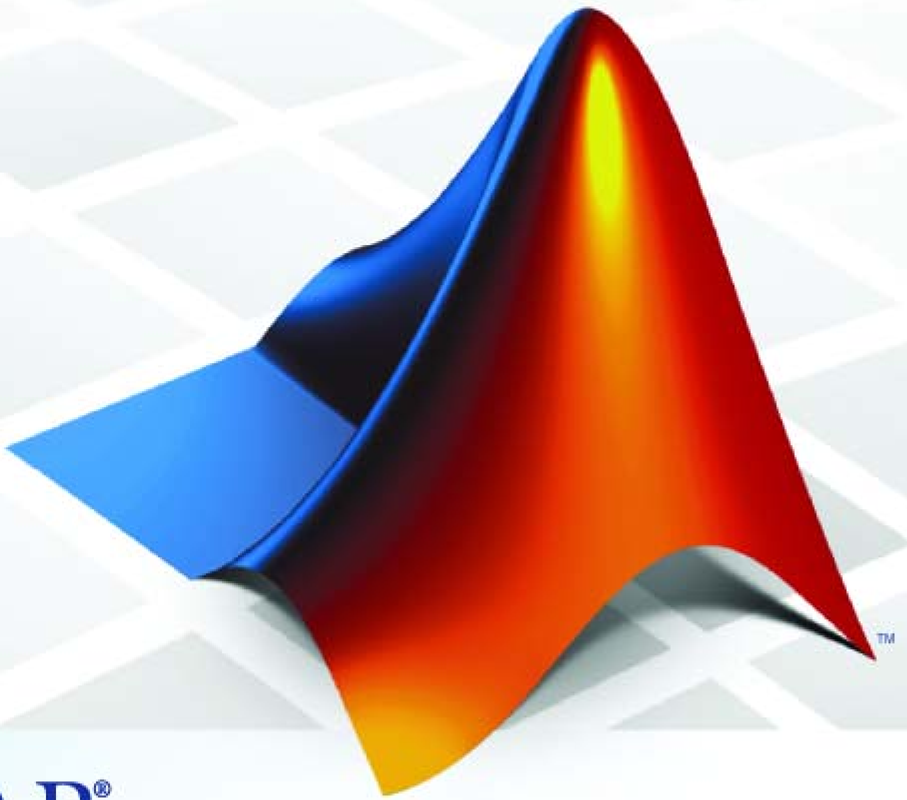
Video and Image Processing Blockset™ 2

Reference



MATLAB® & SIMULINK®

The MathWorks™

Accelerating the pace of engineering and science

How to Contact The MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support
`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

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

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

Video and Image Processing Blockset™ Reference

Revision History

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

Contents

Block Reference

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

Apply Geometric Transformation	Apply projective or affine transformation to an image
Estimate Geometric Transformation	Compute transformation matrix between greatest number of point pairs of two images
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 (Obsolete)	Generate histogram of each input matrix
2-D Mean (Obsolete)	Find mean value of each input matrix

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

Text & Graphics

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

Transforms

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

Utilities

Block Processing	Repeat user-specified operation on submatrices of input matrix
Image Pad	Pad signal along its rows, columns, or both
Variable Selector	Specify subset of rows or columns from input

2

Blocks — Alphabetical List

2-D Autocorrelation

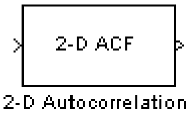
Purpose

Compute 2-D autocorrelation of input matrix

Library

Statistics

Description



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

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

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

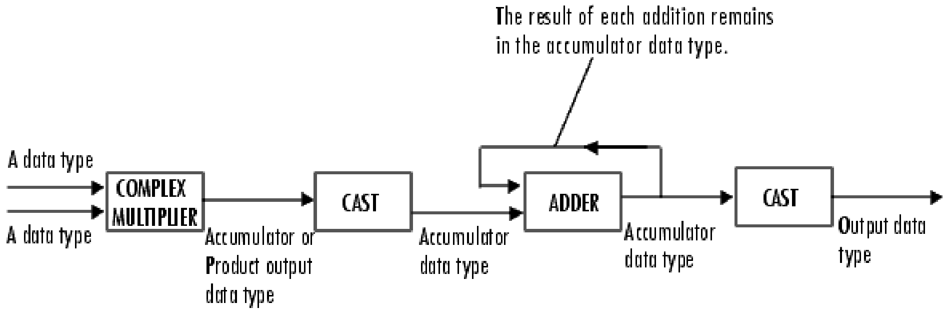
The output of this block has dimensions $(2Ma - 1, 2Na - 1)$.

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	• 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.

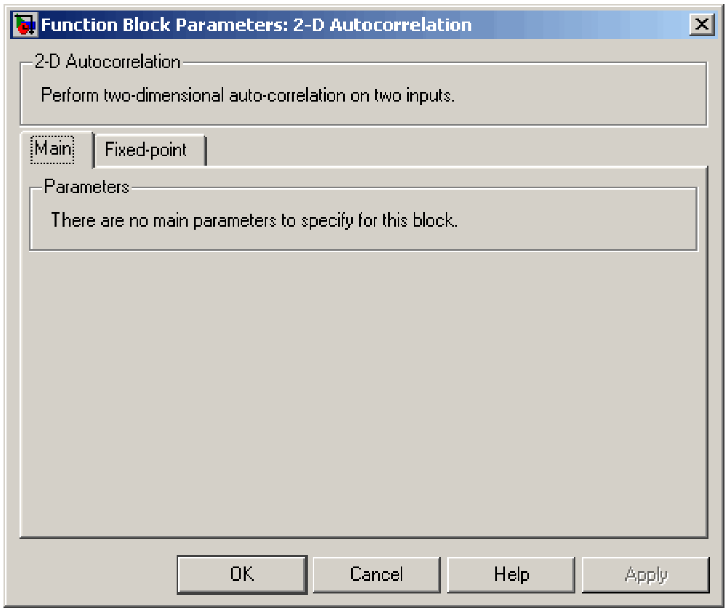


You can set the product output, accumulator, and output data types in the block mask as discussed in "Dialog Box" on page 2-4.

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.

Dialog Box

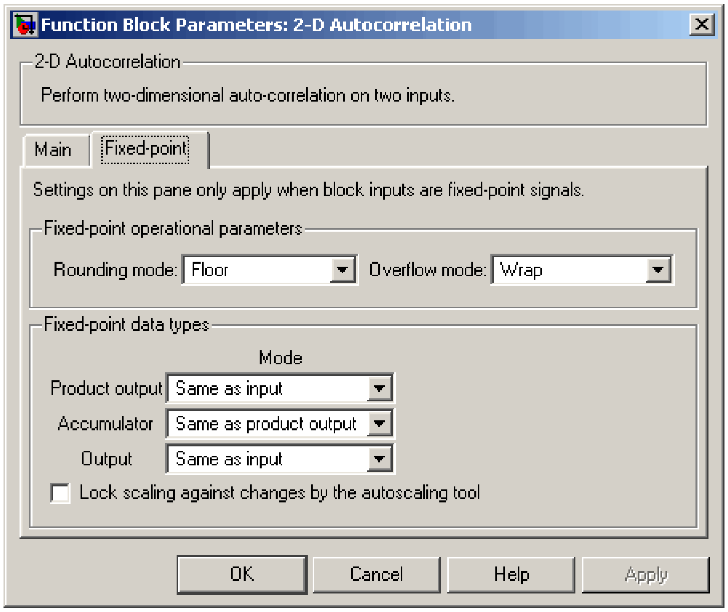
The **Main** pane of the 2-D Autocorrelation dialog box appears as shown in the following figure.



Perform two-dimensional auto-correlation on two inputs.

There are no main parameters to specify for this block.

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


Function Block Parameters: 2-D Autocorrelation

2-D Autocorrelation

Perform two-dimensional auto-correlation on two inputs.

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

Fixed-point operational parameters

Rounding mode: Floor   Overflow mode: Wrap

Fixed-point data types

	Mode
Product output	Same as input
Accumulator	Same as product output
Output	Same as input

☐ Lock scaling against changes by the autoscaling tool

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.

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

Accumulator

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

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

Output

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

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the the Video and Image Processing Blockset software 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 software
Histogram	Video and Image Processing Blockset software
Mean	Video and Image Processing Blockset software
Median	Video and Image Processing Blockset software
Standard Deviation	Video and Image Processing Blockset software
Variance	Video and Image Processing Blockset software
Maximum	Signal Processing Blockset software
Minimum	Signal Processing Blockset software

2-D Convolution

Purpose

Compute 2-D discrete convolution of two input matrices

Library

Filtering

Description

2-D Convolution

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

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	• 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.

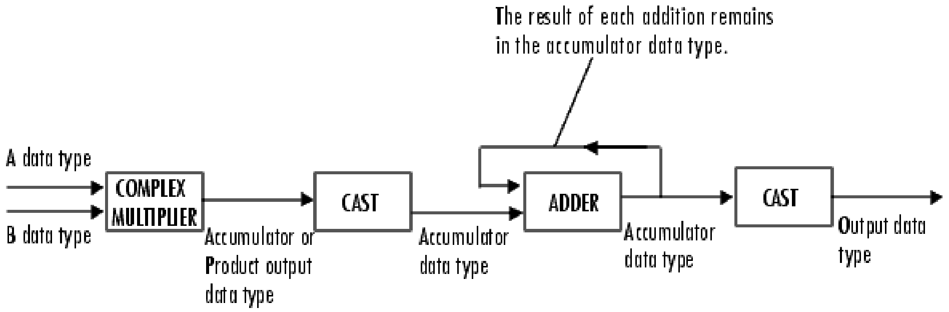
The dimensions of the output are dictated by the **Output size** parameter. Assume that the input at port I1 has dimensions (Ma, Na) and the input at port I2 has dimensions (Mb, Nb). If, for the **Output size** parameter, you choose `Full`, the output is the full two-dimensional convolution with dimensions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

If you select the **Output normalized convolution** check box, the block's output is divided by `sqrt(sum(dot(I1p,I1p))*sum(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.



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:

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

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

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:

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

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

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

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

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 = [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]
```

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

```
I2 = [8    1    6
      3    5    7
      4    9    2]
```

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

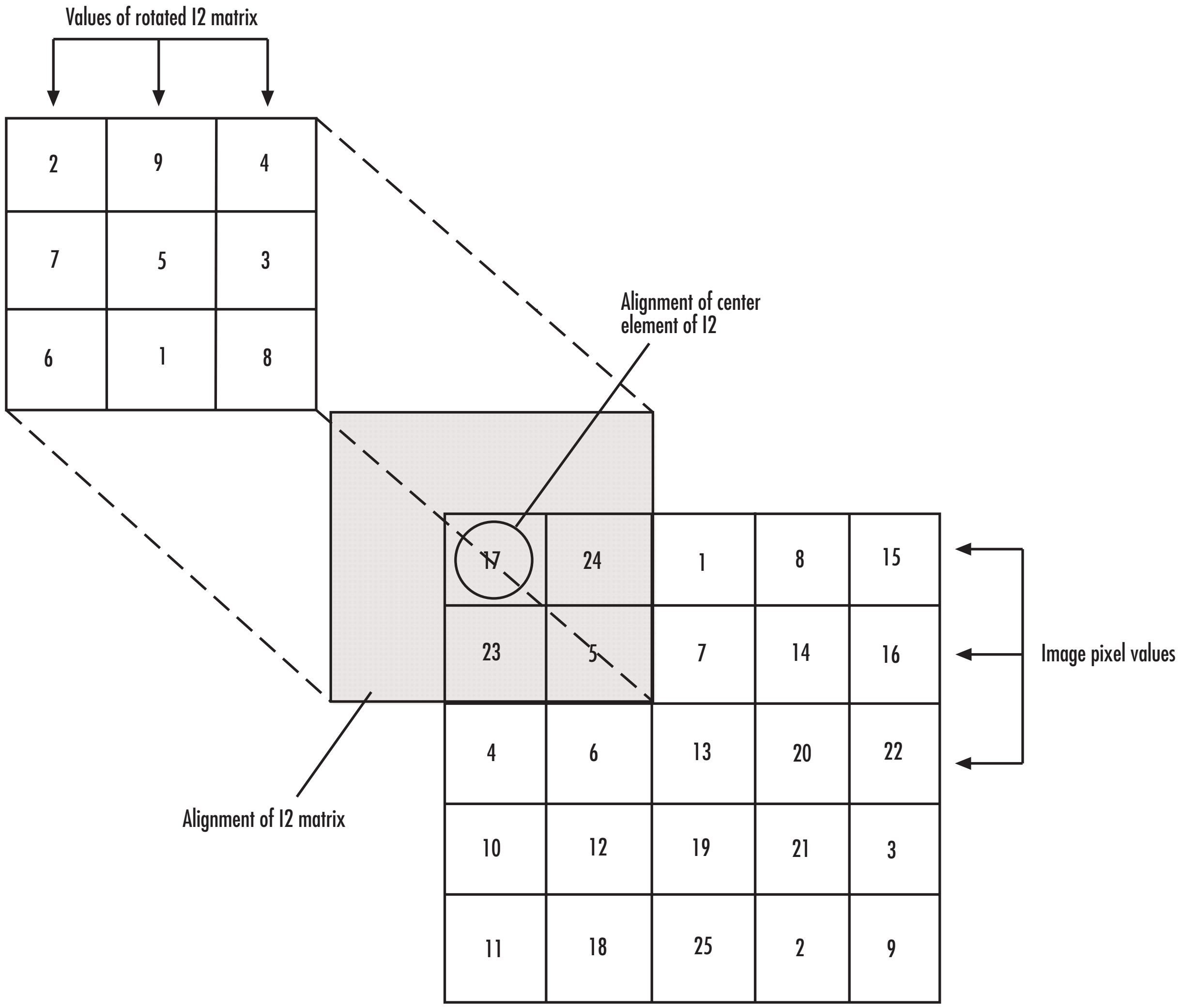
1 Rotate the second input matrix, I2, 180 degrees about its center element.

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

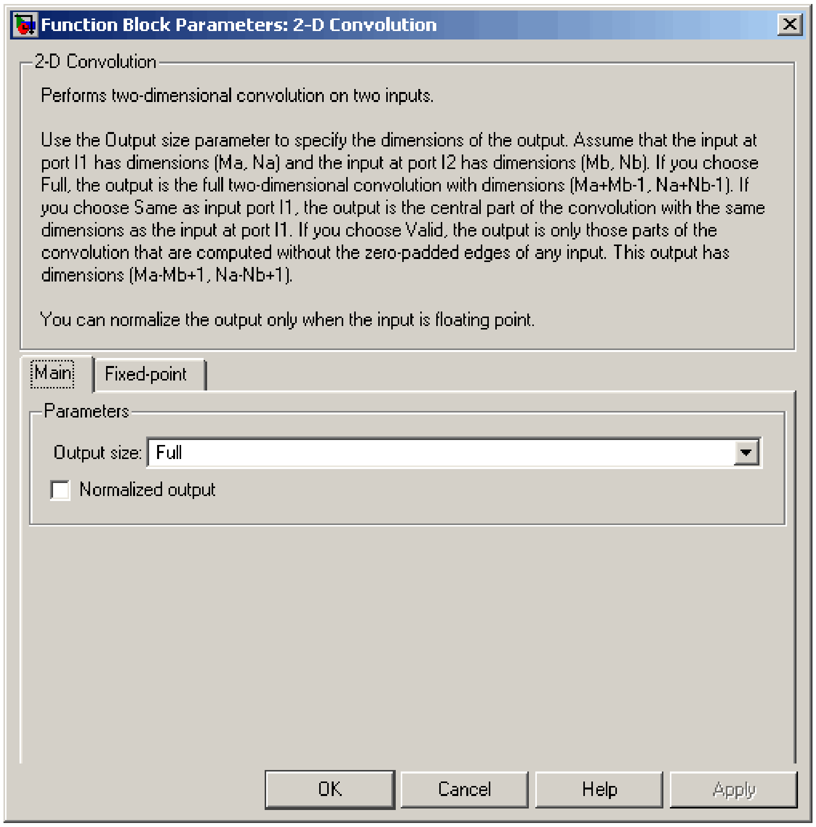


Computing the (1,1) Output of Convolution

The normalized convolution of the (1,1) output element is `220/sqrt(sum(dot(I1p,I1p))*sum(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.


Function Block Parameters: 2-D Convolution

2-D Convolution

Performs two-dimensional convolution on two inputs.

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

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

Main | Fixed-point

Parameters

Output size: Full

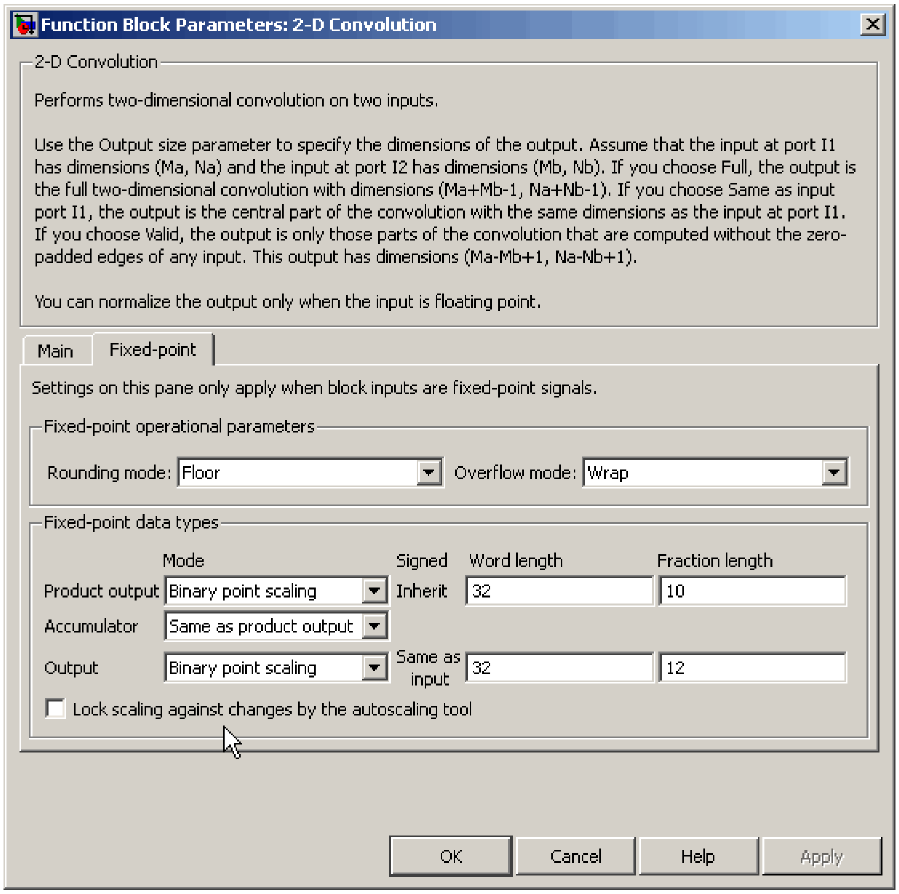
☐ Normalized output

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).

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.


Function Block Parameters: 2-D Convolution

2-D Convolution

Performs two-dimensional convolution on two inputs.

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

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

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

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

☐ Lock scaling against changes by the autoscaling tool

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 theVideo and Image Processing Blockset software is 0.

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

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

Accumulator

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

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

Output

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

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

Lock 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 software

2-D Correlation

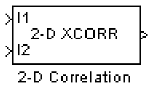
Purpose

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

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

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

Port	Input/Output	Supported Data Types	Complex Values Supported
I1	Vector or matrix of intensity values	• 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

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

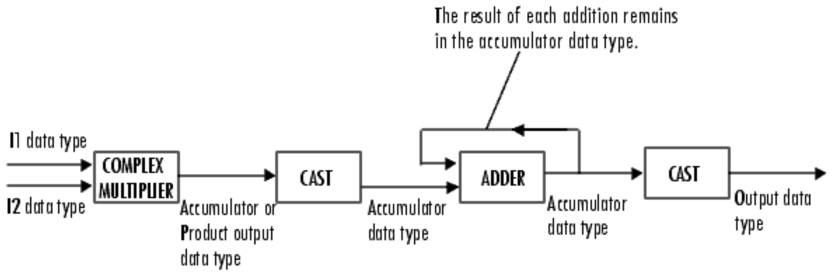
The dimensions of the output are dictated by the **Output size** parameter and the sizes of the inputs at ports I1 and I2. For example, assume that the input at port I1 has dimensions (Ma, Na) and the input at port I2 has dimensions (Mb, Nb). If, for the **Output size** parameter, you choose `Full`,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

If you select the **Normalized output** check box, the block's output is divided by `sqrt(sum(dot(I1p,I1p))*sum(dot(I2,I2)))`, where `I1p` is the portion of the I1 matrix that aligns with the I2 matrix. See "Example 2" on page 2-24 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 2-27.

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:

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

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

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

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

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 = [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]
```

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

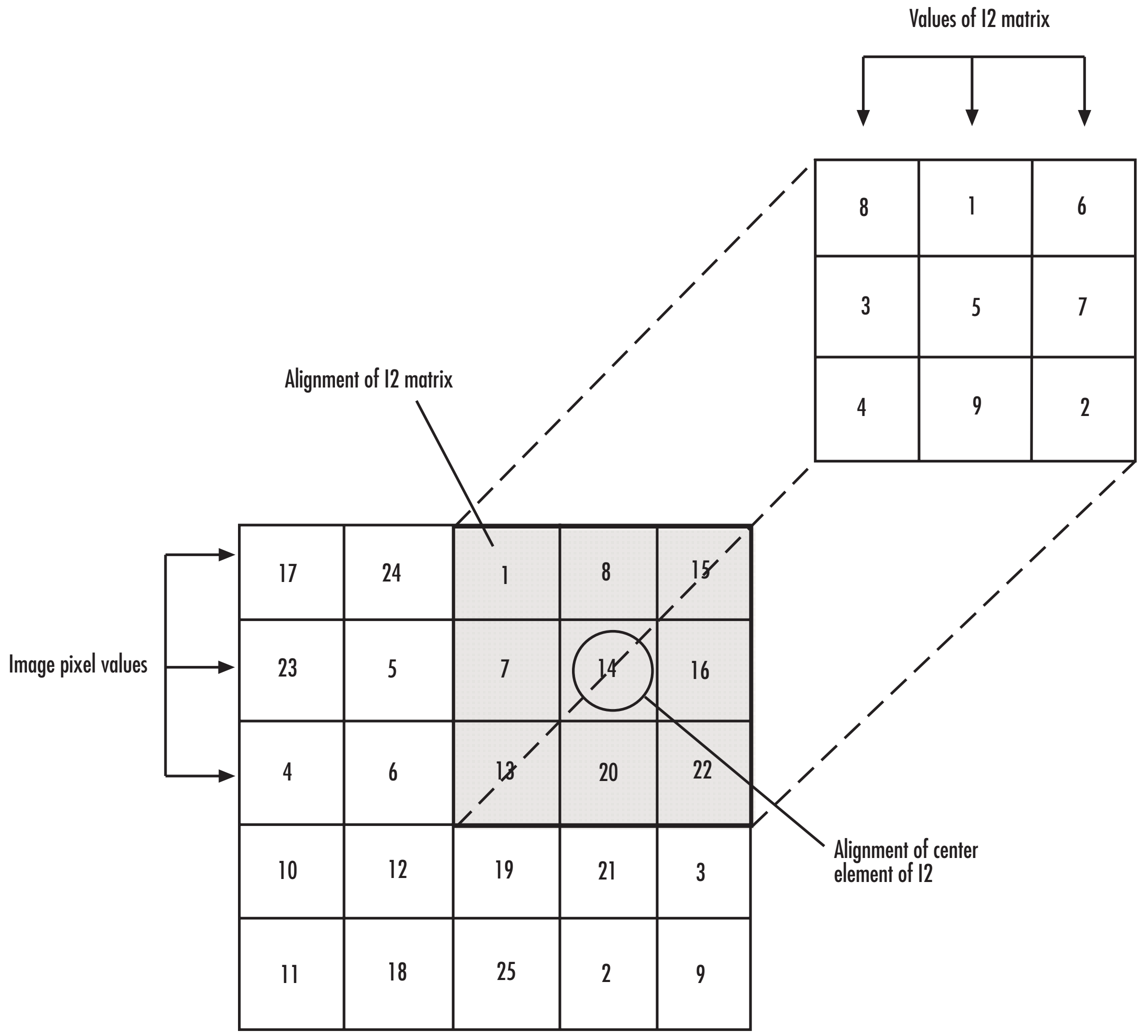
```
I2 = [8    1    6
      3    5    7
      4    9    2]
```

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

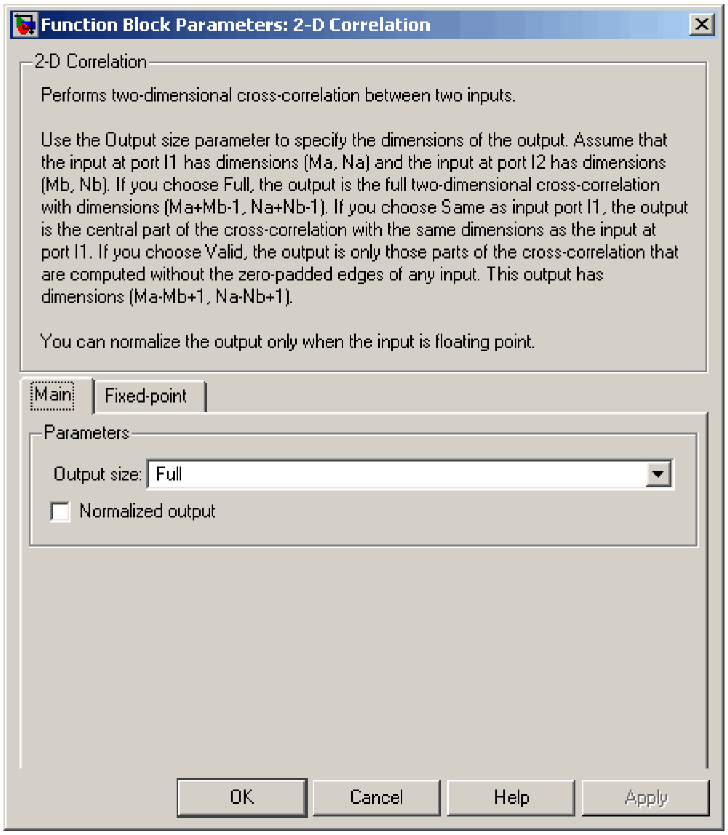


Computing the (2,4) Output of Cross-Correlation

The normalized cross-correlation of the (2,4) output element is `585/sqrt(sum(dot(I1p,I1p))*sum(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.


Function Block Parameters: 2-D Correlation

2-D Correlation

Performs two-dimensional cross-correlation between two inputs.

Use the Output size parameter to specify the dimensions of the output. Assume that the input at port I1 has dimensions (Ma, Na) and the input at port I2 has dimensions (Mb, Nb). If you choose Full, the output is the full two-dimensional cross-correlation with dimensions (Ma+Mb-1, Na+Nb-1). If 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 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).

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

Parameters

Output size: Full

☐ Normalized output

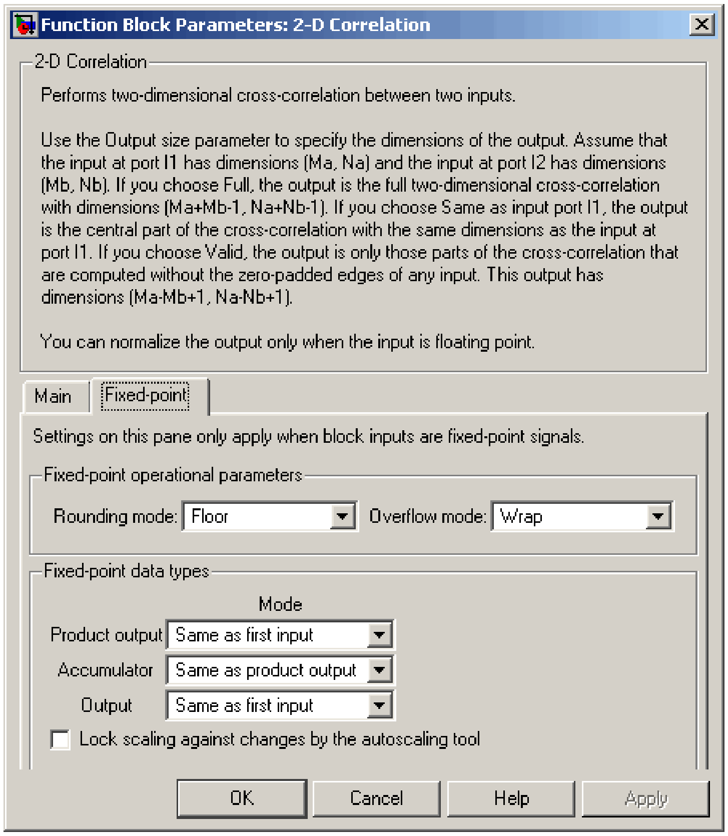
Output size

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 (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).

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.


Function Block Parameters: 2-D Correlation

2-D Correlation

Performs two-dimensional cross-correlation between two inputs.

Use the Output size parameter to specify the dimensions of the output. Assume that the input at port I1 has dimensions (Ma, Na) and the input at port I2 has dimensions (Mb, Nb). If you choose Full, the output is the full two-dimensional cross-correlation with dimensions (Ma+Mb-1, Na+Nb-1). If 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 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).

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

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

	Mode
Product output	Same as first input
Accumulator	Same as product output
Output	Same as first input

☐ Lock scaling against changes by the autoscaling tool

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 2-21 and “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the product output data type in this block:

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

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

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

Accumulator

Use this parameter to specify how to designate the accumulator word and fraction lengths. Refer to “Fixed-Point Data Types” on page 2-21 and “Multiplication Data Types” in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block. The accumulator data type is only used when both inputs to the multiplier are complex:

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

Output

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

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

Lock scaling against changes by the autoscaling tool

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

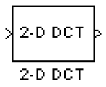
See Also

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

Purpose Compute 2-D discrete cosine transform (DCT)

Library Transforms

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.

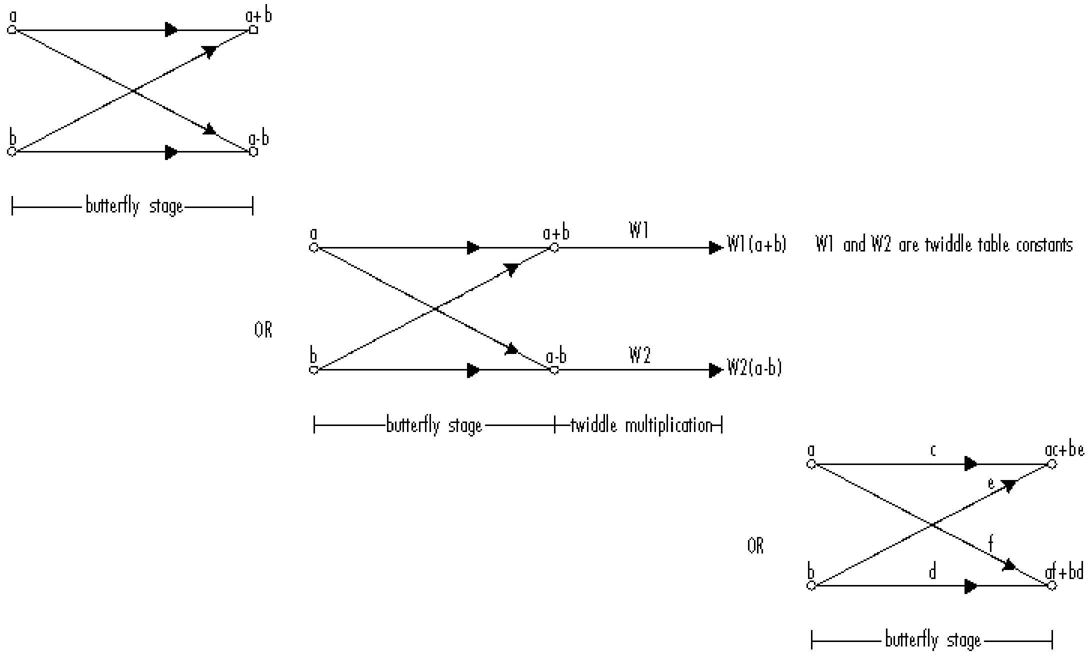
Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer	No
Output	2-D DCT of the input	Same as Input port	No

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

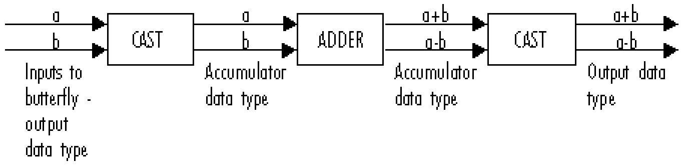
Use the **Sine and cosine computation** parameter to specify how the block computes the sine and cosine terms in the DCT algorithm. If you select `Trigonometric fcn`, the block computes the sine and cosine values during the simulation. If you select `Table lookup`, the block computes and stores the trigonometric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

Fixed-Point Data Types

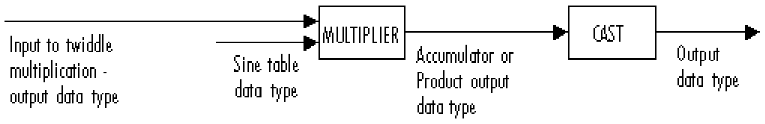
The following diagram shows the data types used in the 2-D DCT block for fixed-point signals. Inputs are first cast to the output data type and stored in the output buffer. Each butterfly stage processes signals in the accumulator data type, with the final output of the butterfly being cast back into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



Butterfly Stage Data Types



Twiddle Multiplication Data Types

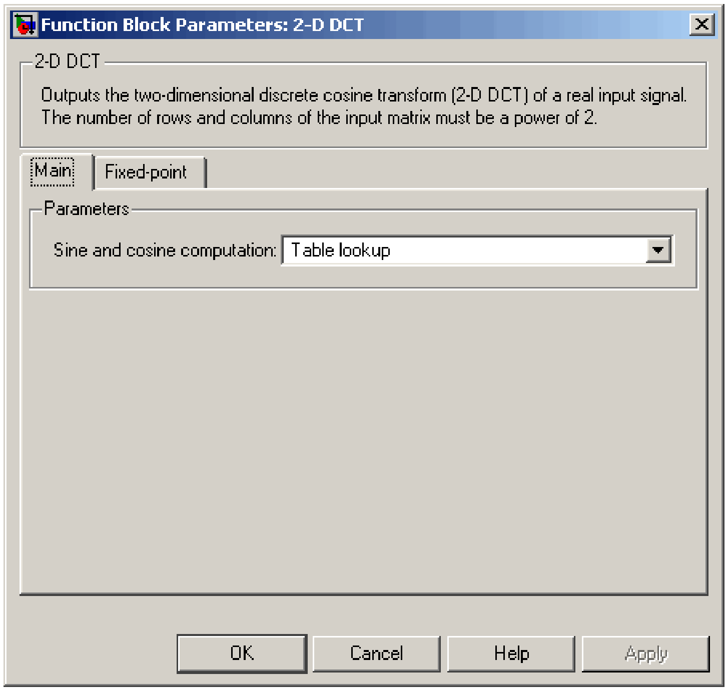


MULTIPLIER
CAST
Input to twiddle multiplication - output data type
Sine table data type
Accumulator or Product output data type
Output data type

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.



Function Block Parameters: 2-D DCT

2-D DCT

Outputs the two-dimensional discrete cosine transform (2-D DCT) of a real input signal. The number of rows and columns of the input matrix must be a power of 2.

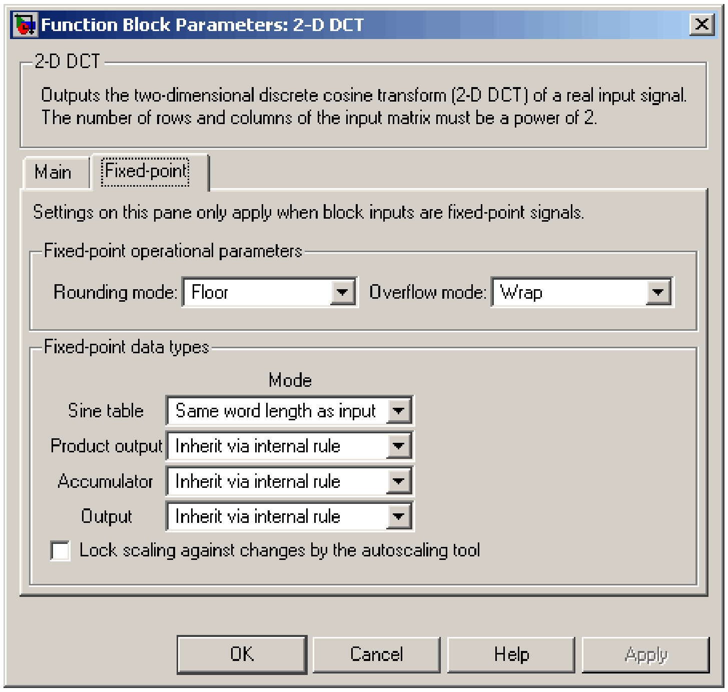
Parameters

Sine and cosine computation: Table lookup

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.


Function Block Parameters: 2-D DCT

2-D DCT

Outputs the two-dimensional discrete cosine transform (2-D DCT) of a real input signal. The number of rows and columns of the input matrix must be a power of 2.

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

	Mode
Sine table	Same word length as input
Product output	Inherit via internal rule
Accumulator	Inherit via internal rule
Output	Inherit via internal rule

☐ Lock scaling against changes by the autoscaling tool

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 2-45 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 the Video and Image Processing Blockset software is 0.

Accumulator

Use this parameter to specify how to designate the accumulator word and fraction lengths. Refer to “Fixed-Point Data Types” on page 2-45 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 the Video and Image Processing Blockset software 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(log 2((M-1)(N-1)))+1*

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

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

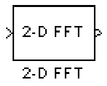
Purpose

Compute 2-D FFT of input

Library

Transforms

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 $0 \le m \le M-1$ and $0 \le n \le N-1$.

Port	Description	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	• 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

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

Optimizing the Table of Trigonometric Values

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

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

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

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

Ordering Output Column Entries

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

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

For more information ordering of the output, see “Bit-Reversed Order” on page 2-46. The 2-D FFT block bit-reverses the order of both the columns and the rows.

Algorithms Used for FFT Computation

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

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

Floating-Point Signals

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

Floating-Point Signals (Continued)

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

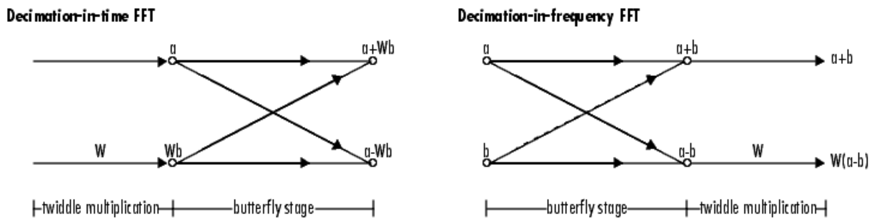
Fixed-Point Signals

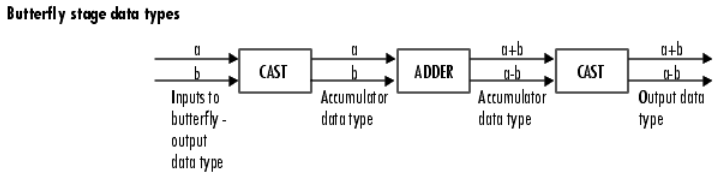
Complexity of Input	Output Ordering	Algorithms Used for FFT Computation
Real or complex	Linear	Bit-reversal operation and radix-2 DIT
Real or complex	Bit-reversed	Radix-2 DIF

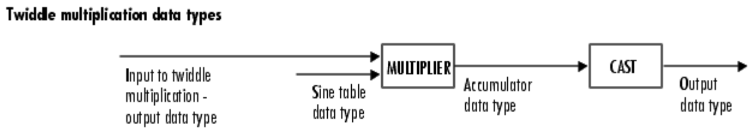
Fixed-Point Data Types

The following diagrams show the data types used in the 2-D FFT block for fixed-point signals. You can set the sine table, accumulator, product output, and output data types displayed in the diagrams in the 2-D FFT dialog box as discussed in "Dialog Box" on page 2-49.

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






Twiddle multiplication data types
Input to twiddle multiplication - output data type
Sine table data type
MULTIPLIER
Accumulator data type
CAST
Output data type

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

Example

Bit-Reversed Order

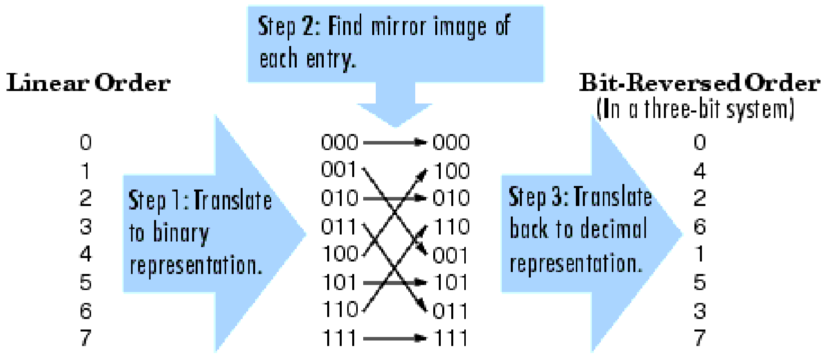
Two numbers are bit-reversed values of each other when the binary representation of one is the mirror image of the binary representation of the other. For example, in a three-bit system, one and four are bit-reversed values of each other because the three-bit binary representation of one, 001, is the mirror image of the three-bit binary representation of four, 100. The following diagram shows the row indices in linear order. To put them in bit-reversed order

1. Translate the indices into their binary representation with the minimum number of bits. In this example, the minimum number of bits is three because the binary representation of 7 is 111.

2. Find the mirror image of each binary entry, and write it beside the original binary representation.

3. Translate the indices back to their decimal representation.

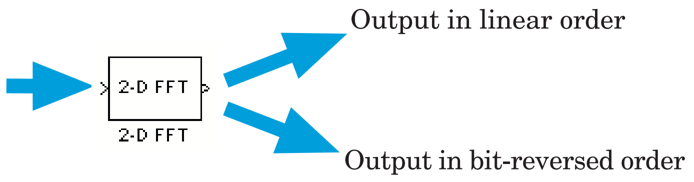
   The row indices now appear in bit-reversed order.



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

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

Linearly ordered row and column indices

$$\begin{array}{c|cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 0 \rightarrow & 245 & 13.9-0.4i & 10-5i & -15.9+21.6i & -13 & -15.9-21.6i & 10+5i & 13.9 \\ 1 \rightarrow & -4.3-10.3i & -27.6-6.6i & -5.6+13.1i & -3.4+8.7i & 1.1-i & -2.6i & -11.5-11i & 6.2+13i \\ 2 \rightarrow & 18-5i & -4.3-10.4i & 19-24i & 12.4-11.4i & 6-3i & -5.7+16.4i & 5+4i & 5.5+1.4i \\ 3 \rightarrow & 8.4-2.4i & -0.6+2.7i & -4.5+1.1i & 17.6+9.4i & 11-9i & -2.2-13i & -18.4+25.1i & 34+0.5i \\ 4 \rightarrow & -9 & 16.3+5.9i & 14-31i & 17.7+23.9i & 1 & 17.7-23.9i & 14+31i & 16.3-5.9i \\ 5 \rightarrow & 8.4+2.4i & 3.4-5.4i & -18.4-25.1i & -2.2+13.1i & 11+9i & 17.6-9.4i & -4.5-1.1i & -1-2.7i \\ 6 \rightarrow & 18+5i & 5.5-1.4i & 5-4i & -5.7-16.4i & 6+3i & 12.5+11.3i & 19+24i & -4.3+10.4i \\ 7 \rightarrow & -4.4+10.3i & 6.2-13i & -11.5+11i & 2.6i & 1.1+i & -3.4-8.7i & -5.6-13.1i & -27.6+6.6i \end{array}$$

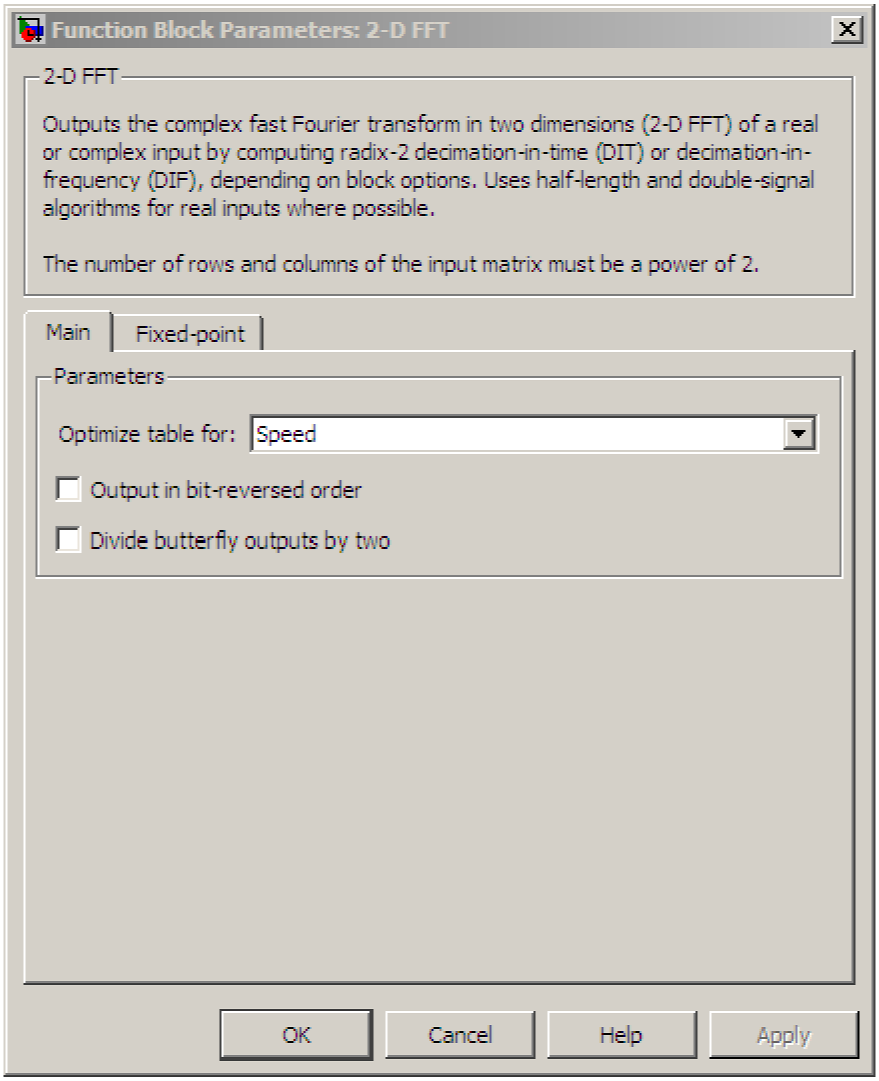
Output in bit-reversed order

Bit-reversed row and column indices

$$\begin{array}{c|cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 0 \rightarrow & 245 & -13 & 10-5i & 10+5i & 13.9-0.4i & -15.9-21.6i & -15.9+21.6i & 13.9 \\ 1 \rightarrow & -9 & 1 & 14-31i & 14+31i & 16.3+5.9i & 17.7-23.9i & 17.7+23.9i & 16.3-5.9i \\ 2 \rightarrow & 18-5i & 6-3i & 19-24i & 5+4i & -4.3-10.4i & -5.7+16.4i & 12.4-11.4i & 5.5+1.4i \\ 3 \rightarrow & 18+5i & 6+3i & 5-4i & 19+24i & 5.5-1.4i & 12.5+11.3i & -5.7-16.4i & 34+0.5i \\ 4 \rightarrow & -4.3-10.3i & 1.1-i & -5.6+13.1i & -11.5-11i & -27.6-6.6i & -2.6i & -3.4+8.7i & 6.2+13i \\ 5 \rightarrow & 8.4+2.4i & 11+9i & -18.4-25.1i & -4.5-1.1i & 3.4-5.4i & 17.6-9.4i & -2.2+13i & -1-2.7i \\ 6 \rightarrow & 8.4-2.4i & 11-9i & -4.5+1.1i & -18.4+25.1i & -0.6+2.7i & -2.2-13i & 17.6+9.4i & 34+0.5i \\ 7 \rightarrow & -4.4+10.3i & 1.1+i & -11.5+11i & -5.6-13.1 & 6.2-13i & -3.4-8.7i & 2.6i & -27.6+6.6i \end{array}$$

Dialog Box

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



Function Block Parameters: 2-D FFT

2-D FFT

Outputs the complex fast Fourier transform in two dimensions (2-D FFT) of a real or complex input by computing radix-2 decimation-in-time (DIT) or decimation-in-frequency (DIF), depending on block options. Uses half-length and double-signal algorithms for real inputs where possible.

The number of rows and columns of the input matrix must be a power of 2.

Optimize table for: Speed

☐ Output in bit-reversed order

☐ Divide butterfly outputs by two

Optimize table for

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

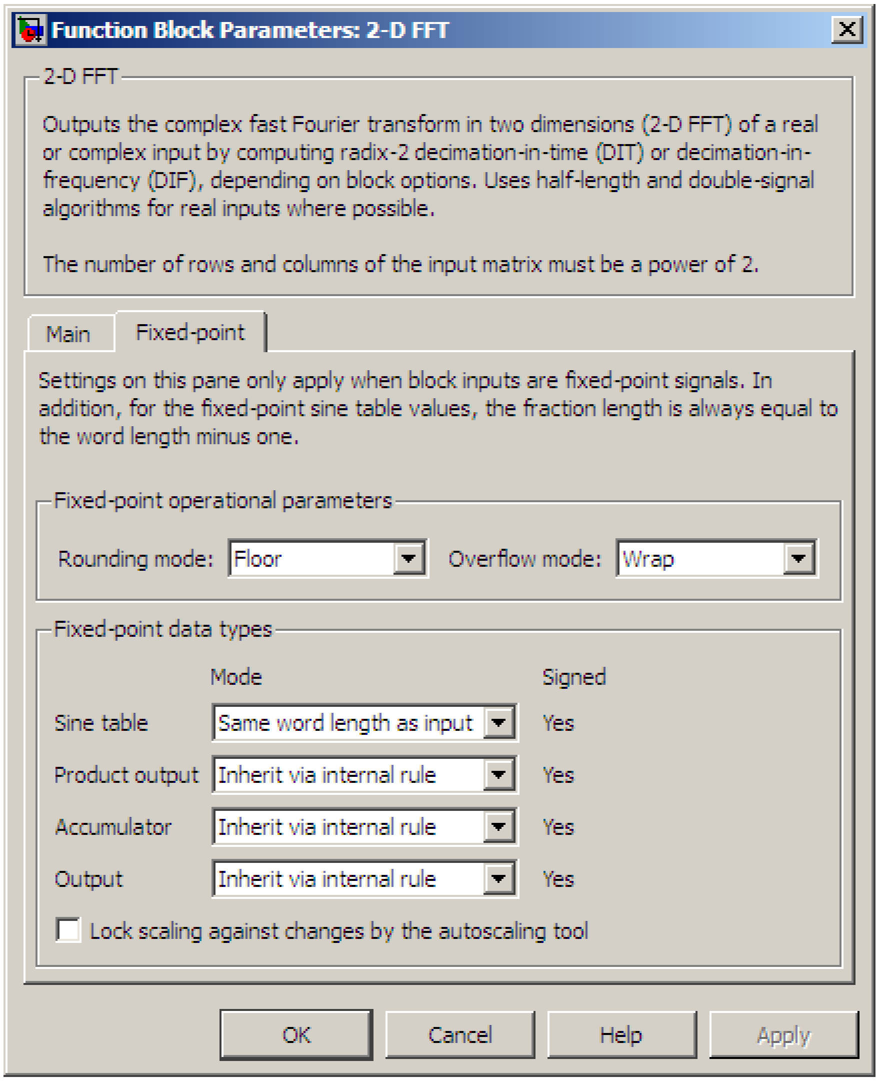
Output in bit-reversed order

Designate the order of the output channel elements relative to the ordering of the input elements. When selected, the output channel elements appear in bit-reversed order relative to the input ordering. Otherwise, the output column elements display in linear order relative to the input ordering. Linearly ordering the output requires extra data sorting manipulation. For more information, see “Bit-Reversed Order” on page 2-46.

Divide butterfly outputs by two

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

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



Function Block Parameters: 2-D FFT

2-D FFT

Outputs the complex fast Fourier transform in two dimensions (2-D FFT) of a real or complex input by computing radix-2 decimation-in-time (DIT) or decimation-in-frequency (DIF), depending on block options. Uses half-length and double-signal algorithms for real inputs where possible.

The number of rows and columns of the input matrix must be a power of 2.

Main | Fixed-point

Settings on this pane only apply when block inputs are fixed-point signals. In addition, for the fixed-point sine table values, the fraction length is always equal to the word length minus one.

Fixed-point operational parameters

Rounding mode: Floor   Overflow mode: Wrap

Fixed-point data types

	Mode	Signed
Sine table	Same word length as input	Yes
Product output	Inherit via internal rule	Yes
Accumulator	Inherit via internal rule	Yes
Output	Inherit via internal rule	Yes

☐ Lock scaling against changes by the autoscaling tool

Rounding mode

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

Overflow mode

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

Sine table

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

- When you select `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 2-45 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 the Video and Image Processing Blockset software is 0.

Accumulator

Use this parameter to specify how to designate the accumulator word and fraction lengths. Refer to "Fixed-Point Data Types" on page 2-45 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 the Video and Image Processing Blockset software 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(log 2((M-1)(N-1)))+1*

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

  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 the Video and Image Processing Blockset software is 0.

See Also

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

2-D FIR Filter

Purpose

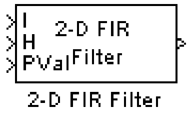
Perform 2-D FIR filtering on input matrix

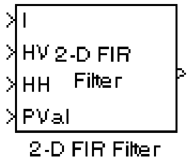
Library

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	• 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

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

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

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

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

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

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

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

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

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

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

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

The block outputs the result of the filtering operation at the Output port. The `Output size` parameter and the sizes of the inputs at ports **I** and **H** dictate the dimensions of the output. For example, assume that the input at port I has dimensions (Mi, Ni) and the input at port H has dimensions (Mh, Nh). If you select an**Output size** of `Full`, the output has dimensions (Mi+Mh-1, Ni+Nh-1). If you select an **Output size** of `Same as input port I`, the output has the same dimensions as the input at port I. If you select an **Output size** of `Valid`, the block filters the input image only where the coefficient matrix fits entirely within it, so no padding is required. The output has dimensions (Mi-Mh+1, Ni-Nh+1). However, if `all(size(I)<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 repetition of its elements, select `Circular`. For more information on padding, see the Image Pad block reference page.

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

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

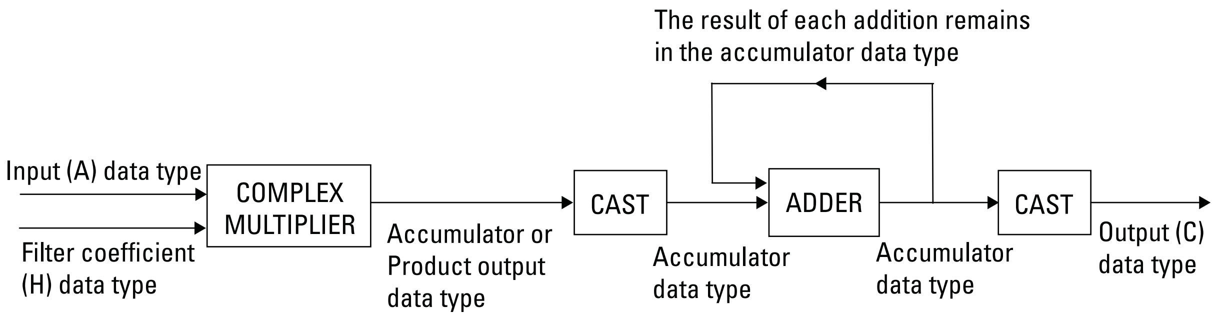
$$C(i,j) = \sum_{m=0}^{(Ma-1)} \sum_{n=0}^{(Na-1)} A(m,n) \cdot conj(H(m+i, n+j))$$

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

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

Fixed-Point Data Types

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

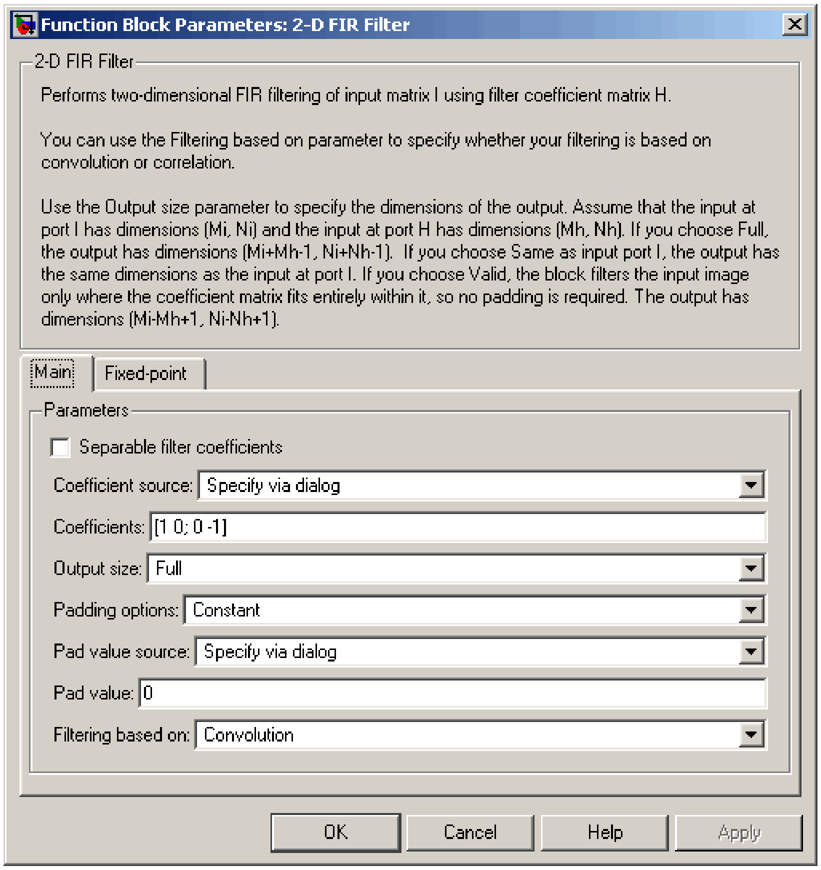

The result of each addition remains in the accumulator data type
Input (A) data type
Filter coefficient (H) data type
COMPLEX MULTIPLIER
Accumulator or Product output data type
CAST
Accumulator data type
ADDER
Accumulator data type
CAST
Output (C) data type

You can set the coefficient, product output, accumulator, and output data types in the block mask as discussed in “Dialog Box” on page 2-61.

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

Dialog Box

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



Function Block Parameters: 2-D FIR Filter

2-D FIR Filter

Performs two-dimensional FIR filtering of input matrix I using filter coefficient matrix H.

You can use the Filtering based on parameter to specify whether your filtering is based on convolution or correlation.

Use the Output size parameter to specify the dimensions of the output. Assume that the input at port I has dimensions (Mi, Ni) and the input at port H has dimensions (Mh, Nh). If you choose Full, the output has dimensions (Mi+Mh-1, Ni+Nh-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, the block filters the input image only where the coefficient matrix fits entirely within it, so no padding is required. The output has dimensions (Mi-Mh+1, Ni-Nh+1).

Main | Fixed-point

Parameters

☐ Separable filter coefficients

Coefficient source: Specify via dialog

Coefficients: [1 0; 0 -1]

Output size: Full

Padding options: Constant

Pad value source: Specify via dialog

Pad value: 0

Filtering based on: Convolution

Separable filter coefficients

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

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

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

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

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

Output size
This parameter controls the size of the filtered output. If you choose `Full`, the output has dimensions (Ma+Mh-1, Na+Nh-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 (Ma-Mh+1, Na-Nh+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 appears if you select an **Output size** of `Full` or `Same as input port I`.

Pad value source

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

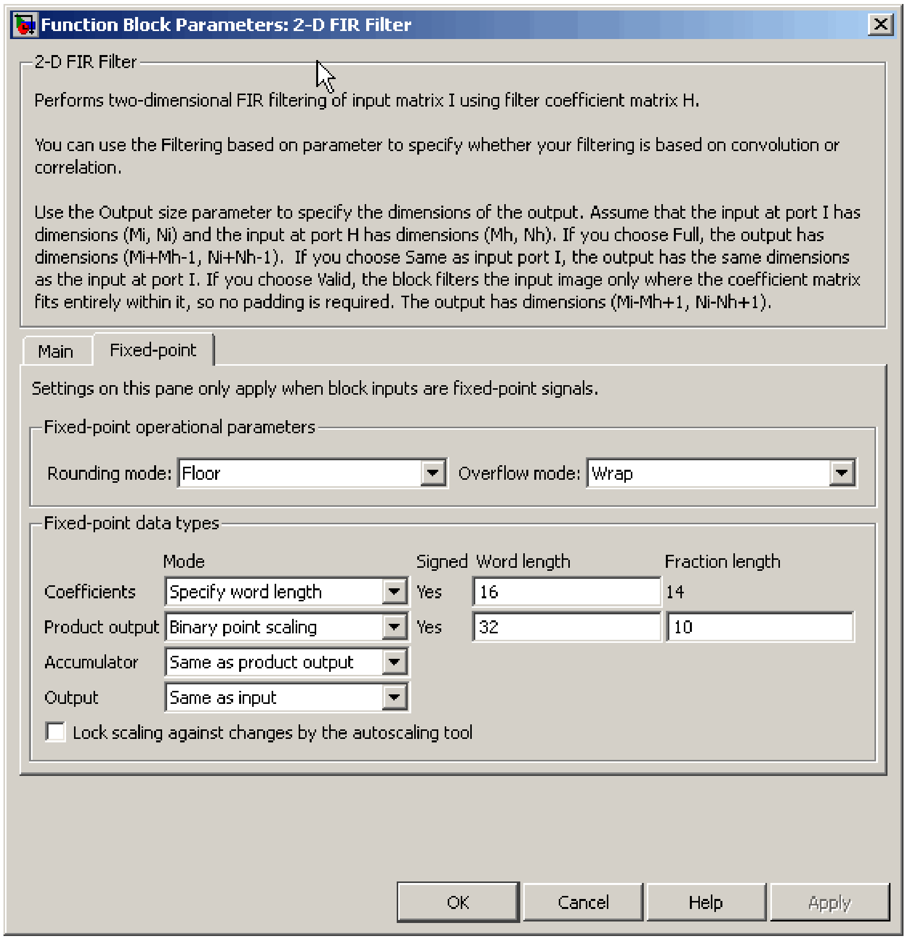
Pad value

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

Filtering based on

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

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



Function Block Parameters: 2-D FIR Filter

2-D FIR Filter

Performs two-dimensional FIR filtering of input matrix I using filter coefficient matrix H.

You can use the Filtering based on parameter to specify whether your filtering is based on convolution or correlation.

Use the Output size parameter to specify the dimensions of the output. Assume that the input at port I has dimensions (Mi, Ni) and the input at port H has dimensions (Mh, Nh). If you choose Full, the output has dimensions (Mi+Mh-1, Ni+Nh-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, the block filters the input image only where the coefficient matrix fits entirely within it, so no padding is required. The output has dimensions (Mi-Mh+1, Ni-Nh+1).

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

	Mode	Signed	Word length	Fraction length
Coefficients	Specify word length	Yes	16	14
Product output	Binary point scaling	Yes	32	10
Accumulator	Same as product output			
Output	Same as input			

☐ Lock scaling against changes by the autoscaling tool

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Coefficients

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

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

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

Product output

Use this parameter to specify how to designate the product output word and fraction lengths. Refer to "Fixed-Point Data Types" on page 2-59 and "Multiplication Data Types" in the Signal Processing Blockset documentation for illustrations depicting the use of the product output data type in this block:

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

- When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the product output. All signals in the Video and Image Processing Blockset software have a bias of 0.

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

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

Accumulator

Use this parameter to specify how to designate the accumulator word and fraction lengths. Refer to "Fixed-Point Data Types" on page 2-59 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. All signals in the Video and Image Processing Blockset software have a bias of 0.

Output

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

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

Lock 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

imfilter	Image Processing Toolbox

2-D Histogram (Obsolete)

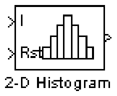
Purpose

Generate histogram of each input matrix

Library

Statistics

Description



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

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

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

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

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

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

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

Port	Input/Output	Supported Data Types	Complex Values Supported
Input / I	Vector or matrix of intensity values	• 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	• 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

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

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

The block outputs `[3 3 3 0]`.

If you select the **Normalized** check box, the block scales each element of the output so that `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.

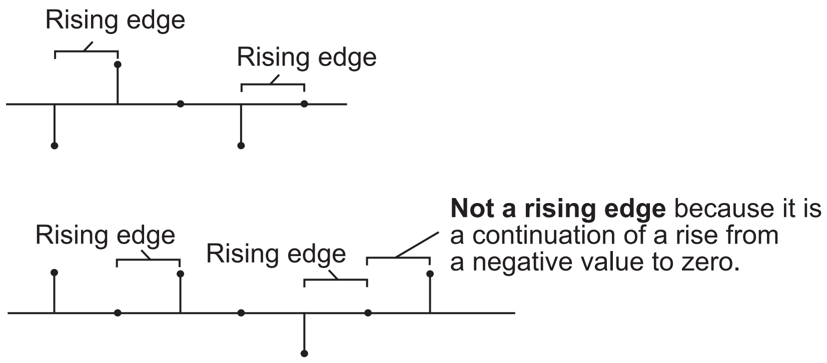
Resetting the Running Histogram

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.

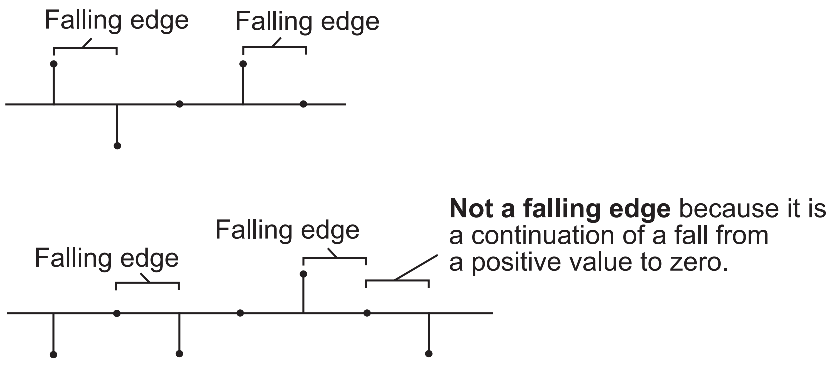
To enable the Rst port, select the **Reset port** parameter. You specify the reset event in the **Trigger type** parameter, and can be one of the following:

- `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)


Rising edge
Rising edge
Rising edge
Rising edge
Not a rising edge because it is a continuation of a rise from a negative value to zero.

- `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)


Falling edge
Falling edge
Falling edge
Falling edge
Not a falling edge because it is a continuation of a fall from a positive value to zero.

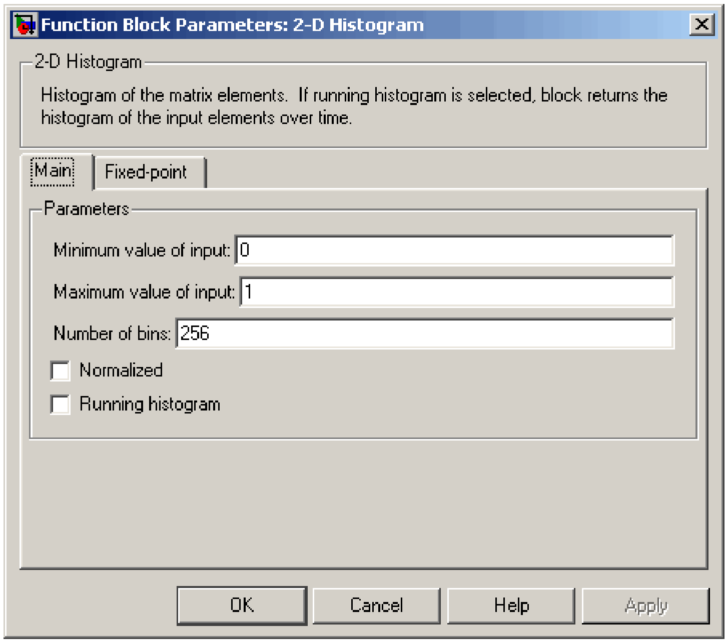
- `Either edge` – Triggers a reset operation when the `Rst` input is a `Rising edge` or `Falling edge` (as described previously)

- `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, 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:


Function Block Parameters: 2-D Histogram

2-D Histogram

Histogram of the matrix elements. If running histogram is selected, block returns the histogram of the input elements over time.

Parameters

Minimum value of input: 0

Maximum value of input: 1

Number of bins: 256

☐ Normalized

☐ Running histogram

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.

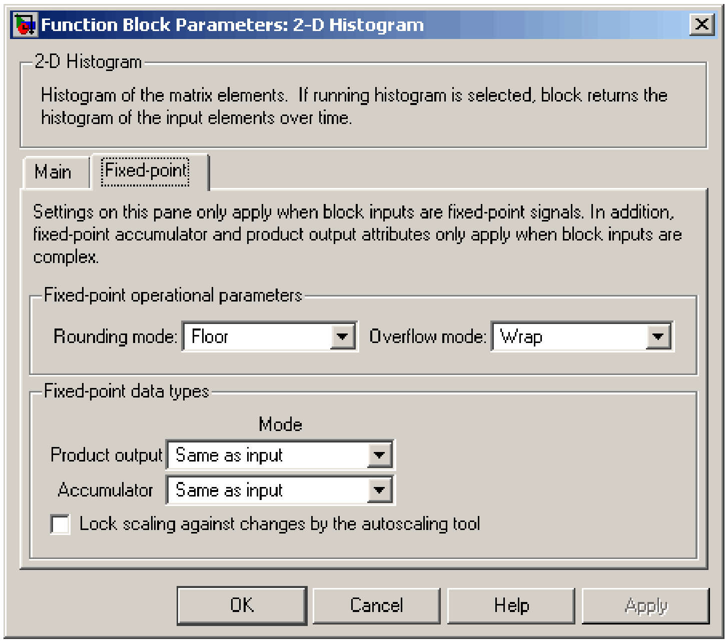
Reset port

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 2-71. This parameter is enabled only when you set the **Reset port** parameter.

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


Function Block Parameters: 2-D Histogram

2-D Histogram

Histogram of the matrix elements. If running histogram is selected, block returns the histogram of the input elements over time.

Main | Fixed-point

Settings on this pane only apply when block inputs are fixed-point signals. In addition, fixed-point accumulator and product output attributes only apply when block inputs are complex.

Fixed-point operational parameters

Rounding mode: Floor Overflow mode: Wrap

Fixed-point data types

	Mode
Product output	Same as input
Accumulator	Same as input

☐ Lock scaling against changes by the autoscaling tool

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

Rounding mode
Select the rounding mode for fixed-point operations.

Overflow mode
Select the overflow mode for fixed-point operations.

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

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

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

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

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

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

See Also

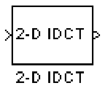
2-D Autocorrelation	Video and Image Processing Blockset software
2-D Correlation	Video and Image Processing Blockset software

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

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

Library Transforms

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 $C(m),C(n) = 1/\sqrt{2}$ 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.

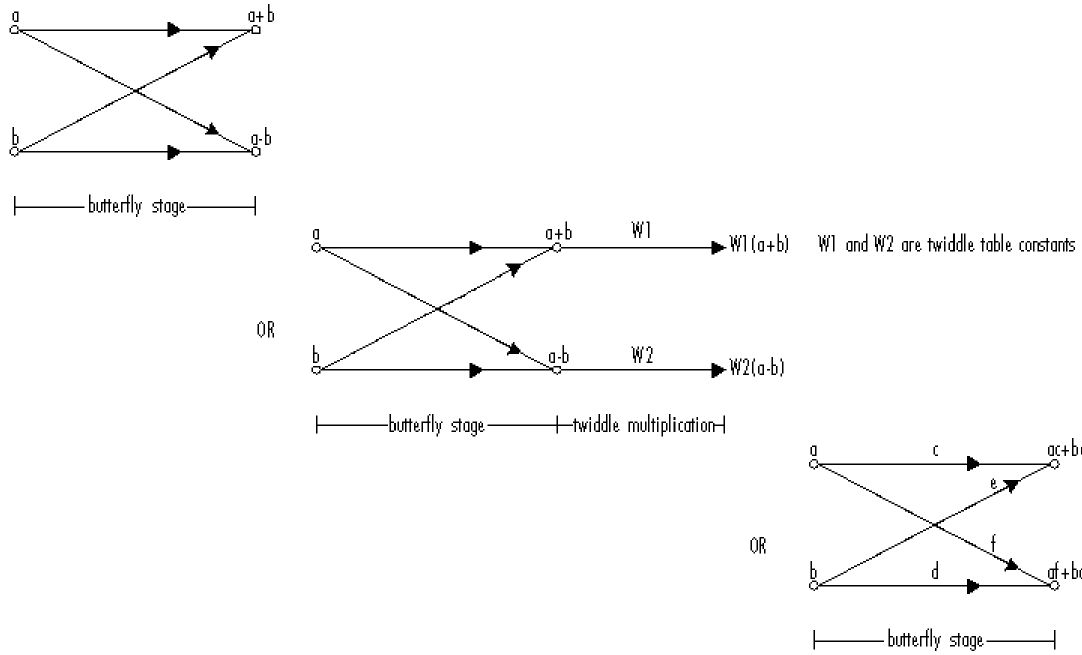
Port	**Input/Output**	**Supported Data Types**	**Complex Values Supported**
Input	Vector or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer	No
Output	2-D IDCT of the input	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.

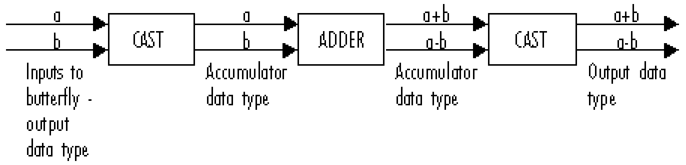
Use the **Sine and cosine computation** parameter to specify how the block computes the sine and cosine terms in the IDCT algorithm. If you select `Trigonometric fcn`, the block computes the sine and cosine values during the simulation. If you select `Table lookup`, the block computes and stores the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

Fixed-Point Data Types

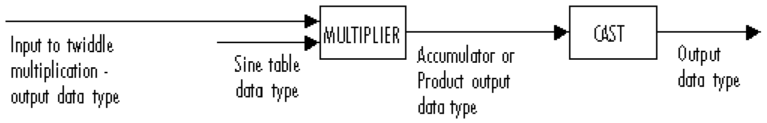
The following diagram shows the data types used in the 2-D IDCT block for fixed-point signals. Inputs are first cast to the output data type and stored in the output buffer. Each butterfly stage processes signals in the accumulator data type, with the final output of the butterfly being cast back into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



Butterfly Stage Data Types



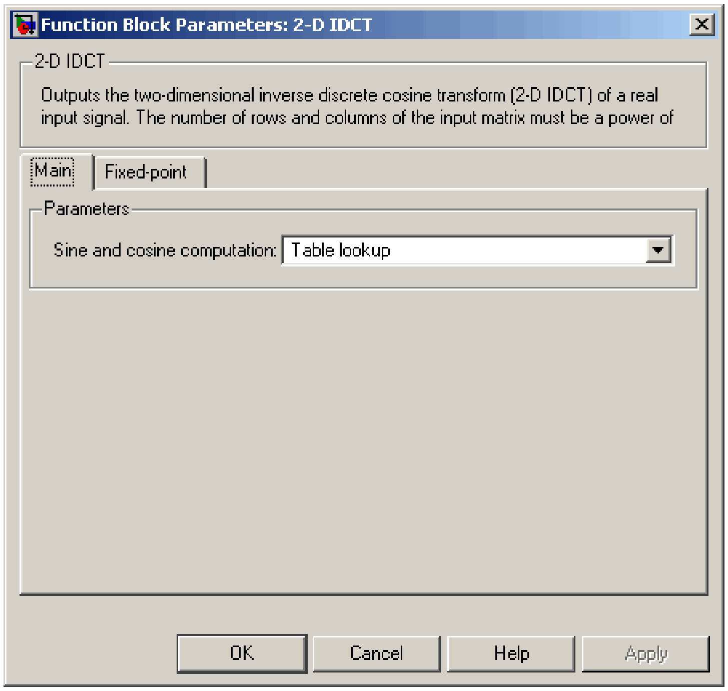
Twiddle Multiplication Data Types


MULTIPLIER
CAST
Input to twiddle multiplication - output data type
Sine table data type
Accumulator or Product output data type
Output data type

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.



Function Block Parameters: 2-D IDCT

2-D IDCT

Outputs the two-dimensional inverse discrete cosine transform (2-D IDCT) of a real input signal. The number of rows and columns of the input matrix must be a power of

Parameters

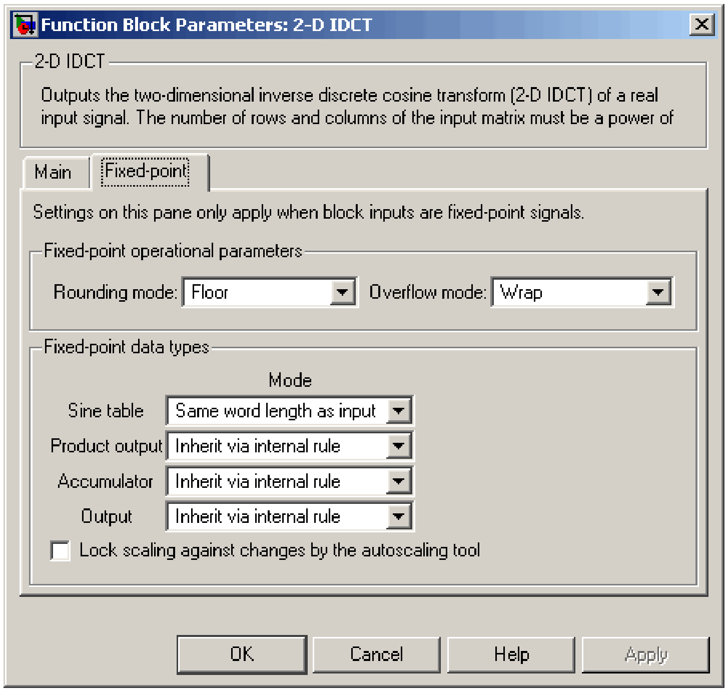
Sine and cosine computation: Table lookup

]

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.



Function Block Parameters: 2-D IDCT

2-D IDCT

Outputs the two-dimensional inverse discrete cosine transform (2-D IDCT) of a real input signal. The number of rows and columns of the input matrix must be a power of

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

	Mode
Sine table	Same word length as input
Product output	Inherit via internal rule
Accumulator	Inherit via internal rule
Output	Inherit via internal rule

☐ Lock scaling against changes by the autoscaling tool

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 2-79 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:

$$\begin{aligned}&\mathit{ideal\ product\ output\ word\ length}\ =\\&\mathit{output\ word\ length} + \mathit{sine\ table\ values\ word\ length}\end{aligned}$$

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

- 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 the Video and Image Processing Blockset software is 0.

Accumulator

Use this parameter to specify how to designate the accumulator word and fraction lengths. Refer to "Fixed-Point Data Types" on page 2-79 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 the Video and Image Processing Blockset software 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(log 2((M-1)(N-1)))+1*

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

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

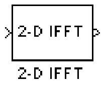
Purpose

Compute 2-D IFFT of input

Library

Transforms

Description



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

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

```
y = 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 $0 \le x \le M-1$ and $0 \le y \le N-1$.

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

Port	Description	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer	Yes
Output	2-D IFFT of the input	Same as Input port	Yes

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

Optimizing the Table of Trigonometric Values

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

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

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

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

Input Order

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

For more information ordering of the output, see "Bit-Reversed Order" on page 2-46. The 2-D FFT block bit-reverses the order of both the columns and the rows..

Conjugate Symmetric Input

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

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

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

Scaled Output

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

The unscaled IFFT is defined by the following equation:

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

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

The scaled version of the IFFT multiplies the above unscaled version by $\frac{1}{MN}$.

Algorithms Used for IFFT Computation

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

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

Algorithms for Floating-Point Signals

Input Complexity	Other Parameter Settings	Algorithms Used for IFFT Computation
Real or complex	☐ Input is in bit-reversed order ☐ Input is conjugate symmetric	Butterfly operation and radix-2 DIT
Real or complex	☑ Input is in bit-reversed order ☐ Input is conjugate symmetric	Radix-2 DIF
Real or complex	☐ Input is in bit-reversed order ☑ Input is conjugate symmetric	Butterfly operation and radix-2 DIT in conjunction with the half-length and double-signal algorithms
Real or complex	☑ Input is in bit-reversed order ☑ Input is conjugate symmetric	Radix-2 DIF in conjunction with the half-length and double-signal algorithms

Algorithms for Fixed-Point Signals

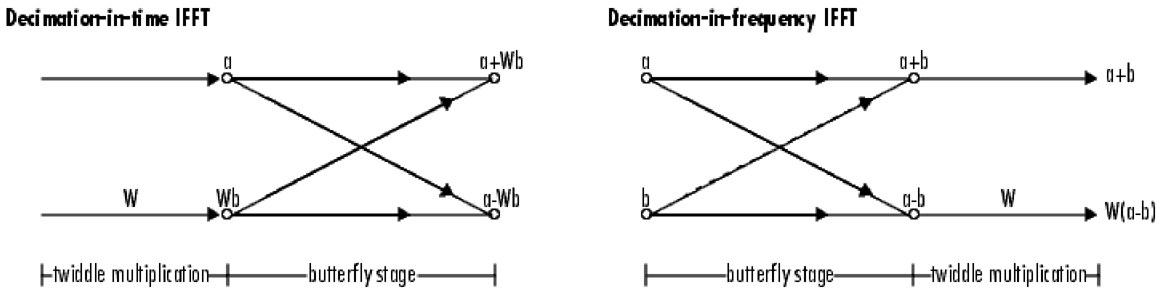
Input Complexity	Other Parameter Settings	Algorithms Used for IFFT Computation
Real or complex	☐ Input is in bit-reversed order ☐ Input is conjugate symmetric	Butterfly operation and radix-2 DIT
Real or complex	☑ Input is in bit-reversed order ☐ Input is conjugate symmetric	Radix-2 DIF

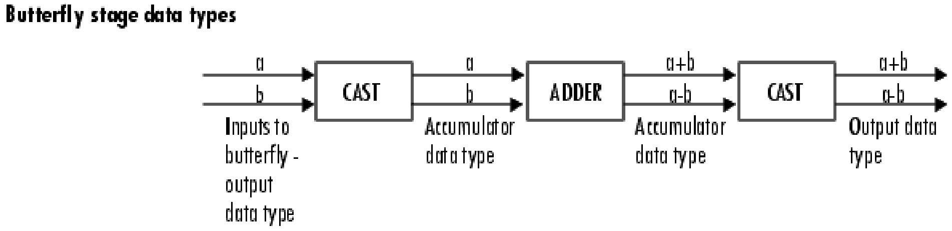
Note The **Input is conjugate symmetric** parameter cannot be used for fixed-point signals.

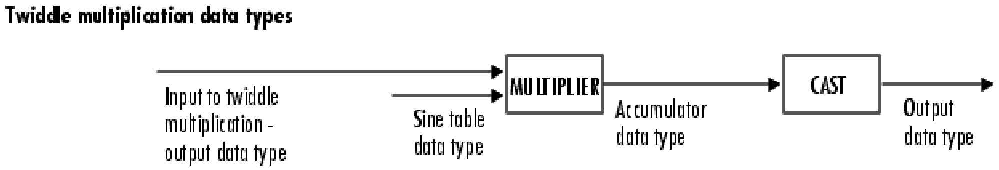
Fixed-Point Data Types

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 2-94.

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



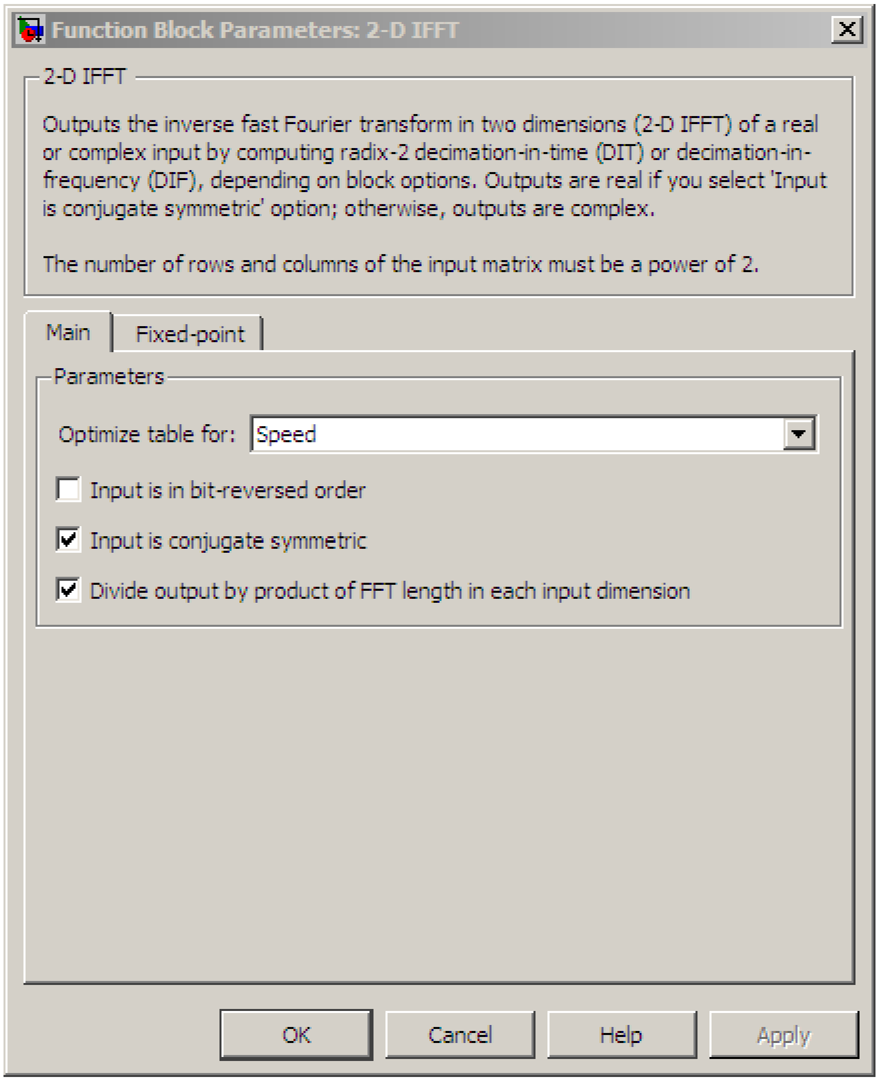



Twiddle multiplication data types
MULTIPLIER
CAST
Input to twiddle multiplication - output data type
Sine table data type
Accumulator data type
Output data type

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

Dialog Box

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



Function Block Parameters: 2-D IFFT

2-D IFFT

Outputs the inverse fast Fourier transform in two dimensions (2-D IFFT) of a real or complex input by computing radix-2 decimation-in-time (DIT) or decimation-in-frequency (DIF), depending on block options. Outputs are real if you select 'Input is conjugate symmetric' option; otherwise, outputs are complex.

The number of rows and columns of the input matrix must be a power of 2.

Main | Fixed-point

Parameters

Optimize table for: Speed

☐ Input is in bit-reversed order

☑ Input is conjugate symmetric

☑ Divide output by product of FFT length in each input dimension

Optimize table for

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

Input is in bit-reversed order

Designate the order of the input channel elements. Select this check box when the input should appear in reversed order, and clear it when the input should appear in linear order. in linear order. The block yields invalid outputs when you do not set this parameter correctly. See “Input Order” on page 2-89.

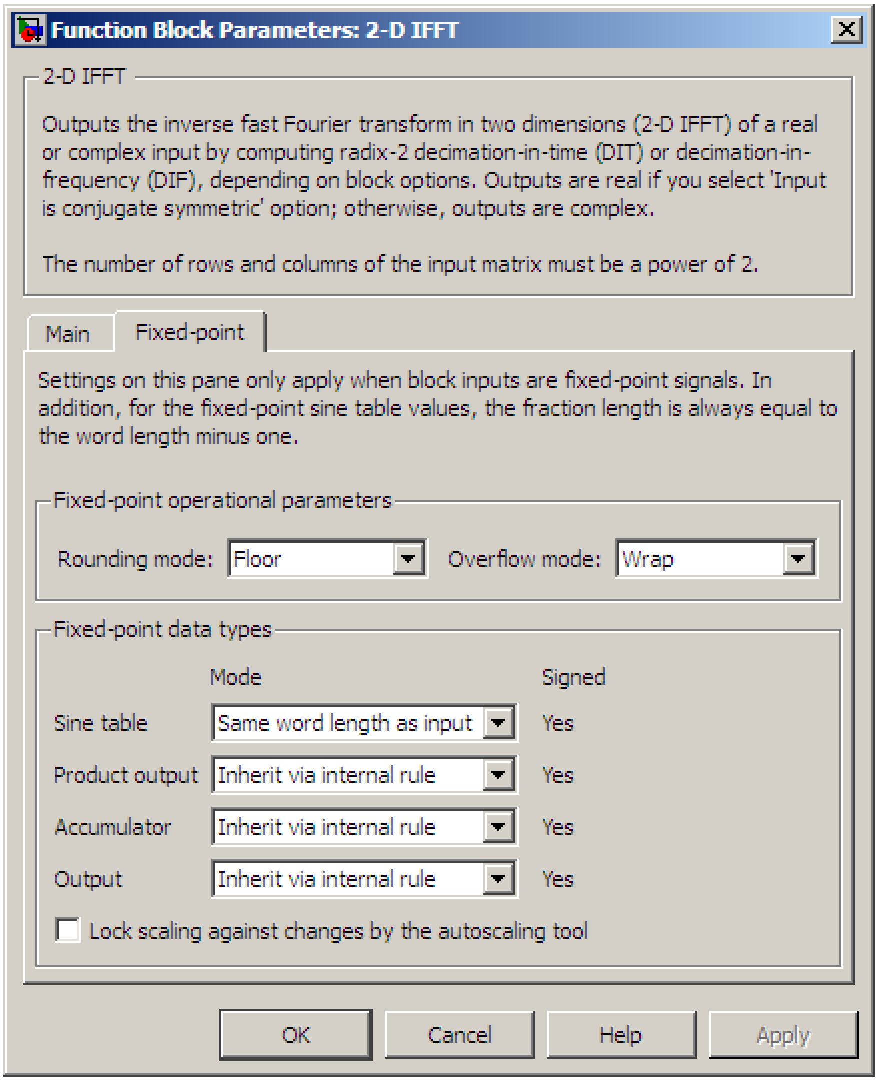
Input is conjugate symmetric

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

Divide output by product of FFT length in each input dimension

Select this check box to compute the scaled IFFT.

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



2-D IFFT

Outputs the inverse fast Fourier transform in two dimensions (2-D IFFT) of a real or complex input by computing radix-2 decimation-in-time (DIT) or decimation-in-frequency (DIF), depending on block options. Outputs are real if you select 'Input is conjugate symmetric' option; otherwise, outputs are complex.

The number of rows and columns of the input matrix must be a power of 2.

Settings on this pane only apply when block inputs are fixed-point signals. In addition, for the fixed-point sine table values, the fraction length is always equal to the word length minus one.

Fixed-point operational parameters

Rounding mode: Floor   Overflow mode: Wrap

Fixed-point data types

	Mode	Signed
Sine table	Same word length as input	Yes
Product output	Inherit via internal rule	Yes
Accumulator	Inherit via internal rule	Yes
Output	Inherit via internal rule	Yes

☐ Lock scaling against changes by the autoscaling tool

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 2-92 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. All signals in the Video and Image Processing Blockset have a bias of 0.

Accumulator

Use this parameter to specify how to designate the accumulator word and fraction lengths. Refer to "Fixed-Point Data Types" on page 2-92 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 block automatically sets the accumulator word length and fraction length 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. All signals in the Video and Image Processing Blockset have a bias of 0.

Output

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

- When you select `Inherit via internal rule`, the block automatically sets the word length and fraction length 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(log 2((M-1)(N-1)))+1*

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

  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. All signals in the Video and Image Processing Blockset have a bias of 0.

See Also

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

IFFT	Signal Processing Blockset software
Pad	Signal Processing Blockset software
`bitrevorder`	Signal Processing Toolbox software
`fft`	MATLAB
`ifft`	MATLAB

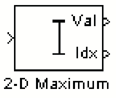
Purpose

Find maximum values in an input or sequence of inputs

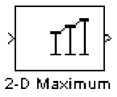
Library

`vipobslib`

Description



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

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

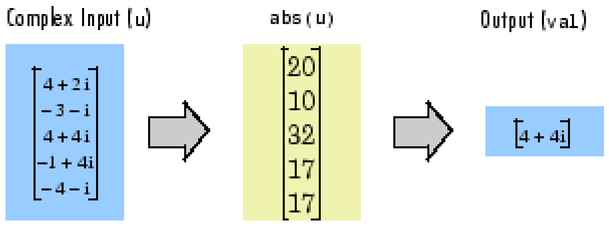
When **Mode** is set to `Value`, the block computes the maximum value in each column of the M-by-N input matrix u independently at each sample time.

