications (`%d`, `%.2f`, etc.). `VARS` is a scalar or a vector having length equal to the number of format specifiers in each element in the specified text string.

`Y = step(H, IMG, COLOR)` uses the given scalar or 3-element vector `COLOR` for the text intensity or color respectively, when you set the `ColorSource` property to `Input port`.

`Y = step(H, IMG, LOC)` places the text at the location given by two-element vector `LOC`, when you set the `LocationSource` property to `Input port`.

`Y = step(H, IMG, OPAC)` uses `OPAC` for the text opacity when you set the `OpacitySource` property to set to `Input port`.

`Y = step(H, IMG, CELLIDX, VARS, COLOR, LOC, OPAC)` draws the specified text onto image `IMG` using text string selection index `CELLIDX`, text variable substitution data `VARS`, intensity or color value `COLOR` at

video.TextInserter.step

location LOC with opacity OPAC. You can use any combination or all possible inputs. Properties must be set appropriately.

Note The object performs an initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

Purpose	Find variance values in an input or sequence of inputs
Description	The Variance object finds variance values in an input or sequence of inputs.
Construction	<p>H = video.Variance returns a System object, H, that computes the variance of an input or a sequence of inputs.</p> <p>H = video.Variance('PropertyName',PropertyValue,...) returns a variance System object, H, with each specified property set to the specified value.</p>
Properties	<p>RunningVariance</p> <p>Enable calculation over time</p> <p>Set this property to true to enable the calculation of the variance over time. The default value of this property is false.</p> <p>ResetInputPort</p> <p>Enable resetting via an input in running variance mode</p> <p>Set this property to true to enable resetting the running variance. When the property is set to true, a reset input must be specified to the step method to reset the running variance. This property applies when you set the RunningVariance property to true. The default value of this property is false.</p> <p>ResetCondition</p> <p>Reset condition for running variance mode</p> <p>Specify the event to reset the running variance as Rising edge, Falling edge, Either edge, or Non-zero. This property applies when you set the ResetInputPort property to true. The default value for this property is Non-zero.</p> <p>CustomDimension</p> <p>Numerical dimension to operate along</p>

video.Variance class

Specify the dimension (one-based value) of the input signal, over which the variance is computed. The value of this property cannot exceed the number of dimensions in the input signal. This property applies when you set the `Dimension` property to `Custom`. The default value of this property is `1`.

Dimension

Numerical dimension to operate along

Specify how the variance calculation is performed over the data as `All`, `Row`, `Column`, or `Custom`. This property applies only when you set the `RunningVariance` property to `false`. The default value for this property is `All`.

ROIForm

Type of region of interest

Specify the type of region of interest as `Rectangles`, `Lines`, `Label matrix`, or `Binary mask`. This property applies when you set the `ROIProcessing` property to `true`. Full ROI processing support requires a Video and Image Processing Blockset license. If you only have the Signal Processing Blockset license, the `ROIForm` property value options are limited to `Rectangles`. The default value for this property is `Rectangles`.

ROIPortion

Calculate over entire ROI or just perimeter

Specify the region over which to calculate the variance as `Entire ROI`, or `ROI perimeter`. This property applies when you set the `ROIForm` property to `Rectangles`. The default value for this property is `Entire ROI`.

ROIProcessing

Enable region of interest processing

Set this property to `true` to enable calculating the variance within a particular region of each image. This property applies when you set the `Dimension` property to `All` and the `RunningVariance`

property to `false`. Full ROI processing support requires a Video and Image Processing Blockset license. If you only have the Signal Processing Blockset license, the `ROIForm` property value options are limited to `Rectangles`. The default value of this property is `false`.

ROIStatistics

Statistics for each ROI, or one for all ROIs

Specify what statistics to calculate as `Individual statistics` for each ROI, or `Single statistic` for all ROIs. This property does not apply when you set the `ROIForm` property to `Binary mask`. The default value of this property is `Individual statistics for each ROI`.

ValidityOutputPort

Produces an output with ROI validity status

Set this property to `true` to return the validity of the specified ROI being completely inside of the image. Set this property to `true` to return the validity of the specified label numbers when you set the `ROIForm` property to `Label Matrix`. This property applies when you set the `ROIForm` property to `Lines` or `Rectangles`. The default value of this property is `false`.

Fixed-Point Properties

RoundingMethod

Rounding method for fixed-point operations

Specify the rounding method as `Ceiling`, `Convergent`, `Floor`, `Nearest`, `Round`, `Simplest`, or `Zero`. The default value for this property is `Floor`.

OverflowAction

Overflow action for fixed-point operations

video.Variance class

Specify the overflow action as `Wrap` or `Saturate`. The default value for this property is `Wrap`.

`InputSquaredProductDataType`

Input squared product and fraction lengths

Specify the input-squared product fixed-point data type as `Same as input` or `Custom`. The default value for this property is `Same as input`.

`CustomInputSquaredProductDataType`

Input squared product word and fraction lengths

Specify the input-squared product fixed-point type as a scaled `numericType` object. This property applies when you set the `InputSquaredProductDataType` property to `Custom`. The default value of this property is `numericType(true, 32, 15)`.

