Video and Image Processing Blockset

For Use with Simulink®



Telegram.me/expertonline
مرکز شبیه سازی مقالات مرجع و
معتبر

- Modeling
- Simulation
- Implementation

کانال تخصصی
پردازش تصویر و هوش مصنوعی
دانشگاه تهران
Telegram.me/digital_image_processing

User's Guide

Version 2

The MathWorks

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 User's Guide

© COPYRIGHT 2004–2006 by The MathWorks, Inc.

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

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

Trademarks

MATLAB, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, and xPC TargetBox are registered trademarks, and SimBiology, SimEvents, and SimHydraulics are trademarks of The MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Patents

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

Revision History

July 2004	First printing	New for Version 1.0 (Release 14)
October 2004	Second printing	Revised for Version 1.0.1 (Release 14SP1)
March 2005	Online only	Revised for Version 1.1 (Release 14SP2)
September 2005	Online only	Revised for Version 1.2 (Release 14SP3)
November 2005	Online only	Revised for Version 2.0 (Release 14SP3+)
March 2006	Online only	Revised for Version 2.1 (Release 2006a)
September 2006	Online only	Revised for Version 2.2 (Release 2006b)

Contents

1 Getting Started

What Is the Video and Image Processing Blockset? . . . **1-2**

Installation . . . **1-3**
- Installing the Video and Image Processing Blockset . . . **1-3**
- Required Products . . . **1-4**
- Related Products . . . **1-4**

Product Demos . . . **1-5**
- Demos in the Help Browser . . . **1-5**
- Demos on the Web . . . **1-6**
- Demos on MATLAB Central . . . **1-6**

Working with the Documentation . . . **1-7**
- Viewing the Documentation . . . **1-7**
- Printing the Documentation . . . **1-8**
- Using This Guide . . . **1-8**

Key Blockset Concepts . . . **1-11**
- Image Types . . . **1-11**
- Video in the Video and Image Processing Blockset . . . **1-12**
- Defining Intensity and Color . . . **1-12**
- Coordinate Systems . . . **1-13**
- Image Data Stored in Column-Major Format . . . **1-15**
- Sample Time . . . **1-16**
- Video Duration and Simulation Time . . . **1-17**
- Normal and Accelerator Mode . . . **1-18**
- Windows Dynamic Library Dependencies . . . **1-19**

Block Data Type Support . . . **1-21**

Image Credits . . . **1-26**

2 Importing and Exporting Video

Working with AVI Files **2-2**
Importing and Viewing AVI Files **2-2**
Exporting to AVI Files **2-5**
Annotating AVI Files **2-9**

Working with Multimedia Files **2-14**
Importing and Viewing Multimedia Files **2-14**
Exporting to Multimedia Files **2-17**
Working with Audio **2-20**

3 Working with MPlay

Viewing Videos from the MATLAB Workspace **3-2**

Viewing Video Files **3-6**

Viewing Video Signals in Simulink **3-8**

4 Conversions

Intensity to Binary Conversion **4-2**
Thresholding Intensity Images Using Relational Operators **4-2**
Thresholding Intensity Images Using the Autothreshold Block **4-7**

Color Space Conversion **4-14**
Converting Color Information from R'G'B' to Intensity . . . **4-14**

Chroma Resampling **4-19**

5 Geometric Transformation

Interpolation Overview **5-2**
Nearest Neighbor Interpolation **5-2**
Bilinear Interpolation **5-3**
Bicubic Interpolation **5-4**

Rotating an Image **5-6**

Resizing an Image **5-14**

Cropping an Image **5-21**

6 Morphological Operations

Overview of Morphology **6-2**

Counting Objects in an Image **6-3**

Correcting for Nonuniform Illumination **6-11**

7 Analysis and Enhancement

Feature Extraction **7-2**
Finding Edges in Images **7-2**
Finding Lines in Images **7-9**
Measuring an Angle Between Lines **7-17**

Image Enhancement **7-26**
Sharpening and Blurring an Image **7-26**
Removing Salt and Pepper Noise from Images **7-33**
Removing Periodic Noise from Video **7-39**
Adjusting the Contrast in Intensity Images **7-46**
Adjusting the Contrast in Color Images **7-52**

Pixel Statistics **7-58**
Finding the Histogram of an Image **7-58**

8 Example Applications

Pattern Matching **8-2**
Tracking an Object Using Correlation **8-2**

Motion Compensation **8-9**

Image Compression **8-11**
Compressing an Image **8-11**
Viewing the Compressed Image **8-18**

9 Blocks — By Category

Analysis & Enhancement **9-2**

Conversions **9-2**

Filtering **9-3**

Geometric Transformations **9-3**

Morphological Operations **9-4**

Sinks **9-4**

Sources **9-5**

Statistics **9-5**

Text & Graphics **9-6**

Transforms **9-6**

Utilities **9-7**

10 Blocks — Alphabetical List

11 Functions — Alphabetical List

Index

1

Getting Started

The Video and Image Processing Blockset is a tool for processing images and video in the Simulink® environment. This chapter provides an introduction to the Video and Image Processing Blockset, its product requirements, and its documentation.

What Is the Video and Image Processing Blockset? (p. 1-2)	Learn more about the Video and Image Processing Blockset and its components.
Installation (p. 1-3)	Install the Video and Image Processing Blockset and learn about the products required to run the models in this manual.
Product Demos (p. 1-5)	View the demos available in the product and on the Web.
Working with the Documentation (p. 1-7)	Learn how to view and print the documentation.
Key Blockset Concepts (p. 1-11)	Understand how your image and video data is interpreted within the Simulink environment.
Block Data Type Support (p. 1-21)	Learn which data types are supported by each Video and Image Processing Blockset block.
Image Credits (p. 1-26)	View a list of the copyright owners of the images used in the Video and Image Processing Blockset documentation.

What Is the Video and Image Processing Blockset?

The Video and Image Processing Blockset is a tool used for the rapid design, prototyping, graphical simulation, and efficient code generation of video processing algorithms. The Video and Image Processing Blockset blocks can import streaming video into the Simulink environment and perform two-dimensional filtering, geometric and frequency transforms, block processing, motion estimation, edge detection and other signal processing algorithms. You can also use the blockset in conjunction with Real-Time Workshop® to automatically generate embeddable C code for real-time execution.

The Video and Image Processing Blockset blocks support floating-point, integer, and fixed-point data types. To use any data type other than double-precision and single-precision floating point, you must install Simulink Fixed Point. For more information about this product, see the Simulink Fixed Point documentation.

Installation

This section describes how to install the Video and Image Processing Blockset software and documentation. It also reviews the other MathWorks products you must install to run the Video and Image Processing Blockset.

This section includes the following topics:

- "Installing the Video and Image Processing Blockset" on page 1-3 — Learn how to install the Video and Image Processing Blockset and its documentation from a CD or a Web download
- "Required Products" on page 1-4 — Links to products you must install to run the Video and Image Processing Blockset
- "Related Products" on page 1-4 — Links to other products that are relevant to the kinds of tasks you can perform with the Video and Image Processing Blockset

Installing the Video and Image Processing Blockset

Before you begin working with the Video and Image Processing Blockset, you need to install the product on your computer.

Installation from a CD

The Video and Image Processing Blockset follows the same installation procedure as the MATLAB® toolboxes:

1. Start the MathWorks installer.
2. When prompted, select the **Product** check boxes for the products you want to install.

   The documentation is installed along with the products.

Installation from a Web Download

You can use your MathWorks Account to download products from the MathWorks Web site:

1. Navigate to `http://www.mathworks.com/web_downloads/`.

2 Click **Download products**.

3 Log in to the system using your MathWorks Account e-mail and password. If you do not have a MathWorks Account, you can create one from this Web page.

4 Select your platform and the products you want to install.

5 Follow the instructions on the **Download and Install** screen, which describe how to download the product(s) and the installer.

6 Double-click the `Installer.exe` file to run the installer.

7 When prompted, enter your Personal License Password.

8 Select the **Product** check boxes for the products you want to install.

The documentation is installed along with the products.

Required Products

The Video and Image Processing Blockset is part of a family of products from The MathWorks. You need to install several products to use the Video and Image Processing Blockset. For more information, see the MathWorks Web site at `http://www.mathworks.com/products/viprocessing/requirements.jsp`.

Related Products

The MathWorks provides several products that are relevant to the kinds of tasks you can perform with the Video and Image Processing Blockset.

For more information about any of these products, see either

- The online documentation for that product if it is installed on your system
- The MathWorks Web site, at `http://www.mathworks.com/products/viprocessing/related.jsp`

Product Demos

The Video and Image Processing Blockset has a number of demo models that solve real-world problems. Begin viewing Video and Image Processing Blockset demos by using the MATLAB Help browser. For additional demo models, navigate to the MathWorks and MATLAB Central Web sites.

This section includes the following topics:

- "Demos in the Help Browser" on page 1-5 -- View and interact with Video and Image Processing Blockset product demos in the Help browser
- "Demos on the Web" on page 1-6 -- View Video and Image Processing Blockset Web demos on the MathWorks Web site
- "Demos on MATLAB Central" on page 1-6 -- View user and developer contributed Video and Image Processing Blockset demos on the MATLAB Central Web site

Demos in the Help Browser

You can find interactive Video and Image Processing Blockset demos is the MATLAB Help browser. This example shows you how to locate and open a typical demo:

1. To open the Help browser to the **Demos** tab, type `demos` at the MATLAB command line.
2. On the left side of the Help browser, double-click **Blocksets**, and then double-click **Video and Image Processing** to see a list of demo categories.
3. The Pattern matching demo, which demonstrates object tracking in a video stream, is a typical Video and Image Processing Blockset demo. To view the description of this demo, double-click **Detection and Tracking**, and then click **Pattern matching**.
4. Click **Open this model** to display the Simulink model for this demo. Run the model by selecting **Start** from the **Simulation** menu in the model window.

Demos on the Web

The MathWorks Web site contains demos that show you how to use the Video and Image Processing Blockset. You can find these demos at `http://www.mathworks.com/products/viprocessing/demos.jsp`.

You can run these demos without having MATLAB or the Video and Image Processing Blockset installed on your system.

Demos on MATLAB Central

MATLAB Central contains files, including demos, contributed by users and developers of the Video and Image Processing Blockset, MATLAB, Simulink and other products. Contributors submit their files to one of a list of categories. You can browse these categories to find submissions that pertain to the Video and Image Processing Blockset or a specific problem that you would like to solve. MATLAB Central is located at `http://www.mathworks.com/matlabcentral/`.

Working with the Documentation

The Video and Image Processing Blockset documentation includes the Video and Image Processing Blockset User's Guide. You can access this documentation using the MATLAB Help browser or on the MathWorks Web site.

This section includes the following topics:

- "Viewing the Documentation" on page 1-7 -- View HTML files on your system or the MathWorks Web site
- "Printing the Documentation" on page 1-8 -- Locate and print PDF files on the MathWorks Web site
- "Using This Guide" on page 1-8 -- Suggestions for learning about the Video and Image Processing Blockset and a description of the chapters in this manual

Viewing the Documentation

You can access the Video and Image Processing Blockset documentation using files you installed on your system or from the Web using the MathWorks Web site.

Documentation in the Help Browser

This procedure shows you how to use the MATLAB Help browser to view the Video and Image Processing Blockset documentation installed on your system:

1. In the MATLAB window, from the **Help** menu, click **Full Product Family Help**. The Help browser opens.
2. From the list of products in the left pane, click **Video and Image Processing Blockset**. In the right pane, the Help browser displays the Video and Image Processing Blockset Roadmap page.
3. Under the section titled **Documentation Set**, select **User's Guide**. The Help browser displays the chapters of this manual.

The Help browser also has a **Demos** tab where you can view product demos. For more information, see "Product Demos" on page 1-5.

Documentation on the Web

You can also view the documentation from the MathWorks Web site. The documentation available on these Web pages is for the latest release, regardless of whether the release was distributed on a CD or as a Web download:

1. Navigate to the Video and Image Processing Blockset Product page at `http://www.mathworks.com/products/viprocessing/`.
2. On the right side of the page, click the **Documentation** link. The Video and Image Processing Blockset documentation is displayed.

Printing the Documentation

The documentation for the Video and Image Processing Blockset is also available in printable PDF format. You need to install Adobe Acrobat Reader 4.0 or later to open and read these files. To download a free copy of Acrobat Reader, see `http://www.adobe.com/products/acrobat/main.html`.

The following procedure shows you how to view the documentation in PDF format:

1. In the MATLAB window, from the **Help** menu, click **Full Product Family Help**. The Help browser opens.
2. From the list of products in the left pane, click **Video and Image Processing Blockset**. In the right pane, the Help browser displays the Video and Image Processing Blockset Roadmap page.
3. Under the **Printing the Documentation Set** heading, click the links to view PDF versions of the Video and Image Processing Blockset documentation.

Using This Guide

To help you effectively read and use this guide, here is a brief description of the chapters and a suggested reading path.

Expected Background

This manual assumes that you are familiar with the following:

- MATLAB, to write scripts and functions with M-code, and to use functions with the command-line interface
- Simulink, to create simple models as block diagrams and simulate those models

What Chapter Should I Read?

Follow the procedures in this guide to become familiar with the blockset's functionality. The User's Guide contains tutorial sections that are designed to help you become familiar with using Simulink and the Video and Image Processing Blockset:

- Read Chapter 1, "Getting Started" to learn about the installation process, the products required to run the Video and Image Processing Blockset, and to view Video and Image Processing Blockset demos.
- Read Chapter 2, "Importing and Exporting Video" to understand how video is interpreted by Simulink. You also learn how to bring video data into a model, display it on your monitor, and export it to an AVI file.
- Read Chapter 3, "Working with MPlay" to learn how to use the MPlay GUI to view videos that are represented as variables in the MATLAB workspace. You can also learn how to use it to view video files or video signals in Simulink models.
- Read Chapter 4, "Conversions" to learn how to convert an intensity image to a binary image, how to convert color information between color spaces, and how to downsample the chroma components of an image.
- Read Chapter 5, "Geometric Transformation" to understand how blocks in the Geometric Transformations library interpolate values. You also learn how to rotate, resize, and crop images.
- Read Chapter 6, "Morphological Operations" to learn about morphological operations and which blocks can be used to perform them. For example, you learn how to count objects in an image and correct for nonuniform illumination.
- Read Chapter 7, "Analysis and Enhancement" to learn how to sharpen, blur, and remove noise from images. You also learn how to find object boundaries and calculate the histogram of the R, G, and B values in an image.

- Read Chapter 8, “Example Applications” to learn how to track the motion of an object in a video stream. Also, learn more about motion compensation and image compression.

For a description of each block’s operation, parameters, and characteristics, see the Block Reference in the Video and Image Processing Blockset documentation on the Web at `http://www.mathworks.com/products/viprocessing/` or in the Help browser.

Key Blockset Concepts

In this section, you learn how the Video and Image Processing Blockset blocks interpret input matrices and arrays. Images are real-valued ordered sets of color or intensity data. The blocks interpret input matrices as images, where each element of the matrix corresponds to a single pixel in the displayed image. Video data is a series of images over time. All blocks in the Video and Image Processing Blockset can process images or video data.

This section includes the following topics:

- "Image Types" on page 1-11 — Learn how to represent binary, intensity, and RGB images
- "Video in the Video and Image Processing Blockset" on page 1-12 — Understand the representation of video
- "Defining Intensity and Color" on page 1-12— Learn how data type determines which values correspond to black and white as well as the absence or saturation of color
- "Coordinate Systems" on page 1-13— Learn how pixel and spatial coordinate systems are defined in the Video and Image Processing Blockset
- "Image Data Stored in Column-Major Format" on page 1-15 — Learn how the blockset stores image data
- "Sample Time" on page 1-16 — Learn how a block's sample time determines when the code behind each block is executed
- "Video Duration and Simulation Time" on page 1-17 — Learn how to control the duration of the simulation
- "Normal and Accelerator Mode" on page 1-18 — Learn how to improve the performance of larger Simulink models
- "Windows Dynamic Library Dependencies" on page 1-19 — Understand when `vip_rt.dll` is needed to run executables generated for certain targets

Image Types

Images can be binary, intensity (grayscale), or RGB.

Binary Images

Binary images are represented by a Boolean matrix of 0s and 1s, which correspond to black and white pixels, respectively.

For more information, see “Binary Images” in the Image Processing Toolbox documentation.

Intensity Images

Intensity images are represented by a matrix of intensity values. While intensity images are not stored with colormaps, you can use a gray colormap to display them.

For more information, see “Grayscale Images” in the Image Processing Toolbox documentation.

RGB Images

RGB images are also known as a true-color images. With the Video and Image Processing Blockset, these images are represented by an array, where the first plane represents the red pixel intensities, the second plane represents the green pixel intensities, and the third plane represents the blue pixel intensities. Blocks that perform color operations expect three input matrices, one for each color component.

For more information, see “Truecolor Images” in the Image Processing Toolbox documentation.

Video in the Video and Image Processing Blockset

Video data is a series of images over time. Video in binary or intensity format is a series of single images. Video in RGB format is a series of matrices grouped into sets of three, where each matrix represents an R, G, or B plane.

Defining Intensity and Color

The values in a binary, intensity, or RGB image can be different data types. The data type of the image values determines which values correspond to black and white as well as the absence or saturation of color. The table below summarizes the interpretation of the upper and lower bound of each data

type. To view the data types of the signals at each port, from the **Format** menu, point to **Port/Signal Displays**, and select **Port Data Types**.

Data Type	Black or Absence of Color	White or Saturation of Color
Fixed point	Minimum data type value	Maximum data type value
Floating point	0	1

Note The Video and Image Processing Blockset considers any data type other than double-precision floating point and single-precision floating point to be fixed point.

For example, for an intensity image whose image values are 8-bit unsigned integers, 0 is black and 255 is white. For an intensity image whose image values are double-precision floating point, 0 is black and 1 is white. For an intensity image whose image values are 16-bit signed integers, -32768 is black and 32767 is white.

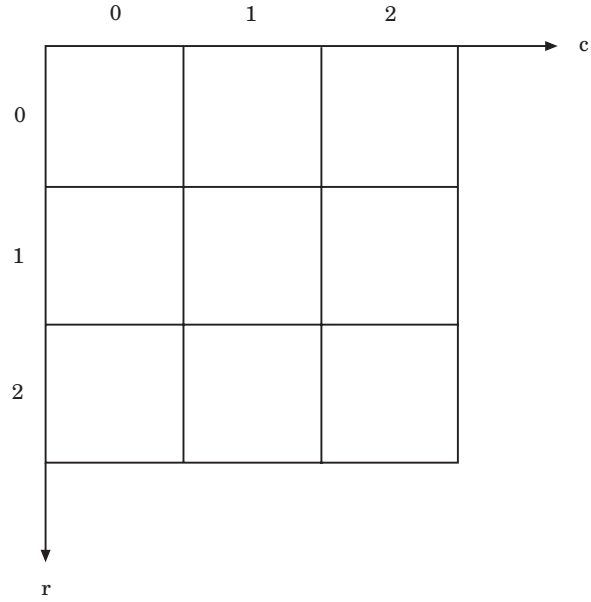
For an RGB image whose image values are 8-bit unsigned integers, 0 0 0 is black, 255 255 255 is white, 255 0 0 is red, 0 255 0 is green, and 0 0 255 is blue. For an RGB image whose image values are double-precision floating point, 0 0 0 is black, 1 1 1 is white, 1 0 0 is red, 0 1 0 is green, and 0 0 1 is blue. For an RGB image whose image values are 16-bit signed integers, -32768 -32768 -32768 is black, 32767 32767 32767 is white, 32767 -32768 -32768 is red, -32768 32767 -32768 is green, and -32768 -32768 32767 is blue.

Coordinate Systems

You can specify locations in images using various coordinate systems. This topic discusses pixel coordinates and spatial coordinates, which are the two main coordinate systems used in the Video and Image Processing Blockset.

Pixel Coordinates

Pixel coordinates enable you to specify locations in images. In this coordinate system, the image is treated as a grid of discrete elements, ordered from top to bottom and left to right, as shown in the following figure:

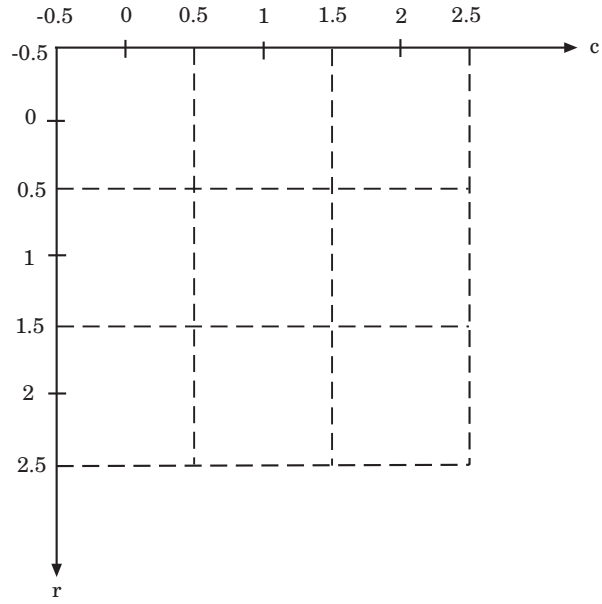


For pixel coordinates, the first component r (the row) increases downward, while the second component c (the column) increases to the right. Pixel coordinates are integer values and range from 0 to the length of the row or column. Note that the pixel coordinates used in the Video and Image Processing Blockset are zero based, while the pixel coordinates used by the Image Processing Toolbox and MATLAB are one based. For more information on the pixel coordinate system used by the Image Processing Toolbox, see "Pixel Coordinates" in the Image Processing Toolbox documentation.

Spatial Coordinates

Spatial coordinates enable you to specify a location in an image with greater granularity than pixel coordinates. For example, in the pixel coordinate system, a pixel is treated as a discrete unit, uniquely identified by an integer row and column pair, such as (3,4). In a spatial coordinate system, locations

in an image can be represented in terms of partial pixels, such as (3.3, 4.7). The following figure illustrates the spatial coordinate system used for images:



This spatial coordinate system corresponds to the pixel coordinate system in the following ways. First, both are defined in terms of row and column positions. Second, the spatial coordinates of the center point of any pixel are identical to the pixel coordinates for that pixel. However, the pixel coordinate system is discrete, while the spatial coordinate system is continuous. This means that, in pixel coordinates, the upper-left corner of an image is (0,0), while in spatial coordinates, this location is (-0.5,-0.5). The spatial coordinate system used by the Video and Image Processing Blockset differs from the one used by the Image Processing Toolbox. For more information on this spatial coordinate system, see “Spatial Coordinates” in the Image Processing Toolbox documentation.

Image Data Stored in Column-Major Format

MATLAB and the Video and Image Processing Blockset use a column-major numbering scheme to represent data elements internally. That means that

they internally store data elements from the first column first, then data elements from the second column second, and so on through the last column.

If you have imported an image or a video stream into the MATLAB workspace using a function from MATLAB or the Image Processing Toolbox, the Video and Image Processing Blockset blocks will display this image or video stream correctly. If you have written your own function or code to import images into MATLAB, you must take the column-major convention into account.

Sample Time

Because the Video and Image Processing blocks calculate values directly rather than solving differential equations, you must configure the Simulink Solver to behave like a scheduler. The following steps show you how to do this:

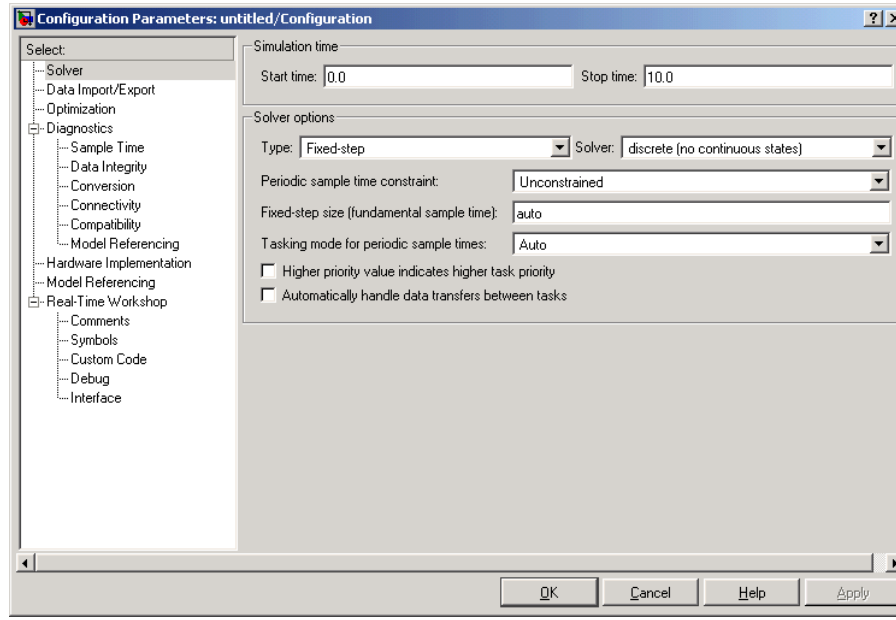
1. From the model's **Simulation** menu, select **Configuration Parameters**.

   The Configuration dialog box opens.

2. From the **Type** list, choose `Fixed-step`.

3. From the **Solver** list, choose `discrete (no continuous states)`.

   The following figure shows the correctly configured Configuration dialog box.

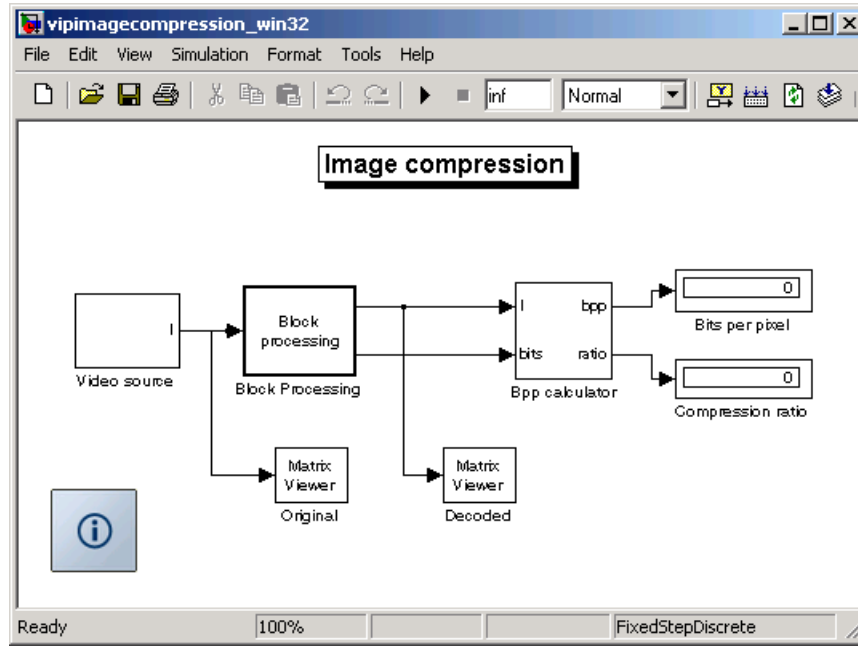


The Solver, while in scheduler mode, uses a block's sample time to determine when the code behind each block is executed. For example, if the sample time of a Video From Workspace block is `0.05`, the Solver executes the code behind this block, and every other block with this sample time, once every 0.05 second.

Video Duration and Simulation Time

The duration of the simulation is controlled by the **Stop time** parameter — not the input video. If you want the simulation to run for the duration of the input video, you must adjust the **Stop time** parameter. If your video is being cropped, increase the parameter value. If your video is complete and the display window is black, decrease the parameter value. To view the first *N* frames of your video, set the **Stop time** parameter to $(N\text{-}1)*T_s$, where T_s is the sample time of your source block.

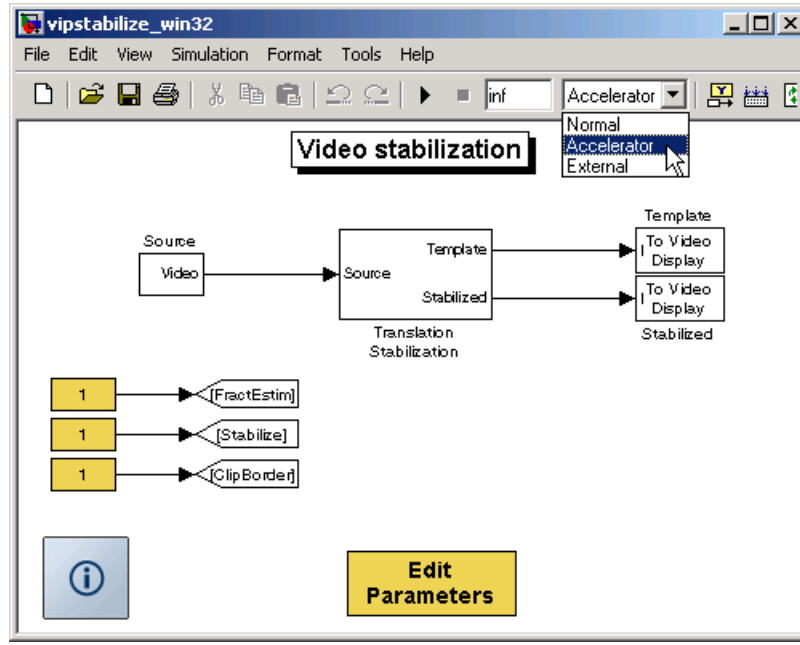
You can access the **Stop time** parameter in the model window, as shown in the figure below, or on the **Solver** pane of the Configuration dialog box. For more information, see "Solver Pane" in the Simulink documentation.



Normal and Accelerator Mode

The Simulink Accelerator removes much of the computational overhead required by Simulink models. It works by replacing blocks that are designed to handle any possible configuration in Simulink with compiled versions customized to your particular model's configuration. Through this method, the Accelerator can achieve substantial improvements in performance for larger Simulink models. The performance gains are tied to the size and complexity of your model. Therefore, large models that contain Video and Image Processing Blockset blocks run faster in `Accelerator` mode. You must have the Simulink Accelerator installed on your system to take advantage of this functionality.

To change between `Accelerator` and `Normal` mode, use the drop-down list at the top of the model window.



For more information, see “The Simulink Accelerator” in the Simulink documentation.

Windows Dynamic Library Dependencies

To run executables generated for Generic Real-Time (GRT), Embedded Real-Time (ERT), and S-Function targets, you need `vip_rt.dll` if both these conditions exist:

- The Real-Time Workshop target is a Windows platform.
- You are using the default Real-Time Workshop optimization parameters.

For more information about Real-Time Workshop optimization parameters, see “Generated Source Files and File Dependencies” in the Real-Time Workshop documentation.

If you want to run these executables on a Windows machine where the Video and Image Processing Blockset is not installed, copy `vip_rt.dll` from the

machine where the Video and Image Processing Blockset is installed to a directory on the system path of the other machine.

The library `vip_rt.dll` resides in `$matlabroot/bin/win32` on the machine where MATLAB and the Video and Image Processing Blockset are installed.

Block Data Type Support

The following table shows what data types are accepted on the main input data port of each Video and Image Processing Blockset block, unless otherwise noted. If the block is a source, the table shows what data types are accepted on the main output data port of each source block.

- If the Double, Single, Boolean, and/or Custom Data Types columns are populated by an x, the block supports those data types.
- If the Base Integer and/or Fixed-Point columns are populated with an s, the block supports signed integers and/or fixed-point data types.
- If the Base Integer and/or Fixed-Point columns are populated with a u, the block supports unsigned integers and/or fixed-point data types.

Note All blocks support code generation with Real-Time Workshop.

Block	Double	Single	Boolean	Base Integer	Fixed-Point
2-D Autocorrelation	x	x		s, u	s, u
2-D Convolution	x	x		s, u	s, u
2-D Correlation	x	x		s, u	s, u
2-D DCT	x	x		s, u	s, u
2-D FFT	x	x		s, u	s, u
2-D FIR Filter	x	x		s, u	s, u
2-D Histogram	x	x		s, u	s, u
2-D IDCT	x	x		s, u	s, u
2-D IFFT	x	x		s, u	s, u
2-D Mean	x	x		s, u	s, u
2-D Median	x	x		s, u	s, u
2-D Pad	x	x	x	s, u	s, u

Block	Double	Single	Boolean	Base Integer	Fixed-Point
2-D Standard Deviation	x	x			
2-D Variance	x	x		s, u	s, u
Autothreshold	x	x		s, u	s, u
Blob Analysis	x (Output)	x (Output)	x (Input)	s (Output)	s, u (Output)
Block Matching	x	x		s, u	s, u
Block Processing	The blocks inside the subsystem dictate the data types supported by this block.				
Bottom-hat	x	x	x	s, u	s, u
Chroma Resampling	x	x		u (8–bit unsigned integers only)	
Closing	x	x	x	s, u	s, u
Color Space Conversion	x	x		u (8–bit unsigned integers only)	
Compositing	x	x	x	s, u	s, u
Contrast Adjustment	x	x		s, u	s, u
Deinterlacing	x	x		s, u	s, u
Demosaic	x	x		s, u	s, u
Dilation	x	x	x	s, u	s, u
Draw Markers	x	x	x	s, u	s, u
Draw Shapes	x	x	x	s, u	s, u

Block	Double	Single	Boolean	Base Integer	Fixed-Point
Edge Detection	x	x		s, u	s, u
Erosion	x	x	x	s, u	s, u
Find Local Maxima	x	x		s, u	s, u
Frame Rate Display	x	x	x	s, u	s, u
From Multimedia File	This is a Signal Processing Blockset block.				
Gamma Correction	x	x		s, u	s, u
Gaussian Pyramid	x	x		s, u	s, u
Histogram Equalization	x	x		s, u	s, u
Hough Lines	x	x		s	s (Word length less than or equal to 32)
Hough Transform	x (Output)	x (Output)	x	u (Output)	u (Output)
Image Complement	x	x	x	s, u	
Image Data Type Conversion	x	x	x	s, u (Word length less than or equal to 16)	s, u (Word length less than or equal to 16)
Image From File	x	x	x	s, u	s, u
Image From Workspace	x	x	x	s, u	s, u
Insert Text	x	x	x	s, u	s, u
Label			x	u (Output)	

Block	Double	Single	Boolean	Base Integer	Fixed-Point
Maximum	This is a Signal Processing Blockset block.				
Median Filter	x	x	x	s, u	s, u
Minimum	This is a Signal Processing Blockset block.				
Opening	x	x	x	s, u	s, u
Optical Flow	x	x			
Projective Transformation	x	x	x	s, u	s, u
PSNR	x	x		s, u	s, u
Read Binary File				s, u	
Resize	x	x		s, u	s, u
Rotate	x	x		s, u	s, u
SAD	x	x	x	s, u	s, u
Shear	x	x		s, u	s, u
To Multimedia File	This is a Signal Processing Blockset block.				
To Video Display	x	x	x	s, u	
Top-hat	x	x	x	s, u	s, u
Trace Boundaries			x		
Translate	x	x		s, u	s, u
Variable Selector	This is a Signal Processing Blockset block.				
Video From Workspace	x	x	x	s, u	s, u
Video To Workspace	x	x	x	s, u	s, u

Block	Double	Single	Boolean	Base Integer	Fixed-Point
Video Viewer	x	x	x	s, u	
Write AVI File	x	x	x	s, u	
Write Binary File				s, u	

Image Credits

This table lists the copyright owners of the images used in the Video and Image Processing Blockset documentation.

Image	Source
`cameraman`	Copyright Massachusetts Institute of Technology. Used with permission.
`circuit`	Micrograph of 16-bit A/D converter circuit, courtesy of Steve Decker and Shujaat Nadeem, MIT, 1993.
`moon`	Copyright Michael Myers. Used with permission.

2

Importing and Exporting Video

In this chapter, you learn how to bring video data into a model, display it on your monitor, and export it to an AVI or multimedia file.

Working with AVI Files (p. 2-2)	Use the From Multimedia File block to import video data into your Simulink model and the Write AVI File block to export video data to an AVI file
Working with Multimedia Files (p. 2-14)	Use the From Multimedia File block to import video data into your Simulink model and the To Video Display block to view it. Use the To Multimedia File block to export video data to a multimedia file. These procedures assume you are working on a Windows platform.

Working with AVI Files

The Video and Image Processing Blockset enables you to work with video data within the Simulink environment. Before you can analyze or operate on your data, you must import it into your Simulink model. Blocks from the Sources library, such as the From Multimedia File block, can help you with this type of task.

This section includes the following topics:

- "Importing and Viewing AVI Files" on page 2-2 -- Use the From Multimedia File block to import video data into your Simulink model
- "Exporting to AVI Files" on page 2-5 -- Use the Write AVI File block to export video data to an AVI file
- "Annotating AVI Files" on page 2-9 — Use the Insert Text block to add descriptive text to a video

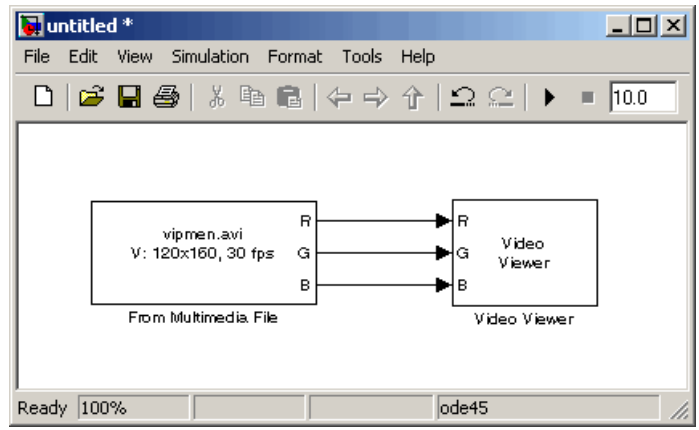
Importing and Viewing AVI Files

In this section, you use the From Multimedia File block to import an AVI file into your model and the Video Viewer block to view it:

1 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
From Multimedia File	Video and Image Processing Blockset / Sources	1
Video Viewer	Video and Image Processing Blockset / Sinks	1

2 Connect the blocks so your model looks similar to the figure below.



3 Locate an AVI file that you want to import into Simulink. If you do not have access to an AVI file, the Video and Image Processing Blockset has sample AVI files you can use to complete this procedure.

4 Use the From Multimedia File block to import the AVI file into the model. Double-click the From Multimedia File block.

- If you do not have your own AVI file, enter `barcodes.avi` for the **File name** parameter.
- If the AVI file is on your MATLAB path, enter the AVI filename for the **File name** parameter.
- If the file is not on your MATLAB path, use the **Browse** button to locate the AVI filename.

By default, the **Number of times to play file** parameter is set to `inf`. The model continues to play the file until the simulation stops.

5 Use the Video Viewer block to view the AVI file. Use the default parameters.

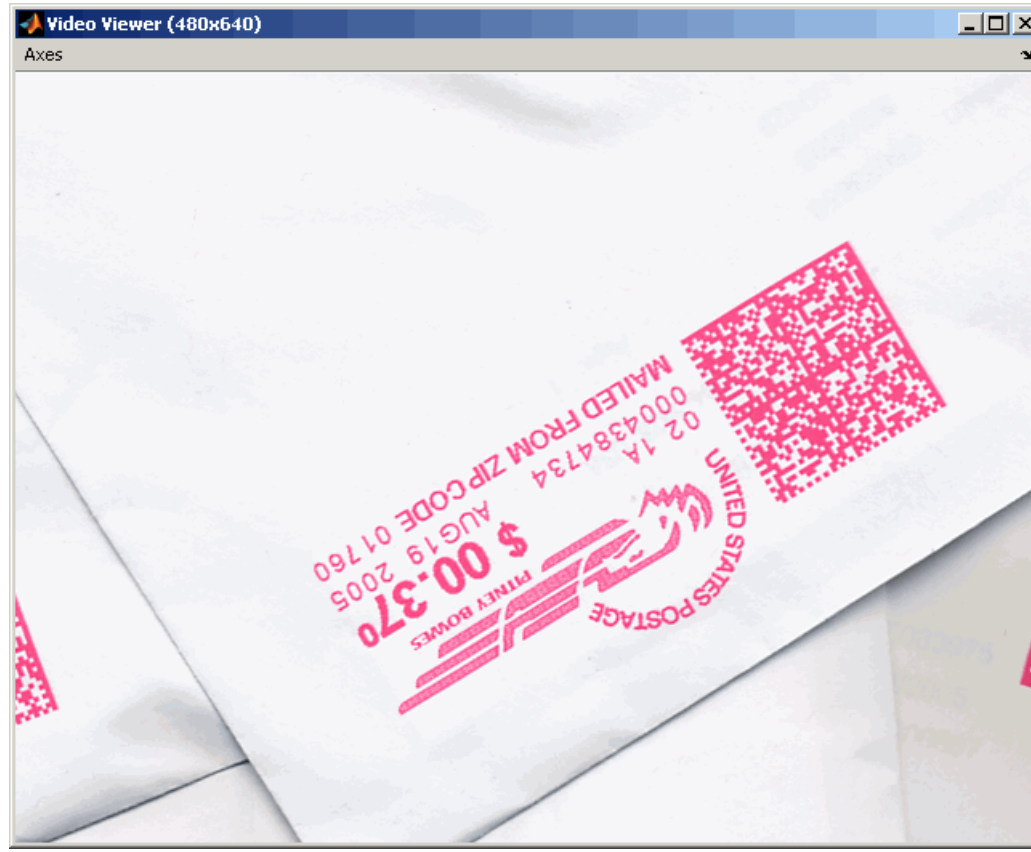
6 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `20`
- **Solver** pane, **Type** = `Fixed-step`

- **Solver** pane, **Solver** = `discrete (no continuous states)`

7 Run your model.

View your video in the Video Viewer window that automatically appears when you start your simulation. To view the video at its true size, right-click the window and select **Set Display To True Size**. To save the size and the position of the Video Viewer window, right-click and select **Save Position**.



Note The video that is displayed in the Video Viewer window runs as fast as Simulink processes the video frames. If you are on a Windows platform and you want to run the video at the frame rate that corresponds to the input sample time, use the To Video Display block.

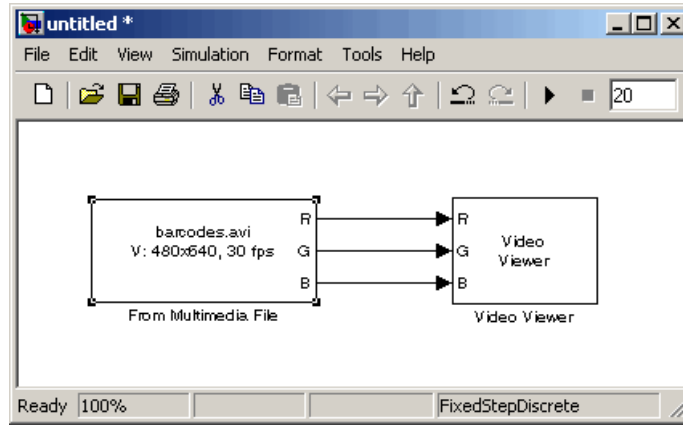
You have now imported and displayed video data in your Simulink model. In "Exporting to AVI Files" on page 2-5, you manipulate your video stream and export it to an AVI file. For more information on the blocks used in this example, see the From Multimedia File and Video Viewer block reference pages. To listen to audio associated with an AVI file, use the To Wave Device block in the Signal Processing Blockset.

Note The Video Viewer block is supported on all platforms, but it does not support code generation. If you are on a Windows platform, you can use the To Video Display block to display video data. This block supports code generation. For more information, see the To Video Display block reference page.

Exporting to AVI Files

The Video and Image Processing Blockset enables you to export video data from your Simulink model. In this section, you use the Write AVI File block to export an AVI file from your model:

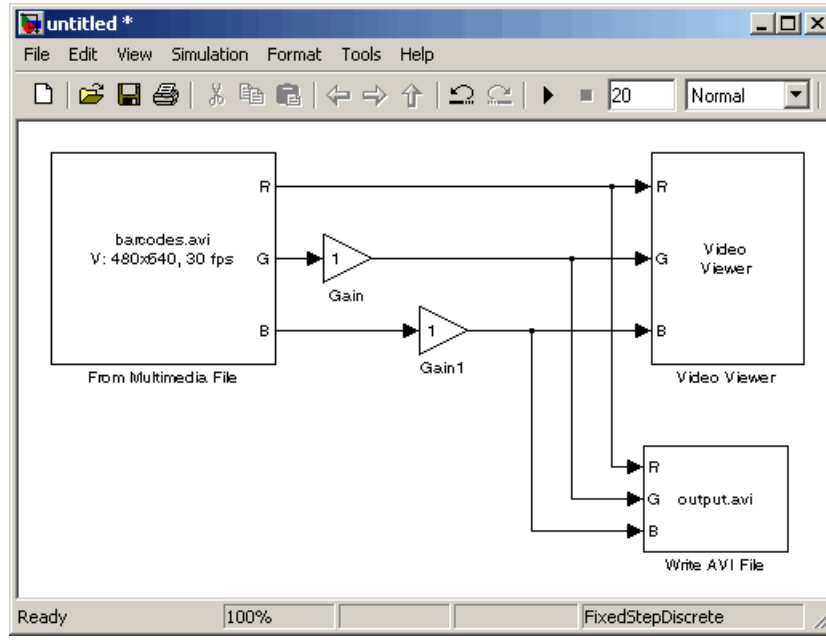
1 Open the model you created in "Importing and Viewing AVI Files" on page 2-2.



2 Click-and-drag the following blocks into your model.

Block	Library	Quantity
Write AVI File	Video and Image Processing Blockset / Sinks	1
Gain	Simulink / Math Operations	2

3 Connect the blocks as shown in the following figure. You might need to resize some blocks to do so.



You are now ready to set your block parameters by double-clicking the blocks, modifying the block parameter values, and clicking **OK**.

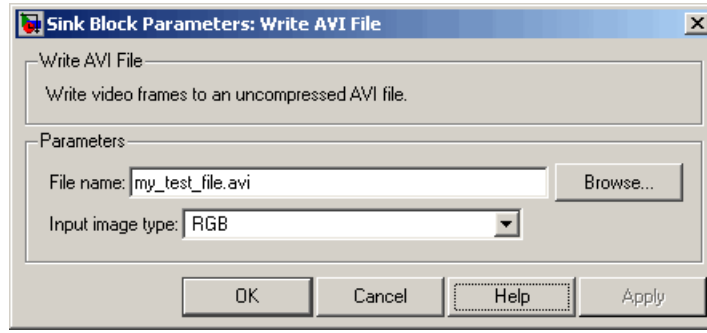
4 Use the Gain block to change the green values of the video stream. Set the block parameters as follows:

- **Main** pane, **Gain** = `0.3`
- **Signal data types** pane, **Output data type mode** = `Same as input`

5 Use the Gain1 block to change the blue values of the video stream. Set the block parameters as follows:

- **Main** pane, **Gain** = `1.5`
- **Signal data types** pane, **Output data type mode** = `Same as input`

6 Use the Write AVI File block to export the video to an AVI file. Set the **File name** parameter to `my_test_file.avi`.

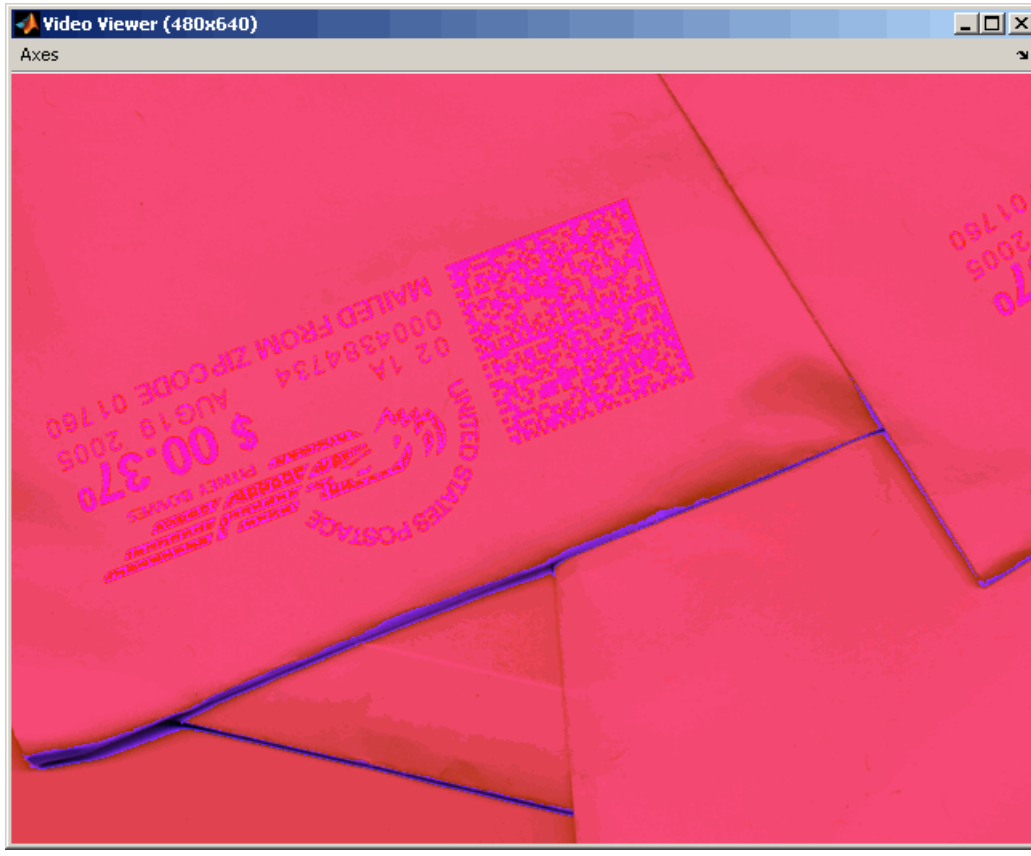


7 If you have not already done so, set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `20`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

8 Run your model.

You can view your video in the Video Viewer window. The Write AVI File block exports the video data from the Simulink model to an AVI file that it creates in your current directory.



You have now manipulated your video stream and exported it from a Simulink model to an AVI file. For more information, see the Write AVI File block reference page.

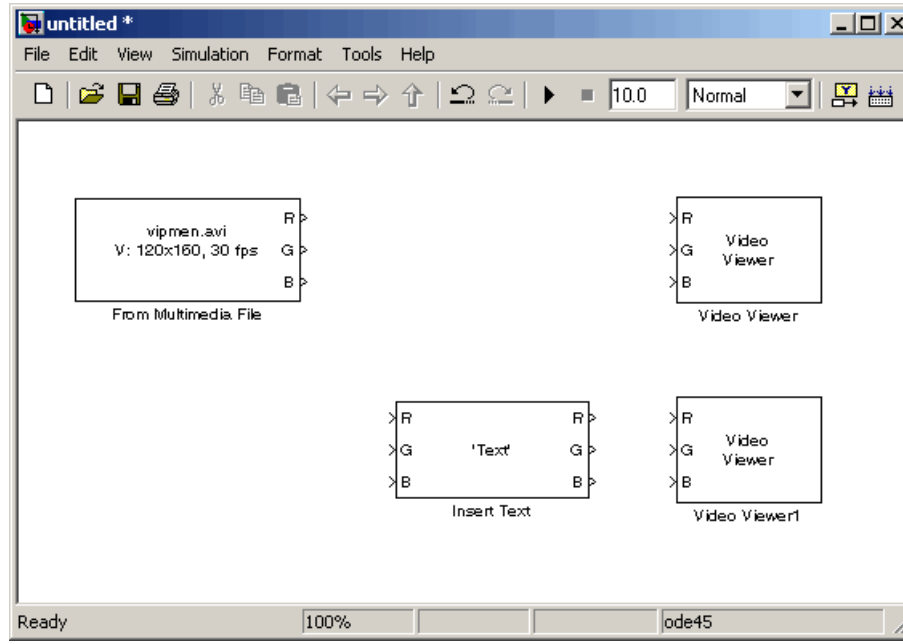
Annotating AVI Files

You can use the Insert Text block to overlay text on video stream. In this example, you add a running count of the number of video frames to a video.

1 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
From Multimedia File	Video and Image Processing Blockset / Sources	1
Insert Text	Video and Image Processing Blockset / Text & Graphics	1
Video Viewer	Video and Image Processing Blockset / Sinks	2

2 Position the blocks as shown in the following figure.



3 Use the From Multimedia File block to import the video into the Simulink model. Select the **Output intensity video** check box to work with intensity video.

4 Open the Surveillance Recording demo by typing

```
vipsurveillance
```

at the MATLAB command prompt.

5 Click-and-drag the Frame Counter block from the demo model into your model. This block counts the number of frames in an input video.

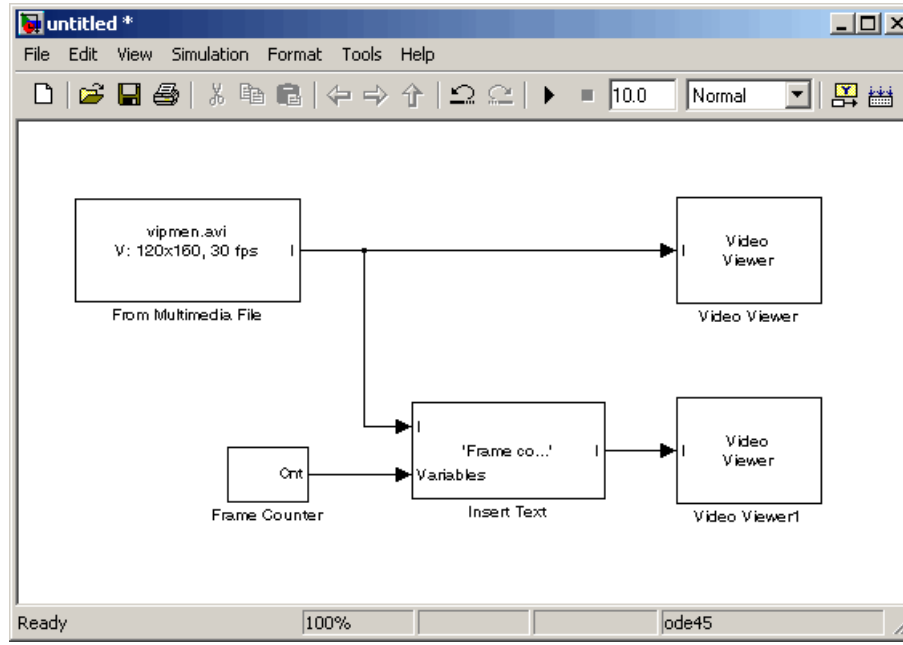
6 Use the Insert Text block to annotate the video stream with a running frame count. Set the block parameters as follows:

- **Main** pane, **Input type** = `Intensity`
- **Main** pane, **Text** = `['Frame count' sprintf('\n') 'Source frame: %d']`
- **Main** pane, **Location** = `[85 2]`
- **Main** pane, **Intensity** = `1`
- **Font** pane, **Font face** = `LucindaTypewriterRegular`

By setting the **Text** parameter to `['Frame count' sprintf('\n') 'Source frame:  %d']`, you are asking the block to print `Frame count` on one line and the`Source frame:` on a new line. Because you specified `%d`, an ANSI C printf-style format specification, the Variables port appears on the block. The block takes the port input (it is expecting a decimal) and substitutes it for the `%d` in the string. You used the **Location** parameter to specify where to print the text. In this case, the location is 85 rows down and 2 rows over from the top left corner of the image.

7 Use the Video Viewer blocks to view the original and annotated videos. Set the **Input image type** parameters to `Intensity`.

8 Connect the blocks as shown in the following figure.



9 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

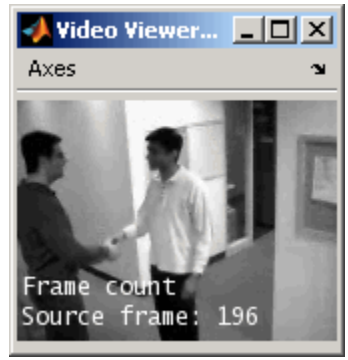
- **Solver** pane, **Stop time** = `inf`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

10 Run the model.

The original video appears in the Video Viewer window.



The annotated video appears in the Video Viewer1 window.



You have now added descriptive text to a video stream. For more information, see the Insert Text block reference page. For related information, see the Draw Shapes and Draw Markers block reference pages.

Working with Multimedia Files

If you are working on a Windows platform, the Video and Image Processing Blockset contains blocks that you can use to import and view multimedia files. These blocks include the From Multimedia File block, the To Multimedia File block, and the To Video Display block. These blocks perform best on platforms with DirectX Version 9.0 or later and Windows Media Version 9.0 or later. They also support code generation. If you generate code from a model that contains a To Video Display block, you can view the video stream when you run the executable.

This section includes the following topics:

- "Importing and Viewing Multimedia Files" on page 2-14 — Use the From Multimedia File block to import video data into your Simulink model and the To Video Display block to view it
- "Exporting to Multimedia Files" on page 2-17 — Use the To Multimedia File block to export video data to a multimedia file
- "Working with Audio" on page 2-20 — Use the To Multimedia File block to write separate audio and video data to a single multimedia file

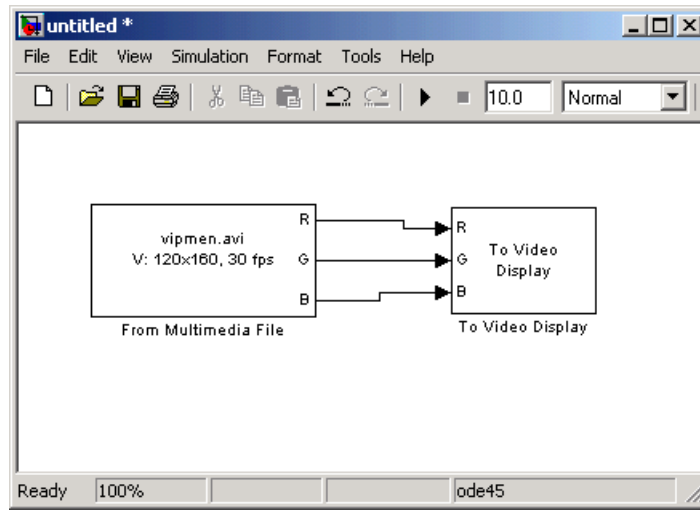
Importing and Viewing Multimedia Files

In this example, you use the From Multimedia File block to import a video stream into a Simulink model and the To Video Display block to view it. This procedure assumes you are working on a Windows platform:

1 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
From Multimedia File	Video and Image Processing Blockset / Sources	1
To Video Display	Video and Image Processing Blockset / Sinks	1

2 Connect the blocks so your model looks similar to the figure below.



3 Locate a multimedia file that you want to import into Simulink. If you do not have access to a multimedia file, the Video and Image Processing Blockset has sample multimedia files you can use to complete this procedure.

4 Use the From Multimedia File block to import the multimedia file into the model. Double-click the From Multimedia File block:

- If you do not have your own multimedia file, enter `vipmen.avi` for the **Input file name** parameter.
- If the multimedia file is on your MATLAB path, enter the filename for the **Input file name** parameter.
- If the file is not on your MATLAB path, use the **Browse** button to locate the multimedia file.

By default, the **Number of times to play file** parameter is set to `inf`. The model continues to play the file until the simulation stops.

Note The From Multimedia File block supports any multimedia file format supported by Microsoft Windows Media Player. Additionally, this block supports specialized file formats that are associated with any codec supported by Microsoft Windows Media Player.

5 Use the To Video Display block to view the multimedia file. Use the default parameters.

6 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. On the **Solver** pane, set the parameters as follows:

- **Stop time** = `20`
- **Type** = `Fixed-step`
- **Solver** = `discrete (no continuous states)`

7 Run your model.

View your video in the To Video Display window that automatically appears when you start your simulation. This window closes as soon as the simulation stops.



Note The video that is displayed in the To Video Display window runs at the frame rate that corresponds to the input sample time. To run the video as fast as Simulink processes the video frames, use the Video Viewer block.

You have now imported and displayed a multimedia file in your Simulink model. In “Exporting to Multimedia Files” on page 2-17, you manipulate your video stream and export it to a multimedia file. For more information on the blocks used in this example, see the From Multimedia File and To Video Display block reference pages. To listen to audio associated with an AVI file, use the To Wave Device block in the Signal Processing Blockset.

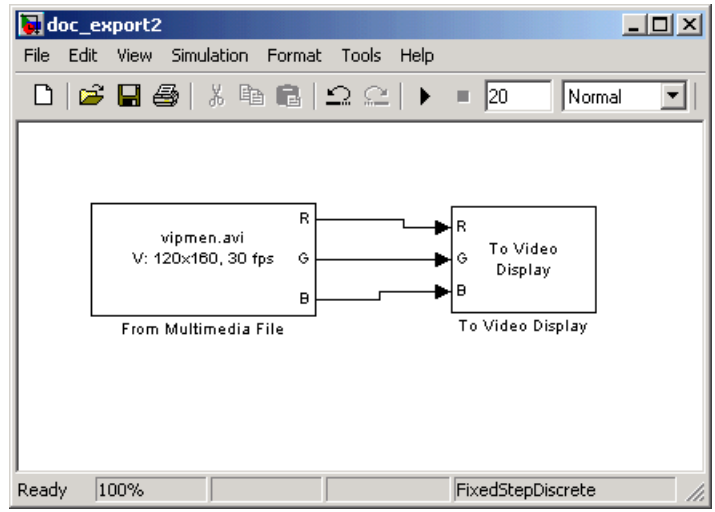
Exporting to Multimedia Files

The Video and Image Processing Blockset enables you to export video data from your Simulink model. In this section, you use the To Multimedia File block to export an multimedia file from your model. This procedure assumes you are working on a Windows platform.