```
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

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.

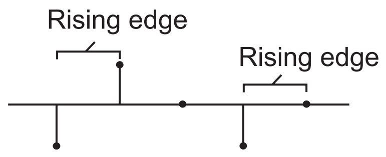
Resetting the Running Maximum

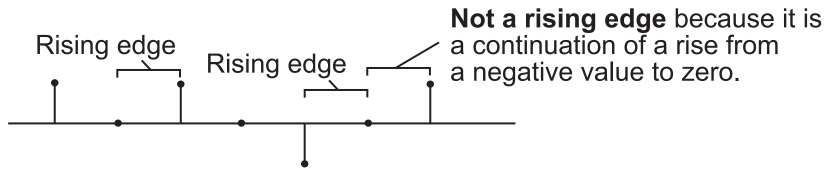
The block resets the running maximum whenever a reset event is detected at the optional Rst port. The rate of the reset signal must be a positive integer multiple of the rate of the data signal input.

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.

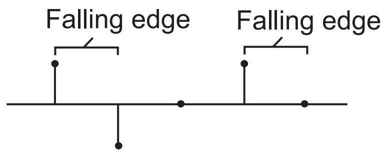
You specify the reset event in the **Reset port** menu:

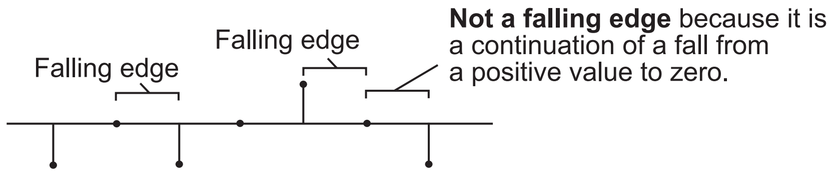
- `None` —- Disables the Rst port.
- `Rising edge` — Triggers a reset operation when the `Rst` input does one of the following:
  - Rises from a negative value to a positive value or 0
  - Rises from 0 to a positive value, where the rise is not a continuation of a rise from a negative value to 0 (see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




Rising edge
Rising edge
Not a rising edge because it is a continuation of a rise from a negative value to zero.

- `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)




Falling edge
Falling edge
Not a falling edge because it is a continuation of a fall from a positive value to zero.

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

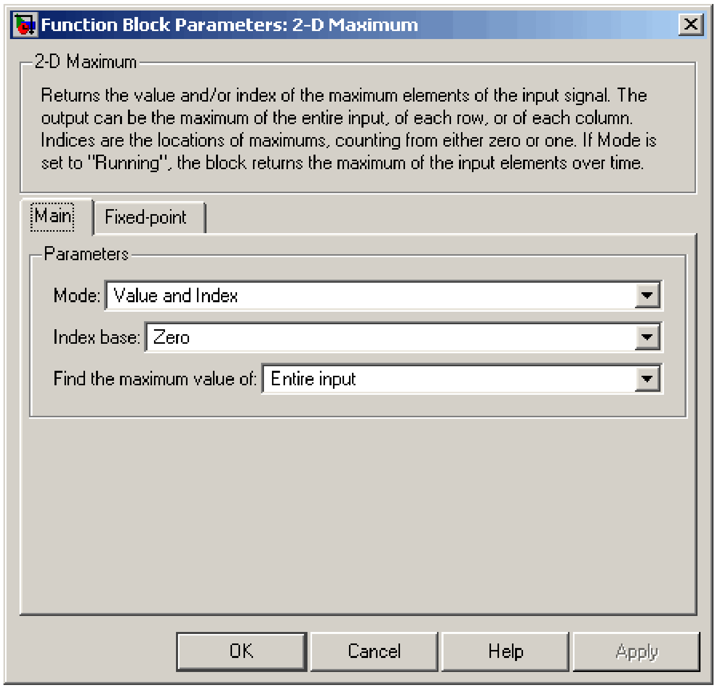
Note When running simulations in the Simulink `MultiTasking` mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking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

Fixed-Point Data Types

The parameters on the **Fixed-point** pane of the dialog box are only used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before a comparison is made, as described in "Value Mode" on page 2-102. 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.



Function Block Parameters: 2-D Maximum

2-D Maximum

Returns the value and/or index of the maximum elements of the input signal. The output can be the maximum of the entire input, of each row, or of each column. Indices are the locations of maximums, counting from either zero or one. If Mode is set to "Running", the block returns the maximum of the input elements over time.

Parameters

Mode: Value and Index

Index base: Zero

Find the maximum value of: Entire input

Mode

Specify the block's mode of operation:

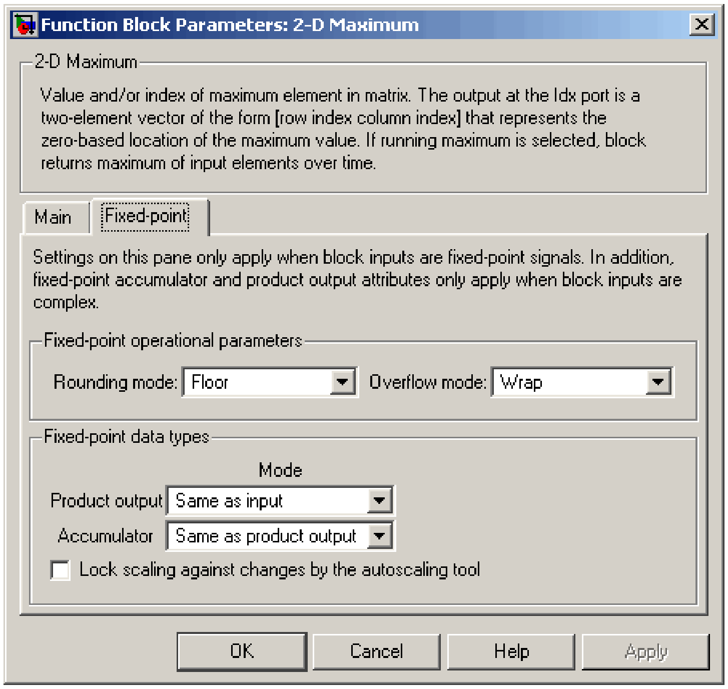
- `Value` — Output the maximum value of each input matrix
- `Index` — Output the zero-based index location of the maximum value
- `Value and Index` — Output both the value and the index location
- `Running` — Track the maximum value of the input sequence over time

Index base
Specify whether the index is zero based or one based.

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

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

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


Function Block Parameters: 2-D Maximum

2-D Maximum

Value and/or index of maximum element in matrix. The output at the Idx port is a two-element vector of the form [row index column index] that represents the zero-based location of the maximum value. If running maximum is selected, block returns maximum of input elements over time.

Main | Fixed-point

Settings on this pane only apply when block inputs are fixed-point signals. In addition, fixed-point accumulator and product output attributes only apply when block inputs are complex.

Fixed-point operational parameters

Rounding mode: Floor   Overflow mode: Wrap

Fixed-point data types

Mode

Product output: Same as input

Accumulator: Same as product output

☐ Lock scaling against changes by the autoscaling tool

OK | Cancel | Help | Apply

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 2-102. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ignored for other types of inputs.

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

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

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

Accumulator

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

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

Lock scaling against changes by the autoscaling tool

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

2-D Mean (Obsolete)

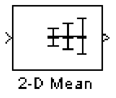
Purpose

Find mean value of each input matrix

Library

Statistics

Description



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

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

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	• 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	• 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 — • 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 — • 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.	• 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.	• 8-, 16-, and 32-bit unsigned integer	No

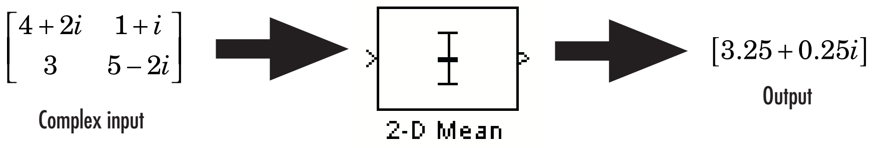
Port	Input/Output	Supported Data Types	Complex Values Supported
Output/Out	Without ROI processing — Mean of each M-by-N input matrix or the mean for each element of a series of M-by-N inputs. With ROI processing — Vector of separate statistical values for each ROI or a scalar value that represents the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

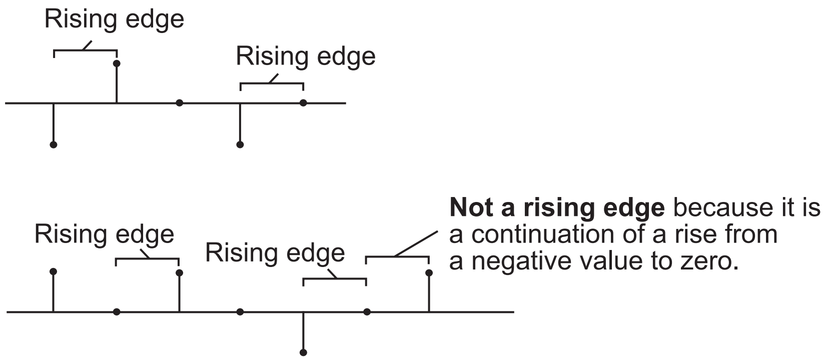
Resetting the Running Mean

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.

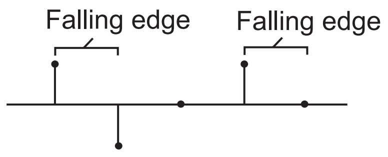
When the block is reset, the running mean associated with each element is initialized to the value in the corresponding location of the current input.

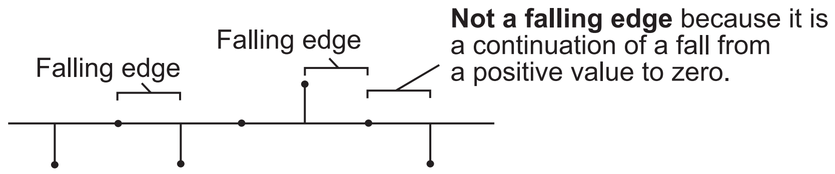
You specify the reset event using the **Reset port** parameter:

- `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)


Rising edge
Rising edge
Rising edge
Rising edge
Not a rising edge because it is a continuation of a rise from a negative value to zero.

- `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)




Falling edge
Falling edge
Not a falling edge because it is a continuation of a fall from a positive value to zero.

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

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

ROI Processing

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

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

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

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

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

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

Output = Individual statistics for each ROI

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

Output = Single statistic for all ROIs

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

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

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

Output = Individual statistics for each ROI

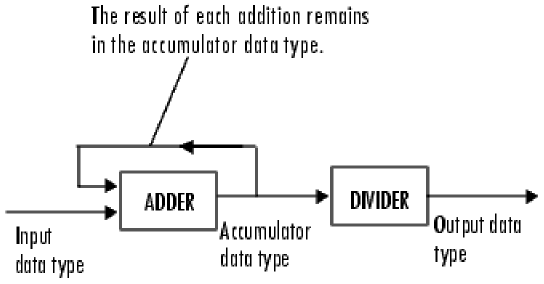
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

Output = Single statistic for all ROIs

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

Fixed-Point Data Types

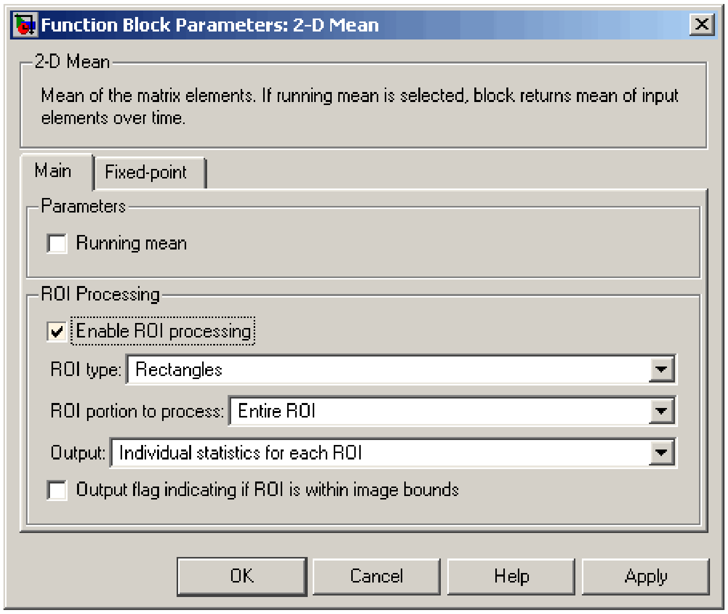
The following diagram shows the data types used in the 2-D Mean block for fixed-point signals.


The result of each addition remains in the accumulator data type.
ADDER
DIVIDER
Input data type
Accumulator data type
Output data type

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.



Function Block Parameters: 2-D Mean

2-D Mean

Mean of the matrix elements. If running mean is selected, block returns mean of input elements over time.

Main | Fixed-point

Parameters

☐ Running mean

ROI Processing

☑ Enable ROI processing

ROI type: Rectangles

ROI portion to process: Entire ROI

Output: Individual statistics for each ROI

☐ Output flag indicating if ROI is within image bounds

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`.

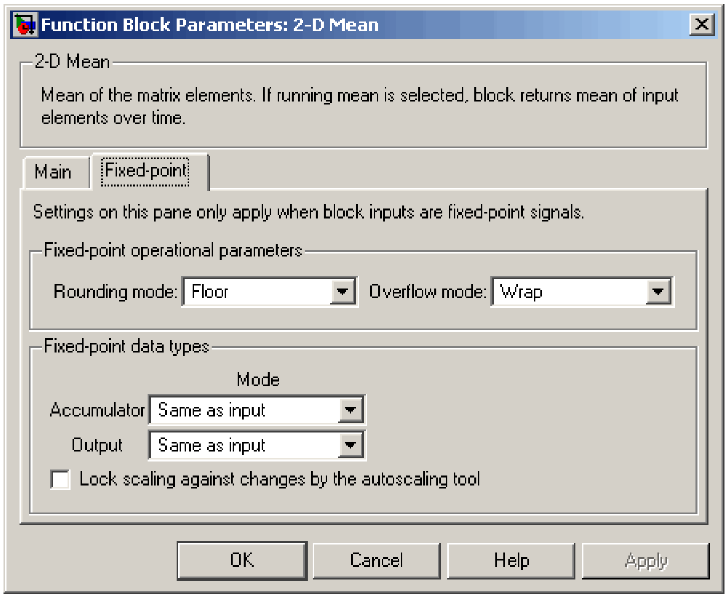
ROI portion to process
Specify whether you want to calculate the statistical value for the entire ROI or just the ROI perimeter. This parameter is only visible if, for the **ROI type** parameter, you specify `Rectangles`.

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 2-117. 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 2-117. 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.


Function Block Parameters: 2-D Mean

2-D Mean

Mean of the matrix elements. If running mean is selected, block returns mean of input elements over time.

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

Accumulator: Same as input

Output: Same as input

☐ Lock scaling against changes by the autoscaling tool

Rounding mode
: Select the rounding mode for fixed-point operations.

Overflow mode
: Select the overflow mode for fixed-point operations.

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

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

Output

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

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

Lock scaling against changes by the autoscaling tool

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

See Also

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

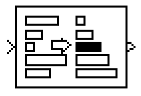
Minimum	Signal Processing Blockset software
`mean`	MATLAB software

2-D Median (Obsolete)

