# Projective Transformation

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<br>• Single-precision floating point<br>• Fixed point<br>• Boolean<br>• 8-, 16-, 32-bit signed integer<br>• 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 |
| 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.)<br>• 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.)<br>• 8-, 16-, 32-bit signed integer<br>• 8-, 16-, 32-bit unsigned integer<br><br>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 |

| Port | Output | Supported Data Types | Complex Values Supported |
|---|---|---|---|
| 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 |
| 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.

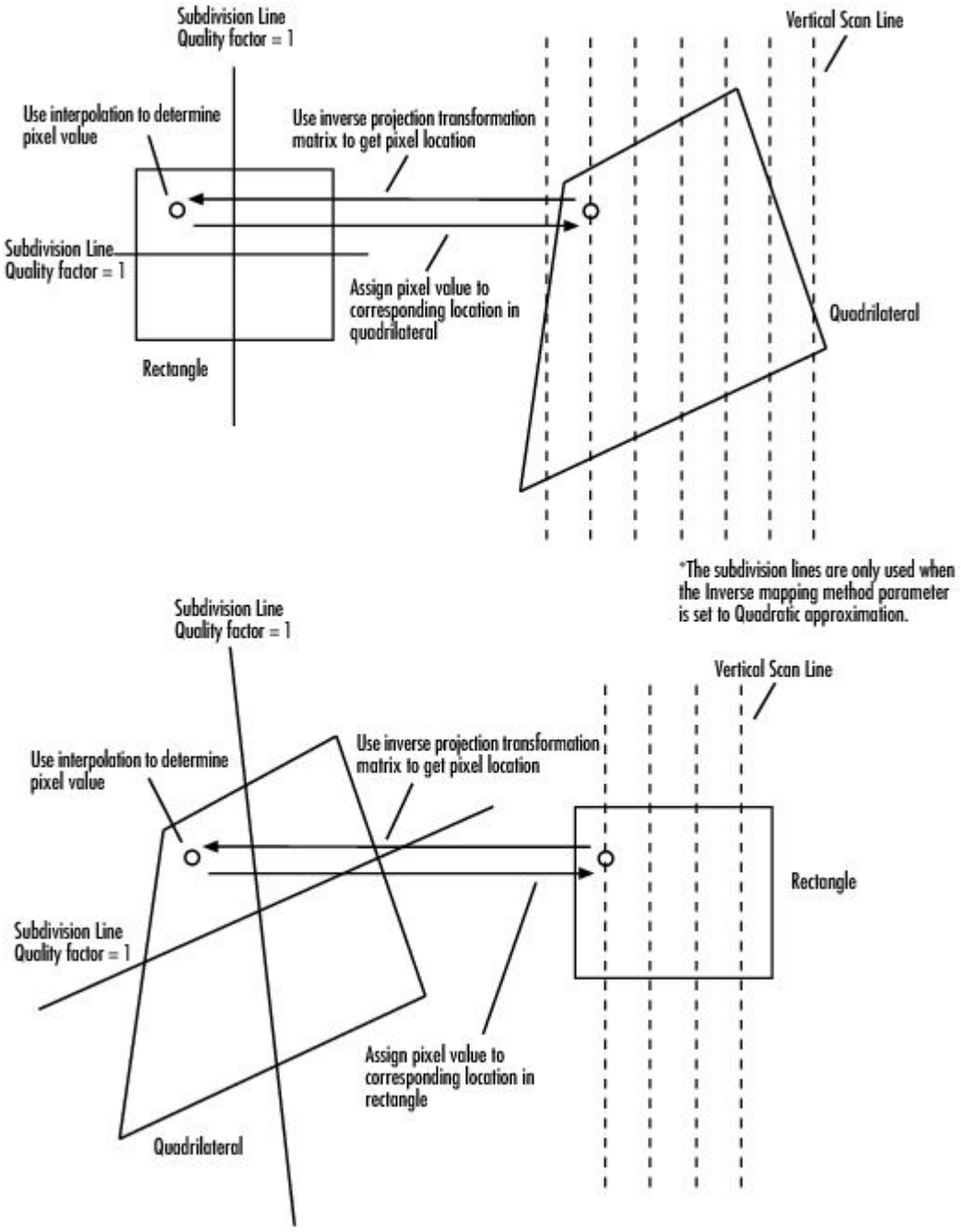
The following sections summarize the behavior of the Projective Transformation block in its three modes.

### 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.




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.

**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.

**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.

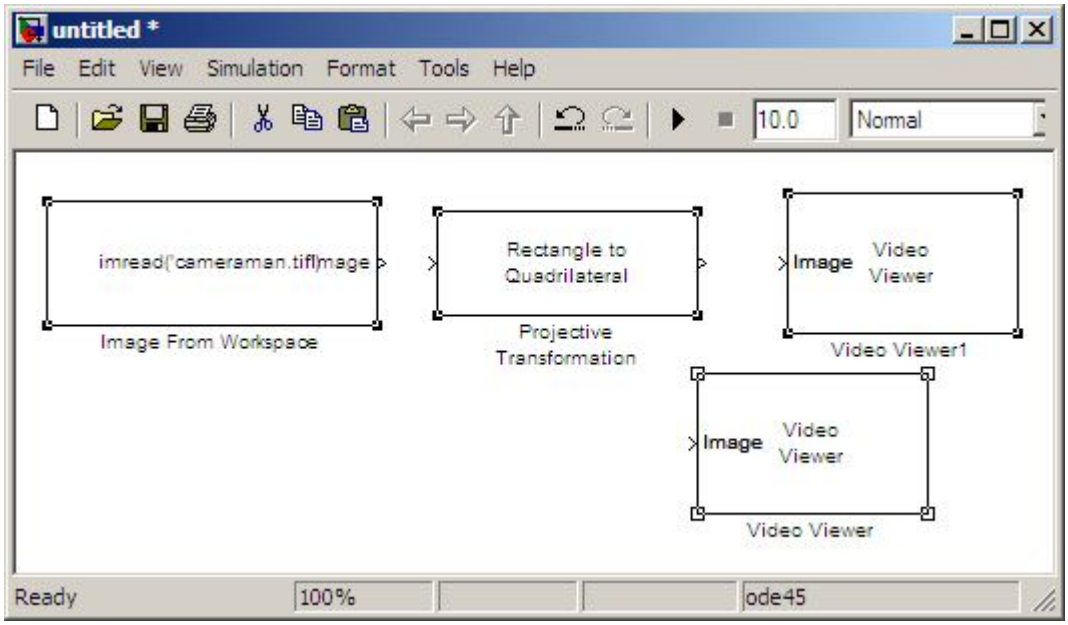
## Example

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.