`InputSumSquaredProductDataType`

Input-sum-squared product and fraction lengths

Specify the input-sum-squared product fixed-point data type as `Same as input-squared product` or `Custom`. The default value for this property is `Same as input-squared product`.

`CustomInputSumSquaredProductDataType`

Input sum-squared product and fraction lengths

Specify the input-sum-squared product fixed-point type as a scaled `numericType` object. This property applies when you set the `InputSumSquaredProductDataType` property to `Custom`. The default value of this property is `numericType(true, 32, 23)`.

`AccumulatorDataType`

Data type of the accumulator

Specify the accumulator fixed-point data type as `Same as input`, or `Custom`. The default value for this property is `Same as input`.

`CustomAccumulatorDataType`

Accumulator word and fraction lengths

Specify the accumulator fixed-point type as a scaled `numericType` object. This property applies when you set the `AccumulatorDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

`OutputDataType`

Data type of output

Specify the output fixed-point data type as `Same as accumulator`, `Same as input`, or `Custom`. The default value of this property is `Same as accumulator`.

`CustomOutputDataType`

Output word and fraction lengths

Specify the output fixed-point type as a scaled `numericType` object.

This property applies when you set the `OutputDataType` property to `Custom`. The default value of this property is `numericType(true,32,30)`.

Methods

<code>clone</code>	Create variance object with same property values
<code>getNumInputs</code>	Number of expected inputs to step method
<code>getNumOutputs</code>	Number of outputs from step method
<code>isLocked</code>	Locked status (logical) for input attributes and non-tunable properties
<code>reset</code>	Reset the internal states of the variance object
<code>step</code>	Compute variance of input

video.Variance class

Examples

Determine the variance in a grayscale image.

```
img = im2single(rgb2gray(imread('peppers.png')));  
hvar2d = video.Variance;  
var2d = step(hvar2d,img);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Variance block reference page. The object properties correspond to the block parameters. except for:

- **Treat sample-based row input as a column** block parameter is not supported by the Variance object.
- **Reset port** block parameter corresponds to both the `ResetCondition` and the `ResetInputPort` object properties.

See Also

`signalblks.Variance`

Purpose

Create variance object with same property values

Syntax

```
C = clone(H)
```

Description

`C = clone(H)` creates a `Variance System` object `C`, with the same property values as `H`. The clone method creates a new unlocked object with uninitialized states.

video.Variance.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method.

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

Purpose Number of outputs from step method

Syntax `N = getNumOutputs(H)`

Description `N = getNumOutputs(H)` returns the number of outputs for the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.Variance.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the Variance System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.

Purpose Reset the internal states of the variance object

Syntax reset(H)

Description reset(H) resets the internal states of System object H to their initial values.

video.Variance.step

Purpose Compute variance of input

Syntax

```
Y = step(H,X)
Y = step(H,X,R)
VAR2D = step(H,X,ROI)
VAR2D = step(H,X,LABEL,LABELNUMBERS)
[VAR2D, FLAG] = step(H,X,ROI)
[VAR2D, FLAG] = step(H,X,LABEL,LABELNUMBERS)
```

Description

`Y = step(H,X)` computes the variance of input `X`. Computes the variance of the input elements over time, `Y`, when you set the `RunningVariance` property to true.

`Y = step(H,X,R)` computes the variance of the input elements over time, `Y`, and optionally resets its state based on the value of the reset signal `R`, the `ResetInputPort` property and the `ResetCondition` property. This option applies when you set the `RunningVariance` property to true and the `ResetInputPort` to true.

`VAR2D = step(H,X,ROI)` computes the variance of input image `X` within the given region of interest `ROI` when you set the `ROIProcessing` property to true and the `ROIForm` property to `Lines`, `Rectangles` or `Binary mask`.

`VAR2D = step(H,X,LABEL,LABELNUMBERS)` computes the variance of input image `X` for region labels contained in vector `LABELNUMBERS`, with matrix `LABEL` marking pixels of different regions. This option applies when you set the `ROIProcessing` property to true and the `ROIForm` property to `Label matrix`.

`[VAR2D, FLAG] = step(H,X,ROI)` also returns `FLAG`, which indicates whether the given region of interest is within the image bounds when you set both the `ROIProcessing` and the `ValidityOutputPort` properties to true and the `ROIForm` property to `Lines`, `Rectangles` or `Binary mask`.

`[VAR2D, FLAG] = step(H,X,LABEL,LABELNUMBERS)` also returns `FLAG`, which indicates whether the input label numbers are valid when you set

both the ROIProcessing and ValidityOutputPort properties to true and the ROIForm property to Label matrix.

Note The object performs an initialization the first time the step method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a non-tunable property or an input specification, the System object issues a warning and re-initializes.

video.VideoPlayer class

Purpose	Play video or display image sequences								
Description	The VideoPlayer object plays video or displays image sequences.								
Construction	<p>H = video.VideoPlayer returns a video player object, H, to view video or image sequences.</p> <p>H = video.VideoPlayer('PropertyName',PropertyValue, ...) returns a video player object, H, with each property set to the specified value.</p>								
Properties	<p>WindowCaption</p> <p>Caption display on video player window</p> <p>Specify the caption to display on the video player window as a string. The property defaults to Video.</p> <p>WindowPosition</p> <p>Size and position of the video player window in pixels</p> <p>Specify the size and position of the video player window in pixels as a four-element vector of the form: [left bottom width height]. The default value of this property is dependent on the screen resolution and positions the window in the center of the screen with a width and height of 410 and 300 pixels. This property is tunable.</p>								
Methods	<table><tr><td>clone</td><td>Create video player with same property values</td></tr><tr><td>close</td><td>Release video resources</td></tr><tr><td>getNumInputs</td><td>Number of expected inputs to step method</td></tr><tr><td>getNumOutputs</td><td>Number of outputs from step method</td></tr></table>	clone	Create video player with same property values	close	Release video resources	getNumInputs	Number of expected inputs to step method	getNumOutputs	Number of outputs from step method
clone	Create video player with same property values								
close	Release video resources								
getNumInputs	Number of expected inputs to step method								
getNumOutputs	Number of outputs from step method								

isLocked	Locked status (logical) for input attributes and non-tunable properties
reset	Reset displayed frame number to zero
step	Play video or image sequence

Examples

Play back a video on the screen:

```
hmfr = video.MultimediaFileReader;
hvp = video.VideoPlayer;
while ~isDone(hmfr)
    frame = step(hmfr);
    step(hvp, frame);
end
close(hmfr);
close(hvp);
```

Algorithm

This object implements the algorithm, inputs, and outputs described on the Video Viewer block reference page. The object properties correspond to the block parameters, except for:

- The WindowCaption property can only be set directly in the video player object.
- The WindowPosition property can only be set directly in the video player object.
- The **Image Signal** block parameter allows you to specify whether the block accepts the color video signal as One Multi-Dimensional Signal or Separate Color Signals. The object does not have a property that corresponds to the **Image Signal** block parameter. You must always provide the input image to the step method of the object as a single multidimensional signal.

See Also

video.DeployableVideoPlayer | video.MultimediaFileWriter

video.VideoPlayer.clone

Purpose Create video player with same property values

Syntax `C = clone(H)`

Description `C = clone(H)` creates an instance of the current video player object with the same property values. The `clone` method creates a new unlocked object with uninitialized states.

Purpose Release video resources

Syntax `close(H)`

Description `close(H)` releases the system resources used by the video player.

video.VideoPlayer.getNumInputs

Purpose Number of expected inputs to step method

Syntax `N = getNumInputs(H)`

Description `N = getNumInputs(H)` returns the number of expected inputs, N to the step method

The `getNumInputs` method returns a positive integer representing the number of expected inputs to the `step` method. This value will change if any properties that turn inputs on or off are changed. The `step` method must be called with a number of input arguments equal to the result of `getNumInputs(H)`.

video.VideoPlayer.getNumOutputs

Purpose Number of outputs from step method

Syntax N = getNumOutputs(H)

Description N = getNumOutputs(H) returns the number of arguments from the step method.

The `getNumOutputs` method returns a positive integer representing the number of outputs from the `step` method. This value will change if any properties that turn inputs on or off are changed.

video.VideoPlayer.isLocked

Purpose Locked status (logical) for input attributes and non-tunable properties

Syntax TF = isLocked(H)

Description TF = isLocked(H) returns the locked status, TF of the VideoPlayer System object.

The `isLocked` method returns a logical value to indicate whether input attributes and non-tunable properties are locked for the object. The object performs an internal initialization the first time the `step` method is executed. This initialization locks non-tunable properties and input specifications, such as dimensions, complexity, and data type of the input data. Once this occurs, the `isLocked` method returns a true value.


Purpose	Reset displayed frame number to zero
Syntax	<code>reset(H)</code>
Description	<code>reset(H)</code> resets the displayed frame number of the video player to zero.

video.VideoPlayer.step

Purpose Play video or image sequence

Syntax `step(H,I)`

Description `step(H,I)` sends one frame of a multidimensional video I, or image sequence to the video player.

Purpose	Open top-level Video and Image Processing Blockset library
Syntax	<code>viplib</code>
Description	<code>viplib</code> opens the top-level Video and Image Processing Blockset block library model.
Examples	View and gain access to the Video and Image Processing Blockset blocks: <code>viplib</code>
Alternatives	To view and gain access to the Video and Image Processing Blockset blocks using the Simulink library browser: <ul style="list-style-type: none">• Type <code>simulink</code> at the MATLAB command line, and then expand the Video and Image Processing Blockset node in the library browser.• Click the Simulink icon  from the MATLAB desktop or from a model.

Function Reference

Video and Image Processing
Functions (p. 5-2)

Video & Image Processing Functions

Video and Image Processing Functions

<code>isfilterseparable</code>	Determine whether filter coefficients are separable
<code>mplay</code>	View video from MATLAB workspace, multimedia file, or Simulink model
<code>viplib</code>	Open top-level Video and Image Processing Blockset library