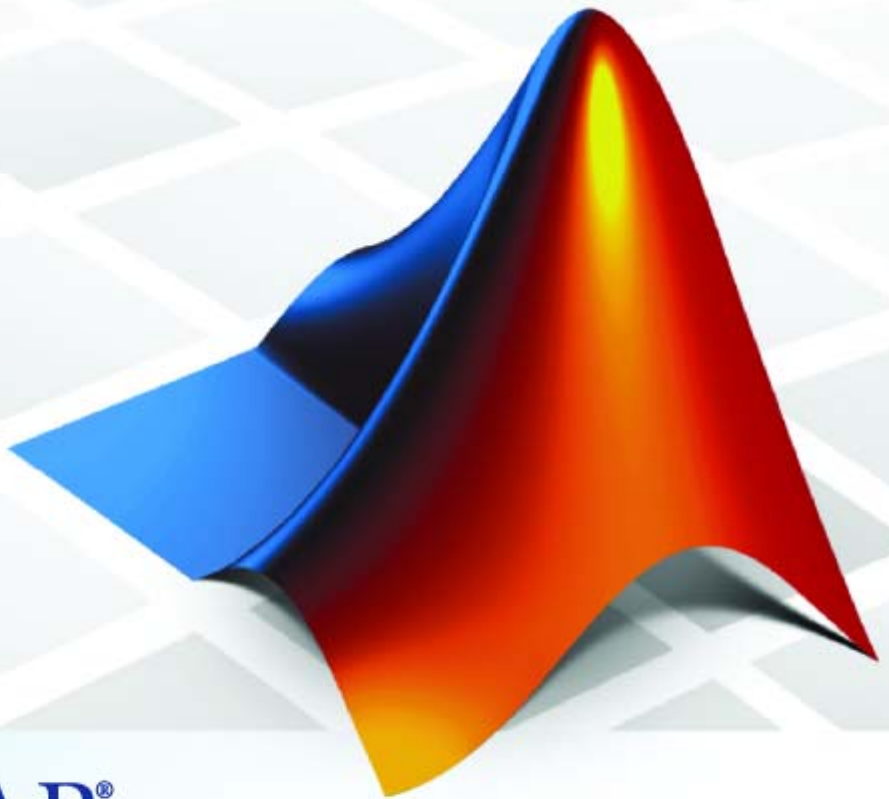


Video and Image Processing Blockset 2

Reference



MATLAB[®]
& **SIMULINK[®]**

How to Contact The MathWorks



www.mathworks.com
comp.soft-sys.matlab
www.mathworks.com/contact_TS.html

Web
Newsgroup
Technical Support



suggest@mathworks.com
bugs@mathworks.com
doc@mathworks.com
service@mathworks.com
info@mathworks.com

Product enhancement suggestions
Bug reports
Documentation error reports
Order status, license renewals, passcodes
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 –2007 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used 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.

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, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, SimBiology, SimHydraulics, SimEvents, and xPC TargetBox are registered trademarks and The MathWorks, the L-shaped membrane logo, Embedded MATLAB, and PolySpace are trademarks of The MathWorks, Inc.

Other product or brand names are 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
September 2007 Online only

New for Version 2.3 (Release 2007a)
Revised for Version 2.4 (Release 2007b)

Blocks — By Category

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	1-6
Transforms	1-6
Utilities	1-7

Blocks — Alphabetical List

2

Functions — Alphabetical List

3

Index

Blocks — By Category

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. 1-6)	Annotate images or video
Transforms (p. 1-6)	Perform transform operations such as 2-D FFT and 2-D DCT
Utilities (p. 1-7)	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
SAD	Perform 2-D sum of absolute differences (SAD)
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

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	Send video data to display devices
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	Generate histogram of each input matrix
2-D Mean	Find mean value of each input matrix
2-D Median	Find median value of each input matrix
2-D Standard Deviation	Find standard deviation of each input matrix
2-D Variance	Compute variance of each input matrix
Blob Analysis	Compute statistical values for labeled regions
Find Local Maxima	Find local maxima in matrices

Maximum	Find maximum values in input or sequence of inputs
Minimum	Find minimum values in input or sequence of inputs
PSNR	Compute peak signal-to-noise ratio (PSNR) between images

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

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

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

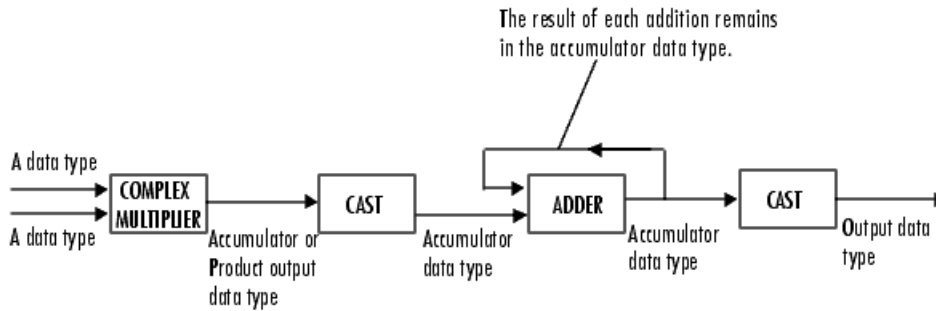
The output of this block has dimensions $(2Ma - 1, 2Na - 1)$.

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.



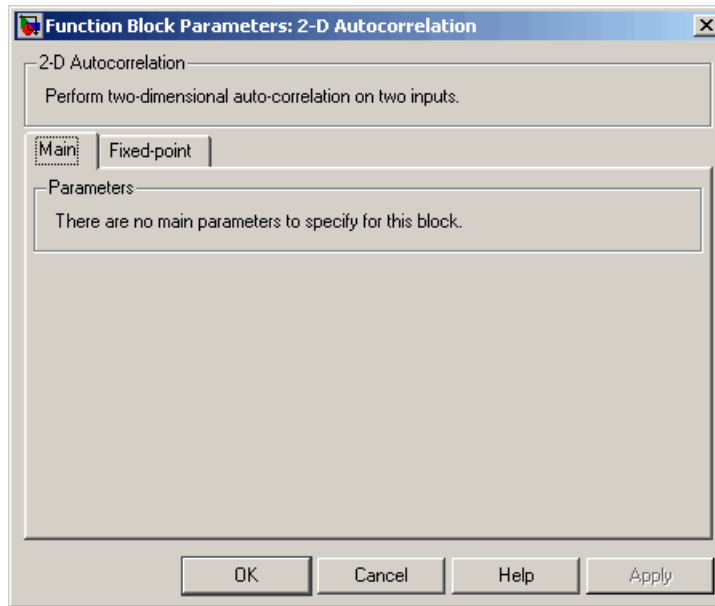
You can set the product output, accumulator, and output data types in the block mask as discussed in “Dialog Box” on page 2-4.

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.

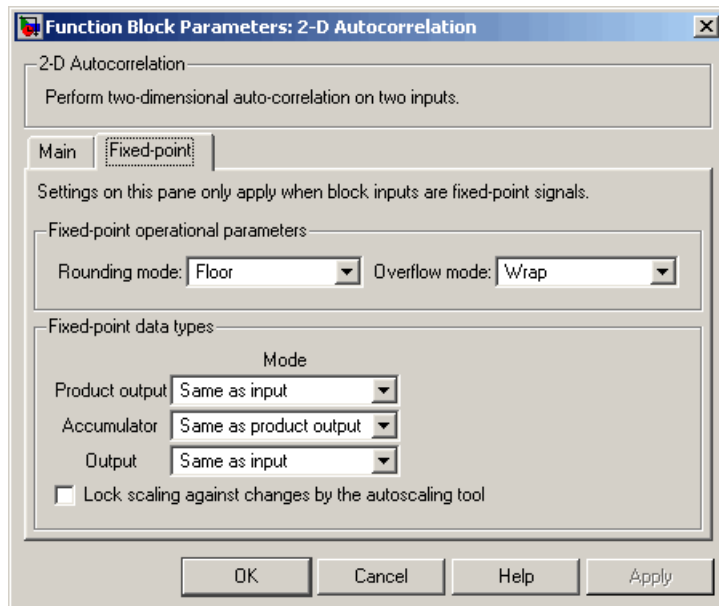
2-D Autocorrelation

Dialog Box

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



The **Fixed-point** 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 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.

2-D Autocorrelation

- 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 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

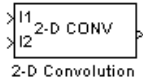
2-D Correlation	Video and Image Processing Blockset
2-D Histogram	Video and Image Processing Blockset
2-D Mean	Video and Image Processing Blockset
2-D Median	Video and Image Processing Blockset
2-D Standard Deviation	Video and Image Processing Blockset
2-D Variance	Video and Image Processing Blockset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

2-D Convolution

Purpose Compute 2-D discrete convolution of two input matrices

Library Filtering

Description The 2-D Convolution block computes the two-dimensional convolution of two input matrices. Assume that matrix A has dimensions (Ma, Na) and matrix B has dimensions (Mb, Nb). When the block calculates the full output size, the equation for the 2-D discrete convolution is



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

where $0 \leq i < Ma + Mb - 1$ and $0 \leq j < Na + Nb - 1$.

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 represents one plane of the RGB video stream	Same as I1 port	Yes
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.

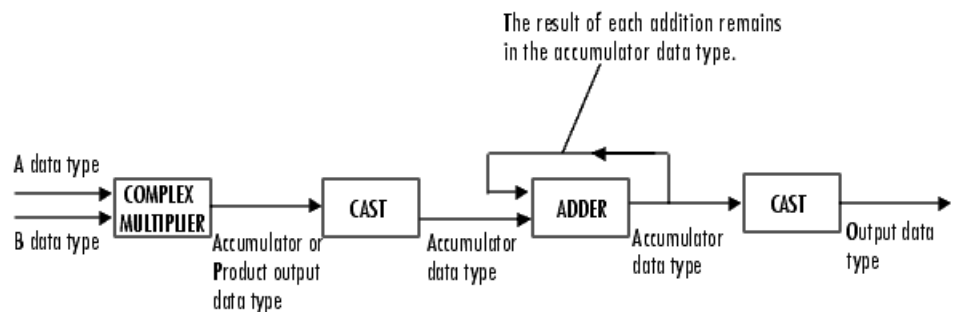
The dimensions of the output are dictated by the **Output size** parameter. Assume that the input at port I1 has dimensions (M_a, N_a) and the input at port I2 has dimensions (M_b, N_b) . If, for the **Output size** parameter, you choose **Full**, the output is the full two-dimensional convolution with dimensions (M_a+M_b-1, N_a+N_b-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 (M_a-M_b+1, N_a-N_b+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:

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

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

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

2-D Convolution

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

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.

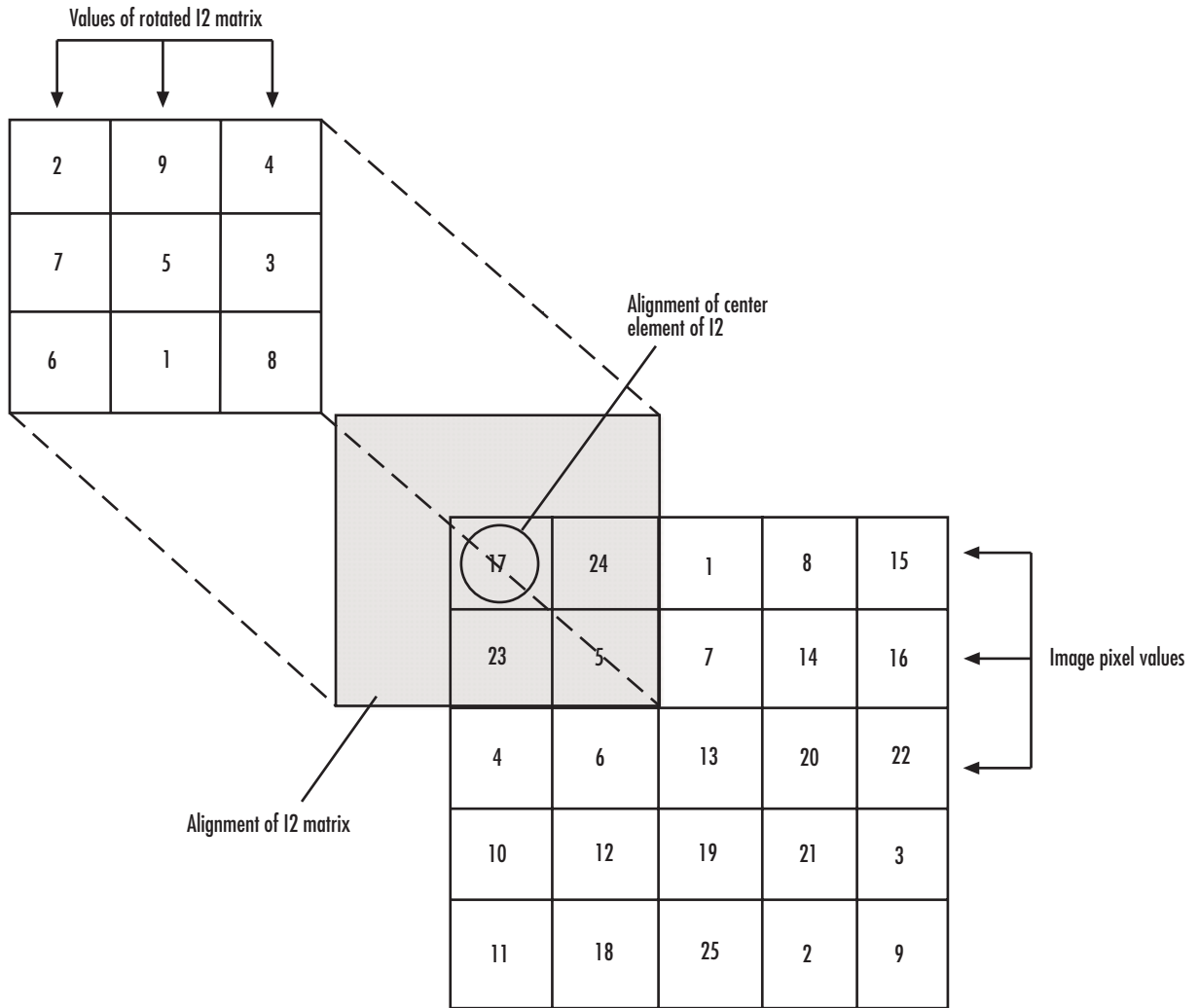
3 Multiply each element of the rotated I2 matrix by the element of I1 underneath.

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

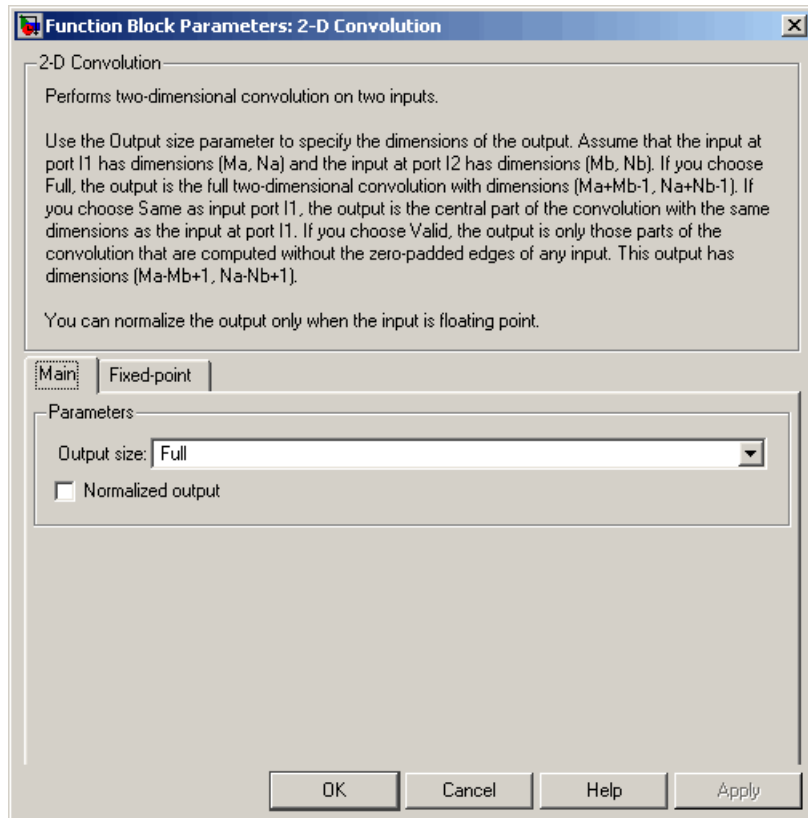


Computing the (1,1) Output of Convolution

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.



Output size

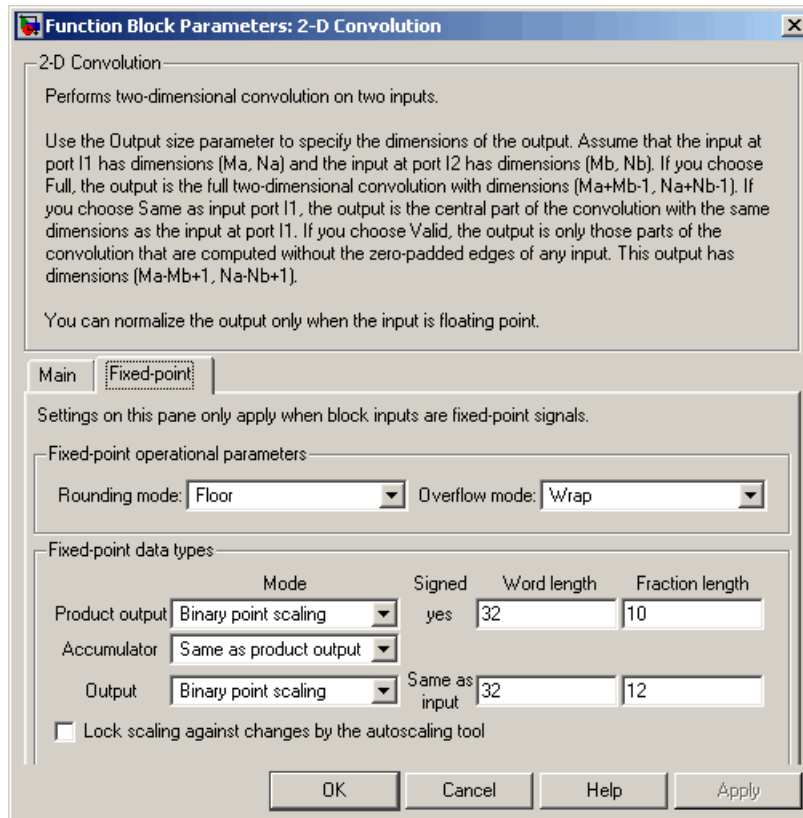
This parameter controls the size of the output scalar, vector, or matrix produced as a result of the convolution between the two inputs. If you choose Full, the output has dimensions (Ma+Mb-1, Na+Nb-1). If you choose Same as input port I1, the output has the same dimensions as the input at port I1. If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

2-D Convolution

Output normalized convolution

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

The **Fixed-point** pane of the 2-D Convolution 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 2-9 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 Video and Image Processing Blockset 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 2-9 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 Video and Image Processing Blockset is 0.

2-D Convolution

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 Video and Image Processing Blockset is 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

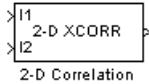
2-D FIR Filter

Video and Image Processing Blockset

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

Library Statistics

Description



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

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

where $0 \leq i < Ma + Mb - 1$ and $0 \leq j < Na + Nb - 1$.

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.

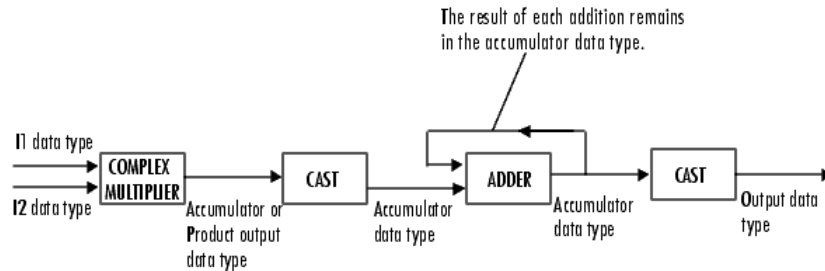
The dimensions of the output are dictated by the **Output size** parameter and the sizes of the inputs at ports I1 and I2. For example, assume that the input at port I1 has dimensions (Ma, Na) and the input at port I2 has dimensions (Mb, Nb). If, for the **Output size** parameter, you choose Full, the 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 $\text{all}(\text{size}(I1) < \text{size}(I2))$, 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 2-23 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 2-26.

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

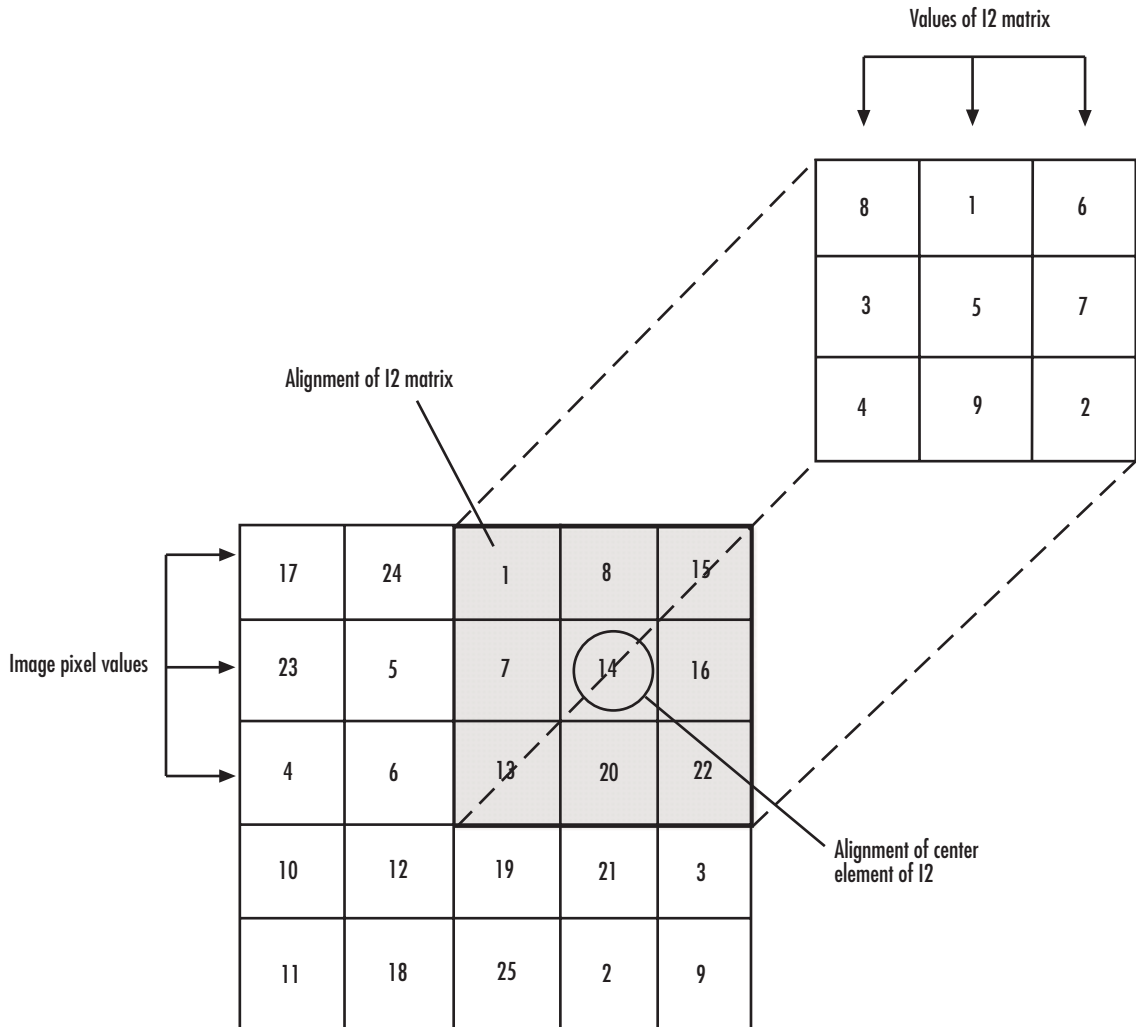
2 Multiply each weight in I2 by the element of I1 underneath.

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



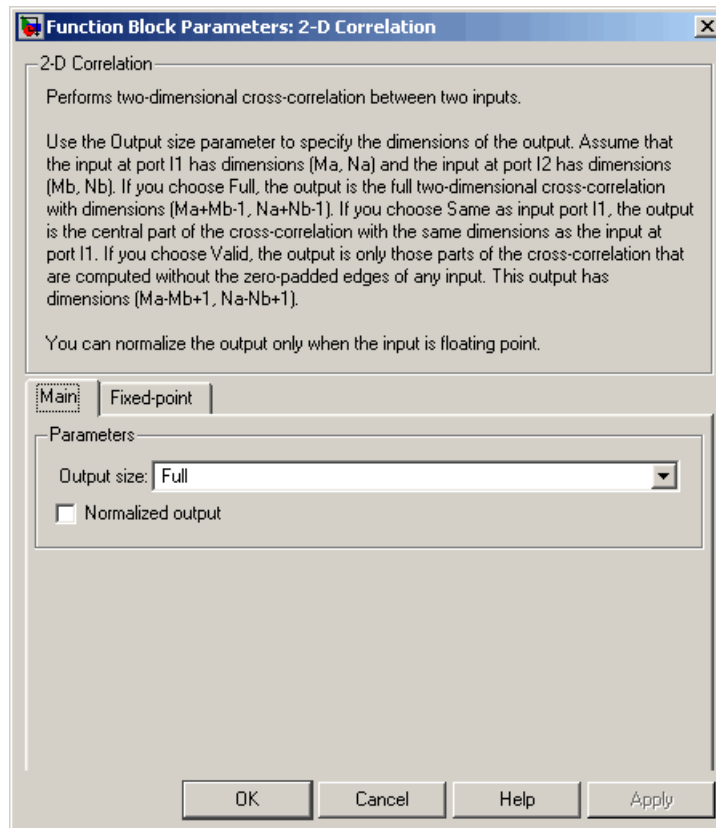
Computing the (2,4) Output of Cross-Correlation

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.



Output size

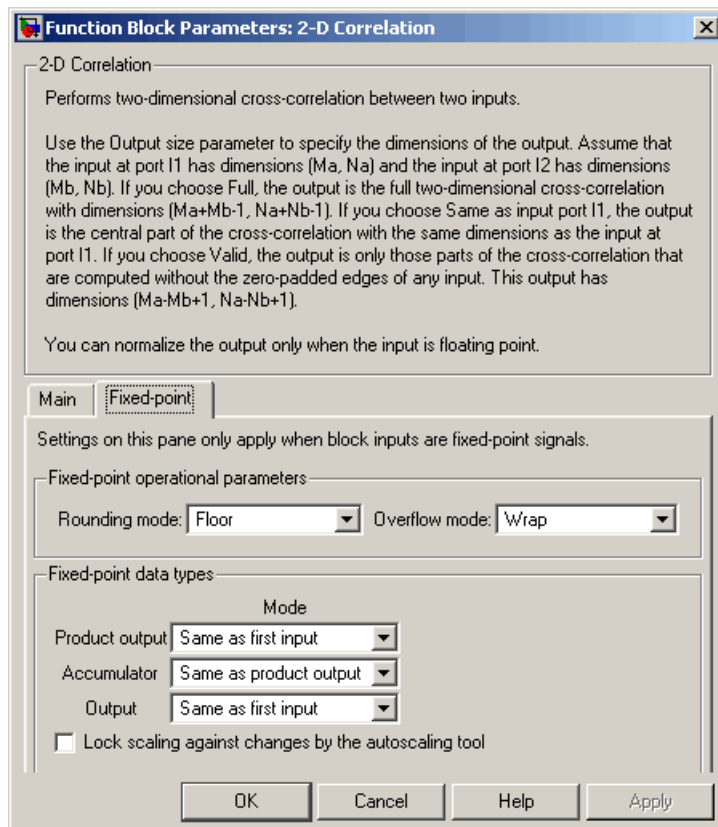
This parameter controls the size of the output scalar, vector, or matrix produced as a result of the cross-correlation between

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 **Fixed-point** 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 2-20 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 Video and Image Processing Blockset 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 2-20 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 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

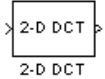
2-D Autocorrelation	Video and Image Processing Blockset
2-D Histogram	Video and Image Processing Blockset
2-D Mean	Video and Image Processing Blockset
2-D Median	Video and Image Processing Blockset
2-D Standard Deviation	Video and Image Processing Blockset
2-D Variance	Video and Image Processing Blockset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

2-D DCT

Purpose Compute 2-D discrete cosine transform (DCT)

Library Transforms

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.

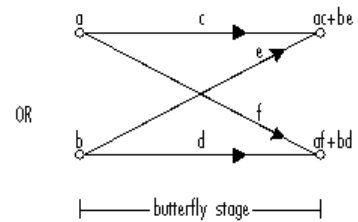
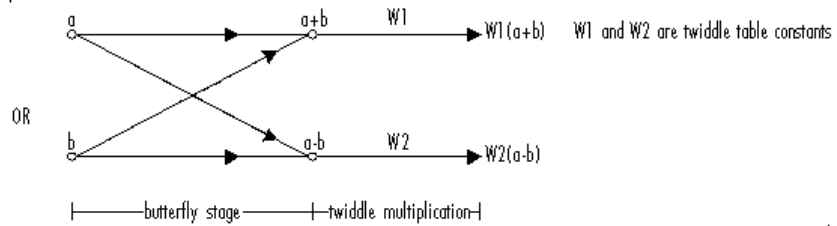
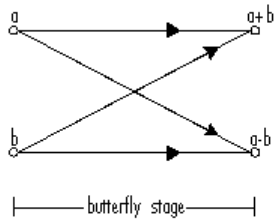
Use the **Sine and cosine computation** parameter to specify how the block computes the sine and cosine terms in the DCT algorithm. If you select **Trigonometric fcn**, the block computes the sine and cosine

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.

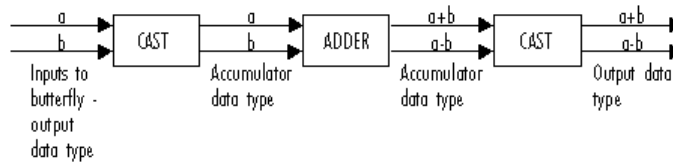
Fixed-Point Data Types

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.

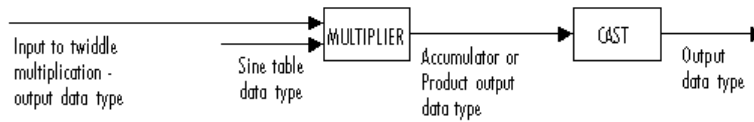
2-D DCT



Butterfly Stage Data Types



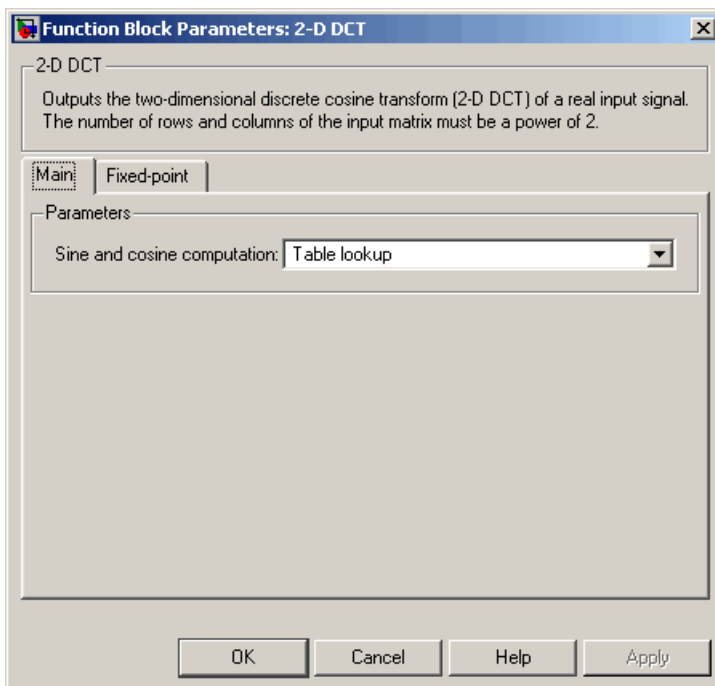
Twiddle Multiplication Data Types



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.

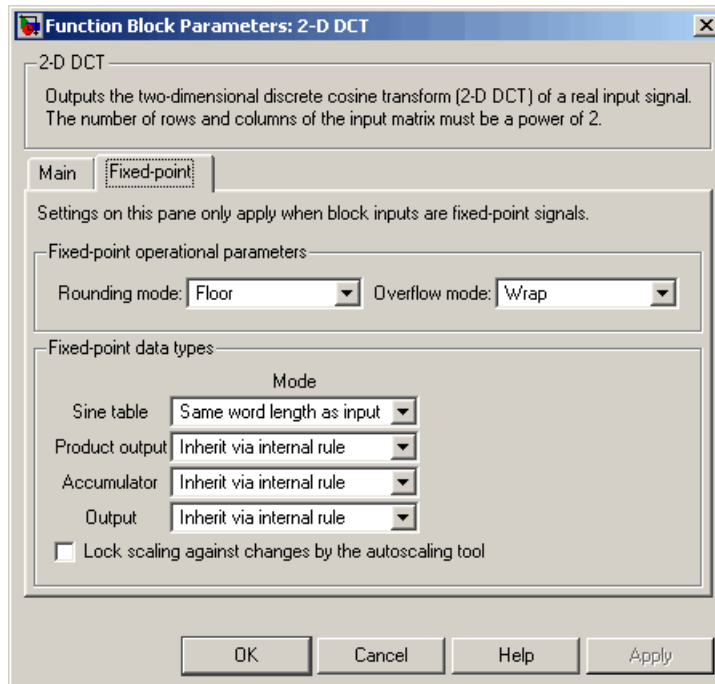


2-D DCT

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

The **Fixed-point** 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 are always saturated and rounded to Nearest.

Overflow mode

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

Sine table

Choose how to specify the word length of the values of the sine table. The fraction length of the sine table values is always equal to the word length minus 1:

- When you select `Same word length as input`, the word length of the sine table values match that of the input to the block.
- 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 are 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 2-43 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 `Inherit via internal rule`, the product output word length and fraction length are automatically set according to the following equations:

*ideal product output word length =
output word length + sine table values word length*

*ideal product output fraction length =
output fraction length + sine table values 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 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 Video and Image Processing Blockset 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 2-43 and “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block:

- When you select **Inherit via internal rule**, the accumulator word length and fraction length are automatically set according to the following equations:

ideal accumulator word length = product output word length + 1

ideal accumulator fraction length = product output fraction length

- 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 Video and Image Processing Blockset is 0.

Output

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

- When you select `Inherit` via `internal rule`, the output word length and fraction length are automatically set according to the following equations, where the input matrix is M-by-N:

If $M > 1$ and $N > 1$, *output word length* = *input word length* + $\text{floor}(\log_2((M-1)(N-1))) + 1$

If $M > 1$ and $N = 1$, *output word length* = *input word length* + $\text{floor}(\log_2(M-1)) + 1$

If $M = 1$ and $N > 1$, *output word length* = *input word length* + $\text{floor}(\log_2(N-1)) + 1$

output fraction length = *input 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 Video and Image Processing Blockset is 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.

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-D DCT

[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.

See Also

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

Purpose Compute 2-D FFT of input

Library Transforms

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	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
Output	2-D FFT of the input	Same as Input port	Yes

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

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

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. This means that the m th row element is located at the k th position, where k is the bit reversed value of m . Also, the n th column element is located 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 With the 2-D FFT block, linearly ordering the output requires a butterfly operation. So, it might be better to output in bit-reversed order in some situations.

For more information ordering of the output, see “Bit-Reversed Order” on page 2-44. The 2-D FFT block bit-reverses the order of the columns as well as the rows.

Algorithms Used for FFT Computation

Depending on whether the block input is floating-point or fixed-point, real or complex, and whether you want the output in linear or bit-reversed order, 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

Floating-Point Signals

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

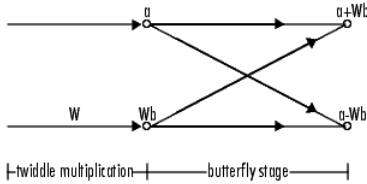
Fixed-Point Data Types

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 2-47.

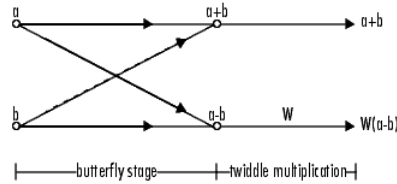
Inputs to the 2-D FFT block are first cast to the output data type and stored in the output buffer. Each butterfly stage then processes signals in the accumulator data type, with the final output of the butterfly being cast back into the output data type. A twiddle factor is multiplied in before each butterfly stage in a decimation-in-time FFT, and after each butterfly stage in a decimation-in-frequency FFT.

2-D FFT

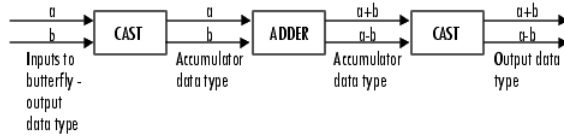
Decimation-in-time FFT



Decimation-in-frequency FFT



Butterfly stage data types



Twiddle multiplication data types



The output of the multiplier is in the accumulator data type since 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

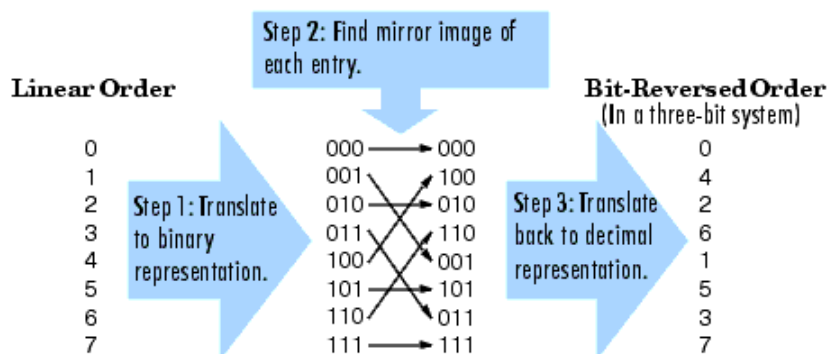
Bit-Reversed Order

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, since the three-bit binary representation of one, 001, is the mirror image of the three-bit binary

representation of four, 100. In the following diagram, the row indices are 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.

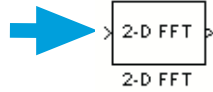
The row indices are now in bit-reversed order.



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 the columns as well as 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

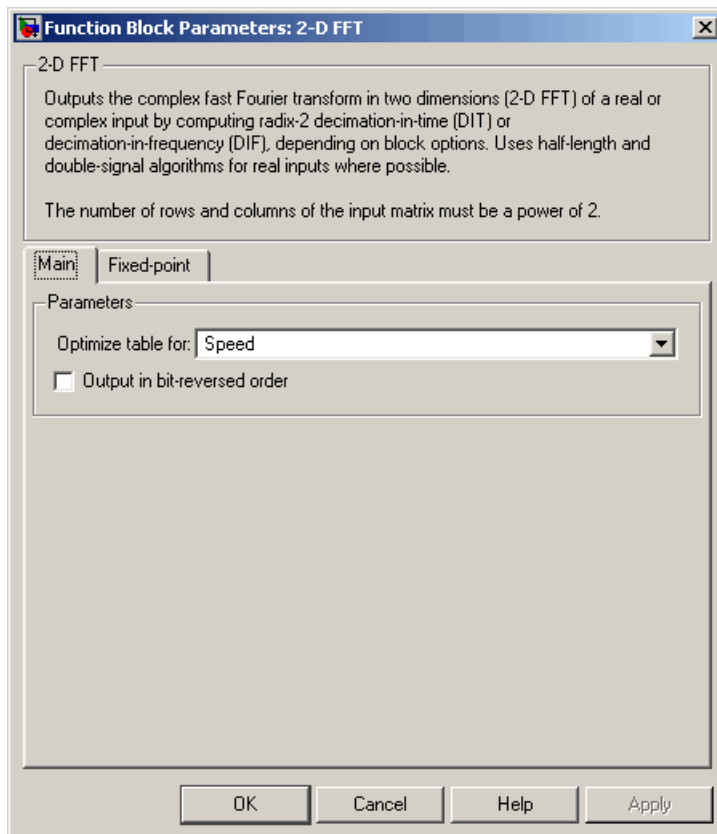
		0	1	2	3	4	5	6	7	
Linearly ordered row and column indices		↓	↓	↓	↓	↓	↓	↓	↓	
0 →	[245	13.9-0.4i	10-5i	-15.9+21.6i	-13	-15.9-21.6i	10+5i	13.9]
1 →		-4.3-10.3i	-27.6-6.6i	-5.6+13.1i	-3.4+8.7i	1.1-i	-2.6i	-11.5-11i	6.2+13i	
2 →		18-5i	-4.3-10.4i	19-24i	12.4-11.4i	6-3i	-5.7+16.4i	5+4i	5.5+1.4i	
3 →		8.4-2.4i	-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 of the table of twiddle factor values for Speed or Memory. This parameter must be set to Speed for fixed-point signals.

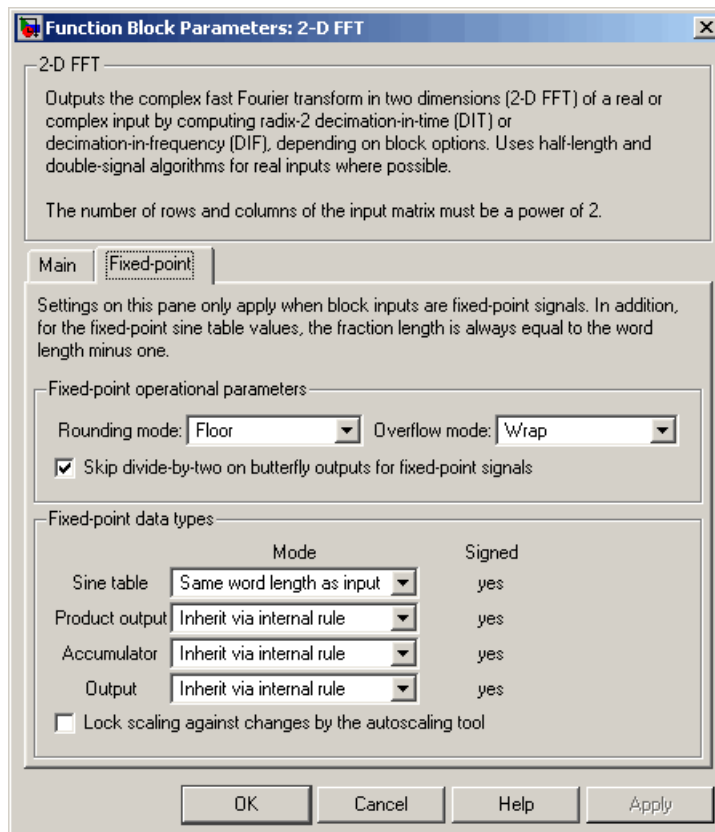
Output in bit-reversed order

Designate the order of the output channel elements relative to the ordering of the input elements. When selected, the output channel

2-D FFT

elements are in bit-reversed order relative to the input ordering. Otherwise, the output column elements are linearly ordered relative to the input ordering. Linearly ordering the output requires extra data sorting manipulation. For more information, see “Bit-Reversed Order” on page 2-44.

The **Fixed-point** 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; they always round to Nearest.

Overflow mode

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

Skip divide-by-two on butterfly outputs for fixed-point signals

When this parameter is selected, no scaling occurs. When this parameter is not selected, the output of each butterfly of the FFT is divided by two for fixed-point signals.

Sine table

Choose how to specify the word length of the values of the sine table. The fraction length of the sine table values is always equal to the word length minus one:

- When you select `Same word length as input`, the word length of the sine table values match that of the input to the block.
- 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 are 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 2-43 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 `Inherit via internal rule`, the product output word length and fraction length are automatically set according to the following equations:

*ideal product output word length =
output word length + sine table values word length*

*ideal product output fraction length =
output fraction length + sine table values 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 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 Video and Image Processing Blockset 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 2-43 and “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block:

- When you select **Inherit via internal rule**, the accumulator word length and fraction length are automatically set according to the following equations:

ideal accumulator word length = product output word length + 1

ideal accumulator fraction length = product output fraction length

- 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 Video and Image Processing Blockset is 0.

Output

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

- When you select `Inherit` via `internal rule`, the output word length and fraction length are automatically set according to the following equations, where the input matrix is M-by-N:

If $M > 1$ and $N > 1$, *output word length = input word length + floor(log₂((M-1)(N-1)))+1*

If $M > 1$ and $N = 1$, *output word length = input word length + floor(log₂(M-1))+1*

If $M = 1$ and $N > 1$, *output word length = input word length + floor(log₂(N-1))+1*

output fraction length = input 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 Video and Image Processing Blockset is 0.

See Also

2-D DCT	Video and Image Processing Blockset
2-D IDCT	Video and Image Processing Blockset
2-D IFFT	Video and Image Processing Blockset
FFT	Signal Processing Blockset
IFFT	Signal Processing Blockset
Pad	Signal Processing Blockset

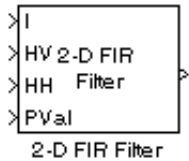
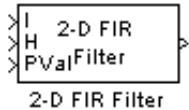
2-D FFT

<code>bitrevorder</code>	Signal Processing Toolbox
<code>fft</code>	Signal Processing Toolbox
<code>ifft</code>	Signal Processing Toolbox

Purpose Perform 2-D FIR filtering on input matrix

Library Filtering

Description The 2-D 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

2-D FIR Filter

Port	Input/Output	Supported Data Types	Complex Values Supported
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

If the data type of the input is floating point, the output of the block is 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 aren't sure 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. Later, you learn how to use these variables in the block mask.

Use the **Coefficient source** parameter to specify how to define your filter coefficients. If you select the **Separable filter coefficients** check box and, for the **Coefficient source** parameter, you select 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 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, for the **Coefficient source** parameter, you select 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, for the **Coefficient source** parameter, you select 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, for the **Coefficient source** parameter, you select 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 dimensions of the output are dictated by the **Output size** parameter and the sizes of the inputs at ports I and H. 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, for the **Output size** parameter, you choose Full, the output has dimensions $(M_i + M_h - 1, N_i + N_h - 1)$. If, for the **Output size** parameter, you choose Same as input port I, the output has the same dimensions as the input at port I. If, for the **Output size** parameter, you choose 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 $\text{all}(\text{size}(I) < \text{size}(H))$, 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

2-D FIR Filter

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, for the **Pad value source** parameter, you select **Input port**, the **PVal** port appears on the block. Use this port to specify the constant value with which to pad your matrix.

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

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

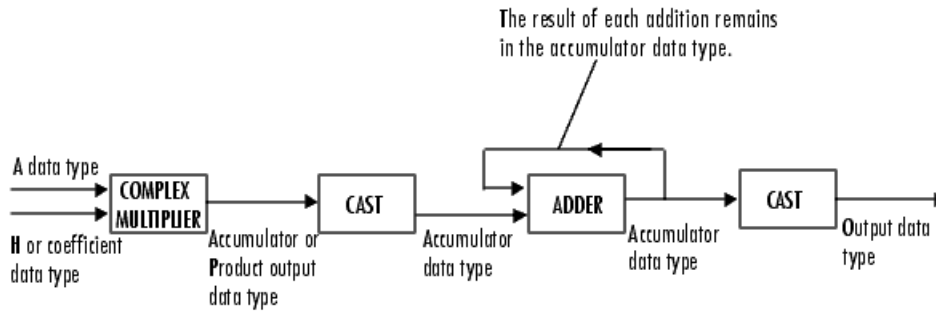
where $0 \leq i < Ma + Mh - 1$ and $0 \leq j < Na + Nh - 1$. If you select **Correlation** and set the **Output size** parameter to **Full**, the block filters your input using the following algorithm

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

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

Fixed-Point Data Types

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



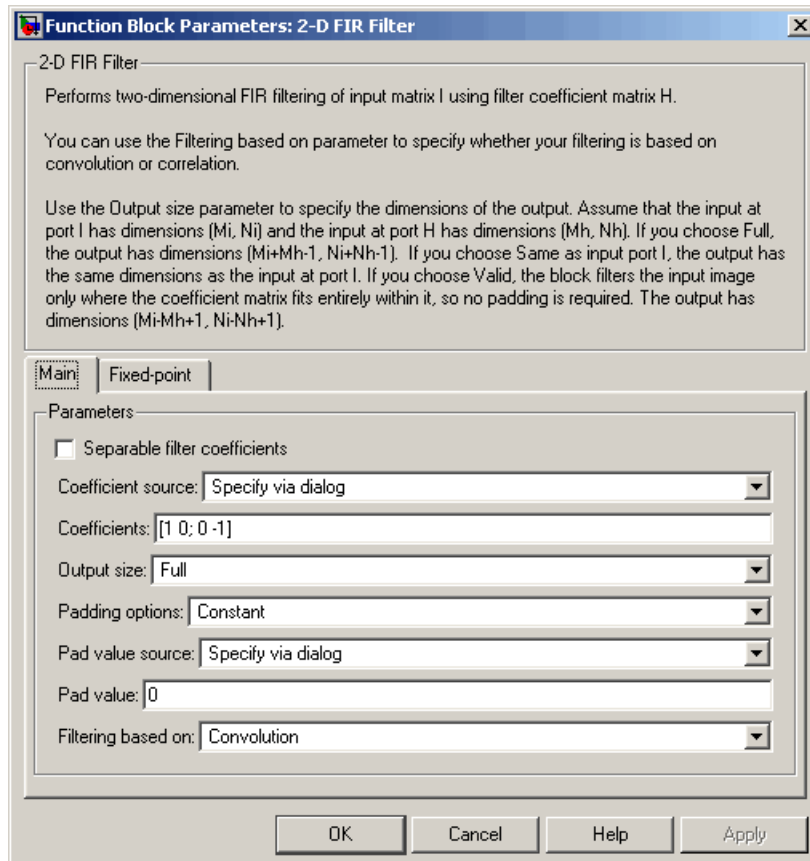
You can set the coefficient, product output, accumulator, and output data types in the block mask as discussed in “Dialog Box” on page 2-58.

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 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 is visible if you clear the **Separable filter coefficients** check box and, for the **Coefficient source** parameter, you select **Specify via dialog**. Tunable.

Vertical coefficients (across height)

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

Horizontal coefficients (across width)

Enter the vector of horizontal filter coefficients for your separable filter. This parameter is visible if you select the **Separable filter coefficients** check box and, for the **Coefficient source** parameter, you select **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 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 is visible if, for the

2-D FIR Filter

Output size parameter, you select 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 is visible if, for the **Padding options** parameter, you select Constant.

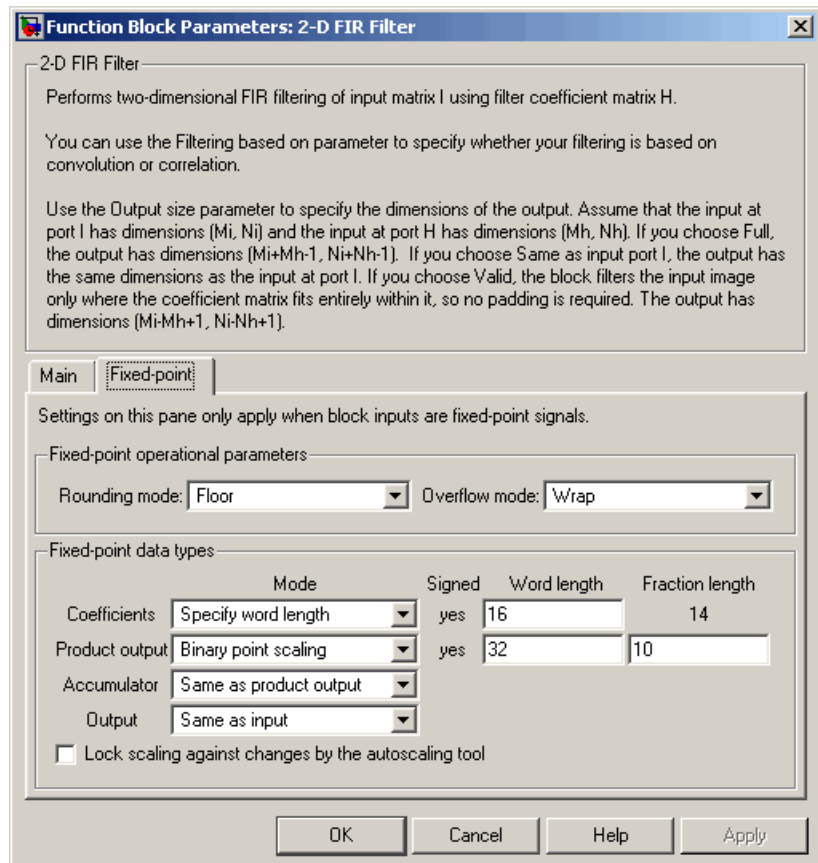
Pad value

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.

Filtering based on

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

The **Fixed-point** pane of the 2-D FIR Filter 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.

Coefficients

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

2-D FIR Filter

- 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 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. 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 **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 Video and Image Processing Blockset is 0.

The filter coefficients do not obey the **Rounding mode** and the **Overflow mode** parameters; they are 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 2-56 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. The bias of all signals in Video and Image Processing Blockset 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 2-56 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 0.

2-D FIR Filter

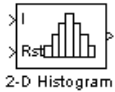
Lock scaling against changes by the autoscaling tool

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.

Purpose Generate histogram of each input matrix

Library Statistics

Description



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 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

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

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.

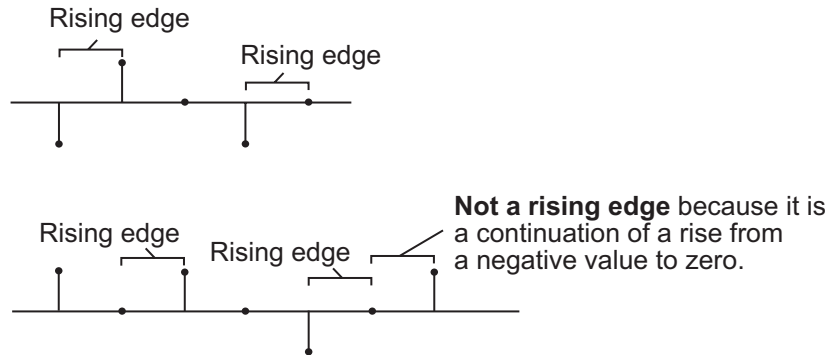
Resetting the Running Histogram

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.

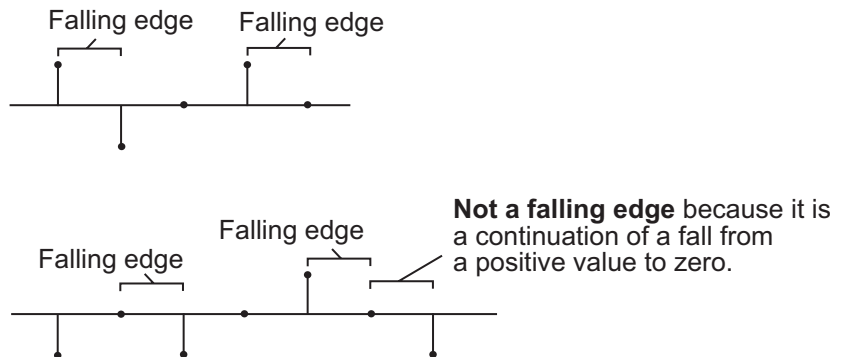
To enable the Rst port, select the **Reset port** parameter. You specify the reset event in the **Trigger type** parameter, and can be one of the following:

- **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

- 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)

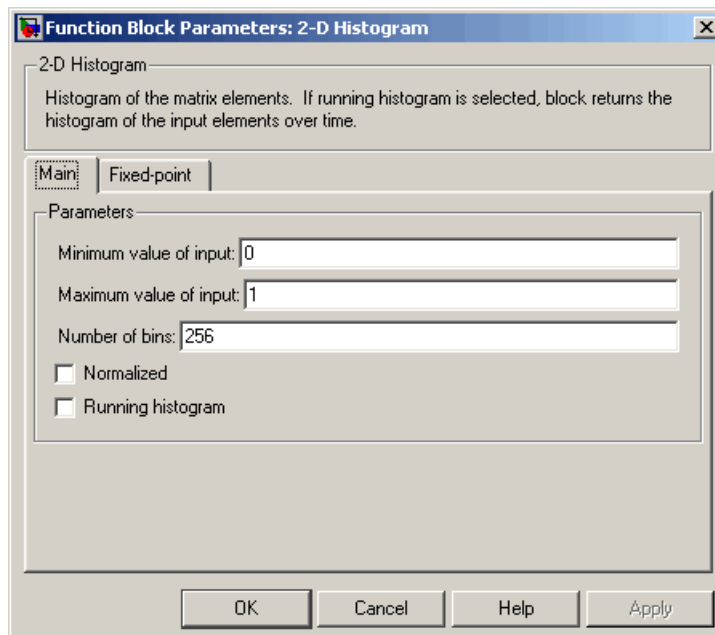
2-D Histogram

- 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 appears as shown in the following figure.



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

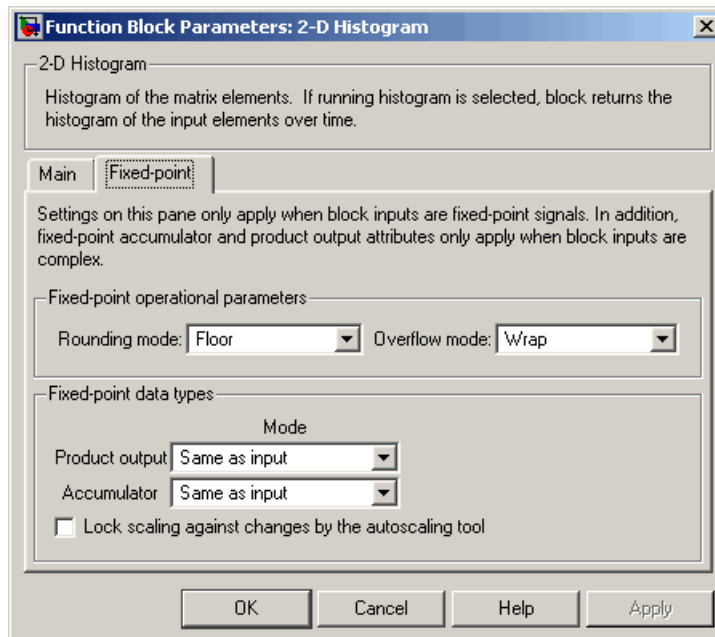
Enables the Rst input port when selected. The reset signal and the input data signal must be the same rate. This parameter is visible if you select the **Running histogram** check box.

Trigger type

The type of event that resets the running histogram. For more information, see "Resetting the Running Histogram" on page 2-68. This parameter is enabled only when you set the **Reset port** parameter.

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

2-D Histogram



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
2-D Correlation	Video and Image Processing Blockset
2-D Mean	Video and Image Processing Blockset
2-D Median	Video and Image Processing Blockset

2-D Histogram

2-D Standard Deviation	Video and Image Processing Blockset
2-D Variance	Video and Image Processing Blockset
Histogram	Signal Processing Blockset
Maximum	Signal Processing Blockset
Minimum	Signal Processing Blockset
hist	MATLAB
imhist	Image Processing Toolbox

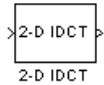
Purpose

Compute 2-D inverse discrete cosine transform (IDCT)

Library

Transforms

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.