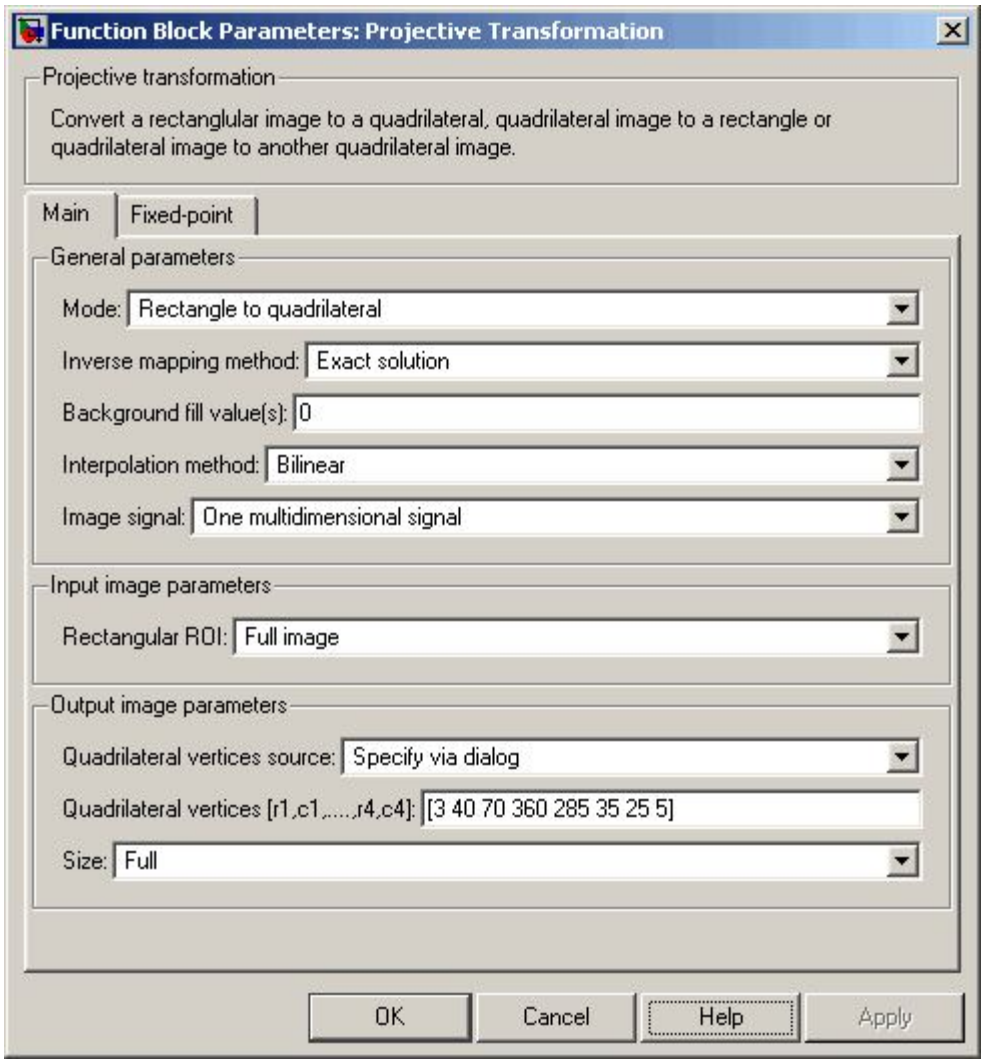
1. Create a new Simulink model.
2. Click-and-drag the following blocks into your model.

| 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.



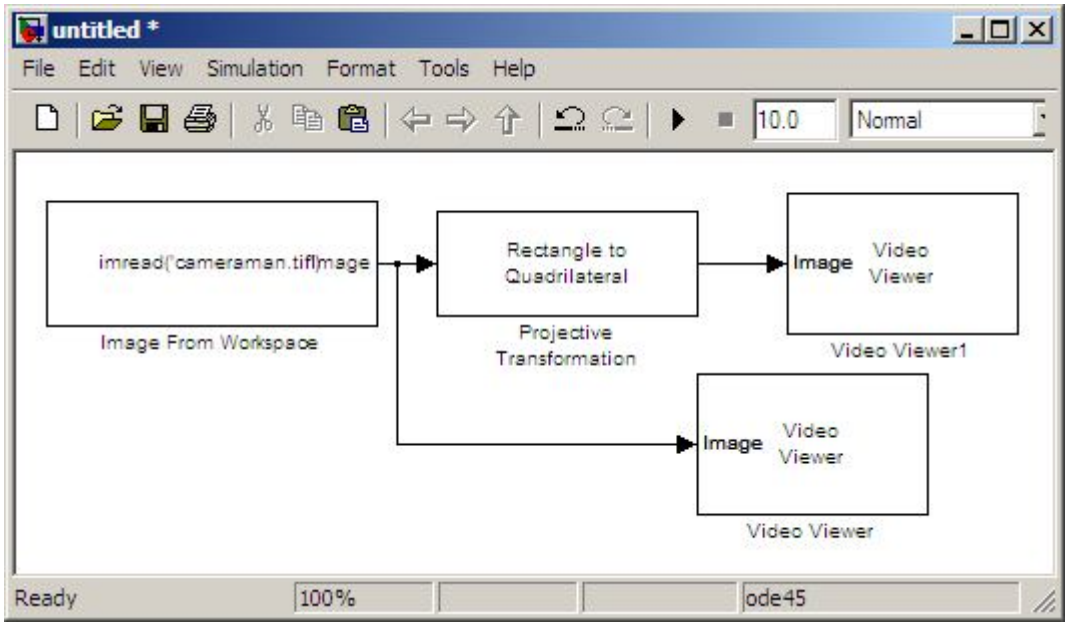

4. Use the Image From Workspace block to import an image into your model.
   - Set the **Value** parameter to `imread('cameraman.tif')`.
5. Use the Projective Transformation block to transform your rectangular image into a quadrilateral.
   - Set the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter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




**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

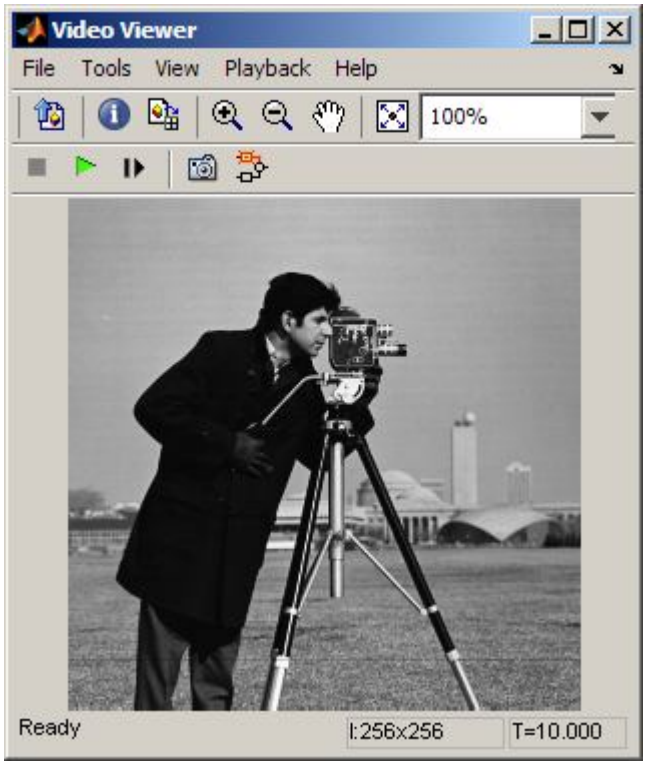
clockwise direction around the image.

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.