Purpose Find median value of each input matrix

Library Statistics

Description



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

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

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	• 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.

```
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 2-128. 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.

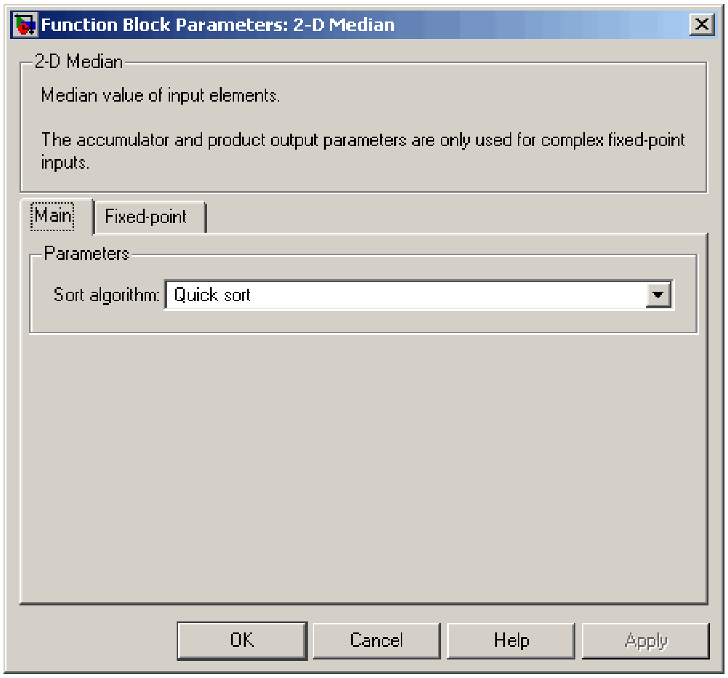
The accumulator and product output parameters are used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before the input elements are sorted.

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.



Function Block Parameters: 2-D Median

2-D Median

Median value of input elements.

The accumulator and product output parameters are only used for complex fixed-point inputs.

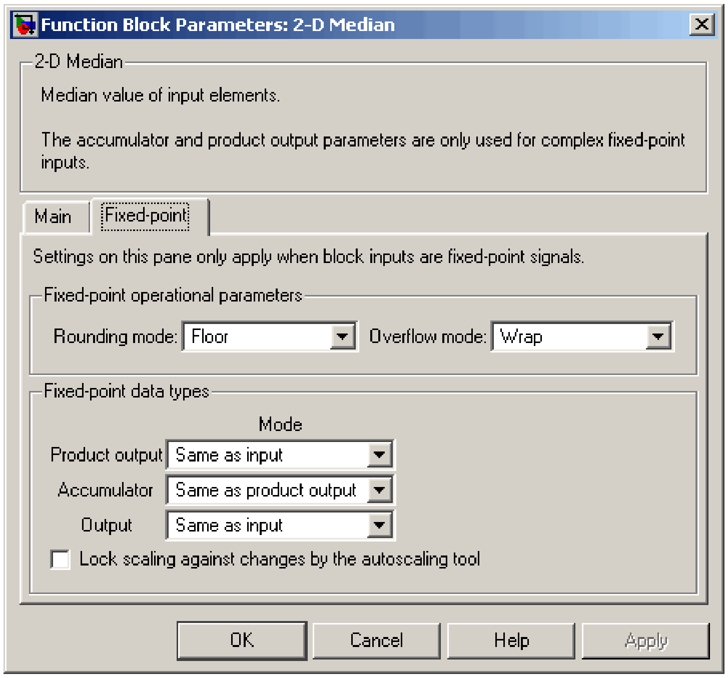
Parameters

Sort algorithm: Quick sort

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.


Function Block Parameters: 2-D Median

2-D Median

Median value of input elements.

The accumulator and product output parameters are only used for complex fixed-point inputs.

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

	Mode
Product output	Same as input
Accumulator	Same as product output
Output	Same as input

☐ Lock scaling against changes by the autoscaling tool

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 2-127 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.

See Also

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

2-D Minimum (Obsolete)

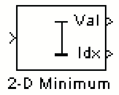
Purpose

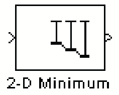
Find minimum values in an input or sequence of inputs

Library

`vipobslib`

Description





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

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

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

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

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

Value Mode

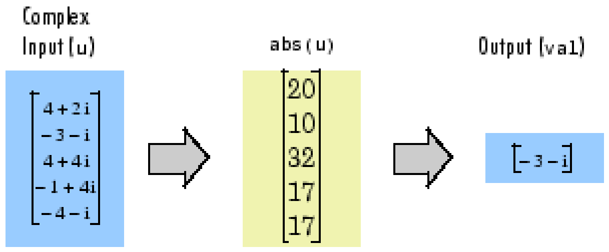
When **Mode** is set to `Value`, the block computes the minimum value in each column of the *M*-by-*N* input matrix *u* independently at each sample time.

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

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

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

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



Index Mode

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

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

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

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

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

Value and Index Mode

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

Running Mode

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

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

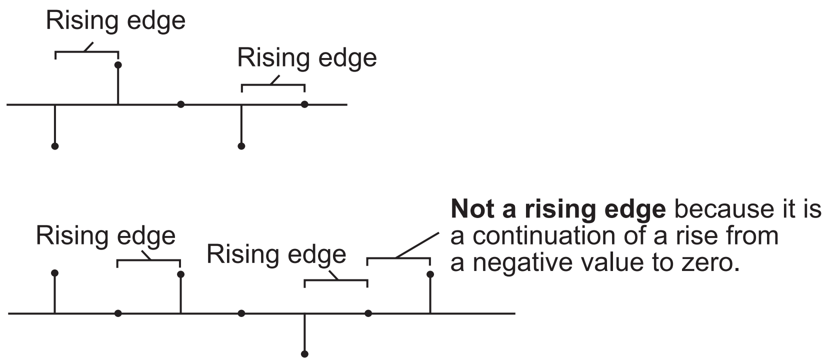
Resetting the Running Minimum

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.

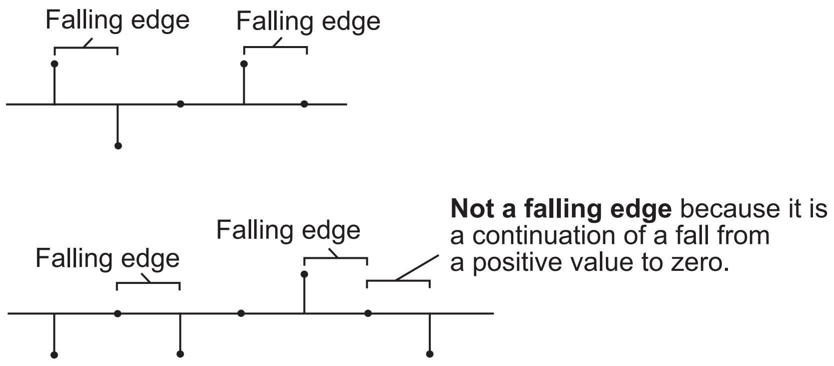
You specify the reset event by the **Reset port** parameter:

- `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)


Rising edge
Rising edge
Rising edge
Rising edge
Not a rising edge because it is a continuation of a rise from a negative value to zero.

- `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)


Falling edge
Falling edge
Falling edge
Falling edge
Not a falling edge because it is a continuation of a fall from a positive value to zero.

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

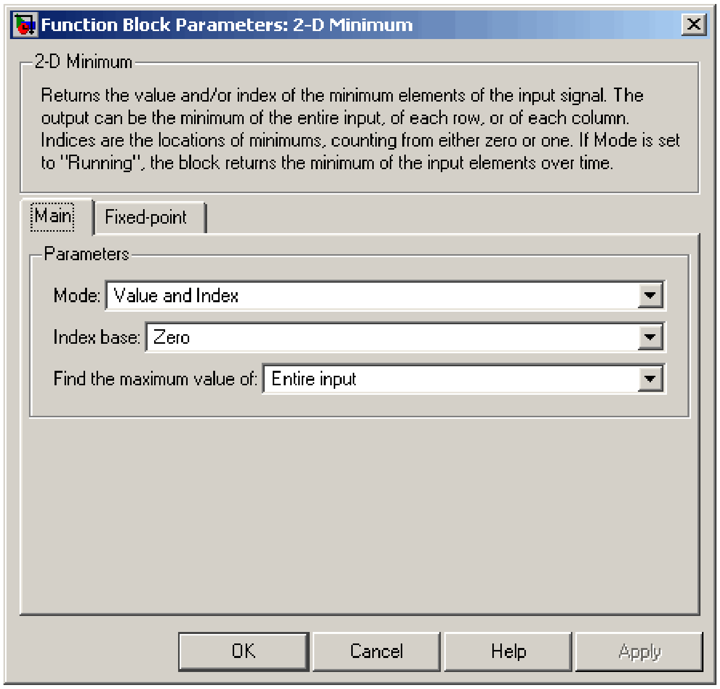
Note When running simulations in the Simulink `MultiTasking` mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking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

Fixed-Point Data Types

The parameters on the **Fixed-point** pane of the dialog box are only used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before a comparison is made, as described in "Value Mode" on page 2-133. 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.


Function Block Parameters: 2-D Minimum

2-D Minimum

Returns the value and/or index of the minimum elements of the input signal. The output can be the minimum of the entire input, of each row, or of each column. Indices are the locations of minimums, counting from either zero or one. If Mode is set to "Running", the block returns the minimum of the input elements over time.

Mode: Value and Index

Index base: Zero

Find the maximum value of: Entire input

Mode

Specify the block's mode of operation:

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

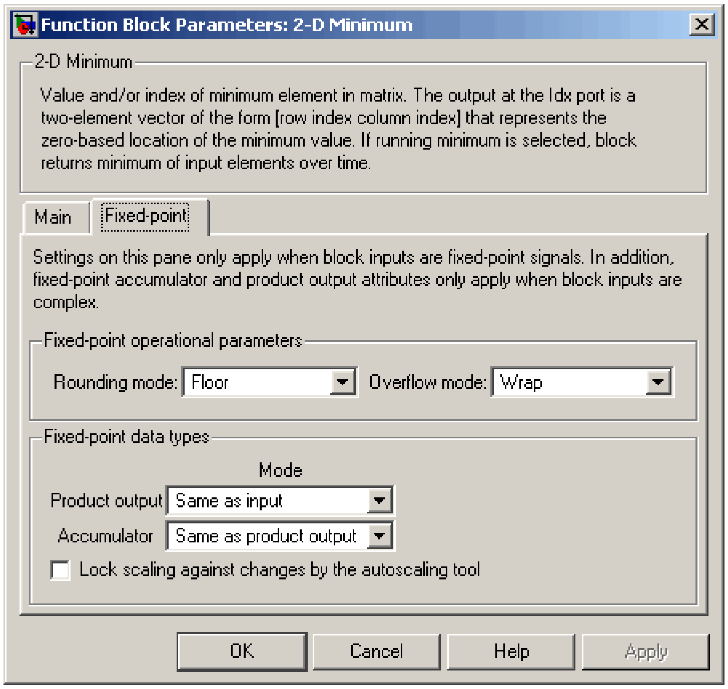
- `Running` — Track the minimum value of the input sequence over time

Index base
Specify whether the index is zero based or one based.

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

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

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


Function Block Parameters: 2-D Minimum

2-D Minimum

Value and/or index of minimum element in matrix. The output at the Idx port is a two-element vector of the form [row index column index] that represents the zero-based location of the minimum value. If running minimum is selected, block returns minimum of input elements over time.

Settings on this pane only apply when block inputs are fixed-point signals. In addition, fixed-point accumulator and product output attributes only apply when block inputs are complex.

Fixed-point operational parameters

Rounding mode: Floor   Overflow mode: Wrap

Fixed-point data types

Mode

Product output: Same as input

Accumulator: Same as product output

☐ Lock scaling against changes by the autoscaling tool

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 2-133. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ignored for other types of inputs.

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode
Select the overflow mode for fixed-point operations.

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

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

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

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

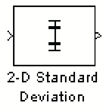
Lock scaling against changes by the autoscaling tool
Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Tool. For more information, see `fxptdlg`, a reference page on the Fixed-Point Tool in the Simulink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

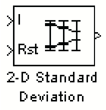
2-D Standard Deviation (Obsolete)

Purpose Find standard deviation of each input matrix

Library Statistics

Description





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

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

Port	Input/Output	Supported Data Types	Complex Values Supported
Input / I	Vector or matrix of intensity values	• Double-precision floating point • Single-precision floating point	Yes

Port	Input/Output	Supported Data Types	Complex Values Supported
Rst	Signal that triggers a reset event	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
ROI	• 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 — • 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 — • 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.	• 8-, 16-, and 32-bit unsigned integer	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Label Numbers	Vector containing the label numbers for the regions for which the block will compute the statistics.	• 8-, 16-, and 32-bit unsigned integer	No
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.

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

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

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

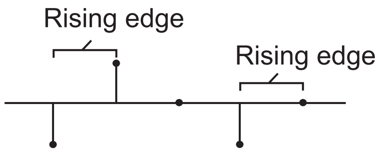
For the next input, the block computes the standard deviation for each element of the first three inputs, and so on.

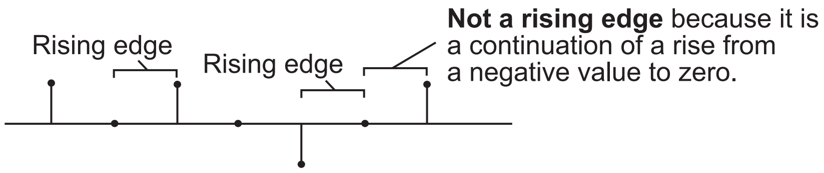
Resetting the Running Standard Deviation

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.

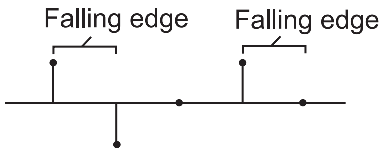
You specify the reset event using the **Reset port** parameter:

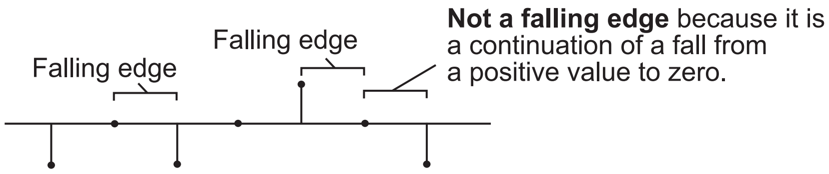
- `None` — Disables the Rst port
- `Rising edge` — Triggers a reset operation when the Rst input does one of the following:
  - Rises from a negative value to a positive value or 0
  - Rises from 0 to a positive value, where the rise is not a continuation of a rise from a negative value to 0 (see the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




Rising edge
Rising edge
Not a rising edge because it is a continuation of a rise from a negative value to zero.

- `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)




Falling edge
Falling edge
Not a falling edge because it is a continuation of a fall from a positive value to zero.

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

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

ROI Processing

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

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

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

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

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

Output = Individual statistics for each ROI

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

Output = Single statistic for all ROIs

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

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

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

Output = Individual statistics for each ROI

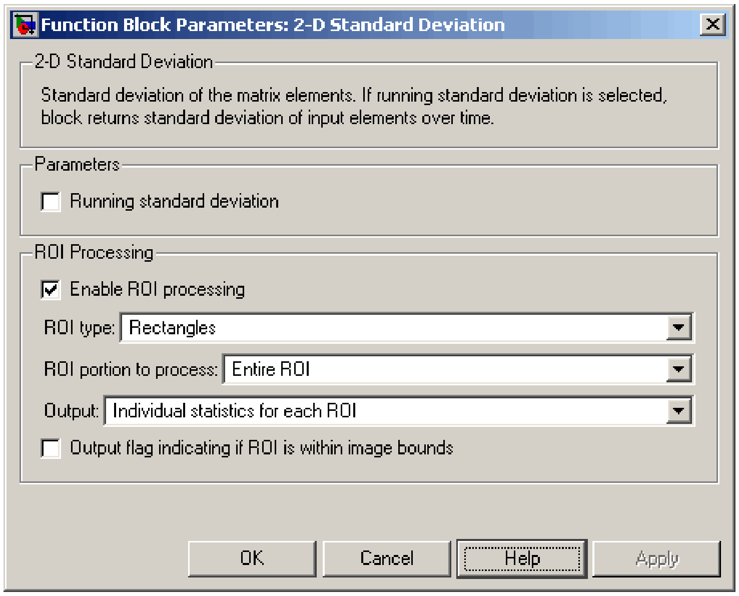
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

Output = Single statistic for all ROIs

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

Dialog Box

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



Standard deviation of the matrix elements. If running standard deviation is selected, block returns standard deviation of input elements over time.

☐ Running standard deviation

☑ Enable ROI processing

ROI type: Rectangles

ROI portion to process: Entire ROI

Output: Individual statistics for each ROI

☐ Output flag indicating if ROI is within image bounds

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

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

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

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 2-148. 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 2-148. This parameter is visible if, for the **ROI type** parameter, you select `Label matrix`.

See Also

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

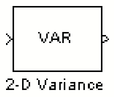
Median	Video and Image Processing Blockset software
Variance	Video and Image Processing Blockset software
Maximum	Signal Processing Blockset software
Minimum	Signal Processing Blockset software
Standard Deviation	Signal Processing Blockset software
`std`	MATLAB software

2-D Variance (Obsolete)

Purpose Compute variance of each input matrix

Library Statistics

Description



I
Running
VAR
Rst
2-D Variance

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

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

Port	Input/Output	Supported Data Types	Complex Values Supported
Input / I	Vector or matrix of intensity values	• 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	• 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	• 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 - 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 - 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.	• 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.	• 8-, 16-, and 32-bit unsigned integer	No

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:

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

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

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

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

Running Operation

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

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

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

and

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

The block computes the variance,

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

and outputs

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

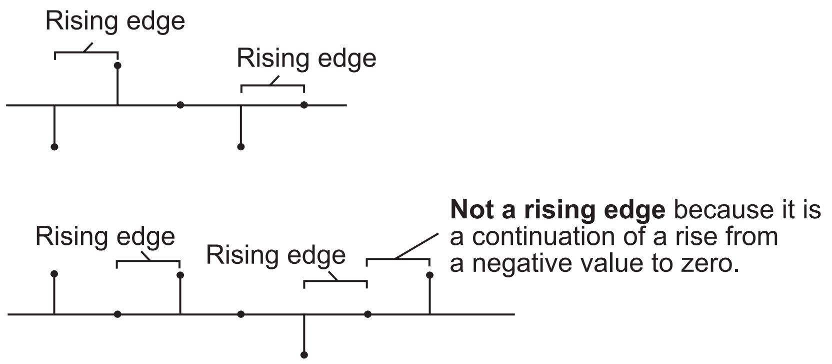
For the next input, the block computes the variance for each element of the first three inputs, and so on.

Resetting the Running Variance

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

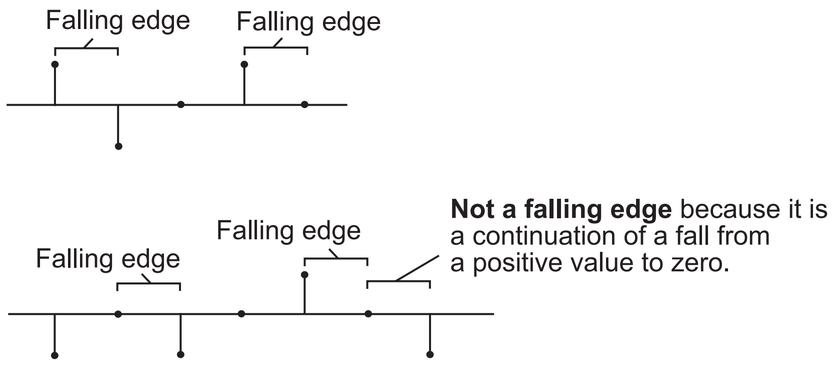
You specify the reset event using the **Reset port** parameter:

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


Rising edge
Rising edge
Rising edge
Rising edge
Not a rising edge because it is a continuation of a rise from a negative value to zero.

- `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)


Falling edge
Falling edge
Falling edge
Falling edge
Not a falling edge because it is a continuation of a fall from a positive value to zero.

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

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

ROI Processing

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

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

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

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

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

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

Output = Individual Statistics for Each ROI

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

Output = Single Statistic for All ROIs

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

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

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

Output = Individual Statistics for Each ROI

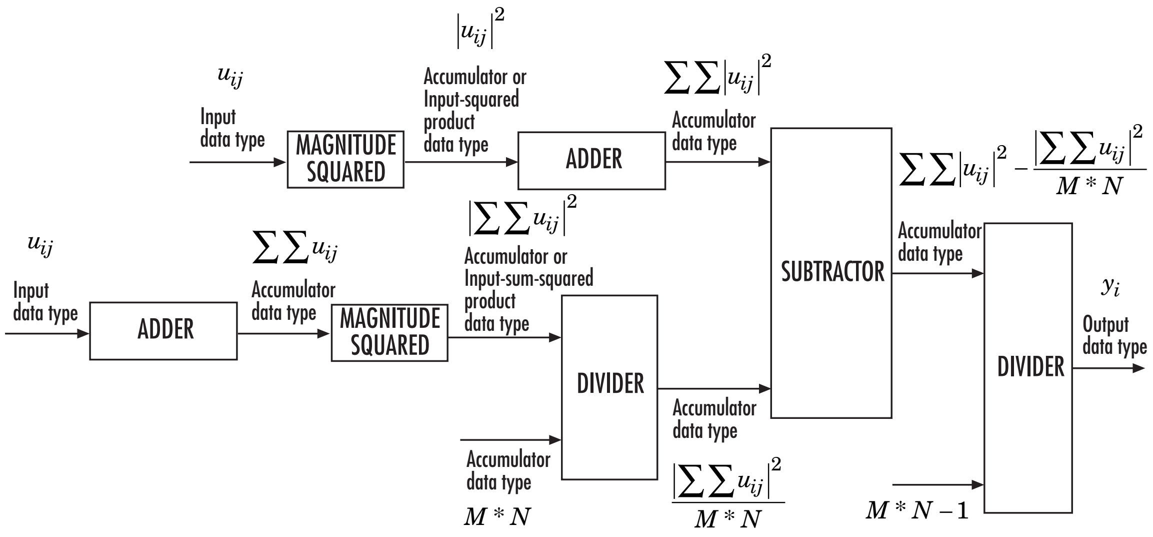
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

Output = Single Statistic for All ROIs

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

Fixed-Point Data Types

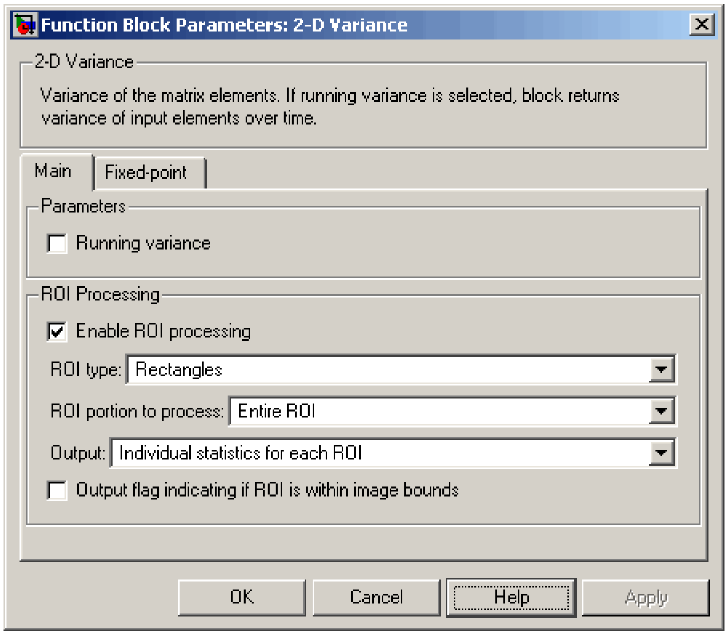
The following diagram shows the data types used in the 2-D Variance block for fixed-point signals.



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

Dialog Box

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



Function Block Parameters: 2-D Variance

2-D Variance

Variance of the matrix elements. If running variance is selected, block returns variance of input elements over time.

Parameters

☐ Running variance

ROI Processing

☑ Enable ROI processing

ROI type: Rectangles

ROI portion to process: Entire ROI

Output: Individual statistics for each ROI

☐ Output flag indicating if ROI is within image bounds

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`.

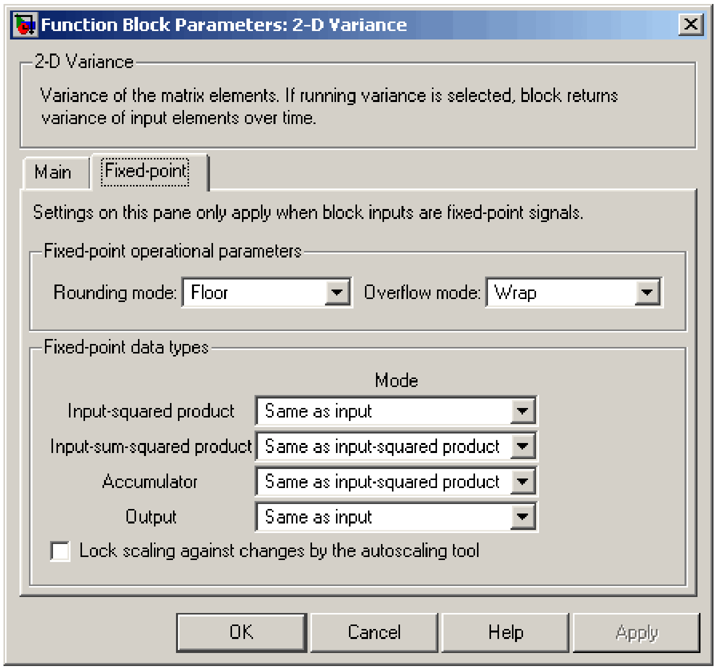
Output flag indicating if ROI is within image bounds

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 2-159. 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 2-159. 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:


Function Block Parameters: 2-D Variance

2-D Variance

Variance of the matrix elements. If running variance is selected, block returns variance of input elements over time.

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

Fixed-point operational parameters

Rounding mode: Floor   Overflow mode: Wrap

Fixed-point data types

	Mode
Input-squared product	Same as input
Input-sum-squared product	Same as input-squared product
Accumulator	Same as input-squared product
Output	Same as input

☐ Lock scaling against changes by the autoscaling tool

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 2-162 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.

Accumulator

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

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

Output

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

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

Lock scaling against changes by the autoscaling tool

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

See Also

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

Apply Geometric Transformation

Purpose

Apply projective or affine transformation to an image

Library

Geometric Transformations

Description

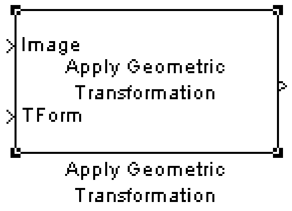


Image
Apply Geometric Transformation
TForm
Apply Geometric Transformation

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

Input	Description
Image	*M*-by-*N* or *M*-by-*N*-by-*P* input matrix. *M*: Number of rows in the image. *N*: Number of columns in the image. *P*: Number of color planes in the image.
TForm	When you set the **Transformation matrix source** parameter to `Input port`, the **TForm** input will accept: • *2*-by-*3* matrix (affine transform) or *6*-by-*Q* matrix (multiple affine transforms) • *3*-by-*3* matrix (projective transform) or *9*-by-*Q* matrix (multiple projective transforms) *Q*: Number of transformations.
ROI	When you set the ROI source parameter to Input port, the ROI input will accept: • *4-element* vector rectangle ROI

Input	Description
	• *2L-element* vector polygon ROI • *4*-by-*R* matrix for multiple rectangle ROIs • *2L*-by-*R* matrix for multiple polygon ROIs *R*: Number of Region of Interests (ROIs). *L* (*L* ≥ *3*): Number of vertices in a polygon ROI.

Transformations

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

Affine Transformation

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

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

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

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

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

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

If you use more than one transformation, the transformation coefficients must be arranged as a *6*-by-*Q* matrix, where each column has the format of $H = [h_1 \quad h_2 \quad h_3 \quad h_4 \quad h_5 \quad h_6]^T$, and Q is the number of transformations as in:

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

Projective Transformation

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

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

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

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

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

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

If you use more than one transformation, the transformation coefficients must be arranged as a *9*-by-*Q* matrix, where each column has the format of $H = [h_1 \quad h_2 \quad h_3 \quad h_4 \quad h_5 \quad h_6 \quad h_7 \quad h_8 \quad h_9]^T$, and Q is the number of transformations. For example,

$$H = \begin{bmatrix} h_{11} & h_{21} & \dots & h_{Q1} \\ h_{12} & h_{22} & \dots & h_{Q2} \\ . & . & \dots & . \\ h9 & h_{29} & \dots & h_{Q9} \end{bmatrix}$$

Region of Interest

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

Region of Interest (ROI) Types

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

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

A **polygon** ROI is specified by the vertices of the polygon in clockwise or counter-clockwise order, with at least three or more vertices. If you specify one ROI of *L* vertices, it must be a *2L-element* vector of format $[r1, \quad c1, \quad r2, \quad c2, \quad \dots \quad rL, \quad cL]^T$. If you specify more than one ROI, it must be a *2L*-by-*R* matrix, where now *L* is the maximum number of vertices in the ROIs. For ROI with vertices fewer than *L*, its last vertex can be repeated to form a vector or *2L*. See "Defining Shapes to Draw" on page 2-331 for details.

ROI Processing

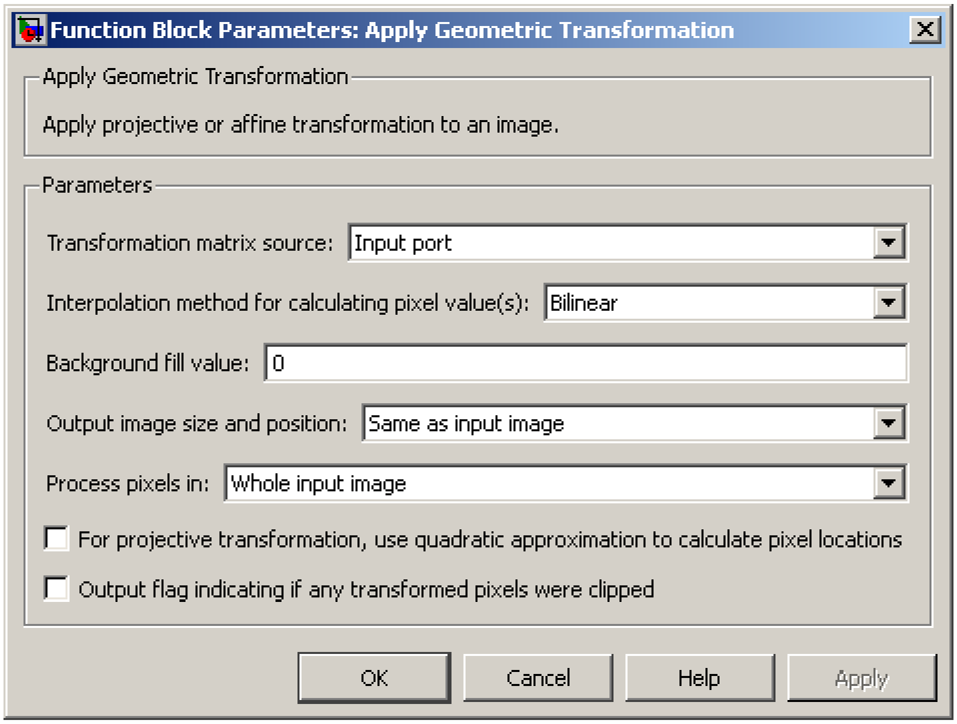
The transformations will be applied on the whole image, or on specified multiple ROIs. The table below outlines how transformation matrices are handled with an entire image and with single and multiple ROIs.

Number of Transformation Matrices	Region of Interest
One transformation matrix	You can apply the transformation on the entire image, single ROI or multiple ROIs.
Multiple transformation matrices	• You can apply multiple transformation matrices on one ROI or on the entire image. The transformations are done in the order they are entered in the **TForm**. • You can apply multiple transformation matrices on multiple ROIs. Each transformation matrix is applied to one ROI. The first transformation matrix specified is applied to the first ROI specified. The second transformation matrix is applied to the second ROI specified, and so on. The number of transformation matrices must be equal to the number of ROIs.

Quadratic Approximation Mode for Projective Transformation

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

Dialog Box


Function Block Parameters: Apply Geometric Transformation

Apply Geometric Transformation

Apply projective or affine transformation to an image.

Parameters

Transformation matrix source: Input port

Interpolation method for calculating pixel value(s): Bilinear

Background fill value: 0

Output image size and position: Same as input image

Process pixels in: Whole input image

☐ For projective transformation, use quadratic approximation to calculate pixel locations

☐ Output flag indicating if any transformed pixels were clipped

Transformation matrix source

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

Transformation matrix

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

Interpolation method for calculating pixel value(s)

Specify interpolation method, either `Nearest neighbor`, `Bilinear`, or `Bicubic` interpolation to calculate output pixel values. See Geometric Transformation Interpolation Methods for an overview of these methods.

Background fill value

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

Output image size and position

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

Size [height width]

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

Location of the upper left corner [row col]

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

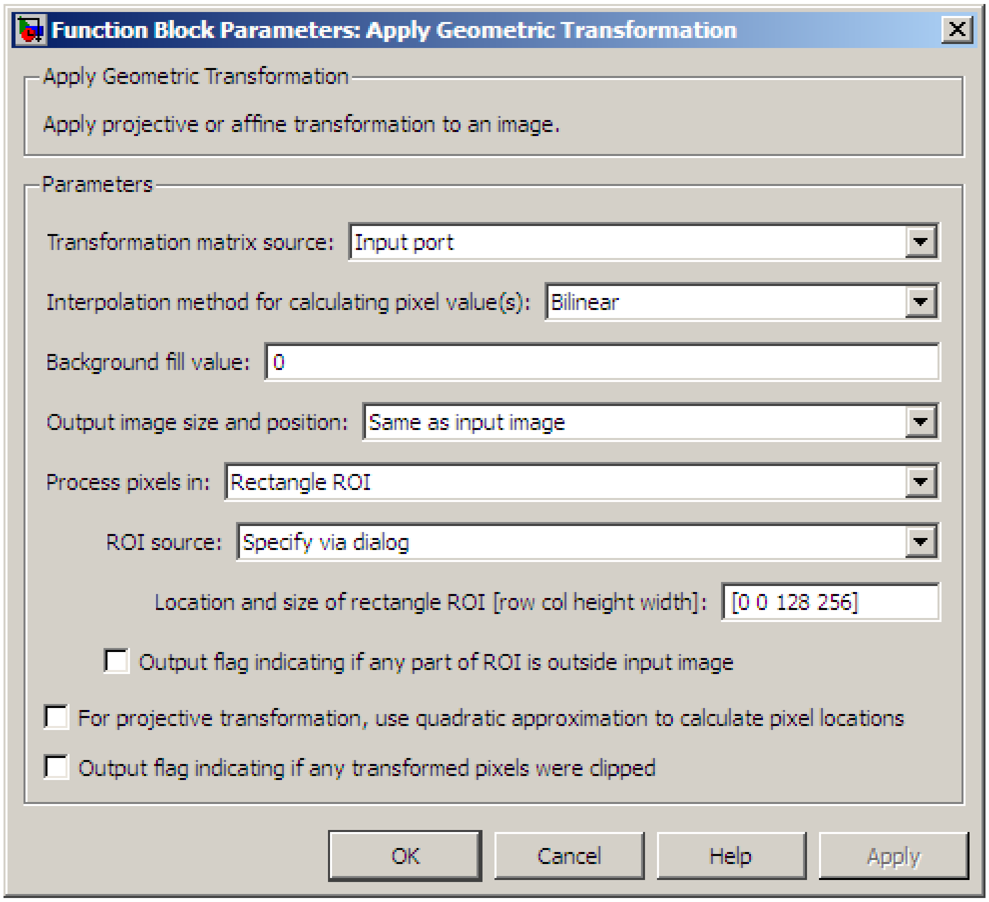
Process pixels in

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

ROI source

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

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



Function Block Parameters: Apply Geometric Transformation

Apply Geometric Transformation

Apply projective or affine transformation to an image.

Parameters

Transformation matrix source: Input port

Interpolation method for calculating pixel value(s): Bilinear

Background fill value: 0

Output image size and position: Same as input image

Process pixels in: Rectangle ROI

ROI source: Specify via dialog

Location and size of rectangle ROI [row col height width]: [0 0 128 256]

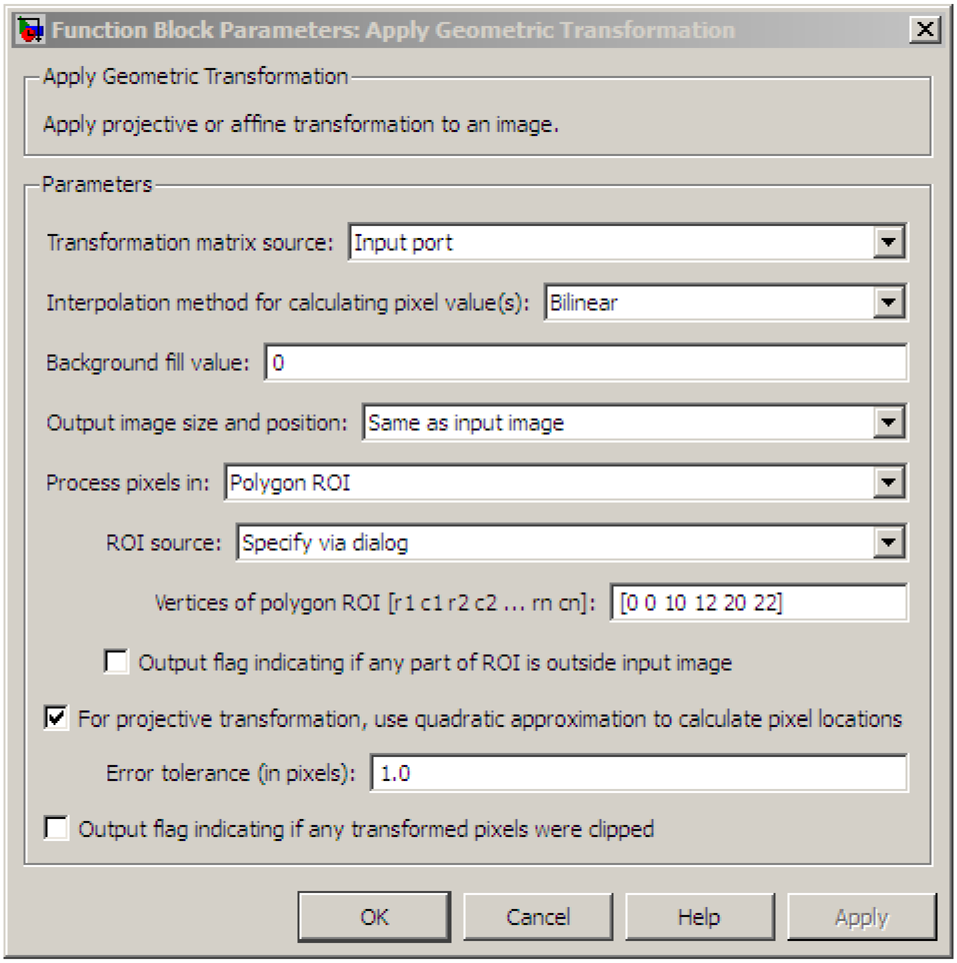
☐ Output flag indicating if any part of ROI is outside input image

☐ For projective transformation, use quadratic approximation to calculate pixel locations

☐ Output flag indicating if any transformed pixels were clipped

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

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



Apply Geometric Transformation

Apply projective or affine transformation to an image.

Parameters

Transformation matrix source: Input port

Interpolation method for calculating pixel value(s): Bilinear

Background fill value: 0

Output image size and position: Same as input image

Process pixels in: Polygon ROI

ROI source: Specify via dialog

Vertices of polygon ROI [r1 c1 r2 c2 ... rn cn]: [0 0 10 12 20 22]

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

☑ For projective transformation, use quadratic approximation to calculate pixel locations

Error tolerance (in pixels): 1.0

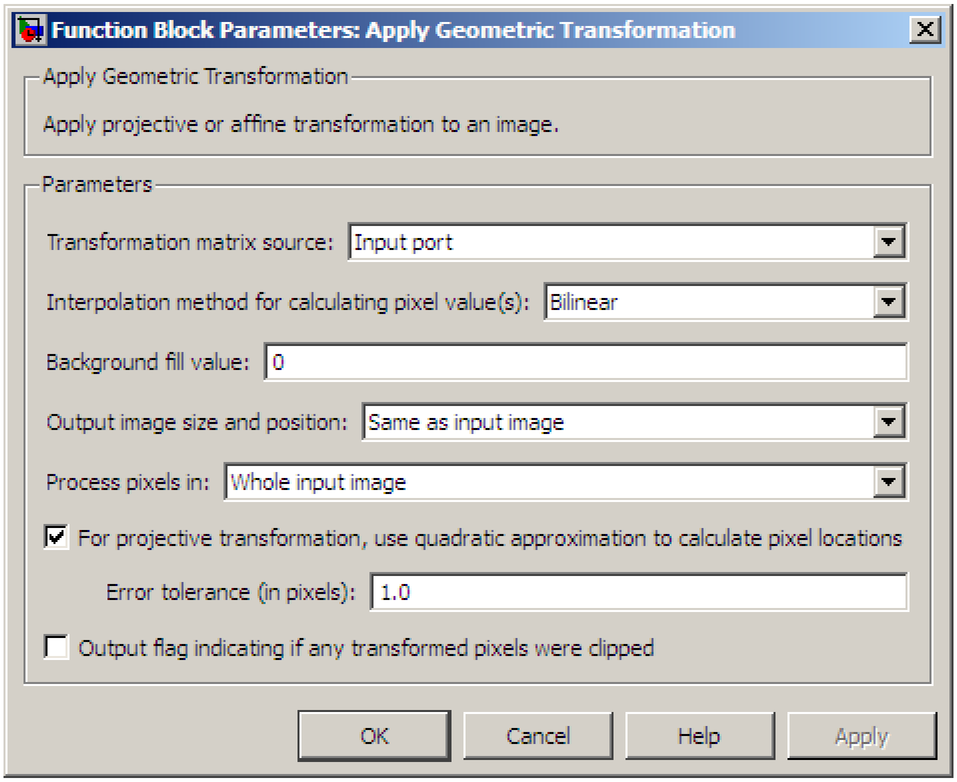
☐ Output flag indicating if any transformed pixels were clipped

Specify a *2L-element* vector or *2L*-by-*R* matrix, (where *L* is the number of vertices in a polygon and *R* represents the number of ROIs). This parameter becomes available when you set **Process pixels in** parameter to `Polygon ROI`.

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

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

Error tolerance (in pixels)


Function Block Parameters: Apply Geometric Transformation

Apply Geometric Transformation

Apply projective or affine transformation to an image.

Parameters

Transformation matrix source: Input port

Interpolation method for calculating pixel value(s): Bilinear

Background fill value: 0

Output image size and position: Same as input image

Process pixels in: Whole input image

[✓] For projective transformation, use quadratic approximation to calculate pixel locations

Error tolerance (in pixels): 1.0

[ ] Output flag indicating if any transformed pixels were clipped

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

Output flag indicating if any transformed pixels were clipped

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

Examples

"Apply a Projective Transformation to an Image" on page 2-179	A simple model using the Apply Geometric Transformation block.
"Create an Image of a Cube using Three Regions of Interest" on page 2-180	A model which uses the Apply Geometric Transformation block to create an image of a cube using ROIs.

Apply a Projective Transformation to an Image

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

1. Add two Constant blocks for the input image and the transformation matrix. Set the **Constant value** parameters for the constant blocks as follows:
   - for the input image, `"checker_board"`, and
   - for the transformation matrix,`[0.06 -0.15 15; 0.04 0.1 2; -0.01 0 1]`

2 Add two Video Viewer blocks, connecting one directly to the input image output port, and the other one to the Apply Geometric Transformation output port.

Create an Image of a Cube using Three Regions of Interest

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

1 Change the **Process pixels in** parameter to `Polygon ROI`.

2 Change the **Background fill value** to `[0.5 0.5 0.75]`

3 Add three Constant blocks for the input image, transformation matrix, and ROI vertices. Set the **Constant value** parameters for the three blocks as follows:

- For the input image, `"checker_board"`
- For the transformation matrix,`[1 0 -15 0 1 15; 0.4082 0 15 -0.4082 1.0204 35; 1 -0.4082 5.4082 0 0.4082 44.5918]'`, and
- For the polygon ROI, `[50 0 50 49 99 49 99 0; 0 0 0 49 49 49 49 0; 50 50 50 99 99 99 99 50]'`.

4 Add two Video Viewer blocks, connecting one directly to the Constant block containing the input image. The other Video Viewer block connected to the Apply Geometric Transformation output port.

References

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

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

Supported Data Types

Port	Supported Data Types
Image	• Double-precision floating point • Single-precision floating point
TForm	• Double-precision floating point • Single-precision floating point
ROI	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers
Output	Same as input
Err_roi	Boolean
Err_clip	Boolean

See Also

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

Blob Analysis	Video and Image Processing Blockset
`Video and Image Processing Demos`	Video and Image Processing Blockset

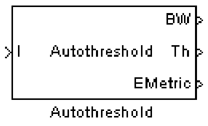
Purpose

Convert intensity image to binary image

Library

Conversions

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	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
BW	Scalar, vector, or matrix that represents a binary image	Boolean	No
Th	Threshold value	Same as I port	No
EMetric	Effectiveness metric	Same as I port	No

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

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

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

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

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

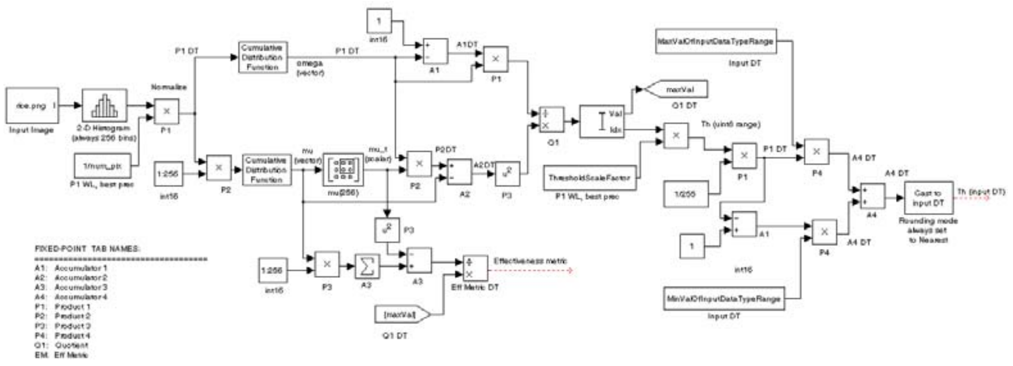
- `Ignore` — Proceed with the computation and do not issue a warning message. If you choose this option, the block performs the most efficient computation. However, if the input values exceed the expected range, the block produces incorrect results.
- `Saturate` — Change any input values outside the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

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

Fixed-Point Data Types

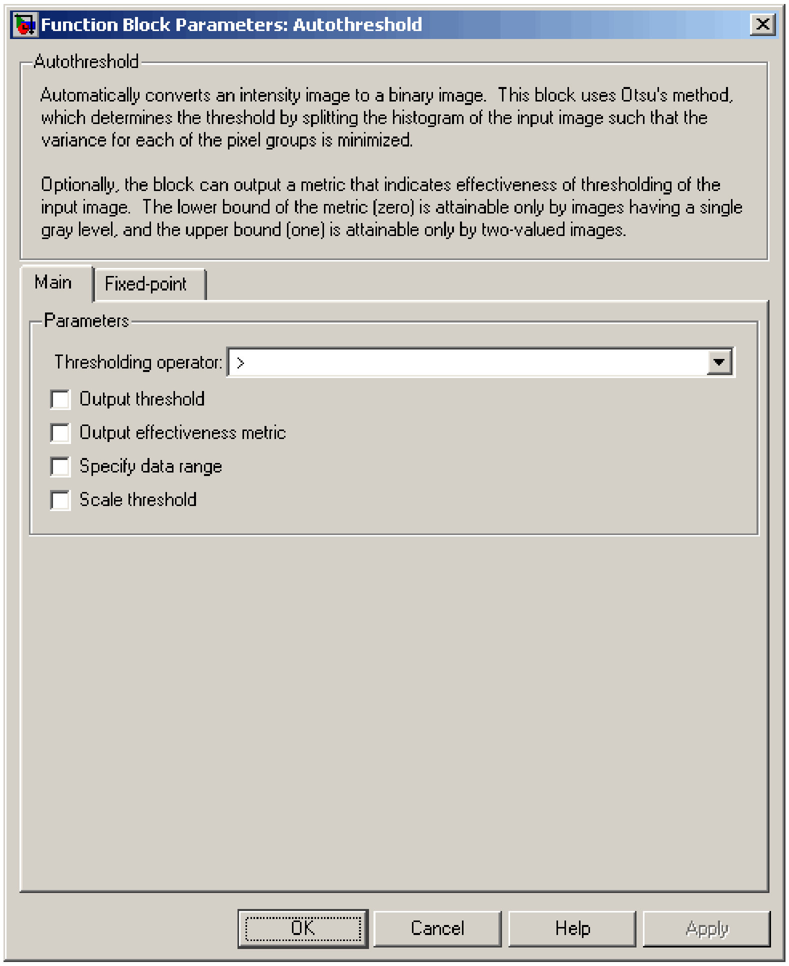
The following diagram shows the data types used in the Autothreshold block for fixed-point signals. You can use the default fixed-point parameters if your input has a word length less than or equal to 16.

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



Dialog Box

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



Function Block Parameters: Autothreshold

Autothreshold

Automatically converts an intensity image to a binary image. This block uses Otsu's method, which determines the threshold by splitting the histogram of the input image such that the variance for each of the pixel groups is minimized.

Optionally, the block can output a metric that indicates effectiveness of thresholding of the input image. The lower bound of the metric (zero) is attainable only by images having a single gray level, and the upper bound (one) is attainable only by two-valued images.

Main | Fixed-point

Parameters

Thresholding operator: >

☐ Output threshold

☐ Output effectiveness metric

☐ Specify data range

☐ Scale threshold

Thresholding operator

Specify the condition the block places on the input matrix values. If you select > or <=, the block outputs 0 or 1 depending on whether the input matrix values are above, below, or equal to the threshold value.

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

Output effectiveness metric
Select this check box to output values that represent the effectiveness of the thresholding at the EMetric port.

Specify data range
If you clear this check box, the block assumes that floating-point input values range from 0 to 1. To specify a different data range, select this check box.

Minimum value of input
Enter the minimum value of your input data. This parameter is visible if you select the **Specify data range** check box.

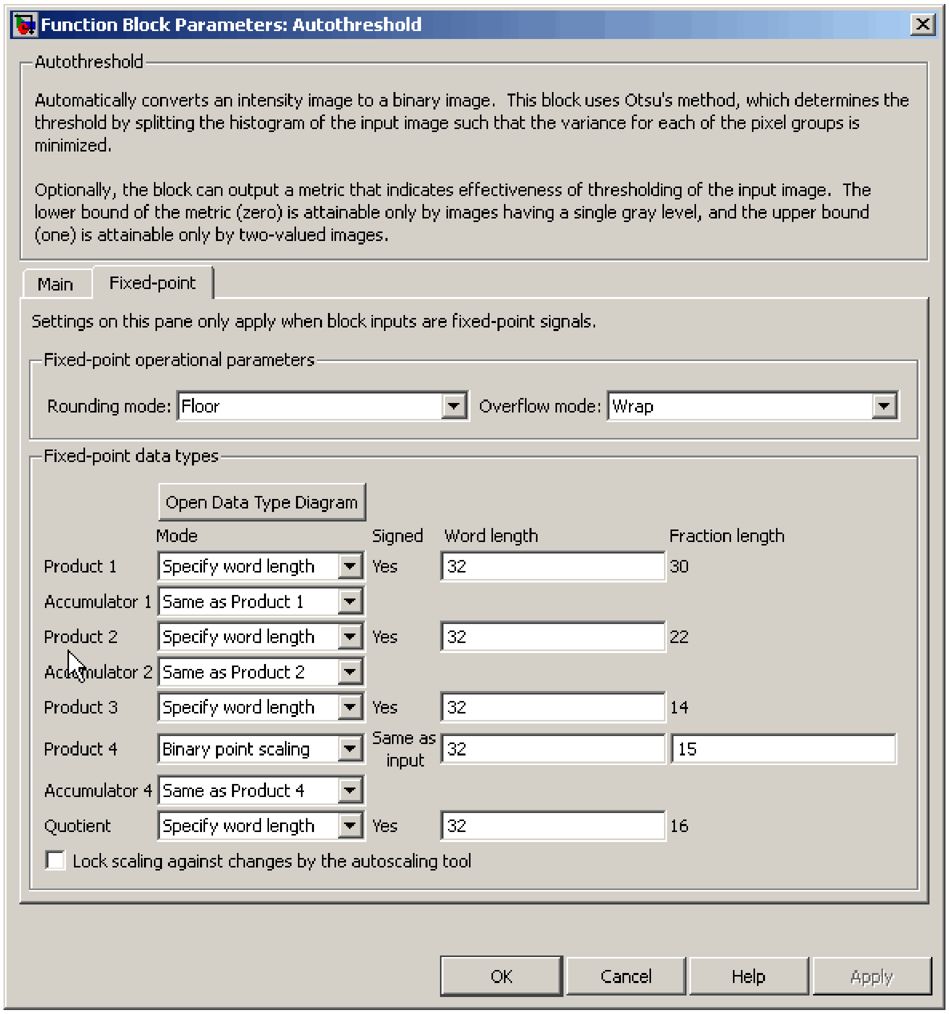
Maximum value of input
Enter the maximum value of your input data. This parameter is visible if you select the **Specify data range** check box.

When data range is exceeded
Specify the block's behavior when the input values are outside the expected range. Your options are `Ignore`, `Saturate`, `Warn and saturate`, or `Error`. This parameter is visible if you select the **Specify data range** check box.

Scale threshold
Select this check box to scale the threshold value computed by Otsu's method.

Threshold scaling factor
Enter a scalar value. The block multiplies this scalar value with the threshold value computed by Otsu's method and uses the 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.



Autothreshold

Automatically converts an intensity image to a binary image. This block uses Otsu's method, which determines the threshold by splitting the histogram of the input image such that the variance for each of the pixel groups is minimized.

Optionally, the block can output a metric that indicates effectiveness of thresholding of the input image. The lower bound of the metric (zero) is attainable only by images having a single gray level, and the upper bound (one) is attainable only by two-valued images.

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

Fixed-point operational parameters

Rounding mode: Floor   Overflow mode: Wrap

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Product 1	Specify word length	Yes	32	30
Accumulator 1	Same as Product 1			
Product 2	Specify word length	Yes	32	22
Accumulator 2	Same as Product 2			
Product 3	Specify word length	Yes	32	14
Product 4	Binary point scaling	Same as input	32	15
Accumulator 4	Same as Product 4			
Quotient	Specify word length	Yes	32	16

☐ Lock scaling against changes by the autoscaling tool

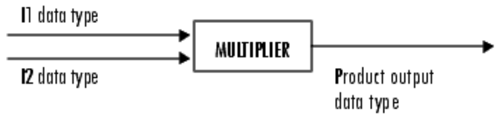
Rounding mode

Select the rounding mode for fixed-point operations. This parameter does not apply to the Cast to input DT step shown in "Fixed-Point Data Types" on page 2-185. For this step, **Rounding mode** is always set to `Nearest`.

Overflow mode

Select the overflow mode for fixed-point operations.

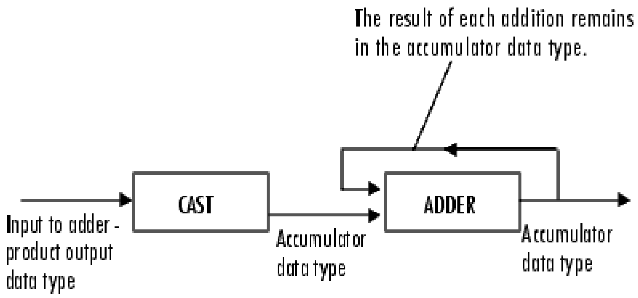
Product 1, 2, 3, 4



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

- When you select `Specify word length`, you can enter the word length of the product values in bits. The block sets the fraction length to give you the best precision.
- When you select `Same as input`, the characteristics match those of the input to the block. This choice is only available for the **Product 4** parameter.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output in bits.
- When you select `Slope and bias scaling`, you can enter the word length in bits and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset software is 0.

Accumulator 1, 2, 3, 4


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type

As shown previously, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate the accumulator word and fraction lengths.

- When you select `Same as Product`, these characteristics match those of the product output.
- When you select `Specify word length`, you can enter the word length of the accumulator values in bits. The block sets the fraction length to give you the best precision. This choice is not available for the **Accumulator 4** parameter because it is dependent on the input data type.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator in bits.
- When you select `Slope and bias scaling`, you can enter the word length in bits and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset software is 0.

The **Accumulator 3** parameter is only visible if, on the **Main** pane, you select the **Output effectiveness metric** check box.

Quotient

Choose how to specify the word length and fraction length of the quotient data type:

- When you select `Specify word length`, you can enter the word length of the quotient values in bits. The block sets the fraction length to give you the best precision.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the quotient, in bits.
- When you select `Slope and bias scaling`, you can enter the word length in bits and the slope of the quotient. The bias of all signals in the Video and Image Processing Blockset software is 0.

Eff Metric

Choose how to specify the word length and fraction length of the effectiveness metric data type. This parameter is only visible if, on the **Main** tab, you select the **Output effectiveness metric** check box.

- When you select `Specify word length`, you can enter the word length of the effectiveness metric values, in bits. The block sets the fraction length to give you the best precision.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the effectiveness metric in bits.
- When you select `Slope and bias scaling`, you can enter the word length in bits and the slope of the effectiveness metric. The bias of all signals in the Video and Image Processing Blockset software is 0.

Lock 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

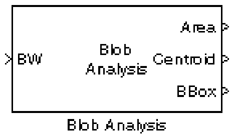
Purpose

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 quantities such as the Centroid, label matrix, and blob count.

The Blob Analysis block supports variable size signals at the input and output.

For information about how pixel and spatial coordinate systems are defined in the Video and Image Processing Blockset documentation, see "Coordinate Systems" in the *Video and Image Processing Blockset software User's Guide*.

Use the Variable Selector block to select certain blobs based on their statistics. For more information about this block, see the Variable Selector block reference page in the Signal Processing Blockset documentation.

Port	Input/Output	Supported Data Types	Complex Values Supported
BW	Vector or matrix that represents a binary image	Boolean	No
Area	Vector that represents the number of pixels in labeled regions	• 32-bit signed integer	No

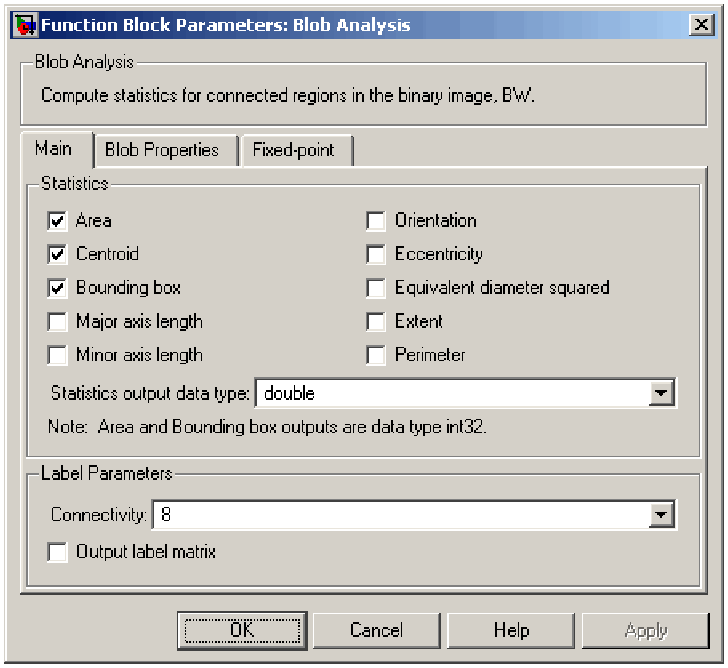
Port	Input/Output	Supported Data Types	Complex Values Supported
Centroid	2-by-N matrix of centroid coordinates, where N is the number of blobs	• Double-precision floating point • Single-precision floating point • Fixed point	No
BBox	4-by-N matrix of bounding box coordinates, where N is the number of blobs	• 32-bit signed integer	No
MajorAxis	Vector that represents the lengths of major axes of ellipses	• 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 *x*-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

Port	Input/Output	Supported Data Types	Complex Values Supported
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
Perimeter	Vector containing an estimate of the perimeter length, in pixels, for each blob	Same as Centroid port	No
Label	Label matrix	• 8-, 16-, or 32-bit unsigned integer	No
Count	Scalar value that represents the actual number of labeled regions in each image	Same as Label port	No

Dialog Box

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

Main tab


Function Block Parameters: Blob Analysis

Blob Analysis

Compute statistics for connected regions in the binary image, BW.

Main | Blob Properties | Fixed-point

Statistics

- [✓] Area
- [✓] Centroid
- [✓] Bounding box
- [ ] Major axis length
- [ ] Minor axis length
- [ ] Orientation
- [ ] Eccentricity
- [ ] Equivalent diameter squared
- [ ] Extent
- [ ] Perimeter

Statistics output data type: double

Note: Area and Bounding box outputs are data type int32.

Label Parameters

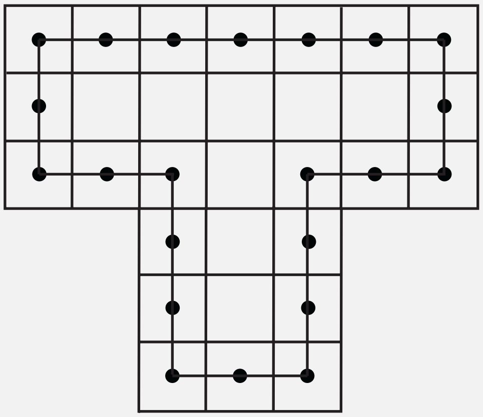
Connectivity: 8

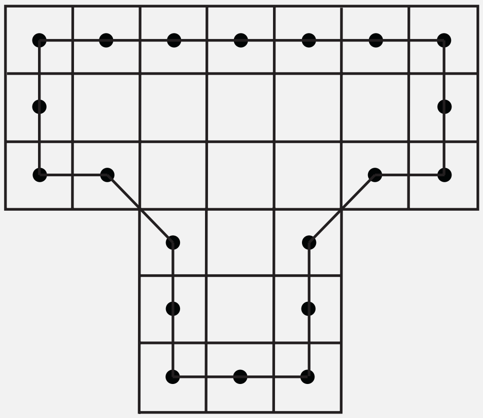
- [ ] Output label matrix

Parameter	Value
Area	If checked, the block output provides a vector that represents the number of pixels in labeled regions

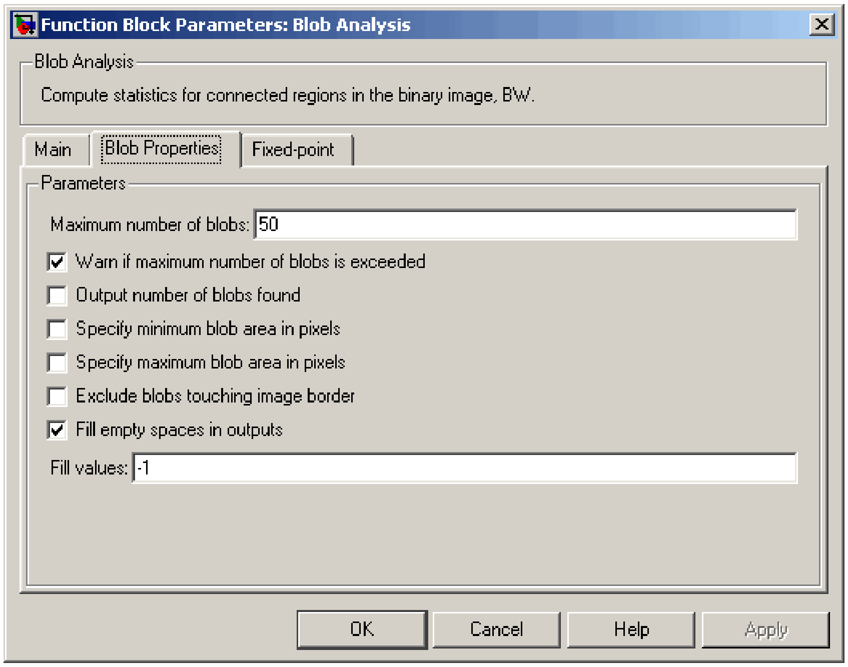
Parameter	Value
Centroid	If checked, the block output provides a 2-by-N matrix whose columns represent the coordinates of the centroid of each region, where N is the number of blobs. For example, if there are two blobs and the row and column coordinates of their centroids are *r1*, *c1* and *r2*, *c2*, respectively, the block outputs $\begin{bmatrix} r1 & r2 \\ c1 & c2 \end{bmatrix}$ at the Centroid port.
Bounding box	If checked, the block output provides a 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 $\begin{bmatrix} r1 & r2 \\ c1 & c2 \\ h1 & h2 \\ w1 & w2 \end{bmatrix}$ at the BBox port.
Major axis length	If checked, the block output provides a 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	If checked, the block output provides a vector that represents the lengths of the minor axes of ellipses that have the same normalized second central moments as the labeled regions

Parameter	Value
Orientation	If checked, the block output provides a vector that represents the angles between the major axes of the ellipses and the *x*-axis. The angle values are in radians and range between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$
Eccentricity	If checked, the block output provides a vector that represents the eccentricities of ellipses that have the same second moments as the region
Equivalent diameter squared	If checked, the block output provides a vector that represents the equivalent diameters squared
Extent	If checked, the block output provides a vector that represents the results of dividing the areas of the blobs by the area of their bounding boxes
Perimeter	If checked, the block output provides a N-by-1 vector containing estimates of the perimeter lengths, in pixels, of each blob, where N is the number of blobs
Statistics output data type	Use the **Statistics output data type** parameter to specify the data type of the outputs at the **Centroid**, **MajorAxis**, **MinorAxis**, **Orientation**, **Eccentricity**, **Equivalent diameter squared**, and **Extent** ports. If you select `Fixed-point`, the block cannot calculate the major axis, minor axis, orientation, or eccentricity and the associated check boxes become unavailable.

Parameter	Value
Connectivity	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. For example; 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.

Parameter	Value
	 The block takes a different path around the blob and estimates the perimeter to be $18+2\sqrt{2}$.
Output label matrix	If you select the **Output label matrix** check box, the block outputs the label matrix at the **Label** port., 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.

Blob Properties tab


Function Block Parameters: Blob Analysis

Blob Analysis

Compute statistics for connected regions in the binary image, BW.

Main | Blob Properties | Fixed-point

Parameters

Maximum number of blobs: 50

[✓] Warn if maximum number of blobs is exceeded

[ ] Output number of blobs found

[ ] Specify minimum blob area in pixels

[ ] Specify maximum blob area in pixels

[ ] Exclude blobs touching image border

[✓] Fill empty spaces in outputs

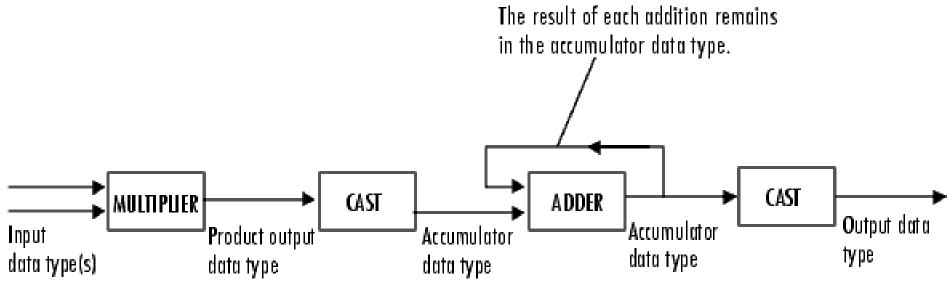
Fill values: -1

Parameter	Value
Maximum number of blobs	Specify the maximum number of labeled regions in each input image. The block uses this value to preallocate vectors and matrices so that they are long enough to hold all of the statistical values.
Warn if maximum number of blobs is exceeded	If checked, 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 checked, the block outputs a scalar value that represents the actual number of connected regions in each image at the **Count** port.

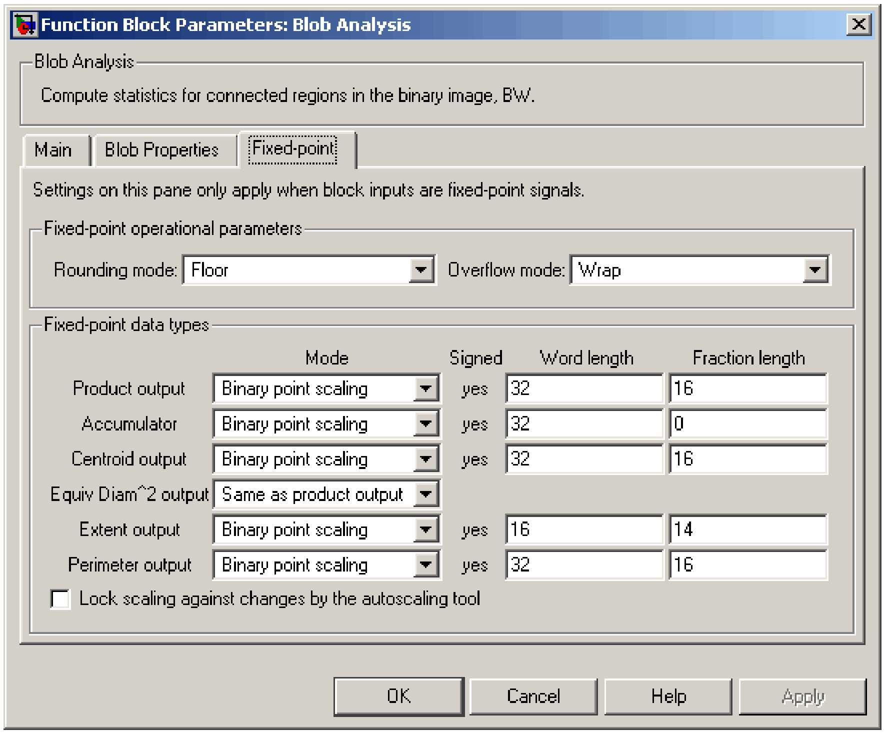
Parameter	Value
Specify minimum blob area in pixels	If checked, you can enter the minimum blob area in the edit box that appears beside the check box. Blobs that do not meet this criteria are not labeled. This parameter is tunable.
Specify maximum blob area in pixels	If checked, you can enter the maximum blob area in the edit box that appears beside the check box. Blobs that do not meet this criteria are not labeled. The maximum allowable value is maximum of `unit32` data type. This parameter is tunable.
Exclude blobs touching image border	If checked, label blobs that contain at least one border pixel will be excluded.
Output blob statistics as a variable-size signal	If checked, the output blob statistics will support variable-size signals. When this is checked, you do not need to specify fill values.
Fill empty spaces in outputs	If checked, the block fills empty spaces in the statistical vectors with the value(s) you specify in the **Fill values** parameter. The **Fill empty spaces in outputs** check box will not appear when the **Output blob statistics as a variable-size signal** is checked.
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. 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. The **Fill values** parameter will not appear when the **Output blob statistics as a variable-size signal** is checked.

Fixed-Point Data Types

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



The parameters on the **Fixed-point** tab are applicable only when the **Statistics output data type** parameter is set to `Specify via Fixed-point tab`.



Compute statistics for connected regions in the binary image, BW.

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

Fixed-point operational parameters

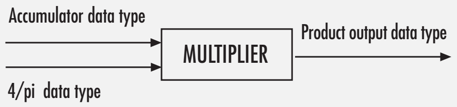
Rounding mode: Floor   Overflow mode: Wrap

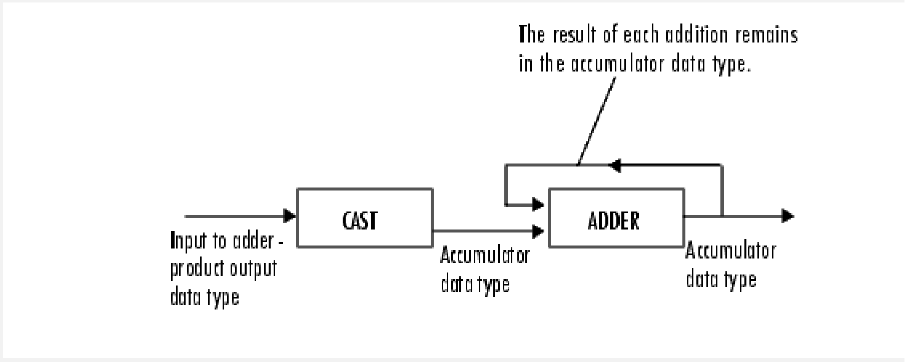
Fixed-point data types

	Mode	Signed	Word length	Fraction length
Product output	Binary point scaling	yes	32	16
Accumulator	Binary point scaling	yes	32	0
Centroid output	Binary point scaling	yes	32	16
Equiv Diam^2 output	Same as product output			
Extent output	Binary point scaling	yes	16	14
Perimeter output	Binary point scaling	yes	32	16

☐ Lock scaling against changes by the autoscaling tool

Parameter	Value
Rounding mode	Select the rounding mode `Floor`, `Ceiling`, `Nearest` or `Zero` for fixed-point operations.
Overflow mode	Select the overflow mode, `Wrap` or `Saturate` for fixed-point operations.

Parameter	Value
Product output	When you select `Binary point scaling`, you can enter the **Word length** and the **Fraction length** of the product output, in bits. When you select `Slope and bias scaling`, you can enter the **Word length** in bits, and the **Slope** of the product output. The bias of all signals in the Video and Image Processing Blockset software is 0.  Accumulator data type 4/pi data type MULTIPLIER Product output data type 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.
Accumulator	When you select `Same as product output` the characteristics match those of the product output. When you select `Binary point scaling`, you can enter the **Word length** and the **Fraction length** of the accumulator, in bits. When you select `Slope and bias scaling`, you can enter the **Word length**, in bits, and the **Slope** of the **Accumulator**. The bias of all signals in the Video and Image Processing Blockset software is 0.

Parameter	Value
	 The result of each addition remains in the accumulator data type. CAST ADDER Input to adder - product output data type Accumulator data type Accumulator data type 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:
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 **Fraction length** of the output, in bits. When you select `Slope and bias scaling`, you can enter the **Word length**, in bits, and the **Slope** of the output. The bias of all signals in the Video and Image Processing Blockset software is 0.

Parameter	Value
Equiv Diam^2 output	Choose how to specify the **Word length** and **Fraction length** of the output at the **Diameter^2** 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 **Fraction length** of the output, in bits. When you select `Slope and bias scaling`, you can enter the **Word length**, in bits, and the **Slope** of the output. The bias of all signals in the Video and Image Processing Blockset software is 0.
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 **Fraction length** of the output, in bits. When you select `Slope and bias scaling`, you can enter the **Word length**, in bits, and the **Slope** of the output. The bias of all signals in the Video and Image Processing Blockset software is 0.
Fill empty spaces in outputs	If checked, the block fills empty spaces in the statistical vectors with the value(s) you specify in the **Fill values** parameter. The **Fill empty spaces in outputs** check box will not appear when the **Output blob statistics as a variable-size signal** is checked.

Parameter	Value
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 **Fraction length** of the output, in bits. When you select `Slope and bias scaling`, you can enter the word length, in bits, and the **Slope** of the output. The bias of all signals in the Video and Image Processing Blockset software is 0.
Lock 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 software
Variable Selector	Signal Processing Blockset software
`regionprops`	Image Processing Toolbox software

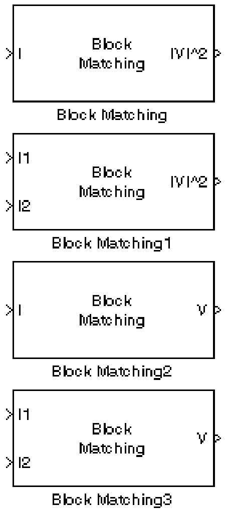
Purpose

Estimate motion between images or video frames

Library

Analysis & Enhancement

Description



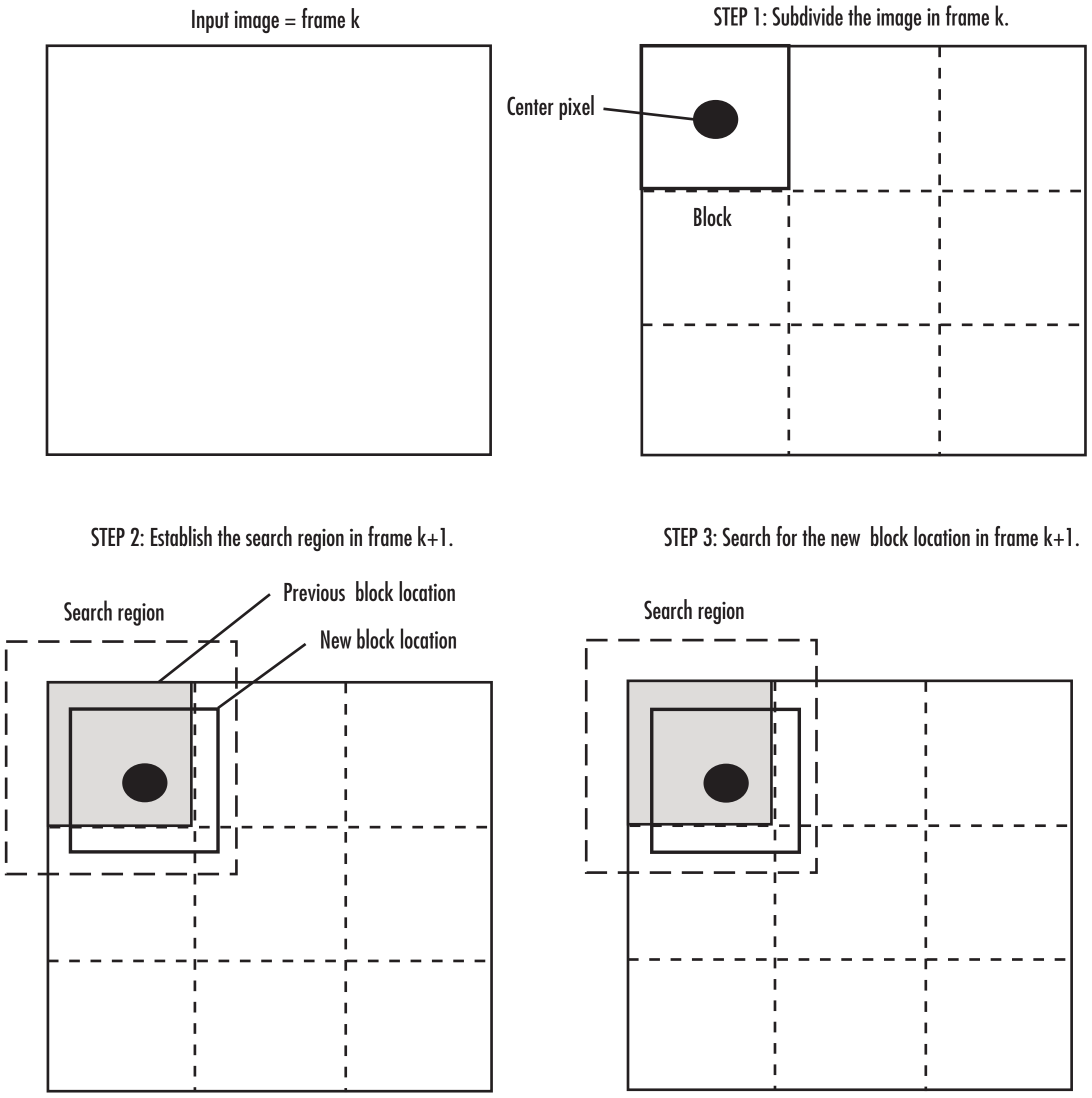
The Block Matching block estimates motion between two images or two video frames using “blocks” of pixels. The Block Matching block matches the block of pixels in frame k to a block of pixels in frame k+1 by moving the block of pixels over a search region.

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

1 The block subdivides this frame using the values you enter for the **Block size [height width]** and **Overlap [r c]** parameters. In the following example, the **Overlap [r c]** parameter is `[0 0]`.

2 For each subdivision or block in frame k+1, the Block Matching block establishes a search region based on the value you enter for the **Maximum displacement [r c]** parameter.

3 The block searches for the new block location using either the `Exhaustive` or `Three-step` search method.


Input image = frame k
STEP 1: Subdivide the image in frame k.
Center pixel
Block
STEP 2: Establish the search region in frame k+1.
Search region
Previous block location
New block location
STEP 3: Search for the new block location in frame k+1.
Search region

Port	Output	Supported Data Types	Complex Values Supported
I/I1	Scalar, vector, or matrix of intensity values	• 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:

$$MSE(d_1,d_2) = \frac{1}{N_1 \times N_2} \sum_{(n_1,n_2),\in B} \sum [s(n_1,n_2,k) - s(n_1+d_1,n_2+d_2,k+1)]^2$$

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:

$$MAD(d_1,d_2) = \frac{1}{N_1 \times N_2} \sum_{(n_1,n_2),\in B} \sum | s(n_1,n_2,k) - s(n_1+d_1,n_2+d_2,k+1) |$$

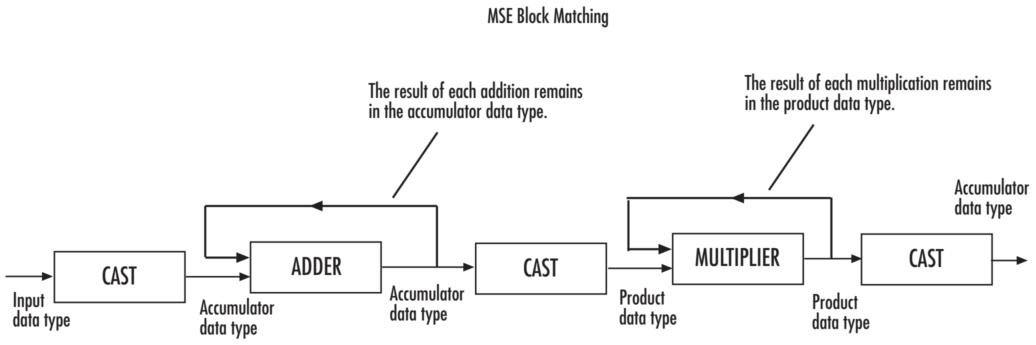
Use the **Block size [height width]** and **Overlap [r c]** parameters to specify how the block subdivides the input image. For a graphical description of these parameters, see the first figure in this reference page. If the **Overlap [r c]** parameter is not `[0 0]`, the blocks would overlap each other by the number of pixels you specify.

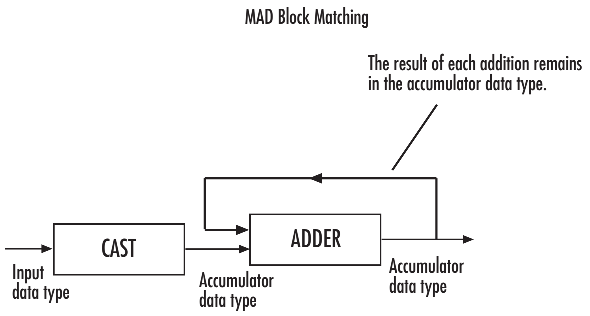
Use the **Maximum displacement [r c]** parameter to specify the maximum number of pixels any center pixel in a block of pixels might move from image to image or frame to frame. The block uses this value to determine the size of the search region.

Use the **Velocity output** parameter to specify the block's output. If you select `Magnitude-squared`, the block outputs the optical flow matrix where each element is of the form u^2+v^2. If you select `Horizontal and vertical components in complex form`, the block outputs the optical flow matrix where each element is of the form $u + jv$. The real part of each value is the horizontal velocity component and the imaginary part of each value is the vertical velocity component.

Fixed-Point Data Types

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

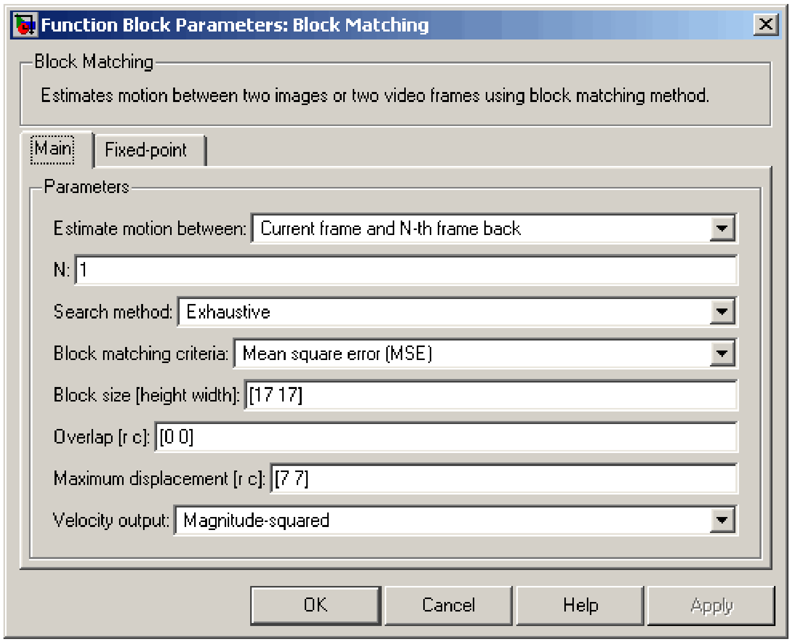




You can set the accumulator and output data types in the block mask as discussed in the next section.

Dialog Box

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



Block Matching

Estimates motion between two images or two video frames using block matching method.

Parameters

Estimate motion between: Current frame and N-th frame back

N: 1

Search method: Exhaustive

Block matching criteria: Mean square error (MSE)

Block size [height width]: [17 17]

Overlap [r c]: [0 0]

Maximum displacement [r c]: [7 7]

Velocity output: Magnitude-squared

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)`.

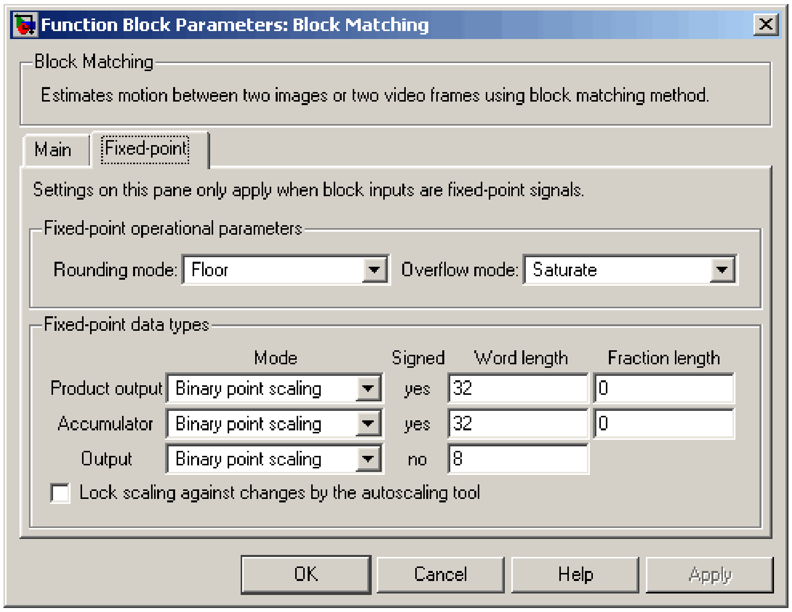
Block size [height width]
: Specify the size of the block of pixels.

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.



Function Block Parameters: Block Matching

Block Matching

Estimates motion between two images or two video frames using block matching method.

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: Saturate

Fixed-point data types

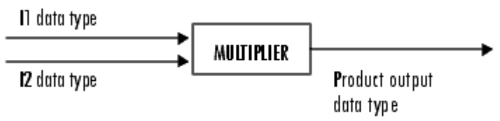
	Mode	Signed	Word length	Fraction length
Product output	Binary point scaling	yes	32	0
Accumulator	Binary point scaling	yes	32	0
Output	Binary point scaling	no	8	

☐ Lock scaling against changes by the autoscaling tool

Rounding mode
Select the rounding mode for fixed-point operations.

Overflow mode
Select the overflow mode for fixed-point operations.

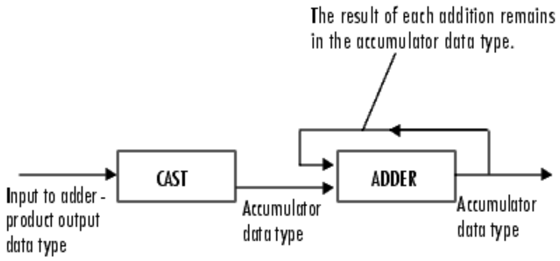
Product output



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 the Video and Image Processing Blockset software is 0.

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type

As depicted previously, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

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

Output

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

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

Lock 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 software

Block Processing

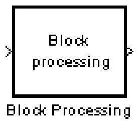
Purpose

Repeat user-specified operation on submatrices of input matrix

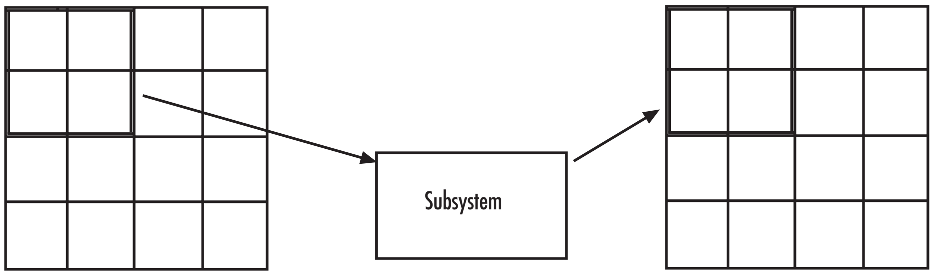
Library

Utilities

Description



The Block Processing block extracts submatrices of a user-specified size from each input matrix. It sends each submatrix to a subsystem for processing, and then reassembles each subsystem output into the output matrix.



Note Because you modify the Block Processing block's subsystem, the link between this block and the block library is broken when you click-and-drag a Block Processing block into your model. As a result, this block will not be automatically updated if you upgrade to a newer version of the Video and Image Processing Blockset software. If you right-click on the block and select **Look under mask**, you can delete blocks from this subsystem without triggering a warning. Lastly, if you search for library blocks in a model, this block will not be part of the results.

The blocks inside the subsystem dictate the frame status of the input and output signals, whether single channel or multichannel signals are supported, and which data types are supported by this block.

Use the **Number of inputs** and **Number of outputs** parameters to specify the number of input and output ports on the Block Processing block.

Use the **Block size** parameter to specify the size of each submatrix in cell array format. Each vector in the cell array corresponds to one input; the block uses the vectors in the order you enter them. If you have one input port, enter one vector. If you have more than one input port, you can enter one vector that is used for all inputs or you can specify a different vector for each input. For example, if you want each submatrix to be 2-by-3, enter `{[2 3]}`.

Use the **Overlap** parameter to specify the overlap of each submatrix in cell array format. Each vector in the cell array corresponds to the overlap of one input; the block uses the vectors in the order they are specified. If you enter one vector, each overlap is the same size. For example, if you want each 3-by-3 submatrix to overlap by 1 row and 2 columns, enter `{[1 2]}`.

The **Traverse order** parameter determines how the block extracts submatrices from the input matrix. If you select `Row-wise`, the block extracts submatrices by moving across the rows. If you select `Column-wise`, the block extracts submatrices by moving down the columns.

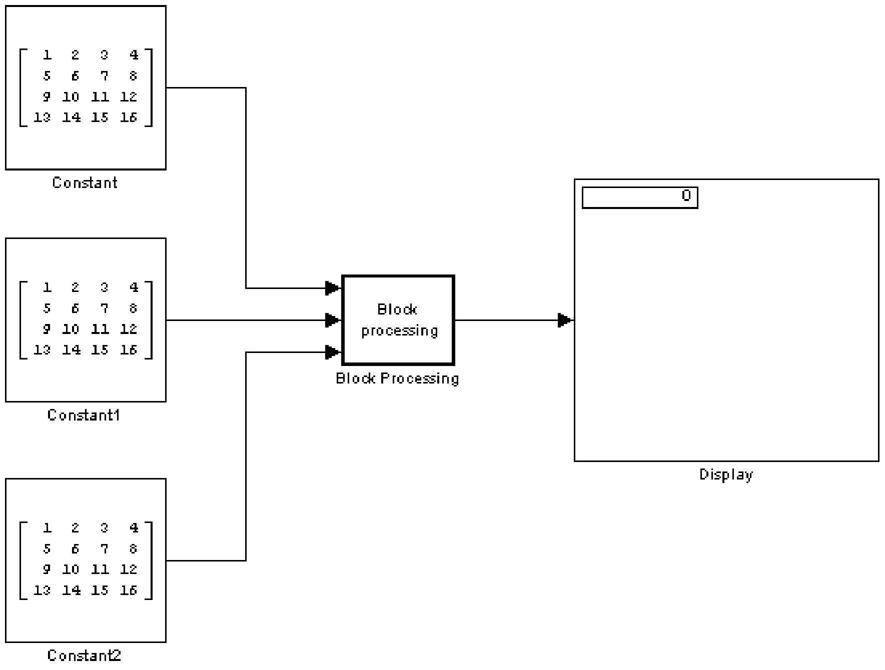
Click the **Open Subsystem** button to open the block's subsystem. Click-and-drag blocks into this subsystem to define the processing operation(s) the block performs on the submatrices. The input to this subsystem are the submatrices whose size is determined by the **Block size** parameter.

Note When you place an Assignment block inside a Block Processing block's subsystem, the Assignment block behaves as though it is inside a For Iterator block. For a description of this behavior, see the "Iterated Assignment" section of the Assignment block reference page. To achieve the normal behavior of the Assignment block, use an Overwrite Values block inside the Block Processing block's subsystem.

Example

Example 1 – Multiple Inputs

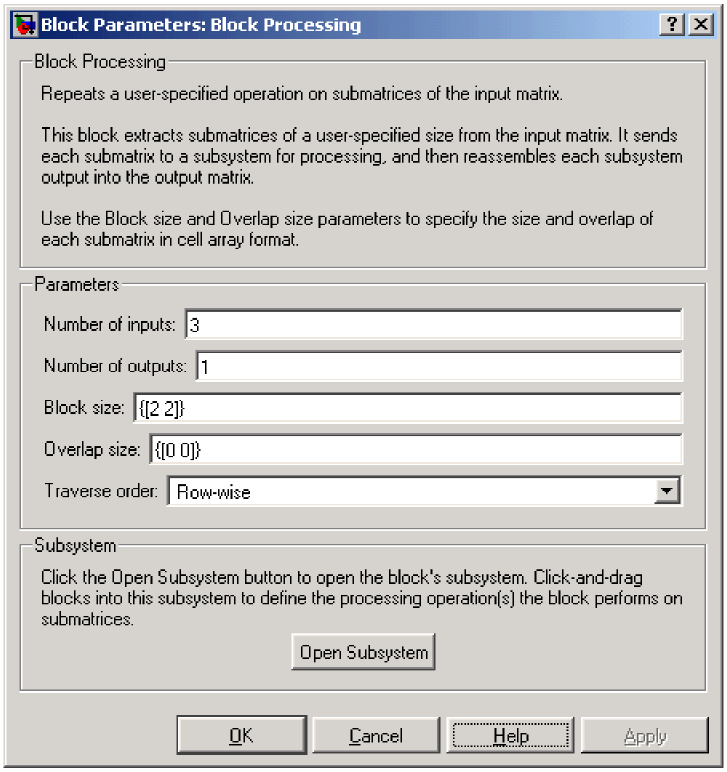
In this example, you multiply each element of three input matrices by two and add the results using the Block Processing block. Suppose you have the following model:



$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$
Constant
Constant1
Constant2
Block processing
Block Processing
0
Display

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 Parameters: Block Processing

Block Processing

Repeats a user-specified operation on submatrices of the input matrix.

This block extracts submatrices of a user-specified size from the input matrix. It sends each submatrix to a subsystem for processing, and then reassembles each subsystem output into the output matrix.

Use the Block size and Overlap size parameters to specify the size and overlap of each submatrix in cell array format.

Parameters

Number of inputs: 3

Number of outputs: 1

Block size: {[2 2]}

Overlap size: {[0 0]}

Traverse order: Row-wise

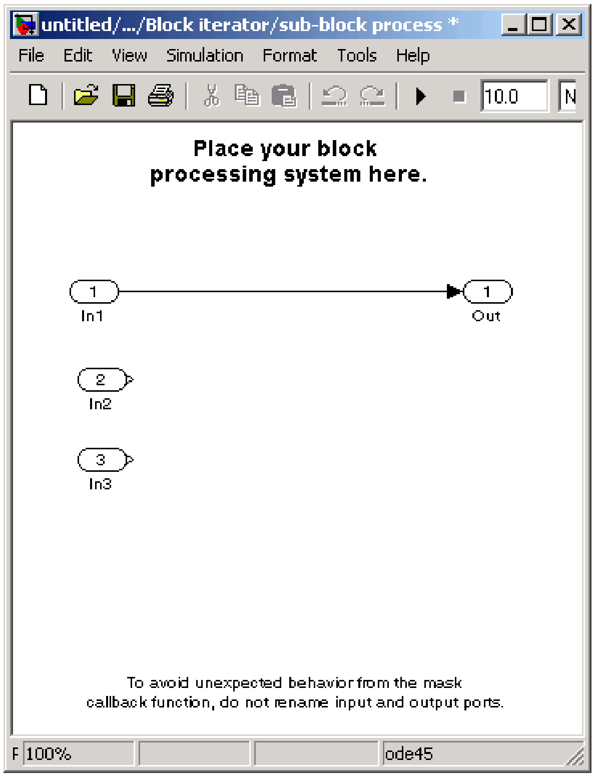
Subsystem

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 submatrices.

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.


untitled/.../Block iterator/sub-block process *
File Edit View Simulation Format Tools Help
10.0
1
In1
1
Out
2
In2
3
In3
100%
ode45

Place your block processing system here.

To avoid unexpected behavior from the mask callback function, do not rename input and output ports.

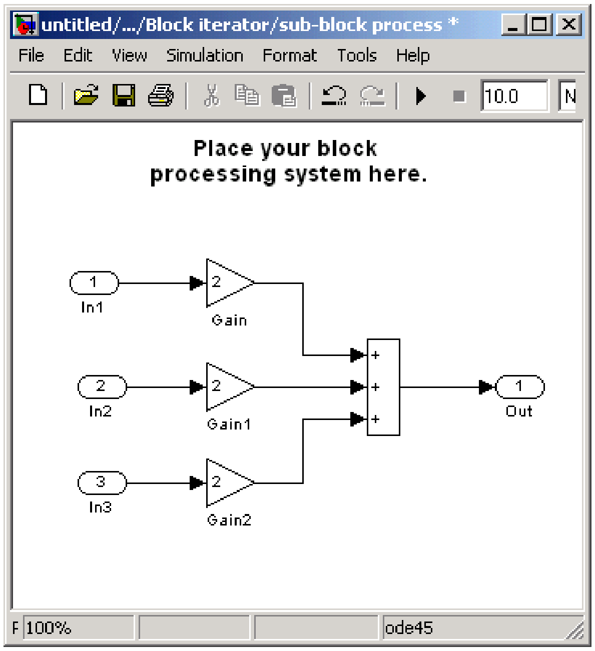
3 Click and drag the following blocks into the subsystem:

Block	Library	Quantity
Gain	Simulink / Math Operations	3
Sum	Simulink / Math Operations	1

4 Use the Gain blocks to multiply the elements of each submatrix by two. Set the **Gain** parameter to `2`.

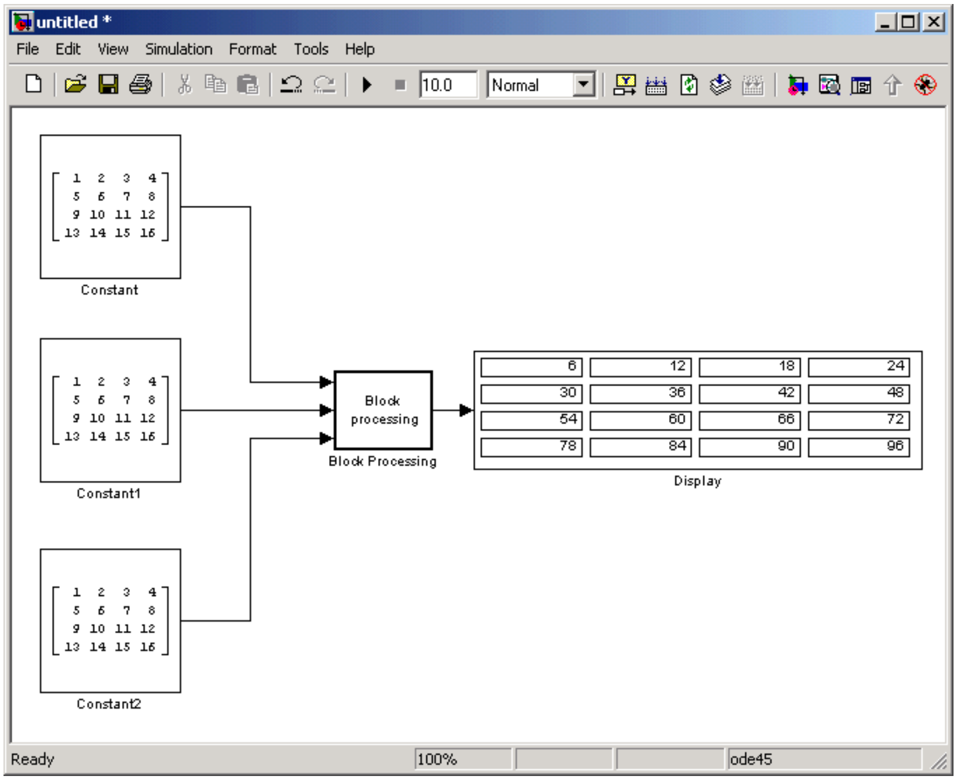
5 Use the Sum block to add the values. Set the **Icon shape** parameter to `rectangular` and the **List of signs** parameter to `+++`.

6 Connect the blocks as shown in the following figure.


untitled/.../Block iterator/sub-block process *
File Edit View Simulation Format Tools Help
10.0
Place your block processing system here.
1
In1
2
Gain
2
In2
2
Gain1
3
In3
2
Gain2
+
+
+
1
Out
100%
ode45

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

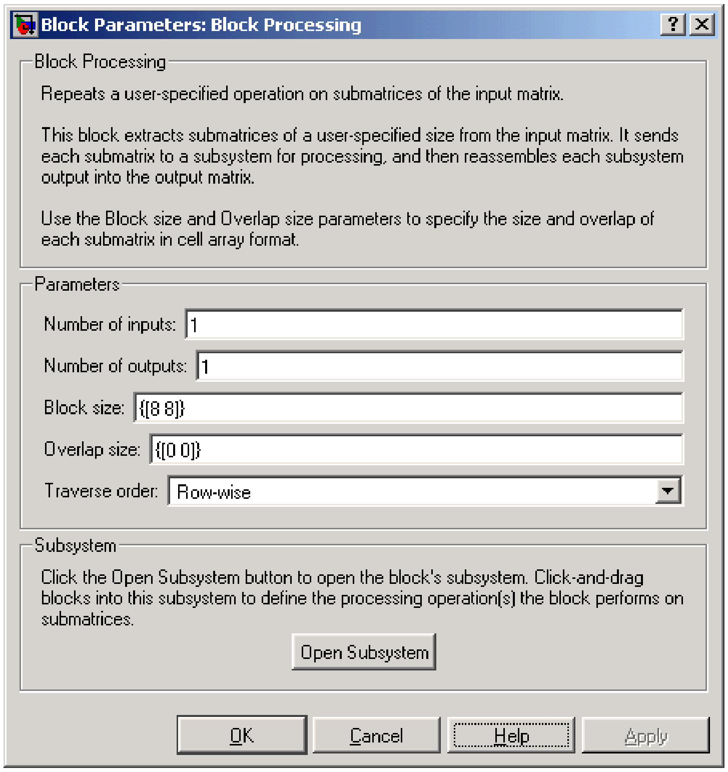
8 Run the model.



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

Dialog Box

The Block Processing dialog box appears as shown in the following figure.



Block Parameters: Block Processing

Block Processing

Repeats a user-specified operation on submatrices of the input matrix.

This block extracts submatrices of a user-specified size from the input matrix. It sends each submatrix to a subsystem for processing, and then reassembles each subsystem output into the output matrix.

Use the Block size and Overlap size parameters to specify the size and overlap of each submatrix in cell array format.

Parameters

Number of inputs: 1

Number of outputs: 1

Block size: {[8 8]}

Overlap size: {[0 0]}

Traverse order: Row-wise

Subsystem

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 submatrices.

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

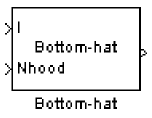
Purpose

Perform bottom-hat filtering on intensity or binary images

Library

Morphological Operations

Description



Use the Bottom-hat block to perform bottom-hat filtering on an intensity or binary image using a predefined neighborhood or structuring element. Bottom-hat filtering is the equivalent of subtracting the input image from the result of performing a morphological closing operation on the input image. This block uses flat structuring elements only.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	• 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`.

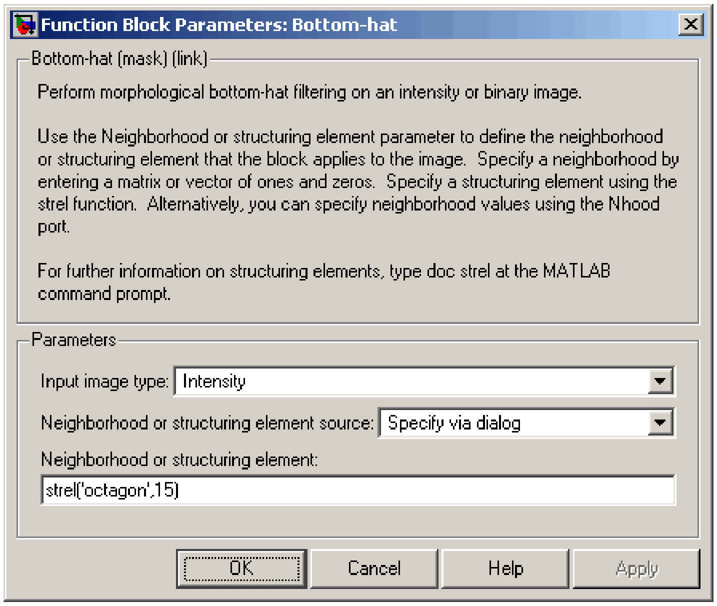
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select `Specify via dialog`, the **Neighborhood or structuring**

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 the Image Processing Toolbox. If the structuring element is decomposable into smaller elements, the block executes at higher speeds due to the use of a more efficient algorithm.

Dialog Box

The Bottom-hat dialog box appears as shown in the following figure.



Bottom-hat (mask) (link)

Perform morphological bottom-hat filtering on an intensity or binary image.

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 ones and zeros. Specify a structuring element using the strel function. Alternatively, you can specify neighborhood values using the Nhood port.

For further information on structuring elements, type doc strel at the MATLAB command prompt.

Parameters

Input image type: Intensity

Neighborhood or structuring element source: Specify via dialog

Neighborhood or structuring element:

strel('octagon',15)

Input image type

If your input image is a binary image, select `Binary`. If your input image is an intensity image, select `Intensity`.

Neighborhood or structuring element source

Specify how to enter your neighborhood or structuring element values. Select `Specify via dialog` to enter the values in the dialog box. Select `Input port` to use the Nhood port to specify the neighborhood values. You can only specify a structuring element using the dialog box.

Neighborhood or structuring element

If you are specifying a neighborhood, this parameter must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the `strel` function from the Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or structuring element source** parameter, you select `Specify via dialog`.

See Also

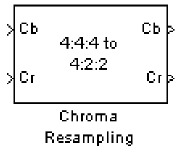
Closing	Video and Image Processing Blockset software
Dilation	Video and Image Processing Blockset software
Erosion	Video and Image Processing Blockset software
Label	Video and Image Processing Blockset software
Opening	Video and Image Processing Blockset software
Top-hat	Video and Image Processing Blockset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

Chroma Resampling

Purpose Downsample or upsample chrominance components of images

Library Conversions

Description The Chroma Resampling block downsamples or upsamples chrominance components of pixels to reduce the bandwidth required for transmission or storage of a signal.

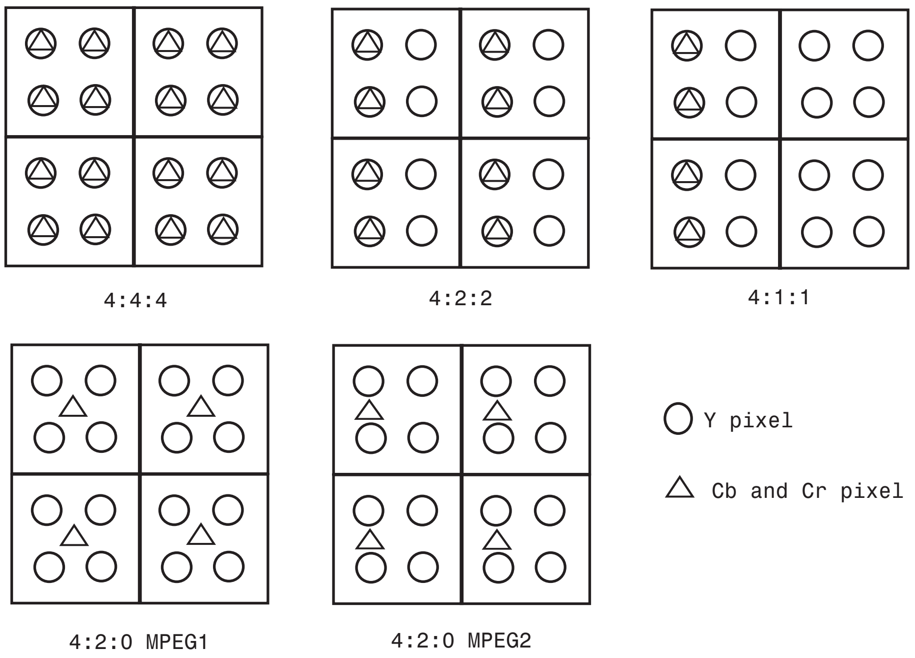


Port	Input/Output	Supported Data Types	Complex Values Supported
Cb	Matrix that represents one chrominance component of an image	• Double-precision floating point • Single-precision floating point • 8-bit unsigned integer	No
Cr	Matrix that represents one chrominance component of an image	Same as Cb port	No

The data type of the output signals is the same as the data type of the input signals.

Chroma Resampling Formats

The Chroma Resampling block supports the formats shown in the following diagram.



Downsampling

If, for the **Resampling** parameter, you select `4:4:4 to 4:2:2`, `4:4:4 to 4:2:0 (MPEG1)`, `4:4:4 to 4:2:0 (MPEG2)`, `4:4:4 to 4:1:1`, `4:2:2 to 4:2:0 (MPEG1)`, or `4:2:2 to 4:2:0 (MPEG2)`, the block performs a downsampling operation. When the block downsamples from one format to another, it can bandlimit the input signal by applying a lowpass filter to prevent aliasing.

If, for the **Antialiasing filter** parameter, you select `Default`, the block uses a built-in lowpass filter to prevent aliasing.

If, for the **Resampling** parameter, you select `4:4:4 to 4:2:2`, `4:4:4 to 4:2:0 (MPEG1)`, `4:4:4 to 4:2:0 (MPEG2)`, or `4:4:4 to 4:1:1` and, for the **Antialiasing filter** parameter, you select

`User-defined`, the **Horizontal filter coefficients** parameter appears on the dialog box. Enter the filter coefficients to apply to your input.

If, for the **Resampling** parameter, you select `4:4:4 to 4:2:0 (MPEG1)`, `4:4:4 to 4:2:0 (MPEG2)`, `4:2:2 to 4:2:0 (MPEG1)`, or `4:2:2 to 4:2:0 (MPEG2)` and, for the **Antialiasing filter** parameter, you select `User-defined`. **Vertical filter coefficients** parameters appear on the dialog box. Enter an even number of filter coefficients to apply to your input signal.

If, for the **Antialiasing filter** parameter, you select `None`, the block does not filter the input signal.

Upsampling

If, for the **Resampling** parameter, you select `4:2:2 to 4:4:4`, `4:2:0 (MPEG1) to 4:2:2`, `4:2:0 (MPEG1) to 4:4:4`, `4:2:0 (MPEG2) to 4:2:2`, `4:2:0 (MPEG2) to 4:4:4`, or `4:1:1 to 4:4:4`, the block performs an upsampling operation.

When the block upsamples from one format to another, it uses interpolation to approximate the missing chrominance values. If, for the **Interpolation** parameter, you select `Linear`, the block uses linear interpolation to calculate the missing values. If, for the **Interpolation** parameter, you select `Pixel replication`, the block replicates the chrominance values of the neighboring pixels to create the upsampled image.

Row-Major Data Format

The MATLAB environment and the Video and Image Processing Blockset software use column-major data organization. However, the Chroma Resampling block gives you the option to process data that is stored in row-major format. When you select the **Input image is transposed (data order is row major)** check box, the block assumes that the input buffer contains contiguous data elements from the first row first, then data elements from the second row second, and so on through the last row. Use this functionality only when you meet all the following criteria:

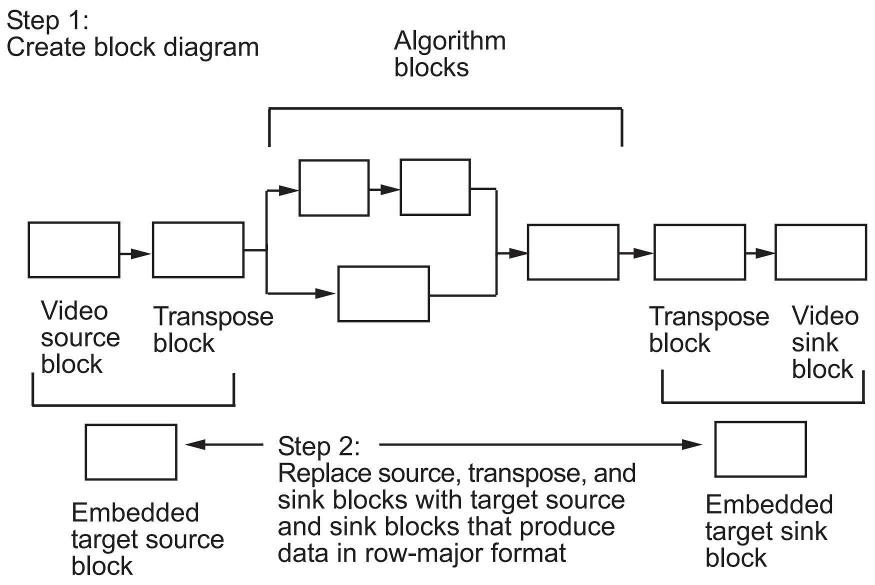
- You are developing algorithms to run on an embedded target that uses the row-major format.
- You want to limit the additional processing required to take the transpose of signals at the interfaces of the row-major and column-major systems.

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

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

  For example, if you use the 2-D FIR Filter block, you need to verify that your filter coefficients are transposed. If you are using the Rotate block, you need to use negative rotation angles, etc.
- Only three blocks have the **Input image is transposed (data order is row major)** check box. They are the Chroma Resampling, Deinterlacing, and Insert Text blocks. You need to select this check box to enable row-major functionality in these blocks. All other blocks must be properly configured to process data in row-major format.

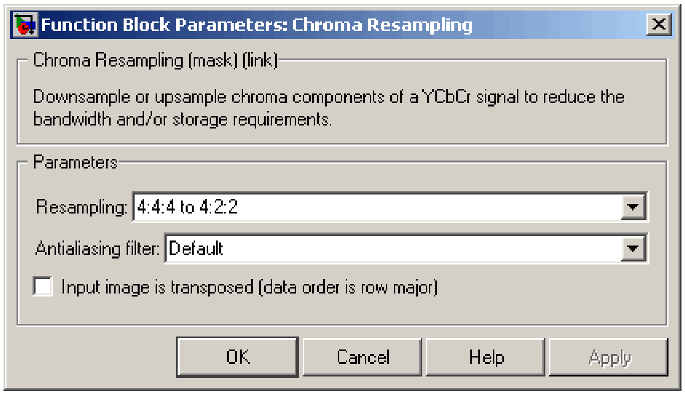
Use the following two-step workflow to develop algorithms in row-major format to run on an embedded target.


Step 1:
Create block diagram
Algorithm blocks
Video source block
Transpose block
Transpose block
Video sink block
Embedded target source block
Step 2:
Replace source, transpose, and sink blocks with target source and sink blocks that produce data in row-major format
Embedded target sink block

See the DM642 EVM Video ADC and DM642 EVM Video DAC reference pages in the *Target Support Package™ TC6 User's Guide* for more information about data order in embedded targets.

Dialog Box

The Chroma Resampling dialog box appears as shown in the following figure.



Function Block Parameters: Chroma Resampling

Chroma Resampling (mask) (link)

Downsample or upsample chroma components of a YCbCr signal to reduce the bandwidth and/or storage requirements.

Parameters

Resampling: 4:4:4 to 4:2:2

Antialiasing filter: Default

☐ Input image is transposed (data order is row major)

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 4:2:2` , or `4:2:0 (MPEG2) to 4:2:2` .

Input image is transposed (data order is row major)

When you select this check box, the block assumes that the input buffer contains data elements from the first row first, then data elements from the second row second, and so on through the last row.

References

[1] Haskell, Barry G., Atul Puri, and Arun N. Netravali. *Digital Video: An Introduction to MPEG-2.* New York: Chapman & Hall, 1996.

[2] Recommendation ITU-R BT.601-5, Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide Screen 16:9 Aspect Ratios.

[3] Wang, Yao, Jorn Ostermann, Ya-Qin Zhang. *Video Processing and Communications.* Upper Saddle River, NJ: Prentice Hall, 2002.

See Also

Autothreshold	Video and Image Processing Blockset software
Color Space Conversion	Video and Image Processing Blockset software
Image Complement	Video and Image Processing Blockset software

Closing

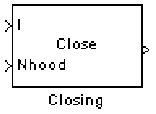
Purpose

Perform morphological closing on binary or intensity images

Library

Morphological Operations

Description



The Closing block performs a dilation operation followed by an erosion operation using a predefined neighborhood or structuring element. This block uses flat structuring elements only.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
Nhood	Matrix or vector of ones and zeros that represents the neighborhood values	Boolean	No
Output	Vector or matrix of intensity values that represents the closed image	Same as I port	No

The output signal has the same data type as the input to the I port.

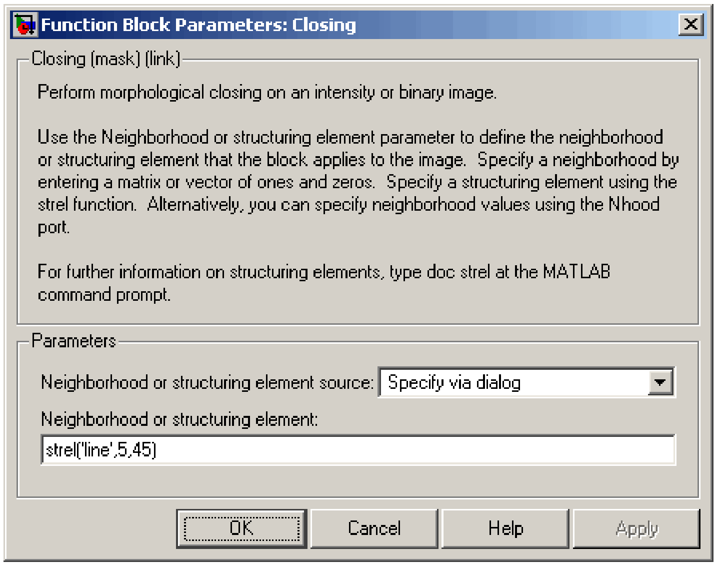
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values.

If you select `Specify via dialog`, the **Neighborhood or structuring element** parameter appears in the dialog box. If you select `Input port`, the Nhood port appears on the block. Use this port to enter your neighborhood values as a matrix or vector of 1s and 0s. You can only specify a structuring element using the dialog box.

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

Dialog Box

The Closing dialog box appears as shown in the following figure.



Perform morphological closing on an intensity or binary image.

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 ones and zeros. Specify a structuring element using the strel function. Alternatively, you can specify neighborhood values using the Nhood port.

For further information on structuring elements, type doc strel at the MATLAB command prompt.

Parameters

Neighborhood or structuring element source: Specify via dialog

Neighborhood or structuring element: strel('line',5,45)

Neighborhood or structuring element source

Specify how to enter your neighborhood or structuring element values. Select `Specify via dialog` to enter the values in the dialog box. Select `Input port` to use the Nhood port to specify the neighborhood values. You can only specify a structuring element using the dialog box.

Neighborhood or structuring element
If you are specifying a neighborhood, this parameter must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the `strel` function from the Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or structuring element source** parameter, you select `Specify via dialog`.

References

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

See Also

Bottom-hat	Video and Image Processing Blockset software
Dilation	Video and Image Processing Blockset software
Erosion	Video and Image Processing Blockset software
Label	Video and Image Processing Blockset software
Opening	Video and Image Processing Blockset software
Top-hat	Video and Image Processing Blockset software
`imclose`	Image Processing Toolbox software
`strel`	Image Processing Toolbox software

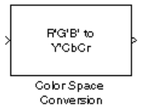
Purpose

Convert color information between color spaces

Library

Conversions

Description



The Color Space Conversion block converts color information between color spaces. Use the **Conversion** parameter to specify the color spaces you are converting between. Your choices are `R'G'B' to Y'CbCr`, `Y'CbCr to R'G'B'`, `R'G'B' to intensity`, `R'G'B' to HSV`, `HSV to R'G'B'`, `sR'G'B' to XYZ`, `XYZ to sR'G'B'`, `sR'G'B' to L*a*b*`, and `L*a*b* to sR'G'B'`.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input / Output	M-by-N-by-P color video signal where P is the number of color planes	• 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

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	• Double-precision floating point • Single-precision floating point	No
S	Matrix that represents represent the saturation component of an image	Same as the H port	No
V	Matrix that represents the value (brightness) component of an image	Same as the H port	No
X	Matrix that represents the X component of an image	Same as the H port	No
Y	Matrix that represents the Y component of an image	Same as the H port	No
Z	Matrix that represents the Z component of an image	Same as the H port	No
L*	Matrix that represents the luminance portion of an image	Same as the H port	No
a*	Matrix that represents the a* component of an image	Same as the H port	No
b*	Matrix that represents the b* component of an image	Same as the H port	No

The data type of the output signal is the same as the data type of the input signal.

Use the **Image signal** parameter to specify how to input and output a color video signal. If you select `One multidimensional signal`, the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select `Separate color signals`, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

Note The prime notation indicates that the signals are gamma corrected.

Conversion Between R'G'B' and Y'CbCr Color Spaces

The R'G'B' to Y'CbCr conversion and the Y'CbCr to R'G'B' conversion are defined by the following equations:

$$\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + A \times \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = B \times \left(\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \right)$$

The values in the A and B matrices are based on your choices for the **Use conversion specified by** and **Scanning standard** parameters. The following table summarizes the possible values:

Matrix	Use conversion specified by = Rec. 601 (SDTV)	Use conversion specified by = Rec. 709 (HDTV)	
		Scanning standard = 1125/60/2:1	Scanning standard = 1250/50/2:1
A	$\begin{bmatrix} 0.25678824 & 0.50412941 & 0.09790588 \\ -0.1482229 & -0.29099279 & 0.43921569 \\ 0.43921569 & -0.36778831 & -0.07142737 \end{bmatrix}$	$\begin{bmatrix} 0.18258588 & 0.61423059 & 0.06200706 \\ -0.10064373 & -0.33857195 & 0.43921569 \\ 0.43921569 & -0.39894216 & -0.04027352 \end{bmatrix}$	$\begin{bmatrix} 0.25678824 & 0.50412941 & 0.09790588 \\ -0.1482229 & -0.29099279 & 0.43921569 \\ 0.43921569 & -0.36778831 & -0.07142737 \end{bmatrix}$
B	$\begin{bmatrix} 1.1643836 & 0 & 1.5960268 \\ 1.1643836 & -0.39176229 & -0.81296765 \\ 0.16438356 & 2.0172321 & 0 \end{bmatrix}$	$\begin{bmatrix} 1.16438356 & 0 & 1.79274107 \\ 1.16438356 & -0.21324861 & -0.53290933 \\ 1.16438356 & 2.11240179 & 0 \end{bmatrix}$	$\begin{bmatrix} 1.1643836 & 0 & 1.5960268 \\ 1.1643836 & -0.39176229 & -0.81296765 \\ 0.16438356 & 2.0172321 & 0 \end{bmatrix}$

Conversion R'B'G' to Intensity

The conversion from the R'B'G' color space to intensity is defined by the following equation:

$$\text{intensity} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

Conversion Between R'G'B' and HSV Color Spaces

The R'G'B' to HSV conversion is defined by the following equations. In these equations, *MAX* and *MIN* represent the maximum and minimum values of each R'G'B' triplet, respectively. *H*, *S*, and *V* vary from 0 to 1, where 1 represents the greatest saturation and value.

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

$$S = \frac{MAX - MIN}{MAX}$$

$$V = MAX$$

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

$$H_i = \lfloor 6H \rfloor$$
$$f = 6H - H_i$$
$$p = 1 - S$$
$$q = 1 - fS$$
$$t = 1 - (1 - f)S$$
$$\text{if} \quad H_i = 0, \quad R_{tmp} = 1, \quad G_{tmp} = t, \quad B_{tmp} = p$$
$$\text{if} \quad H_i = 1, \quad R_{tmp} = q, \quad G_{tmp} = 1, \quad B_{tmp} = p$$
$$\text{if} \quad H_i = 2, \quad R_{tmp} = p, \quad G_{tmp} = 1, \quad B_{tmp} = t$$
$$\text{if} \quad H_i = 3, \quad R_{tmp} = p, \quad G_{tmp} = q, \quad B_{tmp} = 1$$
$$\text{if} \quad H_i = 4, \quad R_{tmp} = t, \quad G_{tmp} = p, \quad B_{tmp} = 1$$
$$\text{if} \quad H_i = 5, \quad R_{tmp} = 1, \quad G_{tmp} = p, \quad B_{tmp} = q$$
$$u = V / \max(R_{tmp}, G_{tmp}, B_{tmp})$$
$$R' = uR_{tmp}$$
$$G' = uG_{tmp}$$
$$B' = uB_{tmp}$$

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

Conversion Between sR'G'B' and XYZ Color Spaces

The sR'G'B' to XYZ conversion is a two-step process. First, the block converts the gamma-corrected sR'G'B' values to linear sRGB values using the following equations:

$$\begin{aligned}
&\text{If} \quad R'_{sRGB}, G'_{sRGB}, B'_{sRGB} \le 0.03928 \\
&R_{sRGB} = R'_{sRGB} / 12.92 \\
&G_{sRGB} = G'_{sRGB} / 12.92 \\
&B_{sRGB} = B'_{sRGB} / 12.92 \\
&\text{otherwise,} \quad \text{if} \quad R'_{sRGB}, G'_{sRGB}, B'_{sRGB} > 0.03928 \\
&R_{sRGB} = \left[{(R'_{sRGB} + 0.055)} \big/ {1.055} \right]^{2.4} \\
&G_{sRGB} = \left[{(G'_{sRGB} + 0.055)} \big/ {1.055} \right]^{2.4} \\
&B_{sRGB} = \left[{(B'_{sRGB} + 0.055)} \big/ {1.055} \right]^{2.4}
\end{aligned}$$

Then the block converts the sRGB values to XYZ values using the following equation:

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

The XYZ to sR'G'B' conversion is also a two-step process. First, the block converts the XYZ values to linear sRGB values using the following equation:

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

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

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

Note Video and Image Processing Blockset software uses a D65 white point, which is specified in Recommendation ITU-R BT.709, for this conversion. In contrast, the Image Processing Toolbox conversion is based on ICC profiles, and it uses a D65 to D50 Bradford adaptation transformation to the D50 white point. If you are using these two products and comparing results, you must account for this difference.

Conversion Between sR'G'B' and L*a*b* Color Spaces

The Color Space Conversion block converts sR'G'B' values to L*a*b* values in two steps. First it converts sR'G'B' to XYZ values using the equations described in "Conversion Between sR'G'B' and XYZ Color Spaces" on page 2-248. Then it uses the following equations to transform the XYZ values to L*a*b* values. Here, X_n, Y_n, and Z_n are the tristimulus values of the reference white point you specify using the **White point** parameter:

$$L^* = 116(Y/Y_n)^{1/3} - 16, \textit{for} \quad Y/Y_n > 0.008856$$
$$L^* = 903.3\, Y/Y_n, \quad \text{otherwise}$$

$$a^* = 500(f(X/X_n) - f(Y/Y_n))$$
$$b^* = 200(f(Y/Y_n) - f(Z/Z_n)),$$
$$\text{where} \quad f(t) = t^{1/3}, \textit{for} \quad t > 0.008856$$
$$f(t) = 7.787t + 16/166, \quad \text{otherwise}$$

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:

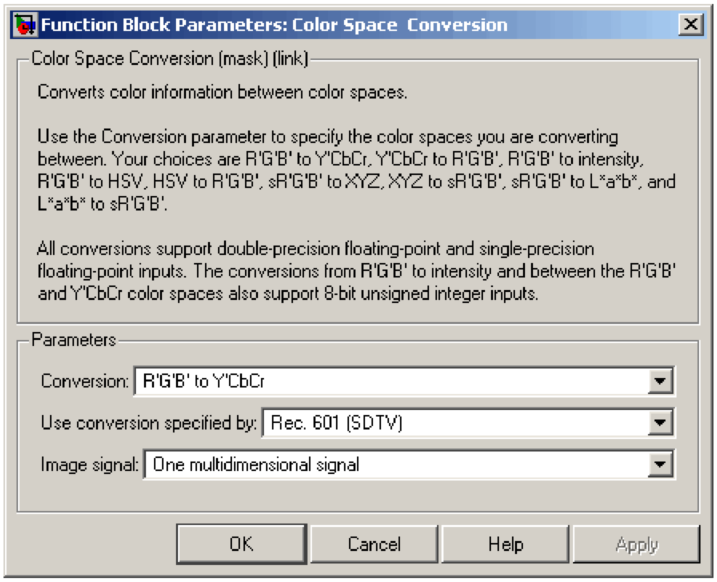
$$\text{For} \quad Y/Y_n > 0.008856$$

$$X = X_n(P + a^*/500)^3$$
$$Y = Y_n P^3$$
$$Z = Z_n(P - b^*/200)^3,$$

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

Dialog Box

The Color Space Conversion dialog box appears as shown in the following figure.



Color Space Conversion (mask) (link)

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'.

All conversions support double-precision floating-point and single-precision floating-point inputs. The conversions from R'G'B' to intensity and between the R'G'B' and Y'CbCr color spaces also support 8-bit unsigned integer inputs.

Parameters

Conversion: R'G'B' to Y'CbCr

Use conversion specified by: Rec. 601 (SDTV)

Image signal: One multidimensional signal

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	Video and Image Processing Blockset software
`rgb2hsv`	MATLAB software

`hsv2rgb`	MATLAB software
`rgb2ycbcr`	Image Processing Toolbox software
`ycbcr2rgb`	Image Processing Toolbox software
`rgb2gray`	Image Processing Toolbox software
`makecform`	Image Processing Toolbox software
`applycform`	Image Processing Toolbox software

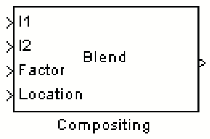
Purpose

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

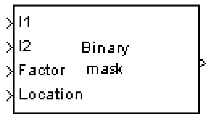
Library

Text & Graphics

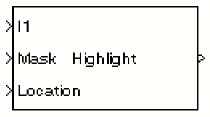
Description



Compositing



Compositing



Compositing

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 $0 \le X \le 1$, defines the amount by which to scale each pixel value before combining them.

You can use the Compositing block to overlay a Image 2 over Image 1. The masking factor and the location determine which Image 1 pixels are overwritten. The masking factor(s) can be 0 or 1, where 0 corresponds to not overwriting pixels and 1 corresponds to overwriting pixels.

You can use the Compositing block to highlight selected pixels in the input image. Use a binary image, input at the Mask port, to specify which pixels to highlight.

Note This block supports intensity and color images on its ports.

Port	Input/Output	Supported Data Types	Complex Values Supported
Image 1	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean	No

Port	Input/Output	Supported Data Types	Complex Values Supported
		• 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	
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	• 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
Mask	Binary image that specifies which pixels to highlight	Same as Factor port When the **Operation** parameter is set to `Highlight selected pixel`, the input to the Mask port must be a Boolean data type.	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Location	Two-element vector that specifies the position of the upper-left corner of the image input at port I2	• Double-precision floating point. (Only supported if the input to the Image 1 and Image 2 ports is a floating-point data type.) • Single-precision floating point. (Only supported if the input to the Image 1 and Image 2 ports is a floating-point data type.) • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
Output	Vector or matrix of intensity or color values	Same as Image 1 port	No

Use the **Operation** parameter to specify the operation you want the block to perform. If you choose `Blend`, the block linearly combines the pixels of one image with another image. If you choose `Binary mask`, the block overwrites the pixel values of one image with the pixel values of another image. If you choose `Highlight selected pixel`, the block uses the binary image input at the Mask port to determine which pixels are set to the maximum value supported by their data type. For example, for every 1 in the binary image, the block sets the corresponding pixel in input image to the maximum value supported by its data type. For every 0 in the binary image, the block leaves the pixel value alone.

If, for the **Operation** parameter, you choose `Blend`, the **Opacity factor(s) source** parameter appears on the dialog box. Use this parameter to indicate where to specify the opacity factor(s).

- If you choose `Specify via dialog`, the **Opacity factor(s)** parameter appears on the dialog box. Use this parameter to define the amount by which the block scales each I2 pixel value before combining them with the Image 1 pixel values. You can enter a scalar value used for all pixels or a matrix of values that is the same size as Image 2.
- If you choose `Input port`, the Factor port appears on the block. The input to this port must be a scalar or matrix of values as described for the **Opacity factor(s)** parameter. If the input to the Image 1 and Image 2 ports is floating point, the input to this port must be the same floating-point data type.

If, for the **Operation** parameter, you choose `Binary mask`, the **Mask source** parameter appears on the dialog box. Use this parameter to indicate where to specify the masking factor(s).

- If you choose `Specify via dialog`, the **Mask** parameter appears on the dialog box. Use this parameter and the location of the I2 image to define which pixels are overwritten. You can enter 0 or 1, which is used for all pixels in I2, or a matrix of 0s and 1s that defines the factor for each I2 pixel.
- If you choose `Input port`, the Factor port appears on the block. The input to this port must be a 0 or 1 whose data type is Boolean or a matrix of 0s or 1s whose data type is Boolean as described for the **Mask** parameter.

Use the **Location source** parameter to specify where to enter the zero-based location of the upper-left corner of the image input at port I2.

- If you choose `Specify via dialog`, the **Location [row column]** parameter appears on the dialog box. Enter a two-element vector that specifies the row and column position of the upper-left corner of the image input at port Image 2 relative to the upper-left corner of the image input at port Image 1. Positive values move the image down and to the right; negative values move the image up and to the left. If the first element is greater than the number of rows in the

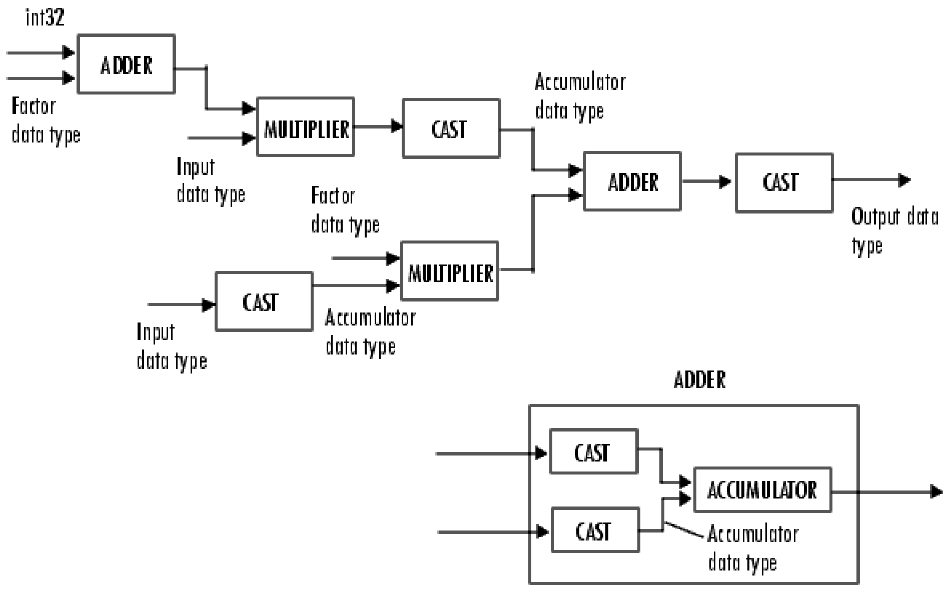
Image 1 matrix, the value is clipped to the total number of rows. If the second element is greater than the number of columns in the Image 1 matrix, the value is clipped to the total number of columns.

- If you choose `Input port`, the Location port appears on the block. The input to this port must be a two-element vector as described for the **Location [row column]** parameter.

If, for the **Operation** parameter, you choose `Highlight selected pixels`, the **Location source** parameter appears on the dialog box. This parameter is described above.

Fixed-Point Data Types

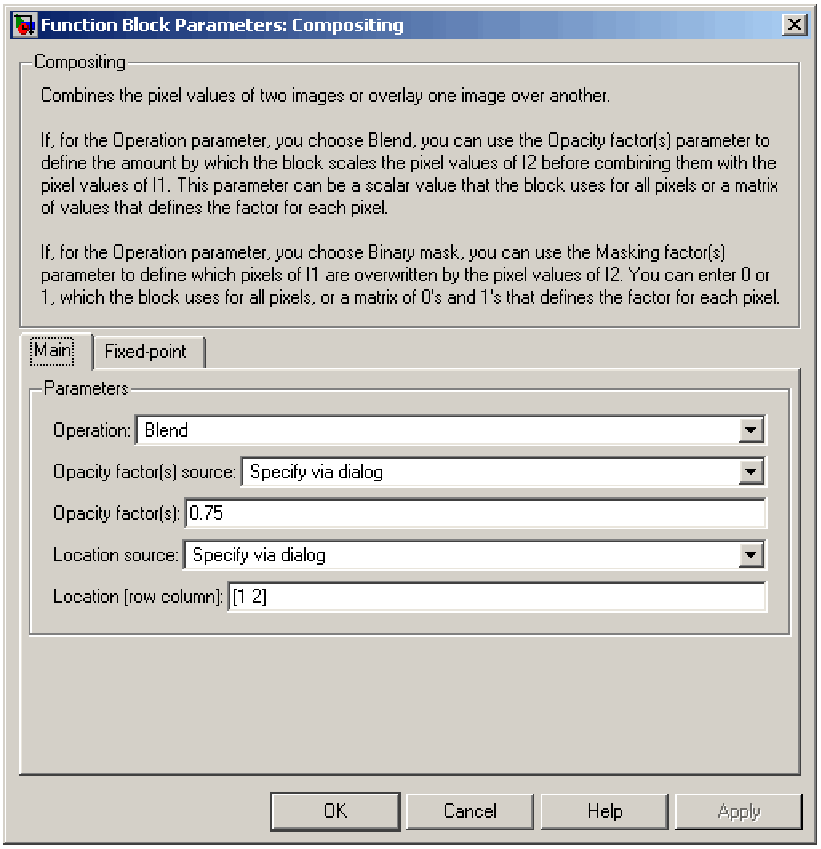
The following diagram shows the data types used in the Compositing block for fixed-point signals. It is only applicable when the **Operation** parameter is set to `Blend`.



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

Dialog Box

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



Function Block Parameters: Compositing

Compositing

Combines the pixel values of two images or overlay one image over another.

If, for the Operation parameter, you choose Blend, you can use the Opacity factor(s) parameter to define the amount by which the block scales the pixel values of I2 before combining them with the pixel values of I1. This parameter can be a scalar value that the block uses for all pixels or a matrix of values that defines the factor for each pixel.

If, for the Operation parameter, you choose Binary mask, you can use the Masking factor(s) parameter to define which pixels of I1 are overwritten by the pixel values of I2. You can enter 0 or 1, which the block uses for all pixels, or a matrix of 0's and 1's that defines the factor for each pixel.

Main | Fixed-point

Parameters

Operation: Blend

Opacity factor(s) source: Specify via dialog

Opacity factor(s): 0.75

Location source: Specify via dialog

Location [row column]: [1 2]

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.

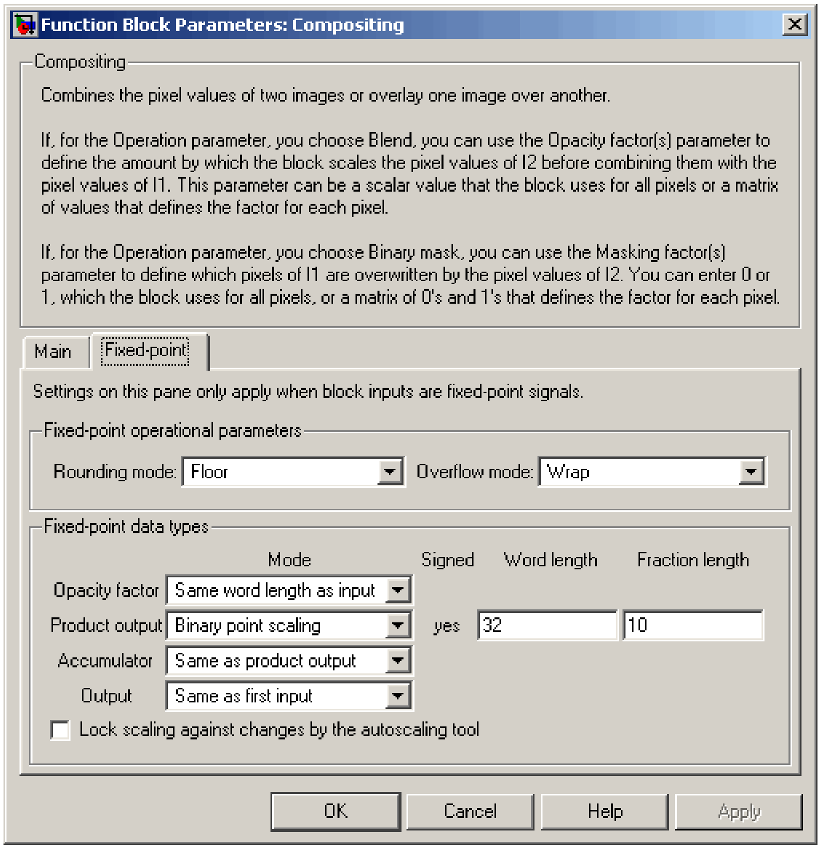
Mask source
: Indicate where to specify the masking factor(s). Your choices are `Specify via dialog` and `Input port`. This parameter is visible if, for the **Operation** parameter you choose `Binary mask`.

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`.



Function Block Parameters: Compositing

Compositing

Combines the pixel values of two images or overlay one image over another.

If, for the Operation parameter, you choose Blend, you can use the Opacity factor(s) parameter to define the amount by which the block scales the pixel values of I2 before combining them with the pixel values of I1. This parameter can be a scalar value that the block uses for all pixels or a matrix of values that defines the factor for each pixel.

If, for the Operation parameter, you choose Binary mask, you can use the Masking factor(s) parameter to define which pixels of I1 are overwritten by the pixel values of I2. You can enter 0 or 1, which the block uses for all pixels, or a matrix of 0's and 1's that defines the factor for each pixel.

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

	Mode	Signed	Word length	Fraction length
Opacity factor	Same word length as input			
Product output	Binary point scaling	yes	32	10
Accumulator	Same as product output			
Output	Same as first input			

☐ Lock scaling against changes by the autoscaling tool

Rounding mode
: Select the rounding mode for fixed-point operations.

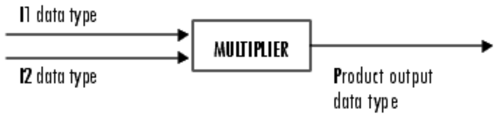
Overflow mode
: Select the overflow mode for fixed-point operations.

Opacity factor

Choose how to specify the word length and fraction length of the opacity factor:

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

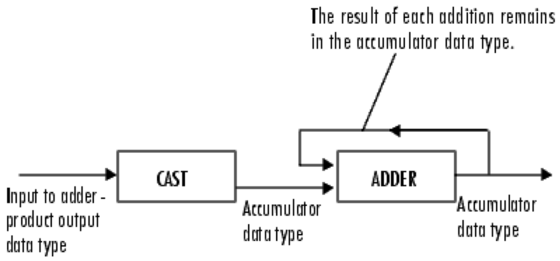
Product output



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

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

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type

As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

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

Output

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

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

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

Lock 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

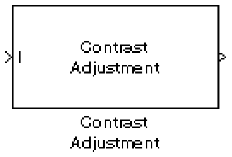
Insert Text	Video and Image Processing Blockset software

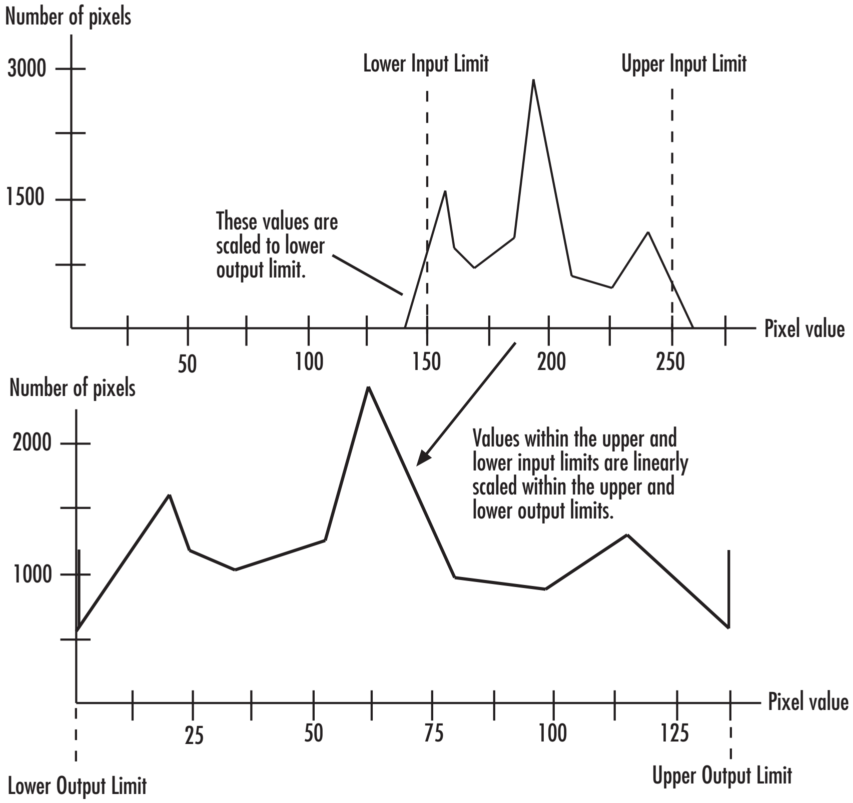
Contrast Adjustment

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.





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*]:

$$Output = \begin{cases} low\_out, \quad Input \le low\_in \\ low\_out + (Input - low\_in)\dfrac{high\_out - low\_out}{high\_in - low\_in}, \quad low\_in < Input < high\_in \\ high\_out, \quad Input \ge high\_in \end{cases}$$

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
Output	Scalar, vector, or matrix of intensity values or a scalar, vector, or matrix that represents one plane of the RGB video stream	Same as I port	No

Specifying upper and lower limits

Use the **Adjust pixel values from** and **Adjust pixel values to** parameters to specify the upper and lower input and output limits. All options are described below.

Input limits

Use the **Adjust pixel values from** parameter to specify the upper and lower input limits.

If you select `Full input data range [min max]`, uses the minimum input value as the lower input limit and the maximum input value as the upper input limit.

If you select `User-defined`, the **Range [low high]** parameter associated with this option appears. Enter a two-element vector of scalar values, where the first element corresponds to the lower input limit and the second element corresponds to the upper input limit.

If you select `Range determined by saturating outlier pixels`, the **Percentage of pixels to saturate [low high] (in %)**, **Specify number of histogram bins (used to calculate the range when outliers are eliminated)**, and **Number of histogram bins** parameters appear on the block. The block uses these parameter values to calculate the input limits in this three-step process:

1. Find the minimum and maximum input values, [*min_in max_in*].
2. Scale the pixel values from [**min_in max_in**] to [0 *num_bins*-1], where *num_bins* is the scalar value you specify in the **Number of histogram bins** parameter. This parameter always displays the value used by the block. Then the block calculates the histogram of the scaled input. For additional information about histograms, see the Histogram block reference page.
3. Find the lower input limit such that the percentage of pixels with values smaller than the lower limit is at most the value of the first element of the **Percentage of pixels to saturate [low high] (in %)** parameter. Similarly, find the upper input limit such that the percentage of pixels with values greater than the upper limit is at least the value of the second element of the parameter.

Output limits

Use the **Adjust pixel values to** parameter to specify the upper and lower output limits.

If you select `Full data type range`, the block uses the minimum value of the input data type as the lower output limit and the maximum value of the input data type as the upper out

If you select `User-defined range`, the **Range [low high]** parameter appears on the block. Enter a two-element vector of scalar values, where the first element corresponds to the lower output limit and the second element corresponds to the upper output limit.

For INF, -INF and NAN Input Values

If any input pixel value is either `INF` or `-INF`, the Contrast Adjustment block will change the pixel value according to how the parameters are set. The following table shows how the block handles these pixel values.

If Adjust pixel values from parameter is set to...	Contrast Adjustment block will:
Full data range [min,max]	Set the entire output image to the lower limit of the **Adjust pixel values to** parameter setting.
Range determined by saturating outlier pixels	
User defined range	Lower and higher limits of the **Adjust pixel values to** parameter set to `-INF` and `INF,` respectively.

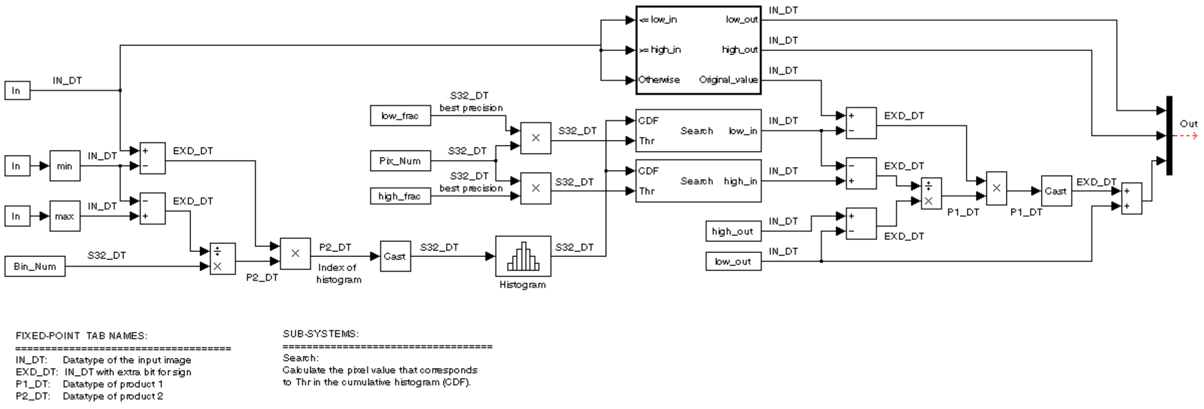
If any input pixel has a `NAN` value, the block maps the pixels with valid numerical values according to the user-specified method. It maps the `NAN` pixels to the lower limit of the **Adjust pixels values to** parameter.

Examples

See "Adjusting the Contrast in Intensity Images" in the *Video and Image Processing Blockset User's Guide*.

Fixed-Point Data Types

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

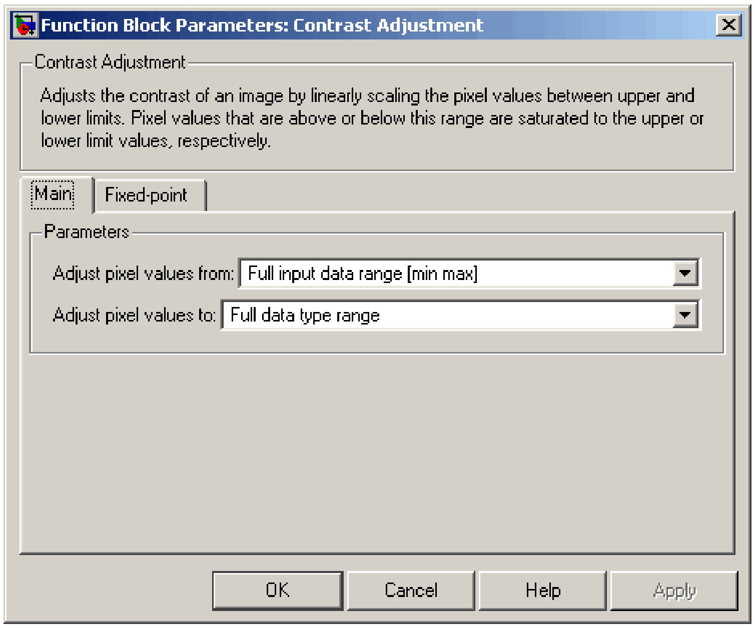


```
FIXED-POINT  TAB NAMES:
========================================
IN_DT:     Datatype of the input image
EXD_DT:  IN_DT with extra bit for sign
P1_DT:     Datatype of product 1
P2_DT:     Datatype of product 2
```