1 If the model you created in “Importing and Viewing Multimedia Files” on page 2-14 is not open on your desktop, you can open an equivalent model by typing

```
doc_export2
```

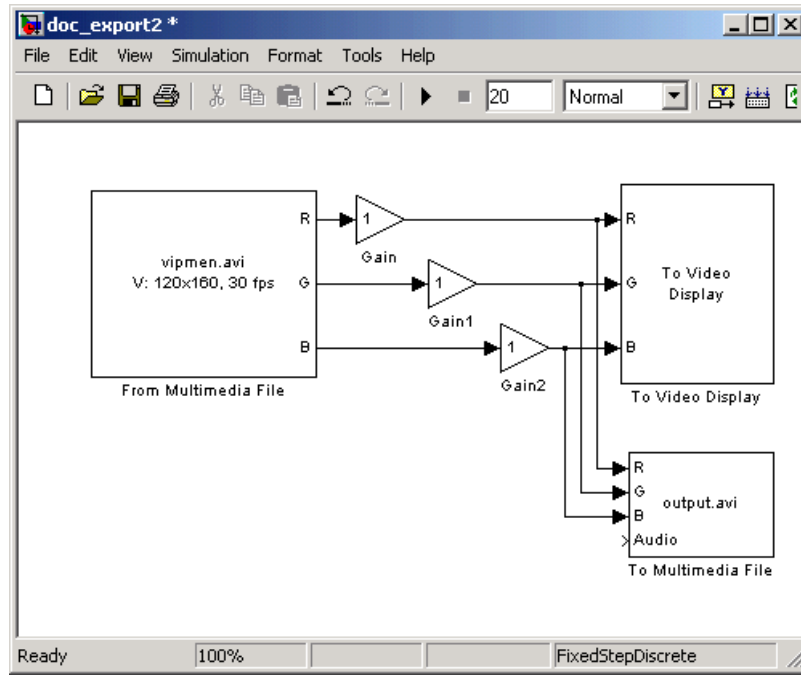
at the MATLAB command prompt.



2 Click-and-drag the following blocks into your model.

Block	Library	Quantity
To Multimedia File	Video and Image Processing Blockset / Sinks	1
Gain	Simulink / Math Operations	3

3 Connect the blocks as shown in the following figure. You might need to resize some blocks to do so.



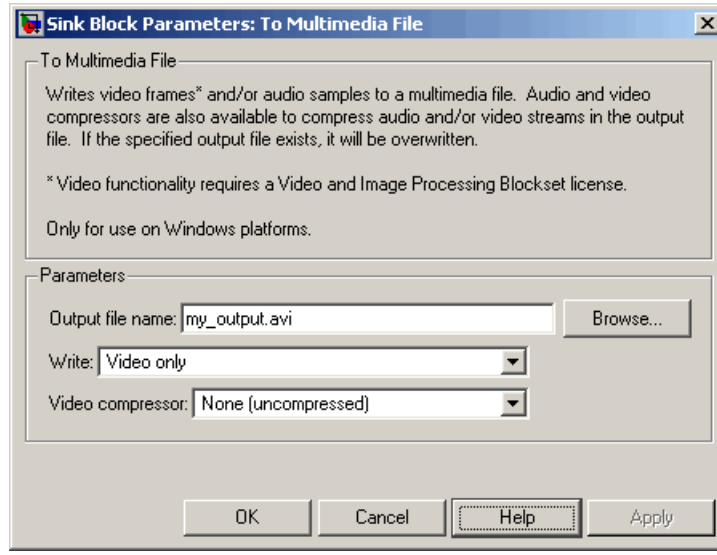
You are now ready to set your block parameters by double-clicking the blocks, modifying the block parameter values, and clicking **OK**.

4 Use the Gain blocks to increase the red, green, and blue values of the video stream. This increases the contrast of the video. Set the block parameters as follows:

- **Main** pane, **Gain** = `1.2`
- **Signal data types** pane, **Output data type mode** = `Same as input`

5 Use the To Multimedia File block to export the video to a multimedia file. Set the block parameters as follows:

- **Output file name** = `my_output.avi`
- **Write** = `Video only`



6 If you have not already done so, set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. On the **Solver** pane, set the parameters as follows:

- **Stop time** = `20`
- **Type** = `Fixed-step`
- **Solver** = `discrete (no continuous states)`

7 Run your model.

You can view your video in the To Video Display window. Note that increasing the red, green, and blue color values increased the contrast of

the video. The To Multimedia File block exports the video data from the Simulink model to a multimedia file that it creates in your current directory.



You have now manipulated your video stream and exported it from a Simulink model to a multimedia file. For more information, see the To Multimedia File block reference page.

Working with Audio

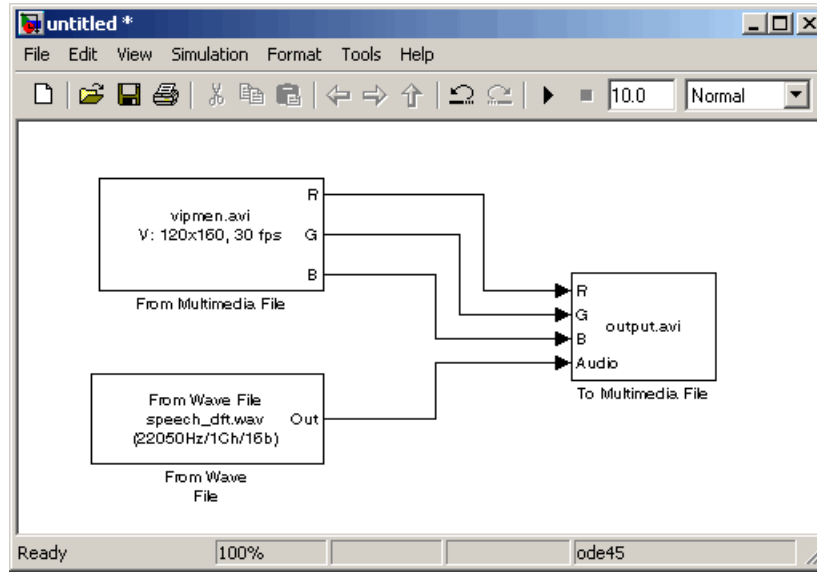
In this example, you use the From Multimedia File block to import a video stream into a Simulink model. You also use the Signal Processing Blockset From Wave File block to import an audio stream into the model. Then you write this audio and video to a single file using the To Multimedia File block.

This procedure assumes you are working on a Windows platform:

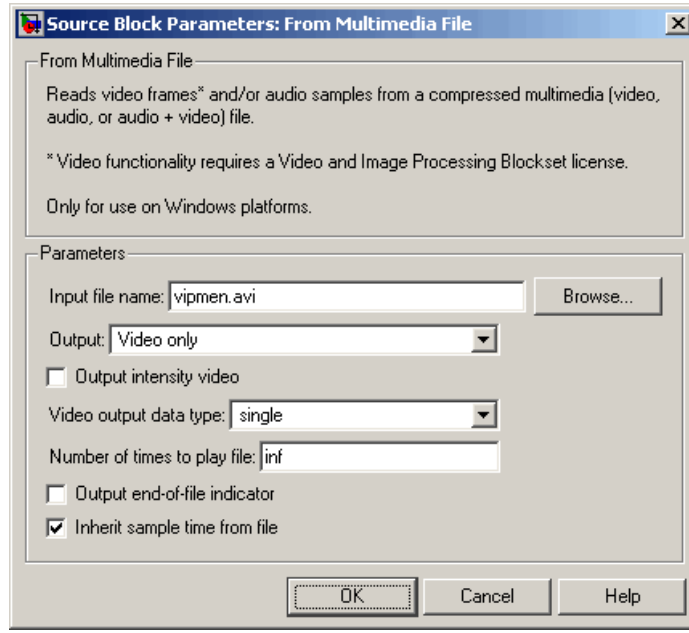
1 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
From Multimedia File	Video and Image Processing Blockset / Sources	1
From Wave File	Signal Processing Blockset / Platform Specific I/O	1
To Multimedia File	Video and Image Processing Blockset / Sinks	1

2 Connect the blocks so your model looks similar to the following figure.

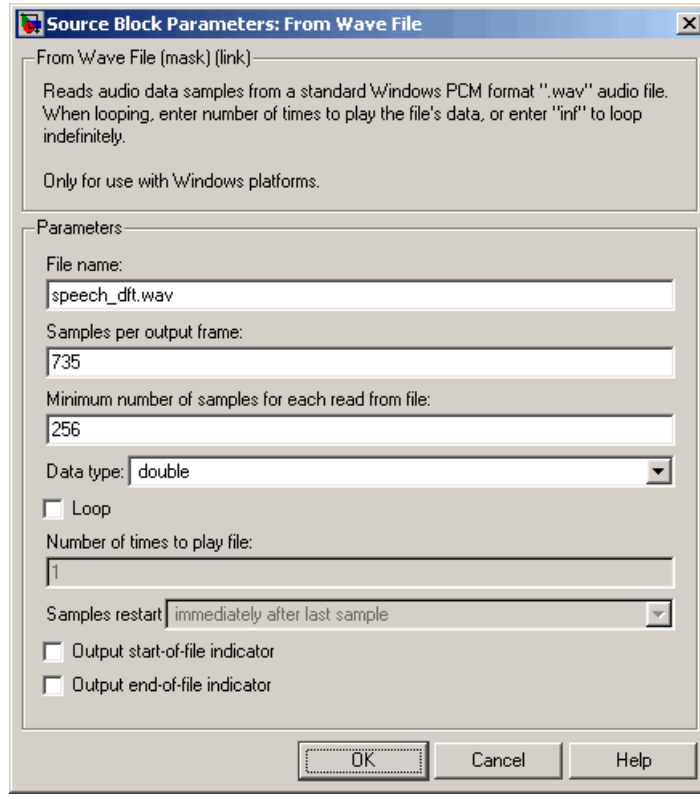


3 Use the From Multimedia File block to import a multimedia file into the model. Use the following default parameters.



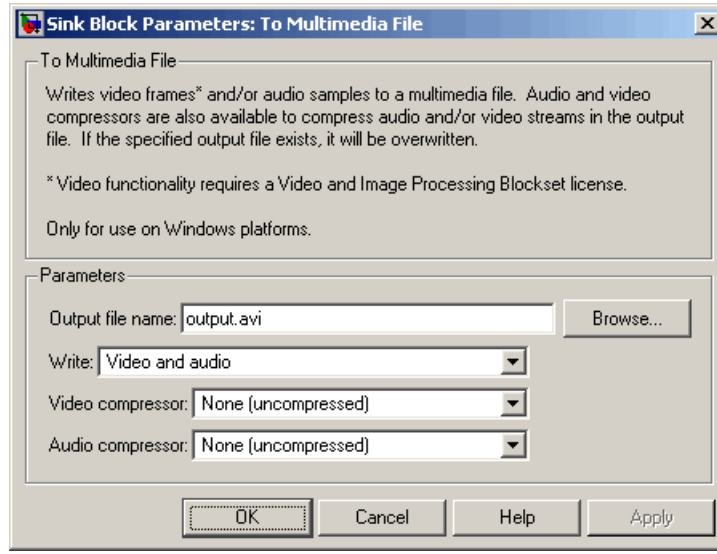
Note that the From Multimedia File block inherits its sample time from `vipmen.avi`. For video signals, the sample time is equivalent to the frame period. Because this file's frame rate is 30 frames per second (fps) and the frame period is defined as 1/frame rate, the frame period of this block is 0.0333 seconds per frame.

4 Use the From Wave File block to import an audio file into the model. To calculate the output frame size, divide the frequency of the audio signal (22050 samples per second) by the frame rate (30 frames per second) to get 735 samples per frame. Set the **Samples per output frame** parameter to 735.



The frame period of the audio signal must match the frame period of the video signals, which is 0.0333 seconds per frame. Since the frame period is also defined as the frame size divided by frequency, you can calculate the frame period of the audio signal by dividing the frame size of the audio signal (735 samples per frame) by the frequency (22050 samples per second) to get 0.0333 seconds per frame. Alternatively, you can verify that the frame period of the audio and video signals is the same using a Simulink Probe block.

5 Use the To Multimedia File to output the audio and video signals to a single multimedia file. Use the default parameters.



6 Set the configuration parameters. Open the Configuration dialog box by selecting **Simulation > Configuration Parameters**. On the **Solver** pane, set the parameters as follows:

- **Stop time** = `10`
- **Type** = `Fixed-step`
- **Solver** = `discrete (no continuous states)`

7 Run your model. The model creates a multimedia file called `output.avi` in your current directory.

8 Play the multimedia file using a media player. The original video file now has an audio component to it.

You have now combined audio and video information into a single file using the To Multimedia File block. For more information, see the To Multimedia File block reference page.

3

Working with MPlay

The MPlay GUI enables you to view videos that are represented as variables in the MATLAB workspace. You can also use it to view video files or video signals in Simulink models.

Viewing Videos from the MATLAB Workspace (p. 3-2)	Use MPlay to view videos in the MATLAB workspace.
Viewing Video Files (p. 3-6)	Use MPlay to view videos stored in AVI files.
Viewing Video Signals in Simulink (p. 3-8)	Use MPlay to view video signals in Simulink models.

Viewing Videos from the MATLAB Workspace

The MPlay GUI enables you to view videos that are represented as variables in the MATLAB workspace, such as video data exported to the workspace by the Video To Workspace block. The following procedure shows you how to use the MPlay GUI to view such a video:

1. Define a variable that represents a video sequence in the MATLAB workspace. For example, to read an entire video into memory, type

   ```
   d = aviread('vipmen.avi');
   ```

   at the MATLAB command prompt.

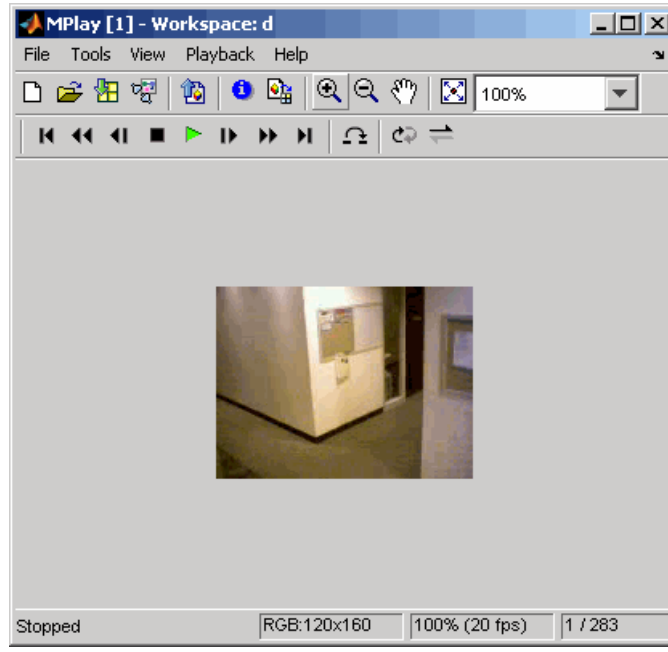
2. Open an MPlay GUI by typing

   ```
   mplay
   ```

   at the MATLAB command prompt.

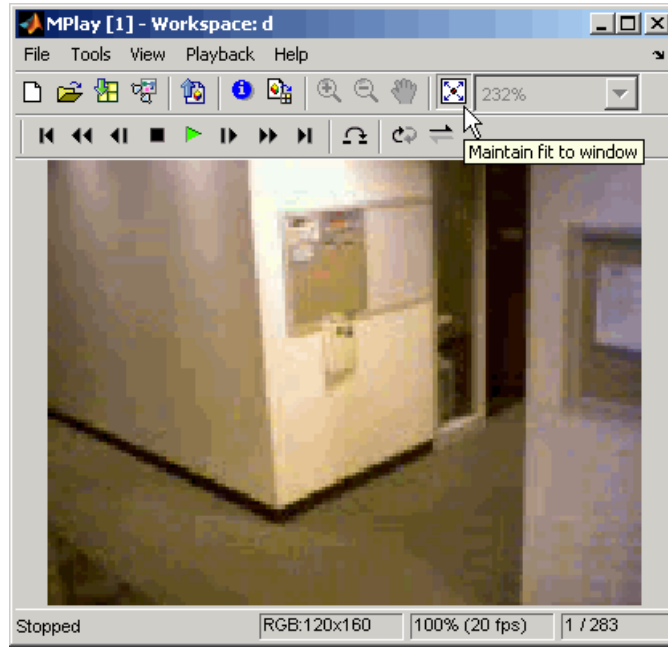
3. Connect the MPlay GUI to the variable in the MATLAB workspace by clicking  on the MPlay GUI. In the Import from Workspace dialog box, select `d` from the list of workspace variables. Then click **Import**.

   The first frame of the video appears in the MPlay window.

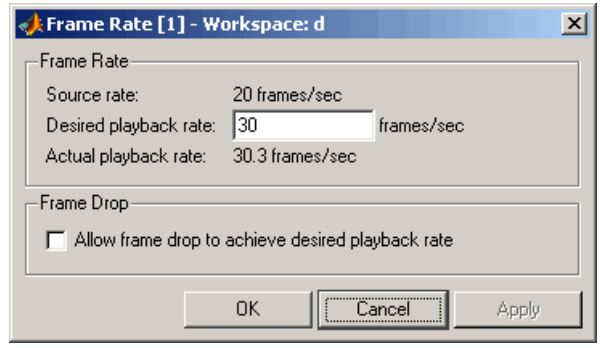


Note The MPlay GUI supports MATLAB variables that are in the movie structure array format. It also supports three-dimensional and four-dimensional arrays, which it interprets as intensity and RGB videos, respectively. You can use a function or any statement that can be evaluated for the **MATLAB variable or expression** parameter.

4 To resize the video to fill the GUI display area, click .



5 Experiment with using the MPlay GUI to play and interact with the video sequence. By default, the GUI assumes that the video data has a frame rate of 20 frames per second (fps). To change the frame rate to 30 fps, click **Playback** > **Frame Rate**. Enter 30 for the **Desired playback rate** parameter.



For more information about the MPlay GUI, see the `mplay` function reference page.

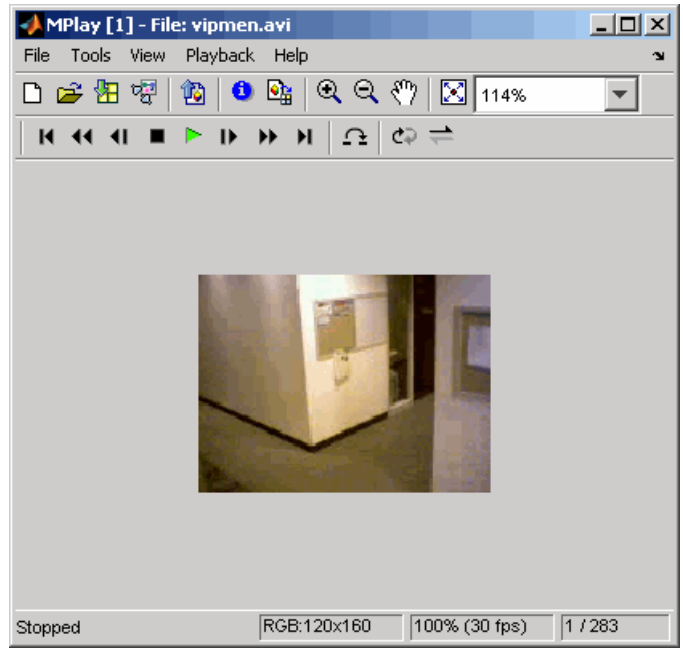
Viewing Video Files

The MPlay GUI enables you to view videos from files without having to load all the video data into memory at once. The following procedure shows you how to use the MPlay GUI to load and view a video one frame at a time:

1. On the MPlay GUI, click .

2. Use the Connect to File dialog box to navigate to the multimedia file you want to view in the MPlay window. For example, navigate to `$matlabroot\toolbox\vipblks\vipdemos\vipmen.avi`. Click **Open**.

   The first frame of the video appears in the MPlay window.



Note The MPlay GUI supports AVI files that are supported by the `aviread` function.

3 Experiment with the MPlay GUI by using it to play and interact with the video stream.

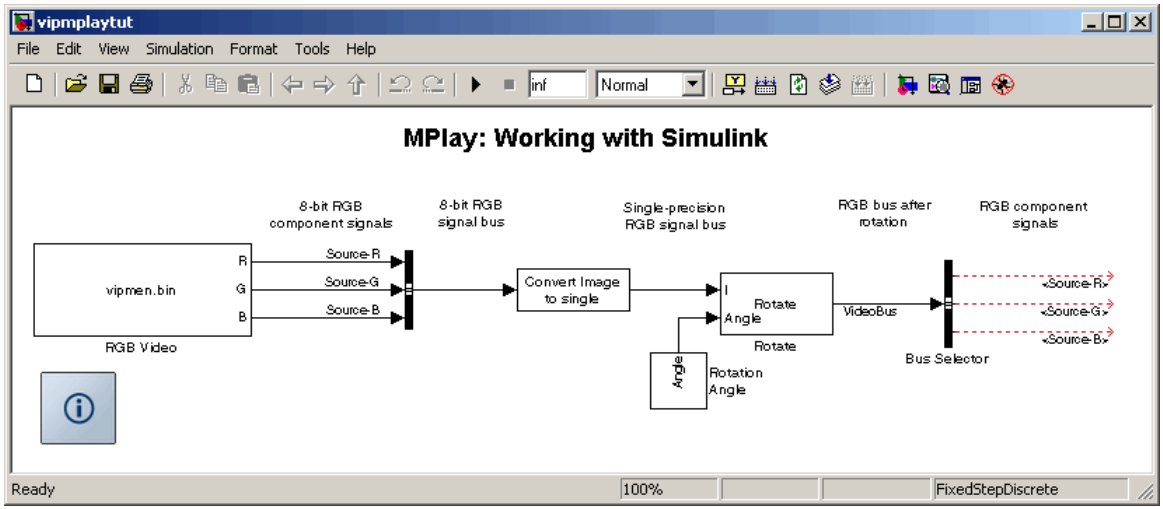
For more information about the MPlay GUI, see the `mplay` function reference page.

Viewing Video Signals in Simulink

The MPlay GUI enables you to view video signals in Simulink models without adding blocks to your model. The following procedure shows you how to use the MPlay GUI to view a Simulink signal:

1 Open a Simulink model. At the MATLAB command prompt, type

```
vipmplaytut
```



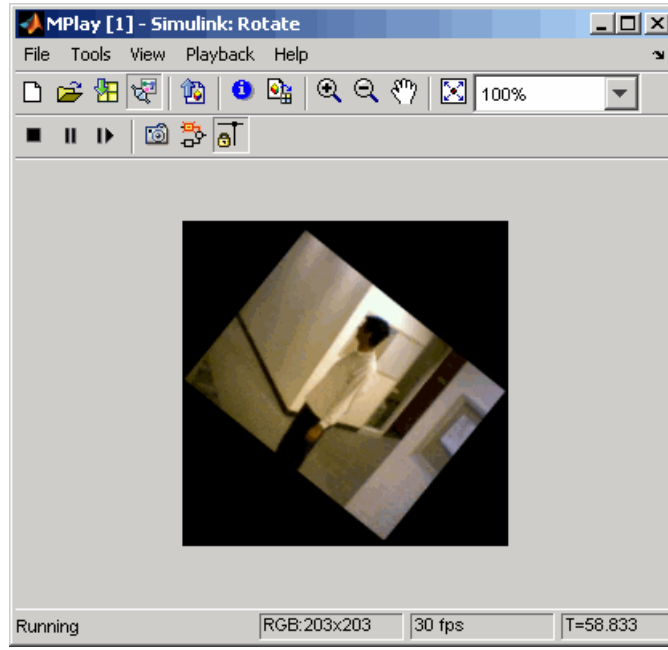
2 Open an MPlay GUI.

3 Run the model.

4 Select the signal line you want to view. For example, select the bus signal coming out of the Rotate block.

5 On the MPlay GUI, click .

The video appears in the MPlay window.



Also, some new buttons appear above the video window.

6 Change to floating-scope mode by clicking the  button.

7 Experiment with selecting different signals and viewing them in the MPlay window. You can also use multiple MPlay GUIs to display different Simulink signals.

For more information about the MPlay GUI, see the `mplay` function reference page.

Conversions

In this chapter, you learn how to convert an intensity image to a binary image and an RGB image to an intensity image. You also learn how to downsample the chroma components of an image.

Intensity to Binary Conversion (p. 4-2)	Learn how to convert an intensity image to a binary image.
Color Space Conversion (p. 4-14)	Learn how to convert color information between color spaces and to intensity values.
Chroma Resampling (p. 4-19)	Use the Chroma Resampling block to downsample the chroma components of an image.

Intensity to Binary Conversion

Binary images contain Boolean pixel values that are either 0 or 1. Pixels with the value 0 are displayed as black; pixels with the value 1 are displayed as white. Intensity images contain pixel values that range between the minimum and maximum values supported by their data type. Intensity images can contain only 0s and 1s, but they are not binary images unless their data type is Boolean.

This section includes the following topics:

- "Thresholding Intensity Images Using Relational Operators" on page 4-2 -- Use the Relational Operator block to convert an intensity image to a binary image
- "Thresholding Intensity Images Using the Autothreshold Block" on page 4-7 -- Use the Autothreshold block to convert an intensity image to a binary image

Thresholding Intensity Images Using Relational Operators

You can use the Relational Operator block to perform a thresholding operation that converts your intensity image to a binary image. This example shows you how to accomplish this task:

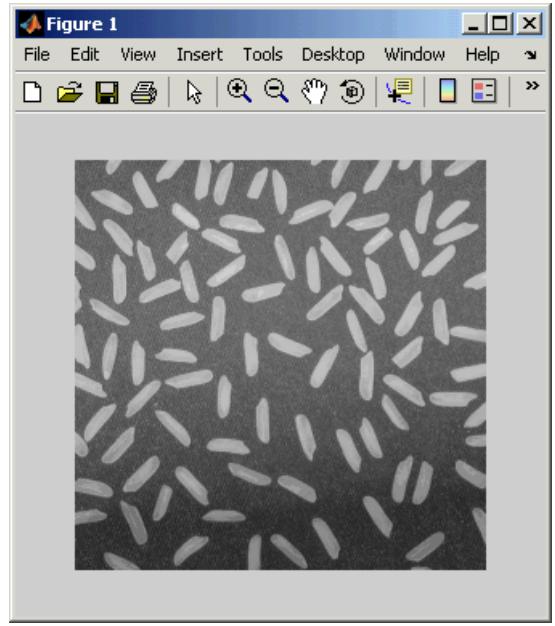
1. Define an intensity image in the MATLAB workspace. To read in an intensity image from a PNG file, at the MATLAB command prompt, type

   ```
   I= imread('rice.png');
   ```

   `I` is a 256-by-256 matrix of 8-bit unsigned integer values that range from 0 to 255.

2. To view the image this matrix represents, at the MATLAB command prompt, type

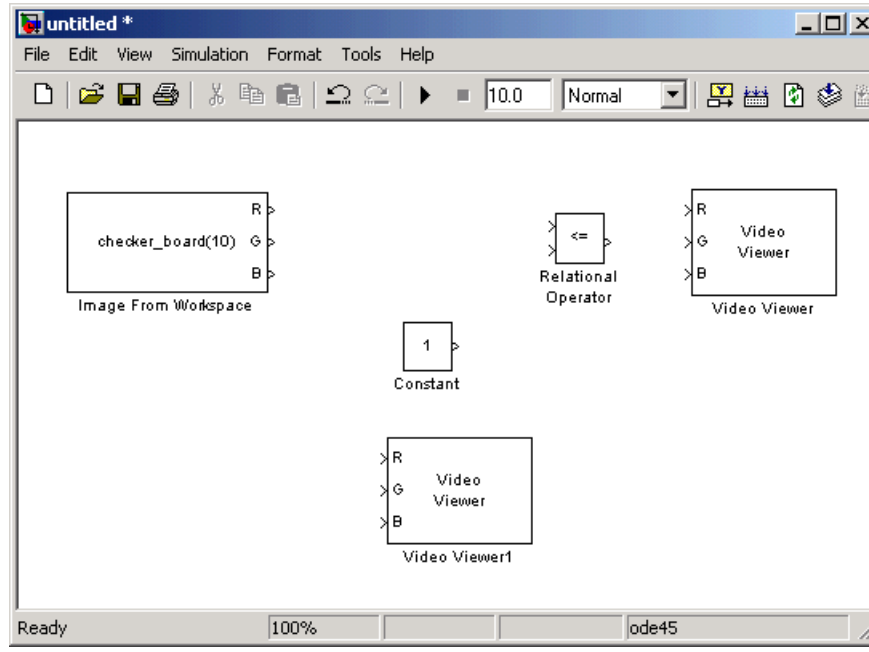
   ```
   imshow(I)
   ```



3 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
Video Viewer	Video and Image Processing Blockset / Sinks	2
Relational Operator	Simulink / Logic and Bit Operations	1
Constant	Simulink / Sources	1

4 Position the blocks as shown in the following figure.



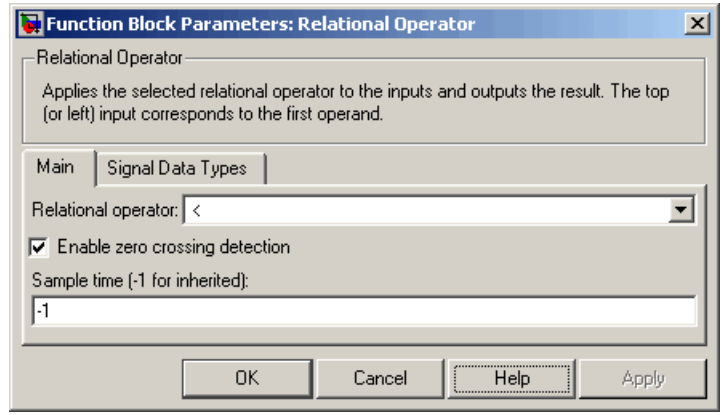
5 Use the Image from Workspace block to import your image from the MATLAB workspace. Set the block parameters as follows:

- **Value** = `I`
- **Output port labels** = `Image`

6 Use the Video Viewer1 block to view the original intensity image. Set the **Input image type** parameter to `Intensity`.

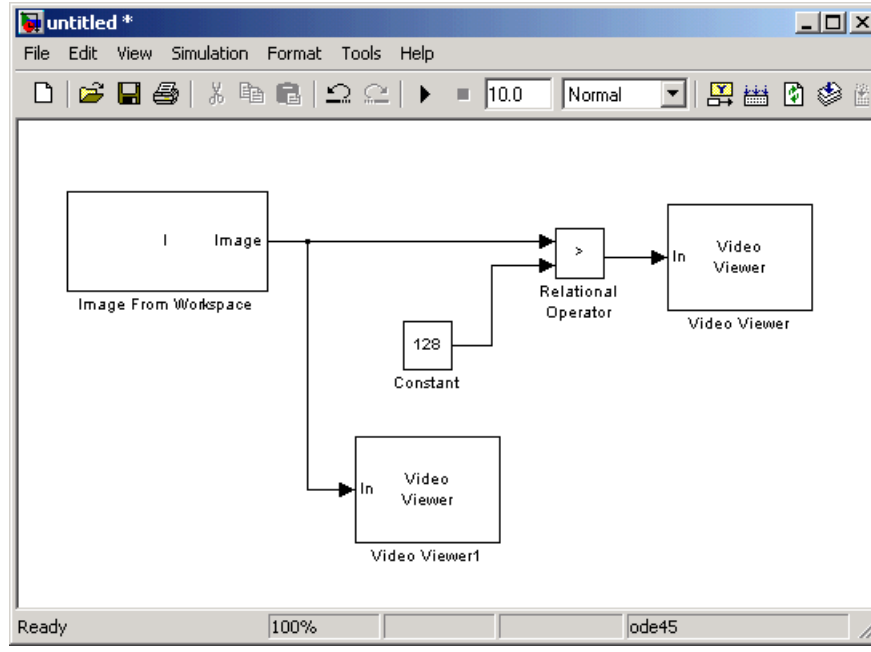
7 Use the Constant block to define a threshold value for the Relational Operator block. Since the pixel values range from 0 to 255, set the **Constant value** parameter to `128`. Note that this value is image dependent.

8 Use the Relational Operator block to perform a thresholding operation that converts your intensity image to a binary image. Set the **Relational Operator** parameter to >. If the input to the Relational Operator block is greater than 128, its output is a Boolean 1; otherwise, its output is a Boolean 0.



9 Use the Video Viewer block to view the binary image. Set the **Input image type** parameter to `Intensity`.

10 Connect the blocks as shown in the following figure.



11 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

12 Run your model.

The original intensity image appears in the Video Viewer1 window.



The binary image appears in the Video Viewer window.



Note A single threshold value was unable to effectively threshold this image due to its uneven lighting. For information on how to address this problem, see "Correcting for Nonuniform Illumination" on page 6-11.

You have used the Relational Operator block to convert an intensity image to a binary image. For more information about this block, see the Relational Operator block reference page in the Simulink documentation. For another example that uses this technique, see "Counting Objects in an Image" on page 6-3. For additional information, see "Converting Between Image Types" in the Image Processing Toolbox documentation.

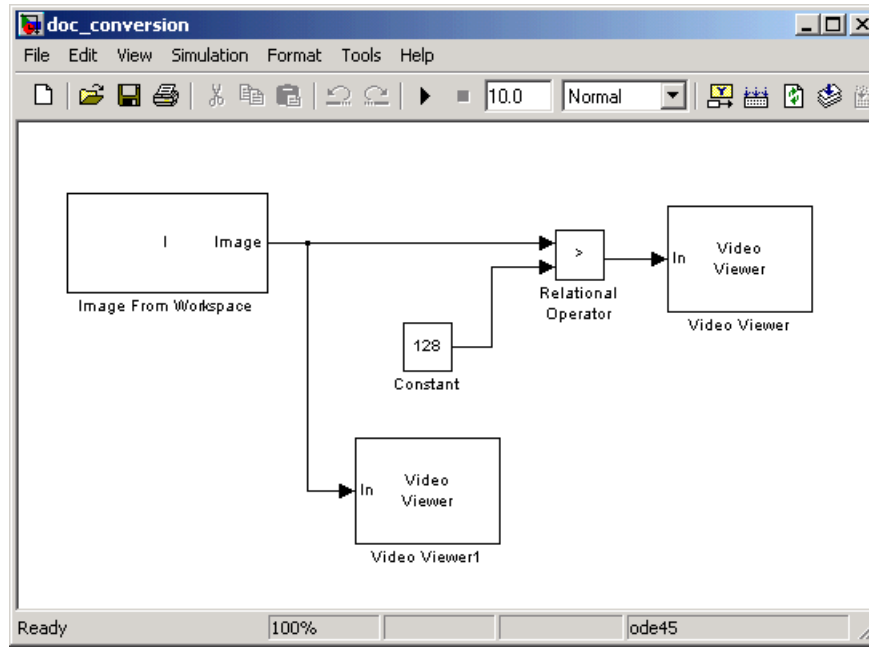
Thresholding Intensity Images Using the Autothreshold Block

In the previous topic, you used the Relational Operator block to convert an intensity image into a binary image. In this topic, you use the Autothreshold block to accomplish the same task. Use the Autothreshold block when lighting conditions vary and the threshold needs to change for each video frame.

1 If the model you created in "Thresholding Intensity Images Using Relational Operators" on page 4-2 is not open on your desktop, you can open an equivalent model by typing

```
doc_conversion
```

at the MATLAB command prompt.



2 If you have not already done so, define an intensity image in the MATLAB workspace. At the MATLAB command prompt, type

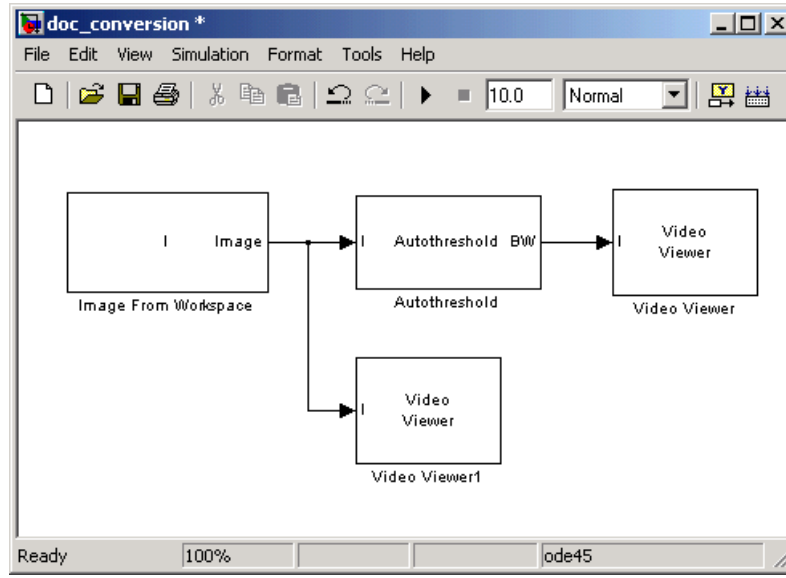
```
I= imread('rice.png');
```

`I` is a 256-by-256 matrix of 8-bit unsigned integer values that range from 0 to 255.

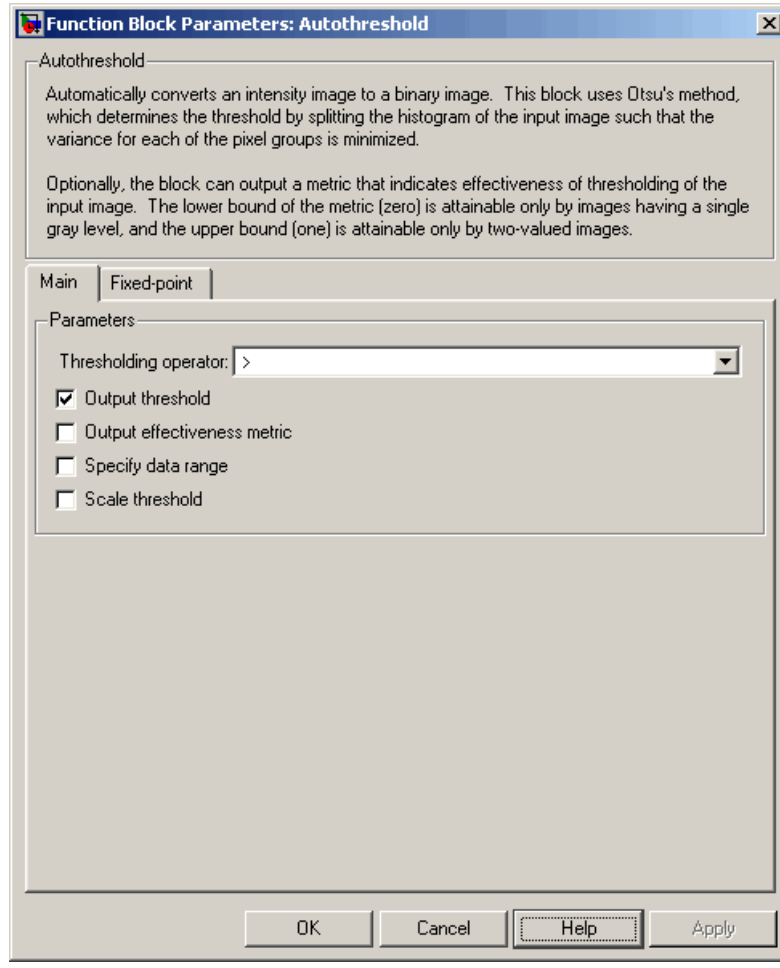
3 Delete the Constant and the Relational Operator blocks in this model.

4 From the Video and Image Processing Blockset library, and then from the Conversions library, click-and-drag an Autothreshold block into your model.

5 Connect the blocks as shown in the figure below.



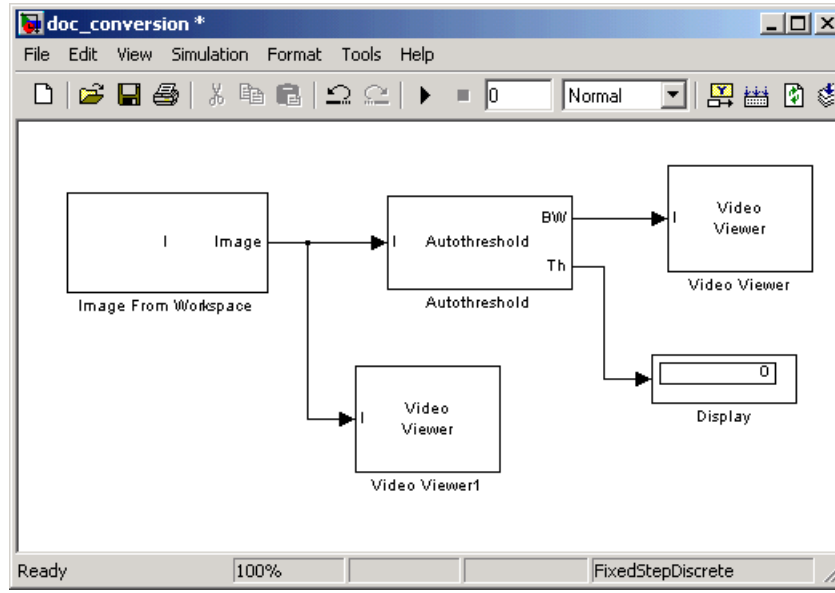
6 Use the Autothreshold block to perform a thresholding operation that converts your intensity image to a binary image. Select the **Output threshold** check box.



The block outputs the calculated threshold value at the Th port.

7 From the Signal Processing Blockset library, and then from the Signal Processing Sinks library, click-and-drag a Display block into the model. Connect it to the Th port.

Your model should look similar to the following figure:



8 Double-click the Image From Workspace block. On the **Data Types** pane, set the **Output data type** parameter to `double`.

9 If you have not already done so, set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

10 Run the model.

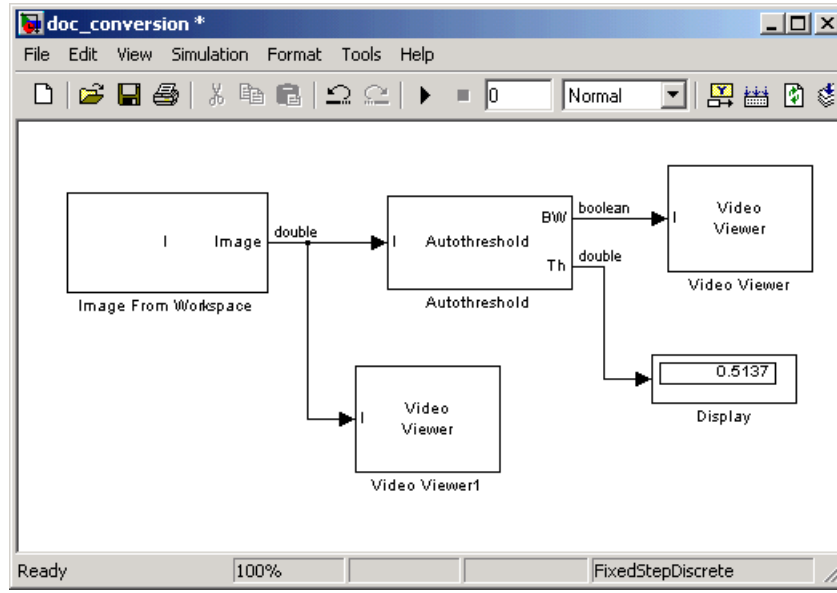
The original intensity image appears in the Video Viewer1 window.



The binary image appears in the Video Viewer window.



In the model window, the Display block shows the threshold value, calculated by the Autothreshold block, that separated the rice grains from the background.



You have used the Autothreshold block to convert an intensity image to a binary image. For more information about this block, see the Autothreshold block reference page. To open a demo model that uses this block, type `vipstaples` at the MATLAB command prompt.

Color Space Conversion

The Color Space Conversion block enables you to convert color information from the R'G'B' color space to the Y'CbCr color space and from the Y'CbCr color space to the R'G'B' color space as specified by Recommendation ITU-R BT.601-5. This block can also be used to convert from the R'G'B' color space to intensity. The prime notation indicates that the signals are gamma corrected.

This section includes the following topic:

- "Converting Color Information from R'G'B' to Intensity" on page 4-14 -- Use the Color Space Conversion block to convert an R'G'B' image to intensity.

Converting Color Information from R'G'B' to Intensity

Some image processing algorithms are customized for intensity images. If you want to use one of these algorithms, you must first convert your image to intensity. In this topic, you learn how to use the Color Space Conversion block to accomplish this task. You can use this procedure to convert any R'G'B' image to an intensity image:

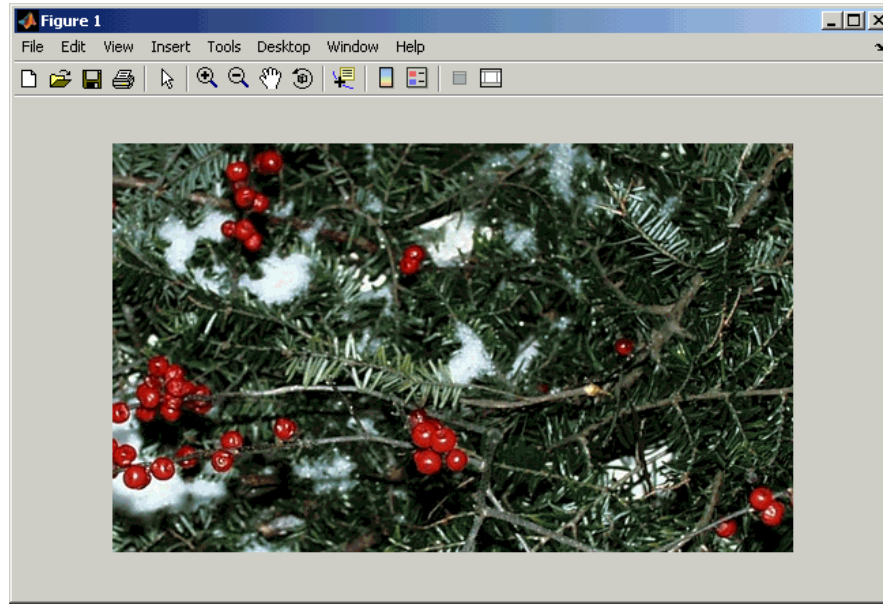
1. Define an R'G'B' image in the MATLAB workspace. To read in an R'G'B' image from a JPG file, at the MATLAB command prompt, type

   ```
   I= imread('greens.jpg');
   ```

   `I` is a 300-by-500-by-3 array of 8-bit unsigned integer values. Each plane of this array represents the red, green, or blue color values of the image.

2. To view the image this matrix represents, at the MATLAB command prompt, type

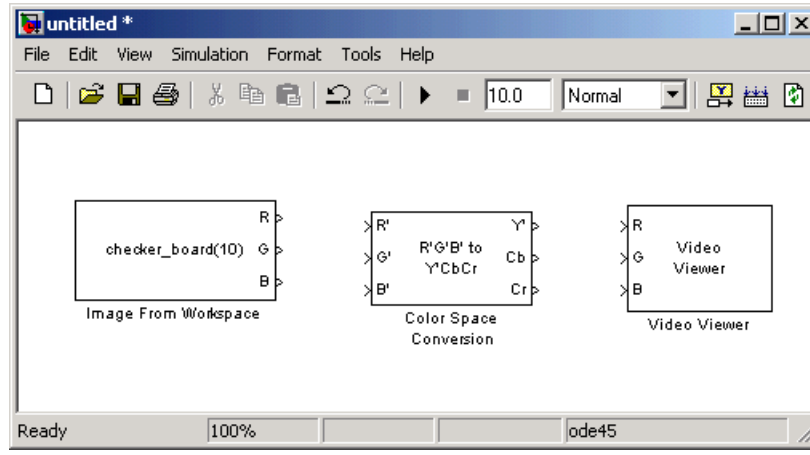
   ```
   imshow(I)
   ```



3 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
Color Space Conversion	Video and Image Processing Blockset / Conversions	1
Video Viewer	Video and Image Processing Blockset / Sinks	1

4 Position the blocks as shown in the following figure.

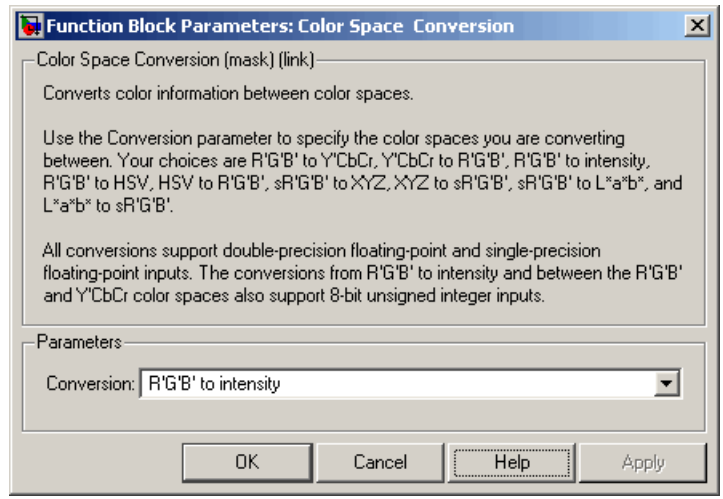


Once you have assembled the blocks needed to convert a R'G'B' image to an intensity image, you are ready to set your block parameters. To do this, double-click the blocks, modify the block parameter values, and click **OK**.

5 Use the Image from Workspace block to import your image from the MATLAB workspace. Set the block parameters as follows:

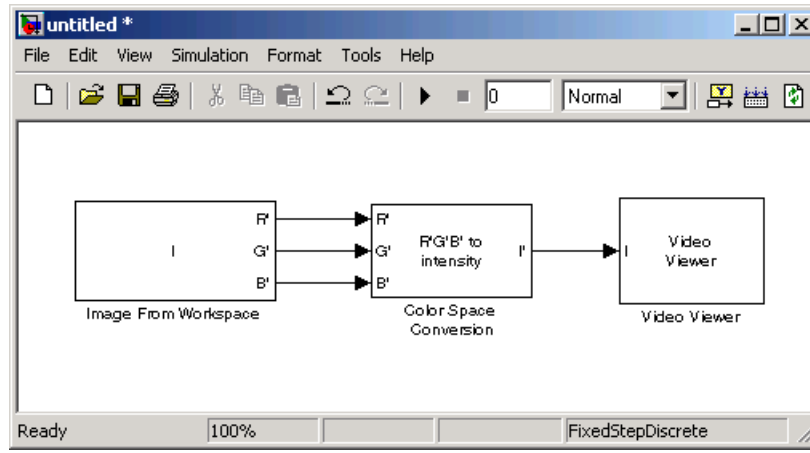
- **Main** pane, **Value** = `I`
- **Output port labels** = `R'|G'|B'`

6 Use the Color Space Conversion block to convert the input values from the R'G'B' color space to intensity. Set the **Conversion** parameter to `R'G'B' to intensity`.



7 View the modified image using the Video Viewer block. Set the **Input image type** to `Intensity`.

8 Connect the blocks so that your model is similar to the following figure.



9 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

10 Run your model.

The image displayed in the Video Viewer window is the intensity version of the `greens.jpg` image. To view the image at its true size, right-click the window and select **Set Display To True Size**.



In this topic, you used the Color Space Conversion block to convert color information from the R'G'B' color space to intensity. For more information on this block, see the Color Space Conversion block reference page.

Chroma Resampling

The Y'CbCr color space separates the luma (Y') component of an image from the chroma (Cb and Cr) components. Luma and chroma, which are calculated using gamma corrected R, G, and B (R', G', B') signals, are different quantities than the CIE chrominance and luminance. Because the human eye is more sensitive to changes in luma than to changes in chroma, you can reduce the bandwidth required for transmission or storage of a signal by removing some of the color information. For this reason, this color space is often used for digital encoding and transmission applications. In the following example, you use the Chroma Resampling block to downsample the Cb and Cr components of an image:

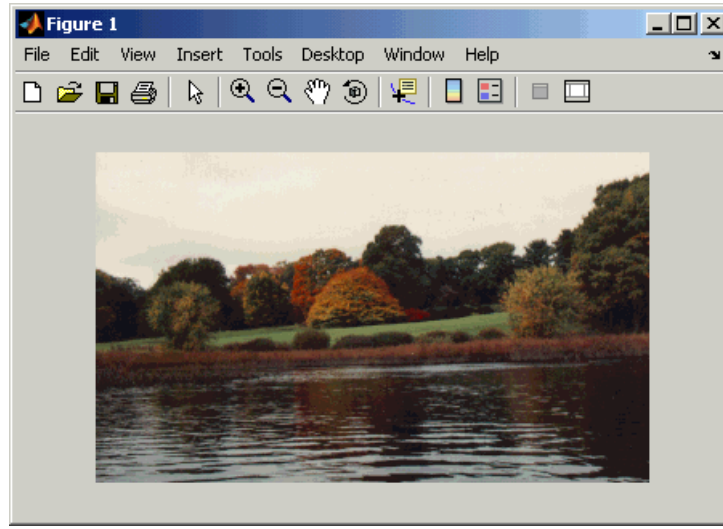
1. Define an RGB image in the MATLAB workspace. To read in an RGB image from a TIF file, at the MATLAB command prompt, type

   ```
   I= imread('autumn.tif');
   ```

   `I` is a 206-by-345-by-3 array of 8-bit unsigned integer values. Each plane of this array represents the red, green, or blue color values of the image.

2. To view the image this array represents, at the MATLAB command prompt, type

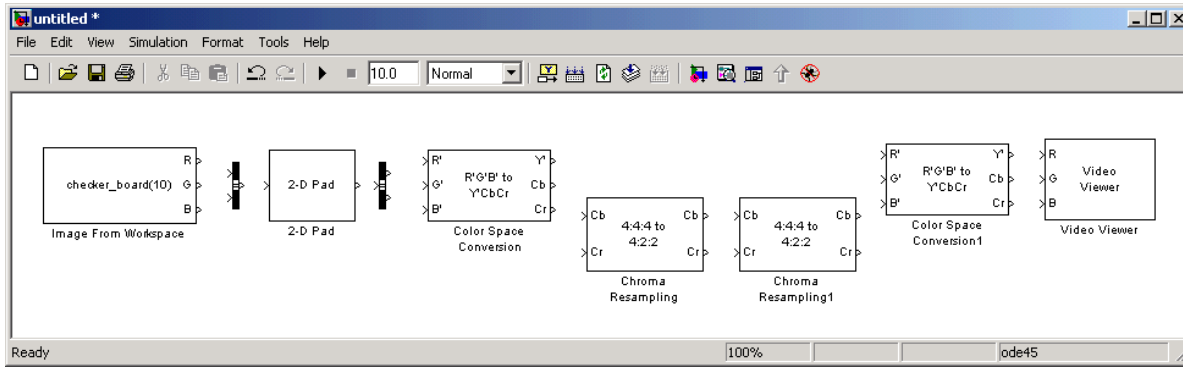
   ```
   imshow(I)
   ```



3 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
2-D Pad	Video and Image Processing Blockset / Utilities	1
Color Space Conversion	Video and Image Processing Blockset / Conversions	2
Chroma Resampling	Video and Image Processing Blockset / Conversions	2
Video Viewer	Video and Image Processing Blockset / Sinks	1
Bus Creator	Simulink / Signal Routing	1
Bus Selector	Simulink / Signal Routing	1

4 Position the blocks as shown in the following figure.



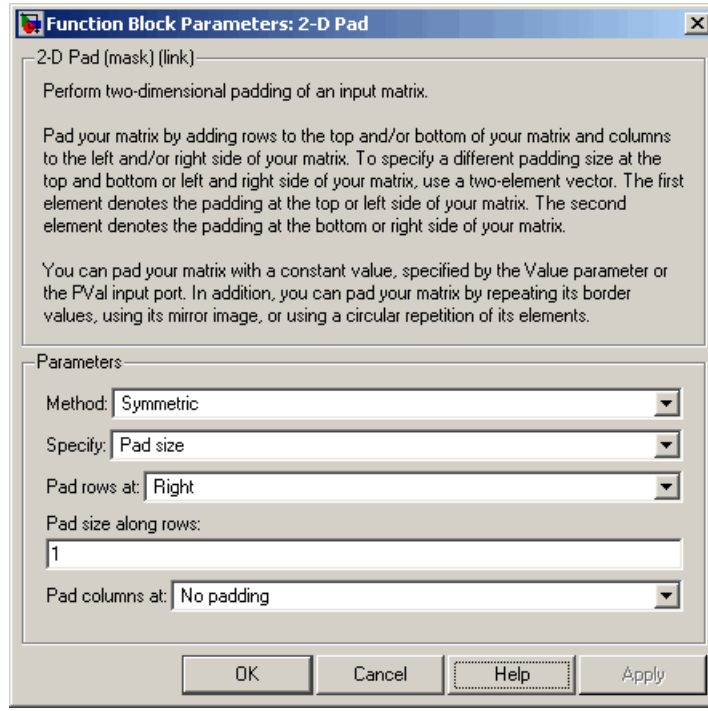
The blocks to the left of and including the Chroma Resampling block represent the transmission portion of the model. The remaining blocks represent the receiving portion of the model. Once you have assembled these blocks, you are ready to set your block parameters. To do this, double-click the blocks, modify the block parameter values, and click **OK**.

5 Use the Image from Workspace block to import your image from the MATLAB workspace. Set the **Value** parameter to `I`.

6 Use the Bus Creator block to combine the R, G, and B, signals into one signal so you can process it with one 2-D Pad block. Set the **Number of inputs** parameter to 3.

7 Use the 2-D Pad block to change the dimensions of the I array from 206-by-345-by-3 to 206-by-346-by-3. You are changing these dimensions because the Chroma Resampling block requires that the dimensions of the input be divisible by 2. Set the block parameters as follows:

- **Method** = `Symmetric`
- **Pad rows at** = `Right`
- **Pad size along rows** = `1`
- **Pad columns at** = `No padding`



The 2-D Pad block adds one column to the right of each plane of the array by repeating its border values. This padding minimizes the effect of the pixels outside the image on the processing of the image.

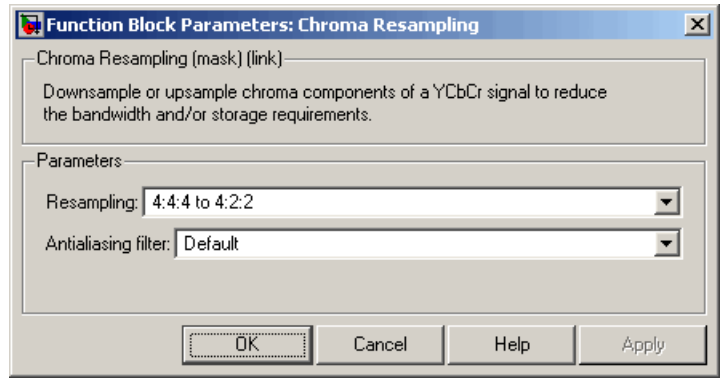
Note When processing video streams, it is computationally expensive to pad every video frame. We recommend changing the dimensions of the video stream before you process it with Video and Image Processing Blockset blocks.

---

8 Use the Bus Selector block to expand the input signal into three separate R, G, and B, signals. You must set the block parameters of this block after you connect a signal to its input port. You configure this block later in this procedure.

9 Use the Color Space Conversion block to convert the input values from the R'G'B' color space to the Y'CbCr color space. The prime symbol indicates a gamma corrected signal. Use the default parameters.

10 Use the Chroma Resampling block to downsample the chroma components of the image from the 4:4:4 format to the 4:2:2 format. Use the default parameters.



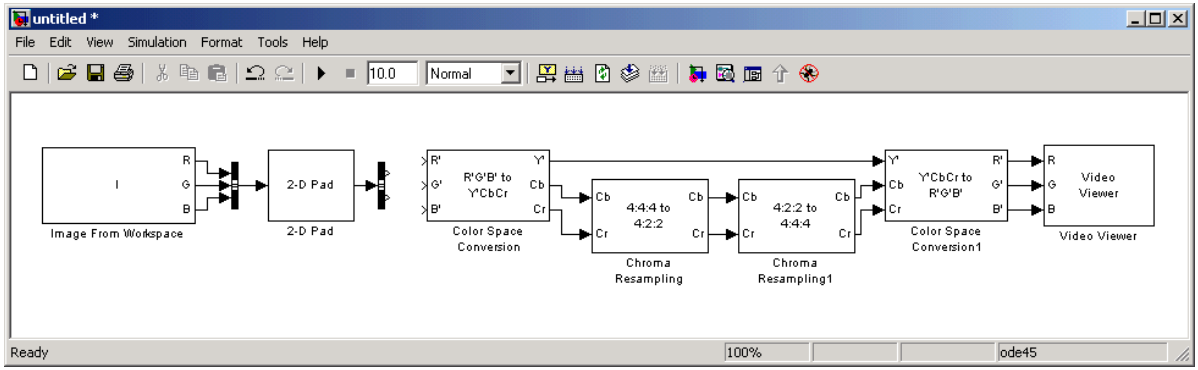
The dimensions of the output of the Chroma Resampling block are smaller than the dimensions of the input. Therefore, the output signal requires less bandwidth for transmission.

11 Use the Chroma Resampling1 block to upsample the chroma components of the image from the 4:2:2 format to the 4:4:4 format. Set the **Resampling** parameter to `4:2:2 to 4:4:4`.

12 Use the Color Space Conversion1 block to convert the input values from the Y'CbCr color space to the R'G'B' color space. Set the **Conversion** parameter to `Y'CbCr to R'G'B'`.

13 Use the Video Viewer block to display the recovered image. Use the default parameters.

14 Connect the blocks as shown in the following figure.



Note that the Bus Selector block, which is to the right of the 2-D Pad block, still needs to be connected to the Color Space Conversion block. You cannot configure the parameters of this block until you connect an input signal to it.