2-D IDCT

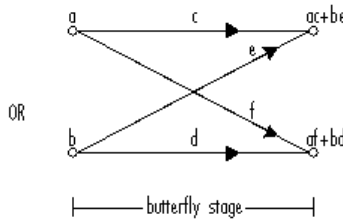
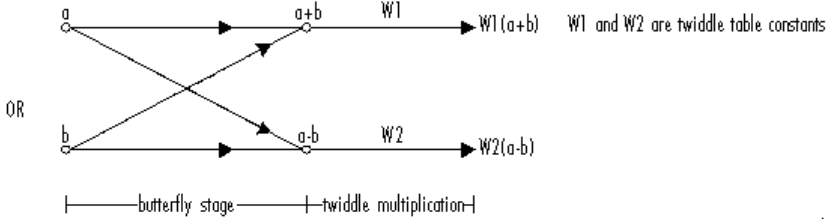
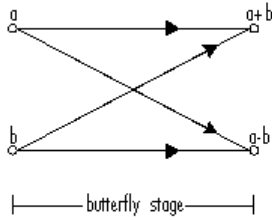
Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer	No
Output	2-D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

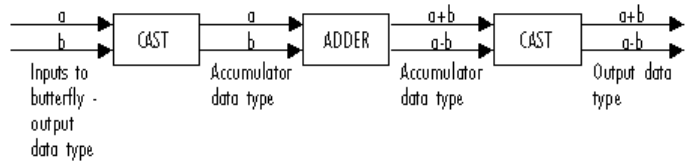
Use the **Sine and cosine computation** parameter to specify how the block computes the sine and cosine terms in the IDCT algorithm. If you select `Trigonometric fcn`, the block computes the sine and cosine values during the simulation. If you select `Table lookup`, the block computes and stores the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

Fixed-Point Data Types

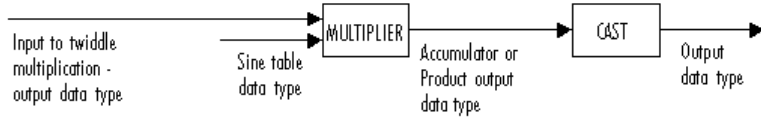
The following diagram shows the data types used in the 2-D IDCT block for fixed-point signals. Inputs are first cast to the output data type and stored in the output buffer. Each butterfly stage processes signals in the accumulator data type, with the final output of the butterfly being cast back into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



Butterfly Stage Data Types



Twiddle Multiplication Data Types

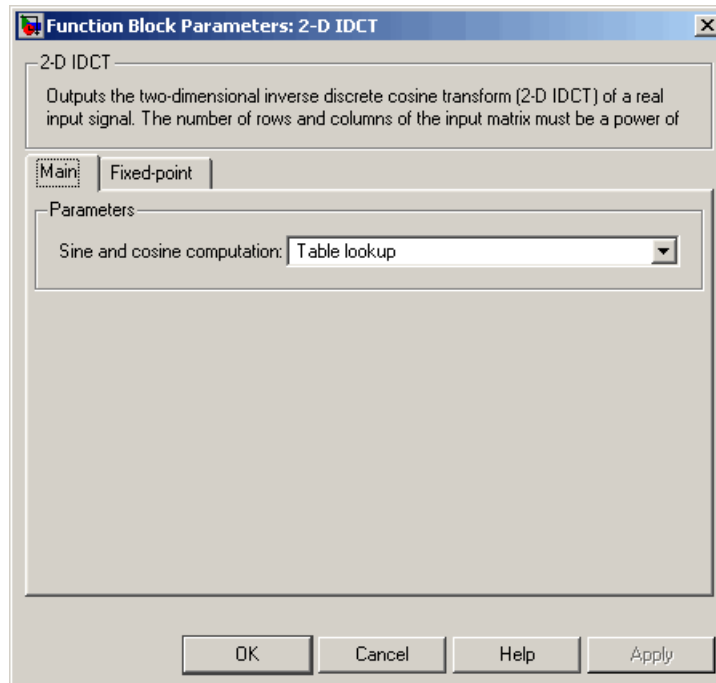


2-D IDCT

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

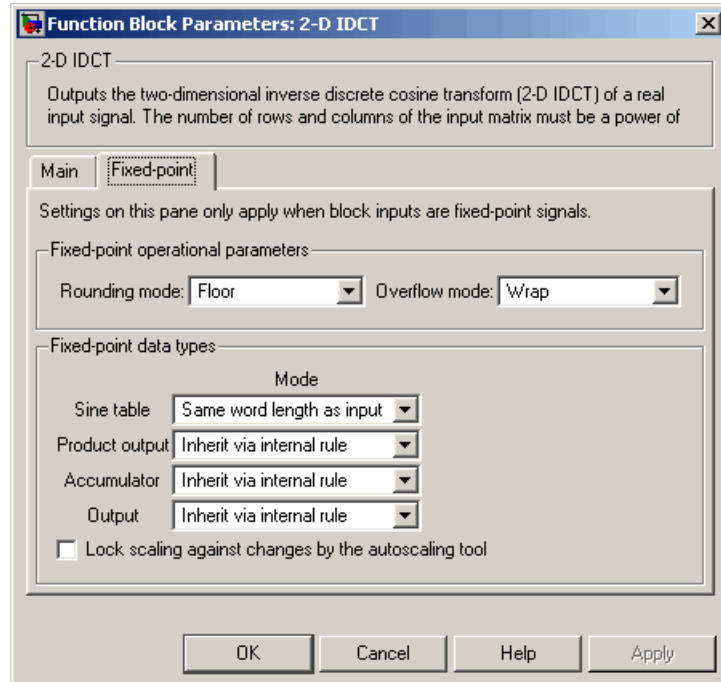
The **Main** pane of the 2-D IDCT dialog box appears as shown in the following figure.



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 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 **Fixed-point** 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 are always saturated and rounded to Nearest.

Overflow mode

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

Sine table

Choose how to specify the word length of the values of the sine table. The fraction length of the sine table values is always equal to the word length minus one:

- When you select **Same word length as input**, the word length of the sine table values match that of the input to the block.
- 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 are 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 2-76 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 **Inherit via internal rule**, the product output word length and fraction length are automatically set according to the following equations:

$$\textit{ideal product output word length} = \textit{output word length} + \textit{sine table values word length}$$

$$\textit{ideal product output fraction length} = \textit{output fraction length} + \textit{sine table values fraction length}$$

- 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 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 Video and Image Processing Blockset 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 2-76 and “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block:

- When you select `Inherit via internal rule`, the accumulator word length and fraction length are automatically set according to the following equations:

$$\textit{ideal accumulator word length} = \textit{product output word length} + 1$$

$$\textit{ideal accumulator fraction length} = \textit{product output fraction length}$$

- 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 Video and Image Processing Blockset is 0.

Output

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

- When you select **Inherit via internal rule**, the output word length and fraction length are automatically set according to the following equations, where the input matrix is M-by-N:

If $M > 1$ and $N > 1$, $\text{output word length} = \text{input word length} + \text{floor}(\log_2((M-1)(N-1))) + 1$

If $M > 1$ and $N = 1$, $\text{output word length} = \text{input word length} + \text{floor}(\log_2(M-1)) + 1$

If $M = 1$ and $N > 1$, $\text{output word length} = \text{input word length} + \text{floor}(\log_2(N-1)) + 1$

$\text{output fraction length} = \text{input fraction length}$

- 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 Video and Image Processing Blockset is 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.

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.

See Also

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

2-D IFFT

Purpose Compute 2-D IFFT of input

Library Transforms

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). Then 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 = ifft(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.

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
Output	2-D IFFT of the input	Same as Input port	Yes

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

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

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.

2-D IFFT

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 ordering of the column elements of the input is 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 2-44. The 2-D FFT block bit-reverses the order of the columns as well as 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

By default, the **Skip scaling** check box is not selected. If your signal is a floating-point signal, the block computes the scaled version of the IFFT. If your signal is a fixed-point signal, the output of each butterfly of the IFFT is divided by two. If you select the **Skip scaling** check box, the block computes the unscaled IFFT as defined by the following equation:

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

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

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

For floating-point signals:

2-D IFFT

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

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

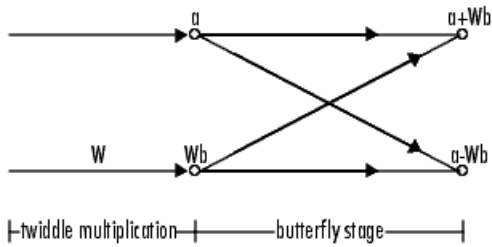
The following diagrams show the data types used in the IFFT block for fixed-point signals. You can set the sine table, accumulator, product

output, and output data types displayed in the diagrams in the IFFT dialog box as discussed in “Dialog Box” on page 2-91.

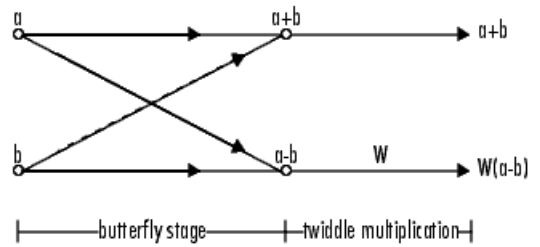
Inputs to the IFFT block are first cast to the output data type and stored in the output buffer. Each butterfly stage then processes signals in the accumulator data type, with the final output of the butterfly being cast back into the output data type. A twiddle factor is multiplied in before each butterfly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

2-D IFFT

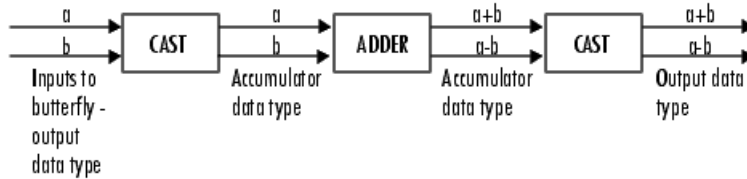
Decimation-in-time IFFT



Decimation-in-frequency IFFT



Butterfly stage data types



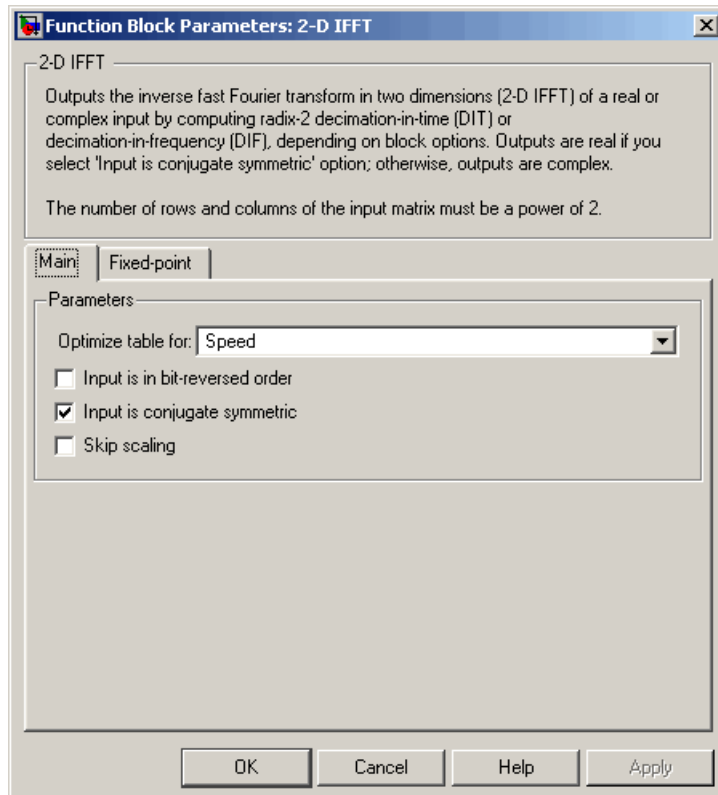
Twiddle multiplication data types



The output of the multiplier is in the accumulator data type since 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.

Dialog Box

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



Optimize table for

Select the optimization of 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 when the input is in bit-reversed order, and clear when the input is in

linear order. The block yields invalid outputs when you do not set this parameter correctly. See “Input Order” on page 2-86.

Input is conjugate symmetric

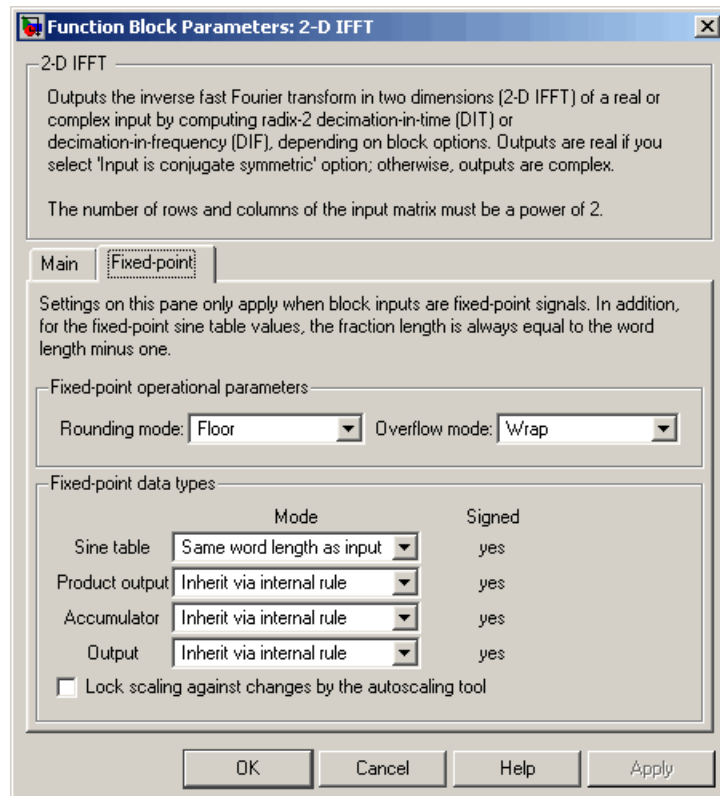
Select when the input to the block is both floating point and conjugate symmetric, and you want real-valued outputs. The block output is invalid when you set this parameter when the input is not conjugate symmetric. This parameter cannot be used for fixed-point signals.

Skip scaling

When you select this check box, no scaling occurs. When this parameter is cleared, scaling does occur:

- For floating-point signals, rather than computing the IDFT, the block computes a scaled version of the IDFT. This scaled version of the IDFT does not include the multiplication factor of $1/M$.
- For fixed-point signals, the output of each butterfly of the IFFT is divided by two.

The **Fixed-point** pane of the 2-D IFFT 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; they are always saturated.

Sine table

Choose how to specify the word length of the values of the sine table. The fraction length of the sine table values is always equal to the word length minus 1:

- When you select **Same word length as input**, the word length of the sine table values match that of the input to the block.
- 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 are 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 2-88 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 **Inherit via internal rule**, the product output word length and fraction length are automatically set according to the following equations:

$$\textit{ideal product output word length} = \textit{output word length} + \textit{sine table values word length}$$

$$\textit{ideal product output fraction length} = \textit{output fraction length} + \textit{sine table values fraction length}$$

- 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 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 Video and Image Processing Blockset 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 2-88 and “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block:

- When you select `Inherit via internal rule`, the accumulator word length and fraction length are automatically set according to the following equations:

$$\textit{ideal accumulator word length} = \textit{product output word length} + 1$$

$$\textit{ideal accumulator fraction length} = \textit{product output fraction length}$$

- 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 Video and Image Processing Blockset is 0.

Output

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

- When you select `Inherit via internal rule`, the output word length and fraction length are automatically set according to the following equations, where the input matrix is M-by-N:

$$\textit{If } M > 1 \textit{ and } N > 1, \textit{ output word length} = \textit{input word length} + \textit{floor}(\log_2((M-1)(N-1))) + 1$$

$$\textit{If } M > 1 \textit{ and } N = 1, \textit{ output word length} = \textit{input word length} + \textit{floor}(\log_2(M-1)) + 1$$

2-D IFFT

If $M=1$ and $N>1$, *output word length = input word length + floor(log 2(N-1))+1*

output fraction length = input 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 Video and Image Processing Blockset is 0.

See Also

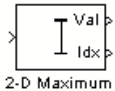
2-D DCT	Video and Image Processing Blockset
2-D FFT	Video and Image Processing Blockset
2-D IDCT	Video and Image Processing Blockset
FFT	Signal Processing Blockset
IFFT	Signal Processing Blockset
Pad	Signal Processing Blockset
bitrevorder	Signal Processing Toolbox
fft	Signal Processing Toolbox
ifft	Signal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

2-D Maximum (Obsolete)

Purpose Find maximum values in an input or sequence of inputs

Library vipobslib

Description



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

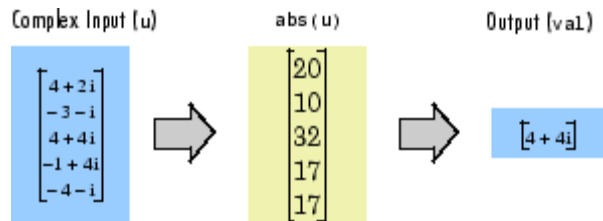
When **Mode** is set to **Value**, the block computes the maximum value in each column of the M -by- N input matrix u independently at each sample time.

```
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 $a^2 + b^2$.



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

2-D Maximum (Obsolete)

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.

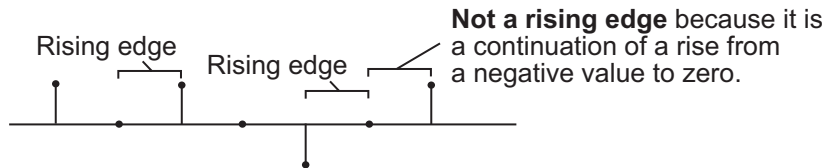
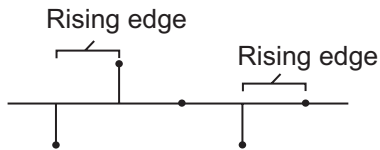
Resetting the Running Maximum

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.

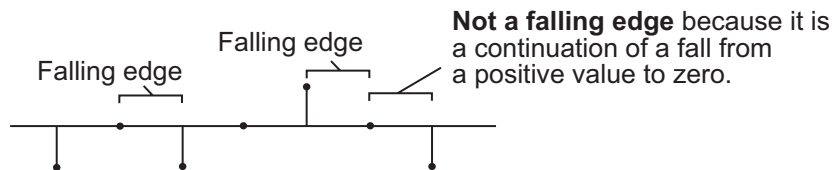
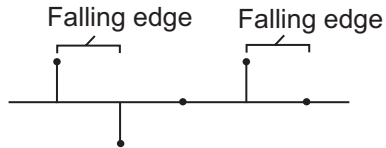
For sample-based inputs, a reset event causes the running maximum for each channel to be initialized to the value in the corresponding channel of the current input. For frame-based inputs, a reset event causes the running maximum for each channel to be initialized to the earliest value in each channel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



- 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 Maximum (Obsolete)

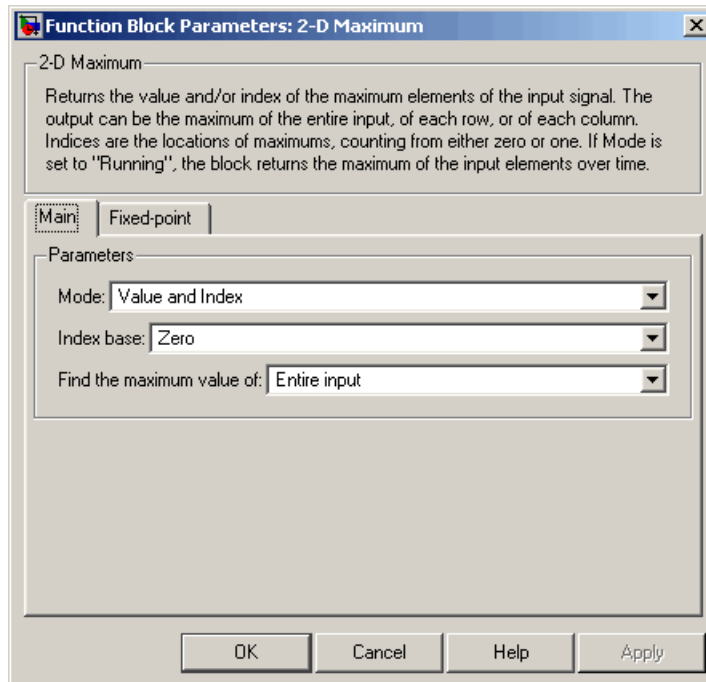
Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes,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

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 2-98. 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.

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

2-D Maximum (Obsolete)

Index base

Specify whether the index is zero based or one based.

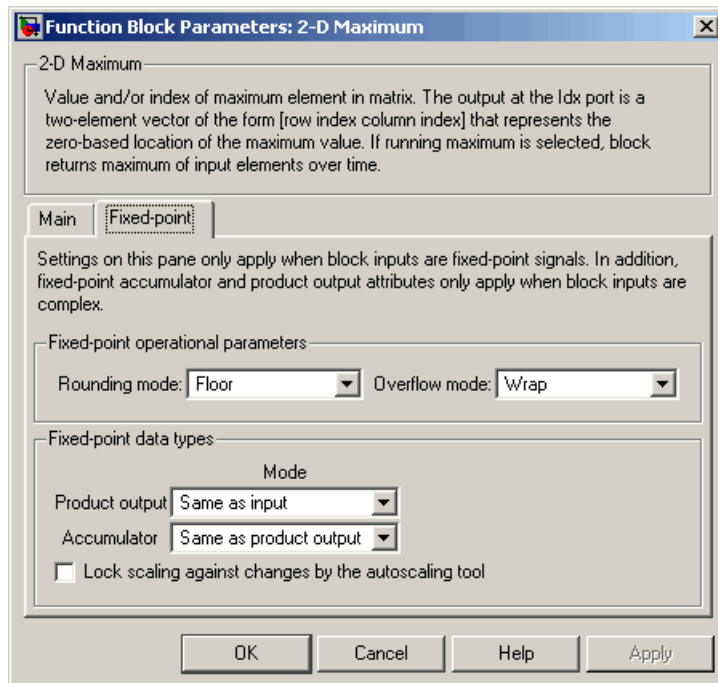
Find the maximum value of

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.



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 2-98. 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.

2-D Maximum (Obsolete)

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.

Lock scaling against changes by the autoscaling tool

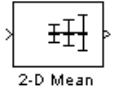
Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

2-D Mean

Purpose Find mean value of each input matrix

Library Statistics

Description 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

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

2-D Mean

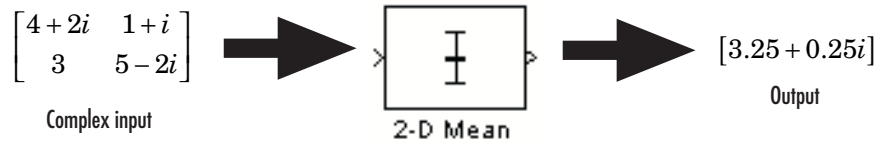
Port	Input/Output	Supported Data Types	Complex Values Supported
Output/Out	<p>Without ROI processing — Mean of each M-by-N input matrix or the mean for each element of a series of M-by-N inputs.</p> <p>With ROI processing — Vector of separate statistical values for each ROI or a scalar value that represents 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.



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.

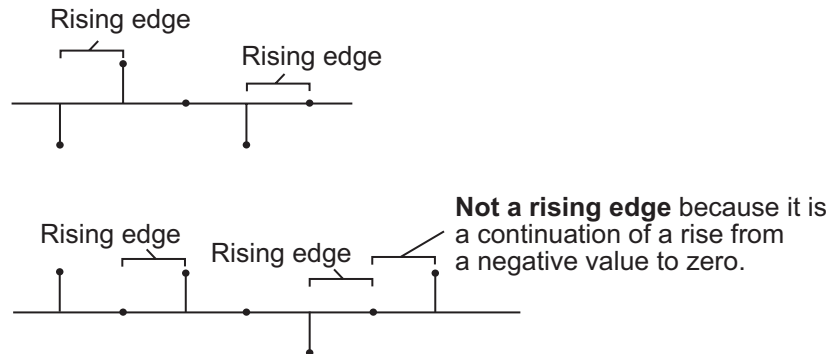
Resetting the Running Mean

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.

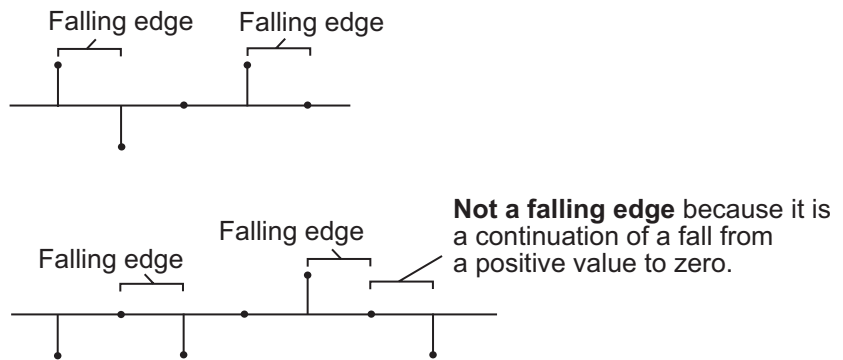
When the block is reset, the running mean associated with each element is initialized to the value in the corresponding location of the current input.

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)



- 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

2-D Mean

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.

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 Mean

Output = Individual statistics for each ROI

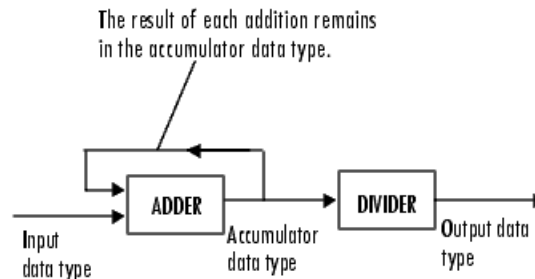
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

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

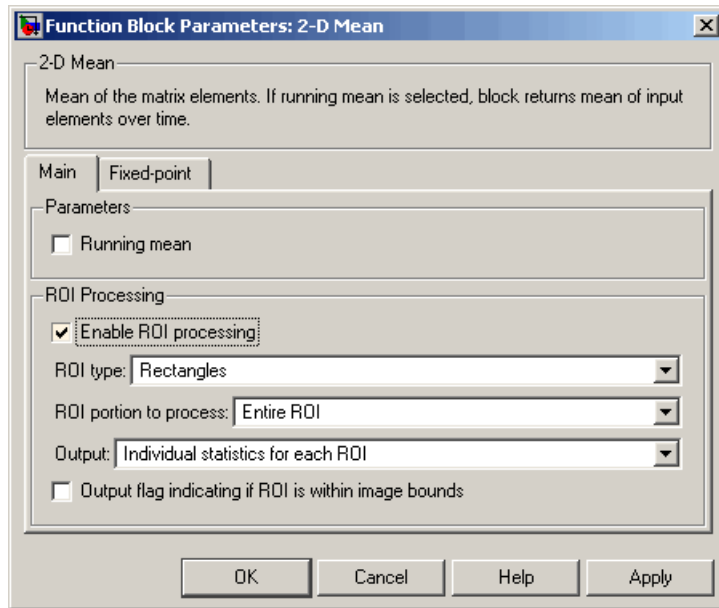
The following diagram shows the data types used in the 2-D Mean block for fixed-point signals.



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

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.

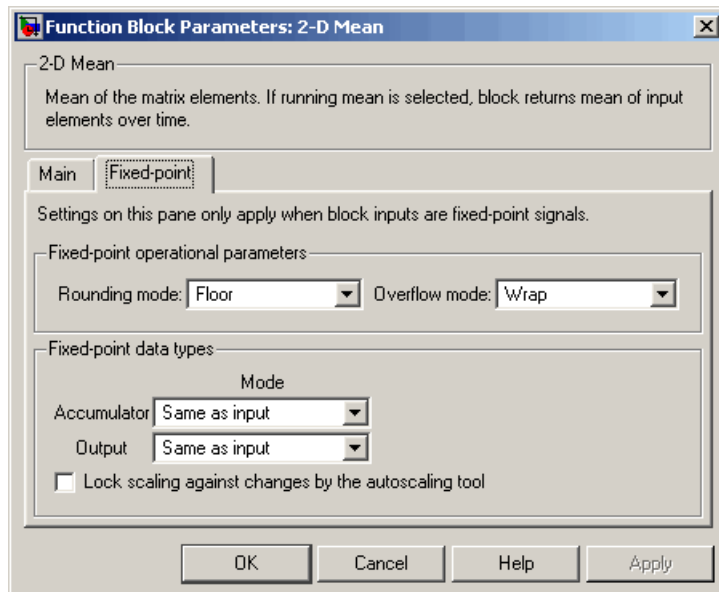
Output flag indicating if ROI is within image bounds

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 2-113. 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 2-113. 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.



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.

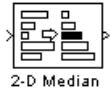
See Also

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

Purpose Find median value of each input matrix

Library Statistics

Description 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.

```
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.

2-D Median

```
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 2-123. 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

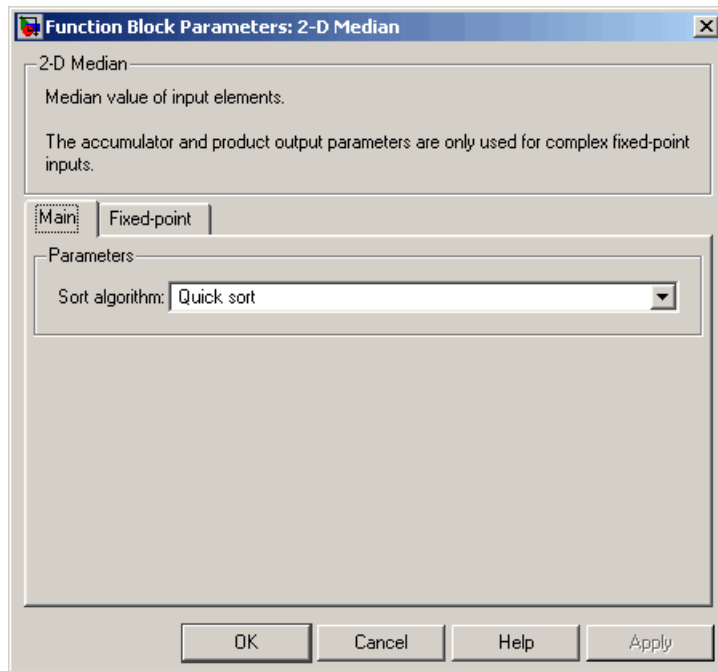
The accumulator and output data types and scalings are used for fixed-point signals when M is even. The result of the sum performed while calculating the average of the two central rows of the input matrix is stored in the accumulator data type and scaling. The total result of the average is then put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

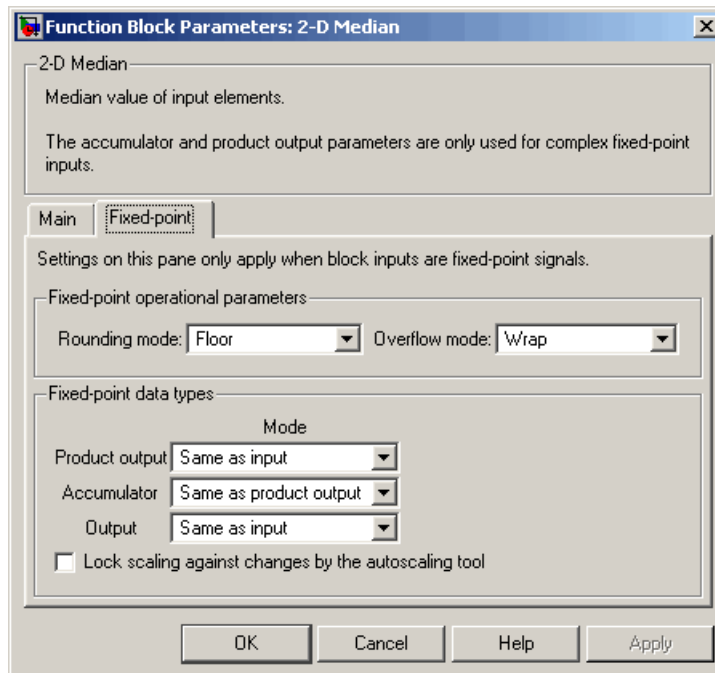


Sort algorithm

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

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

2-D Median



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 2-122 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.
- 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.

2-D Median

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.

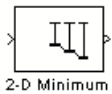
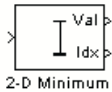
See Also

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

Purpose Find minimum values in an input or sequence of inputs

Library vipobslib

Description



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

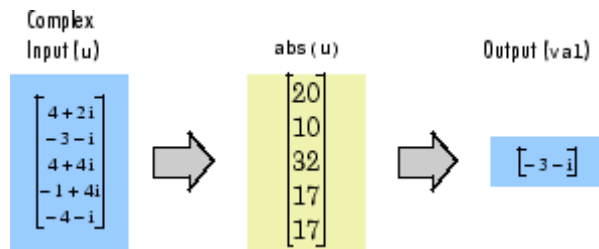
When **Mode** is set to **Value**, the block computes the minimum value in each column of the M -by- N input matrix u independently at each sample time.

```
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

2-D Minimum (Obsolete)

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.

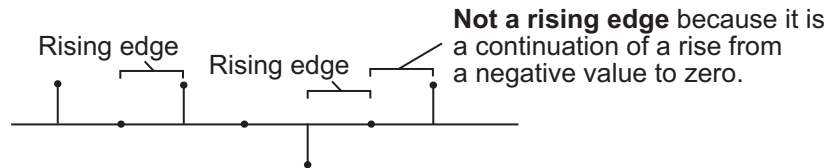
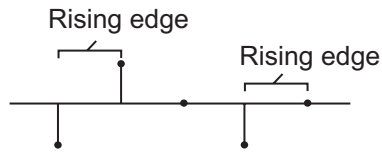
Resetting the Running Minimum

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.

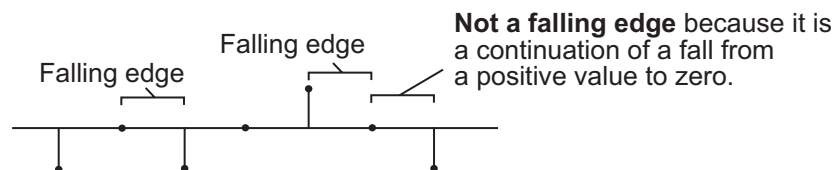
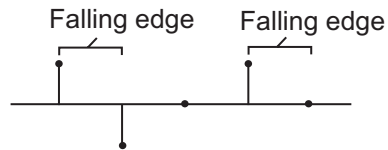
When the block is reset for sample-based inputs, the running minimum for each channel is initialized to the value in the corresponding channel of the current input. For frame-based inputs, the running minimum for each channel is initialized to the earliest value in each channel of the current input.

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

2-D Minimum (Obsolete)

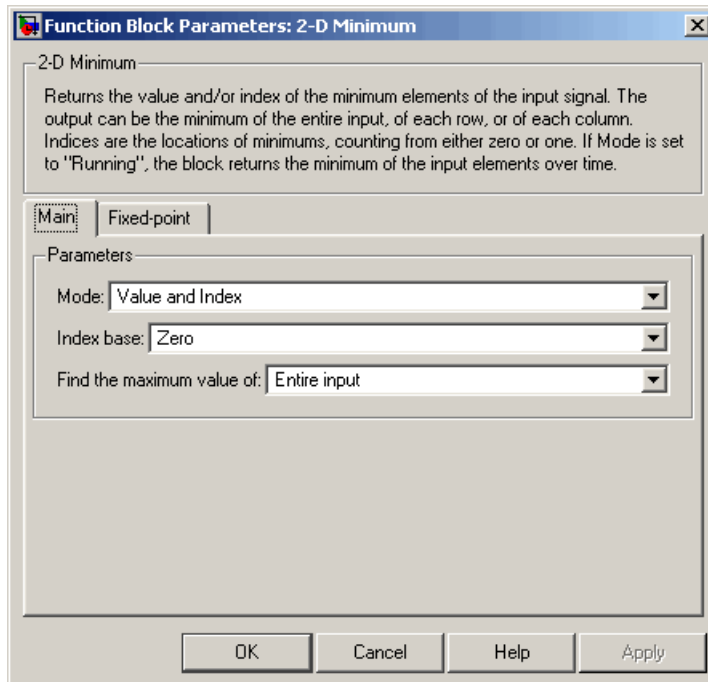
Note When running simulations in the Simulink MultiTasking mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes,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

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 2-128. 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.

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

2-D Minimum (Obsolete)

Index base

Specify whether the index is zero based or one based.

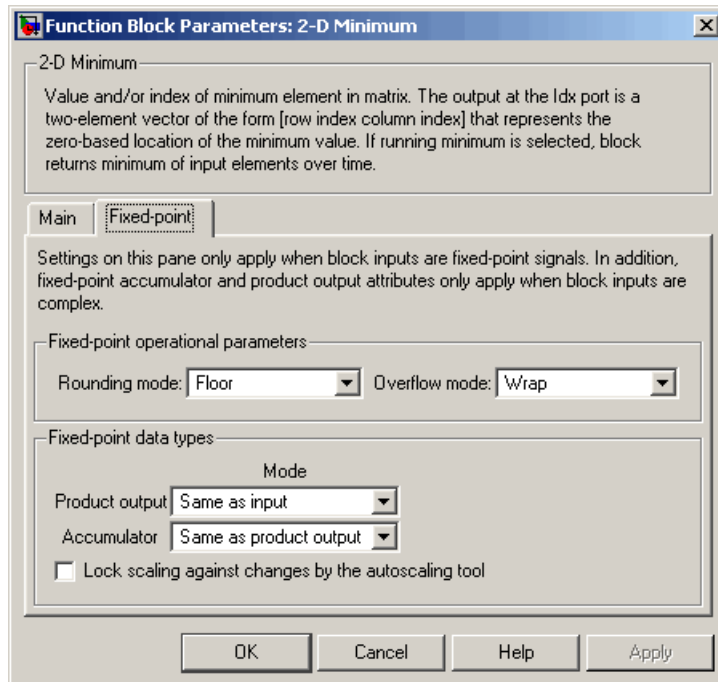
Find the maximum value of

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.



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 2-128. 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.

2-D Minimum (Obsolete)

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.

Lock scaling against changes by the autoscaling tool

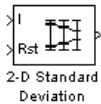
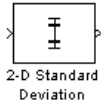
Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink documentation.

2-D Standard Deviation

Purpose Find standard deviation of each input matrix

Library Statistics

Description



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.

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

2-D Standard Deviation

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

2-D Standard Deviation

Port	Input/Output	Supported Data Types	Complex Values Supported
Output / Out	Without ROI processing — Standard deviation of each M-by-N input matrix, or the standard deviation of 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 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

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

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

2-D Standard Deviation

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

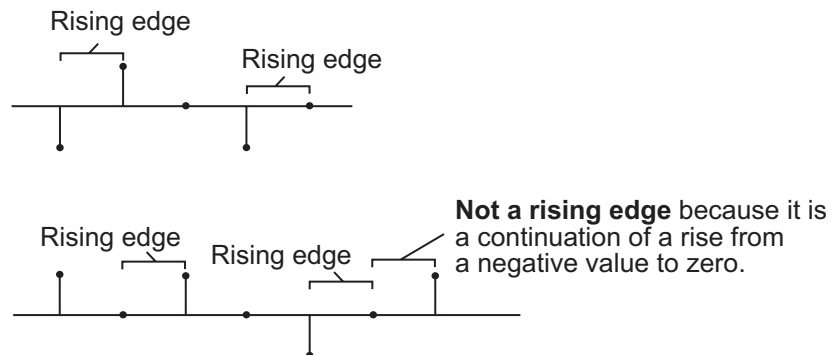
For the next input, the block computes the standard deviation for each element of the first three inputs, and so on.

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.

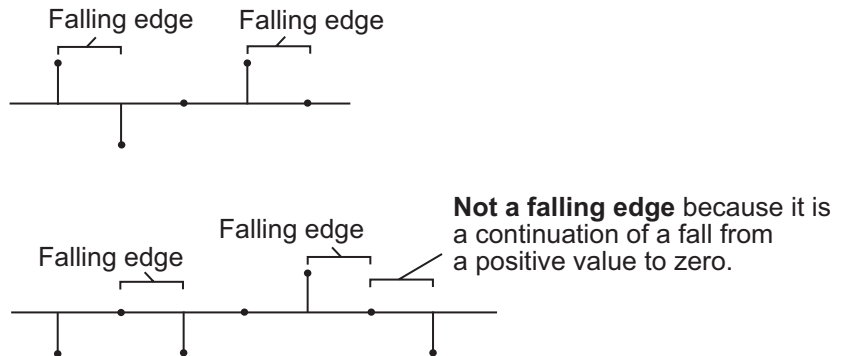
You specify the reset event using the **Reset port** parameter:

- 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)



- 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.

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

2-D Standard Deviation

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.

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

Output = Individual statistics for each ROI

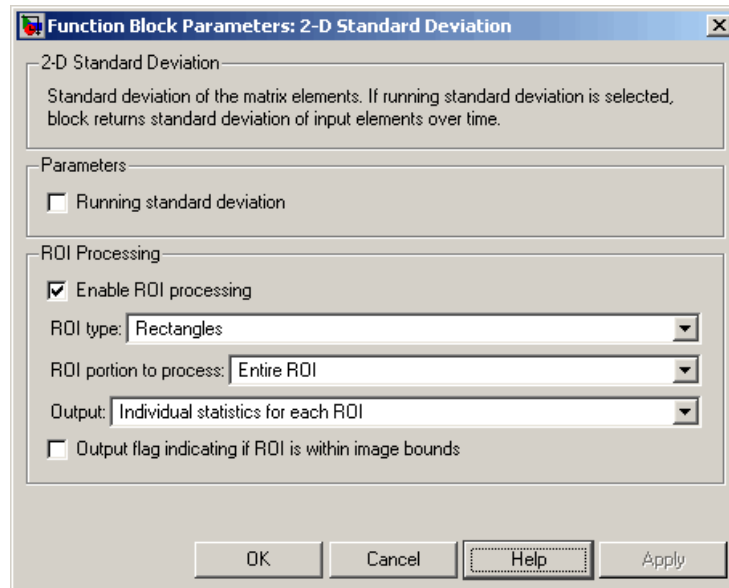
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

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.

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

2-D Standard Deviation

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.

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 2-143. 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 2-143. This parameter is visible if, for the **ROI type** parameter, you select Label matrix.

See Also

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

2-D Variance

Purpose Compute variance of each input matrix

Library Statistics

Description 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.



Note This block's functionality is different from the Signal Processing Blockset Variance block, which computes the variance of each column in the input.

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

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

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:

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

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,

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

2-D Variance

and outputs

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

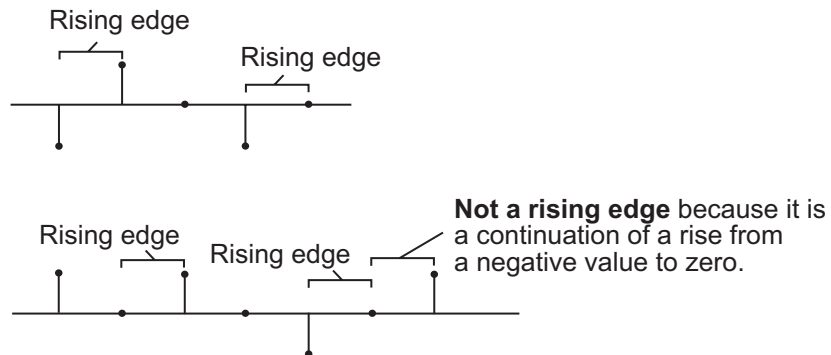
For the next input, the block computes the variance for each element of the first three inputs, and so on.

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.

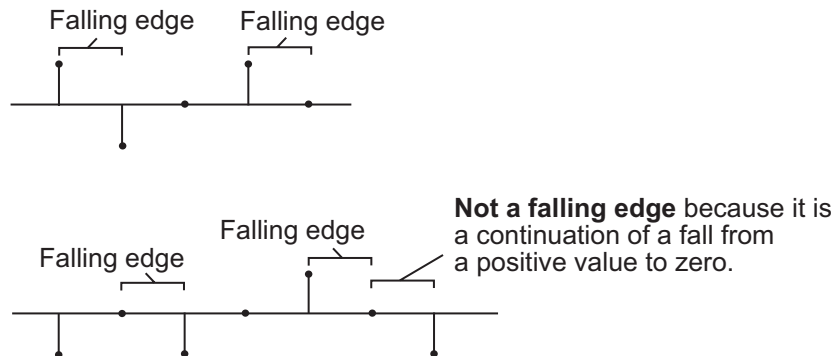
You specify the reset event using the **Reset port** parameter:

- 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:

- 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.

2-D Variance

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.

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 Variance

Output = Individual Statistics for Each ROI

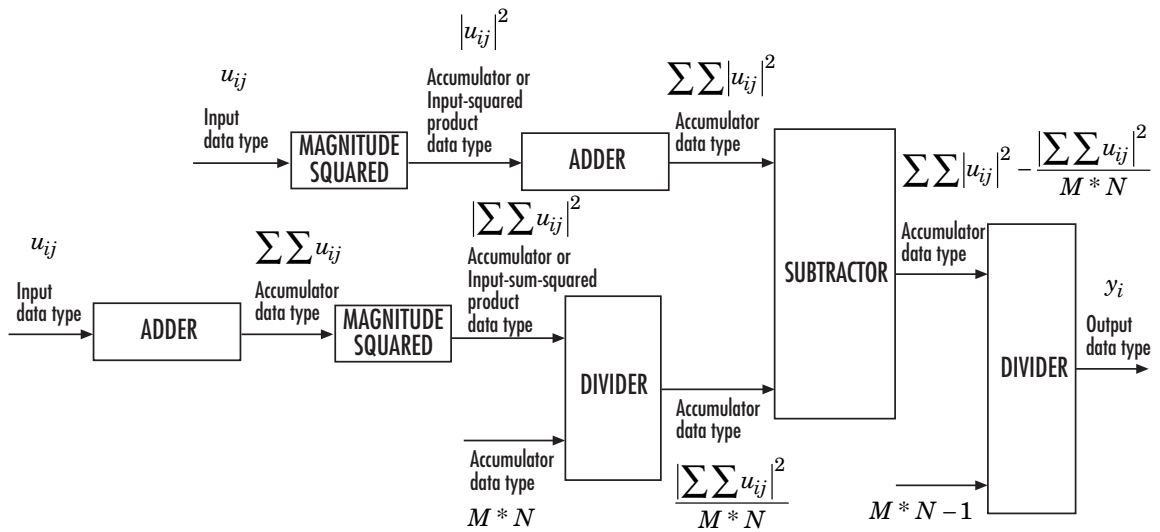
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

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.

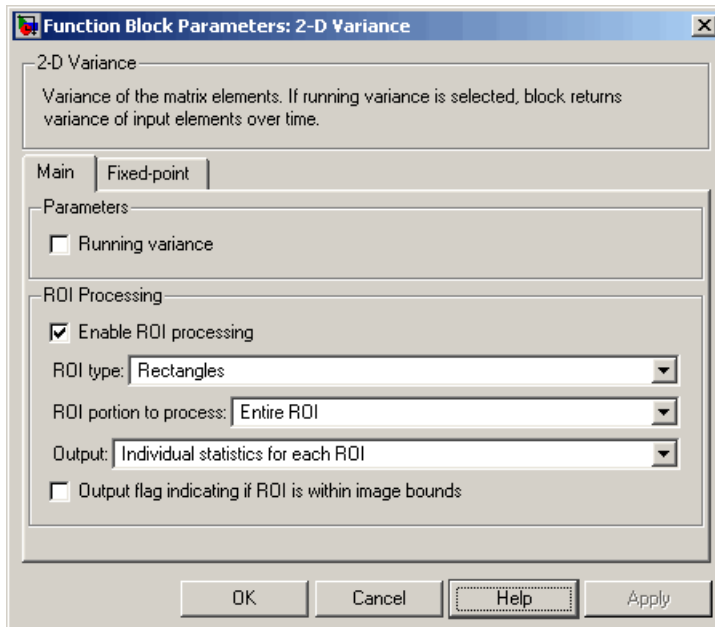


The results of the magnitude squared calculations in the preceding diagram are in the product output data type. You can set the accumulator, product output, and output data types in the dialog box.

2-D Variance

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

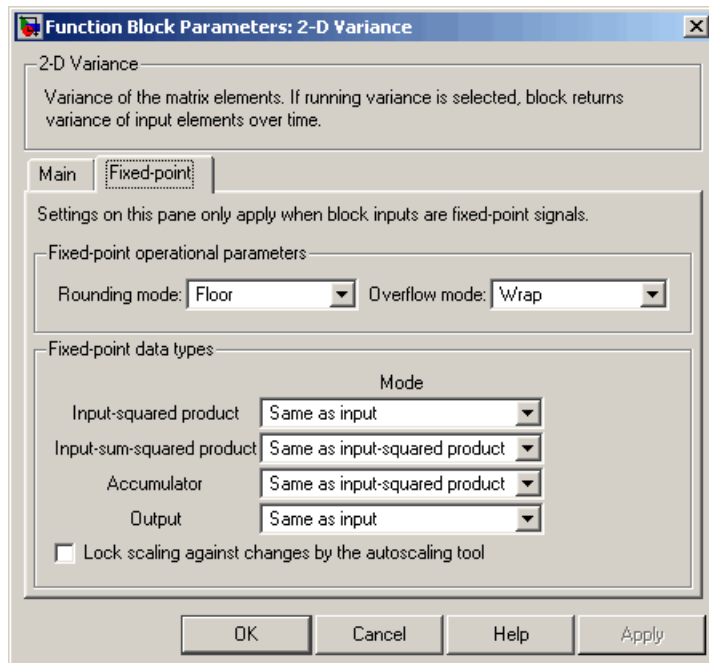
If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in “ROI Processing” on page 2-155. This parameter is visible if, for the **ROI type** parameter, you select Rectangles or Lines.

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 2-155. 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



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 2-158 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

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

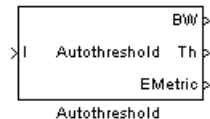
2-D Autocorrelation	Video and Image Processing Blockset
2-D Correlation	Video and Image Processing Blockset
2-D Histogram	Video and Image Processing Blockset
2-D Mean	Video and Image Processing Blockset
2-D Median	Video and Image Processing Blockset
2-D Standard Deviation	Video and Image Processing Blockset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

Autothreshold

Purpose Convert intensity image to binary image

Library Conversions

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 \leq and the input value is less 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.

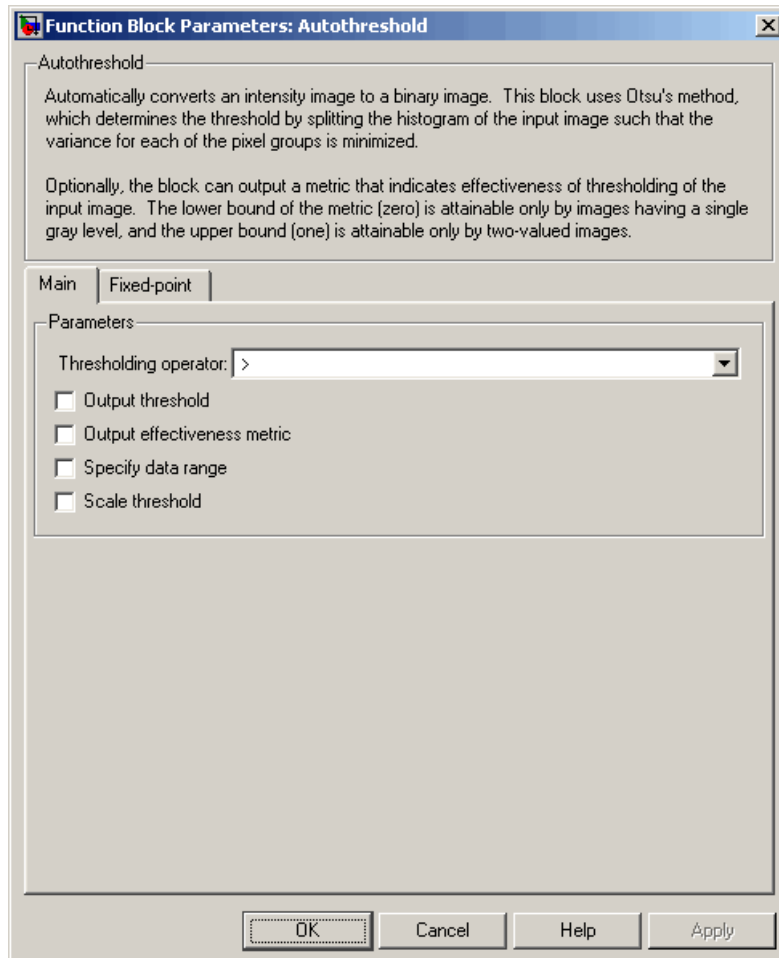
Use the **When data range is exceeded** parameter to specify the block's behavior when the input values are outside the expected range. The following options are available:

- **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

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

Autothreshold

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. Tunable.

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. Tunable.

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.

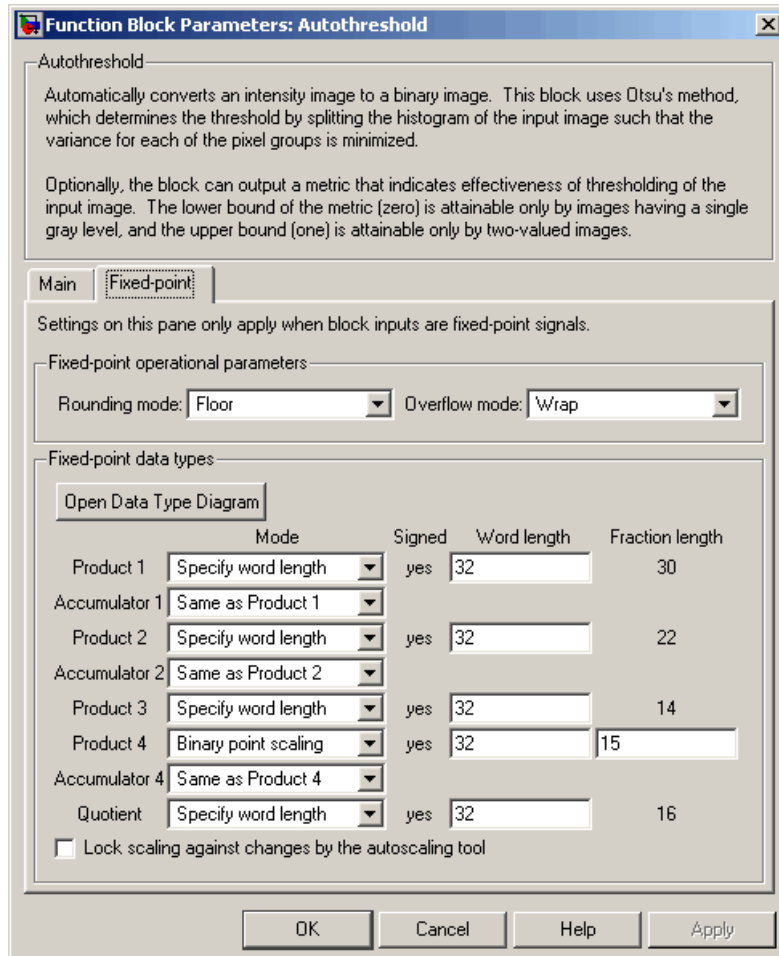
Scale threshold

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 result as the new threshold value. This parameter is visible if you select the **Scale threshold** check box.

The **Fixed-point** 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.



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 2-168. For this step, **Rounding mode** is always set to Nearest.

Overflow mode

Select the overflow mode for fixed-point operations.

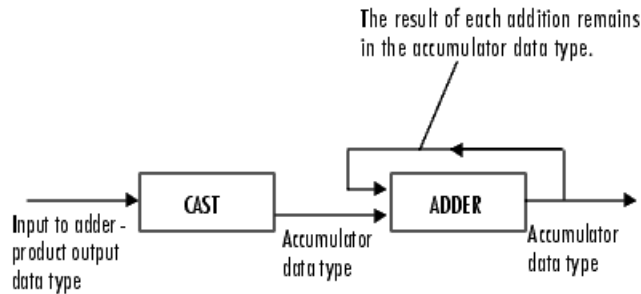
Product 1, 2, 3, 4



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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 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

Compare To Constant

Simulink

Relational Operator

Simulink

graythresh

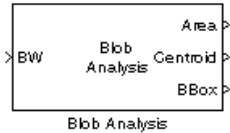
Image Processing Toolbox

Blob Analysis

Purpose Compute statistical values for labeled regions

Library Statistics

Description Use the Blob Analysis block to calculate statistics for labeled regions in a binary image. The block returns some quantities, such as the Centroid, as values that represent spatial coordinate locations. For information about how pixel and spatial coordinate systems are defined in Video and Image Processing Blockset, 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	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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

Main Pane

Use the check boxes to specify the statistics values you want the block to output. The following table summarizes the block's 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.

Check Box	Block Output
Area	Vector that represents the number of pixels in labeled regions
Centroid	<p>2-by-N matrix whose columns represent the coordinates of the centroid of each region, where N is the number of blobs. For example, if there are two blobs and 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>

Check Box	Block Output
Bounding box	<p>4-by-N matrix whose columns represent the coordinates of each bounding box, where N is the number of blobs. For example, suppose there are two blobs, where r and c define the row and column location of the upper-left corner of the bounding box and 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	Vector that represents the lengths of the major axes of ellipses that have the same normalized second central moments as the labeled regions
Minor axis length	Vector that represents the lengths of the minor axes of ellipses that have the same normalized second central moments as the labeled regions
Orientation	<p>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> $-\frac{\pi}{2} \text{ and } \frac{\pi}{2}$
Eccentricity	Vector that represents the eccentricities of ellipses that have the same second moments as the origin
Equivalent diameter squared	Vector that represents the equivalent diameters squared

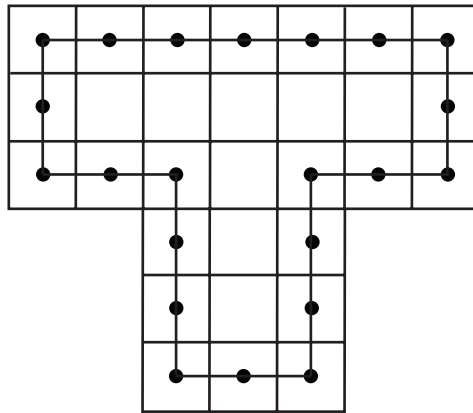
Blob Analysis

Check Box	Block Output
Extent	Vector that represents the results of dividing the areas of the blobs by the area of their bounding boxes
Perimeter	N-by-1 vector containing estimates of the perimeter lengths, in pixels, of each blob, where N is the number of blobs

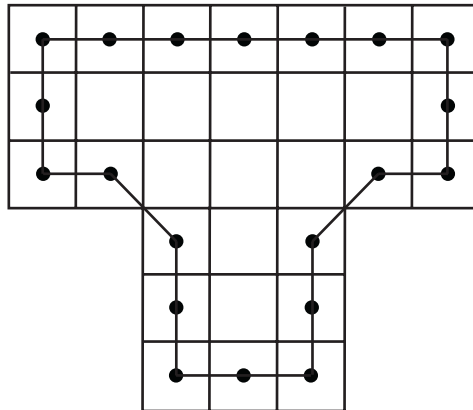
Use the **Statistics output data type** parameter to specify the data type of the outputs at the Centroid, MajorAxis, MinorAxis, Orientation, Eccentricity, Diameter², and Extent ports. If you select Fixed-point, the block cannot calculate the major axis, minor axis, orientation, or eccentricity and these check boxes become unavailable.