```
SUB-SYSTEMS:
========================================
Search:
Calculate the pixel value that corresponds
to Thr in the cumulative histogram (CDF).
```

Dialog Box

The Contrast Adjustment dialog box appears as shown in the following figure.



Contrast Adjustment

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 values, respectively.

Parameters

Adjust pixel values from: Full input data range [min max]

Adjust pixel values to: Full data type range

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`.

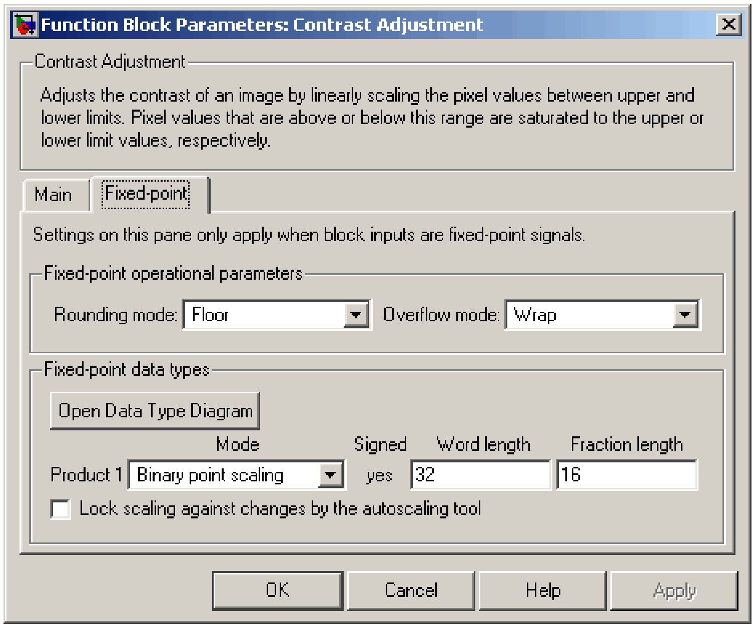
Specify number of histogram bins (used to calculate the range when outliers are eliminated)
Select this check box to change the number of histogram bins. This parameter is editable if, for the **Adjust pixel values from** parameter, you select `Range determined by saturating outlier pixels`.

Number of histogram bins
Enter the number of histogram bins to use to calculate the scaled input values. This parameter is available if you select the **Specify number of histogram bins (used to calculate the range when outliers are eliminated)** check box.

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

Range [low high]
Enter a two-element vector of scalar values. The first element corresponds to the lower output limit and the second element corresponds to the upper output limit. This parameter is 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.


Function Block Parameters: Contrast Adjustment

Contrast Adjustment

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 values, respectively.

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	16

☐ Lock scaling against changes by the autoscaling tool

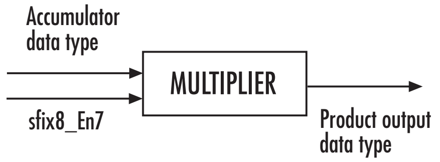
OK | Cancel | Help | Apply

Rounding mode
Select the rounding mode for fixed-point operations.

Overflow mode
Select the overflow mode for fixed-point operations.

Product 1

The product output type when the block calculates the ratio between the input data range and the number of histogram bins.



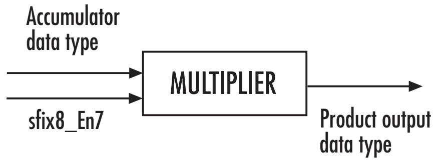
As shown in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.

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

Product 2

The product output type when the block calculates the bin location of each input value.



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

When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.

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

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

Lock 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

Histogram	Video and Image Processing Blockset software
Histogram Equalization	Video and Image Processing Blockset software

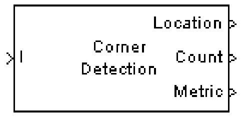
Purpose

Calculate corner metric matrix and find corners in images

Library

Analysis & Enhancement

Description



Corner Detection

The Corner Detection block finds corners in an image using the Harris corner detection, minimum eigenvalue, or local intensity comparison method. The block finds the corners in the image based on the pixels that have the largest corner metric values.

For the most accurate results, use the "Minimum Eigenvalue Method" on page 2-275. For the fastest computation, use the "Local Intensity Comparison" on page 2-276. For the trade-off between accuracy and computation, use the "Harris Corner Detection Method" on page 2-276.

Input/Output	Description
I	Matrix of intensity values
Location	2-by-N matrix that represents the locations of the corners where N is the maximum number of corners
Count	Scalar value that represents the number of detected corners
Metric	Matrix of corner metric values that is the same size as the input image

Minimum Eigenvalue Method

This method is more computationally expensive than the Harris corner detection algorithm because it directly calculates the eigenvalues of the sum of the squared difference matrix, *M*.

The sum of the squared difference matrix, *M*, is defined as follows:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

The previous equation is based on the following values:

$$A = (I_x)^2 \otimes w$$
$$B = (I_y)^2 \otimes w$$
$$C = (I_x I_y)^2 \otimes w$$

where I_x and I_y are the gradients of the input image, I, in the x and y direction, respectively. The $\otimes$ symbol denotes a convolution operation.

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

The block calculates the smaller eigenvalue of the sum of the squared difference matrix. This minimum eigenvalue corresponds to the corner metric matrix.

Harris Corner Detection Method

The Harris corner detection method avoids the explicit computation of the eigenvalues of the sum of squared differences matrix by solving for the following corner metric matrix, R:

$$R = AB - C^2 - k(A + B)^2$$

A, B, C are defined in the previous section, "Minimum Eigenvalue Method" on page 2-275.

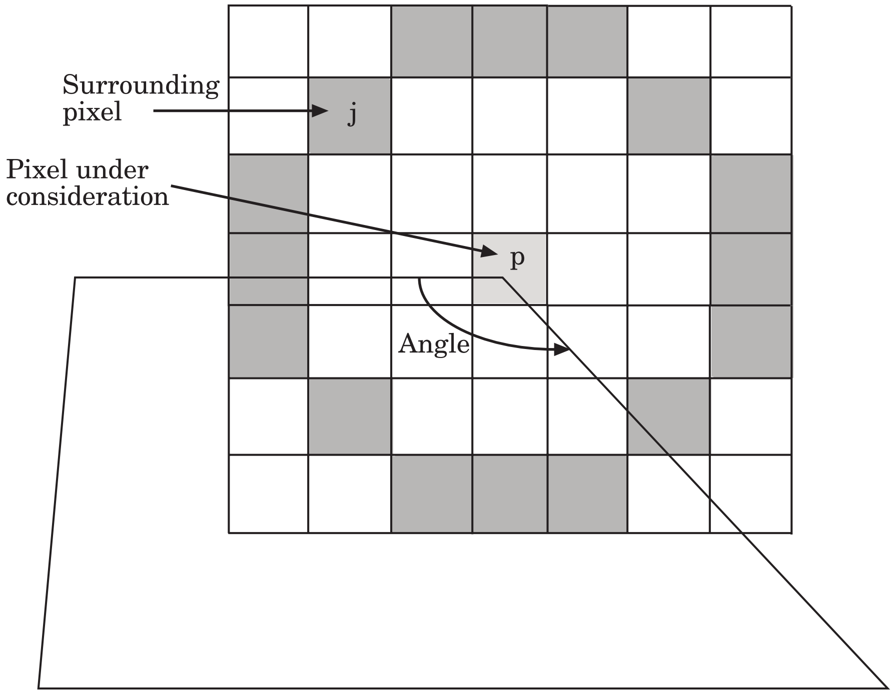
The variable k corresponds to the sensitivity factor. You can specify its value using the **Sensitivity factor (0<k<0.25)** parameter. The smaller the value of k, the more likely it is that the algorithm can detect sharp corners.

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

Local Intensity Comparison

This method determines that a pixel is a possible corner if it has either, N contiguous valid bright surrounding pixels, or N contiguous dark surrounding pixels. Specifying the value of *N* is discussed later in this section. The next section explains how the block finds these surrounding pixels.

Suppose that *p* is the pixel under consideration and *j* is one of the pixels surrounding *p*. The locations of the other surrounding pixels are denoted by the shaded areas in the following figure.



I_p and I_j are the intensities of pixels *p* and *j*, respectively. Pixel *j* is a valid bright surrounding pixel if $I_j - I_p \geq T$. Similarly, pixel *j* is a valid dark surrounding pixel if $I_p - I_j \geq T$. In these equations, *T* is the value you specified for the **Intensity comparison threshold** parameter.

The block repeats this process to determine whether the block has *N* contiguous valid surrounding pixels. The value of *N* is related to the value you specify for the **Maximum angle to be considered a corner (in degrees)**, as shown in the following table.

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:

$$R = \max\left(\sum_{j:I_j \geq I_p + T} \left| I_p - I_j \right| - T, \sum_{j:I_j \leq I_p - T} \left| I_p - I_j \right| - T, \right)$$

Block Output

Use the **Output** parameter to determine whether the block outputs the corner location, corner location and metric matrix, or the metric matrix. The block outputs the corner locations in a 2-by-N matrix where each row stores the row and column locations of the corners and N is the maximum number of corners. The block outputs the corner metric values in a matrix that is the same size as the input image.

If you set the **Output** parameter to `Corner location` or `Corner location and metric matrix`, the **Maximum number of corners**, **Minimum metric value that indicates a corner**, and

Neighborhood size (suppress region around detected corners) parameters appear on the block.

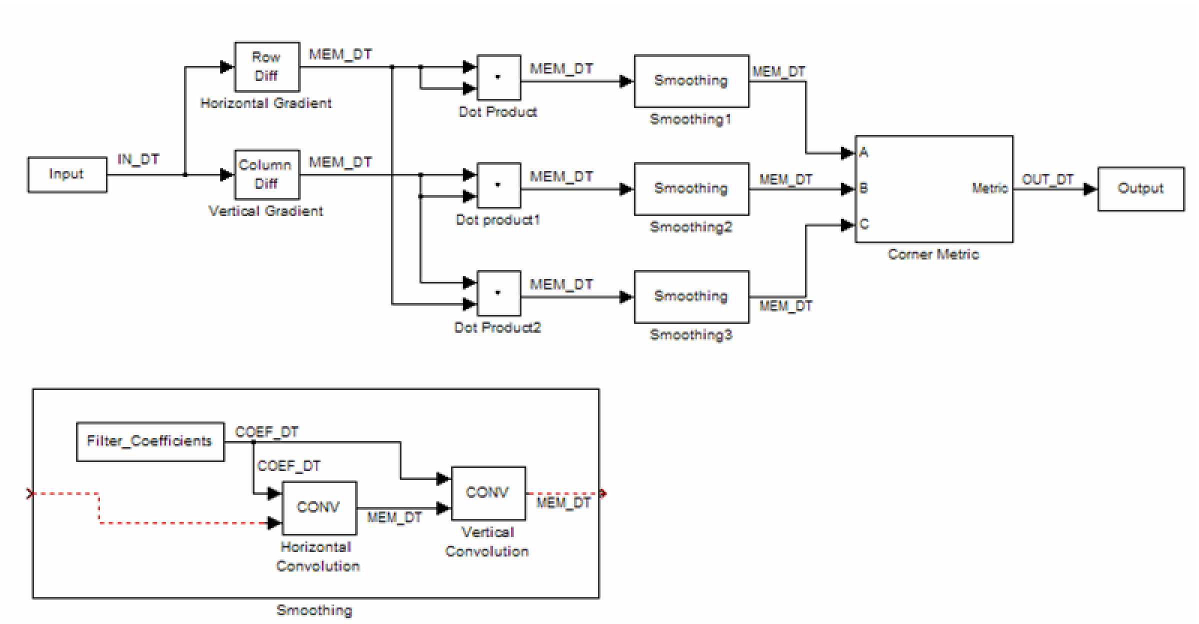
To determine the final corner values, the block follows this process:

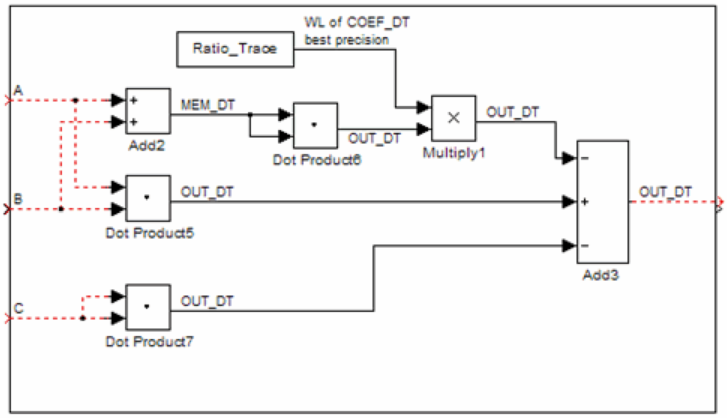
1. Find the pixel with the largest corner metric value.
2. Verify that the metric value is greater than or equal to the value you specified for the **Minimum metric value that indicates a corner** parameter.
3. Suppress the region around the corner value by the size defined in the **Neighborhood size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

The block repeats this process until it finds all the corners in the image or it finds the number of corners you specified in the **Maximum number of corners** parameter.

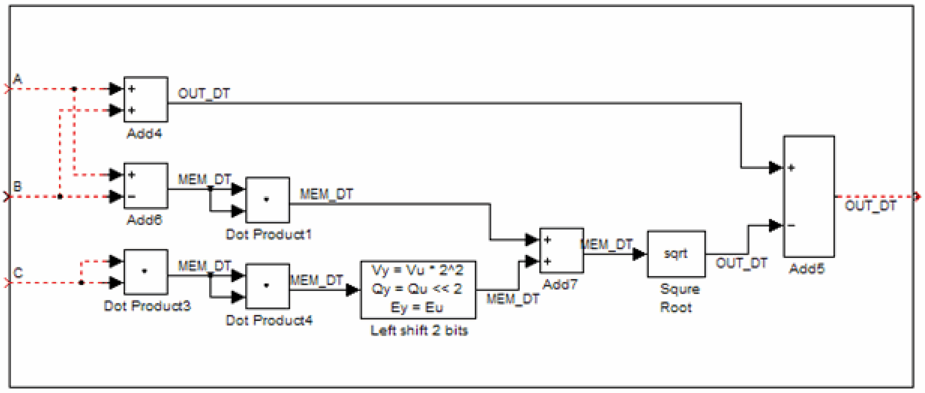
Fixed-Point Data Types

The following diagram shows the data types used in the Corner Detection block for fixed-point signals. These diagrams apply to the Harris corner detection and minimum eigenvalue methods only.


Input
IN_DT
Row Diff
Horizontal Gradient
Column Diff
Vertical Gradient
MEM_DT
Dot Product
Dot product1
Dot Product2
Smoothing
Smoothing1
Smoothing2
Smoothing3
A
B
C
Metric
Corner Metric
OUT_DT
Output
Filter_Coefficients
COEF_DT
CONV
Horizontal Convolution
Vertical Convolution
Smoothing



Corner Metric by Harris Algorithm



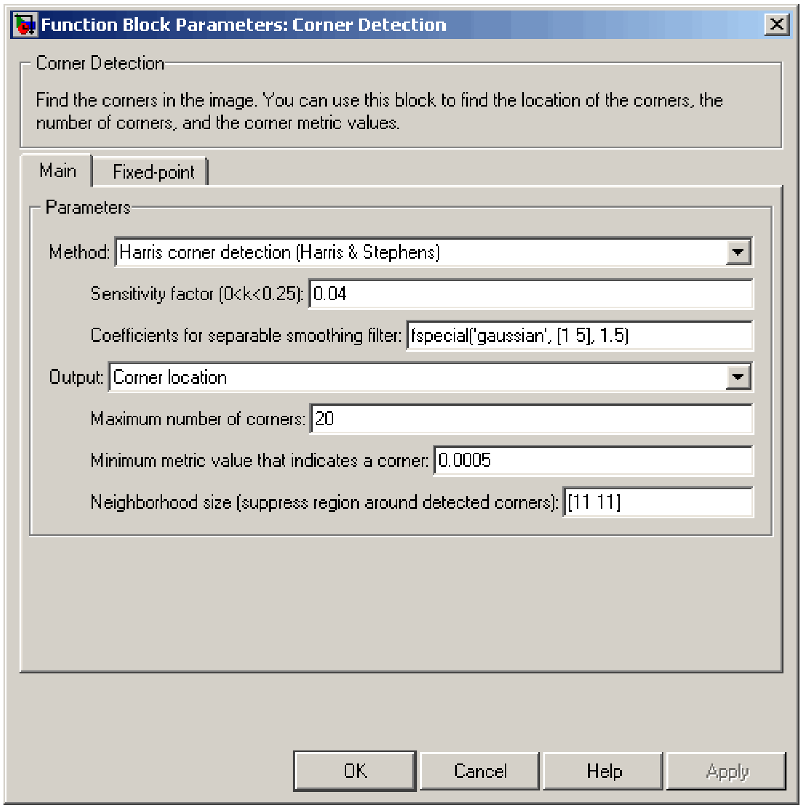
Corner Metric by Minimum Eigenvalue Algorithm

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

Variable Name	Definition
IN_DT	Input data type
MEM_DT	Memory data type
OUT_DT	Metric output data type
COEF_DT	Coefficients data type

Dialog Box

The Corner Detection dialog box appears as shown in the following figure.



Function Block Parameters: Corner Detection

Corner Detection

Find the corners in the image. You can use this block to find the location of the corners, the number of corners, and the corner metric values.

Method: Harris corner detection (Harris & Stephens)

Sensitivity factor (0<k<0.25): 0.04

Coefficients for separable smoothing filter: fspecial('gaussian', [1 5], 1.5)

Output: Corner location

Maximum number of corners: 20

Minimum metric value that indicates a corner: 0.0005

Neighborhood size (suppress region around detected corners): [11 11]

Method

Specify the method to use to find the corner values. Your choices are `Harris corner detection (Harris & Stephens)`, `Minimum eigenvalue (Shi & Tomasi)`, and `Local intensity comparison (Rosen & Drummond)`.

Sensitivity factor (0<k<0.25)
: Specify the sensitivity factor, *k*. The smaller the value of *k* the more likely the algorithm is to detect sharp corners. This parameter is visible if you set the **Method** parameter to `Harris corner detection (Harris & Stephens)`. This parameter is tunable.

Coefficients for separable smoothing filter
: Specify a vector of filter coefficients for the smoothing filter. This parameter is visible if you set the **Method** parameter to `Harris corner detection (Harris & Stephens)` or `Minimum eigenvalue (Shi & Tomasi)`.

Intensity comparison threshold
: Specify the threshold value used to find valid surrounding pixels. This parameter is visible if you set the **Method** parameter to `Local intensity comparison (Rosen & Drummond)`. This parameter is tunable.

Maximum angle to be considered a corner (in degrees)
: Specify the maximum corner angle. This parameter is visible if you set the **Method** parameter to `Local intensity comparison (Rosen & Drummond)`. This parameter is tunable for Simulation only.

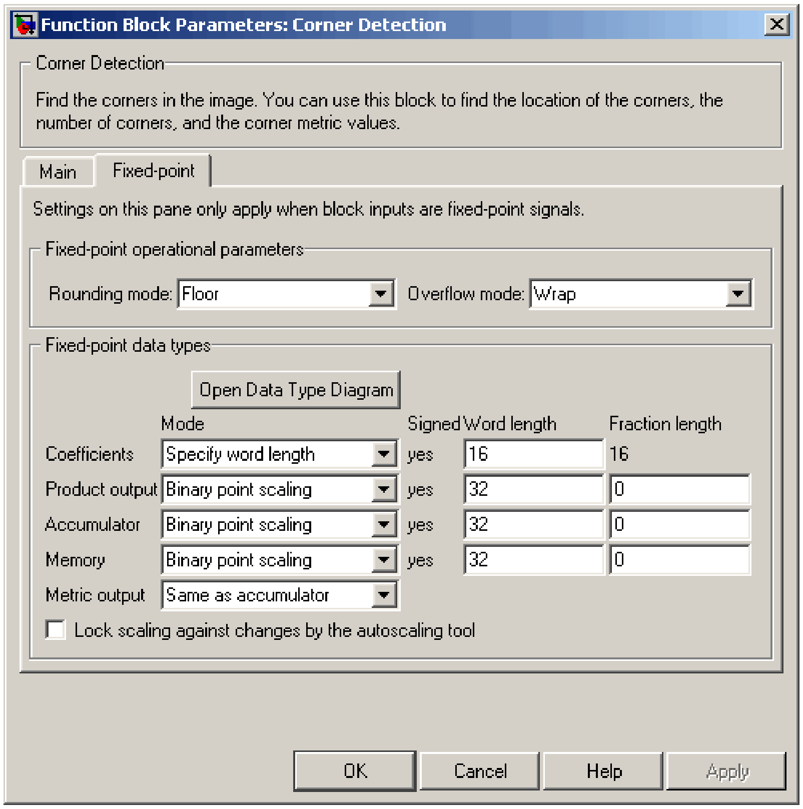
Output
: Specify the block output. Your choices are `Corner location`, `Corner location and metric matrix`, and `Metric matrix`.

Maximum number of corners
: Enter the maximum number of corners you want the block to find. This parameter is visible if you set the **Output** parameter to `Corner location` or `Corner location and metric matrix`.

Minimum metric value that indicates a corner
: Specify the minimum corner metric value. This parameter is visible if you set the **Output** parameter to `Corner location` or `Corner location and metric matrix`. This parameter is tunable.

Neighborhood size (suppress region around detected corners)
Specify the size of the neighborhood around the corner metric value over which the block zeros out the values. Enter a two-element vector of positive odd integers, [r c]. Here, r is the number of rows in the neighborhood and c is the number of columns. This parameter is visible if you set the **Output** parameter to `Corner location` or `Corner location and metric matrix`.

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



Function Block Parameters: Corner Detection

Corner Detection

Find the corners in the image. You can use this block to find the location of the corners, the number of corners, and the corner metric values.

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

Fixed-point operational parameters

Rounding mode: Floor   Overflow mode: Wrap

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Coefficients	Specify word length	yes	16	16
Product output	Binary point scaling	yes	32	0
Accumulator	Binary point scaling	yes	32	0
Memory	Binary point scaling	yes	32	0
Metric output	Same as accumulator			

☐ Lock scaling against changes by the autoscaling tool

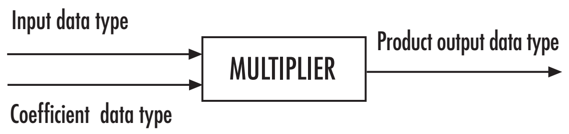
Rounding mode
Select the rounding mode for fixed-point operations.

Overflow mode
Select the overflow mode for fixed-point operations.

Coefficients
Choose how to specify the word length and the fraction length of the coefficients:

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

Product output
As shown in the following figure, the output of the multiplier is placed into the product output data type and scaling.

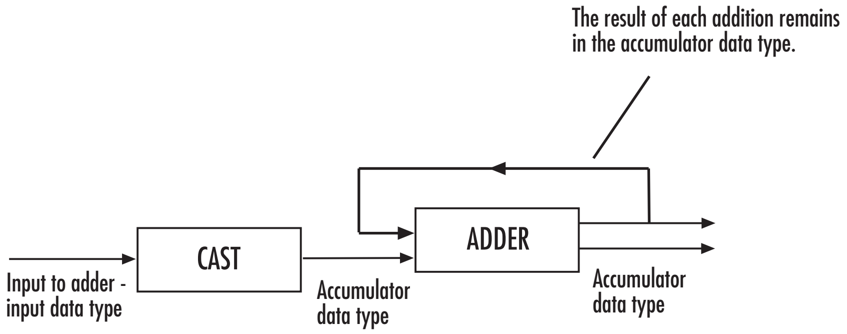


Use this parameter to specify how to designate the product output word and fraction lengths.

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

Accumulator

As shown in the following figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it.


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - input data type
Accumulator data type
Accumulator data type

Use this parameter to specify how to designate this accumulator word and fraction lengths:

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

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

Memory
Choose how to specify the memory word length and fraction length:

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

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

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

Lock 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.

Supported Data Types

Port	Supported Data Types
I	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer
Location	32-bit unsigned integer
Count	32-bit unsigned integer
Metric	Same as I port

See Also

Find Local Maxima	Video and Image Processing Blockset software
Estimate Geometric Transformation	Video and Image Processing Blockset software

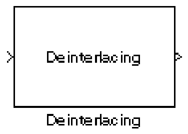
Purpose

Remove motion artifacts by deinterlacing input video signal

Library

Analysis & Enhancement

Description



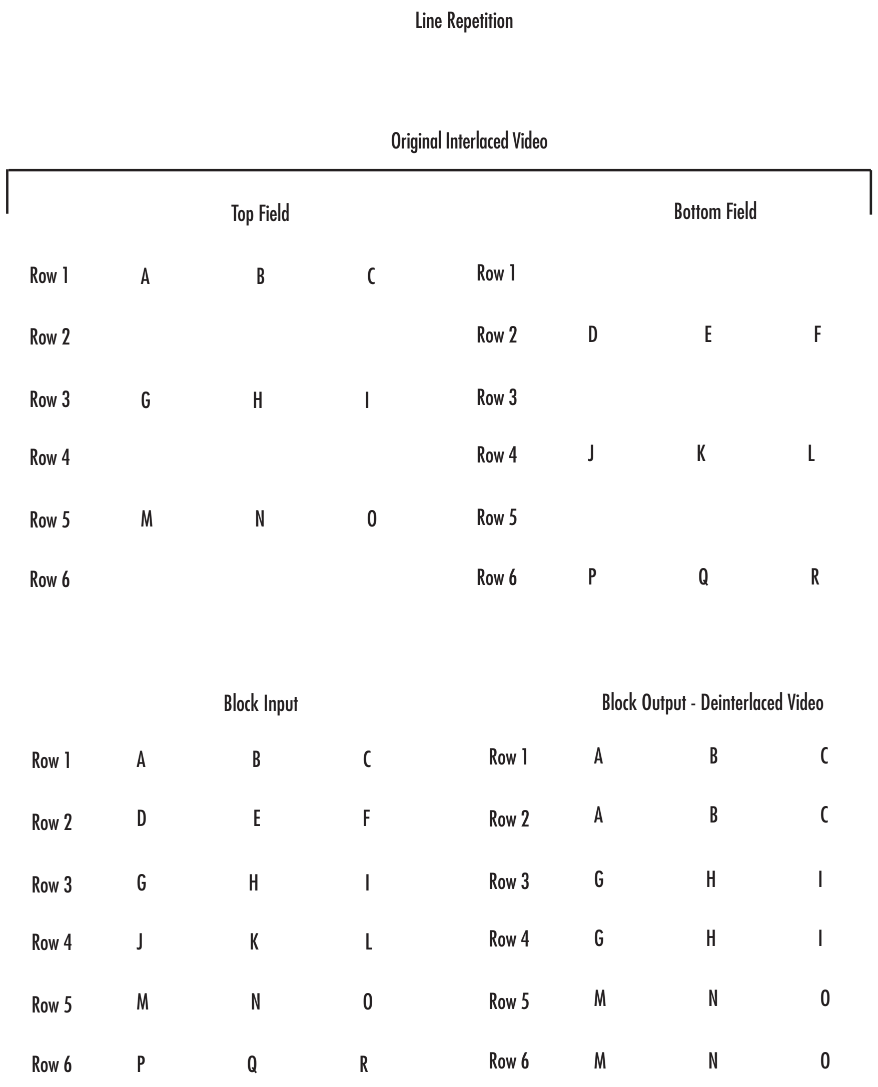
The Deinterlacing block takes the input signal, which is the combination of the top and bottom fields of the interlaced video, and converts it into deinterlaced video using line repetition, linear interpolation, or vertical temporal median filtering.

Note This block supports intensity and color images on its ports.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Combination of top and bottom fields of interlaced video	• 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`.



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

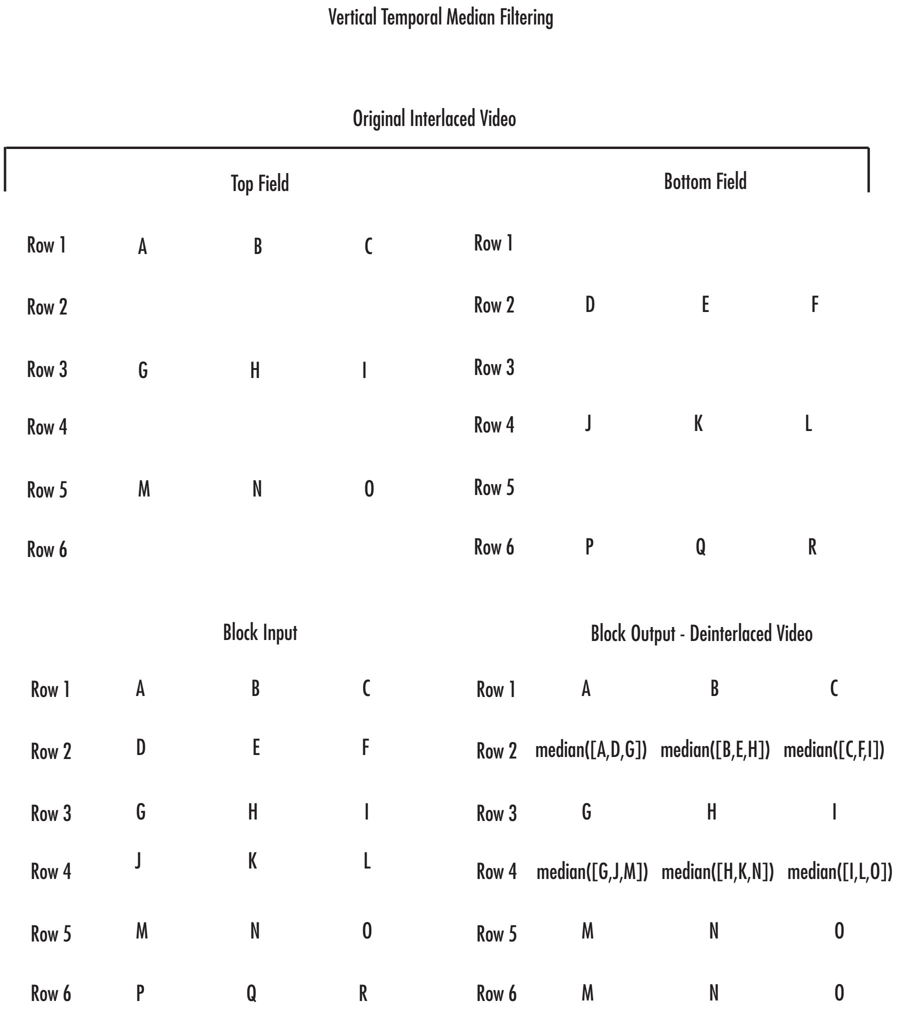
Linear Interpolation

Original Interlaced Video

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

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

The following figure illustrates the block's behavior if you select `Vertical temporal median filtering`.



Row-Major Data Format

The MATLAB enviroment and the Video and Image Processing Blockset software use column-major data organization. However, the Deinterlacing block gives you the option to process data that is stored in row-major format. When you select the **Input image is transposed (data order is row major)** check box, the block assumes that the input buffer contains contiguous data elements from the first row first, then data elements from the second row second, and so on through the last row. Use this functionality only when you meet all the following criteria:

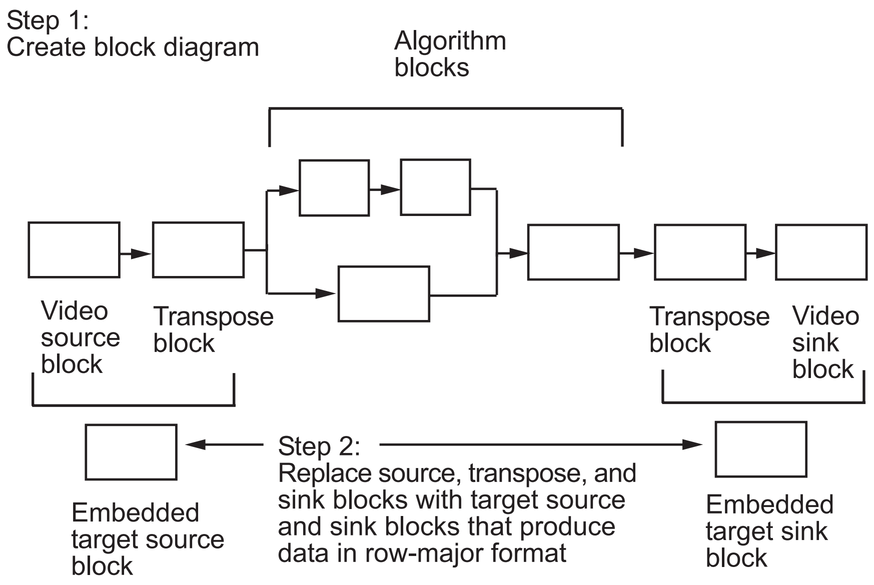
- You are developing algorithms to run on an embedded target that uses the row-major format.
- You want to limit the additional processing required to take the transpose of signals at the interfaces of the row-major and column-major systems.

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

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

  For example, if you use the 2-D FIR Filter block, you need to verify that your filter coefficients are transposed. If you are using the Rotate block, you need to use negative rotation angles, etc.
- Only three blocks have the **Input image is transposed (data order is row major)** check box. They are the Chroma Resampling, Deinterlacing, and Insert Text blocks. You need to select this check box to enable row-major functionality in these blocks. All other blocks must be properly configured to process data in row-major format.

Use the following two-step workflow to develop algorithms in row-major format to run on an embedded target.


Step 1:
Create block diagram
Algorithm blocks
Video source block
Transpose block
Transpose block
Video sink block
Step 2:
Replace source, transpose, and sink blocks with target source and sink blocks that produce data in row-major format
Embedded target source block
Embedded target sink block

See the DM642 EVM Video ADC and DM642 EVM Video DAC reference pages in the *Target Support Package TC6 User's Guide* for more information about data order in embedded targets.

Example

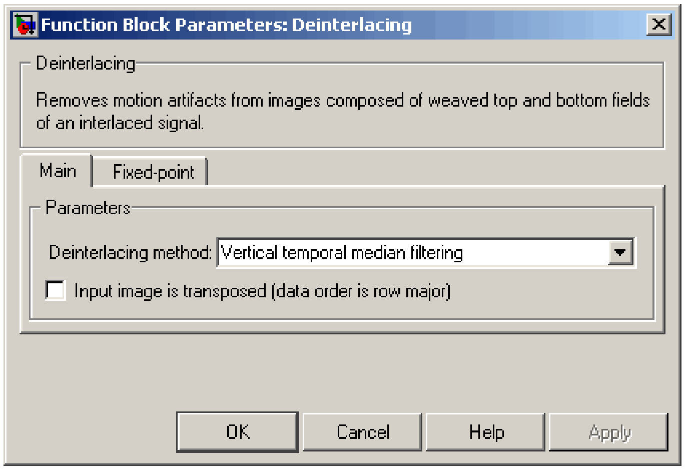
The following example shows you how to use the Deinterlacing block to remove motion artifacts from an image.

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`.



Function Block Parameters: Deinterlacing

Deinterlacing

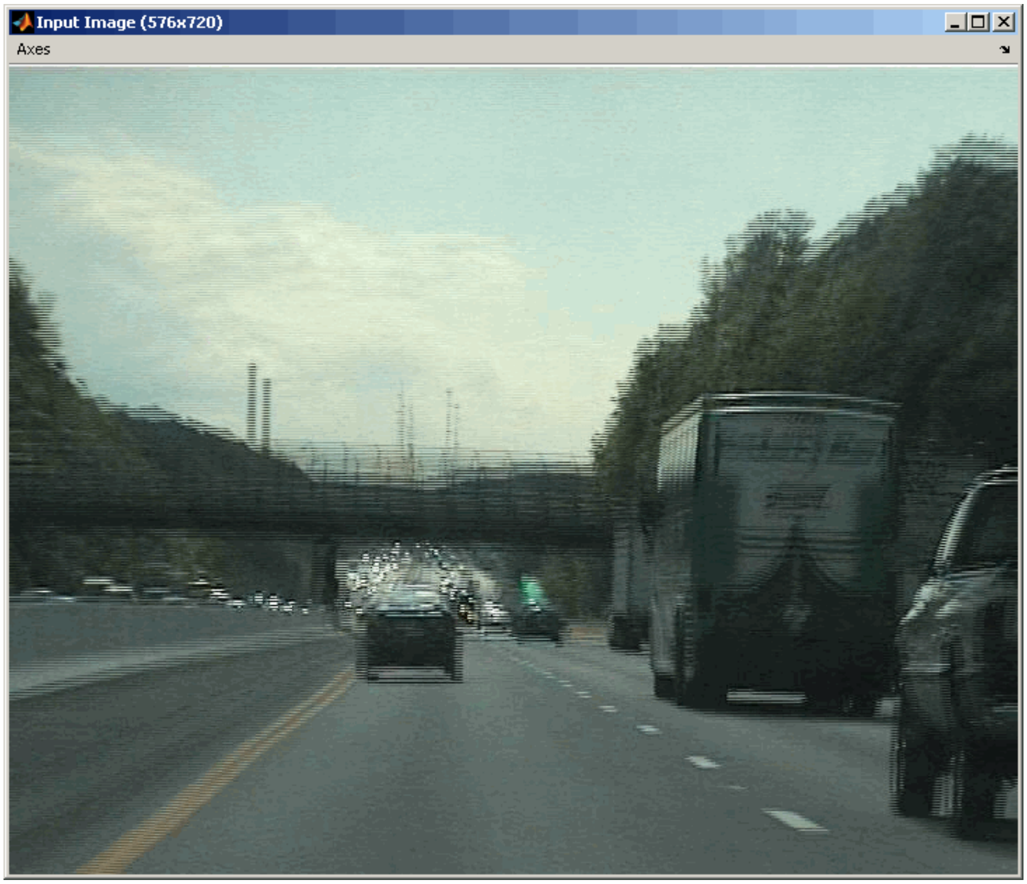
Removes motion artifacts from images composed of weaved top and bottom fields of an interlaced signal.

Deinterlacing method: Vertical temporal median filtering

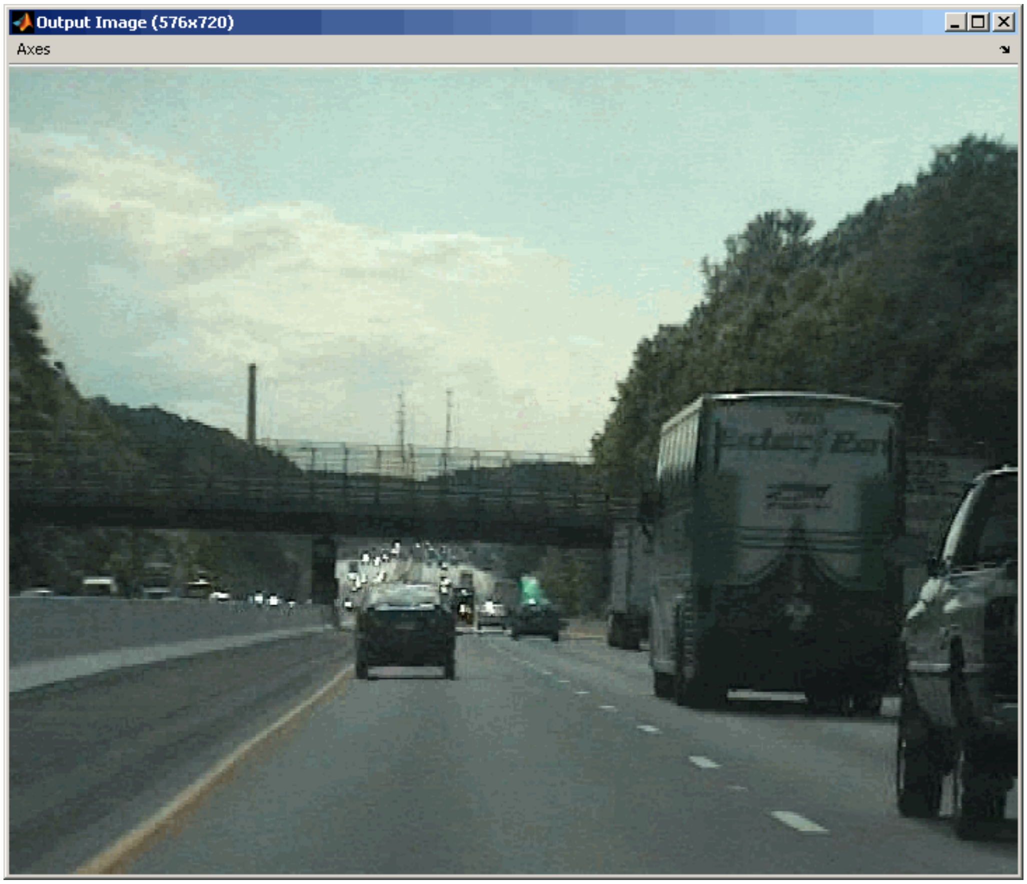
☐ Input image is transposed (data order is row major)

3 Run the model.

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

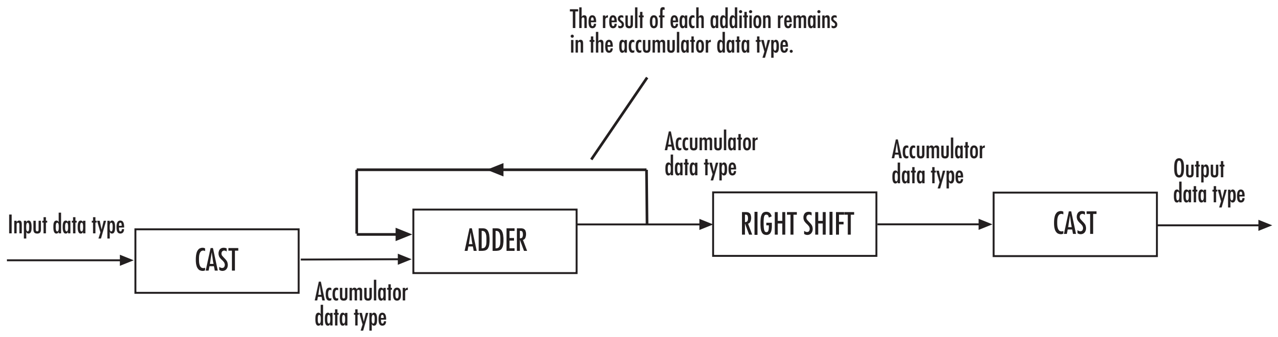


The clearer output image appears in the Output Image window.



Fixed-Point Data Types

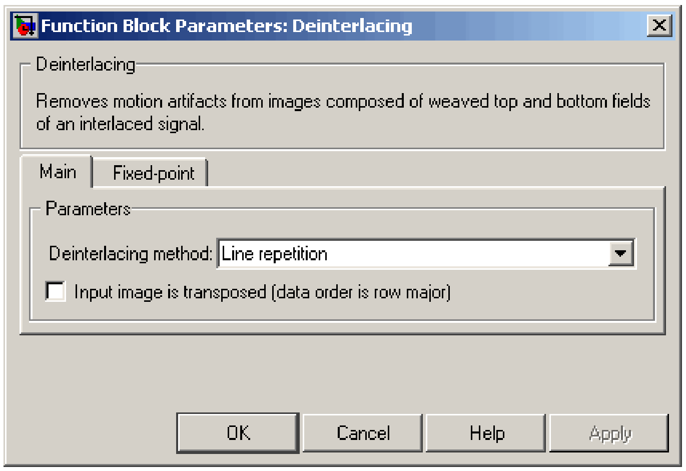
The following diagram shows the data types used in the Deinterlacing block for fixed-point signals.



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.


Function Block Parameters: Deinterlacing

Deinterlacing

Removes motion artifacts from images composed of weaved top and bottom fields of an interlaced signal.

Parameters

Deinterlacing method: Line repetition

☐ Input image is transposed (data order is row major)

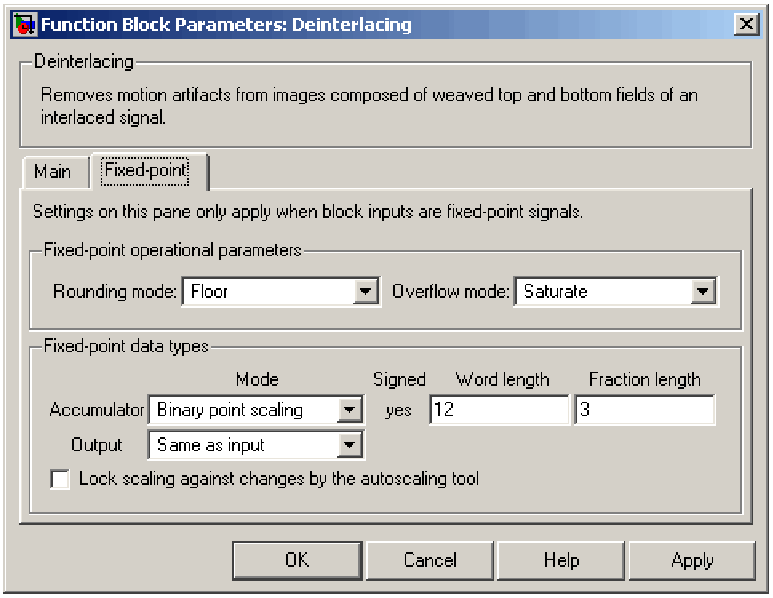
Deinterlacing method

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

Input image is transposed (data order is row major)

When you select this check box, the block assumes that the input buffer contains data elements from the first row first, then data elements from the second row second, and so on through the last row.

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



Function Block Parameters: Deinterlacing

Deinterlacing

Removes motion artifacts from images composed of weaved top and bottom fields of an interlaced signal.

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

Fixed-point operational parameters

Rounding mode: Floor   Overflow mode: Saturate

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Accumulator	Binary point scaling	yes	12	3
Output	Same as input			

☐ Lock scaling against changes by the autoscaling tool

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

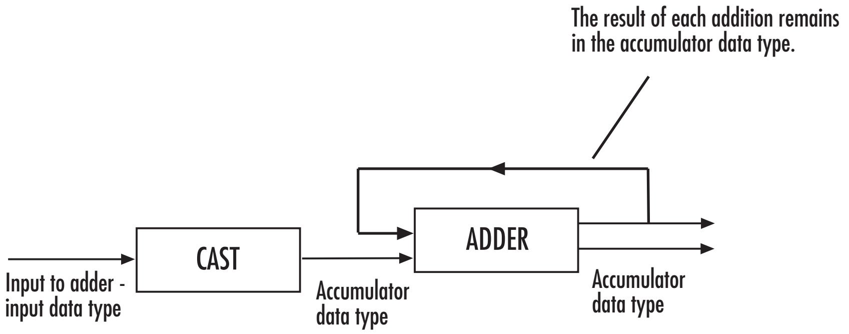
Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - input data type
Accumulator data type
Accumulator data type

As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths:

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

Output

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

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

- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- 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.

Demosaic

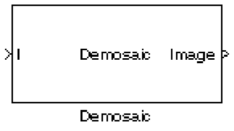
Purpose

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.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Matrix of intensity values • If, for the **Interpolation algorithm** parameter, you select `Bilinear`, the number of rows and	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integer	No

Port	Input/Output	Supported Data Types	Complex Values Supported
	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.	• 8-, 16-, and 32-bit unsigned integer	
R, G, B	Matrix that represents one plane of the input RGB video stream. Outputs from the R, G, or B ports have the same data type.	Same as I port	No
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.	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].

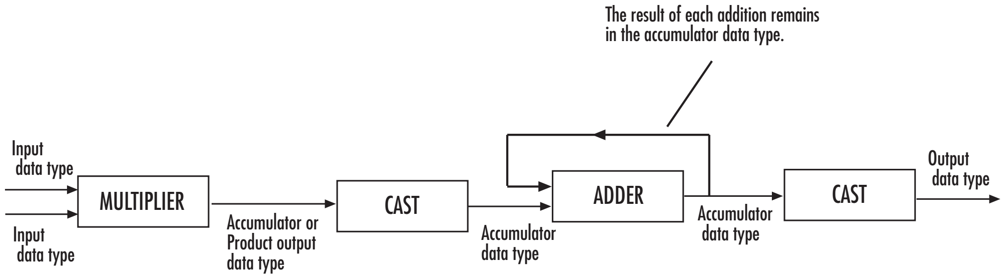
Use the **Sensor alignment** parameter to specify the alignment of the input image. Select the sequence of R, G and B pixels that correspond to the 2-by-2 block of pixels in the top-left corner of the image. You specify the sequence in left-to-right, top-to-bottom order. For example, for the image at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

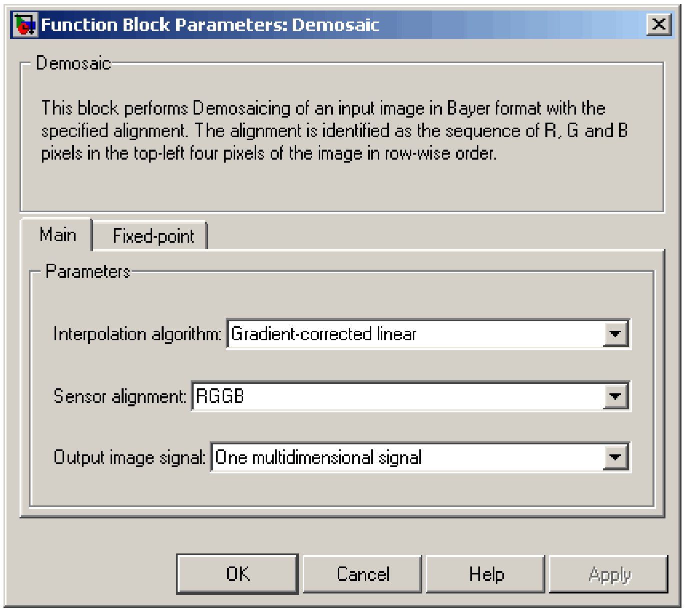
The following diagram shows the data types used in the Demosaic block for fixed-point signals.



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

Dialog Box

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



Demosaic

This block performs Demosaicing of an input image in Bayer format with the specified alignment. The alignment is identified as the sequence of R, G and B pixels in the top-left four pixels of the image in row-wise order.

Interpolation algorithm: Gradient-corrected linear

Sensor alignment: RGGB

Output image signal: One multidimensional signal

Interpolation algorithm

Specify the algorithm the block uses to calculate the missing color information. Your choices are `Bilinear` or `Gradient-corrected linear`.

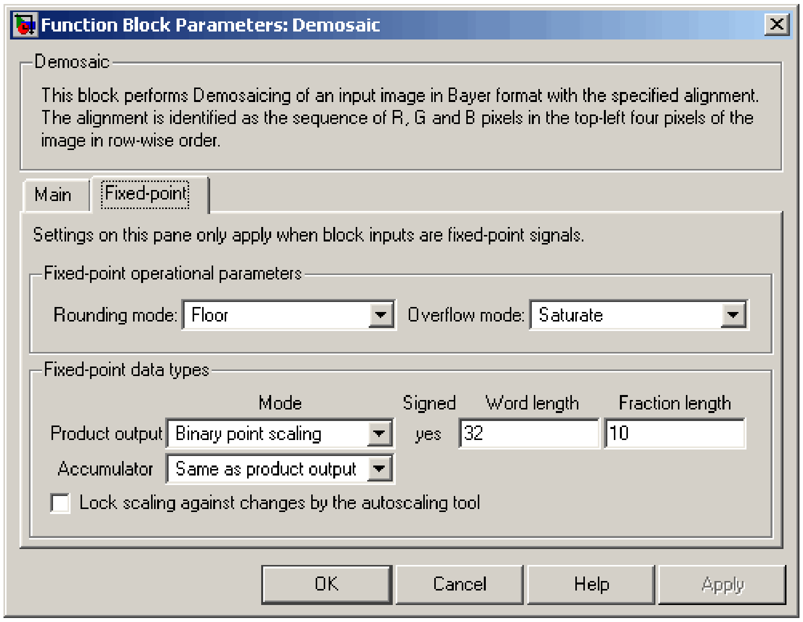
Sensor alignment

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.



Function Block Parameters: Demosaic

Demosaic

This block performs Demosaicing of an input image in Bayer format with the specified alignment. The alignment is identified as the sequence of R, G and B pixels in the top-left four pixels of the image in row-wise order.

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: Saturate

Fixed-point data types

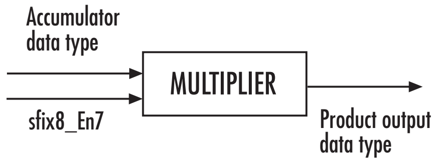
	Mode	Signed	Word length	Fraction length
Product output	Binary point scaling	yes	32	10
Accumulator	Same as product output			

☐ Lock scaling against changes by the autoscaling tool

Rounding mode
: Select the rounding mode for fixed-point operations.

Overflow mode
: Select the overflow mode for fixed-point operations.

Product output



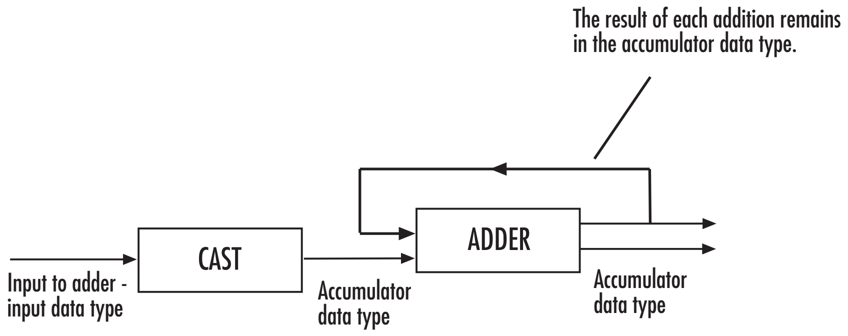
As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

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

When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.

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

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - input data type
Accumulator data type
Accumulator data type

As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths:

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

Lock 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.

Purpose

Find local maxima in binary or intensity images

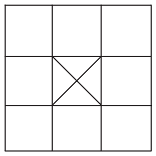
Library

Morphological Operations

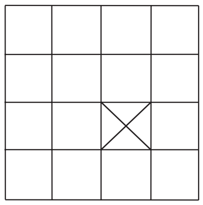
Description

I
Dilate
Nhood
Dilation

The Dilation block rotates the neighborhood or structuring element 180 degrees. Then it slides the neighborhood or structuring element over an image, finds the local maxima, and creates the output matrix from these maximum values. If the neighborhood or structuring element has a center element, the block places the maxima there, as illustrated in the following figure.



If the neighborhood or structuring element does not have an exact center, the block has a bias toward the lower-right corner, as a result of the rotation. The block places the maxima there, as illustrated in the following figure.



This block uses flat structuring elements only.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
Nhood	Matrix or vector of ones and zeros that represents the neighborhood values	Boolean	No
Output	Vector or matrix of intensity values that represents the dilated image	Same as I port	No

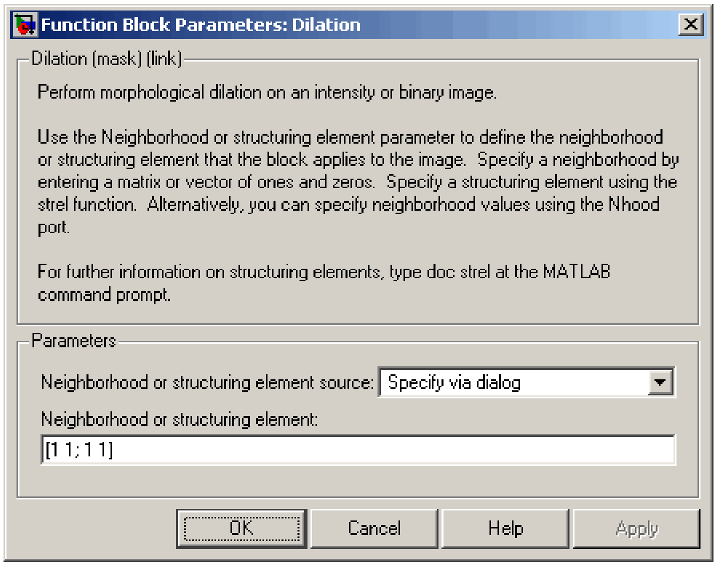
The output signal has the same data type as the input to the I port.

Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select `Specify via dialog`, the **Neighborhood or structuring element** parameter appears in the dialog box. If you select `Input port`, the Nhood port appears on the block. Use this port to enter your neighborhood values as a matrix or vector of 1s and 0s. You can only specify a structuring element using the dialog box.

Use the **Neighborhood or structuring element** parameter to define the neighborhood or structuring element that the block applies to the image. Specify a neighborhood by entering a matrix or vector of 1s and 0s. Specify a structuring element with the `strel` function from the Image Processing Toolbox. If the structuring element is decomposable into smaller elements, the block executes at higher speeds due to the use of a more efficient algorithm. If you enter an array of STREL objects, the block applies each object to the entire matrix in turn.

Dialog Box

The Dilation dialog box appears as shown in the following figure.


Function Block Parameters: Dilation

Dilation (mask) (link)

Perform morphological dilation on an intensity or binary image.

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 ones and zeros. Specify a structuring element using the strel function. Alternatively, you can specify neighborhood values using the Nhood port.

For further information on structuring elements, type doc strel at the MATLAB command prompt.

Parameters

Neighborhood or structuring element source: Specify via dialog

Neighborhood or structuring element:

[1 1; 1 1]

Neighborhood or structuring element source

Specify how to enter your neighborhood or structuring element values. Select `Specify via dialog` to enter the values in the dialog box. Select `Input port` to use the Nhood port to specify the neighborhood values. You can only specify a structuring element using the dialog box.

Neighborhood or structuring element

If you are specifying a neighborhood, this parameter must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the `strel` function from the Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or structuring element source** parameter, you select `Specify via dialog`.

References

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

See Also

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

Draw Markers

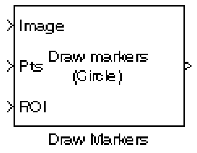
Purpose

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. Overwriting the pixel values embeds the shapes.

This block uses Bresenham's circle drawing algorithm to draw circles and Bresenham's line drawing algorithm to draw all other markers.

Port	Input/Output	Supported Data Types	Complex Values Supported
Image	*M-by-N* matrix of intensity values or an *M-by-N-by-P* color values where *P* is the number of color planes	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
R, G, B	Scalar, vector, or matrix that represents one plane of the input RGB video stream. Inputs to the R, G, and B ports must have the same dimensions and data type.	Same as Image port	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Pts	*2-by-N* matrix of row and column pairs, $\begin{bmatrix} r_1 & r_2 & \cdots & r_N \\ c_1 & c_2 & \cdots & c_N \end{bmatrix}$ where *N* is the total number of markers and each row and column pair defines the center of a marker.	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer If the input to the Image port is an integer, fixed point, or boolean data type, the input to the Pts port must also be an integer data type.	No
ROI	Four-element vector of integers that define a rectangular area in which to draw the markers. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the area. The second two elements represent the height and width of the area.	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
Clr	*P*-element vector or *P-by-N* matrix where *P* is the number of color planes	Same as Image port	No
Output	Scalar, vector, or matrix of pixel values that contain the marker(s)	Same as Image port	No

The output signal is the same size and data type as the inputs to the Image, R, G, and B ports.

Use the **Marker shape** parameter to specify one of the following types of markers:

- `Circle`
- `X-mark`
- `Plus`
- `Star`
- `Square`

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

If, for the **Marker shape** parameter, you select:

- `Circle`, `X-mark`, or `Star`

and you then select the,

- **Use antialiasing** check box

the block performs a smoothing algorithm. The Draw Markers block uses an algorithm similar to the poly2mask function to determine which subpixels to draw.

Use the **Draw markers in** parameter to define one of the following types of areas in which to draw the markers.

- `Entire image`, enables you to draw markers in the entire image.
- `Specify region of interest via port`, the ROI port appears on the block. Enter a four-element vector of integer values, `[r c height width]`, where `r` and `c` are the row and column coordinates of the upper-left corner of the area, and `height` and `width` represent

the height (in rows) and width (in columns) of the area. If you specify values that are outside the image, the block clips the values to the image boundaries.

Use the **Image signal** parameter to specify one of the following ways to input and output a color video signal.

- `One multidimensional signal`, the block accepts an *M-by-N-by-P* color video signal, where *P* is the number of color planes, at one port.
- `Separate color signals`, additional ports appear on the block. Each port accepts one *M-by-N* plane of an RGB video stream.

Selecting Marker Fill and Border Colors

You can set the marker fill or border color via the input port or via the input dialog. Use the color input or color parameter to determine the appearance of the rectangle(s), line(s), polygon(s), or circle(s).

- "Fill Color" on page 2-316
- "Border Color" on page 2-317
- "Color Values" on page 2-317
- "Opacity Factor" on page 2-317

Fill Color

If you select the **Filled** check box, the **Fill color source**, **Fill color** and **Opacity factor (between 0 and 1)** parameters appear in the dialog box. Use the **Fill color source** parameter to specify either `Input port` or `Specify via dialog` for the color source. If `Specify via dialog` is selected, you can specify either `Black`, `White`, or `User-specified value` for the **Fill color** parameter for the shading inside the shape. The **Color value(s)** parameter is applicable when the `User-specified value` is selected. Use the **Opacity factor (between 0 and 1)** parameter to specify the opacity of the shading inside the shape, where 0 is transparent and 1 is opaque.

Border Color

If the **Filled** check box is not selected, the **Border color source**, and **Border color** parameters are available. Use the **Border color source** parameter to specify either `Input port` or `Specify via dialog` for the color source. If `Specify via dialog` is selected, you can specify either `Black`, `White`, or `User-specified value` for the **Border color** parameter. If the color is user specified, the **Color value(s)** parameter is used to enter the color.

Color Values

The following table describes what to enter for the **Color Value(s)** parameter based on the block input and the number of markers you are drawing. This parameter is applicable when `User-specified value` is selected for the border color source.

Block Input	Color Value(s) for Drawing One Marker or Multiple Markers with the Same Color	Color Value(s) for Drawing Multiple Markers with Unique Color
Intensity image	Scalar intensity value	*R*-element vector where *R* is the number of markers
Color image	P-element vector where P is the number of color planes	*P-by-R* matrix where *P* is the number of color planes and *R* is the number of markers

For each value in the parameter, enter a number between the minimum and maximum values that can be represented by the data type of the input image. If you enter a value outside this range, the block produces an error message.

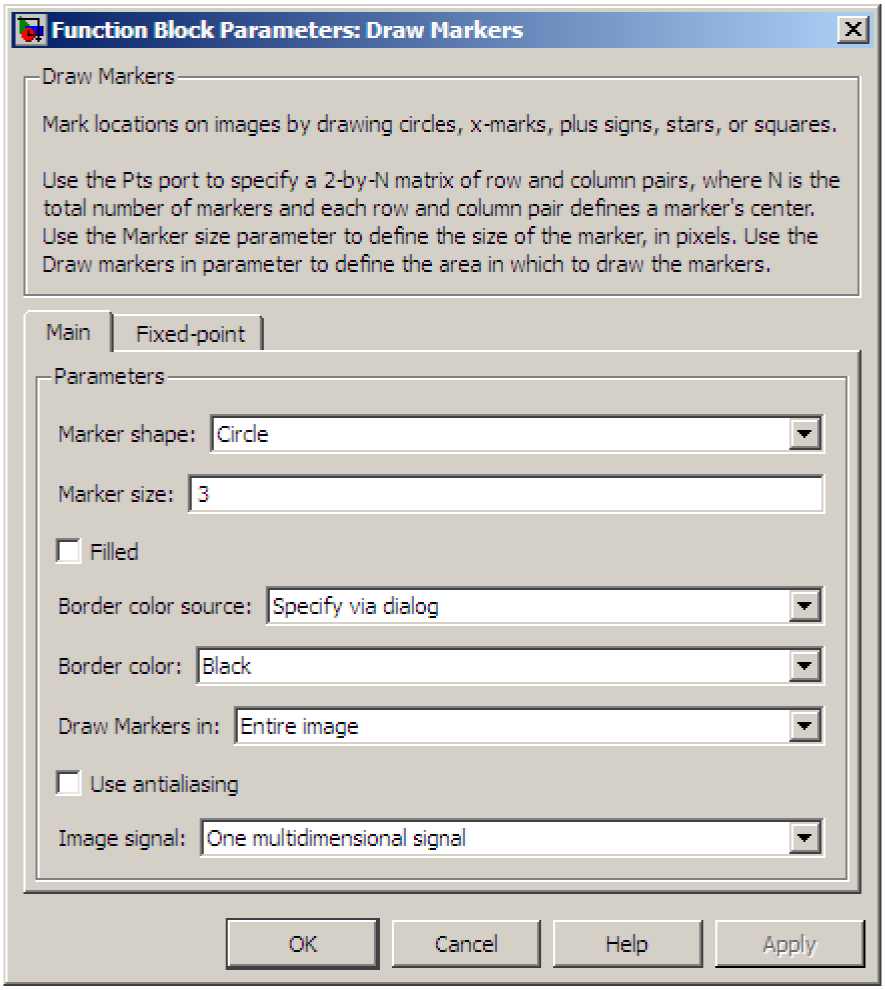
Opacity Factor

The following table describes what to enter for the **Opacity factor(s) (between 0 and 1)** parameter based on the block input and the number of markers you are drawing. This parameter is applicable when the `Filled` check box is selected.

Opacity Factor value for Drawing One Marker or Multiple Markers with the Same Color	**Oopacity Factor value for Drawing Multiple Marker with Unique Color**
Scalar intensity value	R-element vector where R is the number of markers

Dialog Box

The Draw Markers dialog box appears as shown in the following figure.



Function Block Parameters: Draw Markers

Draw Markers

Mark locations on images by drawing circles, x-marks, plus signs, stars, or squares.

Use the Pts port to specify a 2-by-N matrix of row and column pairs, where N is the total number of markers and each row and column pair defines a marker's center. Use the Marker size parameter to define the size of the marker, in pixels. Use the Draw markers in parameter to define the area in which to draw the markers.

Main | Fixed-point

Parameters

Marker shape: Circle

Marker size: 3

☐ Filled

Border color source: Specify via dialog

Border color: Black

Draw Markers in: Entire image

☐ Use antialiasing

Image signal: One multidimensional signal

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`.

Fill color source
: Specify source for fill color value to either `Specify via dialog` or `Input port`. This parameter is visible if you select the **Filled** check box.

Fill color
: If you select `Black`, the marker is black. If you select `White`, the marker is white. If you select `User-specified value`, the **Color value(s)** parameter appears in the dialog box. This parameter is visible if you select the **Filled** check box.

Border color source
: Specify source for the border color value to either `Specify via dialog` or `Input port`. Border color options are visible when the fill shapes options are not selected. This parameter is visible if you select the **Filled** check box.

Border color
: Specify the appearance of the shape's border. If you select `Black`, the border is black. If you select `White`, the border is white. If you select `User-specified value`, the **Color value(s)** parameter appears in the dialog box. This parameter is visible if you clear the **Fill shapes** check box.

Color value(s)
: Specify an intensity or color value for the marker's border or fill. This parameter is visible if, for the **Border color** or **Fill color** parameter, you select `User-specified value`. Tunable.

Opacity factor (between 0 and 1)
: Specify the opacity of the shading inside the marker, where 0 is transparent and 1 is opaque. This parameter is visible if you select the **Filled** check box. This parameter is tunable.

Draw markers in
: Define the area in which to draw the markers. If you select `Entire image`, you can draw markers in the entire image. If you select `Specify region of interest via port`, the ROI port appears on the block. Enter a four-element vector, `[r c height width]`, where `r` and `c` are the row and column coordinates of the upper-left corner of the area, and `height` and `width` represent the height (in rows) and width (in columns) of the area.

Use antialiasing
: Perform a smoothing algorithm on the marker. This parameter is visible if, for the **Marker shape** parameter, you select `Circle`, `X-mark`, or `Star`.

Image signal
: Specify how to input and output a color video signal. If you select `One multidimensional signal`, the block accepts an *M-by-N-by-P* color video signal, where *P* is the number of color planes, at one port. If you select `Separate color signals`, additional ports appear on the block. Each port accepts one *M-by-N* plane of an RGB video stream.

See Also

Draw Shapes	Video and Image Processing Blockset software
Insert Text	Video and Image Processing Blockset software

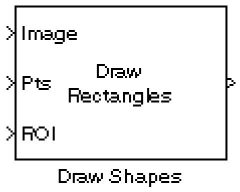
Purpose

Draw rectangle around region of interest (ROI)

Library

`vipobslib`

Description



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

The Draw Shape block draws a rectangle around a user-defined ROI by overwriting pixel values. As a result, the rectangle is embedded on the output image.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean	No

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.	• 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.

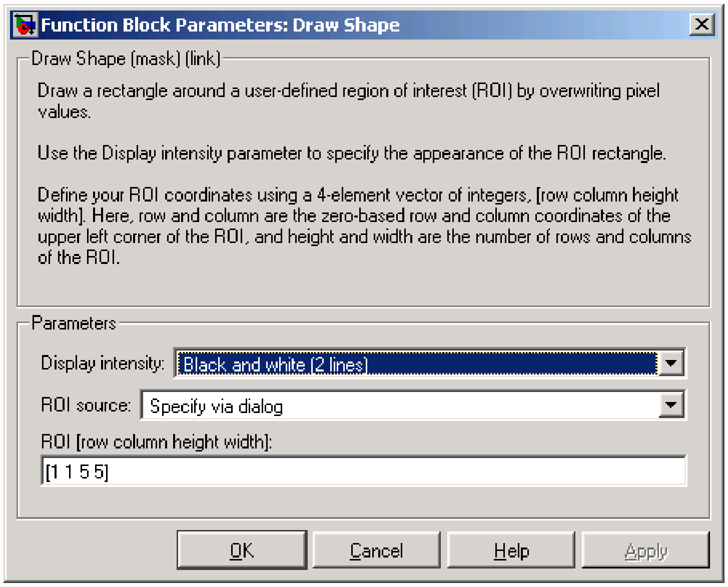
Use the **Display intensity** parameter to determine the appearance of the ROI rectangle. If you select `Black` or `White`, the rectangle is black or white, respectively. If you select `Black and white (2 lines)`, the rectangle is created by a black line on the outside and a white line on the inside. 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.

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.



Function Block Parameters: Draw Shape

Draw Shape (mask) (link)

Draw a rectangle around a user-defined region of interest (ROI) by overwriting pixel values.

Use the Display intensity parameter to specify the appearance of the ROI rectangle.

Define your ROI coordinates using a 4-element vector of integers, [row column height width]. Here, row and column are the zero-based row and column coordinates of the upper left corner of the ROI, and height and width are the number of rows and columns of the ROI.

Parameters

Display intensity: Black and white (2 lines)

ROI source: Specify via dialog

ROI [row column height width]:

[1 1 5 5]

Display intensity

Specify the appearance of the ROI rectangle. If you select `Black` or `White`, the rectangle is black or white. If you select `Black and white (2 lines)`, the rectangle is created by a black line on the outside and a white line on the inside. If you select `User-specified intensity`, the **Intensity value (0 to 1)** parameter appears in the dialog box.

Intensity value

Enter a scalar intensity value from 0 to 1, where 0 corresponds to black and 1 corresponds to white. This parameter is visible if, for the **Display intensity** parameter, you select `User-specified intensity`. Tunable.

ROI source

Specify how to enter your ROI coordinates. If you select `Specify via dialog`, the **ROI [row column height width]** parameter appears on the dialog box. If you select `Input port`, the ROI port appears on the dialog box. The input to this port must be a four-element vector of integers. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the ROI. The second two elements represent the height and width of the ROI.

ROI [row column height width]

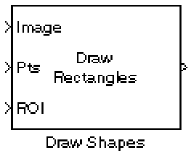
Enter a four-element vector of integers. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the ROI. The second two elements represent the height and width of the ROI. Tunable.

Draw Shapes

Purpose Draw rectangles, lines, polygons, or circles on images

Library Text & Graphics

Description The Draw Shapes block draws multiple rectangles, lines, polygons, or circles on images by overwriting pixel values. As a result, the shapes are embedded on the output image.



This block uses Bresenham's line drawing algorithm to draw lines, polygons, and rectangles. It uses Bresenham's circle drawing algorithm to draw circles.

Port	Input/Output	Supported Data Types	Complex Values Supported
Image	*M-by-N* matrix of intensity values or an *M-by-N-by-P* color values where *P* is the number of color planes	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
R, G, B	Scalar, vector, or matrix that represents one plane of the input RGB video stream. Inputs to the R, G, and B ports must have the same dimensions and data type.	Same as Image port	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Pts	Use integer values to define zero-based shape coordinates. If you enter noninteger values, the block rounds them to the nearest integer. For more information about how to specify shape coordinates for different shapes, see "Defining Shapes to Draw" on page 2-331.	• 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	*4*-element vector of integers that defines a rectangular area in which to draw the shapes. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the area. The second two elements represent the height and width of the area.	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
Clr	P-element vector or P-*by*-N matrix, where P is the number of color planes	Same as Image port	No
Output	Scalar, vector, or matrix of pixel values that contain the shape(s)	Same as Image port	No

The output signal is the same size and data type as the inputs to the Image, R, G, and B ports.

Use the **Shape** parameter to specify one of the following types type of shape(s) to draw.

- `Rectangles`
- `Lines`
- `Polygons`
- `Circles`

Use the **Draw shapes in** parameter to define one of the following types of area in which to draw the shapes.

- `Entire image`, enables you to draw shapes in the entire image.
- `Specify region of interest via port`, the ROI port appears on the block. Enter a four-element vector of integer values, `[r c height width]`, where `r` and `c` are the row and column coordinates of the upper-left corner of the area, and `height` and `width` represent the height (in rows) and width (in columns) of the area.

  Note If you specify values that are outside the image, the block sets the values to the image boundaries.

If, for the **Shape** parameter, you select:

- `Lines`, `Polygons`, or `Circles`

and you then select the,

- **Use antialiasing** check box,

the block performs a smoothing algorithm. The Draw Shapes block uses an algorithm similar to the poly2mask function to determine which subpixels to draw.

Use the **Image signal** parameter to specify one of the following ways to input and output a color video signal.

- `One multidimensional signal`, the block accepts an *M-by-N-by-P* color video signal, where *P* is the number of color planes, at one port.
- `Separate color signals`, additional ports appear on the block. Each port accepts one *M-by-N* plane of an RGB video stream.

Selecting Shape Fill and Border Colors

You can set the shape fill or border color via the input port or via the input dialog. Use the color input or color parameter to determine the appearance of the rectangle(s), line(s), polygon(s), or circle(s).

- Fill Color on page 329
- Border Color on page 330
- Color Values on page 330
-

Fill Color

If you select the **Fill shapes** check box, the **Fill color source**, **Fill color** and **Opacity factor (between 0 and 1)** parameters appear in the dialog box. Use the **Fill color source** parameter to specify either `Input port` or `Specify via dialog` for the color source. If `Specify via dialog` is selected, you can specify either `Black`, `White`, or `User-specified value` for the **Fill color** parameter for the shading inside the shape. The **Color value(s)** parameter is applicable when the `User-specified value` is selected. Use the **Opacity factor (between 0 and 1)** parameter to specify the opacity of the shading inside the shape, where 0 is transparent and 1 is opaque.

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

Border Color

If the **Fill shapes** check box is not selected, the **Border color source**, and **Border color** parameters are available. Use the **Border color source** parameter to specify either `Input port` or `Specify via dialog` for the color source. If `Specify via dialog` is selected, you can specify either `Black`, `White`, or `User-specified value` for the **Border color** parameter. If the color is user specified, the **Color value(s)** parameter is used to enter the color.

Color Values

The following table describes what to enter for the **Color Value(s)** parameter based on the block input and the number of shapes you are drawing.

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

For each value in the **Color Value(s)** parameter, enter a number between the minimum and maximum values that can be represented by the data type of the input image. If you enter a value outside this range, the block produces an error message.

Opacity Factor

The following table describes what to enter for the **Opacity factor(s) (between 0 and 1)** parameter based on the block input and the number of shapes you are drawing. This parameter is applicable when the `Filled` check box is selected.

Opacity Factor value for Drawing One Shape or Multiple Shapes with the Same Color	Oopacity Factor value for Drawing Multiple Shapes with Unique Color
Scalar intensity value	R-element vector where R is the number of shapes

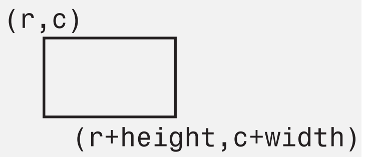
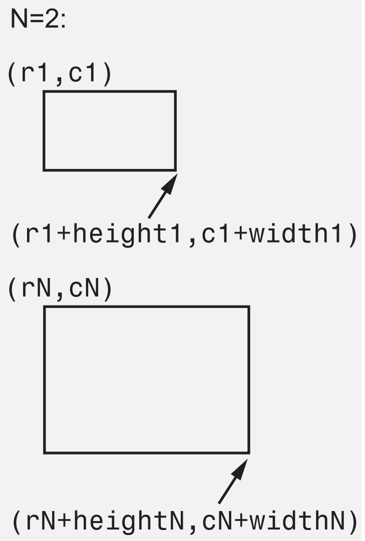
Defining Shapes to Draw

This section explains how to use the **Shape** parameter and the Pts port to draw the following shapes:

- Drawing Rectangles on page 331
- Drawing Lines and Polylines on page 332
- Drawing Polygons on page 335
- Drawing Circles on page 337

Drawing Rectangles

The Draw Shapes block lets you draw one or more rectangles. Set the **Shape** parameter to `Rectangles`, and then follow the instructions in the table to specify the input to the Pts port to obtain the desired number of rectangles.

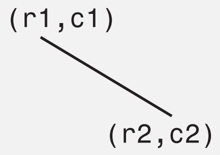
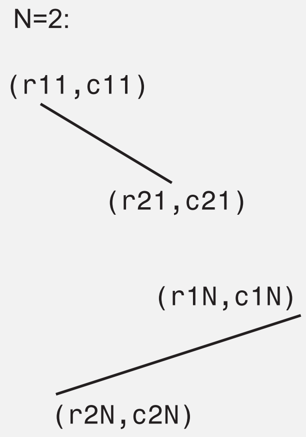
Shape	Input to the Pts Port	Drawn Shape
Single Rectangle	Four-element row or column vector `[r c height width]` where • `r` and `c` are the zero-based row and column coordinates of the upper-left corner of the rectangle. • `height` and `width` are the height, in pixels, and width, in pixels, of the rectangle. Here, `height` and `width` must be greater than 0.	
N Rectangles	*4-by-N* matrix $\begin{bmatrix} r_1 & \cdots & r_N \\ c_1 & \cdots & c_N \\ height_1 & \cdots & height_N \\ width_1 & \cdots & width_N \end{bmatrix}$ where each column of the matrix corresponds to a different rectangle and is of the same form as the vector for a single rectangle.	 N=2: (r1,c1) (r1+height1,c1+width1) (rN,cN) (rN+heightN,cN+widthN)

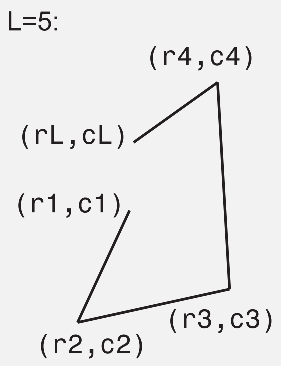
For an example of how to use the Draw Shapes block to draw a rectangle, see “Tracking an Object Using Correlation”.

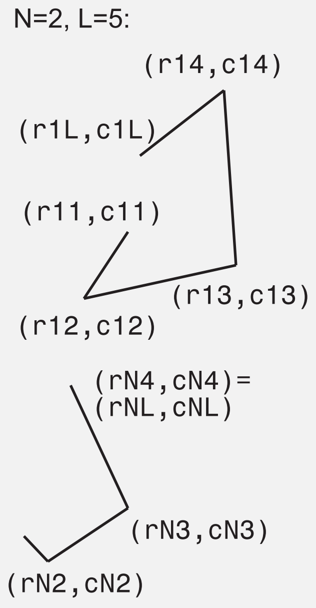
Drawing Lines and Polylines

The Draw Shapes block lets you draw either a single line, or one or more polylines, where each polyline is a series of connected line segments.

Set the **Shape** parameter to `Lines`, and then follow the instructions in the table to specify the input to the Pts port to obtain the desired shape.

Shape	Input to the Pts Port	Drawn Shape
Single Line	Four-element row or column vector `[r1 c1 r2 c2]` where • `r1` and `c1` are the row and column coordinates of the beginning of the line. • `r2` and `c2` are the row and column coordinates of the end of the line.	
N Lines	*4-by-N* matrix $\begin{bmatrix} r_{11} & \cdots & r_{1N} \\ c_{11} & \cdots & c_{1N} \\ r_{21} & \cdots & r_{2N} \\ c_{21} & \cdots & c_{2N} \end{bmatrix}$ where each column of the matrix corresponds to a different line and is of the same form as the vector for a single line.	

Shape	Input to the Pts Port	Drawn Shape
Single Polyline with L-1 Segments	Vector of size *2L* `[r1 c1 r2 c2 ... rL cL]` where • `r1` and `c1` are the row and column coordinates of the beginning of the first line segment. • `r2` and `c2` are the row and column coordinates of the end of the first line segment and the beginning of the second line segment • `rL` and `cL` are the row and column coordinates of the end of the L-1th line segment.	

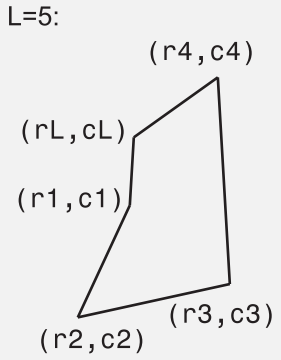
Shape	Input to the Pts Port	Drawn Shape
	The block produces an error message if the number of rows is less than two or is not a multiple of two.	
N Polylines with the largest number of line segments in any line being L-1	*2L-by-N* matrix $$\begin{bmatrix} r_{11} & \cdots & r_{N1} \\ c_{11} & \cdots & c_{N1} \\ r_{12} & \cdots & r_{N2} \\ c_{12} & \cdots & c_{N2} \\ \vdots & \ddots & \vdots \\ r_{1L} & \cdots & r_{NL} \\ c_{1L} & \cdots & cL \end{bmatrix}$$ where each column of the matrix corresponds to a different polyline and is of the same form as the vector for a single polyline. If some polylines are shorter than others, repeat the ending coordinates to fill the polyline matrix. The block produces an error message if the number of rows is less than two or is not a multiple of two.	 N=2, L=5: (r14,c14) (r1L,c1L) (r11,c11) (r13,c13) (r12,c12) (rN4,cN4)= (rNL,cNL) (rN3,cN3) (rN2,cN2)

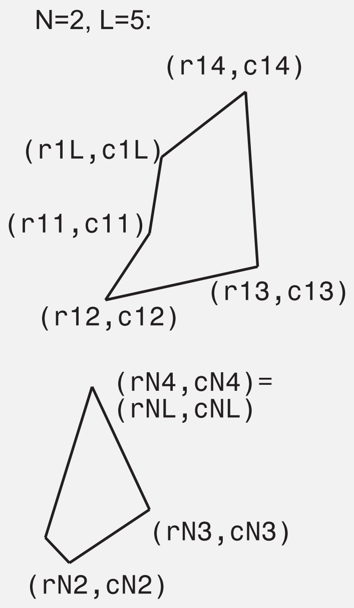
If you select the **Use antialiasing** check box, the block applies an edge smoothing algorithm.

For examples of how to use the Draw Shapes block to draw a line, see "Finding Lines in Images" and "Measuring an Angle Between Lines".

Drawing Polygons

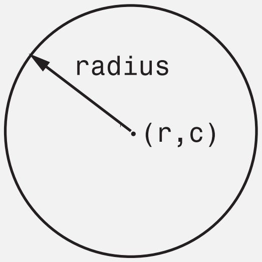
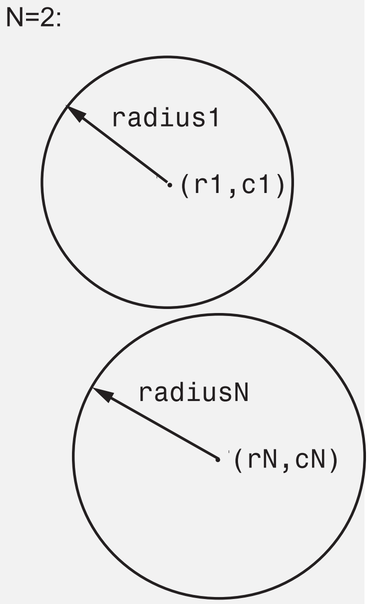
The Draw Shapes block lets you draw one or more polygons. Set the **Shape** parameter to `Polygons`, and then follow the instructions in the table to specify the input to the Pts port to obtain the desired number of polygons.

Shape	Input to the Pts Port	Drawn Shape
Single Polygon with L line segments	Row or column vector of size 2L `[r1 c1 r2 c2 ... rL cL]` where • `r1` and `c1` are the row and column coordinates of the beginning of the first line segment. • `r2` and `c2` are the row and column coordinates of the end of the first line segment and the beginning of the second line segment • `rL` and `cL` are the row and column coordinates of the end of the L-1th line segment and the beginning of the Lth line segment. The block connects `[r1 c1]` to `[rL cL]` to complete the polygon. The block	

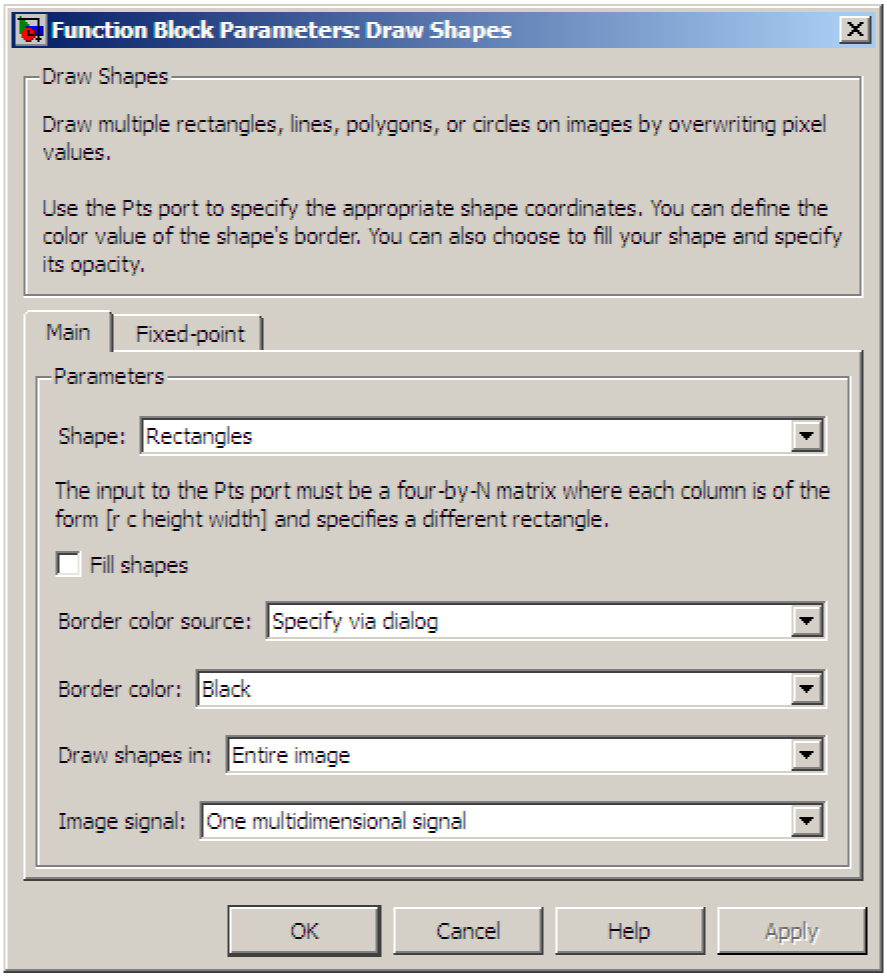
Shape	Input to the Pts Port	Drawn Shape
	produces an error if the number of rows is negative or not a multiple of two.	
N Polygons with the largest number of line segments in any line being L	2L-by-N matrix $$\begin{bmatrix} r_{11} & \cdots & r_{N1} \\ c_{11} & \cdots & c_{N1} \\ r_{12} & \cdots & r_{N2} \\ c_{12} & \cdots & c_{N2} \\ \vdots & \ddots & \vdots \\ r_{1L} & \cdots & r_{NL} \\ c_{1L} & \cdots & cL \end{bmatrix}$$ where each column of the matrix corresponds to a different polygon and is of the same form as the vector for a single polygon. If some polygons are shorter than others, repeat the ending coordinates to fill the polygon matrix. The block produces an error message if the number of rows is less than two or is not a multiple of two.	 N=2, L=5: (r14,c14) (r1L,c1L) (r11,c11) (r13,c13) (r12,c12) (rN4,cN4)= (rNL,cNL) (rN3,cN3) (rN2,cN2)

Drawing Circles

The Draw Shapes block lets you draw one or more circles. Set the **Shape** parameter to `Circles`, and then follow the instructions in the table to specify the input to the Pts port to obtain the desired number of circles.

Shape	Input to the Pts Port	Drawn Shape
Single Circle	Three-element row or column vector `[r c radius]` where • `r` and `c` are the row and column coordinates of the center of the circle. • `radius` is the radius of the circle, which must be greater than 0.	
N Circles	*3-by-N* matrix $$\begin{bmatrix} r_1 & \cdots & r_N \\ c_1 & \cdots & c_N \\ radius_1 & \cdots & radius_N \end{bmatrix}$$ where each column of the matrix corresponds to a different circle and is of the same form as the vector for a single circle.	

Dialog Box



Function Block Parameters: Draw Shapes

Draw Shapes

Draw multiple rectangles, lines, polygons, or circles on images by overwriting pixel values.

Use the Pts port to specify the appropriate shape coordinates. You can define the color value of the shape's border. You can also choose to fill your shape and specify its opacity.

Main | Fixed-point

Parameters

Shape: Rectangles

The input to the Pts port must be a four-by-N matrix where each column is of the form [r c height width] and specifies a different rectangle.

☐ Fill shapes

Border color source: Specify via dialog

Border color: Black

Draw shapes in: Entire image

Image signal: One multidimensional signal

Shape
: Specify the type of shape(s) to draw. Your choices are `Rectangles`, `Lines`, `Polygons`, or `Circles`.

Fill shapes
: Fill the shape with an intensity value or a color.

Fill color source
Specify source for fill color value to either `Specify via dialog` or `Input port`. This parameter is visible if you select the **Fill shapes** check box.

Fill color
If you select `Black`, the border is black. If you select `White`, the border is white. If you select `User-specified value`, the **Color value(s)** parameter appears in the dialog box. This parameter is visible if you select the **Fill shapes** check box.

Border color source
Specify source for the border color value to either `Specify via dialog` or `Input port`. Border color options are visible when the fill shapes options are not selected. This parameter is visible if you select the **Fill shapes** check box.

Border color
Specify the appearance of the shape's border. If you select `Black`, the border is black. If you select `White`, the border is white. If you select `User-specified value`, the **Color value(s)** parameter appears in the dialog box. This parameter is visible if you clear the **Fill shapes** check box.

Color value(s)
Specify an intensity or color value for the shape's border or fill. This parameter is visible if, for the **Border color** or **Fill color** parameter, you select `User-specified value`. This parameter is tunable.

Opacity factor (between 0 and 1)
Specify the opacity of the shading inside the shape, where 0 is transparent and 1 is opaque. This parameter is visible if you select the **Fill shapes** check box.

Draw shapes in
Define the area in which to draw the shapes. If you select `Entire image`, you can draw shapes in the entire image. If you select `Specify region of interest via port`, the ROI port appears on the block. Enter a four-element vector, `[r c height`

`width]`, where `r` and `c` are the row and column coordinates of the upper-left corner of the area, and `height` and `width` represent the height (in rows) and width (in columns) of the area.

Use antialiasing

Perform a smoothing algorithm on the line, polygon, or circle. This parameter is visible if, for the **Shape** parameter, you select `Lines`, `Polygons`, or `Circles`.

Image signal

Specify how to input and output a color video signal. If you select `One multidimensional signal`, the block accepts an *M-by-N-by-P* color video signal, where *P* is the number of color planes, at one port. If you select `Separate color signals`, additional ports appear on the block. Each port accepts one *M-by-N* plane of an RGB video stream.

See Also

Draw Markers	Video and Image Processing Blockset software
Insert Text	Video and Image Processing Blockset software

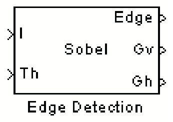
Purpose

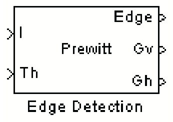
Find edges of objects in images using Sobel, Prewitt, Roberts, or Canny method

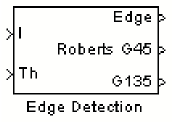
Library

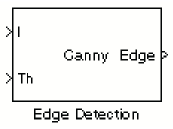
Analysis & Enhancement

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	• Double-precision floating point • Single-precision floating point • Fixed point (not supported for the Canny method) • 8-, 16-, 32-bit signed integer (not supported for the Canny method)	No

Port	Input/Output	Supported Data Types	Complex Values Supported
		• 8-, 16-, 32-bit unsigned integer (not supported for the Canny method)	
Th	Matrix of intensity values	Same as I port	No
Edge	Matrix that represents a binary image	Boolean	No
Gv	Matrix of gradient values in the vertical direction	Same as I port	No
Gh	Matrix of gradient values in the horizontal direction	Same as I port	No
G45	Matrix of gradient values	Same as I port	No
G135	Matrix of gradient values	Same as I port	No

The output of the Gv, Gh, G45, and G135 ports is the same data type as the input to the I port. The input to the Th port must be the same data type as the input to the I port.

Use the **Method** parameter to specify which algorithm to use to find edges. You can select `Sobel`, `Prewitt`, `Roberts`, or `Canny` to find edges using the Sobel, Prewitt, Roberts, or Canny method.

Sobel, Prewitt, and Roberts Methods

Use the **Output type** parameter to select the format of the output. If you select `Binary image`, the block outputs a Boolean matrix at the Edge port. The nonzero elements of this matrix correspond to the edge pixels and the zero elements correspond to the background pixels. If you select `Gradient components` and, for the **Method** parameter, you select `Sobel` or `Prewitt`, the block outputs the gradient components that correspond to the horizontal and vertical edge responses at the Gh and Gv ports, respectively. If you select `Gradient components` and, for the **Method** parameter, you select `Roberts`, the block outputs the gradient components that correspond to the 45 and 135 degree edge responses at the G45 and G135 ports, respectively. If you select `Binary image and gradient components`, the block outputs both the binary image and the gradient components of the image.

Select the **User-defined threshold** check box to define a threshold values or values. If you clear this check box, the block computes the threshold for you.

Use the **Threshold source** parameter to specify how to enter your threshold value. If you select `Specify via dialog`, the **Threshold** parameter appears in the dialog box. Enter a threshold value that is within the range of your input data. If you choose `Input port`, use input port Th to specify a threshold value. This value must have the same data type as the input data. Gradient magnitudes above the threshold value correspond to edges.

The Edge Detection block computes the automatic threshold using the mean of the gradient magnitude squared image. However, you can adjust this threshold using the **Threshold scale factor (used to automatically calculate threshold value)** parameter. The block multiplies the value you enter with the automatic threshold value to determine a new threshold value.

Select the **Edge thinning** check box to reduce the thickness of the edges in your output image. This option requires additional processing time and memory resources.

Note This block is most efficient in terms of memory usage and processing time when you clear the **Edge thinning** check box and use the **Threshold** parameter to specify a threshold value.

Canny Method

Select the **User-defined threshold** check box to define the low and high threshold values. If you clear this check box, the block computes the threshold values for you.

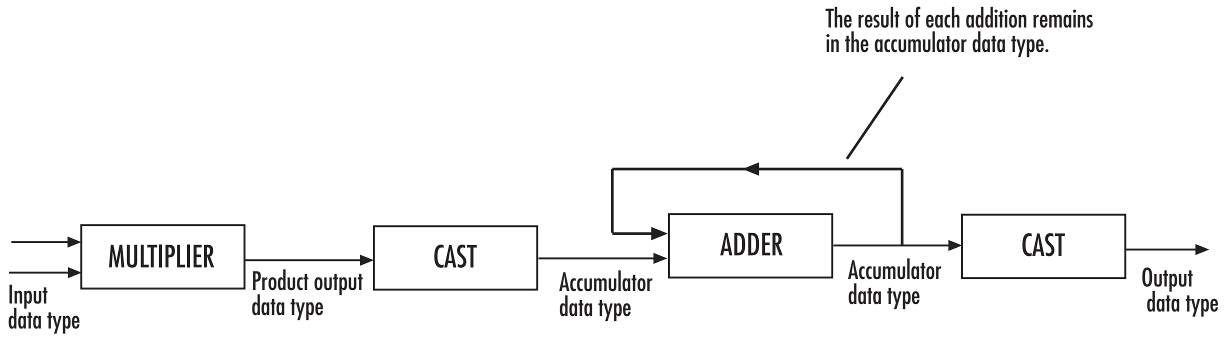
Use the **Threshold source** parameter to specify how to enter your threshold values. If you select `Specify via dialog`, the **Threshold [low high]** parameter appears in the dialog box. Enter the threshold values. If a pixel's magnitude in the gradient image, which is formed by convolving the input image with the derivative of the Gaussian filter, exceeds the high threshold, then the pixel corresponds to a strong edge. Any pixel connected to a strong edge and having a magnitude greater than the low threshold corresponds to a weak edge. If, for the **Threshold source** parameter, you choose `Input port`, use input port Th to specify a two-element vector of threshold values. These values must have the same data type as the input data.

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

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

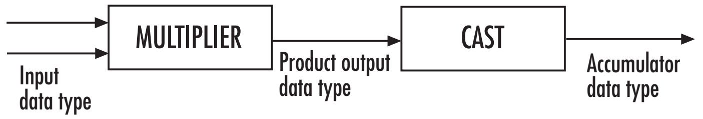
Fixed-Point Data Types

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

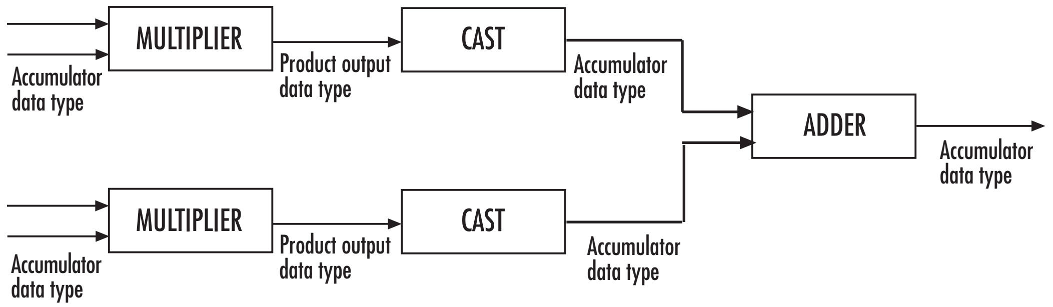


The block squares the threshold and compares it to the sum of the squared gradients to avoid using square roots.

Threshold:



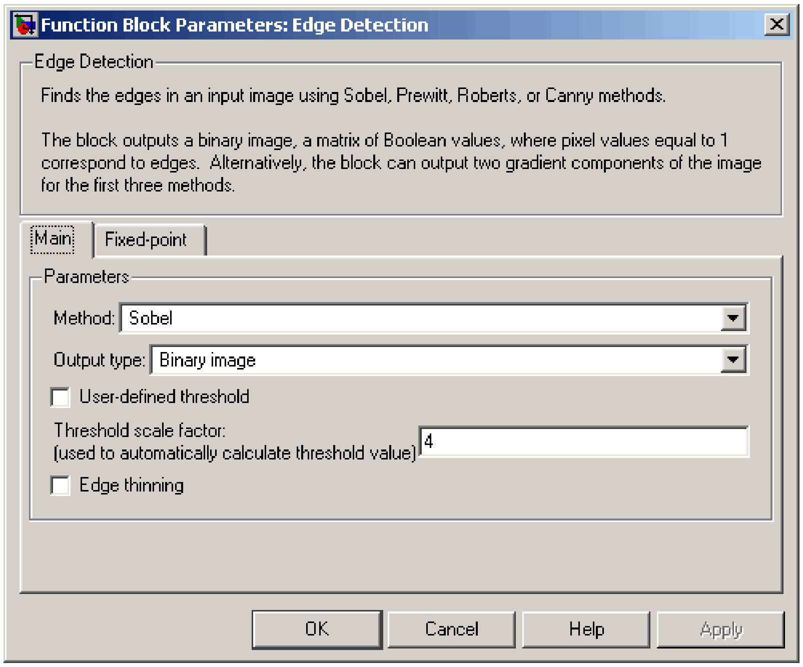
Gradients:



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

Dialog Box

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



Edge Detection

Finds the edges in an input image using Sobel, Prewitt, Roberts, or Canny methods.

The block outputs a binary image, a matrix of Boolean values, where pixel values equal to 1 correspond to edges. Alternatively, the block can output two gradient components of the image for the first three methods.

Parameters

Method: Sobel

Output type: Binary image

☐ User-defined threshold

Threshold scale factor: (used to automatically calculate threshold value) 4

☐ Edge thinning

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`. .

Threshold [low high]

Enter the low and high threshold values that define the weak and strong edges. This parameter is visible if, for the **Method** parameter, you select `Canny`. Then you select the **User-defined threshold** check box, and, for **Threshold source** parameter, you select `Specify via dialog`. Tunable.

Threshold scale factor (used to automatically calculate threshold value)

Enter a multiplier that is used to adjust the calculation of the automatic threshold. This parameter is visible if, for the **Method** parameter, you select `Sobel`, `Prewitt`, or `Roberts`, and you clear the **User-defined threshold** check box. Tunable.

Edge thinning

Select this check box if you want the block to perform edge thinning. This option requires additional processing time and memory resources. This parameter is visible if, for the **Method** parameter, you select `Sobel`, `Prewitt`, or `Roberts`, and for the **Output type** parameter, you select `Binary image` or `Binary image and gradient components`.

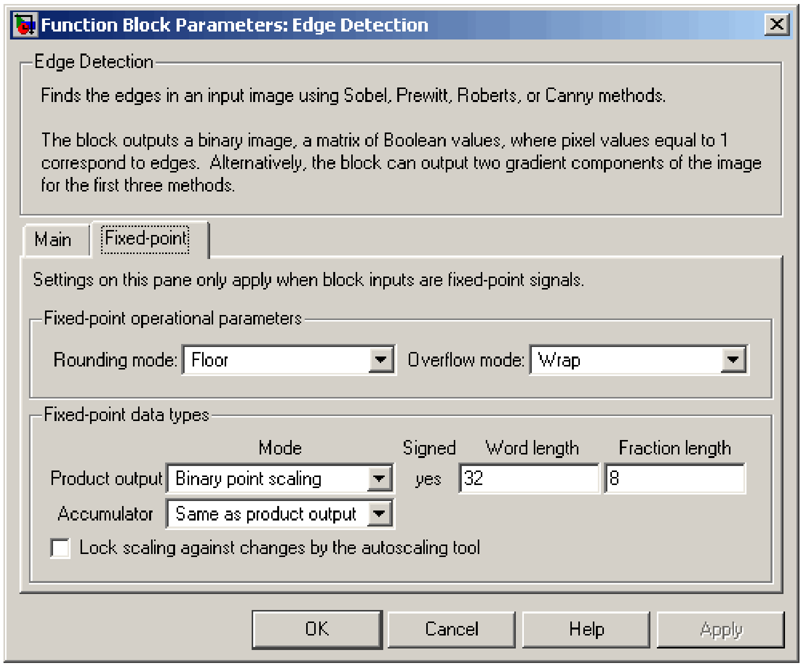
Approximate percentage of weak edge and nonedge pixels (used to automatically calculate threshold values)

Enter the approximate percentage of weak edge and nonedge image pixels. The block computes the automatic threshold values using this approximation. This parameter is visible if, for the **Method** parameter, you select `Canny`. Tunable.

Standard deviation of Gaussian filter

Enter the standard deviation of the Gaussian filter whose derivative is convolved with the input image. This parameter is visible if, for the **Method** parameter, you select `Canny`.

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



Edge Detection

Finds the edges in an input image using Sobel, Prewitt, Roberts, or Canny methods.

The block outputs a binary image, a matrix of Boolean values, where pixel values equal to 1 correspond to edges. Alternatively, the block can output two gradient components of the image for the first three methods.

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

Fixed-point operational parameters

Rounding mode: Floor   Overflow mode: Wrap

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Product output	Binary point scaling	yes	32	8
Accumulator	Same as product output			

☐ Lock scaling against changes by the autoscaling tool

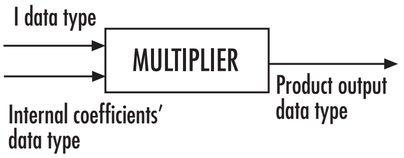
Rounding mode

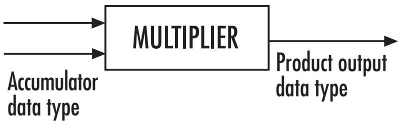
Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

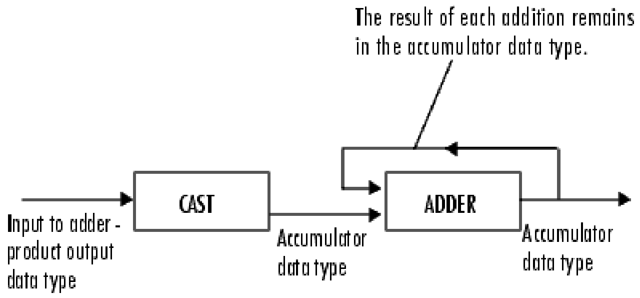




Here, the internal coefficients are the Sobel, Prewitt, or Roberts masks. As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths.

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

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type

As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

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

Gradients

Choose how to specify the word length and fraction length of the outputs of the Gv and Gh ports. This parameter is visible if, for the **Output type** parameter, you choose `Gradient components` or `Binary image and gradient components`:

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

Lock 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

Erosion

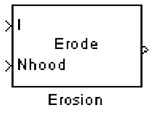
Purpose

Find local minima in binary or intensity images

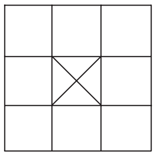
Library

Morphological Operations

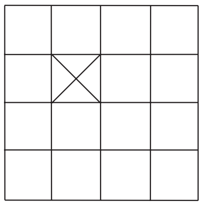
Description



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.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
Nhood	Matrix or vector of 1s and 0s that represents the neighborhood values	Boolean	No
Output	Vector or matrix of intensity values that represents the eroded image	Same as I port	No

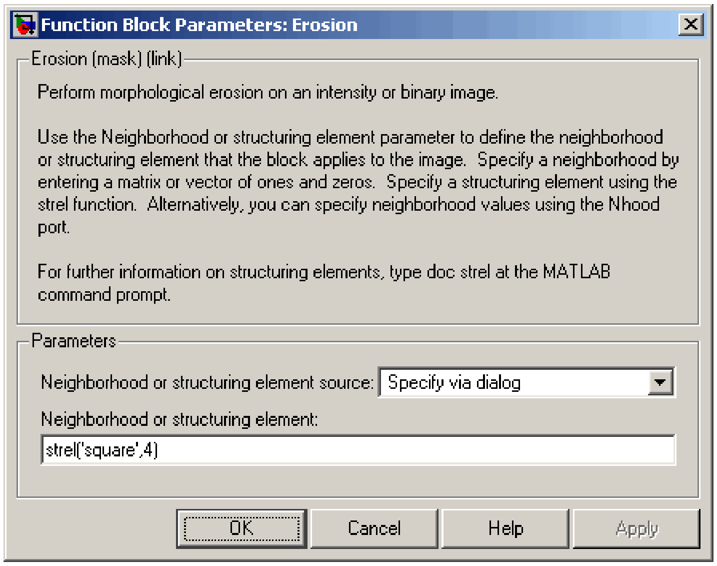
The output signal is the same data type as the input to the I port.

Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select `Specify via dialog`, the **Neighborhood or structuring element** parameter appears in the dialog box. If you select `Input port`, the Nhood port appears on the block. Use this port to enter your neighborhood values as a matrix or vector of 1s and 0s. You can only specify a structuring element using the dialog box.

Use the **Neighborhood or structuring element** parameter to define the neighborhood or structuring element that the block applies to the image. Specify a neighborhood by entering a matrix or vector of 1s and 0s. Specify a structuring element with the `strel` function from the Image Processing Toolbox. If the structuring element is decomposable into smaller elements, the block executes at higher speeds due to the use of a more efficient algorithm. If you enter an array of STREL objects, the block applies each object to the entire matrix in turn.

Dialog Box

The Erosion dialog box appears as shown in the following figure.


Function Block Parameters: Erosion

Erosion (mask) (link)

Perform morphological erosion on an intensity or binary image.

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 ones and zeros. Specify a structuring element using the strel function. Alternatively, you can specify neighborhood values using the Nhood port.

For further information on structuring elements, type doc strel at the MATLAB command prompt.

Parameters

Neighborhood or structuring element source: Specify via dialog

Neighborhood or structuring element:

strel('square',4)

Neighborhood or structuring element source
: Specify how to enter your neighborhood or structuring element values. Select `Specify via dialog` to enter the values in the dialog box. Select `Input port` to use the Nhood port to specify the neighborhood values. You can only specify a structuring element using the dialog box.

Neighborhood or structuring element
: If you are specifying a neighborhood, this parameter must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the `strel` function from the Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or structuring element source** parameter, you select `Specify via dialog`.

References

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

See Also

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

Estimate Geometric Transformation

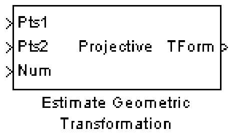
Purpose

Compute transformation matrix between greatest number of point pairs of two images

Library

Geometric Transformations

Description



Use the Estimate Geometric Transformation block to find the transformation matrix which maps the greatest number of point pairs between two images. A *point pair* refers to a point in the input image and its related point on the image created using the transformation matrix. You can select to use the RANdom SAmple Consensus (RANSAC) or the Least Median Squares algorithm to exclude outliers and to calculate the transformation matrix. You can also use all input points to calculate the transformation matrix.

Port	Input/Output	Supported Data Types	Complex Values Supported
Pts1/Pts2	2xN Matrix, (where N is the maximum number of points) coordinates of the input points	• Double • Single • 8, 16, 32-bit signed integer • 8, 16, 32-bit unsigned integer	No
Num	Scalar value that represents the number of valid points in Pts1 and Pts 2	• 8, 16, 32-bit signed integer • 8, 16, 32-bit unsigned integer	No

Port	Input/Output	Supported Data Types	Complex Values Supported
TForm	2x3 or 3x3, the transformation matrix	• Double • Single	No
Inlier	1xN, indicates which points have been used to calculate TForm	Boolean	No

Ports `Pts1` and `Pts2` are the points on two images that have the same data type. When `Pts 1` and `Pts 2` are single or double, the output transformation matrix will also have single or double data type. When `Pts1` and `Pts2` images are built-in integers, the option is available to set the transformation matrix data type to either `Single` or `Double`. The `TForm` output provides the transformation matrix. The `Inlier` output port provides the Inlier points on which the transformation matrix is based. This output appears when you select the **Output Boolean signal indication which point pairs are inliers** checkbox.

RANSAC and Least Median Squares Algorithms

The RANSAC algorithm relies on a distance threshold. A pair of points, p_i^a (image a, Pts1) and p_i^b (image b, Pts 2) is an inlier only when the distance between p_i^b and the projection of p_i^a based on the transformation matrix falls within the specified threshold. The distance metric used in the RANSAC algorithm is as follows:

$$d = \sum_{i=1}^{Num} \min(D(p_i^b, \psi(p_i^a : H)), t)$$

The Least Median Squares algorithm assumes at least 50% of the point pairs can be mapped by a transformation matrix. The algorithm does not need to explicitly specify the distance threshold. Instead, it uses the median distance between all input point pairs. The distance metric used in the Least Median of Squares algorithm is as follows:

$$d = median(D(p_1^b, \psi(p_1^a : H)), D(p_2^b, \psi(p_2^a : H)), \ldots, D(p_{Num}^b, \psi(p_N^a : H)))$$

For both equations:

p_i^a is a point in image a (`Pts1`)

p_i^b is a point in image b (`Pts2`)

$\psi(p_i^a : H)$ is the projection of a point on image a based on transformation matrix H

$D(p_i^b, p_j^b)$ is the distance between two point pairs on image b

t is the threshold

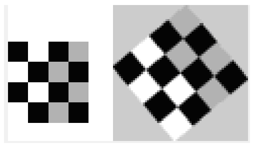
Num is the number of points

The smaller the distance metric, the better the transformation matrix and therefore the more accurate the projection image.

Transformations

The Estimate Geometric Transformation block supports `Nonreflective similarity`, `affine`, and `projective` transformation types, which are described in this section.

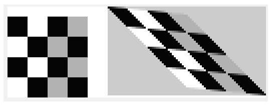
Nonreflective similarity transformation supports translation, rotation, and isotropic scaling. It has four degrees of freedom and requires two pairs of points.



The transformation matrix is: $H = \begin{bmatrix} h_1 & h_2 & h_3 \\ -h_2 & h_1 & h_4 \end{bmatrix}$

The projection of a point $[x \quad y]^T$ by H is: $[\hat{x} \quad \hat{y}]^T = H[x \quad y \quad 1]^T$

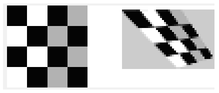
affine transformation supports nonisotropic scaling in addition to all transformations that the nonreflective similarity transformation supports. It has six degrees of freedom that can be determined from three pairs of noncollinear points.



The transformation matrix is: $H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \end{bmatrix}$

The projection of a point $[x \quad y]^T$ by H is: $[\hat{x} \quad \hat{y}]^T = H[x \quad y \quad 1]^T$

Projective transformation supports tilting in addition to all transformations that the affine transformation supports.



The transformation matrix is : $h = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$

The projection of a point $[x \quad y]^T$ by H is represented by homogeneous coordinates as: $[\hat{u} \quad \hat{v} \quad \hat{w}]^T = H[x \quad y \quad 1]^T$

Distance Measurement

For computational simplicity and efficiency, this block uses algebraic distance. The algebraic distance for a pair of points, $\begin{bmatrix} x^a & y^a \end{bmatrix}^T$ on image a, and $\begin{bmatrix} x^b & y^b \end{bmatrix}^T$ on image b , according to transformation H, is defined as follows;

For projective transformation:

$$D(p_i^b, \psi(p_i^a : H)) = ((u^a - w^a x^b)^2 + (v^a - w^a y^b)^2)^{\frac{1}{2}}, \text{ where}$$

$$\begin{bmatrix} \hat{u}^a & \hat{v}^a & \hat{w}^a \end{bmatrix}^T = H \begin{bmatrix} x^a & y^a & 1 \end{bmatrix}^T$$

For Nonreflective similarity or affine transformation:

$$D(p_i^b, \psi(p_i^a : H)) = ((x^a - x^b)^2 + (y^a - y^b)^2)^{\frac{1}{2}},$$

where $\begin{bmatrix} \hat{x}^a & \hat{y}^a \end{bmatrix}^T = H \begin{bmatrix} x^a & y^a & 1 \end{bmatrix}^T$

Algorithm

The block performs a comparison and repeats it M number of times between successive transformation matrices. If you select the **Find and exclude outliers** option, the RANSAC and Least Median Squares (LMS) algorithms become available. These algorithms calculate and compare a distance metric. The transformation matrix that produces the smaller distance metric becomes the new transformation matrix that the next comparison uses. A final transformation matrix is resolved when either:

- M number of random samplings is performed
- The RANSAC algorithm, when enough number of inlier point pairs can be mapped, (dynamically updating M)

The Estimate Geometric Transformation algorithm follows these steps:

1

A transformation matrix H is initialized to zeros

2

Set `count = 0` (Randomly sampling).

3

While `count < M`, where `M` is total number of random samplings to perform, perform the following;

a

Increment the count; `count = count + 1`.

b

Randomly select pair of points from images a and b, (2 pairs for Nonreflective similarity, 3 pairs for affine, or 4 pairs for projective).

c

Calculate a transformation matrix H , from the selected points.

d

If H has a distance metric less than that of H , then replace H with H .

(Optional for RANSAC algorithm only)

i.

Update `M` dynamically.

ii.

Exit out of sampling loop if enough number of point pairs can be mapped by H .

4

Use all point pairs in images a and b that can be mapped by H to calculate a refined transformation matrix H

5

Iterative Refinement, (Optional for RANSAC and LMS algorithms)

a

Denote all point pairs that can be mapped by H as inliers.

b

Use inlier point pairs to calculate a transformation matrix H .

c

If H has a distance metric less than that of H , then replace H with H , otherwise exit the loop.

Number of Random Samplings