15 Configure the Bus Selector block. Double-click the block. In the **Signals in the bus** pane, select `signal3`. Click **Select** to move `signal3` to the **Selected signals** pane. Click **OK**.

The Bus Selector block now has three output ports.

16 Connect the Bus Selector block to the Color Space Conversion block.

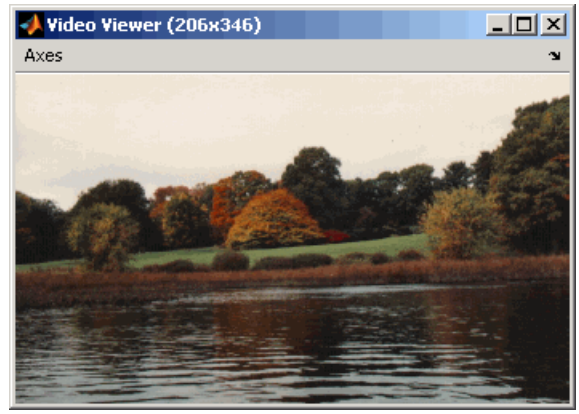
17 Configure Simulink to display signal dimensions next to each signal line. From the **Format** menu, point to **Port/Signal Displays**, and select **Signal Dimensions**.

18 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

19 Run the model.

The recovered image appears in the Video Viewer window. To view the image at its true size, right-click the window and select **Set Display To True Size**.



20 Examine the signal dimensions in your model. The Chroma Resampling block downsamples the Cb and Cr components of the image from 206-by-346 matrices to 206-by-173 matrices. These matrices require less bandwidth for transmission while still communicating the information necessary to recover the image after it is transmitted.

You have used the Chroma Resampling block to downsample the Cb and Cr components of an image. For more information about this block, see the Chroma Resampling block reference page.

5

Geometric Transformation

The Geometric Transformations library contains blocks that enable you to rotate, translate, shear, and resize images.

Interpolation Overview (p. 5-2)	Understand how blocks in the Geometric Transformations library interpolate values.
Rotating an Image (p. 5-6)	Use the Rotate block to continuously rotate an image.
Resizing an Image (p. 5-14)	Use the Resize block to reduce the size of an image.
Cropping an Image (p. 5-21)	Use the Selector block to trim an image down to a region of interest.

Interpolation Overview

The Video and Image Processing Blockset contains blocks that perform geometric transformations. These blocks use interpolation to calculate the appropriate pixel values so that images appear rotated, translated, resized, or sheared.

Note The examples in this section are illustrations of interpolation methods. The block algorithms are implemented in a slightly different way so that they are optimized for speed and memory.

The following sections illustrate the geometric transformation blocks' interpolation methods:

- "Nearest Neighbor Interpolation" on page 5-2 — Understand the basic concepts of nearest neighbor interpolation
- "Bilinear Interpolation" on page 5-3 — Understand the basic concepts of bilinear interpolation
- "Bicubic Interpolation" on page 5-4 — Understand the basic concepts of bicubic interpolation

Nearest Neighbor Interpolation

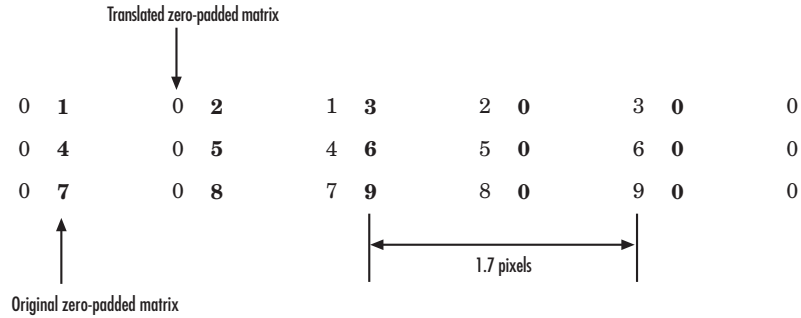
For nearest neighbor interpolation, the block uses the value of nearby translated pixel values for the output pixel values.

For example, suppose this matrix,

```
1 2 3
4 5 6
7 8 9
```

represents your input image. You want to translate this image 1.7 pixels in the positive horizontal direction using nearest neighbor interpolation. The Translate block's nearest neighbor interpolation algorithm is illustrated by the following steps:

1 Zero pad the input matrix and translate it by 1.7 pixels to the right.



2 Create the output matrix by replacing each input pixel value with the translated value nearest to it. The result is the following matrix:

```
0 0 0 1 2 3
0 0 0 4 5 6
0 0 0 7 8 9
```

Note that you wanted to translate the image by 1.7 pixels, but this method translated the image by 2 pixels. Nearest neighbor interpolation is computationally efficient but not as accurate as bilinear or bicubic interpolation.

For more information, see “Interpolation” in the Image Processing Toolbox documentation.

Bilinear Interpolation

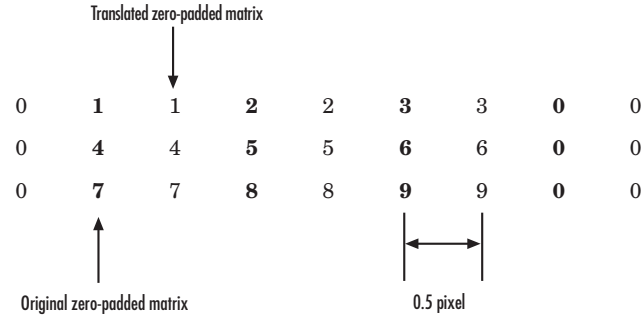
For bilinear interpolation, the block uses the weighted average of two translated pixel values for each output pixel value.

For example, suppose this matrix,

```
1 2 3
4 5 6
7 8 9
```

represents your input image. You want to translate this image 0.5 pixel in the positive horizontal direction using bilinear interpolation. The Translate block's bilinear interpolation algorithm is illustrated by the following steps:

1. Zero pad the input matrix and translate it by 0.5 pixel to the right.



2. Create the output matrix by replacing each input pixel value with the weighted average of the translated values on either side. The result is the following matrix:

$$\begin{matrix} 0.5 & 1.5 & 2.5 & 1.5 \\ 2 & 4.5 & 5.5 & 3 \\ 3.5 & 7.5 & 8.5 & 4.5 \end{matrix}$$

Note that the output matrix has one more column than the input matrix.

For more information, see "Interpolation" in the Image Processing Toolbox documentation.

Bicubic Interpolation

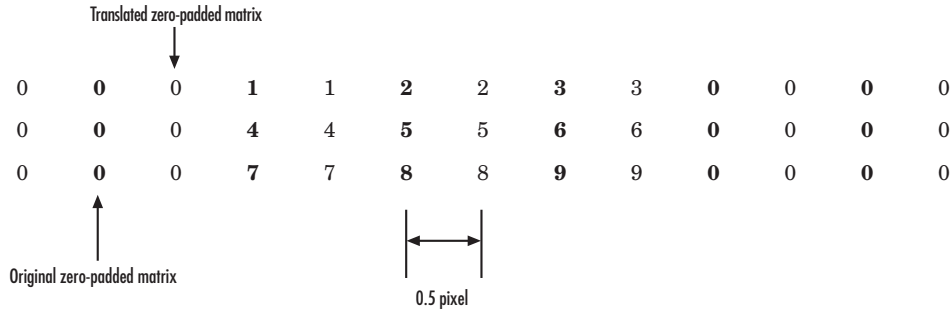
For bicubic interpolation, the block uses the weighted average of four translated pixel values for each output pixel value.

For example, suppose this matrix,

1 2 3
4 5 6
7 8 9

represents your input image. You want to translate this image 0.5 pixel in the positive horizontal direction using bicubic interpolation. The Translate block's bicubic interpolation algorithm is illustrated by the following steps:

1 Zero pad the input matrix and translate it by 0.5 pixel to the right.



2 Create the output matrix by replacing each input pixel value with the weighted average of the two translated values on either side. The result is the following matrix:

0.375 1.5 3 1.625
1.875 4.875 6.375 3.125
3.375 8.25 9.75 4.625

Note that the output matrix has one more column than the input matrix.

For more information, see "Interpolation" in the Image Processing Toolbox documentation.

Rotating an Image

You can use the Rotate block to rotate your image or video stream by a specified angle. In this example, you learn how to use the Rotate block to continuously rotate an image:

1. Define an RGB image in the MATLAB workspace. At the MATLAB command prompt, type

   ```
   A = checker_board;
   ```

   `A` is a 100-by-100-by-3 array of double-precision values. Each plane of the array represents the reg, green, or blue color values of the image.

2. To view the image this matrix represents, at the MATLAB command prompt, type

   ```
   imshow(A)
   ```

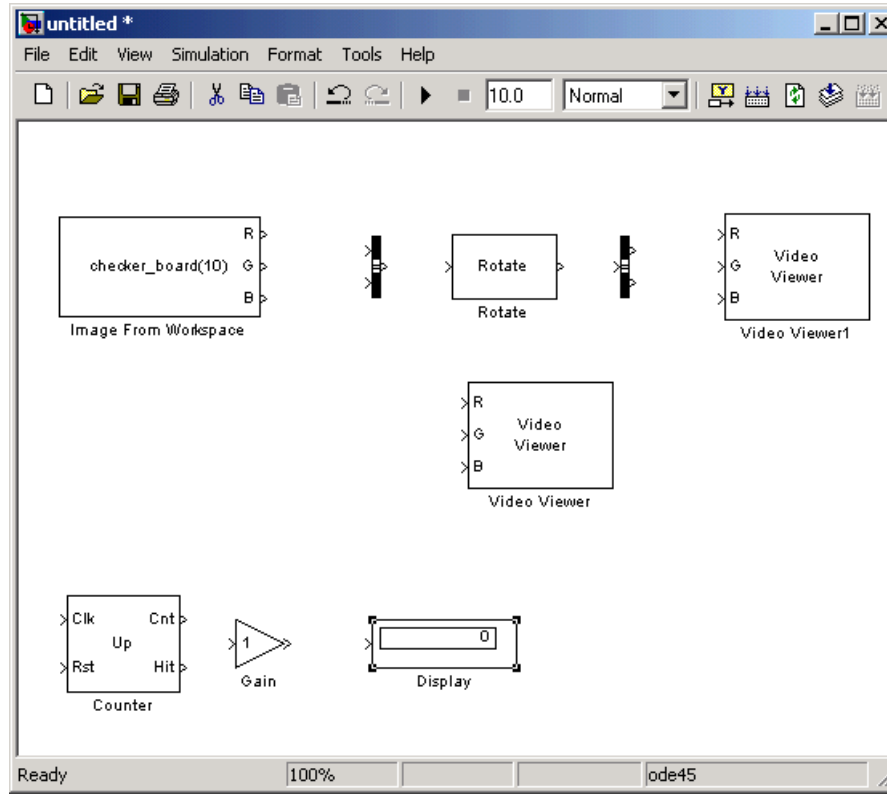


3. Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1

Block	Library	Quantity
Rotate	Video and Image Processing Blockset / Geometric Transformations	1
Video Viewer	Video and Image Processing Blockset / Sinks	2
Bus Creator	Simulink / Signal Routing	1
Bus Selector	Simulink / Signal Routing	1
Gain	Simulink / Math Operations	1
Display	Signal Processing Blockset / Signal Processing Sinks	1
Counter	Signal Processing Blockset / Signal Management / Switches and Counters	1

4 Position the blocks as shown in the following figure.



You are now ready to set your block parameters by double-clicking the blocks, modifying the block parameter values, and clicking **OK**.

5 Use the Image From Workspace block to import the RGB image from the MATLAB workspace. On the **Main** pane, set the **Value** parameter to `A`.

The block outputs the R, G, and B planes of the A array at the output ports.

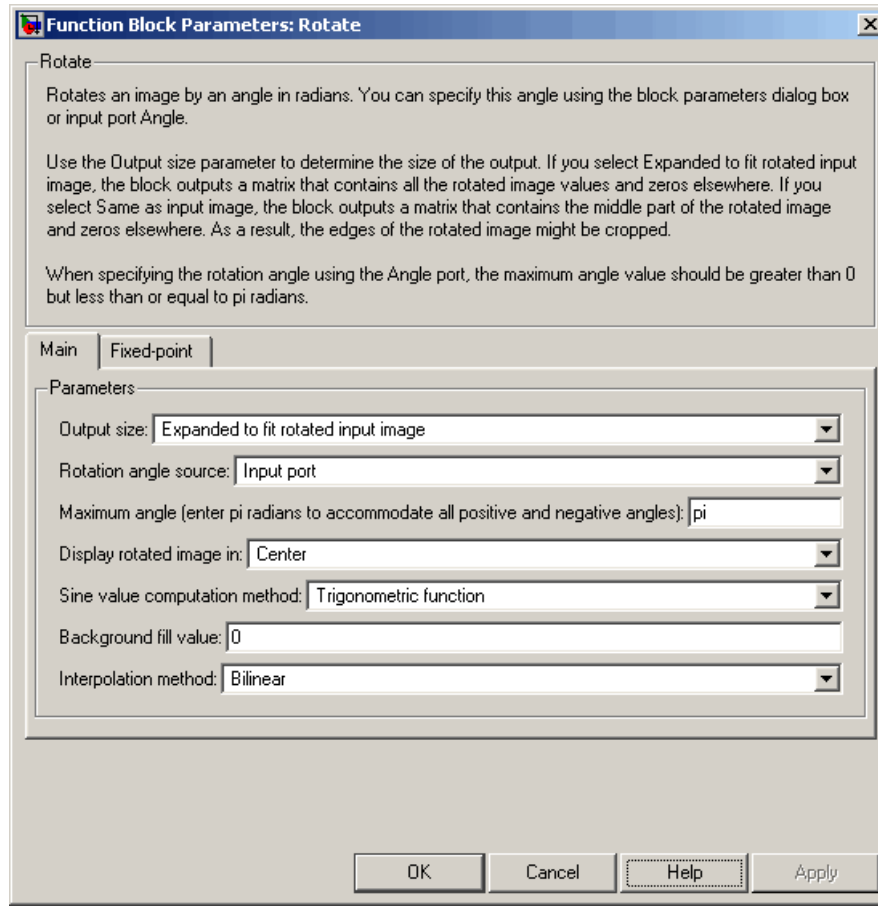
6 Use the Video Viewer block to display the original image. Use the default parameters.

The Video Viewer block automatically displays the original image in the Video Viewer window when you run the model. Because the image is represented by double-precision floating-point values, a value of 0 corresponds to black and a value of 1 corresponds to white.

7 Use the Bus Creator block to combine the R, G, and B planes of the A array into a single signal. This enables you to use one Rotate block to rotate the image. Set the **Number of inputs** parameter to 3.

8 Use the Rotate block to rotate the image. Set the block parameters as follows:

- **Rotation angle source** = `Input port`
- **Sine value computation method** = `Trigonometric function`



The Angle port appears on the block. You will use this port to input a steadily increasing angle. Setting the **Output size** parameter to `Expanded to fit rotated input image` ensures that the block does not crop the output.

9 Use the Bus Selector block to split the R, G, and B planes of the A array into separate signals. You must set the block parameters of this block after you connect a signal to its input port. You configure this block later in this procedure.

10 Use the Video Viewer1 block to display the rotating image. Use the default parameters.

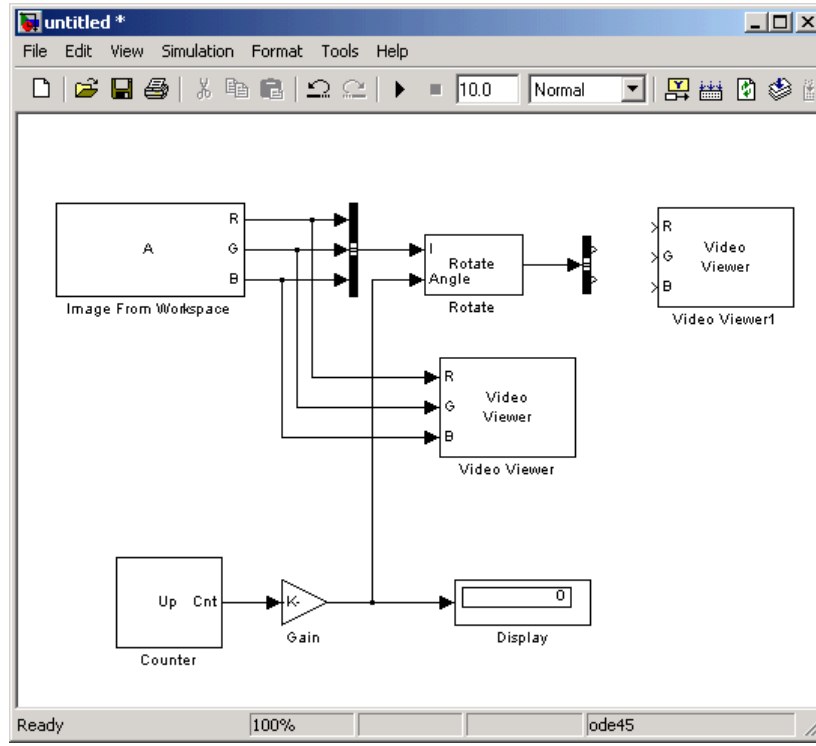
11 Use the Counter block to create a steadily increasing angle. Set the block parameters as follows:

- **Count event** = `Free running`
- **Counter size** = `16 bits`
- **Output** = `Count`
- Clear the **Reset input** check box.
- **Sample time** = `1/30`

The Counter block counts upward until it reaches the maximum value that can be represented by 16 bits. Then, it starts again at zero. You can view its output value on the Display block while the simulation is running. You are using the Counter block from the Signal Processing Blockset because its **Count data type** parameter enables you to specify the data type of its output.

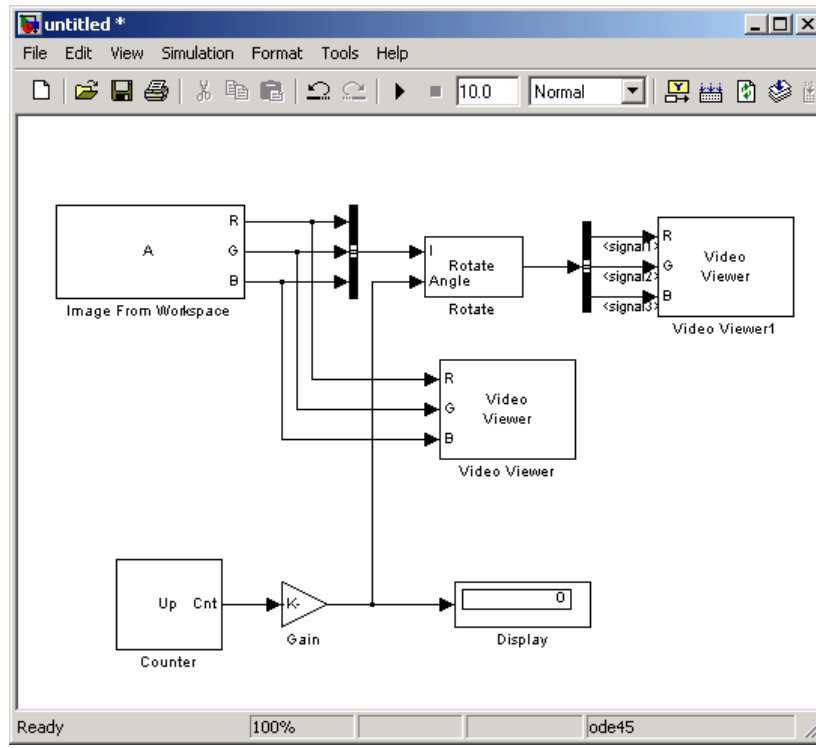
12 Use the Gain block to convert the output of the Counter block from degrees to radians. Set the **Gain** parameter to `pi/180`.

13 Connect the block as shown in the following figure.



14 Double-click the Bus Selector block. In the **Signals in the bus** pane, select `signal3`, and click **Select**. Now, `signal3` appears in the **Selected signals** pane. Click **OK**. Then, connect the Bus Selector block to the Video Viewer1 block.

Your model should look similar to the figure below.

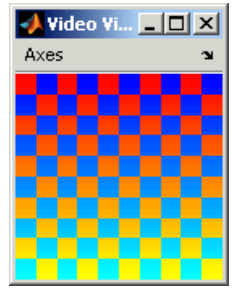


15 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

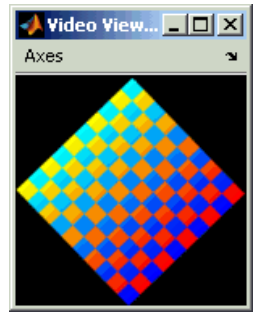
- **Solver** pane, **Stop time** = `inf`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

16 Run the model.

The original image appears in the Video Viewer window.



The rotating image appears in the Video Viewer1 window.



In this example, you used the Rotate block to continuously rotate your image. For more information about this block, see the Rotate block reference page. For more information about other geometric transformation blocks, see the Resize and Shear block reference pages.

Note If you are on a Windows operating system, you can replace the Video Viewer block with the To Video Display block, which supports code generation.

Resizing an Image

You can use the Resize block to change the size of your image or video stream. In this example, you learn how to use the Resize block to reduce the size of an image:

1. Define an intensity image in the MATLAB workspace. At the MATLAB command prompt, type

   ```
   I = imread('moon.tif');
   ```

   `I` is a 537-by-358 matrix of 8-bit unsigned integer values.

2. To view the image this matrix represents, at the MATLAB command prompt, type

   ```
   imshow(I)
   ```

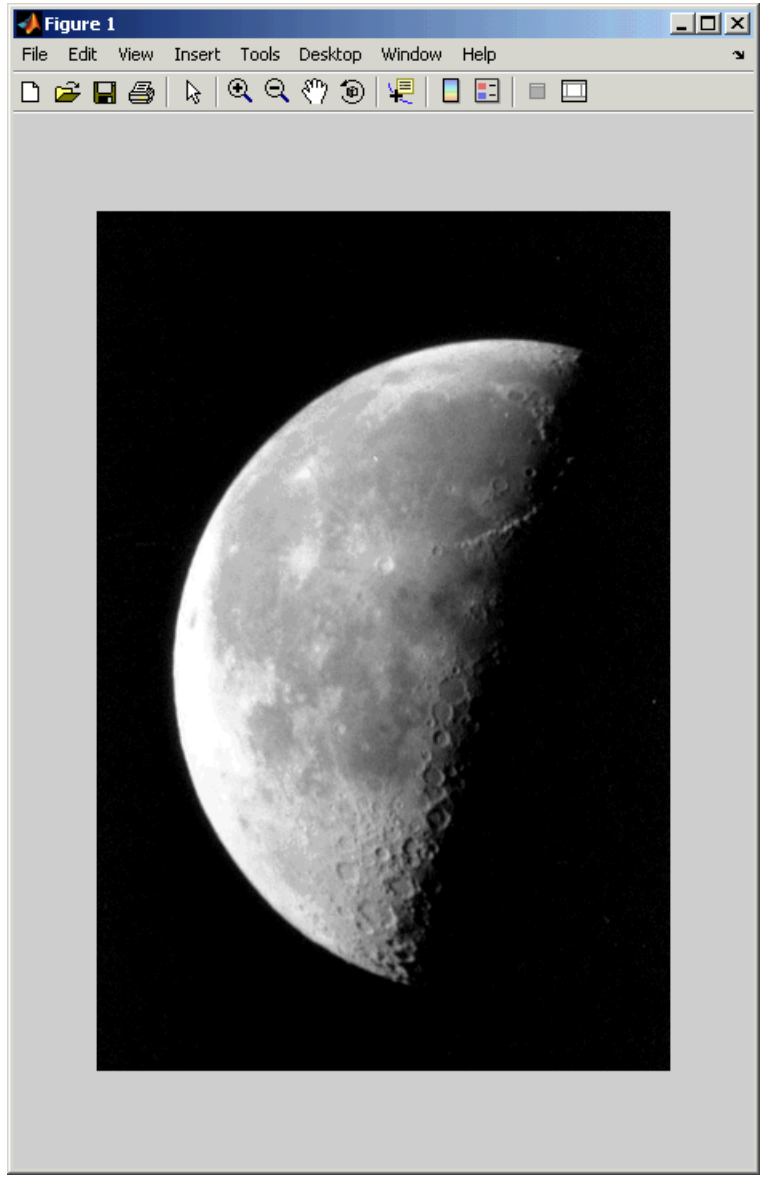
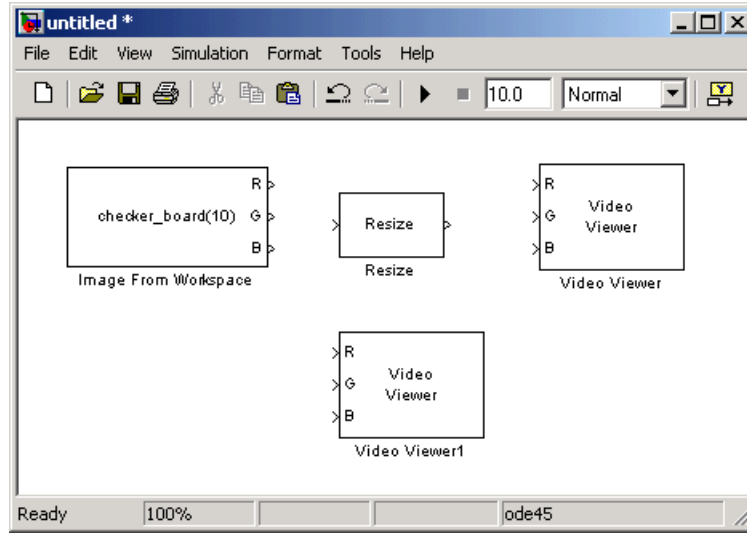


Image Courtesy of Michael Myers

3 Create a new Simulink model, and add to it the blocks shown in the following table.

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

4 Position the blocks as shown in the following figure.



5 Use the Image From Workspace block to import the intensity image from the MATLAB workspace. Set the parameters as follows:

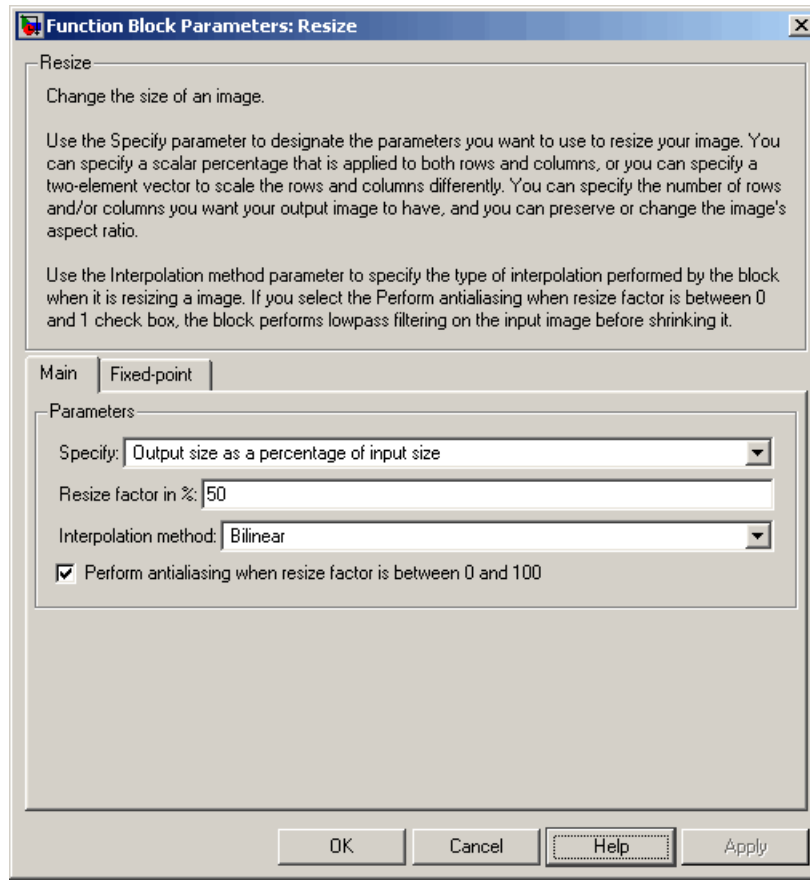
- **Main** pane, **Value** = `I`
- **Main** pane, **Output port labels** = `Image`

The block outputs the image at the Image port.

6 Use the Video Viewer1 block to display the original image. Set the **Input image type** parameter to `Intensity`.

The Video Viewer1 block automatically displays the original image in the Video Viewer1 window when you run the model.

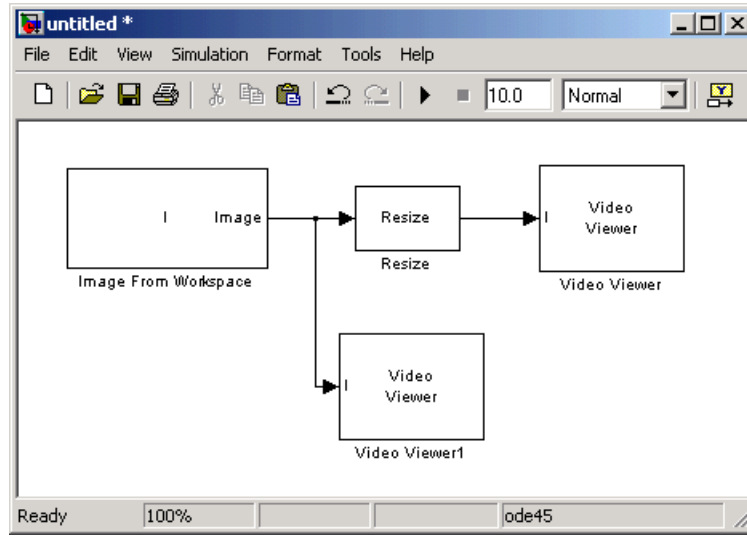
7 Use the Resize block to shrink the image. Set the **Resize factor in %** parameter to `50`.



The Resize block shrinks the image to half its original size.

8 Use the Video Viewer block to display the modified image. Set the **Input image type** parameter to `Intensity`.

9 Connect the blocks as shown in the figure below.



10 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

11 Run the model.

The original image appears in the Video Viewer1 window. To view the image at its true size, right-click the window and select **Set Display To True Size**.



The reduced image appears in the Video Viewer window. Right-click the window and select **Set Display To True Size**. The smaller image is half the size of the original image.



In this example, you used the Resize block to shrink an image. For more information about this block, see the Resize block reference page. For more information about other geometric transformation blocks, see the Rotate, Shear, and Translate block reference pages.

Cropping an Image

You can use the Selector block to crop your image or video stream. In this example, you learn how to use the Selector block to trim an image down to a particular region of interest:

1. Define an intensity image in the MATLAB workspace. At the MATLAB command prompt, type

   ```
   I = imread('coins.png');
   ```

   `I` is a 246-by-300 matrix of 8-bit unsigned integer values.

2. To view the image this matrix represents, at the MATLAB command prompt, type

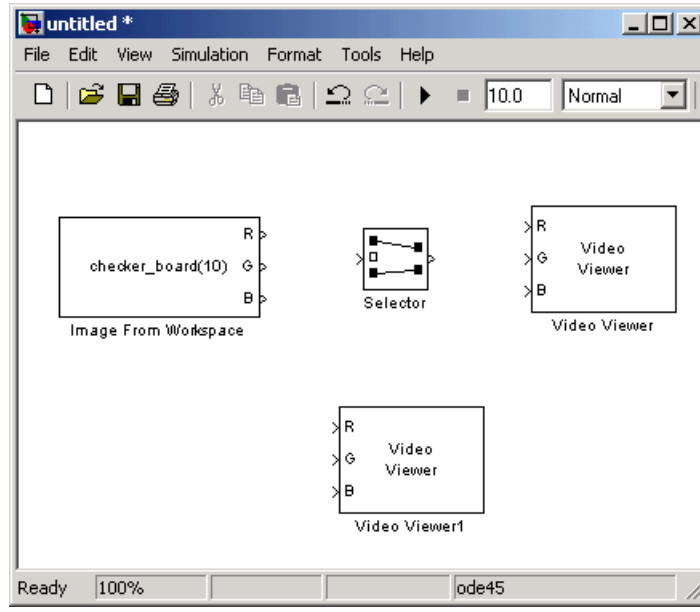
   ```
   imshow(I)
   ```



3. Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
Video Viewer	Video and Image Processing Blockset / Sinks	2
Selector	Simulink / Signal Routing	1

4 Position the blocks as shown in the following figure.



5 Use the Image From Workspace block to import the intensity image from the MATLAB workspace. Set the parameters as follows:

- **Main** pane, **Value** = `I`
- **Main** pane, **Output port labels** = `Image`

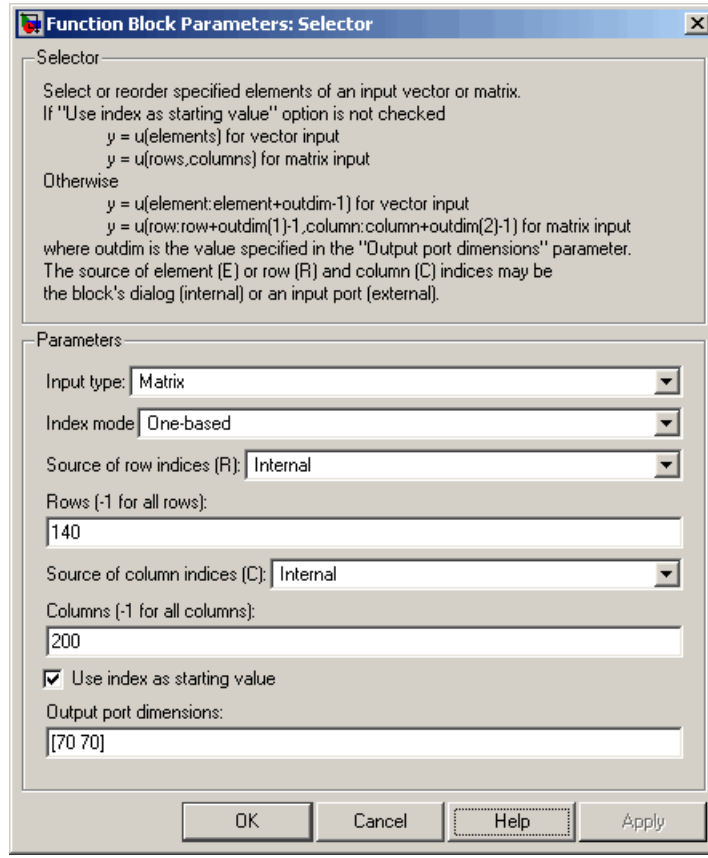
The block outputs the image at the Image port.

6 Use the Video Viewer1 block to display the original image. Set the **Input image type** parameter to `Intensity`.

   The Video Viewer1 block automatically displays the original image in the Video Viewer1 window when you run the model.

7 Use the Selector block to crop the image. Set the block parameters as follows:

   - **Input type** = `Matrix`
   - **Rows** = `140`
   - **Columns** = `200`
   - Select the **Use index as starting value** check box.
   - **Output port dimensions** = `[70 70]`

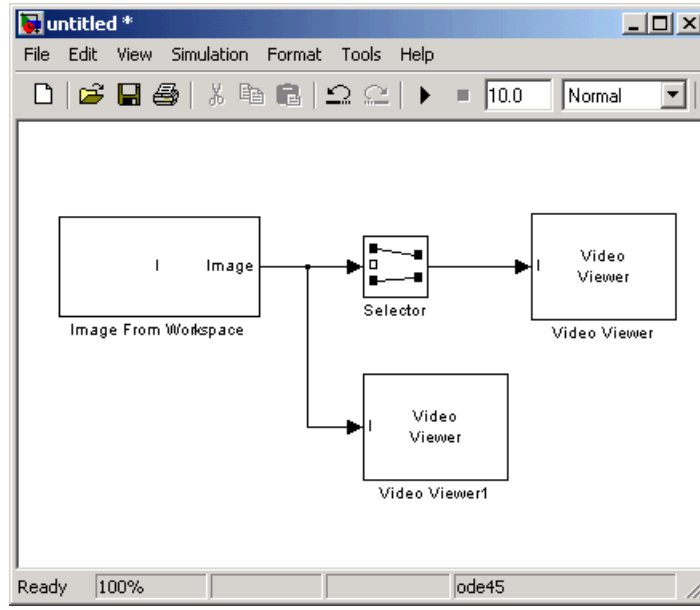


The Selector block starts at row 140 and column 200 of the image and outputs the next 70 rows and columns of the image.

8 Use the Video Viewer block to display the cropped image. Set the **Input image type** parameter to `Intensity`.

The Video Viewer block automatically displays the modified image in the Video Viewer window when you run the model.

9 Connect the blocks as shown in the figure below.



10 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

11 Run the model.

The original image appears in the Video Viewer1 window. To view the image at its true size, right-click the window and select **Set Display To True Size**.



The cropped image appears in the Video Viewer window. The image below is shown at its true size.



In this example, you used the Selector block to crop an image. For more information about the Selector block, see the Simulink documentation. For information about the `imcrop` function, see the Image Processing Toolbox documentation.

6

Morphological Operations

Morphological image analysis can be used to perform image filtering, image segmentation, and measurement operations.

Overview of Morphology (p. 6-2)	Learn about morphological operations and which Video and Image Processing blocks can be used to perform them.
Counting Objects in an Image (p. 6-3)	Use the Opening and Label blocks to determine the number of spokes in a wheel.
Correcting for Nonuniform Illumination (p. 6-11)	Use the Opening block to correct for uneven lighting in an image.

Overview of Morphology

Morphology is the study of the shape and form of objects. Morphological image analysis can be used to perform

- Object extraction
- Image filtering operations, such as removal of small objects or noise from an image
- Image segmentation operations, such as separating connected objects
- Measurement operations, such as texture analysis and shape description

The Video and Image Processing Blockset contains blocks that perform morphological operations such as erosion, dilation, opening, and closing. Often, you need to use a combination of these blocks to perform your morphological image analysis. The examples in this chapter show you how to use blocks from the Morphological Operations library to count the number of objects in an image and how to correct for uneven illumination.

For more information, see “Morphological Operations” in the Image Processing Toolbox documentation.

Counting Objects in an Image

In this example, you import an intensity image of a wheel from the MATLAB workspace and convert it to binary. Then, using the Opening and Label blocks, you count the number of spokes in the wheel. You can use similar techniques to count objects in other intensity images. However, you might need to use additional morphological operators and different structuring elements:

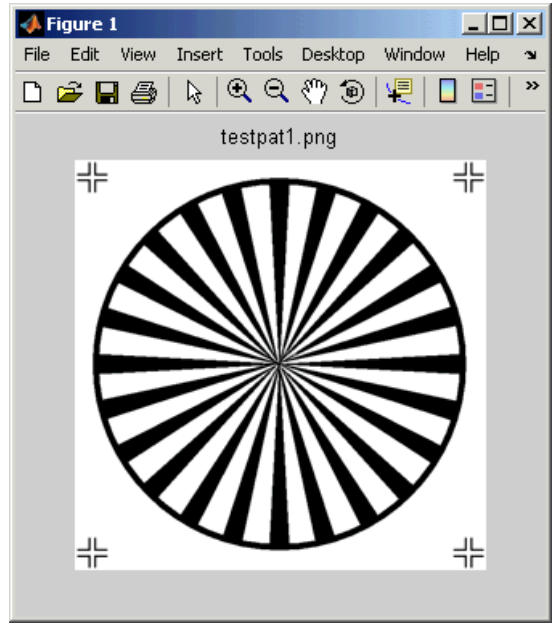
1. Define an intensity image in the MATLAB workspace. To read in an intensity image from a PNG file, at the MATLAB command prompt, type

   ```
   I= imread('testpat1.png');
   ```

   `I` is a 256-by-256 matrix of 8-bit unsigned integers.

2. To view the image this matrix represents, at the MATLAB command prompt, type

   ```
   imshow(I)
   ```

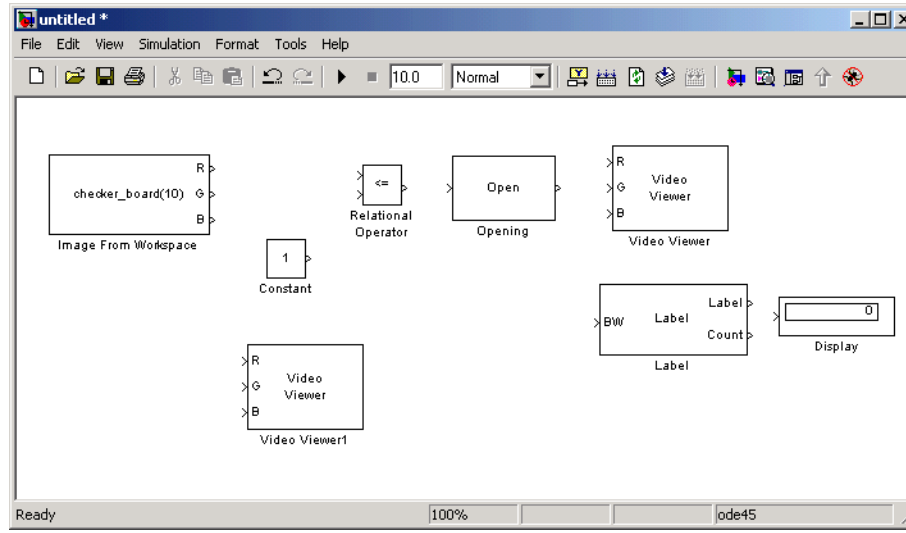


The file `testpat1.png` is an intensity image of a wheel that contains 24 black spokes.

3 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
Opening	Video and Image Processing Blockset / Morphological Operations	1
Label	Video and Image Processing Blockset / Morphological Operations	1
Video Viewer	Video and Image Processing Blockset / Sinks	2
Constant	Simulink / Sources	1
Relational Operator	Simulink / Logic and Bit Operations	1
Display	Signal Processing Blockset/ Signal Processing Sinks	1

4 Position the blocks as shown in the following figure. The unconnected ports disappear when you set block parameters.



You are now ready to set your block parameters by double-clicking the blocks, modifying the block parameter values, and clicking **OK**.

5 Use the Image From Workspace block to import your image from the MATLAB workspace. Set the block parameters as follows:

- **Main** pane, **Value** = `I`
- **Main** pane, **Output port labels** = `Image`

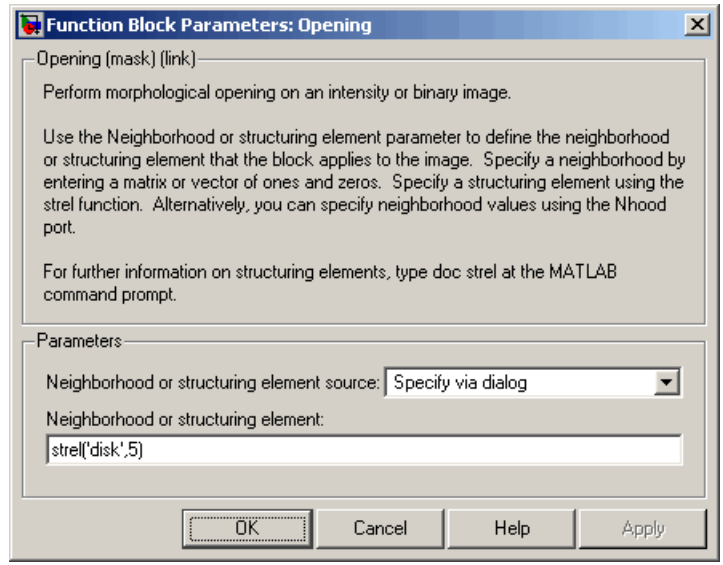
6 Use the Constant block to define a threshold value for the Relational Operator block. Set the **Constant value** parameter to `200`.

7 Use the Video Viewer1 block to view the original image. Set the **Input image type** parameter to `Intensity`.

8 Use the Relational Operator block to perform a thresholding operation that converts your intensity image to a binary image. Set the **Relational Operator** parameter to <.

If the input to the Relational Operator block is less than 200, its output is 1; otherwise, its output is 0. You must threshold your intensity image because the Label block expects binary input. Also, the objects it counts must be white.

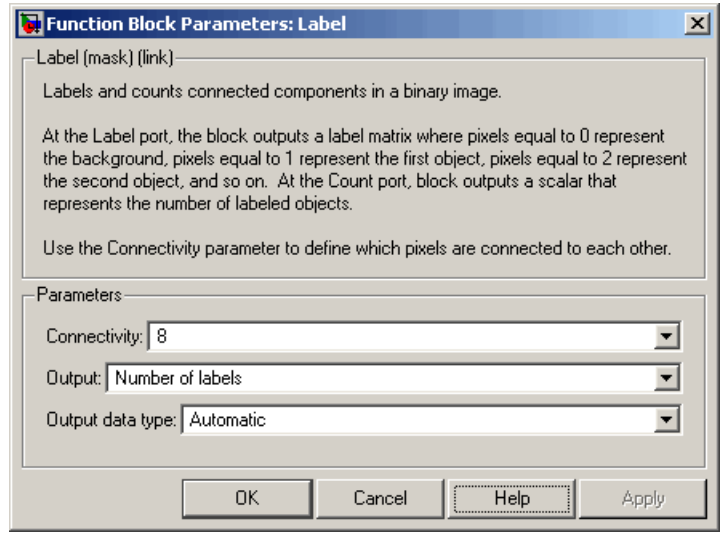
9 Use the Opening block to separate the spokes from the rim and from each other at the center of the wheel. Use the default parameters.



The `strel` function creates a circular STREL object with a radius of 5 pixels. When working with the Opening block, pick a STREL object that fits within the objects you want to keep. It often takes experimentation to find the neighborhood or STREL object that best suits your application.

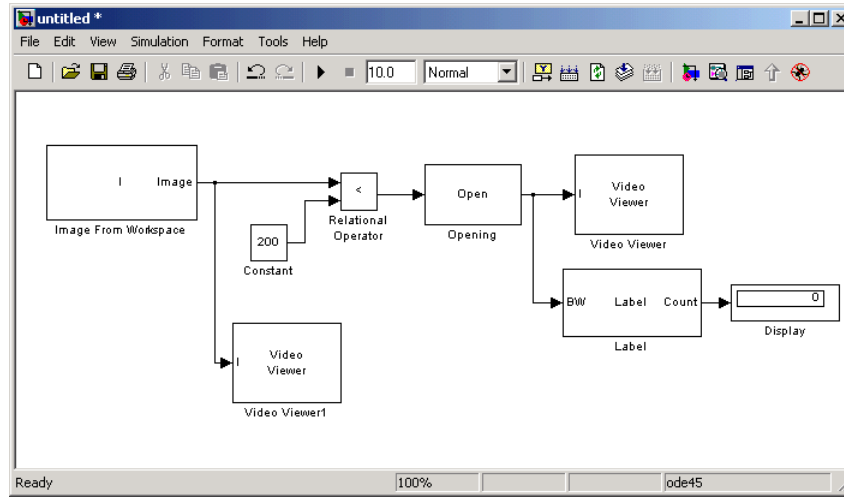
10 Use the Video Viewer block to view the opened image. Set the **Input image type** parameter to `Intensity`.

11 Use the Label block to count the number of spokes in the input image. Set the **Output** parameter to `Number of labels`.



12 The Display block displays the number of spokes in the input image. Use the default parameters.

13 Connect the block as shown in the figure below.

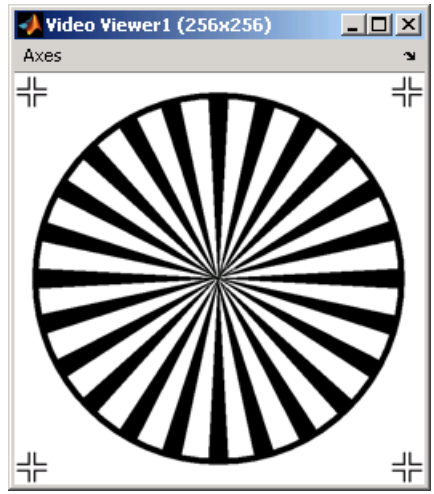


14 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

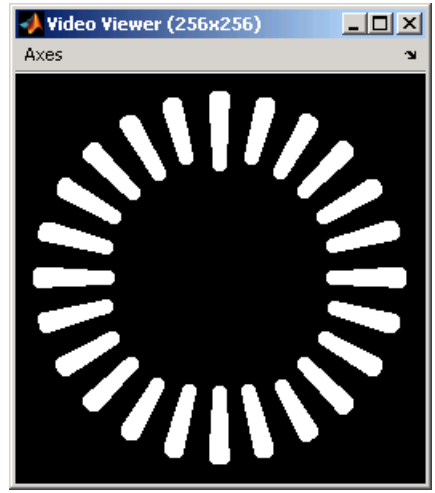
- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

15 Run the model.

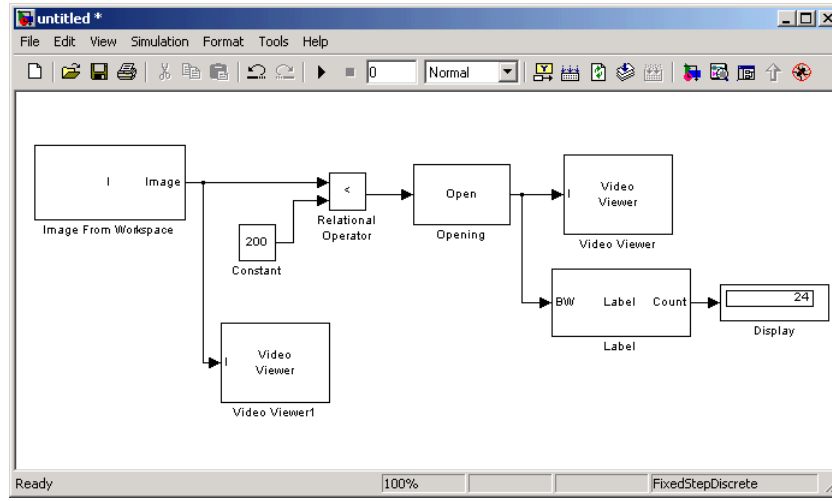
The original image appears in the Video Viewer1 window. To view the image at its true size, right-click the window and select **Set Display To True Size**.



The opened image appears in appears in the Video Viewer window. The image below is shown at its true size.



As you can see in the figure above, the spokes are now separate white objects. In the model, the Display block correctly indicates that there are 24 distinct spokes.



You have used the Opening and Label blocks to count the number of spokes in an image. For more information about these blocks, see the Opening and Label block reference pages. If you want to send the number of spokes to

the MATLAB workspace, use the To Workspace block in Simulink or the Signal to Workspace block in the Signal Processing Blockset. For more information about STREL objects, see `strel` in the Image Processing Toolbox documentation.

Correcting for Nonuniform Illumination

Global threshold techniques, which are often the first step in object measurement, cannot be applied to unevenly illuminated images. To correct this problem, you can change the lighting conditions and take another picture, or you can use morphological operators to even out the lighting in the image. Once you have corrected for nonuniform illumination, you can pick a global threshold that delineates every object from the background. In this topic, you use the Opening block to correct for uneven lighting in an intensity image:

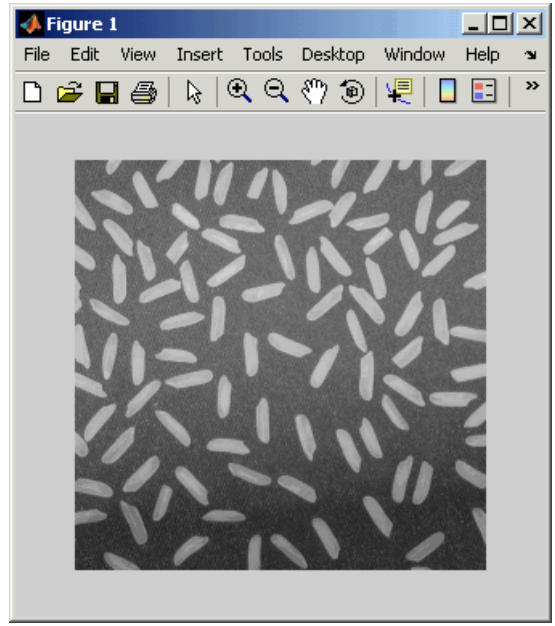
1. Define an intensity image in the MATLAB workspace. To read in an intensity image from a PNG file, at the MATLAB command prompt, type

   ```
   I= imread('rice.png');
   ```

   `I` is a 256-by-256 matrix of 8-bit unsigned integer values.

2. To view the image this matrix represents, at the MATLAB command prompt, type

   ```
   imshow(I)
   ```

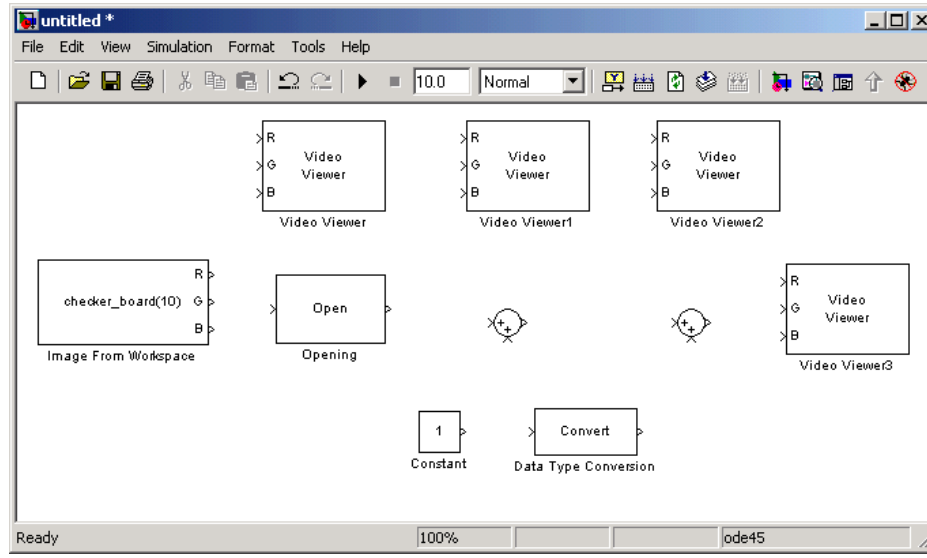


This image is darker at the bottom than at the top. You want to create a model to even out this lighting.

3 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
Opening	Video and Image Processing Blockset / Morphological Operations	1
Video Viewer	Video and Image Processing Blockset / Sinks	4
Constant	Simulink / Sources	1
Sum	Simulink / Math Operations	2
Data Type Conversion	Simulink / Signal Attributes	1

4 Position the blocks as shown in the following figure.



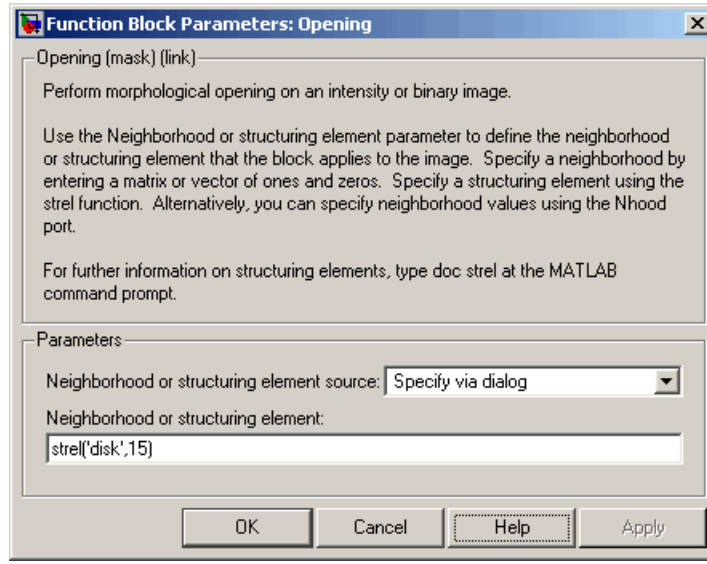
Once you have assembled the blocks required to correct for uneven illumination, you need to set your block parameters. To do this, double-click the blocks, modify the block parameter values, and click **OK**.

5 Use the Image From Workspace block to import the intensity image into your model. Set the block parameters as follows:

- **Main** pane, **Value** = `R`
- **Main** pane, **Output port labels** = `Image`

6 Use the Video Viewer block to view the original image. Set the **Input image type** parameter to `Intensity`.

7 Use the Opening block to estimate the background of the image. Set the **Neighborhood or structuring element** parameter to `strel('disk',15)`.



The `strel` function creates a circular STREL object with a radius of 15 pixels. When working with the Opening block, pick a STREL object that fits within the objects you want to keep. It often takes experimentation to find the neighborhood or STREL object that best suits your application.

8. Use the Video Viewer1 block to view the background estimated by the Opening block. Set the **Input image type** parameter to `Intensity`.

9. Use the first Sum block to subtract the estimated background from the original image. Set the block parameters as follows:

   - **Icon shape** = `rectangular`
   - **List of signs** = `-+`

10. Use the Video Viewer2 block to view the result of subtracting the background from the original image. Set the **Input image type** parameter to `Intensity`.

11. Use the Constant block to define an offset value. Set the **Constant value** parameter to `80`.

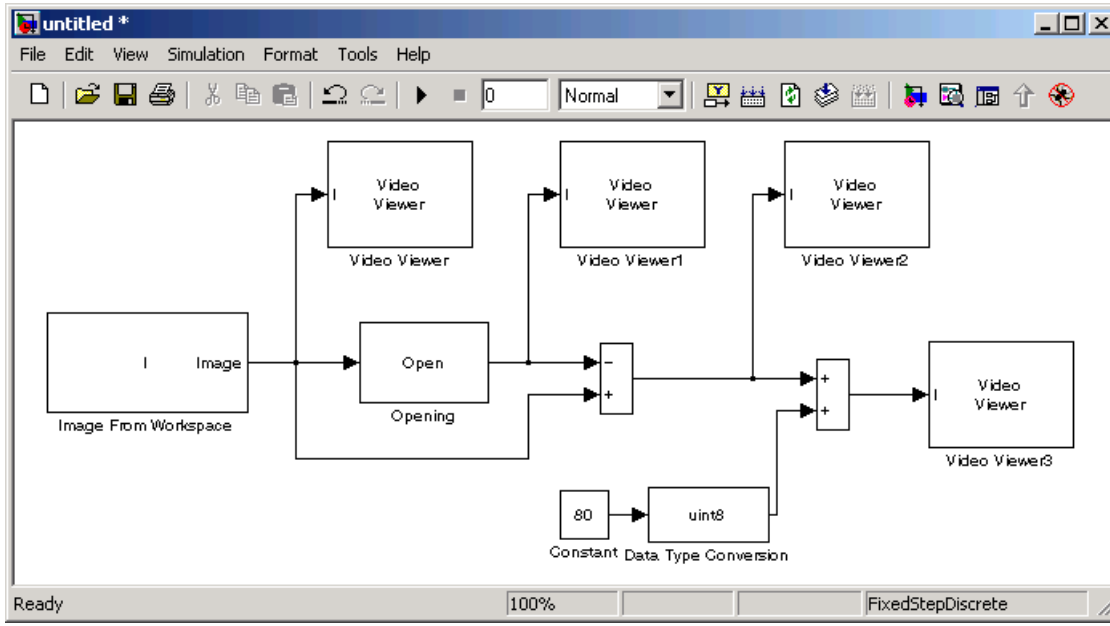
12 Use the Data Type Conversion block to convert the offset value to an 8-bit unsigned integer. Set the **Output data type mode** parameter to `uint8`.

13 Use the second Sum block to lighten the image so that it has the same brightness as the original image. Set the block parameters as follows:

- **Icon shape** = `rectangular`
- **List of signs** = `++`

14 Use the Video Viewer3 block to view the corrected image. Set the **Input image type** parameter to `Intensity`.

15 Connect the blocks as shown in the figure below.



16 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`

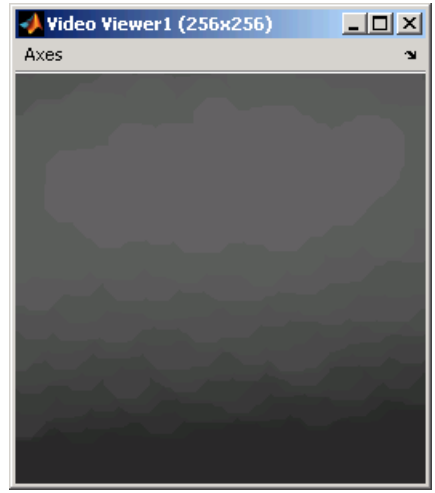
- **Solver** pane, **Solver** = `discrete (no continuous states)`

17 Run the model.

The original image appears in the Video Viewer window. To view the image at its true size, right-click the window and select **Set Display To True Size**.



The estimated background appears in the Video Viewer1 window. The image below is shown at its true size.

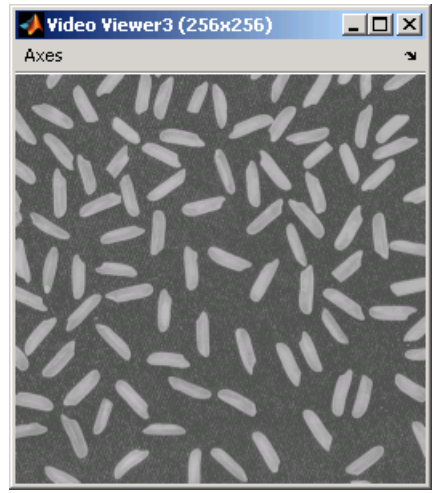


The image without the estimated background appears in the Video Viewer2 window. The image below is shown at its true size.



The image shown above is too dark. The Constant block provides an offset value that you used to brighten the image.

The corrected image appears in the Video Viewer3 window. Note that the corrected image has even lighting. The image below is shown at its true size.