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. 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. The following figure illustrates how the block calculates the perimeter when you set the **Connectivity** parameter to 4.



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.



The block takes a different path around the blob and estimates the perimeter to be $18 + 2\sqrt{2}$.

Blob Analysis

If you select the **Output label matrix** check box, the block outputs the 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, at the Label port.

Blob Properties Pane

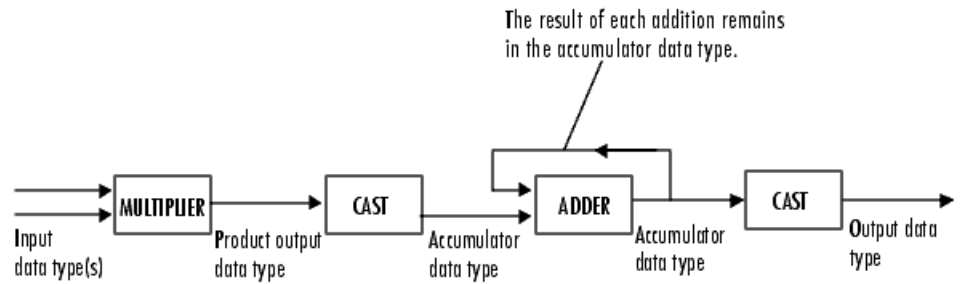
Use the **Maximum number of blobs** parameter to specify the maximum number of labeled regions in each input image. The block uses this value to preallocate vectors and matrices so that they are long enough to hold all of the statistical values. If you select the **Warn if maximum number of blobs is exceeded** check box, the block produces a warning if the number of blobs in an image is greater than the value you entered for the **Maximum number of blobs** parameter. If you select the **Output number of blobs found** check box, the block outputs a scalar value that represents the actual number of connected regions in each image at the Count port.

If you select the **Specify minimum blob area in pixels** and/or **Specify maximum blob area in pixels** check boxes, you can enter the minimum and maximum pixel area for the blobs. Blobs that do not meet this area criteria are not labeled. Select the **Exclude blobs touching image border** check box if you do not want to label blobs that contain at least one border pixel.

If the number of blobs found in an image is less than the scalar value entered for the **Maximum number of blobs** parameter, the block has empty spaces in the statistics output arrays. If you select the **Fill empty spaces in outputs** check box, the block fills the empty spaces with the scalar value you specify in the **Fill values** parameter. If you enter a vector for the **Fill values** parameter, 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. Otherwise, we recommend leaving it selected.

Fixed-Point Data Types

The following diagram shows the data types used in the Blob Analysis block for fixed-point signals.

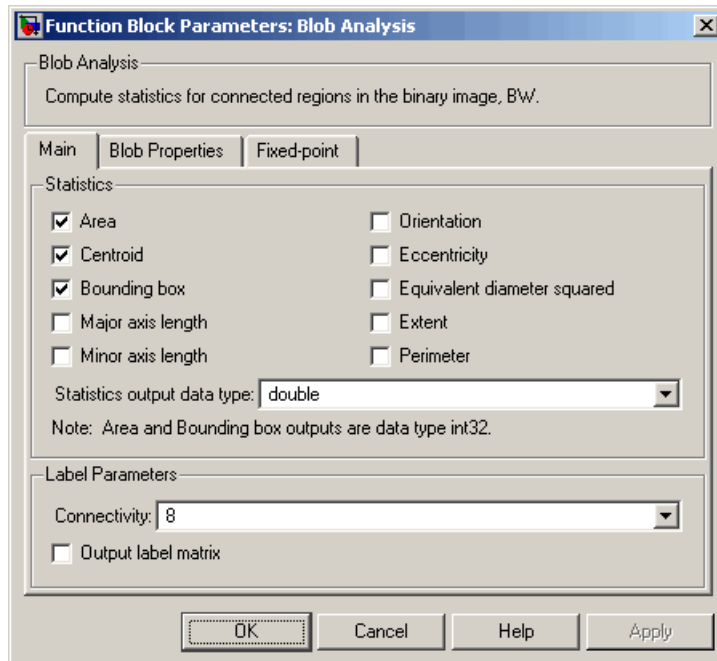


You can set the product output, accumulator, centroid output, equivalent diameter squared output, and extent output data types in the block mask as discussed in the next section.

Blob Analysis

Dialog Box

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



Area

Select this check box to output the area of the blobs.

Centroid

Select this check box to output the 2-by-N matrix whose columns represent the coordinates of the centroid of each labeled region.

Bounding box

Select this check box to output the coordinates of the bounding boxes.

Major axis length

Select this check box 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.

Minor axis length

Select this check box 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.

Orientation

Select this check box to output a vector whose values represent the angles between the major axes of the ellipses and the x -axis. The angle values are in radians and range between $-\pi/2$ and $\pi/2$.

Eccentricity

Select this check box to output a vector whose values represent the eccentricities of the ellipses that have the same second moments as the origin.

Equivalent diameter squared

Select this check box to output a vector whose values represent the equivalent diameters squared.

Extent

Select this check box to output a vector whose values represent the results of dividing the areas of the blobs by the area of their bounding boxes.

Perimeter

Select this check box to output a vector whose values represent estimates of the perimeter lengths, in pixels, of each blob.

Statistics output data type

Specify the data type of the outputs at the Centroid, MajorAxis, MinorAxis, Orientation, Eccentricity, Diameter², and Extent ports.

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

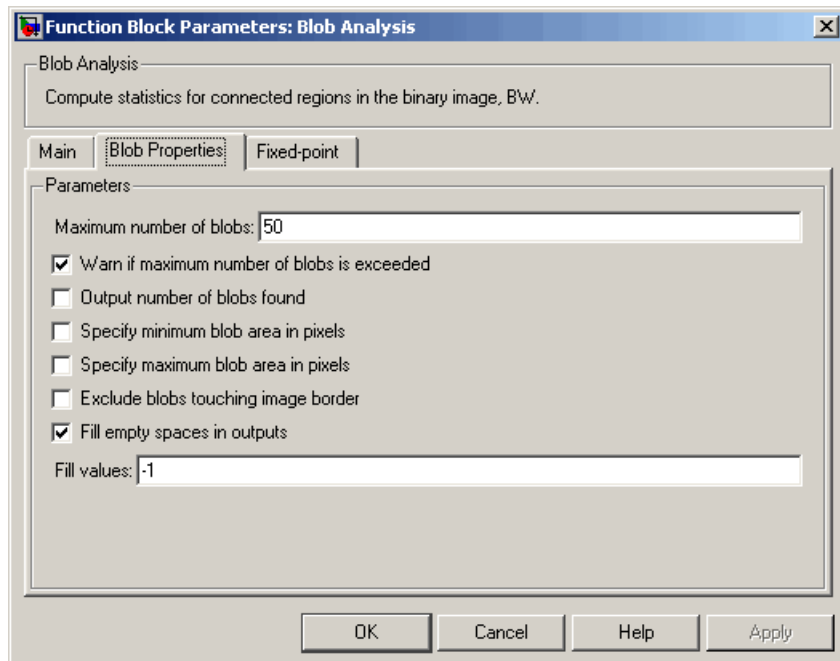
Blob Analysis

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 label matrix

If you select this check box, the block outputs the label matrix at the Label port.

The **Blob Properties** pane of the Blob Analysis dialog box appears as shown in the following figure.



Maximum number of blobs

Specify the maximum number of labeled regions in each input image.

Warn if maximum number of blobs is exceeded

If you select this check box, the block produces a warning if the number of blobs in an image is greater than the value you entered for the **Maximum number of blobs** parameter.

Output number of blobs found

If you select this check box, the block outputs a scalar value that represents the actual number of labeled regions in each image at the Count port.

Specify minimum blob area in pixels

If you select this check box, you can enter the minimum blob area in the edit box that appears beside the check box. Blobs that do not meet this criteria are not labeled.

Specify maximum blob area in pixels

If you select this check box, you can enter the maximum blob area in the edit box that appears beside the check box. Blobs that do not meet this criteria are not labeled.

Exclude blobs touching image border

Select this check box if you do not want to label blobs that contain at least one border pixel.

Fill empty spaces in outputs

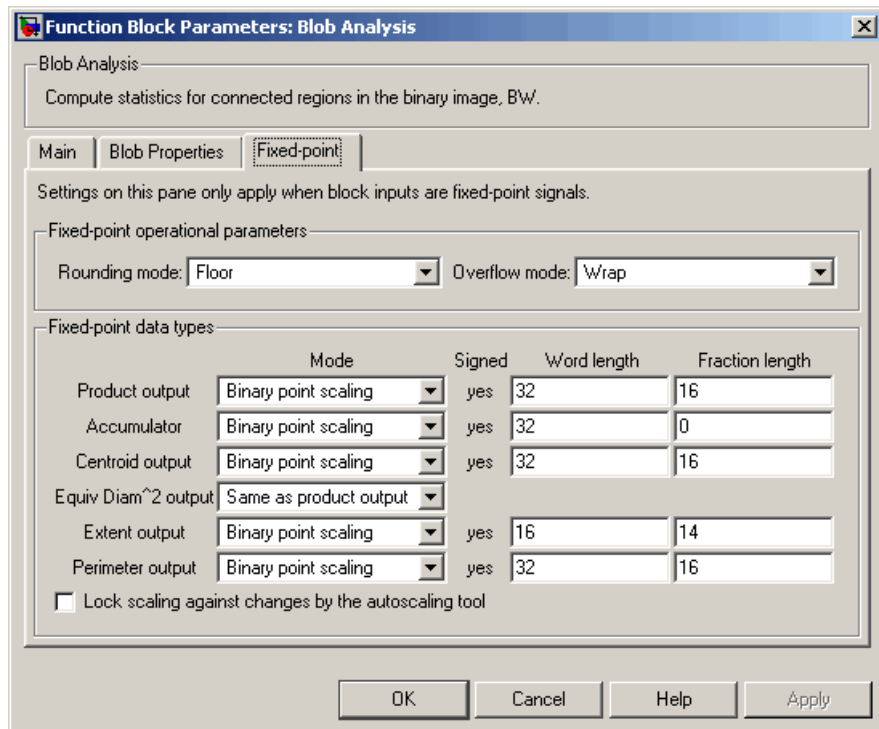
If you select this check box, the block fills the empty spaces in the statistical vectors with the value(s) you specify in the **Fill values** parameter.

Fill values

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.

The **Fixed-point** pane of the Blob Analysis dialog box appears as follows. These parameters are applicable only when the **Statistics output data type** parameter is set to Specify via Fixed-point tab.

Blob Analysis



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

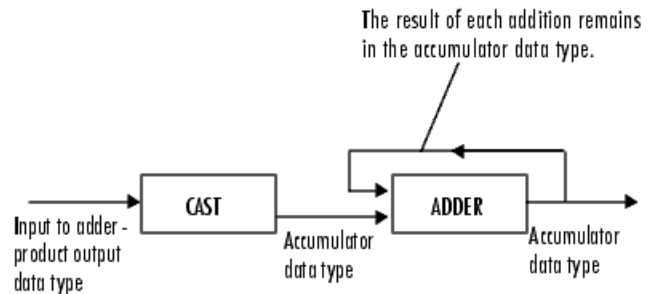


As depicted previously, the output of the multiplier is placed into the product output data type and scaling. The product output data

type is used during the computation of the equivalent diameter squared. During this computation, the blob area, which is stored in the accumulator, is multiplied by the $4/\pi$ factor. This factor has a word length that is equal to the equivalent diameter squared output data type's word length and a fraction length that is equal to its word length minus two. 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 Video and Image Processing Blockset 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 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 Video and Image Processing Blockset is 0.

Centroid output

Choose how to specify the word length and fraction length of the output at the Centroid port:

- When you select **Same as accumulator**, these characteristics match those of the accumulator.
- 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 Video and Image Processing Blockset is 0.

Equiv Diam² output

Choose how to specify the word length and fraction length of the output at the Diameter² port:

- 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 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 Video and Image Processing Blockset is 0.

Extent output

Choose how to specify the word length and fraction length of the output at the Extent port:

- When you select `Same as accumulator`, these characteristics match those of the accumulator.
- 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 Video and Image Processing Blockset is 0.

Perimeter output

Choose how to specify the word length and fraction length of the output at the Perimeter port:

- When you select `Same as accumulator`, these characteristics match those of the accumulator.
- 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 Video and Image Processing Blockset is 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

Label	Video and Image Processing Blockset
Variable Selector	Signal Processing Blockset
<code>regionprops</code>	Image Processing Toolbox

Block Matching

Purpose Estimate motion between images or video frames

Library Analysis & Enhancement

Description



Block Matching



Block Matching1



Block Matching2



Block Matching3

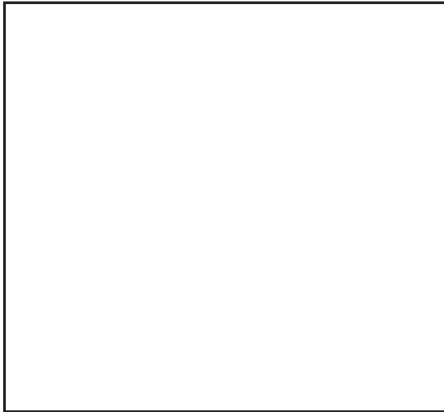
The Block Matching block estimates motion between two images or two video frames using “blocks” of pixels. The Block Matching block matches the block of pixels in frame k to a block of pixels in frame $k+1$ by moving the block of pixels over a search region.

Suppose the input to the block is frame k . The Block Matching block performs the following steps:

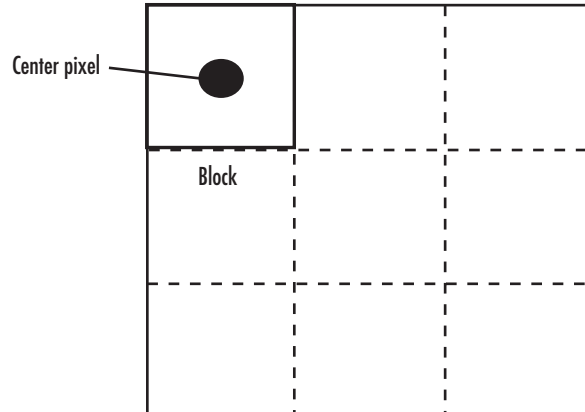
- 1 The block subdivides this frame using the values you enter for the **Block size [height width]** and **Overlap [r c]** parameters. In the following example, the **Overlap [r c]** parameter is [0 0].
- 2 For each subdivision or block in frame $k+1$, the Block Matching block establishes a search region 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

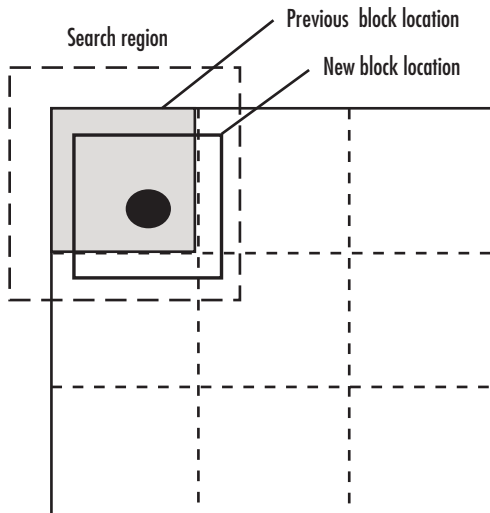
Input image = frame k



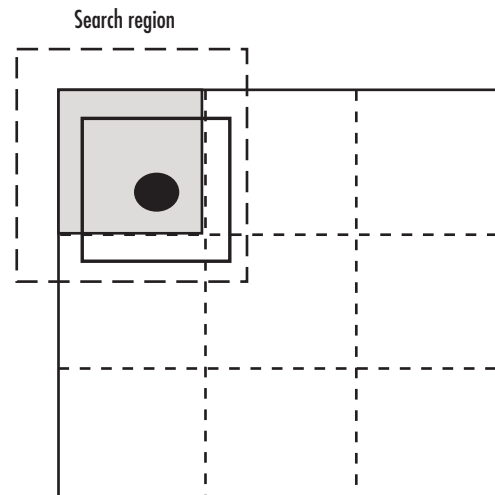
STEP 1: Subdivide the image in frame k.



STEP 2: Establish the search region in frame k+1.



STEP 3: Search for the new block location in frame k+1.



Block Matching

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

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

Block Matching

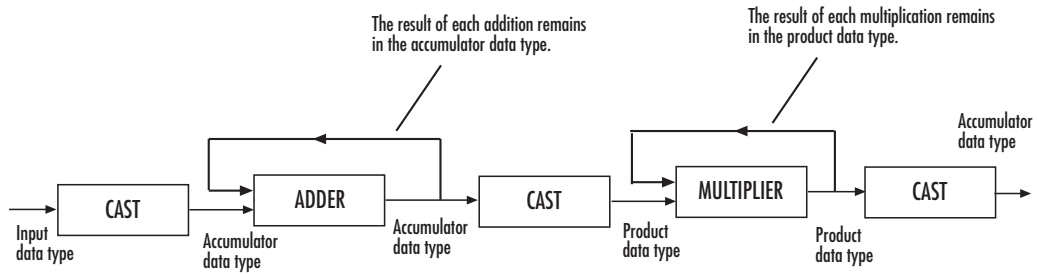
move from image to image or frame to frame. The block uses this value to determine the size of the search region.

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.

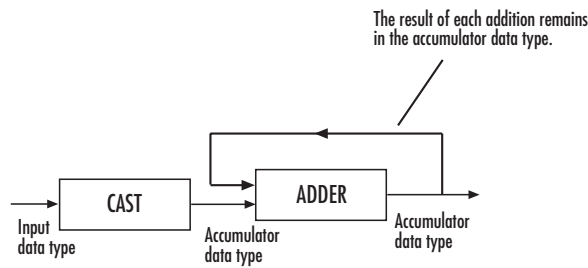
Fixed-Point Data Types

The following diagram shows the data types used in the Block Matching block for fixed-point signals.

MSE Block Matching



MAD Block Matching

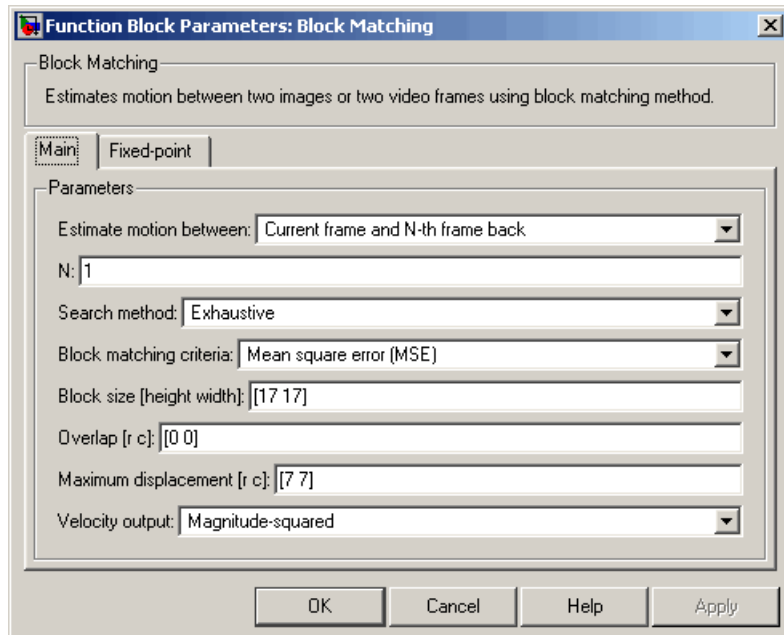


You can set the accumulator and output data types in the block mask as discussed in the next section.

Block Matching

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.

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]

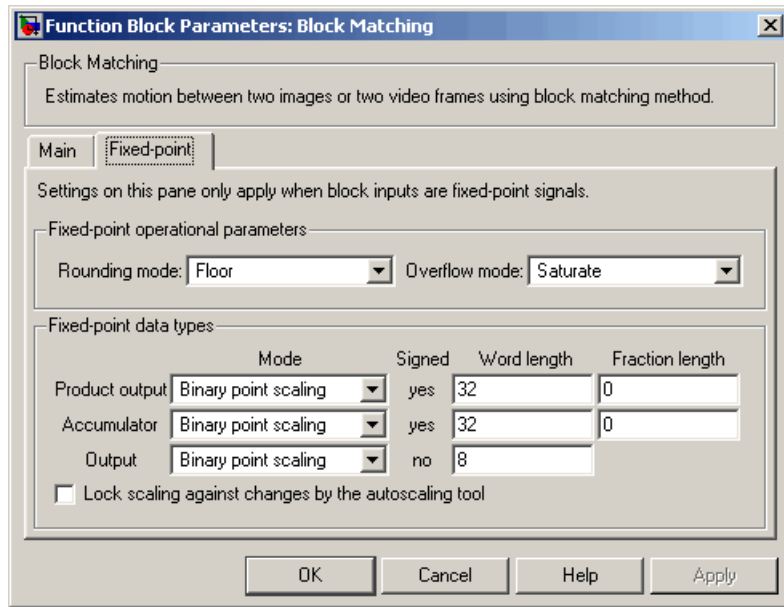
Specify the maximum number of pixels any center pixel in a block of pixels might move from image to image or frame to frame. The block uses this value to determine the size of the search region.

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 **Fixed-point** pane of the Block Matching dialog box appears as shown in the following figure.

Block Matching



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

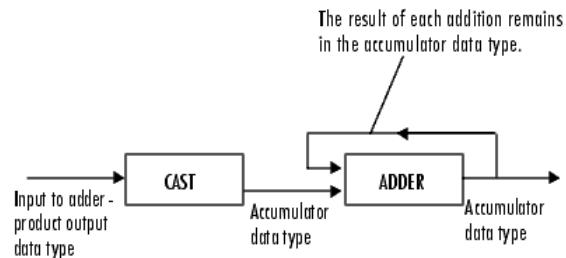
Product output



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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 0.

Output

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

Block Matching

- 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 Video and Image Processing Blockset is 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

Optical Flow

Video and Image Processing Blockset

Purpose

Repeat user-specified operation on submatrices of input matrix

Library

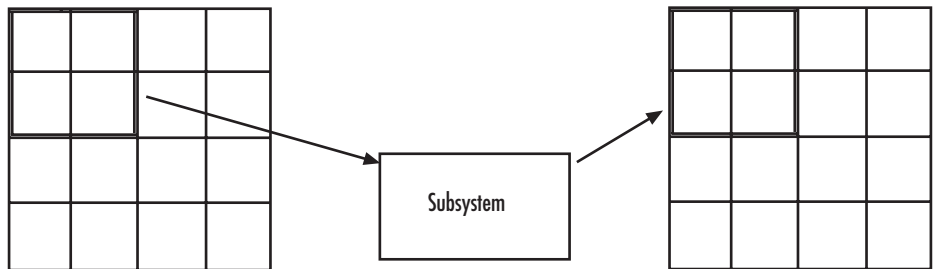
Utilities

Description



Block Processing

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 Video and Image Processing Blockset. 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.

Block Processing

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.

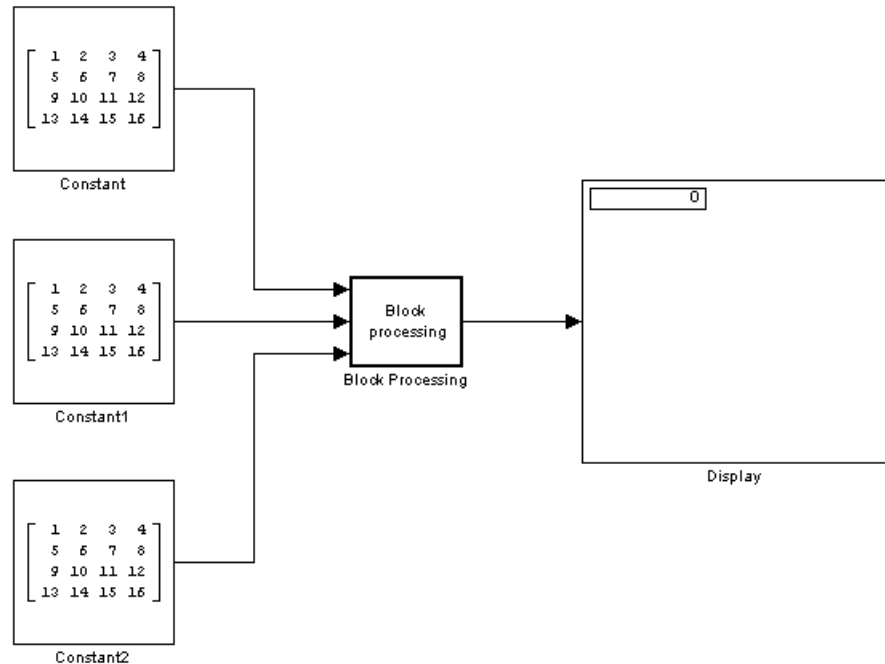
Click the **Open Subsystem** button to open the block's subsystem. Click-and-drag blocks into this subsystem to define the processing operation(s) the block performs on the submatrices. The input to this subsystem are the submatrices whose size is determined by the **Block size** parameter.

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.

Example

Example 1 -- Multiple Inputs

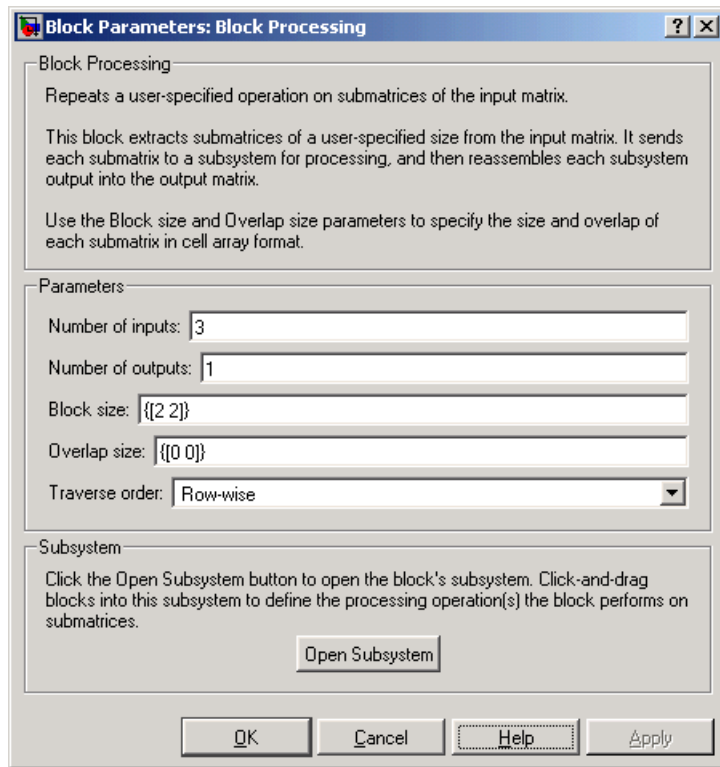
In this example, you multiply each element of three input matrices by two and add the results using the Block Processing block. Suppose you have the following model:



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** = $\{[2 \ 2]\}$

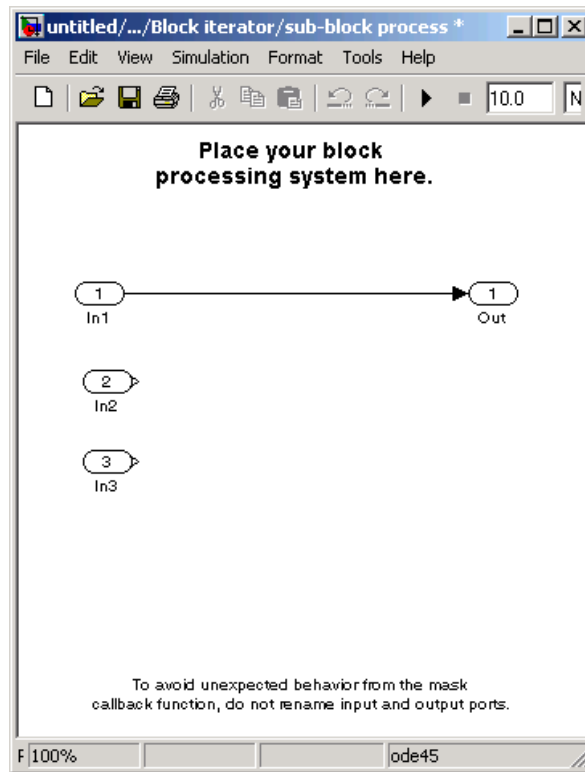
Block Processing



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.



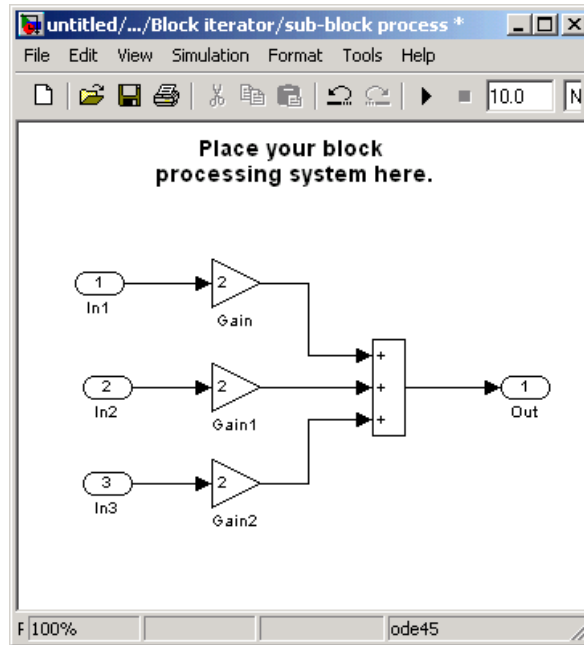
- 3 Click and drag the following blocks into the subsystem:

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 +++.

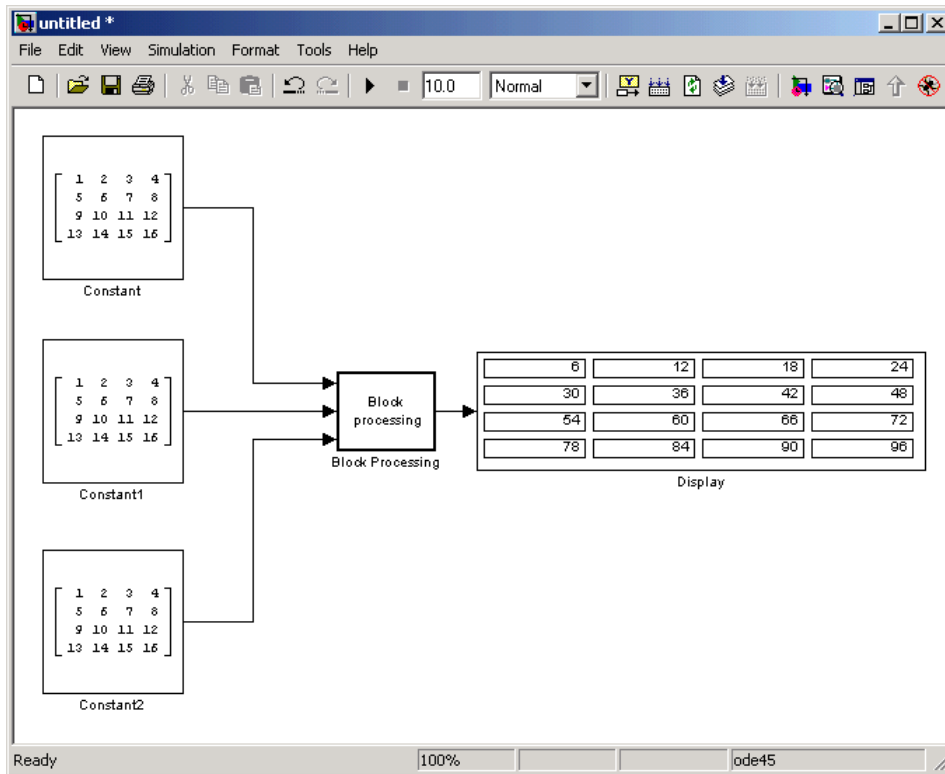
Block Processing

6 Connect the blocks as shown in the following figure.



7 Close the subsystem and Click **OK**.

8 Run the model.

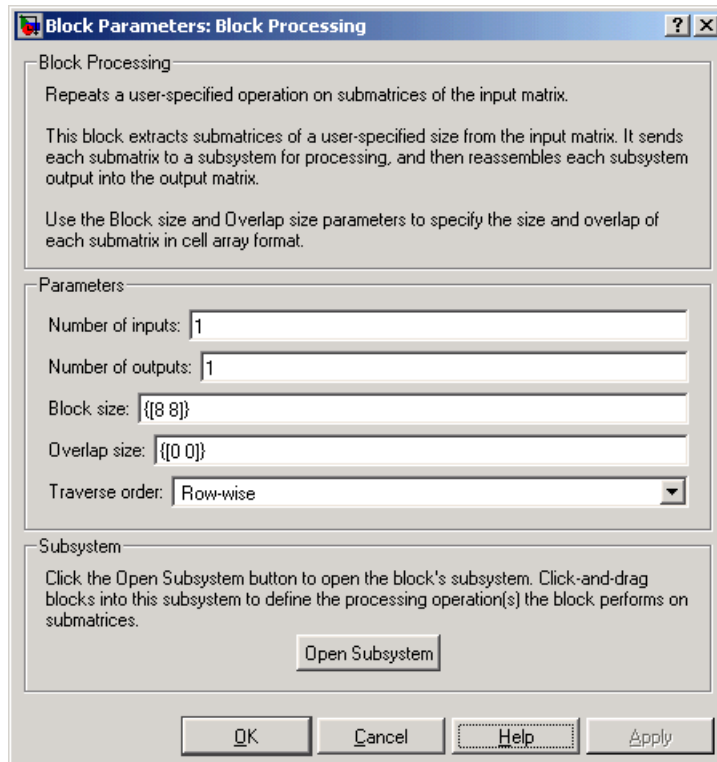


The Block Processing block operates on the submatrices and assembles the results into an output matrix that is displayed using the Display block.

Block Processing

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.

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

Simulink

blkproc

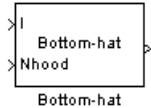
Image Processing Toolbox

Bottom-hat

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 result of performing a morphological closing 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 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.

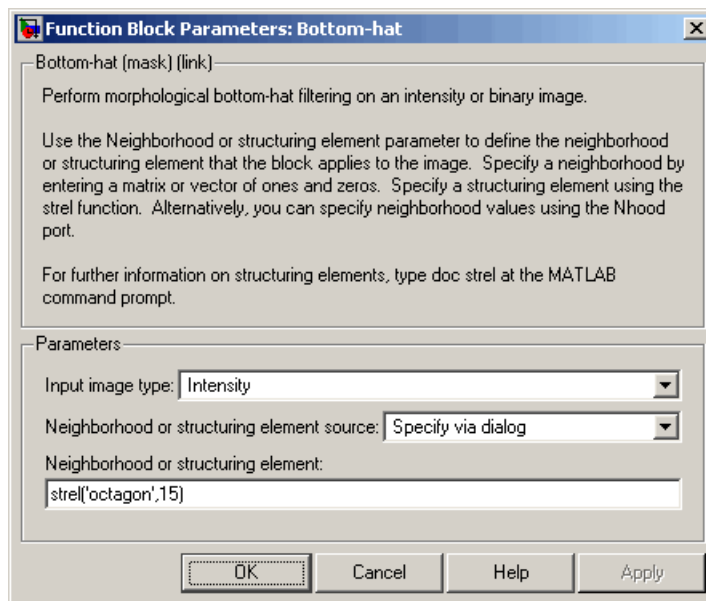
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select Specify via dialog, the **Neighborhood or structuring**

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 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 Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or structuring element source** parameter, you select Specify via dialog.

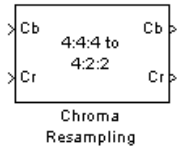
See Also

Closing	Video and Image Processing Blockset
Dilation	Video and Image Processing Blockset
Erosion	Video and Image Processing Blockset
Label	Video and Image Processing Blockset
Opening	Video and Image Processing Blockset
Top-hat	Video and Image Processing Blockset
imbothat	Image Processing Toolbox
strel	Image Processing Toolbox

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.



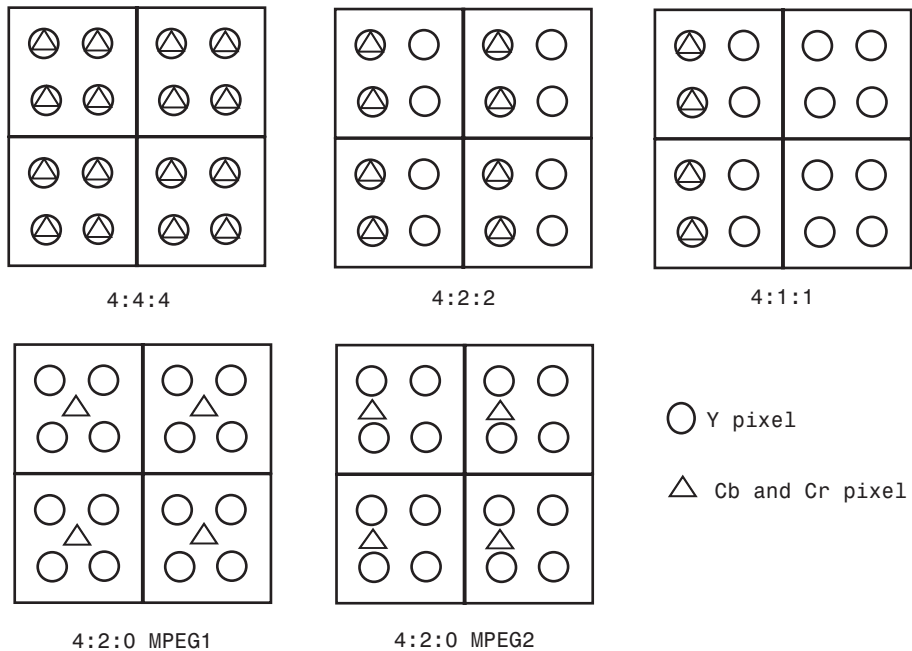
Port	Input/Output	Supported Data Types	Complex Values Supported
Cb	Matrix that represents one chrominance component of an image	<ul style="list-style-type: none"> • Double-precision floating point • Single-precision floating point • 8-bit unsigned integer 	No
Cr	Matrix that represents one chrominance 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.

Chroma Resampling



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

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

MATLAB and Video and Image Processing Blockset 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:

Chroma Resampling

- 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:

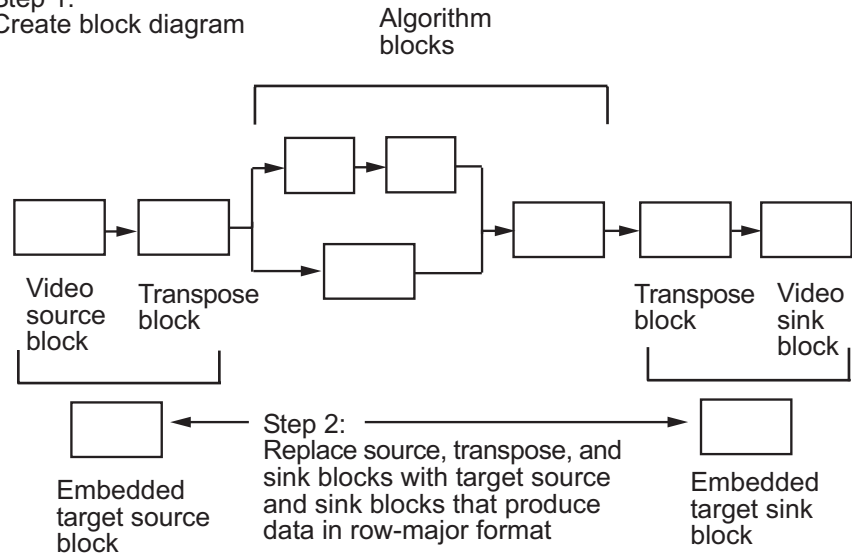
- When you select this check box, the signal dimensions of the Chroma Resampling block's input are swapped.
- All Video and Image Processing Blockset blocks can be used to process data that is in the row-major format, but you need to know the image dimensions when you develop your algorithms.

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.

Step 1:
Create block diagram

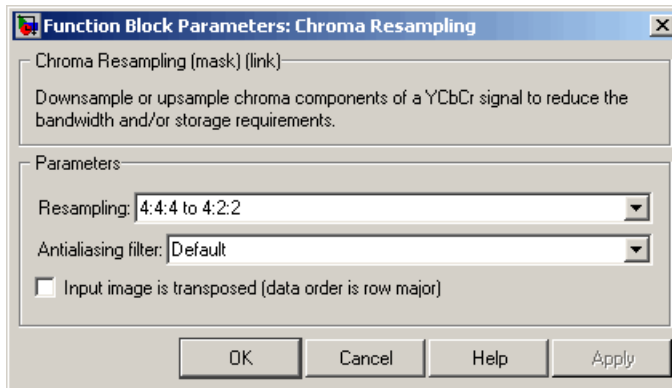


See the DM642 EVM Video ADC and DM642 EVM Video DAC reference pages in the *Target for TI C6000 User's Guide* for more information about data order in embedded targets.

Chroma Resampling

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

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 4:4:4 , 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.

Chroma Resampling

See Also

Autothreshold

Video and Image Processing Blockset

Color Space
Conversion

Video and Image Processing Blockset

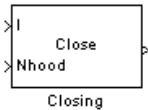
Image Complement

Video and Image Processing Blockset

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.

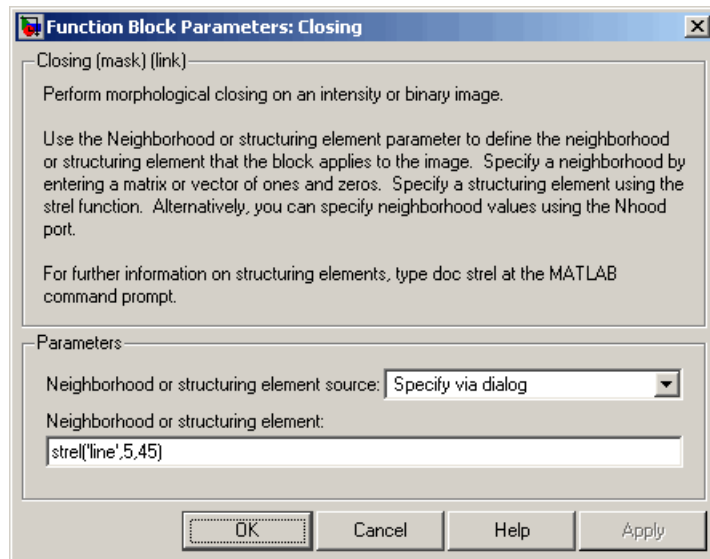
Closing

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 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 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
Dilation	Video and Image Processing Blockset
Erosion	Video and Image Processing Blockset
Label	Video and Image Processing Blockset
Opening	Video and Image Processing Blockset
Top-hat	Video and Image Processing Blockset
<code>imclose</code>	Image Processing Toolbox
<code>strel</code>	Image Processing Toolbox

Color Space Conversion

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.

Color Space Conversion

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.

Color Space Conversion

$$H = \begin{cases} \left(\frac{G' - B'}{MAX - MIN} \right) / 6, & \text{if } R' = MAX \\ \left(2 + \frac{B' - R'}{MAX - MIN} \right) / 6, & \text{if } G' = MAX \\ \left(4 + \frac{R' - G'}{MAX - MIN} \right) / 6, & \text{if } B' = MAX \end{cases}$$
$$S = \frac{MAX - MIN}{MAX}$$
$$V = MAX$$

The HSV to R'G'B' conversion is defined by the following equations:

$$H_i = \lfloor 6H \rfloor$$
$$f = 6H - H_i$$
$$p = 1 - S$$
$$q = 1 - fS$$
$$t = 1 - (1 - f)S$$

if $H_i = 0$, $R_{tmp} = 1$, $G_{tmp} = t$, $B_{tmp} = p$
if $H_i = 1$, $R_{tmp} = q$, $G_{tmp} = 1$, $B_{tmp} = p$
if $H_i = 2$, $R_{tmp} = p$, $G_{tmp} = 1$, $B_{tmp} = t$
if $H_i = 3$, $R_{tmp} = p$, $G_{tmp} = q$, $B_{tmp} = 1$
if $H_i = 4$, $R_{tmp} = t$, $G_{tmp} = p$, $B_{tmp} = 1$
if $H_i = 5$, $R_{tmp} = 1$, $G_{tmp} = p$, $B_{tmp} = q$

$$u = V / \max(R_{tmp}, G_{tmp}, B_{tmp})$$
$$R' = uR_{tmp}$$
$$G' = uG_{tmp}$$
$$B' = uB_{tmp}$$

For more information about the HSV color space, see “HSV Color Space” in the Image Processing Toolbox documentation.

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:

$$\text{If } R'_{sRGB}, G'_{sRGB}, B'_{sRGB} \leq 0.03928$$

$$R_{sRGB} = R'_{sRGB} / 12.92$$

$$G_{sRGB} = G'_{sRGB} / 12.92$$

$$B_{sRGB} = B'_{sRGB} / 12.92$$

$$\text{otherwise, if } R'_{sRGB}, G'_{sRGB}, B'_{sRGB} > 0.03928$$

$$R_{sRGB} = \left[\frac{(R'_{sRGB} + 0.055)}{1.055} \right]^{2.4}$$

$$G_{sRGB} = \left[\frac{(G'_{sRGB} + 0.055)}{1.055} \right]^{2.4}$$

$$B_{sRGB} = \left[\frac{(B'_{sRGB} + 0.055)}{1.055} \right]^{2.4}$$

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:

$$\begin{bmatrix} R_{sRGB} \\ G_{sRGB} \\ B_{sRGB} \end{bmatrix} = \begin{bmatrix} 0.41239079926596 & 0.35758433938388 & 0.18048078840183 \\ 0.21263900587151 & 0.71516867876776 & 0.07219231536073 \\ 0.01933081871559 & 0.11919477979463 & 0.95053215224966 \end{bmatrix}^{-1} \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Color Space Conversion

Then the block applies gamma correction to obtain the sR'G'B' values. This process is described by the following equations:

$$\begin{aligned} &\text{If } R_{sRGB}, G_{sRGB}, B_{sRGB} \leq 0.00304 \\ &R'_{sRGB} = 12.92R_{sRGB} \\ &G'_{sRGB} = 12.92G_{sRGB} \\ &B'_{sRGB} = 12.92B_{sRGB} \\ &\text{otherwise, if } R_{sRGB}, G_{sRGB}, B_{sRGB} > 0.00304 \\ &R'_{sRGB} = 1.055R_{sRGB}^{(1.0/2.4)} - 0.055 \\ &G'_{sRGB} = 1.055G_{sRGB}^{(1.0/2.4)} - 0.055 \\ &B'_{sRGB} = 1.055B_{sRGB}^{(1.0/2.4)} - 0.055 \end{aligned}$$

Note Video and Image Processing Blockset 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 2-231. 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:

$$L^* = 116(Y/Y_n)^{1/3} - 16, \text{ for } Y/Y_n > 0.008856$$

$$L^* = 903.3Y/Y_n, \quad \text{otherwise}$$

$$a^* = 500(f(X/X_n) - f(Y/Y_n))$$

$$b^* = 200(f(Y/Y_n) - f(Z/Z_n)),$$

$$\text{where } f(t) = t^{1/3}, \text{ for } t > 0.008856$$

$$f(t) = 7.787t + 16/166, \quad \text{otherwise}$$

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:

$$\text{For } Y/Y_n > 0.008856$$

$$X = X_n(P + a^*/500)^3$$

$$Y = Y_n P^3$$

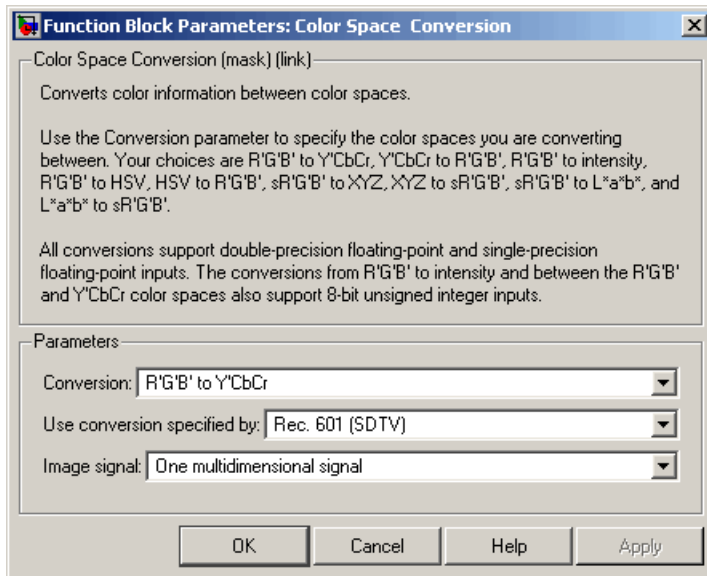
$$Z = Z_n(P - b^*/200)^3,$$

$$\text{where } P = (L^* + 16)/116$$

Color Space Conversion

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

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
rgb2hsv

Video and Image Processing Blockset
MATLAB

Color Space Conversion

hsv2rgb

MATLAB

rgb2ycbcr

Image Processing Toolbox

ycbcr2rgb

Image Processing Toolbox

rgb2gray

Image Processing Toolbox

makecform

Image Processing Toolbox

applycform

Image Processing Toolbox

Purpose

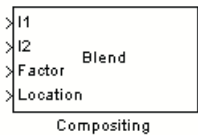
Combine pixel values of two images, overlay one image over another, or highlight selected pixels

Library

Text & Graphics

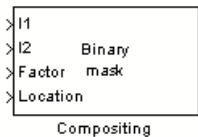
Description

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:

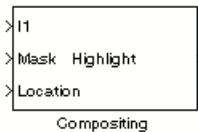


$$O(i, j) = (1 - X) * I1(i, j) + X * I2(i, j)$$

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.

Compositing

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

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 <code>Highlight selected pixel</code> , 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.

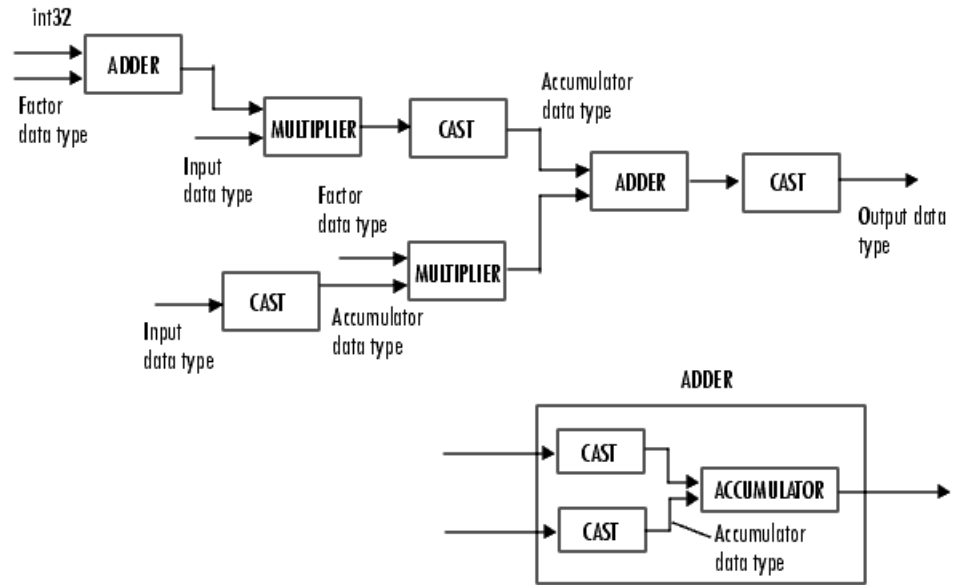
- If you choose `Specify via dialog`, the **Location [row column]** parameter appears on the dialog box. Enter a two-element vector that specifies the row and column position of the upper-left corner of the image input at port Image 2 relative to the upper-left corner of the image input at port Image 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`.

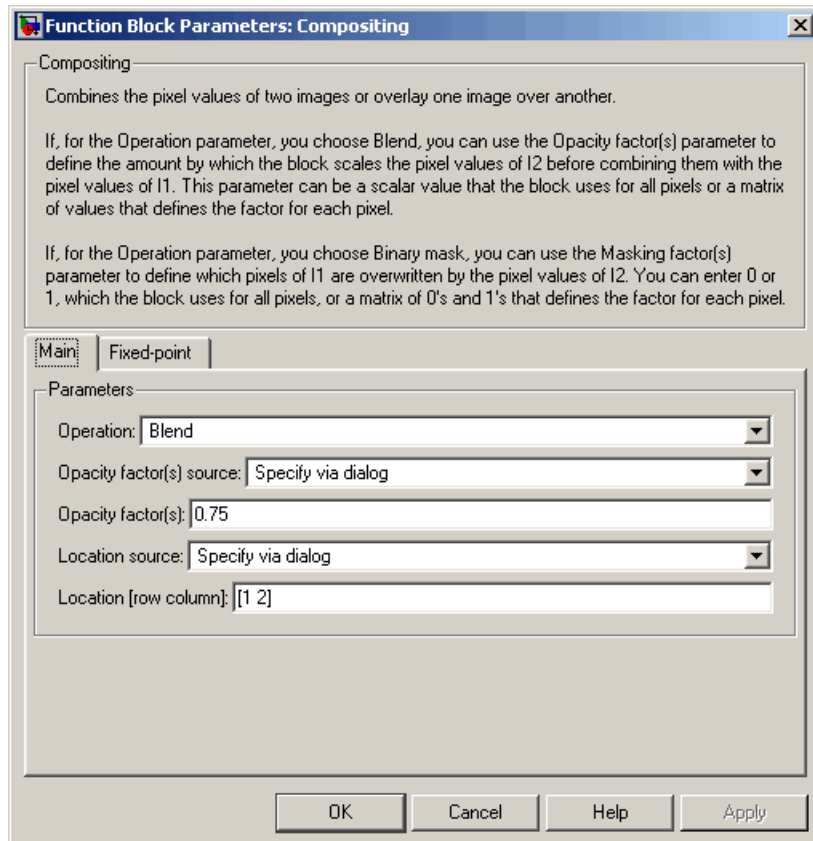
Compositing



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 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.

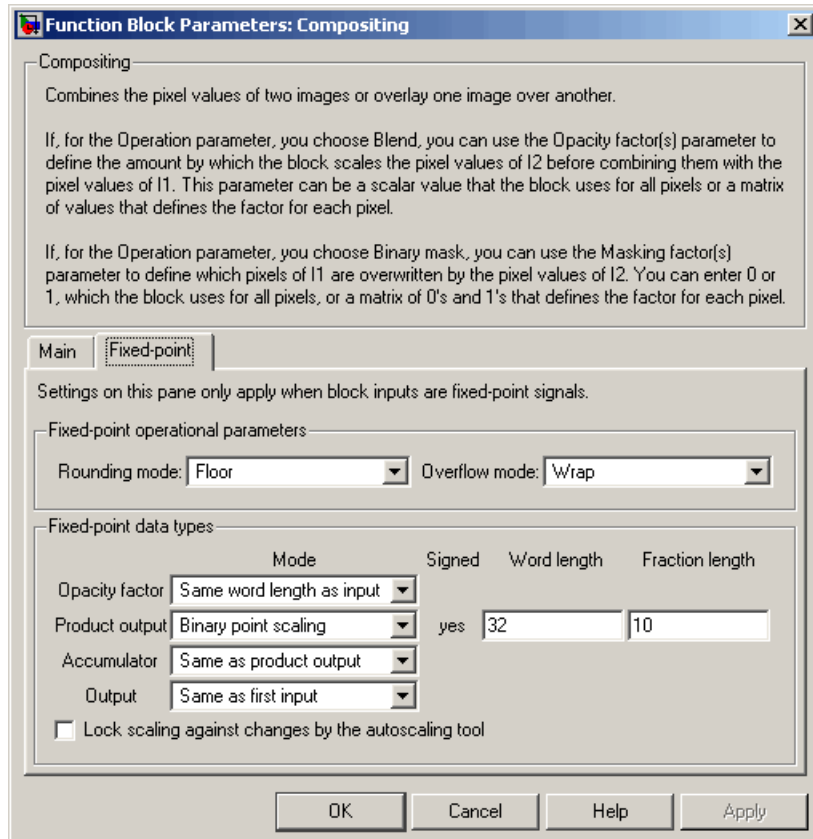
Location source

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 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 **Fixed-point** pane of the Compositing dialog box appears as follows. These parameters are applicable only when the **Operation** parameter is set to Blend.



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 Video and Image Processing Blockset is 0.

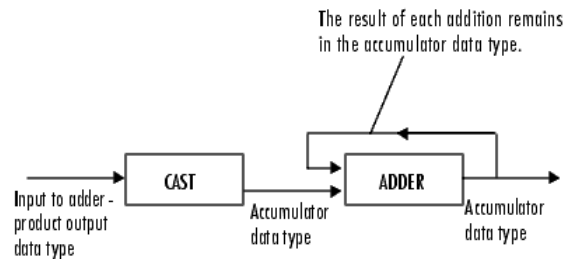
Product output



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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 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

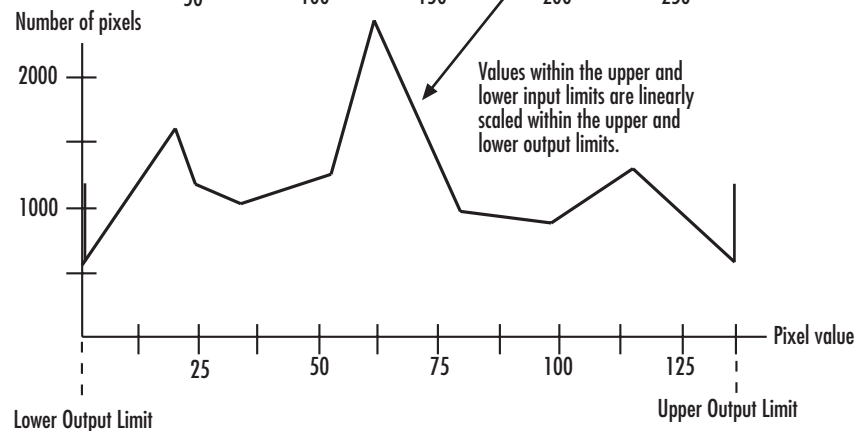
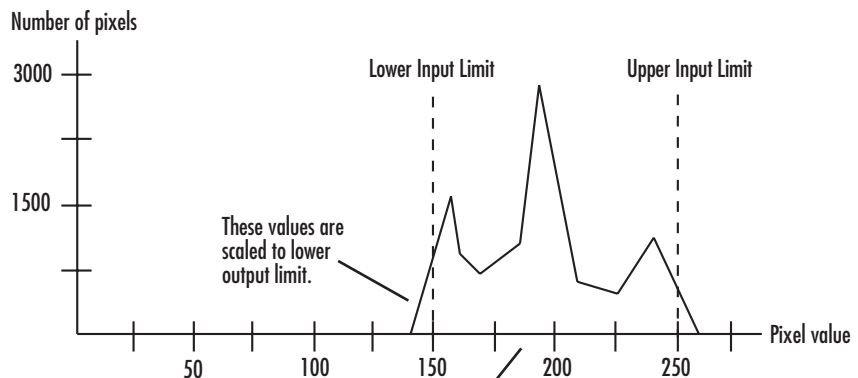
[Insert Text](#)

[Video and Image Processing Blockset](#)

Purpose Adjust image contrast by linearly scaling pixel values

Library Analysis & Enhancement

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

Use the **Adjust pixel values from** parameter to specify the upper and lower input limits. If you select Full input data range [min max], the block 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 appears on the block. 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 outliers are eliminated), 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 2-D 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.