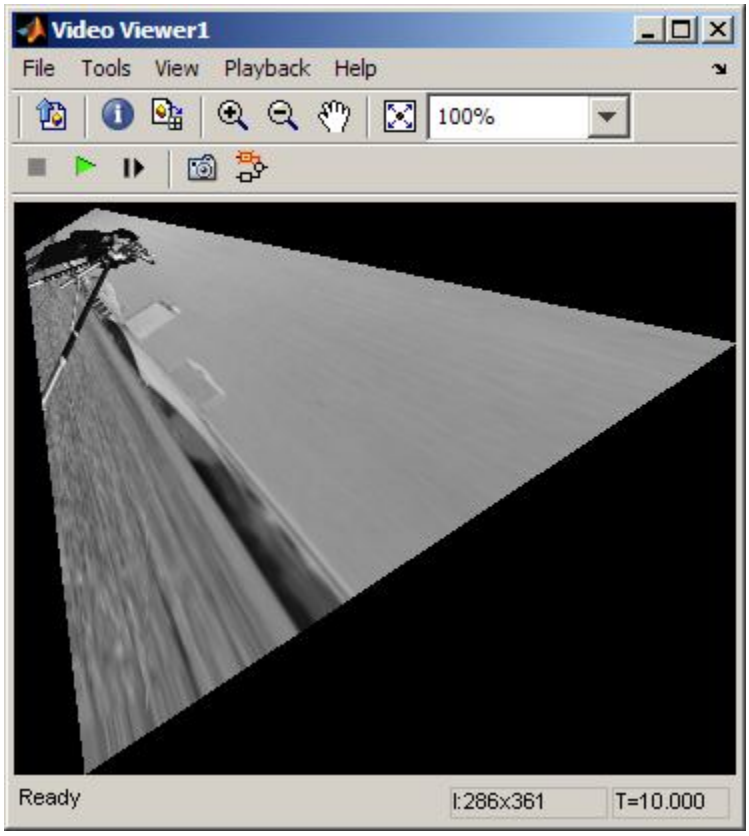



8. Run the model.

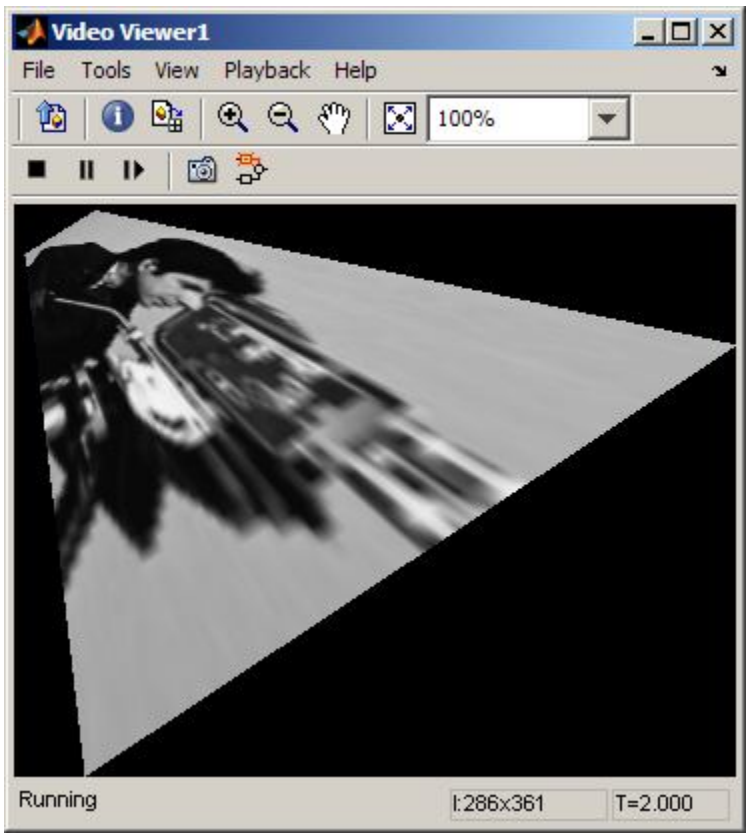
   The original rectangular image appears in the Video Viewer1 window.




   The quadrilateral image appears in the Video Viewer window.



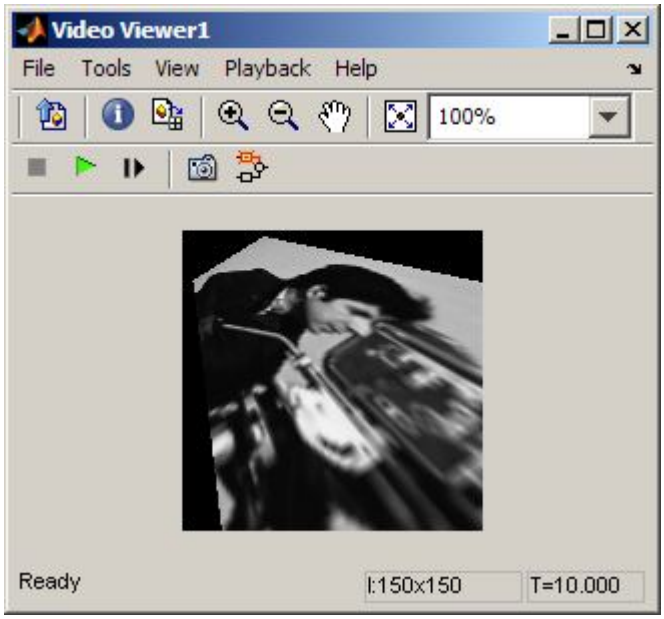

9. You can change the dimensions of the input image using the parameters in the **Input image parameters** section of the Projective Transformation dialog box. Set the block parameters as follows:
    - **Rectangular ROI** = `User-defined`
    - **ROI [r,c,height,width]** = `[30 20 100 140]`
10. Run 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]`

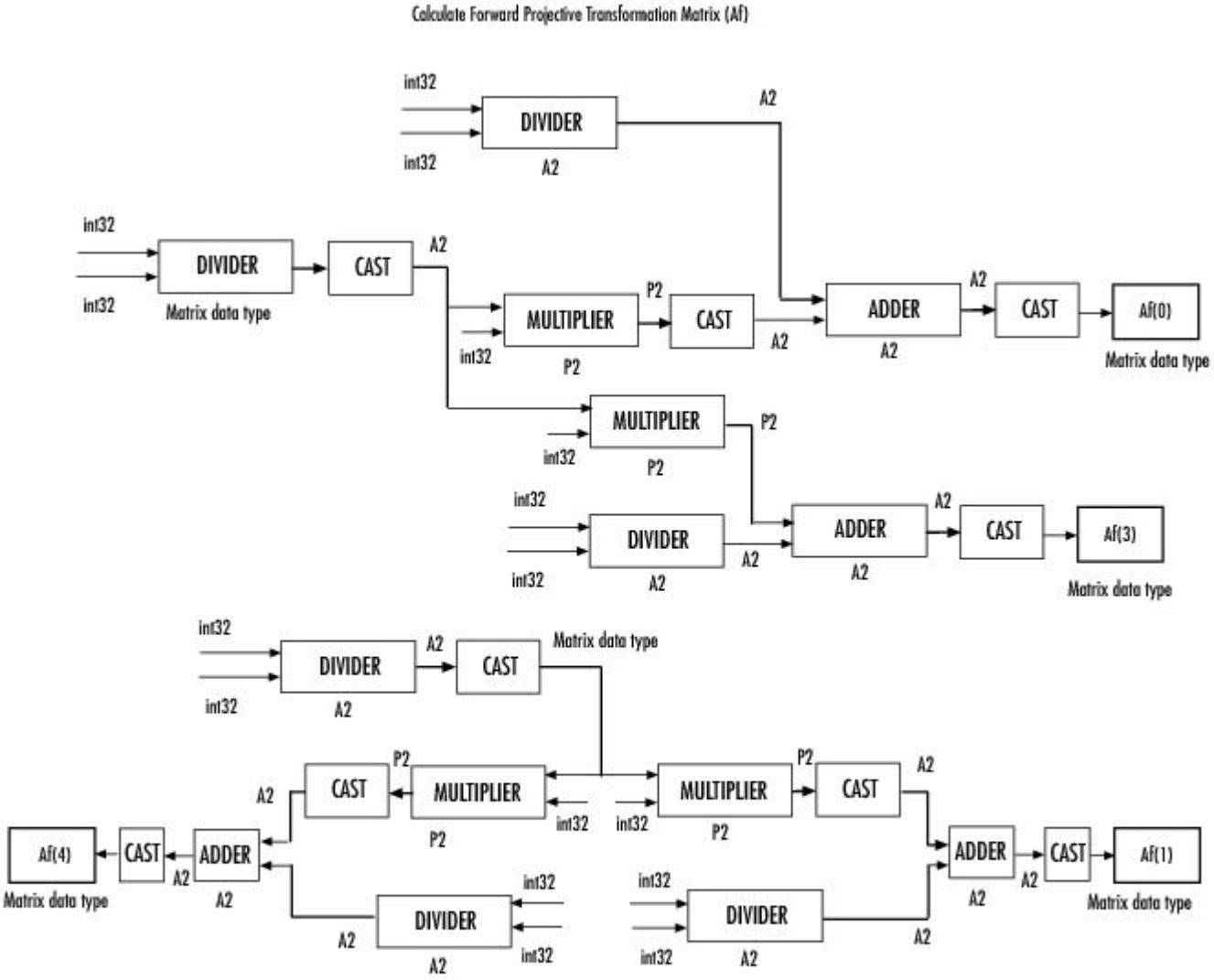
The **Location and size [r,c,height,width]** parameter defines the row and column coordinates as well as the height and width of the output image.