The number of random samplings can be specified by the user for the RANSAC and Least Median Squares algorithms. You can use an additional option with the RANSAC algorithm, which calculates this number based on an accuracy requirement. The **Desired Confidence** level drives the accuracy.

The calculated number of random samplings, M used with the RANSAC algorithm, is as follows:

$$M = \frac{\log(1-p)}{\log(1-q^s)}$$

where

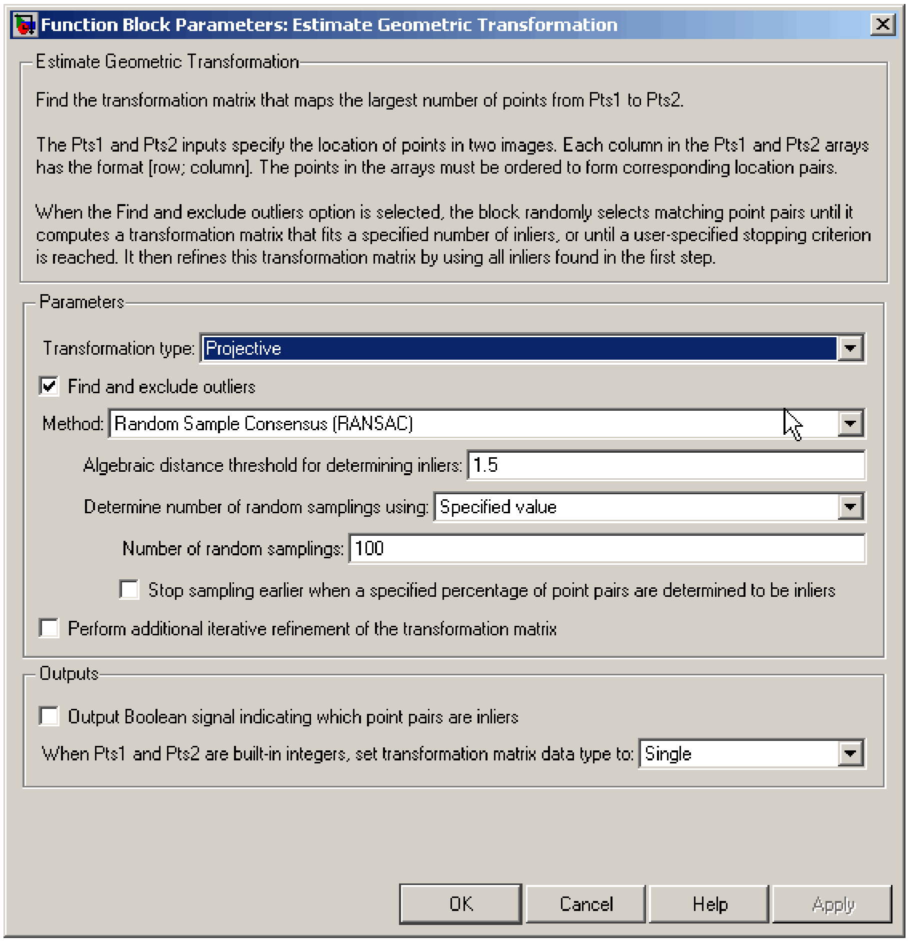
- p is the probability of independent point pairs belonging to the largest group that can be mapped by the same transformation. The probability is dynamically calculated based on the number of inliers found versus the total number of points. As the probability increases, the number of samplings, M , decreases.

- *q* is the probability of finding the largest group that can be mapped by the same transformation.
- s is equal to the value 2, 3, or 4 for Nonreflective similarity, affine, and projective transformation, respectively.

Iterative Refinement of Transformation Matrix

The transformation matrix calculated from all inliers can be used to calculate a refined transformation matrix. The refined transformation matrix is then used to find a new set of inliers. This procedure can be repeated until the transformation matrix cannot be further improved. This iterative refinement is optional.

Dialog Box


Function Block Parameters: Estimate Geometric Transformation

Estimate Geometric Transformation

Find the transformation matrix that maps the largest number of points from Pts1 to Pts2.

The Pts1 and Pts2 inputs specify the location of points in two images. Each column in the Pts1 and Pts2 arrays has the format [row; column]. The points in the arrays must be ordered to form corresponding location pairs.

When the Find and exclude outliers option is selected, the block randomly selects matching point pairs until it computes a transformation matrix that fits a specified number of inliers, or until a user-specified stopping criterion is reached. It then refines this transformation matrix by using all inliers found in the first step.

Parameters

Transformation type: Projective

[✓] Find and exclude outliers

Method: Random Sample Consensus (RANSAC)

Algebraic distance threshold for determining inliers: 1.5

Determine number of random samplings using: Specified value

Number of random samplings: 100

[ ] Stop sampling earlier when a specified percentage of point pairs are determined to be inliers

[ ] Perform additional iterative refinement of the transformation matrix

Outputs

[ ] Output Boolean signal indicating which point pairs are inliers

When Pts1 and Pts2 are built-in integers, set transformation matrix data type to: Single

Transformation Type

Specify transformation type, either `Nonreflective similarity`, `affine`, or `projective` transformation. If you select `projective` transformation, you can also specify a scalar algebraic distance threshold for determining inliers. If you select either `affine` or `projective` transformation, you can specify the distance threshold for determining inliers in pixels. See "Transformations" on page 2-360 for a more detailed discussion.

Find and exclude outliers
When selected, the block finds and excludes outliers from the input points and uses only the inlier points to calculate the transformation matrix. When this option is not selected, all input points are used to calculate the transformation matrix.

Method
Select either the `RANdom SAmple Consensus (RANSAC)` or the `Least Median of Squares` algorithm to find outliers. See "RANSAC and Least Median Squares Algorithms" on page 2-359 for a more detailed discussion.

Algebraic distance threshold for determining inliers
Specify a scalar threshold value for determining inliers. The threshold controls the upper limit used to find the algebraic distance in the RANSAC algorithm. This parameter appears when the **Method** parameter is `Random Sample Consensus (RANSAC)` and the **Transformation type** parameter is `projective`.

Distance threshold for determining inliers (in pixels)
Specify the upper limit distance a point can differ from the projection location of its associating point. This parameter appears when the **Method** parameter is set to `Random Sample Consensus (RANSAC)` and the value of the **Transformation type** parameter is set to `Nonreflective similarity` or `affine`

Determine number of random samplings using
Select `Specified value` to enter a positive integer value for number of random samplings, or select `Desired confidence` to set the number of random samplings as a percentage and a maximum number. This parameter appears when you select `Find and exclude outliers parameter`, and the value of the **Method** parameter is `Random Sample Consensus (RANSAC)`.

Number of random samplings
: Specify the number of random samplings for the algorithm to perform. This parameter appears when the value of the **Determine number of random samplings using** parameter is `Specified value`.

Desired confidence (in %)
: Specify a percent by entering a number between `0` and `100`. The `Desired confidence` is the probability to find the largest group of points that can be mapped by a transformation matrix. This parameter is visible when the **Determine number of random samplings using** is `Desired confidence`.

Maximum number of random samplings
: Specify an integer number for the maximum number of random samplings. This parameter appears when the **Method** parameter is set to `Random Sample Consensus (RANSAC)` and the value of **Determine number of random samplings using** parameter is `Desired confidence`.

Stop sampling earlier when a specified percentage of point pairs are determined to be inlier
: Specify to stop random sampling when a percentage of input points have been found as inliers. This parameter appears when the **Method** parameter is `Random Sample Consensus (RANSAC)`.

Perform additional iterative refinement of the transformation matrix
: Specify whether to perform refinement on the transformation matrix. This parameter appears when you select **Find and exclude outliers** parameter.

Output Boolean signal indicating which point pairs are inliers
: Select this option to output the inlier point pairs that were used to calculate the transformation matrix. This parameter appears when you select **Find and exclude outliers** parameter. This parameter is not used when the data type of points is signed or double.

When Pts1 and Pts2 are built-in integers, set transformation matrix date type to

Specify transformation matrix data type as `Single` or `Double` when the input points are built-in integers. This parameter is not used when the data type of points is signed or double.

Parameter Name	Default Value	Visibility	Tunability
Transformation type	`projective`	Always	No
Find and exclude outliers	Checked	Always	No
Method	`RANSAC`	Visible when **Find and exclude outliers** parameter is selected	No
Algebraic distance threshold for determining inliers	`1.5`	Visible when **Method** parameter is `Random Sample Consensus (RANSAC)` and the **Transformatinon type** parameter is `projective`	Yes
Distance threshold for determining inliers (in pixels)	`1.5`	Visible when **Method** parameter is `Random Sample Consensus (RANSAC)` and the **Transformatinon type** parameter is `Nonreflective similarity` or `affine`	Yes
Determine number of random samplings	`Specified value`	Visible when `Find and exclude outliers parameter` is selected	No

Parameter Name	Default Value	Visibility	Tunability
Number of random samplings	`100`	Visible when **Determine number of random samplings using** parameter is `Specified value`	Yes
Maximum number of random samplings	`200`	Visible when the **Method** parameter is set to `Random Sample Consensus (RANSAC)` and **Determine number of random samplings using** parameter is `Desired confidence`	Yes
Desired confidence (in%)	`99`	Visible when the **Determine number of random samplings using** is `Desired confidence`	Yes
Stop sampling earlier when a specified percentage of point pairs are determined to be inliers	Unchecked	Visible when the **Method** parameter is `Random Sample Consensus (RANSAC)`	No
Inlier percentage	`75`	Visible when **Stop sampling earlier when a specified percentage of point pairs are determined to be inliers** parameter is checked	Yes

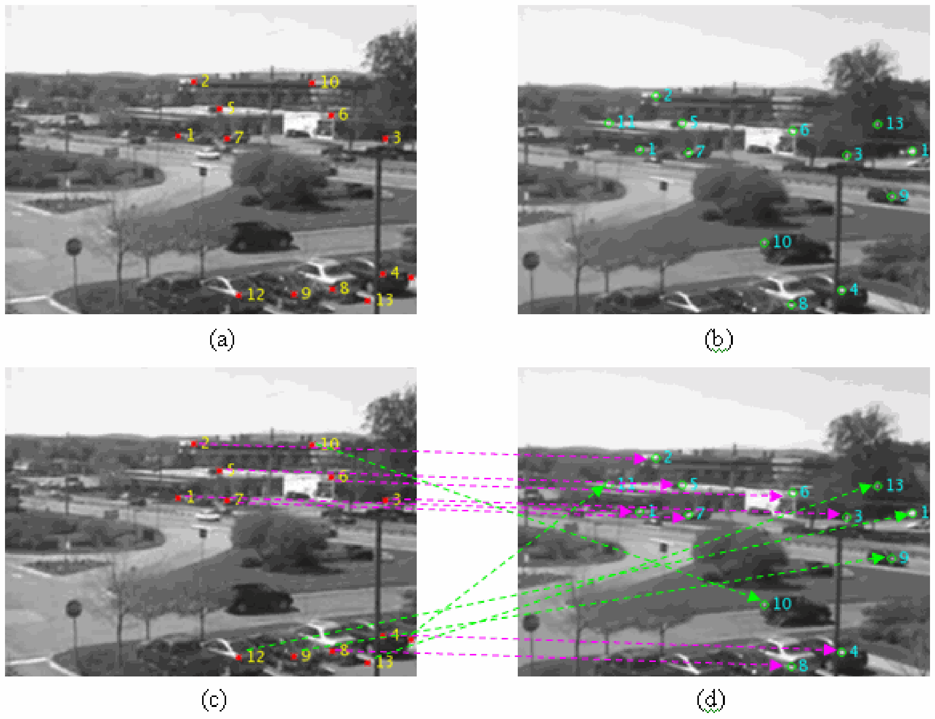
Parameter Name	Default Value	Visibility	Tunability
Perform additional iterative refinement of the transformation matrix	Unchecked	Visible when **Find and exclude outliers** parameter is selected	No
Output Boolean signal indicating which input points are inliers	Unchecked	Visible when **Find and exclude outliers** parameter is selected	No
When Pts1 and Pts2 are built-in integers, set transformation matrix data type to	`Single`	Always	No

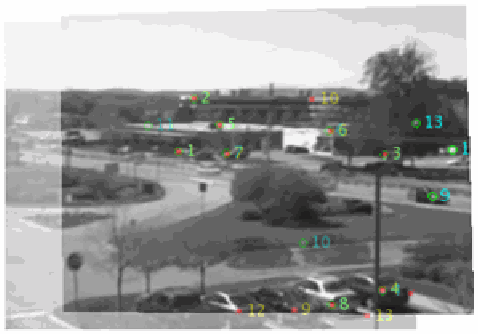
Examples

Calculate transformation matrix from largest group of point pairs

Examples of input data and application of the Estimate Geometric Transformation block appear in the following figures. Figures (a) and (b) show the point pairs. The points are denoted by stars or circles, and the numbers following them show how they are paired. Some point pairs can be mapped by the same transformation matrix. Other point pairs require a different transformation matrix. One matrix exists that maps the largest number of point pairs, the block calculates and returns this matrix. The block finds the point pairs in the largest group and uses them to calculate the transformation matrix. The point pairs connected by the magenta lines are the largest group.

The transformation matrix can then be used to stitch the images as shown in Figure (e).


(a) (b) (c) (d)


(e)

Video Mosaicking

To see an example of the Estimate Geometric Transformation block used in a model with other blocks, see the **Video Mosaicking Demo**, `vipmosaicking`.

References

R. Hartley and A. Ziserman, "Multiple View Geometry in Computer Vision," Second edition, Cambridge University Press, 2003

See Also

cp2tform	Image Processing Toolbox software
`vipmosaicking`	Video and Image Processing Blockset demo

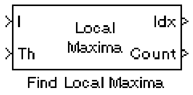
Purpose

Find local maxima in matrices

Library

Statistics

Description



The Find Local Maxima block finds the local maxima in the input matrix. It does this by comparing the maximum value in the matrix to a user-specified threshold. If the maximum value is greater than or equal to this threshold, the block considers the value a valid local maximum. Then, it sets all the matrix values in the neighborhood, an area around and including the maximum value, to 0. This step ensures that this maximum is not included in subsequent searches. The size of the neighborhood must be appropriate for the data set. That is, it must eliminate enough of the values around the maximum so that false peaks are not discovered. The block repeats this entire process until either it finds all the valid maxima or it finds the number of local maxima equal to the **Maximum number of local maxima (N)** parameter value, whichever comes first.

The block outputs the zero-based row and column coordinates of the maxima at the Idx port and the number of valid local maxima found at the Count port.

Port	Input/Output	Supported Data Types	Complex Values Supported
I/ Hough	Matrix in which you want to find the maxima	• 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

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.	• 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`.

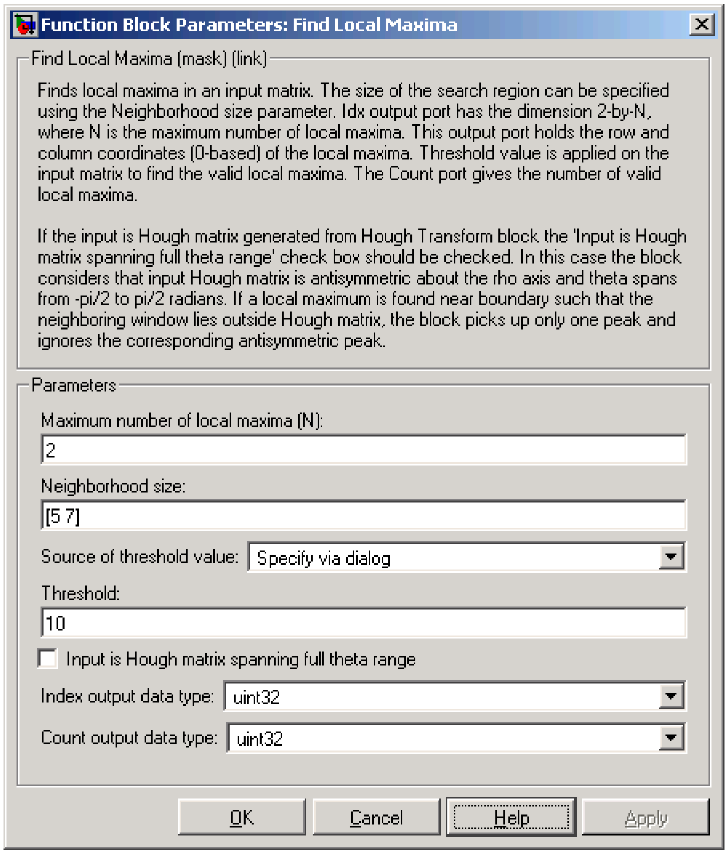
Use the **Count output data type** parameter to specify the data type of the Count port output. Your choices are `double`, `single`, `uint8`, `uint16`, or `uint32`.

Examples

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

Dialog Box

The Find Local Maxima dialog box appears as shown in the following figure.



Function Block Parameters: Find Local Maxima

Find Local Maxima (mask) (link)

Finds local maxima in an input matrix. The size of the search region can be specified using the Neighborhood size parameter. Idx output port has the dimension 2-by-N, where N is the maximum number of local maxima. This output port holds the row and column coordinates (0-based) of the local maxima. Threshold value is applied on the input matrix to find the valid local maxima. The Count port gives the number of valid local maxima.

If the input is Hough matrix generated from Hough Transform block the 'Input is Hough matrix spanning full theta range' check box should be checked. In this case the block considers that input Hough matrix is antisymmetric about the rho axis and theta spans from -pi/2 to pi/2 radians. If a local maximum is found near boundary such that the neighboring window lies outside Hough matrix, the block picks up only one peak and ignores the corresponding antisymmetric peak.

Parameters

Maximum number of local maxima (N): 2

Neighborhood size: [5 7]

Source of threshold value: Specify via dialog

Threshold: 10

☐ Input is Hough matrix spanning full theta range

Index output data type: uint32

Count output data type: uint32

Maximum number of local maxima (N)

Specify the maximum number of maxima you want the block to find.

Neighborhood size
Specify the size of the neighborhood around the maxima over which the block zeros out the values. Enter a two-element vector of positive odd integers, [r c].

Source of threshold value
Specify how to enter the threshold value. If you select `Input port`, the Th port appears on the block. If you select `Specify via dialog`, the **Threshold** parameter appears in the dialog box.

Threshold
Enter a scalar value that represents the value all maxima should meet or exceed. This parameter is visible if, for the **Source of threshold value** parameter, you choose `Specify via dialog`.

Input is Hough matrix spanning full theta range
If you select this check box, the block assumes that the Hough port input is antisymmetric about the rho axis and theta ranges from -pi/2 to pi/2 radians.

Index output data type
Specify the data type of the Peaks port output. Your choices are `double`, `single`, `uint8`, `uint16`, or `uint32`.

Count output data types
Specify the data type of the Count port output. Your choices are `double`, `single`, `uint8`, `uint16`, or `uint32`.

See Also

Hough Lines	Video and Image Processing Blockset software
Hough Transform	Video and Image Processing Blockset software
`houghpeaks`	Image Processing Toolbox sofware

Frame Rate Display

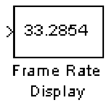
Purpose

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	• 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

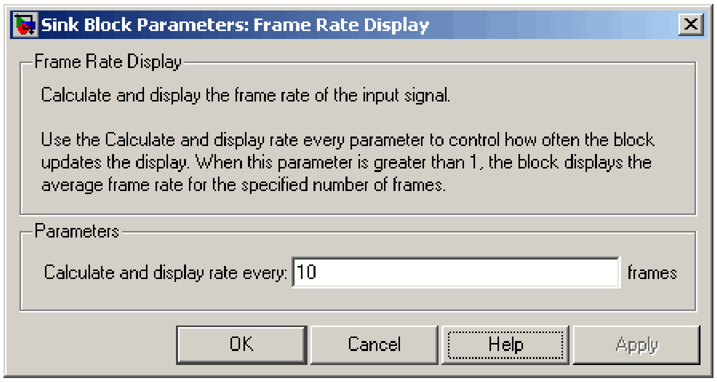
Use the **Calculate and display rate every** parameter to control how often the block updates the display. When this parameter is greater than 1, the block displays the average update rate for the specified number of video frames. For example, if you enter 10, the block calculates the amount of time 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.

Frame Rate Display

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.



Sink Block Parameters: Frame Rate Display

Frame Rate Display

Calculate and display the frame rate of the input signal.

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 frame rate for the specified number of frames.

Parameters

Calculate and display rate every: 10 frames

Calculate and display rate every

Use this parameter to control how often the block updates the display.

See Also

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

Purpose Read video frames and audio samples from compressed multimedia file

Library Sources

Description The From Multimedia File block is a Signal Processing Blockset block. For more information, see the From Multimedia File block reference page in the Signal Processing Blockset software documentation.

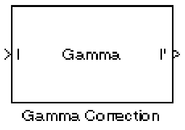
Purpose

Apply or remove gamma correction from images or video streams

Library

Conversions

Description



I'
De-gamma
I
Gamma Correction1

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:

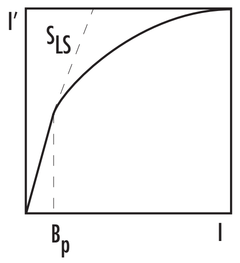
$$S_{LS} = \frac{1}{\dfrac{\gamma}{B_P^{(1/\gamma - 1)}} - \gamma B_P + B_P}$$

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

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

$$I' = \begin{Bmatrix} S_{LS} I, & I \le B_P \\ F_S I^{1/\gamma} - C_O, & I > B_P \end{Bmatrix}$$

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.



For normalized input signals, the block removes gamma correction, which linearizes the input video stream, as defined by the following equation:

$$I = \begin{cases} I'/S_{LS}, & I' \le B_P \\ \left(\dfrac{I' + C_O}{F_S}\right)^{\gamma}, & I' > B_P \end{cases}$$

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	• 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

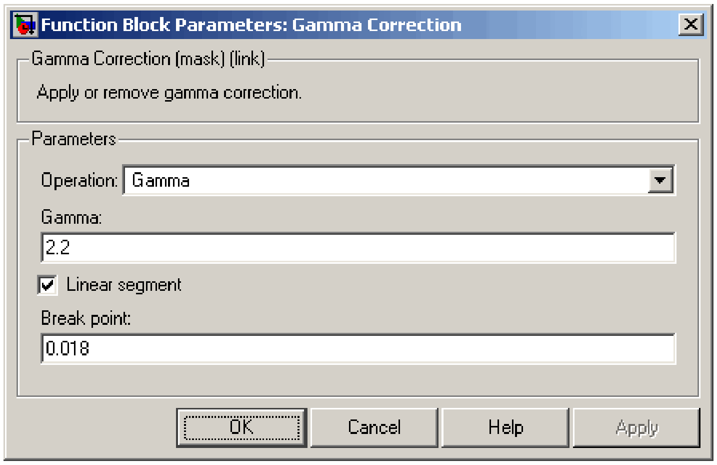
Use the **Operation** parameter to specify the block's operation. If you want to perform gamma correction, select `Gamma`. If you want to linearize the input signal, select `De-gamma`.

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 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.



Function Block Parameters: Gamma Correction

Gamma Correction (mask) (link)

Apply or remove gamma correction.

Parameters

Operation: Gamma

Gamma:

2.2

☑ Linear segment

Break point:

0.018

Operation

Specify the block's operation. Your choices are `Gamma` or `De-gamma`.

Gamma

If, for the **Operation** parameter, you select `Gamma`, enter the desired gamma value of the output video stream. This value must be greater than or equal to 1. If, for the **Operation** parameter, you select `De-gamma`, enter the gamma value of the input video stream.

Linear segment

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

Break point

Enter a scalar value that indicates the *I*-axis value of the end of the linear segment. This parameter is visible if you select the **Linear segment** check box.

References

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

See Also

Color Space Conversion	Video and Image Processing Blockset software
`imadjust`	Image Processing Toolbox software

Purpose

Perform Gaussian pyramid decomposition

Library

Transforms

Description

Gaussian Pyramid

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 2-389.

Note This block supports intensity and color images on its ports.

Port	Output	Supported Data Types	Complex Values Supported
Input	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. 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.	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer	No
Output	In `Reduce` mode, the output can be a scalar, vector, or matrix that	Same as Input port	No

Port	Output	Supported Data Types	Complex Values Supported
	represents one level of a Gaussian pyramid. In `Expand` mode, the output can be a matrix that represents one level of a Gaussian pyramid.		

Use the **Operation** parameter to specify whether to reduce or expand the input image. If you select `Reduce`, the block applies a lowpass filter and then downsamples the input image. If you select `Expand`, the block upsamples and then applies a lowpass filter to the input image.

Use the **Pyramid level** parameter to specify the number of times the block upsamples or downsamples each dimension of the image by a factor of 2. For example, suppose you have a 4-by-4 input image. You set the **Operation** parameter to `Reduce` and the **Pyramid level** to `1`. The block filters and downsamples the image and outputs a 2-by-2 pixel output image. If you have an M-by-N input image and you set the **Operation** parameter to `Reduce`, you can calculate the dimensions of the output image using the following equation:

$$\text{ceil}\left(\frac{M}{2}\right) - \text{by} - \text{ceil}\left(\frac{N}{2}\right)$$

You must repeat this calculation for each successive pyramid level. If you have an M-by-N input image and you set the **Operation** parameter to `Expand`, you can calculate the dimensions of the output image using the following equation:

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

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.

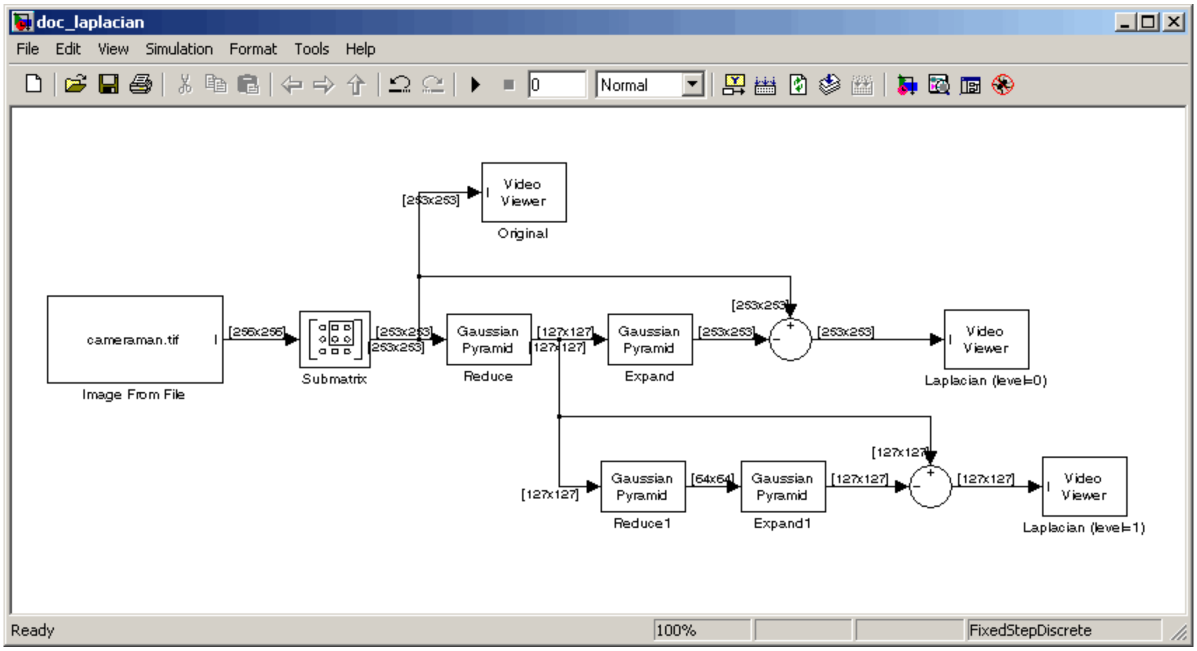
Examples

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

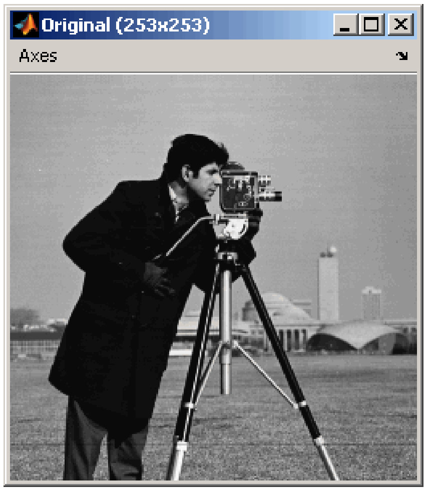
1. Open this model by typing

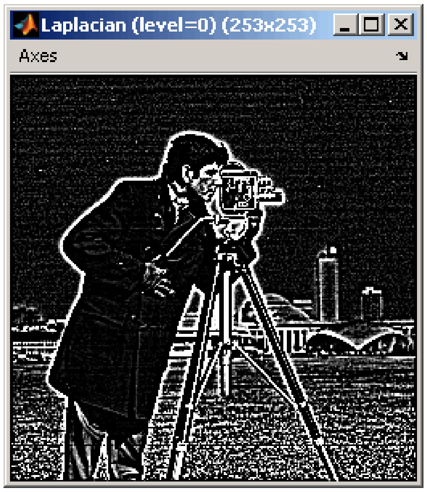
   ```
   doc_laplacian
   ```

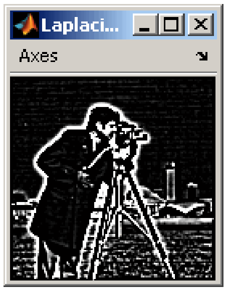
   at the MATLAB command prompt.



2 Run the model to see the following results.



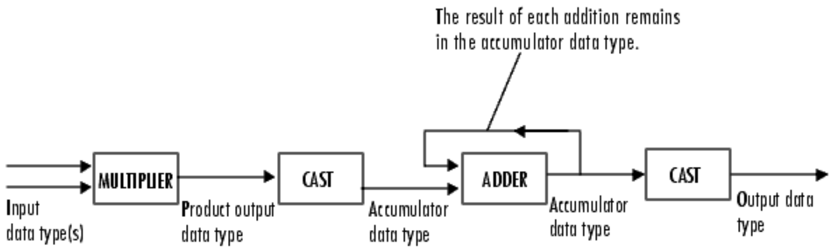




You can construct a Laplacian pyramid if the dimensions of the input image, R-by-C, satisfy $R = M_R 2^N + 1$ and $C = M_c 2^N + 1$, where M_R, M_C, and N are integers. In this example, you have an input matrix that is 256-by-256. If you set M_R and M_C equal to 63 and N equal to 2, you find that the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

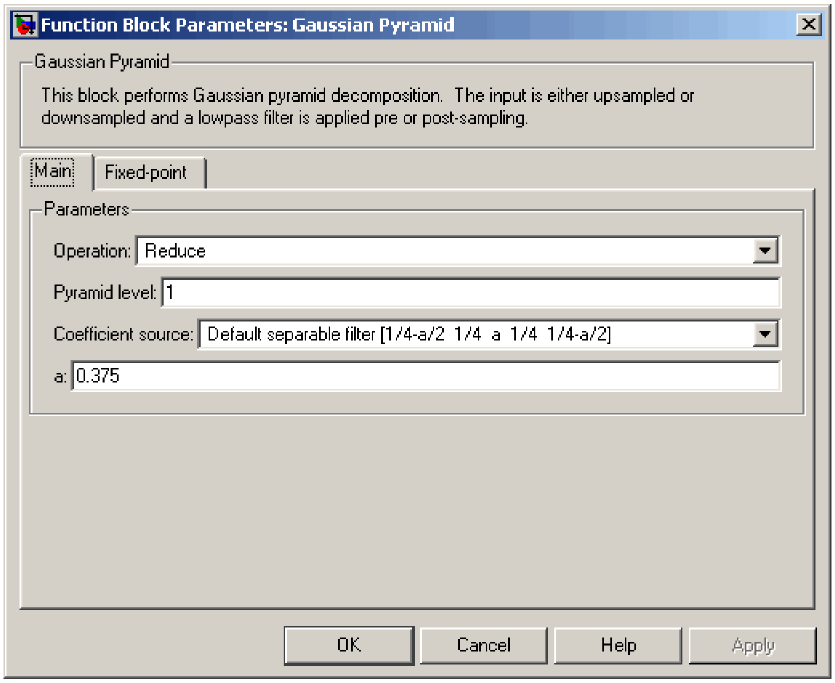
The following diagram shows the data types used in the Gaussian Pyramid block for fixed-point signals:



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

Dialog Box

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


Function Block Parameters: Gaussian Pyramid

Gaussian Pyramid

This block performs Gaussian pyramid decomposition. The input is either upsampled or downsampled and a lowpass filter is applied pre or post-sampling.

Main | Fixed-point

Parameters

Operation: Reduce

Pyramid level: 1

Coefficient source: Default separable filter [1/4-a/2 1/4 a 1/4 1/4-a/2]

a: 0.375

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`.

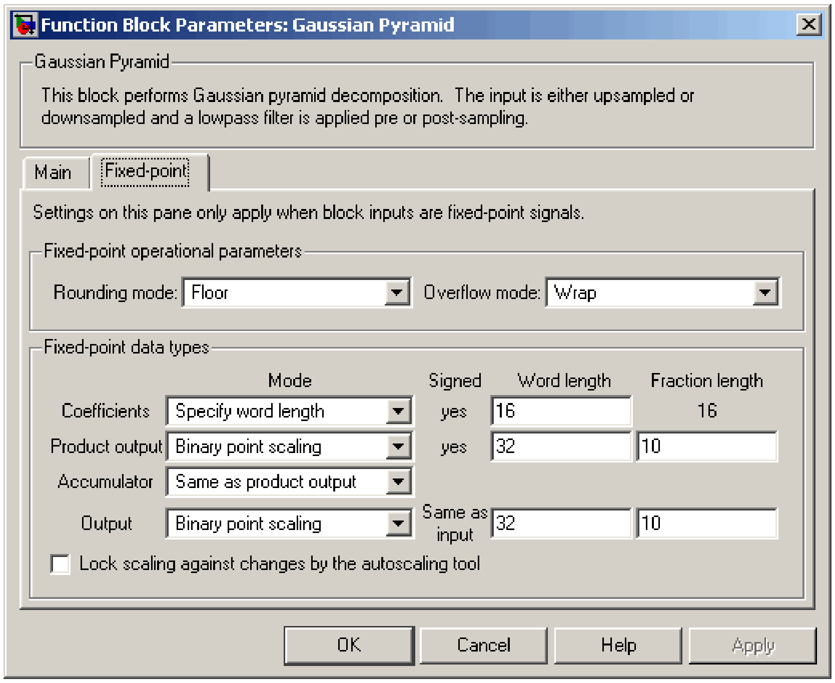
a

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.



Function Block Parameters: Gaussian Pyramid

Gaussian Pyramid

This block performs Gaussian pyramid decomposition. The input is either upsampled or downsampled and a lowpass filter is applied pre or post-sampling.

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

	Mode	Signed	Word length	Fraction length
Coefficients	Specify word length	yes	16	16
Product output	Binary point scaling	yes	32	10
Accumulator	Same as product output			
Output	Binary point scaling	Same as input	32	10

☐ Lock scaling against changes by the autoscaling tool

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

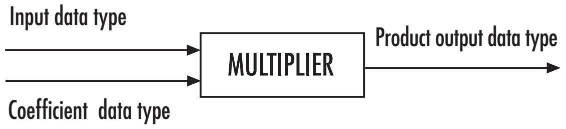
Select the overflow mode for fixed-point operations.

Coefficients

Choose how to specify the word length and the fraction length of the coefficients:

- When you select `Same word length as input`, the word length of the coefficients match that of the input to the block. In this mode, the fraction length of the coefficients is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the coefficients.
- When you select `Specify word length`, you can enter the word length of the coefficients, in bits. The block automatically sets the fraction length to give you the best precision.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the coefficients, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the coefficients. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

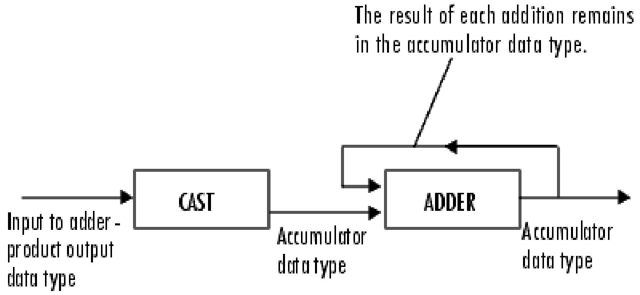
Product output



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

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

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type

As shown in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate the accumulator word and fraction lengths.

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

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

Output

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

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

Lock 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 software

Purpose Generate histogram of each input matrix

Library Statistics

Description The Histogram block is a Signal Processing Blockset block. For more information, see the Histogram block reference page in the Signal Processing Blockset software documentation.

Purpose Enhance contrast of images using histogram equalization

Library Analysis & Enhancement

Description The Histogram Equalization block enhances the contrast of images by transforming the values in an intensity image so that the histogram of the output image approximately matches a specified histogram.

I
Hist
Histogram
Equalization

Histogram Equalization

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Matrix of intensity values	• 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	• 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.

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.

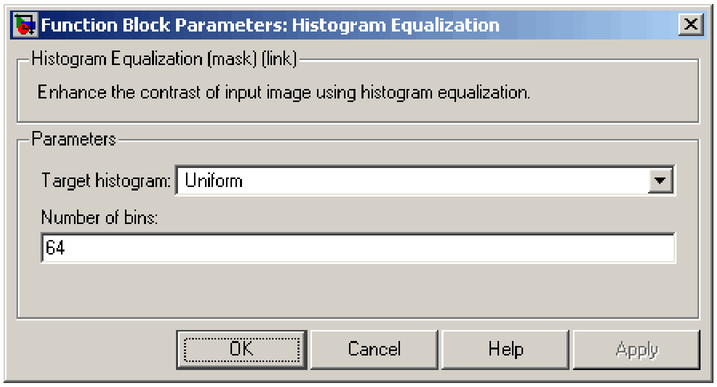
Note The vector input to the Hist port must be normalized such that the sum of the values in all the bins is equal to the number of pixels in the input image. The block does not error if the histogram is not normalized.

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.


Function Block Parameters: Histogram Equalization

Histogram Equalization (mask) (link)

Enhance the contrast of input image using histogram equalization.

Parameters

Target histogram: Uniform

Number of bins:

64

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`.

See Also

`imadjust`	Image Processing Toolbox
`histeq`	Image Processing Toolbox

Hough Lines

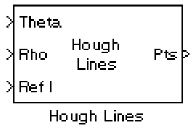
Purpose

Find Cartesian coordinates of lines described by rho and theta pairs

Library

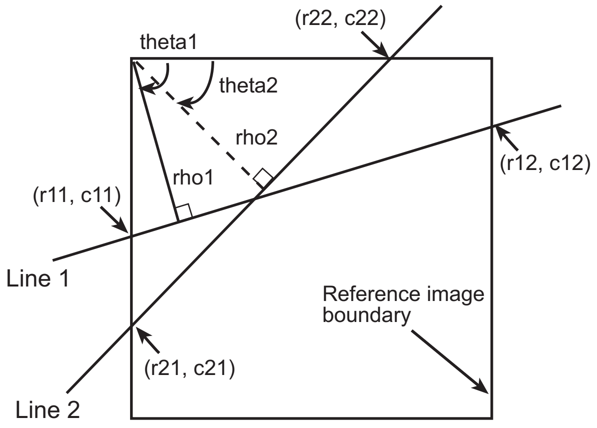
Transforms

Description

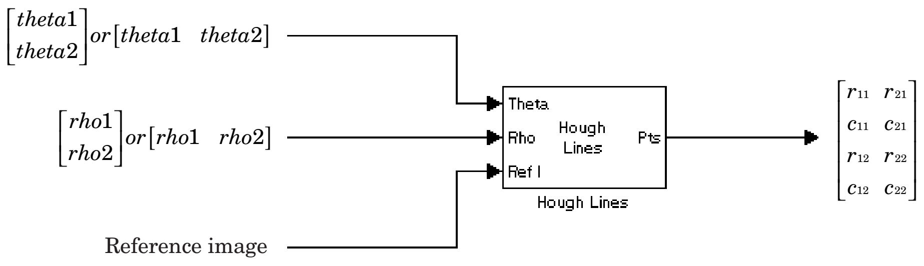


The Hough Lines block inputs are the theta and rho values of lines and a reference image. The block outputs the zero-based row and column positions of the intersections between the lines and two of the reference image boundary lines. The boundary lines are the left and right vertical boundaries and the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



$\begin{bmatrix} theta1 \\ theta2 \end{bmatrix} or [theta1 \quad theta2]$
$\begin{bmatrix} rho1 \\ rho2 \end{bmatrix} or [rho1 \quad rho2]$
Reference image
Theta
Rho
Ref I
Hough Lines
Pts
Hough Lines
$\begin{bmatrix} r_{11} & r_{21} \\ c_{11} & c_{21} \\ r_{12} & r_{22} \\ c_{12} & c_{22} \end{bmatrix}$

Port	Input/Output	Supported Data Types	Complex Values Supported
Theta	Vector of theta values that represent input line(s)	• 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
Ref I	Matrix that represents a binary or intensity image or matrix that represents one plane of an RGB image	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • Custom data types • Boolean • 8-, 16-, and 32-bit signed integer	No

Port	Input/Output	Supported Data Types	Complex Values Supported
		• 8-, 16-, and 32-bit unsigned integer	
Pts	4-by-N matrix of intersection values, where N is the number of input lines	• 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`.

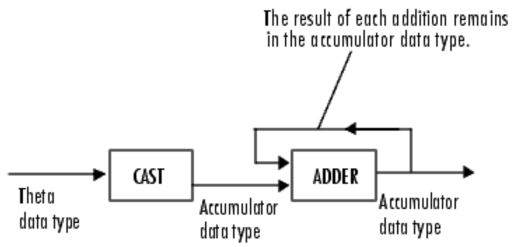
If, for the **Sine value computation method** parameter, you select `Table lookup`, the **Theta resolution (radians)** parameter appears in the dialog box. Use this parameter to specify the spacing of the theta-axis.

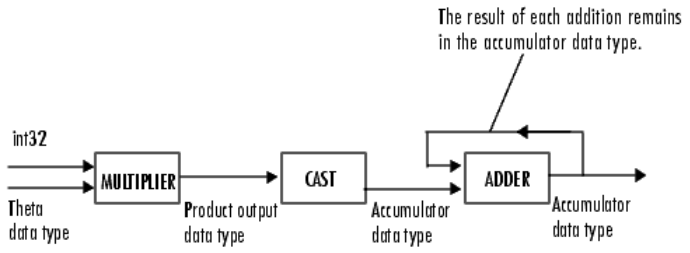
Examples

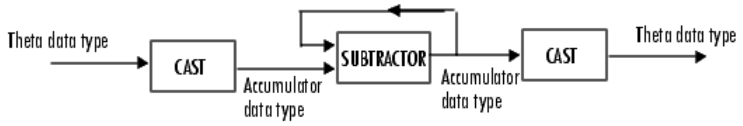
See "Finding Lines in Images" and "Measuring an Angle Between Lines" in the *Video and Image Processing Blockset User's Guide*.

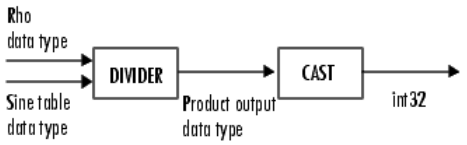
Fixed-Point Data Types

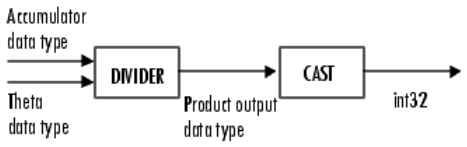
The following diagram shows the data types used in the Hough Lines block for fixed-point signals.







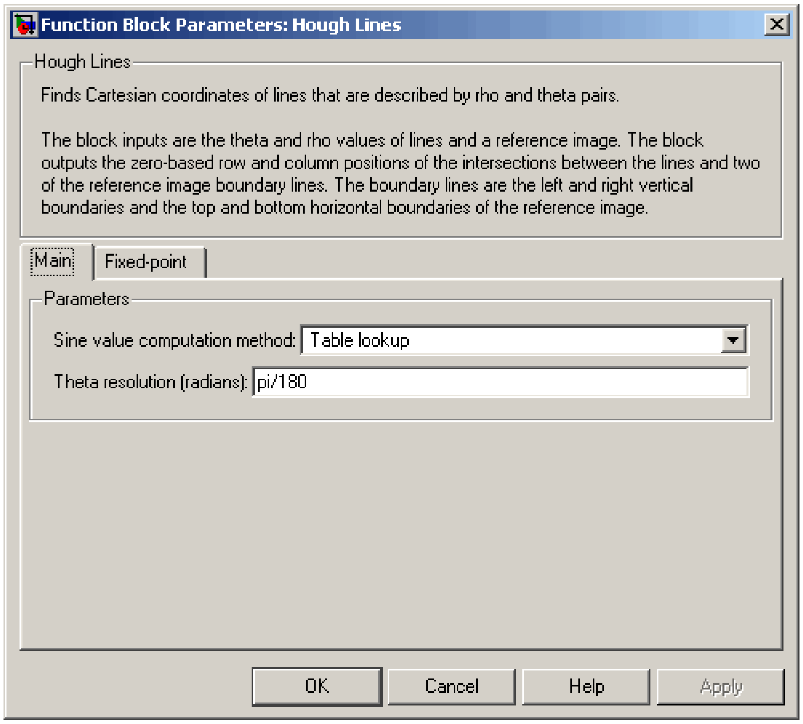




You can set the sine table, product output, and accumulator output data types in the block mask as discussed in the next section.

Dialog Box

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



Finds Cartesian coordinates of lines that are described by rho and theta pairs.

The 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.

Sine value computation method: Table lookup

Theta resolution (radians): pi/180

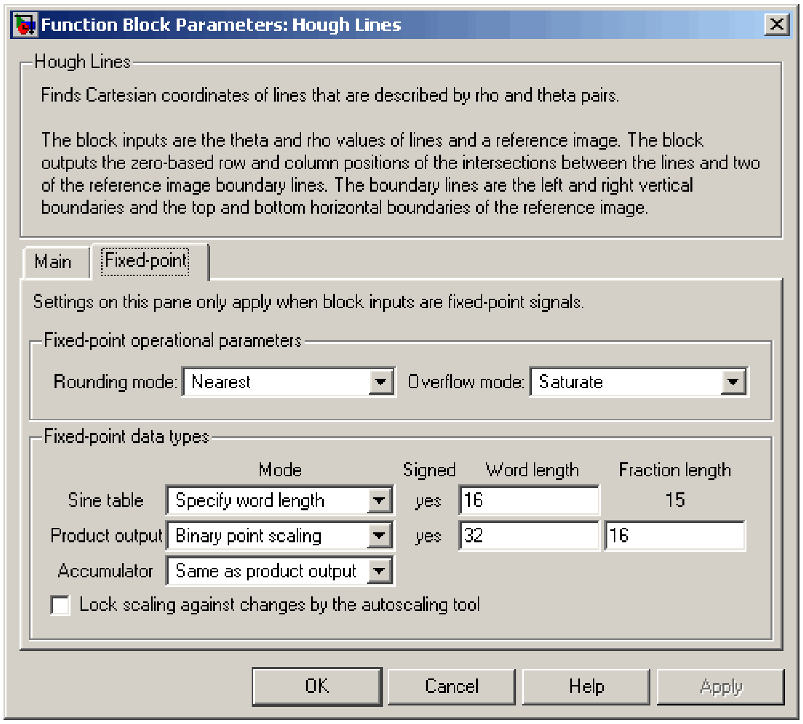
Sine value computation method

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.



Function Block Parameters: Hough Lines

Hough Lines

Finds Cartesian coordinates of lines that are described by rho and theta pairs.

The 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.

Main | Fixed-point

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

Fixed-point operational parameters

Rounding mode: Nearest   Overflow mode: Saturate

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Sine table	Specify word length	yes	16	15
Product output	Binary point scaling	yes	32	16
Accumulator	Same as product output			

☐ Lock scaling against changes by the autoscaling tool

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

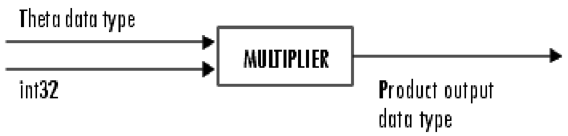
Sine table

Choose how to specify the word length of the values of the sine table. The fraction length of the sine table values 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



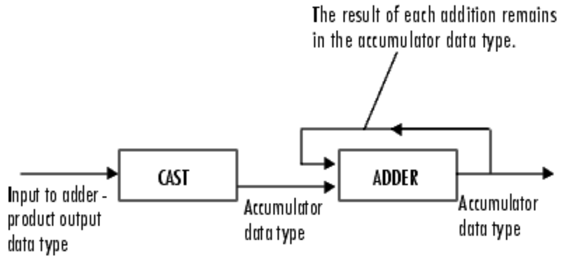
As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

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 the Video and Image Processing Blockset blocks is 0.

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type

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 the Video and Image Processing Blockset blocks is 0.

See Also

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

2-D IFFT	Video and Image Processing Blockset software
Find Local Maxima	Video and Image Processing Blockset software
Hough Transform	Video and Image Processing Blockset software

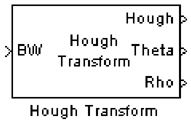
Purpose

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 *rho* and *theta* values, respectively. Peak values in this matrix represent potential lines in the input image. The *theta* 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	• 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	• 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.

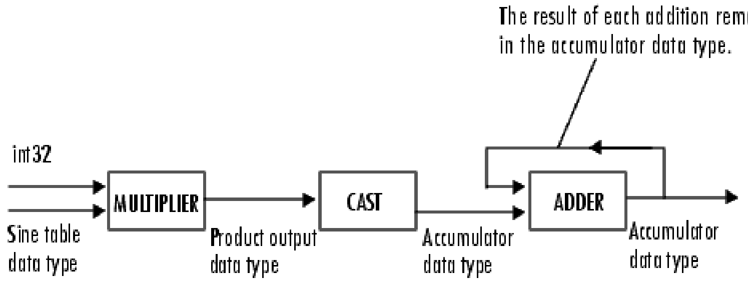
Use the **Output data type** parameter to specify the data type of your output signal.

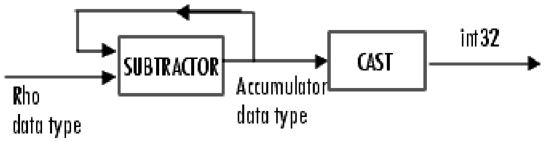
Examples

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.

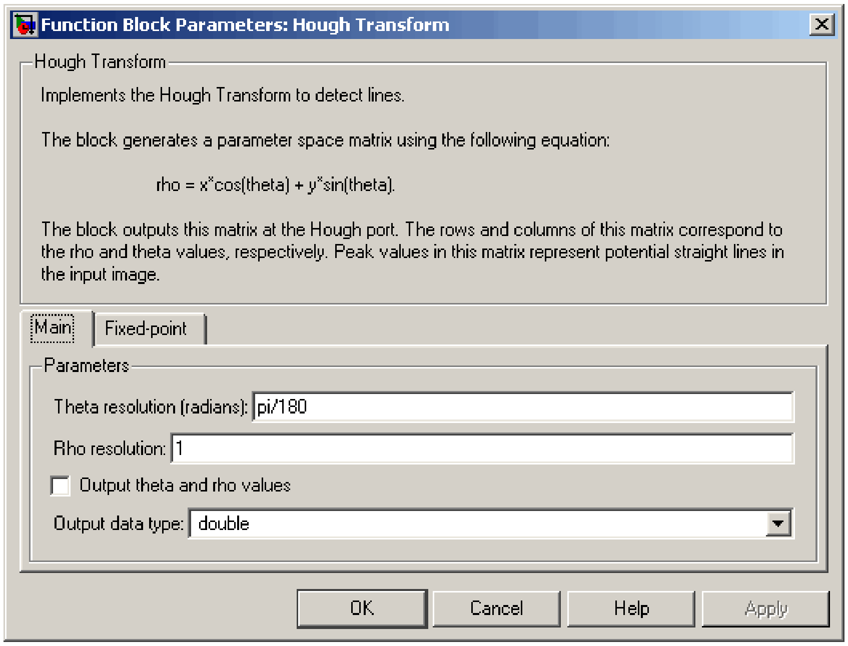




You can set the sine table, rho, product output, accumulator, Hough output, and Theta output data types in the block mask as discussed in the next section.

Dialog Box

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


Function Block Parameters: Hough Transform

Hough Transform

Implements the Hough Transform to detect lines.

The block generates a parameter space matrix using the following equation:

rho = x*cos(theta) + y*sin(theta).

The block outputs this matrix at the Hough port. The rows and columns of this matrix correspond to the rho and theta values, respectively. Peak values in this matrix represent potential straight lines in the input image.

Main | Fixed-point

Parameters

Theta resolution (radians): pi/180

Rho resolution: 1

☐ Output theta and rho values

Output data type: double

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.

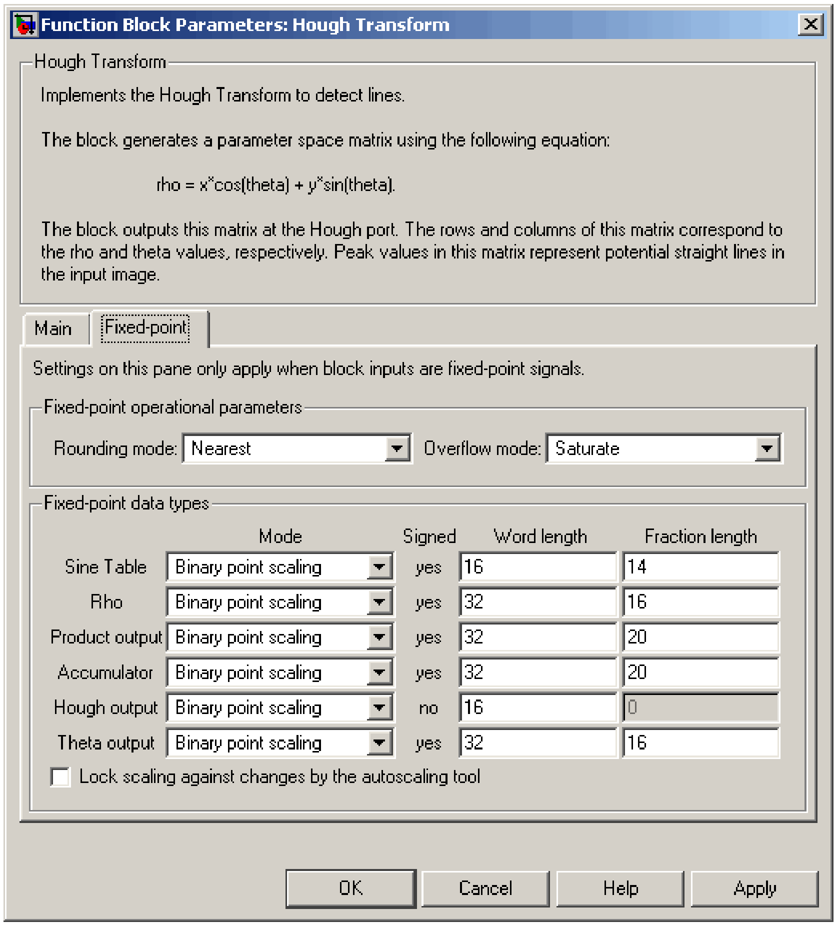
Output theta and rho values

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.


Function Block Parameters: Hough Transform

Hough Transform

Implements the Hough Transform to detect lines.

The block generates a parameter space matrix using the following equation:

rho = x*cos(theta) + y*sin(theta).

The block outputs this matrix at the Hough port. The rows and columns of this matrix correspond to the rho and theta values, respectively. Peak values in this matrix represent potential straight lines in the input image.

Main | Fixed-point

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

Fixed-point operational parameters

Rounding mode: Nearest   Overflow mode: Saturate

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Sine Table	Binary point scaling	yes	16	14
Rho	Binary point scaling	yes	32	16
Product output	Binary point scaling	yes	32	20
Accumulator	Binary point scaling	yes	32	20
Hough output	Binary point scaling	no	16	0
Theta output	Binary point scaling	yes	32	16

☐ Lock scaling against changes by the autoscaling tool

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.

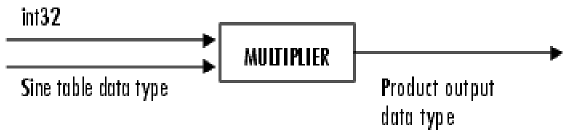
The sine table values do not obey the **Rounding mode** and **Overflow mode** parameters; they are always saturated and rounded to `Nearest`.

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 blocks is 0.

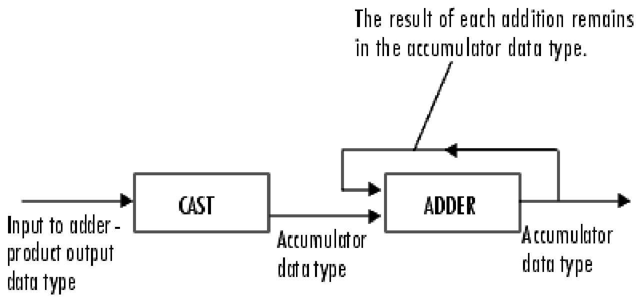
Product output

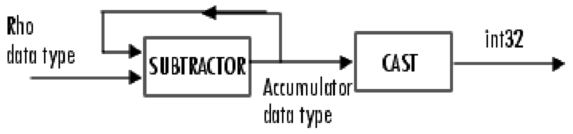


As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

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

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type



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

See Also

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

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

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.

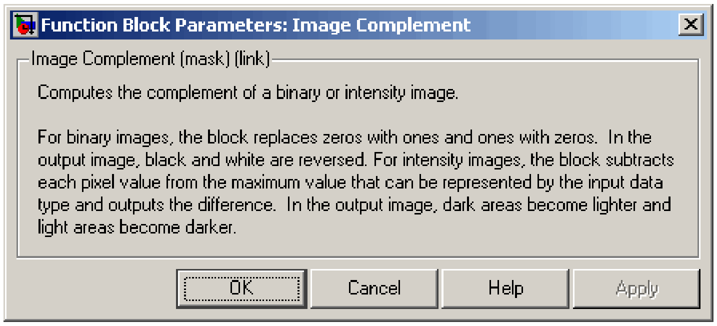
For example, suppose the input pixel values are given by *x*(i) and the output pixel values are given by *y*(i). If the data type of the input is double or single precision floating-point, the block outputs *y*(i) = 1.0-*x*(i). If the input is an 8-bit unsigned integer, the block outputs *y*(i) = 255-*x*(i).

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Vector or matrix of intensity values	• 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.


Function Block Parameters: Image Complement
Image Complement (mask) (link)
OK
Cancel
Help
Apply

Computes the complement of a binary or intensity image.

For binary images, the block replaces zeros with ones and ones with zeros. In the output image, black and white are reversed. For intensity images, the block subtracts each pixel value from the maximum value that can be represented by the input data type and outputs the difference. In the output image, dark areas become lighter and light areas become darker.

See Also

Autothreshold	Video and Image Processing Blockset software
Chroma Resampling	Video and Image Processing Blockset software
Color Space Conversion	Video and Image Processing Blockset software
`imcomplement`	Image Processing Toolbox software

Image Data Type Conversion

Purpose

Convert and scale input image to specified output data type

Library

Conversions

Description

The Image Data Type Conversion block changes the data type of the input to the user-specified data type and scales the values to the new data type's dynamic range. To convert between data types without scaling, use the Simulink Data Type Conversion block.

When converting between floating-point data types, the block casts the input into the output data type and clips values outside the range to 0 or 1. When converting to the Boolean data type, the block maps 0 values to 0 and all other values to one. When converting to or between all other data types, the block casts the input into the output data type and scales the data type values into the dynamic range of the output data type. For double- and single-precision floating-point data types, the dynamic range is between 0 and 1. For fixed-point data types, the dynamic range is between the minimum and maximum values that can be represented by the data type.

Note This block supports intensity and color images on its ports.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	• Double-precision floating point • Single-precision floating point • Fixed point (word length less than or equal to 16) • Boolean • 8-, 16-bit signed integer	No

Port	Input/Output	Supported Data Types	Complex Values Supported
		• 8-, 16-bit unsigned integer	
Output	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	Same as Input port	No

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

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

Dialog Box

The Image Data Type Conversion dialog box appears as shown in the following figure.

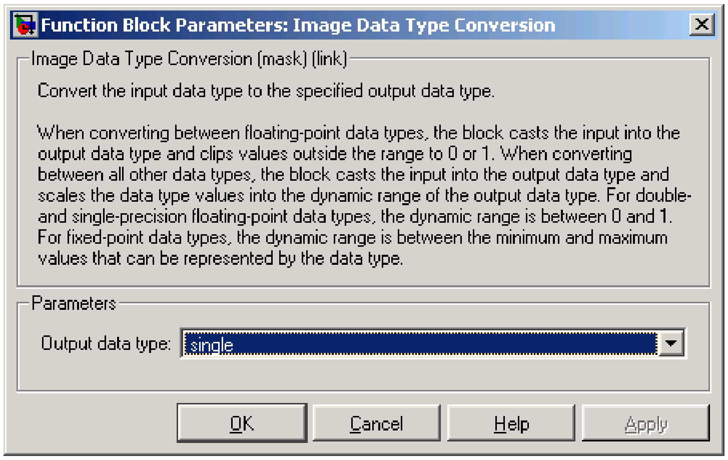

Function Block Parameters: Image Data Type Conversion

Image Data Type Conversion (mask) (link)

Convert the input data type to the specified output data type.

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 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.

Parameters

Output data type: single

Output data type

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

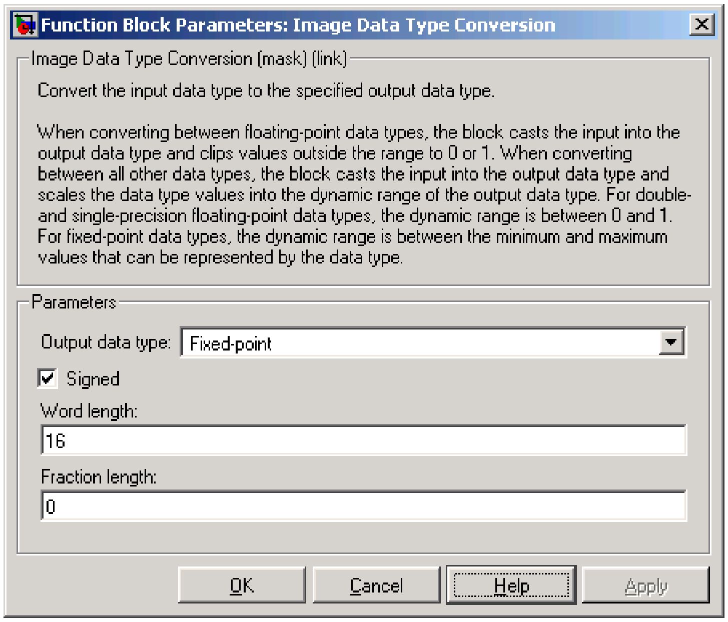


Image Data Type Conversion (mask) (link)

Convert the input data type to the specified output data type.

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 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.

Parameters

Output data type: Fixed-point

[✓] Signed

Word length: 16

Fraction length: 0

Signed

Select this check box if you want the output fixed-point data to be signed. This parameter is visible if, for the **Output data type** parameter, you choose `Fixed-point`.

Word length

Use this parameter to specify the word length of your fixed-point output. This parameter is visible if, for the **Output data type** parameter, you choose `Fixed-point`.

Fraction length

Use this parameter to specify the fraction length of your fixed-point output. This parameter is visible if, for the **Output data type** parameter, you choose `Fixed-point`.

See Also

Autothreshold	Video and Image Processing Blockset software

Image From File

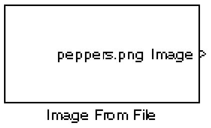
Purpose

Import image from image file

Library

Sources

Description



Use the Image From File block to import an image from a supported image file. For a list of supported file formats, see the `imread` function reference page in the MATLAB documentation. If the image is a M-by-N array, the block outputs a binary or intensity image, where M and N are the number of rows and columns in the image. If the image is a M-by-N-by-P array, the block outputs a color image, where M and N are the number of rows and columns in each color plane, P.

Port	Output	Supported Data Types	Complex Values Supported
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer	Yes
R, G, B	Scalar, vector, or matrix that represents one plane of the input RGB video stream. Outputs from the R, G, or B ports have the same dimensions.	Same as I port	Yes

For the Video and Image Processing Blockset blocks to display video data properly, double- and single-precision floating-point pixel values must be between 0 and 1. If the input pixel values have a different data type than the one you select using the **Output data type** parameter, the block scales the pixel values, adds an offset to the pixel values so that they are within the dynamic range of their new data type, or both.

Use the **File name** parameter to specify the name of the graphics file that contains the image to import into the Simulink modeling and simulation software. If the file is not on the MATLAB path, use the **Browse** button to locate the file. This parameter supports URL paths.

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

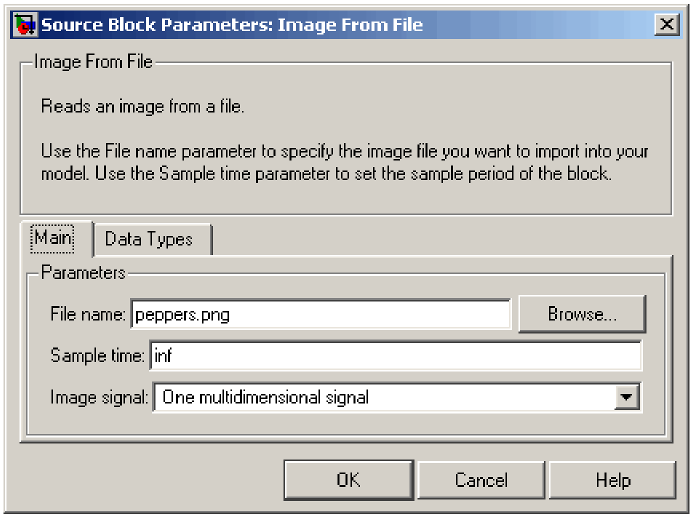
Use the **Image signal** parameter to specify how the block outputs a color video signal. If you select `One multidimensional signal`, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select `Separate color signals`, additional ports appear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

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

On the **Data Types** pane, use the **Output data type** parameter to specify the data type of your output signal.

Dialog Box

The **Main** pane of the Image From File dialog box appears as shown in the following figure.



Source Block Parameters: Image From File

Image From File

Reads an image from a file.

Use the File name parameter to specify the image file you want to import into your model. Use the Sample time parameter to set the sample period of the block.

File name: peppers.png

Sample time: inf

Image signal: One multidimensional signal

File name
: Specify the name of the graphics file that contains the image to import into the Simulink environment.

Sample time
: Enter the sample period of the output signal.

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

Output port labels

Enter the labels for your output ports using the spacer character, `|`, as the delimiter. This parameter is visible if you set the **Image signal** parameter to `Separate color signals`.

The **Data Types** pane of the Image From File dialog box appears as shown in the following figure.

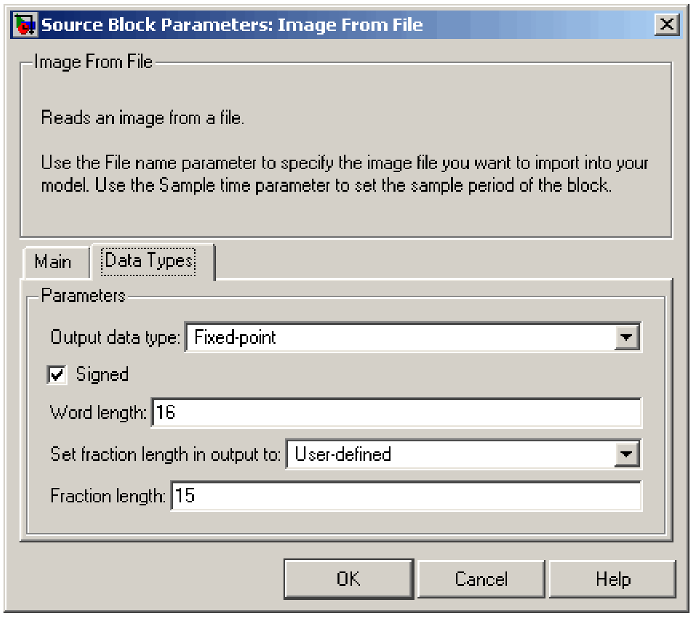

Source Block Parameters: Image From File

Image From File

Reads an image from a file.

Use the File name parameter to specify the image file you want to import into your model. Use the Sample time parameter to set the sample period of the block.

Main | Data Types

Parameters

Output data type: Fixed-point

☑ Signed

Word length: 16

Set fraction length in output to: User-defined

Fraction length: 15

Output data type

Specify the data type of your output signal.

Signed

Select to output a signed fixed-point signal. Otherwise, the signal will be unsigned. This parameter is only visible if, from the **Output data type** list, you select `Fixed-point`.

Word length

Specify the word length, in bits, of the fixed-point output data type. This parameter is only visible if, from the **Output data type** list, you select `Fixed-point`.

Set fraction length in output to

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

- Choose `Best precision` to have the output scaling automatically set such that the output signal has the best possible precision.
- Choose `User-defined` to specify the output scaling in the **Fraction length** parameter.

This parameter is only visible if, from the **Output data type** list, you select `Fixed-point` or when you select `User-defined`.

Fraction length

For fixed-point output data types, specify the number of fractional bits, or bits to the right of the binary point. This parameter is only visible when you select `Fixed-point` or `User-defined` for the **Output data type** parameter and `User-defined` for the **Set fraction length in output to** parameter.

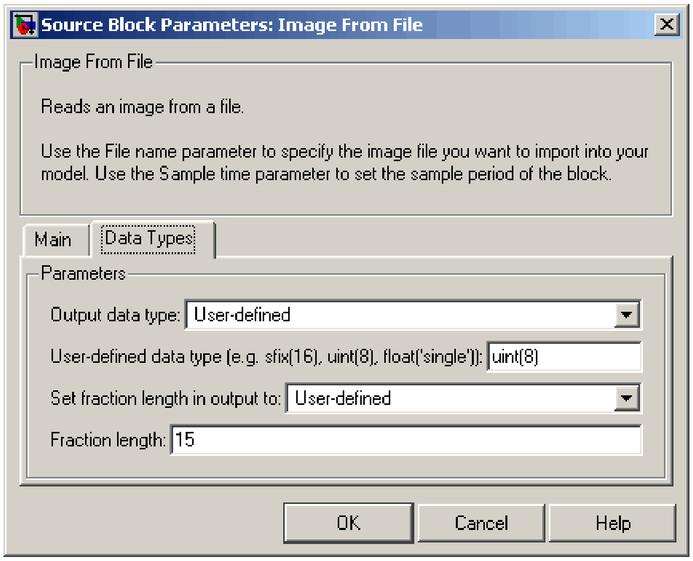

Source Block Parameters: Image From File

Image From File

Reads an image from a file.

Use the File name parameter to specify the image file you want to import into your model. Use the Sample time parameter to set the sample period of the block.

Main | Data Types

Parameters

Output data type: User-defined

User-defined data type (e.g. sfix(16), uint(8), float('single')): uint(8)

Set fraction length in output to: User-defined

Fraction length: 15

User-defined data type
Specify any built-in or fixed-point data type. You can specify fixed-point data types using the `sfix`, `ufix`, `sint`, `uint`, `sfrac`, and `ufrac` functions from the Simulink Fixed Point library. This parameter is only visible when you select `User-defined` for the **Output data type** parameter.

See Also

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

`im2double`	Image Processing Toolbox software
`im2uint8`	Image Processing Toolbox software
`imread`	MATLAB

Image From Workspace

Purpose

Import image from MATLAB workspace

Library

Sources

Description

checker_board(10) Image
Image From Workspace

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	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer	No
R, G, B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B ports have the same dimensions.	Same as I port	No

For the Video and Image Processing Blockset blocks to display video data properly, double- and single-precision floating-point pixel values must be between 0 and 1. If the input pixel values have a different data type than the one you select using the **Output data type** parameter, the block scales the pixel values, adds an offset to the pixel values so that they are within the dynamic range of their new data type, or both.

Use the **Value** parameter to specify the MATLAB workspace variable that contains the image you want to import into Simulink environment.

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

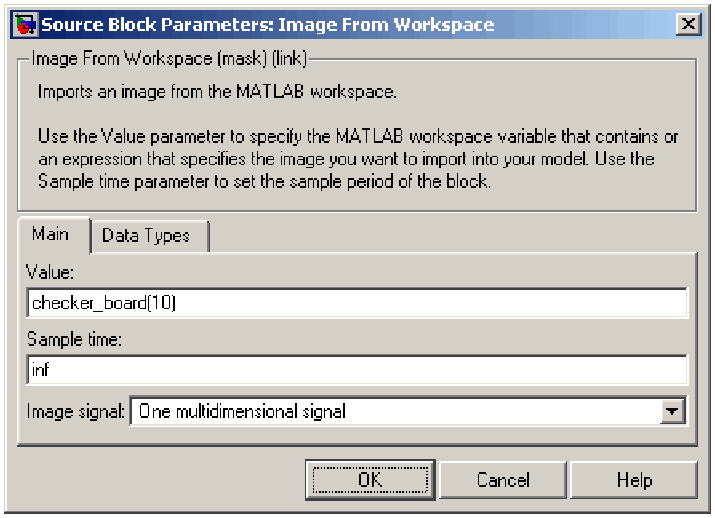
Use the **Image signal** parameter to specify how the block outputs a color video signal. If you select `One multidimensional signal`, the block outputs an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select `Separate color signals`, additional ports appear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

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

On the **Data Types** pane, use the **Output data type** parameter to specify the data type of your output signal.

Dialog Box

The **Main** pane of the Image From Workspace dialog box appears as shown in the following figure.



Source Block Parameters: Image From Workspace

Image From Workspace (mask) (link)

Imports an image from the MATLAB workspace.

Use the Value parameter to specify the MATLAB workspace variable that contains or an expression that specifies the image you want to import into your model. Use the Sample time parameter to set the sample period of the block.

Value: checker_board(10)

Sample time: inf

Image signal: One multidimensional signal

Value

Specify the MATLAB workspace variable that you want to import into Simulink environment.

Sample time

Enter the sample period of the output signal.

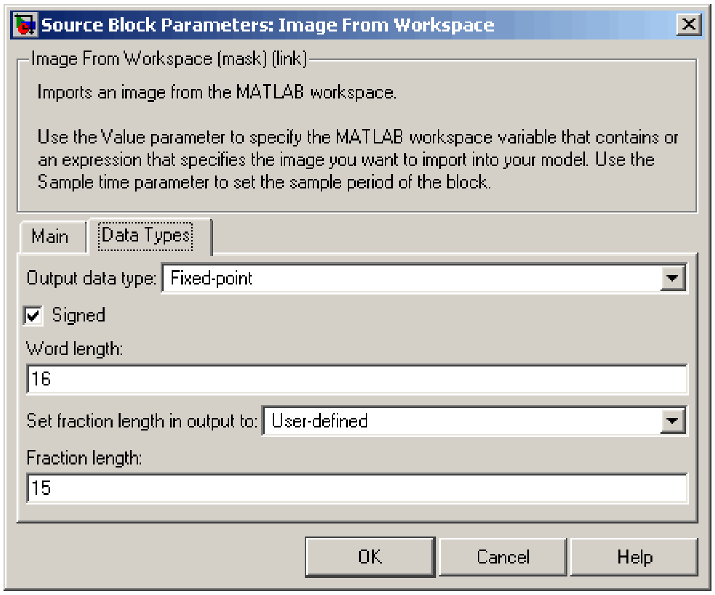
Image signal

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

Output port labels

Enter the labels for your output ports using the spacer character, `|`, as the delimiter. This parameter is visible if you set the **Image signal** parameter to `Separate color signals`.

The **Data Types** pane of the Image From Workspace dialog box appears as shown in the following figure.



Source Block Parameters: Image From Workspace

Image From Workspace (mask) (link)

Imports an image from the MATLAB workspace.

Use the Value parameter to specify the MATLAB workspace variable that contains or an expression that specifies the image you want to import into your model. Use the Sample time parameter to set the sample period of the block.

Output data type: Fixed-point

[✓] Signed

Word length: 16

Set fraction length in output to: User-defined

Fraction length: 15

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.

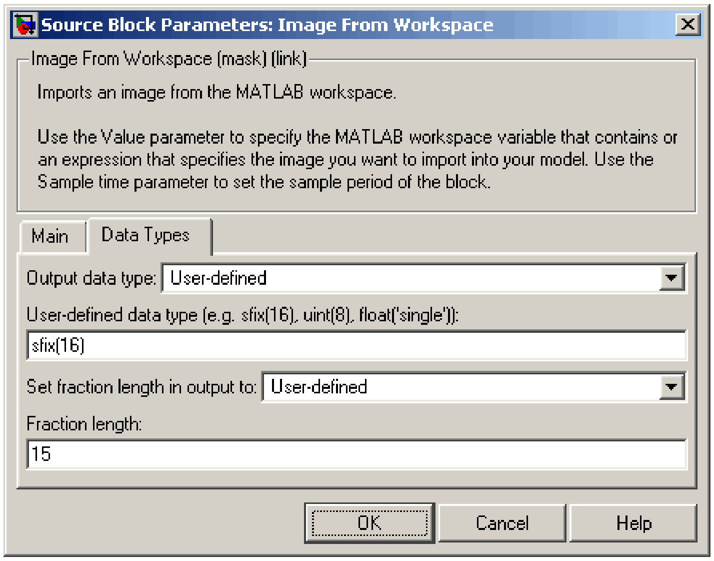


Image From Workspace (mask) (link)

Imports an image from the MATLAB workspace.

Use the Value parameter to specify the MATLAB workspace variable that contains or an expression that specifies the image you want to import into your model. Use the Sample time parameter to set the sample period of the block.

Output data type: User-defined

User-defined data type (e.g. sfix(16), uint(8), float('single')): sfix(16)

Set fraction length in output to: User-defined

Fraction length: 15

User-defined data type

Specify any built-in or fixed-point data type. You can specify fixed-point data types using the `sfix`, `ufix`, `sint`, `uint`, `sfrac`, and `ufrac` functions from the Simulink Fixed Point library. This parameter is only visible when you select `User-defined` for the **Output data type** parameter.

See Also

From Multimedia File	Video and Image Processing Blockset software
To Video Display	Video and Image Processing Blockset software
Video From Workspace	Video and Image Processing Blockset software
Video Viewer	Video and Image Processing Blockset software
`im2double`	Image Processing Toolbox software
`im2uint8`	Image Processing Toolbox software

Purpose

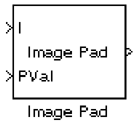
Pad signal along its rows, columns, or both

Library

Utilities

Description

The Image Pad block expands or crops the dimensions of a signal by padding or truncating its rows, columns, or both.



Port	Input/Output	Supported Data Types	Complex Values Supported
Image / I	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	• 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

- `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.

- `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 2-443 — Demonstrates the block's behavior when the **Method** parameter is set to `Constant`.
- "Example 2" on page 2-444— Demonstrates the block's behavior when the **Method** parameter is set to `Replicate`.

- "Example 3" on page 2-445— Demonstrates the block's behavior when the **Method** parameter is set to `Symmetric`.
- "Example 4" on page 2-446— 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:

$$\begin{bmatrix} 0 & 0 & 0 & a_{00} & a_{01} & a_{02} \\ 0 & 0 & 0 & a_{10} & a_{11} & a_{12} \\ 0 & 0 & 0 & a_{20} & a_{21} & a_{22} \end{bmatrix}$$

Example 2

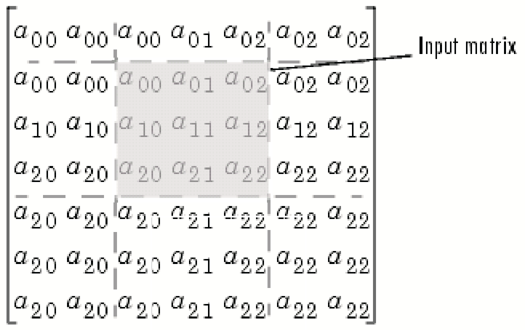
Suppose you want to pad your input signal with its border values, and your input signal is defined as follows:

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

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:



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`

- **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:

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

The block flips the original input matrix and each matrix it creates about their top, bottom, left, and right sides to populate the 7-by-13 output signal. For example, in the preceding figure, you can see how the block flips the input matrix about its right side to create the matrix directly to its right.

Example 4

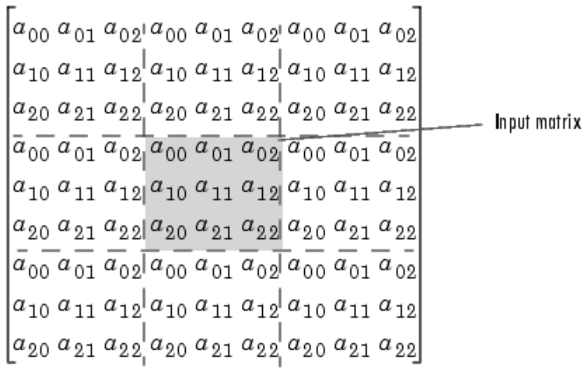
Suppose you want to pad your input signal using a circular repetition of its values. Your input signal is defined as follows:

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

Set the Image Pad block parameters as follows:

- **Method** = `Circular`
- **Specify** = `Output size`
- **Pad rows at** = `Both left and right`
- **Output row mode** = `User-specified`
- **Row size** = 9
- **Pad columns at** = `Both top and bottom`
- **Output column mode** = `User-specified`
- **Column size** = 9

The Image Pad block outputs the following signal:



The block repeats the values of the input signal in a circular pattern to populate the 9-by-9 output matrix.

Dialog Box

The Image Pad dialog box appears as shown in the following figure.

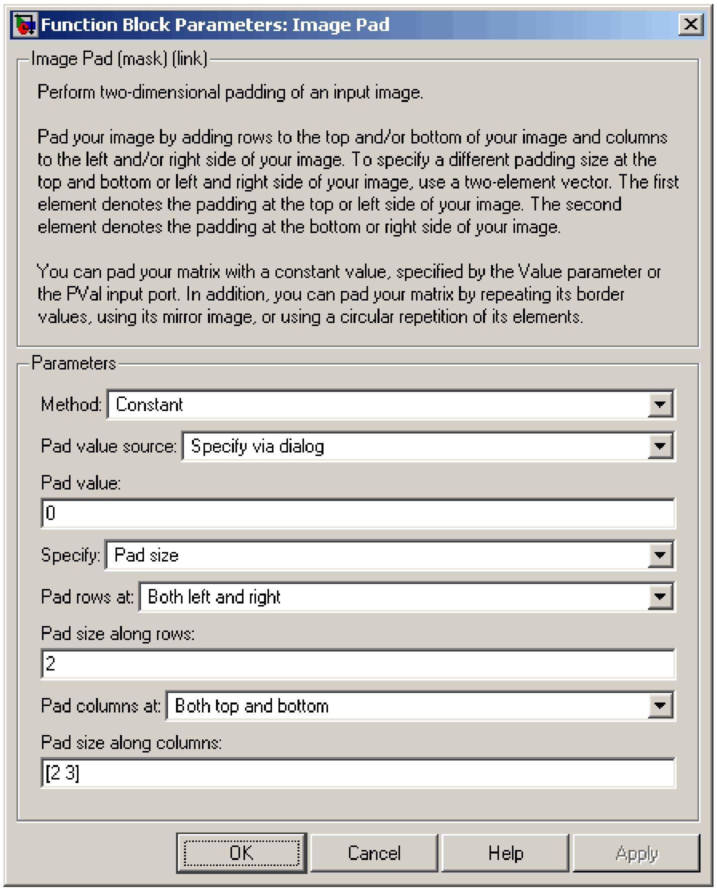


Image Pad (mask) (link)

Perform two-dimensional padding of an input image.

Pad your image by adding rows to the top and/or bottom of your image and columns to the left and/or right side of your image. To specify a different padding size at the top and bottom or left and right side of your image, use a two-element vector. The first element denotes the padding at the top or left side of your image. The second element denotes the padding at the bottom or right side of your image.

You can pad your matrix with a constant value, specified by the Value parameter or the PVal input port. In addition, you can pad your matrix by repeating its border values, using its mirror image, or using a circular repetition of its elements.

Parameters

Method: Constant

Pad value source: Specify via dialog

Pad value: 0

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: [2 3]

Method

Specify how you want the block to pad your signal.

Pad value source

If you select `Input port`, the PVal port appears on the block. Use this port to specify the constant value with which to pad your signal. If you select `Specify via dialog`, the **Pad value** parameter becomes available. This parameter is visible if, for the **Method** parameter, you select `Constant`.

Pad value
Enter the constant value with which to pad your signal. This parameter is visible if, for the **Pad value source** parameter, you select `Specify via dialog`. This parameter is tunable.

Specify
If you select `Pad size`, you can enter the size of the padding in the horizontal and vertical directions. If you select `Output size`, you can enter the total number of output columns and rows.

Pad rows at
Select `Left` to add additional columns to the left side of the signal. Select `Right` to add additional columns to the right side of the signal. Select `Both left and right` to add additional columns to the left and right side of the signal. If you select `No padding`, the block does not change the number of columns of the input signal.

Pad size along rows
This parameter controls how many columns are added to the right and/or left side of your input signal. Enter a scalar value, and the block adds this number of columns to the left, right, or both sides of your signal. If, for the **Pad rows at** parameter you selected `Both left and right`, enter a two-element vector. The left element controls the number of columns the block adds to the left side of the signal and the right element controls how many columns the block adds to the right side of the signal. This parameter is visible if, for the **Specify** parameter, you select `Pad size`.

Output row mode
Describe how to pad the input signal. If you select `User-specified`, the **Row size** parameter appears on the block dialog box. If you select `Next power of two`, the block pads the input signal along the rows until the length of the rows is equal to a power of two. This parameter is visible if, for the **Specify** parameter, you select `Output size`.

Row size

Enter a scalar value that represents the total number of output columns. This parameter is visible if you set the **Output row mode** parameter to `User-specified`.

Pad columns at

Select `Top` to add additional rows at the top of the input signal. Select `Bottom` to add additional rows at the bottom of the signal. Select `Both top and bottom` to add additional rows at the top and bottom of the signal. If you select `No padding`, the block does not change the number of rows of the input signal.

Pad size along columns

This parameter controls how many rows are added to the top, bottom, or both of your input signal. Enter a scalar value and the block adds this number of columns to the top, bottom, or both of your signal. If, for the **Pad columns at** parameter you selected `Both top and bottom`, enter a two-element vector. The left element controls the number of rows the block adds to the top of the signal and the right element controls how many rows the block adds to the bottom of the signal. This parameter is visible if you set the **Specify** parameter to `Pad size`.

Output column mode

Describe how to pad the input signal. If you select `User-specified`, the **Column size** parameter appears on the block dialog box. If you select `Next power of two`, the block pads the input signal along the columns until the length of the columns is equal to a power of two. This parameter is visible if, for the **Specify** parameter, you select `Output size`.

Column size

Enter a scalar value that represents the total number of output columns. This parameter is visible if you set the **Output column mode** parameter to `User-specified`.

Action when truncation occurs

Choose `None` when you do not want to be notified that the input signal is truncated. Select `Warning` to display a warning when the input signal is truncated. Choose `Error` when you want an error dialog box displayed and the simulation terminated when the input signal is truncated.

Insert Text

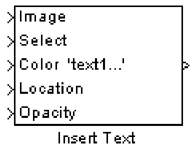
Purpose

Draw text on image or video stream.

Library

Text & Graphics

Description



The Insert Text block draws formatted text or numbers on an image or video stream. The block uses the FreeType 2.3.5 library, an open-source font engine, to produce stylized text bitmaps. To learn more about the FreeType Project, visit `http://www.freetype.org/`. The Insert Text block does not support character sets other than ASCII.

The Insert Text block lets you draw one or more instances of one or more strings, including:

- A single instance of one text string
- Multiple instances of one text string
- Multiple instances of text, with a different text string at each location

Input/Output	Description
Image	*M*-by-*N* matrix of intensity values or an *M*-by-*N*-by-*P* color video signal where *P* is the number of color planes.
R, G, B	Matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B ports have the same dimensions and data type.
Select	Zero-based index value that indicates which text string to display.
Variables	Vector or matrix whose values are used to replace ANSI C `printf`-style format specifications.
Color	Intensity input — Scalar value used for all strings or 1-by-*N* vector of intensity values whose length is equal to the number of strings. Color input — Three-element vector that specifies one color for all strings or a 3-by-*N* matrix of color values, where *N* is the number of strings.

Input/Output	Description
Location	2-by-*N* matrix, where *N* is the number of text strings to insert, that specifies the top-left corner of the text string bounding boxes.
Opacity	Scalar value that is used for all strings or vector of opacity values whose length is equal to the number of strings.

Use the **Text** parameter to specify the text string to be drawn on the image or video frame. This parameter can be a single text string, such as `'Figure1'`, a cell array of strings, such as `{'Figure1','Figure2'}`, or an ANSI C `printf`-style format specifications, such as %s.

If, for the **Text** parameter, you enter a cell array of strings, the Insert Text block does not display all of the strings simultaneously. Instead, the **Select** port appears on the block to let you indicate which text string to display. The input to this port must be a scalar value, where 0 indicates the first string. If the input is less than 0 or greater than one less than the number of strings in the cell array, the block does not draw text on the image or video frame.

If, for the **Text** parameter, you enter ANSI C `printf`-style format specifications, such as %d, %f, or %s, the **Variables** port appears on the block. The block replaces the format specifications in the **Text** parameter with each element of the input vector in turn. Use the %s option to specify a set of text strings for the block to display simultaneously at different locations. For example, using a Constant block, enter `[uint8('Text1') 0 uint8('Text2')]` for the **Constant value** parameter. The following table summarizes the supported conversion specifications.

Text Parameter Supported Conversion Specifications

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

Use the **Location source** parameter to indicate where to specify the text location:

- `Specify via dialog` — the **Location [row column]** parameter appears on the dialog box.
- `Input port` — the Location port appears on the block.

The following table describes how to format the location of the text strings depending on the number of strings you specify to insert. You can specify more than one location regardless of how many text strings you specify, but the only way to get a different text string at each location is to use the %s option for the **Text** parameter to specify a set of text strings. You can enter negative values or values that exceed the dimensions of the input image or video frame, but the text might not be visible.

Location Parameter Text String Insertion

Parameter	One Instance of One Text String	Multiple Instances of the Same Text String	Multiple Instances of Unique Text Strings
Location [row column] parameter setting or the input to the Location port	Two-element vector of the form [*row column*] that indicates the top-left corner of the text bounding box.	2-by-N matrix, where N is the number of locations at which to display the text string. Each column indicates the row and column coordinate of the top-left corner of the text bounding box for the string, e.g., `[[row1 column1]' [row2 column2]']`	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 the corresponding string, e.g., `[[row1 column1]' [row2 column2]']`.

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

- `Specify via dialog` — the **Color value** parameter appears on the dialog box.
- `Input port` — the Color port appears on the block.

The following table describes how to format the color of the text strings depending on the block input and the number of strings you want to insert. If the input image is a floating-point data type, the color values must be between 0 and 1. If the input image is an 8-bit unsigned integer data type, the color values must range between 0 and 255.

Text String Color Values

Block Input	One Text String	Multiple Text Strings
Intensity image	**Color value** parameter or the input to the Color port = Scalar intensity value	**Color value** parameter or the input to the Color port = Vector of intensity values whose length is equal to the number of strings
Color image	**Color value** parameter or the input to the Color port = RGB triplet that specifies the color of the text	**Color value** parameter or the input to the Color port = 3-by-*N* matrix of color values, where *N* is the number of strings

Use the **Opacity source** parameter to indicate where to specify the text's opaqueness:

- `Specify via dialog` — the **Opacity** parameter appears on the dialog box.
- `Input port` — the Opacity port appears on the block.

The following table describes how to format the opacity of the text strings depending on the number of strings you want to insert.

Text String Opacity Values

Parameter	One Text String	Multiple Text Strings
Opacity parameter setting or the input to the Opacity port	Scalar value between 0 and 1, where 0 is translucent and 1 is opaque	Vector whose length is equal to the number of strings

Use the **Image signal** parameter to specify how to input and output a color video signal:

- `One multidimensional signal` — the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port.
- `Separate color signals` — additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

Use the **Font face** parameter to specify the font of your text. The block populates this list with the TrueType fonts installed on your system. On Windows, the block searches the system registry for font files. On UNIX, the block searches the X Server's font path for font files.

Use the **Font size (points)** parameter to specify the font size.

If you select the **Anti-aliased** check box, the block smooths the edges of the text, which can be computationally expensive. If you want your model to run faster, clear this check box.

Row-Major Data Format

MATLAB and the Video and Image Processing Blockset blocks use column-major data organization. However, the Insert Text block gives you the option to process data that is stored in row-major format. When you select the **Input image is transposed (data order is row major)** check box, the block assumes that the input buffer contains contiguous data elements from the first row first, then data elements from the second row second, and so on through the last row. Use this functionality only when you meet all the following criteria:

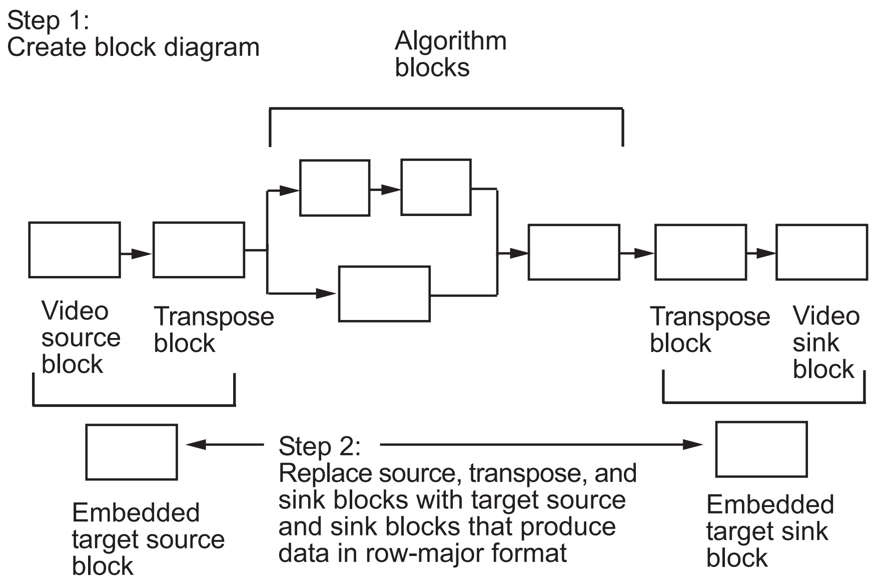
- You are developing algorithms to run on an embedded target that uses the row-major format.
- You want to limit the additional processing required to take the transpose of signals at the interfaces of the row-major and column-major systems.

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

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

  For example, if you use the 2-D FIR Filter block, you need to verify that your filter coefficients are transposed. If you are using the Rotate block, you need to use negative rotation angles, etc.
- Only three blocks have the **Input image is transposed (data order is row major)** check box. They are the Chroma Resampling, Deinterlacing, and Insert Text blocks. You need to select this check box to enable row-major functionality in these blocks. All other blocks must be properly configured to process data in row-major format.

Use the following two-step workflow to develop algorithms in row-major format to run on an embedded target.


Step 1:
Create block diagram
Algorithm blocks
Video source block
Transpose block
Transpose block
Video sink block
Step 2:
Replace source, transpose, and sink blocks with target source and sink blocks that produce data in row-major format
Embedded target source block
Embedded target sink block

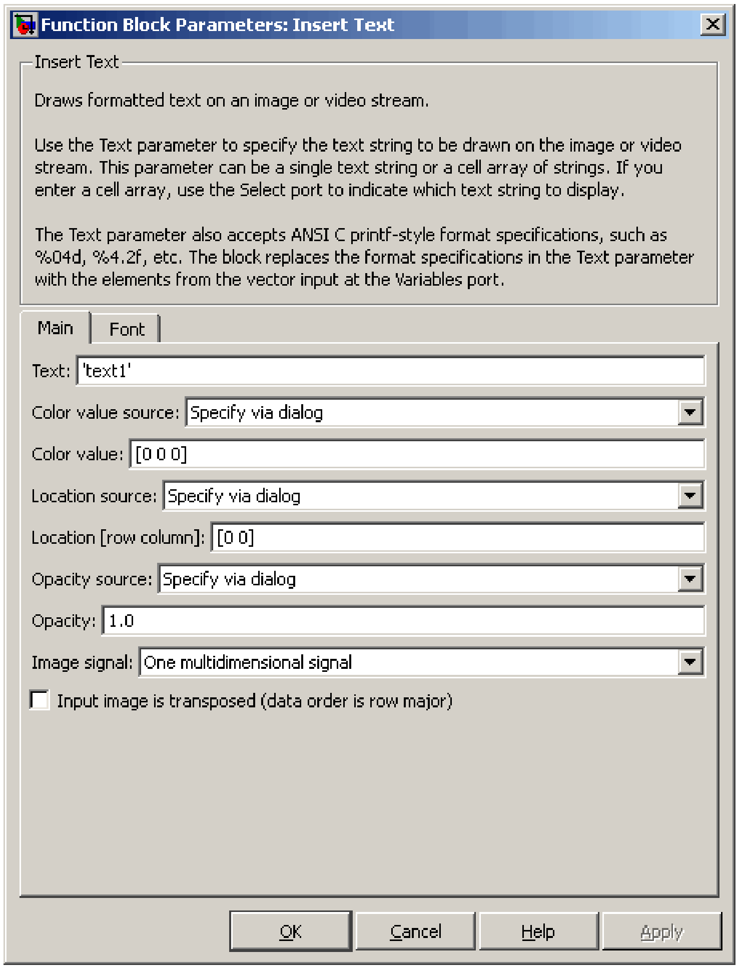
See the DM642 EVM Video ADC and DM642 EVM Video DAC reference pages in the *Target Support Package TC6 User's Guide* for more information about data order in embedded targets.

Examples

See "Annotating AVI Files with Video Frame Numbers" and "Annotating AVI Files at Two Separate Locations" in the *Video and Image Processing Blockset User's Guide*. Many of the Video and Image Processing Blockset Demos make use of the Insert Text block.

Dialog Box

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



Insert Text

Draws formatted text on an image or video stream.

Use the Text parameter to 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. If you enter a cell array, use the Select port to indicate which text string to display.

The Text parameter also accepts ANSI C printf-style format specifications, such as %04d, %4.2f, etc. The block replaces the format specifications in the Text parameter with the elements from the vector input at the Variables port.

Text: 'text1'

Color value source: Specify via dialog

Color value: [0 0 0]

Location source: Specify via dialog

Location [row column]: [0 0]

Opacity source: Specify via dialog

Opacity: 1.0

Image signal: One multidimensional signal

☐ Input image is transposed (data order is row major)

Text

Specify the text string to be drawn on the image or video stream. This parameter can be a single text string or a cell array of strings.

To create a **Select** port enter a cell array of strings. To create a **Variables** port, enter ANSI C printf-style format specifications, such as `%d, %f, or %s`.

Color value source
Indicate where you want to specify the text color. Your choices are `Specify via dialog` or `Input port`.

Color value
Specify the intensity or color of the text. This parameter is visible if, for the **Color source** parameter, you select `Specify via dialog`. Tunable.

Location source
Indicate where you want to specify the text location. Your choices are `Specify via dialog` or `Input port`.

Location [row column]
Specify the text location. This parameter is visible if, for the **Location source** parameter, you select `Specify via dialog`. Tunable.

Opacity source
Indicate where you want to specify the text's opaqueness. Your choices are `Specify via dialog` or `Input port`.

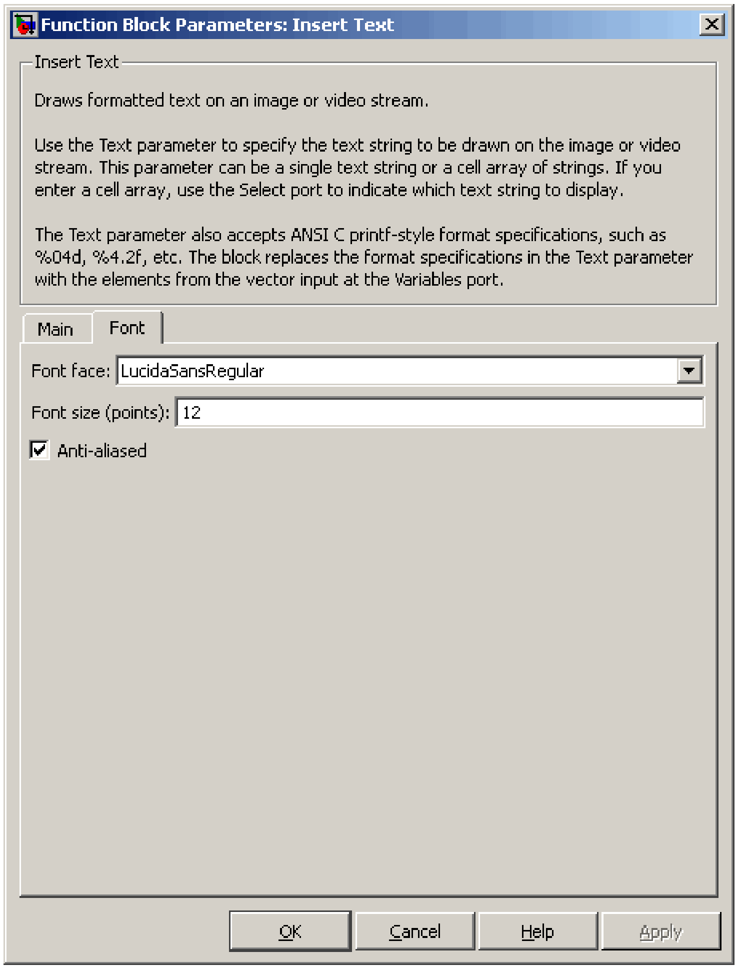
Opacity
Specify the opacity of the text. This parameter is visible if, for the **Opacity source** parameter, you select `Specify via dialog`. Tunable.

Image signal
Specify how to input and output a color video signal. If you select `One multidimensional signal`, the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select `Separate color signals`, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

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.


Function Block Parameters: Insert Text

Insert Text

Draws formatted text on an image or video stream.

Use the Text parameter to 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. If you enter a cell array, use the Select port to indicate which text string to display.

The Text parameter also accepts ANSI C printf-style format specifications, such as %04d, %4.2f, etc. The block replaces the format specifications in the Text parameter with the elements from the vector input at the Variables port.

Main | Font

Font face: LucidaSansRegular

Font size (points): 12

☑ Anti-aliased

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.

Anti-aliased

Select this check box if you want the block to smooth the edges of the text.

Supported Data Types

Port	Supported Data Types
Input/Image	• Double-precision floating point • Single-precision floating point • Fixed point(signed, word length less than or equal to 32.) • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer
R, G, B	Same as Input port
Select	• Double-precision floating point. (This data type is only supported if the input to the I or R, G, and B ports is a floating-point data type.) • Single-precision floating point. (This data type is only supported if the input to the I or R, G, and B ports is a floating-point data type.) • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer

Port	Supported Data Types
Variables	The data types supported by this port depend on the conversion specification you are using in the **Text** parameter. %d, %i, and %u: • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer %c and %s: • 8-bit unsigned integer %f: • Double-precision floating point • Single-precision floating point %o, %x, %X, %e, %E, %g, and %G: • Double-precision floating point • Single-precision floating point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer
Color	Same as Input port (The input to this port must be the same data type as the input to the Input port.)

Port	Supported Data Types
Location	• Double-precision floating point. (This data type is only supported if the input to the I or R, G, and B ports is a floating-point data type.) • Single-precision floating point. (This data type is only supported if the input to the I or R, G, and B ports is a floating-point data type.) • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer
Opacity	• 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.) • `ufix8_En7` (This data type is only supported if the input to the I or R, G, and B ports is a fixed-point data type.)

See Also

Draw Shapes	Video and Image Processing Blockset software

Purpose Predict or estimate states of dynamic systems

Library Filtering

Description The Kalman Filter block is a Signal Processing Blockset block. For more information, see the Kalman Filter block reference page in the Signal Processing Blockset software documentation.

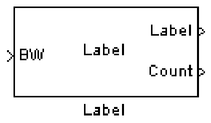
Purpose

Label connected components in binary images

Library

Morphological Operations

Description

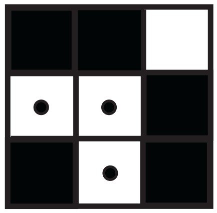


The Label block labels the objects in a binary image, BW, where the background is represented by pixels equal to 0 (black) and objects are represented by pixels equal to 1 (white). At the Label port, the block outputs a label matrix that is the same size as the input matrix. In the label matrix, pixels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

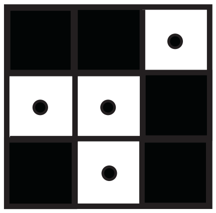
Port	Input/Output	Supported Data Types	Complex Values Supported
BW	Vector or matrix that represents a binary image	Boolean	No
Label	Label matrix	• 8-, 16-, and 32-bit unsigned integer	No
Count	Scalar that represents the number of labeled objects	Same as Label port	No

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.

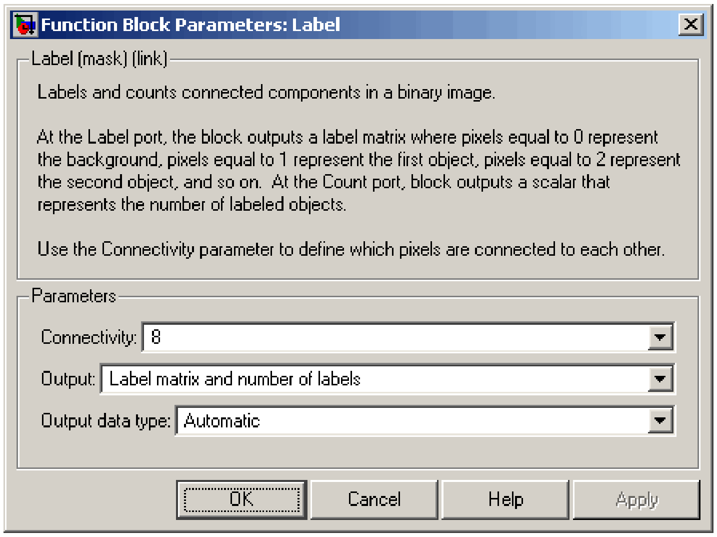


Use the **Output** parameter to determine the block's output. If you select `Label matrix and number of labels`, ports Label and Count appear on the block. The block outputs the label matrix at the Label port and the number of labeled objects at the Count port. If you select `Label matrix`, the Label 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 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.



Function Block Parameters: Label

Label (mask) (link)

Labels and counts connected components in a binary image.

At the Label port, the block outputs a label matrix where pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on. At the Count port, block outputs a scalar that represents the number of labeled objects.

Use the Connectivity parameter to define which pixels are connected to each other.

Parameters

Connectivity: 8

Output: Label matrix and number of labels

Output data type: Automatic

Connectivity
: Specify which pixels are connected to each other. If you want a pixel to be connected to the pixels on the top, bottom, left, and right, select `4`. If you want a pixel to be connected to the pixels on the top, bottom, left, right, and diagonally, select `8`.

Output
: Determine the block's output. If you select `Label matrix and number of labels`, the Label and Count ports appear on the block. The block outputs the label matrix at the Label port and the number of labeled objects at the Count port. If you select

`Label matrix`, the Label port appears on the block. If you select `Number of labels`, the Count port appears on the block.

Output data type
: Set the data type of the outputs at the Label and Count ports. If you select `Automatic`, the block determines the appropriate data type for the output. If you select `uint32`, `uint16`, or `uint8`, the data type of the output is 32-, 16-, or 8-bit unsigned integers, respectively.

If label exceeds data type size, mark remaining regions using
: Use this parameter to specify the block's behavior if the number of found objects exceeds the maximum number that can be represented by the output data type. If you select `Maximum value of the output data type`, the remaining regions are labeled with the maximum value of the output data type. If you select `Zero`, the remaining regions are labeled with zeroes. This parameter is visible if, for the **Output data type** parameter, you choose `uint16` or `uint8`.

See Also

Bottom-hat	Video and Image Processing Blockset software
Closing	Video and Image Processing Blockset software
Dilation	Video and Image Processing Blockset software
Erosion	Video and Image Processing Blockset software
Opening	Video and Image Processing Blockset software
Top-hat	Video and Image Processing Blockset software

Label

`bwlabel`	Image Processing Toolbox software
`bwlabeln`	Image Processing Toolbox software

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 software documentation.

Mean

Purpose Find mean value of each input matrix

Library Statistics

Description The Mean block is a Signal Processing Blockset block. For more information, see the Mean block reference page in the Signal Processing Blockset software documentation.

Purpose Find median value of each input matrix

Library Statistics

Description The Median block is a Signal Processing Blockset block. For more information, see the Median block reference page in the Signal Processing Blockset software documentation.

Median Filter

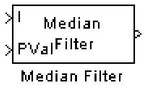
Purpose

Perform 2-D median filtering

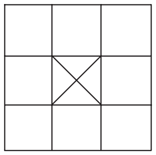
Library

Filtering / Analysis & Enhancement

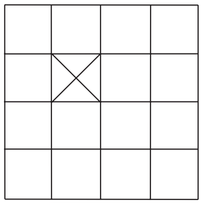
Description



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.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Matrix of intensity values	• 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