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 output limit. 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.

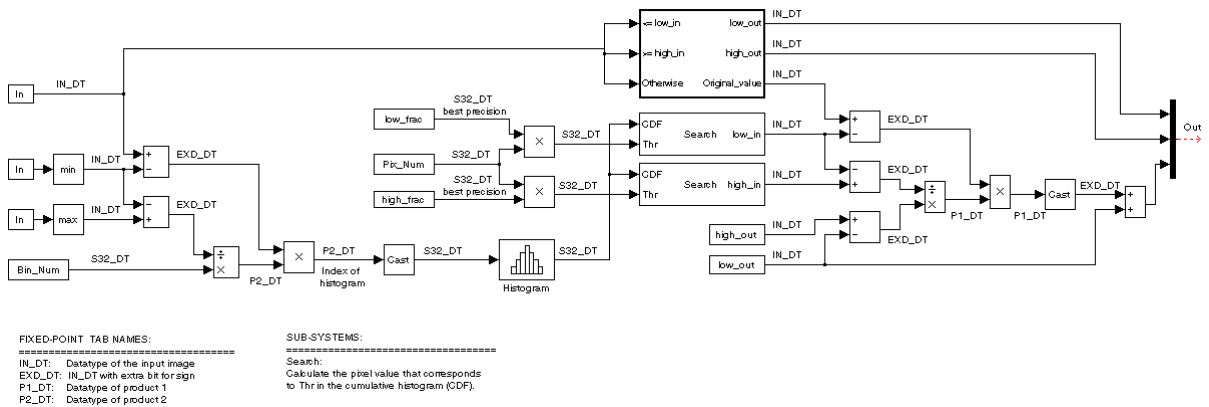
Examples

See “Adjusting the Contrast in Intensity Images” in the *Video and Image Processing Blockset User’s Guide*.

Fixed-Point Data Types

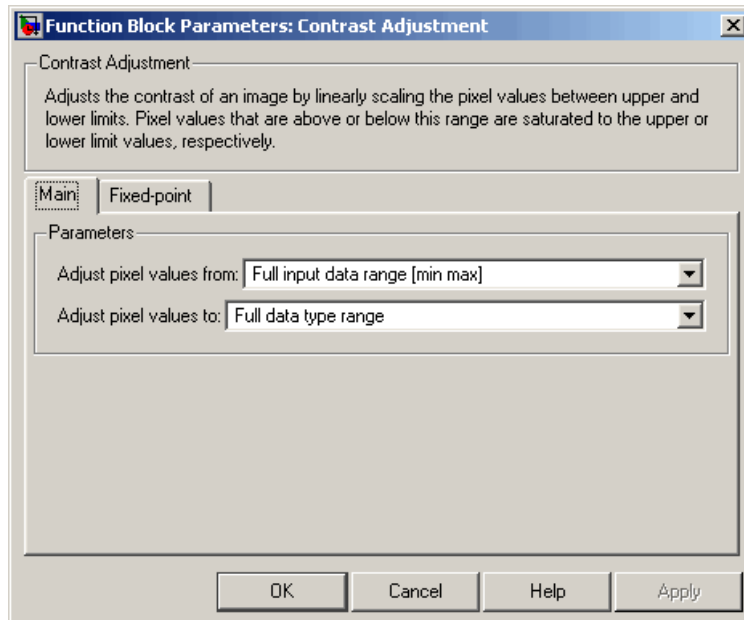
The following diagram shows the data types used in the Contrast Adjustment block for fixed-point signals:

Contrast Adjustment



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.

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 histograms 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

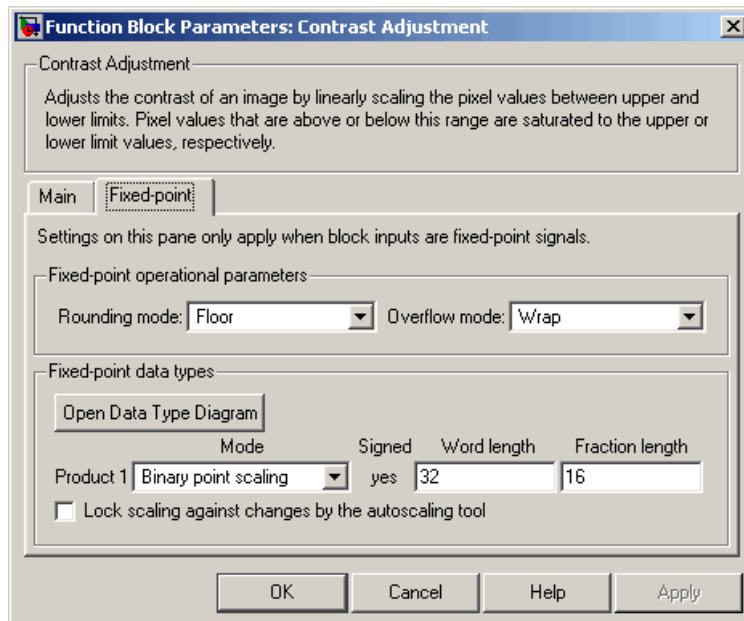
Specify the upper and lower output limits. If you select Full data type range, the block uses the minimum value of the input data type as the lower output limit and the maximum value of the input data type as the upper output limit. If you select User-defined range, the **Range [low high]** parameter appears on the block.

Contrast Adjustment

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 visible if, for the **Adjust pixel values to** parameter, you select User-defined range

The **Fixed-point** pane of the Contrast Adjustment dialog box appears as shown in the following figure.



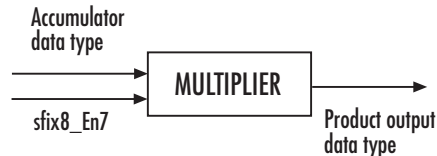
Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product 1



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 Video and Image Processing Blockset is 0.

Product 2



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 Video and Image Processing Blockset is 0.

Contrast Adjustment

This parameter is visible if, for the **Adjust pixel values from** parameter, you select Range determined by saturating outlier pixels.

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 Histogram

Video and Image Processing Blockset

Histogram

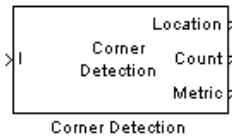
Video and Image Processing Blockset

Equalization

Purpose Calculate corner metric matrix and find corners in images

Library Analysis & Enhancement

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.



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-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer 	No
Location	2-by-N matrix that represents the locations of the corners where N is the maximum number of corners	32-bit unsigned integer	No
Count	Scalar value that represents the number of detected corners	32-bit unsigned integer	No
Metric	Matrix of corner metric values that is the same size as the input image	Same as I port	No

Corner Detection

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 2-258.

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.

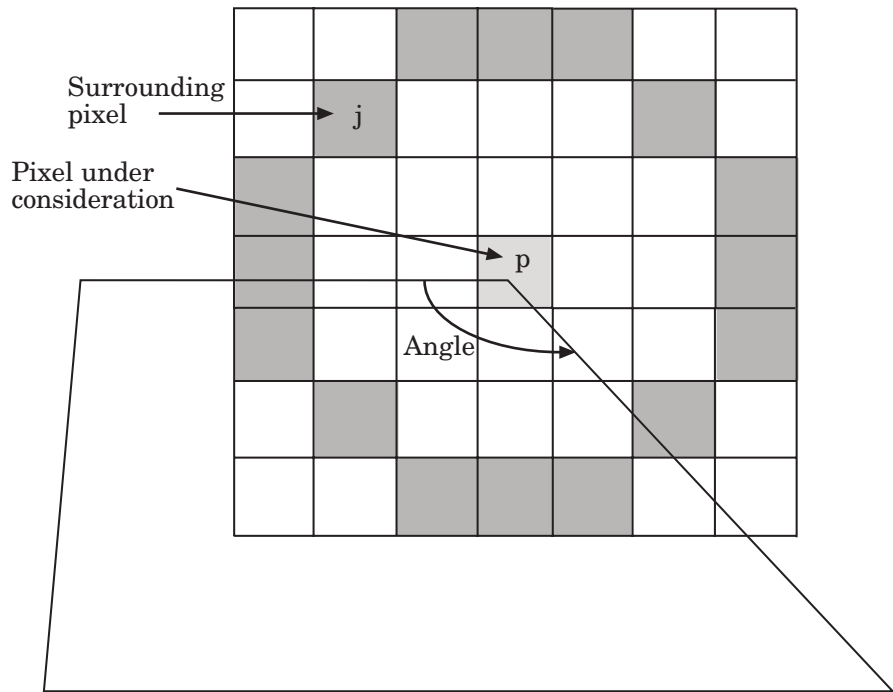
Use the **Coefficients for separable smoothing filter** parameter to define a vector of filter coefficients. The block multiplies this vector of coefficients by its transpose to create a matrix of filter coefficients, w .

Local Intensity Comparison

This method determines that a pixel is a possible corner if it has an N , contiguous valid bright or 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.

Corner Detection



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.

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 **Neighborhoo 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.

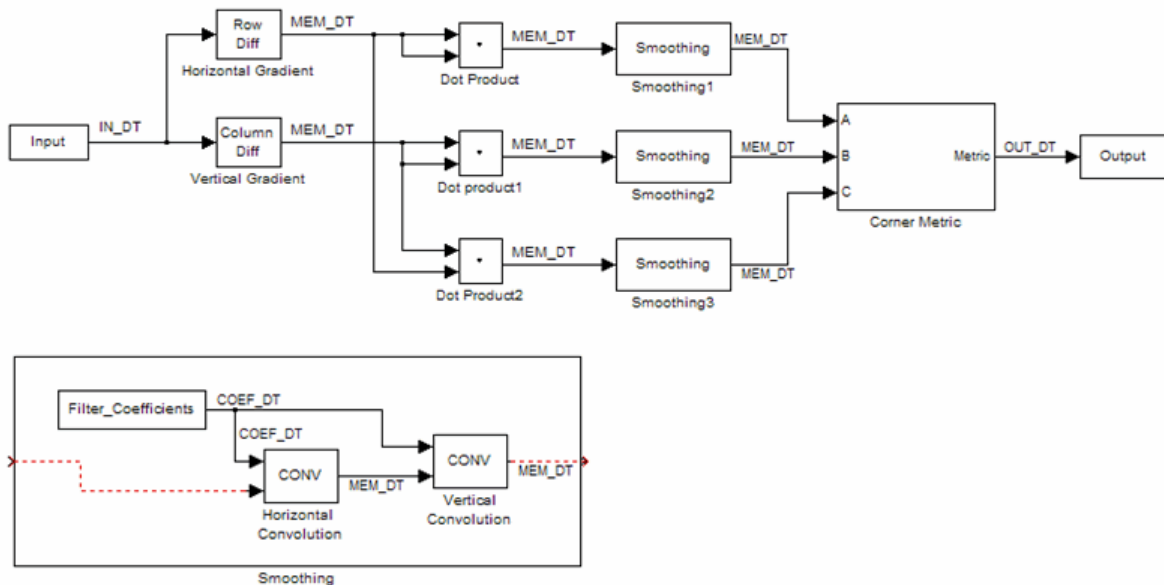
Corner Detection

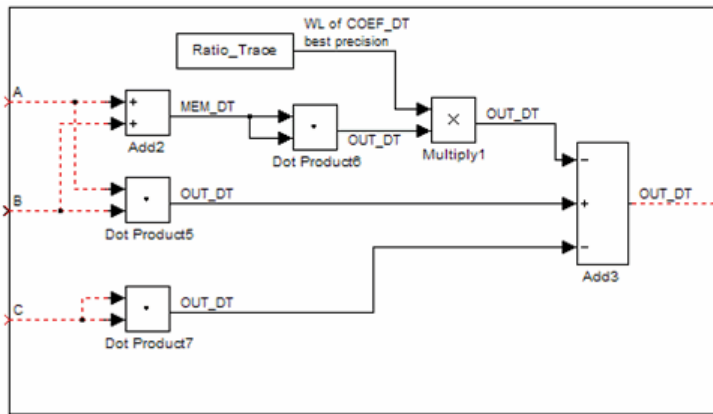
- 2 Verify that the metric value is greater than or equal to the value you specified for the **Minimum metric value that indicates a corner** parameter.
- 3 Suppress the region around the corner value by the size defined in the **Neighborhood size (suppress region around detected corners)** parameter.

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.

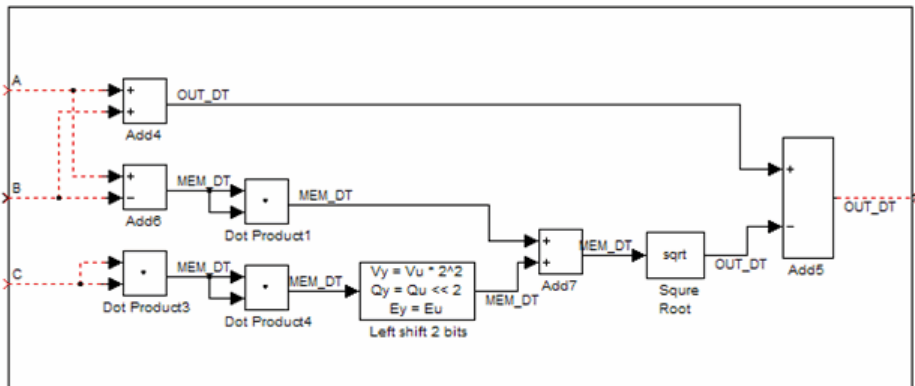
Fixed-Point Data Types

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 Metric by Harris Algorithm



Corner Metric by Minimum Eigenvalue Algorithm

The following table summarizes the variables used in the previous diagrams.

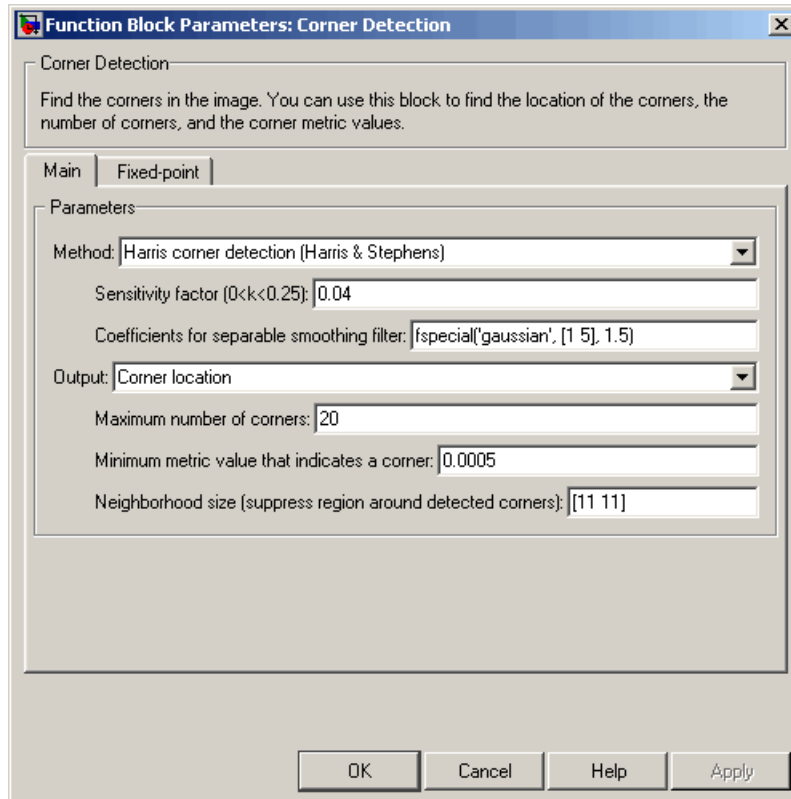
Variable Name	Definition
IN_DT	Input data type
MEM_DT	Memory data type

Corner Detection

Variable Name	Definition
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.



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).

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).

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).

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.

Corner Detection

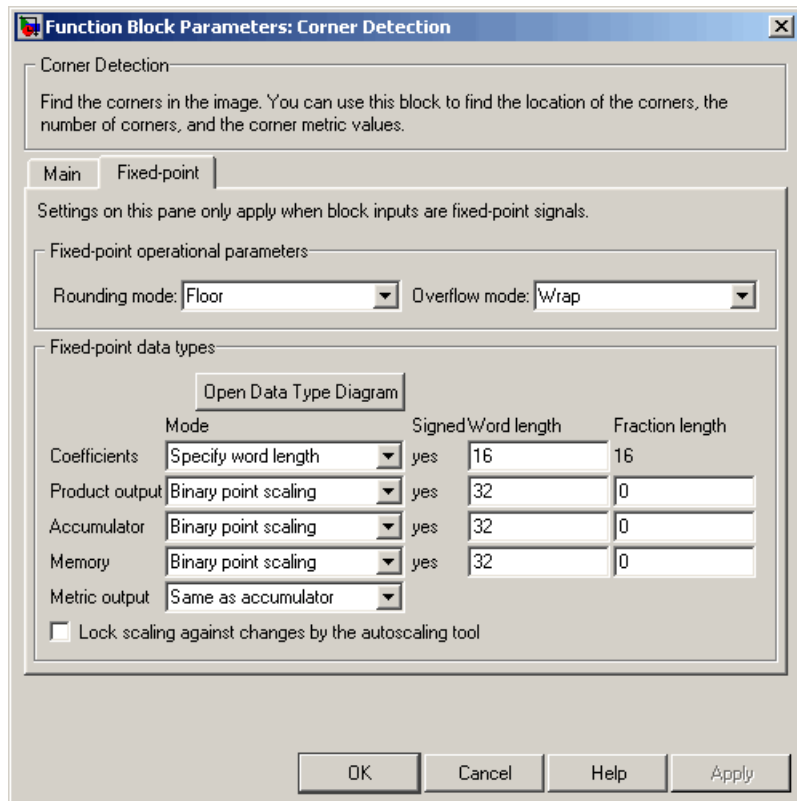
Minimum metric value that indicates a corner

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.

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 **Fixed-point** pane of the Corner 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.

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

Corner Detection

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 Video and Image Processing Blockset 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.

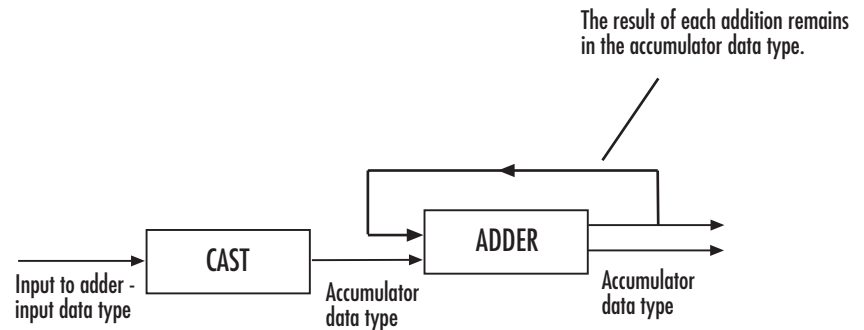


Use this parameter to specify how to designate the product output word and fraction lengths.

- When you select `Same as input`, these characteristics match those of the input to the block.
- 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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.

Corner Detection

- 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 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] C. Harris and M. Stephens. “A Combined Corner and Edge Detector.” *Proceedings of the 4th Alvey Vision Conference*. August 1988, pp. 147–151.
- [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.

See Also

[Find Local Maxima](#)

[Video and Image Processing Blockset](#)

Deinterlacing

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.

Line Repetition

Original Interlaced Video

Top Field			Bottom Field			
Row 1	A	B	C	Row 1		
Row 2				Row 2	D	E
Row 3	G	H	I	Row 3		
Row 4				Row 4	J	K
Row 5	M	N	O	Row 5		
Row 6				Row 6	P	Q

Block Input			Block Output - Deinterlaced Video			
Row 1	A	B	C	Row 1	A	B
Row 2	D	E	F	Row 2	A	B
Row 3	G	H	I	Row 3	G	H
Row 4	J	K	L	Row 4	G	H
Row 5	M	N	O	Row 5	M	N
Row 6	P	Q	R	Row 6	M	N

The following figure illustrates the block's behavior if you select Linear interpolation.

Deinterlacing

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.

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

MATLAB and Video and Image Processing Blockset use column-major data organization. However, the Deinterlacing block gives you the option to process data that is stored in row-major format. When

Deinterlacing

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.

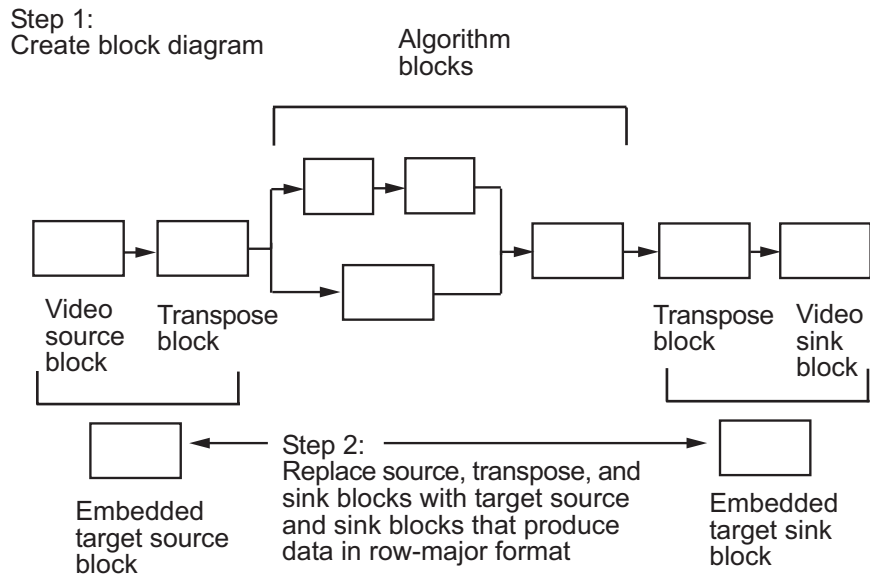
When you use the row-major functionality, you must consider the following issues:

- When you select this check box, the first two signal dimensions of the Deinterlacing block's input are swapped.
- All 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.



See the DM642 EVM Video ADC and DM642 EVM Video DAC reference pages in the *Target for TI C6000 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.

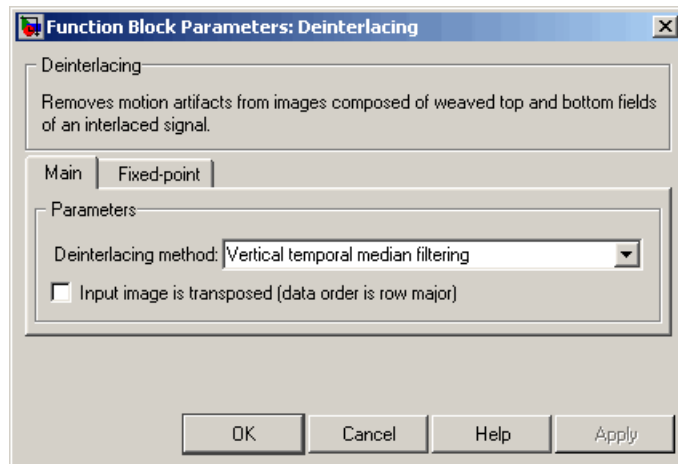
- 1 Open the example model by typing

```
doc_deinterlace
```

at the MATLAB command prompt.

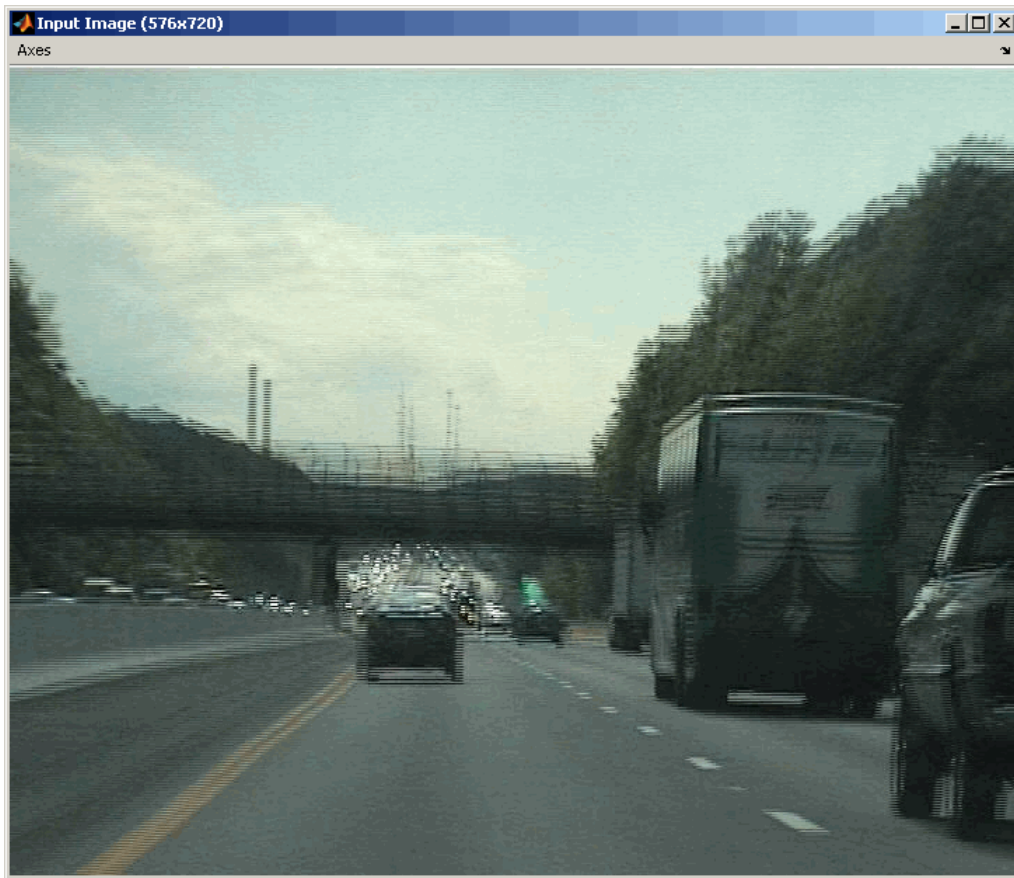
- 2 Double-click the Deinterlacing block. The model uses this block to remove the motion artifacts from the input image. The **Deinterlacing method** parameter is set to Vertical temporal median filtering.

Deinterlacing



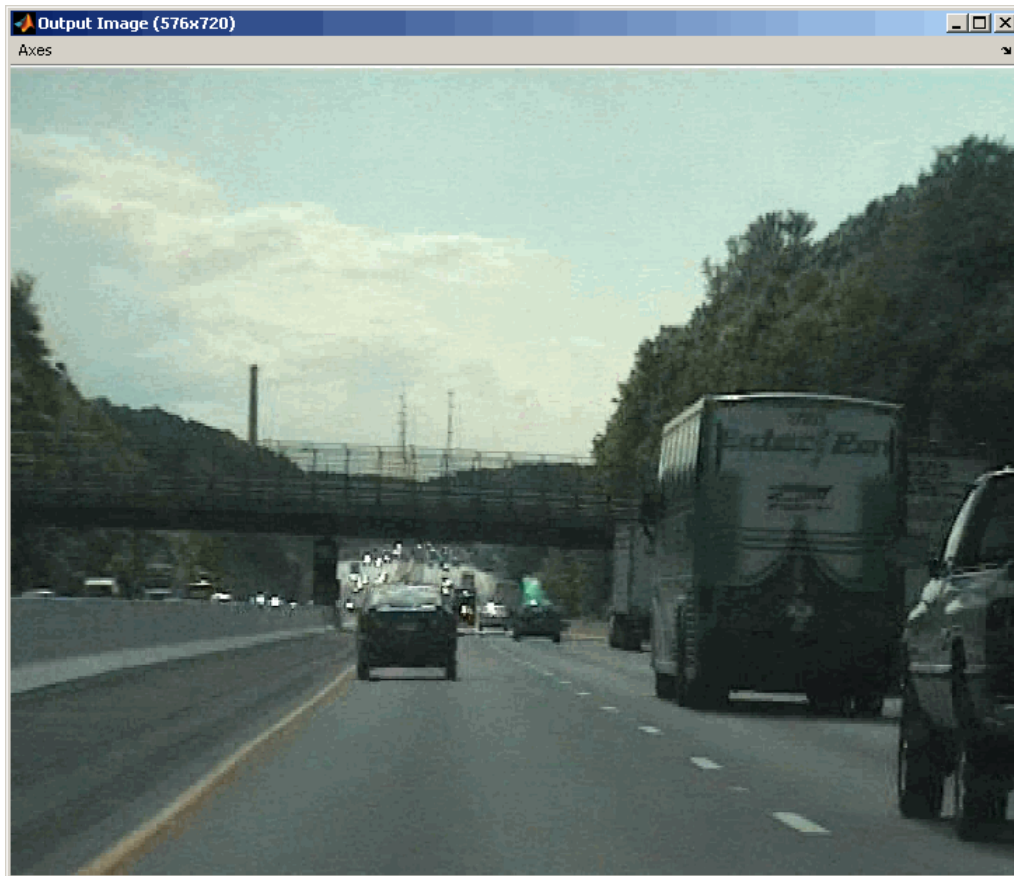
3 Run the model.

The original image that contains the motion artifacts appears in the Input Image window.



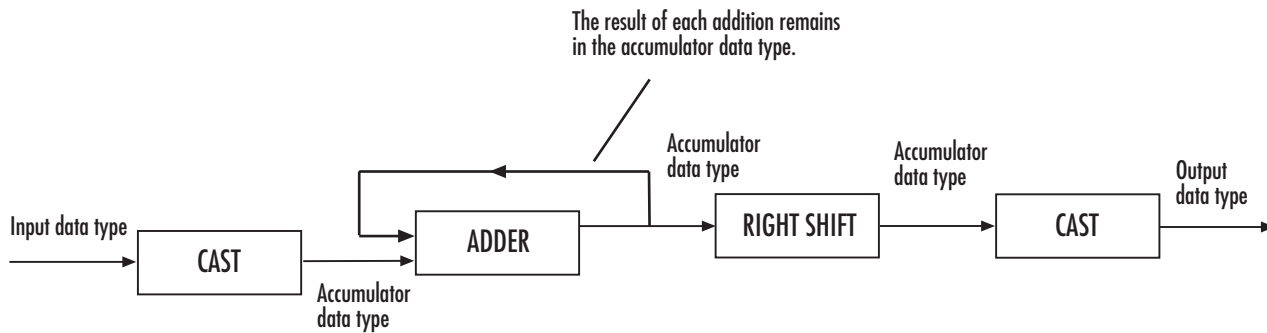
The clearer output image appears in the Output Image window.

Deinterlacing



Fixed-Point Data Types

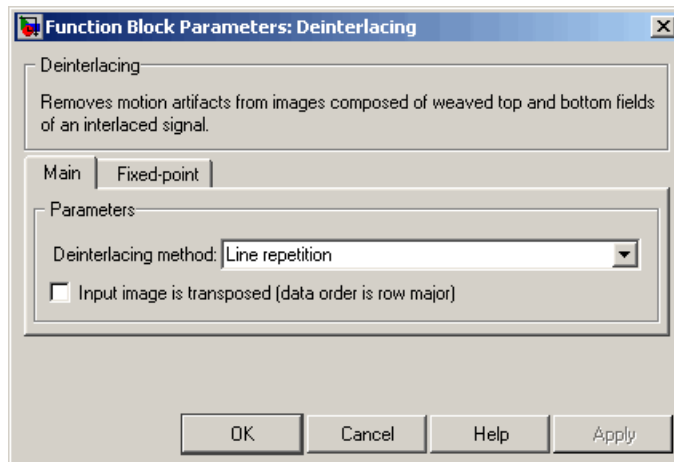
The following diagram shows the data types used in the Deinterlacing block for fixed-point signals.



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

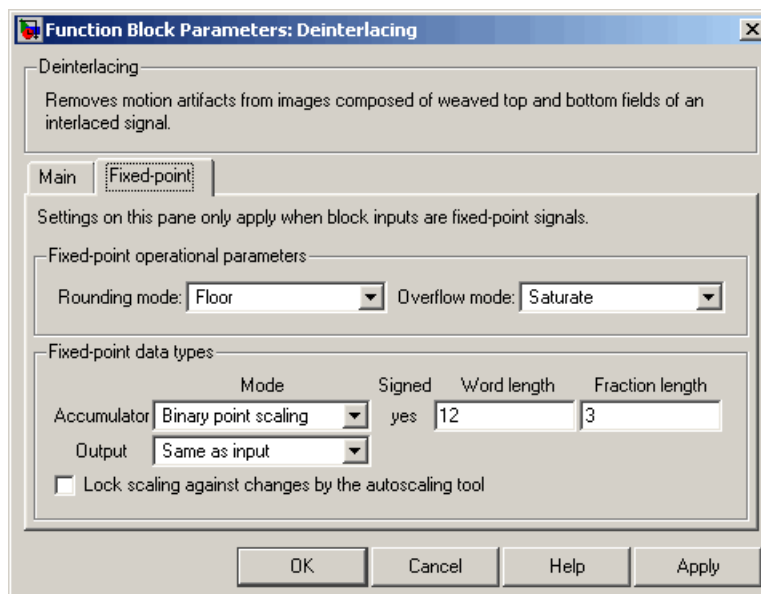
Specify how the block deinterlaces the video. Your choices are Line repetition, Linear interpolation, or Vertical temporal median filtering.

Deinterlacing

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 **Fixed-point** pane of the Deinterlacing dialog box appears as shown in the following figure.



Note The parameters on the **Fixed-point** pane are only available if, for the **Deinterlacing method**, you select **Linear** interpolation.

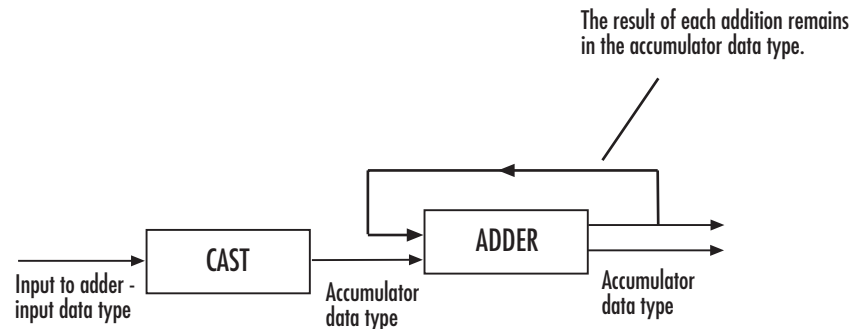
Rounding mode

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 Video and Image Processing Blockset 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.

Deinterlacing

- 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.

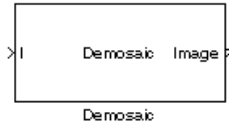
Lock scaling against changes by the autoscaling tool

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.

Purpose Demosaic Bayer's format images

Library Conversions

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].

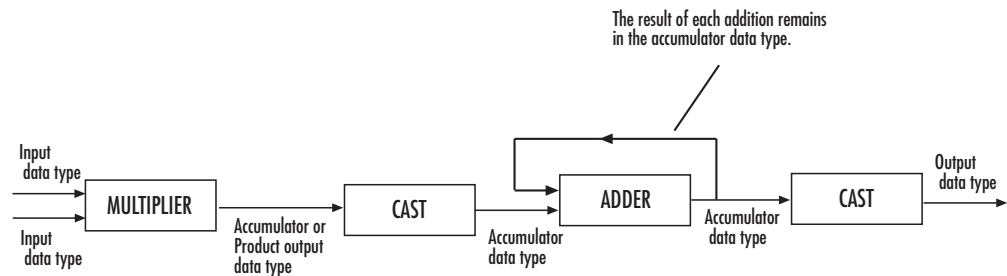
Use the **Sensor alignment** parameter to specify the alignment of the input image. Select the sequence of R, G and B pixels that correspond to the 2-by-2 block of pixels in the top-left corner of the image. You specify the sequence in left-to-right, top-to-bottom order. For example, for the image at 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.

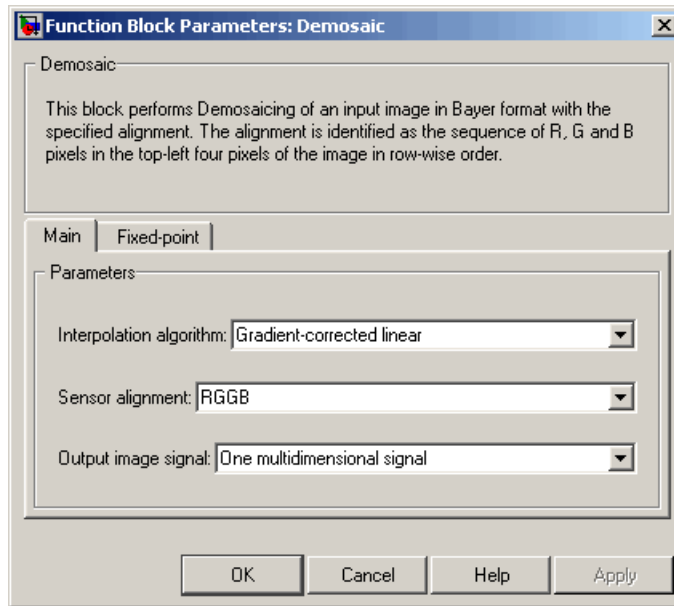


You can set the product output and accumulator data types in the block mask as discussed in the next section.

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

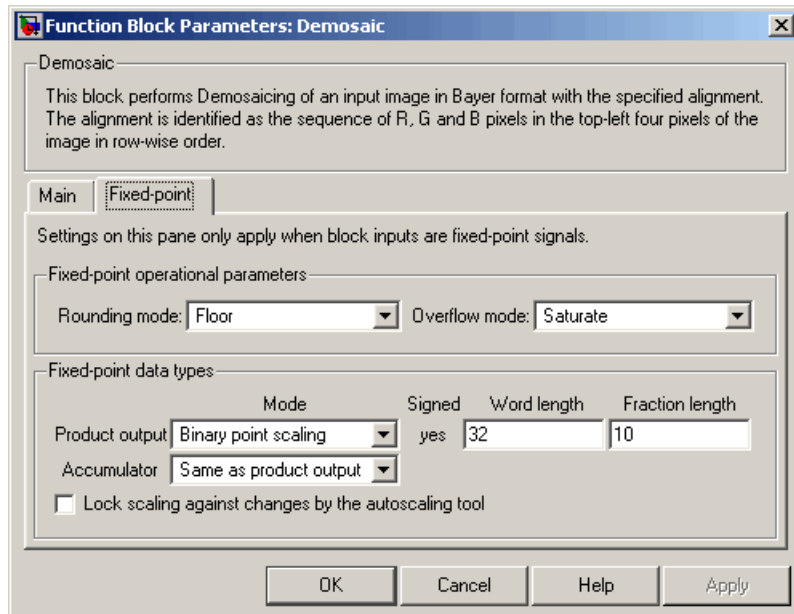
Select the sequence of R, G and B pixels that correspond to the 2-by-2 block of pixels in the top left corner of the image. You specify the sequence in left-to-right, top-to-bottom order.

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 **Fixed-point** pane of the Demosaic dialog box appears as shown in the following figure.



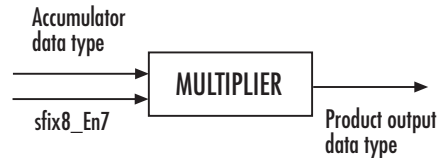
Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output



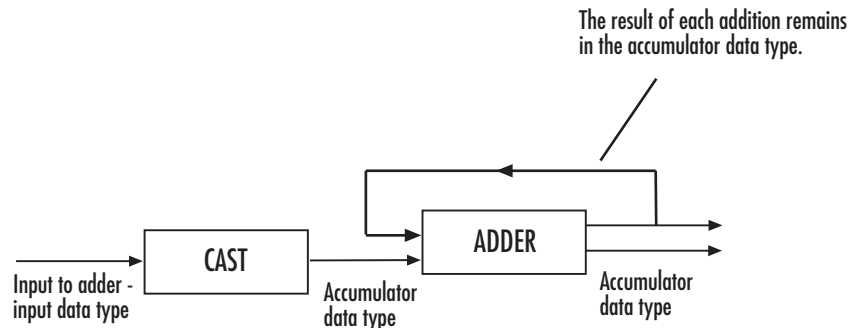
As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 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.

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.

Dilation

Purpose

Find local maxima in binary or intensity images

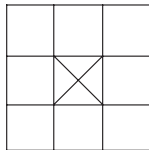
Library

Morphological Operations

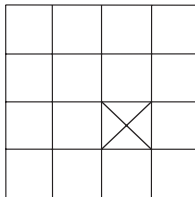
Description



The Dilate 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.

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.

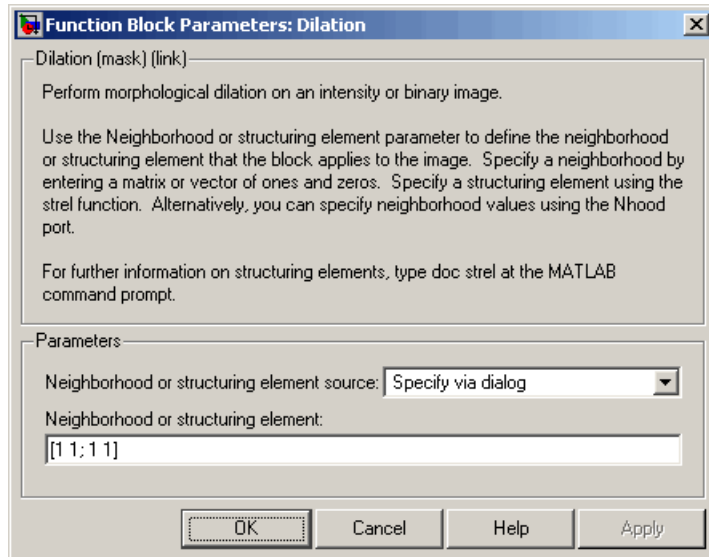
Use the **Neighborhood or structuring element** parameter to define the neighborhood or structuring element that the block applies to the image. Specify a neighborhood by entering a matrix or vector of 1s and 0s. Specify a structuring element with the `strel` function from Image Processing Toolbox. If the structuring element is decomposable into smaller elements, the block executes at higher speeds due to the use of

Dilation

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 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

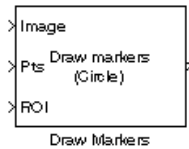
Bottom-hat	Video and Image Processing Blockset
Closing	Video and Image Processing Blockset
Erosion	Video and Image Processing Blockset
Label	Video and Image Processing Blockset
Opening	Video and Image Processing Blockset
Top-hat	Video and Image Processing Blockset
imdilate	Image Processing Toolbox
strel	Image Processing Toolbox

Draw Markers

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. As a result, the shapes are embedded on the output image.



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 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
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 I port	No

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 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 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
Output	Scalar, vector, or matrix of pixel values that contain the shape(s)	Same as I 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 the shape of the markers. Your choices are Circle, X-mark, Plus, Star, or Square.

Draw Markers

Use the **Marker size** parameter to define the size of the marker, in pixels. Enter a scalar value, M , that defines a $2M$ -by- $2M$ pixel square into which the marker fits. M must be greater than or equal to 1.

If you clear the **Filled** check box, the **Border value** parameter appears in the dialog box. Use this parameter to determine the appearance of any markers. If you select **Black**, the border is black. If you select **White**, the border is white. If you select **User-specified value**, the **Value(s)** parameter appears in the dialog box. The following table describes the what to enter for the **Value(s)** parameter based on the block input and the number of markers you are drawing:

Block Input	Drawing One Shape	Drawing Multiple Shapes
Intensity image	Value(s) = Scalar intensity value	Value(s) = R -element vector where R is the number of shapes
Color image	Value(s) = P -element vector where P is the number of color planes	Value(s) = P -by- R matrix where P is the number of color planes and R is the number of shapes

For each value in the **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.

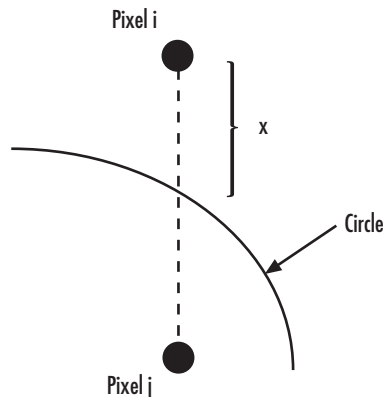
If you select the **Filled** check box, the **Fill value** and **Opacity factor (between 0 and 1)** parameters appear in the dialog box. Use the **Fill value** parameter to specify the shading inside the shape. If you select **Black**, the shape is black. If you select **White**, the shape is white. If you select **User-specified value**, the **Value(s)** parameter appears in the dialog box. For a description of this parameter, see the preceding table. 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.

Note If you are generating code and you select the **Filled** check box, the word length of the block input(s) cannot exceed 16 bits.

Use the **Draw markers in** parameter to 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 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. If you specify values that are outside the image, the block clips the values to the image boundaries.

If, for the **Marker shape** parameter, you select `X-mark`, `Plus`, or `Star`, and you select the **Use antialiasing** check box, the block performs the smoothing algorithm described in [1].

If, for the **Marker shape** parameter, you select `Circle`, and you select the **Use antialiasing** check box, the block performs a smoothing algorithm that is described next. First, the block calculates the distance from a point on the circle to the nearest pixel.



Draw Markers

Then, it uses the following equations to calculate the intensity at pixels i and j :

$$I_{i_{NEW}} = factor * [I * (1 - x) + I_{i_{OLD}} * x] + I_{i_{OLD}} * (1 - factor)$$

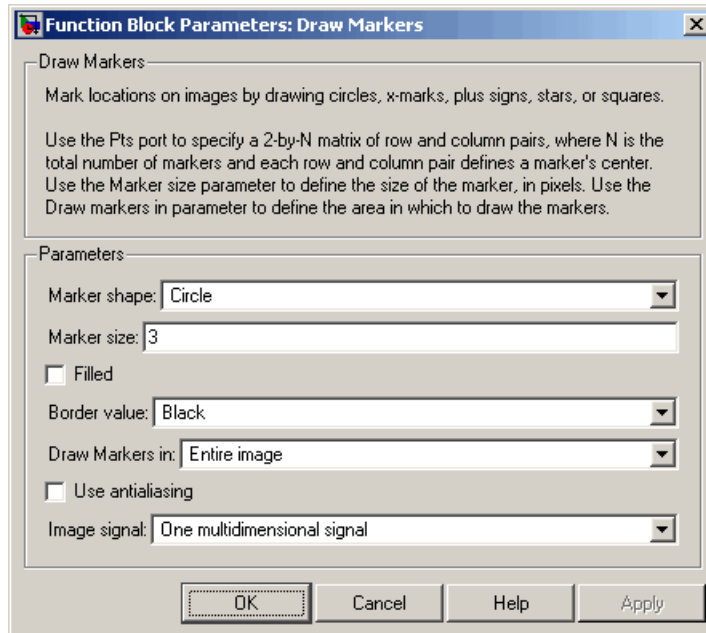
$$I_{j_{NEW}} = factor * [I * x + I_{j_{OLD}} * (1 - x)] + I_{j_{OLD}} * (1 - factor)$$

In the previous equations, *factor* is the value you entered for the **Opacity factor (between 0 and 1)** parameter. If you did not enter an opacity factor, the block sets it to 1. *I* is the value you entered for the **Border value** or **Fill value** parameter.

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.

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.

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.

Draw Markers

Border value

Specify the appearance of the marker. If you select **Black**, the marker is black. If you select **White**, the marker border is white. If you select **User-specified value**, the **Value(s)** parameter appears in the dialog box.

Fill value

Specify how to fill the marker. If you select **Black**, the block fills the marker with black. If you select **White**, the block fills the marker with white. If you select **User-specified value**, the **Value(s)** parameter appears in the dialog box. This parameter is visible if you select the **Filled** check box.

Value(s)

Specify an intensity or color value for the marker's border. This parameter is visible if, for the **Border value** or **Fill value** parameter, you select **User-specified value**. This parameter is 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

Select this check box to perform a smoothing algorithm on the marker. This parameter is visible if, for the **Marker shape** parameter, you select **Circle**, **X-mark**, **Plus**, 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.

References

[1] Gupta, S. and R.F. Sproull, "Filtering Edges for Gray-Scale Displays", *Computer Graphics*, Vol. 15, No. 3, August 1981.

See Also

Draw Shapes

Video and Image Processing
Blockset

Insert Text

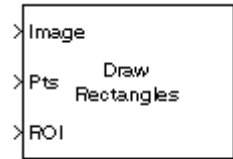
Video and Image Processing
Blockset

Draw Shape (Obsolete)

Purpose Draw rectangle around region of interest (ROI)

Library vipobslib

Description The Draw Shape block is obsolete. It may be removed in a future version of Video and Image Processing Blockset. Use the replacement block Draw Shapes.



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

Draw Shape (Obsolete)

Port	Input/Output	Supported Data Types	Complex Values Supported
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
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.

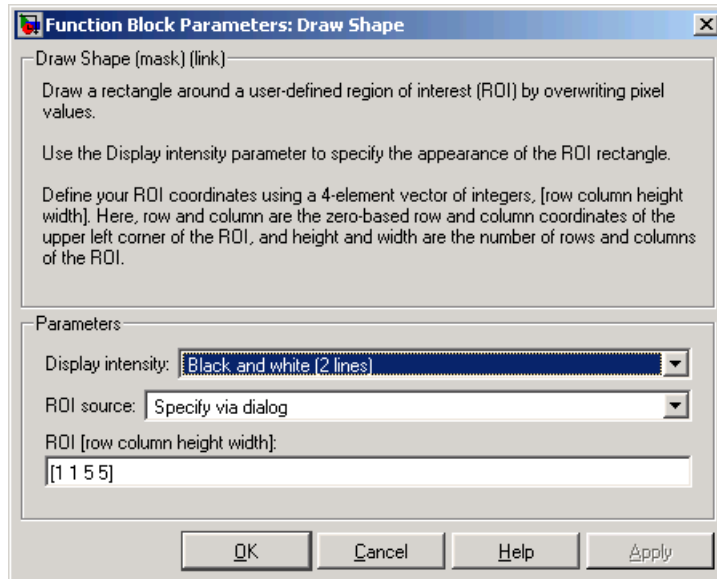
Use the **ROI source** parameter to determine how to enter your ROI coordinates. If you select **Specify via dialog**, the **ROI [row column height width]** parameter appears on the dialog box. Enter a four-element vector of integers. The first two elements represent the zero-based row and column coordinates 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.

Draw Shape (Obsolete)

The input to this port must be a four-element of integers as previously defined.

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]

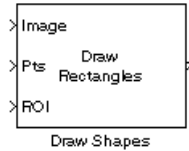
Enter a four-element vector of integers. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the ROI. The second two elements represent the height and width of the ROI. Tunable.

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 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
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 I port	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Pts	Use integer values to define zero-based shape coordinates. If you enter noninteger values, the block rounds them to the nearest integer.	<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	Four-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.	<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
Output	Scalar, vector, or matrix of pixel values that contain the shape(s)	Same as I 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 **Shape** parameter to specify the type of shape(s) to draw. Your choices are Rectangles, Lines, Polygons, or Circles.

Draw Shapes

If you clear the **Fill shapes** check box, the **Border value** parameter appears in the dialog box. Use this parameter determine the appearance of the rectangle(s), line(s), polygon(s), or circle(s). If you select **Black**, the border is black. If you select **White**, the border is white. If you select **User-specified value**, the **Value(s)** parameter appears in the dialog box. The following table describes what to enter for the **Value(s)** parameter based on the block input and the number of shapes you are drawing.

Block Input	Drawing One Shape	Drawing Multiple Shapes
Intensity image	Value(s) = Scalar intensity value	Value(s) = R-element vector where R is the number of shapes
Color image	Value(s) = P-element vector where P is the number of color planes	Value(s) = P-by-R matrix where P is the number of color planes and R is the number of shapes

For each value in the **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.

If you select the **Fill shapes** check box, the **Fill value** and **Opacity factor (between 0 and 1)** parameters appear in the dialog box. Use the **Fill value** parameter to specify the shading inside the shape. If you select **Black**, the shape is black. If you select **White**, the shape is white. If you select **User-specified value**, the **Value(s)** parameter appears in the dialog box. For a description of this parameter, see the preceding table. 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.

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.

Use the **Draw shapes in** parameter to 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 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. If you specify values that are outside the image, the block sets the values to the image boundaries.

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.

Drawing Rectangles

If to draw one rectangle, for the **Shape** parameter, choose *Rectangles*. The input to the Pts port must be a four-element vector of the form [r c height width], where r and c are the zero-based row and column coordinates of the upper-left corner of the rectangle, and height and width represent the height, in rows, and width, in columns, of the rectangle. Here, height and width must be greater than 0.

If you want to draw N rectangles, for the **Shape** parameter, choose *Rectangles*. The input to the Pts port must be a 4-by-N matrix where each column defines a rectangle as described in the previous paragraph. For example, to draw two rectangles enter the following at the Pts port:

Draw Shapes

$$\begin{bmatrix} r_1 & r_2 \\ c_1 & c_2 \\ height_1 & height_2 \\ width_1 & width_2 \end{bmatrix}$$

Drawing Lines and Polylines

If you want to draw one line, for the **Shape** parameter, choose `Lines`. The input to the `Pts` port must be a four-element vector of the form `[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.

If you want to draw one polyline, which is a series of connected line segments, for the **Shape** parameter, choose `Lines`. For a polyline with `L-1` line segments, the input to the `Pts` port must be a vector of size `2L`, `[r1 c1 r2 c2 ... rL cL]`. Here, `r1` and `c1` are the row and column coordinates of the beginning of the first polyline, `r2` and `c2` are the row and column coordinates of the end of the first polyline and the beginning of the second polyline, etc. The block produces an error message if the number of rows is less than two or is not a multiple of two.

If you want to draw `N` polylines, for the **Shape** parameter, choose `Lines`. If the biggest polyline has `L-1` line segments, the input to the `Pts` port must be a `2L-by-N` matrix, where each column of the matrix corresponds to a different polyline, as described in the previous paragraph. If some polylines are shorter than others, repeat the ending coordinates to fill the polyline matrix. The block produces an error message if the number of rows is less than two or is not a multiple of two.

If you select the **Use antialiasing** check box, the block performs a smoothing algorithm. See [1].

Drawing Polygons

If you want to draw one polygon, for the **Shape** parameter, choose `Polygons`. For a polygon with `L` line segments, the input to the `Pts` port must be a vector of size `2L`, `[r1 c1 r2 c2 ... rL cL]`. In this case, `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, etc. 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.

If you want to draw N polygons, for the **Shape** parameter, choose **Polygons**. If the biggest polygon has L line segments, the input to the **Pts** port must be a $2L$ -by- N matrix, where each column of the matrix corresponds to a different polygon, as described in the previous paragraph. If some polygons have fewer line segments, repeat the ending coordinates until you fill the polygon matrix. The block produces an error if the number of rows is negative or not a multiple of two.

If you select the **Use antialiasing** check box, the block performs a smoothing algorithm. See [1].

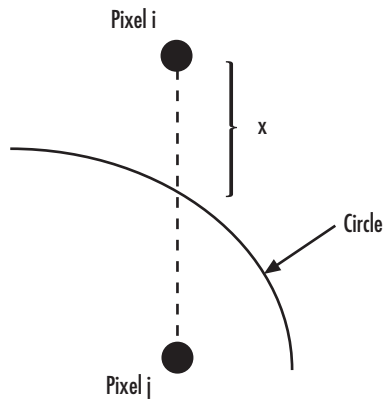
Drawing Circles

If you want to draw one circle, for the **Shape** parameter, choose **Circles**. The input to the **Pts** port must be a three-element vector of the form $[r\ c\ radius]$, where r and c are the row and column coordinates of the center of the circle. The radius is the radius of the circle, which must be greater than 0.

If you want to draw N circles, for the **Shape** parameter, choose **Circles**. The input to the **Pts** port must be a 3-by- N matrix, where each column defines a circle as described in the previous paragraph.

If you select the **Use antialiasing** check box, the block performs a smoothing algorithm that is described next. First, the block calculates the distance from a point on the circle to the nearest pixel.

Draw Shapes



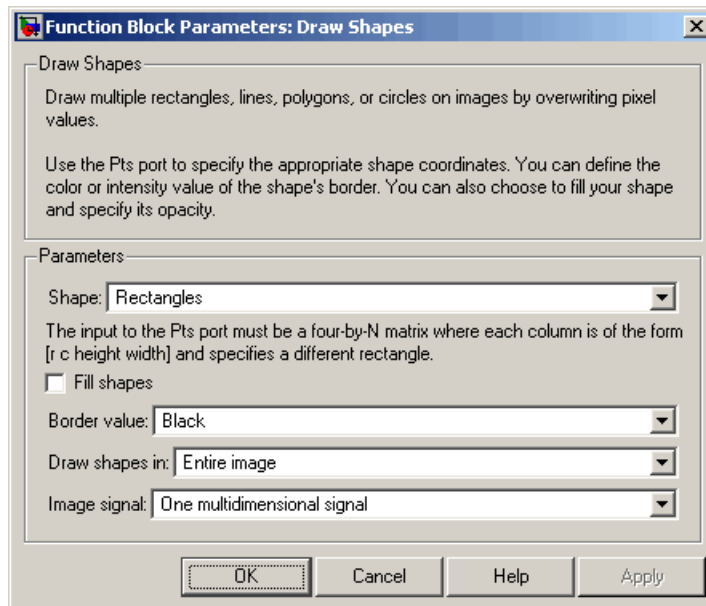
Then, it uses the following equations to calculate the intensity at pixels i and j :

$$I_{i_{NEW}} = factor * [I_B * (1 - x) + I_{i_{OLD}} * x] + I_{i_{OLD}} * (1 - factor)$$

$$I_{j_{NEW}} = factor * [I_B * x + I_{j_{OLD}} * (1 - x)] + I_{j_{OLD}} * (1 - factor)$$

In the previous equations, *factor* is the value you entered for the **Opacity factor** parameter. If you did not enter an opacity factor, the block sets it to 1. I_B is the value you entered for the **Border intensity** parameter.

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 value

Specify the intensity of the shading inside the shape. This parameter is visible if you select the **Fill shapes** check box.

Border value

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 **Value(s)** parameter appears in the dialog box. This parameter is visible if you clear the **Fill shapes** check box.

Draw Shapes

Value(s)

Specify an intensity or color value for the shape's border. This parameter is visible if, for the **Border value** or **Fill value** 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 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.

References

[1] Gupta, S. and R.F. Sproull, "Filtering Edges for Gray-Scale Displays", *Computer Graphics*, Vol. 15, No. 3, August 1981.

See Also

[Draw Markers](#)

[Video and Image Processing Blockset](#)

[Insert Text](#)

[Video and Image Processing Blockset](#)

Edge Detection

Purpose

Find edges of objects in images using Sobel, Prewitt, Roberts, or Canny method

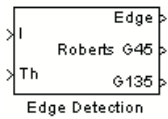
Library

Analysis & Enhancement

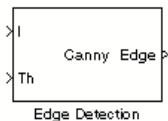
Description



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) • 8-, 16-, 32-bit unsigned integer (not supported for the Canny method) 	No
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.

Edge Detection

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.

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.