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.

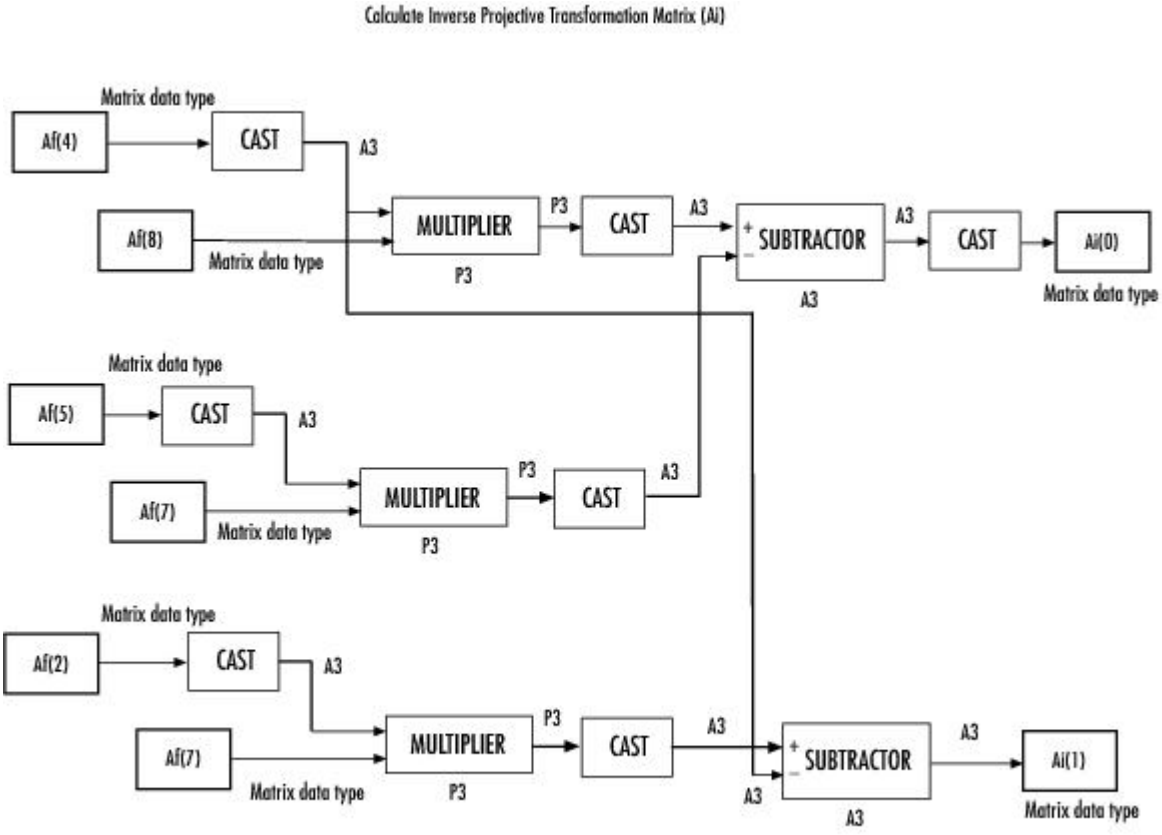



## Fixed-Point Data Types

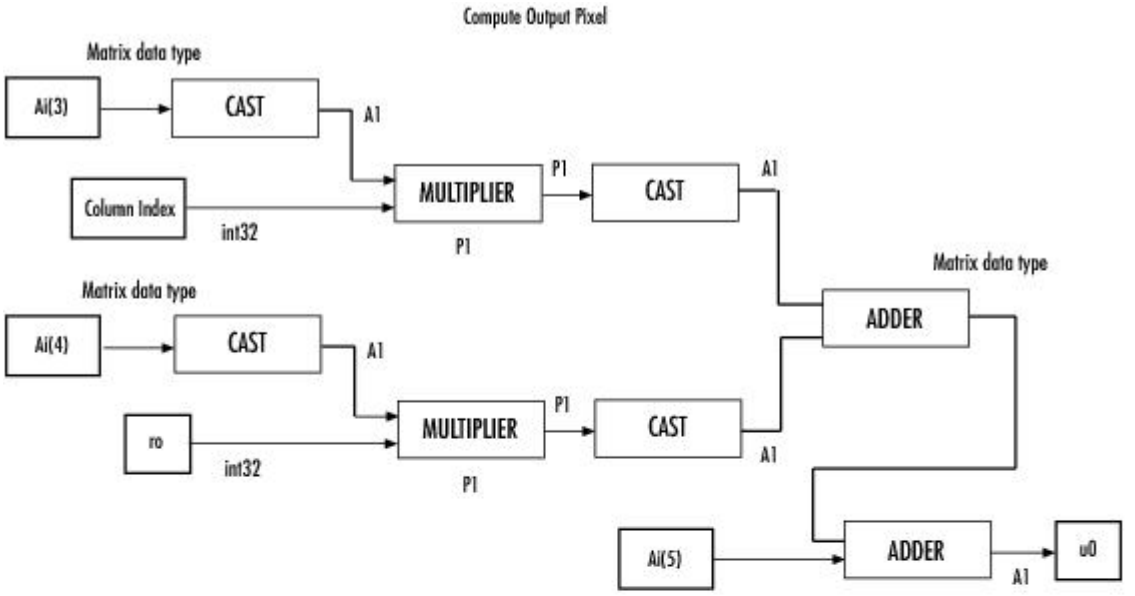
The following diagram shows the data types used in the Projective Transformation block for fixed-point signals:

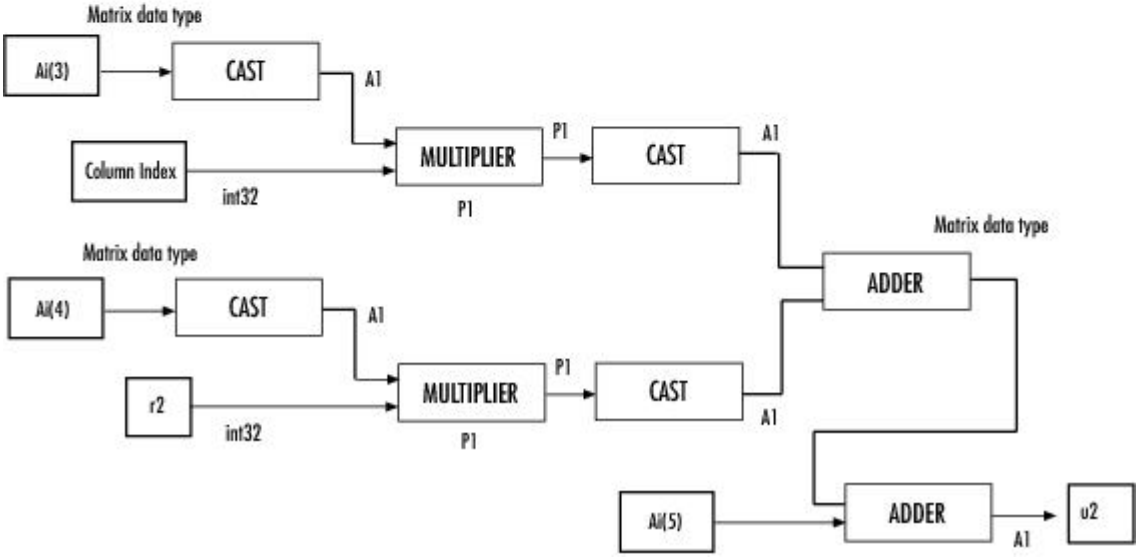


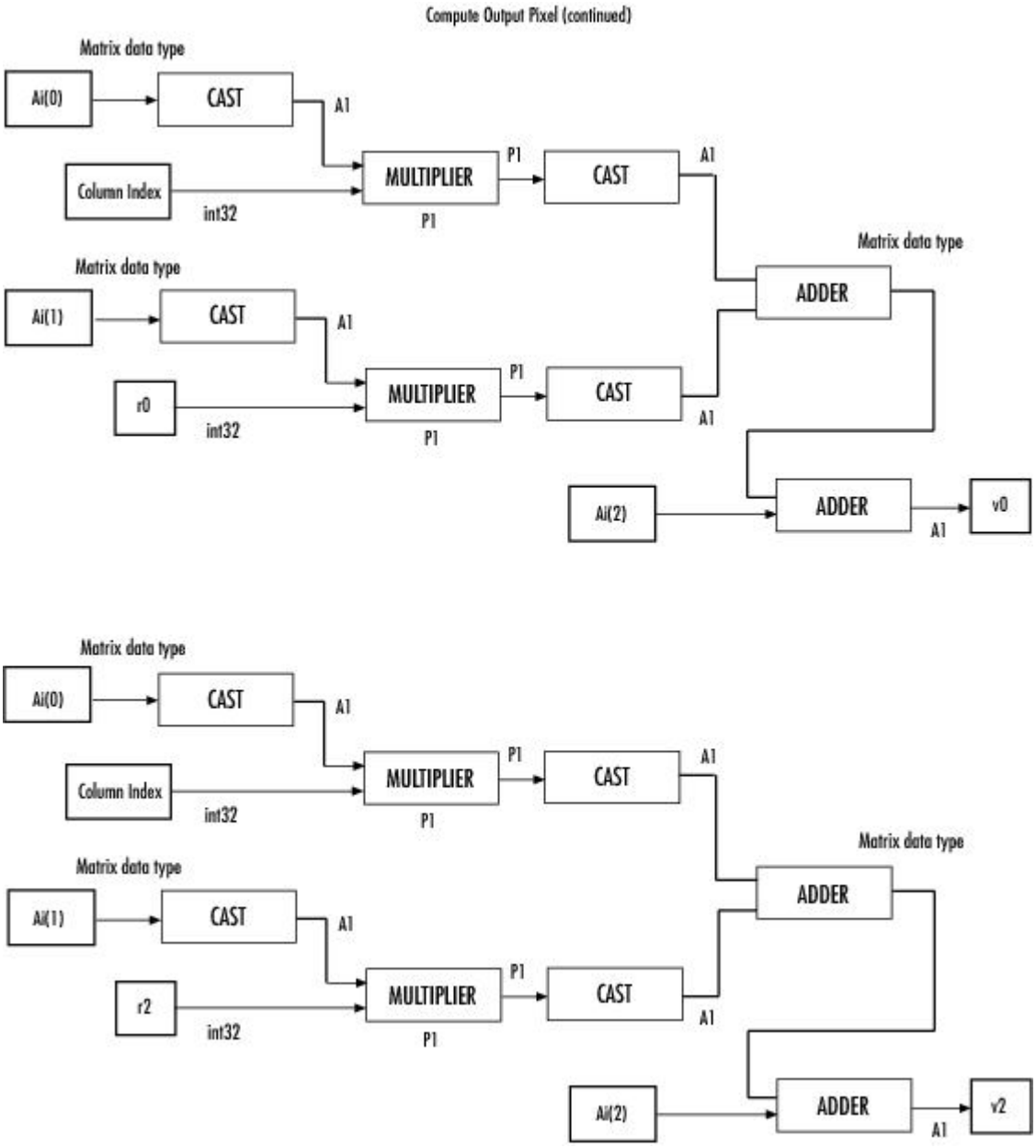

The block uses a similar computation to calculate Af(2), Af(4), Af(5), Af(6), Af(7), and Af(8).



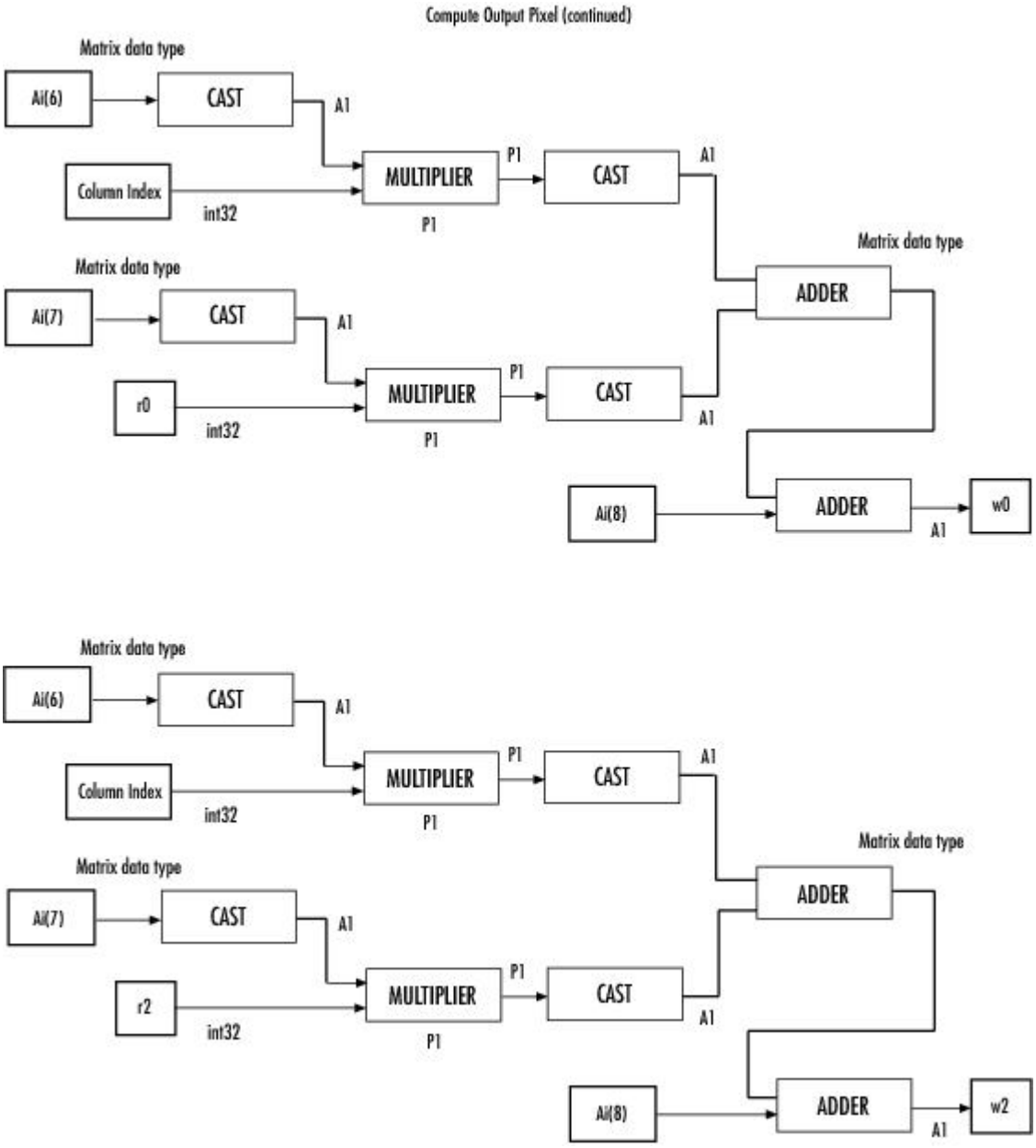

The block uses a similar computation to calculate Ai(2), Ai(3), Ai(4), Ai(5), Ai(6), Ai(7), and Ai(8).