```
output rows = input rows - neighborhood rows + 1
output columns = input columns - neighborhood columns + 1
```

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 2-480. 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

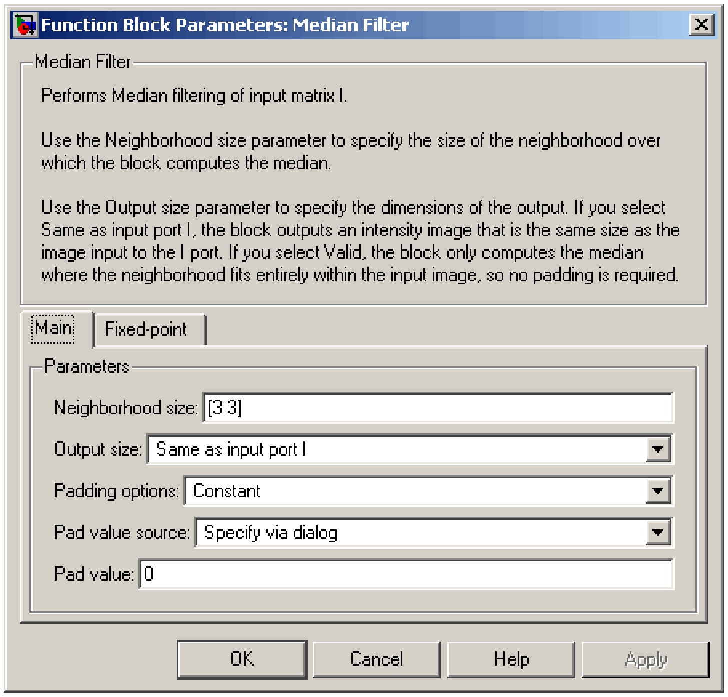
The accumulator and output data types and scalings are used for fixed-point signals when `M` is even. The result of the sum performed while calculating the average of the two central rows of the input matrix is stored in the accumulator data type and scaling. The total result of the average is then put 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 Median Filter dialog box appears as shown in the following figure.



Performs Median filtering of input matrix I.

Use the Neighborhood size parameter to specify the size of the neighborhood over which the block computes the median.

Use the Output size parameter to specify the dimensions of the output. 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.

Neighborhood size: [3 3]

Output size: Same as input port I

Padding options: Constant

Pad value source: Specify via dialog

Pad value: 0

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` `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`.

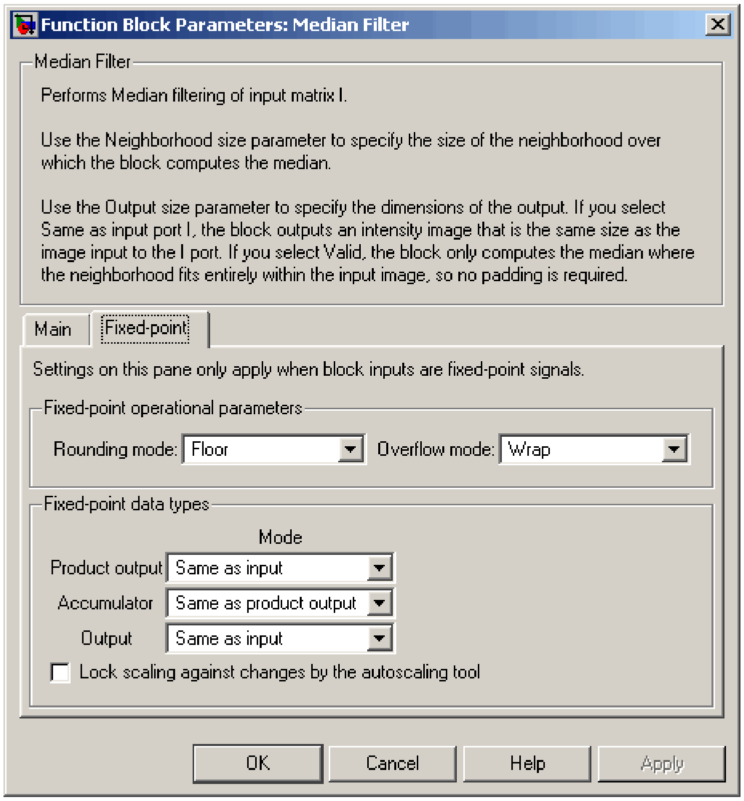
Pad value source

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.



Median Filter

Performs Median filtering of input matrix I.

Use the Neighborhood size parameter to specify the size of the neighborhood over which the block computes the median.

Use the Output size parameter to specify the dimensions of the output. 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.

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

Fixed-point operational parameters

Rounding mode: Floor   Overflow mode: Wrap

Fixed-point data types

	Mode
Product output	Same as input
Accumulator	Same as product output
Output	Same as input

☐ Lock scaling against changes by the autoscaling tool

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 2-478 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 software
2-D FIR Filter	Video and Image Processing Blockset software
`medfilt2`	Image Processing Toolbox software

Purpose Find minimum values in input or sequence of inputs

Library Statistics

Description The Minimum block is a Signal Processing Blockset block. For more information, see the Minimum block reference page in the Signal Processing Blockset software documentation.

Opening

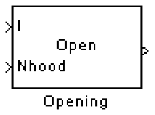
Purpose

Perform morphological opening on binary or intensity images

Library

Morphological Operations

Description



The Opening block performs an erosion operation followed by a dilation operation using a predefined neighborhood or structuring element. This block uses flat structuring elements only.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
Nhood	Matrix or vector of ones and zeros that represents the neighborhood values	Boolean	No
Output	Scalar, vector, or matrix of intensity values that represents the opened image	Same as I port	No

The output signal has the same data type as the input to the I port.

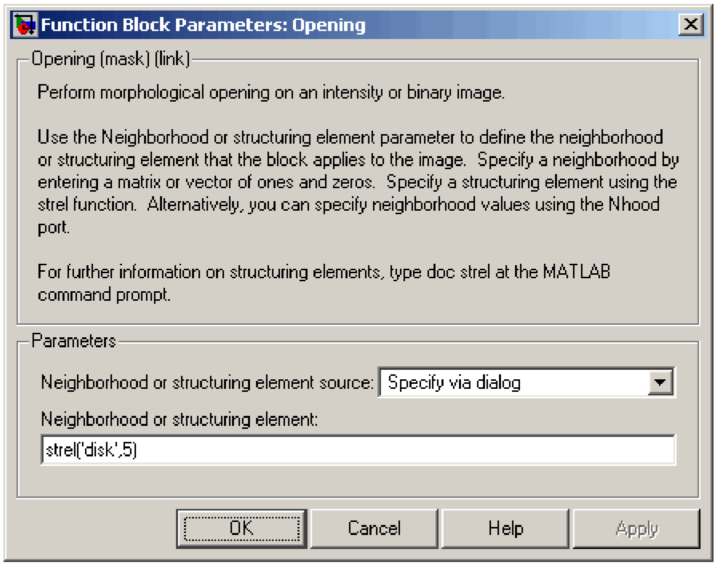
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select `Specify via dialog`, the **Neighborhood or structuring**

element parameter appears in the dialog box. If you select `Input port`, the Nhood port appears on the block. Use this port to enter your neighborhood values as a matrix or vector of 1s and 0s. You can only specify a structuring element using the dialog box.

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

Dialog Box

The Opening dialog box appears as shown in the following figure.


Function Block Parameters: Opening

Opening (mask) (link)

Perform morphological opening on an intensity or binary image.

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 ones and zeros. Specify a structuring element using the strel function. Alternatively, you can specify neighborhood values using the Nhood port.

For further information on structuring elements, type doc strel at the MATLAB command prompt.

Parameters

Neighborhood or structuring element source: Specify via dialog

Neighborhood or structuring element:

strel('disk',5)

Neighborhood or structuring element source

Specify how to enter your neighborhood or structuring element values. Select `Specify via dialog` to enter the values in the dialog box. Select `Input port` to use the Nhood port to specify the neighborhood values. You can only specify a structuring element using the dialog box.

Neighborhood or structuring element

If you are specifying a neighborhood, this parameter must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the `strel` function from the Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or structuring element source** parameter, you select `Specify via dialog`.

References

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

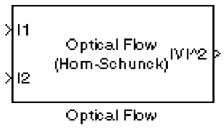
See Also

Bottom-hat	Video and Image Processing Blockset software
Closing	Video and Image Processing Blockset software
Dilation	Video and Image Processing Blockset software
Erosion	Video and Image Processing Blockset software
Label	Video and Image Processing Blockset software
Top-hat	Video and Image Processing Blockset software
`imopen`	Image Processing Toolbox software
`strel`	Image Processing Toolbox software

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	• 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, $[u \quad v]^T$, that minimizes this equation:

$$E = \iint (I_x u + I_y v + I_t)^2 dxdy + \alpha \iint \left\{ \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \right\} dxdy$$

In this equation, $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial y}$ are the spatial derivatives of the optical velocity component u, and α scales the global smoothness term. The Horn-Schunck method minimizes the previous equation to obtain the velocity field, [u v], for each pixel in the image, which is given by the following equations:

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

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

In this equation, $\begin{bmatrix} u_{x,y}^k & v_{x,y}^k \end{bmatrix}$ is the velocity estimate for the pixel at (x,y), and $\begin{bmatrix} \bar{u}_{x,y}^k & \bar{v}_{x,y}^k \end{bmatrix}$ is the neighborhood average of $\begin{bmatrix} u_{x,y}^k & v_{x,y}^k \end{bmatrix}$. For $k=0$, the initial velocity is 0.

If you set the **Method** parameter to `Horn-Schunck`, the block solves for u and v as follows:

1. Compute I_x and I_y using the Sobel convolution kernel: $[-1 \;\; -2 \;\; -1; \;\; 0 \;\; 0 \;\; 0; \;\; 1 \;\; 2 \;\; 1]$, and its transposed form for each pixel in the first image.
2. Compute I_t between images 1 and 2 using the $[-1 \;\; 1]$ kernel.
3. Assume the previous velocity to be 0, and compute the average velocity for each pixel using $[0 \;\; 1 \;\; 0; \;\; 1 \;\; 0 \;\; 1; \;\; 0 \;\; 1 \;\; 0]$ 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 $[u \quad v]^T$ 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 $[-1 \quad 8 \quad 0 \quad -8 \quad 1]/12$ 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 $[-1 \quad 1]$ 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 $[1 \quad 4 \quad 6 \quad 4 \quad 1]/16$. 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 2-490.

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 $\lambda_2 < \tau$

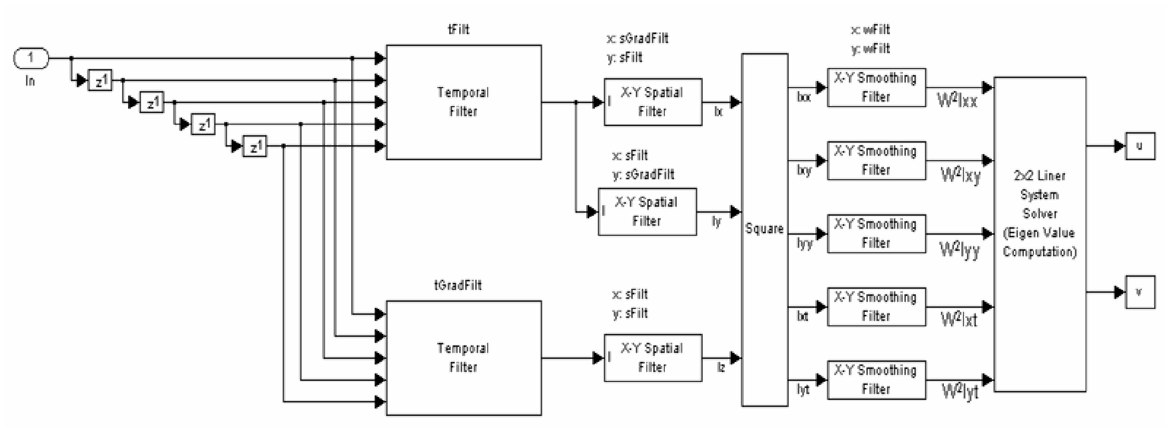
  The optical flow, u and v, is 0.

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 2-490.

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.



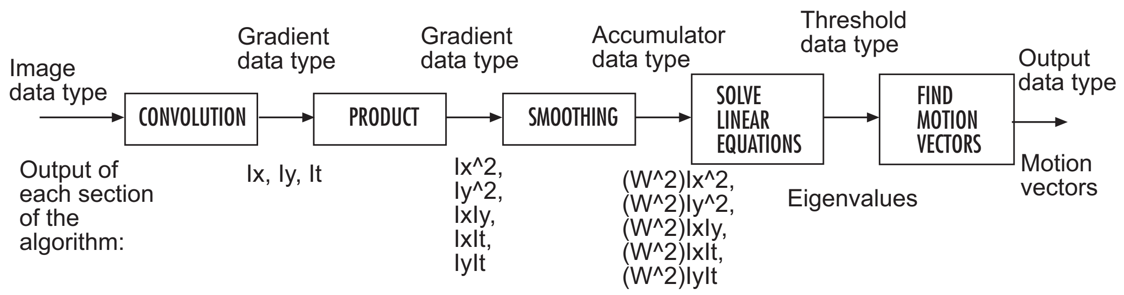
tFilt = Coefficients of Gaussian Filter
tGradFilt = Coefficients of the Derivative of a Gaussian Filter

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

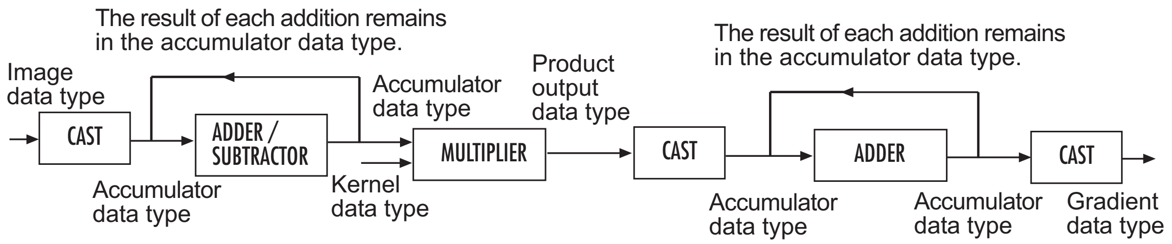
Fixed-Point Data Type Diagram

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`.

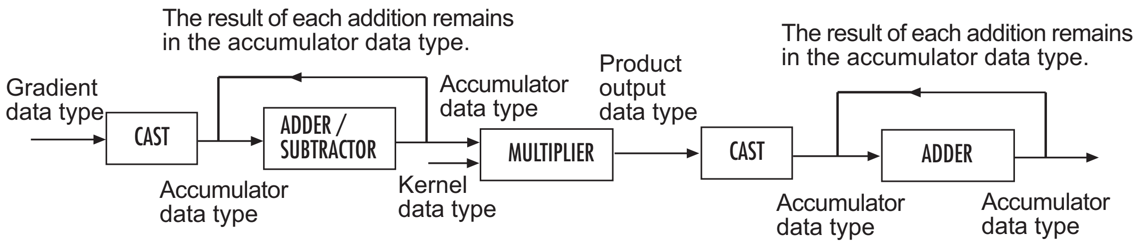
Data type diagram for Optical Flow block's overall algorithm



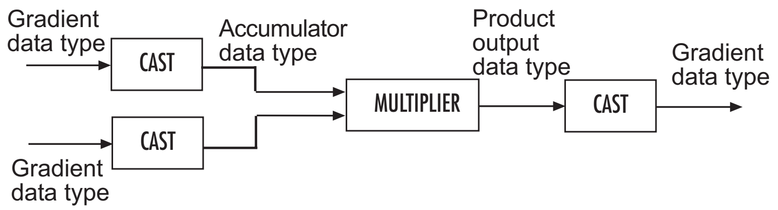
Data type diagram for convolution algorithm



Data type diagram for smoothing algorithm

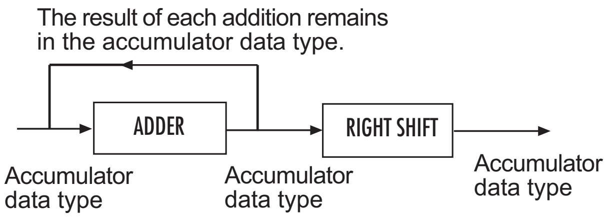


Data type diagram for product algorithm

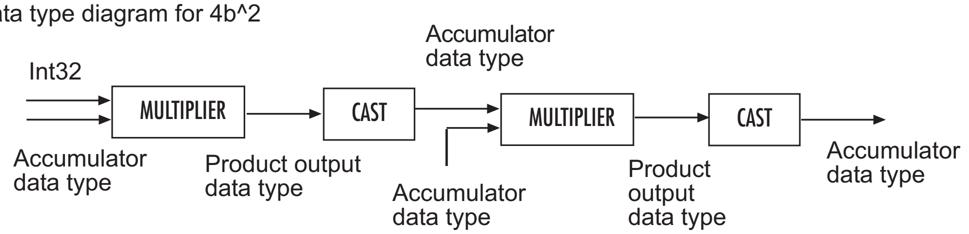


Solving linear equations to compute eigenvalues
(see Step 4 in the Lucas-Kanade Method section for the eigenvalue equations)

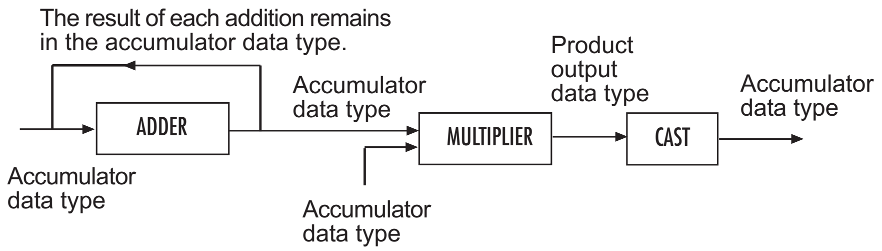
Data type diagram for P



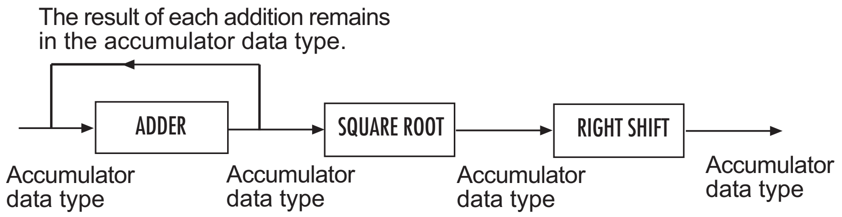
Data type diagram for 4b^2



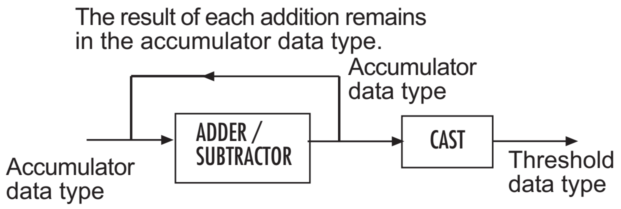
Data type diagram for (a-c)^2



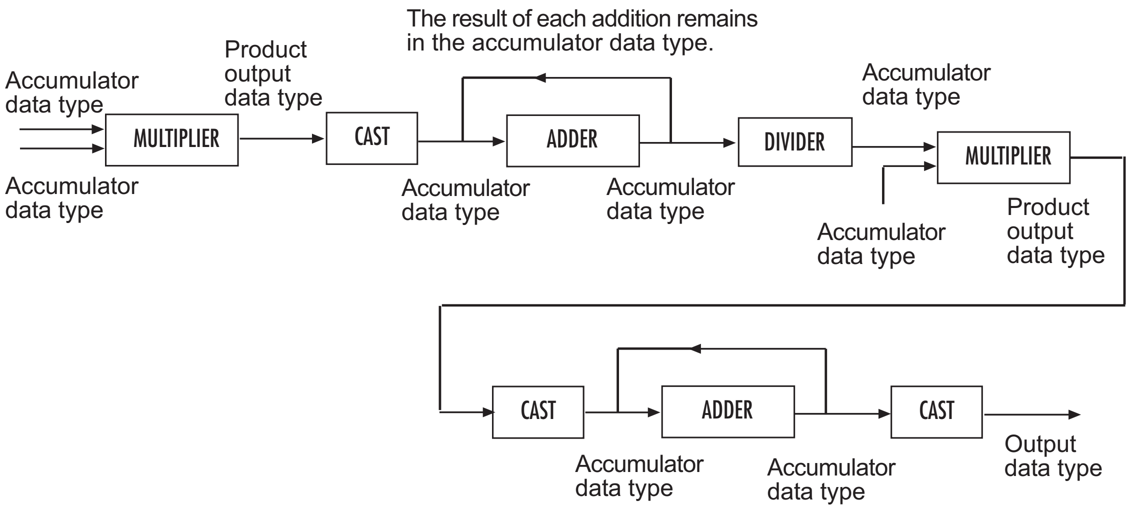
Data type diagram for Q



Data type diagram for eigenvalues



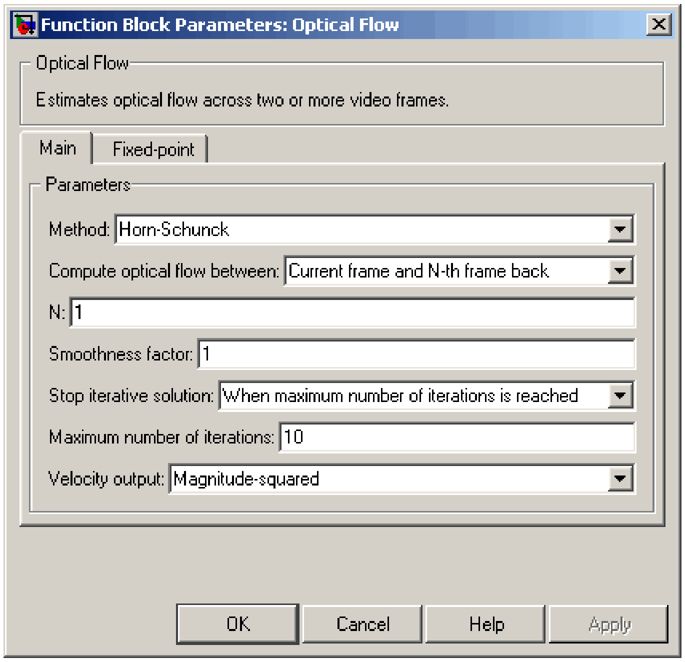
Data type diagram for finding the motion vectors algorithm



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

Dialog Box

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



Optical Flow

Estimates optical flow across two or more video frames.

Parameters

Method: Horn-Schunck

Compute optical flow between: Current frame and N-th frame back

N: 1

Smoothness factor: 1

Stop iterative solution: When maximum number of iterations is reached

Maximum number of iterations: 10

Velocity output: Magnitude-squared

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)

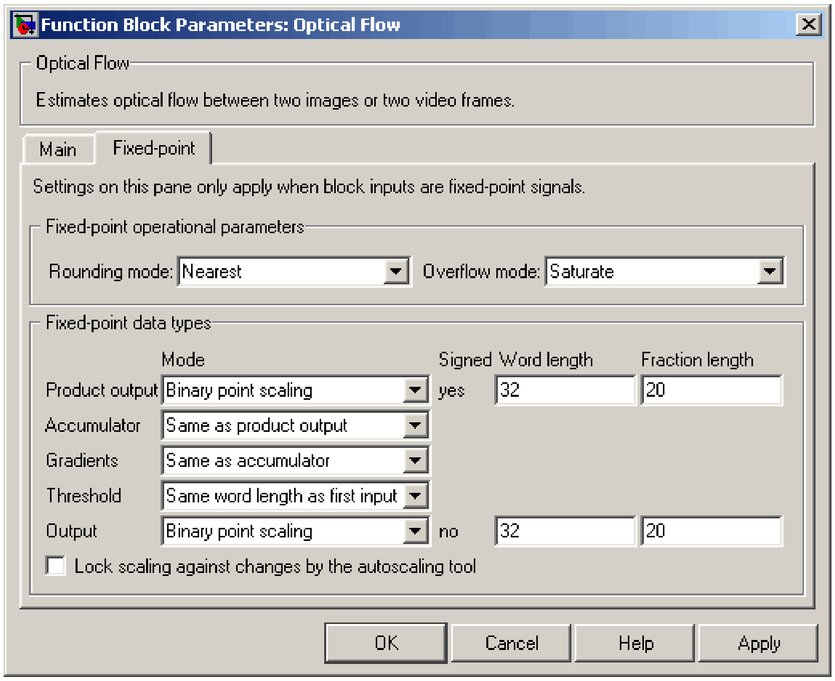
Select this check box if you want the block to output the image that corresponds to the motion vector being output by the block.

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.



Optical Flow

Estimates optical flow between two images or two video frames.

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

Fixed-point operational parameters

Rounding mode: Nearest Overflow mode: Saturate

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Product output	Binary point scaling	yes	32	20
Accumulator	Same as product output			
Gradients	Same as accumulator			
Threshold	Same word length as first input			
Output	Binary point scaling	no	32	20

☐ Lock scaling against changes by the autoscaling tool

Rounding mode

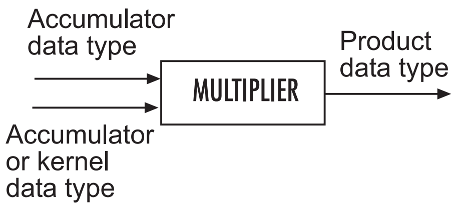
Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

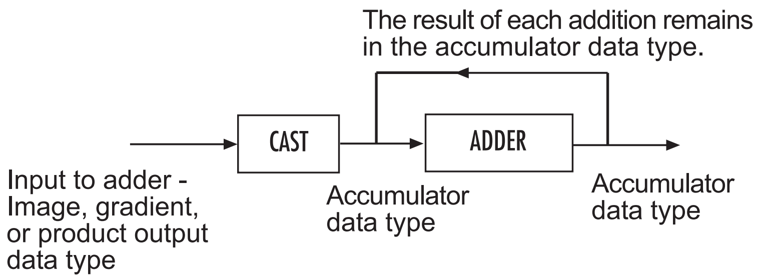
Use this parameter to specify how to designate the product output word and fraction lengths.



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

Accumulator

Use this parameter to specify how to designate this accumulator word and fraction lengths.


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - Image, gradient, or product output data type
Accumulator data type
Accumulator data type

- When you select `Same as product output`, these characteristics match those of the product output.

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

Gradients

Choose how to specify the word length and fraction length of the gradients data type:

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

Threshold

Choose how to specify the word length and fraction length of the threshold data type:

- When you select `Same word length as first input`, the threshold word length matches that of the first input.
- When you select `Specify word length`, enter the word length of the threshold data type.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the threshold, in bits.
- When you select `Slope and bias scaling`, you can enter the word length in bits and the slope of the threshold. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Output

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

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

Lock 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 software
Gaussian Pyramid	Video and Image Processing Blockset software

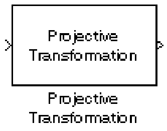
Purpose

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	• 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

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	• 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 The block rounds the values input at this port and converts them to 32-bit signed integers.	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

Port	Output	Supported Data Types	Complex Values Supported
OutSize	Four-element vector, [r c height width], that represents the row and column coordinates of the upper-left corner of the rectangular output image as well as its height and width	Same as InPts port	No
Valid	Boolean value that indicates whether or not three quadrilateral vertices are collinear	Boolean	No
InPtsValid	Boolean value that indicates whether or not three input quadrilateral vertices are collinear	Boolean	No
OutPtsValid	Boolean value that indicates whether or not three output quadrilateral vertices are collinear	Boolean	No

Use the **Image signal** parameter to specify how to input and output a color video signal. If you select `One multidimensional signal`, the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select `Separate color signals`, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

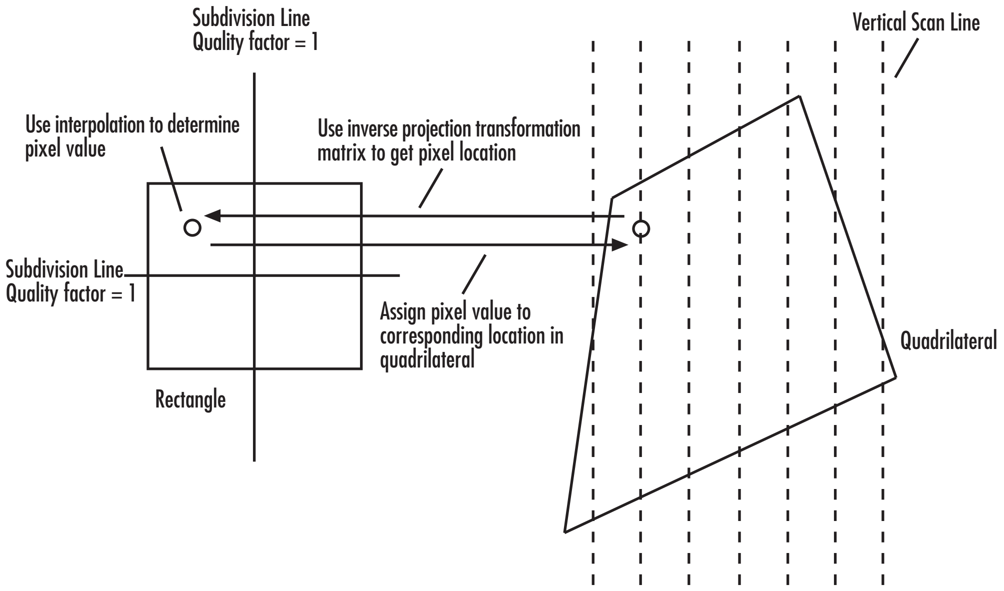
The following sections summarize the behavior of the Projective Transformation block in its three modes.

Rectangle to Quadrilateral Mode

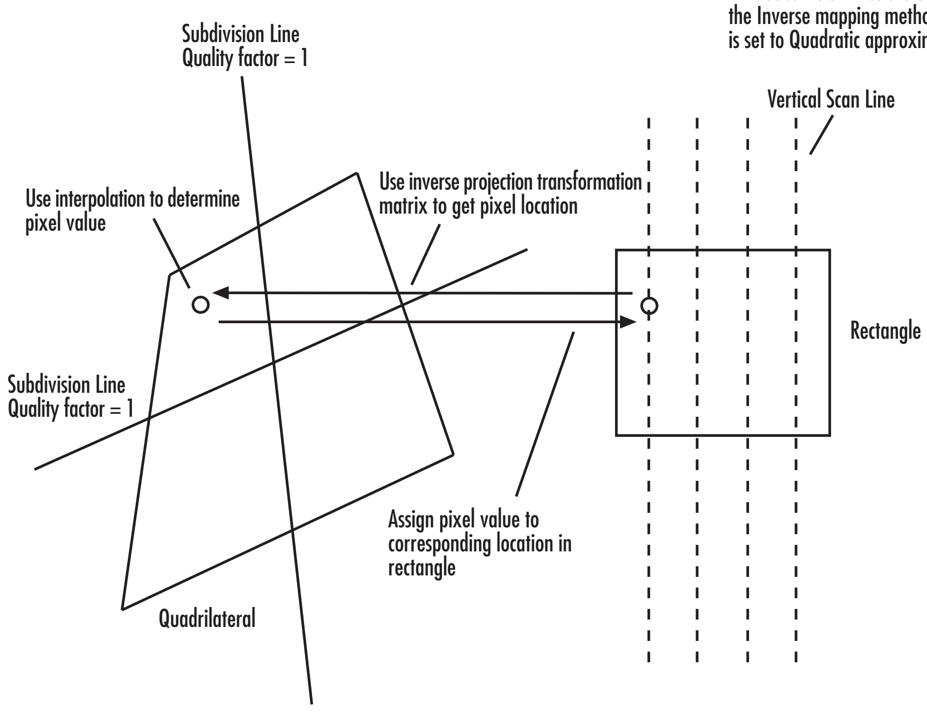
Use the **Inverse mapping method** parameter to specify the algorithm the block uses to implement the projective transformation. If you choose `Exact solution`, the block divides the output shape using vertical scan lines. For each pixel location on a scan line, it uses an inverse projection transformation matrix to find the corresponding pixel location in the input image. When this pixel location is not located directly on a pixel in the input image, the block uses 2-D interpolation to calculate the pixel value. Then it assigns this pixel value to the corresponding location in the output image.

If you choose `Quadratic approximation`, the block divides the input shape using subdivision lines and the output shape using vertical scan lines. For the first pixel location on a scan line, the block uses an inverse projection transformation matrix to find the corresponding pixel location in the input image. If this pixel location is not located directly on a pixel in the input image, the block uses 2-D interpolation to calculate the pixel value. Then it assigns this pixel value to the corresponding location in the output image. The block calculates the remaining pixel locations using x and y offsets that it computes from the inverse projection transformation matrix. Then it repeats the interpolation process to find all the pixel values in the output image.

The following figures summarize two operations of the Projective Transformation block.


Subdivision Line
Quality factor = 1
Vertical Scan Line
Use interpolation to determine pixel value
Use inverse projection transformation matrix to get pixel location
Subdivision Line
Quality factor = 1
Assign pixel value to corresponding location in quadrilateral
Quadrilateral
Rectangle

*The subdivision lines are only used when the Inverse mapping method parameter is set to Quadratic approximation.


Subdivision Line
Quality factor = 1
Vertical Scan Line
Use interpolation to determine pixel value
Use inverse projection transformation matrix to get pixel location
Rectangle
Subdivision Line
Quality factor = 1
Assign pixel value to corresponding location in rectangle
Quadrilateral

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 2-512.

Input image parameters — Use the **Quadrilateral vertices source** parameter to specify how to define the input quadrilateral vertices. If you select `Specify via dialog`, the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter appears in the dialog box. Enter an eight-element vector of values that represent the row and column coordinates of the four corners of the quadrilateral. If you select `Input port`, the InPts port appears on the block. The input to this port must be an eight-element vector of scalar values that represent the row and column coordinates of the four corners of the input quadrilateral. Use the **If vertices are outside input image** parameter to specify the block's behavior if the input to the InPts port contains vertices outside the input image. Your choices are `Clip`, `Clip and warn`, or `Error`. If you select `Clip`, the block changes the row or column values of the vertices so that the quadrilateral fits entirely within the input image. If you select the **Output validity of quadrilateral vertices (three points cannot be collinear)** check box, the Valid port appears on the block. If the quadrilateral vertices are not collinear, the block outputs 1 at this port. Otherwise it outputs 0, and the block does not compute an output image.

Output image parameters — Use the **Rectangle size source** parameter to specify how to define the output rectangle size. If you select `Specify via dialog`, the **Rectangle location and size [r,c,height,width]** and **Size** parameters appear in the dialog box. For the **Rectangle location and size [r,c,height,width]** parameter, enter scalar values that represent the row and column coordinates as well as the height and width of the output rectangle. Use the **Size** parameter to specify the size of the block's output image, which can differ from the size of the output rectangle. If you select `Full`, the block output size is determined by the values you enter for the **Rectangle location and size [r,c,height,width]** parameter. That is, the block output is big enough so you see the entire output rectangle. If you select `User-defined`, use the **Location and size [r,c,height,width]** parameter to define the row and column coordinates as well as the height and width of the output image. If, for the **Rectangle size source** parameter, you select `Input port`, the OutSize port appears on the block. The input to this port must be a four-element vector of scalar values that represent the row and column coordinates of the upper-left corner of the output rectangle as well as its height and width. Use the **Location and size [r,c,height,width]** parameter to define 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 2-512.

Input image parameters — Use the **Quadrilateral vertices source** parameter to specify how to define the input quadrilateral vertices. If you select `Specify via dialog`, the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter appears in the dialog box. Enter an eight-element vector of values that represent the row and column coordinates of the four corners of the input quadrilateral. If you select `Input port`, the InPts port appears on the block. The input to this port must be an eight-element vector of scalar values that represent the row and column coordinates of the four corners of the input quadrilateral. Use the **If vertices are outside input image** parameter to specify the block's behavior if the input to the InPts port contains vertices outside the input image. Your choices are `Clip`, `Clip and warn`, or `Error`. If you select `Clip`, the block changes the row or column values of the vertices so that the quadrilateral fits entirely within the input image. If you select the **Output validity of quadrilateral vertices (three points cannot be collinear)** check box, the InPtsValid port appears on the block. If the quadrilateral vertices are not collinear, the block outputs 1 at this port. Otherwise it outputs 0, and the block does not compute an output image.

Output image parameters — Use the **Quadrilateral vertices source** parameter to specify how to define the output quadrilateral vertices. If you select `Specify via dialog`, the **Quadrilateral vertices [r1,c1,...,r4,c4]** and **Size** parameters appear in the dialog box. For the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter, enter an eight-element vector of values that represent the row and column coordinates of the four corners of the output quadrilateral. If, for the **Size** parameter, you select `Full`, the block output image size is determined by the values you enter for the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter. If you select `User-defined`, use the **Location and size [r,c,height,width]** parameter to define the row and column coordinates as well as the height and width of the output image, which can differ from the size of the output quadrilateral. If, for the **Quadrilateral vertices source**, you select `Input port`, the OutPts port appears on the block. The input to this port must be an eight-element vector of scalar values that represent the row and column coordinates of the four corners of the output quadrilateral. Use the **Location and size [r,c,height,width]** parameter to define the row and column coordinates as well as the height and width of the block's output image. If you select the **Output validity of quadrilateral vertices (three points cannot be collinear)** check box, the OutPtsValid port appears on the block. If the quadrilateral vertices are not collinear, the block outputs 1 at this port. Otherwise it outputs 0, and the block does not compute an output image.

Note If you set the **Inverse mapping method** parameter to `Quadratic approximation` and the **Quality factor (number of subdivisions)** parameter to a value greater than 0, the subquadrilaterals formed by the subdivision lines might have three collinear vertices. In this case, the block does not compute an output image.

Example

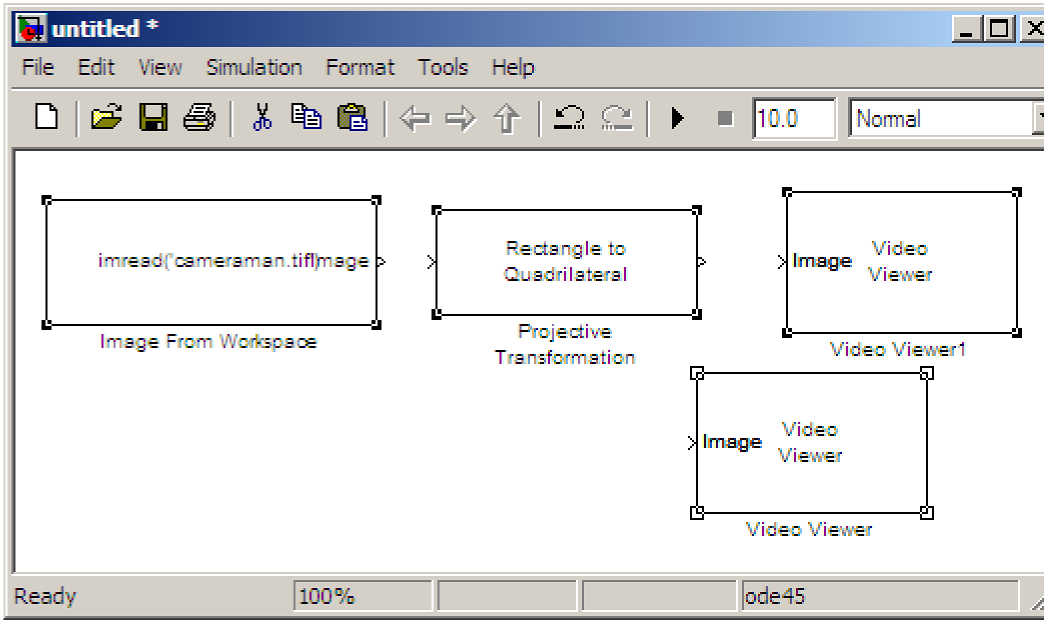
The following example shows you how to convert a rectangular image into a quadrilateral. It also shows you how to change the sizes of the input and output images.

1 Create a new Simulink model.

2 Click-and-drag the following blocks into your model.

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

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

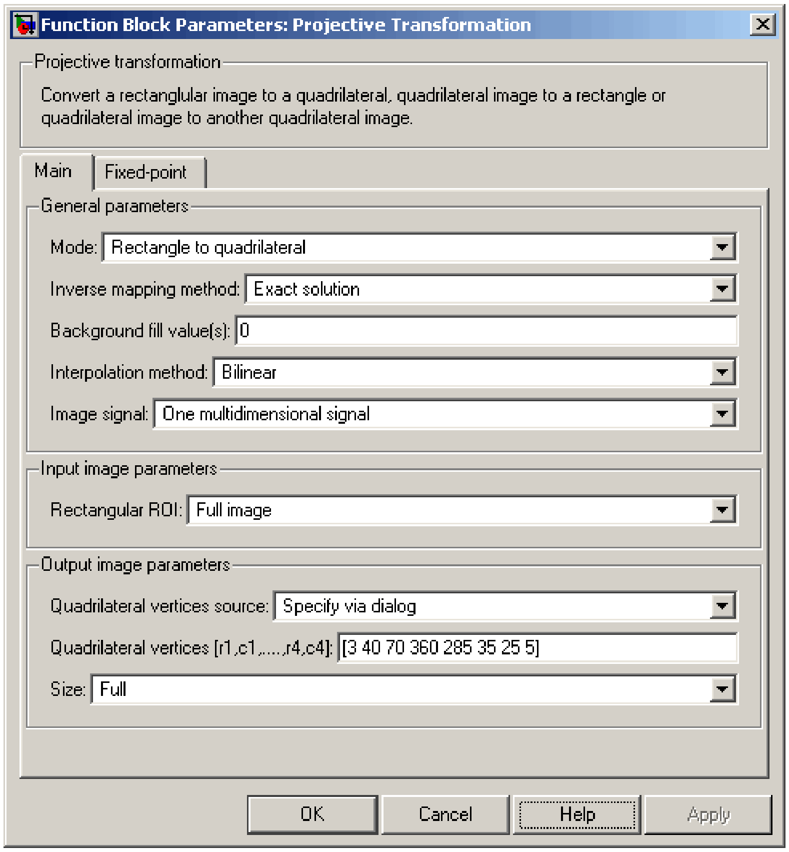


4 Use the Image From Workspace block to import an image into your model.

- 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]`.



Function Block Parameters: Projective Transformation

Projective transformation

Convert a rectanglular image to a quadrilateral, quadrilateral image to a rectangle or quadrilateral image to another quadrilateral image.

General parameters

Mode: Rectangle to quadrilateral

Inverse mapping method: Exact solution

Background fill value(s): 0

Interpolation method: Bilinear

Image signal: One multidimensional signal

Input image parameters

Rectangular ROI: Full image

Output image parameters

Quadrilateral vertices source: Specify via dialog

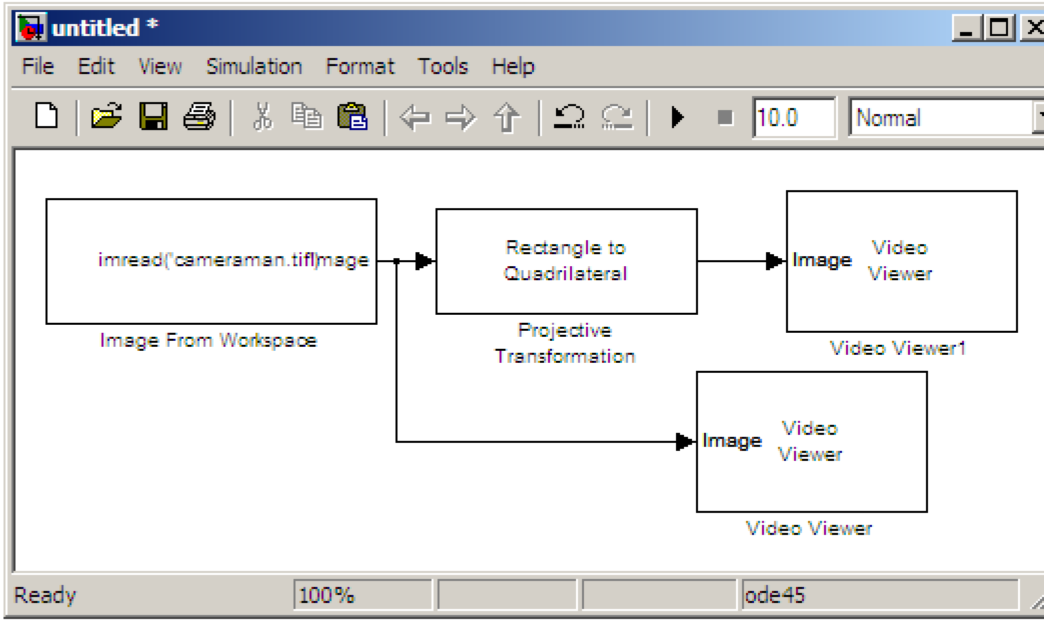
Quadrilateral vertices [r1,c1,....,r4,c4]: [3 40 70 360 285 35 25 5]

Size: Full

Note The order in which you enter the quadrilateral vertices in the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter affects the appearance of the output image. The block assumes that the first row and column pair correspond to the new location of the upper-left corner of the image. The second row and column pair correspond to the new location of the upper-right corner, and so on in a clockwise direction around the image.

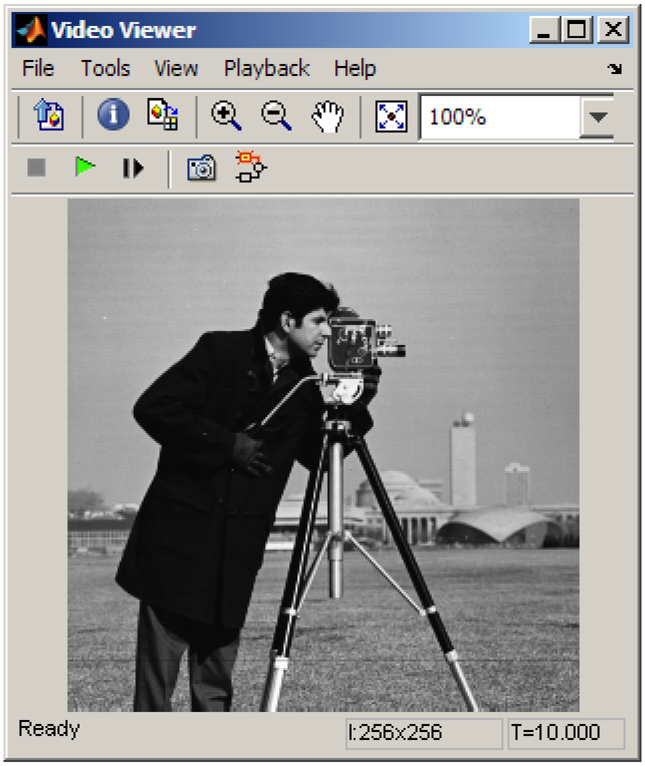
6 Use the Video Viewer and Video Viewer1 blocks to view the rectangular and quadrilateral images, respectively. Use the default parameters.

7 Connect the blocks so that your model resembles the following figure.

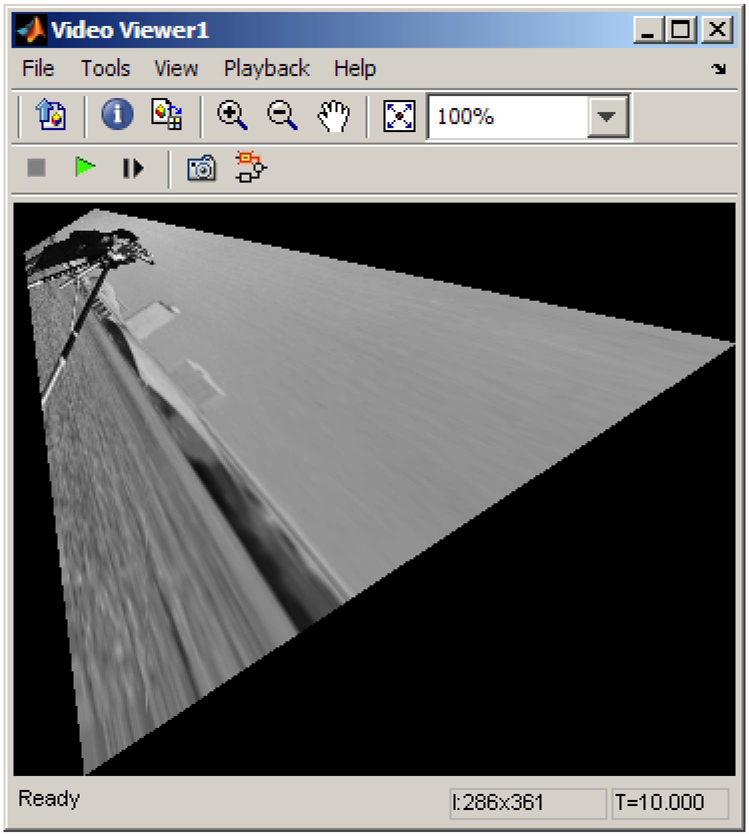


8 Run the model.

The original rectangular image appears in the Video Viewer1 window.



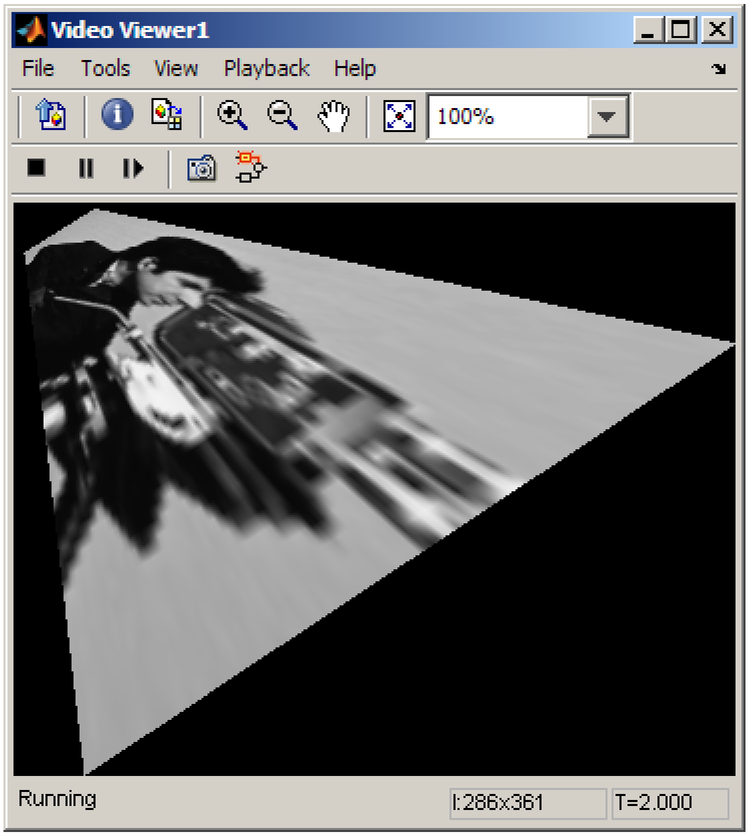
The quadrilateral image appears in the Video Viewer window.



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.

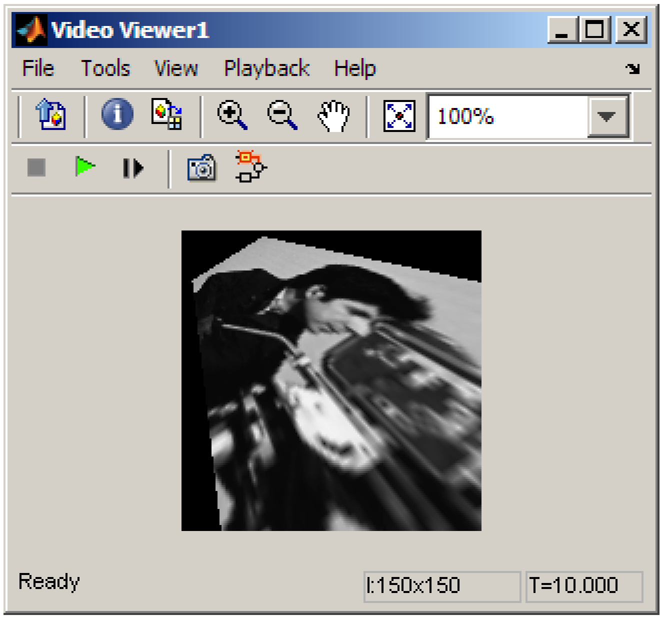


11 You can resize of the output image using the parameters in the **Output image parameters** section of the Projective Transformation dialog box. Set the block parameters as follows:

- **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.

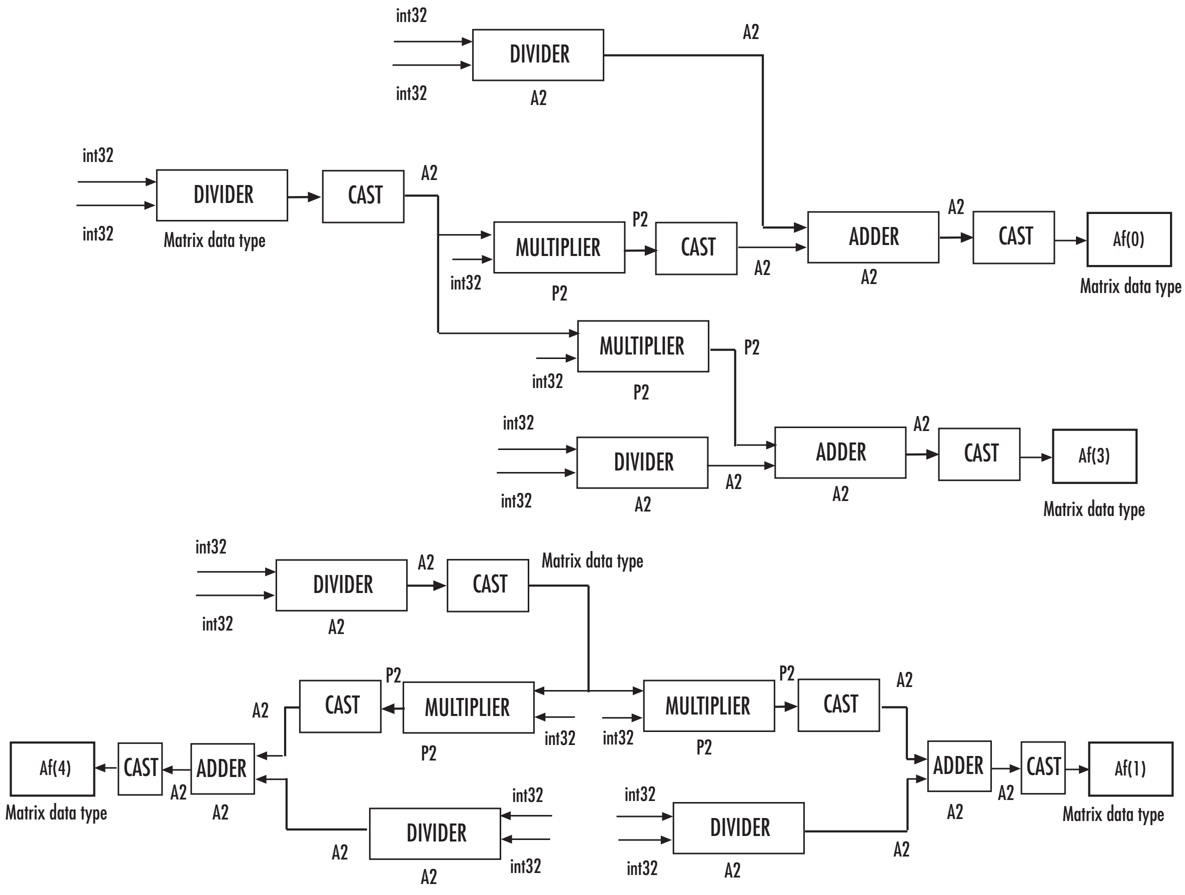
12 Run your model. The Projective Transformation block outputs a portion of the quadrilateral image, so you can no longer see all of the quadrilateral corners.



Fixed-Point Data Types

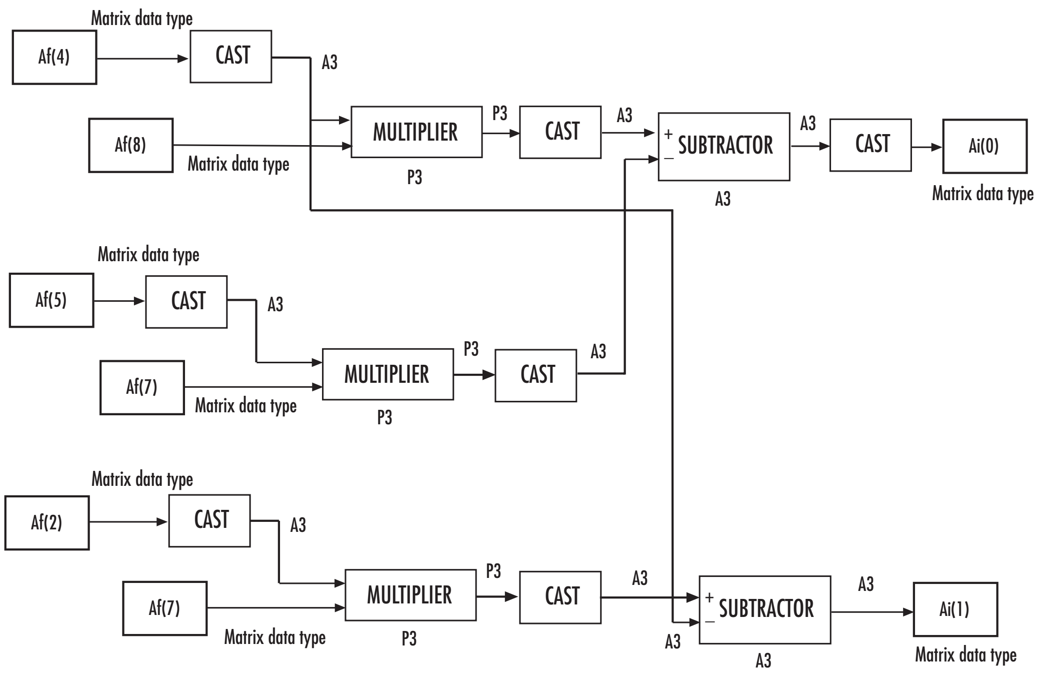
The following diagram shows the data types used in the Projective Transformation block for fixed-point signals:

Calculate Forward Projective Transformation Matrix (Af)

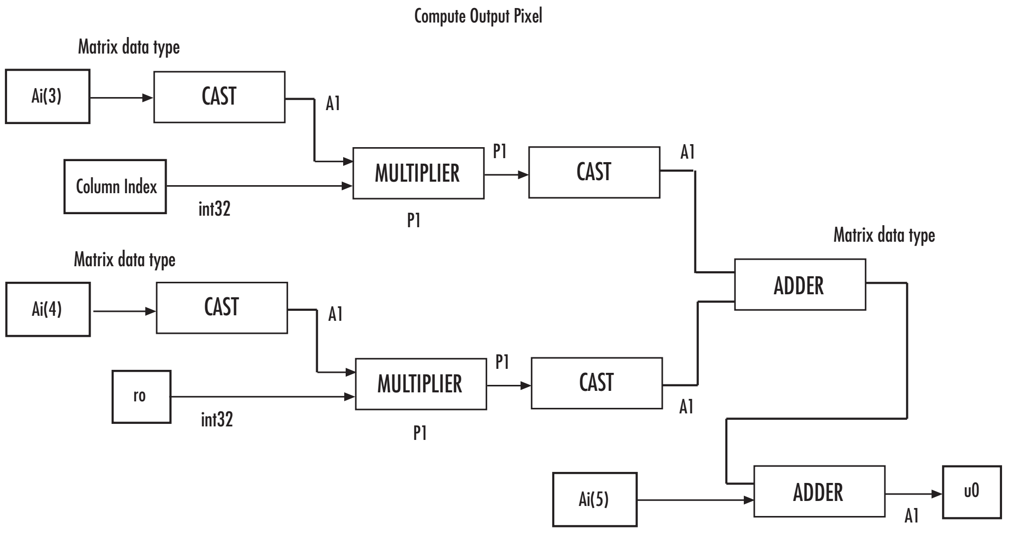


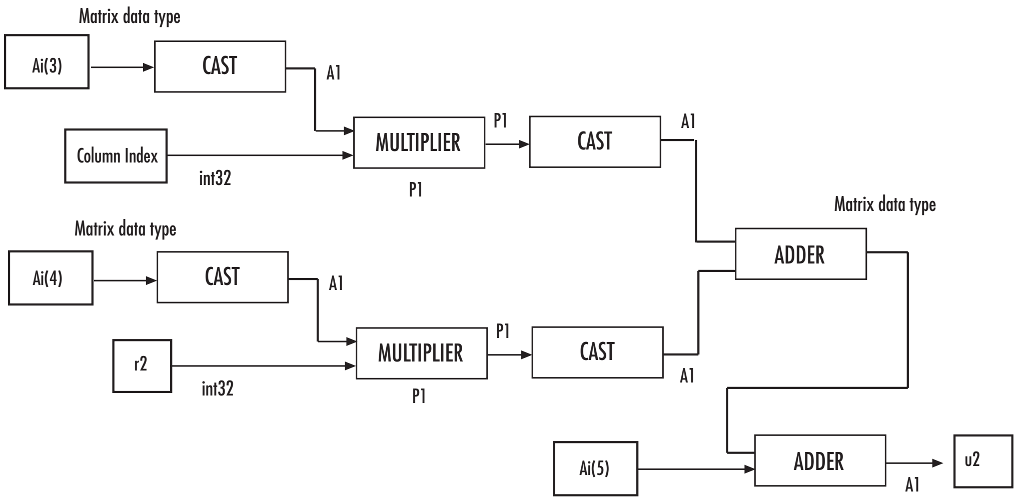
The block uses a similar computation to calculate Af(2), Af(4), Af(5), Af(6), Af(7), and Af(8).

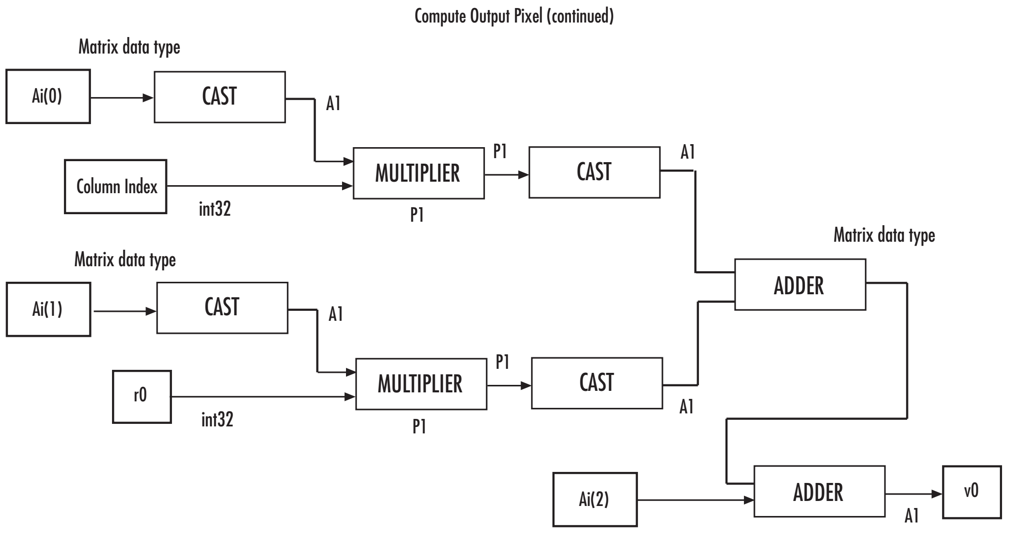
Calculate Inverse Projective Transformation Matrix (Ai)

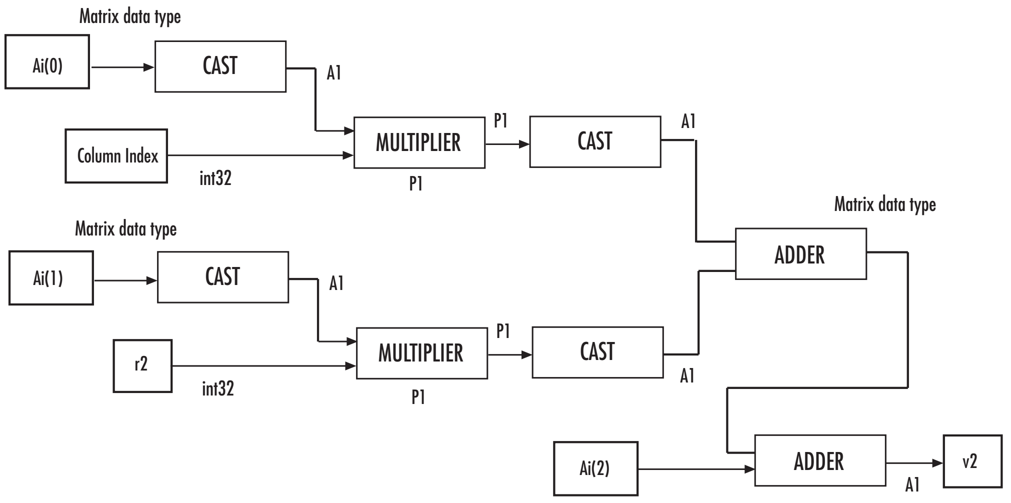


The block uses a similar computation to calculate Ai(2), Ai(3), Ai(4), Ai(5), Ai(6), Ai(7), and Ai(8).

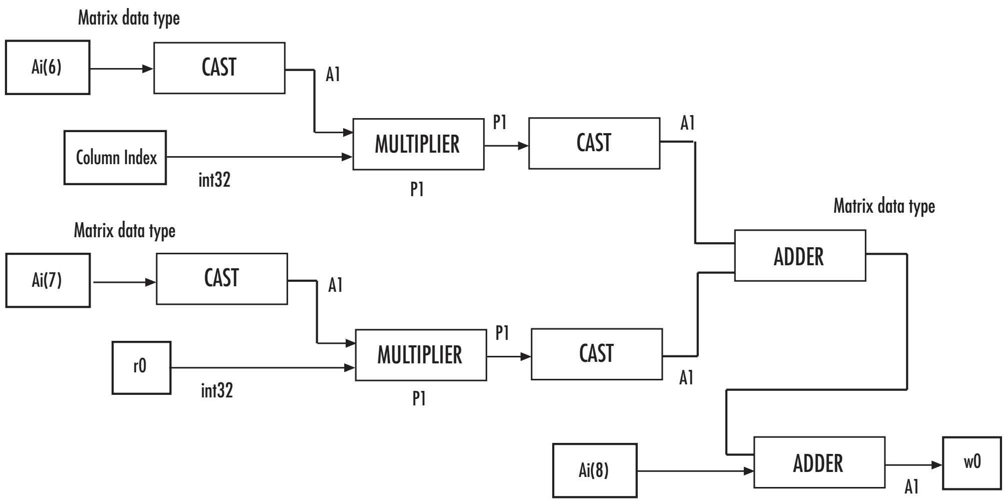

Compute Output Pixel
Matrix data type
Ai(3)
CAST
A1
Column Index
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
ADDER
Matrix data type
Ai(4)
CAST
A1
ro
int32
MULTIPLIER
P1
P1
CAST
A1
Ai(5)
ADDER
u0
A1

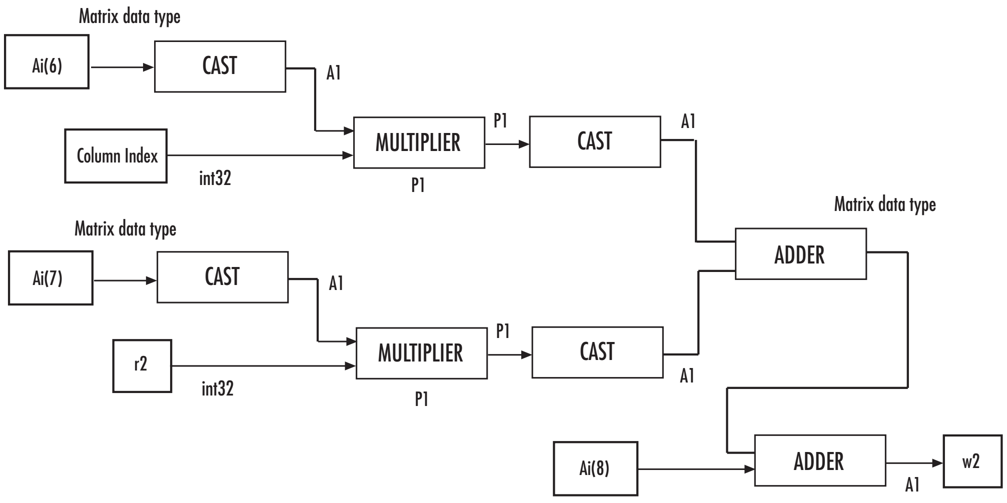

Matrix data type
Ai(3)
CAST
A1
Column Index
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
ADDER
Matrix data type
Ai(4)
CAST
A1
r2
int32
MULTIPLIER
P1
P1
CAST
A1
Ai(5)
ADDER
u2
A1

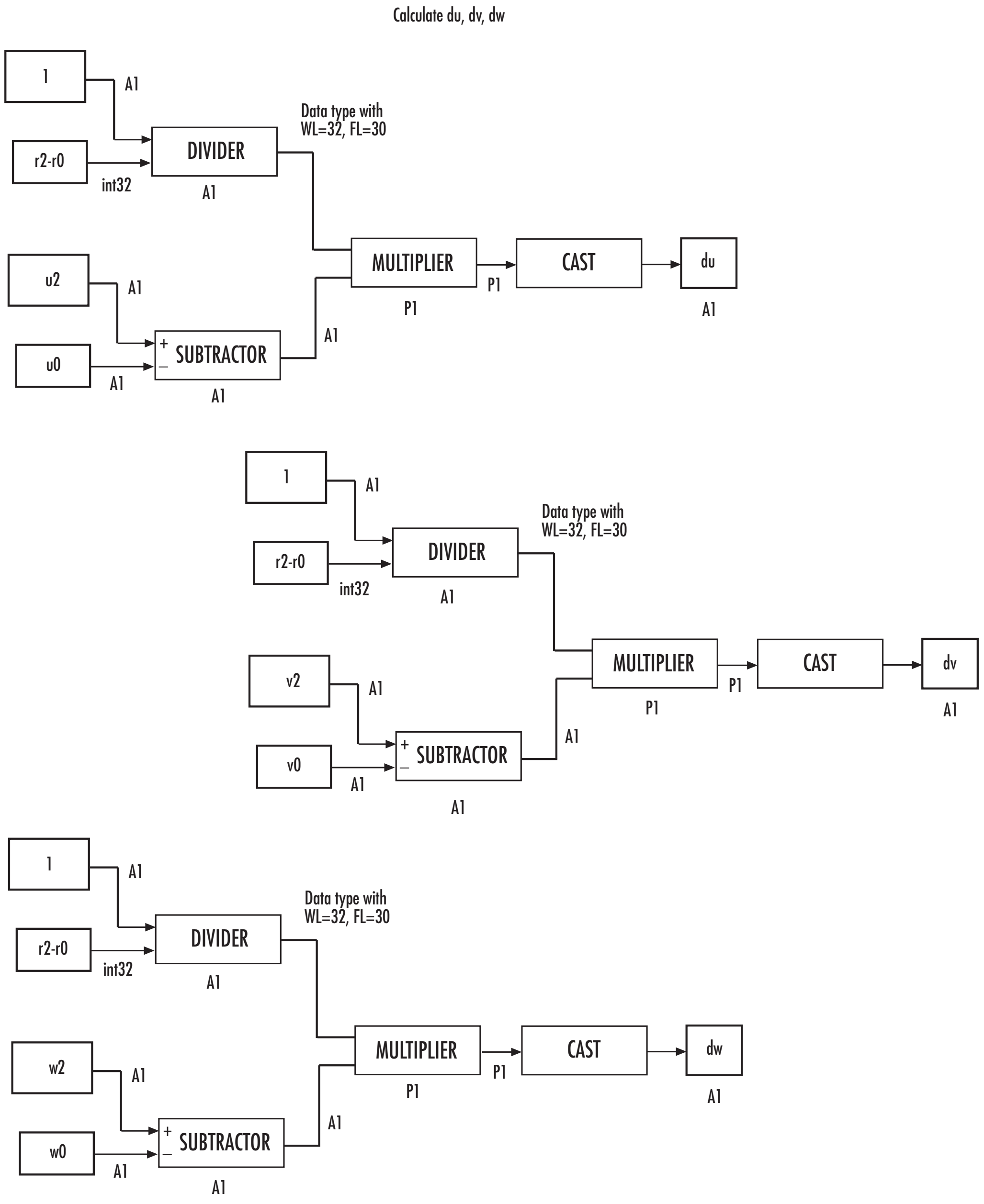

Compute Output Pixel (continued)
Matrix data type
Ai(0)
CAST
A1
Column Index
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
Ai(1)
CAST
A1
r0
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
ADDER
Ai(2)
ADDER
A1
v0


Matrix data type
Ai(0)
CAST
A1
Column Index
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
Ai(1)
CAST
A1
r2
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
ADDER
Ai(2)
ADDER
A1
v2

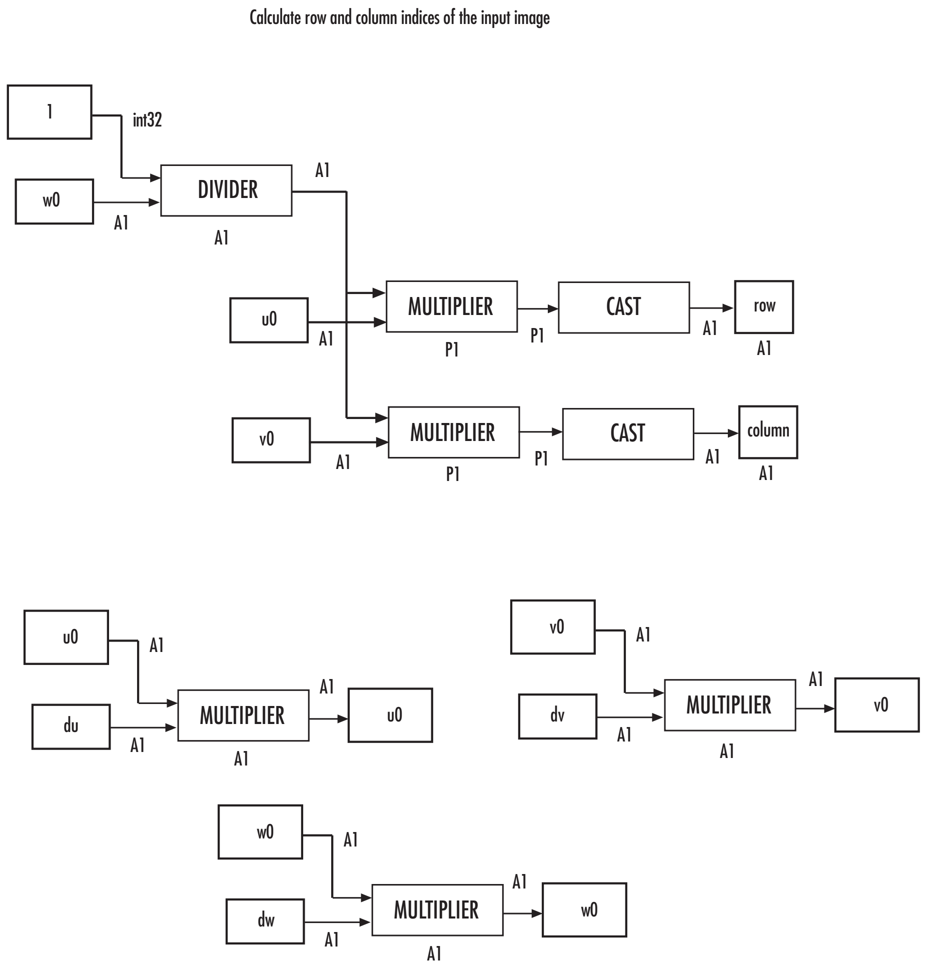
Compute Output Pixel (continued)

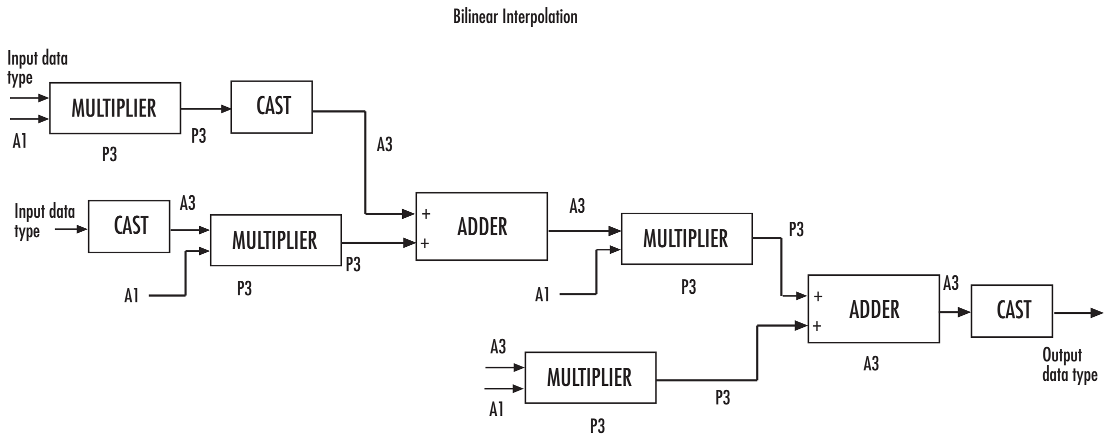






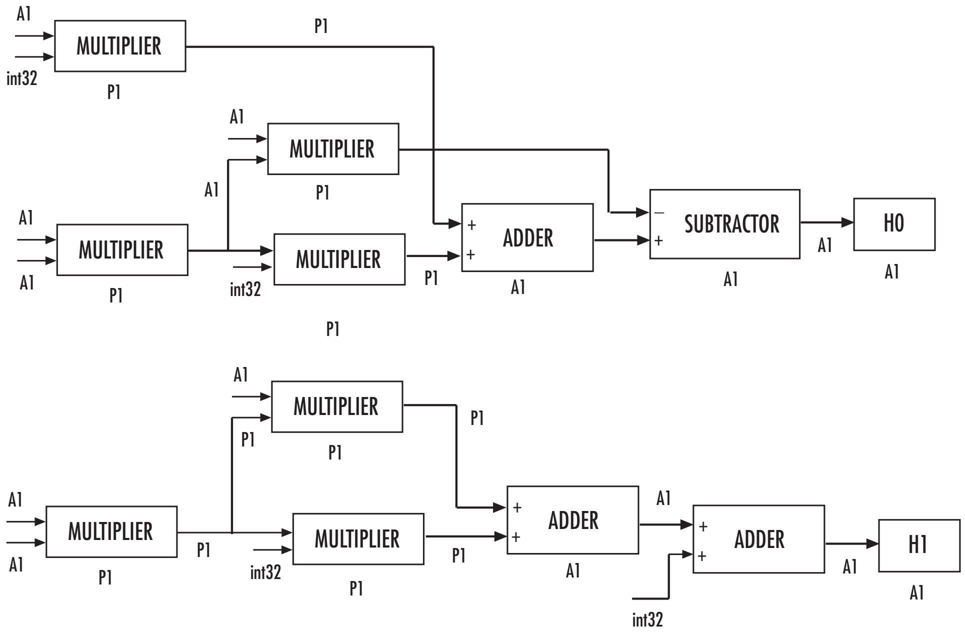
Calculate du, dv, dw
1
A1
r2-r0
int32
DIVIDER
A1
Data type with WL=32, FL=30
u2
A1
u0
A1
SUBTRACTOR
A1
A1
MULTIPLIER
P1
P1
CAST
du
A1
1
A1
r2-r0
int32
DIVIDER
A1
Data type with WL=32, FL=30
v2
A1
v0
A1
SUBTRACTOR
A1
A1
MULTIPLIER
P1
P1
CAST
dv
A1
1
A1
r2-r0
int32
DIVIDER
A1
Data type with WL=32, FL=30
w2
A1
w0
A1
SUBTRACTOR
A1
A1
MULTIPLIER
P1
P1
CAST
dw
A1


Calculate row and column indices of the input image
1
int32
w0
A1
DIVIDER
A1
A1
u0
A1
MULTIPLIER
P1
P1
CAST
A1
row
A1
v0
A1
MULTIPLIER
P1
P1
CAST
A1
column
A1
u0
A1
du
A1
MULTIPLIER
A1
A1
u0
v0
A1
dv
A1
MULTIPLIER
A1
A1
v0
w0
A1
dw
A1
MULTIPLIER
A1
A1
w0


Bilinear Interpolation
Input data type
A1
MULTIPLIER
P3
P3
CAST
A3
Input data type
CAST
A3
MULTIPLIER
P3
P3
A1
ADDER
A3
MULTIPLIER
P3
P3
A1
ADDER
A3
A3
CAST
Output data type
A3
MULTIPLIER
A1
P3
P3

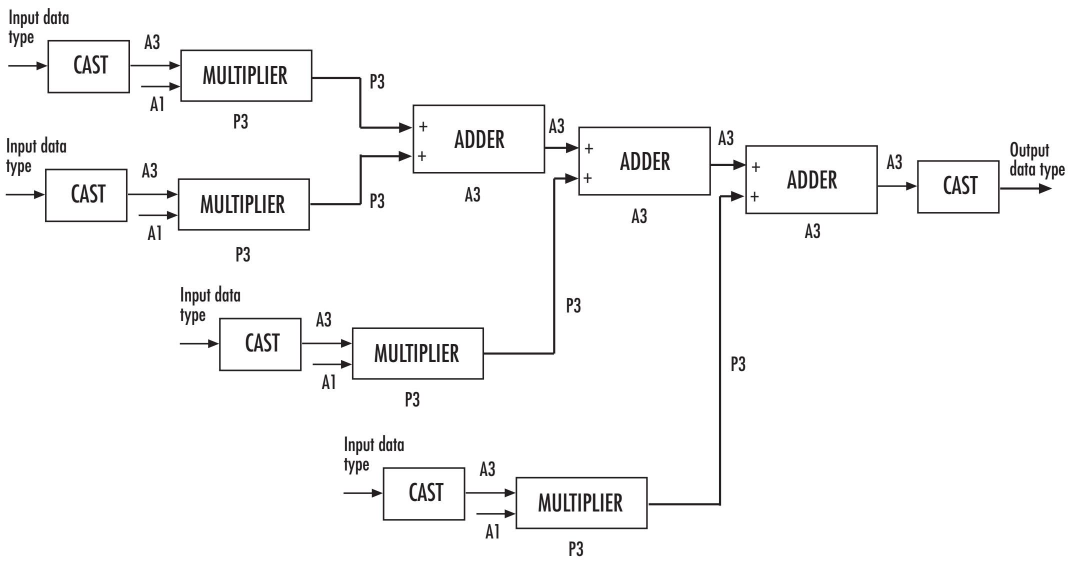
Bicubic Interpolation

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



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

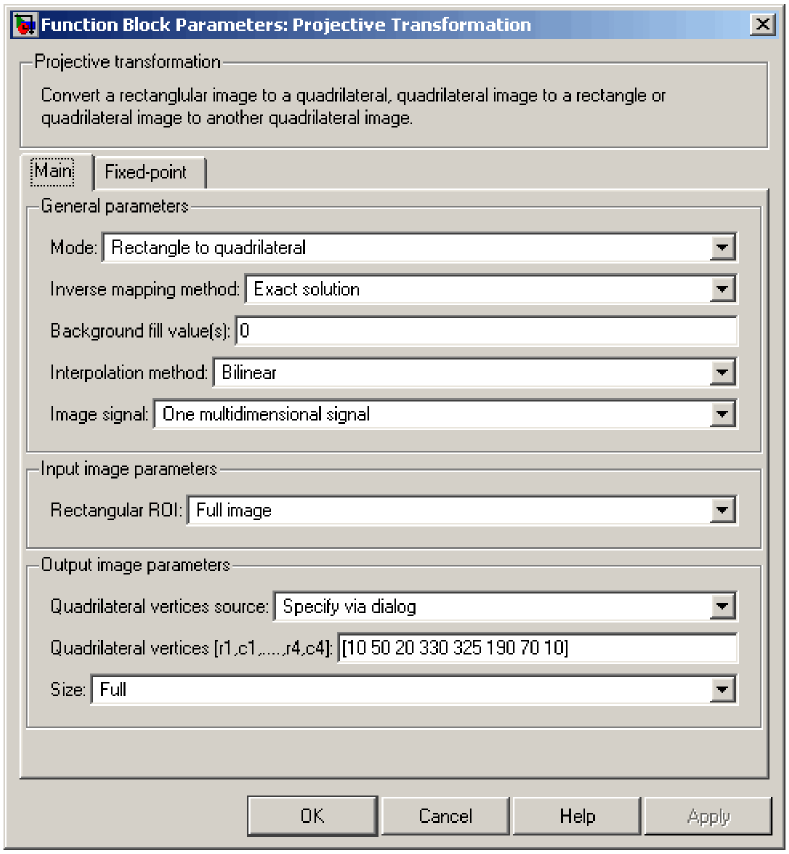
Bicubic Interpolation (continued)



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

Dialog Box

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



Projective transformation

Convert a rectanglular image to a quadrilateral, quadrilateral image to a rectangle or quadrilateral image to another quadrilateral image.

General parameters

Mode: Rectangle to quadrilateral

Inverse mapping method: Exact solution

Background fill value(s): 0

Interpolation method: Bilinear

Image signal: One multidimensional signal

Input image parameters

Rectangular ROI: Full image

Output image parameters

Quadrilateral vertices source: Specify via dialog

Quadrilateral vertices [r1,c1,....,r4,c4]: [10 50 20 330 325 190 70 10]

Size: Full

Mode

Select the shape you want to convert. Your choices are `Rectangle to quadrilateral`, `Quadrilateral to rectangle`, or `Quadrilateral to quadrilateral`.

Inverse mapping method
Specify the algorithm the block uses to implement the projective transformation. Your choices are `Exact solution` or `Quadratic approximation`.

Quality factor (number of subdivisions)
Enter a scalar integer value greater than or equal to 0 and less than or equal to the height or width of the input image, whichever is smaller. The larger the quality factor, the closer the approximate solution is to the exact solution. This parameter is visible if, for the **Inverse mapping method** parameter, you select `Quadratic approximation`. Tunable in some modes.

Background fill value(s)
Set the background of the output image. If the block outputs an intensity image, enter a scalar value. If the block outputs an RGB image, enter a scalar value or a three-element vector that specifies an RGB triplet. Tunable in some modes.

Interpolation method
Specify how the block calculates the pixel intensities in the output image. If you select `Nearest neighbor`, the block uses the value of the nearest pixel for the new pixel intensity. If you select `Bilinear`, the new pixel value is the weighted average of the four nearest pixel intensities. If you select `Bicubic`, the new pixel value is the weighted average of the 16 nearest pixel intensities.

Image signal
Specify how to input and output a color video signal. If you select `One multidimensional signal`, the block accepts an M-by-N-by-P color video signal, where P is the number of color planes, at one port. If you select `Separate color signals`, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

Rectangular ROI
Define the portion of the input image that the block transforms into a quadrilateral. Your choices are `Full image` or `User-defined`.

Rectangular ROI source

Specify whether you want to define the ROI using the Projective Transformation dialog box or an input port. This parameter is visible if, for the **Rectangular ROI** parameter, you select `User-defined`.

ROI [r,c,height,width]

Enter the row and column coordinates of the upper-left corner as well as the height and width of the ROI. This parameter is visible if, for the **Rectangular ROI source** parameter, you select `Specify via dialog`. Tunable in some modes.

If ROI is invalid

Specify the block's behavior if the four-element vector input to the InROI port contains values that are outside the input image. Your choices are `Clip`, `Clip and warn`, or `Error`. During code generation with Real-Time Workshop, this parameter is automatically set to `Clip`. This parameter is visible if, for the **Rectangular ROI source** parameter, you select `Input port`.

Quadrilateral vertices source

Specify how to define the quadrilateral vertices. Your choices are `Specify via dialog` or `Input port`.

Quadrilateral vertices [r1,c1,...,r4,c4]

Enter an eight-element vector of values that represent the row and column coordinates of the four corners of the quadrilateral. This parameter is visible if, for the **Quadrilateral vertices source** parameter, you select `Specify via dialog`.

Size

Specify the size of the output image. If you select `Full`, the block output size is determined by the values you enter for the **Quadrilateral vertices [r1,c1,...,r4,c4]** or **Rectangle location and size [r,c,height,width]** parameter. If you select `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`.

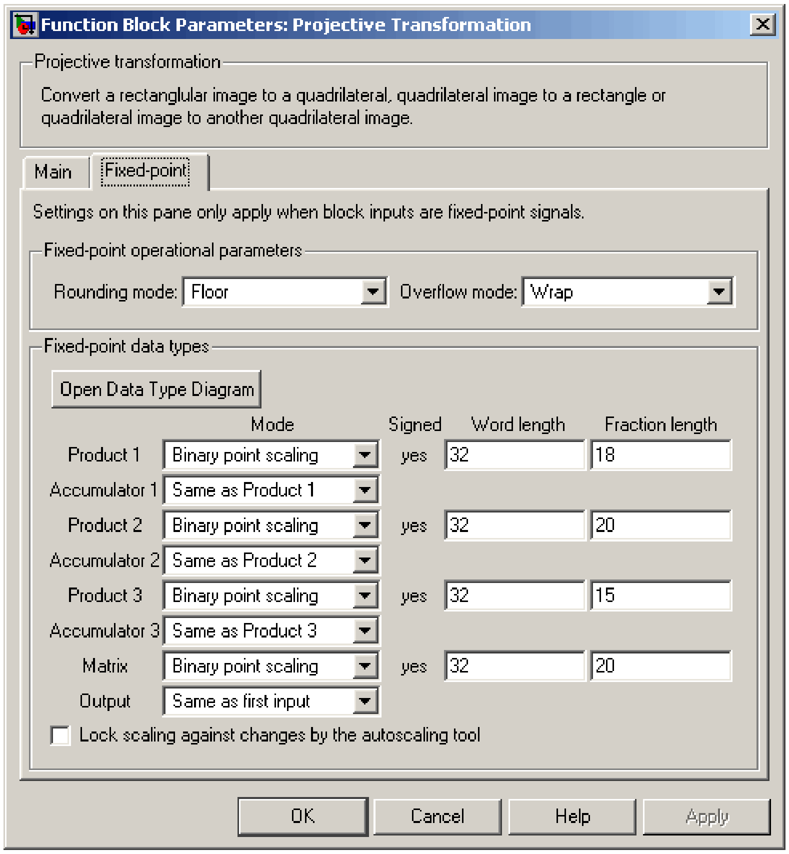
If vertices are outside input image
: Specify the block's behavior if the input to the InPts port is invalid. Your choices are `Clip`, `Clip and warn`, or `Error`. During code generation with Real-Time Workshop, this parameter is automatically set to `Clip`. This parameter is visible if, for the **Mode** 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.



Function Block Parameters: Projective Transformation

Projective transformation

Convert a rectanglular 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

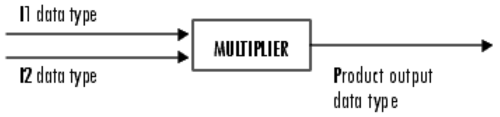
Rounding mode

Select the rounding mode for fixed-point operations. For Boolean input, the **Product 3** and **Accumulator 3Rounding mode** parameter is always set to `Nearest`.

Overflow mode

Select the overflow mode for fixed-point operations.

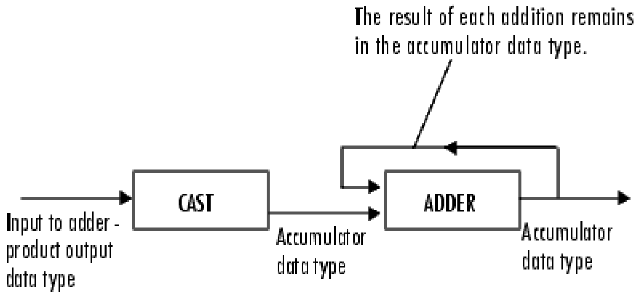
Product 1, 2, 3



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

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

Accumulator 1, 2, 3, 4


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type

As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

- When you select `Same as Product 1, 2, 3`, these characteristics match those of the product output.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator in bits.
- When you select `Slope and bias scaling`, you can enter the word length in bits and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Matrix

Choose how to specify the word length and fraction length of the matrix data type:

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

Output

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

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

The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Lock 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 software
Rotate	Video and Image Processing Blockset software
Shear	Video and Image Processing Blockset software
Translate	Video and Image Processing Blockset software

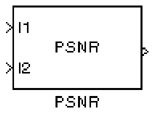
Purpose

Compute peak signal-to-noise ratio (PSNR) between images

Library

Statistics

Description



The PSNR block computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is often used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image.

The *Mean Square Error (MSE)* and the *Peak Signal to Noise Ratio (PSNR)* are the two error metrics used to compare image compression quality. The MSE represents the cumulative squared error between the compressed and the original image, whereas PSNR represents a measure of the peak error. The lower the value of MSE, the lower the error.

To compute the PSNR, the block first calculates the mean-squared error using the following equation:

$$MSE = \frac{\sum_{M,N} [I_1(m,n) - I_2(m,n)]^2}{M * N}$$

In the previous equation, M and N are the number of rows and columns in the input images, respectively. Then the block computes the PSNR using the following equation:

$$PSNR = 10\log_{10}\left(\frac{R^2}{MSE}\right)$$

In the previous equation, R is the maximum fluctuation in the input image data type. For example, if the input image has a double-precision floating-point data type, then R is 1. If it has an 8-bit unsigned integer data type, R is 255, etc.

Recommendation for Computing PSNR for Color Images

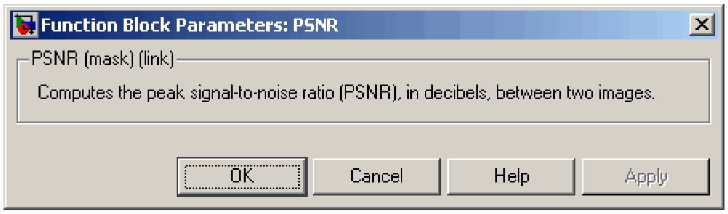
Different approaches exist for computing the PSNR of a color image. Because the human eye is most sensitive to luma information, compute the PSNR for color images by converting the image to a color space that separates the intensity (luma) channel, such as YCbCr. The Y (luma), in YCbCr represents a weighted average of R, G, and B. G is given the most weight, again because the human eye perceives it most easily. With this consideration, compute the PSNR only on the luma channel.

Ports

Port	Output	Supported Data Types	Complex Values Supported
I1	Scalar, vector, or matrix of intensity values	• 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	• Double-precision floating point 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.	No

Dialog Box

The PSNR dialog box appears as shown in the following figure.



PSNR (mask) (link)

Computes the peak signal-to-noise ratio (PSNR), in decibels, between two images.

Read AVI File (Obsolete)

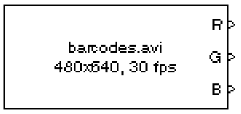
Purpose

Read uncompressed video frames from AVI file

Library

`vipobslib`

Description



Read AVI File

Note The Read AVI File block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset blocks. Use the replacement block From Multimedia File.

The Read AVI File block reads video frames from an uncompressed AVI file and import them into a Simulink model. You can view the video frames using a To Video Display block or Video Viewer block. This block does not support audio samples. Also, this block is supported for simulation only. It produces an error during Real-Time Workshop code generation.

The output ports of the Read AVI File block change according the content of the AVI file. If the file contains RGB video frames, the R, G, and B ports appear on the block. If the file contains intensity video frames, the I port appears on the block.

Port	Output	Supported Data Types	Complex Values Supported
I	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • 8-, 16- 32-bit signed integer • 8-, 16- 32-bit unsigned integer	No

Port	Output	Supported Data Types	Complex Values Supported
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.

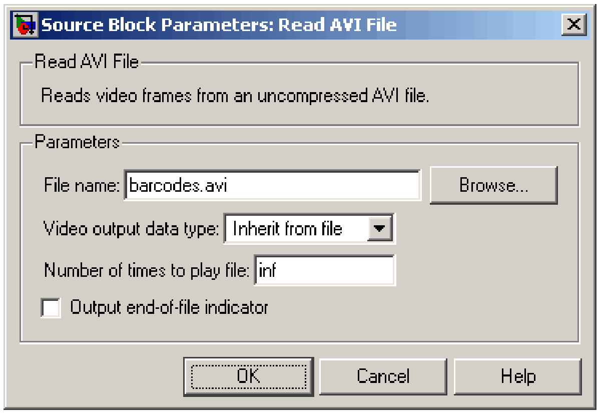
Use the **Video output data type** parameter to set the data type of the values output from the block. You can choose `double`, `single`, `int8`, `uint8`, `int16`, `uint16`, `int32`, `uint32`, and `Inherit from file`. If you choose `double` or `single`, the block scales the input pixels values and outputs 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.

Dialog Box

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



Source Block Parameters: Read AVI File

Read AVI File

Reads video frames from an uncompressed AVI file.

Parameters

File name: barcodes.avi

Video output data type: Inherit from file

Number of times to play file: inf

☐ Output end-of-file indicator

File name
: Specify the name of the AVI file from which to read.

Video output data type
: Set the data type of the video data output from the block.

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

Output end-of-file indicator
: Use this check box to determine whether the output is the last video frame in the AVI file.

See Also

From Multimedia File	Video and Image Processing Blockset software
Image From File	Video and Image Processing Blockset software
Image From Workspace	Video and Image Processing Blockset software
To Multimedia File	Signal Processing Blocksetsoftware
To Video Display	Video and Image Processing Blockset software
Video From Workspace	Video and Image Processing Blockset software
Video Viewer	Video and Image Processing Blockset software

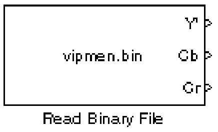
Purpose

Read binary video data from files

Library

Sources

Description



The Read Binary File block reads video data from a binary file and imports it into a Simulink model.

This block takes user specified parameters that describe the format of the video data. These parameters together with the raw binary file, which stores only raw pixel values, creates the video data signal for a Simulink model. The video data read by this block must be stored in row major format.

Note This block supports code generation only for platforms that have file I/O available. You cannot use this block to do code generation with RTWin (Real-Time Windows Target™).

Port	Output	Supported Data Types	Complex Values Supported
Output	Scalar, vector, or matrix of integer values	• 8-, 16- 32-bit signed integer • 8-, 16- 32-bit unsigned integer	No
EOF	Scalar value	Boolean	No

Four Character Code Video Formats

Four Character Codes (FOURCC) identify video formats. For more information about these codes, see `http://www.fourcc.org`.

Use the **Four character code** parameter to identify the binary file format. Then, use the **Rows** and **Cols** parameters to define the size of the output matrix. These dimensions should match the matrix dimensions of the data inside the file.

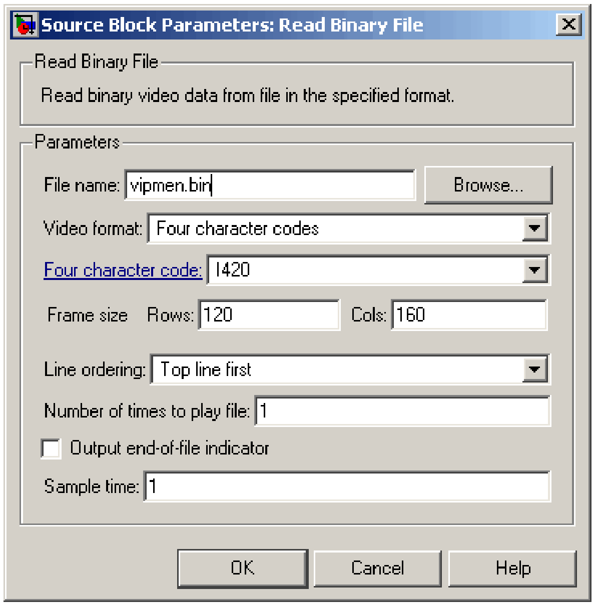
Custom Video Formats

If your binary file contains data that is not in FOURCC format, you can configure the Read Binary File block to understand a custom format:

- Use the **Bit stream format** parameter to specify whether your data is planar or packed. If your data is packed, use the **Rows** and **Cols** parameters to define the size of the output matrix.
- Use the **Number of output components** parameter to specify the number of components in the binary file. This number corresponds to the number of block output ports.
- Use the **Component**, **Bits**, **Rows**, and **Cols** parameters to specify the component name, bit size, and size of the output matrices, respectively. The block uses the **Component** parameter to label the output ports.
- Use the **Component order in binary file** parameter to specify how the components are arranged within the file.
- Select the **Interlaced video** check box if the binary file contains interlaced video data.
- Select the **Input file has signed data** check box if the binary file contains signed integers.
- Use the **Byte order in binary file** to indicate whether your binary file has little endian or big endian byte ordering.

Dialog Box

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



Source Block Parameters: Read Binary File

Read Binary File

Read binary video data from file in the specified format.

Parameters

File name: vipmen.bin

Video format: Four character codes

Four character code: I420

Frame size Rows: 120 Cols: 160

Line ordering: Top line first

Number of times to play file: 1

Output end-of-file indicator

Sample time: 1

File name
: Specify the name of the binary file to read. If the location of this file is on your MATLAB path, enter the filename. If the location of this file is not on your MATLAB path, use the **Browse** button to specify the full path to the file as well as the filename.

Video format
: Specify the format of the binary video data. Your choices are `Four character codes` or `Custom`. See “Four Character Code Video Formats” on page 2-552 or “Custom Video Formats” on page 2-553 for more details.

Four character code
: From the drop-down list, select the binary file format.

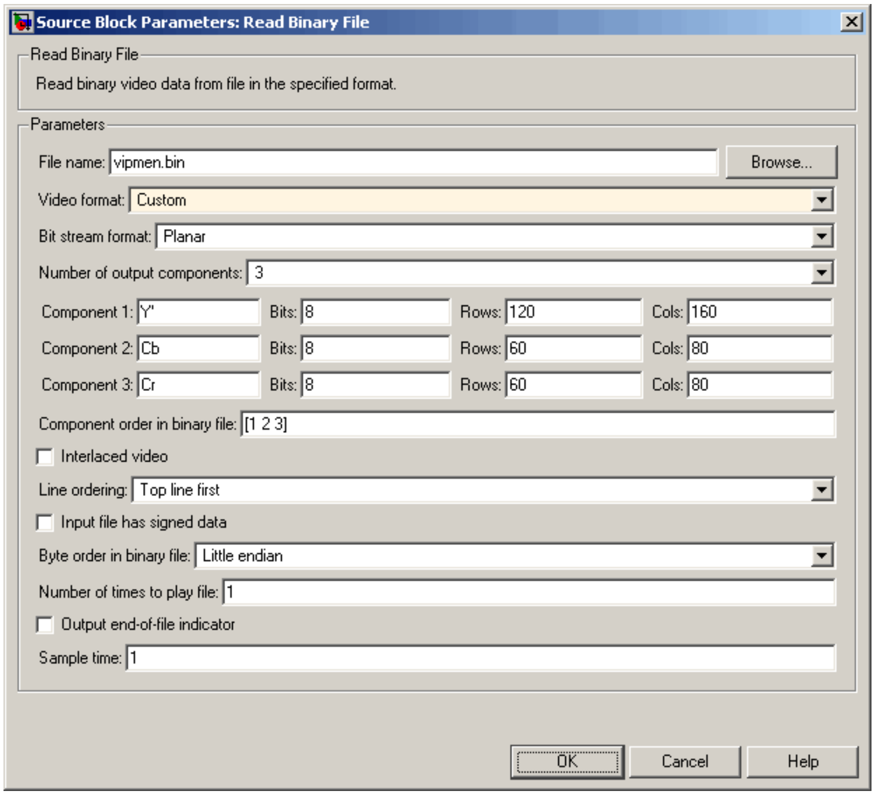
Frame size: Rows, Cols
Define the size of the output matrix. These dimensions should match the matrix dimensions of the data inside the file.

Line ordering
Specify how the block fills the output matrix.If you select `Top line first`, the block first fills the first row of the output matrix with the contents of the binary file. It then fills the other rows in increasing order. If you select `Bottom line first`, the block first fills the last row of the output matrix. It then fills the other rows in decreasing order.

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

Output end-of-file indicator
Specifies the output is the last video frame in the binary file. When you select this check box, a Boolean output port labeled EOF appears on the block. The output from the EOF port is 1 when the last video frame in the binary file is output from the block. Otherwise, the output from the EOF port is 0.

Sample time
Specify the sample period of the output signal.


Source Block Parameters: Read Binary File

Read Binary File

Read binary video data from file in the specified format.

Parameters

File name: vipmen.bin

Video format: Custom

Bit stream format: Planar

Number of output components: 3

Component 1: Y'	Bits: 8	Rows: 120	Cols: 160
Component 2: Cb	Bits: 8	Rows: 60	Cols: 80
Component 3: Cr	Bits: 8	Rows: 60	Cols: 80

Component order in binary file: [1 2 3]

☐ Interlaced video

Line ordering: Top line first

☐ Input file has signed data

Byte order in binary file: Little endian

Number of times to play file: 1

☐ Output end-of-file indicator

Sample time: 1

Bit stream format

Specify whether your data is planar or packed.

Frame size: Rows, Cols

Define the size of the output matrix. This parameter appears when you select a **Bit stream format** parameter of `Packed`.

Number of output components

Specify the number of components in the binary file.

Component, Bits, Rows, Cols

Specify the component name, bit size, and size of the output matrices, respectively.

Component order in binary file
: Specify the order in which the components appear in the binary file.

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

Input file has signed data
: Select this check box if the binary file contains signed integers.

Byte order in binary file
: Use this parameter to indicate whether your binary file has little endian or big endian byte ordering.

See Also

From Multimedia File	Video and Image Processing Blockset
Write Binary File	Video and Image Processing Blockset

Resize

Purpose

Enlarge or shrink image sizes

Library

Geometric Transformations

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	• 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	• 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< resize factor <100, the block shrinks the image. For resize factor =100, the block does not change the image. For 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** 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.

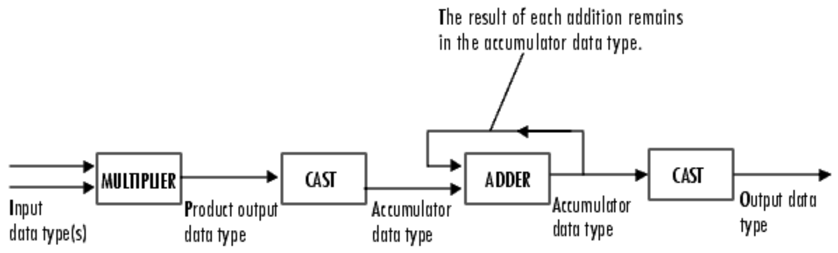
If you select the **Enable ROI processing** check box, the ROI port appears on the block. Use this port to define a region of interest (ROI) in the input matrix, I, that you want to resize. The input to this port must be a four-element vector, [row column height width]. The first two elements define 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

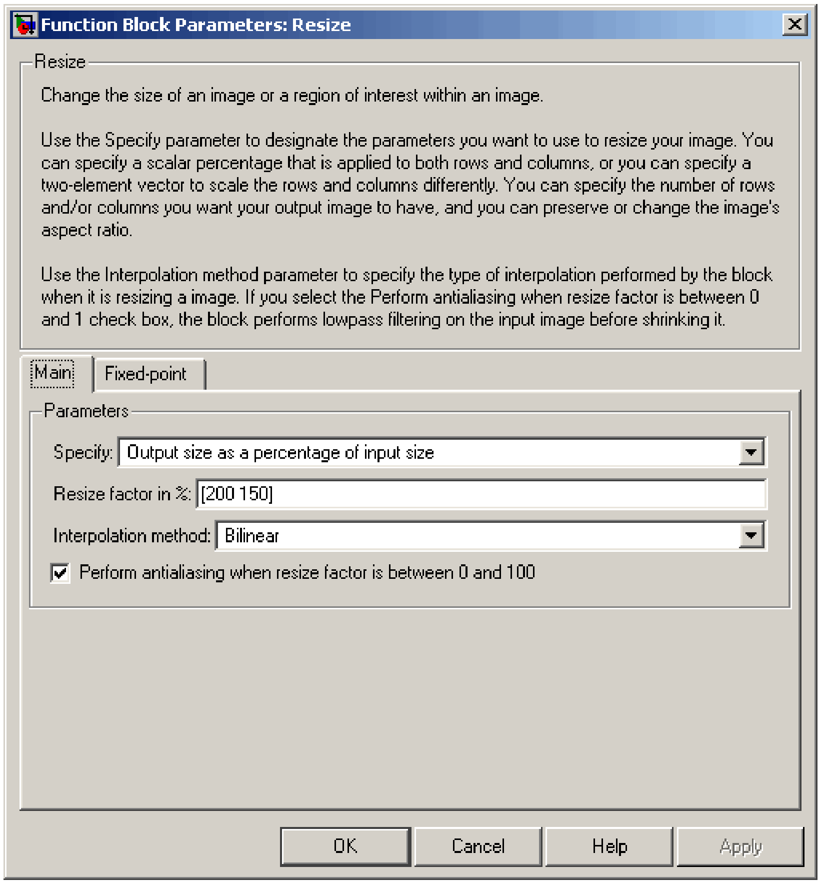
The following diagram shows the data types used in the Resize block for fixed-point signals.



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

Dialog Box

The **Main** pane of the Resize dialog box appears as shown in the following figure:



Resize

Change the size of an image or a region of interest within an image.

Use the Specify parameter to designate the parameters you want to use to resize your image. You can specify a scalar percentage that is applied to both rows and columns, or you can specify a two-element vector to scale the rows and columns differently. You can specify the number of rows and/or columns you want your output image to have, and you can preserve or change the image's aspect ratio.

Use the Interpolation method parameter to specify the type of interpolation performed by the block when it is resizing a image. If you select the Perform antialiasing when resize factor is between 0 and 1 check box, the block performs lowpass filtering on the input image before shrinking it.

Specify: Output size as a percentage of input size

Resize factor in %: [200 150]

Interpolation method: Bilinear

☑ Perform antialiasing when resize factor is between 0 and 100

Specify

Specify which aspects of the image to resize. Your choices are `Output size as a percentage of input size`, `Number of output columns and preserve aspect ratio`, `Number of` `output rows and preserve aspect ratio`, `Number of output rows and columns`.

Resize factor in %
Enter a scalar percentage value that is applied to both rows and columns or a two-element vector, where the first element is the percentage by which to resize the rows and the second element is the percentage by which to resize the columns. This parameter is visible if, for the **Specify** parameter, you select `Output size as a percentage of input size`.

Number of output columns
Enter a scalar value that represents the number of columns you want the output image to have. This parameter is visible if, for the **Specify** parameter, you select `Number of output columns and preserve aspect ratio`.

Number of output rows
Enter a scalar value that represents the number of rows you want the output image to have. This parameter is visible if, for the **Specify** parameter, you select `Number of output rows and preserve aspect ratio`.

Number of output rows and columns
Enter a two-element vector, where the first element is the number of rows in the output image and the second element is the number of columns. This parameter is visible if, for the **Specify** parameter, you select `Number of output rows and columns`.

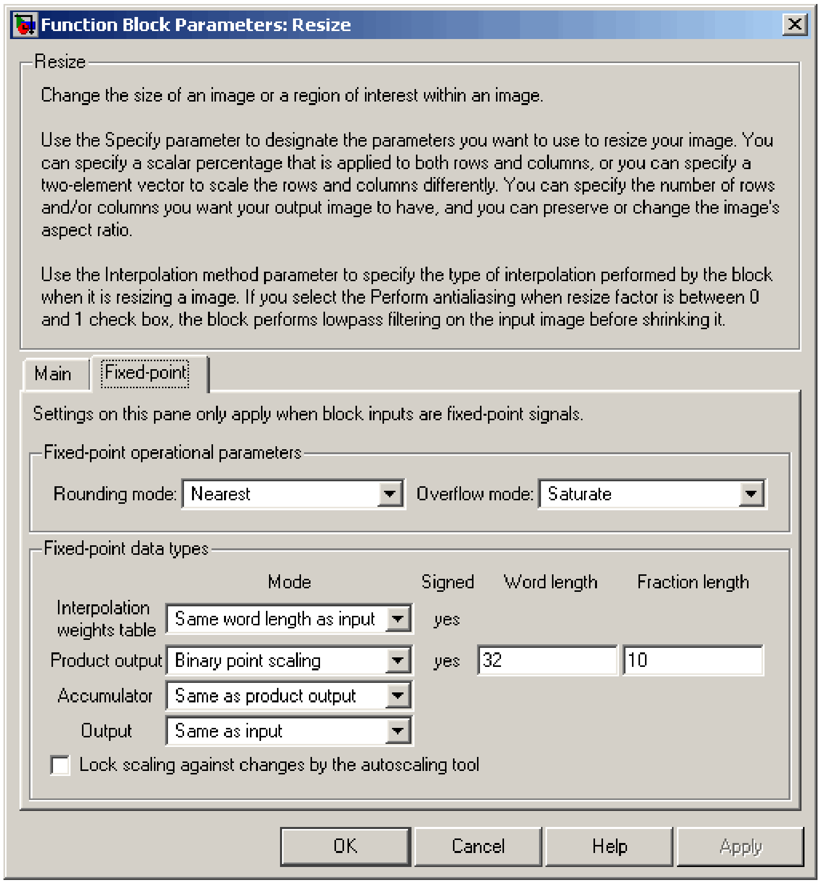
Interpolation method
Determine which interpolation method the block uses to resize the image. If you select `Nearest neighbor`, the block uses one nearby pixel to interpolate the pixel value. If you select `Bilinear`, the block uses two nearby pixels to interpolate the pixel value. If you select `Bicubic` or `Lanczos2`, the block uses four nearby pixels to interpolate the pixel value. If you select `Lanczos3`, the block uses six surrounding pixels to interpolate the pixel value.

Perform antialiasing when resize factor is between 0 and 100
If you select this check box, the block performs low-pass filtering on the input image before shrinking it to prevent aliasing.

Enable ROI processing
Select this check box to resize a particular region of each image. This parameter is available when the **Specify** parameter is set to `Number of output rows and columns`, the **Interpolation method** parameter is set to `Nearest neighbor`, `Bilinear`, or `Bicubic`, and the **Perform antialiasing when resize factor is between 0 and 100** check box is not selected.

Output flag indicating if any part of ROI is outside image bounds
If you select this check box, the Flag port appears on the block. The block outputs 1 at this port if the ROI is completely or partially outside the input image. Otherwise, it outputs 0.

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


Function Block Parameters: Resize

Resize

Change the size of an image or a region of interest within an image.

Use the Specify parameter to designate the parameters you want to use to resize your image. You can specify a scalar percentage that is applied to both rows and columns, or you can specify a two-element vector to scale the rows and columns differently. You can specify the number of rows and/or columns you want your output image to have, and you can preserve or change the image's aspect ratio.

Use the Interpolation method parameter to specify the type of interpolation performed by the block when it is resizing a image. If you select the Perform antialiasing when resize factor is between 0 and 1 check box, the block performs lowpass filtering on the input image before shrinking it.

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

Fixed-point operational parameters

Rounding mode: Nearest Overflow mode: Saturate

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Interpolation weights table	Same word length as input	yes		
Product output	Binary point scaling	yes	32	10
Accumulator	Same as product output			
Output	Same as input			

☐ Lock scaling against changes by the autoscaling tool

Rounding mode
Select the rounding mode for fixed-point operations.

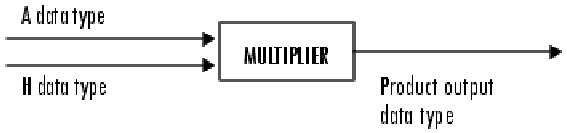
Overflow mode
Select the overflow mode for fixed-point operations.

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.

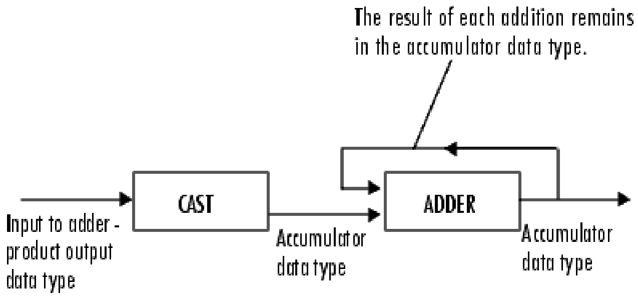
Product output



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 the Video and Image Processing Blockset blocks is 0.

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type

As depicted in the preceding diagram, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

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

Output

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

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

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

References

[1] Ward, Joseph and David R. Cok. "Resampling Algorithms for Image Resizing and Rotation", *Proc. SPIE Digital Image Processing Applications*, vol. 1075, pp. 260-269, 1989.

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

See Also

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

Rotate

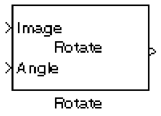
Purpose

Rotate image by specified angle

Library

Geometric Transformations

Description



Use the Rotate block to rotate an image by an angle specified in radians.

Note This block supports intensity and color images on its ports.

Port	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	• 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 appears 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

π

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.

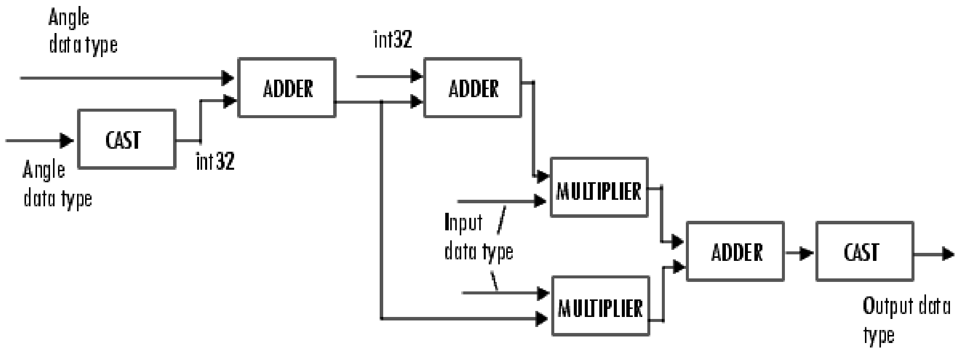
Use the **Interpolation method** parameter to specify which interpolation method the block uses to rotate the image. If you select `Nearest neighbor`, the block uses the value of the nearest pixel for the new pixel value. If you select `Bilinear`, the new pixel value is the weighted average of the four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

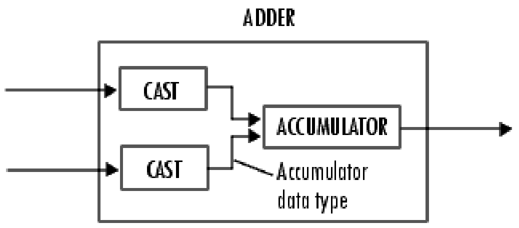
The number of pixels the block considers affects the complexity of the computation. Therefore, the nearest-neighbor interpolation is the most computationally efficient. However, because the accuracy of the method is proportional to the number of pixels considered, the bicubic method is the most accurate. 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 the Image Processing Toolbox. See page 208 of [1] for more information about this algorithm.

Fixed-Point Data Types

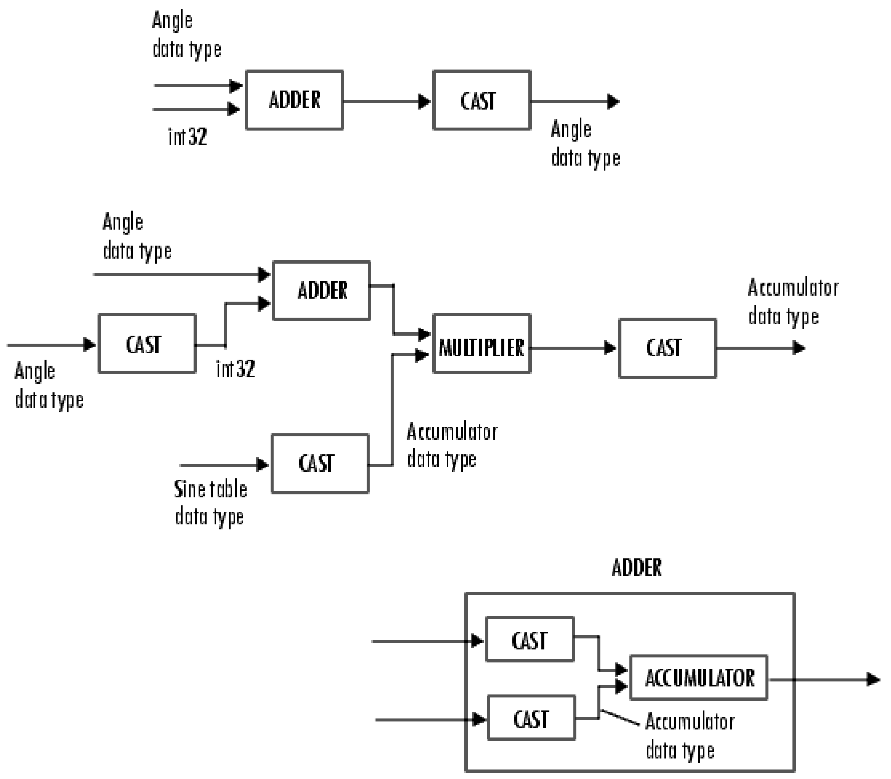
The following diagram shows the data types used in the Rotate block for bilinear interpolation of fixed-point signals.





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

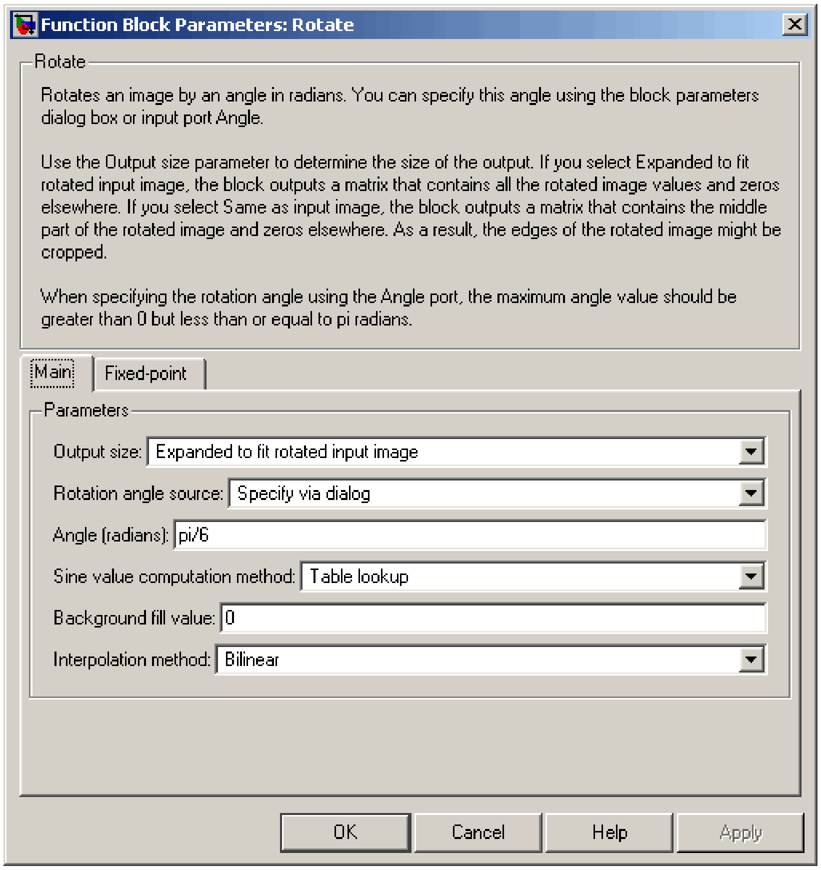
The Rotate block requires additional data types. The Sine table value has the same word length as the angle data type and a fraction length that is equal to its word length minus one. The following diagram shows how these data types are used inside the block.



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.



Function Block Parameters: Rotate

Rotate

Rotates an image by an angle in radians. You can specify this angle using the block parameters dialog box or input port Angle.

Use the Output size parameter to determine the size of the output. If you select Expanded to fit rotated input image, the block outputs a matrix that contains all the rotated image values and zeros elsewhere. If you select Same as input image, the block outputs a matrix that contains the middle part of the rotated image and zeros elsewhere. As a result, the edges of the rotated image might be cropped.

When specifying the rotation angle using the Angle port, the maximum angle value should be greater than 0 but less than or equal to pi radians.

Main | Fixed-point

Parameters

Output size: Expanded to fit rotated input image

Rotation angle source: Specify via dialog

Angle (radians): pi/6

Sine value computation method: Table lookup

Background fill value: 0

Interpolation method: Bilinear

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.

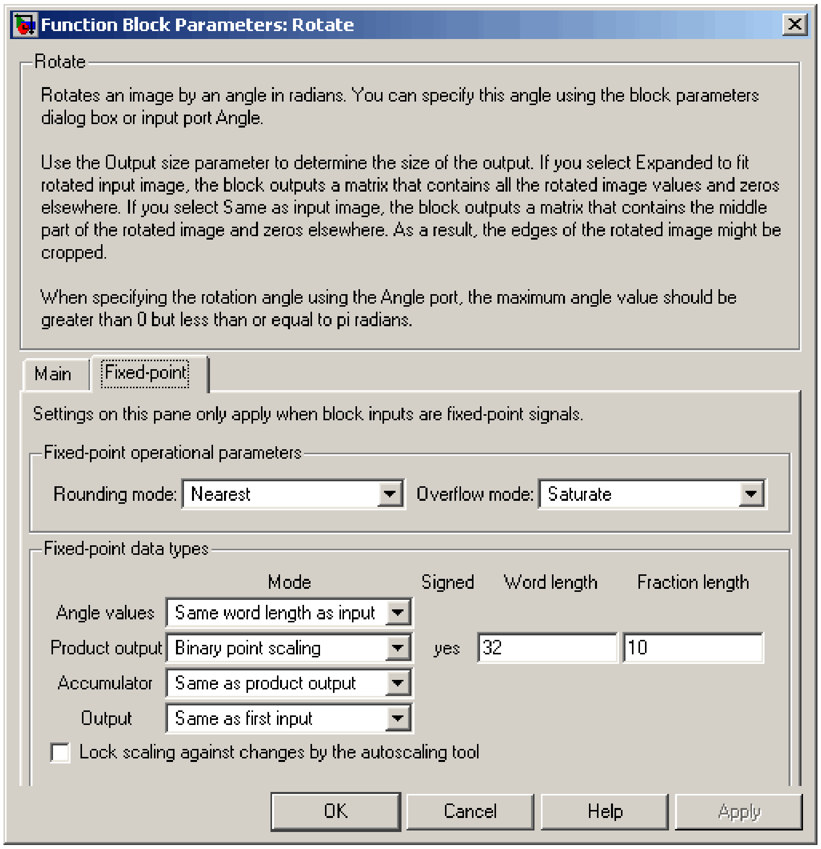
Background fill value

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

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



Function Block Parameters: Rotate

Rotate

Rotates an image by an angle in radians. You can specify this angle using the block parameters dialog box or input port Angle.