In this section, you have used the Opening block to remove irregular illumination from an image. For more information about this block, see the Opening block reference page. For related information, see the Top-hat block reference page. For more information about STREL objects, see the `strel` function in the Image Processing Toolbox documentation.

7

Analysis and Enhancement

You can use Video and Image Processing Blockset blocks to learn more about the structure of images as well as to improve them.

Feature Extraction (p. 7-2)	Learn more about the content of images.
Image Enhancement (p. 7-26)	Understand how to improve image characteristics.
Pixel Statistics (p. 7-58)	Determine information about the data values that make up an image using blocks from the Statistics library.

Feature Extraction

Feature extraction techniques return information about the structure of an image. For example, you can use them to find edges, locations, and attributes of objects.

This section includes the following topics:

- "Finding Edges in Images" on page 7-2 — Use the Edge Detection block to find object boundaries
- "Finding Lines in Images" on page 7-9 — Use the Hough Transform, Find Local Maxima, and Hough Lines blocks to find the longest line in an image
- "Measuring an Angle Between Lines" on page 7-17— Use the Hough Transform, Hough Lines, and Draw Shapes block to draw two lines on an image and measure the angle between them

Finding Edges in Images

You can use the Edge Detection block to find the edges of objects in an image. This block finds the pixel locations where the magnitude of the gradient of intensity is larger than a threshold value. These locations typically occur at the boundaries of objects. In this section, you use the Edge Detection block to find the edges of rice grains in an intensity image:

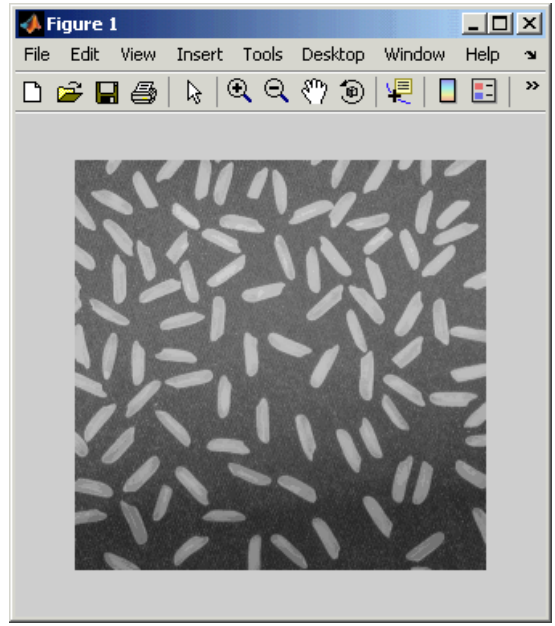
1. Define an intensity image in the MATLAB workspace. To read in an intensity image from a PNG file, at the MATLAB command prompt, type

   ```
   I= imread('rice.png');
   ```

   `I` is a 256-by-256 matrix of 8-bit unsigned integers.

2. To view the image this matrix represents, at the MATLAB command prompt, type

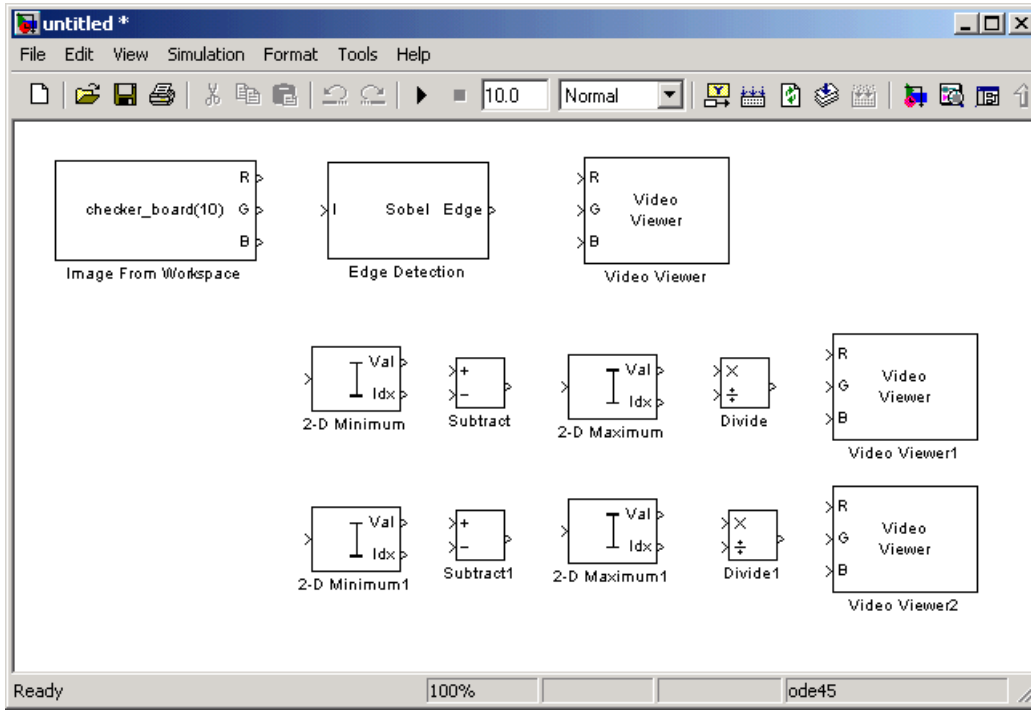
   ```
   imshow(I)
   ```



3 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
Edge Detection	Video and Image Processing Blockset / Analysis & Enhancement	1
2-D Minimum	Video and Image Processing Blockset / Statistics	2
2-D Maximum	Video and Image Processing Blockset / Statistics	2
Video Viewer	Video and Image Processing Blockset / Sinks	3
Subtract	Simulink / Math Operations	2
Divide	Simulink / Math Operations	2

4 Place the blocks so that your model resembles the following figure.



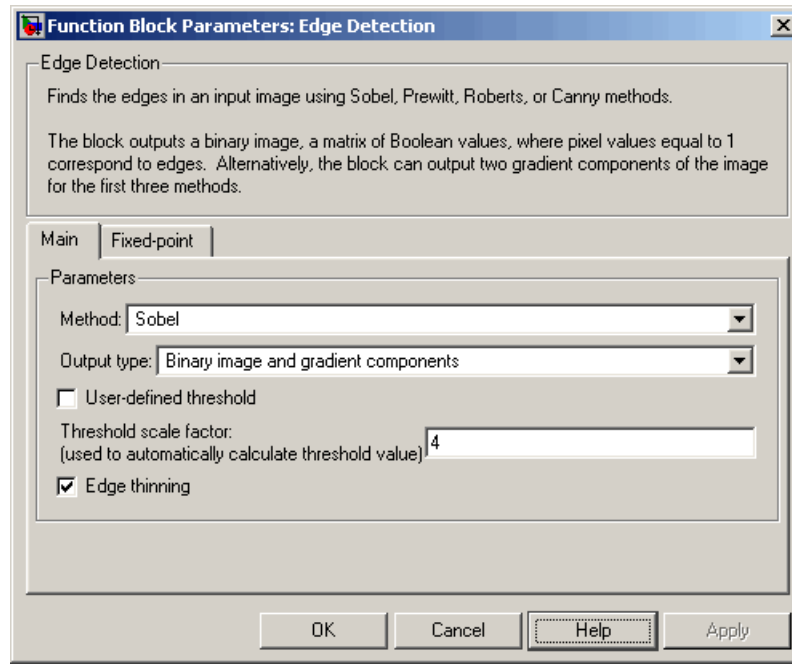
You are now ready to set your block parameters by double-clicking the blocks, modifying the block parameter values, and clicking **OK**.

5 Use the Image From Workspace block to import your image from the MATLAB workspace. Set the block parameters as follows:

- **Main** pane, **Value** = `I`
- **Main** pane, **Output port labels** = `Image`
- **Data Types** pane, **Output data type** = `double`

6 Use the Edge Detection block to find the edges in the image. Set the block parameters as follows:

- **Output type** = `Binary image and gradient components`
- Select the **Edge thinning** check box.



The Edge Detection block convolves the input matrix with the Sobel kernel to calculate the gradient components of the image that correspond to the horizontal and vertical edge responses. The block outputs these components at the Gh and Gv ports, respectively. Then it performs a thresholding operation on these gradient components to find the binary image, a matrix filled with 1s and 0s. The nonzero elements of this matrix correspond to the edge pixels and the zero elements correspond to the background pixels. The block outputs the binary image at the Edge port.

7 View the binary image using the Video Viewer block. Set the **Input image type** parameter to `Intensity`.

The matrices output from the Gv and Gh ports of the Edge Detection block are composed of double-precision floating-point values. You must scale these matrix values between 0 in 1 to display them using the Video Viewer blocks.

8 Use the 2-D Minimum blocks to find the minimum value of Gv and Gh matrices. Set the **Mode** parameters to `Value`.

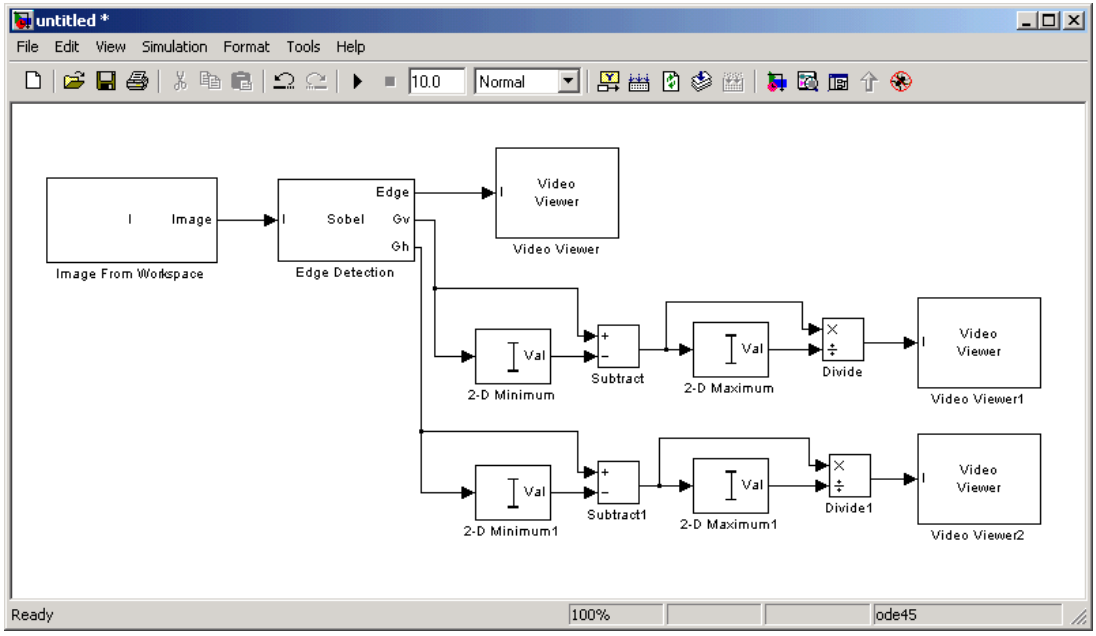
9 Use the Subtract blocks to subtract the minimum values from each element of the Gv and Gh matrices. This process ensures that the minimum value of these matrices is 0. Use the default parameters.

10 Use the 2-D Maximum blocks to find the maximum value of the new Gv and Gh matrices. Set the **Mode** parameters to `Value`.

11 Use the Divide blocks to divide each element of the Gv and Gh matrices by their maximum value. This normalization process ensures that these matrices range between 0 and 1. Use the default parameters.

12 View the gradient components of the image using the Video Viewer1 and Video Viewer2 blocks. Set the **Input image type** parameter to `Intensity`.

13 Connect the blocks as shown in the figure below.

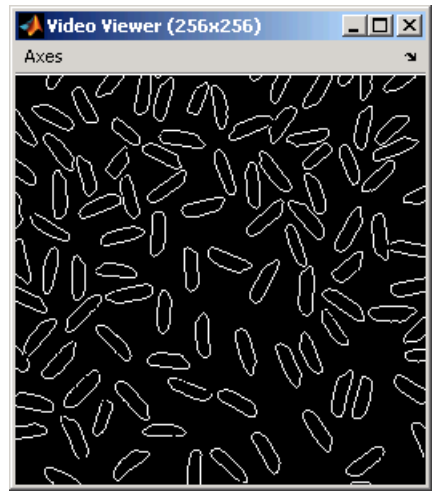


14 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

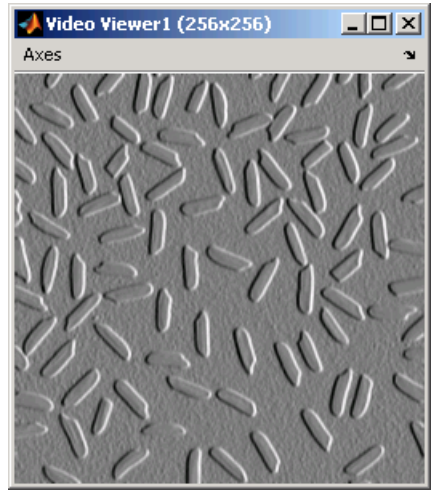
- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

15 Run your model.

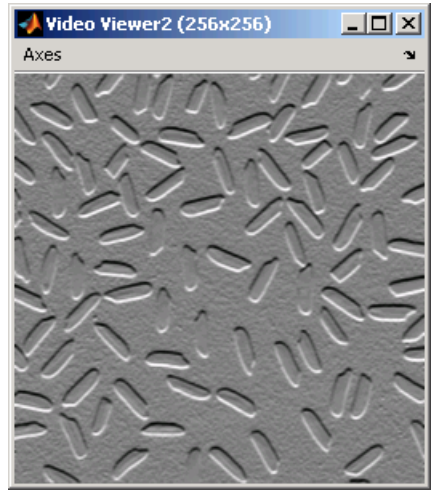
The Video Viewer window displays the edges of the rice grains in white and the background in black. To view the image at its true size, right-click the window and select **Set Display To True Size**.



The Video Viewer1 window displays the intensity image of the vertical gradient components of the image. You can see that the vertical edges of the rice grains are darker and more well defined than the horizontal edges. The image below is shown at its true size.



The Video Viewer2 window displays the intensity image of the horizontal gradient components of the image. In this image, the horizontal edges of the rice grains are more well defined. The image below is shown at its true size.



16 Double-click the Edge Detection block and clear the **Edge thinning** check box.

17 Run your model again.

Your model runs faster because the Edge Detection block is more efficient when you clear the **Edge thinning** check box. However, the edges of rice grains in the Video Viewer window are wider.



You have now used the Edge Detection block to find the object boundaries in an image. For more information on this block, see the Edge Detection block reference page.

Finding Lines in Images

Finding lines within images enables you to detect, measure, and recognize objects. In this section, you use the Hough Transform, Find Local Maxima, and Hough Lines blocks to find the longest line in an image.

1. Define an intensity image in the MATLAB workspace. At the MATLAB command prompt, type

   ```
   I= imread('circuit.tif');
   ```

   `I` is a 280-by-272 matrix of 8-bit unsigned integers.

2. To view the image, at the MATLAB command prompt, type

   ```
   imshow(I)
   ```

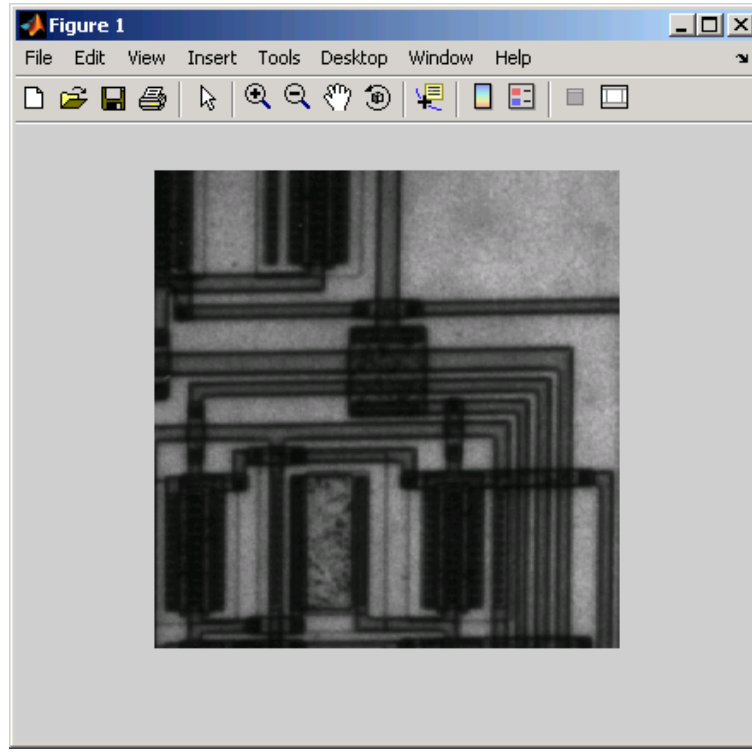


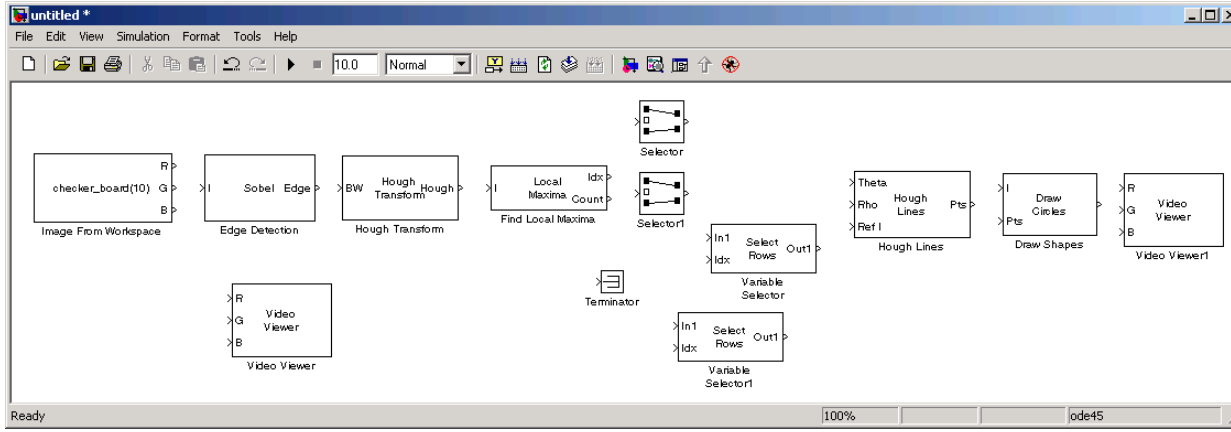
Image Courtesy of Steve Decker and Shujaat Nadeem

3 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
Edge Detection	Video and Image Processing Blockset / Analysis & Enhancement	1
Hough Transform	Video and Image Processing Blockset / Transforms	1

Block	Library	Quantity
Find Local Maxima	Video and Image Processing Blockset / Statistics	1
Selector	Simulink / Signal Routing	2
Variable Selector	Signal Processing Blockset / Signal Management / Indexing	2
Terminator	Simulink / Sinks	1
Hough Lines	Video and Image Processing Blockset / Transforms	1
Draw Shapes	Video and Image Processing Blockset / Text & Graphics	1
Video Viewer	Video and Image Processing Blockset / Sinks	2

4 Place the blocks so that your model resembles the following figure.



You are now ready to set your block parameters by double-clicking the blocks, modifying the block parameter values, and clicking **OK**.

5 Use the Image From Workspace block to import your image from the MATLAB workspace. Set the block parameters as follows:

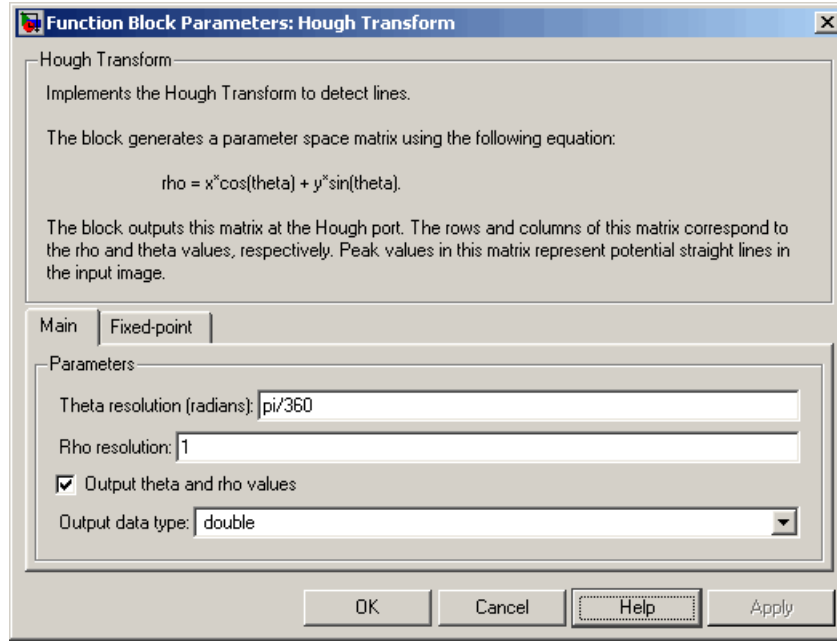
- **Main** pane, **Value** = `I`
- **Main** pane, **Output port labels** = `I`

6 Use the Edge Detection block to find the edges in the intensity image. This process improves the efficiency of the Hough Lines block as it reduces the image area over which the block searches for lines. The block also converts the image to a binary image, which is the required input for the Hough Transform block. Use the default parameters.

7 Use the Video Viewer block to display the edges found by the Edge Detection block. Set the **Input image type** parameter to `Intensity`.

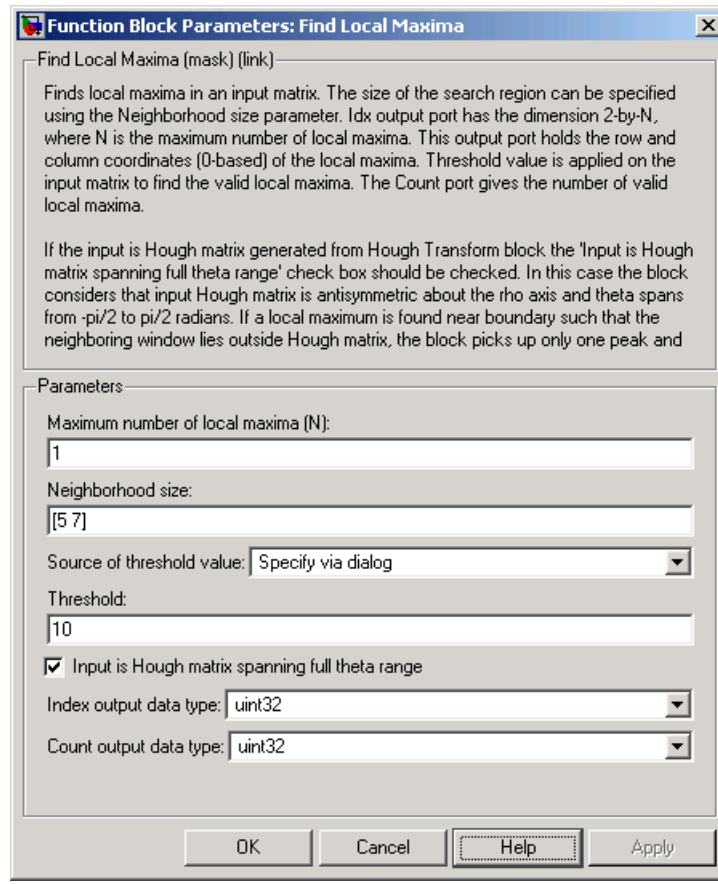
8 Use the Hough Transform block to compute the Hough matrix by transforming the input image into the rho-theta parameter space. The block also outputs the rho and theta values associated with the Hough matrix. Set the block parameters as follows:

- **Theta resolution (radians)** = `pi/360`
- Select the **Output theta and rho values** check box.



9 Use the Find Local Maxima block to find the location of the maximum value in the Hough matrix. Set the block parameters as follows:

- **Maximum number of local maxima (N)** = `1`
- Select the **Input is Hough matrix spanning full theta range** check box.



10 Use the Selector blocks to separate the indices of the rho and theta values, which are output at the Idx port, that are associated with the maximum value in the Hough matrix. Set the Selector block parameters as follows:

- **Index mode** = `Zero-based`

- **Elements (-1 for all elements)** = `0`
- **Input port width** = `2`

Set the Selector1 block parameters as follows:

- **Index mode** = `Zero-based`
- **Elements (-1 for all elements)** = `1`
- **Input port width** = `2`

11 Use the Variable Selector blocks to index into the rho and theta vectors and determine the rho and theta values that correspond to the longest line in the original image. Set the parameters of the Variable Selector blocks as follows:

- **Select** = `Columns`
- **Index mode** = `Zero-based`

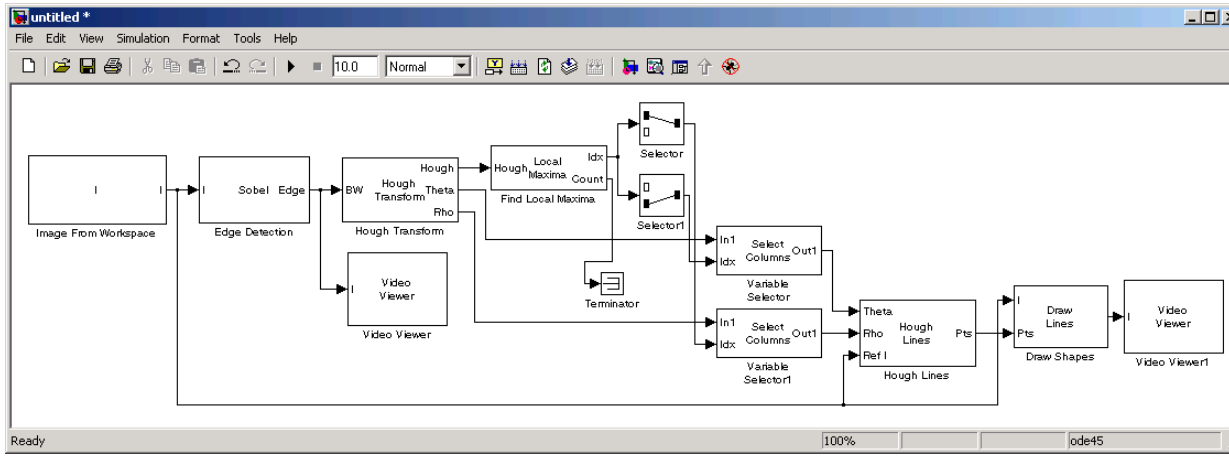
12 Use the Hough Lines block to determine where the longest line intersects the edges of the original image. You use these coordinates to superimpose a white line on the original image. Use the default parameters.

13 Use the Draw Shapes block to draw a white line over the longest line on the original image. Set the block parameters as follows:

- **Input image type** = `Intensity`
- **Shape** = `Lines`
- **Border intensity** = `White`

14 Use the Video Viewer block to display the original image with a white line superimposed over the longest line in the image. Set the **Input image type** parameter to `Intensity`.

15 Connect the blocks as shown in the figure below.

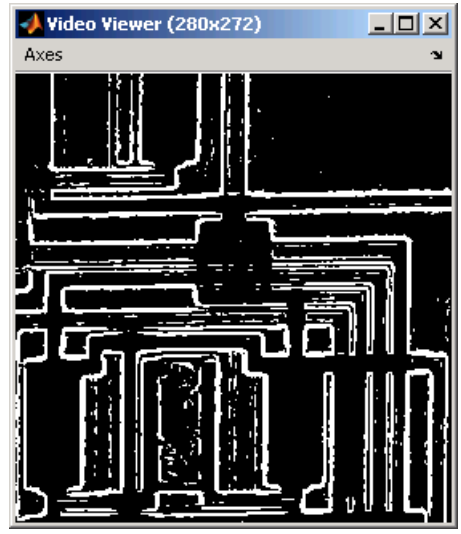


16 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

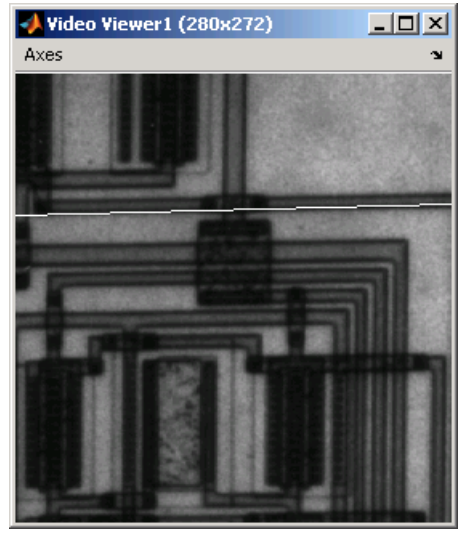
- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

17 Run your model.

The Video Viewer window displays the edges found in the original image in white and the background in black. To view the image at its true size, right-click the window and select **Set Display To True Size**.



The Video Viewer1 window displays the original image with a white line drawn over the longest line in the image.



You have now used the Hough Transform, Find Local Maxima, and Hough Lines blocks to find the longest line in an image. For more information on

these blocks, see the Hough Transform, Find Local Maxima, and Hough Lines block reference pages. For additional examples of the techniques used in this section, see the Lane detection and tracking and Rotation correction demos. You can open these demos by typing `vipdetectlane` and `viphough` at the MATLAB command prompt.

Measuring an Angle Between Lines

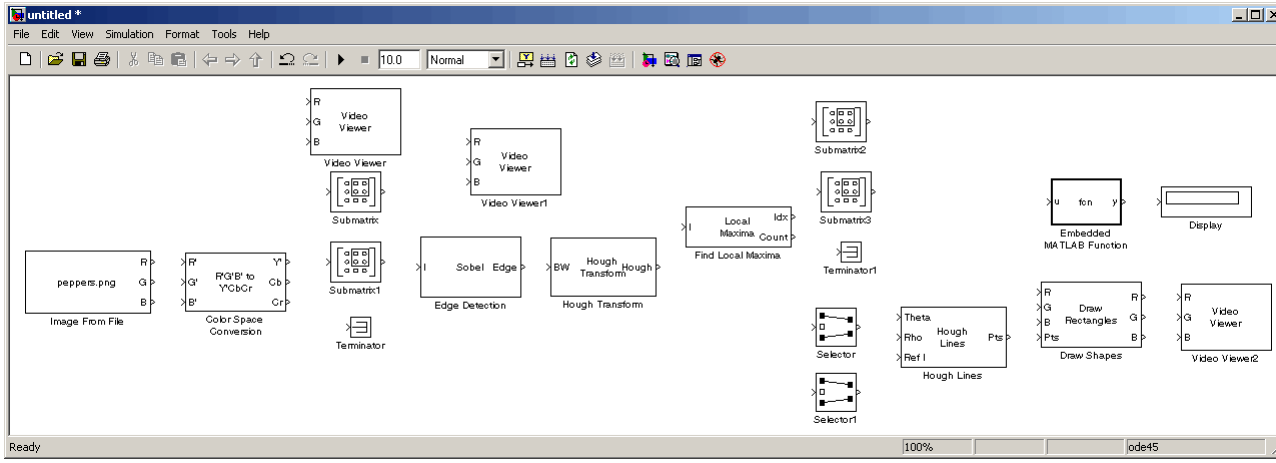
The Hough Transform, Find Local Maxima, and Hough Lines blocks enable you to find lines in images. With the Draw Shapes block, you can annotate images. In the following example, you use these capabilities to draw lines on the edges of two beams and measure the angle between them.

1 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From File	Video and Image Processing Blockset / Sources	1
Color Space Conversion	Video and Image Processing Blockset / Conversions	1
Submatrix	Signal Processing Blockset / Math Functions / Matrices and Linear Algebra / Matrix Operations	4
Terminator	Simulink / Sinks	1
Edge Detection	Video and Image Processing Blockset / Analysis & Enhancement	1
Hough Transform	Video and Image Processing Blockset / Transforms	1
Find Local Maxima	Video and Image Processing Blockset / Statistics	1
Selector	Simulink / Signal Routing	2
Hough Lines	Video and Image Processing Blockset / Transforms	1

Block	Library	Quantity
Embedded MATLAB Function	Simulink / User-Defined Functions	1
Draw Shapes	Video and Image Processing Blockset / Text & Graphics	1
Display	Simulink / Sinks	1
Video Viewer	Video and Image Processing Blockset / Sinks	2

2 Position the blocks as shown in the following figure.



3 Use the Image From File block to import an image into the Simulink model. Set the parameters as follows:

- **File name** = `gantrycrane.png`
- **Sample time** = `1`

4 Use the Color Space Conversion block to convert the RGB image into the Y'CbCr color space. You perform this conversion to separate the luma information from the color information. Use the default parameters.

Note In this example, you segment the image using a thresholding operation that performs best on the Cb channel of the Y'CbCr color space.

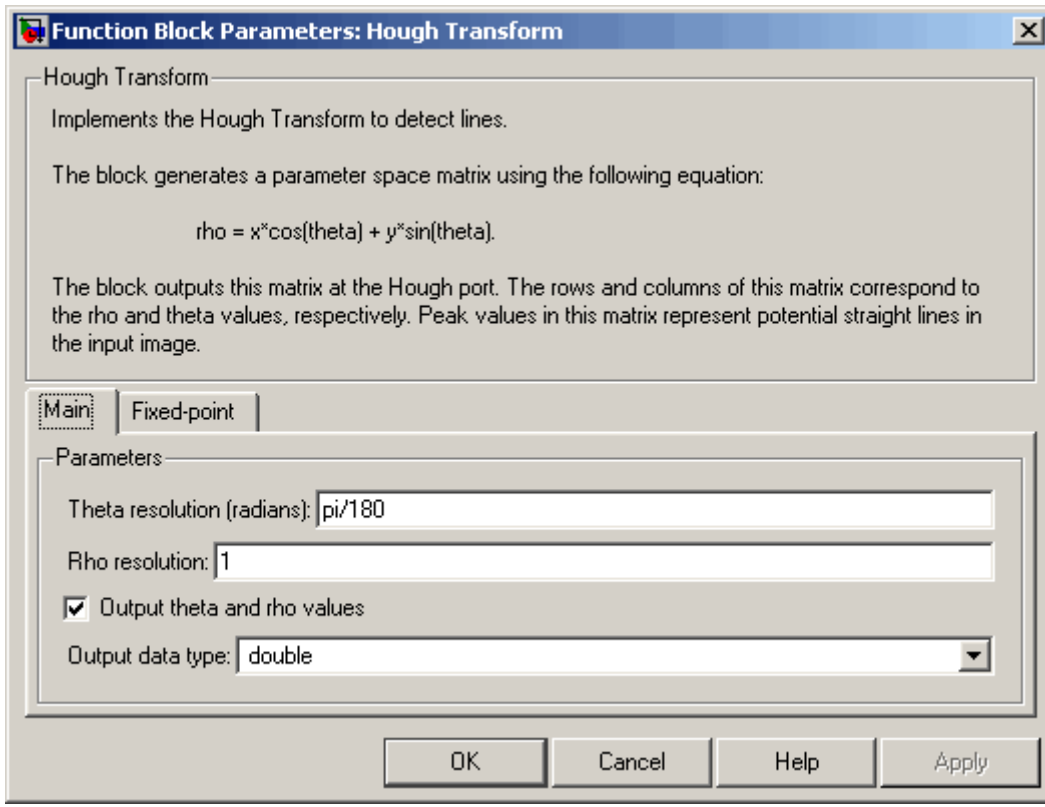
5 Use the Submatrix and Submatrix1 blocks to crop the Y' and Cb matrices to a particular region of interest (ROI). This ROI contains two beams that are at an angle to each other. Set the parameters as follows:

- **Starting row** = `Index`
- **Starting row index** = `66`
- **Ending row** = `Index`
- **Ending row index** = `150`
- **Starting column** = `Index`
- **Starting column index** = `325`
- **Ending column** = `Index`
- **Ending column index** = `400`

6 Use the Edge Detection block to find the edges in the Cb portion of the image. This block outputs a binary image. Set the **Threshold scale factor** parameter to `1`.

7 Use the Hough Transform block to calculate the Hough matrix, which gives you an indication of the presence of lines in an image. Select the **Output theta and rho values** check box as shown in the following figure.

Note In step 11, you find the theta and rho values that correspond to the peaks in the Hough matrix.



8 Use the Find Local Maxima block to find the peak values in the Hough matrix. These values represent potential lines in the input image. Set the parameters as follows:

- **Neighborhood size** = `[11 11]`
- **Input is Hough matrix spanning full theta range** = selected

Because you are expecting two lines, leave the **Maximum number of local maxima (N)** parameter set to 2, and connect the Count port to the Terminator block.

9 Use the Submatrix2 block to find the indices that correspond to the theta values of the two peak values in the Hough matrix. Set the parameters as follows:

- **Starting row** = `Index`
- **Starting row index** = `2`
- **Ending row** = `Index`
- **Ending row index** = `2`

The Idx port of the Find Local Maxima block outputs a matrix whose second row represents the zero-based indices of the theta values that correspond to the peaks in the Hough matrix. Now that you have these indices, you can use a Selector block to extract the corresponding theta values from the vector output of the Hough Transform block.

10 Use the Submatrix3 block to find the indices that correspond to the rho values of the two peak values in the Hough matrix. Set the parameters as follows:

- **Ending row** = `Index`
- **Ending row index** = `1`

The Idx port of the Find Local Maxima block outputs a matrix whose first row represents the zero-based indices of the rho values that correspond to the peaks in the Hough matrix. Now that you have these indices, you can use a Selector block to extract the corresponding rho values from the vector output of the Hough Transform block.

11 Use the Selector blocks to find the theta and rho values that correspond to the peaks in the Hough matrix. These values, output by the Hough Transform block, are located at the indices output by the Submatrix2 and Submatrix3 blocks. Set both block parameters as follows:

- **Index mode** = `Zero-based`
- **Source of element indices (E)** = `External`
- **Input port width** = `-1`

You set the **Index mode** to `Zero-based` because the Find Local Maxima block outputs zero-based indices at the Idx port.

12 Use the Hough Lines block to find the Cartesian coordinates of lines that are described by rho and theta pairs. Set the **Sine value computation method** parameter to `Trigonometric function`.

13 Use the Draw Shapes block to draw the lines on the luminance portion of the ROI. Set the parameters as follows:

- **Input image type** = `Intensity`
- **Shape** = `Lines`
- **Border intensity** = `White`

14 Use the Embedded MATLAB Function block to calculate the angle between the two lines. Copy and paste the following code into the block:

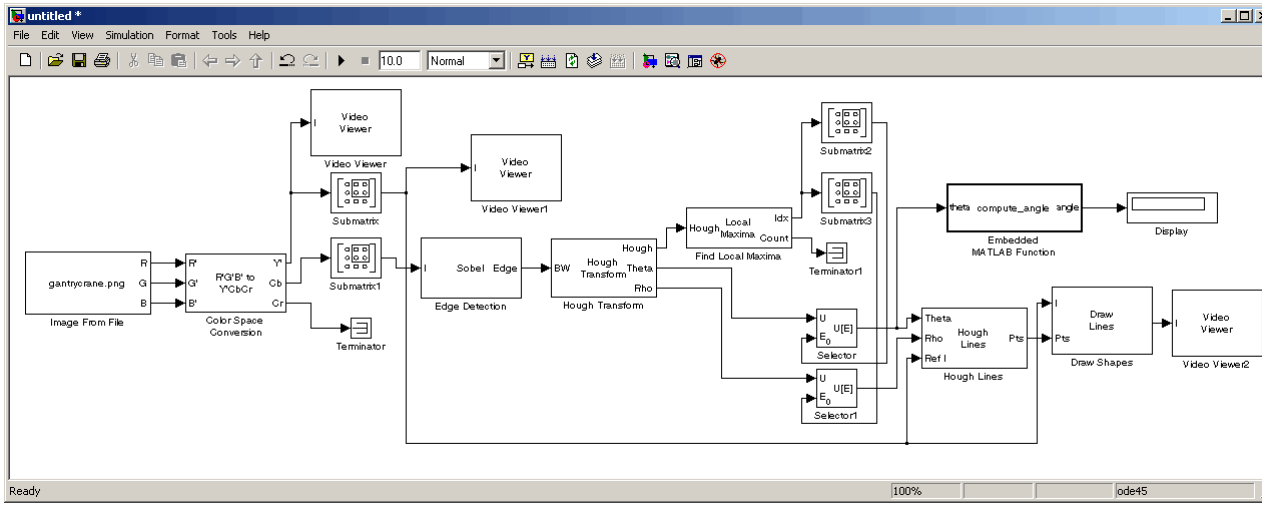
```
function angle = compute_angle(theta)

%Compute the angle value in degrees
angle = abs(theta(1)-theta(2))*180/pi;
%Always return an angle value less than 90 degrees
if (angle>90)
    angle = 180-angle;
end
```

15 Use the Display block to view the angle between the two lines. Use the default parameters.

16 Use the Video Viewer blocks to view the original intensity image, the ROI, and the annotated ROI. Set the **Input image type** parameters to `Intensity`.

17 Connect the blocks as shown in the following figure.

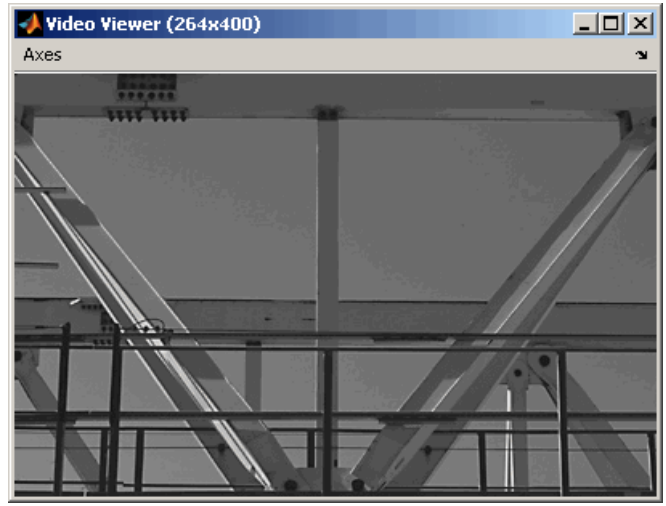


18 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

19 Run the model.

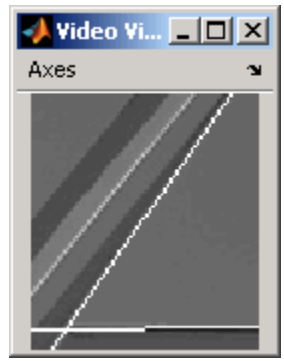
The Video Viewer window displays the original intensity image.



The Video Viewer1 window displays the ROI where two beams intersect.



The Video Viewer2 window displays the ROI that has been annotated with two white lines.



The Display block shows a value of 54, which is the angle in degrees between the two lines on the annotated ROI.

You have now annotated an image with two lines and measured the angle between them. For additional information, see the Hough Transform, Find Local Maxima, Hough Lines, and Draw Shapes block reference pages.

Image Enhancement

Image enhancement techniques improve images. You can use them to remove noise from images, increase the signal-to-noise ratio, make certain features easier to see by modifying the colors or intensities, or sharpen the image.

This section includes the following topics:

- "Sharpening and Blurring an Image" on page 7-26 — Use the 2-D FIR Filter block to improve the clarity of an image
- "Removing Salt and Pepper Noise from Images" on page 7-33 — Use the Median Filter block to eliminate noise from an intensity image
- "Removing Periodic Noise from Video" on page 7-39 — Use the 2-D FIR Filter block to eliminate noise in a video stream
- "Adjusting the Contrast in Intensity Images" on page 7-46 — Use the Contrast Adjustment and Histogram Equalization blocks to modify the contrast of intensity images
- "Adjusting the Contrast in Color Images" on page 7-52 — Use the Histogram Equalization block to modify the contrast of a color image

Sharpening and Blurring an Image

To sharpen a color image, you need to make the luma intensity transitions more acute, while preserving the color information of the image. To do this, you convert an R'G'B' image into the Y'CbCr color space and apply a highpass filter to the luma portion of the image only. Then, you transform the image back to the R'G'B' color space to view the results. To blur an image, you apply a lowpass filter to the luma portion of the image. This example illustrates these two processes. Note that the prime notation indicates that the signals are gamma corrected.

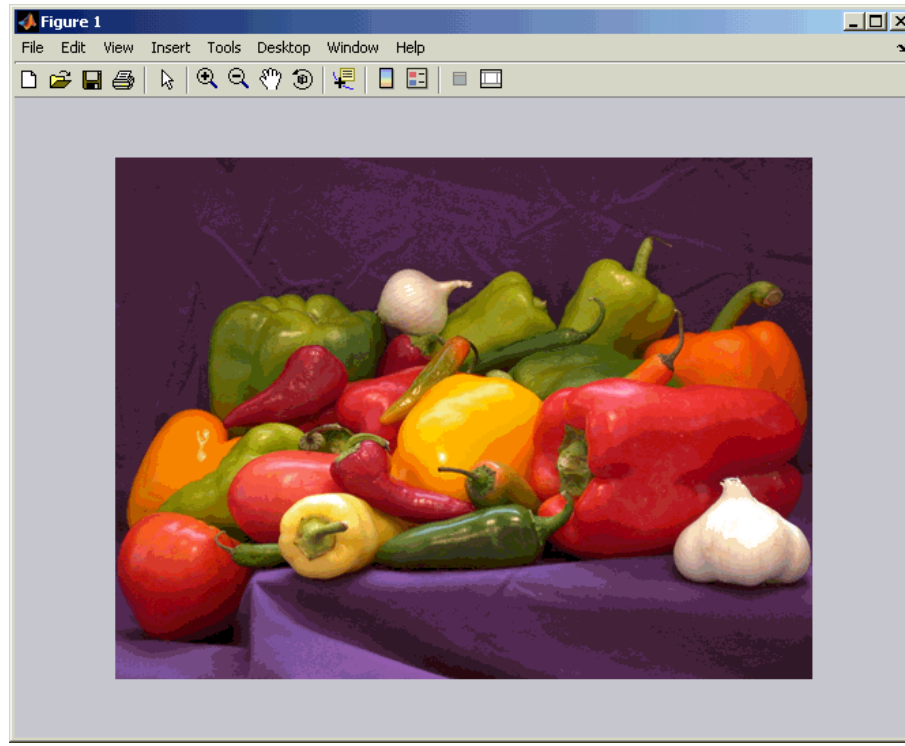
1. Define an R'G'B' image in the MATLAB workspace. To read in an R'G'B' image from a PNG file and cast it to the double-precision data type, at the MATLAB command prompt, type

```
I= im2double(imread('peppers.png'));
```

`I` is a 384-by-512-by-3 array of double-precision floating-point values. Each plane of this array represents the red, green, or blue color values of the image.

2 To view the image this array represents, at the MATLAB command prompt, type

```
imshow(I)
```

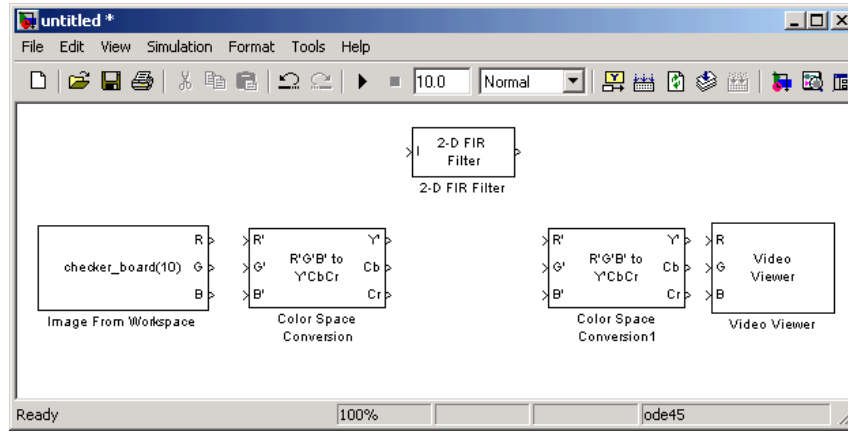


Now that you have defined your image, you can create your model.

3 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
Color Space Conversion	Video and Image Processing Blockset / Conversions	2
2-D FIR Filter	Video and Image Processing Blockset / Filtering	1
Video Viewer	Video and Image Processing Blockset / Sinks	1

4 Position the blocks as shown in the following figure.

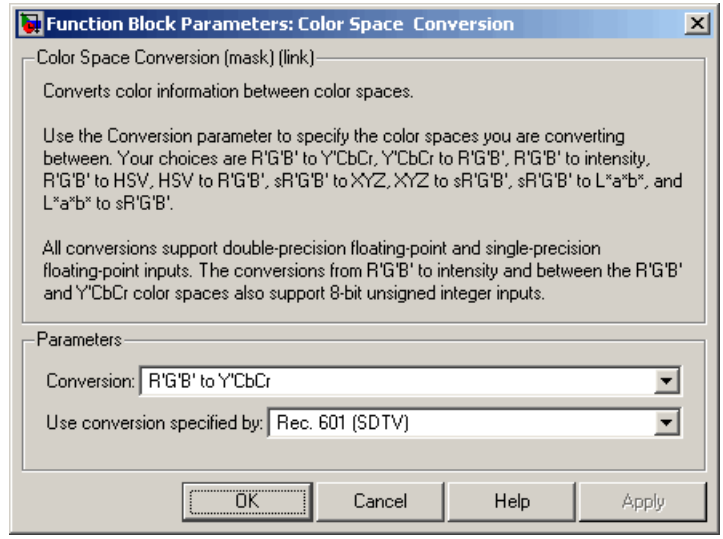


5 Use the Image From Workspace block to import the R'G'B' image from the MATLAB workspace. Set the parameters as follows:

- **Main** pane, **Value** = `I`
- **Main** pane, **Output port labels** = `R'|G'|B'`

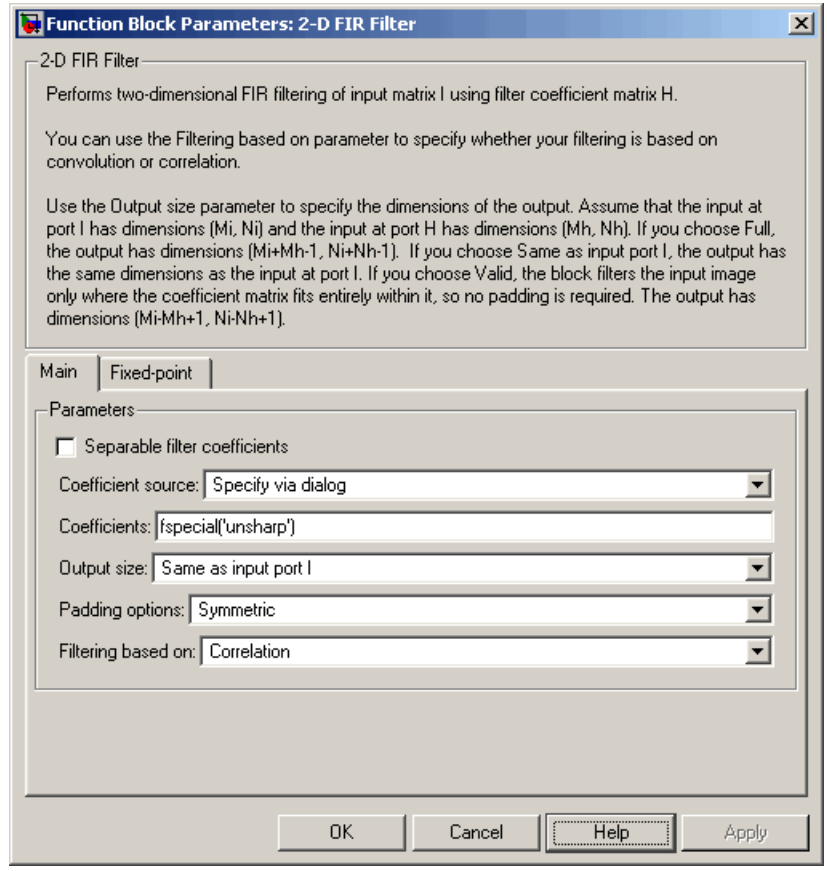
The block outputs the R', G', and B' planes of the `I` array at the output ports.

6 The first Color Space Conversion block converts color information from the R'G'B' color space to the Y'CbCr color space. Use the default parameters.



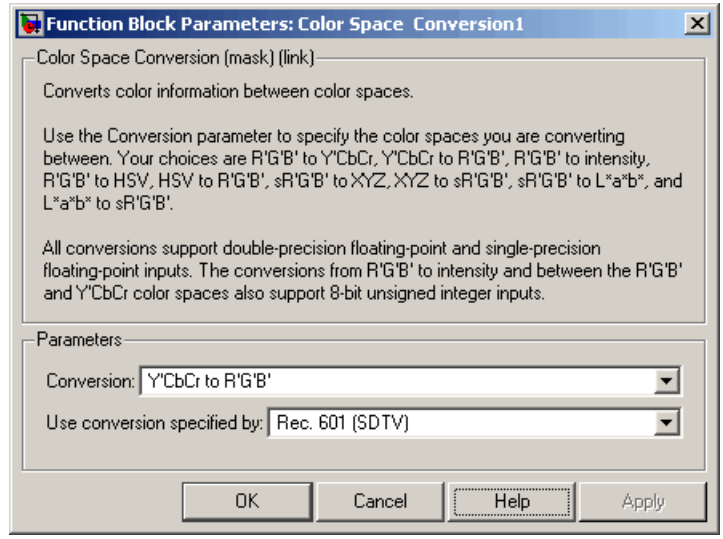
7 Use the 2-D FIR Filter block to filter the luma portion of the image. Set the block parameters as follows:

- **Coefficients** = `fspecial('unsharp')`
- **Output size** = `Same as input port I`
- **Padding options** = `Symmetric`
- **Filtering based on** = `Correlation`



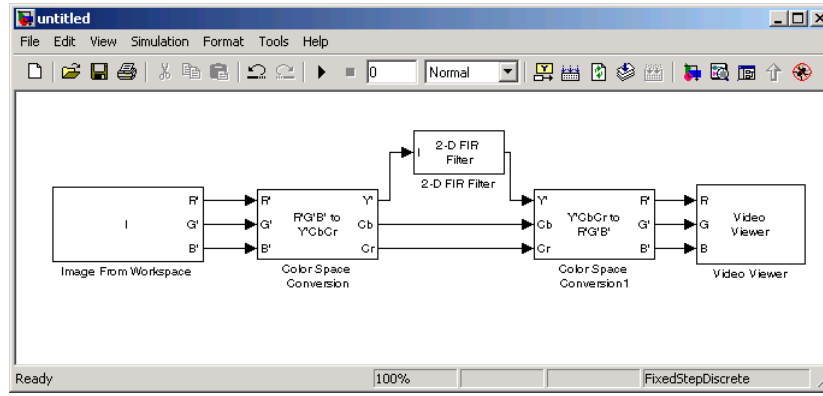
The `fspecial('unsharp')` command creates two-dimensional highpass filter coefficients suitable for correlation. This highpass filter sharpens the image by removing the low frequency noise in it.

8 Use the second Color Space Conversion block to converts the color information from the Y'CbCr color space to the R'G'B' color space. Set the **Conversion** parameter to `Y'CbCr to R'G'B'`.



9 Use the Video Viewer block to automatically display the new, sharper image in the Video Viewer window when you run the model. Use the default parameters.

10 Connect the blocks as shown in the figure below.

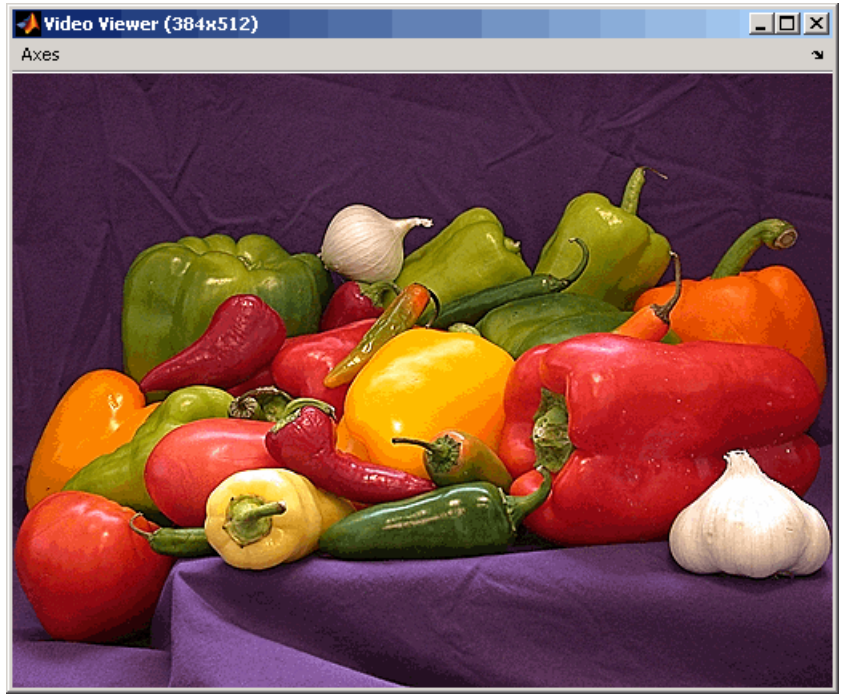


11 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

12 Run the model.

A sharper version of the original image appears in the Video Viewer window. To view the image at its true size, right-click the window and select **Set Display To True Size**.

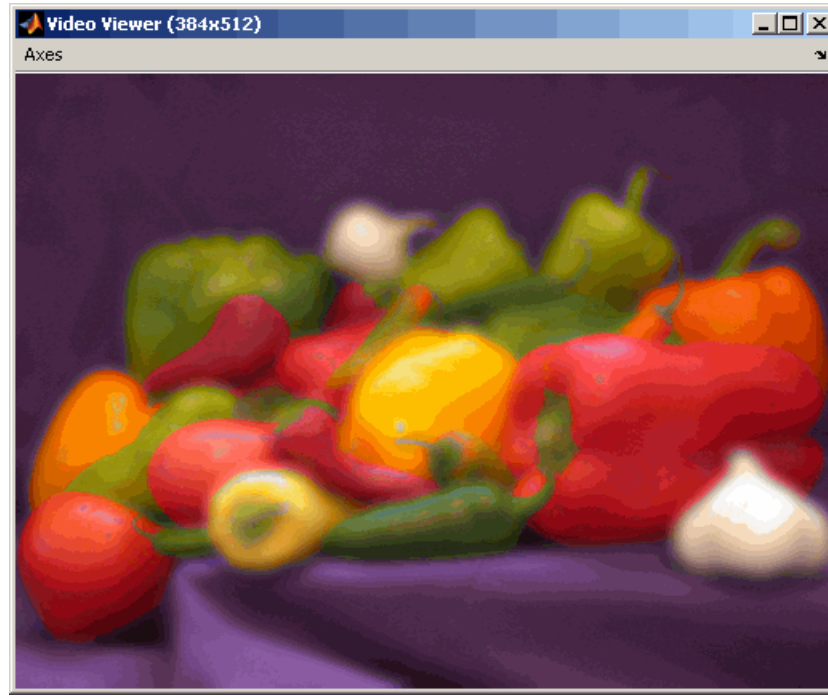


13 To blur the image, double-click the 2-D FIR Filter block. Set **Coefficients** parameter to `fspecial('gaussian',[15 15],7)` and then click **OK**.

The `fspecial('gaussian',[15 15],7)` command creates two-dimensional Gaussian lowpass filter coefficients. This lowpass filter blurs the image by removing the high frequency noise in it.

14 Run the model.

A blurred version of the original image appears in the Video Viewer window. The image below is shown at its true size.



In this example, you used the Color Space Conversion and 2-D FIR Filter blocks to sharpen and blur an image. For more information on these blocks, see the Color Space Conversion and 2-D FIR Filter block reference pages. For more information on the `fspecial` function, see the Image Processing Toolbox documentation.

Removing Salt and Pepper Noise from Images

Median filtering is a common image enhancement technique for removing salt and pepper noise. Because this filtering is less sensitive than linear techniques to extreme changes in pixel values, it can remove salt and pepper noise without significantly reducing the sharpness of an image. In this topic,

you use the Median Filter block to remove salt and pepper noise from an intensity image:

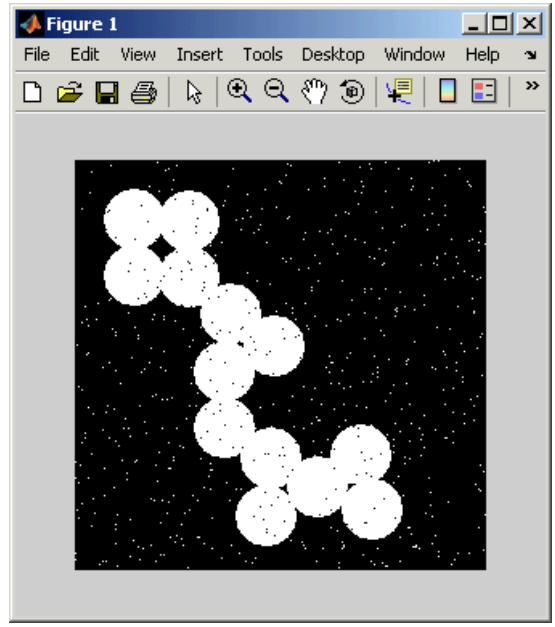
1. Define an intensity image in the MATLAB workspace and add noise to it by typing the following at the MATLAB command prompt:

   ```
   I= double(imread('circles.png'));
   I= imnoise(I,'salt & pepper',0.02);
   ```

   `I` is a 256-by-256 matrix of 8-bit unsigned integer values.