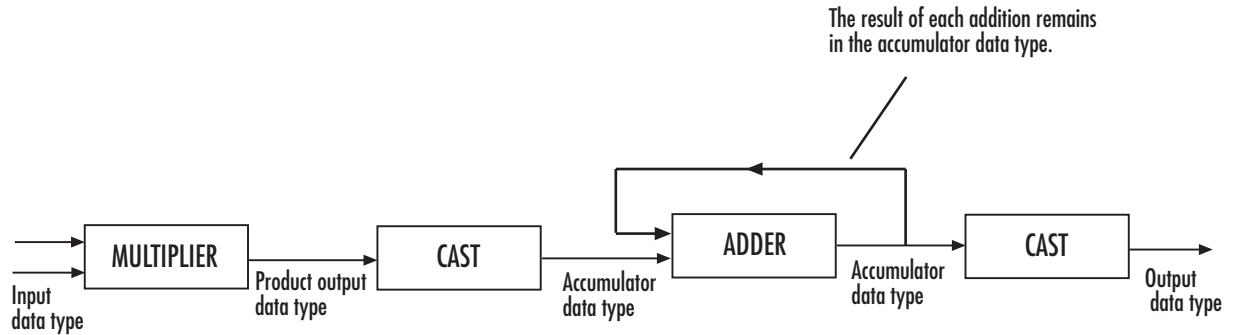
The Edge Detection block computes the automatic threshold values using an approximation of the number of weak and nonedge image pixels. Enter this approximation for the **Approximate percentage of weak edge and nonedge pixels (used to automatically calculate threshold values)** parameter.

Use the **Standard deviation of Gaussian filter** parameter to define the Gaussian filter whose derivative is convolved with the input image.

Edge Detection

Fixed-Point Data Types

The following diagram shows the data types used in the Edge Detection block for fixed-point signals.

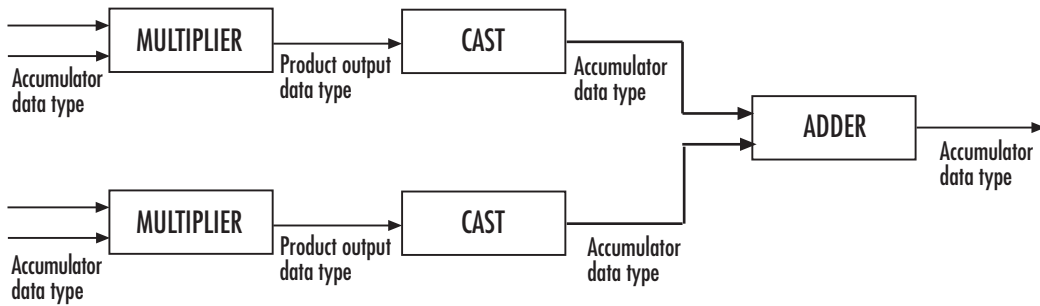


The block squares the threshold and compares it to the sum of the squared gradients to avoid using square roots.

Threshold:



Gradients:

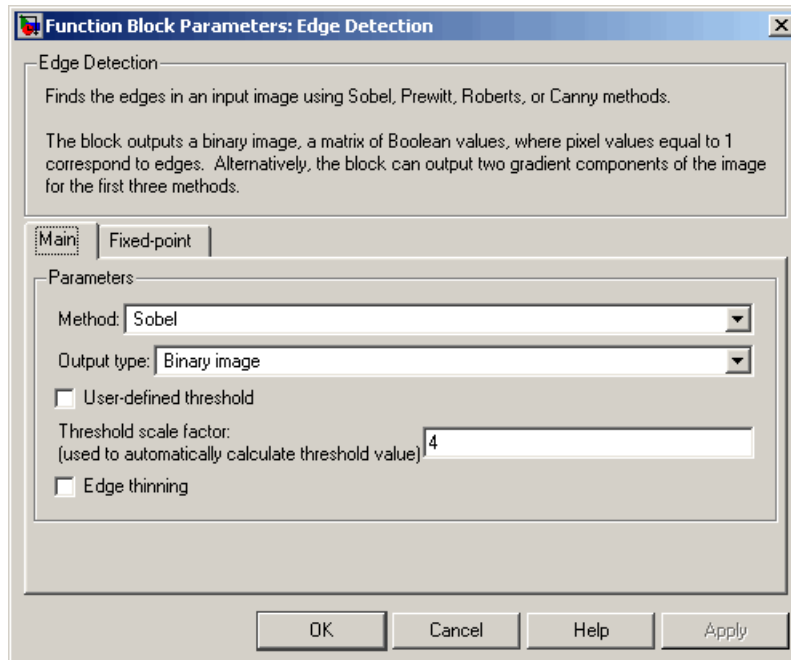


You can set the product output and accumulator data types in the block mask as discussed in the next section.

Edge Detection

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 select Sobel, Prewitt, or Roberts, you select the **User-defined threshold** check box, and, for **Threshold source** parameter, you select Specify via dialog. Tunable.

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.

Edge Detection

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.

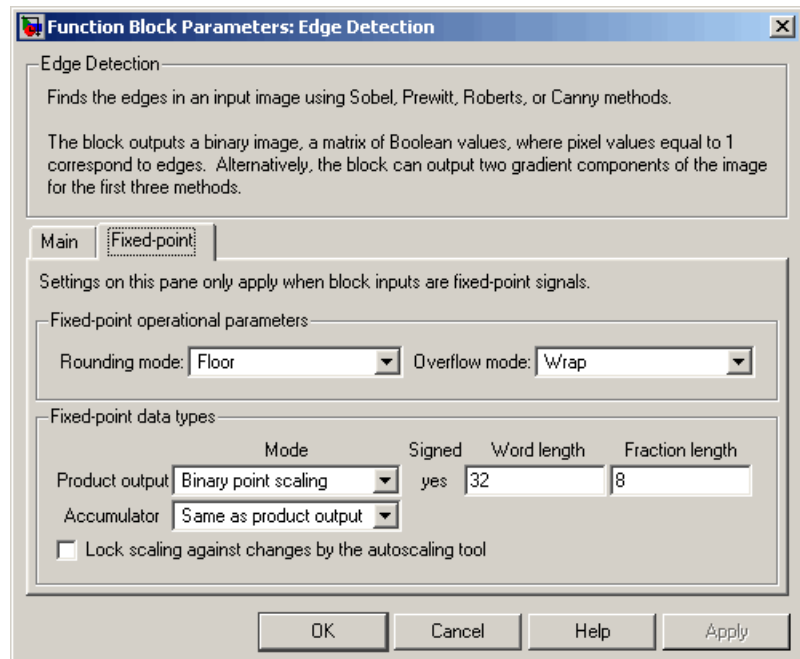
Approximate percentage of weak edge and nonedge pixels (used to automatically calculate threshold values)

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 **Fixed-point** 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

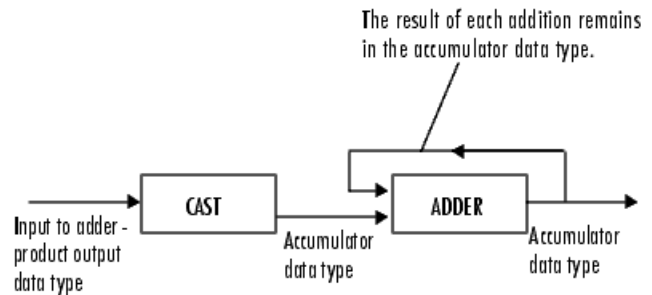
Product output



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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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:

- When you select Same as accumulator, these characteristics match those of the accumulator.

Edge Detection

- 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 Video and Image Processing Blockset is 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.

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

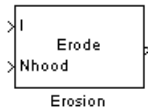
Purpose

Find local minima in binary or intensity images

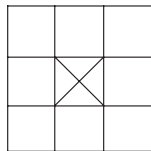
Library

Morphological Operations

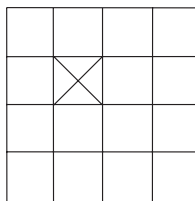
Description



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.

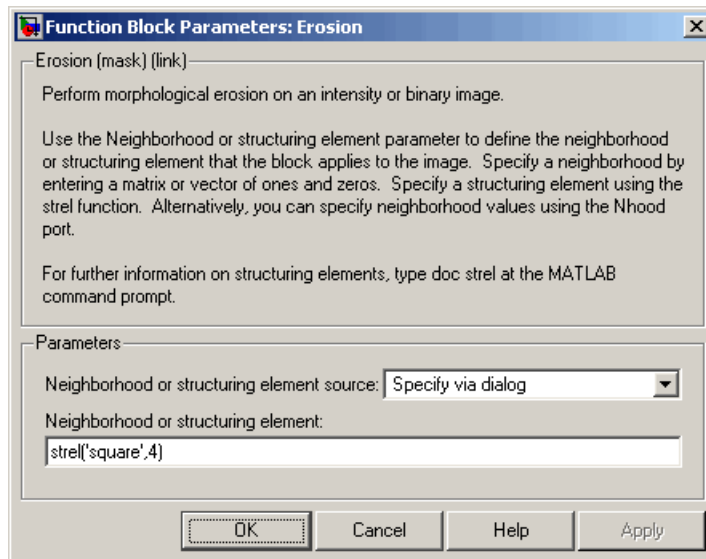
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select Specify via dialog, the **Neighborhood or structuring element** parameter appears in the dialog box. If you select Input port, the Nhood port appears 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 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 Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or structuring element source** parameter, you select `Specify via dialog`.

Erosion

References

[1] Soille, Pierre. *Morphological Image Analysis. 2nd ed.* New York: Springer, 2003.

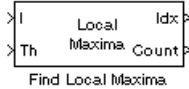
See Also

Bottom-hat	Video and Image Processing Blockset
Closing	Video and Image Processing Blockset
Dilation	Video and Image Processing Blockset
Label	Video and Image Processing Blockset
Opening	Video and Image Processing Blockset
Top-hat	Video and Image Processing Blockset
imerode	Image Processing Toolbox
strel	Image Processing Toolbox

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.

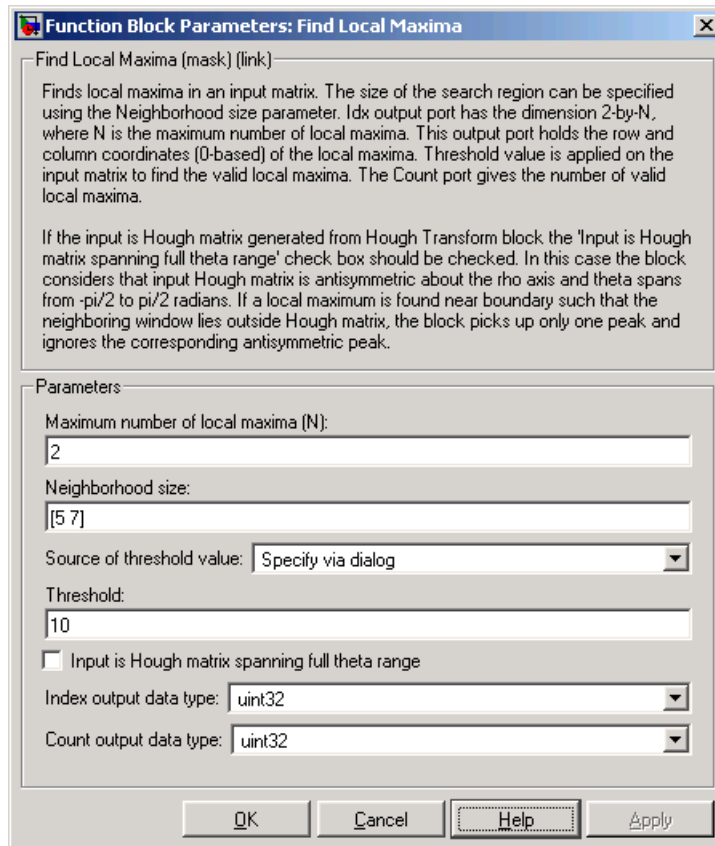
Examples

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

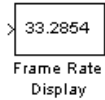
Hough Lines	Video and Image Processing Blockset
Hough Transform	Video and Image Processing Blockset
houghpeaks	Image Processing Toolbox

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.

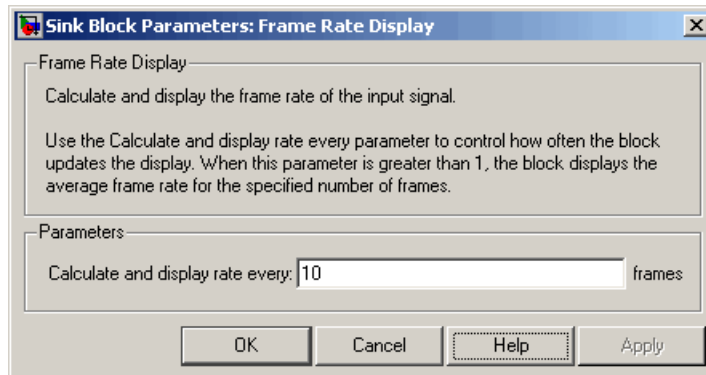
Port	Input	Supported Data Types	Complex Values Supported
Input	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	<ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point• Fixed point• Boolean• 8-, 16-, 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	Video and Image Processing Blockset
To Video Display	Video and Image Processing Blockset
Video To Workspace	Video and Image Processing Blockset
Video Viewer	Video and Image Processing Blockset

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 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:

$$S_{LS} = \frac{1}{\frac{\gamma}{B_P^{(\frac{1}{\gamma}-1)}} - \gamma B_P + B_P}$$

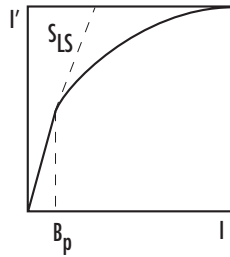
$$F_S = \frac{\gamma S_{LS}}{B_P^{(\frac{1}{\gamma}-1)}}$$

$$C_O = F_S B_P^{\frac{1}{\gamma}} - S_{LS} B_P$$

$$I' = \left\{ \begin{array}{ll} S_{LS} I, & I \leq B_P \\ F_S I^{\frac{1}{\gamma}} - C_O, & I > B_P \end{array} \right\}$$

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.

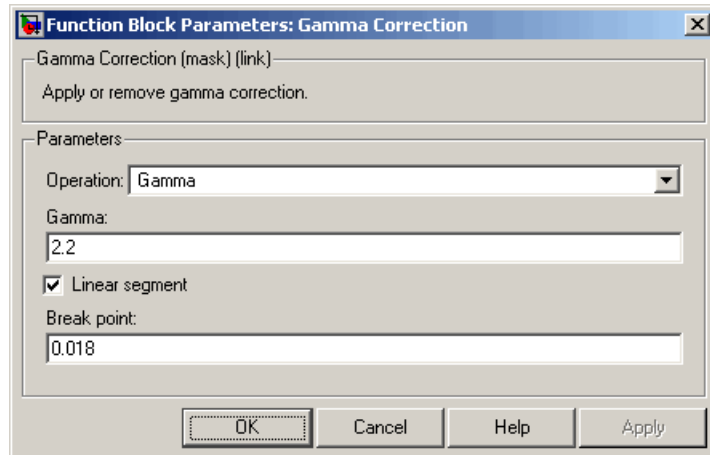
Select the **Linear segment** check box if you want the gamma curve to have a linear portion near black. If you select this check box, the **Break point** parameter appears on the dialog box. Enter a scalar value that

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. Tunable.

Linear segment

Select this check box if you want the gamma curve to have a linear portion near the origin. Tunable.

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. Tunable.

References

[1] Poynton, Charles. *Digital Video and HDTV Algorithms and Interfaces*. San Francisco, CA: Morgan Kaufman Publishers, 2003.

See Also

Color Space Conversion
imadjust

Video and Image Processing Blockset
Image Processing Toolbox

Gaussian Pyramid

Purpose Perform Gaussian pyramid decomposition

Library Transforms

Description



The Gaussian Pyramid block uses Gaussian pyramid decomposition 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 2-350.

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	<p>In Reduce mode, the output can be a scalar, 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>	Same as Input port	No

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

Gaussian Pyramid

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:

$$\left[(M-1)2^l + 1 \right] \text{ by } \left[(N-1)2^l + 1 \right]$$

In the previous equation, l is the scalar value from 1 to inf 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

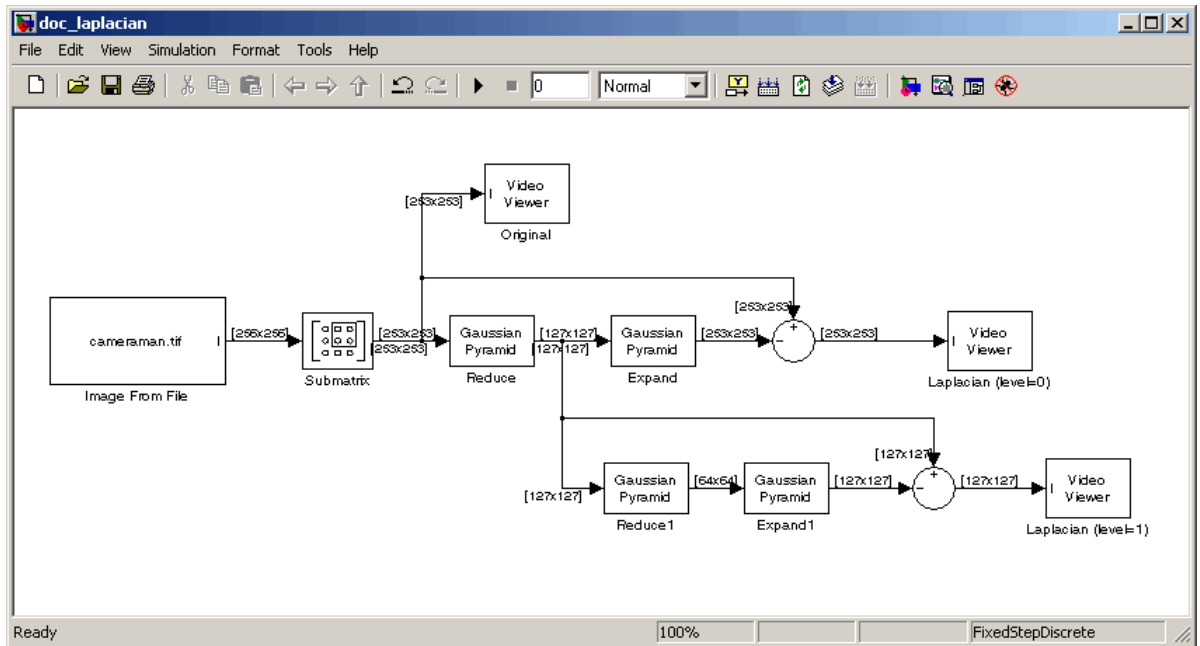
The following example model shows how to construct a Laplacian pyramid:

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

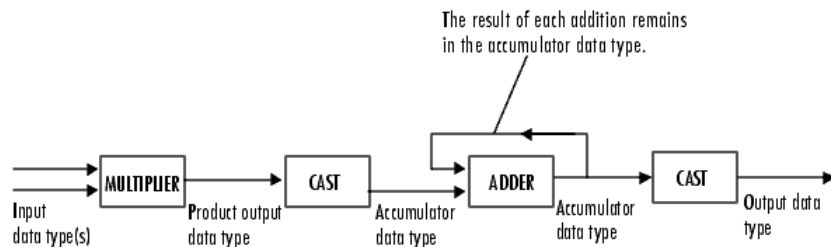




You can construct a Laplacian pyramid if the dimensions of the input image, R-by-C, satisfy $R = M_R 2^N + 1$ and $C = M_C 2^N + 1$, where M_R , M_C , and N are integers. In this example, you have an input matrix that is 256-by-256. If you set M_R and M_C equal to 63 and N equal to 2, you find that the 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:

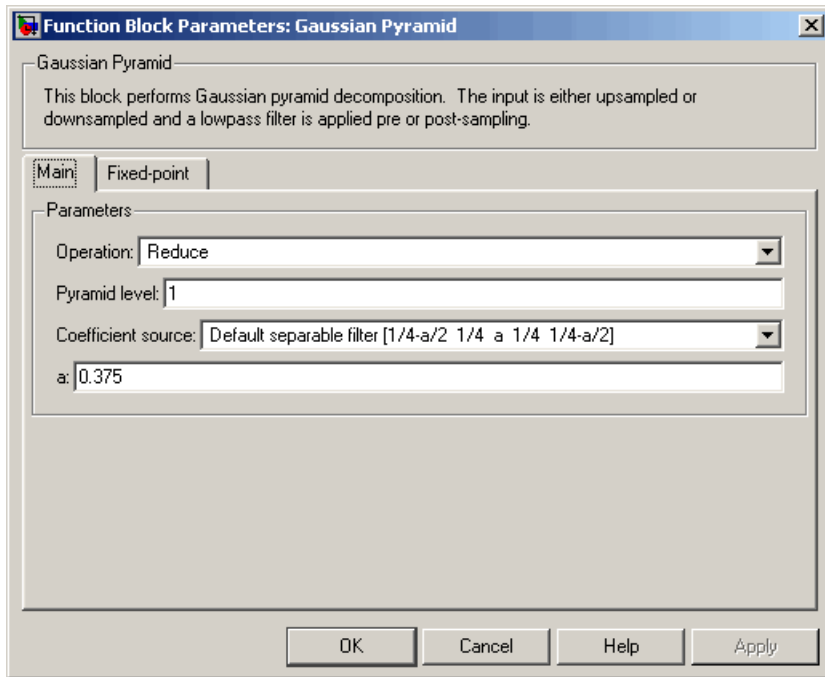


You can set the coefficients table, product output, accumulator, and output data types in the block mask.

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.

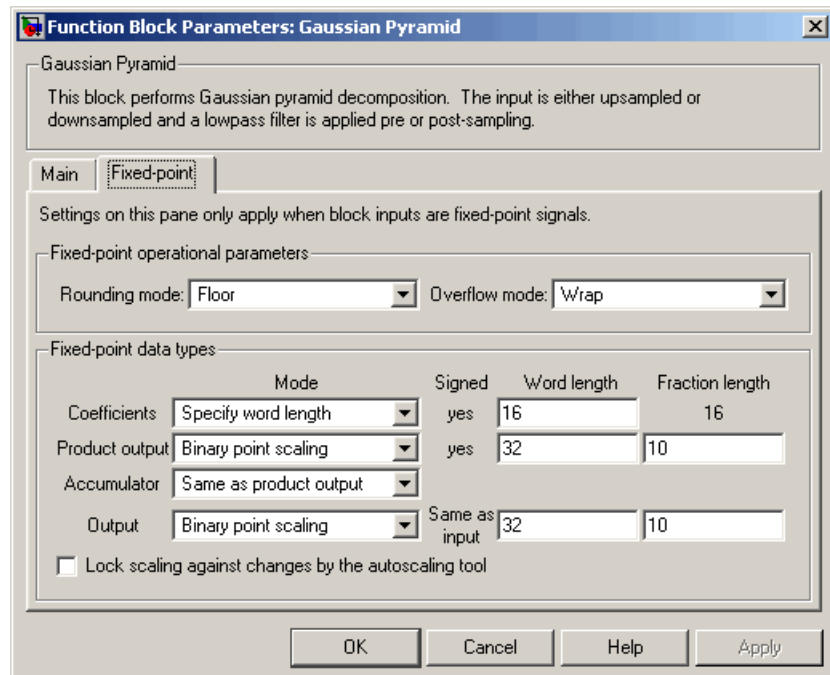
a

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 **Fixed-point** 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 Video and Image Processing Blockset is 0.

Product output

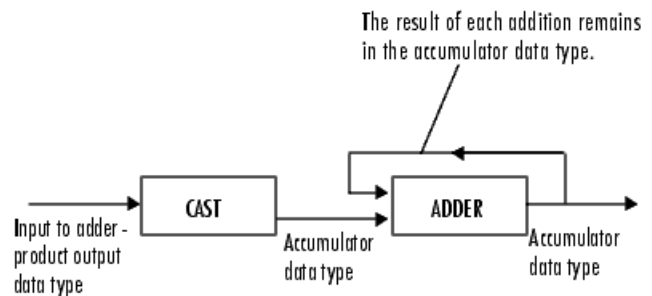


As shown in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate the product output word and fraction lengths.

- 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.
- 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 0.

Gaussian Pyramid

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 Video and Image Processing Blockset is 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

Resize

Video and Image Processing Blockset

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.

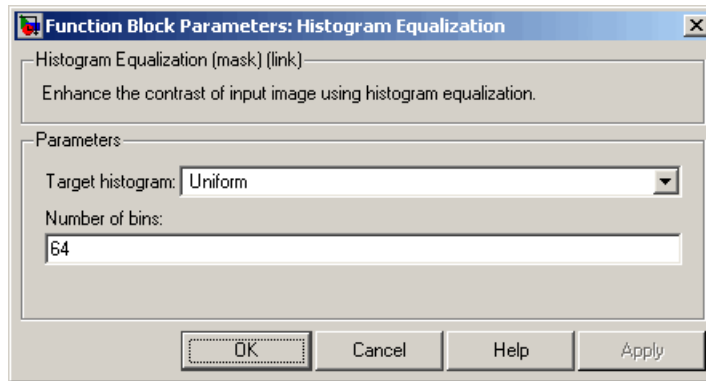
Note The vector input to the Hist port must be normalized such that the sum of the values in all the bins is equal to the number of pixels in the input image. The block does not error if the histogram is not normalized.

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**. Tunable.

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

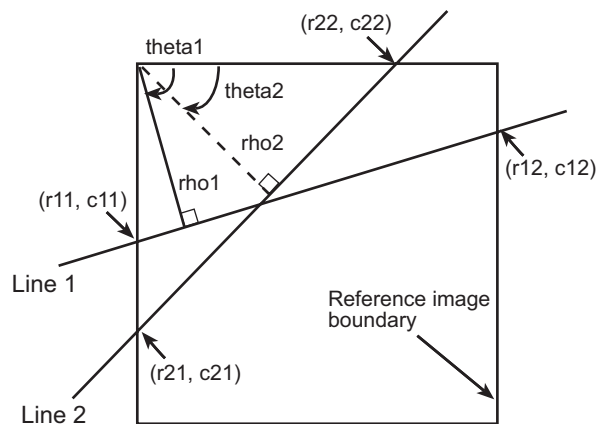
Transforms

Description



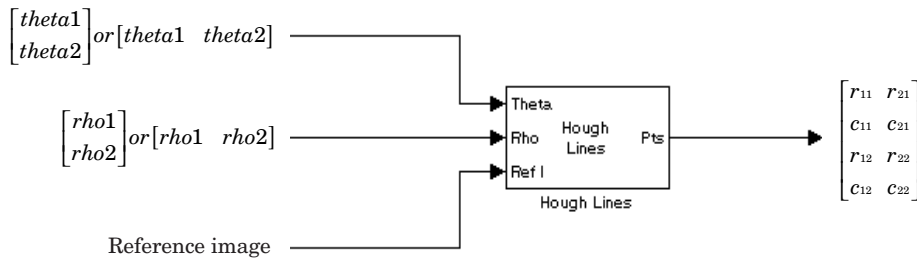
The Hough Lines block inputs are the theta and rho values of lines and a reference image. The block outputs the zero-based row and column positions of the intersections between the lines and two of the reference image boundary lines. The boundary lines are the left and right vertical boundaries and the top and bottom horizontal boundaries of the reference image.

Suppose that Line1 and Line2 intersect the boundaries of the reference image as shown in the following figure.



When the reference image and the theta and rho values of these lines are passed into the Hough Lines block, it outputs the coordinates of the intersections, as shown in the following figure.

Hough Lines



Port	Input/Output	Supported Data Types	Complex Values Supported
Theta	Vector of theta values that represent input line(s)	<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 line(s)	Same as Theta port	No

Port	Input/Output	Supported Data Types	Complex Values Supported
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
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

Use the **Sine value computation method** parameter to specify how much memory the Hough Lines block requires. If you select `Trigonometric function`, the block computes sine and cosine values it needs to calculate the intersections of the lines during the simulation. If you select `Table lookup`, the block computes and stores the trigonometric values it needs to calculate the intersections of the lines before the simulation starts. In this case, the block requires extra memory.

Note For floating-point inputs, the **Sine value computation method** parameter must be set to `Trigonometric function`. For fixed-point inputs, the parameter must be set to `Table lookup`.

If, for the **Sine value computation method** parameter, you select `Table lookup`, the **Theta resolution (radians)** parameter appears

Hough Lines

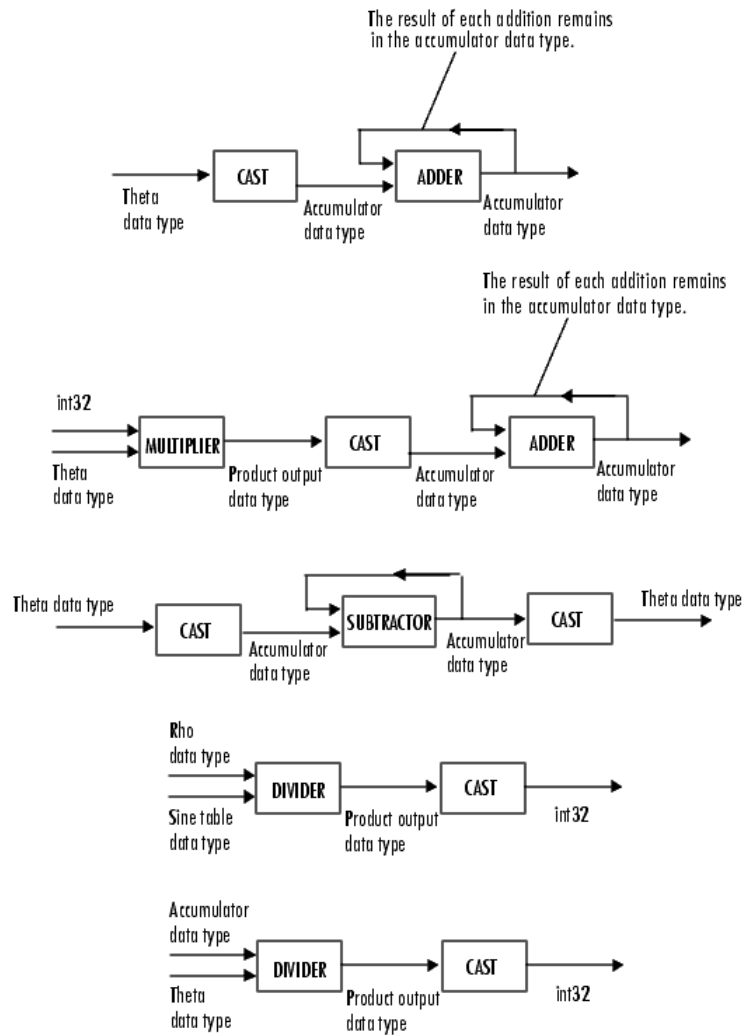
in the dialog box. Use this parameter to specify the spacing of the theta-axis.

Examples

See “Finding Lines in Images” and “Measuring an Angle Between Lines” in the *Video and Image Processing Blockset User’s Guide*.

Fixed-Point Data Types

The following diagram shows the data types used in the Hough Lines block for fixed-point signals.

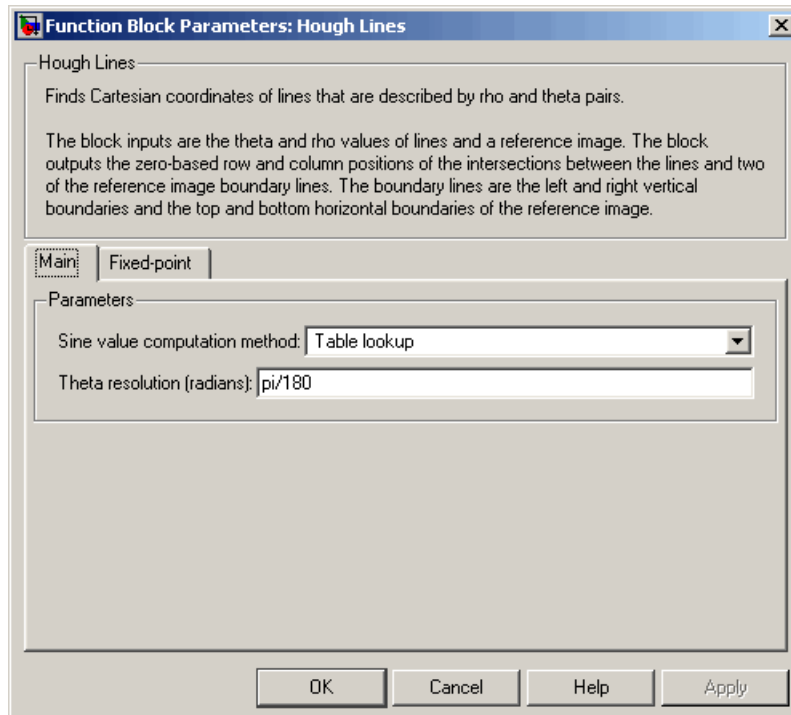


You can set the sine table, product output, and accumulator output data types in the block mask as discussed in the next section.

Hough Lines

Dialog Box

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



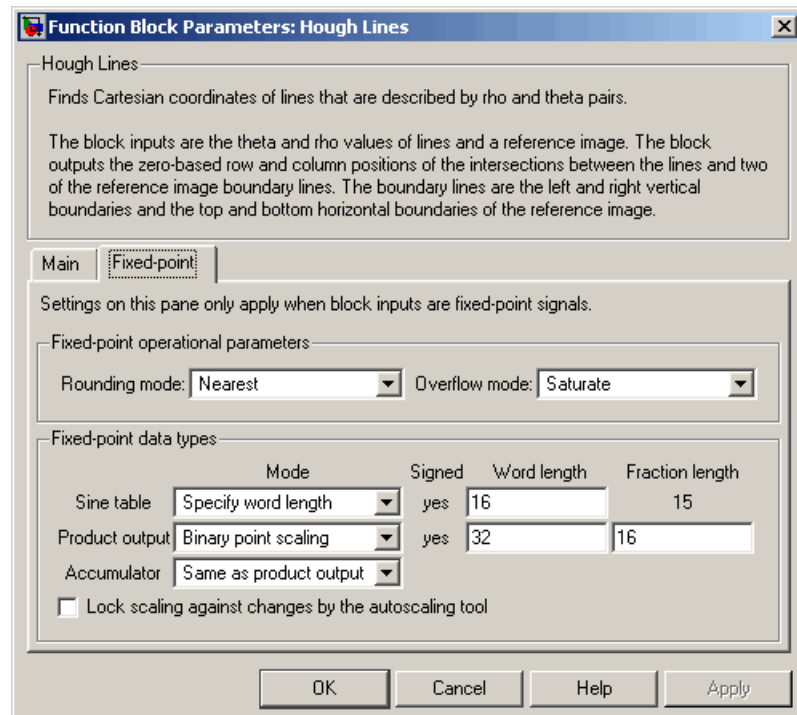
Sine value computation method

If you select `Trigonometric function`, the block computes sine and cosine values it needs to calculate the intersections of the lines during the simulation. If you select `Table lookup`, the block computes and stores the trigonometric values it needs to calculate the intersections of the lines before the simulation starts. In this case, the block requires extra memory. For floating-point inputs, this parameter must be set to `Trigonometric function`. For fixed-point inputs, the parameter must be set to `Table lookup`.

Theta resolution (radians)

Specify the spacing of the theta-axis. This parameter is visible if, for the **Sine value computation method** parameter, you choose Table lookup.

The **Fixed-point** pane of the Hough Lines 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.

Hough Lines

Sine table

Choose how to specify the word length of the values of the sine table. The fraction length of the sine table values is always equal to 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 are always saturated and rounded to Nearest.

Product output



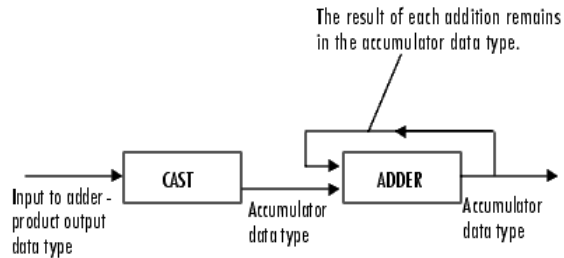
As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

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 Video and Image Processing Blockset 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 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 Video and Image Processing Blockset is 0.

See Also

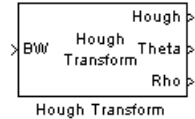
2-D DCT	Video and Image Processing Blockset
2-D FFT	Video and Image Processing Blockset
2-D IDCT	Video and Image Processing Blockset
2-D IFFT	Video and Image Processing Blockset
Find Local Maxima	Video and Image Processing Blockset
Hough Transform	Video and Image Processing Blockset

Hough Transform

Purpose Find lines in images

Library Transforms

Description



Use the Hough Transform block to find lines in an image. The block maps points in the Cartesian image space to curves in the Hough parameter space using the following equation:

$$\rho = x * \cos(\theta) + y * \sin(\theta)$$

The block outputs a parameter space matrix whose rows and columns correspond to the ρ and θ values, respectively. Peak values in this matrix represent potential lines in the input image. The θ values range from $-\pi/2$ radians to $\pi/2$ radians with a step-size determined by the **Theta resolution (radians)** parameter.

Port	Input/Output	Supported Data Types	Complex Values Supported
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

If the output of the Hough port is floating point, the outputs of the Theta and Rho ports are the same data type.

Use the **Theta resolution (radians)** parameter to specify the spacing of the Hough transform bins along the *theta*-axis.

Use the **Rho resolution** parameter to specify the spacing of the Hough transform bins along the *rho*-axis.

If you select the **Output theta and rho values** check box, the Theta and Rho ports appear on the block. The block outputs vectors of theta and rho values at these ports.

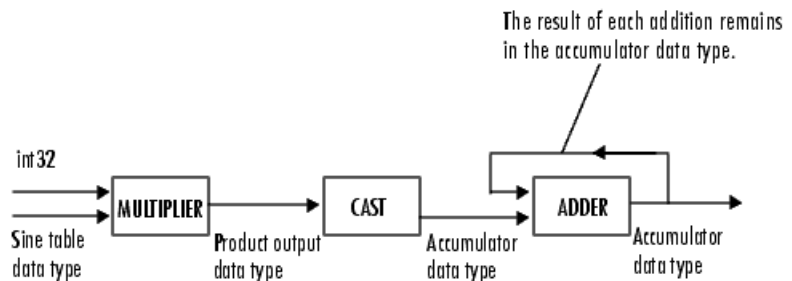
Use the **Output data type** parameter to specify the data type of your output signal.

Examples

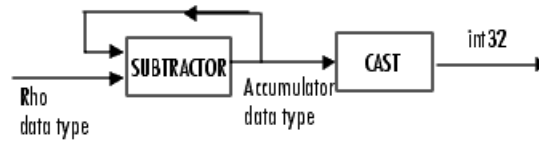
See “Finding Lines in Images” and “Measuring an Angle Between Lines” in the *Video and Image Processing Blockset User’s Guide*.

Fixed-Point Data Types

The following diagram shows the data types used in the Hough Transform block for fixed-point signals.



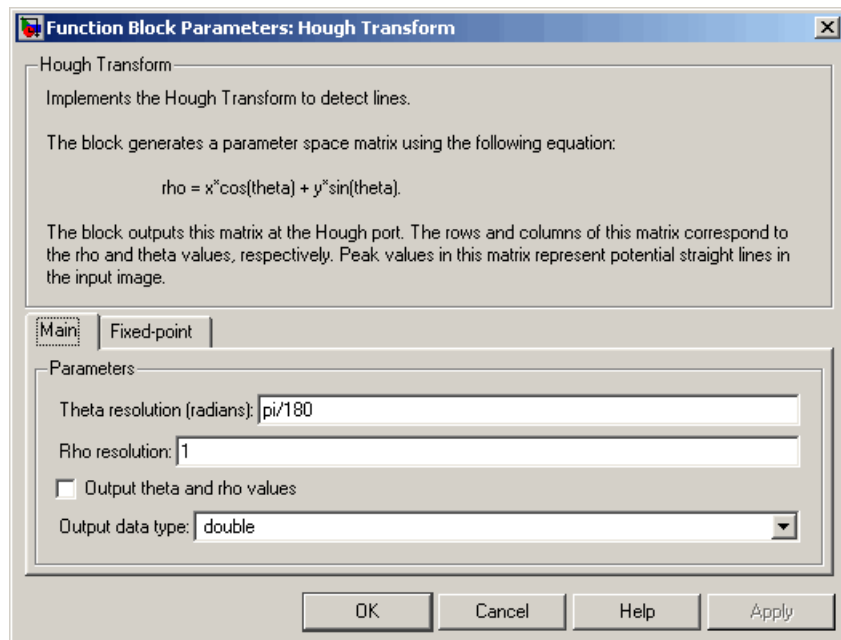
Hough Transform



You can set the sine table, rho, product output, accumulator, Hough output, and Theta output data types in the block mask as discussed in the next section.

Dialog Box

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

Specify the spacing of the Hough transform bins along the *rho*-axis.

Output theta and rho values

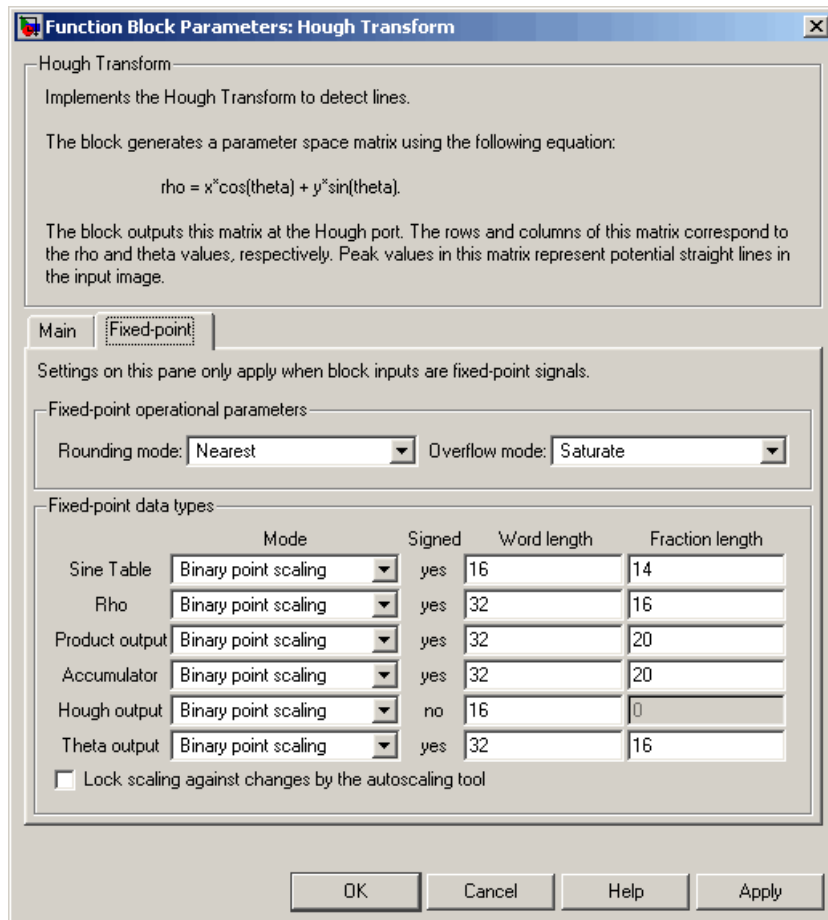
If you select this check box, the Theta and Rho ports appear on the block. The block outputs vectors of theta and rho values at these ports.

Output data type

Specify the data type of your output signal.

The **Fixed-point** pane of the Hough Transform dialog box appears as shown in the following figure.

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 are always saturated and rounded 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. The bias of all signals in Video and Image Processing Blockset is 0.

Product output

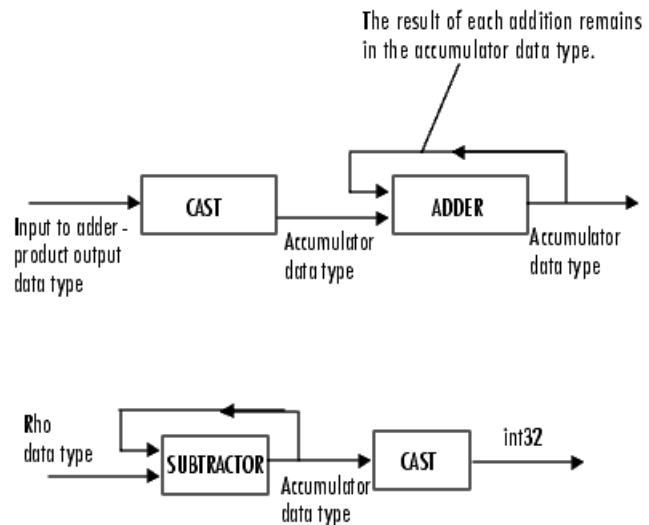


As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

Hough Transform

- 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 Video and Image Processing Blockset 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 **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 Video and Image Processing Blockset is 0.

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

See Also

2-D DCT	Video and Image Processing Blockset
2-D FFT	Video and Image Processing Blockset
2-D IDCT	Video and Image Processing Blockset
2-D IFFT	Video and Image Processing Blockset
Find Local Maxima	Video and Image Processing Blockset
Hough Lines	Video and Image Processing Blockset

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.



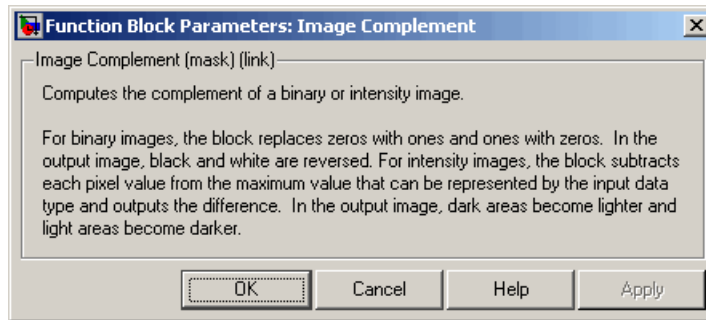
For example, suppose the input pixel values are given by $x(i)$ and the output pixel values are given by $y(i)$. If the data type of the input is double or single precision floating-point, the block outputs $y(i) = 1.0 - x(i)$. If the input is an 8-bit unsigned integer, the block outputs $y(i) = 255 - x(i)$.

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](#)

[Chroma Resampling](#)

[Video and Image Processing Blockset](#)

[Color Space Conversion](#)

[Video and Image Processing Blockset](#)

[imcomplement](#)

[Image Processing Toolbox](#)

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.

Image Data Type Conversion

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
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

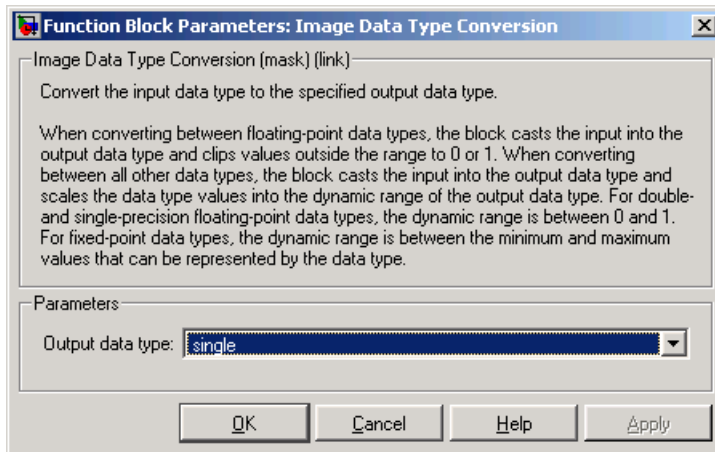
The dimensions, complexity, and frame status of the input and output signals are the same.

Use the **Output data type** parameter to specify the data type of your output signal values.

Image Data Type Conversion

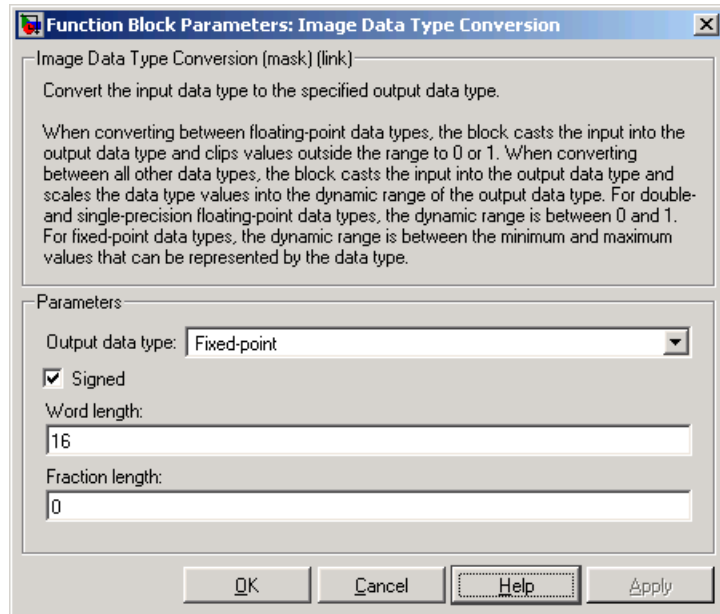
Dialog Box

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.



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

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 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 Simulink. 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.

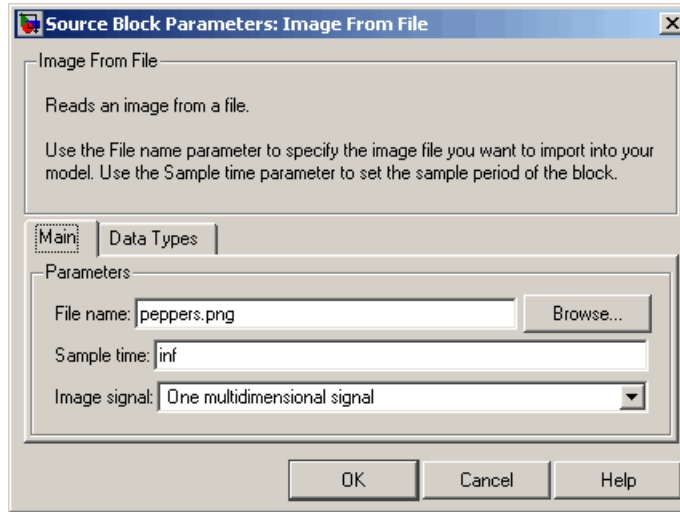
Use the **Output port labels** parameter to label your output ports. Use the spacer character, |, as the delimiter. This parameter is visible if you set the **Image signal** parameter to **Separate color signals**.

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 Simulink.

Sample time

Enter the sample period of the output signal.

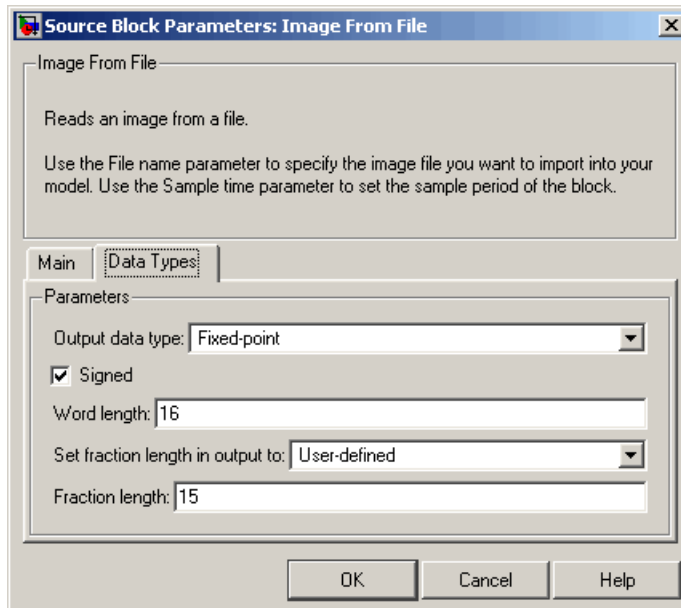
Image signal

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.

Image From File

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

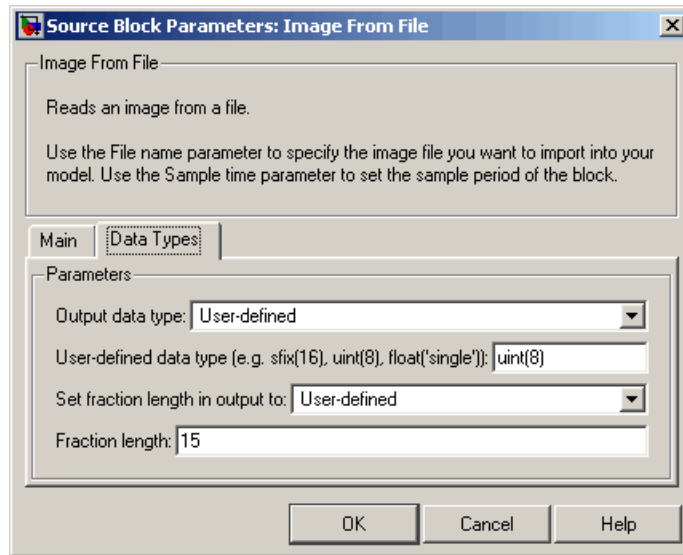
Specify the scaling of the fixed-point output by either of the following two methods:

- 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 Simulink Fixed Point. 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
Image From Workspace	Video and Image Processing Blockset
To Video Display	Video and Image Processing Blockset
Video From Workspace	Video and Image Processing Blockset
Video Viewer	Video and Image Processing Blockset

Image From File

`im2double`

Image Processing Toolbox

`im2uint8`

Image Processing Toolbox

`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 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.

Use the **Sample time** parameter to set the sample period of the output signal.

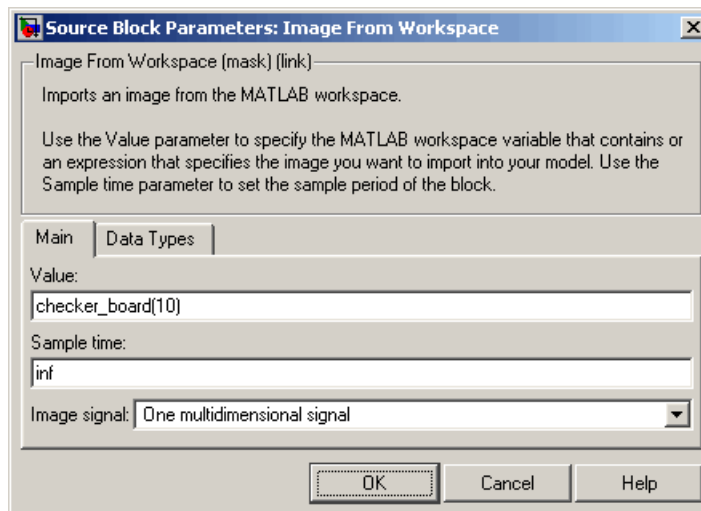
Use the **Image signal** parameter to specify how the block outputs a color video signal. If you select **One multidimensional signal**, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select **Separate color signals**, additional ports 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.

Sample time

Enter the sample period of the output signal.

Image signal

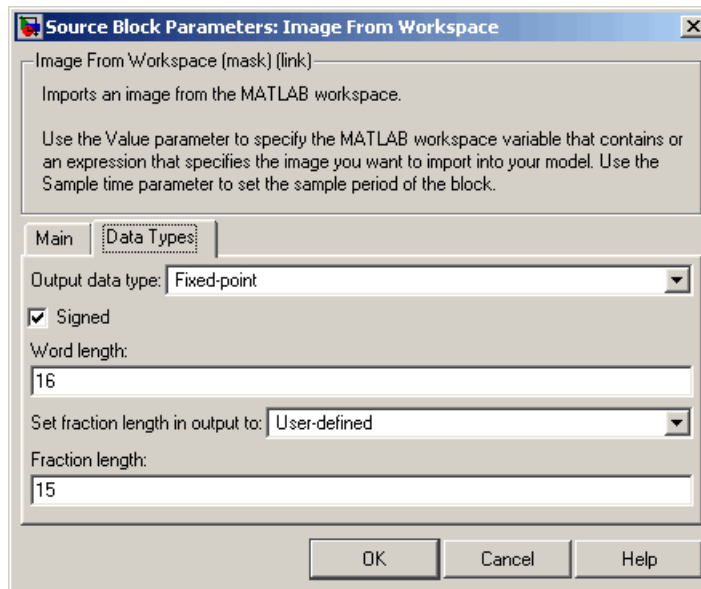
Specify how the block outputs a color video signal. If you select `One multidimensional signal`, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select `Separate color signals`, additional ports appear on the block. Each port outputs one 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

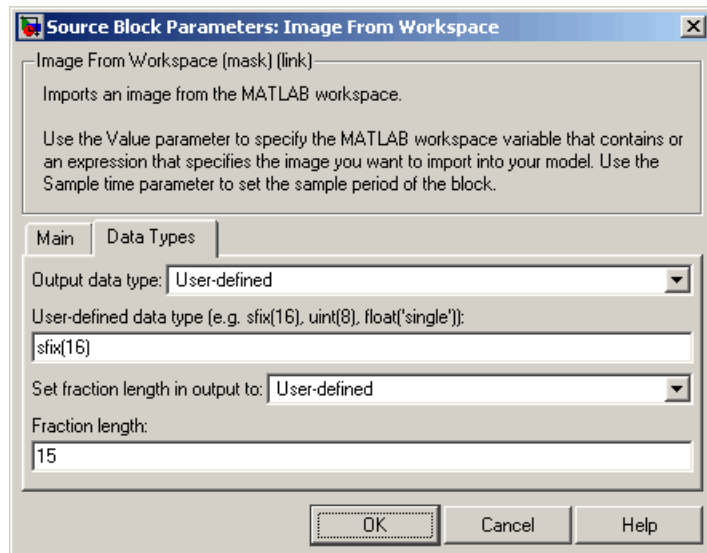
Specify the scaling of the fixed-point output by either of the following two methods:

- 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 `sfixed`, `ufixed`, `sint`, `uint`, `sfrac`,

Image From Workspace

and `ufrac` functions from Simulink Fixed Point. This parameter is only visible when you select User-defined for the **Output data type** parameter.

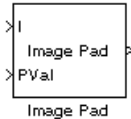
See Also

From Multimedia File	Video and Image Processing Blockset
To Video Display	Video and Image Processing Blockset
Video From Workspace	Video and Image Processing Blockset
Video Viewer	Video and Image Processing Blockset
<code>im2double</code>	Image Processing Toolbox
<code>im2uint8</code>	Image Processing Toolbox

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 2-403 — Demonstrates the block's behavior when the **Method** parameter is set to Constant.
- “Example 2” on page 2-404— Demonstrates the block's behavior when the **Method** parameter is set to Replicate.