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
A1
u0

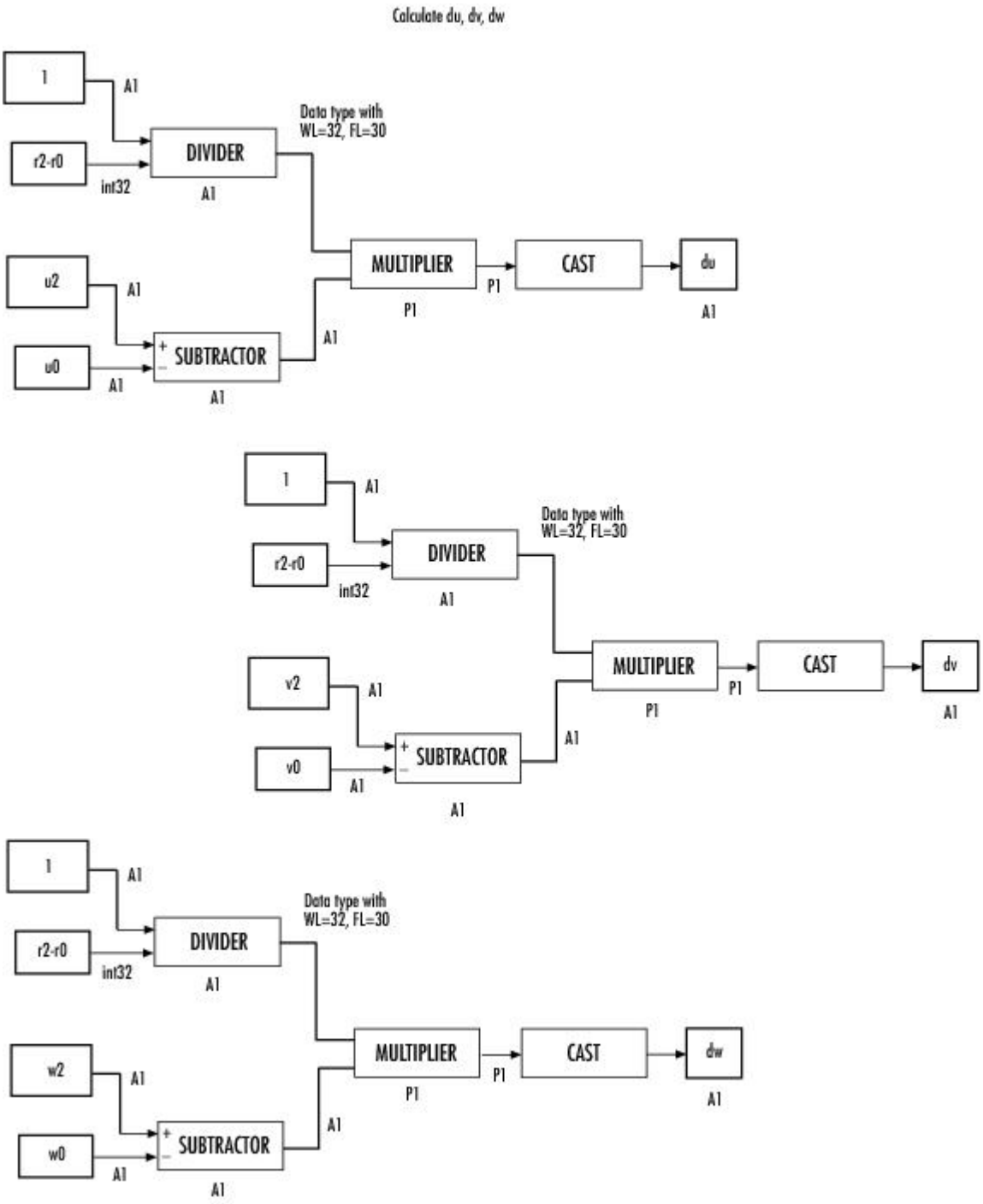

Matrix data type
Ai(3)
CAST
A1
Column Index
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
ADDER
Matrix data type
Ai(4)
CAST
A1
r2
int32
MULTIPLIER
P1
P1
CAST
A1
Ai(5)
ADDER
A1
u2


Compute Output Pixel (continued)
Matrix data type
Ai(0)
CAST
A1
Column Index
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
ADDER
Matrix data type
Ai(1)
CAST
A1
r0
int32
MULTIPLIER
P1
P1
CAST
A1
Ai(2)
ADDER
A1
v0

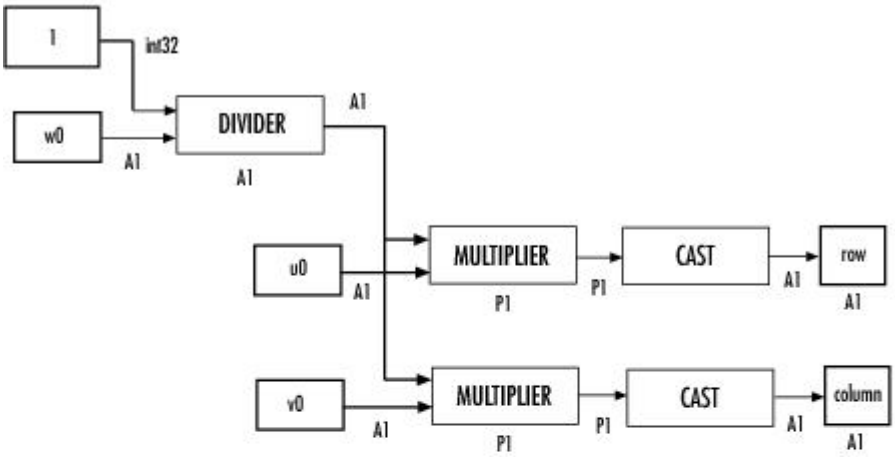

Matrix data type
Ai(0)
CAST
A1
Column Index
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
ADDER
Matrix data type
Ai(1)
CAST
A1
r2
int32
MULTIPLIER
P1
P1
CAST
A1
Ai(2)
ADDER
A1
v2

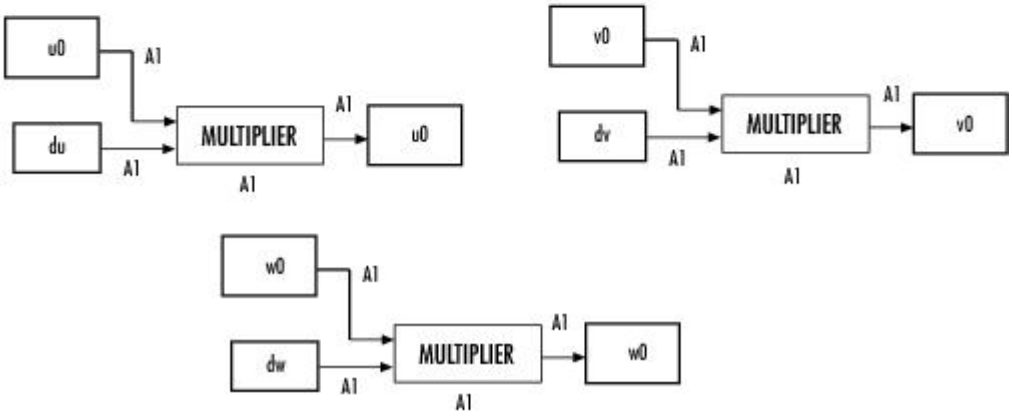

Compute Output Pixel (continued)
Matrix data type
Ai(6)
CAST
A1
Column Index
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
ADDER
Matrix data type
Ai(7)
CAST
A1
r0
int32
MULTIPLIER
P1
P1
CAST
A1
Ai(8)
ADDER
A1
w0


Matrix data type
Ai(6)
CAST
A1
Column Index
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
ADDER
Matrix data type
Ai(7)
CAST
A1
r2
int32
MULTIPLIER
P1
P1
CAST
A1
Ai(8)
ADDER
A1
w2