Use the Output size parameter to determine the size of the output. If you select Expanded to fit rotated input image, the block outputs a matrix that contains all the rotated image values and zeros elsewhere. If you select Same as input image, the block outputs a matrix that contains the middle part of the rotated image and zeros elsewhere. As a result, the edges of the rotated image might be cropped.

When specifying the rotation angle using the Angle port, the maximum angle value should be greater than 0 but less than or equal to pi radians.

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

Fixed-point operational parameters

Rounding mode: Nearest   Overflow mode: Saturate

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Angle values	Same word length as input			
Product output	Binary point scaling	yes	32	10
Accumulator	Same as product output			
Output	Same as first input			

☐ Lock scaling against changes by the autoscaling tool

Rounding mode
Select the rounding mode for fixed-point operations.

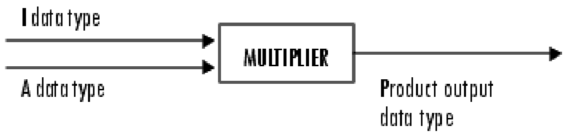
Overflow mode
Select the overflow mode for fixed-point operations.

Angle values
Choose how to specify the word length and the fraction length of the angle values.

- When you select `Same word length as input`, the word length of the angle values match that of the input to the block. In this mode, the fraction length of the angle values is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the angle values.
- When you select `Specify word length`, you can enter the word length of the angle values, in bits. The block automatically sets the fraction length to give you the best precision.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the angle values, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the angle values. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

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

Product output

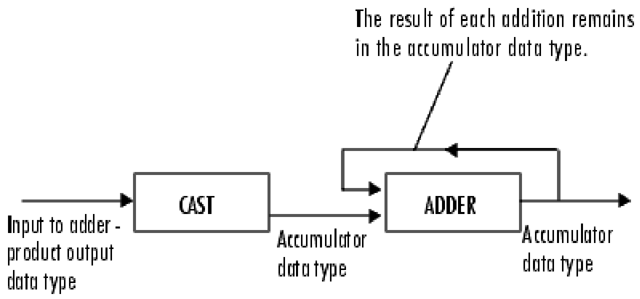


As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths.

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

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

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type

As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

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

bias of all signals in the Video and Image Processing Blockset blocks is 0.

Output

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

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

References

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

See Also

Resize	Video and Image Processing Blockset software
Translate	Video and Image Processing Blockset software
Shear	Video and Image Processing Blockset software
`imrotate`	Image Processing Toolbox software

Purpose

Perform 2-D sum of absolute differences (SAD)

Library

Analysis & Enhancement

Description

I
Template
SAD
Idx
SAD

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 (Mi, Ni) and the input matrix Template has dimensions (Mt, Nt). The equation for the two-dimensional discrete SAD is

$$C(j,k) = \sum_{m=0}^{(Mt-1)} \sum_{n=0}^{(Nt-1)} abs(I(m+j, n+k) - T(m,n))$$

where

$$0 \le j < Mi - Mt + 1$$

and

$$0 \le k < Ni - Nt + 1$$

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Matrix of intensity values	• 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

Port	Input/Output	Supported Data Types	Complex Values Supported
ROI	Four-element vector that defines the ROI	• 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	• 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	• 32-bit signed integers	No
NVals	N-by-N matrix of SAD values centered around the minimum SAD value	Same as Val port	No
NValid	Boolean 0 or 1 that represents whether or not the block went beyond the dimensions of the SAD value matrix to construct an N-by-N matrix around the minimum SAD value	Boolean	No

The data type of the two input signals must be the same. The output signal is the same data type as the input signals.

The dimensions of the output at the Val port are determined by the sizes of the inputs at ports I and Template. If the input at port I has dimensions (Mi, Ni) and the input at the Template port dimensions (Mt, Nt), then the output has dimensions (Mi-Mt+1, Ni-Nt+1).

Use the **Output** parameter to determine the output of the block. If you select `SAD values`, the block outputs the SAD values at the Val port. If you select `Minimum SAD value index`, the block outputs the zero-based index location of the minimum SAD value at the Idx port.

If, for the **Output** parameter, you select `Minimum SAD value index`, the **Search method** parameter appears in the dialog box. If you select `Exhaustive`, the block searches the two input matrices for the minimum difference pixel-by-pixel. This process is described by the previous equation and is computationally expensive.

If, for the **Search method** parameter, you select `Three-step`, the block searches the two input matrices for the minimum difference using a steadily decreasing step size. The block begins with a step size approximately equal to half the maximum search range. In each step, the block compares the central point of the search region to eight search points located on the boundaries of the region and moves the central point to the search point whose values is the closest to that of the central point. The block then 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

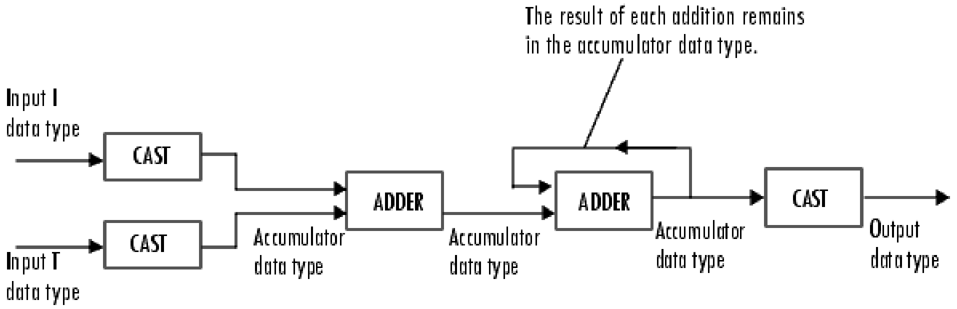
- `Ignore` – Proceed with the computation and do not issue an alert. The output is not valid.
- `Warn` – Display a warning message in the MATLAB Command Window, and continue the simulation. The output is not valid.
- `Error` – Display an error dialog box and terminate the simulation.

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

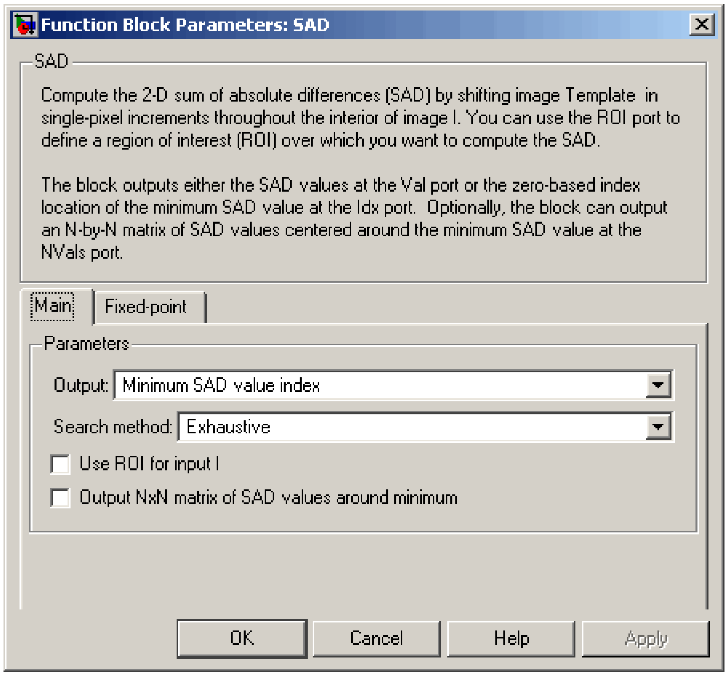
The following diagram shows the data types used in the SAD block for fixed-point signals.



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.



Compute the 2-D sum of absolute differences (SAD) by shifting image Template in single-pixel increments throughout the interior of image I. You can use the ROI port to define a region of interest (ROI) over which you want to compute the SAD.

The block outputs either the SAD values at the Val port or the zero-based index location of the minimum SAD value at the Idx port. Optionally, the block can output an N-by-N matrix of SAD values centered around the minimum SAD value at the NVals port.

Output: Minimum SAD value index

Search method: Exhaustive

☐ Use ROI for input I

☐ Output NxN matrix of SAD values around minimum

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`.

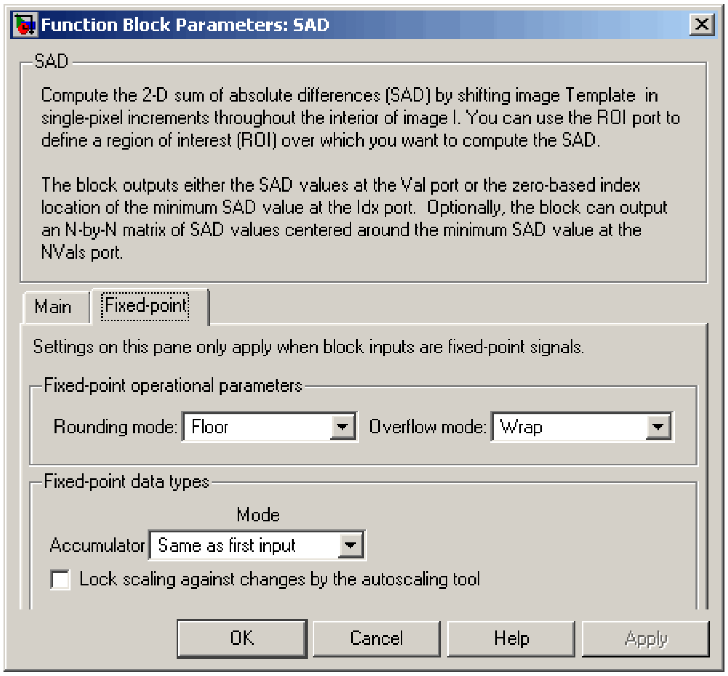
Output NxN matrix of SAD values around minimum

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.



SAD

Compute the 2-D sum of absolute differences (SAD) by shifting image Template in single-pixel increments throughout the interior of image I. You can use the ROI port to define a region of interest (ROI) over which you want to compute the SAD.

The block outputs either the SAD values at the Val port or the zero-based index location of the minimum SAD value at the Idx port. Optionally, the block can output an N-by-N matrix of SAD values centered around the minimum SAD value at the NVals port.

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

Mode

Accumulator: Same as first input

☐ Lock scaling against changes by the autoscaling tool

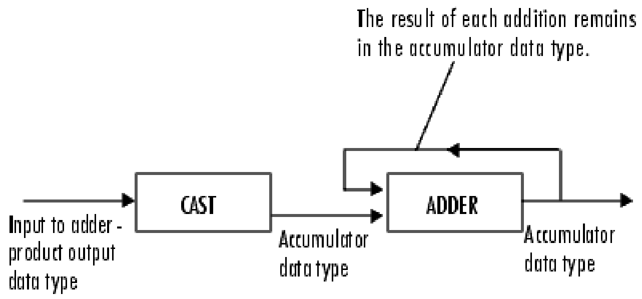
Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type

As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

- When you select `Same as first input`, these characteristics match those of the input to the block. When you have a Boolean input, you cannot select this choice.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

Output

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

- When you select `Same as first input`, these characteristics match those of the first input to the block. When you have a Boolean input, you cannot select this choice.

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

This parameter is not visible if, for the **Output** parameter you select `Minimum SAD value index`, and you clear the **Output NxN matrix of SAD values around minimum** check box.

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

References

[1] Koga, T., et al. *Motion-compensated interframe coding for video conferencing*. In Nat. Telecommun. Conf., Nov. 1981, G5.3.1-5, New Orleans, LA.

[2] Wang, Yao, Jorn Ostermann, Ya-Qin Zhang. *Video Processing and Communications*. Upper Saddle River, NJ: Prentice Hall, 2002.

Shear

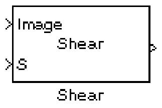
Purpose

Shift rows or columns of image by linearly varying offset

Library

Geometric Transformations

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	• 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.

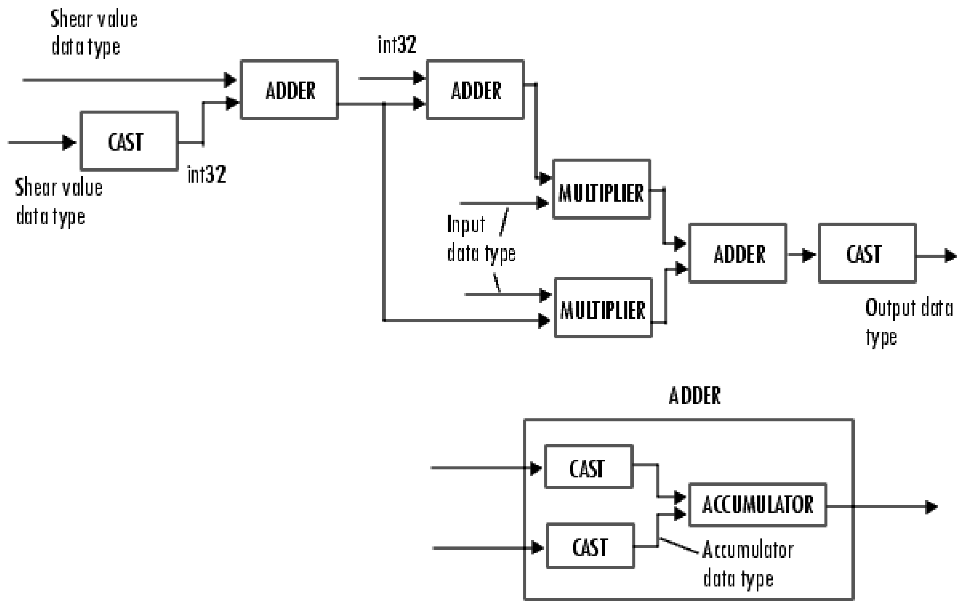
The number of pixels the block considers affects the complexity of the computation. Therefore, the nearest-neighbor interpolation is the most computationally efficient. However, because the accuracy of the method is proportional to the number of pixels considered, the bicubic method is the most accurate.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

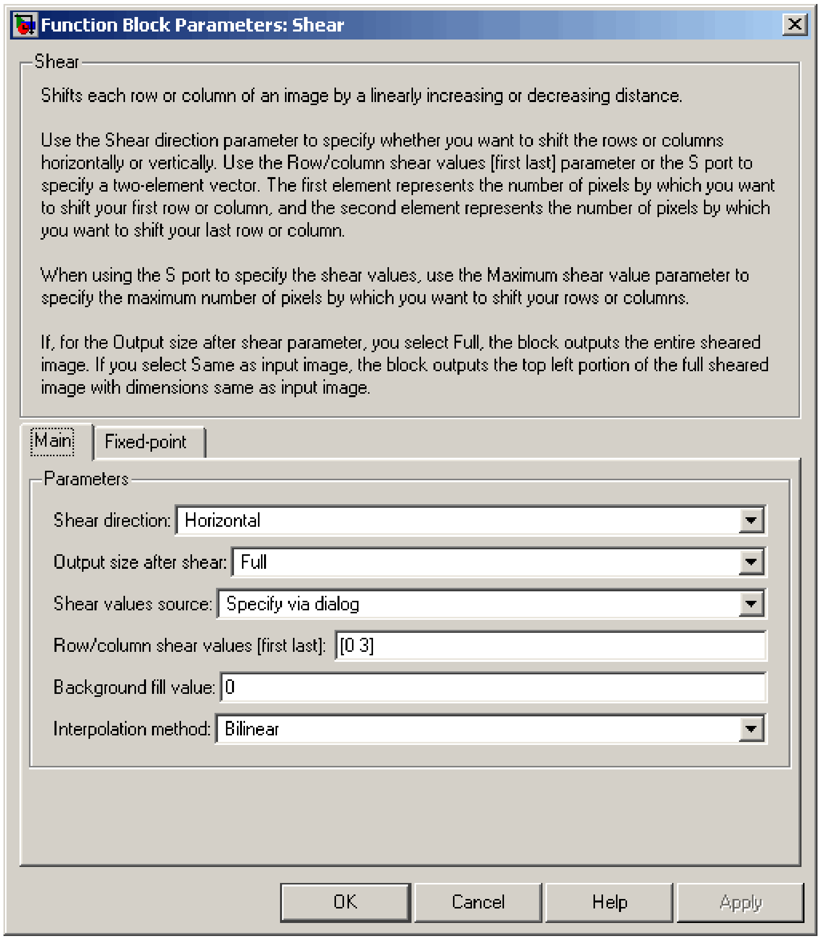
The following diagram shows the data types used in the Shear block for bilinear interpolation of fixed-point signals.



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

Dialog Box

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



Shifts each row or column of an image by a linearly increasing or decreasing distance.

Use the Shear direction parameter to specify whether you want to shift the rows or columns horizontally or vertically. Use the Row/column shear values [first last] parameter or the S port to specify a two-element vector. The first element represents the number of pixels by which you want to shift your first row or column, and the second element represents the number of pixels by which you want to shift your last row or column.

When using the S port to specify the shear values, use the Maximum shear value parameter to specify the maximum number of pixels by which you want to shift your rows or columns.

If, for the Output size after shear parameter, you select Full, the block outputs the entire sheared image. If you select Same as input image, the block outputs the top left portion of the full sheared image with dimensions same as input image.

Shear direction: Horizontal
Output size after shear: Full
Shear values source: Specify via dialog
Row/column shear values [first last]: [0 3]
Background fill value: 0
Interpolation method: Bilinear

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.

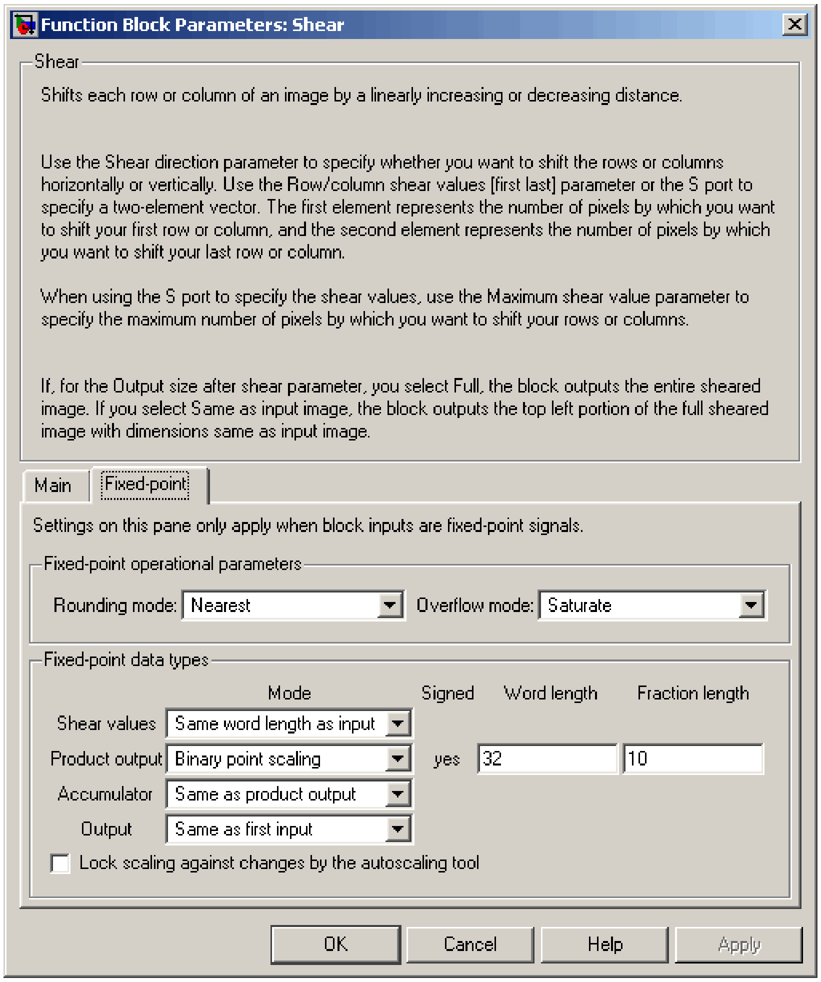
Row/column shear values [first last]
Enter a two-element vector that represents the number of pixels by which to shift your first and last rows or columns. This parameter is visible if, for the **Shear values source** parameter, you select `Specify via dialog`.

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.

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



Shear

Shifts each row or column of an image by a linearly increasing or decreasing distance.

Use the Shear direction parameter to specify whether you want to shift the rows or columns horizontally or vertically. Use the Row/column shear values [first last] parameter or the S port to specify a two-element vector. The first element represents the number of pixels by which you want to shift your first row or column, and the second element represents the number of pixels by which you want to shift your last row or column.

When using the S port to specify the shear values, use the Maximum shear value parameter to specify the maximum number of pixels by which you want to shift your rows or columns.

If, for the Output size after shear parameter, you select Full, the block outputs the entire sheared image. If you select Same as input image, the block outputs the top left portion of the full sheared image with dimensions same as input image.

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

Fixed-point operational parameters

Rounding mode: Nearest   Overflow mode: Saturate

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Shear values	Same word length as input			
Product output	Binary point scaling	yes	32	10
Accumulator	Same as product output			
Output	Same as first input			

☐ Lock scaling against changes by the autoscaling tool

Rounding mode

Select the rounding mode for fixed-point operations.

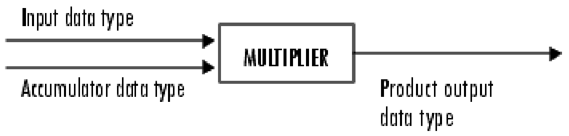
Overflow mode
Select the overflow mode for fixed-point operations.

Shear values
Choose how to specify the word length and the fraction length of the shear values.

- When you select `Same word length as input`, the word length of the shear values match that of the input to the block. In this mode, the fraction length of the shear values is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the shear values.
- When you select `Specify word length`, you can enter the word length of the shear values, in bits. The block automatically sets the fraction length to give you the best precision.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the shear values, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the shear values. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

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

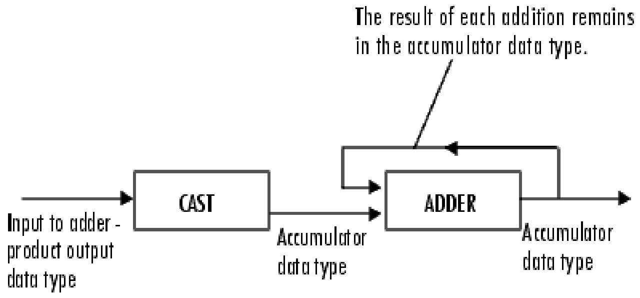
Product output



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

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

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type

As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as first input`, these characteristics match those of the first input to the block at the I port.

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

Output

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

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

References

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

See Also

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

Standard Deviation

Purpose Find standard deviation of each input matrix

Library Statistics

Description The Standard Deviation block is a Signal Processing Blockset block. For more information, see the Standard Deviation block reference page in the Signal Processing Blockset software documentation.

Purpose

Write video frames and audio samples to multimedia file

Library

Sinks

Description

The To Multimedia File block is a Signal Processing Blockset block. For more information, see the To Multimedia File block reference page in the Signal Processing Blockset software documentation.

To Video Display

Purpose

Send video data to display devices

Library

Sinks

Description

Image
To Video Display
To Video Display

The To Video Display block sends video data to your computer screen.

Note This block supports code generation and is only available on Windows platforms that have file I/O available. This excludes RTWin (Real-Time Windows Target). This block performs best on platforms with DirectX Version 9.0 or later and Windows Media Version 9.0 or later.

Input	Description
Image	M-by-N matrix of intensity values or an M-by-N-by-3 color video signal
R, G, B	Matrix that represents one plane of the RGB video stream. Inputs to the R, G, or B ports must have the same dimensions and data type.

For the block to display video data properly, double- and single-precision floating-point pixel values must be from 0 to 1. For any other data type, the pixel values must be between the minimum and maximum values supported by their data type.

Select **Full-screen** to display your video stream in a full-screen window. Use the Esc key to exit or to return to other applications, hold down the **Alt** key and press the **Tab** key. The display window will revert to previous size if other blocks open additional display windows.

Window size and position
Size and position of the display window is saved when the model is saved. Any changes while working with the model should be saved again in order that these preferences are maintained when the model runs.

Host or external monitor visibility

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

1. Replace the To Video Display block with a Video Viewer block.
   or

2. Disable the DirectDraw Acceleration, Direct3D Acceleration, and AGP Texture Acceleration on your system.

   a. **Start > Run**.

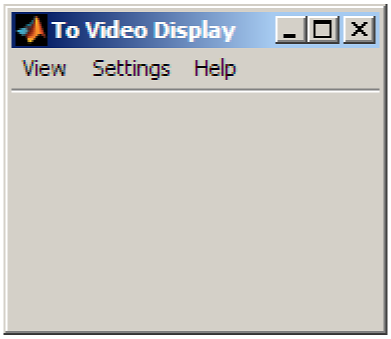
   b. For the **Open** parameter, type `dxdiag`. Click **OK**. The DirectX Diagnostic Tool opens.

   c. On the **Display** tab, click the **Disable** buttons that are next to DirectDraw Acceleration, Direct3D Acceleration, and AGP Texture Acceleration.

   d. Click **Exit**.

Menu Options

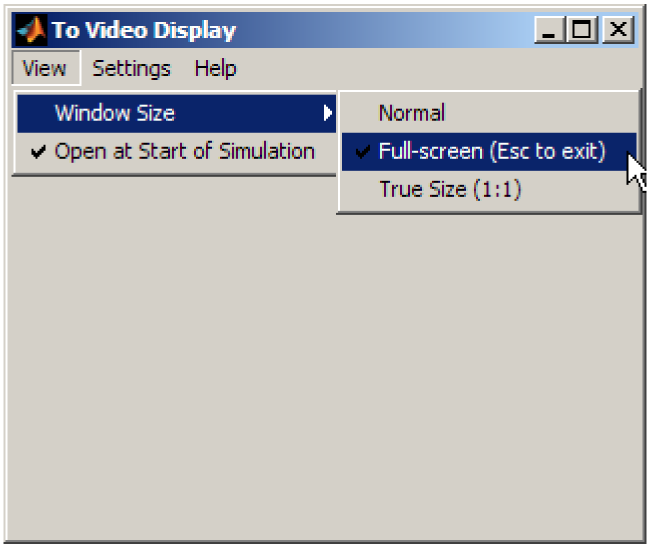
The To Video Display appears as shown in the following figure.



Rapid Accelerator mode

If your model is set to run in Rapid Accelerator mode, the menu options will not be available. In particular, if **Open at Start of Simulation** is unchecked, the block will not be included during the run, and therefore the video display will not be visible. For Rapid Accelerator mode, menu preferences are saved only when the model is compiled. To change any of the menu options, change the model to run in Normal mode, and re-save it. You can then run in Rapid Accelerator mode with the new preferences.

Full-screen



Select `Full-screen` mode to display your video stream in a full screen window.

Open at Start of Simulation

Select `Open at Start of Simulation` from the **View** menu for the display window to appear while running the model. If this is not selected, double click the block to display the window.

Image signal

Select the **Image signal** from the **Settings** menu to specify how the block accepts a color video signal. If you select `One multidimensional signal`, the block accepts an M-by-N-by-3 color video signal at one port. If you select `Separate color signals`, additional ports appear on the block. Each port accepts one M-by-N plane of an RGB video stream.

Preferences

Select **Preferences** from the **Settings** menu to view or modify **Video and Image Processing Blockset Preferences**

Help

The Help menu provides direct links to the To Video Display reference help, Video and Image Processing Blockset reference help, and version information.

Supported Data Types

Port	Supported Data Types
Image	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16, and 32-bit signed integer • 8-, 16, and 32-bit unsigned integer
R, G, B	Same as Image port

See Also

Frame Rate Display	Video and Image Processing Blockset software
From Multimedia File	Video and Image Processing Blockset software
To Multimedia File	Signal Processing Blockset software
Video To Workspace	Video and Image Processing Blockset software
Video Viewer	Video and Image Processing Blockset software

Purpose

Perform top-hat filtering on intensity or binary images

Library

Morphological Operations

Description

I
Top-hat
Nhood
Top-hat

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	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No
Nhood	Matrix or vector of 1s and 0s that represents the neighborhood values	Boolean	No
Output	Scalar, vector, or matrix that represents the filtered image	Same as I port	No

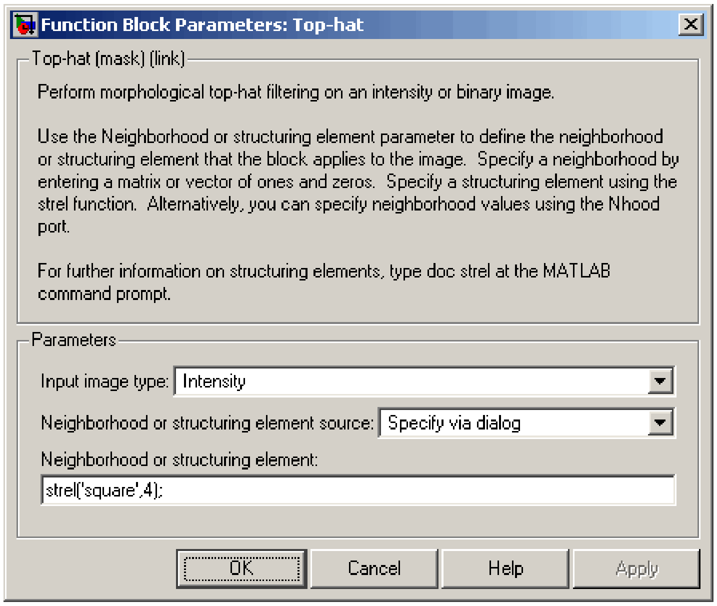
If your input image is a binary image, for the **Input image type** parameter, select `Binary`. If your input image is an intensity image, select `Intensity`.

Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select `Specify via dialog`, the **Neighborhood or structuring element** parameter appears in the dialog box. If you select `Input port`, the Nhood port appears on the block. Use this port to enter your neighborhood values as a matrix or vector of 1s and 0s. Choose your structuring element so that it matches the shapes you want to remove from your image. You can only specify a it using the dialog box.

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

Dialog Box

The Top-hat dialog box appears as shown in the following figure.



Perform morphological top-hat filtering on an intensity or binary image.

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 ones and zeros. Specify a structuring element using the strel function. Alternatively, you can specify neighborhood values using the Nhood port.

For further information on structuring elements, type doc strel at the MATLAB command prompt.

Parameters

Input image type: Intensity

Neighborhood or structuring element source: Specify via dialog

Neighborhood or structuring element:

strel('square',4);

Input image type

If your input image is a binary image, select `Binary`. If your input image is an intensity image, select `Intensity`.

Neighborhood or structuring element source

Specify how to enter your neighborhood or structuring element values. Select `Specify via dialog` to enter the values in the dialog box. Select `Input port` to use the Nhood port to specify the neighborhood values. You can only specify a structuring element using the dialog box.

Neighborhood or structuring element

If you are specifying a neighborhood, this parameter must be a matrix or vector of 1s and 0s. If you are specifying a structuring element, use the `strel` function from the Image Processing Toolbox. This parameter is visible if, for the **Neighborhood or**

structuring element source parameter, you select `Specify via dialog`.

See Also

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

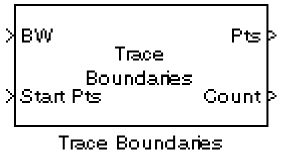
Purpose

Trace object boundaries in binary images

Library

Analysis & Enhancement

Description



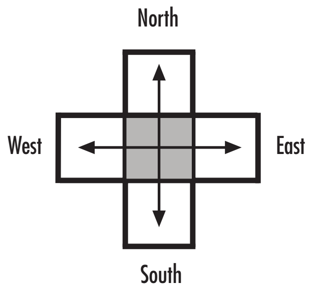
The Trace Boundaries block traces object boundaries in binary images, where nonzero pixels represent objects and 0 pixels represent the background.

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: $\begin{bmatrix} r_1 & r_2 & \cdots & r_N \\ c_1 & c_2 & \cdots & c_N \end{bmatrix}$	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer	No

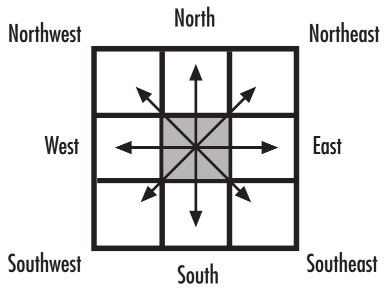
Port	Input/Output	Supported Data Types	Complex Values Supported
Pts	2M-by-N matrix where each column contains the zero-based row and column coordinates of the boundary pixels, M is the maximum number of boundary pixels, and N is the total number of starting points: $\begin{bmatrix} r_{11} & \cdots & r_{N1} \\ c_{11} & \cdots & c_{N1} \\ r_{12} & \cdots & r_{N2} \\ c_{12} & \cdots & c_{N2} \\ \vdots & \ddots & \vdots \\ r_{1M} & \cdots & r_{NM} \\ c_{1M} & \cdots & c_{NM} \end{bmatrix}$	Same as Start Pts port	No
Count	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.	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.

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.

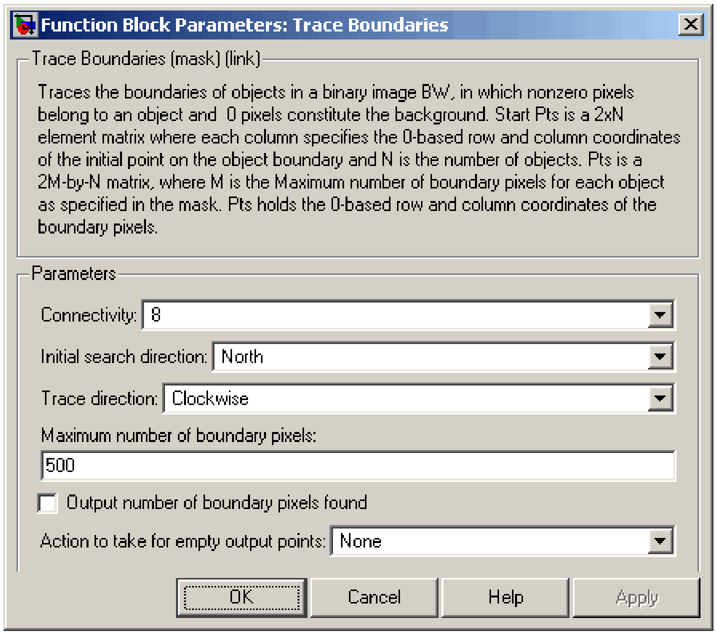
Because you specify the number of rows of the Pts port output matrix using the **Maximum number of boundary pixels** parameter, use the **Action to take for empty output points** parameter to specify what happens to the empty elements in this vector when the number of boundary pixels is less than the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



Trace Boundaries (mask) (link)

Traces the boundaries of objects in a binary image BW, in which nonzero pixels belong to an object and 0 pixels constitute the background. Start Pts is a 2xN element matrix where each column specifies the 0-based row and column coordinates of the initial point on the object boundary and N is the number of objects. Pts is a 2M-by-N matrix, where M is the Maximum number of boundary pixels for each object as specified in the mask. Pts holds the 0-based row and column coordinates of the boundary pixels.

Parameters

Connectivity: 8

Initial search direction: North

Trace direction: Clockwise

Maximum number of boundary pixels: 500

☐ Output number of boundary pixels found

Action to take for empty output points: None

Connectivity

Specify which pixels are connected to each other. If you want a pixel to be connected to the pixels on the top, bottom, left, and right, select `4`. If you want a pixel to be connected to the pixels on the top, bottom, left, right, and diagonally, select `8`.

Initial search direction

Specify the first direction in which to look to find the next boundary pixel that is connected to the starting pixel.

Trace direction

Specify the direction in which to trace the boundary. Your choices are `Clockwise` or `Counterclockwise`.

Maximum number of boundary pixels
Specify the maximum number of boundary pixels. The block uses this value to preallocate the number of rows of the Pts port output matrix so that it can hold all the boundary pixel location values.

Output number of boundary pixels found
If you select this check box, the block outputs a vector at the Count port where each element represents the actual number of boundary pixels found for each starting point.

Action to take for empty output points
Specify how to fill the empty spaces in the Pts port output matrix. If you select `None`, the block takes no action. So, any element that does not contain a boundary pixel location will not have a meaningful value. If you select `Fill with last point found`, the block fills the remaining elements with the position of the last boundary pixel. If you select `Fill with user-defined values`, the **Fill values** parameter appears on the block.

Fill values
Enter a scalar value or two-element vector that you want the block to use to fill in the remaining empty elements. This parameter is visible if, for the **Action to take for empty output points** parameter, you select `Fill with user-defined values`.

See Also

Edge Detection	Video and Image Processing Blockset software
Label	Video and Image Processing Blockset software
`bwboundaries`	Image Processing Toolbox software
`bwtraceboundary`	Image Processing Toolbox software

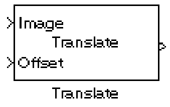
Purpose

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	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer	No
Offset	Vector of values that represent the number of pixels by which to translate the image	Same as I port	No
Output	Translated image	Same as I port	No

The input to the Offset port must be the same data type as the input to the Image port. The output is the same data type as the input to the Image port.

Use the **Output size after translation** parameter to specify the size of the translated image. If you select `Full`, the block outputs a matrix that contains the entire translated image. If you select `Same as input image`, the block outputs a matrix that is the same size as the input image and contains a portion of the translated image. Use the **Background fill value** parameter to specify the pixel values outside the image.

Use the **Translation values source** parameter to specify how to enter your displacement vector. If you select `Specify via dialog`, the **Offset** parameter appears in the dialog box. Use it to enter your displacement vector, a two-element vector, `[r c]`, of real, integer values that represent the number of pixels by which you want to translate your image. The `r` value represents how many pixels up or down to shift your image. The `c` value represents how many pixels left or right to shift your image. The axis origin is the top-left corner of your image. For example, if you enter `[2.5 3.2]`, the block moves the image 2.5 pixels downward and 3.2 pixels to the right of its original location. When the displacement vector contains fractional values, the block uses interpolation to compute the output.

Use the **Interpolation method** parameter to specify which interpolation method the block uses to translate the image. If you translate your image in either the horizontal or vertical direction and you select `Nearest neighbor`, the block uses the value of the nearest pixel for the new pixel value. If you translate your image in either the horizontal or vertical direction and you select `Bilinear`, the new pixel value is the weighted average of the four nearest pixel values. If you translate your image in either the horizontal or vertical direction and you select `Bicubic`, the new pixel value is the weighted average of the sixteen nearest pixel values.

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

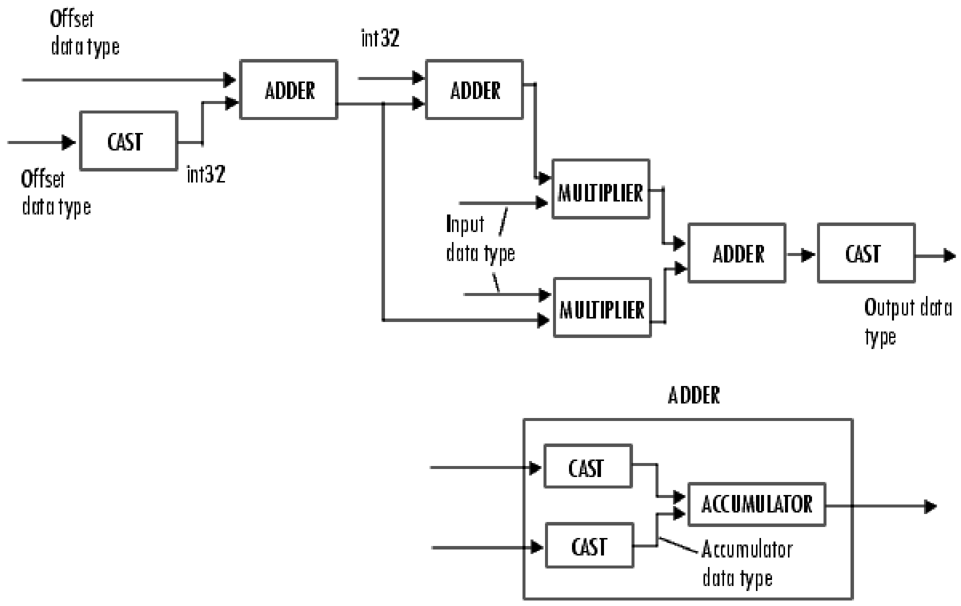
"Geometric Transformation Interpolation Methods" in the *Video and Image Processing Blockset User's Guide*.

If, for the **Output size after translation** parameter, you select `Full`, and for the **Translation values source** parameter, you select `Input port`, the **Maximum offset** parameter appears in the dialog box. Use the **Maximum offset** parameter to enter a two-element vector of real, scalar values that represent the maximum number of pixels by which you want to translate your image. The block uses this parameter to determine the size of the output matrix. If the input to the Offset port is greater than the **Maximum offset** parameter values, the block saturates to the maximum values.

If, for the **Translation values source** parameter, you select `Input port`, the Offset port appears on the block. At each time step, the input to the Offset port must be a vector of real, scalar values that represent the number of pixels by which to translate your image.

Fixed-Point Data Types

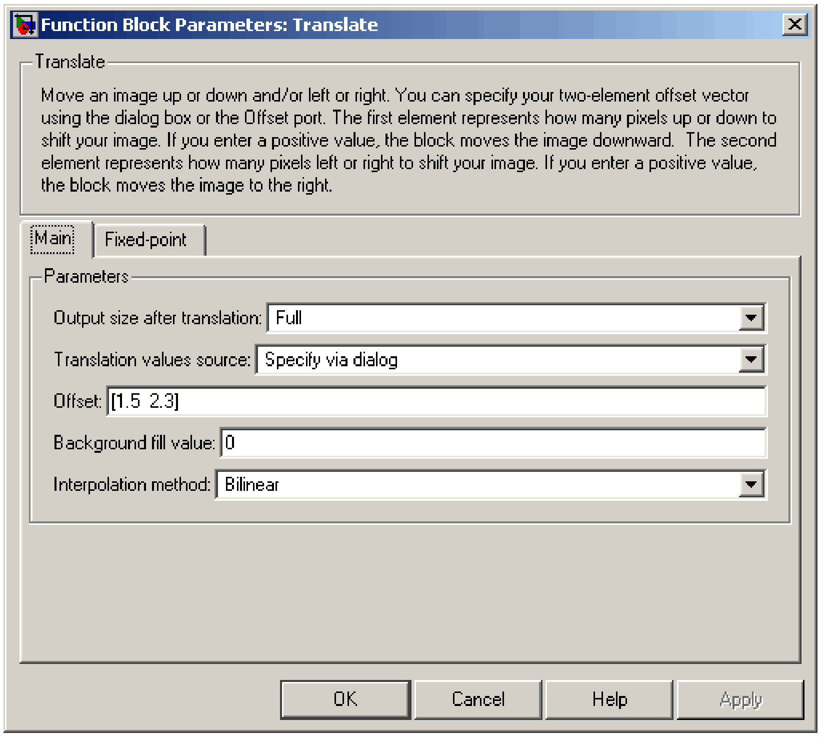
The following diagram shows the data types used in the Translate block for bilinear interpolation of fixed-point signals.



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

Dialog Box

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



Translate

Move an image up or down and/or left or right. You can specify your two-element offset vector using the dialog box or the Offset port. The first element represents how many pixels up or down to shift your image. If you enter a positive value, the block moves the image downward. The second element represents how many pixels left or right to shift your image. If you enter a positive value, the block moves the image to the right.

Parameters

Output size after translation: Full

Translation values source: Specify via dialog

Offset: [1.5 2.3]

Background fill value: 0

Interpolation method: Bilinear

Output size after translation

If you select `Full`, the block outputs a matrix that contains the translated image values. If you select `Same as input image`, the block outputs a matrix that is the same size as the input image and contains a portion of the translated image.

Translation values source

Specify how to enter your translation parameters. If you select `Specify via dialog`, the **Offset** parameter appears in the dialog box. If you select `Input port`, port O appears on the block.

The block uses the input to this port at each time step as your translation values.

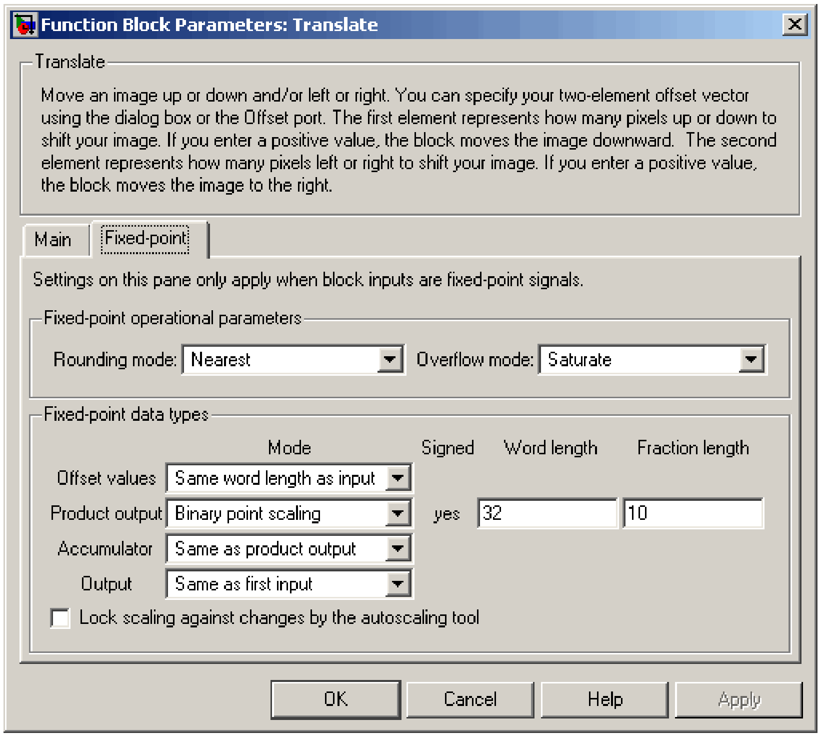
Offset
: Enter a vector of real, scalar values that represent the number of pixels by which to translate your image.

Background fill value
: Specify a value for the pixels that are outside the image.

Interpolation method
: Specify which interpolation method the block uses to translate the image. If you select `Nearest neighbor`, the block uses the value of one nearby pixel for the new pixel value. If you select `Bilinear`, the new pixel value is the weighted average of the four nearest pixel values. If you select `Bicubic`, the new pixel value is the weighted average of the sixteen nearest pixel values.

Maximum offset
: Enter a vector of real, scalar values that represent the maximum number of pixels by which you want to translate your image. This parameter must have the same data type as the input to the Offset port. This parameter is visible if, for the **Output size after translation** parameter, you select `Full` and, for the **Translation values source** parameter, you select `Input port`.

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



Translate

Move an image up or down and/or left or right. You can specify your two-element offset vector using the dialog box or the Offset port. The first element represents how many pixels up or down to shift your image. If you enter a positive value, the block moves the image downward. The second element represents how many pixels left or right to shift your image. If you enter a positive value, the block moves the image to the right.

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

Fixed-point operational parameters

Rounding mode: Nearest   Overflow mode: Saturate

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Offset values	Same word length as input			
Product output	Binary point scaling	yes	32	10
Accumulator	Same as product output			
Output	Same as first input			

☐ Lock scaling against changes by the autoscaling tool

Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Offset values

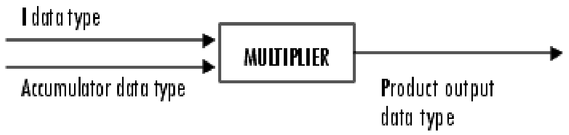
Choose how to specify the word length and the fraction length of the offset values.

- When you select `Same word length as input`, the word length of the offset values match that of the input to the block. In this mode, the fraction length of the offset values is automatically set to the binary-point only scaling that provides you with the best precision possible given the value and word length of the offset values.

- When you select `Specify word length`, you can enter the word length of the offset values, in bits. The block automatically sets the fraction length to give you the best precision.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the offset values, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the offset values. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

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

Product output

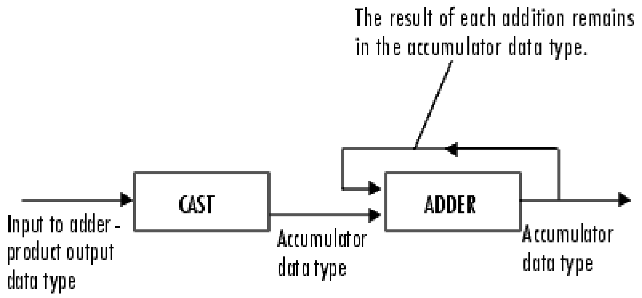


As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths.

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

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

Accumulator


The result of each addition remains in the accumulator data type.
CAST
ADDER
Input to adder - product output data type
Accumulator data type
Accumulator data type

As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

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

Output

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

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

References

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

See Also

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

Purpose Specify subset of rows or columns from input

Library Utilities

Description The Variable Selector block is a Signal Processing Blockset block. For more information, see the Variable Selector block reference page in the Signal Processing Blockset software documentation.

Variance

Purpose Compute variance of input or sequence of inputs

Library Statistics

Description The Variance block is a Signal Processing Blockset block. For more information, see the Variance block reference page in the Signal Processing Blockset software documentation.

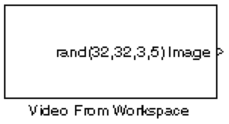
Purpose

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	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer	No

Port	Output	Supported Data Types	Complex Values Supported
R, G, B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B ports have the same dimensions.	Same as I port	No

For the Video and Image Processing Blockset blocks to display video data properly, double- and single-precision floating-point pixel values must be from 0 to 1. This block does not scale pixel values.

Use the **Signal** parameter to specify the MATLAB workspace variable from which to read. For example, to read an AVI file, use the following syntax:

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

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

- 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:

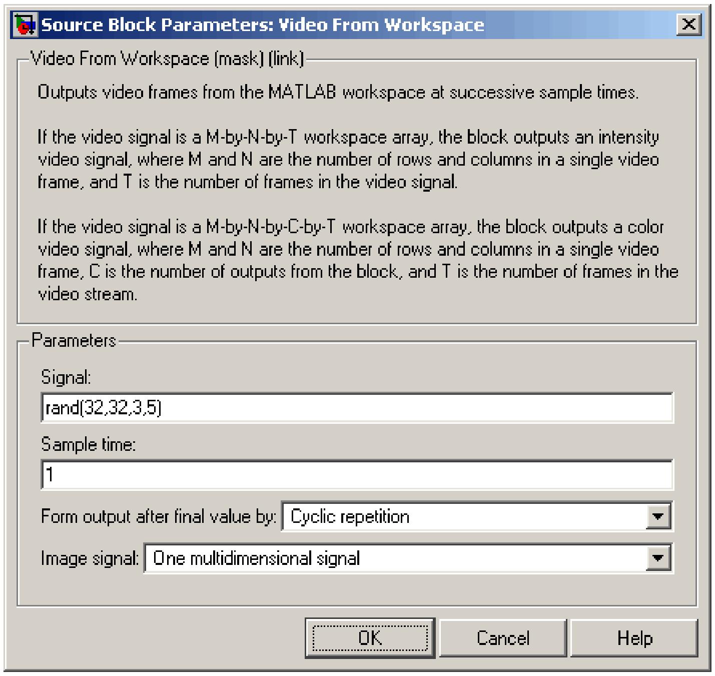
- When you specify `Setting To Zero`, the block generates zero-valued outputs for the duration of the simulation after generating the last frame of the signal.
- When you specify `Holding Final Value`, the block repeats the final frame for the duration of the simulation after generating the last frame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



Source Block Parameters: Video From Workspace

Video From Workspace (mask) (link)

Outputs video frames from the MATLAB workspace at successive sample times.

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 outputs from the block, and T is the number of frames in the video stream.

Parameters

Signal: rand(32,32,3,5)

Sample time: 1

Form output after final value by: Cyclic repetition

Image signal: One multidimensional signal

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`).

Image signal

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

Output port labels

Enter the labels for your output ports using the spacer character, `|`, as the delimiter. This parameter is available when the **Image signal** parameter is set to `Separate color signals`.

See Also

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

Video To Workspace

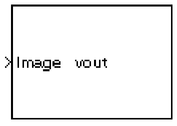
Purpose

Export video signal to MATLAB workspace

Library

Sinks

Description



Video To Workspace

The Video To Workspace block exports a video signal to the MATLAB workspace. If the video signal is represented by intensity values, it appears in the workspace as a three-dimensional M-by-N-by-T array, where M and N are the number of rows and columns in a single video frame, and T is the number of frames in the video signal. If it is a color video signal, it appears in the workspace as a four-dimensional M-by-N-by-C-by-T array, where M and N are the number of rows and columns in a single video frame, C is the number of inputs to the block, and T is the number of frames in the video stream. During code generation, Real-Time Workshop does not generate code for this block.

Note This block supports intensity and color images on its ports.

Port	Input	Supported Data Types	Complex Values Supported
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integer • 8-, 16-, 32-bit unsigned integer	No
R, G, B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Outputs	Same as I port	No

Port	Input	Supported Data Types	Complex Values Supported
	from the R, G, or B ports have the same dimensions.		

Use the **Variable name** parameter to specify the MATLAB workspace variable to which to write the video signal.

Use the **Number of inputs** parameter to determine the number of inputs to the block. If the video signal is represented by intensity values, enter 1. If it is a color (R, G, B) video signal, enter 3.

Use the **Limit data points to last** parameter to determine the number of video frames, T, you want to export to the MATLAB workspace.

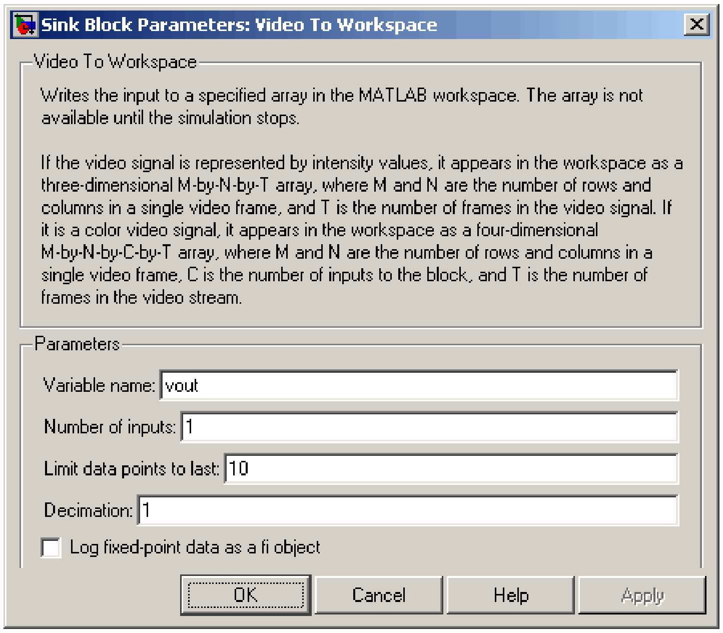
If you want to downsample your video signal, use the **Decimation** parameter to enter your decimation factor.

If your video signal is fixed point and you select the **Log fixed-point data as a fi object** check box, the block creates a fi object in the MATLAB workspace.

Use the **Input port labels** parameter to label your input ports. Use the spacer character, |, as the delimiter. This parameter is available if the **Number of inputs** parameter is greater than 1.

Dialog Box

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



Video To Workspace

Writes the input to a specified array in the MATLAB workspace. The array is not available until the simulation stops.

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.

Parameters

Variable name: vout

Number of inputs: 1

Limit data points to last: 10

Decimation: 1

☐ Log fixed-point data as a fi object

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.

Log fixed-point data as a fi object
If your video signal is fixed point and you select this check box, the block creates a fi object in the MATLAB workspace. For more information of fi objects, see the Fixed-Point Toolbox documentation.

Input port labels
Enter the labels for your input ports using the spacer character, `|`, as the delimiter. This parameter is available if the **Number of inputs** parameter is greater than 1.

See Also

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

Video Viewer

Purpose

Display binary, intensity, or RGB images or video streams

Library

Sinks

Description

Image
Video
Viewer

Video Viewer

The Video Viewer block enables you to view a binary, intensity, or RGB image or a video stream. The block provides simulation controls for play, pause, and step while running the model. The block also provides pixel region analysis tools. During code generation, Real-Time Workshop software does not generate code for this block.

Note The To Video Display block supports code generation.

See the following table for descriptions of both input types.

Input	Description
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes.
R/G/B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Inputs to the R, G, or B ports must have the same dimensions and data type.

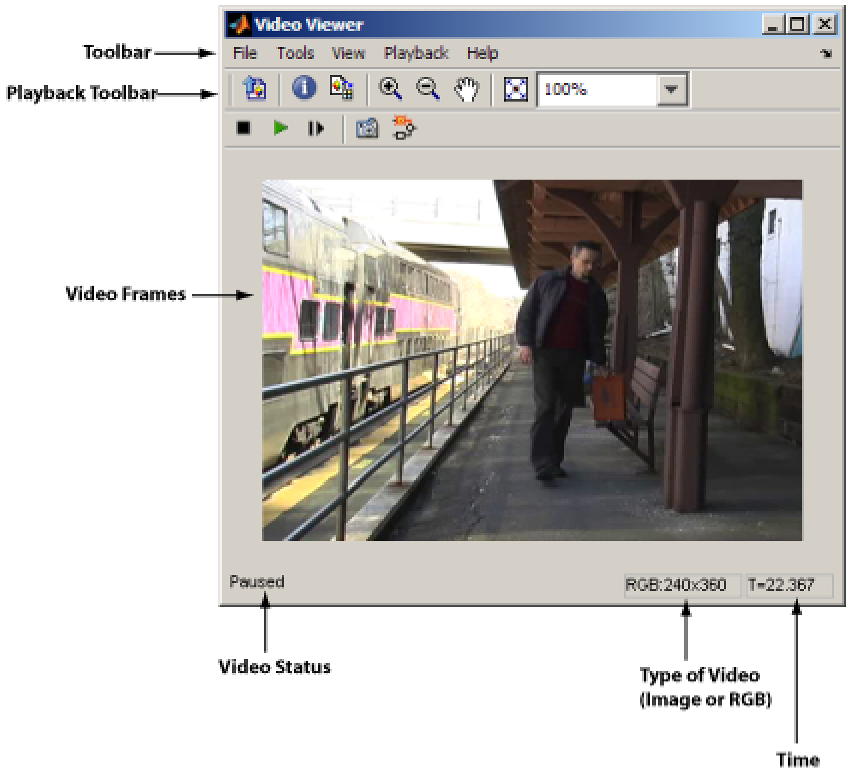
Select **File > Image Signal** to set the input to either **Image** or **RGB**.

Dialogs

- "GUI Buttons" on page 2-641
- "Setting Viewer Configuration" on page 2-644
- "Video Information" on page 2-647
- "Colormap for Intensity Video" on page 3-18
- "Saving the Settings of Multiple Video Viewer GUIs" on page 2-648
- "Message Log" on page 3-21

- "Status Bar" on page 2-649

GUI Buttons



Toolbar

GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
	File > Export to Image Tool	**Ctrl+E**	Send the current video frame to the Image Tool. For more information, see "Using the Image Tool to Explore Images" in the Image Processing Toolbox documentation.

Note The Image Tool can only know that the frame is an intensity image if the colormap of the frame is grayscale (`gray(256)`). Otherwise, the Image Tool assumes the frame is an indexed image and disables the **Adjust Contrast** button.

GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
ℹ	**Tools > Video Information**	**V**	View information about the video data source.
	Tools > Pixel Region	N/A	Open the Pixel Region tool. For more information about this tool, see the Image Processing Toolbox documentation.
🔍	**Tools > Zoom In**	N/A	Zoom in on the video display.
🔍	**Tools > Zoom Out**	N/A	Zoom out of the video display.
✋	**Tools > Pan**	N/A	Move the image displayed in the GUI.

Toolbar (Continued)

GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
⤢	**Tools > Maintain Fit to Window**	N/A	Scale video to fit GUI size automatically. Toggle the button on or off.
110% ▾	N/A	N/A	Enlarge or shrink the video frame. This option becomes available if you do not select the **Maintain Fit to Window**.

Playback Toolbar

GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
■	**Playback > Stop**	S	Stop the video.
▶	**Playback > Play**	**P**, Space bar	Play the video.
⏸	**Playback > Pause**	**P**, Space bar	Pause the video. This button appears only when the video is playing.
▮▶	**Playback > Step Forward**	Right arrow, **Page Down**	Step forward one frame.

Playback Toolbar (Continued)

GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
	Playback > Simulink Snapshot	N/A	Click this button to freeze the display in the viewer window.
	Playback > Highlight Simulink Signal	**Ctrl+L**	In the model window, highlight the Simulink signal the viewer is displaying.

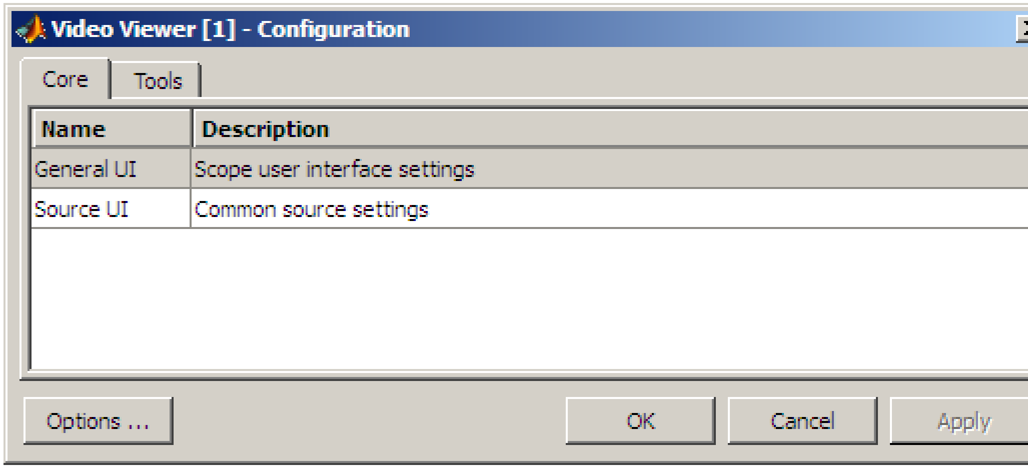
Setting Viewer Configuration

The Video Viewer Configuration preferences enables you to change the behavior and appearance of the graphic user interface (GUI) as well as the behavior of the playback shortcut keys.

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

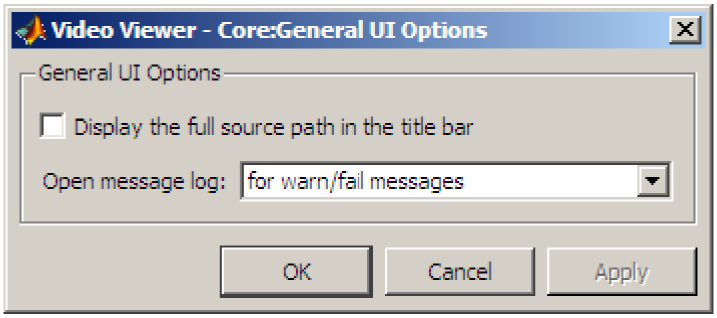
Core Pane

The Core pane controls the GUI's general settings.



Name	Description
General UI	Scope user interface settings
Source UI	Common source settings

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



☐ Display the full source path in the title bar

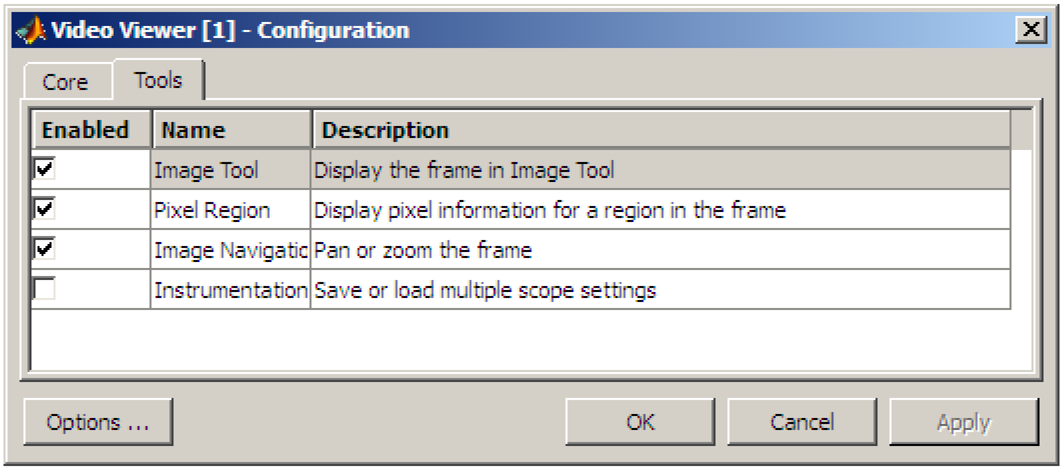
Open message log: for warn/fail messages

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

Use the **Open message log:** parameter to control when the Message log window opens. You can use this window to debug issues with video playback. Your choices are `for any new messages`, `for warn/fail messages`, `only for fail messages`, or `manually`.

Tools Pane

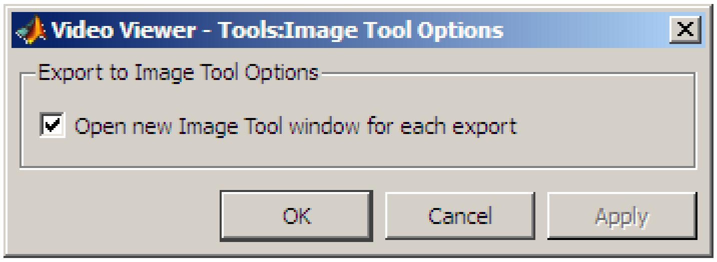
The Tools pane contains the tools that appear on the Video Viewer GUI. Select the **Enabled** check box next to the tool name to specify which tools to include on the GUI.



Enabled	Name	Description
✓	Image Tool	Display the frame in Image Tool
✓	Pixel Region	Display pixel information for a region in the frame
✓	Image Navigatic	Pan or zoom the frame
	Instrumentation	Save or load multiple scope settings

Image Tool

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



Export to Image Tool Options

[✓] Open new Image Tool window for each export

Select the **Open new Image Tool window for export** check box if you want to send each video frame to a different session of Image Tool.

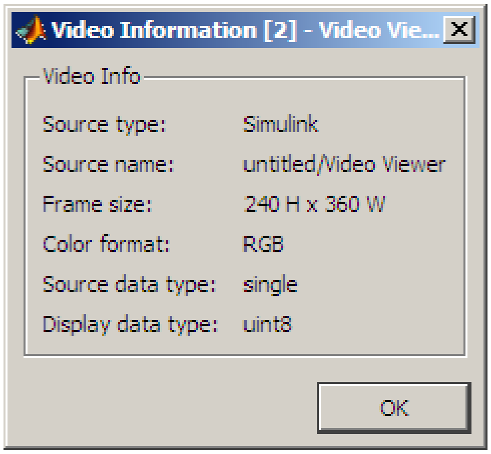
Pixel Region
Select the **Pixel Region** check box to display and enable the pixel region GUI button. For more information on working with pixel regions see Getting Information about the Pixels in an Image.

Image Navigation Tools
Select the **Image Navigation Tools** check box to enable the pan-and-zoom GUI button.

Instrumentation Set
Select the **Instrumentation Set** check box to enable the option to load and save viewer settings. The option appears in the **File** menu.

Video Information

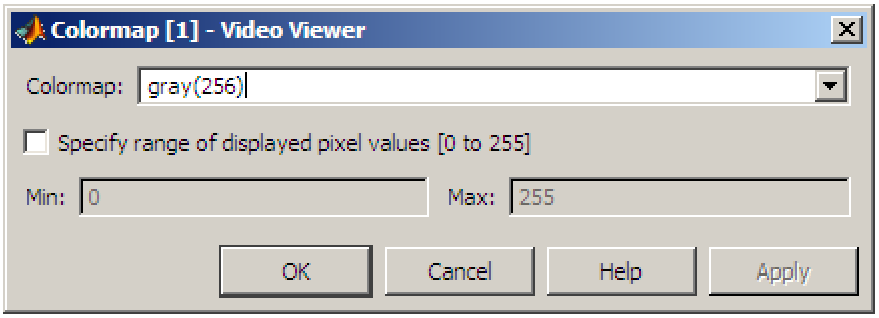
The Video Information dialog box lets you view basic information about the video. To open this dialog box, you can select **Tools > Video Information** , click the information button ⓘ , or press the **V** key.



Source type:	Simulink
Source name:	untitled/Video Viewer
Frame size:	240 H x 360 W
Color format:	RGB
Source data type:	single
Display data type:	uint8

Colormap for Intensity Video

The Colormap dialog box lets you change the colormap of an intensity video. You cannot access the parameters on this dialog box when the GUI displays an RGB video signal. To open this dialog box for an intensity signal, select **Tools > Colormap** or press **C**.



Colormap: gray(256)
Specify range of displayed pixel values [0 to 255]
Min: 0
Max: 255

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.

Saving the Settings of Multiple Video Viewer GUIs

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

Message Log

The Message Log dialog provides a system level record of configurations and extensions used. You can filter what messages to display by **Type** and **Category**, view the records, and display record details.

The **Type** parameter allows you to select either `All`, `Info`, `Warn`, or `Fail` message logs. The **Category** parameter allows you to select either `Configuration` or `Extension` message summaries. The `Configuration` messages indicate when a new configuration file is loaded. The `Extension` messages indicate a component is registered. For example, you might see a `Simulink` message, which indicates the component is registered and available for configuration.

Status Bar

A status bar appear along the bottom of the Video Viewer. It displays information pertaining to the video status (running, paused or ready), type of video (Intensity or RGB) and video time.

Supported Data Types

Port	Supported Data Types
Image	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, and 32-bit signed integer • 8-, 16-, and 32-bit unsigned integer
R/G/B	Same as Image port

See Also

From Multimedia File	Video and Image Processing Blockset software
`mplay`	Video and Image Processing Blockset software
To Multimedia File	Signal Processing Blockset software

To Video Display	Video and Image Processing Blockset software
Video To Workspace	Video and Image Processing Blockset software
implay	Image Processing Toolbox

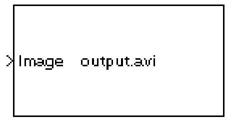
Purpose

Write video frames to uncompressed AVI file

Library

Sinks

Description



Write AVI File

Note The Write AVI File block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset blocks. Use the replacement block To Multimedia File.

The Write AVI File block writes video frames to an uncompressed AVI file from a Simulink model. If the data type of the input pixel values is anything other than 8-bit unsigned integers, the block scales the values. Then, it writes values between the minimum and maximum values supported by the 8-bit unsigned integer data type to the AVI file. This block does not support audio samples. During code generation, Real-Time Workshop does not generate code for this block.

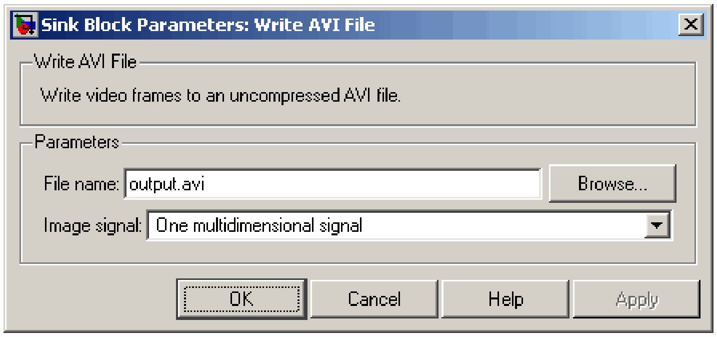
Port	Input	Supported Data Types	Complex Values Supported
Image	M-by-N matrix of intensity values or an M-by-N-by-P color video signal where P is the number of color planes	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16- 32-bit signed integer • 8-, 16- 32-bit unsigned 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. 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.


Sink Block Parameters: Write AVI File

Write AVI File

Write video frames to an uncompressed AVI file.

Parameters

File name: output.avi

Image signal: One multidimensional signal

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	Signal Processing Blockset software
To Video Display	Video and Image Processing Blockset software
Video To Workspace	Video and Image Processing Blockset software
Video Viewer	Video and Image Processing Blockset software

Write Binary File

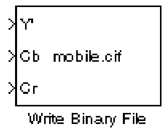
Purpose

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.

This block produces a raw binary file with no header information. It has no encoded information providing the data type, frame rate or dimensionality. The video data for this block appears in row major format.

Note This block supports code generation only for platforms that have file I/O available. You cannot use this block to do code generation with RTWin (Real-Time Windows Target).

Port	Input	Supported Data Types	Complex Values Supported
Input	Matrix that represents the luma (Y') and chroma (Cb and Cr) components of a video stream	• 8-, 16- 32-bit signed integer • 8-, 16- 32-bit unsigned integer	No

Four Character Code Video Formats

Four Character Codes (FOURCC) identify video formats. For more information about these codes, see `http://www.fourcc.org`.

Use the **Four character code** parameter to identify the video format.

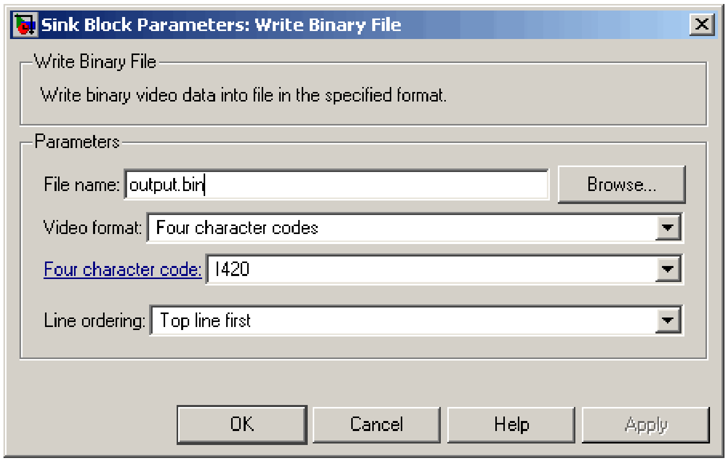
Custom Video Formats

You can use the Write Binary File block to create a binary file that contains video data in a custom format.

- Use the **Bit stream format** parameter to specify whether you want your data in planar or packed format.
- Use the **Number of input components** parameter to specify the number of components in the video stream. This number corresponds to the number of block input ports.
- Select the **Inherit size of components from input data type** check box if you want each component to have the same number of bits as the input data type. If you clear this check box, you must specify the number of bits for each component.
- Use the **Component** parameters to specify the component names.
- Use the **Component order in binary file** parameter to specify how to arrange the components in the binary file.
- Select the **Interlaced video** check box if the video stream represents interlaced video data.
- Select the **Write signed data to output file** check box if your input data is signed.
- Use the **Byte order in binary file** parameter to specify whether the byte ordering in the output binary file is little endian or big endian.

Dialog Box

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


Sink Block Parameters: Write Binary File

Write Binary File

Write binary video data into file in the specified format.

Parameters

File name: output.bin

Video format: Four character codes

Four character code: I420

Line ordering: Top line first

File name

Specify the name of the binary file. If the location of this file is on your MATLAB path, enter the filename. If the location of this file is not on your MATLAB path, use the **Browse** button to specify the full path to the file including the filename.

Video format

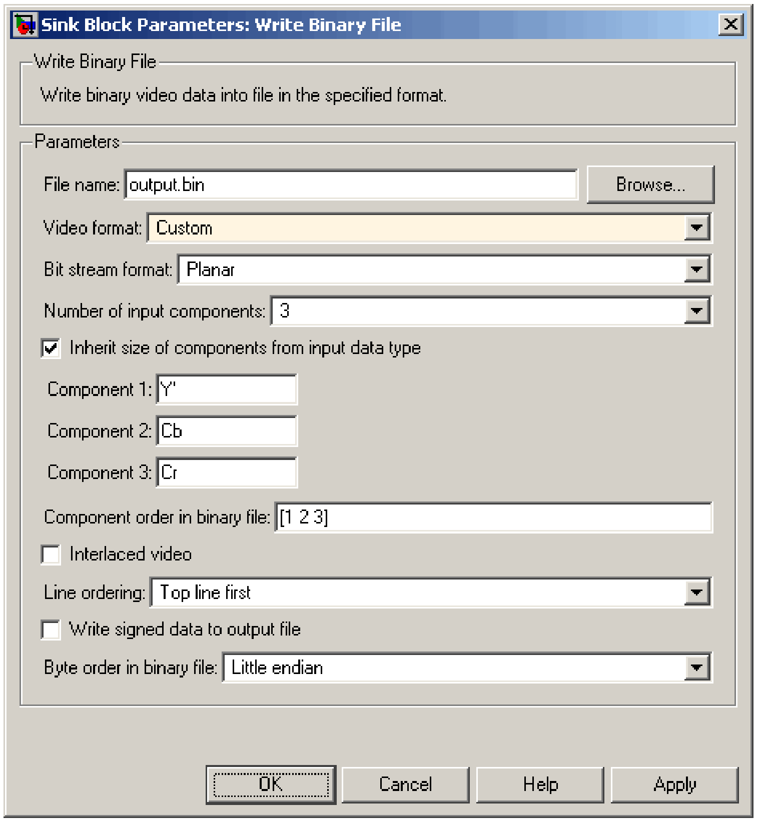
Specify the format of the binary video data as either`Four character codes` or `Custom`. See "Four Character Code Video Formats" on page 2-654 or "Custom Video Formats" on page 2-654 for more details.

Four character code

From the list, select the binary file format.

Line ordering

Specify how the block fills the binary file. If you select `Top line first`, the block first fills the binary file with the first row of the video frame. It then fills the file with the other rows in increasing order. If you select `Bottom line first`, the block first fills the binary file with the last row of the video frame. It then fills the file with the other rows in decreasing order.


Sink Block Parameters: Write Binary File

Write Binary File

Write binary video data into file in the specified format.

Parameters

File name: output.bin

Video format: Custom

Bit stream format: Planar

Number of input components: 3

[✓] Inherit size of components from input data type

Component 1: Y'

Component 2: Cb

Component 3: Cr

Component order in binary file: [1 2 3]

[ ] Interlaced video

Line ordering: Top line first

[ ] Write signed data to output file

Byte order in binary file: Little endian

Bit stream format

Specify whether you want your data in planar or packed format.

Number of input components

Specify the number of components in the video stream. This number corresponds to the number of block input ports.

Inherit size of components from input data type
Select this check box if you want each component to have the same number of bits as the input data type. If you clear this check box, you must specify the number of bits for each component.

Component
Specify the component names.

Component order in binary file
Specify how to arrange the components in the binary file.

Interlaced video
Select this check box if the video stream represents interlaced video data.

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

Byte order in binary file
Use this parameter to specify whether the byte ordering in the output binary file is little endian or big endian.

See Also

Read Binary File	Video and Image Processing Blockset
To Multimedia File	Signal Processing Blockset

Function Reference

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('filename.avi',fps)
mplay(a)
mplay(a,fps)
mplay({line_handles})
mplay({'block',port})
```