2. To view the image this matrix represents, at the MATLAB command prompt, type

   ```
   imshow(I)
   ```

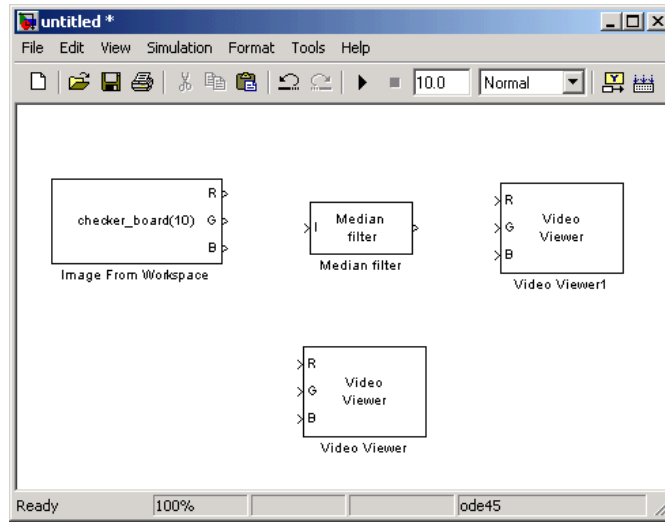


   The intensity image contains noise that you want your model to eliminate.

3. Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
Median Filter	Video and Image Processing Blockset / Filtering	1
Video Viewer	Video and Image Processing Blockset / Sinks	2

4 Place the blocks as shown in the figure below.

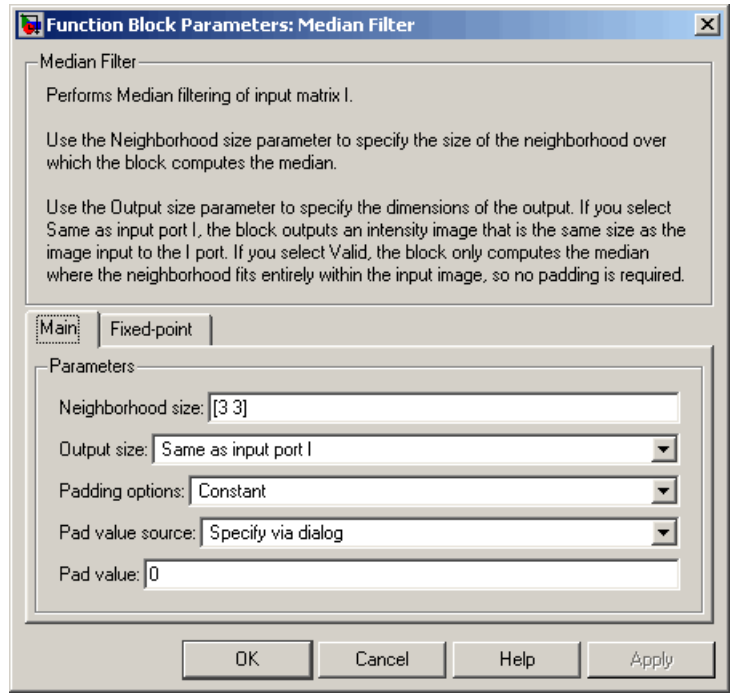


Now that you have assembled the blocks required to remove the noise in your image, you need to set your block parameters. To do this, double-click the blocks, modify the block parameter values, and click **OK**.

5 Use the Image From Workspace block to import the noisy image into your model. Set the block parameters as follows:

- **Main** pane, **Value** = `I`
- **Main** pane, **Output port labels** = `Image`

6 Use the Median Filter block to eliminate the black and white speckles in the image. Use the default parameters.



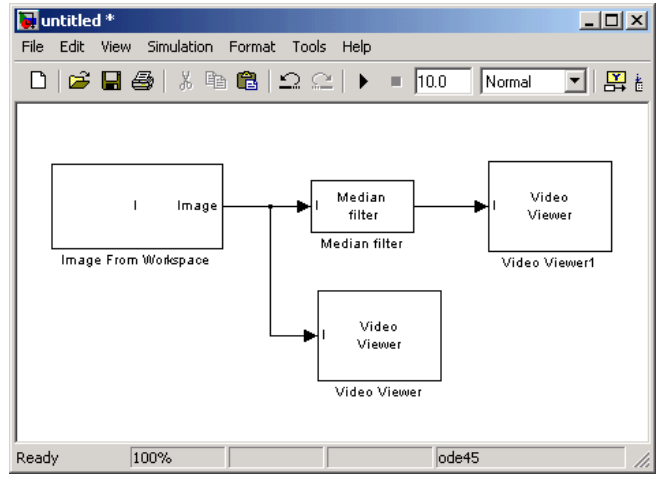
The Median Filter block replaces the central value of the 3-by-3 neighborhood with the median value of the neighborhood. This process removes the noise in the image.

7 Use the Video Viewer block to display the original, noisy image. Set the **Input image type** parameter to `Intensity`.

The Video Viewer block automatically displays the original, noisy image in the Video Viewer1 window when you run the model. Because the image is represented by 8-bit unsigned integers, a value of 0 corresponds to black and a value of 255 corresponds to white.

8 Use the Video Viewer1 block to display the modified image. Set the **Input image type** parameter to `Intensity`.

9 Connect the blocks as shown in the figure below.

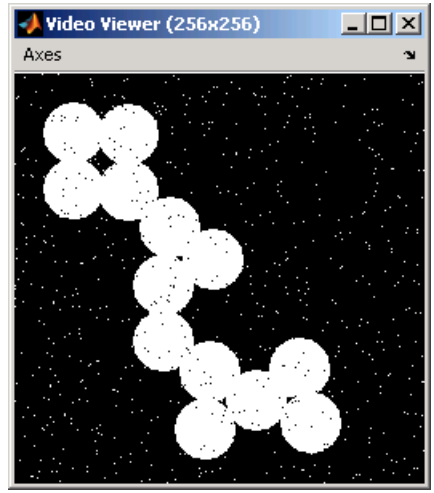


10 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

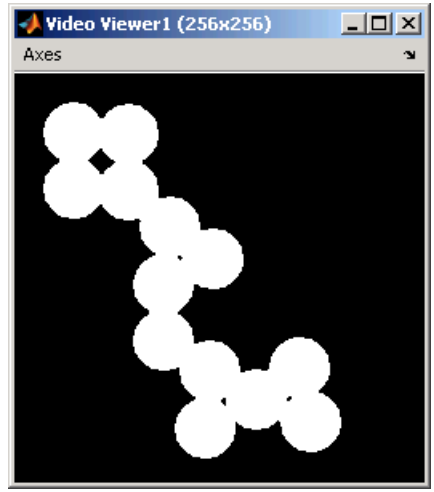
- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

11 Run the model.

The original noisy image appears in the Video Viewer window. To view the image at its true size, right-click the window and select **Set Display To True Size**.



The cleaner image appears in the Video Viewer1 window. The image below is shown at its true size.



You have used the Median Filter block to remove noise from your image. For more information about this block, see the Median Filter block reference page.

Removing Periodic Noise from Video

Periodic noise can be introduced into a video stream during acquisition or transmission due to electrical or electromechanical interference. In this example, you remove periodic noise from an intensity video using the 2-D FIR Filter block. You can use this technique to remove noise from other images or video streams, but you might need to modify the filter coefficients to account for the noise frequency content present in your signal:

1. Create a new Simulink model, and add to it the blocks shown in the following table.

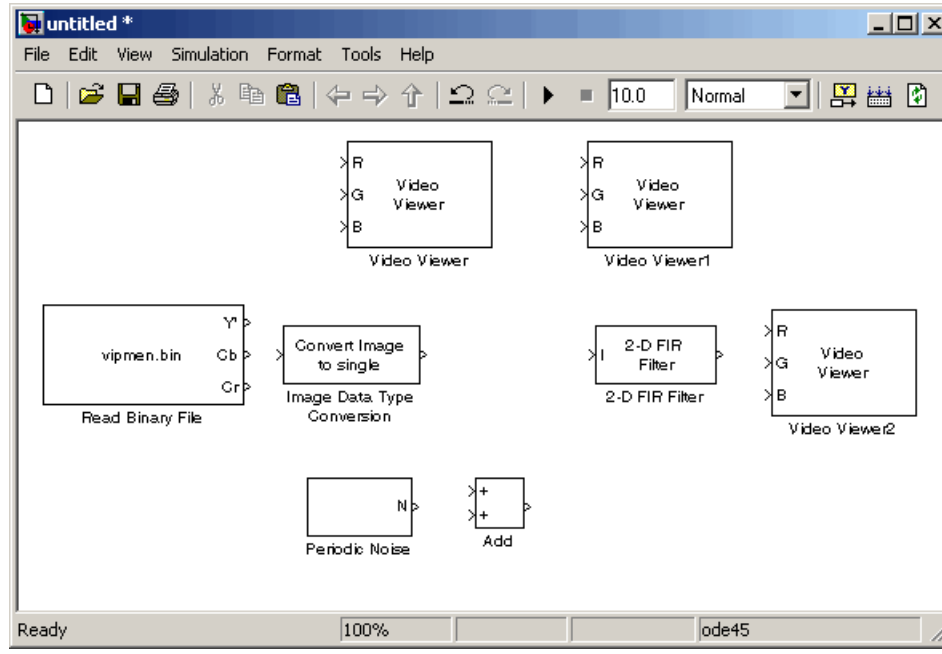
Block	Library	Quantity
Read Binary File	Video and Image Processing Blockset / Sources	1
Image Data Type Conversion	Video and Image Processing Blockset / Conversions	1
2-D FIR Filter	Video and Image Processing Blockset / Filtering	1
Video Viewer	Video and Image Processing Blockset / Sinks	3
Add	Simulink / Math Operations	1

2. Open the Periodic noise reduction demo by typing `vipstripes` at the MATLAB command prompt.

3. Click-and-drag the Periodic Noise block into your model.

   The block outputs a sinusoid with a normalized frequency that ranges between 0.61π and 0.69π radians per sample and a phase that varies between 0 and 3 radians. You are using this sinusoid to represent periodic noise.

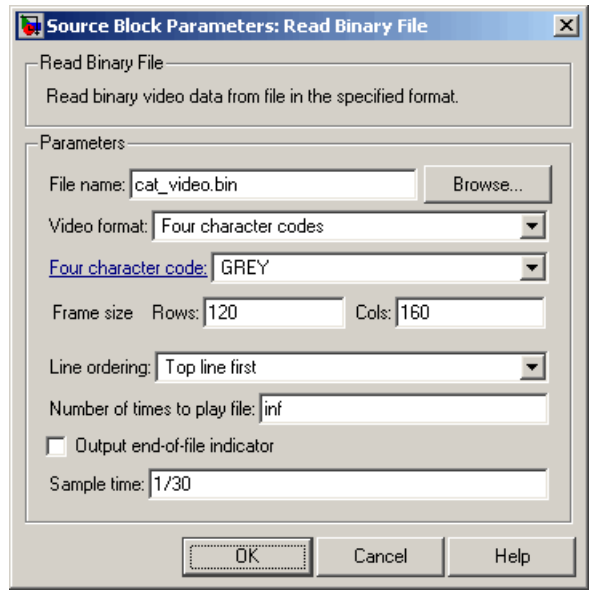
4. Place the blocks so that your model resembles the following figure. The unconnected ports disappear when you set block parameters.



You are now ready to set your block parameters by double-clicking the blocks, modifying the block parameter values, and clicking **OK**.

5 Use the Read Binary File block to import a binary file into the model. Set the block parameters as follows:

- **File name** = `cat_video.bin`
- **Four character code** = `GREY`
- **Number of times to play file** = `inf`
- **Sample time** = `1/30`



6 Use the Image Data Type Conversion block to convert the data type of the video to single-precision floating point. Use the default parameter.

7 Use the Video Viewer block to view the original video. Set the **Input image type** parameter to `Intensity`.

8 Use the Add block to add the noise video to the original video. Use the default parameters.

9 Use the Video Viewer1 block to view the noisy video. Set the **Input image type** parameter to `Intensity`.

10 Define the filter coefficients in the MATLAB workspace. Type the following code at the MATLAB command prompt:

```
vipdh_stripes
```

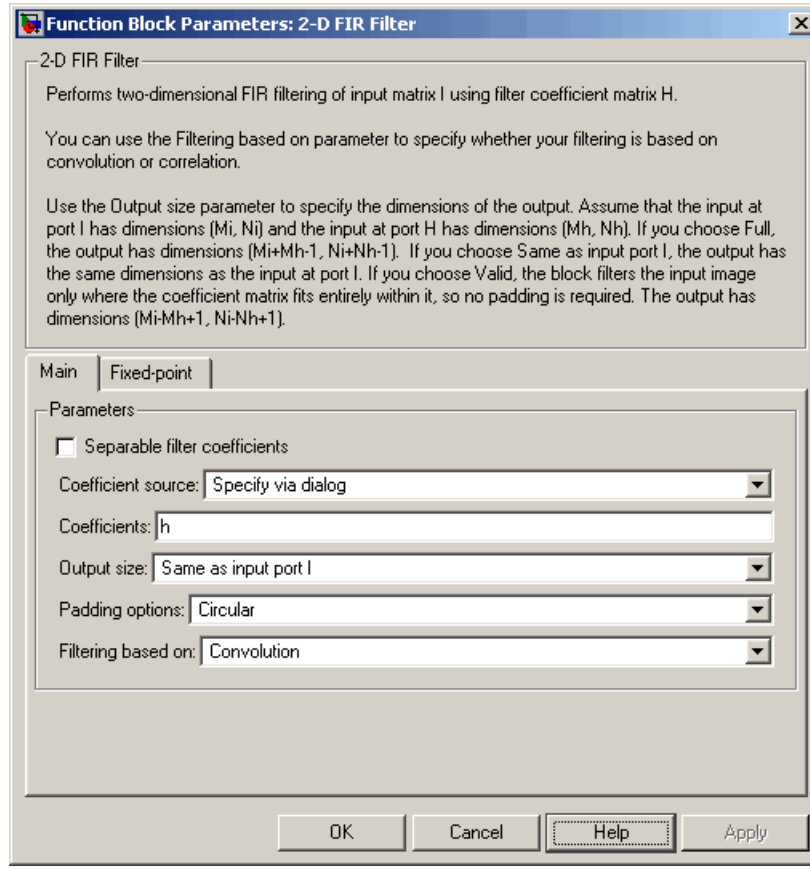
The variable `h`, as well as several others, are loaded into the MATLAB workspace. The variable `h` represents the coefficients of the band reject filter capable of removing normalized frequencies between 0.61π and 0.69π

radians per sample. The coefficients were created using the Filter Design and Analysis Tool (FDATool) and the `ftrans2` function.

11 Use the 2-D FIR Filter block to model a band-reject filter capable of removing the periodic noise from the video. Set the block parameters as follows:

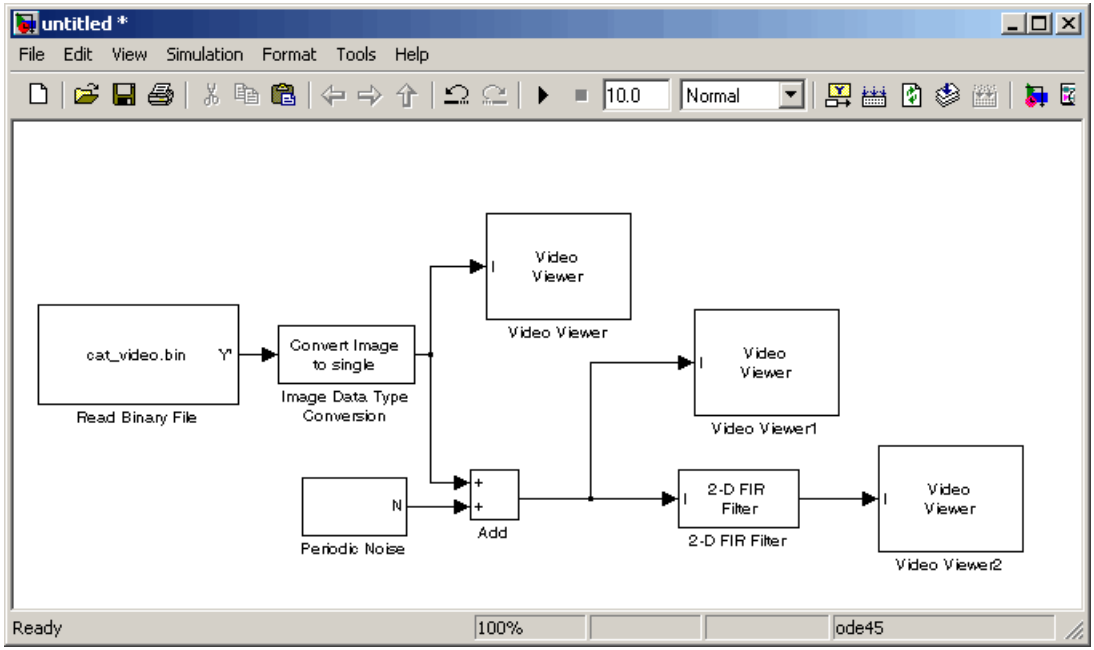
- **Coefficients** = `h`
- **Output size** = `Same as input port I`
- **Padding options** = `Circular`

Choose a type of padding that minimizes the effect of the pixels outside the image on the processing of the image. In this example, circular padding produces the best results because it is most effective at replicating the sinusoidal noise outside the image.



12 Use the Video Viewer2 block to view the approximation of the original video. Set the **Input image type** parameter to `Intensity`.

13 Connect the block as shown in the figure below.

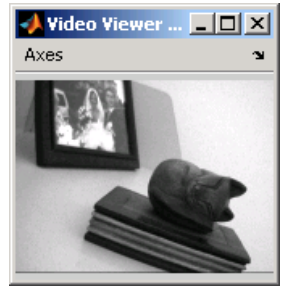


14 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

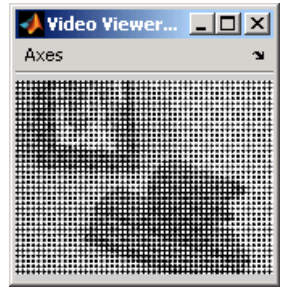
- **Solver** pane, **Stop time** = `inf`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

15 Run the model.

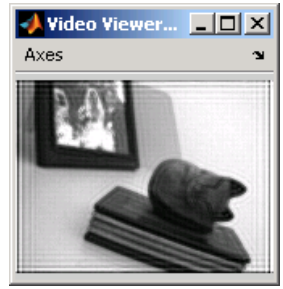
The original video appears in the Video Viewer window. To view the video at its true size, right-click the window and select **Set Display To True Size**.



The noisy video appears in the Video Viewer1 window. The video below is shown at its true size.



The approximation of the original video appears in the Video Viewer2 window. Note that artifacts of the processing appear near the edges of the video. The video below is shown at its true size.



You have used the Read Binary File block to import a binary video into your model, the 2-D FIR Filter to remove periodic noise from this video, and the Video Viewer block to display the results. For more information about these blocks, see the Read Binary File, 2-D FIR Filter, and Video Viewer block

reference pages. For more information about the Filter Design and Analysis Tool (FDATool), see the Signal Processing Toolbox documentation. For information about the `ftrans2` function, see the Image Processing Toolbox documentation.

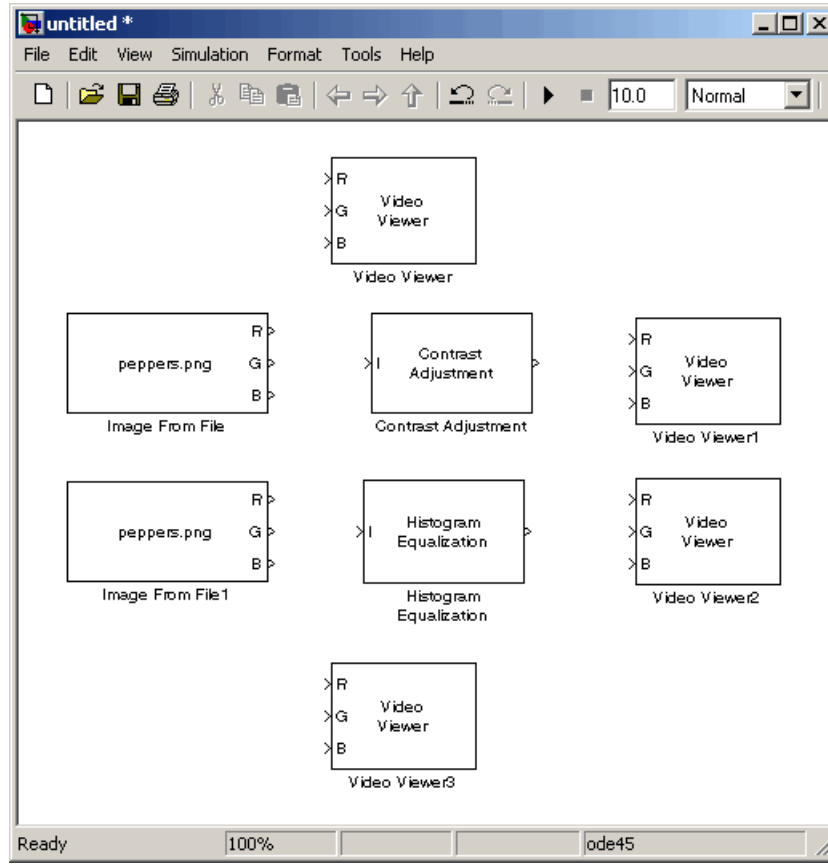
Adjusting the Contrast in Intensity Images

This example shows you how to modify the contrast in two intensity images using the Contrast Adjustment and Histogram Equalization blocks.

1 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From File	Video and Image Processing Blockset / Sources	2
Contrast Adjustment	Video and Image Processing Blockset / Analysis & Enhancement	1
Histogram Equalization	Video and Image Processing Blockset / Analysis & Enhancement	1
Video Viewer	Video and Image Processing Blockset / Sinks	4

2 Place the blocks so that your model resembles the following figure.



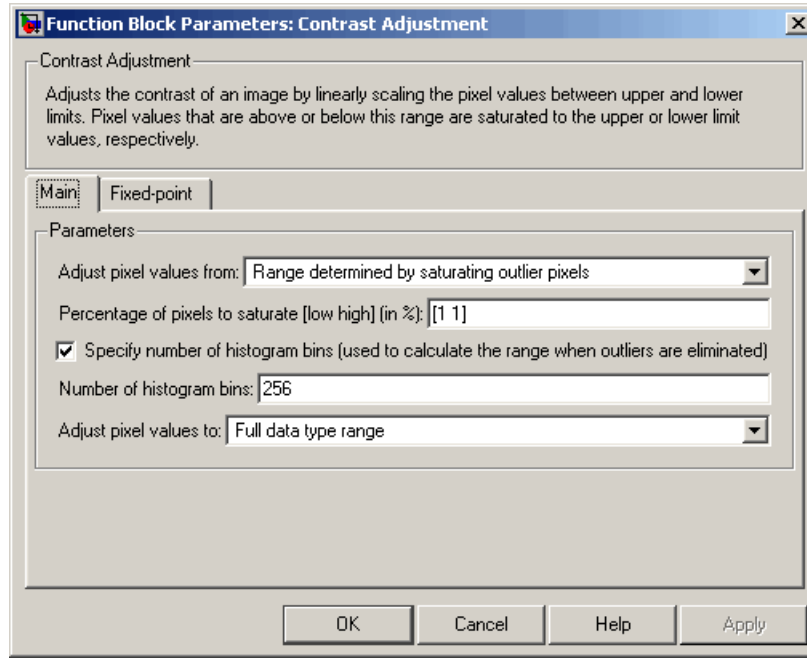
3 Use the Image From File block to import the first image into the Simulink model. Set the block parameters as follows:

- **File name** = `pout.tif`
- **Output port label** = I

4 Use the Image From File1 block to import the second image into the Simulink model. Set the block parameters as follows:

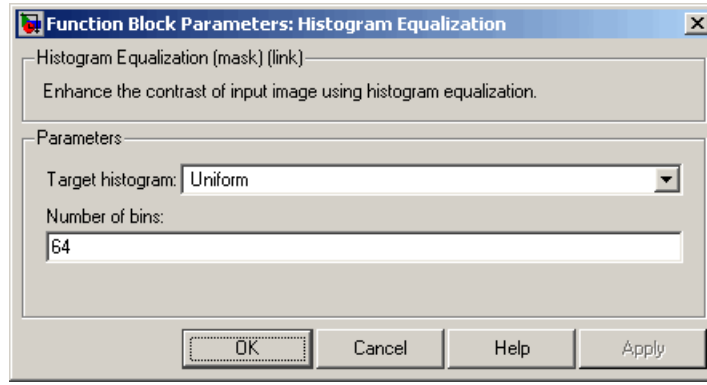
- **File name** = `tire.tif`
- **Output port label** = I

5 Use the Contrast Adjustment block to modify the contrast in `pout.tif`. Set the **Adjust pixel values from** parameter to `Range determined by saturating outlier pixels`, as shown in the following figure.



This block adjusts the contrast of the image by linearly scaling the pixel values between user-specified upper and lower limits.

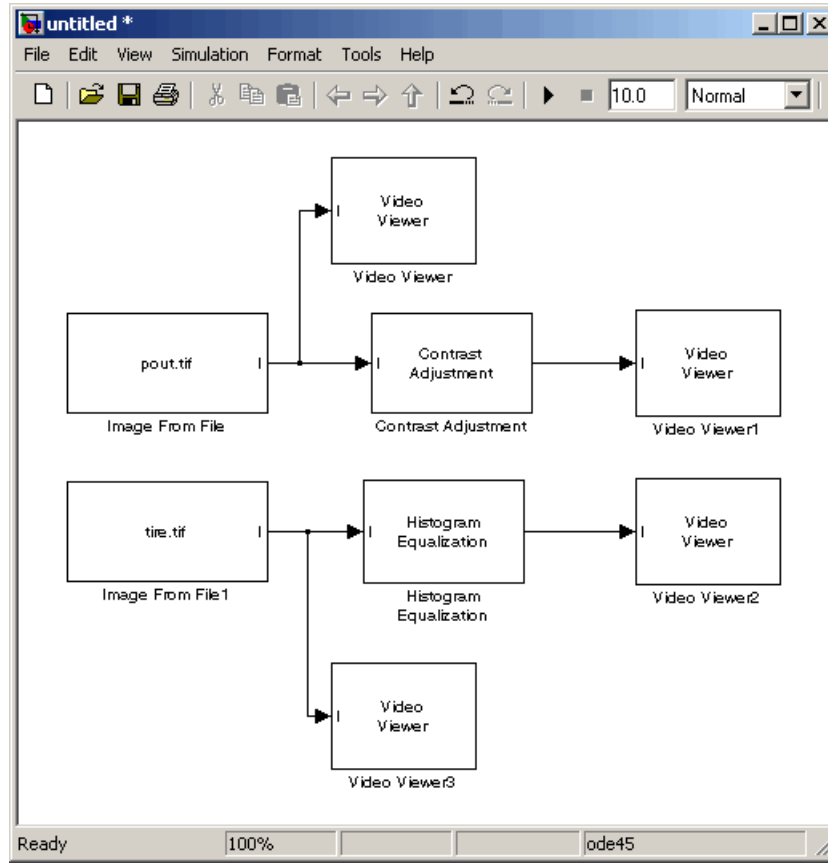
6 Use the Histogram Equalization block to modify the contrast in `tire.tif`. Use the default parameters.



This 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.

7 Use the Video Viewer blocks to view the original and modified images. For each block, set the **Input image type** parameter to `Intensity`.

8 Connect the blocks as shown in the following figure.

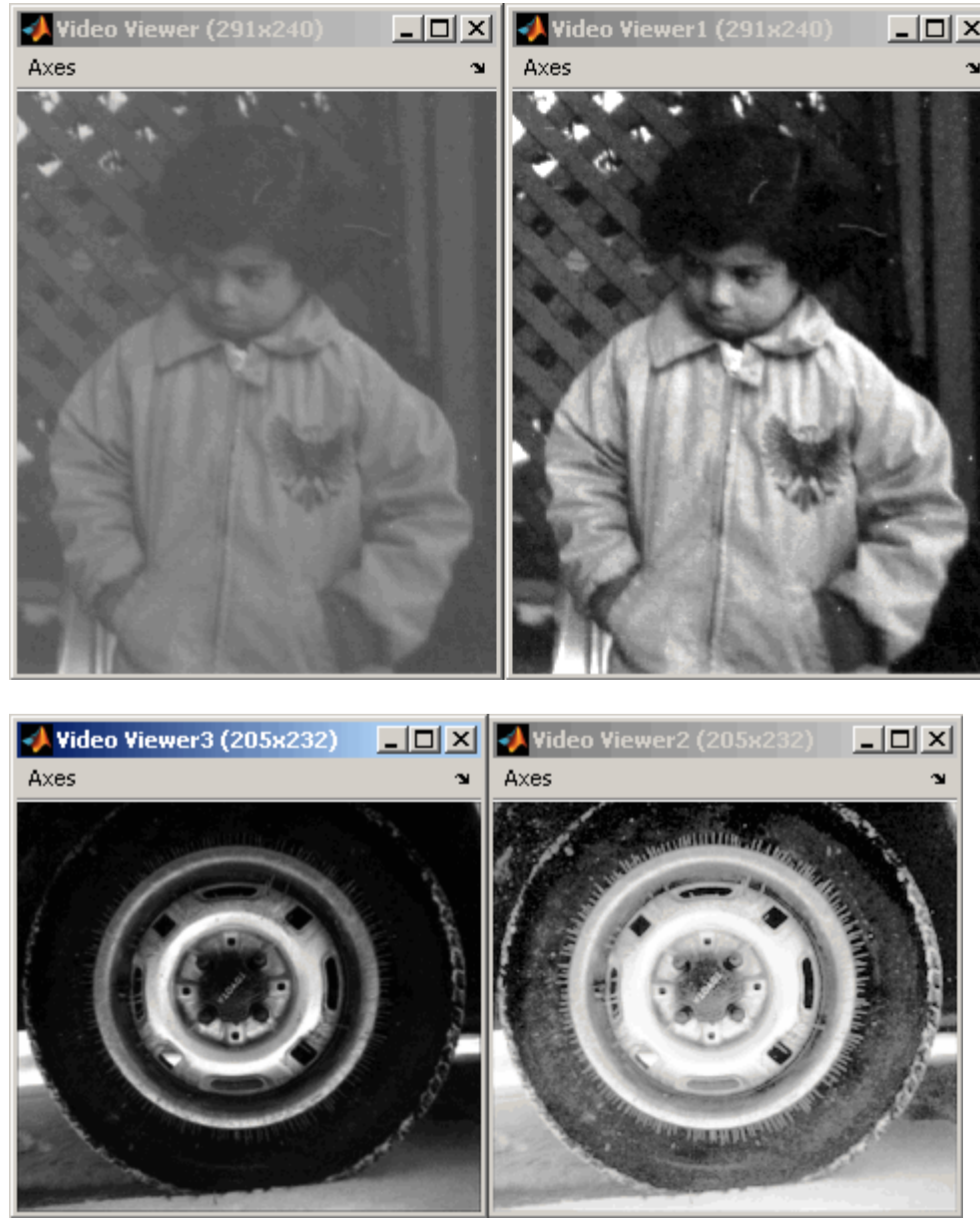


9 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

10 Run the model.

The results appear in the Video Viewer windows.


Video Viewer (291x240)
Axes
Video Viewer1 (291x240)
Axes
Video Viewer3 (205x232)
Axes
Video Viewer2 (205x232)
Axes

In this example, you used the Contrast Adjustment block to linearly scale the pixel values in `pout.tif` between new upper and lower limits. You used the Histogram Equalization block to transform the values in `tire.tif` so that the histogram of the output image approximately matches a uniform histogram. For more information, see the Contrast Adjustment and Histogram Equalization block reference pages.

Adjusting the Contrast in Color Images

This example shows you how to modify the contrast in color images using the Histogram Equalization block.

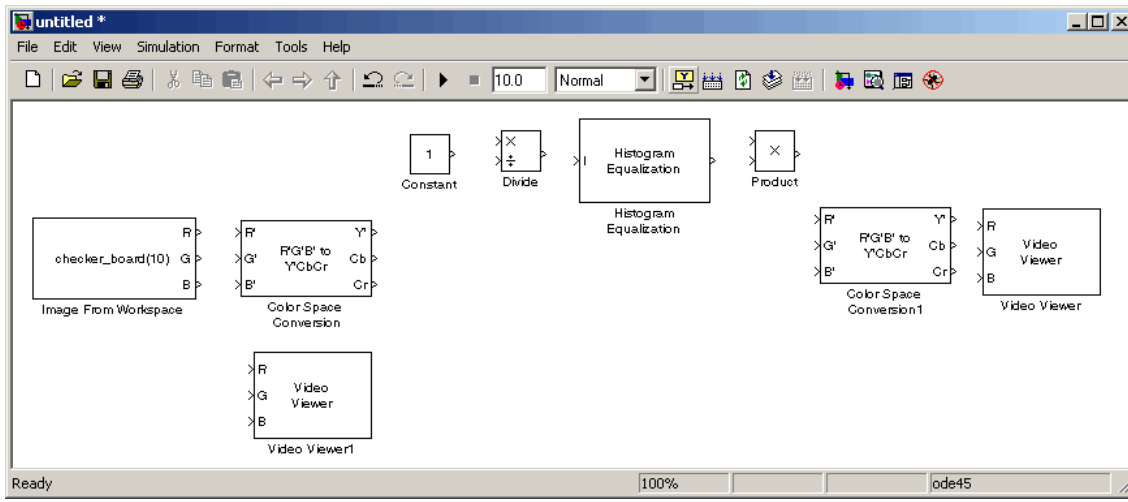
1. Use the following code to read in an indexed RGB image, `shadow.tif`, and convert it to an RGB image.

```
[X map] = imread('shadow.tif');
shadow = ind2rgb(X,map);
```

2. Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
Color Space Conversion	Video and Image Processing Blockset / Conversions	2
Constant	Simulink / Sources	1
Divide	Simulink / Math Operations	1
Histogram Equalization	Video and Image Processing Blockset / Analysis & Enhancement	1
Product	Simulink / Math Operations	1
Video Viewer	Video and Image Processing Blockset / Sinks	2

3. Place the blocks so that your model resembles the following figure.



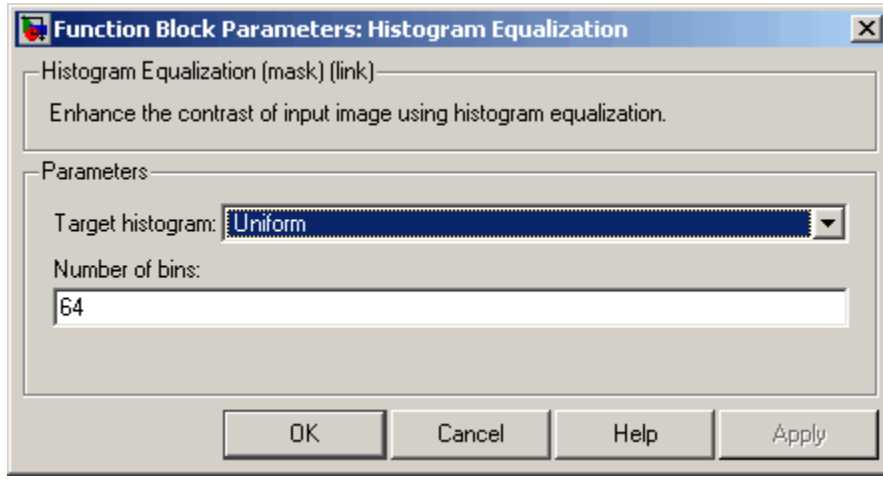
4 Use the Image From Workspace block to import the RGB image from the MATLAB workspace into the Simulink model. Set the **Value** parameter to `shadow`.

5 Use the Color Space Conversion block to separate the luma information from the color information. Set the **Conversion** parameter to `sR'G'B' to L*a*b*` to convert from one color space to the other. Because the range of the L* values is between 0 and 100, you must normalize them to between 0 and 1 before you pass them to the Histogram Equalization block, which expects floating point input in this range.

6 Use the Constant block to define a normalization factor. Set the **Constant value** parameter to `100`.

7 Use the Divide block to normalize the L* values to between 0 and 1. Use the default parameters.

8 Use the Histogram Equalization block to modify the contrast in the image. Use the default parameters.



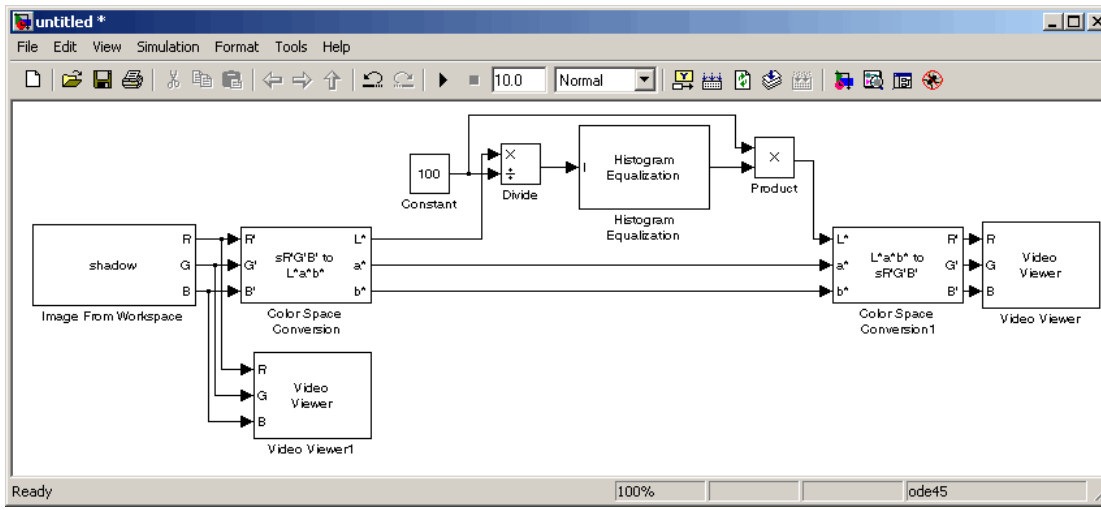
This block enhances the contrast of images by transforming the luma values in the color image so that the histogram of the output image approximately matches a specified histogram.

9 Use the Product block to scale the values back to the 0 to 100 range. Use the default parameters.

10 Set the **Conversion** parameter on the Color Space Conversion1 block to `L*a*b* to sR'G'B'` to convert the image from one color space to the other.

11 Use the Video Viewer blocks to view the original and modified images. For each block, use the default parameters.

12 Connect the blocks as shown in the following figure.



13 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

14 Run the model.

As shown in the following figure, the model displays the original image in the Video Viewer1 window.



As the next figure shows, the model displays the enhanced contrast image in the Video Viewer window.



In this example, you used the Histogram Equalization block to transform the values in a color image so that the histogram of the output image approximately matches a uniform histogram. For more information, see the Histogram Equalization block reference pages.

Pixel Statistics

The Video and Image Processing Blockset contains blocks that can provide information about the data values that make up an image. Blocks from the Statistics library, such as the 2-D Maximum and 2-D Autocorrelation blocks, can help you determine this information.

This section includes the following topic:

- "Finding the Histogram of an Image" on page 7-58 — Use the 2-D Histogram block to calculate the histogram of the R, G, and B values in an image

Finding the Histogram of an Image

The 2-D Histogram block computes the frequency distribution of the elements in each input image by sorting the elements into a specified number of discrete bins. You can use the 2-D Histogram block to calculate the histogram of the R, G, and/or B values in an image. This example shows you how to accomplish this task:

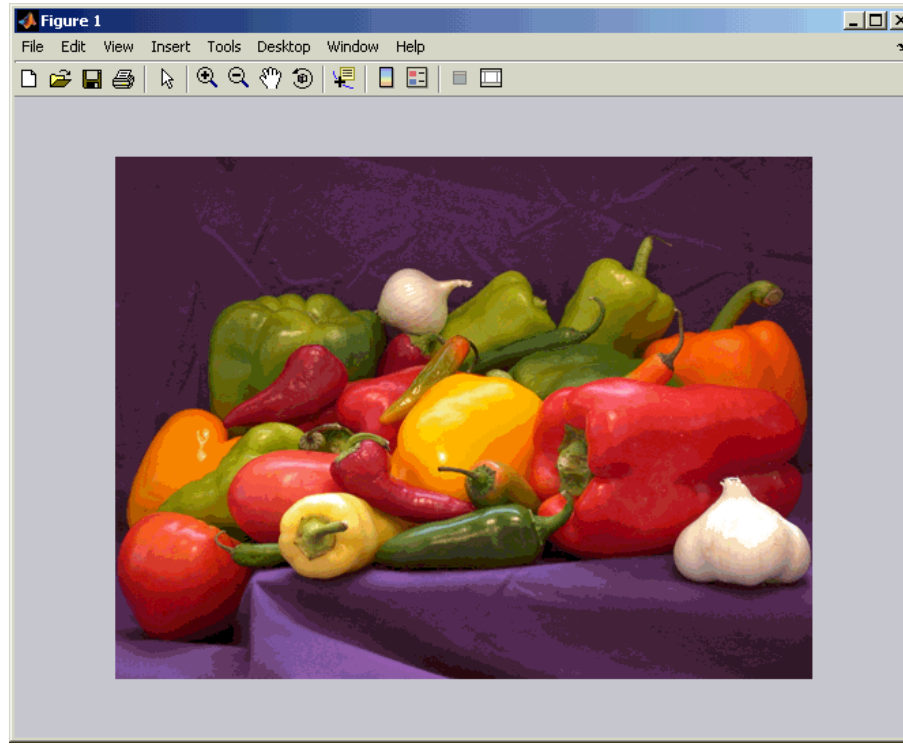
1. Define an RGB image in the MATLAB workspace. To read in an RGB image from a PNG file, at the MATLAB command prompt, type

   ```
   I= im2double(imread('peppers.png'));
   ```

   `I` is a 486-by-732-by-3 array of double-precision floating-point values. Each plane of the array represents the red, green, or blue color values of the image.

2. To view the image this matrix represents, at the MATLAB command prompt, type

   ```
   imshow(I)
   ```

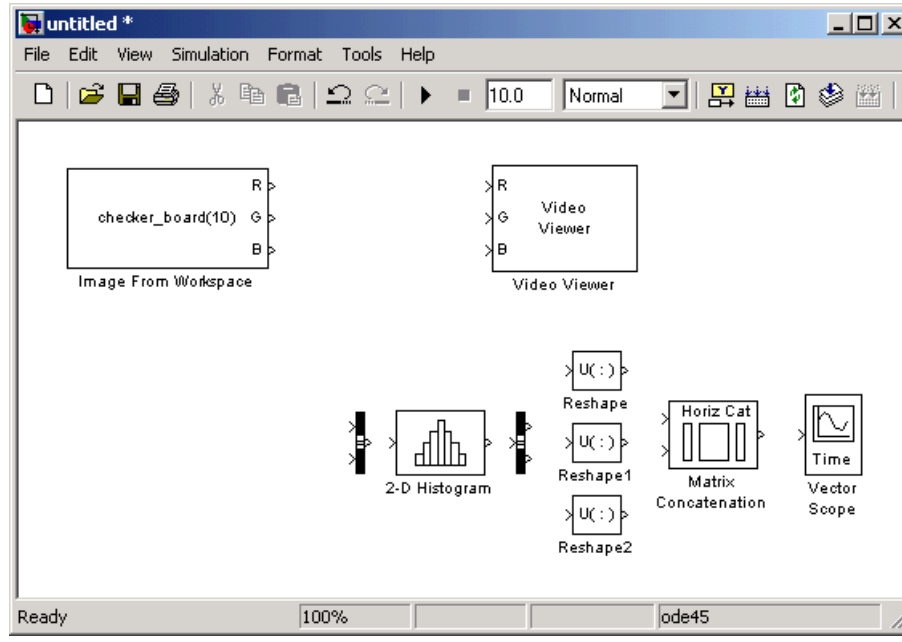


3 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
2-D Histogram	Video and Image Processing Blockset / Statistics	1
Video Viewer	Video and Image Processing Blockset / Sinks	1
Bus Creator	Simulink / Signal Routing	1
Bus Selector	Simulink / Signal Routing	1

Block	Library	Quantity
Reshape	Simulink / Math Operations	3
Matrix Concatenation	Simulink / Math Operations	1
Vector Scope	Signal Processing Blockset / Signal Processing Sinks	1

4 Place the blocks so that your model resembles the following figure.



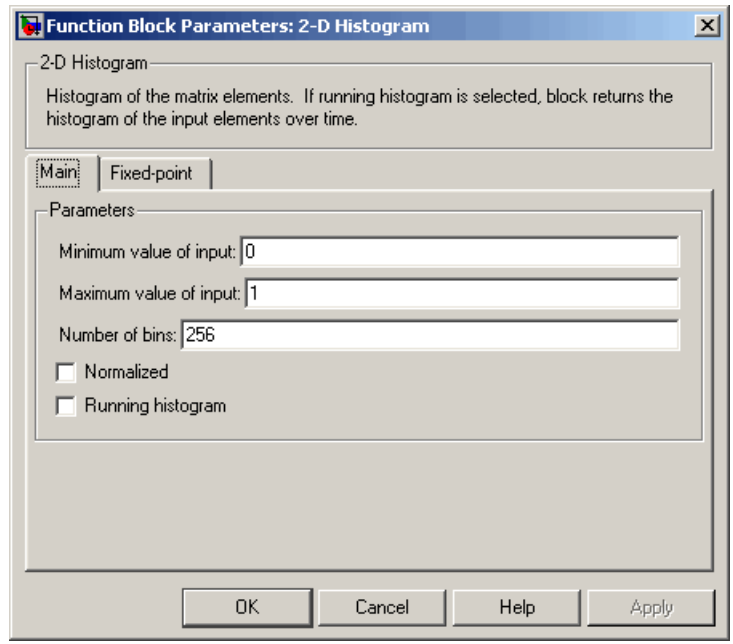
5 Use the Image From Workspace block to import the RGB image from the MATLAB workspace. Set the **Value** parameter to `I`.

The block outputs the R, G, and B planes of the `I` array at the output ports.

6 Use the Video Viewer block to automatically display the original image in the Video Viewer window when you run the model. Use the default parameters.

7 Use the Bus Creator block to combine the R, G, and B, signals into one signal so you can process it with one 2-D Histogram block. Set the **Number of inputs** parameter to 3.

8 Use the 2-D Histogram block to calculate the histogram of the R, G, and B values in the image. Use the default parameters.



The R, G, and B values input to the 2-D Histogram block are double-precision floating point and range between 0 and 1. The block creates 256 bins between the maximum and minimum input values and counts the number of R, G, and B values in each bin.

9 Use the Bus Selector block to expand the input signal into three separate R, G, and B, signals. You must set the block parameters of this block after you connect a signal to its input port. You configure this block later in this procedure.

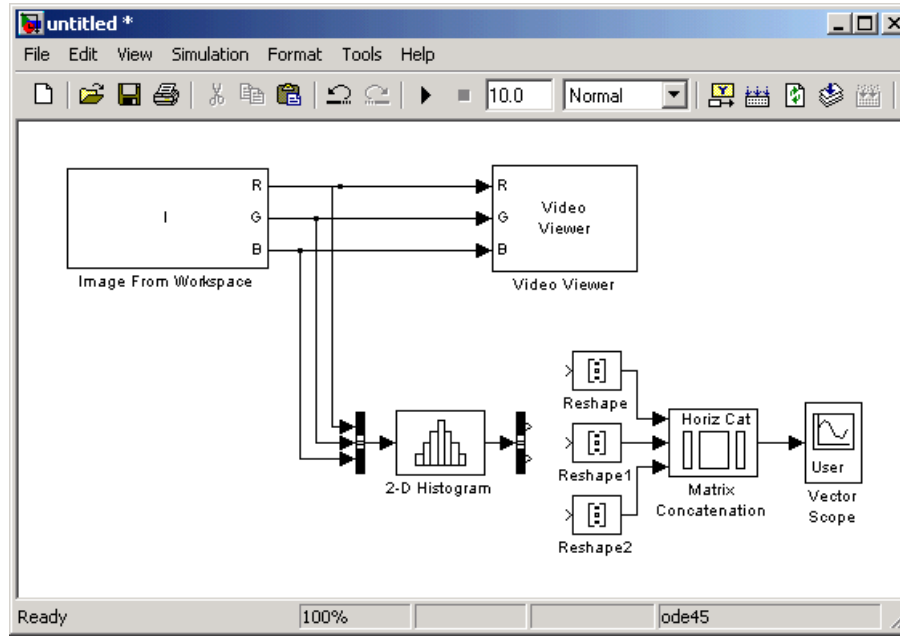
10 Use the Reshape blocks to transform the row vectors output from the Bus Selector block into columns vectors. Set the **Output dimensionality** parameters to `Column vector`.

11 Use the Matrix Concatenation block to concatenate the R, G, and B column vectors into a single matrix so they can be displayed using the Vector Scope block. Set the **Number of inputs** parameter to `3`.

12 Use the Vector Scope block to display the histograms of the R, G, and B values of the input image. Set the block parameters as follows:

- **Scope Properties** pane, **Input domain** = `User-defined`
- **Display Properties** pane, clear the **Frame number** check box
- **Display Properties** pane, select the **Channel legend** check box
- **Display Properties** pane, select the **Compact display** check box
- **Axis Properties** pane, clear the **Inherit sample increment from input** check box.
- **Axis Properties** pane, **Minimum Y-limit** = `0`
- **Axis Properties** pane, **Maximum Y-limit** = `1`
- **Axis Properties** pane, **Y-axis title** = `Count`
- **Line Properties** pane, **Line markers** = `.|s|d`
- **Line Properties** pane, **Line colors** = `[1 0 0]|[0 1 0]|[0 0 1]`

13 Connect the blocks as shown in the following figure.



Note that the Bus Selector block still needs to be connected. You cannot configure the parameters of this block until you connect an input signal to it.

14 Configure the Bus Selector block. Double-click the block. In the **Signals in the bus** pane, select `signal3`. Click **Select** to move `signal3` to the **Selected signals** pane. Click **OK**.

The Bus Selector block now has three output ports.

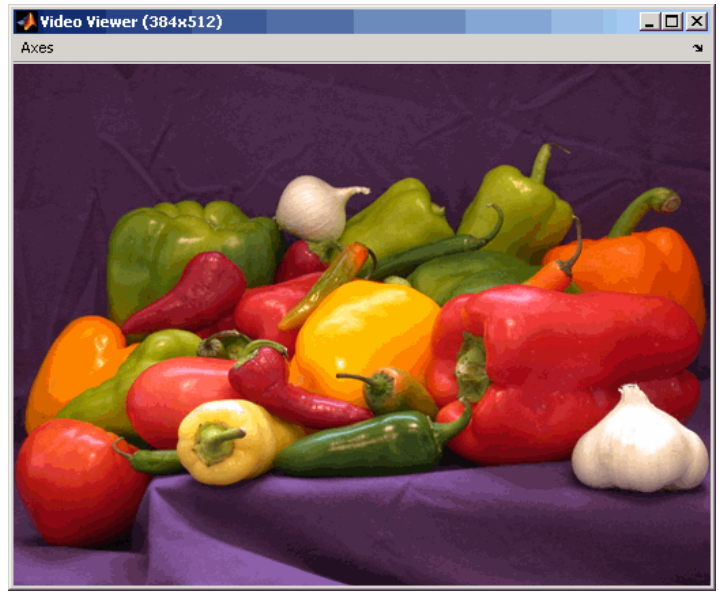
15 Connect the Bus Selector block to the Reshape blocks.

16 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

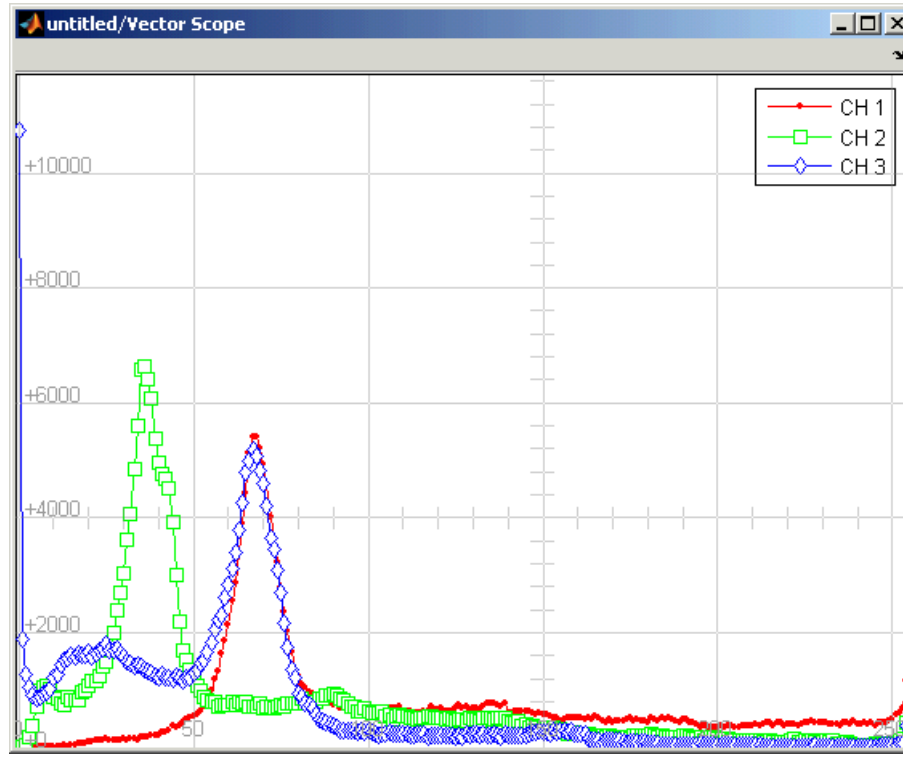
17 Run the model.

The original image appears in the Video Viewer window. To view the image at its true size, right-click the window and select **Set Display To True Size**.



18 Right-click in the Vector Scope window and select **Autoscale**.

The scaled histogram of the image appears in the Vector Scope window.



You have now used the 2-D Histogram block to calculate the histogram of the R, G, and B values in an RGB image. For more information about this block, see the 2-D Histogram block reference page. To open a demo model that illustrates how to use this block to calculate the histogram of the R, G, and B values in an RGB video stream, type `viphistogram` at the MATLAB command prompt.

8

Example Applications

Video and Image Processing Blockset blocks enable you to track objects in an image, stabilize a video stream, and compress images.

Pattern Matching (p. 8-2)	Learn how to track the motion of a sculpture in a video stream.
Motion Compensation (p. 8-9)	Explore the video compression and stabilization demo models.
Image Compression (p. 8-11)	Understand how to compress an image and view the result.

Pattern Matching

Pattern matching can be used to recognize and/or locate specific objects in an image. It can be accomplished using several techniques, one of which is correlation. Correlation provides a direct measure of the similarity between two images. Though sensitive to the scaling or rotation of objects, normalized correlation is robust to changes in lighting.

This section includes the following topic:

- "Tracking an Object Using Correlation" on page 8-2-- Use the 2-D Correlation, 2-D Maximum, and Draw Shapes blocks to track the motion of an object in a video stream

Tracking an Object Using Correlation

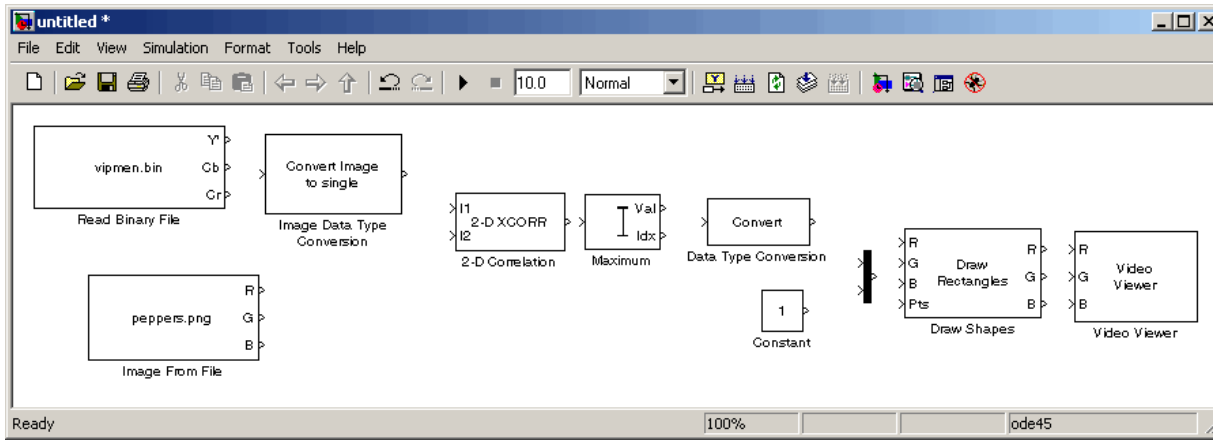
In this example, you use the 2-D Correlation, Maximum, and Draw Shapes blocks to find and indicate the location of a sculpture in each video frame:

1 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Read Binary File	Video and Image Processing Blockset / Sources	1
Image Data Type Conversion	Video and Image Processing Blockset / Conversions	1
Image From File	Video and Image Processing Blockset / Sources	1
2-D Correlation	Video and Image Processing Blockset / Statistics	1
Maximum	Video and Image Processing Blockset / Statistics	1
Draw Shapes	Video and Image Processing Blockset / Text & Graphics	1

Block	Library	Quantity
Video Viewer	Video and Image Processing Blockset / Sinks	1
Data Type Conversion	Simulink / Signal Attributes	1
Constant	Simulink / Sources	1
Mux	Simulink / Signal Routing	1

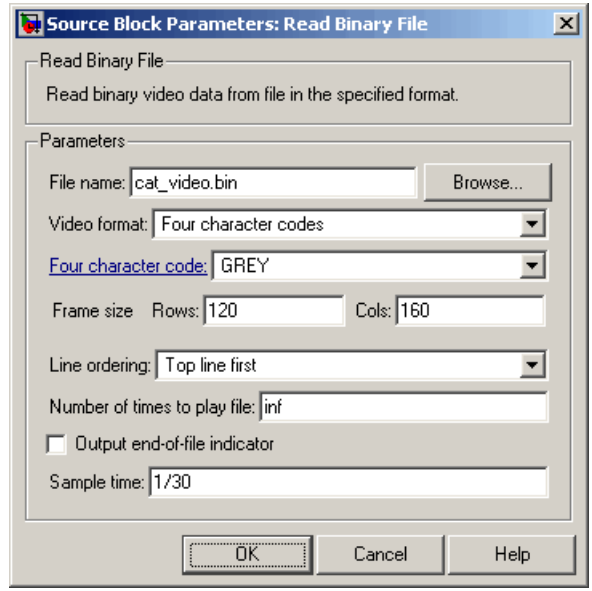
2 Position the blocks as shown in the following figure.



You are now ready to set your block parameters by double-clicking the blocks, modifying the block parameter values, and clicking **OK**.

3 Use the Read Binary File block to import a binary file into the model. Set the block parameters as follows:

- **File name** = `cat_video.bin`
- **Four character code** = `GREY`
- **Number of times to play file** = `inf`
- **Sample time** = `1/30`



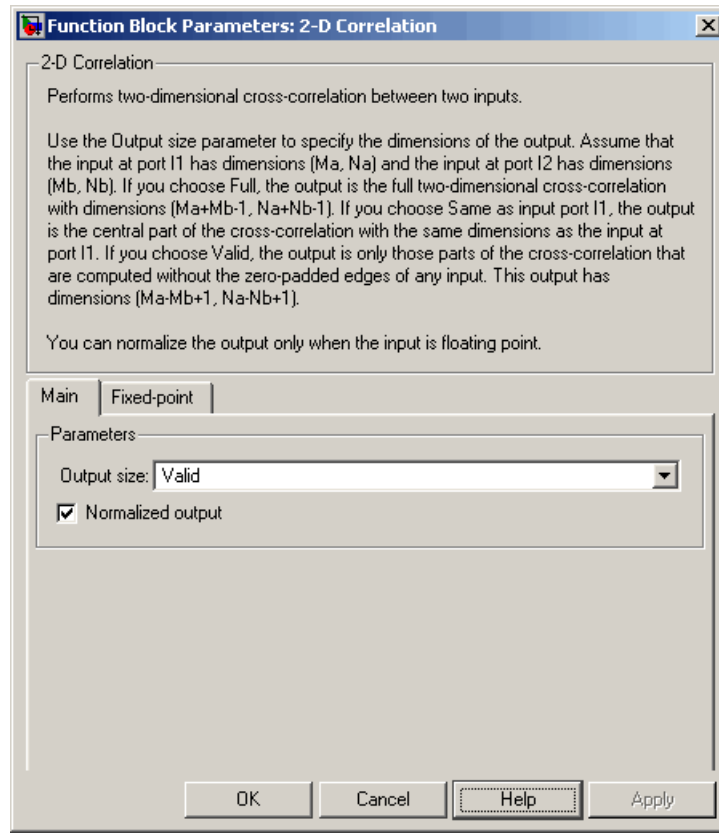
4 Use the Image Data Type Conversion block to convert the data type of the video to single-precision floating point. Use the default parameter.

5 Use the Image From File block to import the image of the cat sculpture, which is the object you want to track. Set the block parameters as follows:

- **Main** pane, **File name** = `cat_target.png`
- **Main** pane, **Output port labels** = `I`
- **Data Types** pane, **Output data type** = `single`

6 Use the 2-D Correlation block to determine the portion of each video frame that best matches the image of the cat sculpture. Set the block parameters as follows:

- **Output size** = `Valid`
- Select the **Normalized output** check box.