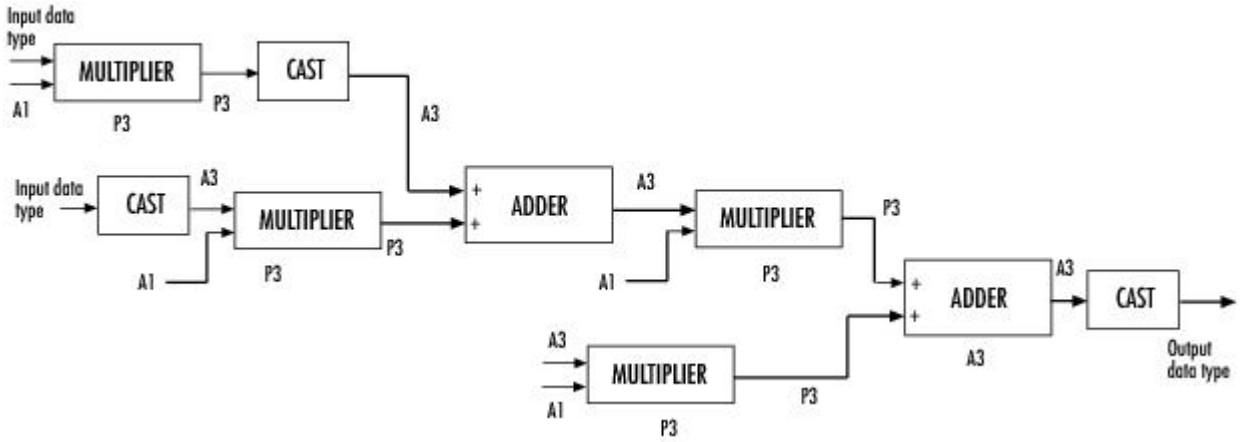

Calculate du, dv, dw
1
A1
r2-r0
int32
DIVIDER
A1
Data type with
WL=32, FL=30
MULTIPLIER
P1
CAST
du
A1
u2
u0
SUBTRACTOR
A1
v2
v0
dv
w2
w0
dw