- “Example 3” on page 2-405— Demonstrates the block’s behavior when the **Method** parameter is set to Symmetric.
- “Example 4” on page 2-406— 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:

$$\begin{array}{c}
 \left[\begin{array}{cccc|cccc|cccc|cccc}
 a_{11} & a_{12} & a_{12} & a_{11} & a_{10} & a_{10} & a_{11} & a_{12} & a_{12} & a_{11} & a_{10} & a_{10} & a_{11} & a_{12} \\
 a_{01} & a_{02} & a_{02} & a_{01} & a_{00} & a_{00} & a_{01} & a_{02} & a_{02} & a_{01} & a_{00} & a_{00} & a_{01} & a_{02} \\
 a_{01} & a_{02} & a_{02} & a_{01} & a_{00} & a_{00} & a_{01} & a_{02} & a_{02} & a_{01} & a_{00} & a_{00} & a_{01} & a_{02} \\
 a_{11} & a_{12} & a_{12} & a_{11} & a_{01} & a_{10} & a_{11} & a_{12} & a_{12} & a_{11} & a_{10} & a_{10} & a_{11} & a_{12} \\
 a_{21} & a_{22} & a_{22} & a_{21} & a_{20} & a_{20} & a_{21} & a_{22} & a_{22} & a_{21} & a_{20} & a_{20} & a_{21} & a_{22} \\
 a_{21} & a_{22} & a_{22} & a_{21} & a_{20} & a_{20} & a_{21} & a_{22} & a_{22} & a_{21} & a_{20} & a_{20} & a_{21} & a_{22} \\
 a_{11} & a_{12} & a_{12} & a_{11} & a_{01} & a_{01} & a_{11} & a_{12} & a_{12} & a_{11} & a_{10} & a_{10} & a_{11} & a_{12}
 \end{array} \right] \text{Input matrix}
 \end{array}$$

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:

$$\begin{bmatrix}
 a_{00} & a_{01} & a_{02} & a_{00} & a_{01} & a_{02} & a_{00} & a_{01} & a_{02} \\
 a_{10} & a_{11} & a_{12} & a_{10} & a_{11} & a_{12} & a_{10} & a_{11} & a_{12} \\
 a_{20} & a_{21} & a_{22} & a_{20} & a_{21} & a_{22} & a_{20} & a_{21} & a_{22} \\
 \hline
 a_{00} & a_{01} & a_{02} & a_{00} & a_{01} & a_{02} & a_{00} & a_{01} & a_{02} \\
 a_{10} & a_{11} & a_{12} & a_{10} & a_{11} & a_{12} & a_{10} & a_{11} & a_{12} \\
 a_{20} & a_{21} & a_{22} & a_{20} & a_{21} & a_{22} & a_{20} & a_{21} & a_{22} \\
 \hline
 a_{00} & a_{01} & a_{02} & a_{00} & a_{01} & a_{02} & a_{00} & a_{01} & a_{02} \\
 a_{10} & a_{11} & a_{12} & a_{10} & a_{11} & a_{12} & a_{10} & a_{11} & a_{12} \\
 a_{20} & a_{21} & a_{22} & a_{20} & a_{21} & a_{22} & a_{20} & a_{21} & a_{22}
 \end{bmatrix}$$

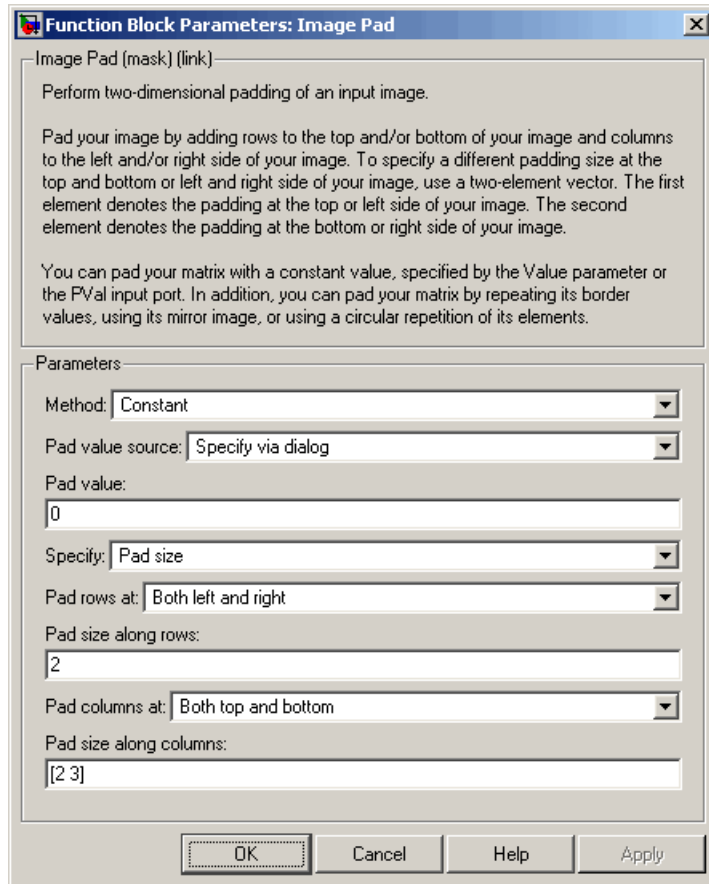
Input matrix

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 library, an open-source font engine, to produce stylized text bitmaps. To learn more about the FreeType Project, visit <http://www.freetype.org/>.



Port	Input/Output	Supported Data Types	Complex Values Supported
Input / 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. (Word length must be less than or equal to 32.)• 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. Outputs from the R, G, or B ports have the same dimensions and data type.	Same as Input port	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Select	Zero-based index value that indicates which text string to display.	<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 	No

Insert Text

Port	Input/Output	Supported Data Types	Complex Values Supported
Variables	Vector or matrix whose values are used to replace ANSI C printf-style format specifications.	<p>The data types supported by this port depend on the conversion specification that you are using in the Text parameter.</p> <p><i>%d</i>, <i>%i</i>, and <i>%u</i>:</p> <ul style="list-style-type: none">• 8-, 16-, 32-bit signed integer• 8-, 16-, 32-bit unsigned integer <p><i>%c</i> and <i>%s</i>:</p> <ul style="list-style-type: none">• 8-bit unsigned integer <p><i>%f</i>:</p> <ul style="list-style-type: none">• Double-precision floating point• Single-precision floating point <p><i>%o</i>, <i>%x</i>, <i>%X</i>, <i>%e</i>, <i>%E</i>, <i>%g</i>, and <i>%G</i>:</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	No

Port	Input/Output	Supported Data Types	Complex Values Supported
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.	<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 	No

Insert Text

Port	Input/Output	Supported Data Types	Complex Values Supported
Color	<p>Intensity input — Scalar value that is used for all strings or 1-by-N vector of intensity values whose length is equal to the number of strings.</p> <p>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.</p>	Same as Input port (The input to this port must be the same data type as the input to the Input port.)	No
Opacity	Scalar value that is used for all strings or vector of opacity values whose length is equal to the number of strings.	<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.) 	No

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', or a cell array of strings, such as {'Figure1', 'Figure2'}.

If, for the **Text** parameter, you enter a cell array of strings, the Select port appears on the block. The input to this port must be a scalar value that indicates which text string to display, 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. The following table summarizes which conversion specifications are supported:

Text Parameter Supported Conversion Specifications

Supported specifications	Support for multiple instances of the same specification?	Support for mixed specifications?
<code>%d</code> , <code>%i</code> , <code>%u</code> , <code>%c</code> , <code>%f</code> , <code>%o</code> , <code>%x</code> , <code>%X</code> , <code>%e</code> , <code>%E</code> , <code>%g</code> , and <code>%G</code>	Yes	No
<code>%s</code>	No	No

Use the **Location source** parameter to indicate where to specify the text location. If you select Specify via dialog, the **Location [row column]** parameter appears on the dialog box. If you select 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 want to insert. 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.

Insert Text

Location Parameter Text String Insertion

Parameter	One Text String	Multiple Text Strings
Location [row column] parameter setting or the input to the Location port	Two-element vector of the form [row column] that indicates the top-left corner of the text bounding box.	2-by-N matrix, where N 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 each string, e.g. [[row1 column1]' [row2 column2]'].

Use the **Color value source** parameter to indicate where to specify the text color.

- If you select Specify via dialog, the **Color value** parameter appears on the dialog box.
- If you select 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.

- If you select Specify via dialog, the **Opacity** parameter appears on the dialog box.
- If you select 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.

Insert Text

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.

- 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.

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 smoothes 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 Video and Image Processing Blockset 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:

- 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:

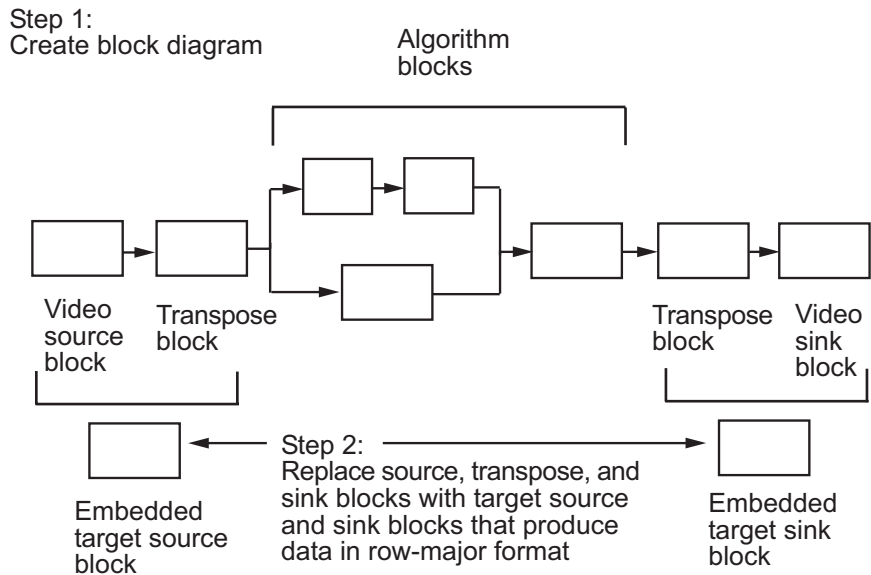
- When you select this check box, the first two signal dimensions of the Insert Text block's input are swapped.
- All Video and Image Processing Blockset blocks can be used to process data that is in the row-major format, but you need to know the image dimensions when you develop your algorithms.

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.

Insert Text



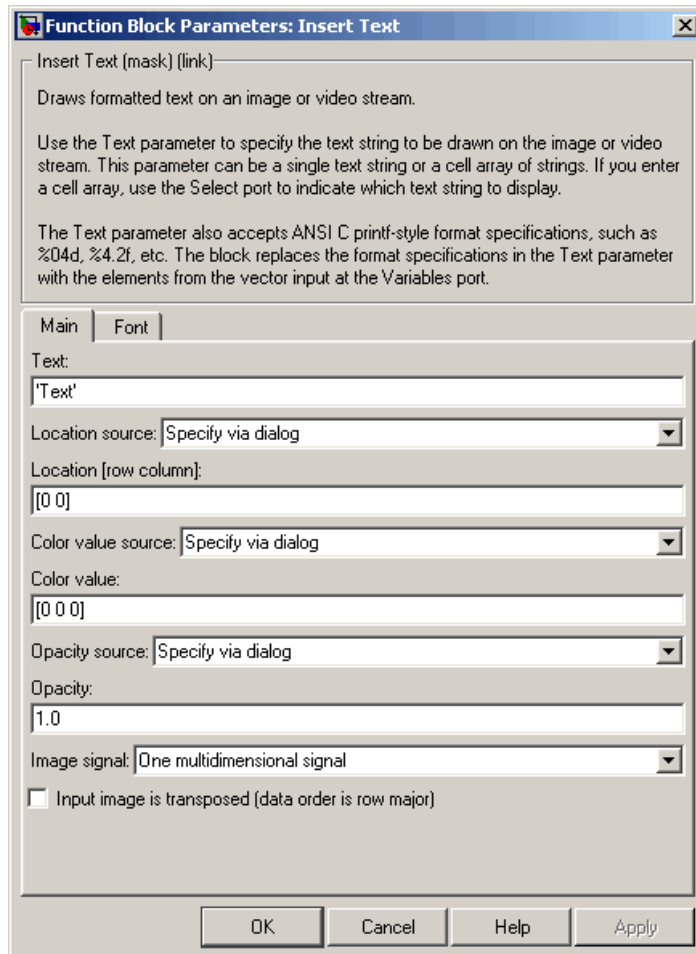
See the DM642 EVM Video ADC and DM642 EVM Video DAC reference pages in the *Target for TI C6000 User's Guide* for more information about data order in embedded targets.

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*.

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.

Insert Text

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.

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.

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.

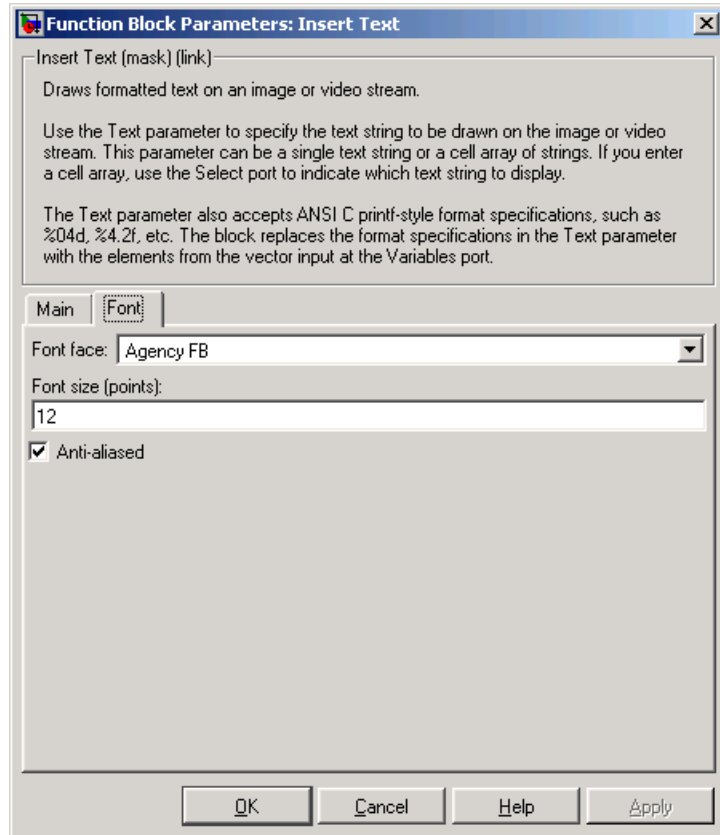
Image signal

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.

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.

See Also

[Draw Shapes](#)

[Video and Image Processing Blockset](#)

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 documentation.

Label

Purpose Label connected components in binary images

Library Morphological Operations

Description

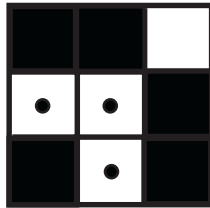


The Label block labels the objects in a binary image, BW, where the background is represented by pixels equal to 0 (black) and objects are represented by pixels equal to 1 (white). At the Label port, the block outputs a label matrix that is the same size as the input matrix. In the label matrix, pixels 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.

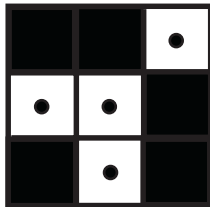
Port	Input/Output	Supported Data Types	Complex Values Supported
BW	Vector or matrix that represents a binary image	Boolean	No
Label	Label matrix	<ul style="list-style-type: none">8-, 16-, and 32-bit unsigned integer	No
Count	Scalar that represents the number of labeled objects	Same as Label port	No

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.

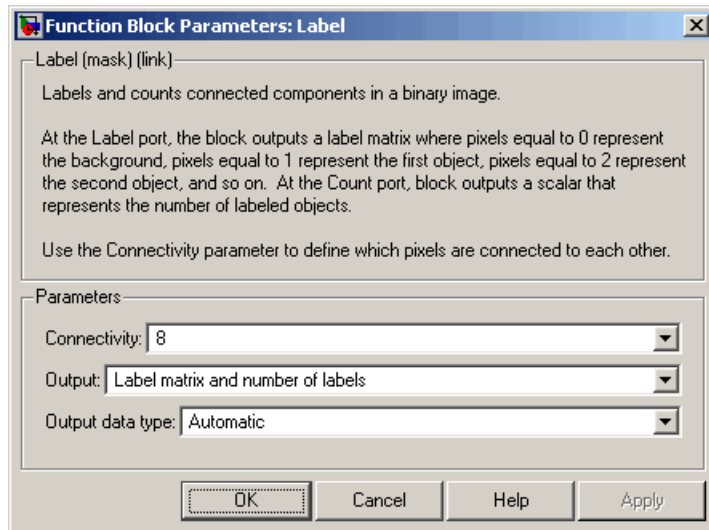
Use the **Output data type** parameter to set the data type of the outputs at the `Label` and `Count` ports. If you select `Automatic`, the block calculates the maximum number of objects that can fit inside the image based on the image size and the connectivity you specified. Based on this calculation, 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
Closing	Video and Image Processing Blockset
Dilation	Video and Image Processing Blockset
Erosion	Video and Image Processing Blockset
Opening	Video and Image Processing Blockset
Top-hat	Video and Image Processing Blockset
bwlabel	Image Processing Toolbox
bwlabeln	Image Processing Toolbox

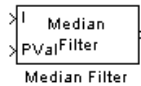
Maximum

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 documentation.

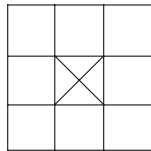
Purpose Perform 2-D median filtering

Library Filtering / Analysis & Enhancement

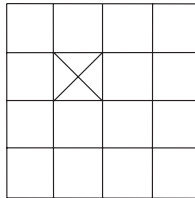
Description



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.



If the neighborhood does not have an exact center, the block has a bias toward the upper-left corner and places the median value there, as illustrated in the following figure.



The block pads the edge of the input image so, pixels within $[M/2 \ N/2]$ of the edges may appear distorted. Because the median value is less sensitive than the mean to extreme values, the Median Filter block can remove salt and pepper noise from an image without significantly reducing the sharpness of the image.

Median Filter

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.

Use the **Neighborhood size** parameter to specify the size of the neighborhood over which the block computes the median. You can enter a scalar value that represents the number of rows and columns in a square matrix or a vector that represents the number of rows and columns in a rectangular matrix.

Use the **Output size** parameter to specify the size of the output matrix. If you select Same as input port I, the block outputs an intensity image that is the same size as the image input to the I port. If you select Valid, the block only computes the median where the neighborhood fits entirely within the input image, so no padding is required. The dimensions of the output image are

$$\begin{aligned}\text{output rows} &= \text{input rows} - \text{neighborhood rows} + 1 \\ \text{output columns} &= \text{input columns} - \text{neighborhood columns} + 1\end{aligned}$$

If, for the **Output size** parameter, you choose Same as input port I, the **Padding options** parameter appear in the dialog box. 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, for the **Pad value source** parameter, you select Input port, the Val port appears on the block. Use this port to specify the constant value with which to pad your matrix.

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, product output, and output data types as discussed in “Dialog Box” on page 2-437. 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

Median Filter

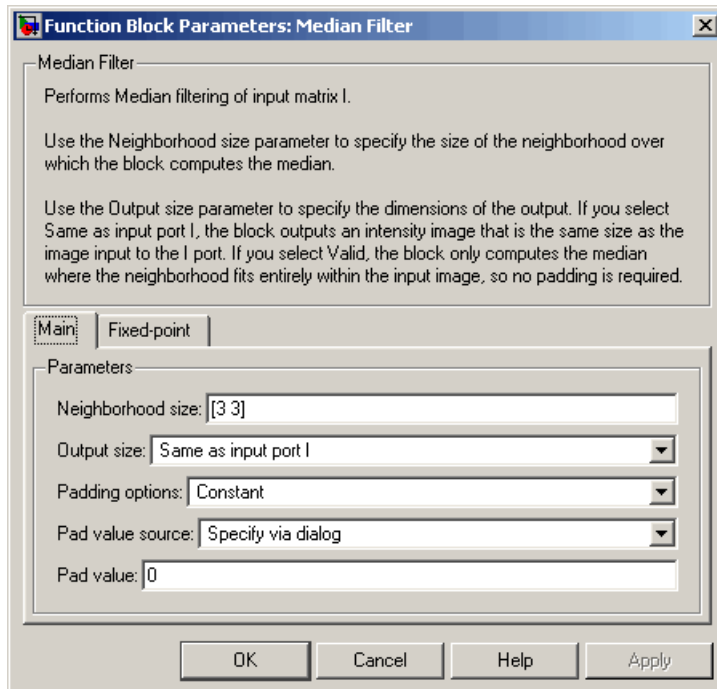
The accumulator and output data types and scalings are used for fixed-point signals when M is even. The result of the sum performed while calculating the average of the two central rows of the input matrix is stored in the accumulator data type and scaling. The total result of the average is then put 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 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. You can enter a scalar value that represents the number of rows and columns in a square matrix or a vector that represents the number of rows and columns in a rectangular matrix.

Output size

This parameter controls the size of the output. If you choose Same as input port I, the output has the same dimensions as the input to port I. If you choose Valid, output rows = input

Median Filter

rows - neighborhood rows + 1 and output columns = input columns - 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. 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 is visible if, for the **Output size** parameter, you select **Same as input port I**.

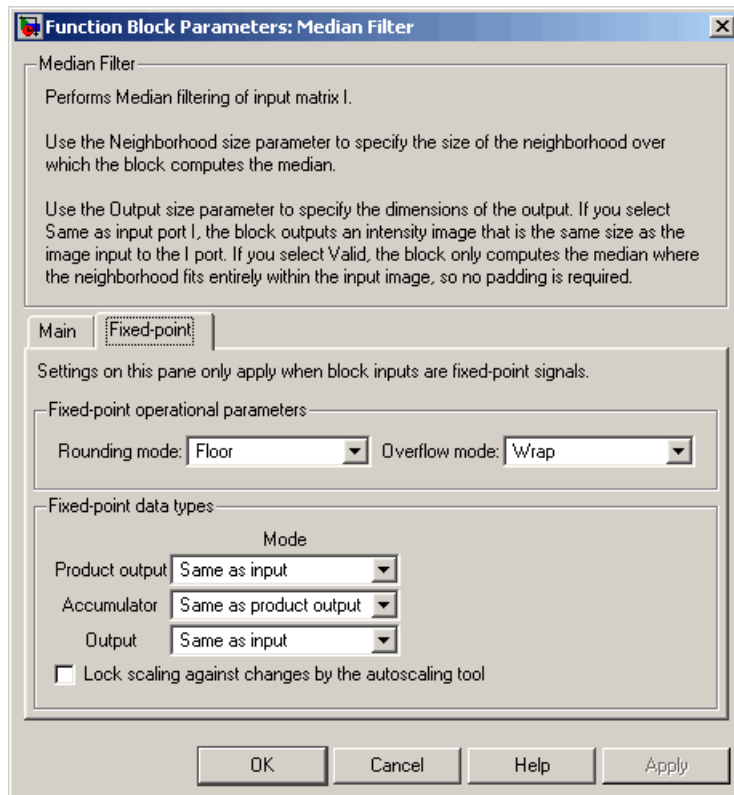
Pad value source

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 **Val** port. This parameter is visible if, for the **Padding options** parameter, you select **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 **Fixed-point** pane of the Median Filter dialog box appears as follows. The parameters on this dialog box are only visible when the dimensions of the neighborhood are even.



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 2-435 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.

- 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.

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

Minimum

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 documentation.

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**

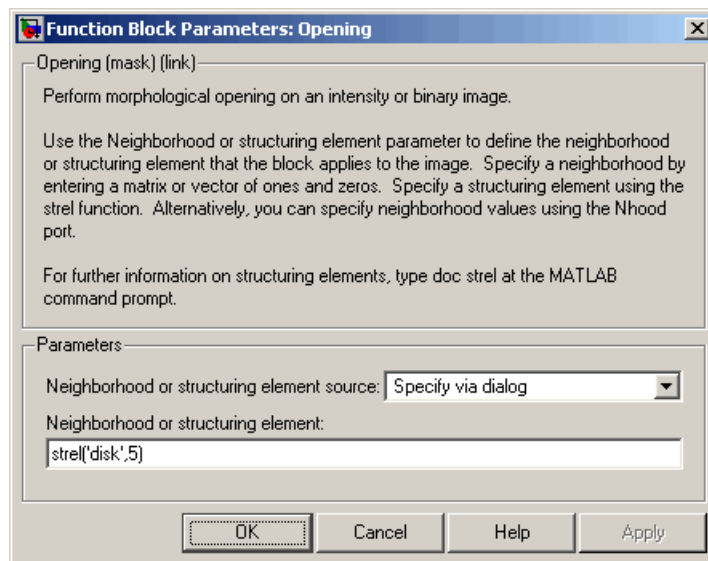
Opening

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 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 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

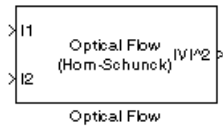
Bottom-hat	Video and Image Processing Blockset
Closing	Video and Image Processing Blockset
Dilation	Video and Image Processing Blockset
Erosion	Video and Image Processing Blockset
Label	Video and Image Processing Blockset
Top-hat	Video and Image Processing Blockset
<code>imopen</code>	Image Processing Toolbox
<code>strel</code>	Image Processing Toolbox

Optical Flow

Purpose Estimate object velocities

Library Analysis & Enhancement

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:

Optical Flow

$$u_{x,y}^{k+1} = \bar{u}_{x,y}^{-k} - \frac{I_x [I_x \bar{u}_{x,y}^{-k} + I_y \bar{v}_{x,y}^{-k} + I_t]}{\alpha^2 + I_x^2 + I_y^2}$$
$$v_{x,y}^{k+1} = \bar{v}_{x,y}^{-k} - \frac{I_y [I_x \bar{u}_{x,y}^{-k} + I_y \bar{v}_{x,y}^{-k} + I_t]}{\alpha^2 + I_x^2 + I_y^2}$$

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 2-447.

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.

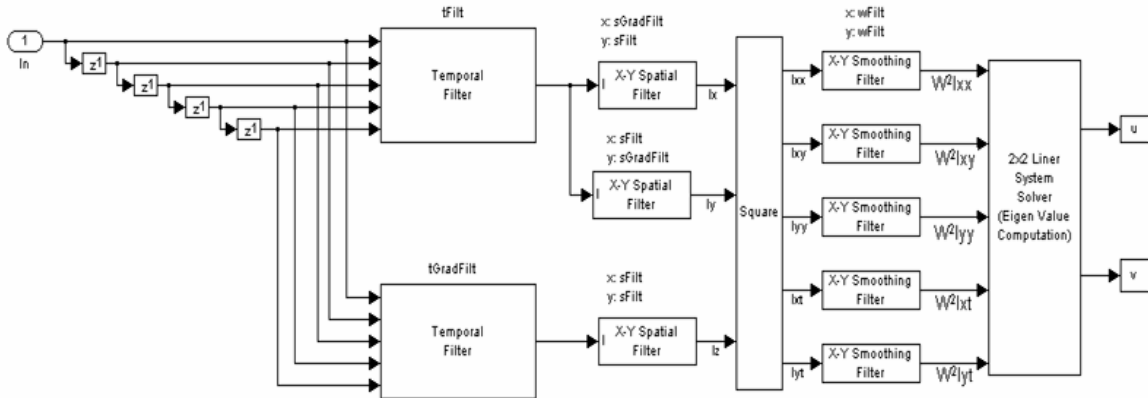
Select the **Discard normal flow estimates when constraint equation is ill-conditioned** check box if you want the block to set the motion vector to zero when the optical flow constraint equation is ill-conditioned. The block calculates these motion vectors on a pixel-by-pixel basis.

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 2-447.

Optical Flow

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.



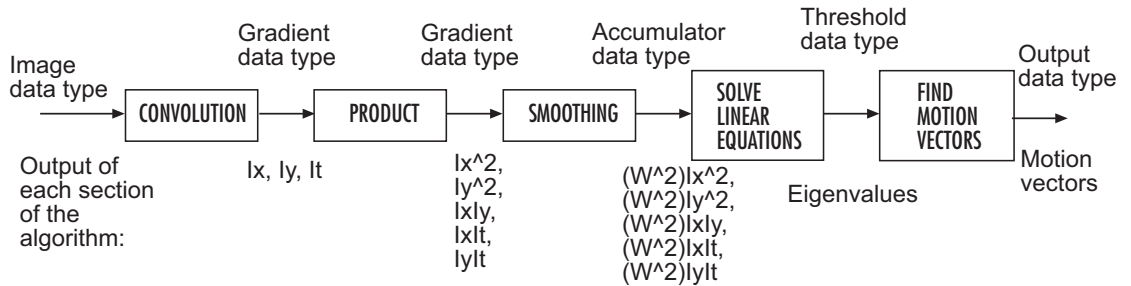
tFilt = Coefficients of Gaussian Filter
tGradFilt = Coefficients of the Derivative of a Gaussian Filter

sFilt = Coefficients of Gaussian Filter
sGradFilt = Coefficients of the Derivative of a Gaussian Filter

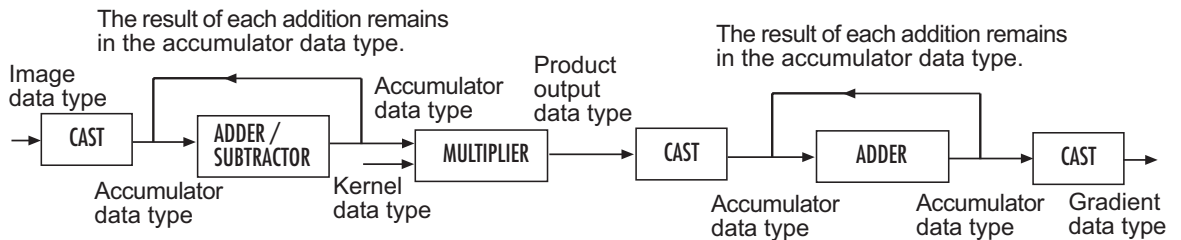
Fixed-Point Data Type Diagram

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.

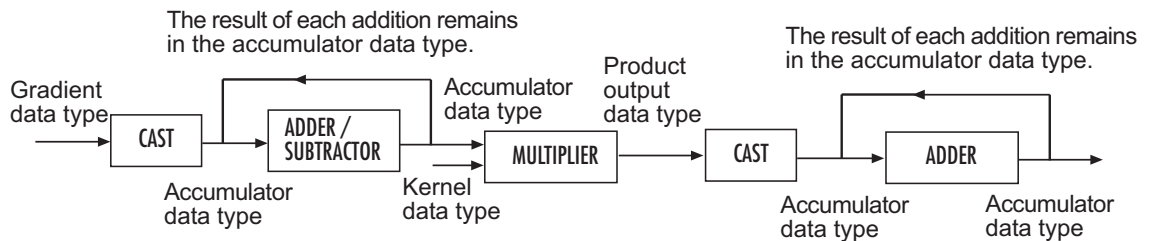
Data type diagram for Optical Flow block's overall algorithm



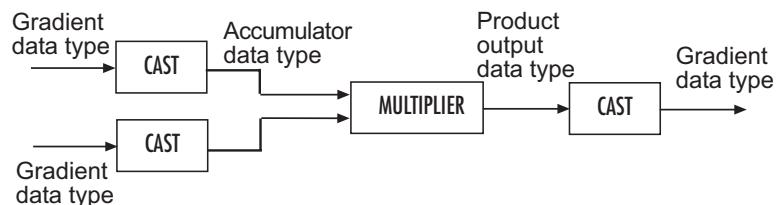
Data type diagram for convolution algorithm



Data type diagram for smoothing algorithm



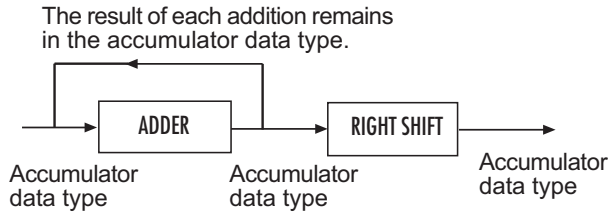
Data type diagram for product algorithm



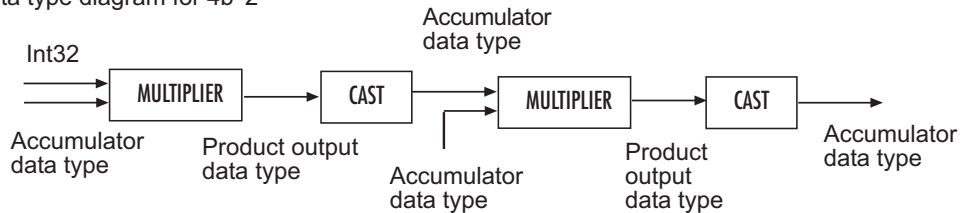
Optical Flow

Solving linear equations to compute eigenvalues
(see Step 4 in the Lucas-Kanade Method section for the eigenvalue equations)

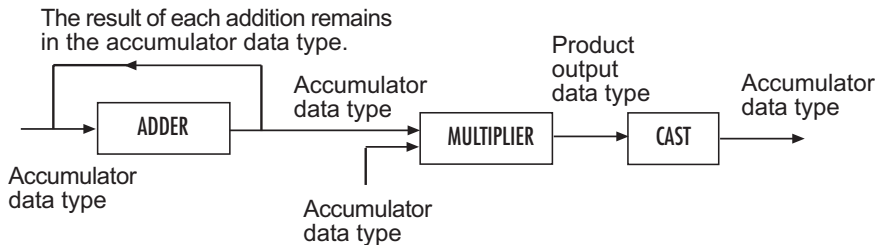
Data type diagram for P



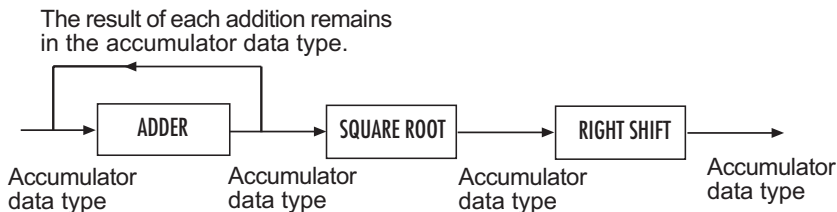
Data type diagram for $4b^2$



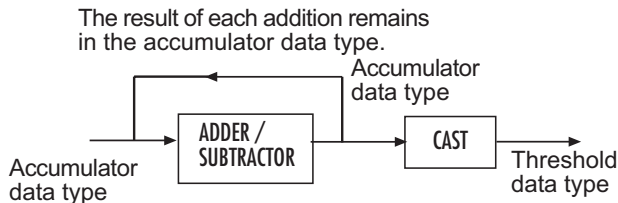
Data type diagram for $(a-c)^2$



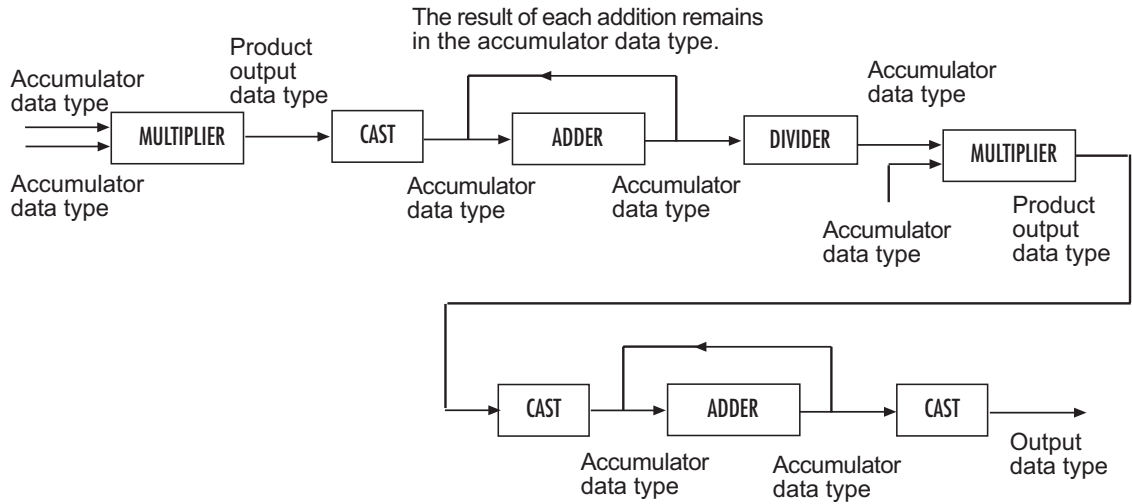
Data type diagram for Q



Data type diagram for eigenvalues



Data type diagram for finding the motion vectors algorithm

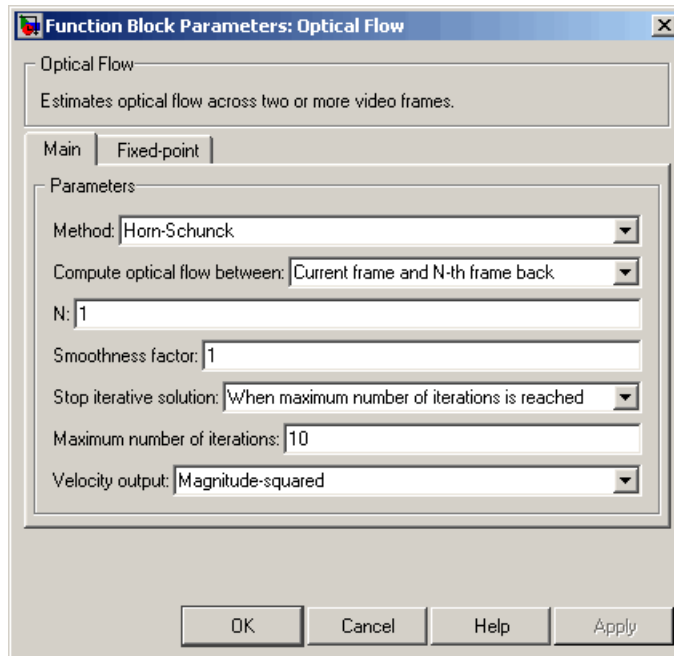


You can set the product output, accumulator, gradients, threshold, and output data types in the block mask.

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)

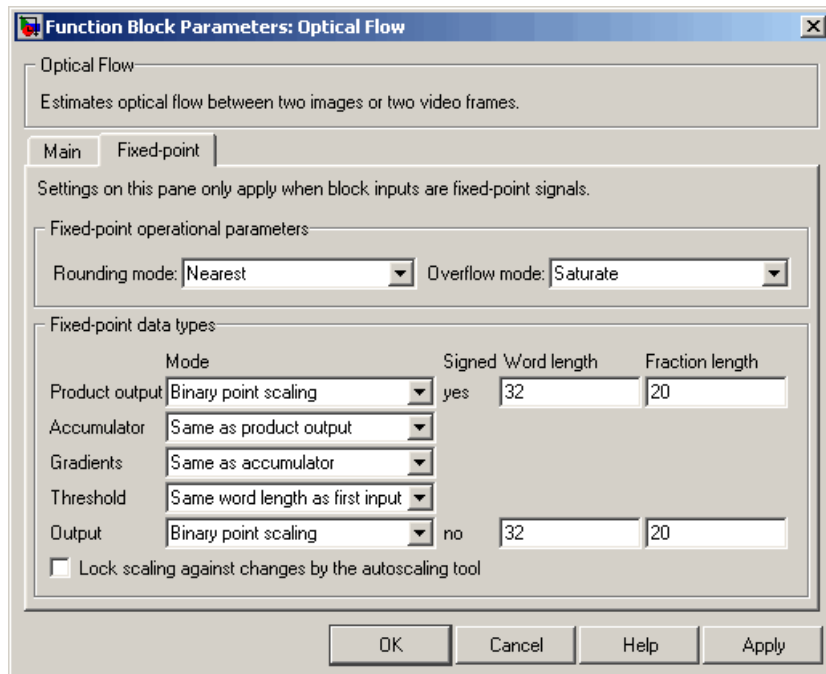
Select this check box if you want the block to output the image that corresponds to the motion vector being output by the block.

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 **Fixed-point** pane of the Optical Flow dialog box appears as shown in the following figure.



Rounding mode

Select the rounding mode for fixed-point operations.

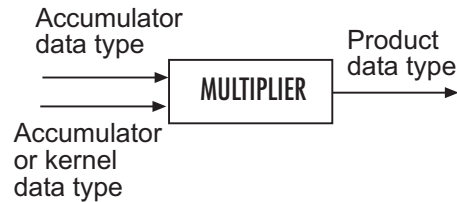
Optical Flow

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

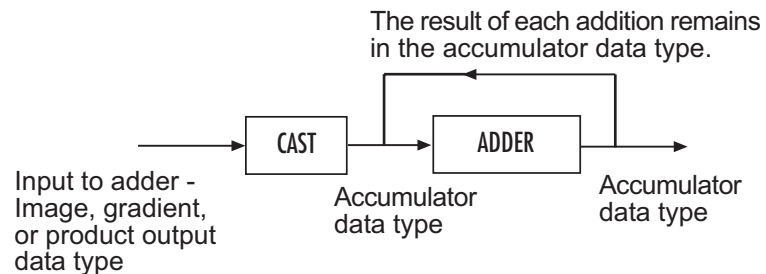
Use this parameter to specify how to designate the product output word and fraction lengths.



- 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 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.

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
Gaussian Pyramid	Video and Image Processing Blockset

Projective Transformation

Purpose Transform quadrilateral into another quadrilateral

Library Geometric Transformations

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.

Projective Transformation

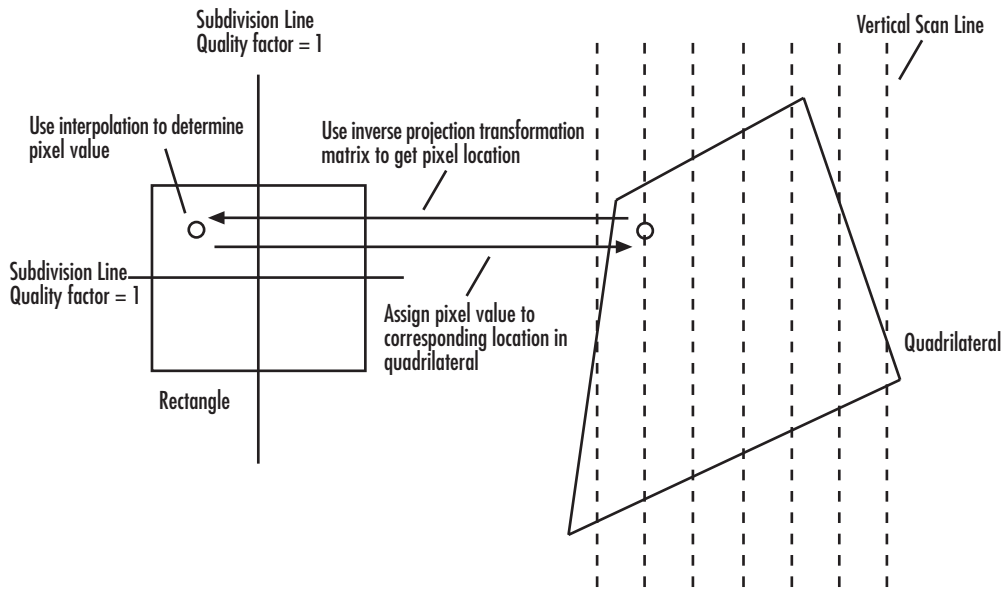
Rectangle to Quadrilateral Mode

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.

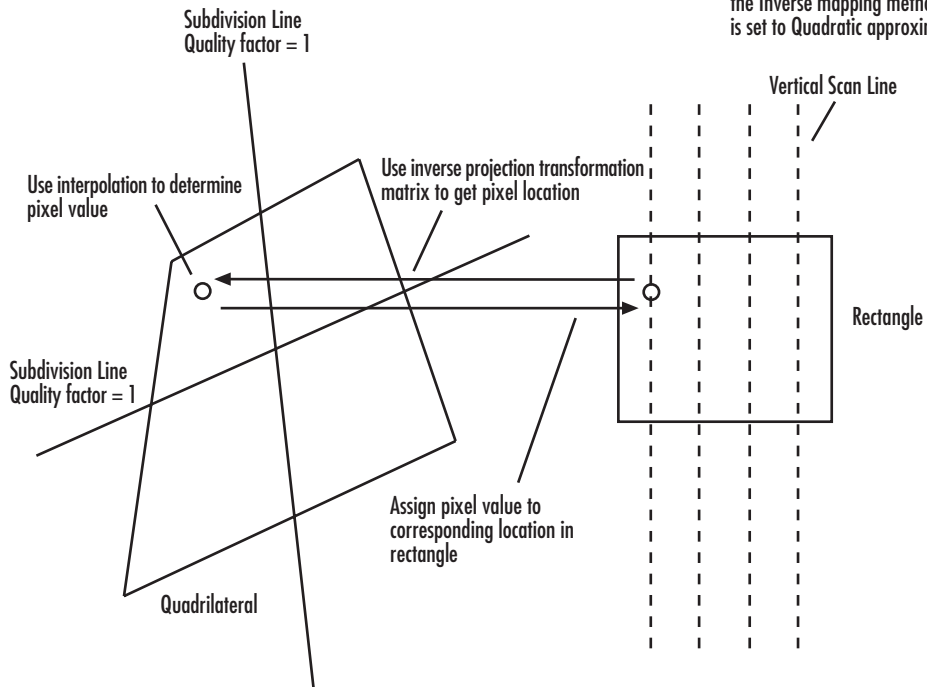
If you choose **Quadratic approximation**, the block divides the input shape using subdivision lines and the output shape using vertical scan lines. For the first pixel location on a scan line, the block uses an inverse projection transformation matrix to find the corresponding pixel location in the 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.



Projective Transformation

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.

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 2-468.

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

Projective Transformation

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 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 2-468.

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.

Projective Transformation

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.

Example

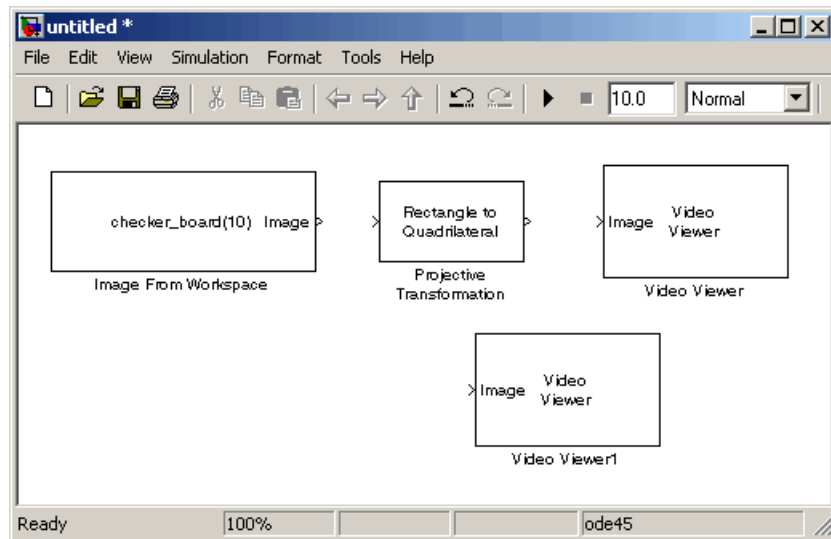
The following example shows you how to convert a rectangular image into a quadrilateral. It also shows you how to change the sizes of the input and output images.

- 1 Create a new Simulink model.
- 2 Click-and-drag the following blocks into your model.

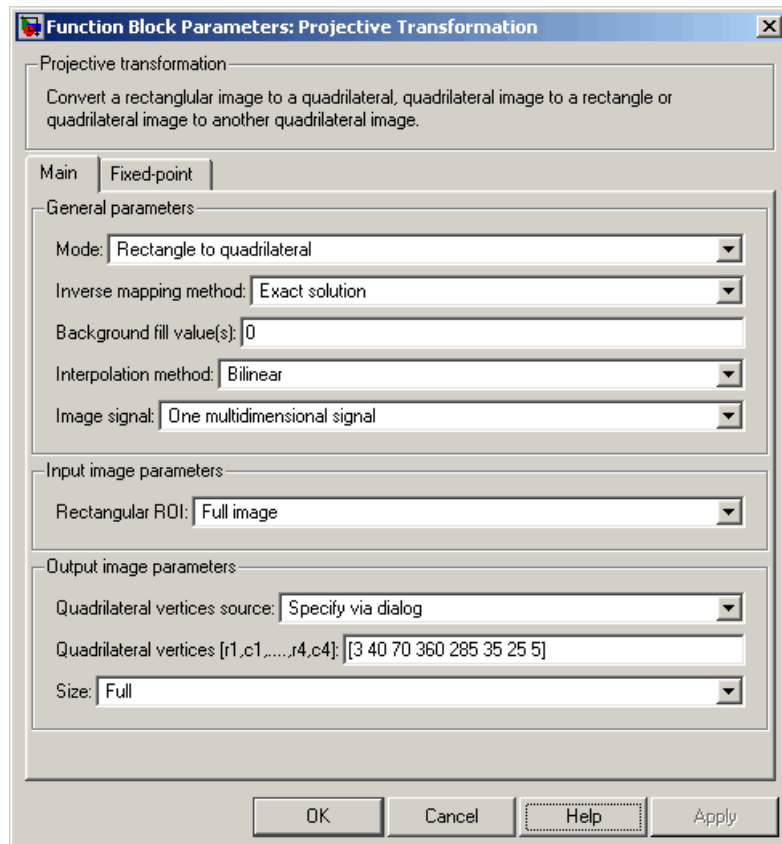
Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
Projective Transformation	Video and Image Processing Blockset / Geometric Transformations	1
Video Viewer	Video and Image Processing Blockset / Sinks	2

- 3 Place the blocks so your model looks similar to the following figure.

Projective Transformation



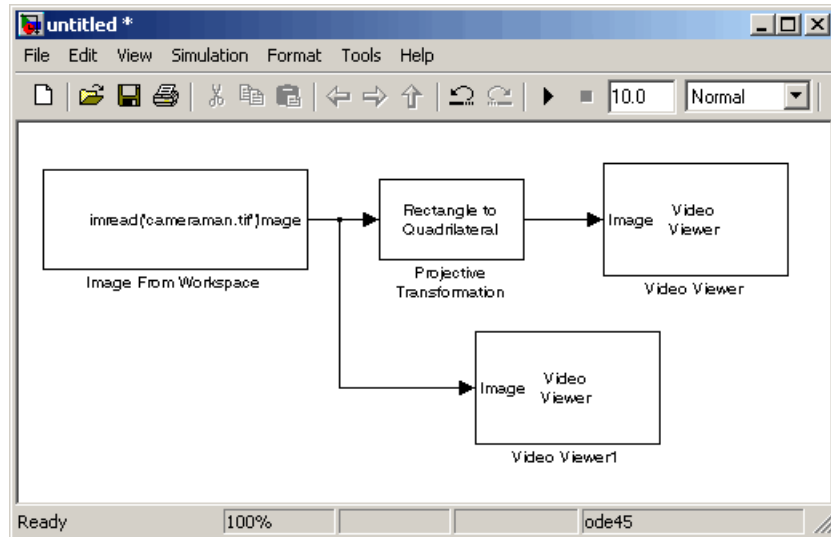
- 4 Use the Image From Workspace block to import an image into your model. Set the **Value** parameter to `imread('cameraman.tif')`.
- 5 Use the Projective Transformation block to transform your rectangular image into a quadrilateral. Set the **Quadrilateral vertices** `[r1,c1,...,r4,c4]` parameter to `[3 40 70 360 285 35 25 5]`.



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.

Projective Transformation

- 6 Use the Video Viewer1 and Video Viewer 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.



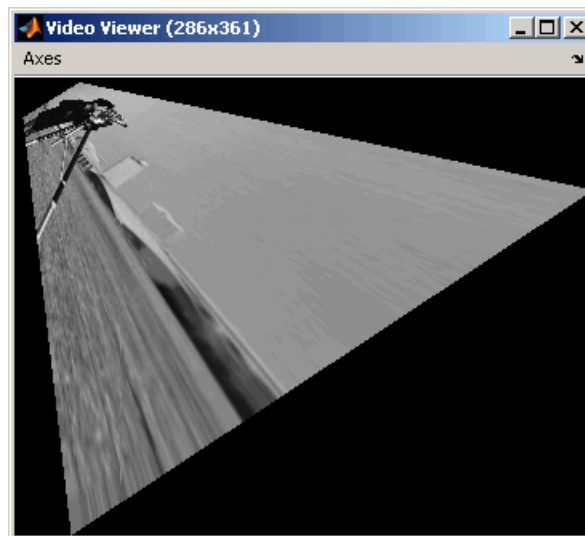
- 8 Run the model.

The original rectangular image appears in the Video Viewer1 window.

Projective Transformation

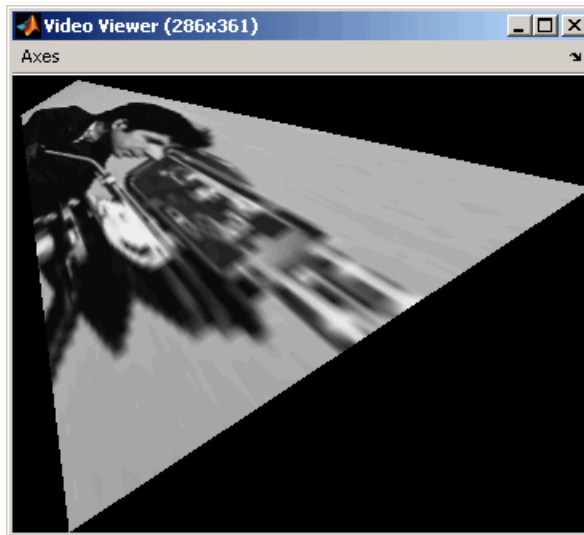


The quadrilateral image appears in the Video Viewer window.



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.

- 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.

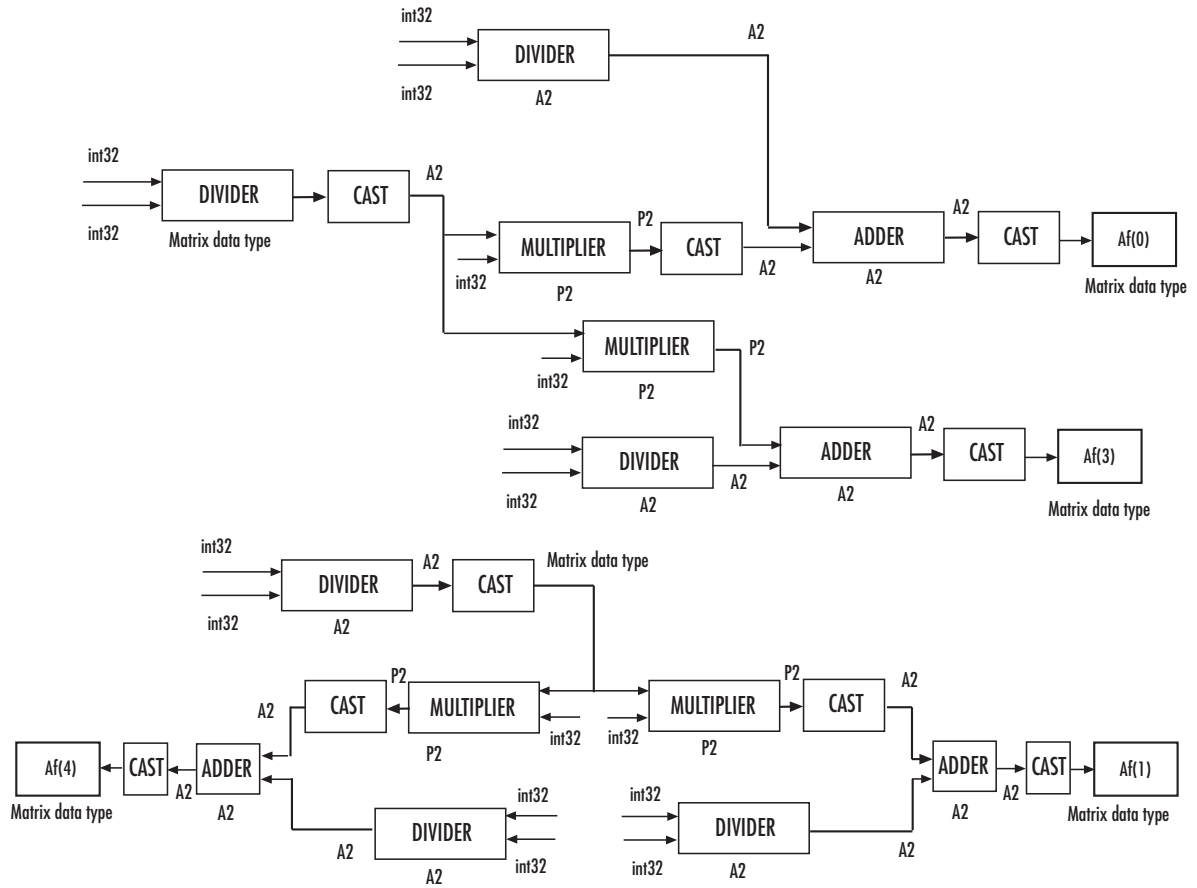


Fixed-Point Data Types

The following diagram shows the data types used in the Projective Transformation block for fixed-point signals:

Projective Transformation

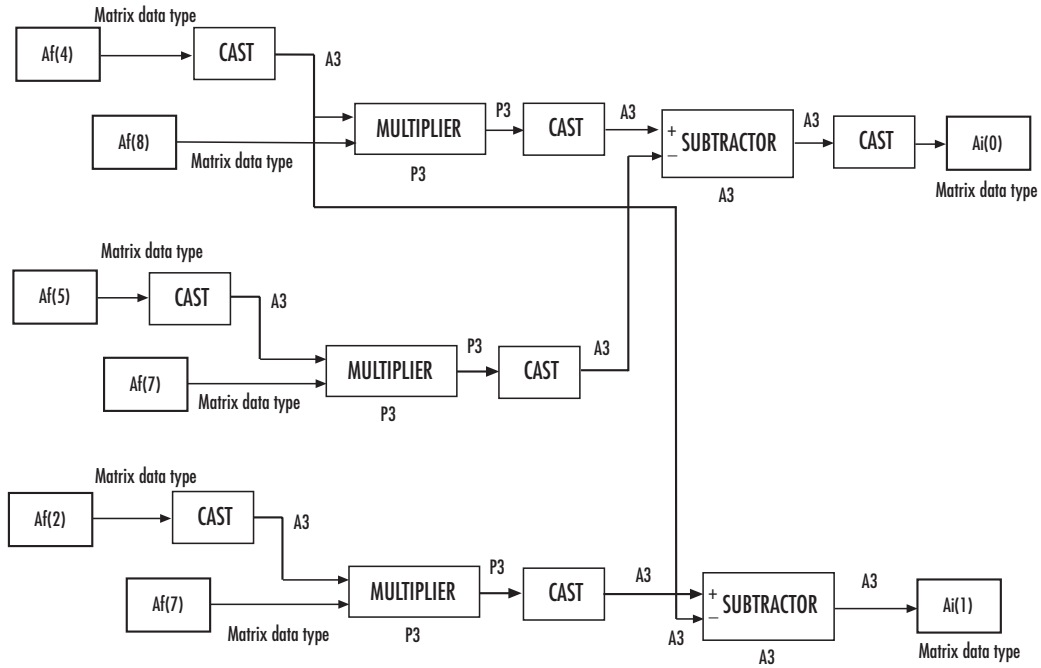
Calculate Forward Projective Transformation Matrix (Af)



The block uses a similar computation to calculate $Af(2)$, $Af(4)$, $Af(5)$, $Af(6)$, $Af(7)$, and $Af(8)$.