Because you chose `Valid` for the **Output size** parameter, the block outputs only those parts of the correlation that are computed without the zero-padded edges of any input.

7 Use the Maximum block to find the index of the maximum value in each input matrix. Set the **Mode** parameter to `Index`.

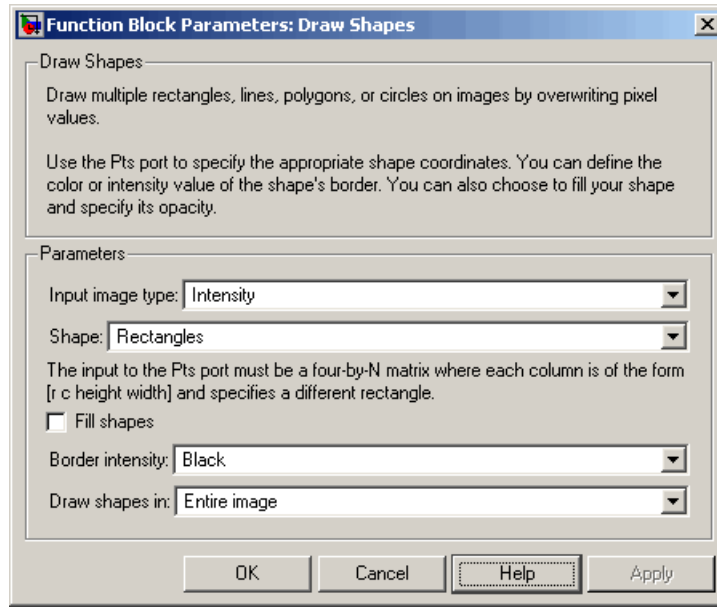
The block outputs the zero-based location of the maximum value as a two-element vector of 32-bit unsigned integers at the Idx port.

8 Use the Data Type Conversion block to change the index values from 32-bit unsigned integers to single-precision floating-point values. Set the **Output data type mode** parameter to `single`.

9 Use the Constant block to define the size of the image of the cat sculpture. Set the **Constant value** parameter to `single([41 41])`.

10 Use the Mux block to concatenate the location of the maximum value and the size of the image of the cat sculpture into a single vector. You use this vector to define a rectangular region of interest (ROI) that you pass to the Draw Shapes block.

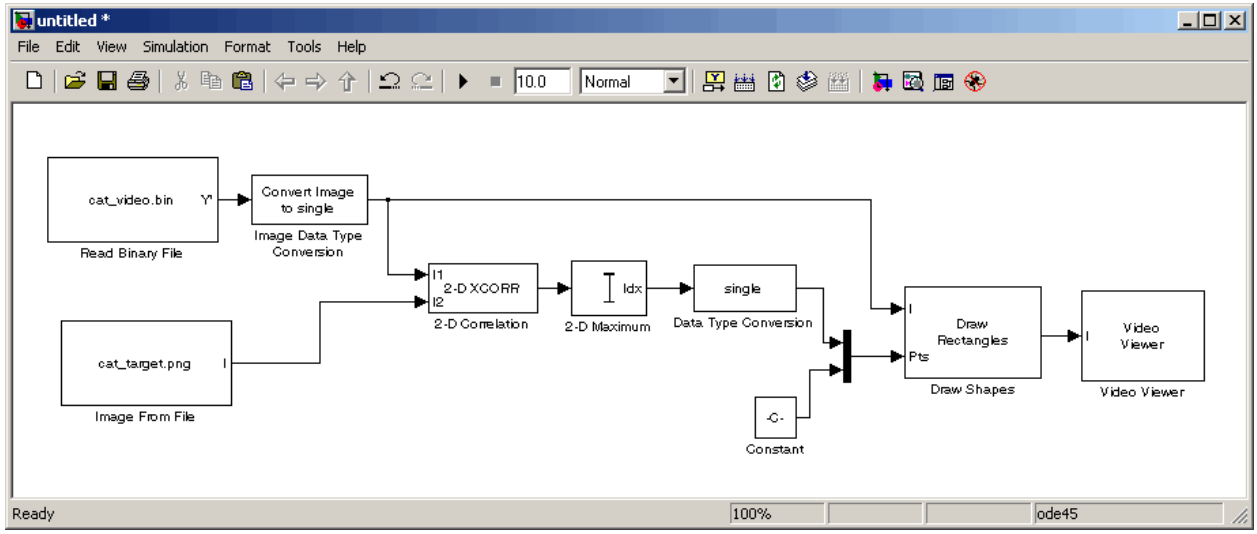
11 Use the Draw Shapes block to draw a rectangle around the portion of each video frame that best matches the image of the cat sculpture. Set the **Input image type** parameter to`Intensity`.



12 Use the Video Viewer block to display the video stream with the ROI displayed on it. Set the **Input image type** parameter to `Intensity`.

The Video Viewer block automatically displays the video in the Video Viewer window when you run the model. Because the image is represented by single-precision floating-point values, a value of 0 corresponds to black and a value of 1 corresponds to white.

13 Connect the blocks as shown in the following figure.

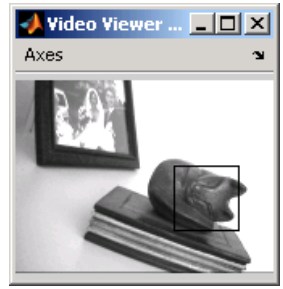


14 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

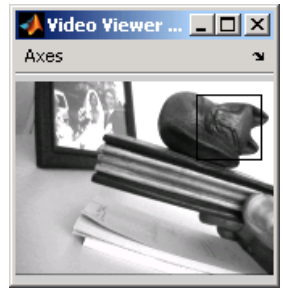
- **Solver** pane, **Stop time** = `inf`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

15 Run the simulation.

The video is displayed in the Video Viewer window and a rectangular box appears around the cat sculpture. To view the video at its true size, right-click the window and select **Set Display To True Size**.



As the video plays, you can watch the rectangular ROI follow the sculpture as it moves.



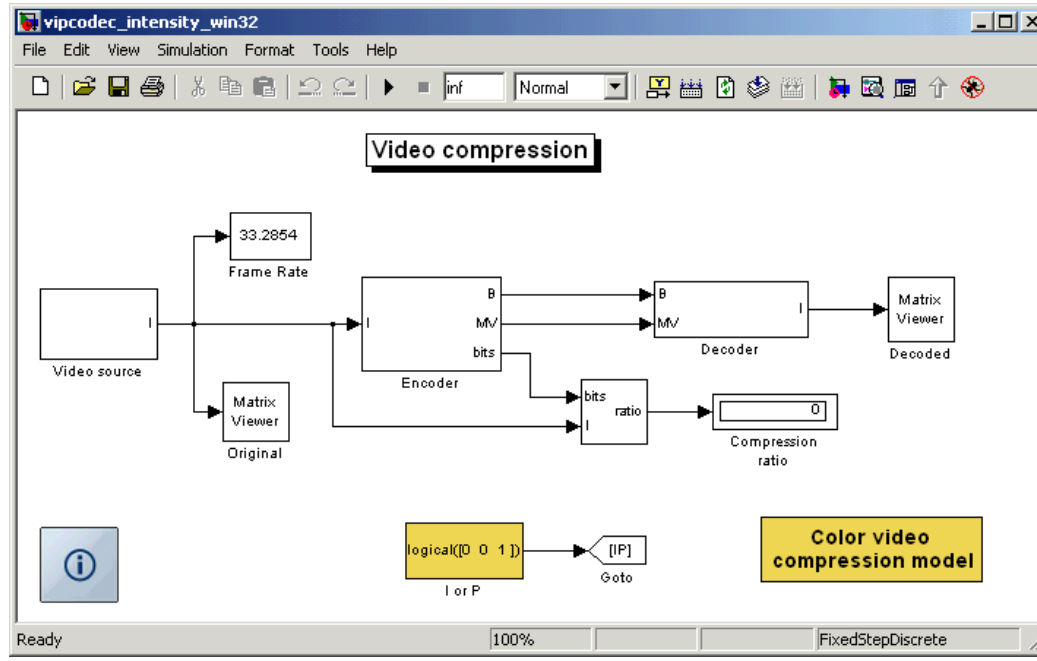
In this example, you used the 2-D Correlation, 2-D Maximum, and Draw Shapes blocks to track the motion of an object in a video stream. For more information about these blocks, see the 2-D Correlation, Maximum, and Draw Shapes block reference pages.

Note This example model does not provide an indication of whether or not the sculpture is present in each video frame. For an example of this type of model, type `vippattern` at the MATLAB command prompt.

Motion Compensation

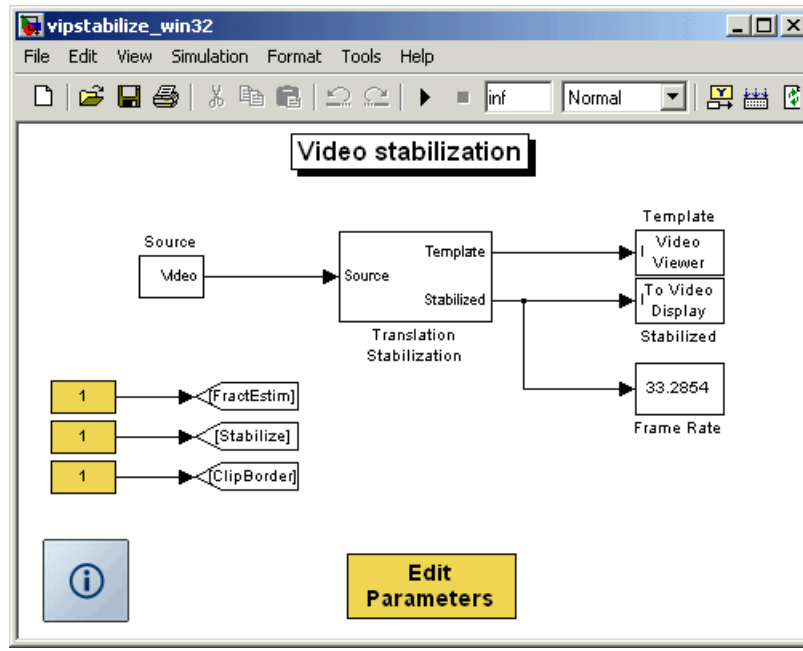
Motion compensation is a set of techniques that take advantage of redundancy in consecutive video frames. These techniques are used in video processing applications such as video compression and video stabilization. For both of these applications, motion compensation is a two-step process of detection and compensation. The detection step results in the specification of a motion vector that relates two consecutive video frames. For video compression, the compensation step involves using the motion vector to predict the current video frame from the previous frame and encoding the prediction residual. For video stabilization, the compensation step involves translating the current frame in the opposite direction of the motion vector to stabilize the video sequence.

The Video and Image Processing Blockset contains a video compression demo model that you can open by typing `vipcodec` at the MATLAB command prompt.



This demo model detects motion by analyzing how much objects move between consecutive video frames. The model aligns two sequential video frames, subtracts them, and codes the residual.

The Video and Image Processing Blockset also contains a video stabilization demo model that you can open by typing `vipstabilize` at the MATLAB command prompt.



The demo illustrates a motion stabilization technique based on the sum of absolute differences (SAD) method. It applies the SAD technique to remove unwanted translational camera motions and generate a stabilized video.

Image Compression

The examples in this section illustrate how to build a Simulink model that is capable of image compression. For image compression algorithms, the input image is divided into blocks and the two-dimensional DCT is computed for each block. The DCT coefficients are then quantized, coded, and transmitted. The receiver decodes the quantized DCT coefficients, computes the inverse two-dimensional DCT of each block, and then puts the blocks back together into a single image. Although there is some loss of quality in the reconstructed image, it is recognizable as an approximation of the original image.

This section includes the following topics:

- "Compressing an Image" on page 8-11 -- Use the 2-D DCT block to compress a matrix of image coefficients
- "Viewing the Compressed Image" on page 8-18 -- Use the 2-D IDCT block to transform the image back to the time domain so it can be viewed

Compressing an Image

You can use image compression to reduce the size of an image before you transmit it. The compressed image retains many of the original image's features but requires less bandwidth. In this topic, you use the 2-D DCT and Selector blocks to compress an intensity image:

1. Define an intensity image in the MATLAB workspace. To read in an intensity image from a TIF file and convert it to double-precision, at the MATLAB command prompt, type

   ```
   I= imread('cameraman.tif');
   ```

   `I` is a 256-by-256 matrix of 8-bit unsigned integer values.

2. To view the image this matrix represents, at the MATLAB command prompt, type

   ```
   imshow(I)
   ```

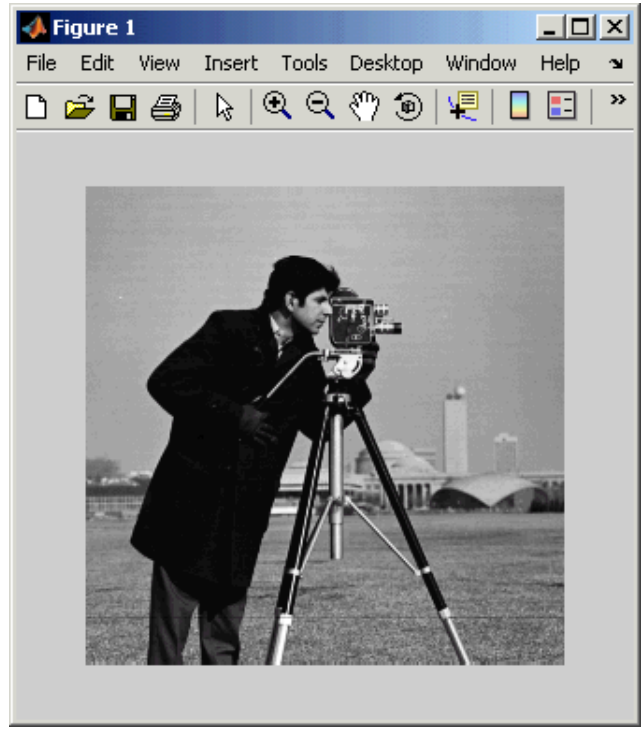
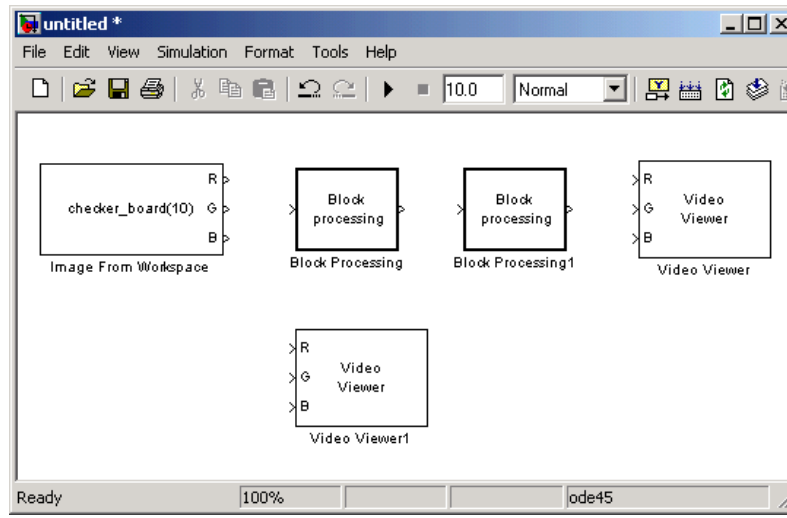


Image Courtesy of MIT

3 Create a new Simulink model, and add to it the blocks shown in the following table.

Block	Library	Quantity
Image From Workspace	Video and Image Processing Blockset / Sources	1
Block Processing	Video and Image Processing Blockset / Utilities	2
Video Viewer	Video and Image Processing Blockset / Sinks	2

4 Position the blocks as shown in the following figure.



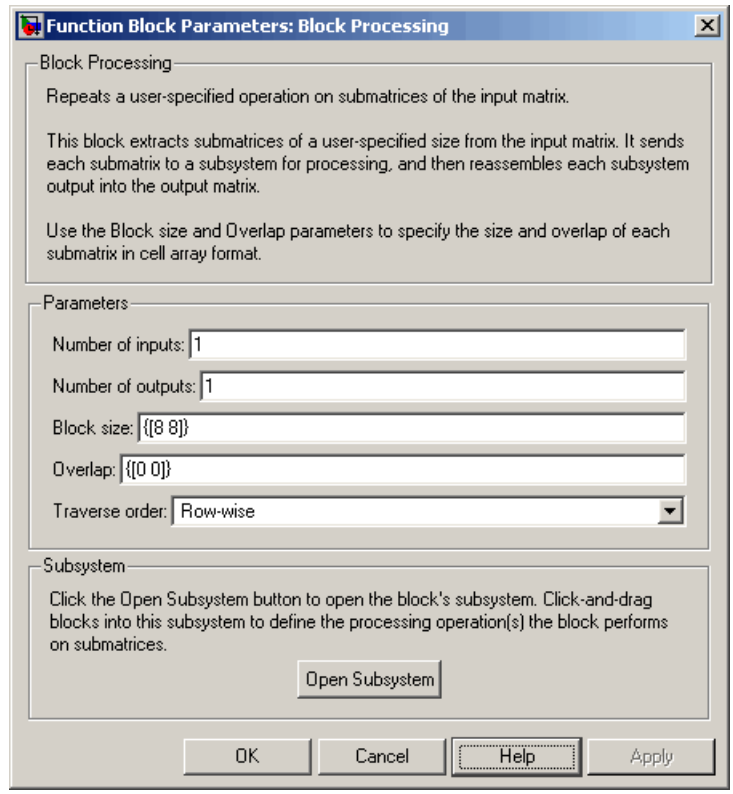
You are now ready to set your block parameters by double-clicking the blocks, modifying the block parameter values, and clicking **OK**.

5 Use the Image From Workspace block to import the intensity image into your model. Set the block parameters as follows:

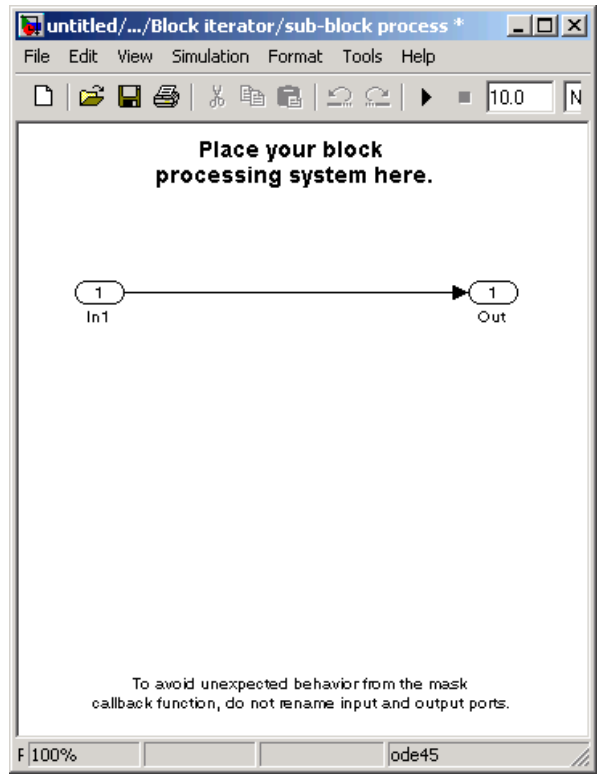
- **Main** pane, **Value** = `I`
- **Main** pane, **Output port labels** = `Image`
- **Data Types** pane, **Output data type** = `double`

6 Use the Video Viewer1 block to view the original intensity image. Set the **Input image type** parameter to `Intensity`.

7 The first Block Processing block represents the transmission portion of the block diagram. This block sends 8-by-8 submatrices of the original matrix to the block's subsystem for processing. Use this block when you want to perform block-based processing on large input images. To view the subsystem, double-click the block and click **Open Subsystem**.



The Block Processing block's subsystem opens.

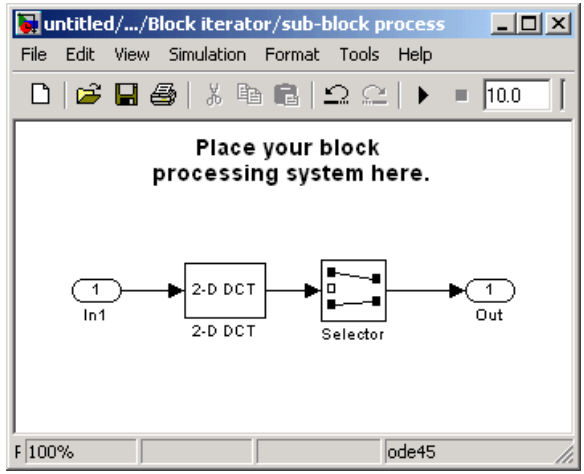


You can drag blocks into this subsystem to process the submatrices.

8 Add the following blocks to your subsystem.

Block	Library	Quantity
2-D DCT	Video and Image Processing Blockset / Transforms	1
Selector	Simulink / Signal Routing	1

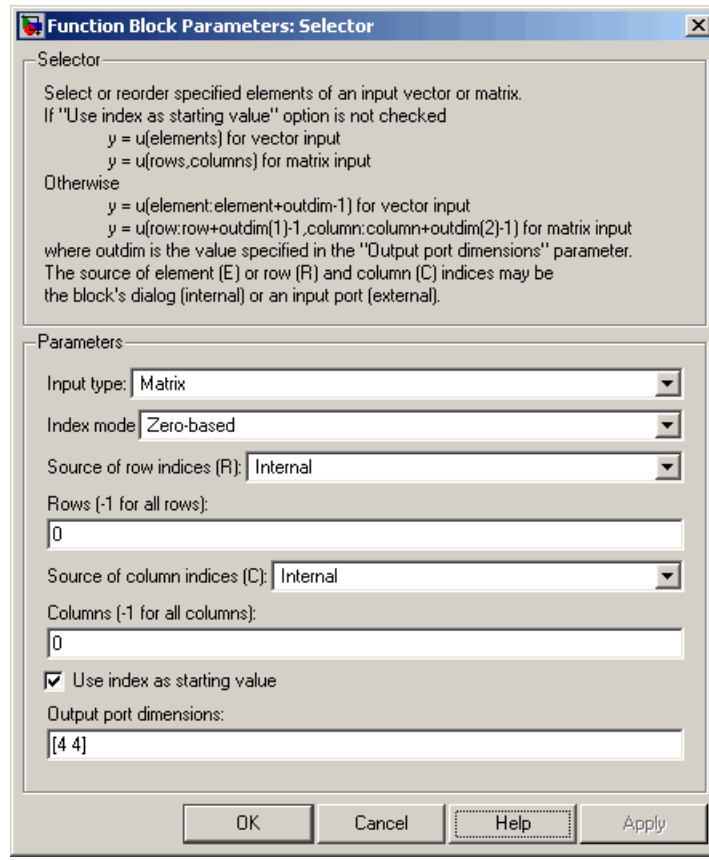
9 Connect the blocks as shown in the figure below.



10 The 2-D DCT block takes the two-dimensional DCT of each submatrix. This process puts most of the energy in the image into the upper left corner of the resulting matrix. Use the default parameters.

11 Use the Selector block to extract the upper left corner of the submatrix. Set the block parameters as follows:

- **Input type** = `Matrix`
- **Index mode** = `Zero-based`
- **Rows (-1 for all rows)** = `0`
- **Columns (-1 for all columns)** = 0
- Select the **Use index as starting value** check box.
- **Output port dimensions** = `[4 4]`



You are using the Selector block to compress the image by extracting the upper left corner of the submatrix, which contains the high energy image coefficients. You want to transmit only this portion of the submatrix because it requires less bandwidth than transmitting the entire submatrix.

12 Close the subsystem and the Block Processing dialog box.

You have now configured the Block Processing and 2-D DCT blocks to compress an image for transmission. In “Viewing the Compressed Image” on page 8-18, you use the 2-D IDCT block to transform the image back to the time domain. Then, you view the compressed image.

Viewing the Compressed Image

In "Compressing an Image" on page 8-11, you compressed an image using the 2-D DCT and Selector blocks. Now, you can use the 2-D IDCT block to transform the image back to the time domain and view the result:

1. If you have not already done so, define an intensity image in the MATLAB workspace by typing

   ```
   I= imread('cameraman.tif');
   ```

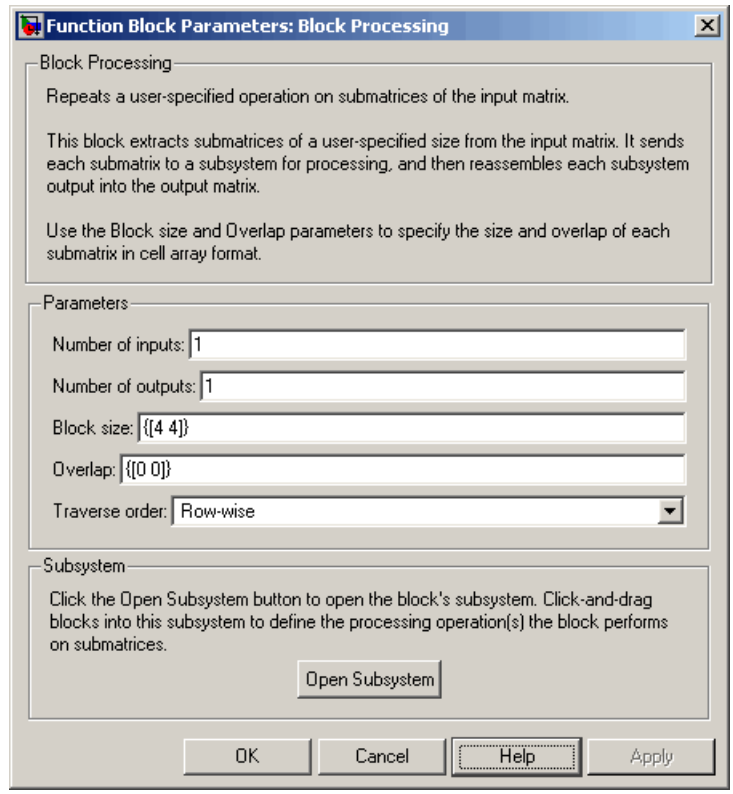
   `I` is a 256-by-256 matrix of 8-bit unsigned integer values.

2. If the model you created in "Compressing an Image" on page 8-11 is not open on your desktop, you can open an equivalent model by typing

   ```
   doc_compression
   ```

   at the MATLAB command prompt.

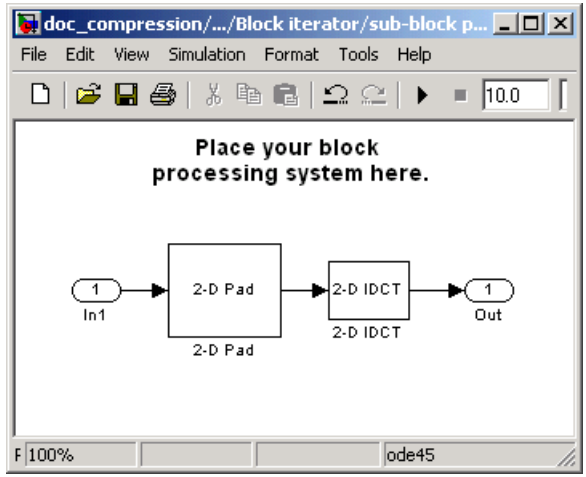
3. Use the Block Processing1 block to set the size of the submatrices that the block passes to the subsystem. Set the **Block size** parameter to `{[4 4]}`.



4 Open the block's subsystem by clicking **Open Subsystem**, and add the following blocks to it.

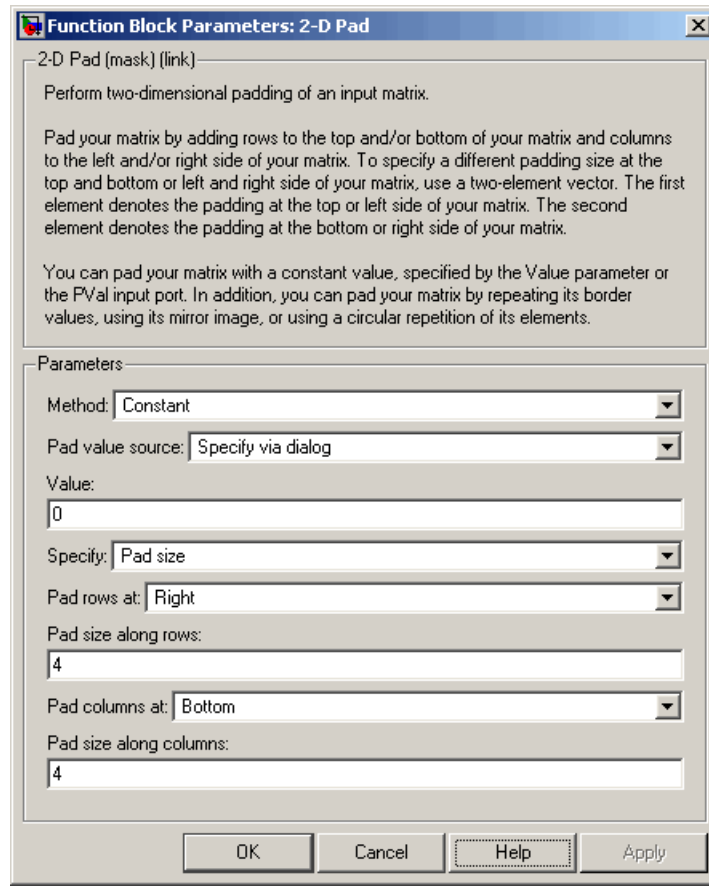
Block	Library	Quantity
2-D Pad	Video and Image Processing Blockset / Utilities	1
2-D IDCT	Video and Image Processing Blockset / Transforms	1

5 Connect the blocks as shown in the figure below.



6 Use the 2-D Pad block to zero pad the 4-by-4 submatrix back to its original 8-by-8 size. Set the block parameters as follows:

- **Pad rows at** = `Right`
- **Pad size along rows** = `4`
- **Pad columns at** = `Bottom`
- **Pad size along columns** = `4`



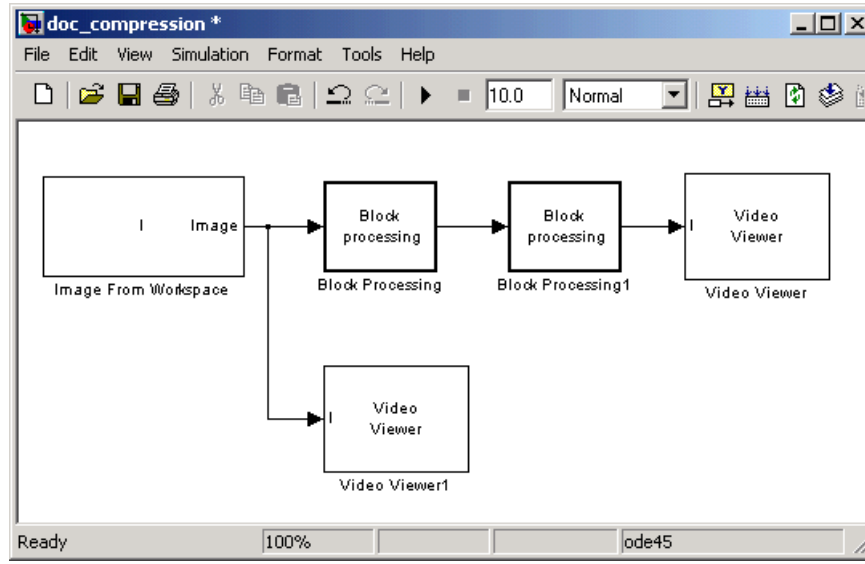
Because zeros are replacing the low energy transform coefficients, the output image is an approximation of the original image.

7 The 2-D IDCT block takes the inverse two-dimensional DCT of the submatrices. Use the default parameters.

8 Close the subsystem and the Block Processing1 dialog box.

9 Use the Video Viewer block to view the compressed image. Set the **Input image type** parameter to `Intensity`.

10 Connect the blocks as shown in the figure below.



11 Set the configuration parameters. Open the Configuration dialog box by selecting **Configuration Parameters** from the **Simulation** menu. Set the parameters as follows:

- **Solver** pane, **Stop time** = `0`
- **Solver** pane, **Type** = `Fixed-step`
- **Solver** pane, **Solver** = `discrete (no continuous states)`

12 Run the model.

The original image is displayed in the Video Viewer1 window. To view the image at its true size, right-click the window and select **Set Display To True Size**.



The compressed image is displayed in the Video Viewer window. The compressed image is not as clear as the original image. However, it still contains many of its features. The image below is shown at its true size.



In this example, you used the 2-D DCT, 2-D Pad 2-D IDCT, and Block Processing blocks to compress an image. For more information on these blocks, see the 2-D DCT, 2-D Pad, 2-D IDCT, and Block Processing block reference

pages. For information on the Selector block, see the Simulink documentation. For more information on sharpening an image, see “Sharpening and Blurring an Image” on page 7-26.

Blocks — By Category

Analysis & Enhancement (p. 9-2)	Analyze or enhance images or video
Conversions (p. 9-2)	Perform conversion operations such as color space conversion
Filtering (p. 9-3)	Filter images or video
Geometric Transformations (p. 9-3)	Manipulate size, shape, and orientation of images or video
Morphological Operations (p. 9-4)	Perform morphological operations such as erosion and dilation
Sinks (p. 9-4)	Export or display images or video
Sources (p. 9-5)	Import images or video into Simulink
Statistics (p. 9-5)	Perform statistical operations on images or video
Text & Graphics (p. 9-6)	Annotate images or video
Transforms (p. 9-6)	Perform transform operations such as 2-D FFT and 2-D DCT
Utilities (p. 9-7)	Perform processing operations such as 2-D Pad and block processing

Analysis & Enhancement

Block Matching	Estimate motion between images or video frames
Contrast Adjustment	Adjust image contrast by linearly scaling pixel values
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
Median Filter	Perform 2-D median filtering

Geometric Transformations

Projective Transformation	Transform quadrilateral into another quadrilateral
Resize	Enlarge or shrink image sizes
Rotate	Rotate image by specified angle
Shear	Shift rows or columns of image by linearly varying offset
Translate	Translate image in 2-D plane using displacement vector

Morphological Operations

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

Sinks

Frame Rate Display	Calculate average update rate of input signal
To Multimedia File	Write video frames and audio samples to multimedia file
To Video Display	Send video data to display devices
Video To Workspace	Export video signal to MATLAB workspace
Video Viewer	Display binary, intensity, or RGB images or video streams
Write AVI File	Write video frames to uncompressed AVI file
Write Binary File	Write binary video data to files

Sources

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

Statistics

2-D Autocorrelation	Compute 2-D autocorrelation of input matrix
2-D Correlation	Compute 2-D cross-correlation of two input matrices
2-D Histogram	Generate histogram of each input matrix
2-D Mean	Find mean value of each input matrix
2-D Median	Find median value of each input matrix
2-D Standard Deviation	Find standard deviation of each input matrix
2-D Variance	Compute variance of each input matrix
Blob Analysis	Compute statistical values for labeled regions
Find Local Maxima	Find local maxima in matrices

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

Text & Graphics

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

Transforms

2-D DCT	Compute 2-D discrete cosine transform (DCT)
2-D FFT	Compute 2-D FFT of input
2-D IDCT	Compute 2-D inverse discrete cosine transform (IDCT)
2-D IFFT	Compute 2-D IFFT of input
Gaussian Pyramid	Perform Gaussian pyramid decomposition

Hough Lines	Find Cartesian coordinates of lines described by rho and theta pairs
Hough Transform	Find lines in images

Utilities

2-D Pad	Pad matrix along its rows and columns
Block Processing	Repeat user-specified operation on submatrices of input matrix
Variable Selector	Specify subset of rows or columns from input

10

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 $0 \le j < 2Na-1$.

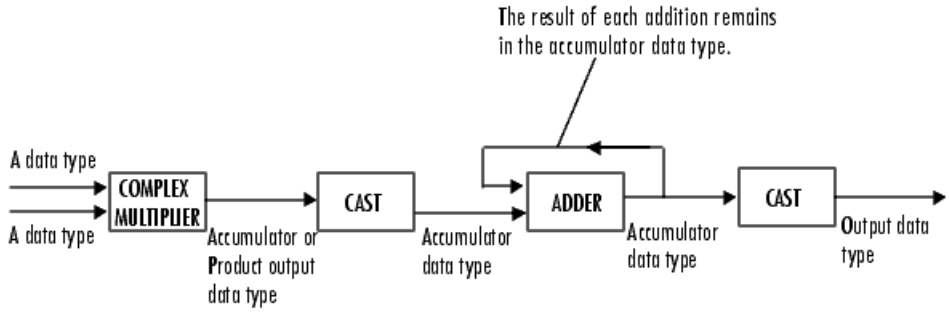
The output of this block has dimensions $(2Ma-1, 2Na-1)$.

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Scalar, 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 integers • 8-, 16-, 32-bit unsigned integers	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. This block supports a signal represented by a Simulink virtual bus.

Fixed-Point Data Types

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

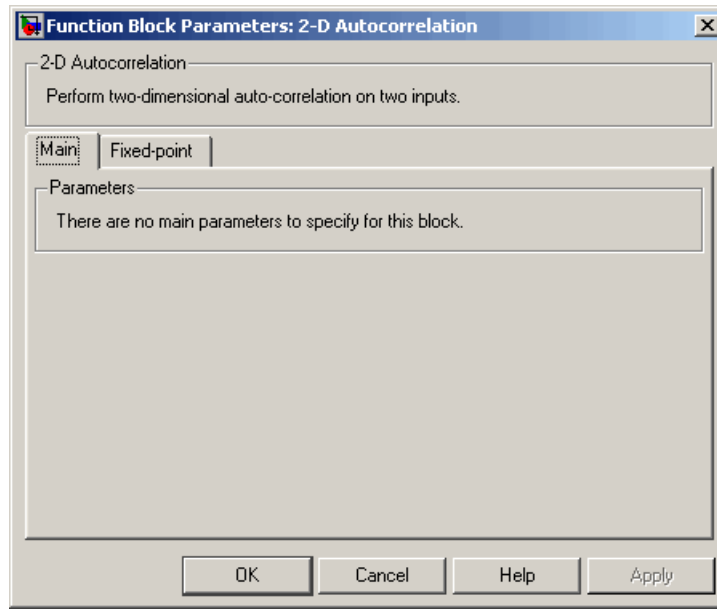


You can set the product output, accumulator, and output data types in the block mask as discussed in “Dialog Box” on page 10-4.

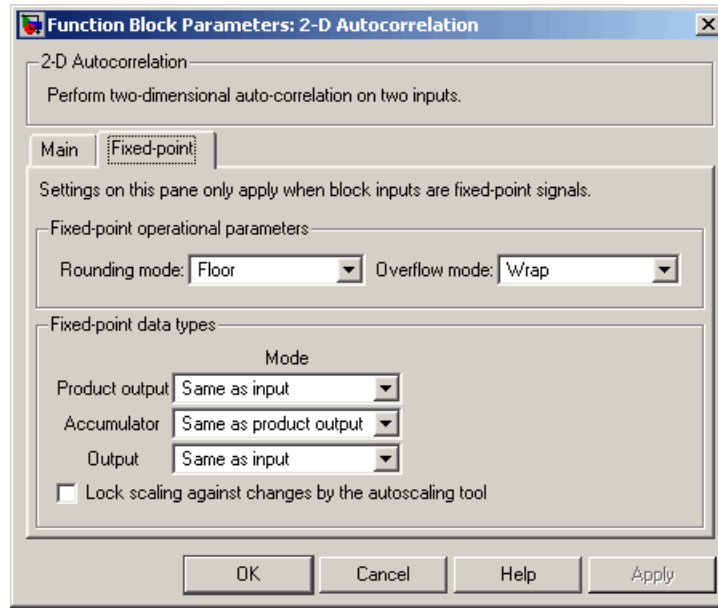
The output of the multiplier is in the product output data type if at least one of the inputs to the multiplier is real. If both of the inputs to the multiplier are complex, the result of the multiplication is in the accumulator data type. For details on the complex multiplication performed, refer to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



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



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

Use this parameter to specify how to designate the product output word and fraction lengths. Refer to "Fixed-Point Data Types" on page 10-3 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 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 10-3 and "Multiplication Data Types" in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block. Note that the accumulator data type is only used when both inputs to the multiplier are complex.

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

Output

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

- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the output. The bias of all signals in the Video and Image Processing Blockset 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

See Also

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

2-D Convolution

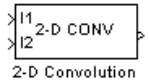
Purpose

Compute 2-D discrete convolution of two input matrices

Library

Filtering

Description



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 integers • 8-, 16-, 32-bit unsigned integers	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. This block supports a signal represented by a Simulink virtual bus.

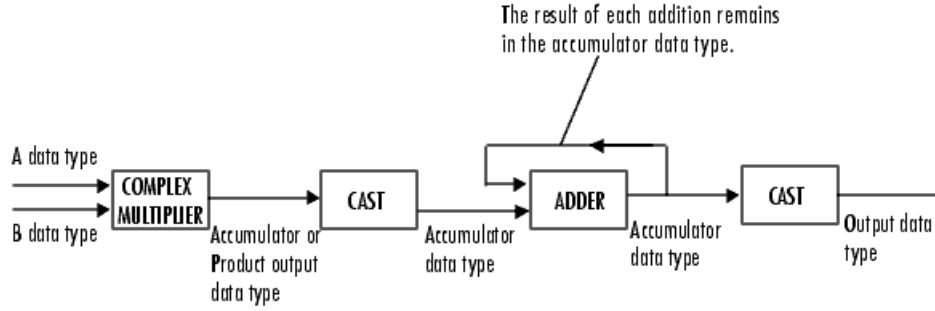
The dimensions of the output are dictated by the **Output size** parameter. Assume that the input at port I1 has dimensions (Ma, Na) and the input at port I2 has dimensions (Mb, Nb). If, for the **Output size** parameter, you choose `Full`, the output is the full two-dimensional convolution with dimensions (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 10-12 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 10-15.

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

Examples

Example 1

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

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

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

The resulting matrix is

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

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

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

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

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

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

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

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

Example 2

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

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

```
I1 = [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:

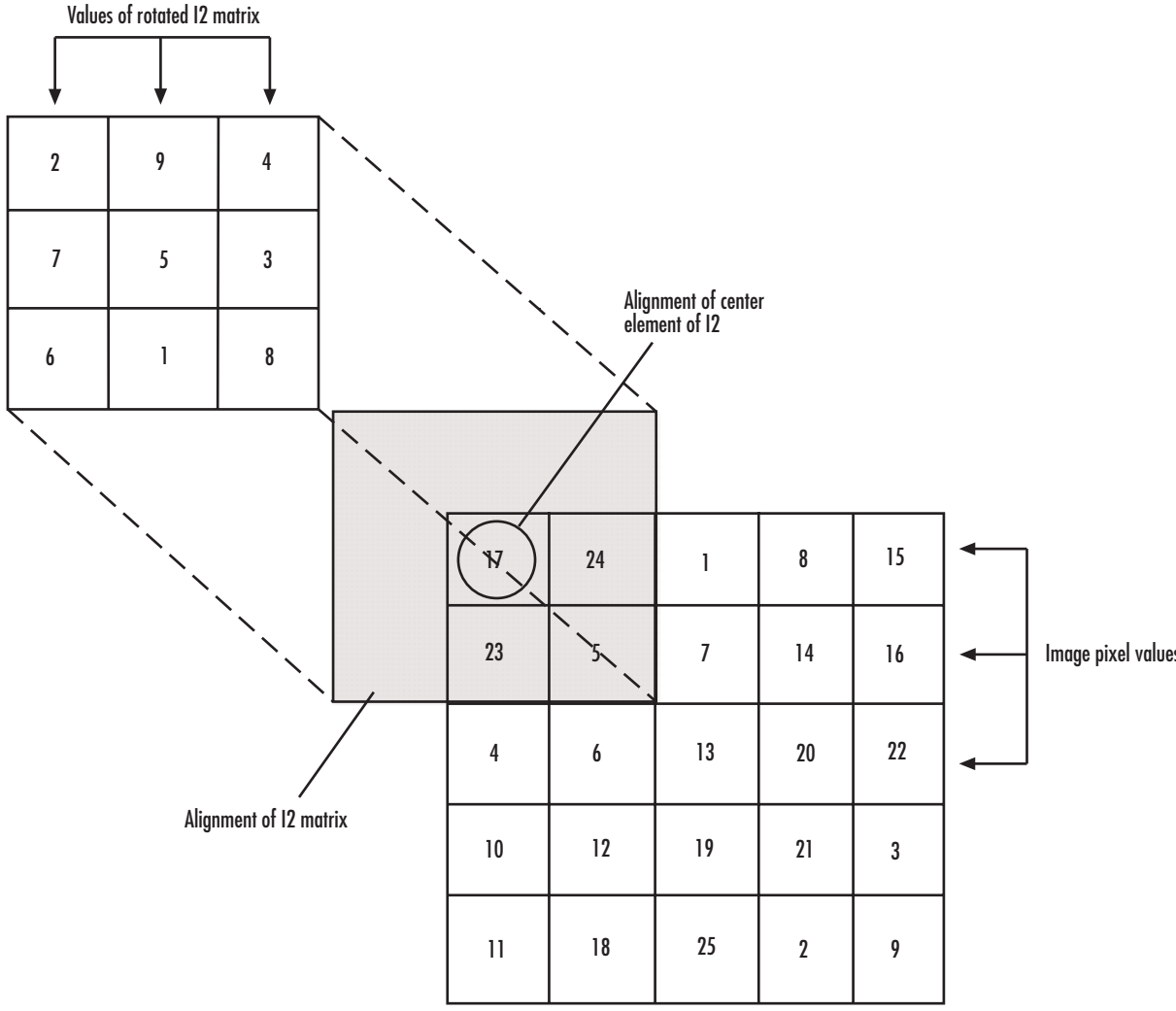
1. Rotate the second input matrix, I2, 180 degrees about its center element.
2. Slide the center element of I2 so that it lies on top of the (0,0) element of I1.

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

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

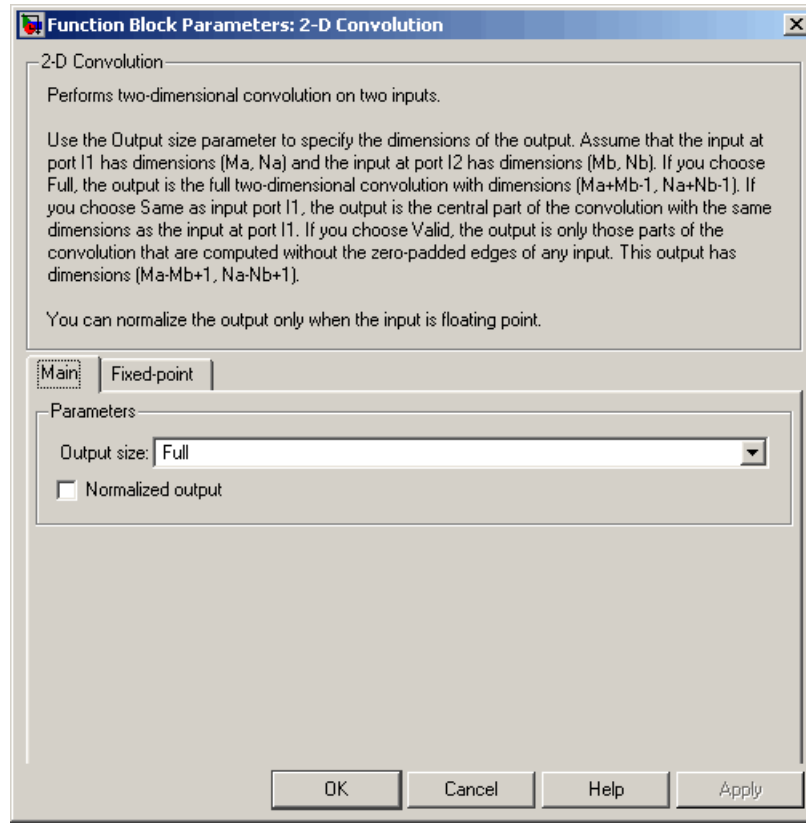


Computing the (1,1) Output of Convolution

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.



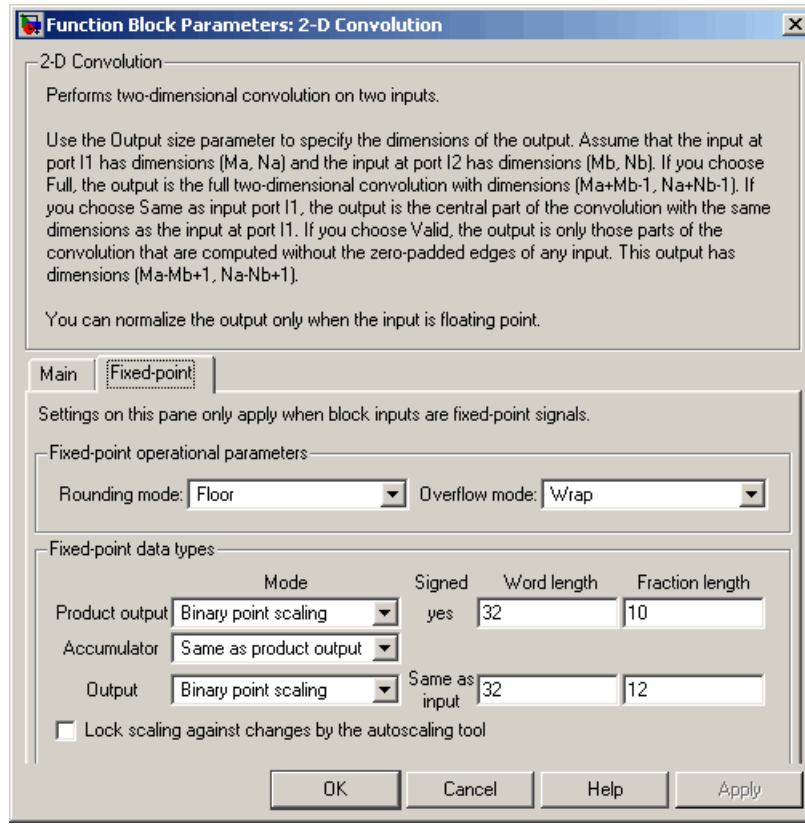
Output size

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.



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 10-9 and "Multiplication Data Types" in the Signal Processing Blockset documentation for illustrations depicting the use of the product output data type in this block:

- When you select `Same as first input`, these characteristics match those of the first input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset 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 10-9 and "Multiplication Data Types" in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block. Note that 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 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 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

See Also

2-D FIR Filter Video and Image Processing Blockset

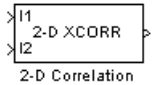
Purpose

Compute 2-D cross-correlation of two input matrices

Library

Statistics

Description



2-D Correlation

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	Scalar, 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 integers • 8-, 16-, 32-bit unsigned integers	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. This block supports a signal represented by a Simulink virtual bus.

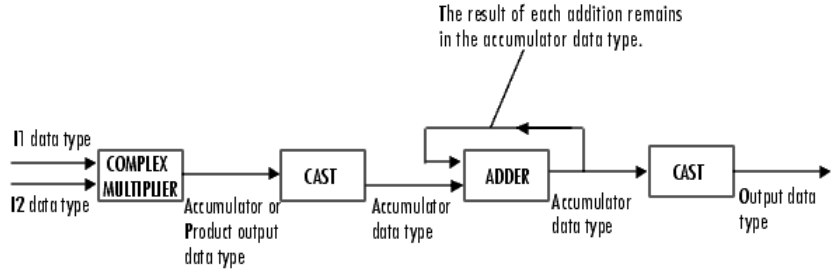
The dimensions of the output are dictated by the **Output size** parameter and the sizes of the inputs at ports I1 and I2. For example, assume that the input at port I1 has dimensions (Ma, Na) and the input at port I2 has dimensions (Mb, Nb). If, for the **Output size** parameter, you choose `Full`, 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 10-23 for more information.

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

Fixed-Point Data Types

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



You can set the product output, accumulator, and output data types in the block mask as discussed in “Dialog Box” on page 10-26.

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

Examples

Example 1

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

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

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

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]
```

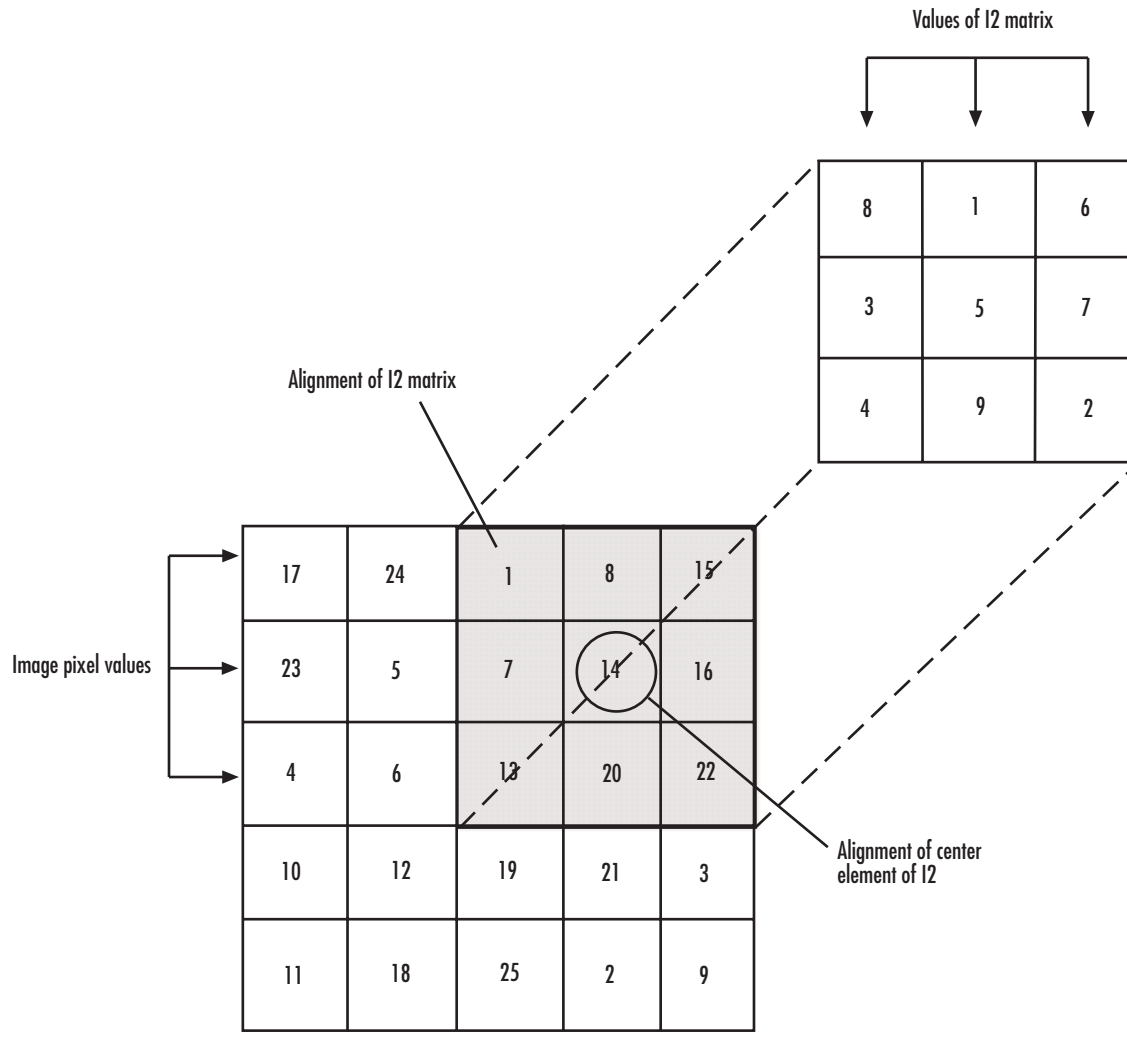
The following figure shows how to compute the (2,4) output element (zero-based indexing) using these steps:

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

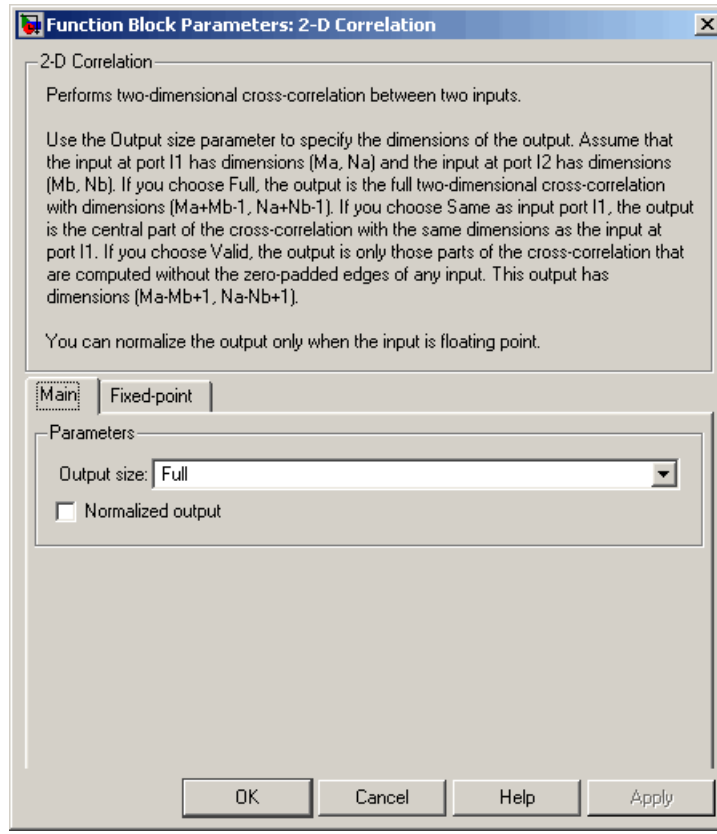


Computing the (2,4) Output of Cross-Correlation

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.



Output size

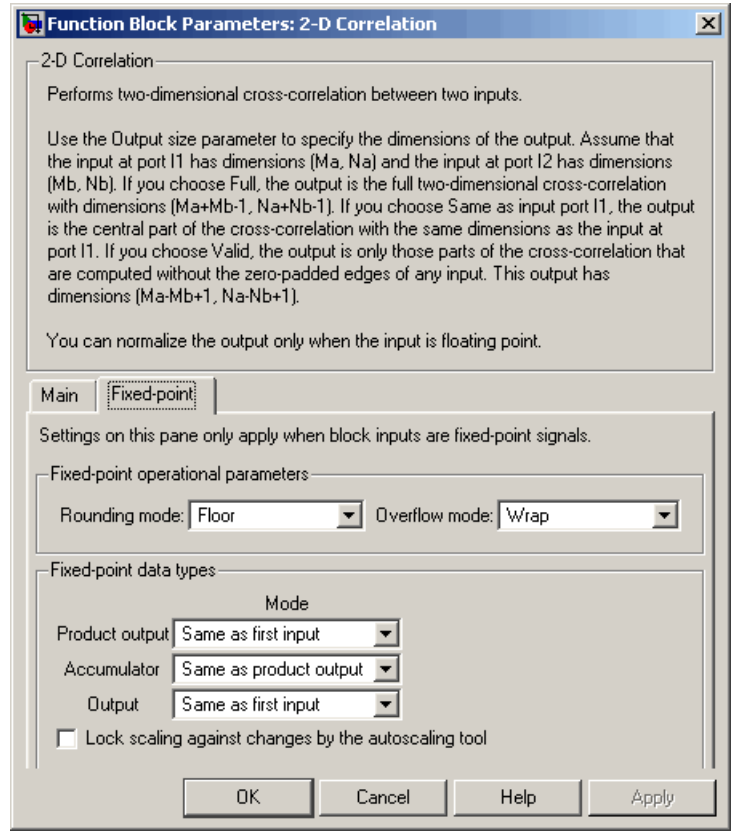
This parameter controls the size of the output scalar, vector, or matrix produced as a result of the cross-correlation between

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.



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 10-20 and "Multiplication Data Types" in the Signal Processing Blockset documentation for illustrations depicting the use of the product output data type in this block:

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

Accumulator
Use this parameter to specify how to designate the accumulator word and fraction lengths. Refer to "Fixed-Point Data Types" on page 10-20 and "Multiplication Data Types" in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block. Note that 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 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 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

See Also

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

Maximum	Signal Processing Blockset
Minimum	Signal Processing Blockset

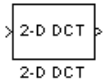
Purpose

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	Scalar, 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 integers • 8-, 16-, 32-bit unsigned integers	No
Output	2-D DCT of the input	Same as Input port	No

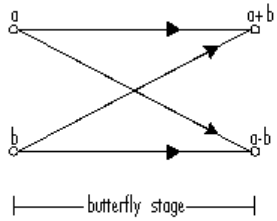
If the data type of the input signal is floating point, the output of the block is the same data type. This block supports a signal represented by a Simulink virtual bus.

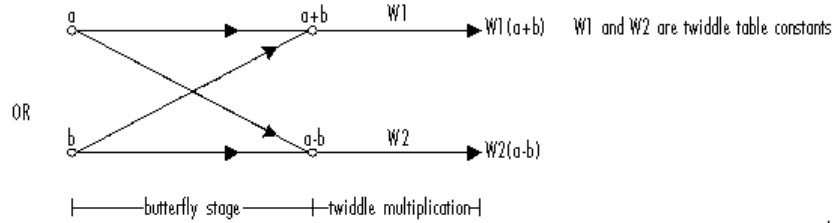
Use the **Sine and cosine computation** parameter to specify how the block computes the sine and cosine terms in the DCT algorithm. If

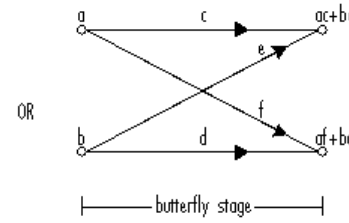
you select `Trigonometric fcn`, the block computes the sine and cosine values during the simulation. If you select `Table lookup`, the block computes and stores the trigonometric values before the simulation starts. In this case, the block requires extra memory.