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. In the model window, you can access this GUI by selecting **Tools > MPlay Video Viewer**. If the video contains audio, the GUI ignores it and plays only the video frames.

`mplay` opens an MPlay GUI.

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

`mplay('filename.avi',fps)` connects the MPlay GUI to the specified AVI file with the specified frame rate in frames per second, `fps`. By default, `fps` is the same as the frame rate specified in the file.

`mplay(a)` connects the MPlay GUI to the variable in the MATLAB workspace, `a`. The variable, `a`, must have one of the following formats:

- MATLAB movie structure
- Intensity movie array (M-by-N-by-T or M-by-N-by-1–by-T array, where the size of each image is M-by-N and there are T image frames).
- RGB movie array (M-by-N-by-3-by-T array, where the size of each RGB image is M-by-N-by-3 and there are T image frames).

`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`.

`mplay({line_handles})` connects the MPlay GUI to one or three Simulink signal lines to display, where all signals must originate from the same block. To get the handles to the Simulink signals, `line_handles`, issue the command `mplay({gsl})`.

`mplay({'block',port})` connects the MPlay GUI to the signal output from the specified block, `'block'`, on output port index `port`. To get the full block path name of the current Simulink block, issue the command `mplay({gcb,1})`. If omitted, `port` is set to 1.

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

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.

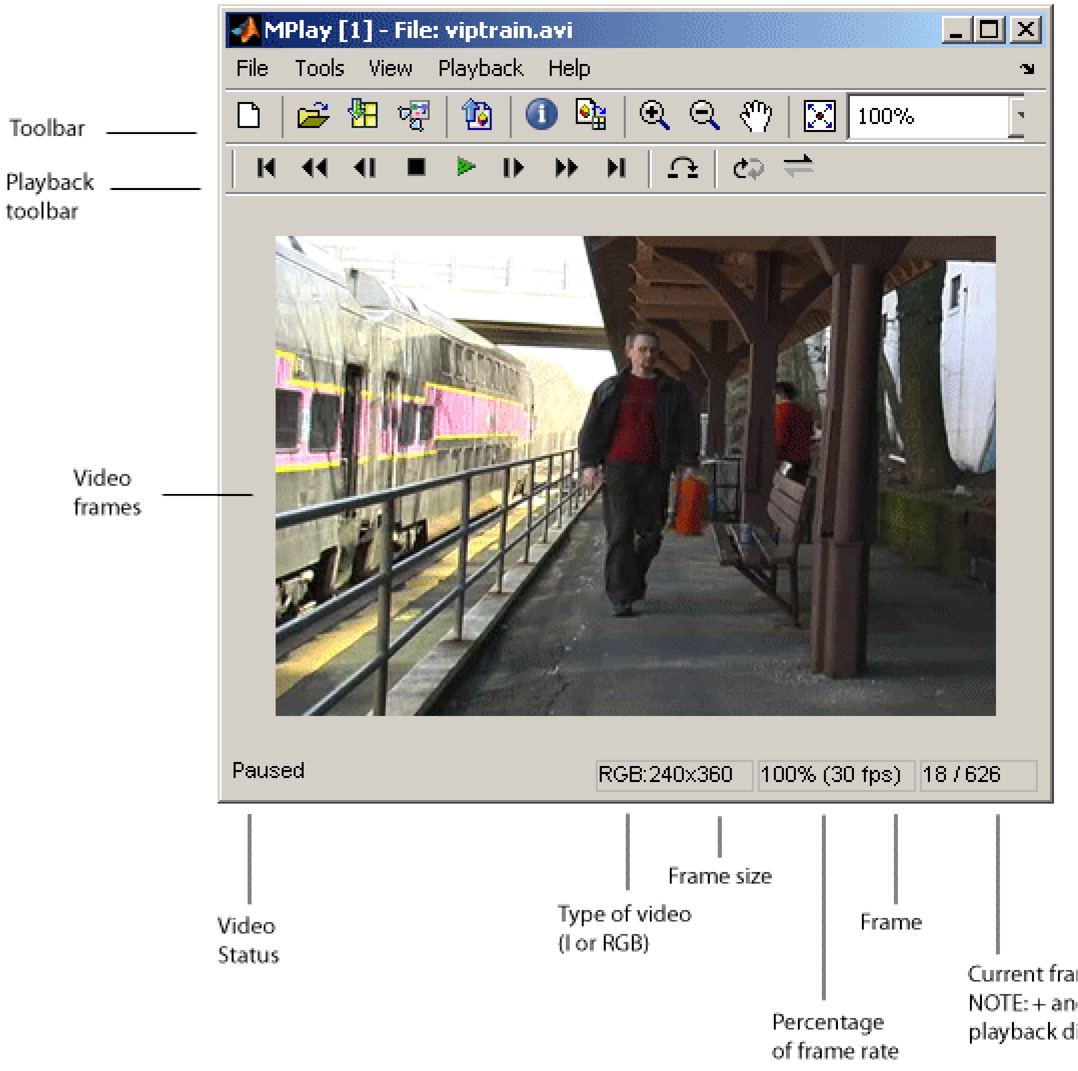
For more information about the MPlay GUI, see "Using the MPlay GUI"in the *Video and Image Processing Blockset User's Guide*. You can also see the "MPlay Simulink Tutorial" in the "Display and Graphics " section of the Video and Image Processing Blockset demos.

Dialogs

- "GUI Elements" on page 3-6
- "MPlay Configuration" on page 3-12
- "Video Information" on page 3-18
- "Colormap for Intensity Video" on page 3-18
- "Frame Rate" on page 3-19

- "Saving the Settings of Multiple MPlay GUIs" on page 3-21
- "Message Log" on page 3-21
- "Status Bar" on page 3-22

GUI Elements



Toolbar

GUI	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.
📤	**File > Export to Image Tool**	**Ctrl+E**	Send the current video frame to the Image Tool. For more information, see "Using the Image Tool to Explore Images" in the Image Processing Toolbox documentation.

Note The Image Tool only knows the frame is an intensity image if the colormap of the frame is grayscale (`gray(256)`). Otherwise, the Image Tool assumes the frame is an indexed image and disables the **Adjust Contrast** button.

Toolbar (Continued)

GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
ℹ	**Tools > Video Information**	**V**	View information about the video data source.
▦	**Tools > Pixel Region**	N/A	Open the Pixel Region tool. For more information about this tool, see the Image Processing Toolbox documentation.
🔍+	**Tools > Zoom In**	N/A	Zoom in on the video display.
🔍−	**Tools > Zoom Out**	N/A	Zoom out of the video display.
✋	**Tools > Pan**	N/A	Move the image displayed in the GUI.
⤢	**Tools > Maintain Fit to Window**	N/A	Scale video to fit GUI size automatically. Toggle the button on or off.
110% ▾	N/A	N/A	Enlarge or shrink the video. This option is available if the **Maintain Fit to Window** button is not selected.

Playback Toolbar — Workspace and File Sources

GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
⏮	**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 appears 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.
↷	**Playback > Jump to**	**J**	Jump to a specific frame.

Playback Toolbar — Workspace and File Sources (Continued)

GUI	Menu Equivalent	Shortcut Keys and Accelerators	Description
⟲	**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.

Playback Toolbar — Simulink Sources

Button	Menu Equivalent	Shortcut Keys and Accelerators	Description
■	**Playback > Stop**	**S**	Stop the video. This button also controls the Simulink model.

Playback Toolbar — Simulink Sources (Continued)

Button	Menu Equivalent	Shortcut Keys and Accelerators	Description
▶	**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 appears only when the video is playing.
❚▶	**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**	N/A	Indicates floating Simulink connection. In this mode, you can click different signals in

Playback Toolbar — Simulink Sources (Continued)

Button	Menu Equivalent	Shortcut Keys and Accelerators	Description
	Connection (selected)		the model, and the MPlay GUI displays them. Only one MPlay GUI can be in floating-scope mode at one time.

MPlay Configuration

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.

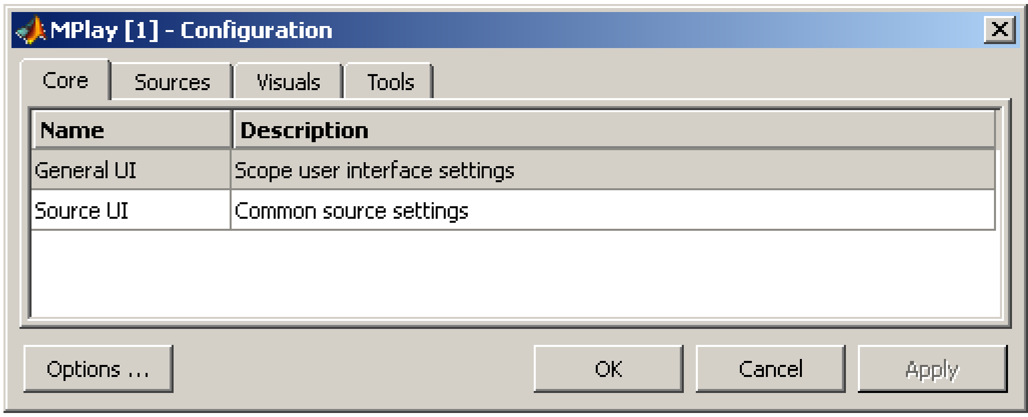
- To open the Configuration dialog box, select **File > Configuration Set > Edit**.
- To save the configuration settings for future use, select **File > Configuration Set > Save as**.

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**.

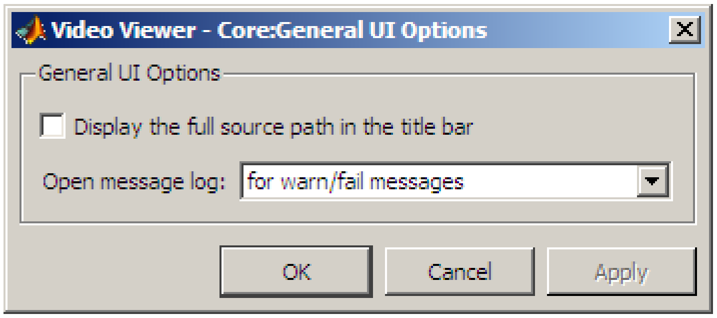
Core Pane

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



Name	Description
General UI	Scope user interface settings
Source UI	Common source settings

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



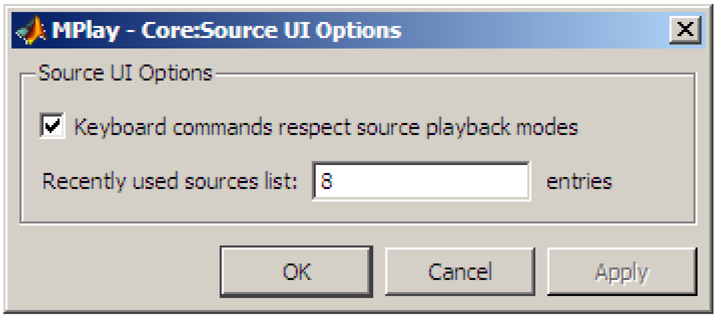
☐ Display the full source path in the title bar

Open message log: for warn/fail messages

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

Use the **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`.

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


MPlay - Core:Source UI Options

Source UI Options

✓ Keyboard commands respect source playback modes

Recently used sources list: 8 entries

If you select the **Keyboard commands respect playback modes** check box, the keyboard shortcut keys behave in response to the playback mode you selected.

Using the Keyboard commands respect playback modes

Open and play a video using MPlay.

1 Select the **Keyboard commands respect playback modes** check box.

2 Select the **Backward playback** button.

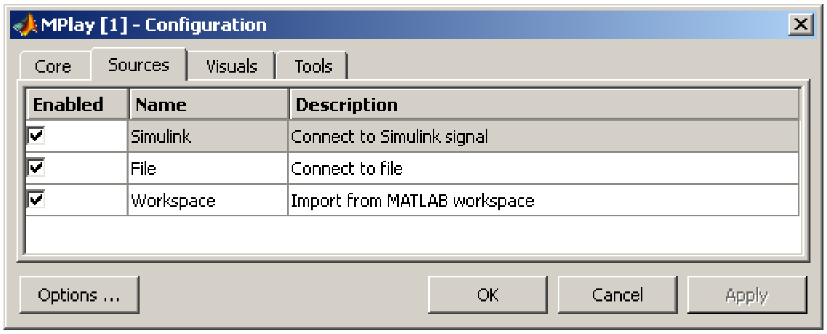
3 Notice that using the right keyboard arrow key moves the video backward and the left keyboard arrow key moves the video forward. Since MPlay is set to play backwards, the keyboard commands “forward” can be thought of as “forward with the direction the video is playing”.

To disconnect the keyboard behavior from the MPlay playback settings, clear the check box.

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

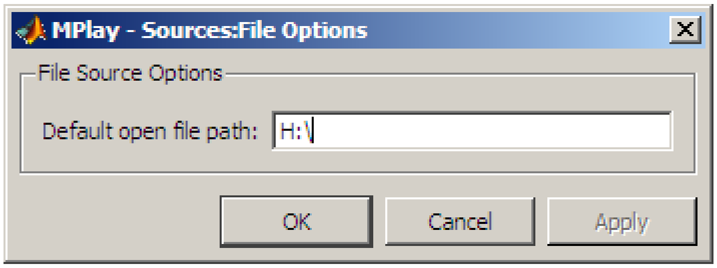
Sources Pane

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.



Enabled	Name	Description
✓	Simulink	Connect to Simulink signal
✓	File	Connect to file
✓	Workspace	Import from MATLAB workspace

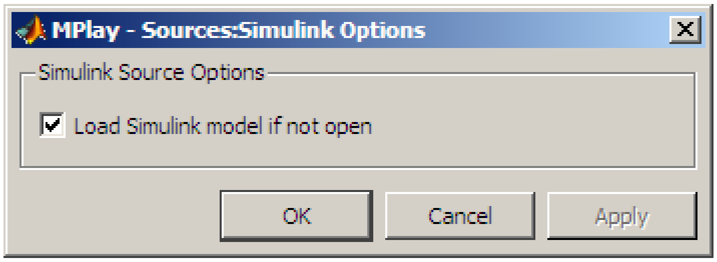
Click **File**, and then click the **Options** button to open the **Sources:File Options** dialog box.



Default open file path: H:\

Use the **Default open file path** parameter to control the directory that is displayed in the Connect to File dialog box when you select **File > Open**.

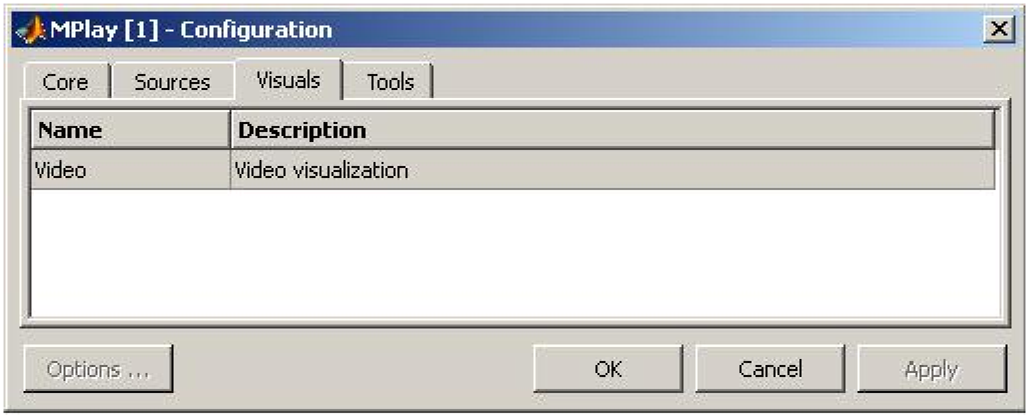
Click **Simulink**, and then click the **Options** button to open the **Sources:Simulink Options** dialog box.


MPlay - Sources:Simulink Options
Simulink Source Options
Load Simulink model if not open
OK
Cancel
Apply

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.

Visuals Pane

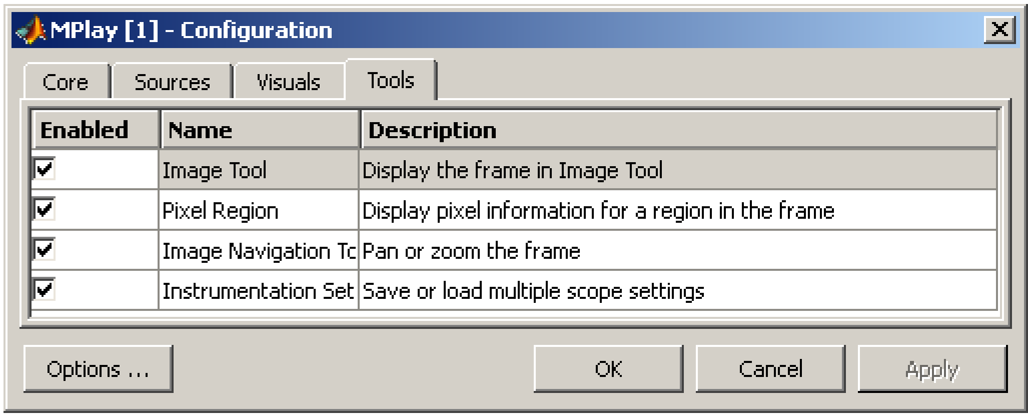
The Visuals pane contains the name of the visual type and its description.



Name	Description
Video	Video visualization

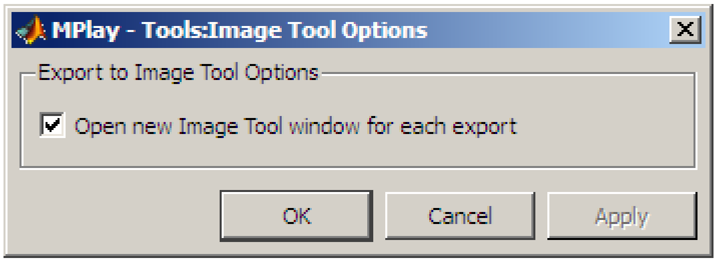
Tools Pane

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.



Enabled	Name	Description
✓	Image Tool	Display the frame in Image Tool
✓	Pixel Region	Display pixel information for a region in the frame
✓	Image Navigation Tc	Pan or zoom the frame
✓	Instrumentation Set	Save or load multiple scope settings

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


MPlay - Tools:Image Tool Options
Export to Image Tool Options
OK
Cancel
Apply

✓ Open new Image Tool window for each export

Select the **Open new Image Tool window for export** check box if you want to send each video frame to a different session of Image Tool.

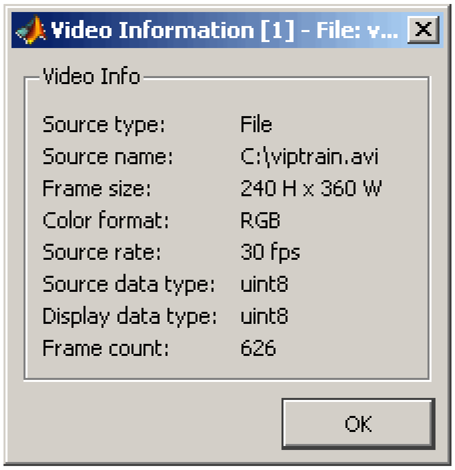
Pixel Region
Select the **Pixel Region** check box to display and enable the pixel region GUI button. For more information on working with pixel regions see Getting Information about the Pixels in an Image.

Image Navigation Tools
Select the **Image Navigation Tools** check box to enable the pan-and-zoom GUI button.

Instrumentation Set
Select the **Instrumentation Set** check box to enable the option to load and save viewer settings. The option appears in the **File** menu.

Video Information

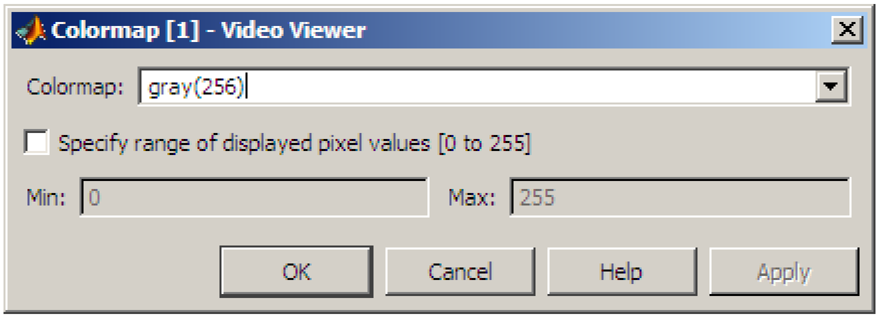
The Video Information dialog box lets you view basic information about the video. To open this dialog box, select **Tools > Video Information** or click the information button ℹ.



```
Source type:        File
Source name:        C:\viptrain.avi
Frame size:         240 H x 360 W
Color format:       RGB
Source rate:        30 fps
Source data type:   uint8
Display data type:  uint8
Frame count:        626
```

Colormap for Intensity Video

The Colormap dialog box lets you change the colormap of an intensity video. You cannot access the parameters on this dialog box when the GUI displays an RGB video signal. To open this dialog box for an intensity signal, select **Tools > Colormap** or press **C**.



Colormap: gray(256)

☐ Specify range of displayed pixel values [0 to 255]

Min: 0 Max: 255

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

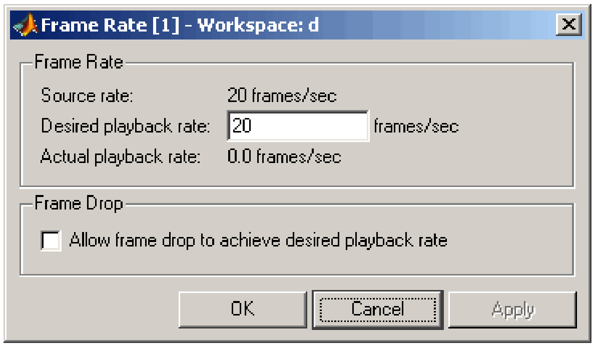
If you know that the pixel values do not use the entire data type range, you can select the **Specify range of displayed pixel values** check box and enter the range for your data. The dialog box automatically displays the range based on the data type of the pixel values.

Frame Rate

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, select **Playback > Frame Rate** or press **T**.



Frame Rate [1] - Workspace: d

Frame Rate

Source rate: 20 frames/sec

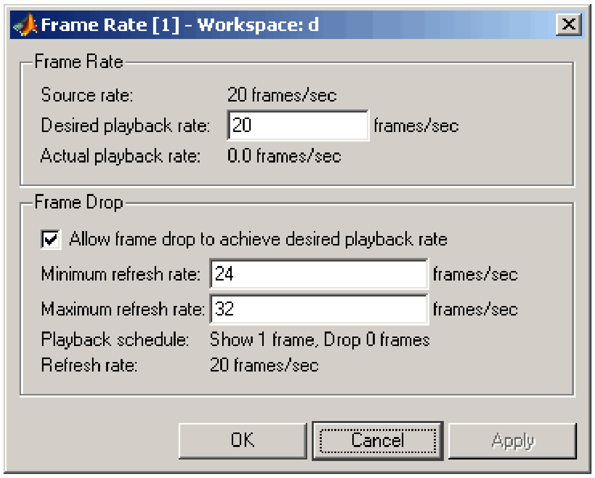
Desired playback rate: 20 frames/sec

Actual playback rate: 0.0 frames/sec

Frame Drop

☐ Allow frame drop to achieve desired playback rate

If you want to increase the actual playback rate, but your system's hardware cannot keep pace 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. Resulting video playback might appear choppy.



Frame Rate [1] - Workspace: d

Frame Rate

Source rate: 20 frames/sec

Desired playback rate: 20 frames/sec

Actual playback rate: 0.0 frames/sec

Frame Drop

☑ Allow frame drop to achieve desired playback rate

Minimum refresh rate: 24 frames/sec

Maximum refresh rate: 32 frames/sec

Playback schedule: Show 1 frame, Drop 0 frames

Refresh rate: 20 frames/sec

You can refine further both the quality of playback versus the hardware burden, by controlling the number of frames to drop per frame or frames displayed. For example, suppose you set the **Desired playback rate** to 80 frames/sec. One way to achieve this desired playback rate is to control the **Playback schedule** to **Show 1 frame, Drop 3 frames** by setting both the refresh rates, (which is how often the GUI updates the screen), to 20 frames/sec. By using these parameter settings, you see that `mplay` can achieve the Desired playback rate (in this case, 80 frames/sec).

In general, the relationship between the **Frame Drop** parameters is:

$$Desired\_rate = refresh\_rate * \frac{show\_frames + drop\_frames}{show\_frames}$$

where the *refresh_rate* includes a more accurate calculation based on both the minimum and maximum refresh rates.

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. Thus, you only need to configure the MPlay GUIs that are associated with your model once. To save the GUI settings, select **File > Instrumentation Sets > Save Set**. To open the preconfigured MPlay GUIs, select **File > Instrumentation Sets > Load Set**.

Message Log

The Message Log dialog provides a system level record of configurations and extensions used. You can filter what messages to display by **Type** and **Category**, view the records, and display record details.

The **Type** parameter allows you to select either `All`, `Info`, `Warn`, or `Fail` message logs. The **Category** parameter allows you to select either

`Configuration` or `Extension` message summaries. The `Configuration` messages indicate when a new configuration file is loaded. The `Extension` messages indicate a component is registered. For example, you might see a `Simulink` message, which indicates the component is registered and available for configuration.

Status Bar

Along the bottom of the MPlay viewer is the status bar. It displays information, such as video status, Type of video playing (`I` or `RGB`), Frame size, Percentage of frame rate, Frame rate, and `Current frame: Total frames`.

Note A minus sign (`-`) for `Current frame` indicates that the video is being played backwards.

Example

```
 % Play a video created in MATLAB workspace
fig=figure;  % create a video
set(gca,'xlim',[-80 80],'ylim',[-80 80],'NextPlot', ...
'replace','Visible','off');
x = -pi:.1:pi;
radius = 0:length(x);
video = []; % initialize video variable
   for i=length(x):-1:1
     patch(sin(x)*radius(i),cos(x)*radius(i), ...
[abs(cos(x(i))) 0 0]);
     F = getframe(gca);
    video = cat(4,video,F.cdata); % video is MxNx3xT
   end
   close(fig);
   mplay(video); % display a video
```

Supported Data Types

Input	Supported Data Types	Converted to uint8
a	double	✓
	single	✓
	int8	
	uint8 (improved performance)	
	int16	
	uint16	
	int32	✓
	uint32	✓

See Also

Working with MPlay	Video and Image Processing Blockset
To Multimedia File	Signal 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