Projective Transformation

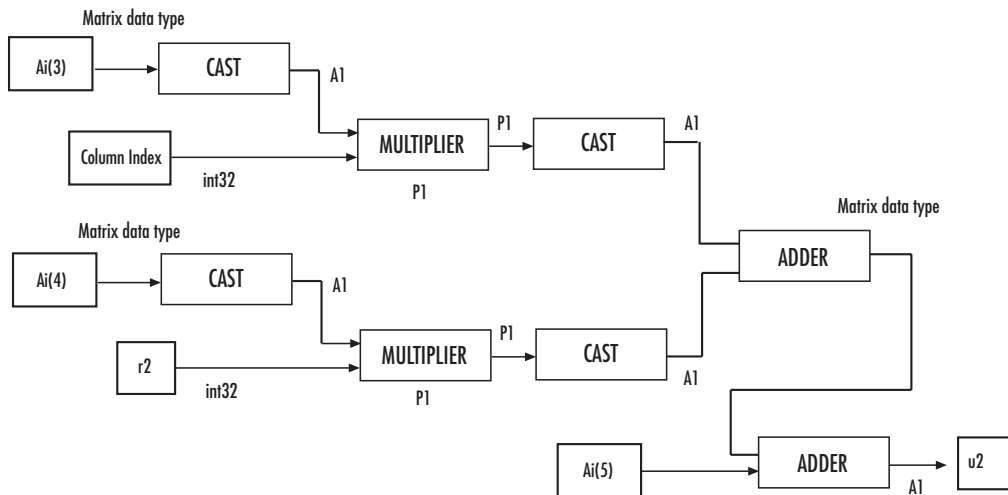
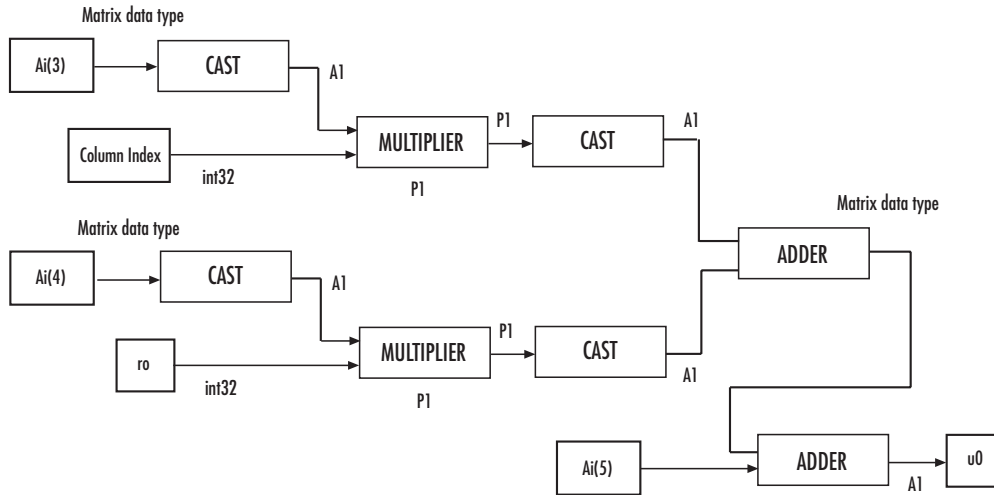
Calculate Inverse Projective Transformation Matrix (A_i)



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 $A_i(8)$.

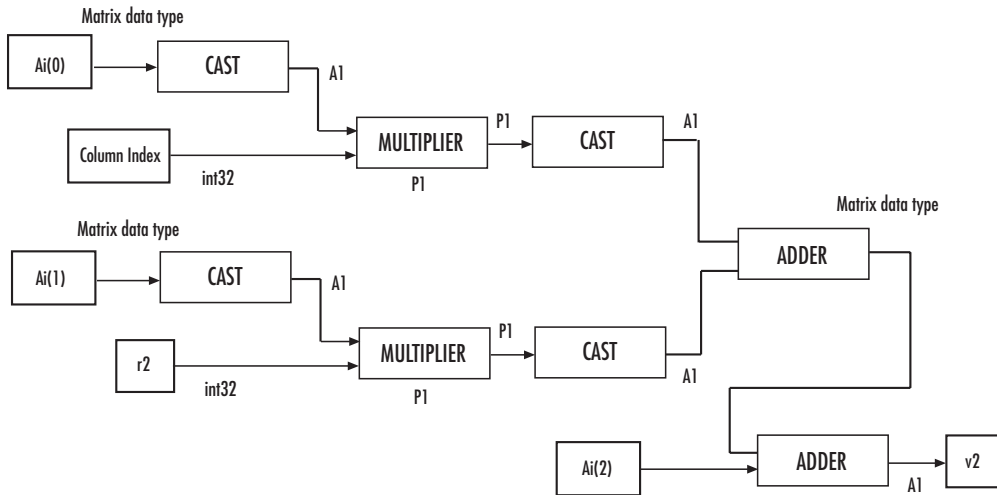
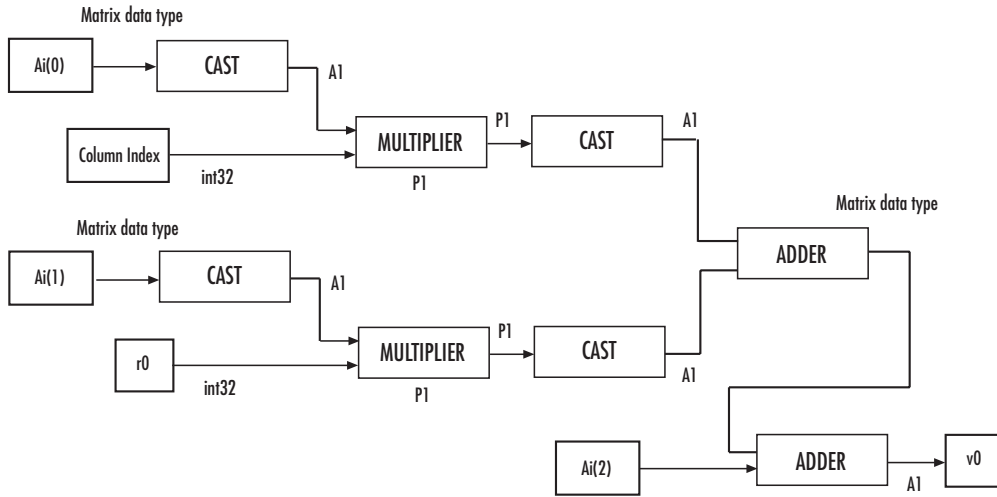
Projective Transformation

Compute Output Pixel



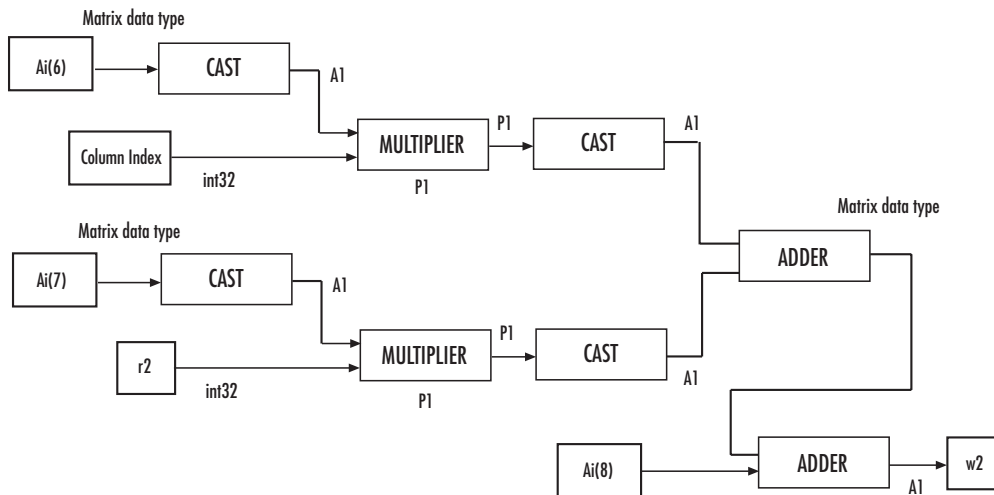
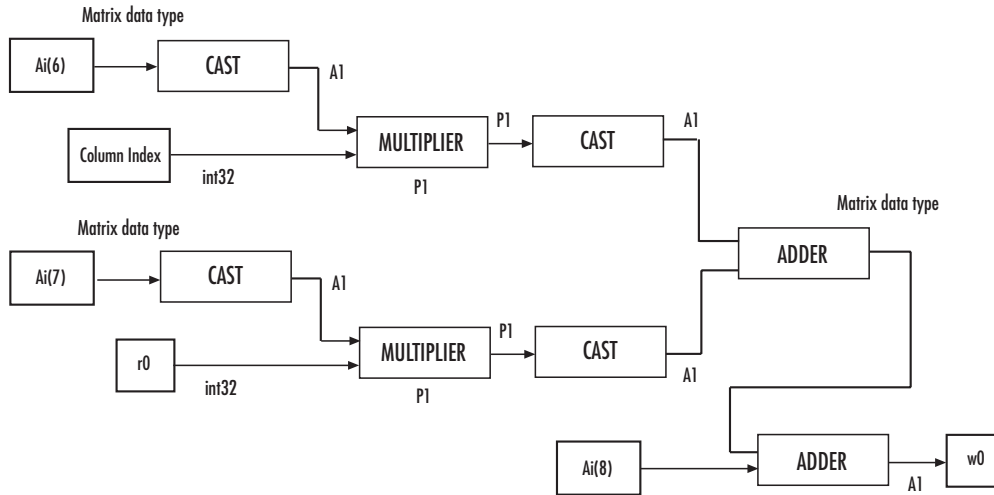
Projective Transformation

Compute Output Pixel (continued)



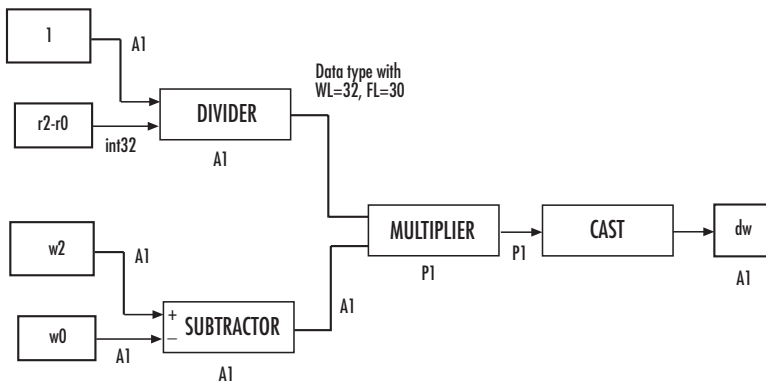
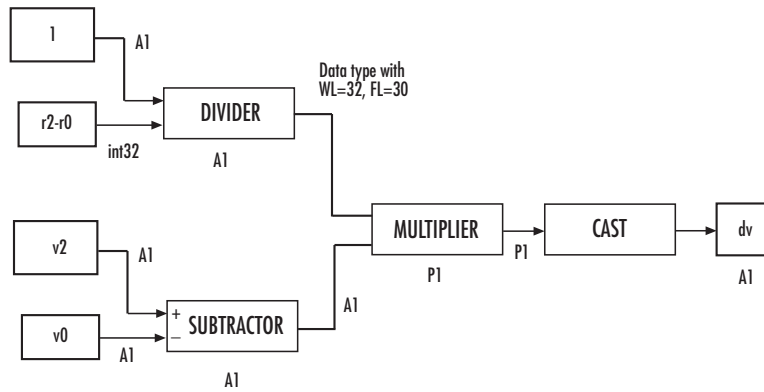
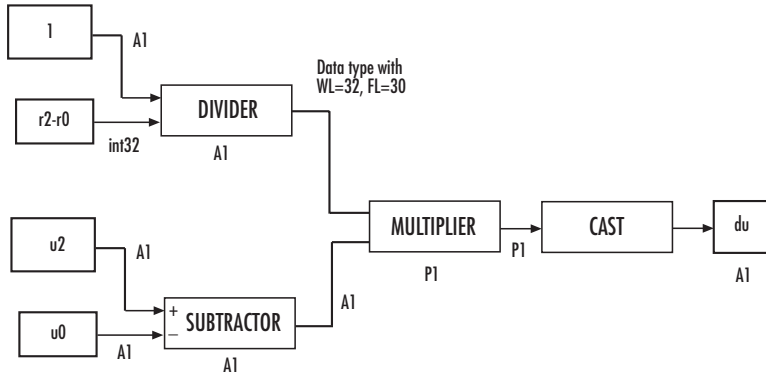
Projective Transformation

Compute Output Pixel (continued)



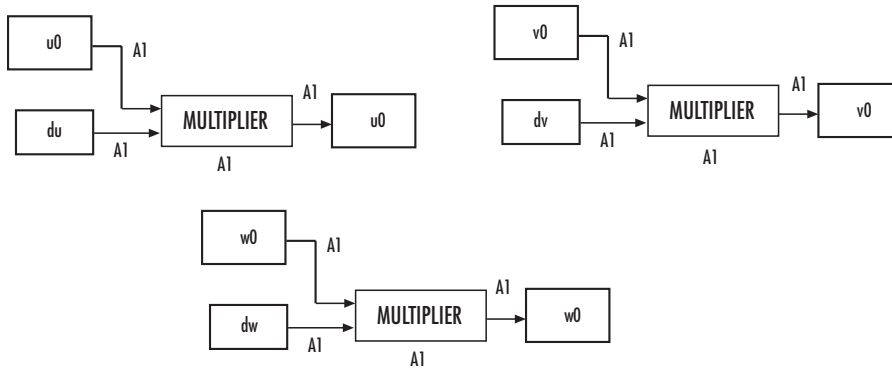
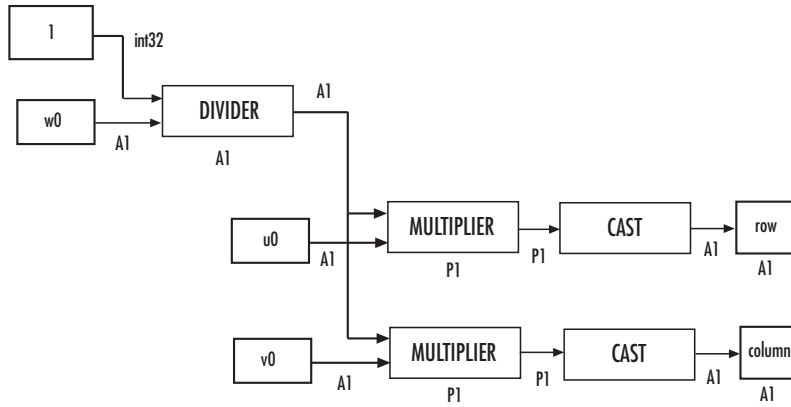
Projective Transformation

Calculate du, dv, dw



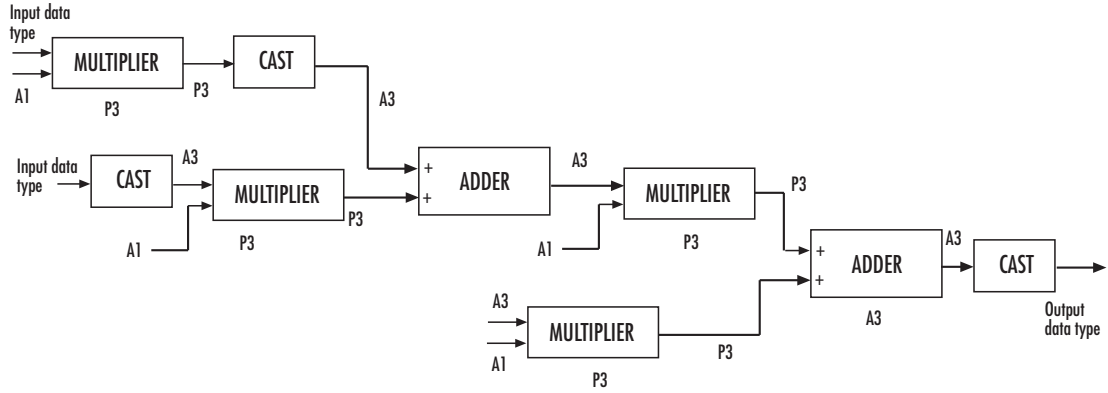
Projective Transformation

Calculate row and column indices of the input image



Projective Transformation

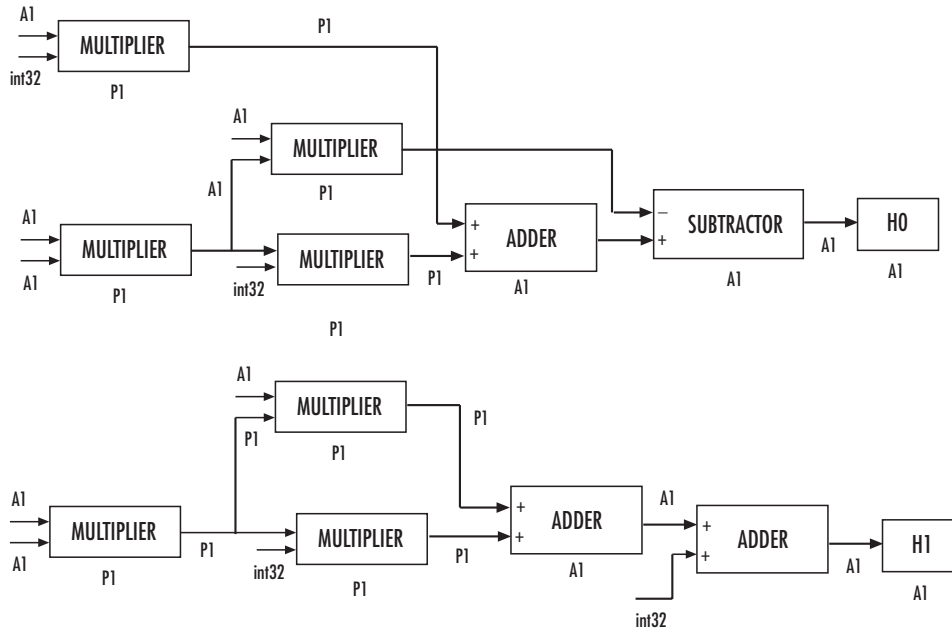
Bilinear Interpolation



Projective Transformation

Bicubic Interpolation

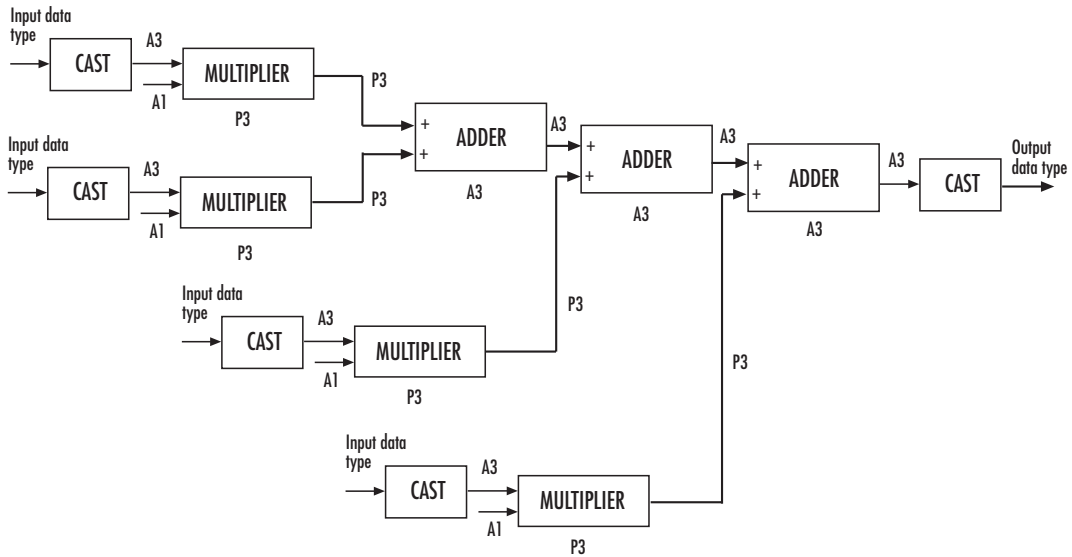
Calculate the bicubic interpolation coefficients, H0, H1, H2, and H3.



The block uses a similar computation to calculate H2 and H3.

Projective Transformation

Bicubic Interpolation (continued)

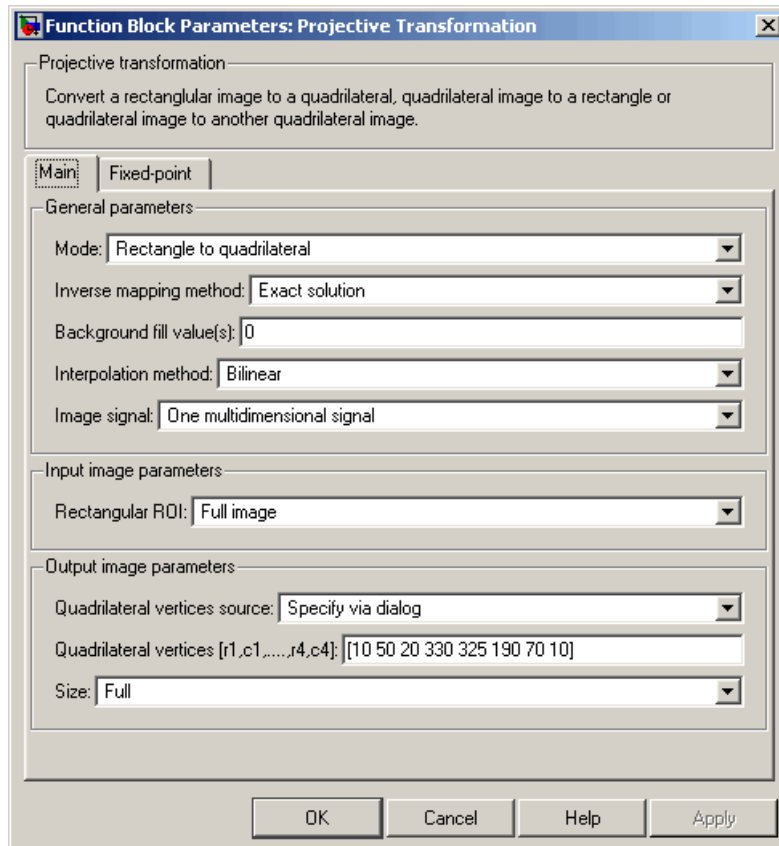


You can set the product, accumulator, matrix, and output data types in the block mask as discussed next.

Projective Transformation

Dialog Box

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



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`.

Projective Transformation

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 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

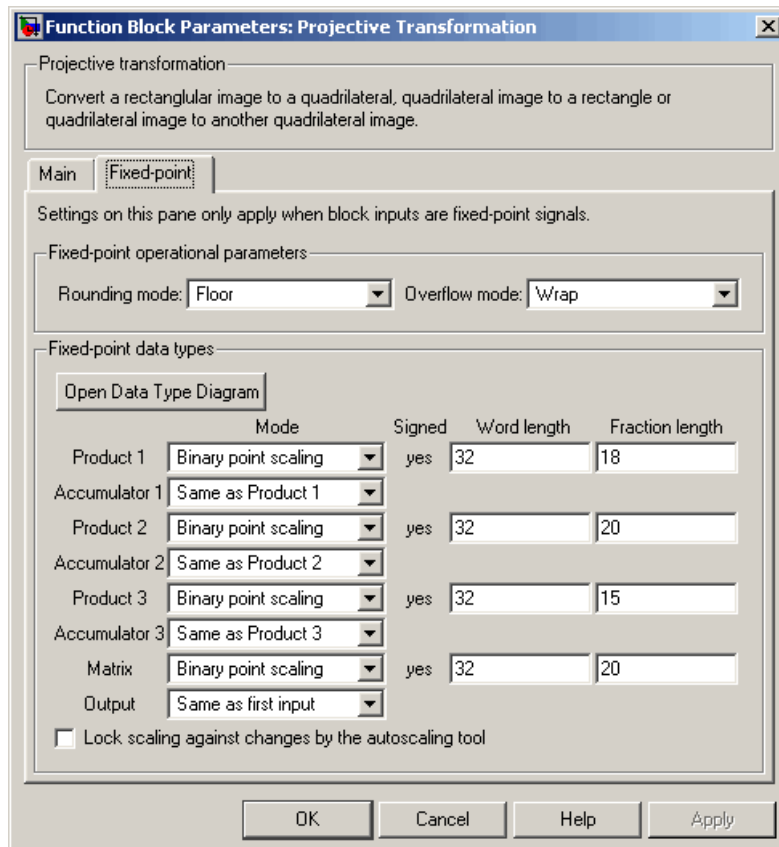
Specify how to define the rectangle size. Your choices are Specify via dialog and Input port.

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 **Fixed-point** pane of the Projective Transformation dialog box appears as follows.

Projective Transformation



The dialog box is titled "Function Block Parameters: Projective Transformation". It contains a description of the block's function, two tabs ("Main" and "Fixed-point"), and several configuration sections. The "Fixed-point" tab is active, showing settings for fixed-point signals. The "Fixed-point operational parameters" section includes "Rounding mode" (set to "Floor") and "Overflow mode" (set to "Wrap"). The "Fixed-point data types" section includes an "Open Data Type Diagram" button and a table for configuring data types. The table has columns for "Mode", "Signed", "Word length", and "Fraction length". It lists parameters for Product 1, Accumulator 1, Product 2, Accumulator 2, Product 3, Accumulator 3, Matrix, and Output. A checkbox for "Lock scaling against changes by the autoscaling tool" is at the bottom. Buttons for "OK", "Cancel", "Help", and "Apply" are at the bottom right.

Projective transformation—
Convert a rectangular image to a quadrilateral, quadrilateral image to a rectangle or quadrilateral image to another quadrilateral image.

Main Fixed-point

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—
Open Data Type Diagram

	Mode	Signed	Word length	Fraction length
Product 1	Binary point scaling	yes	32	18
Accumulator 1	Same as Product 1			
Product 2	Binary point scaling	yes	32	20
Accumulator 2	Same as Product 2			
Product 3	Binary point scaling	yes	32	15
Accumulator 3	Same as Product 3			
Matrix	Binary point scaling	yes	32	20
Output	Same as first input			

Lock scaling against changes by the autoscaling tool

OK Cancel Help Apply

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.

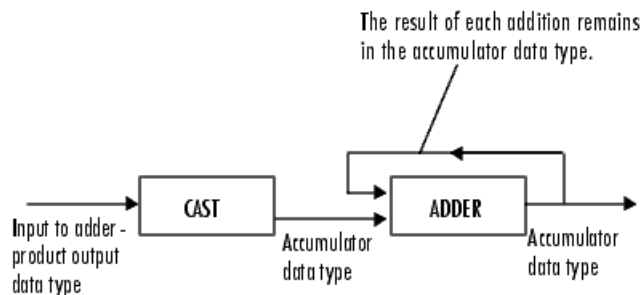
Product 1, 2, 3



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 Video and Image Processing Blockset is 0.

Accumulator 1, 2, 3, 4



Projective Transformation

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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 0.

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 Video and Image Processing Blockset is 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.

References

[1] Wolberg, George. *Digital Image Warping*. Washington: IEEE Computer Society Press, 1990.

See Also

Resize	Video and Image Processing Blockset
Rotate	Video and Image Processing Blockset
Shear	Video and Image Processing Blockset
Translate	Video and Image Processing Blockset

PSNR

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 image.

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.

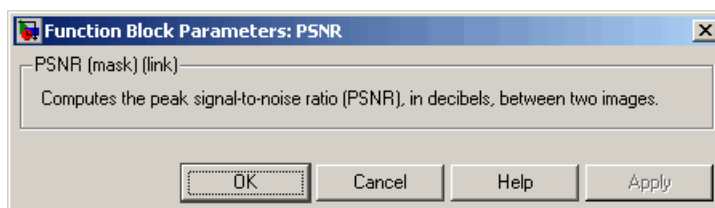
Note To compute the PSNR for color images, convert the images to a color space that separates intensity portion of the image from color information, such as YCbCr. Then perform the PSNR computation only on the intensity portion of the images.

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

Note For fixed-point inputs, the block generates an error during the Real-Time Workshop build process. You must remove this block to generate code for your fixed-point model.

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 The Read AVI File block is obsolete. It may be removed in a future version of Video and Image Processing Blockset. 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, or B ports have the same dimensions.	Same as I port	No
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.

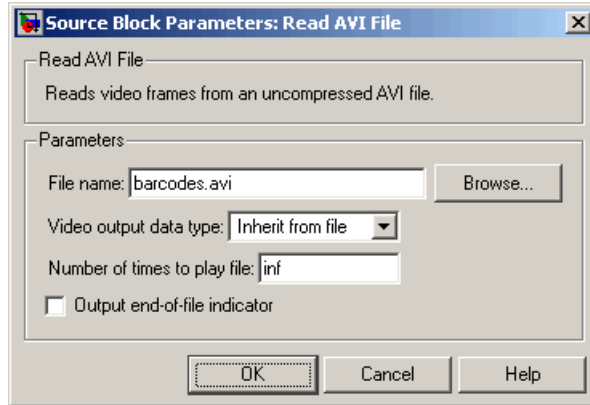
Use the **Number of times to play file** parameter to enter the number of times to play the file. The number you enter must be a positive integer or `inf`, to play the file until you stop the simulation.

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 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.

Read AVI File (Obsolete)

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
Image From File	Video and Image Processing Blockset
Image From Workspace	Video and Image Processing Blockset
To Multimedia File	Video and Image Processing Blockset

To Video Display	Video and Image Processing Blockset
Video From Workspace	Video and Image Processing Blockset
Video Viewer	Video and Image Processing Blockset

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.



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 integer8-, 16- 32-bit unsigned integer	No
EOF	Scalar value	Boolean	No

Use the **File name** parameter to specify the name of the binary 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.

Use the **Video format** parameter to specify the format of the binary video data. Your choices are Four character codes or Custom. See “Four Character Code Video Formats” on page 2-507 or “Custom Video Formats” on page 2-507 for more details.

Use the **Line ordering** parameter to determine 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 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 fills the other rows in decreasing order.

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 binary file is output from the block. 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.

Use the **Sample time** parameter to set the sample period of the output signal.

Four Character Code Video Formats

Four Character Codes (FOURCC) are used to 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.

Read Binary File

Use the **Component order in binary file** parameter to specify how the components are arranged within the file.

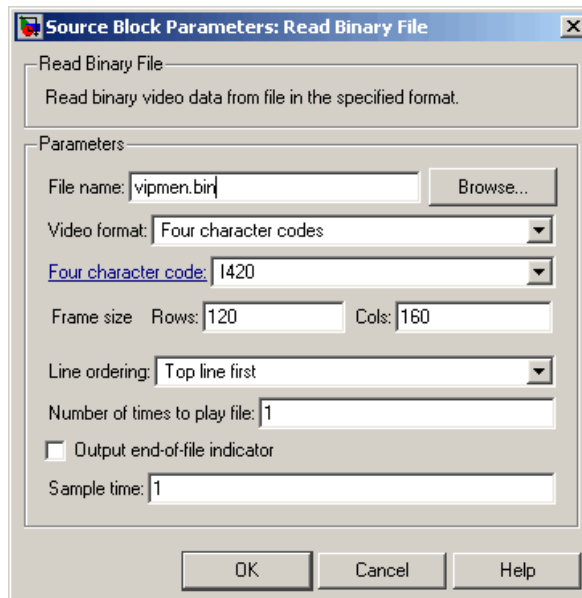
Select the **Interlaced video** check box if the binary file contains interlaced video data.

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.

Dialog Box

The Read Binary File dialog box appears as shown in the following figure.



File name

Specify the name of the binary file.

Video format

Specify the format of the binary video data. Your choices are Four character codes or Custom.

Four character code

From the 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. If you select Bottom line first, the block first fills the last row of the output matrix.

Number of times to play file

Enter a positive integer or `inf` to represent the number of times to play the file.

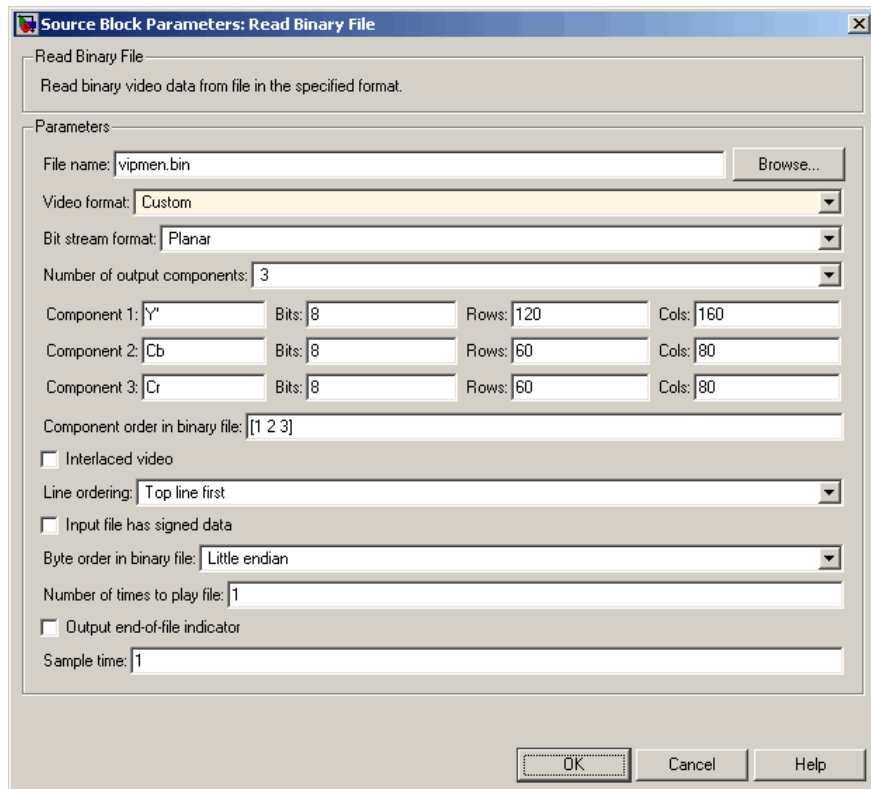
Output end-of-file indicator

Use this check box to determine whether the output is the last video frame in the binary file.

Sample time

Enter 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 is visible if, for the **Bit stream format** parameter, you select 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 how the components are arranged within 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

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.

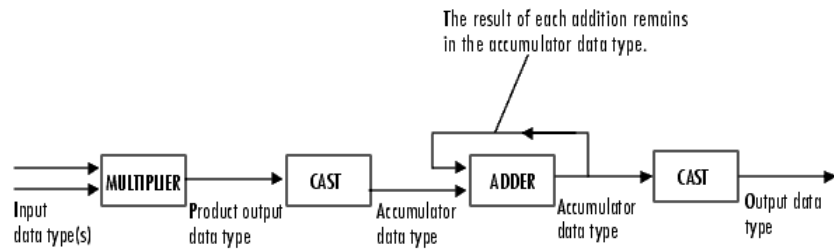
If you select the **Enable ROI processing** check box, the **Output flag indicating if any part of ROI is outside image bounds** check box appears in the dialog box. If you select this check box, the Flag port appears on the block. The following tables describe the Flag port output.

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.

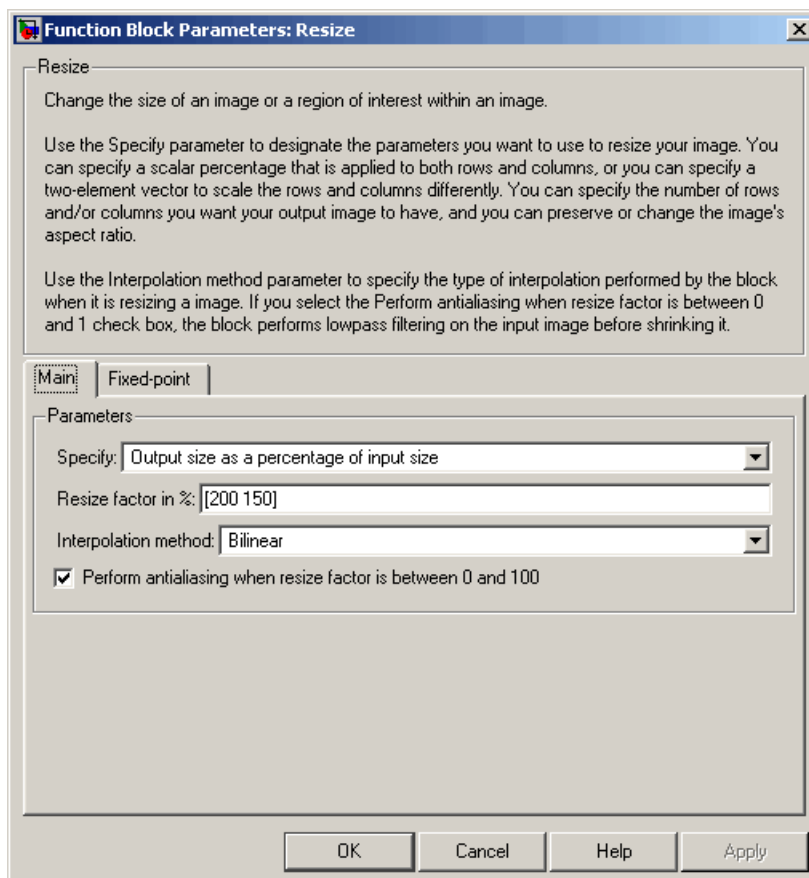
Resize



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:



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

Resize

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

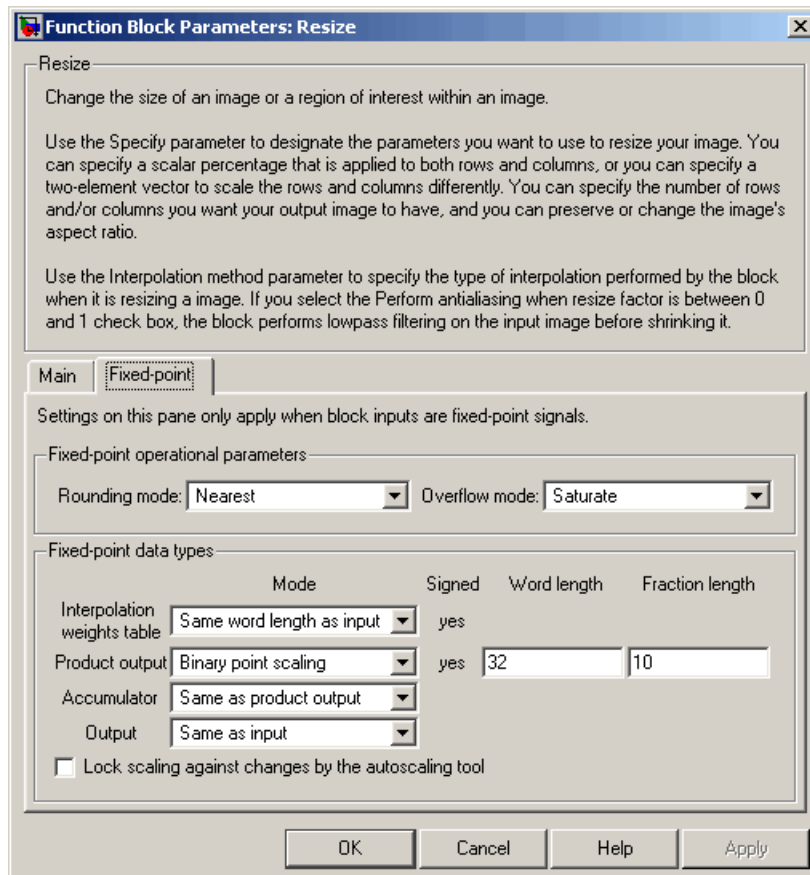
Select this check box to resize a particular region of each image. This parameter is available when the **Specify** parameter is set to Number of output rows and columns, the **Interpolation method** parameter is set to Nearest neighbor, Bilinear, or Bicubic, and the **Perform antialiasing when resize factor is between 0 and 100** check box is 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 **Fixed-point** 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.

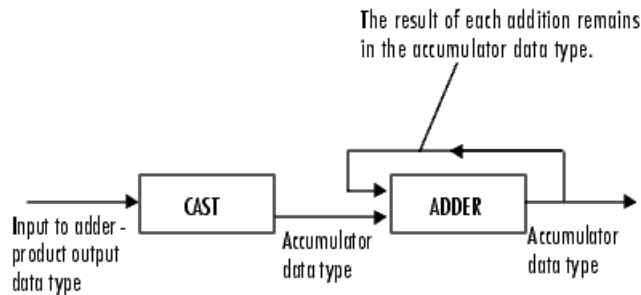
Product output



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.
- 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 0.

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

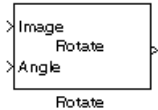
Rotate	Video and Image Processing Blockset
Shear	Video and Image Processing Blockset
Translate	Video and Image Processing Blockset
imresize	Image Processing Toolbox

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	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
Angle	Rotation angle	Same as Image port	No
Output	Rotated matrix	Same as Image port	No

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

Use the **Output size** parameter to 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.

Use the **Rotation angle source** parameter to specify how to enter your rotation angle. If you select Specify via dialog, the **Angle**

(radians) parameter appears in the dialog box. Use it to enter a real, scalar value for your rotation angle.

Note 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.

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**.

If, for the **Output size** parameter, you select **Expanded to fit rotated input image**, and, for **Rotation angle source** parameter, you select **Input port**, the **Maximum angle (enter pi radians to accommodate all positive and negative angles)** and **Display rotated image in** parameters appear in the dialog box. If, for the **Output size** parameter, you select **Same as input image**, and, for **Rotation angle source** parameter, you select **Input port**, the **Maximum angle (enter pi radians to accommodate all positive and negative angles)** and parameters appear in the dialog box.

For the **Maximum angle (enter pi radians to accommodate all positive and negative angles)** parameter, enter a scalar value,

$$0 < \max angle \leq \pi$$

radians, that represents the maximum angle by which you want to rotate the input image. The block determines which angle,

$0 \leq angle \leq \max angle$, requires the largest output matrix and sets the dimensions of the output port accordingly. To accommodate all angles, enter

$$\pi$$

Rotate

radians.

Use the **Display rotated image in** parameter to determine how the image is rotated in the display window. 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 image are always in contact with the top and left side of the Matrix Viewer window.

Use the **Sine value computation method** parameter to specify how much memory the Rotate block requires. 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.

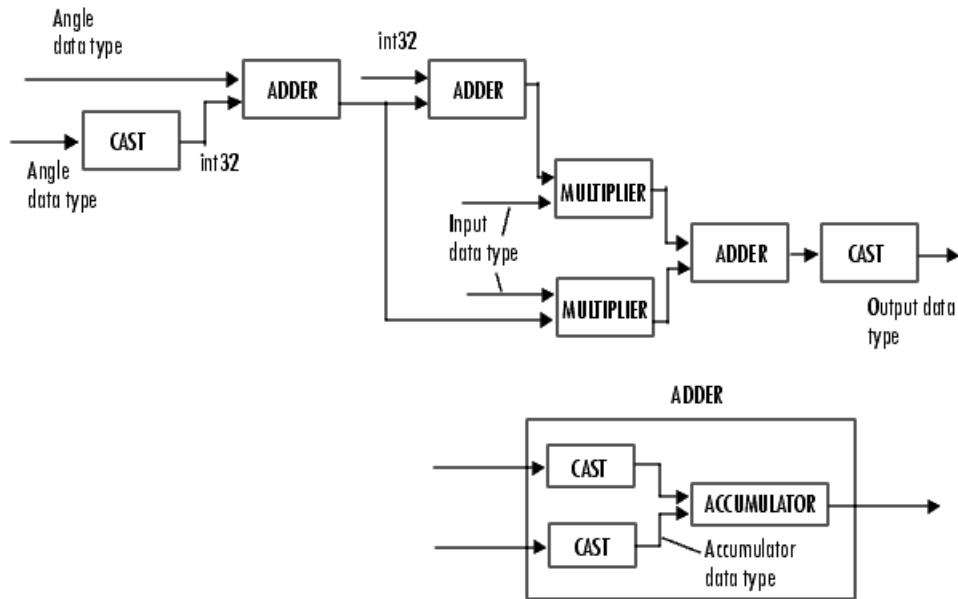
Use the **Interpolation method** parameter to specify which interpolation method the block uses to rotate the image. If you select `Nearest neighbor`, the block uses the value of the nearest pixel for the new pixel value. If you select `Bilinear`, the new pixel value is the weighted average of the two 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*.

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 Image Processing Toolbox. See page 208 of [1] for more information about this algorithm.

Fixed-Point Data Types

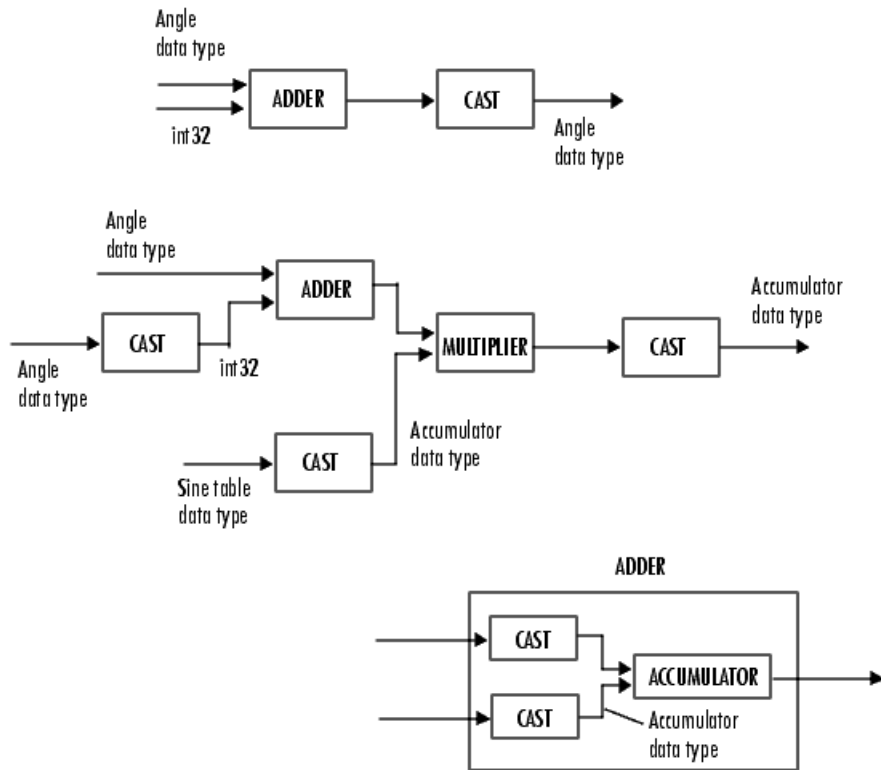
The following diagram shows the data types used in the Rotate block for bilinear interpolation of fixed-point signals.



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.

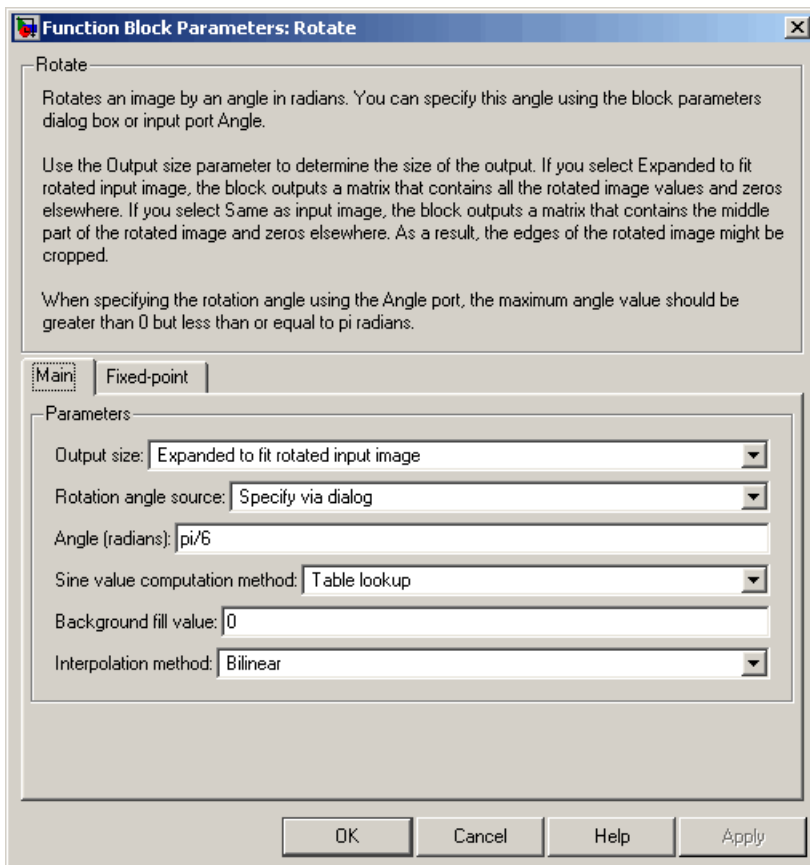
Rotate



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

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.

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.

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`.

Maximum angle (enter pi radians to accommodate all positive and negative angles)

Enter the maximum angle by which to rotate the input image. This parameter is visible if you set 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 image are always in contact with the top and left side of the `Matrix Viewer` window. 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

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

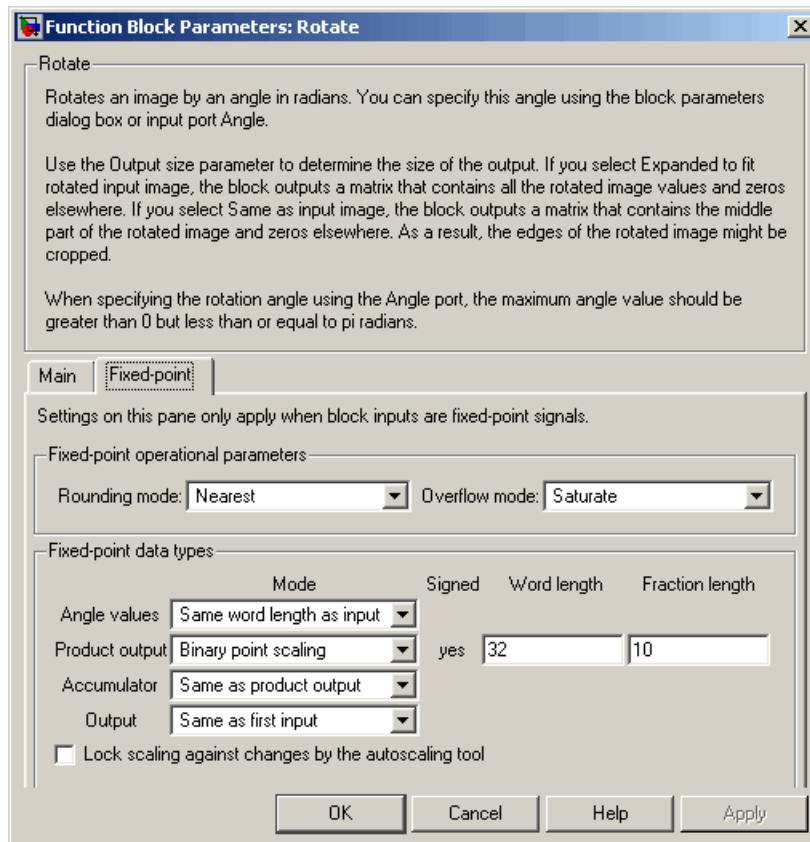
Specify a value for the pixels that are outside the image.

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 two nearest pixel values. If you select **Bicubic**, the new pixel value is the weighted average of the four nearest pixel values.

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

Rotate



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 Video and Image Processing Blockset is 0.

This parameter is only visible if, for the **Rotation angle source** parameter, you select **Specify** via dialog.

Product output



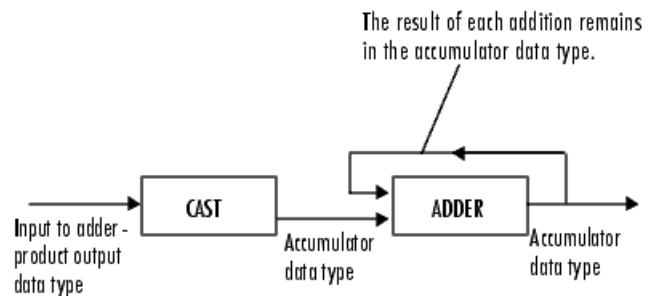
As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths.

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

Rotate

- 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 0.

References

[1] Wolberg, George. *Digital Image Warping*. Washington: IEEE Computer Society Press, 1990.

See Also

<code>Resize</code>	Video and Image Processing Blockset
<code>Translate</code>	Video and Image Processing Blockset
<code>Shear</code>	Video and Image Processing Blockset
<code>imrotate</code>	Image Processing Toolbox

SAD

Purpose Perform 2-D sum of absolute differences (SAD)

Library Analysis & Enhancement

Description



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	Scalar value that represents the zero-based index location of the minimum SAD value	<ul style="list-style-type: none"> • 32-bit signed integers 	No

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 (M_i , N_i) and the input at the Template port dimensions (M_t , N_t), then the output has dimensions ($M_i - M_t + 1$, $N_i - N_t + 1$).

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 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.

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

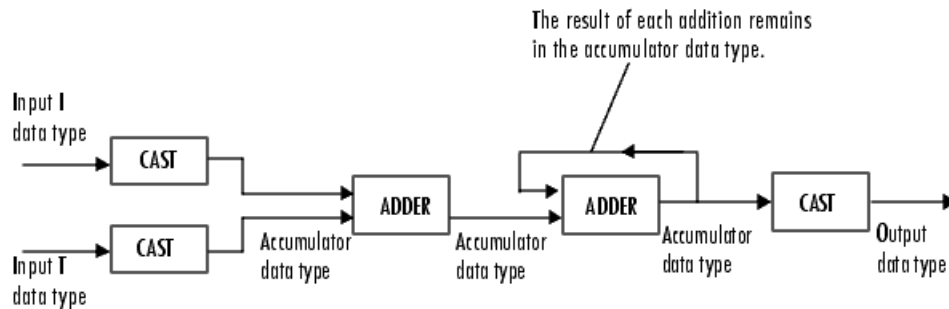
- Ignore -- Proceed with the computation and do not issue an alert. The output is not valid.
- Warn -- Display a warning message in the MATLAB Command Window, and continue the simulation. The output is not valid.
- Error -- Display an error dialog box and terminate the simulation.

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 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

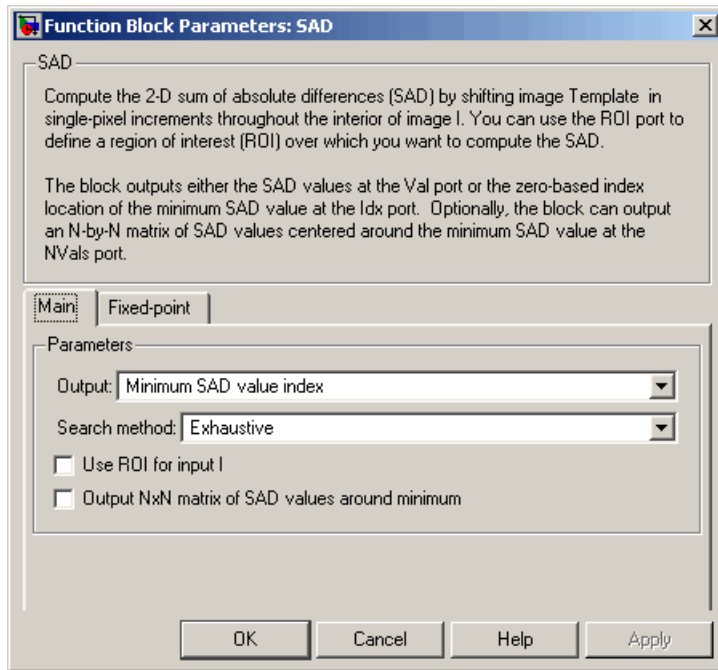
The following diagram shows the data types used in the SAD block for fixed-point signals.



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

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.

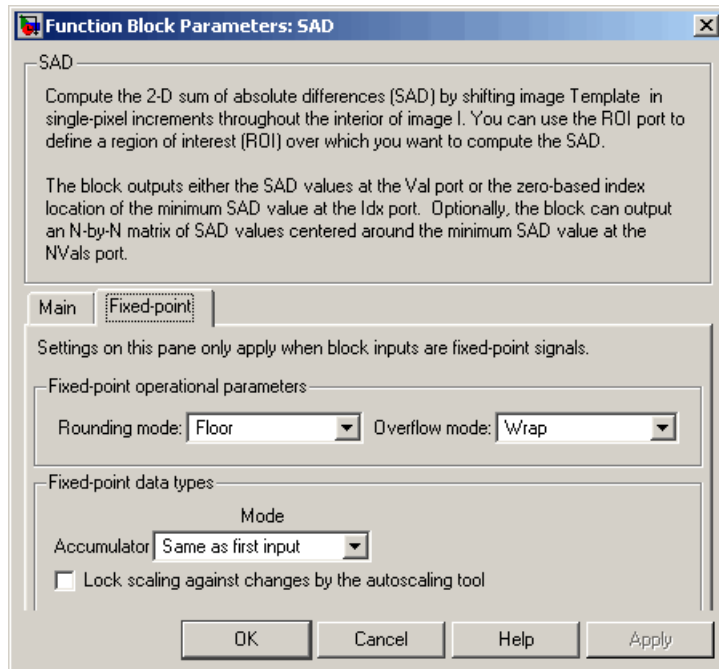
Output NxN matrix of SAD values around minimum

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.

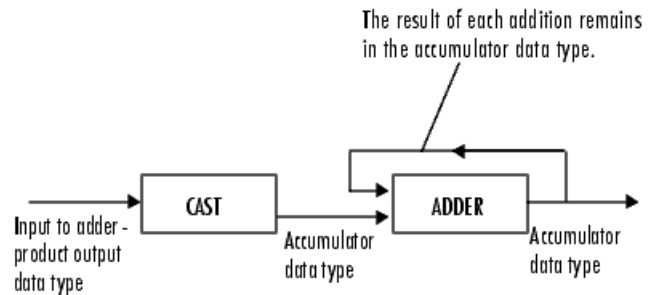
**Rounding mode**

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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

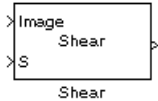
- [1] Koga, T., et al. *Motion-compensated interframe coding for video conferencing*. In Nat. Telecommun. Conf., Nov. 1981, G5.3.1-5, New Orleans, LA.
- [2] Wang, Yao, Jorn Ostermann, Ya-Qin Zhang. *Video Processing and Communications*. Upper Saddle River, NJ: Prentice Hall, 2002.

Shear

Purpose Shift rows or columns of image by linearly varying offset

Library Geometric Transformations

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.

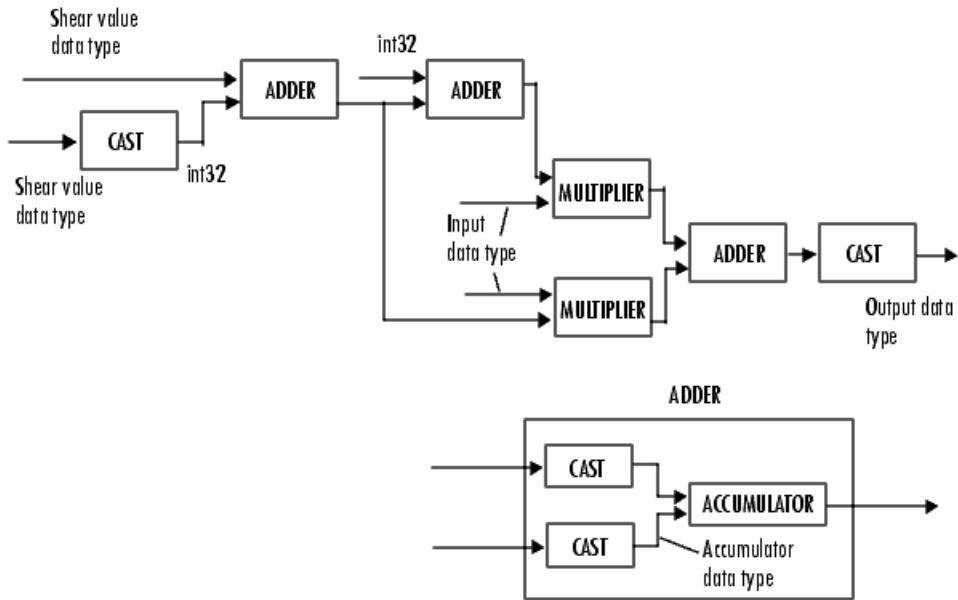
The number of pixels the block considers affects the complexity of the computation. Therefore, the nearest-neighbor interpolation is the most computationally efficient. However, because the accuracy of the method is proportional to the number of pixels considered, the bicubic method is the most accurate. 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.