Fixed-Point Data Types

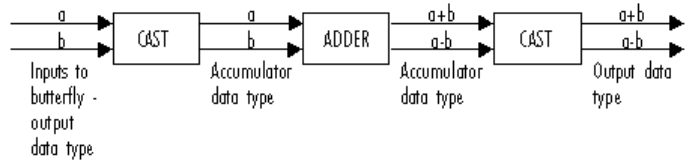
The following diagram shows the data types used in the 2-D DCT block for fixed-point signals. Inputs are first cast to the output data type and stored in the output buffer. Each butterfly stage processes signals in the accumulator data type, with the final output of the butterfly being cast back into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



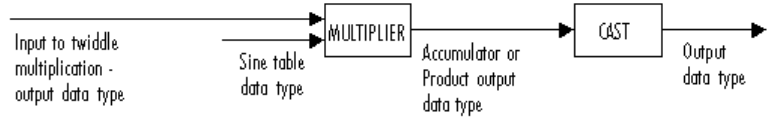




Butterfly Stage Data Types



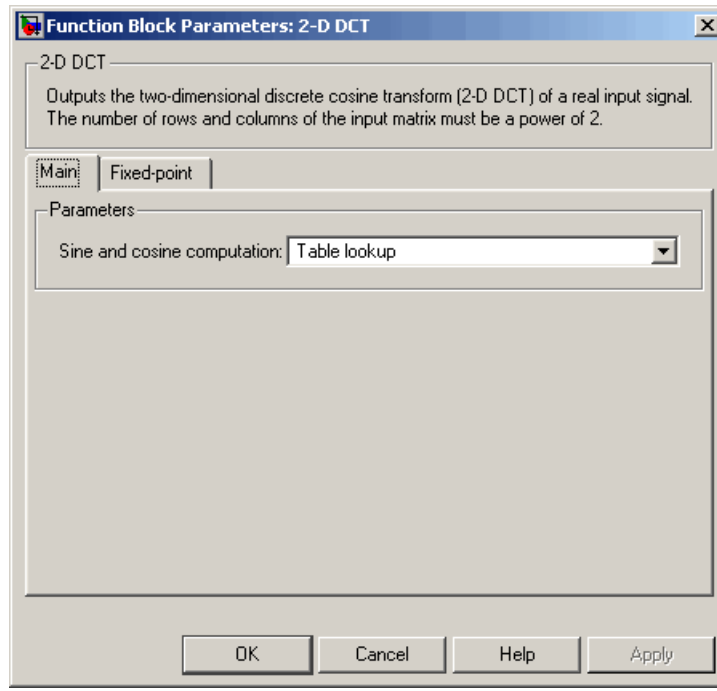
Twiddle Multiplication Data Types



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

Dialog Box

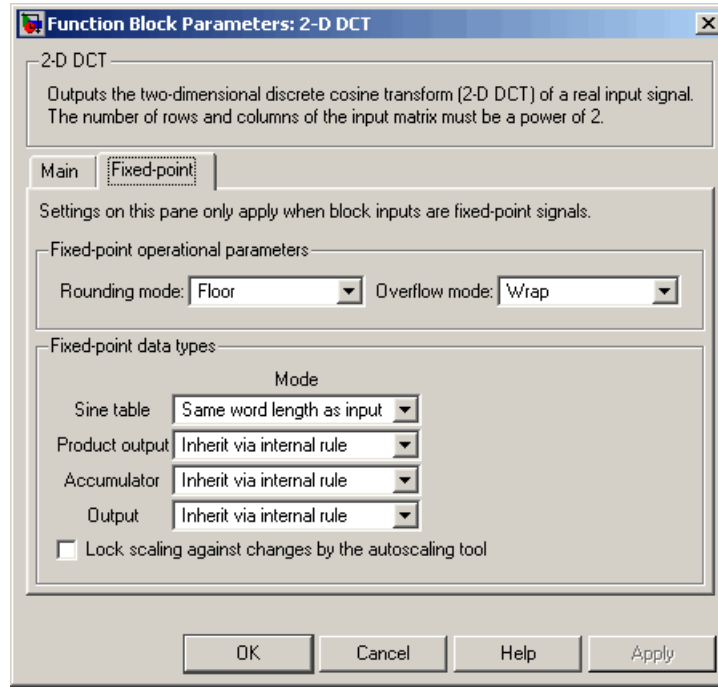
The **Main** pane of the 2-D DCT dialog box appears as shown in the following figure.



Sine and cosine computation

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

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



Rounding mode

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

Overflow mode

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

Sine table

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

- When you select `Same word length as input`, the word length of the sine table values match that of the input to the block.
- When you select `Binary point scaling`, you can enter the word length of the sine table values, in bits.
- When you select `Slope and bias scaling`, you can enter the word length of the sine table values, in bits.

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

Product output

Use this parameter to specify how to designate the product output word and fraction lengths. Refer to “Fixed-Point Data Types” on page 10-44 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:

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

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

- 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 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 10-44 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:

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

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

- 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 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 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

References

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.

Wang, Z. “Fast algorithms for the discrete W transform and for the discrete Fourier transform,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 803-816, Aug. 1984.

See Also

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

2-D FFT

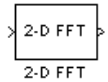
Purpose

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 `Zero 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	Input/Output	Supported Data Types	Complex Values Supported
Input	Scalar, 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 integers • 8-, 16-, 32-bit unsigned integers	Yes
Output	2-D FFT of the input	Same as Input port	Yes

If the data type of the input signal is floating point, the data type of the output signal is the same floating-point data type. Otherwise, the output can be any fixed-point data type. This block supports a signal represented by a Simulink virtual bus.

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 below.

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

Ordering Output Column Entries

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

Note With the 2-D FFT block, linearly ordering the output requires a butterfly operation. So, it might be better to output in bit-reversed order in some situations.

For more information ordering of the output, see “Bit-Reversed Order” on page 10-45. Note that the 2-D FFT block bit-reverses the order of the columns as well as the rows.

Algorithms Used for FFT Computation

Depending on whether the block input is floating-point or fixed-point, real or complex, and whether you want the output in linear or bit-reversed order, the block uses one or more of the following algorithms as summarized in the following tables:

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

Floating-Point Signals

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

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

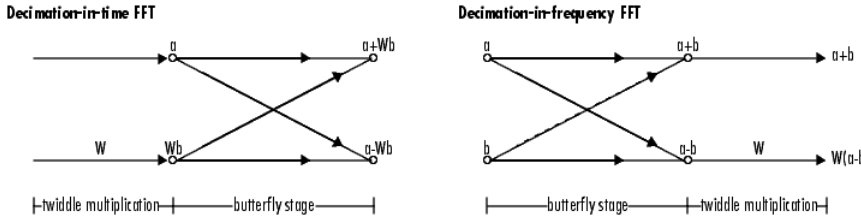
Fixed-Point Signals

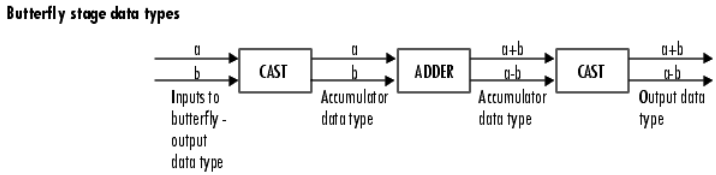
Complexity of Input	Output Ordering	Algorithms Used for FFT Computation
Real or complex	Linear	Butterfly operation and radix-2 DIT
Real or complex	Bit-reversed	Radix-2 DIF

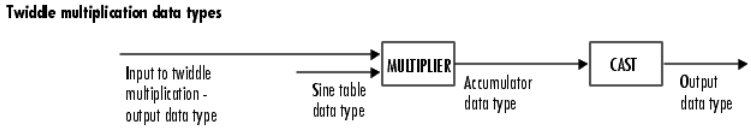
Fixed-Point Data Types

The diagrams below 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 10-48.

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







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

Example

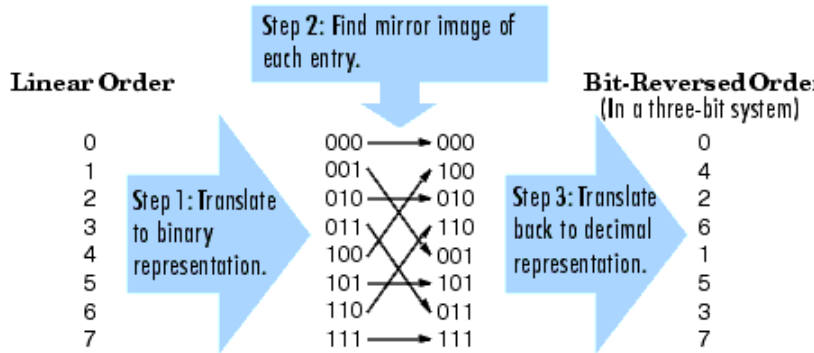
Bit-Reversed Order

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

representation of four, 100. In the diagram below, the row indices are in linear order. To put them in bit-reversed order

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

   The row indices are now in bit-reversed order.



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

Input to FFT block
(must be linear order)

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



Output in linear order

Linearly ordered row and column indices

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

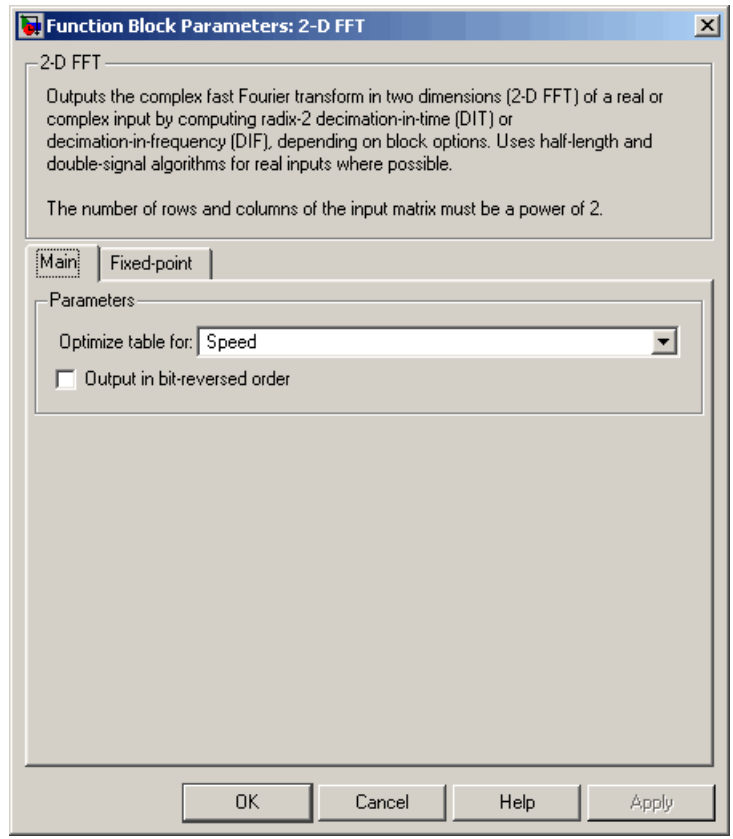
Output in bit-reversed order

Bit-reversed row and column indices

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

Dialog Box

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



Optimize table for

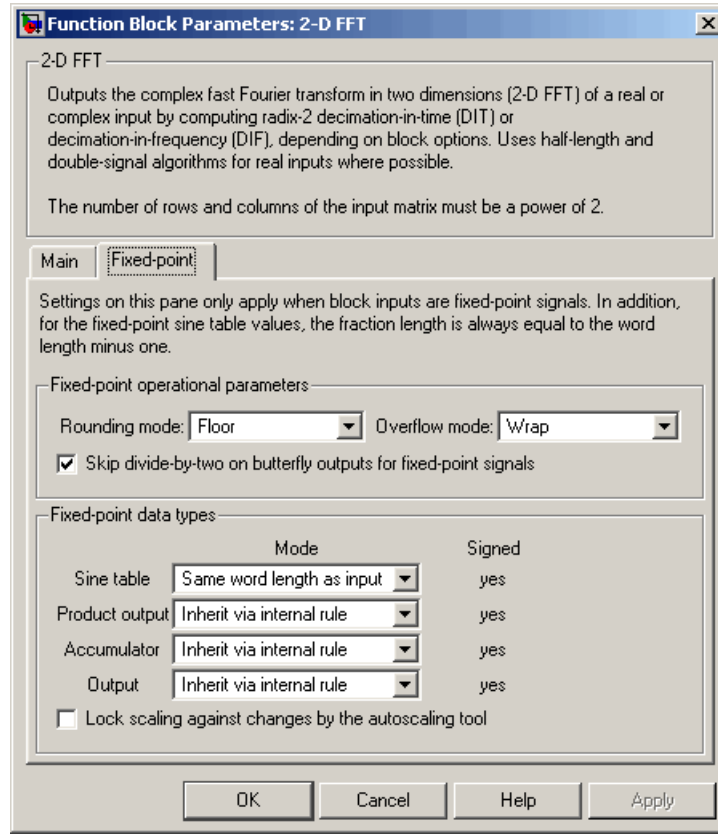
Optimize of the table of twiddle factor values for `Speed` or `Memory`. This parameter must be set to `Speed` for fixed-point signals.

Output in bit-reversed order

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

elements are in bit-reversed order relative to the input ordering. Otherwise, the output column elements are linearly ordered relative to the input ordering. Linearly ordering the output requires extra data sorting manipulation. For more information, see “Bit-Reversed Order” on page 10-45.

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



Rounding mode

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

Overflow mode

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

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

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

Sine table

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

- When you select `Same word length as input`, the word length of the sine table values match that of the input to the block.
- When you select `Binary point scaling`, you can enter the word length of the sine table values, in bits.
- When you select `Slope and bias scaling`, you can enter the word length of the sine table values, in bits.

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

Product output

Use this parameter to specify how to designate the product output word and fraction lengths. Refer to “Fixed-Point Data Types” on page 10-44 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:

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

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

- 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 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 10-44 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:

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

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

- 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 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 is 0.

See Also

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

`bitrevorder`	Signal Processing Toolbox
`fft`	Signal Processing Toolbox
`ifft`	Signal Processing Toolbox

2-D FIR Filter

Purpose

Perform 2-D FIR filtering on input matrix

Library

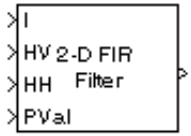
Filtering

Description

The 2-D FIR Filter block filters the input matrix I using the coefficient matrix H or the coefficient vectors HH and HV.



2-D FIR Filter



2-D FIR Filter

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Scalar, 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 integers • 8-, 16-, 32-bit unsigned integers	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 data type of the input is floating point, the output of the block is the same data type. Otherwise, the output can be any fixed-point data type. This block supports a signal represented by a Simulink virtual bus.

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

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

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

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

MAC operations. If you aren't sure whether or not your filter coefficients are separable, use the `isfilterseparable` function.

Here is an example of the function syntax, `[S, HCOL, HROW] = isfilterseparable(H)`. The `isfilterseparable` function takes the filter kernel, `H`, and returns `S`, `HCOL` and `HROW`. Here, `S` is a Boolean variable that is 1 if the filter is separable and 0 if it is not. `HCOL` is a vector of vertical filter coefficients, and `HROW` is a vector of horizontal filter coefficients. Later, you learn how to use these variables in the block mask.

Use the **Coefficient source** parameter to specify how to define your filter coefficients. If you select the **Separable filter coefficients** check box and, for the **Coefficient source** parameter, you select `Specify via dialog`, the **Vertical coefficients (across height)** and **Horizontal coefficients (across width)** parameters appear in the dialog box. You can use these parameters to enter vectors of vertical and horizontal filter coefficients, respectively. You can use the variables `HCOL` and `HROW`, the output of the `isfilterseparable` function, for these parameters. If you select the **Separable filter coefficients** check box and, for the **Coefficient source** parameter, you select `Input port`, ports HV and HH appear on the block. Use these ports to specify vectors of vertical and horizontal filter coefficients. If you clear the **Separable filter coefficients** check box and, for the **Coefficient source** parameter, you select `Specify via dialog`, the **Coefficients** parameter appears in the dialog box. Use this parameter to enter your matrix of filter coefficients. If you clear the **Separable filter coefficients** check box and, for the **Coefficient source** parameter, you select `Input port`, port H appears on the block. Use this port to specify your filter coefficient matrix.

The block outputs the result of the filtering operation at the Output port. The dimensions of the output are dictated by the **Output size** parameter and the sizes of the inputs at ports I and H. For example, assume that the input at port I has dimensions (Mi, Ni) and the input at port H has dimensions (Mh, Nh). If, for the **Output size** parameter, you choose `Full`, the output has dimensions (Mi+Mh-1, Ni+Nh-1). If, for the **Output size** parameter, you choose `Same as input port I`, the output has the same dimensions as the input at port I. If, for the **Output size** parameter, you choose `Valid`, the block filters the input image only where the coefficient matrix fits entirely within it, so no padding is required. The output has dimensions (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 2-D Pad block reference page.

If, for the **Padding options** parameter, you select `Constant`, the **Pad value source** parameter appears in the dialog box. If you select `Specify via dialog`, the **Pad value** parameter appears in the dialog box. Use this parameter to enter the constant value with which to pad your matrix. If, for the **Pad value source** parameter, you select `Input port`, the PVal port appears on the block. Use this port to specify the constant value with which to pad your matrix.

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

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

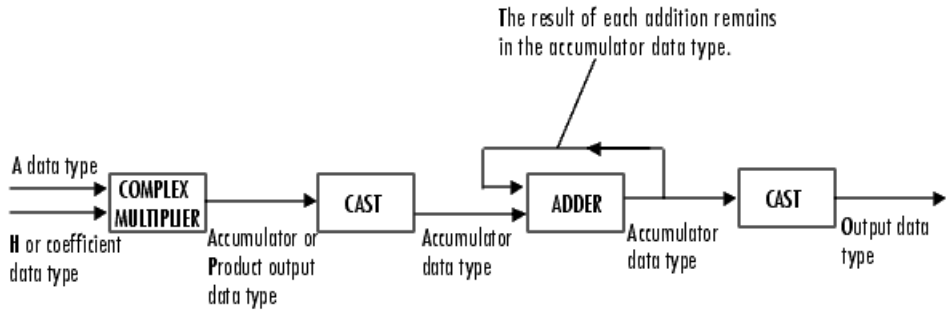
where $0 \le i < Ma + Mh - 1$ and $0 \le j < Na + Nh - 1$. If you select `Correlation` and set the **Output size** parameter to `Full`, the block filters your input using the following algorithm

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

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

Fixed-Point Data Types

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

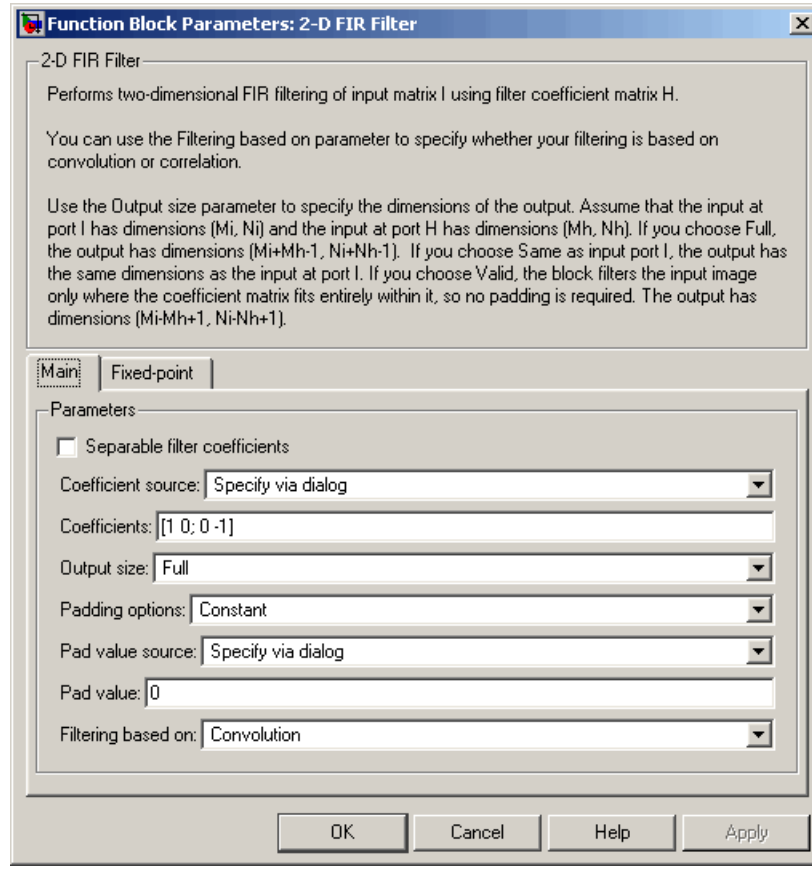


You can set the coefficient, product output, accumulator, and output data types in the block mask as discussed in “Dialog Box” on page 10-59.

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 FIR Filter dialog box appears as shown in the following figure.



Separable filter coefficients

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

Coefficient source

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

Coefficients

Enter your real or complex-valued filter coefficient matrix. This parameter is visible if you clear the **Separable filter coefficients** check box and, for the **Coefficient source** parameter, you select `Specify via dialog`. Tunable.

Vertical coefficients (across height)

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

Horizontal coefficients (across width)

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

Output size

This parameter controls the size of the filtered output. If you choose `Full`, the output has dimensions (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 is visible if, for the

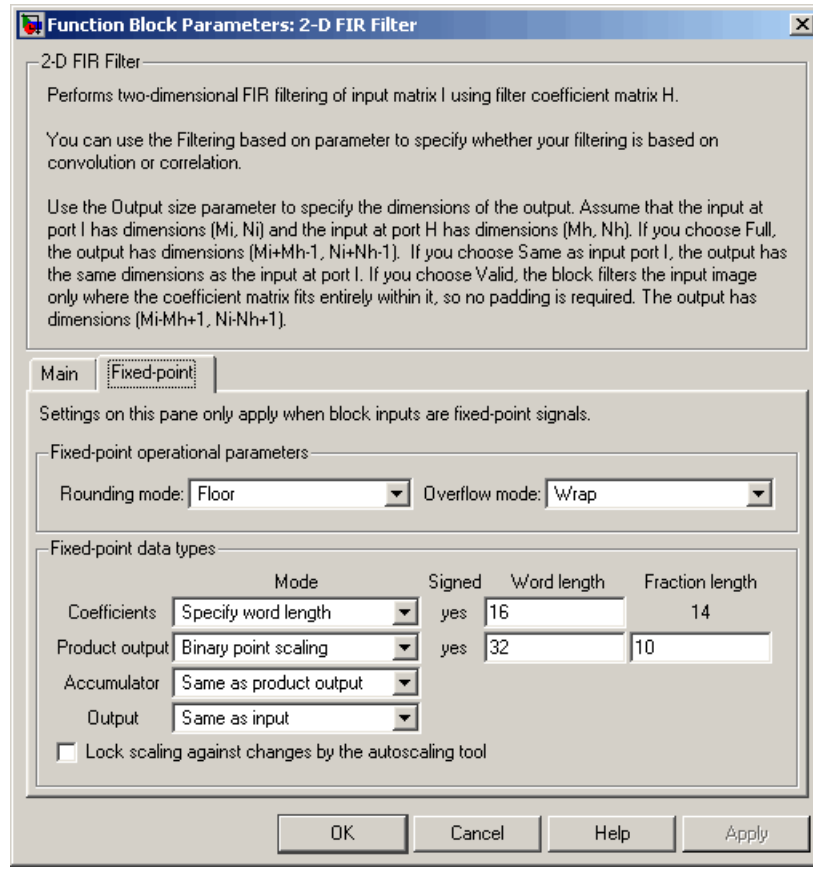
Output size parameter, you select `Full` or `Same as input port I`.

Pad value source
Use this parameter to specify how to define your constant boundary value. Select `Specify via dialog` to enter your value in the block parameters dialog box. Select `Input port` to specify your constant value using the PVal port. This parameter is visible if, for the **Padding options** parameter, you select `Constant`.

Pad value
Enter the constant value with which to pad your matrix. This parameter is visible if, for the **Pad value source** parameter, you select `Specify via dialog`. Tunable.

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

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



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Coefficients

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

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

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

Product output

Use this parameter to specify how to designate the product output word and fraction lengths. Refer to “Fixed-Point Data Types” on page 10-57 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 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 10-57 and "Multiplication Data Types" in the Signal Processing Blockset documentation for illustrations depicting the use of the accumulator data type in this block. Note that the accumulator data type is only used when both inputs to the multiplier are complex:

- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset 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 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

2-D Histogram

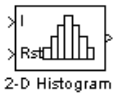
Purpose

Generate histogram of each input matrix

Library

Statistics

Description



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

The output of the 2-D Histogram block is different than the output of the `imhist` function in 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	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the input RGB video stream	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	Yes
Rst	Signal that triggers a reset event	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	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. This block supports a signal represented by a Simulink virtual bus.

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.

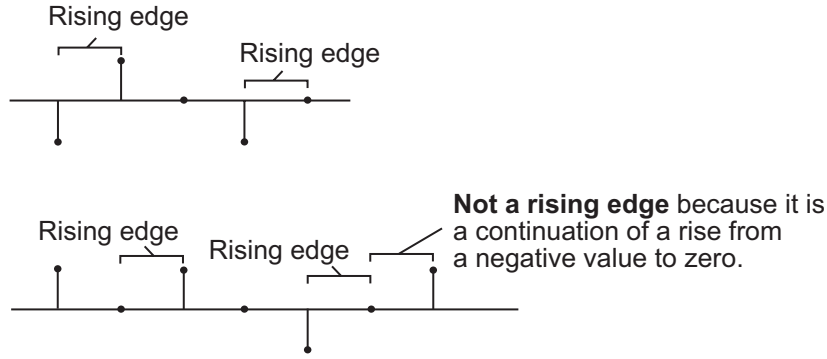
Resetting the Running Histogram

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.

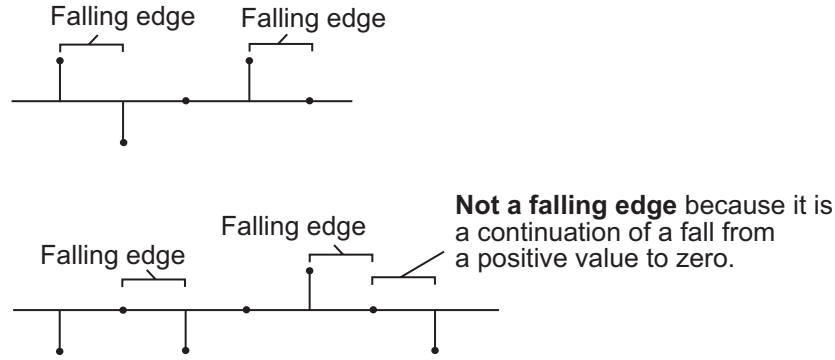
To enable the Rst port, select the **Reset port** parameter. You specify the reset event in the **Trigger type** parameter, and can be one of the following:

- `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)



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



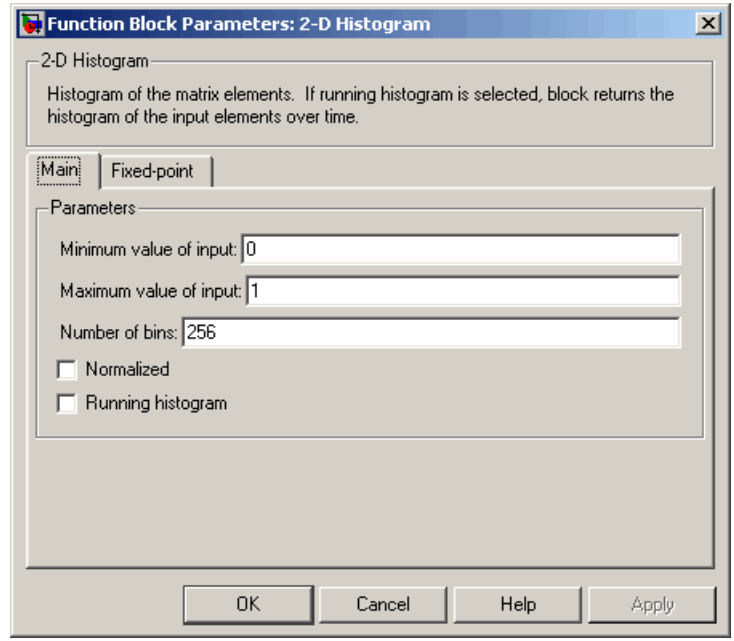
- `Either edge` -- Triggers a reset operation when the `Rst` input is a `Rising edge` or `Falling edge` (as described previously)

- `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 The Configuration Parameters Dialog Box in the Simulink documentation.

Dialog Box

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



Minimum value of input

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

Maximum value of input

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

Number of bins

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

Normalized

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

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

Running histogram

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

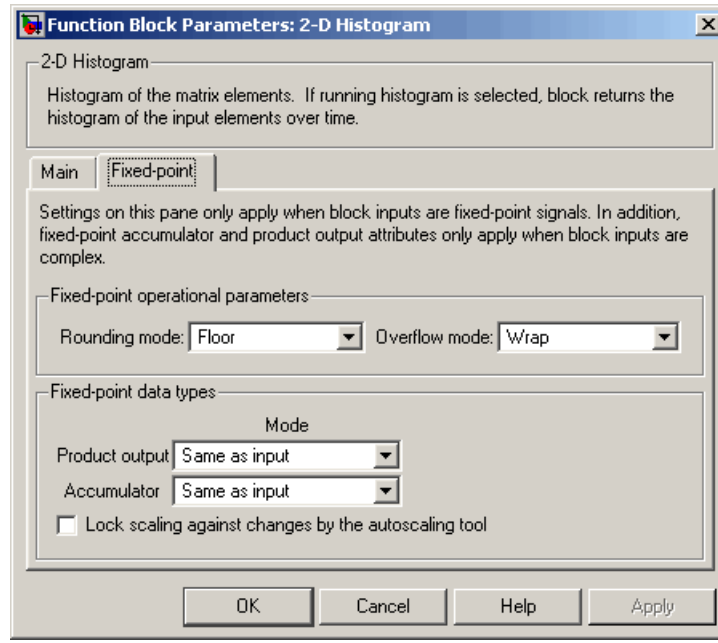
Reset port

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

Trigger type

The type of event that resets the running histogram. For more information, see "Resetting the Running Histogram" on page 10-69. This parameter is enabled only when you set the **Reset port** parameter.

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



Note The fixed-point parameters listed below 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

See Also

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

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

2-D IDCT

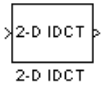
Purpose

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.

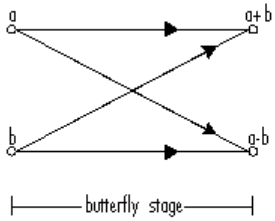
Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Scalar, vector, or matrix of intensity values or a scalar, vector, or matrix that represents one plane of the RGB video stream	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	No
Output	2-D IDCT of the input	Same as Input port	No

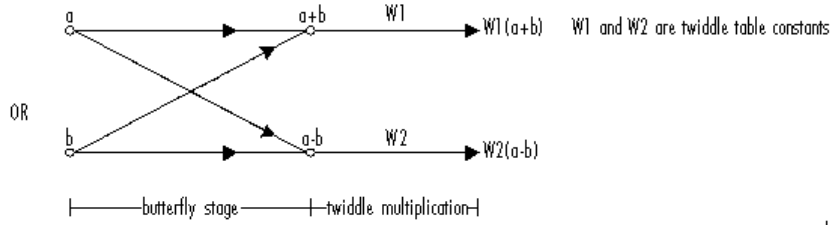
If the data type of the input signal is floating point, the output of the block is the same data type. This block supports a signal represented by a Simulink virtual bus.

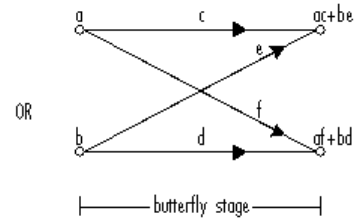
Use the **Sine and cosine computation** parameter to specify how the block computes the sine and cosine terms in the IDCT algorithm. If you select `Trigonometric fcn`, the block computes the sine and cosine values during the simulation. If you select `Table lookup`, the block computes and stores the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

Fixed-Point Data Types

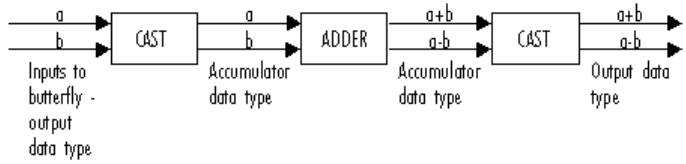
The following diagram shows the data types used in the 2-D IDCT block for fixed-point signals. Inputs are first cast to the output data type and stored in the output buffer. Each butterfly stage processes signals in the accumulator data type, with the final output of the butterfly being cast back into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



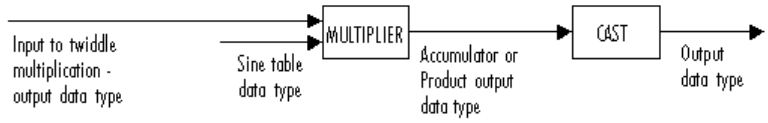




Butterfly Stage Data Types



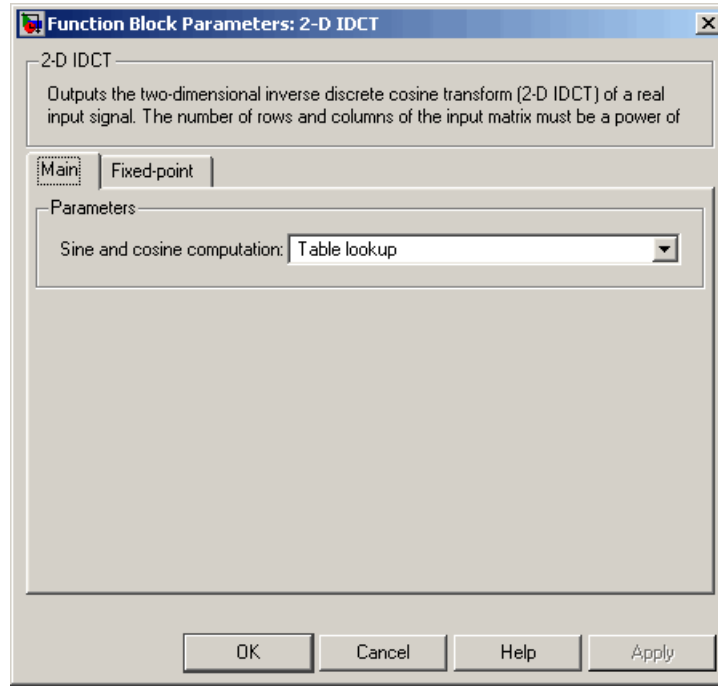
Twiddle Multiplication Data Types



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

Dialog Box

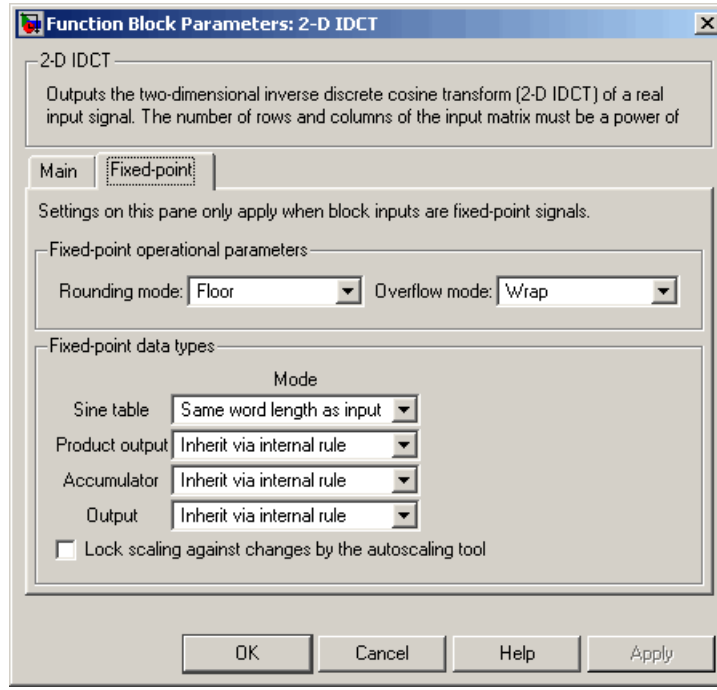
The **Main** pane of the 2-D IDCT dialog box appears as shown in the following figure.



Sine and cosine computation

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

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



Rounding mode

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

Overflow mode

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

Sine table

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

- When you select `Same word length as input`, the word length of the sine table values match that of the input to the block.
- When you select `Binary point scaling`, you can enter the word length of the sine table values, in bits.
- When you select `Slope and bias scaling`, you can enter the word length of the sine table values, in bits.

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

Product output

Use this parameter to specify how to designate the product output word and fraction lengths. Refer to “Fixed-Point Data Types” on page 10-77 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:

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

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

- 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 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 10-77 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 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 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

References

[1] Chen, W.H, C.H. Smith, and S.C. Fralick, "A fast computational algorithm for the discrete cosine transform,"*IEEE Trans. Commun.*, vol. COM-25, pp. 1004-1009. 1977.

[2] Wang, Z. “Fast algorithms for the discrete W transform and for the discrete Fourier transform,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 803-816, Aug. 1984.

See Also

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

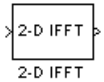
Purpose

Compute 2-D IFFT of input

Library

Transforms

Description



The 2-D IFFT block computes the inverse fast Fourier transform (IFFT) of an M-by-N input matrix in two steps. First it computes the one-dimensional IFFT along one dimension (row or column). Then it computes the IFFT of the output of the first step along the other dimension (column or row). The dimensions of the input matrix, M and N, must be powers of two. To work with other input sizes, use the `Zero 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	Input/Output	Supported Data Types	Complex Values Supported
Input	Scalar, 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 integers • 8-, 16-, 32-bit unsigned integers	Yes
Output	2-D IFFT of the input	Same as Input port	Yes

If the data type of the input signal is floating point, the data type of the output signal is the same floating-point data type. Otherwise, the output can be any fixed-point data type. This block supports a signal represented by a Simulink virtual bus.

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 below.

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

Input Order

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

For more information ordering of the output, see “Bit-Reversed Order” on page 10-45. Note that the 2-D FFT block bit-reverses the order of the columns as well as the rows.

Conjugate Symmetric Input

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

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

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

Scaled Output

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

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

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

Algorithms Used for IFFT Computation

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

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

For floating-point signals:

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

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

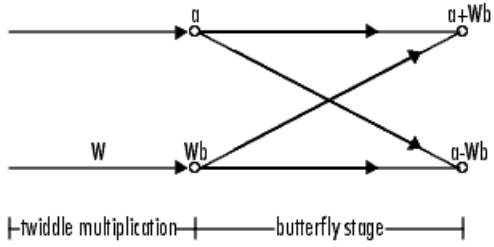
Note that the **Input is conjugate symmetric** parameter cannot be used for fixed-point signals.

Fixed-Point Data Types

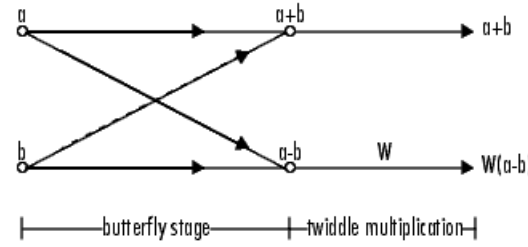
The diagrams below show the data types used in the IFFT block for fixed-point signals. You can set the sine table, accumulator, product output, and output data types displayed in the diagrams in the IFFT dialog box as discussed in “Dialog Box” on page 10-92.

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

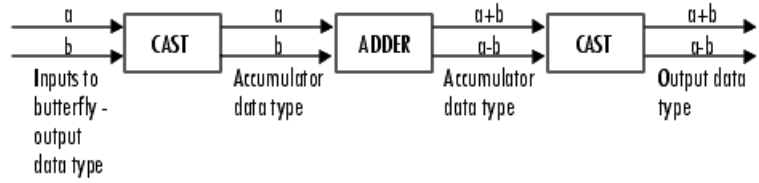
Decimation-in-time IFFT



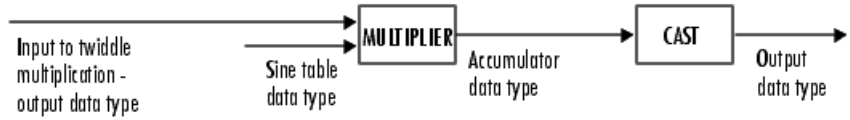
Decimation-in-frequency IFFT



Butterfly stage data types



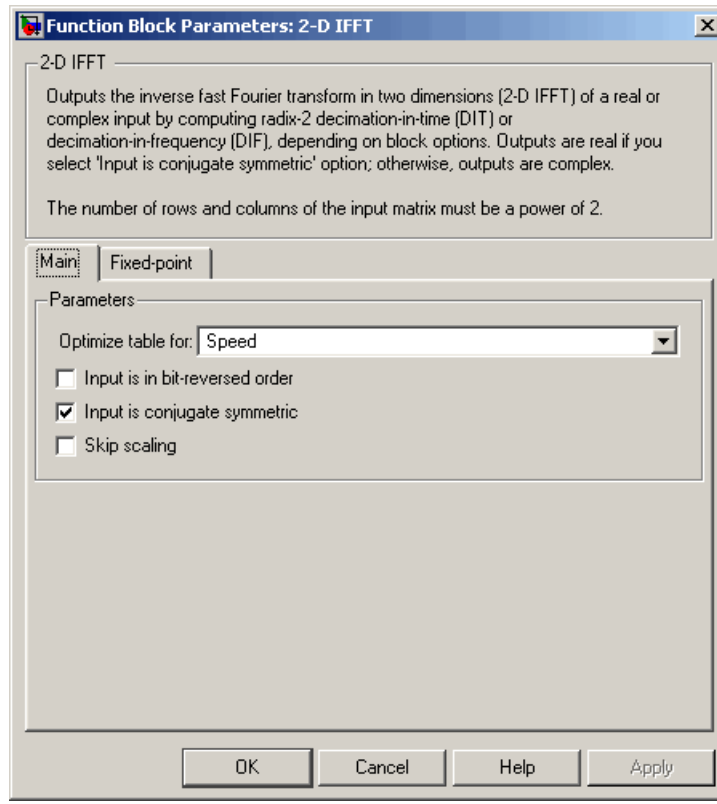
Twiddle multiplication data types



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

Dialog Box

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



Optimize table for

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

Input is in bit-reversed order

Designate the order of the input channel elements. Select when the input is in bit-reversed order, and clear when the input is in

linear order. The block yields invalid outputs when you do not set this parameter correctly. See “Input Order” on page 10-87.

Input is conjugate symmetric

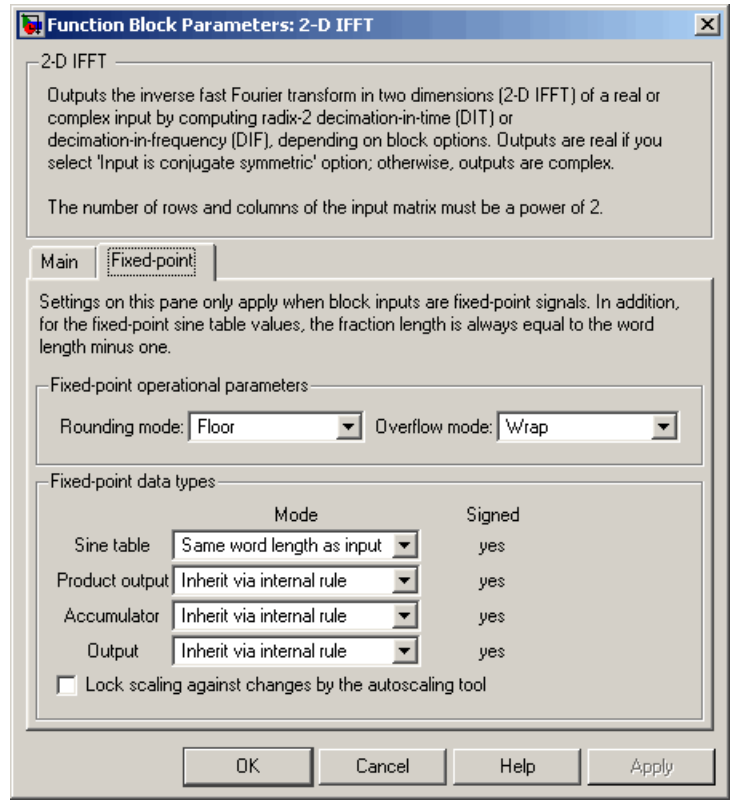
Select when the input to the block is both floating point and conjugate symmetric, and you want real-valued outputs. The block output is invalid when you set this parameter when the input is not conjugate symmetric. This parameter cannot be used for fixed-point signals.

Skip scaling

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

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

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



Rounding mode

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

Overflow mode

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

Sine table

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

- When you select `Same word length as input`, the word length of the sine table values match that of the input to the block.
- When you select `Binary point scaling`, you can enter the word length of the sine table values, in bits.
- When you select `Slope and bias scaling`, you can enter the word length of the sine table values, in bits.

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

Product output

Use this parameter to specify how to designate the product output word and fraction lengths. Refer to "Fixed-Point Data Types" on page 10-89 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 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 10-89 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 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 is 0.

See Also

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

2-D Maximum (Obsolete)

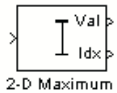
Purpose

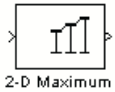
Find maximum values in an input or sequence of inputs

Library

`vipobslib`

Description





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

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

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the input RGB video stream	• 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 integers • 8-, 16-, 32-bit unsigned integers	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

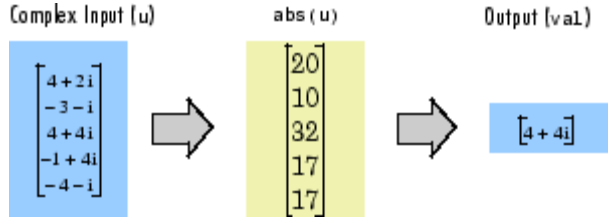
When **Mode** is set to `Value`, the block computes the maximum value in each column of the *M*-by-*N* input matrix *u* independently at each sample time.

```
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 below. 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 jth column of all inputs since the last reset, up to and including element u_{ij} of the current input.

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

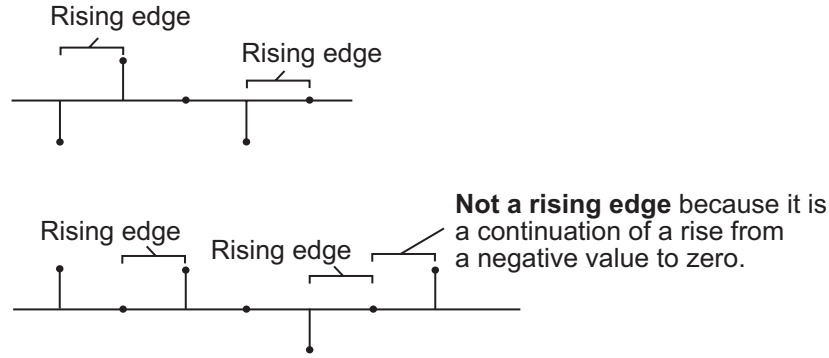
Resetting the Running Maximum

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

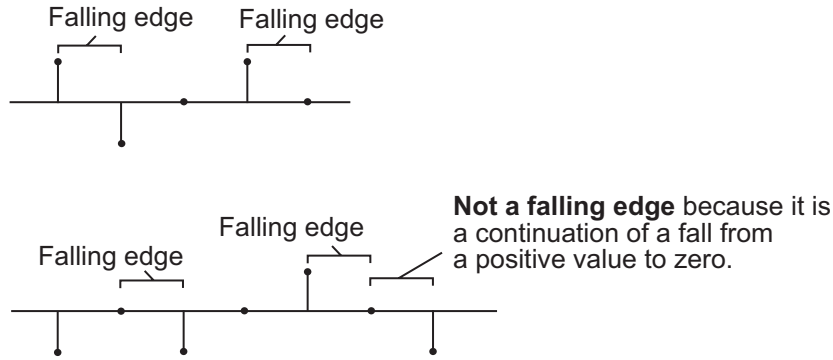
For sample-based inputs, a reset event causes the running maximum for each channel to be initialized to the value in the corresponding channel of the current input. For frame-based inputs, a reset event causes the running maximum for each channel to be initialized to the earliest value in each channel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

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

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



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



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

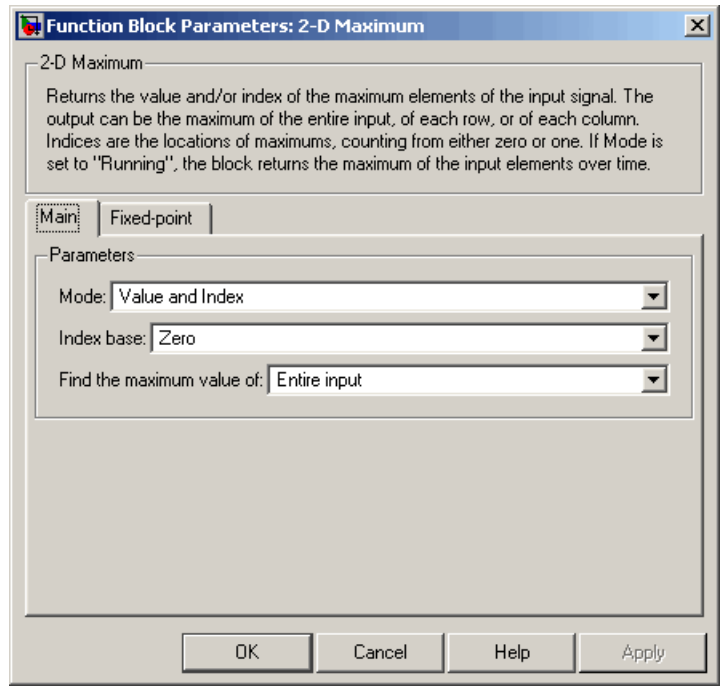
Note When running simulations in the Simulink `MultiTasking` mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see The 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 10-99. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ignored for other types of inputs.

Dialog Box

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



Mode

Specify the block's mode of operation:

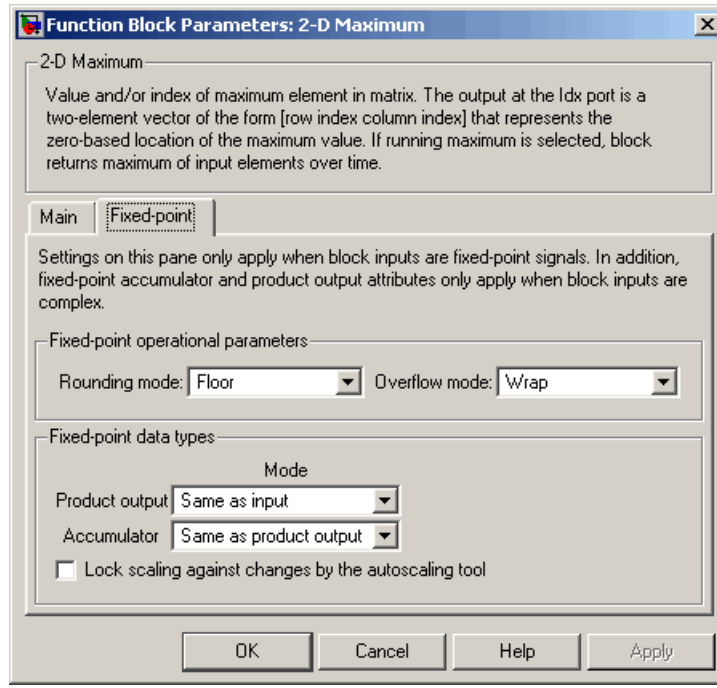
- `Value` — Output the maximum value of each input matrix
- `Index` — Output the zero-based index location of the maximum value
- `Value and Index` — Output both the value and the index location
- `Running` — Track the maximum value of the input sequence over time

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

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

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

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



Note The parameters on the **Fixed-point** pane are only used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before a comparison is made, as described in “Value Mode” on page 10-99. 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

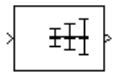
Purpose

Find mean value of each input matrix

Library

Statistics

Description



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

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

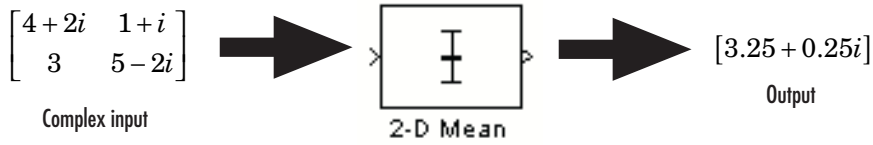
Port	Input/Output	Supported Data Types	Complex Values Supported
Output/Out	Without ROI processing — Mean of each M-by-N input matrix or the mean for each element of a series of M-by-N inputs. With ROI processing — Vector of separate statistical values for each ROI or a scalar value that represents the statistical value for all specified ROIs.	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers	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 below.



Running Operation

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

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

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

and

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

The block computes the mean corresponding to each element,

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

and outputs

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

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

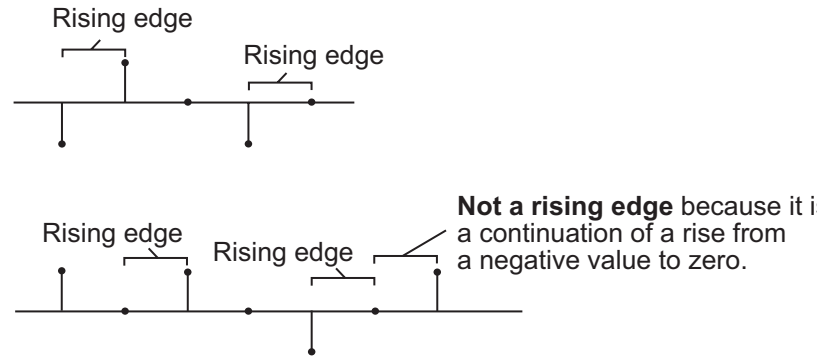
Resetting the Running Mean

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

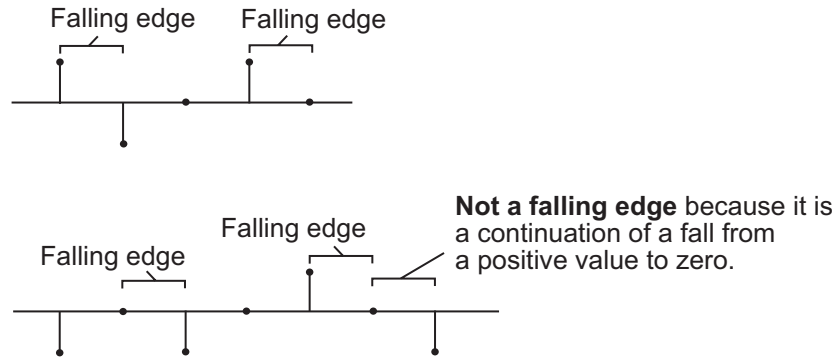
When the block is reset, the running mean associated with each element is initialized to the value in the corresponding location of the current input.

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

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



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



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

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

ROI Processing

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

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

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.

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

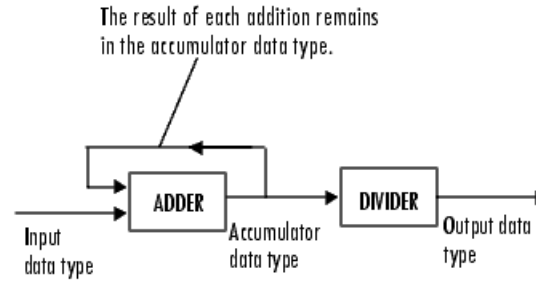
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

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

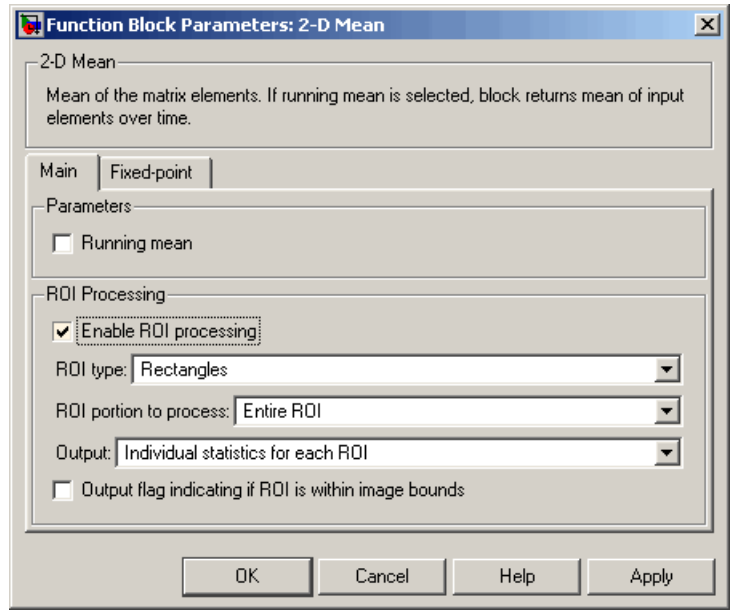
The following diagram shows the data types used in the 2-D Mean block for fixed-point signals.



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

Dialog Box

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



Running mean

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

Reset port

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

Enable ROI processing

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

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

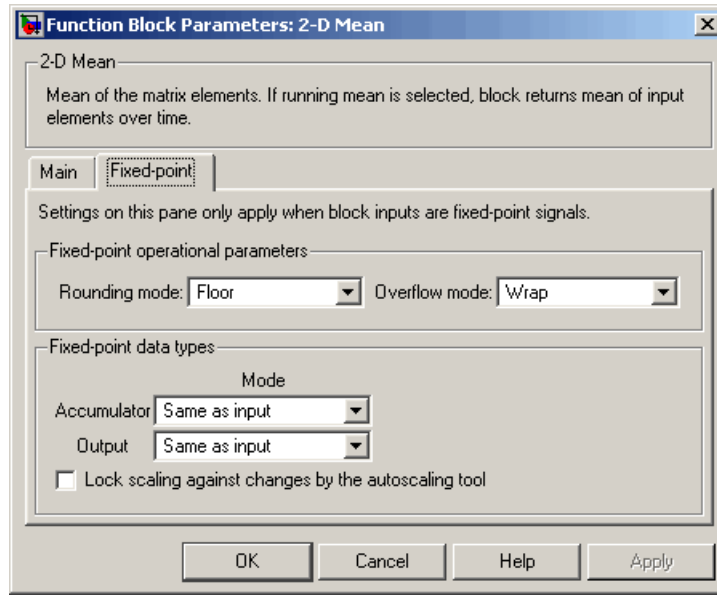
ROI portion to process
Specify whether you want to calculate the statistical value for the entire ROI or just the ROI perimeter. This parameter is only visible if, for the **ROI type** parameter, you specify `Rectangles`.

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

Output flag indicating if ROI is within image bounds
If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in "ROI Processing" on page 10-114. 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 10-114. This parameter is visible if, for the **ROI type** parameter, you select `Label matrix`.

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



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Accumulator

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

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

Output

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

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

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

See Also

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

2-D Median

Purpose

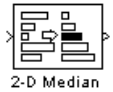
Find median value of each input matrix

Library

Statistics

Description

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



Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the input RGB video stream	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-, and 128-bit signed integers • 8-, 16-, 32-, and 128-bit unsigned integers	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. This block supports a signal represented by a Simulink virtual bus.

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 10-124. Not all these fixed-point parameters are applicable for all types of fixed-point inputs. The following table shows when each kind of data type and scaling is used.

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

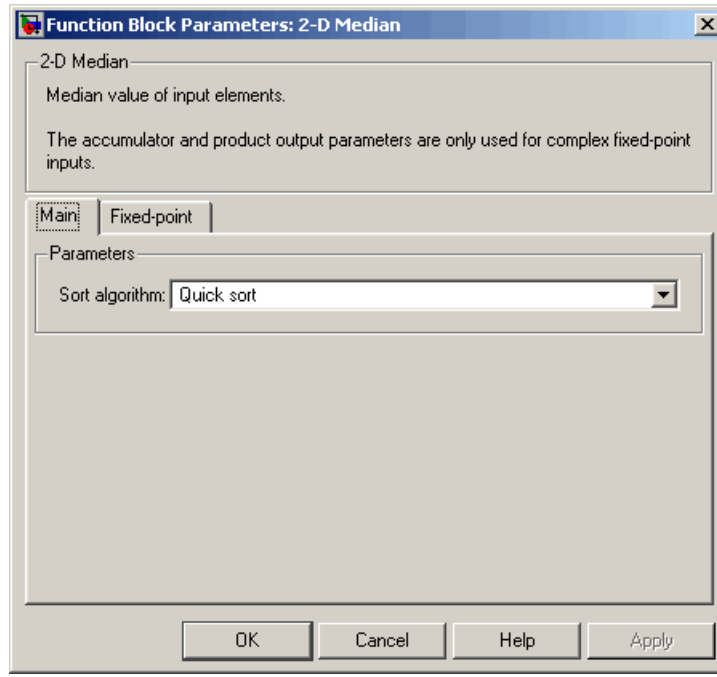
The accumulator and output data types and scalings are used for fixed-point signals when `M` is even. The result of the sum performed while calculating the average of the two central rows of the input matrix is stored in the accumulator data type and scaling. The total result of the average is then put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

The accumulator and product output parameters are used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before the input elements are sorted. The results of the squares of the real and imaginary parts are placed into the product output data type and scaling. The result of the sum of the squares is placed into the accumulator data type and scaling.

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

Dialog Box

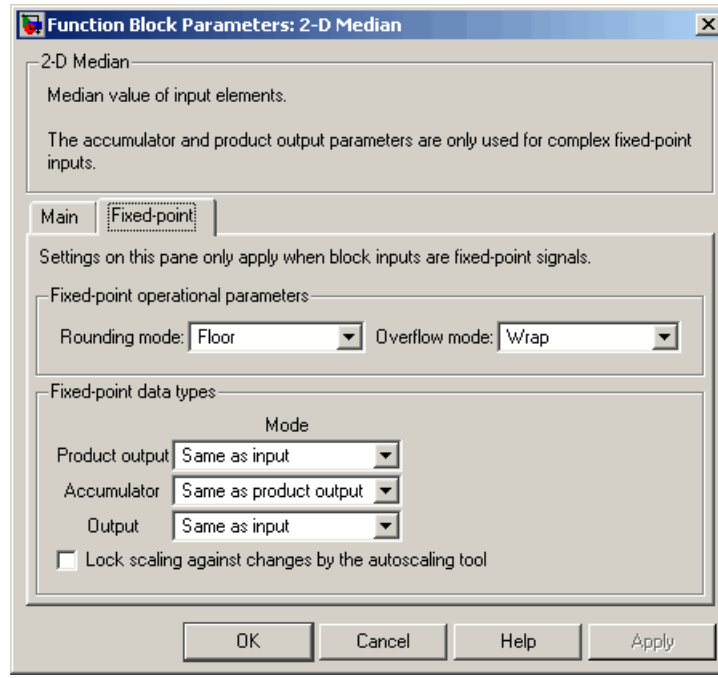
The **Main** pane of the 2-D Median dialog box appears as shown in the following figure.



Sort algorithm

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

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



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 listed below are only used in certain cases. Refer to “Fixed-Point Data Types” on page 10-123 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

See Also

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

2-D Minimum (Obsolete)

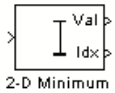
Purpose

Find minimum values in an input or sequence of inputs

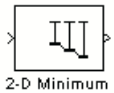
Library

`vipobslib`

Description



2-D Minimum


2-D Minimum

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

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

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

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the input RGB video stream	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	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

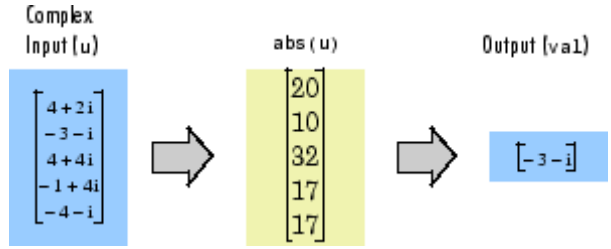
When **Mode** is set to `Value`, the block computes the minimum value in each column of the *M*-by-*N* input matrix *u* independently at each sample time.