Calculate row and column indices of the input image



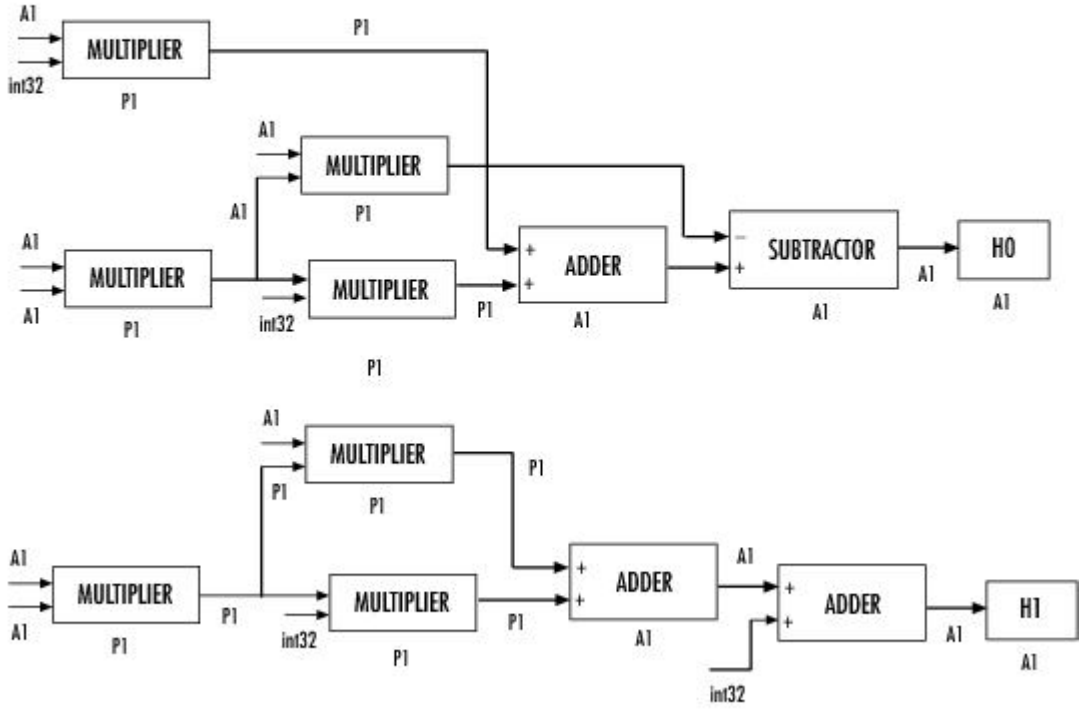




Bilinear Interpolation



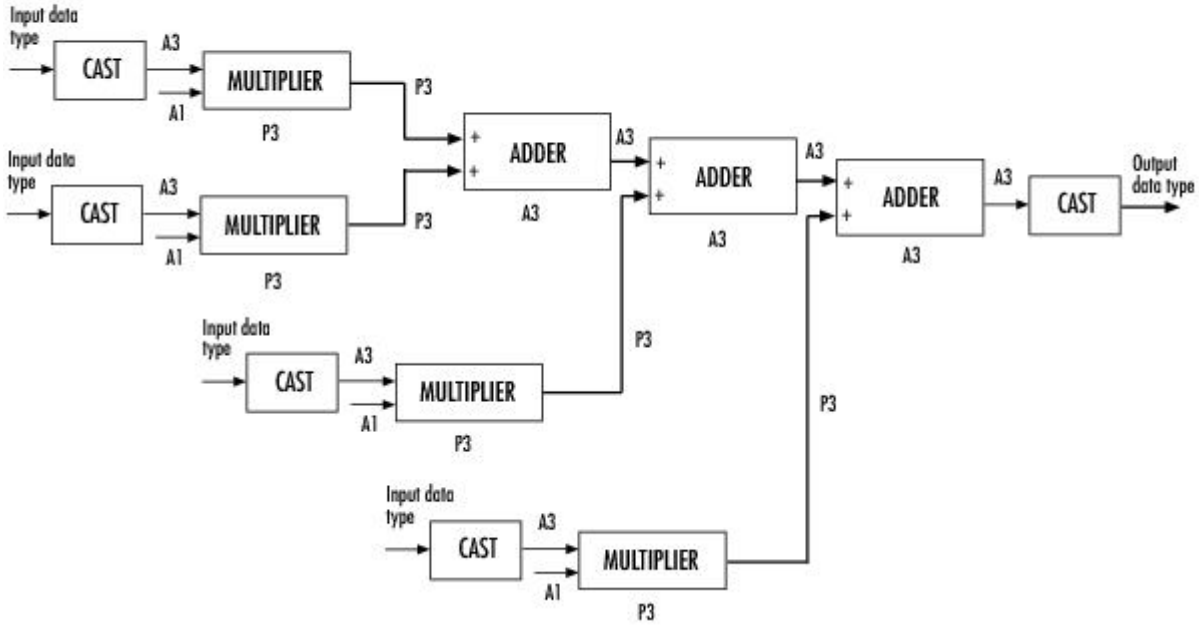
Bicubic Interpolation

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




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

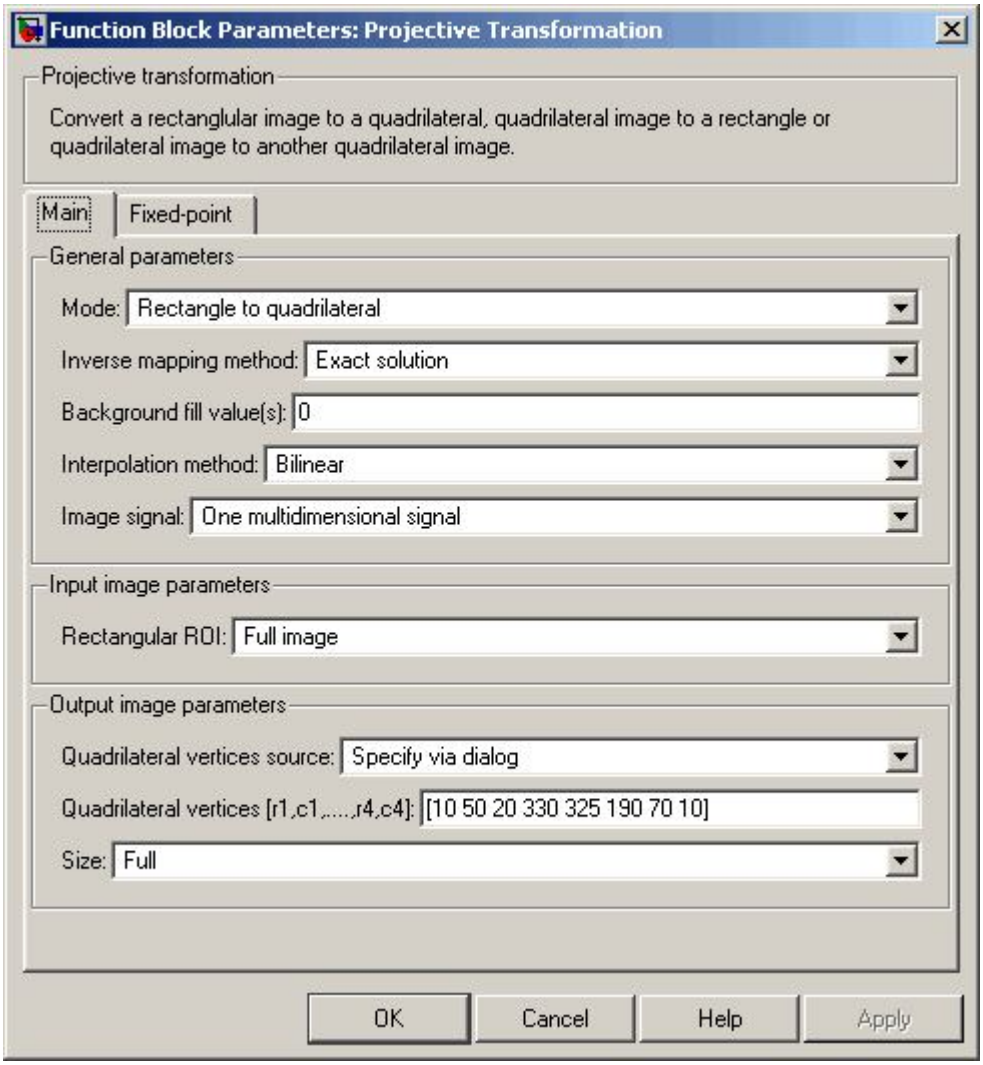
Bicubic Interpolation (continued)




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.




**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.

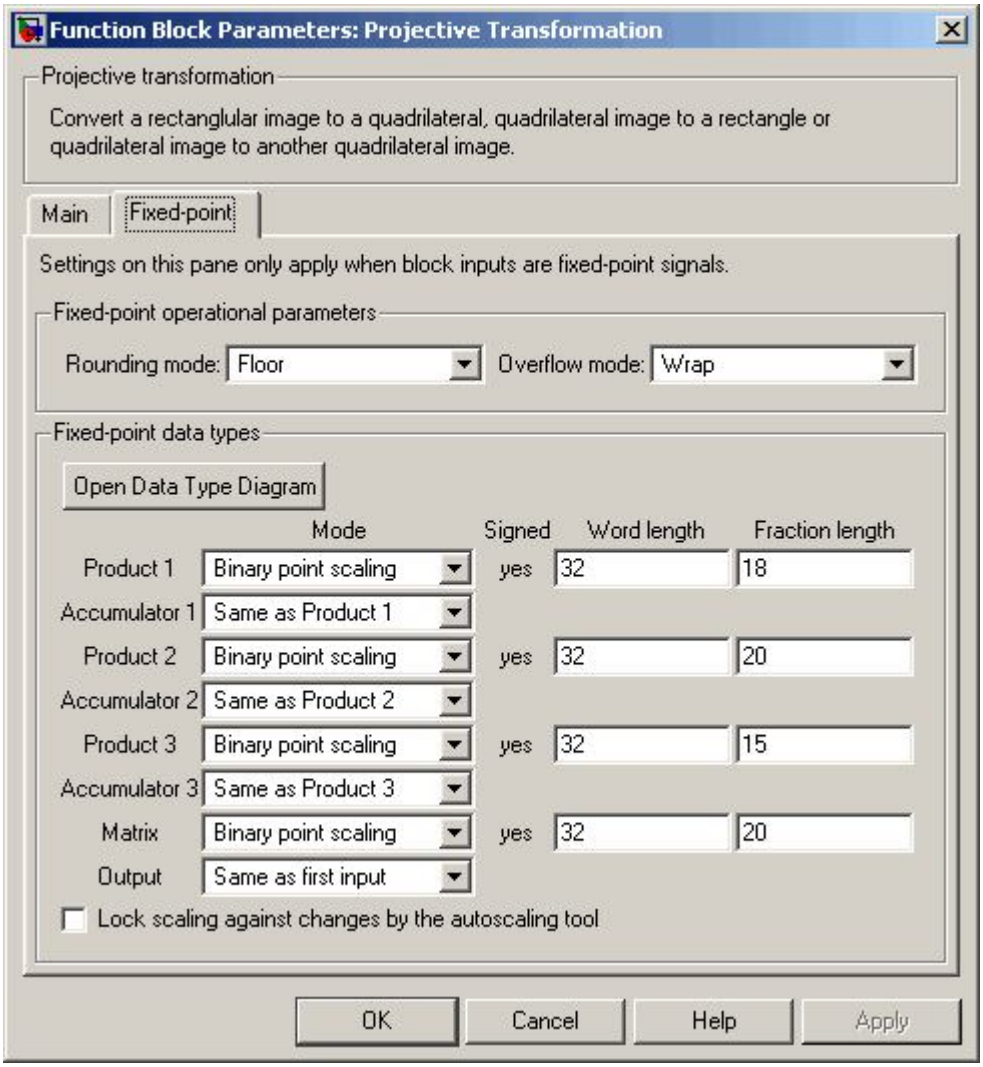
**Rectangle size source**

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.




**Rounding mode**

Select the rounding mode for fixed-point operations. For Boolean input, the **Product 3** and **Accumulator 3** **Rounding mode** parameter is always set to `Nearest`.

**Overflow mode**

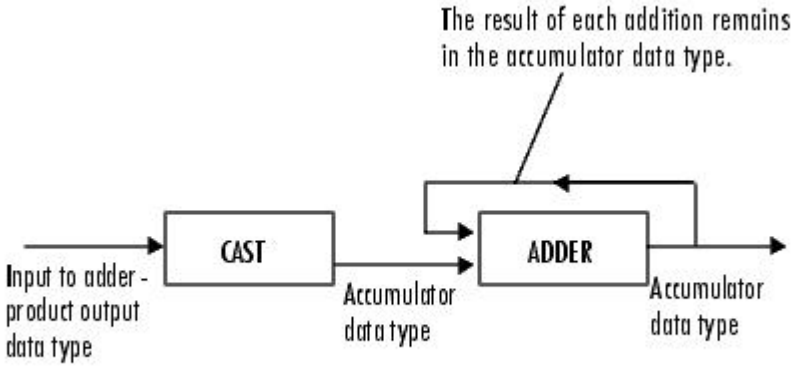
Select the overflow mode for fixed-point operations.

**Product 1, 2, 3**


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

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

**Accumulator 1, 2, 3, 4**




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

- When you select `Same as Product 1, 2, 3`, these characteristics match those of the product output.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator in bits.
- When you select `Slope and bias scaling`, you can enter the word length in bits and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset blocks is 0.

**Matrix**

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

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

**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 |

Optical Flow PSNR

© 1984-2009 The MathWorks, Inc. • Terms of Use • Patents • Trademarks • Acknowledgments