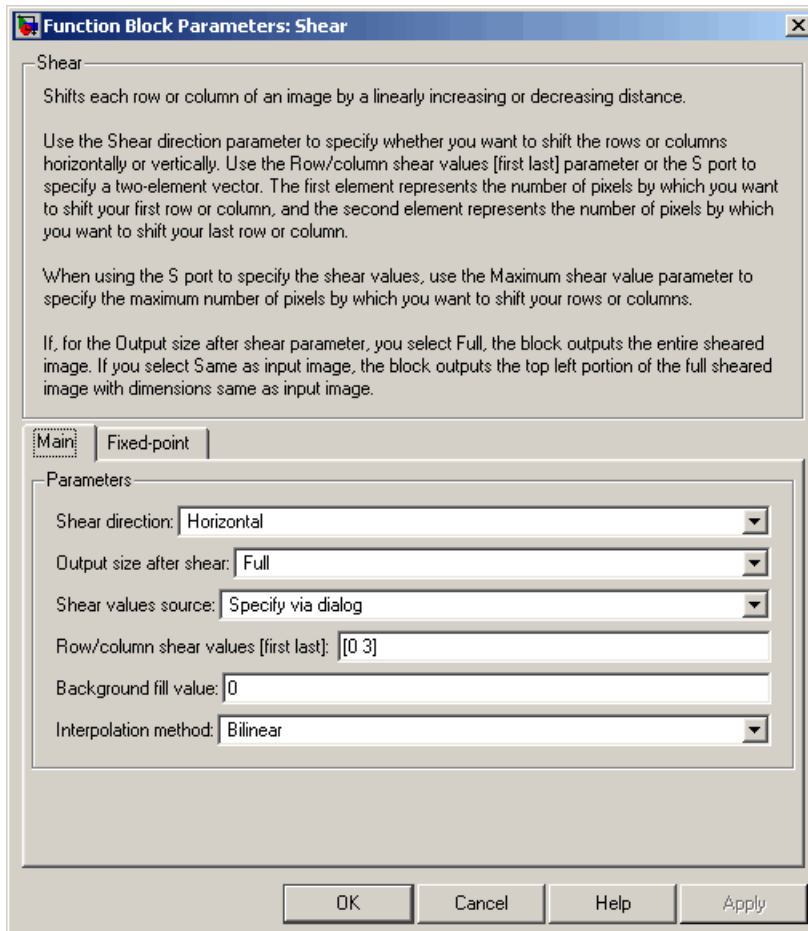


You can set the product output, accumulator, and output data types in the block mask.

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

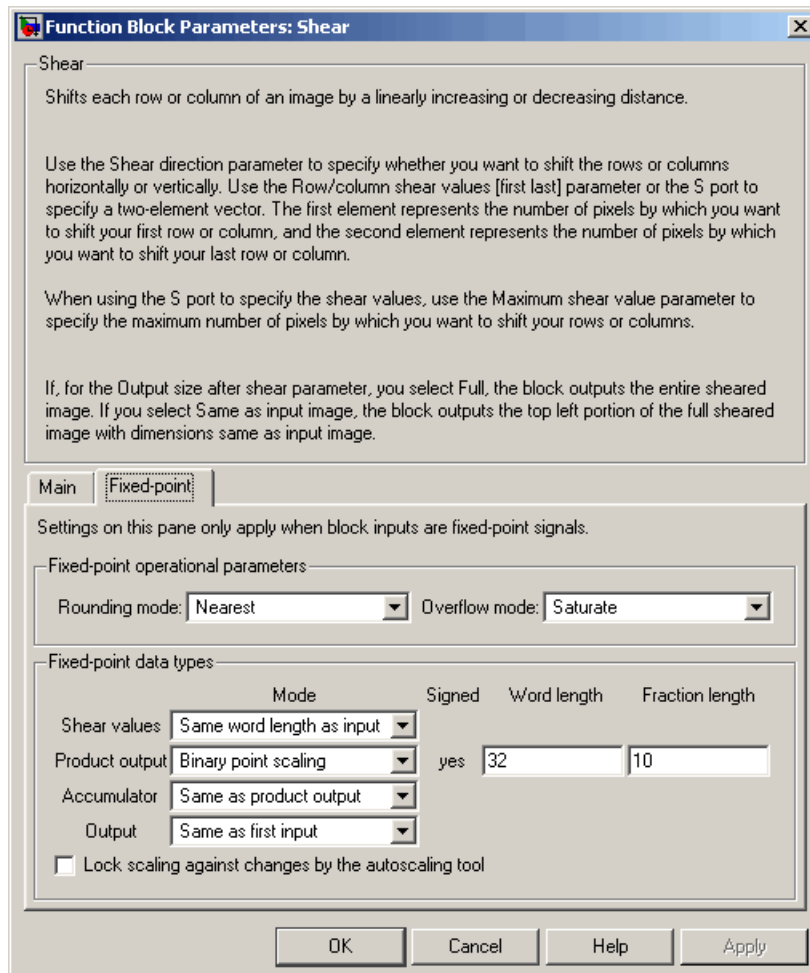
Specify a value for the pixels that are outside the image.

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 pixel values. If you select `Bicubic`, the new pixel value is the weighted average of the four nearest pixel values.

Shear

The **Fixed-point** pane of the Shear 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.

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 Video and Image Processing Blockset is 0.

This parameter is visible if, for the **Shear values source** parameter, you select **Specify** via dialog.

Product output

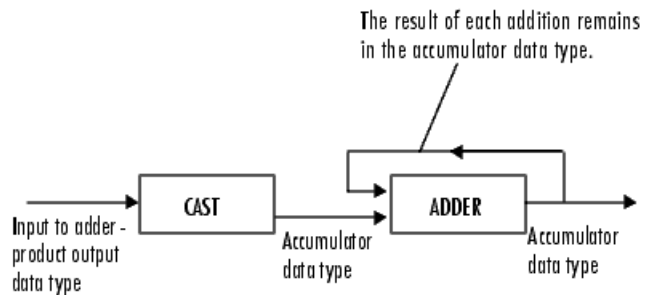


As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this

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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 0.

References

[1] Wolberg, George. *Digital Image Warping*. Washington: IEEE Computer Society Press, 1990.

See Also

Resize	Video and Image Processing Blockset
Rotate	Video and Image Processing Blockset
Translate	Video and Image Processing Blockset

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 documentation.

Purpose Send video data to display devices

Library Sinks

Description



The To Video Display block sends video data to a DirectX supported video output device or video camera. Alternatively, you can send the video data to a separate monitor or view the data in a window on your own computer screen.

Note This block supports code generation and is only available on Windows platforms. This block performs best on platforms with DirectX Version 9.0 or later and Windows Media Version 9.0 or later.

Port	Input	Supported Data Types	Complex Values Supported
Image	M-by-N matrix of intensity values or an M-by-N-by-3 color video signal	<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
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.	Same as I port	No

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.

To Video Display

Use the **Video output device** parameter to specify where you want the video stream to be sent. If you select On-screen video monitor, your video stream is displayed in the **To Video Display** window when you run your model. This window closes automatically when the simulation stops. The other options available in this parameter list are the DirectX supported video output devices installed on your system.

Select the **Full-screen** check box to display your video stream in a full-screen window. To return to other applications, hold down the **Alt** key and press **Tab**. If you have multiple To Video Display blocks in one model and you set the **Video output device** parameter to On-screen video monitor, we recommend selecting the **Full-screen** check box for only one of the blocks.

Select the **Remember video window size** check box if you want the block to save changes you make to the size of the video window.

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-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.

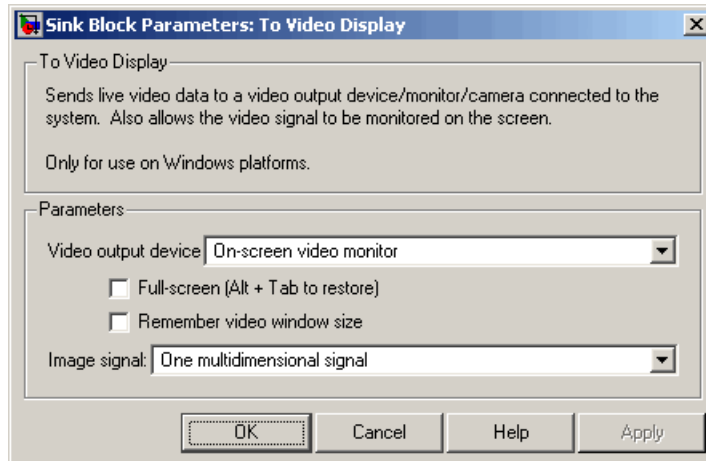
When running a model that contains a To Video Display block, the output of the block might be visible on the host monitor but not the external monitor or vice versa. There are two ways to work around this problem:

- 1** Replace the To Video Display block with a Video Viewer block.
- 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**.

Dialog Box

The To Video Display dialog box appears as shown in the following figure.



Video output device

Choose the video output device you want to use to display your video data.

Full-screen

Select this check box to display your video stream in a full screen window. This parameter is visible if, for the **Video output device** parameter, you select On-screen video monitor.

Remember video window size

Select this check box if you want the block to save changes you make to the size of the video window.

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-3 color video signal at one port. If you select Separate color

To Video Display

signals, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

See Also

Frame Rate Display

Video and Image Processing Blockset

From Multimedia File

Video and Image Processing Blockset

To Multimedia File

Video and Image Processing Blockset

Video To Workspace

Video and Image Processing Blockset

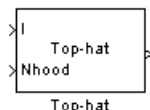
Video Viewer

Video and Image Processing Blockset

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

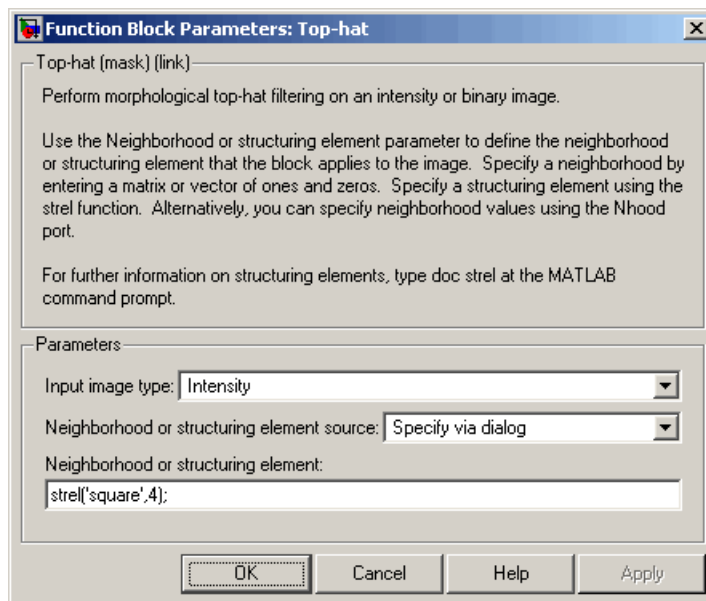
If your input image is a binary image, for the **Input image type** parameter, select Binary. If your input image is an intensity image, select Intensity.

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 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 Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or**

structuring element source parameter, you select Specify via dialog.

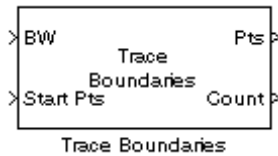
See Also

Bottom-hat	Video and Image Processing Blockset
Closing	Video and Image Processing Blockset
Dilation	Video and Image Processing Blockset
Erosion	Video and Image Processing Blockset
Label	Video and Image Processing Blockset
Opening	Video and Image Processing Blockset
imtophat	Image Processing Toolbox
strel	Image Processing Toolbox

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

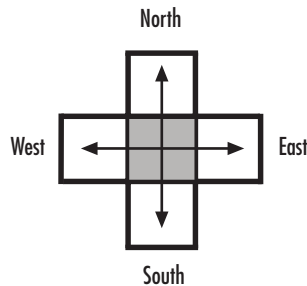
Trace Boundaries

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

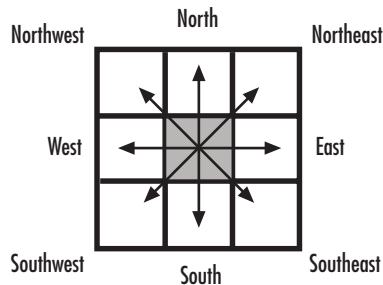
Use the **Connectivity** parameter to define which pixels are connected to each other. If you want a pixel to be connected to the other pixels located on the top, bottom, left, and right, select 4. If you want a pixel to be connected to the other pixels on the top, bottom, left, right, and

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.

Trace Boundaries

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.

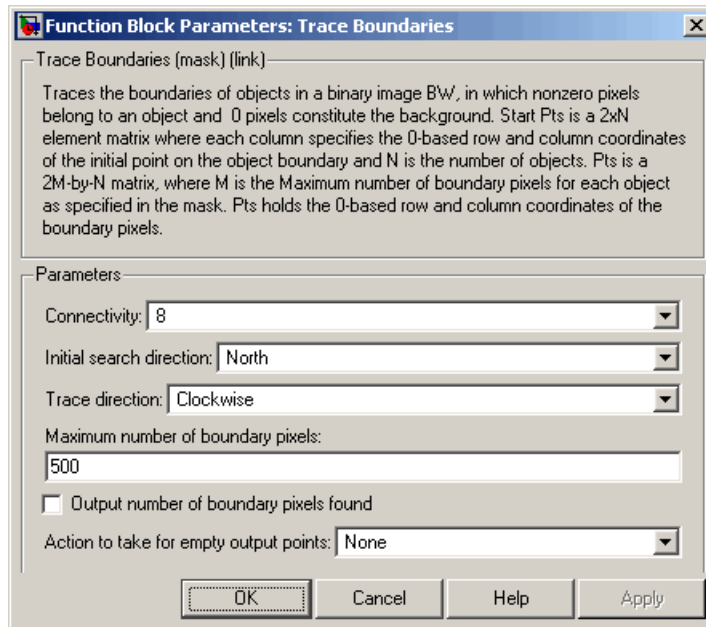
Because you specify the number of rows of the Pts port output matrix using the **Maximum number of boundary pixels** parameter, use the **Action to take for empty output points** parameter to specify what happens to the empty elements in this vector when the number of boundary pixels is less than the 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.

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.

Trace Boundaries

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
Label	Video and Image Processing Blockset
bwboundaries	Image Processing Toolbox
bwtraceboundary	Image Processing Toolbox

Purpose Translate image in 2-D plane using displacement vector

Library Geometric Transformations

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.

Translate

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 two 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 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 roughly 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*.

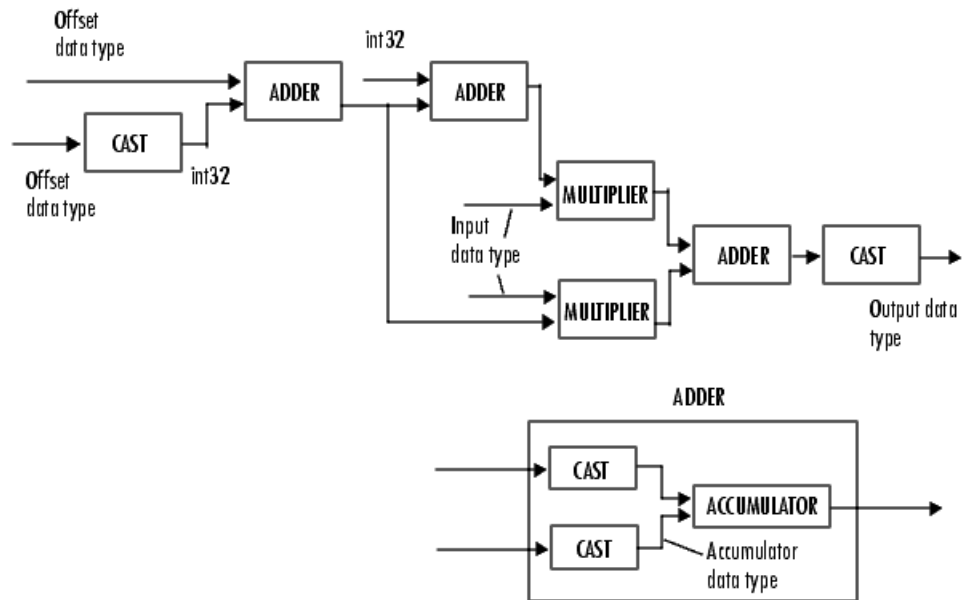
If, for the **Output size after translation** parameter, you select Full, and for the **Translation values source** parameter, you select Input port, the **Maximum offset** parameter appears in the dialog box. Use the **Maximum offset** parameter to enter a two-element vector of real, scalar values that 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.

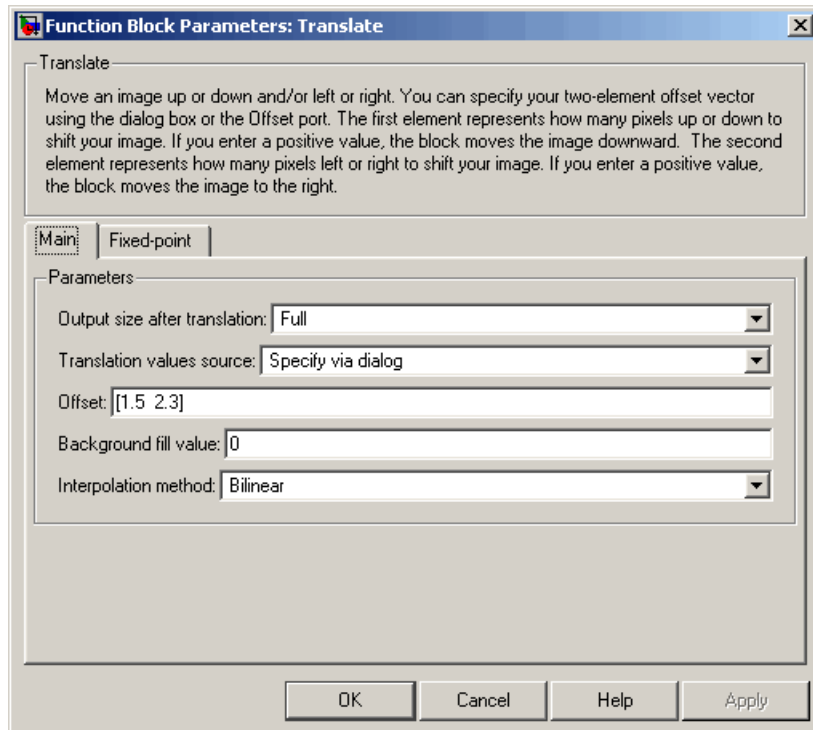
Translate



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 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.

Translate

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.

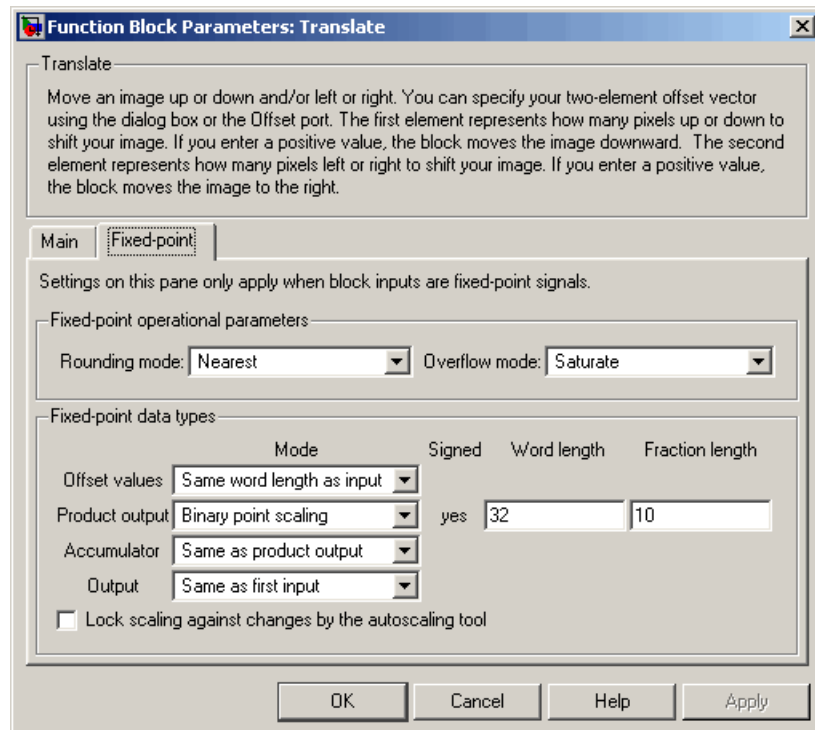
Interpolation method

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 two nearest pixel values. If you select **Bicubic**, the new pixel value is the weighted average of the four 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 **Fixed-point** pane of the Translate 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.

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 Video and Image Processing Blockset is 0.

This parameter is visible if, for the **Translation values source** parameter, you select **Specify via dialog**.

Product output

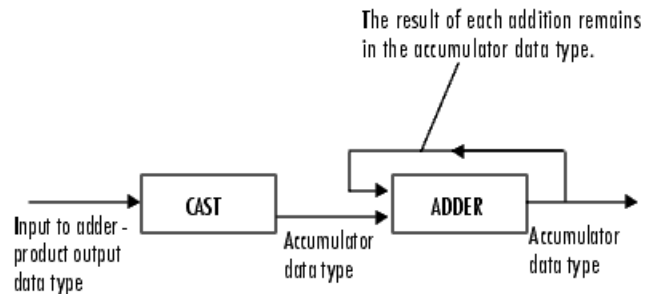


As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths.

- 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset 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 Video and Image Processing Blockset is 0.

References

[1] Wolberg, George. *Digital Image Warping*. Washington: IEEE Computer Society Press, 1990.

See Also

Resize	Video and Image Processing Blockset
Rotate	Video and Image Processing Blockset
Shear	Video and Image Processing Blockset

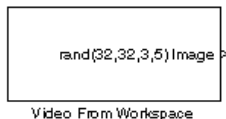
Purpose	Specify subset of rows or columns from input
Library	Utilities
Description	The Variable Selector block is a Signal Processing Blockset block. For more information, see the Variable Selector block reference page in the Signal Processing Blockset documentation.

Video From Workspace

Purpose Import video signal from MATLAB workspace

Library Sources

Description



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 `aviread` 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 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 = aviread('filename.avi')
```

The `aviread` function reads the AVI file into the MATLAB movie structure `mov`. The MATLAB movie structure might be obsolete in the future. For more information, see `aviread` 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:

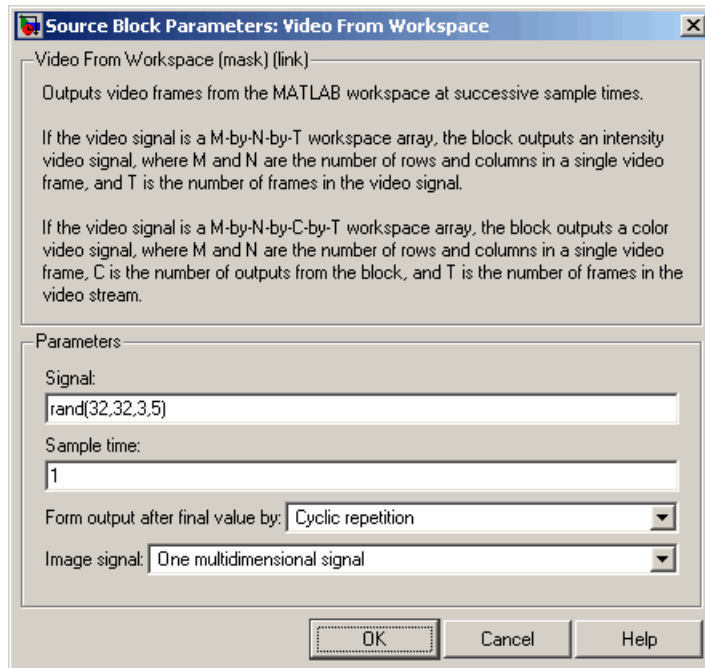
- When you specify `Setting To Zero`, the block generates zero-valued outputs for the duration of the simulation after generating the last frame of the signal.
- When you specify `Holding Final Value`, the block repeats the final frame for the duration of the simulation after generating the last frame 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.



Signal

Specify the MATLAB workspace variable that contains the video signal, or use the `aviread` 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).

Video From Workspace

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
Image From Workspace	Video and Image Processing Blockset
Read Binary File	Video and Image Processing Blockset
To Video Display	Video and Image Processing Blockset
Video Viewer	Video and Image Processing Blockset

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.

Video To Workspace

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 from the R, G, or B ports have the same dimensions.	Same as I port	No

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.

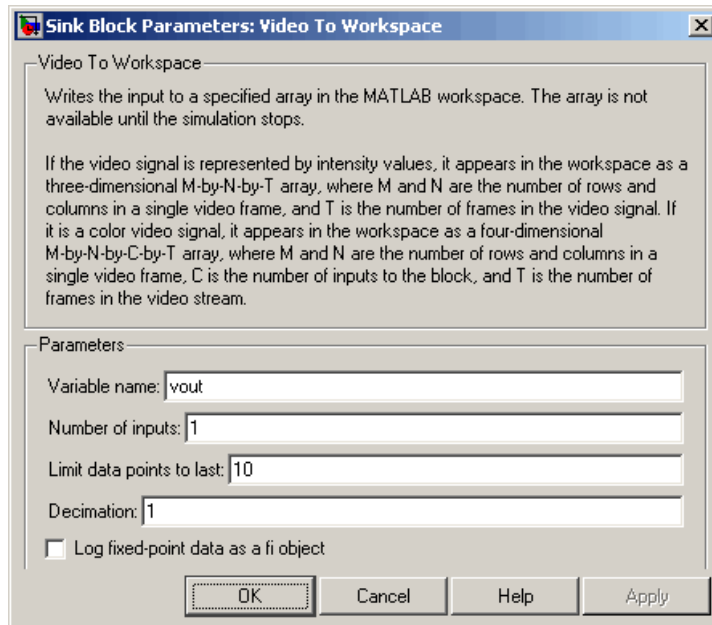
If you want to downsample your video signal, use the **Decimation** parameter to enter your decimation factor.

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

To Multimedia File

To Video Display

Video Viewer

Signal Processing Blockset

Video and Image Processing Blockset

Video and Image Processing Blockset

Video and Image Processing Blockset

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. During code generation, Real-Time Workshop does not generate code for this 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 • 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 RGB video stream. Inputs to the R, G, or B ports must have the same dimensions and data type.	Same as I port	No

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.

If you set the **Image signal** parameter to One multidimensional signal, the **Use colormap** and **Specify range of values** parameters appear on the block dialog box. If you select the **Use colormap** check box, you can use the **Colormap** parameter to specify a colormap or a call to a colormap-generating function.

MATLAB provides a number of functions for generating predefined colormaps, such as `hot`, `cool`, `bone`, and `autumn`. Each of these functions accepts the colormap size as an argument, and can be used in this parameter. For example, if you enter `gray(128)`, the image is displayed in 128 shades of gray. The color in the first row of the colormap matrix represents the minimum input value, and the color in the last row represents the maximum input value. Values between the minimum and maximum are quantized and mapped to the intermediate rows of the colormap matrix. For more information, see the MATLAB colormap function documentation.

If you select the **Specify range of values** check box, you can use the **Minimum input value** and **Maximum input value** parameters to define the range of your input. If you do not specify a range, the block assumes that Boolean, double-precision floating-point, and single-precision floating-point signals range from 0 to 1, and any other signals range between the minimum and maximum values supported by their data type.

Dialog Box

The Video Viewer dialog box appears as shown in the following figure.

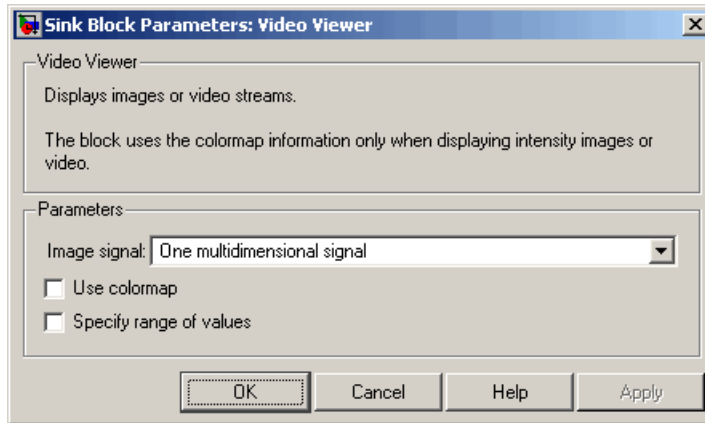


Image signal

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.

Use colormap

Select this check box to specify a colormap. This parameter is available if you set the **Image signal** parameter to **One multidimensional signal**.

Colormap

Specify the colormap to apply to the intensity image or video by entering a 3-column matrix defining the colormap or a call to a colormap-generating function such as **hot**, **cool**, **gray**, or **spring**. This parameter is available if you select the **Use colormap** check box.

Video Viewer

Specify range of values

Select this check box if you want to define the range for the input. This parameter is available if you set the **Image signal** parameter to One multidimensional signal.

Minimum input value

Use this parameter to define the range of the input; enter the smallest input value. This parameter is available if you select the **Specify range of values** check box.

Maximum input value

Use this parameter to define the range of the input; enter the largest input value. This parameter is available if you select the **Specify range of values** check box.

See Also

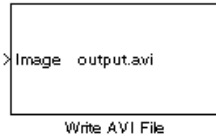
From Multimedia File mplay	Video and Image Processing Blockset
To Multimedia File	Video and Image Processing Blockset
To Video Display	Video and Image Processing Blockset
Video To Workspace	Video and Image Processing Blockset

Write AVI File (Obsolete)

Purpose Write video frames to uncompressed AVI file

Library Sinks

Description



The Write AVI File block is obsolete. It may be removed in a future version of Video and Image Processing Blockset. 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

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.

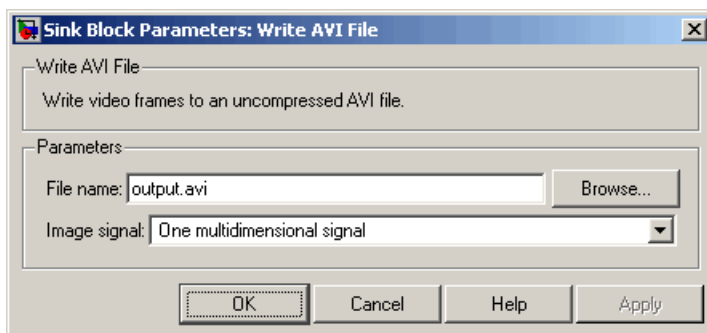
Write AVI File (Obsolete)

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.

See Also

To Multimedia File

Video and Image Processing Blockset

To Video Display

Video and Image Processing Blockset

Video To Workspace

Video and Image Processing Blockset

Video Viewer

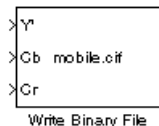
Video and Image Processing Blockset

Write Binary File

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.



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

Use the **File name** parameter to 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 as well as the filename.

Use the **Video format** parameter to specify the format of the binary video data. Your choices are Four character codes or Custom. See “Four Character Code Video Formats” on page 2-599 or “Custom Video Formats” on page 2-599 for more details.

Use the **Line ordering** parameter to determine 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 fills the file with the other rows in increasing order. If you select Bottom line first, the block first fills the binary file with the last row of the video frame. It fills the file with the other rows in decreasing order.

Four Character Code Video Formats

Four Character Codes (FOURCC) are used to 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.

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.

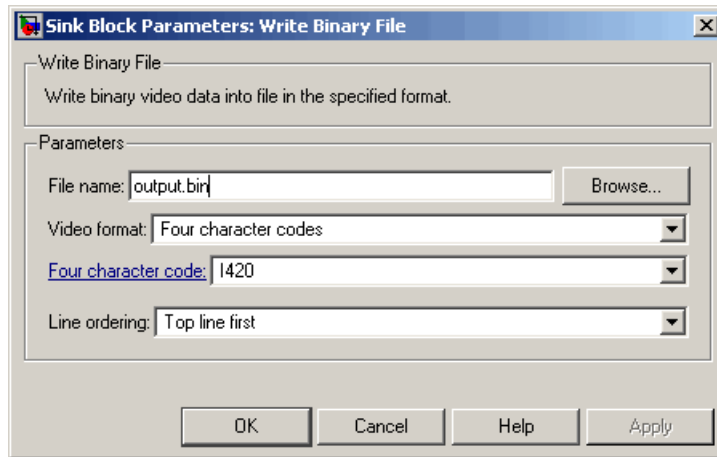
Select the **Write signed data to output file** check box if your input data is signed.

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.

Write Binary File

Dialog Box

The Write Binary File dialog box appears as shown in the following figure.



File name

Specify the name of the binary file.

Video format

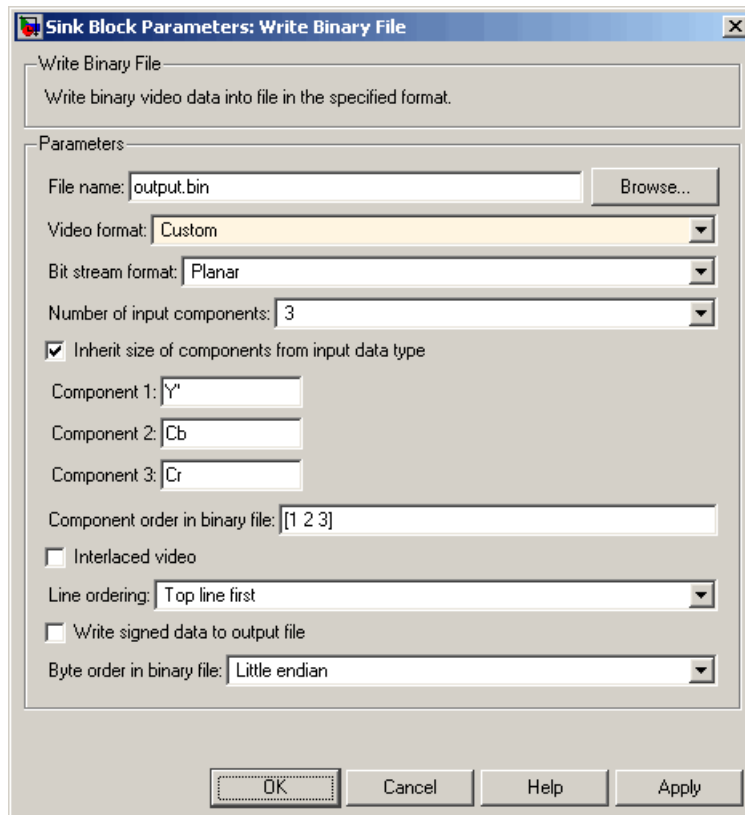
Specify the format of the binary video data. Your choices are Four character codes or Custom.

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. If you select Bottom line first, the block first fills the binary file with the last row of the video frame.



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.

Write Binary File

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 Video and Image Processing Blockset

Functions — Alphabetical List

isfilterseparable

Purpose Determine whether filter coefficients are separable

Syntax [S, HCOL, HROW] = isfilterseparable(H)

Description [S, HCOL, HROW] = isfilterseparable(H) uses the filter kernel, H, to determine whether the filter coefficients are separable. Here, S is a Boolean variable that is 1 if the filter is separable and 0 if it is not. If S is 1, HCOL is a vector of vertical filter coefficients, and HROW is a vector of horizontal filter coefficients. Otherwise, HCOL and HROW do not exist.

See Also 2-D FIR Filter Video and Image Processing Blockset

Purpose View video from MATLAB workspace, multimedia file, or Simulink model

Syntax

```
mplay  
mplay('filename.avi')  
mplay(a)  
mplay(a, fps)
```

Description Use the MPlay GUI to view video from files or the MATLAB workspace. You can also use it to view video signals in Simulink models. If the video contains audio, the GUI ignores it and plays only the video frames. In the model window, you can access this GUI by selecting **Tools > MPlay Video Viewer**.

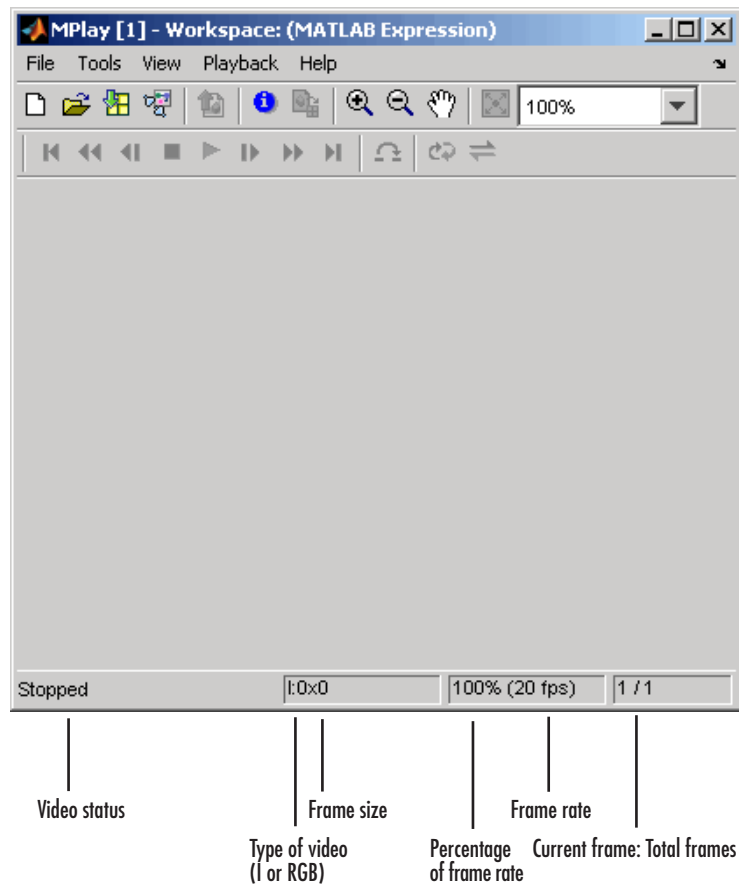
mplay opens an MPlay GUI.

mplay('filename.avi') connects the MPlay GUI to the specified AVI file.

mplay(a) connects the MPlay GUI to the variable in the MATLAB workspace, a.

mplay(a, fps) connects the MPlay GUI to the variable in the MATLAB workspace, a, with the specified frame rate in frames per second, fps. By default, fps is 20.

Note mplay only works when the Simulink simulation mode is set to Normal. It does not work when you use “Acceleration Modes”.



This reference page contains the following sections:





- “MPlay Configuration Dialog Box” on page 3-9
- “Video Information Dialog Box” on page 3-14
- “Colormap Dialog Box” on page 3-14
- “Frame Rate Dialog Box” on page 3-15

- “Saving the Settings of Multiple MPlay GUIs” on page 3-17







You can open several instances of the MPlay GUI simultaneously to view multiple video data sources at once. You can also dock these MPlay GUIs in the MATLAB desktop. Use the figure arrangement buttons in the upper-right corner of the Sinks window to control the placement of the docked GUIs.

For more information about the MPlay GUI, see “Viewing Videos from the MATLAB Workspace”, “Viewing Video Files”, and “Viewing Video Signals in Simulink” in the *Video and Image Processing Blockset User’s Guide*. You can also see the “MPlay Simulink Tutorial” in the “Video Playback” section of the Video and Image Processing Blockset demos.










MPlay GUI Buttons

Button	Menu Equivalent	Shortcut Keys and Accelerators	Description
	File > New MPlay	Ctrl+N	Open a new MPlay GUI.
	File > Open	Ctrl+O	Connect to a video file.
	File > Import from Workspace	Ctrl+I	Connect to a video that is a variable in the MATLAB workspace.
	File > Connect to Simulink Signal	Ctrl+S	Connect to a Simulink signal.









MPlay GUI Buttons (Continued)

Button	Menu Equivalent	Shortcut Keys and Accelerators	Description
	File > Export to Image Tool	Ctrl+E	<p>Send the current video frame to the Image Tool. For more information, see “Using the Image Tool to Explore Images” in the Image Processing Toolbox documentation.</p> <hr/> <p>Note The Image Tool only knows 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> <hr/>
	Tools > Video Information	I	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.






MPlay GUI Buttons (Continued)

Button	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. This option is available if the Maintain Fit to Window button is not selected.
For workspace and file sources:			
	Playback > Go to First	F, Home	Go to the first frame of the video.
	Playback > Rewind	Up arrow	Jump back 10 frames.
	Playback > Step Back	Left arrow, Page Up	Step back one frame.
	Playback > Stop	S	Stop the video.
	Playback > Play	P, Space bar	Play the video.
	Playback > Pause	P, Space bar	Pause the video. This button is visible only when the video is playing.
	Playback > Step Forward	Right arrow, Page Down	Step forward one frame.
	Playback > Fast Forward	Down arrow	Jump forward 10 frames.
	Playback > Go to Last	L, End	Go to the last frame of the video.

MPlay GUI Buttons (Continued)

Button	Menu Equivalent	Shortcut Keys and Accelerators	Description
	Playback > Jump to	J	Jump to a specific frame.
	Playback > Playback Modes > Repeat	R	Repeated video playback.
	Playback > Playback Modes > Forward play	A	Play the video forward.
	Playback > Playback Modes > Backwardplay	A	Play the video backward.
	Playback > Playback Modes > AutoReverse play	A	Play the video forward and backward.
For Simulink sources:			
	Playback > Stop	S	Stop the video. This button also controls the Simulink model.
	Playback > Start	P, Space bar	Play the video. This button also controls the Simulink model.
	Playback > Pause	P, Space bar	Pause the video. This button also controls the Simulink model and is visible only when the video is playing.

MPlay GUI Buttons (Continued)

Button	Menu Equivalent	Shortcut Keys and Accelerators	Description
	Playback > Step Forward	Right arrow, Page Down	Step forward one frame. This button also controls the Simulink model.
	Playback > Simulink Snapshot	N/A	Click this button to freeze the display in the MPlay window.
	Playback > Highlight Simulink Signal	Ctrl+L	In the model window, highlight the Simulink signal the MPlay GUI is displaying.
	Playback > Floating Signal Connection (not selected)	N/A	Indicates persistent Simulink connection. In this mode, the MPlay GUI is always associated with the Simulink signal you selected before you clicked the Connect to Simulink Signal button.
	Playback > Floating Signal Connection (selected)	N/A	Indicates floating Simulink connection. In this mode, you can click different signals in the model, and the MPlay GUI displays them. Only one MPlay GUI can be in floating-scope mode at one time.

MPlay Configuration Dialog Box

The MPlay Configuration dialog box enables you to change the behavior and appearance of the GUI as well as the behavior of the playback shortcut keys.

- To open the Configuration dialog box, select **File > Configuration Set > Edit**.

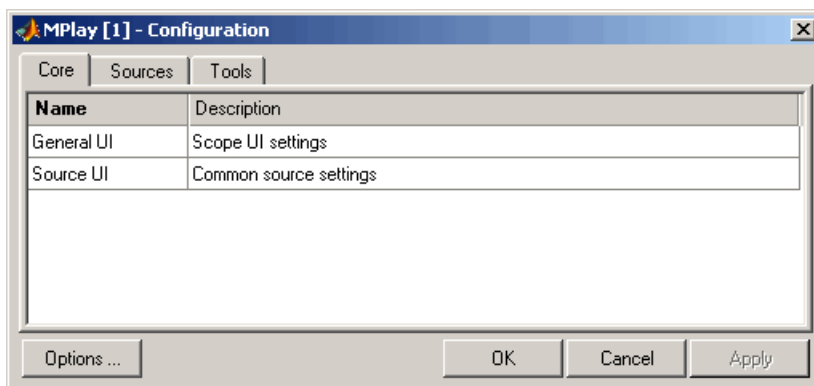
- To save the configuration settings for future use, select **File > Configuration Set > Save as**.

Note By default, the MPlay GUI uses the configuration settings from the file `mplay.cfg`. If you want to store your configuration settings in this file, you should first create a backup copy of the file.

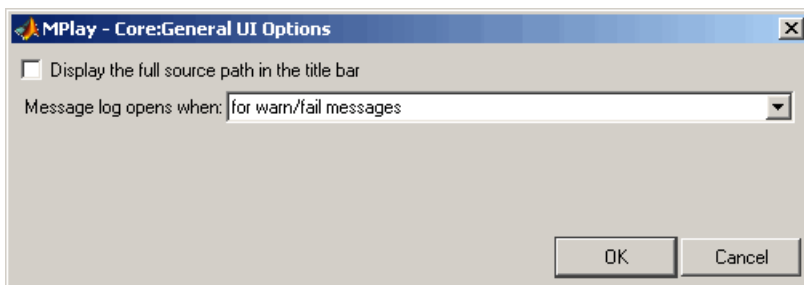
- To load a preexisting configuration set, select **File > Configuration Set > Load**.

Core Pane

The Core pane controls the GUI's general and source settings.



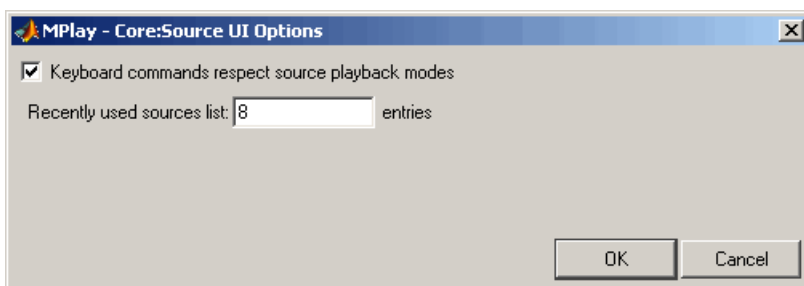
Click General UI, and click the **Options** button to open the General UI Options dialog box.



If you select the **Display the full source path in the title bar** check box, the GUI displays the full path to the video data source in the title bar. Otherwise, it displays a shortened name.

Use the **Message log opens when** 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.

Click Source UI and click the **Options** button to open the Source UI Options dialog box.

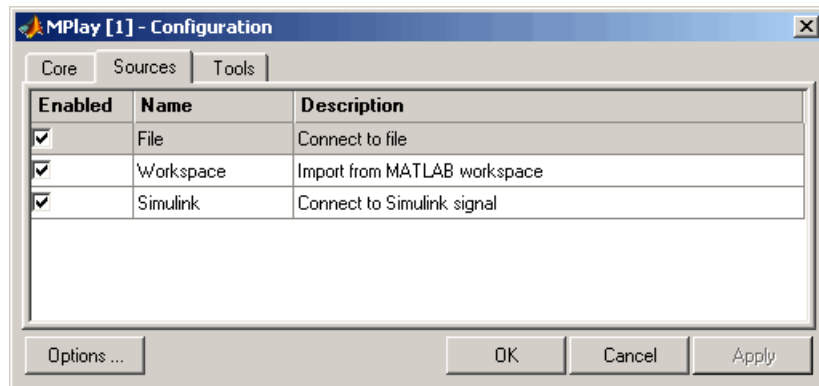


If you select the **Keyboard commands respect playback mode** check box, the keyboard shortcut keys are aware of whether or not you selected **Repeat**, **Forward play**, **Backward play**, or **Fwd/Back play** for the playback mode. If you clear this check box, the keyboard shortcut keys behave as if the playback mode is set to **Forward play** and **Repeat** is set to off.

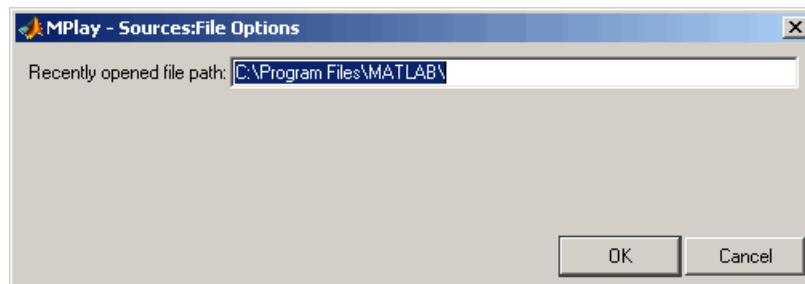
Use the **Recently used sources list** parameter to control the number of sources you see in the **File** menu.

Sources Pane

The Sources pane contains the GUI options that relate to connecting to different sources. Select the **Enabled** check box next to each source type to specify to which type of source you want to connect the GUI.

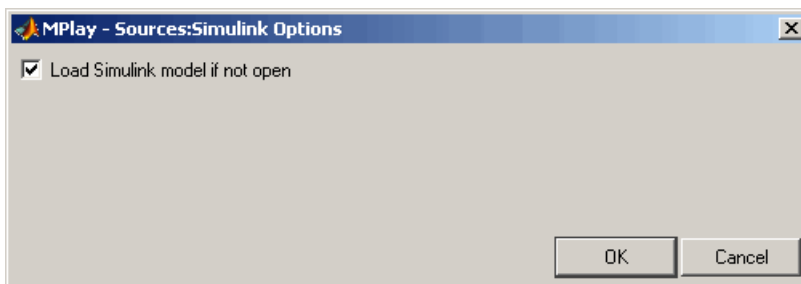


Click **File**, and click the **Options** button to open the File Options dialog box.



Use the **Recently opened file path** parameter to control the directory that is displayed in the Connect to File dialog box when you click **File > Open**.

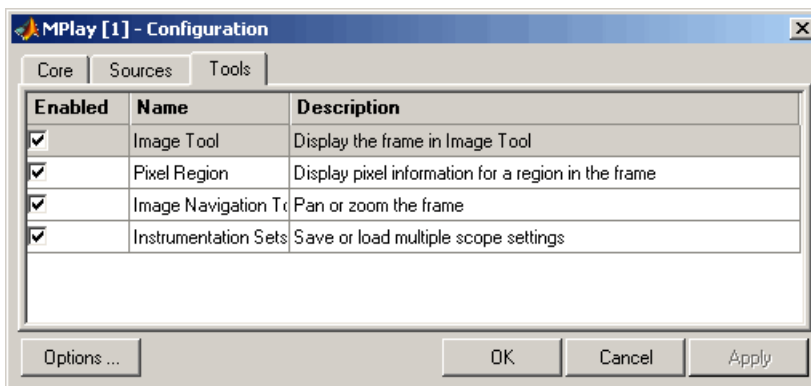
Click Simulink, and click the **Options** button to open the Simulink Options dialog box.



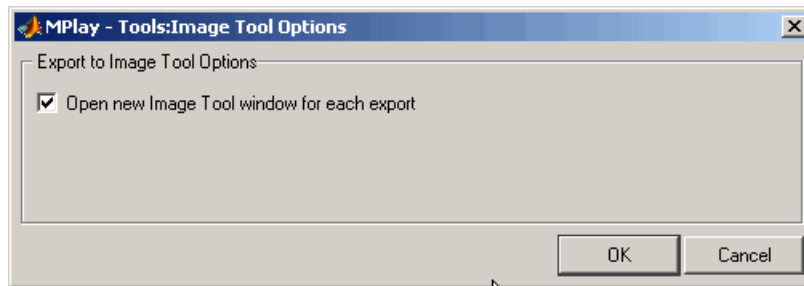
Select the **Load Simulink model if not open** check box if you want the Simulink model associated with an MPlay GUI to open when you open the GUI.

Tools Pane

The Tools pane contains the tools that are available on the MPlay GUI. Select the **Enabled** check box next to the tool name to specify which tools to include on the GUI.




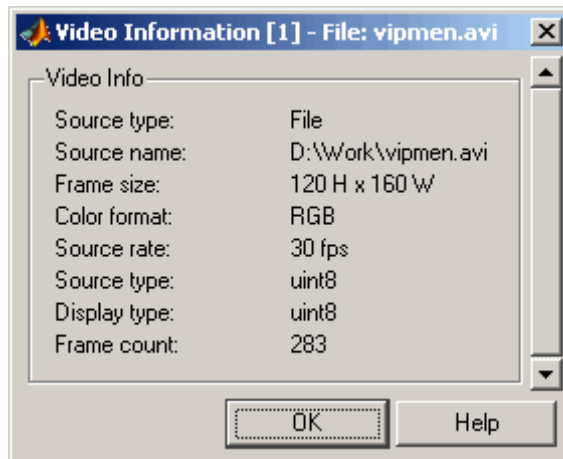
Click Image Tool, and click the **Options** button to open the Image Tool Options dialog box.



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.

Video Information Dialog Box

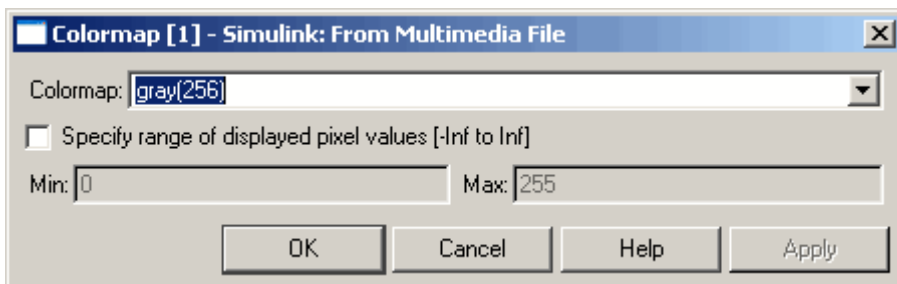
The Video Information dialog box lets you view basic information about the video. To open this dialog box, click **Tools > Video Information** or click  .



Colormap Dialog Box

The Colormap dialog box lets you change the colormap of an intensity video. The parameters on this dialog box are not available when the

GUI is displaying RGB video data. To open this dialog box, click **Tools > Colormap** or press **C**.



Use the **Colormap** parameter to specify the colormap to apply to the intensity video.

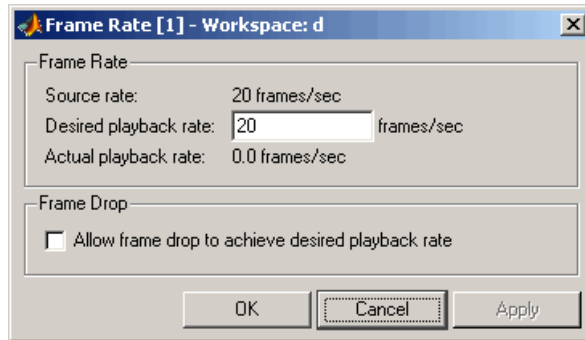
If you know that the pixel values do not use the entire data type range, you can select the **Specify range of displayed pixel values** check box and enter the range for your data. The dialog box automatically displays the range based on the data type of the pixel values.

Frame Rate Dialog Box

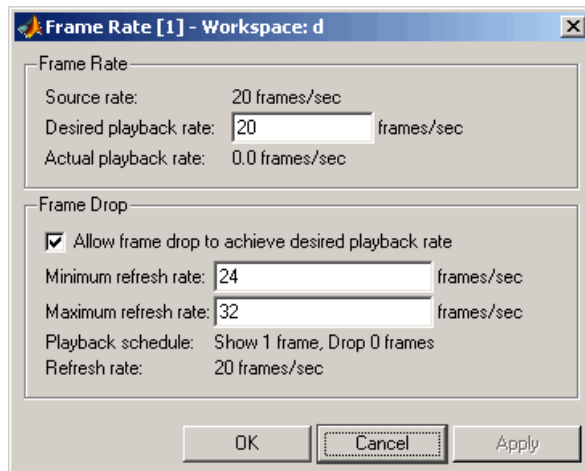
The Frame Rate dialog box displays the frame rate of the source, lets you change the rate at which the MPlay GUI plays the video, and displays the actual playback rate.

Note This dialog box is not available if you are using the MPlay GUI to view a video signal in a Simulink model.

The *playback rate* is the number of frames the GUI processes per second. You can use the **Desired playback rate** parameter to decrease or increase the playback rate. To open this dialog box, click **Playback > Frame Rate** or press **T**.



If you want to increase the actual playback rate, but your system's hardware cannot keep up with the desired rate, select the **Allow frame drop to achieve desired playback rate** check box as shown in the following figure. This parameter enables the MPlay GUI to achieve the playback rate by dropping video frames. As a result, the video playback might appear choppy.



For example, suppose you set the **Desired playback rate** to 150 frames/sec. The MPlay GUI can achieve this desired playback rate by setting the Playback schedule to Show 1 frame, Drop 5 frames, which

results in a frame size of 6. It also sets the Refresh rate, or how often the GUI updates the screen, to 25 frames/sec. If you multiply the refresh rate (25 frames/sec) by the frame size (6), you see that the GUI can achieve the Desired playback rate (150 frames/sec) by using these parameter values.

You can use the **Minimum refresh rate** and **Maximum refresh rate** parameters to adjust the playback schedule of the video displayed in the GUI in the following ways:

- Increase the **Minimum refresh rate** parameter to achieve smoother playback.
- Decrease the **Maximum refresh rate** parameter to reduce the demand on your system's hardware.

Saving the Settings of Multiple MPlay GUIs

The MPlay GUI enables you to save and load the settings of multiple GUI instances. That way, you only need to configure the MPlay GUIs that are associated with your model once. To save the GUI settings, click **File > Instrumentation Sets > Save Set**. To open the preconfigured MPlay GUIs, click **File > Instrumentation Sets > Load Set**.

See Also

To Multimedia File	Video and Image Processing Blockset
To Video Display	Video and Image Processing Blockset
Video To Workspace	Video and Image Processing Blockset
Video Viewer	Video and Image Processing Blockset

- 2-D Autocorrelation block 2-2
- 2-D Convolution block 2-8
- 2-D Correlation block 2-19
- 2-D DCT block 2-30
- 2-D FFT block 2-39
- 2-D FIR Filter block 2-53
- 2-D Histogram block 2-65
- 2-D IDCT block 2-75
- 2-D IFFT block 2-84
- 2-D Maximum block 2-97
- 2-D Mean block 2-108
- 2-D Median block 2-121
- 2-D Minimum block 2-127
- 2-D Standard Deviation block 2-138
- 2-D Variance block 2-150

A

- Autothreshold block 2-166

B

- Blob Analysis block 2-176
- Block Matching block 2-192
- Block Processing block 2-203
- Bottom-hat block 2-212

C

- Chroma Resampling block 2-215
- Closing block 2-223
- Color Space Conversion block 2-226
- Compositing block 2-237
- Contrast Adjustment block 2-249
- Corner Detection block 2-257

D

- Deinterlacing block 2-272
- Demosaic block 2-285
- Dilation block 2-292

- Draw Markers block 2-296
- Draw Shapes block 2-308

E

- Edge Detection block 2-318
- Erosion block 2-331

F

- Find Local Maxima 2-335
- Frame Rate Display block 2-340
- From Multimedia File block 2-342
- function
 - isfilterseparable 3-2
 - mplay 3-3

G

- Gamma Correction block 2-343
- Gaussian Pyramid block 2-348

H

- Histogram Equalization block 2-359
- Hough Lines block 2-363
- Hough Transform block 2-372

I

- Image Complement block 2-380
- Image Data Type Conversion block 2-382
- Image From File block 2-386
- Image From Workspace block 2-393
- Image Pad 2-399
- Insert Text block 2-412
- isfilterseparable function 3-2

K

- Kalman Filter block 2-427

L

Label block 2-428
Laplacian pyramid 2-348

M

Maximum block 2-432
Median Filter block 2-433
Minimum block 2-442
MPlay GUI 3-3

O

Opening block 2-443
Optical Flow block 2-446

P

Projective Transformation block 2-465
PSNR block 2-500

R

Read AVI File block 2-502
Read Binary File block 2-506
Resize block 2-512

Rotate block 2-524

S

SAD block 2-536
Shear block 2-546

T

To Video Display block 2-557
Top-hat block 2-561
Trace Boundaries block 2-565
Translate block 2-571

V

Variable Selector 2-581
Video From Workspace block 2-582
Video To Workspace block 2-587
Video Viewer block 2-591

W

Write AVI File block 2-595
Write Binary File 2-598