```
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 below. 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 jth column of all inputs since the last reset, up to and including element u_{ij} of the current input.

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

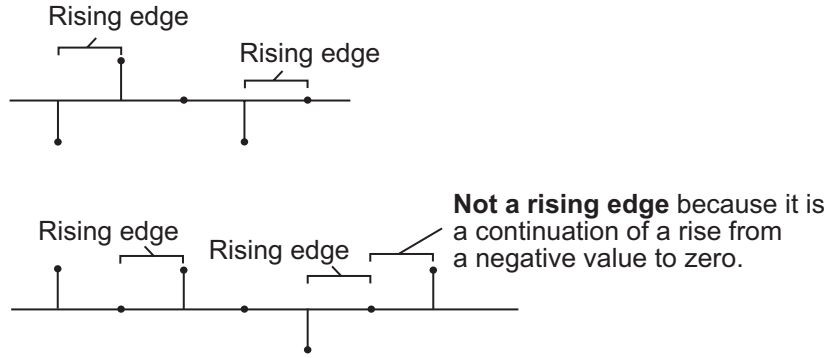
Resetting the Running Minimum

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

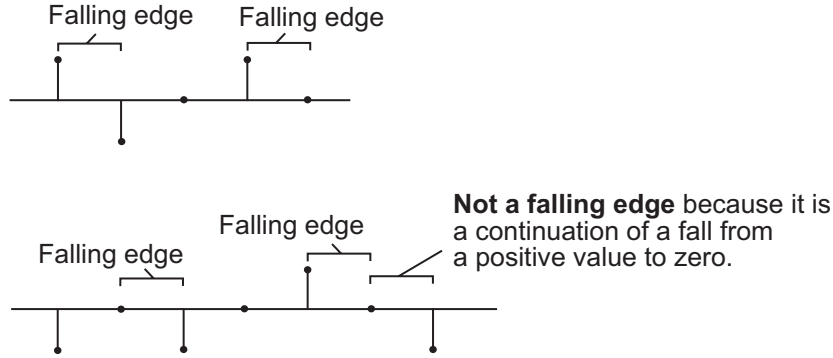
When the block is reset for sample-based inputs, the running minimum for each channel is initialized to the value in the corresponding channel of the current input. For frame-based inputs, the running minimum for each channel is initialized to the earliest value in each channel of the current input.

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

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



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



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

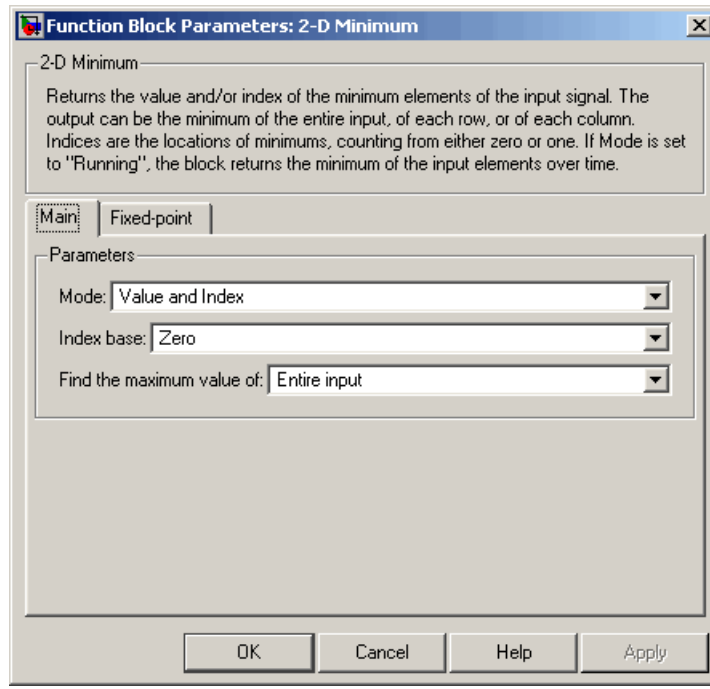
Note When running simulations in the Simulink `MultiTasking` mode, reset signals have a one-sample latency. Therefore, when the block detects a reset event, there is a one-sample delay at the reset port rate before the block applies the reset. For more information on latency and the Simulink tasking modes, see The 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 10-129. The results of the squares of the real and imaginary parts are placed into the product output data type. The result of the sum of the squares is placed into the accumulator data type. These parameters are ignored for other types of inputs.

Dialog Box

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



Mode

Specify the block's mode of operation:

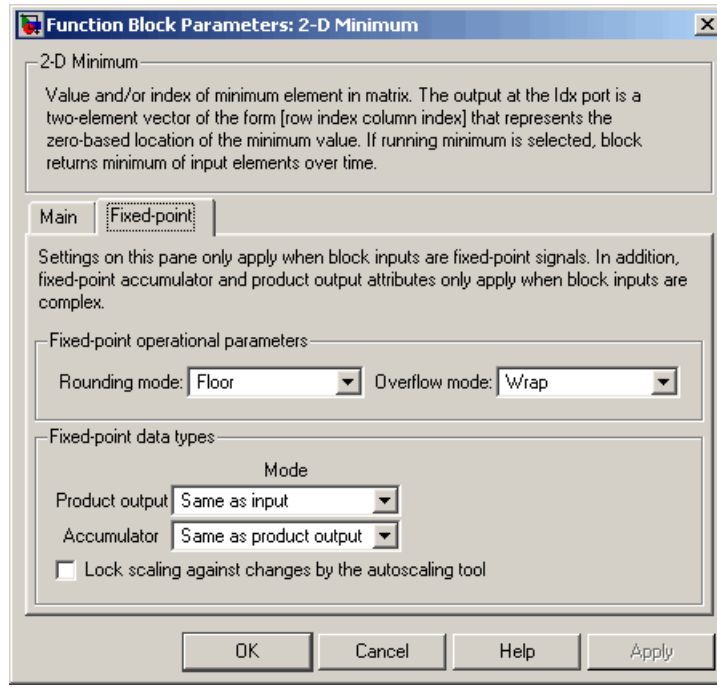
- `Value` — Output the minimum value of each input matrix
- `Index` — Output the zero-based index location of the minimum value
- `Value and Index` — Output both the value and the index location
- `Running` — Track the minimum value of the input sequence over time

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

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

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

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



Note The parameters on the **Fixed-point** pane are only used for complex fixed-point inputs. The sum of the squares of the real and imaginary parts of such an input are formed before a comparison is made, as described in “Value Mode” on page 10-129. 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

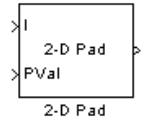
Purpose

Pad matrix along its rows and columns

Library

Utilities

Description



The 2-D Pad block expands the dimensions of a matrix by padding its rows and/or columns at the beginning and/or end with values of your choice.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	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 • Boolean • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	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. This block supports a signal represented by a Simulink virtual bus.

Use the **Method** parameter to specify how you pad the 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`.

If, for the **Method** parameter, you select `Constant`, the **Pad value source** and **Value** parameters appear on the dialog. If, for the **Pad value source** parameter, you select `Input port`, the PVal port appears on the block. Use this port to specify the constant value with which to pad your matrix. If, for the **Pad value source** parameter, you select `Specify via dialog`, the **Value** parameter appears in the dialog box. Enter the constant value with which to pad your matrix.

If, for the **Specify** parameter, 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.

The **Pad rows at** parameter controls the padding at the left and/or right side of the input matrix. If you select `Left`, the block adds additional columns to the left side of the matrix. If you select `Right`, the block adds additional columns to the right side of the matrix. If you select `Both left and right`, the block adds additional columns to the left and right side of the matrix. If you select `No padding`, the block does not change the number of columns of the input matrix.

If, for the **Specify** parameter, you select `Pad size`, the **Pad size along rows** parameter appears in the dialog box. Use this 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 and/or right side of your input matrix. If, for the **Pad rows at** parameter, you selected `Both left and right`, you can also enter a two element vector. The left element controls the number of columns the block adds to the left side of the matrix; the right element controls the number of columns the block adds to the right side of the matrix.

If, for the **Specify** parameter, you select `Output size`, the **Number of output columns** parameter appears in the dialog box. Use this

parameter to enter a scalar value that represents the total number of output columns.

The **Pad columns at** parameter controls the padding at the top and/or bottom of the input matrix. If you select `Top`, the block adds additional rows at the top of the input matrix. If you select `Bottom`, the block adds additional rows at the bottom of the matrix. If you select `Both top and bottom`, the block adds additional rows at the top and bottom of the matrix. If you select `No padding`, the block does not change the number of rows of the input matrix.

If, for the **Specify** parameter, you select `Pad size`, the **Pad size along columns** parameter appears in the dialog box. Use this 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 and/or bottom of your input matrix. If, for the **Pad columns at** parameter, you selected `Both top and bottom`, you can also enter a two element vector. The left element controls the number of rows the block adds to the top of the matrix; the right element controls the number of rows the block adds to the bottom of the matrix.

If, for the **Specify** parameter, you select `Output size`, the **Number of output rows** parameter appears in the dialog box. Use this parameter to enter a scalar value that represents the total number of output rows.

Examples

Example 1

Suppose you want to pad the rows of your input matrix with three initial values equal to 0 and your input matrix is defined as

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

follows:

Set the 2-D Pad block parameters as follows:

- **Method** = `Constant`

- **Pad value source** = `Specify via dialog`
- **Value** = `0`
- **Specify** = `Output size`
- **Pad rows at** = `Left`
- **Number of output columns** = `6`
- **Pad columns at** = `No padding`

The 2-D Pad block outputs the following matrix:

$$\begin{bmatrix} 0 & 0 & 0 & a_{00} & a_{01} & a_{02} \\ 0 & 0 & 0 & a_{10} & a_{11} & a_{12} \\ 0 & 0 & 0 & a_{20} & a_{21} & a_{22} \end{bmatrix}$$

Example 2

Suppose you want to pad your input matrix with its border values, and your input matrix is defined as

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

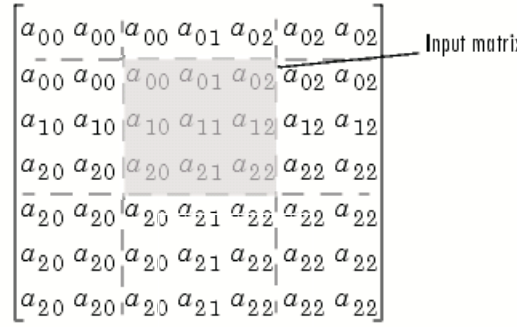
follows:

Set the 2-D Pad block parameters as follows:

- **Method** = `Replicate`
- **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 2-D Pad block outputs the following matrix:



The border values of the input matrix are replicated on the top, bottom, left, and right of the input matrix 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 matrix.

Example 3

Suppose you want to pad your input matrix using its mirror image, and your input matrix is defined as

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

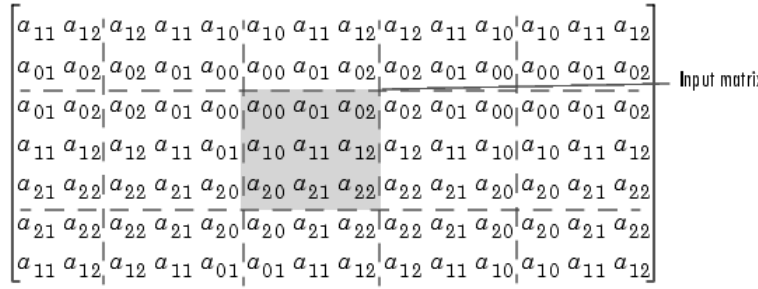
follows:

Set the 2-D Pad block parameters as follows:

- **Method** = `Symmetric`

- **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 2-D Pad block outputs the following 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 matrix.

Example 4

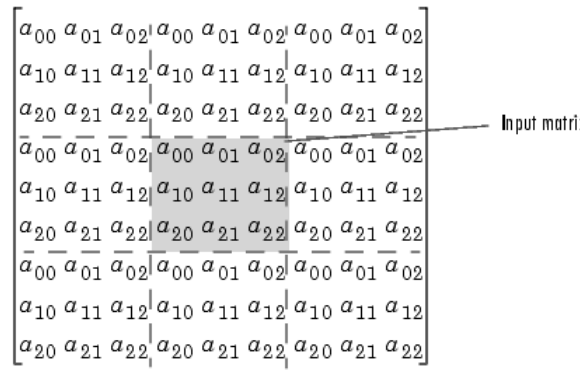
Suppose you want to pad your input matrix using a circular repetition of its values. Your input matrix is defined as

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

Set the 2-D Pad block parameters as follows:

- **Method** = `Circular`
- **Pad rows at** = `Both left and right`
- **Pad size along rows** = `3`
- **Pad columns at** = `Both top and bottom`
- **Pad size along columns** = `3`

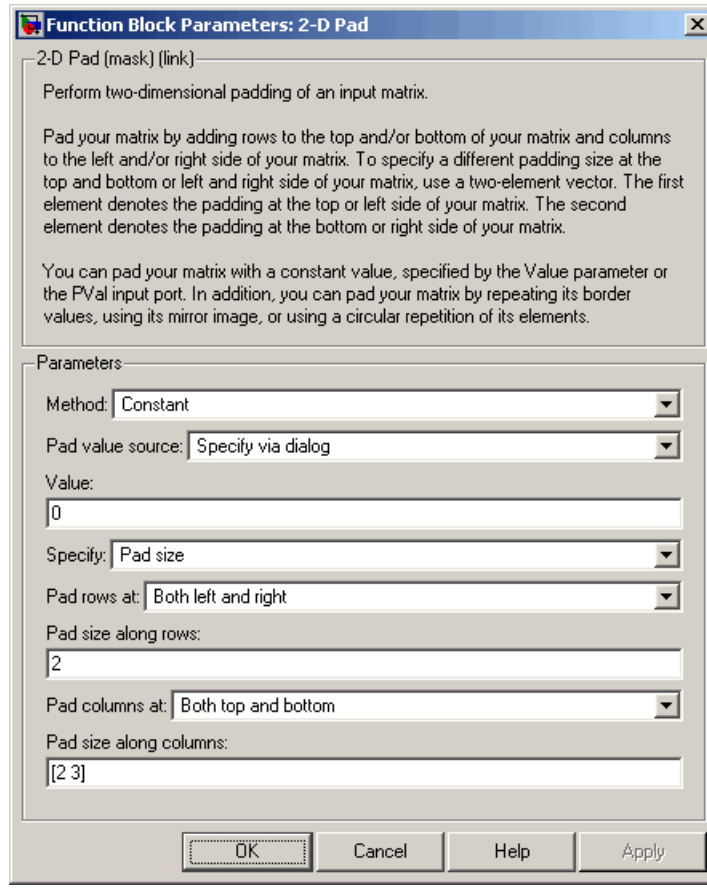
The 2-D Pad block outputs the following matrix:



The block repeats the values of the input matrix in a circular pattern to populate the 9-by-9 output matrix.

Dialog Box

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



Method

Specify how you want the block to pad your matrix. To pad your matrix with a constant value, select `Constant`. To pad your input matrix with 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`.

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 matrix. If you select `Specify via dialog`, the **Value** parameter becomes available. This parameter is visible if, for the **Method** parameter, you select `Constant`.

Value

Enter the constant value with which to pad your matrix. This parameter is visible if, for the **Method** parameter, you select `Constant`. This parameter is available if, for the **Pad value source** parameters, you select `Specify via dialog`. 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 matrix. Select `Right` to add additional columns to the right side of the matrix. Select `Both left and right` to add additional columns to the left and right side of the matrix. If you select `No padding`, the block does not change the number of columns of the input matrix.

Pad size along rows

This parameter controls how many columns are added to the right and/or left side of your input matrix. Enter a scalar value, and the block adds this number of columns to the left and/or right side of your matrix. 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 matrix and the right element controls how many columns the block adds to the right side of the matrix. This parameter is visible if, for the **Specify** parameter, you select `Pad size`.

Number of output columns

Enter a scalar value that represents the total number of output columns. This parameter is visible if, for the **Specify** parameter, you select `Output size`.

Pad columns at

Select `Top` to add additional rows at the top of the input matrix. Select `Bottom` to add additional rows at the bottom of the matrix. Select `Both top and bottom` to add additional rows at the top and bottom of the matrix. If you select `No padding`, the block does not change the number of rows of the input matrix.

Pad size along columns

This parameter controls how many rows are added to the top and/or bottom of your input matrix. Enter a scalar value and the block adds this number of columns to the top and/or bottom of your matrix. 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 matrix and the right element controls how many rows the block adds to the bottom of the matrix. This parameter is visible if, for the **Specify** parameter, you select `Pad size`.

Number of output rows

Enter a scalar value that represents the total number of output rows. This parameter is visible if, for the **Specify** parameter, you select `Output size`.

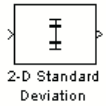
Purpose

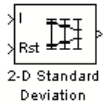
Find standard deviation of each input matrix

Library

Statistics

Description





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

Port	Input/Output	Supported Data Types	Complex Values Supported
Input / I	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the input RGB video stream.	• 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 integers • 8-, 16-, and 32-bit unsigned integers	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 integers • 8-, 16-, and 32-bit unsigned integers 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 integers	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 integers	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.

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

Note that 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

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

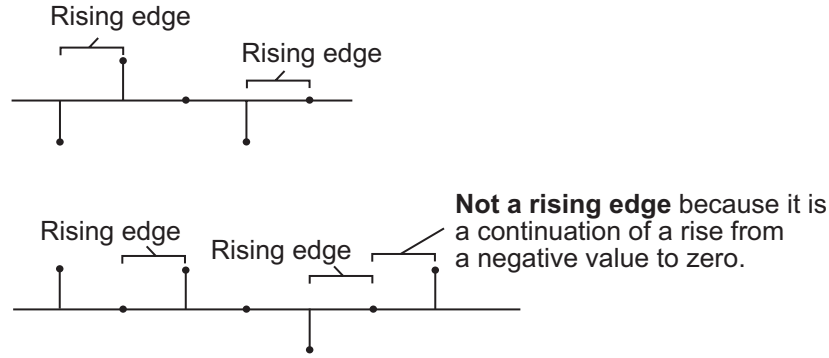
For the next input, the block computes the standard deviation for each element of the first three inputs, and so on.

Resetting the Running Standard Deviation

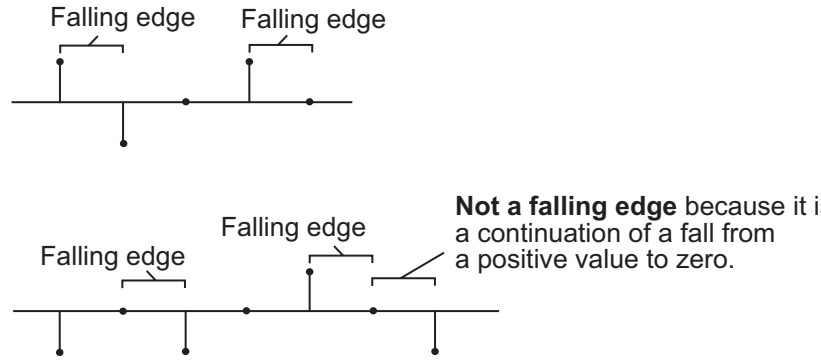
The block resets the running standard deviation whenever a reset event is detected at the optional Rst port. The reset signal rate must be a positive integer multiple of the rate of the data signal input.

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)



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



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

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

ROI Processing

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

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

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

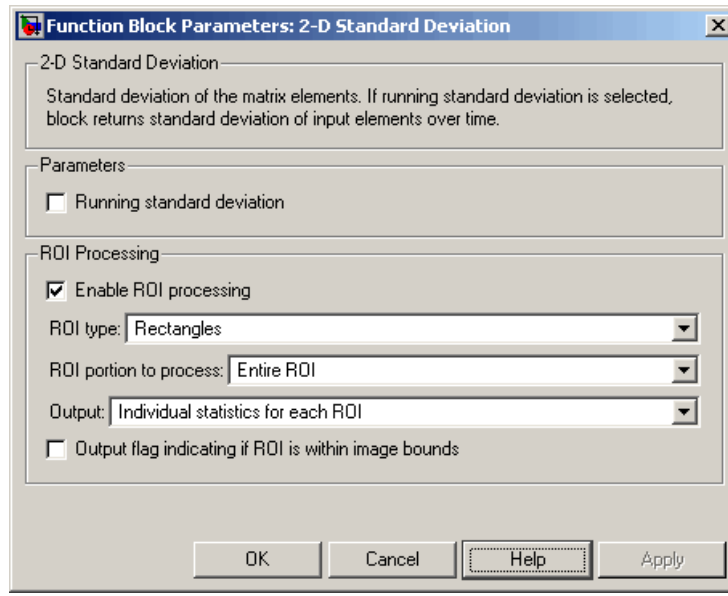
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

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.



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 10-155. This parameter is visible if, for the **ROI type** parameter, you select `Rectangles` or `Lines`.

Output flag indicating if label numbers are valid
If you select this check box, the Flag port appears on the block. For a description of the Flag port output, see the tables in "ROI Processing" on page 10-155. This parameter is visible if, for the **ROI type** parameter, you select `Label matrix`.

See Also

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

Minimum	Signal Processing Blockset
Standard Deviation	Signal Processing Blockset
`std`	MATLAB

Purpose

Compute variance of each input matrix

Library

Statistics

Description



2-D Variance



2-D 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.

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

Port	Input/Output	Supported Data Types	Complex Values Supported
Input / I	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the input RGB video stream.	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	Yes
Rst	Signal that triggers a reset event	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	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 integers - 8-, 16-, and 32-bit unsigned integers • 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 integers	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 integers	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:

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

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,

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

and outputs

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

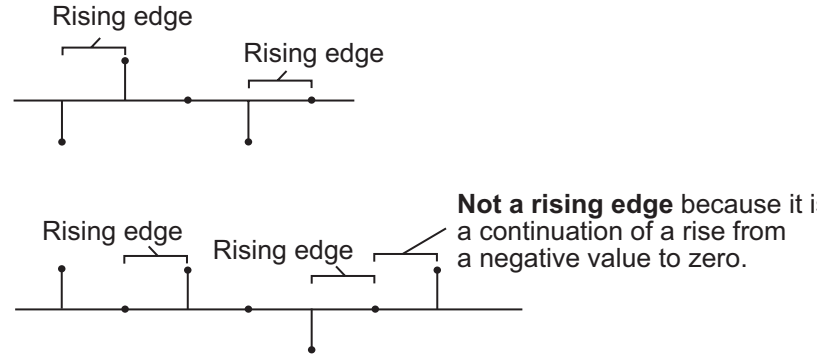
For the next input, the block computes the variance for each element of the first three inputs, and so on.

Resetting the Running Variance

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

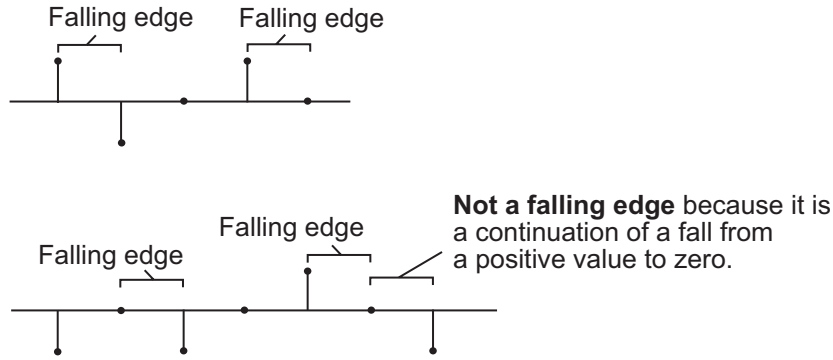
You specify the reset event using the **Reset port** parameter:

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



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

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



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

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

ROI Processing

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

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

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

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

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

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

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

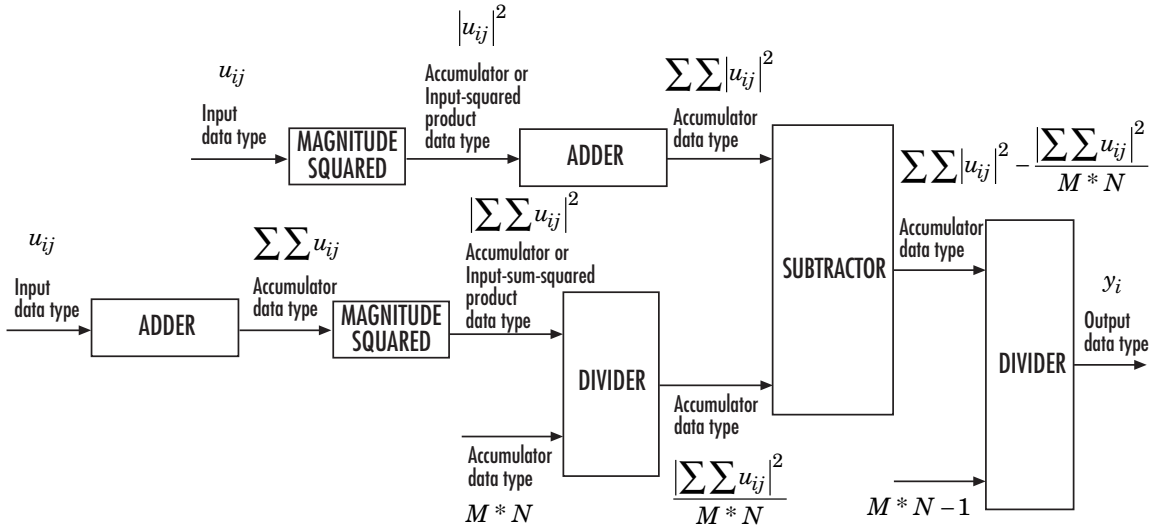
Flag Port Output	Description
0	Label number is not in the label matrix.
1	Label number is in the label matrix.

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

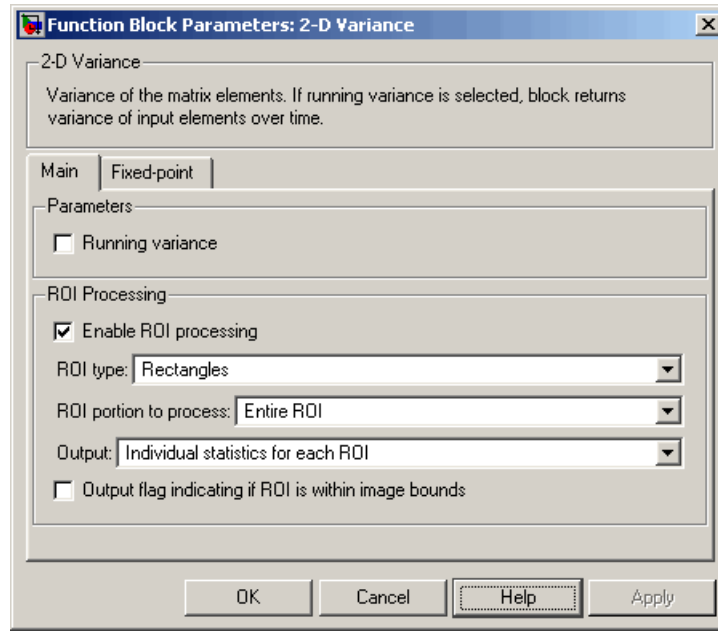
The following diagram shows the data types used in the 2-D Variance block for fixed-point signals.



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.



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`.

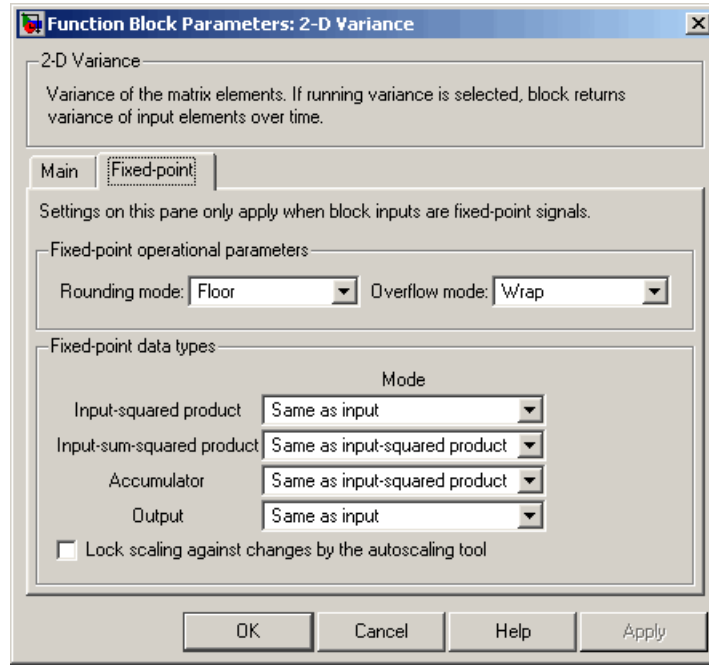
ROI portion to process
Specify whether to calculate the statistical value for the entire ROI or just the ROI perimeter. This parameter is only visible if, for the **ROI type** parameter, you specify `Rectangles`.

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 10-166. 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 10-166. 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:



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 10-169 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

See Also

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

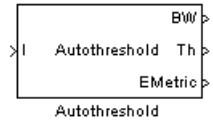
Purpose

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	Scalar, vector, or matrix of intensity values or 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-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	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.

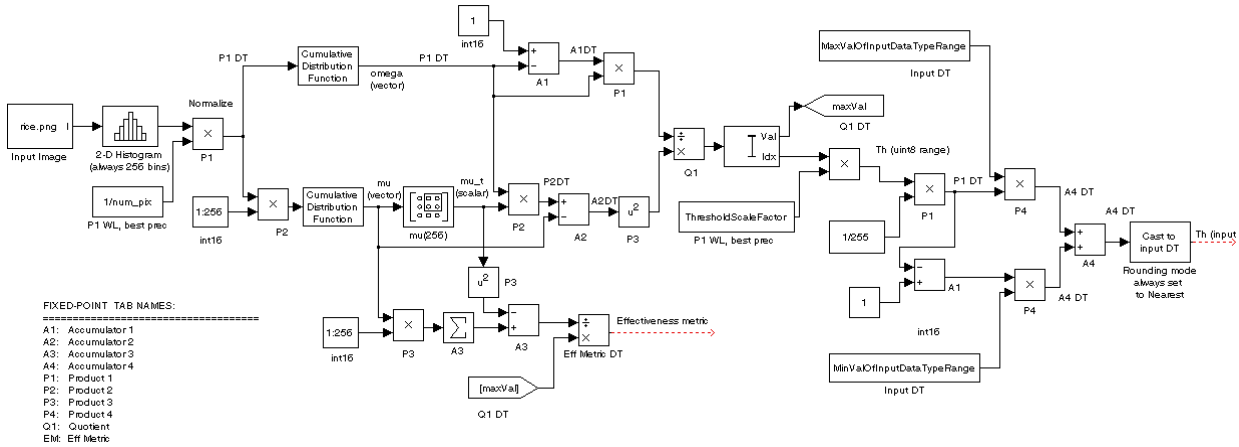
If you clear the **Scale threshold** check box, the block uses the threshold value computed by Otsu's method to convert intensity images into binary images. If you select the **Scale threshold** check box, the **Threshold scaling factor** appears in the dialog box. Enter a scalar value. The block multiplies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

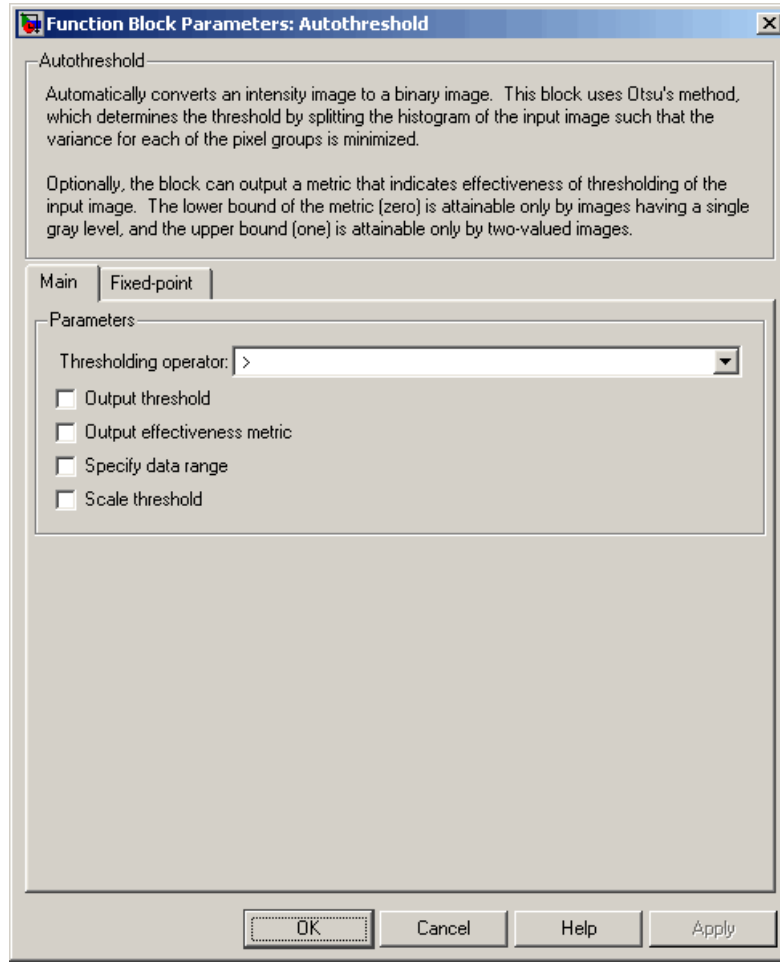
The following diagram shows the data types used in the Autothreshold block for fixed-point signals. You can use the default fixed-point parameters if your input has a word length less than or equal to 16.

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.



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. Tunable.

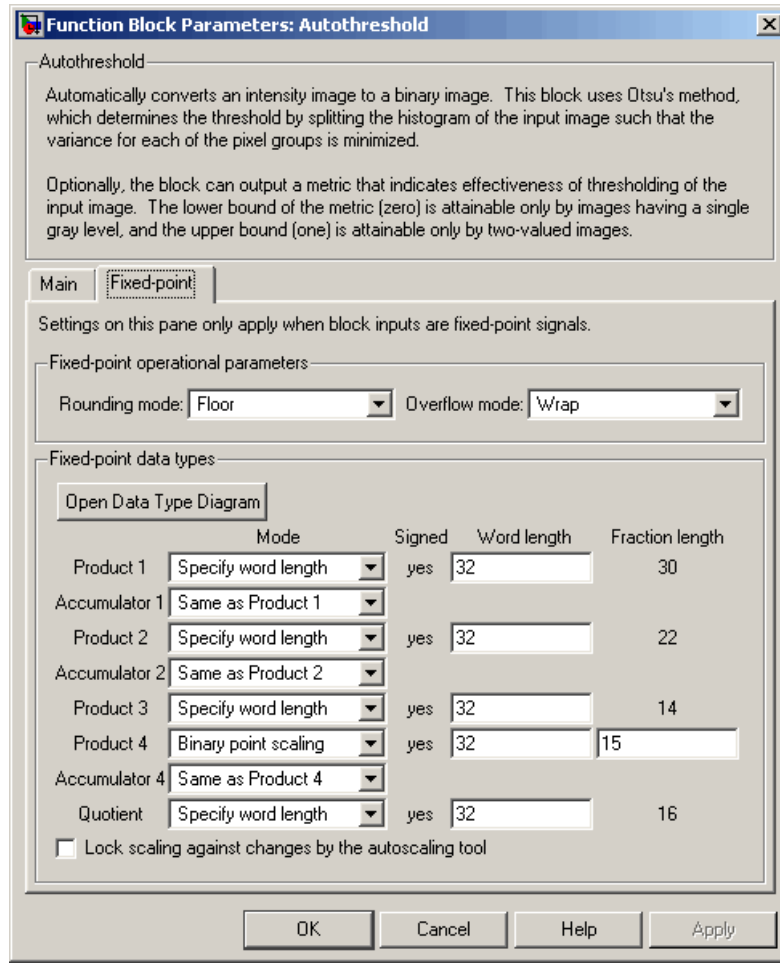
Maximum value of input
Enter the maximum value of your input data. This parameter is visible if you select the **Specify data range** check box. Tunable.

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

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

Threshold scaling factor
Enter a scalar value. The block multiplies this scalar value with the threshold value computed by Otsu's method and uses the result as the new threshold value. This parameter is visible if you select the **Scale threshold** check box.

The **Fixed-point** pane of the Autothreshold dialog box appears as follows. You can use the default fixed-point parameters if your input has a word length less than or equal to 16.



Rounding mode

Select the rounding mode for fixed-point operations. This parameter does not apply to the Cast to input DT step shown in "Fixed-Point Data Types" on page 10-179. For this step, **Rounding mode** is always set to `Nearest`.

Overflow mode

Select the overflow mode for fixed-point operations.

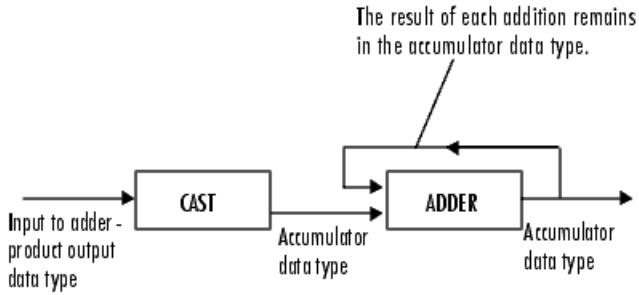
Product 1, 2, 3, 4



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 is 0.

Accumulator 1, 2, 3, 4



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

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

See Also

Compare To Constant	Simulink
Relational Operator	Simulink
`graythresh`	Image Processing Toolbox

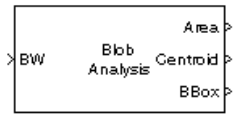
Purpose

Compute statistical values for labeled regions

Library

Statistics

Description



Blob Analysis

Use the Blob Analysis block to calculate statistics for labeled regions in a binary image. The block returns some quantities, such as the Centroid, as values that represent spatial coordinate locations. For information about how pixel and spatial coordinate systems are defined in the Video and Image Processing Blockset, see “Coordinate Systems” on page 1-13.

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 integers	No
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 integers	No

Port	Input/Output	Supported Data Types	Complex Values Supported
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
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

Port	Input/Output	Supported Data Types	Complex Values Supported
Label	Label matrix	• 8-, 16-, or 32-bit unsigned integers	No
Count	Scalar value that represents the actual number of labeled regions in each image	Same as Label port	No

Main Pane

Use the check boxes to specify the statistics values you want the block to output. The following table summarizes the block's behavior based on which check box you select. For a full description of each of these statistics, see the `regionprops` function reference page in the Image Processing Toolbox documentation.

Check Box	Block Output
Area	Vector that represents the number of pixels in labeled regions
Centroid	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.

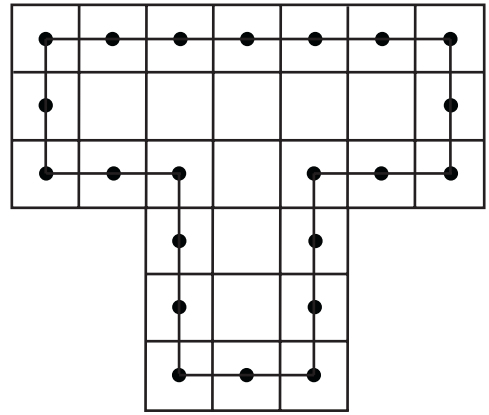
Check Box	Block Output
Bounding box	4-by-N matrix whose columns represent the coordinates of each bounding box, where N is the number of blobs. For example, suppose there are two blobs, where r and c define the row and column location of the upper-left corner of the bounding box and w and h define the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

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

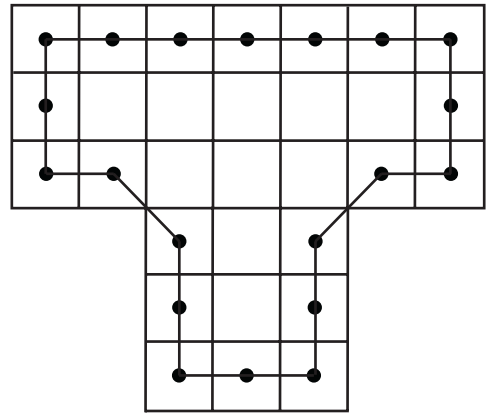
Use the **Statistics output data type** parameter to specify the data type of the outputs at the Centroid, MajorAxis, MinorAxis, Orientation, Eccentricity, Diameter^2, and Extent ports. If you select `Fixed-point`, the block cannot calculate the major axis, minor axis, orientation, or eccentricity and these check boxes become unavailable.

Use the **Connectivity** parameter to define which pixels are connected to each other. If you want a pixel to be connected to the other pixels located on the top, bottom, left, and right, select `4`. If you want a pixel to be connected to the other pixels on the top, bottom, left, right, and diagonally, select `8`. For more information about this parameter, see the Label block reference page.

The **Connectivity** parameter also affects how the block calculates the perimeter of a blob. The following figure illustrates how the block calculates the perimeter when you set the **Connectivity** parameter to `4`.



The blocks calculates the distance between the center of each pixel (marked by the black dots) and estimates the perimeter to be 22. The next figure illustrates how the block calculates the perimeter of a blob when you set the **Connectivity** parameter to 8.



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

If you select the **Output label matrix** check box, the block outputs the label matrix, where pixels equal to 0 represent the background, pixels equal to 1 represent the first object, pixels equal to 2 represent the second object, and so on, at the Label port.

Blob Properties Pane

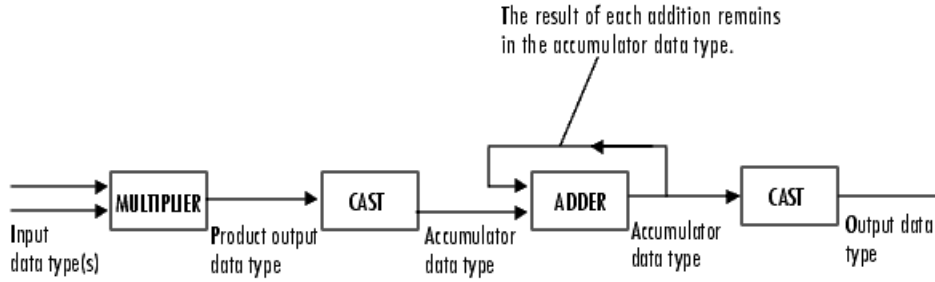
Use the **Maximum number of blobs** parameter to specify the maximum number of labeled regions in each input image. The block uses this value to preallocate vectors and matrices so that they are long enough to hold all of the statistical values. If you select the **Warn if maximum number of blobs is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

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

If the number of blobs found in an image is less than the scalar value entered for the **Maximum number of blobs** parameter, the block has empty spaces in the statistics output arrays. If you select the **Fill empty spaces in outputs** check box, the block fills the empty spaces with the scalar value you specify in the **Fill values** parameter. If you enter a vector for the **Fill values** parameter, it must have the same length as the number of selected statistics. The block uses each vector element to fill a different statistics vector. If the empty spaces do not affect your computation, you can deselect the **Fill empty spaces in outputs** check box. Otherwise, we recommend leaving it selected.

Fixed-Point Data Types

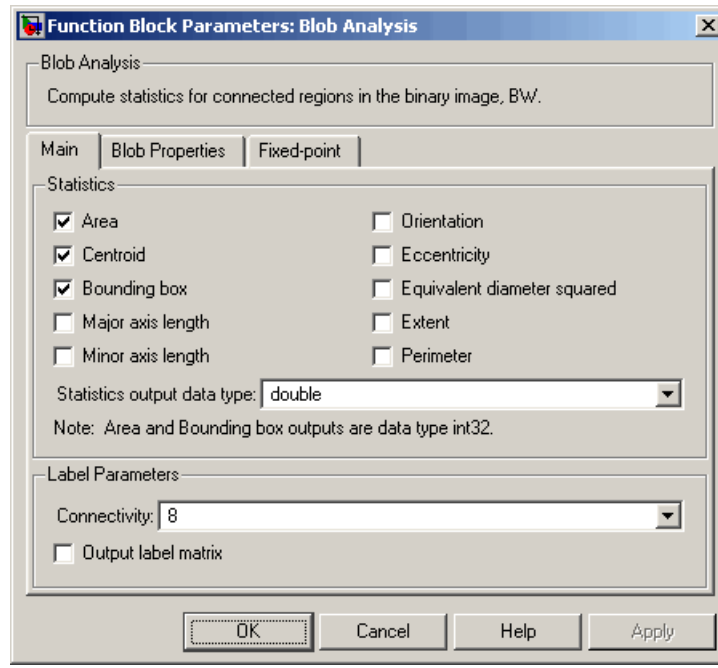
The following diagram shows the data types used in the Blob Analysis block for fixed-point signals.



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

Dialog Box

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



Area

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

Centroid

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

Bounding box

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

Major axis length
Select this check box to output a vector whose values represent the lengths of the major axes of the ellipses that have the same normalized second central moments as the labeled regions.

Minor axis length
Select this check box to output a vector whose values represent the lengths of the minor axes of the ellipses that have the same normalized second central moments as the labeled regions.

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

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

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

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

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

Statistics output data type
Specify the data type of the outputs at the Centroid, MajorAxis, MinorAxis, Orientation, Eccentricity, Diameter^2, and Extent ports.

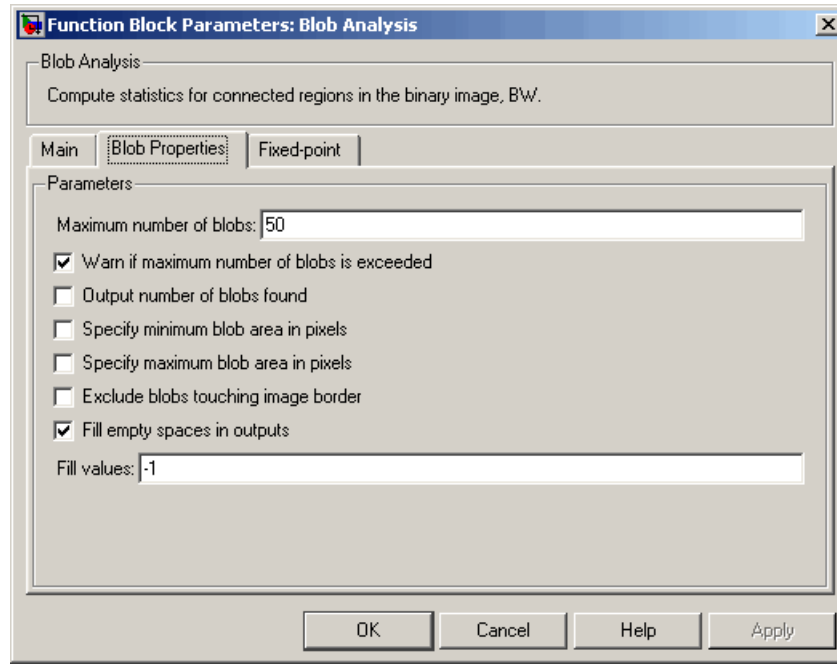
Connectivity
Specify which pixels are connected to each other. If you want a pixel to be connected to the pixels on the top, bottom, left, and

right, select `4`. If you want a pixel to be connected to the pixels on the top, bottom, left, right, and diagonally, select `8`.

Output label matrix

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

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



Maximum number of blobs

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

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

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

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

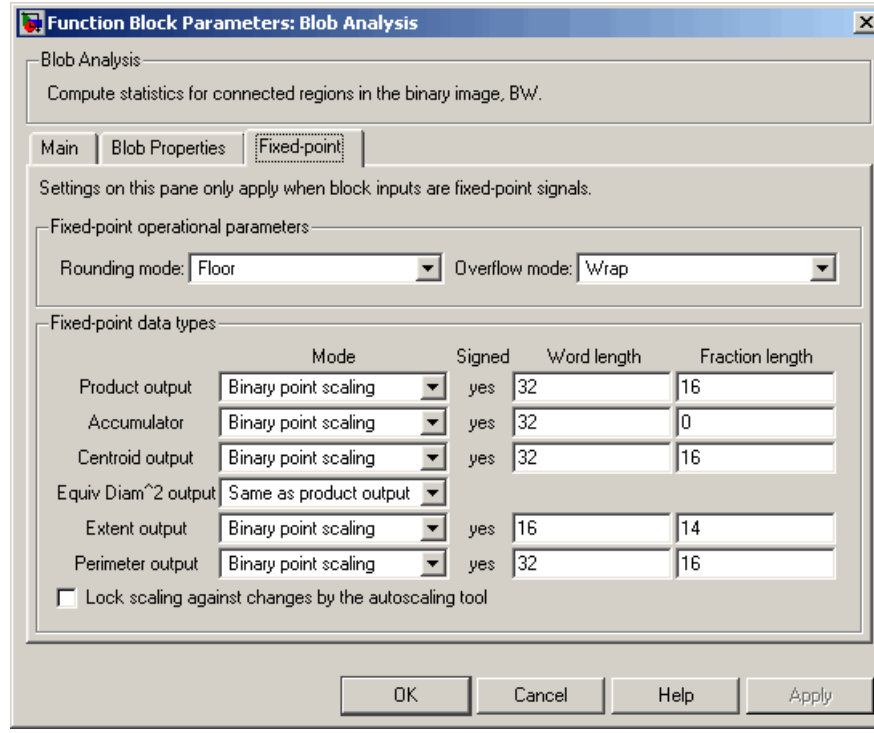
Specify maximum blob area in pixels
If you select this check box, you can enter the maximum blob area in the edit box that appears beside the check box. Blobs that do not meet this criteria are not labeled.

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

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

Fill values
If you enter a scalar value, the block fills all the empty spaces in the statistical vectors with this value. If you enter a vector, it must have the same length as the number of selected statistics. The block uses each vector element to fill a different statistics vector.

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



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output

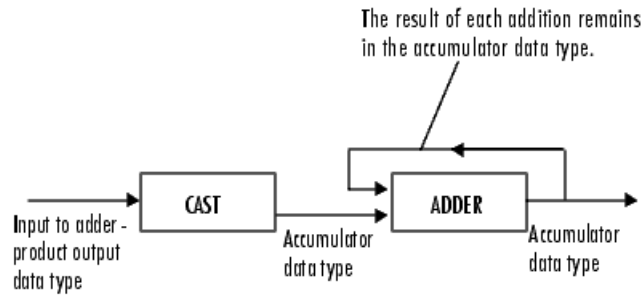


As depicted previously, the output of the multiplier is placed into the product output data type and scaling. The product output data

type is used during the computation of the equivalent diameter squared. During this computation, the blob area, which is stored in the accumulator, is multiplied by the 4/pi factor. This factor has a word length that is equal to the equivalent diameter squared output data type's word length and a fraction length that is equal to its word length minus two. Use this parameter to specify how to designate this product output word and fraction lengths.

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

Accumulator



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

- When you select `Same as product output`, these characteristics match those of the product output.

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

Centroid output

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

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

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 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 is 0.

Extent output

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

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

Perimeter output

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

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

Lock scaling against changes by the autoscaling tool

Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

See Also

Label	Video and Image Processing Blockset
Variable Selector	Signal Processing Blockset
`regionprops`	Image Processing Toolbox

Purpose

Estimate motion between images or video frames

Library

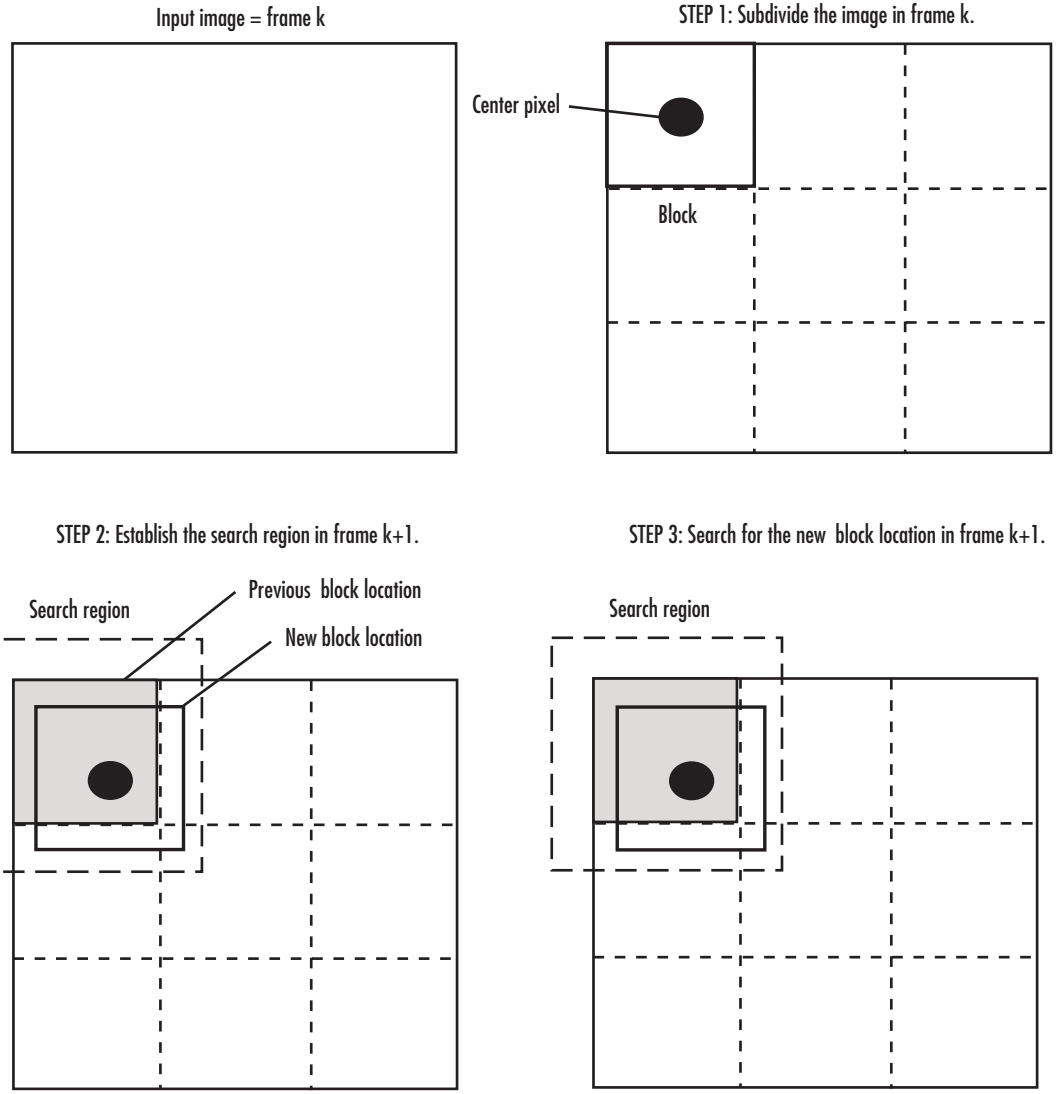
Analysis & Enhancement

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 integers • 8-, 16-, and 32-bit unsigned integers	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 MSE equation, shown below:

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

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 MAD equation, shown below:

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

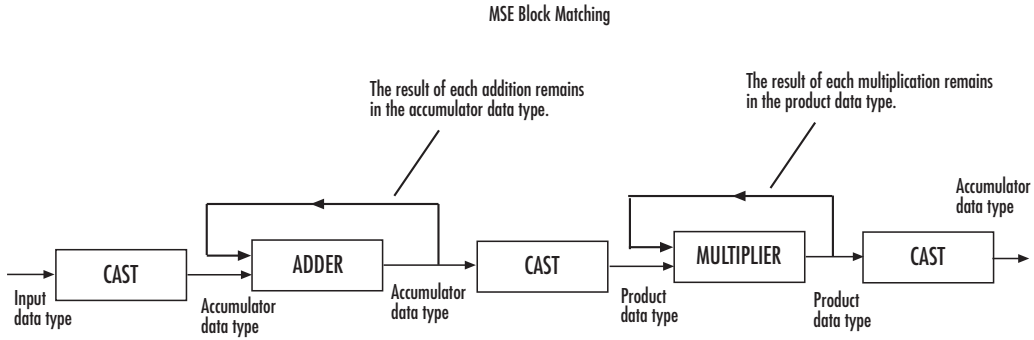
Use the **Block size [height width]** and **Overlap [r c]** parameters to specify how the block subdivides the input image. For a graphical description of these parameters, see the first figure in this reference page. If the **Overlap [r c]** parameter is not `[0 0]`, the blocks would overlap each other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

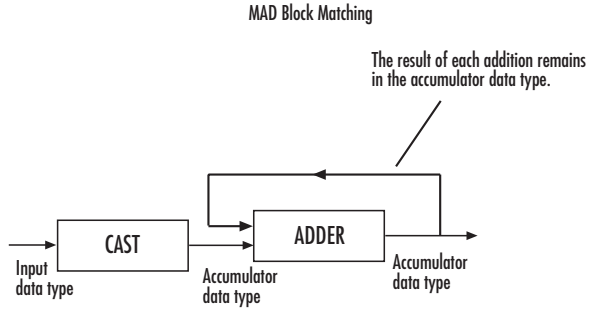
Use the **Maximum displacement [r c]** parameter to specify the maximum number of pixels any center pixel in a block of pixels might move from image to image or frame to frame. The block uses this value to determine the size of the search region.

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.

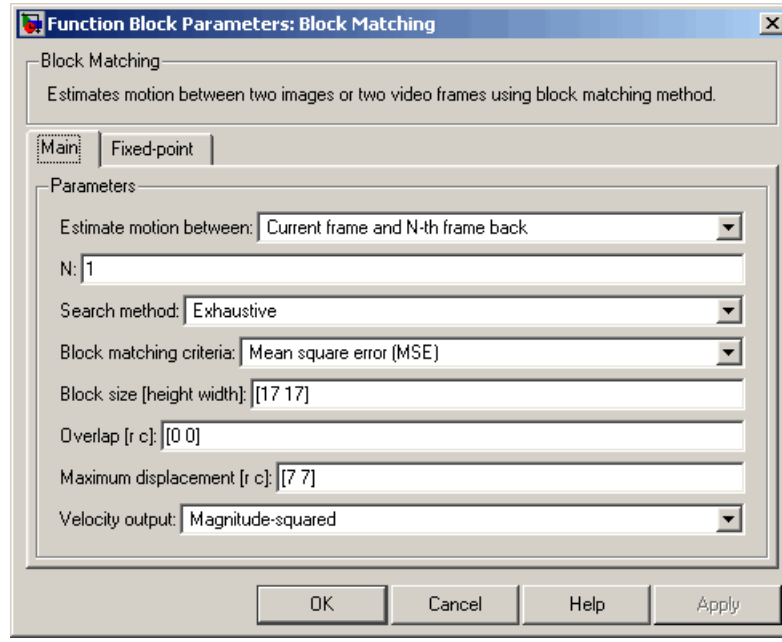




You can set the accumulator and output data types in the block mask as discussed in the next section.

Dialog Box

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



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)`.

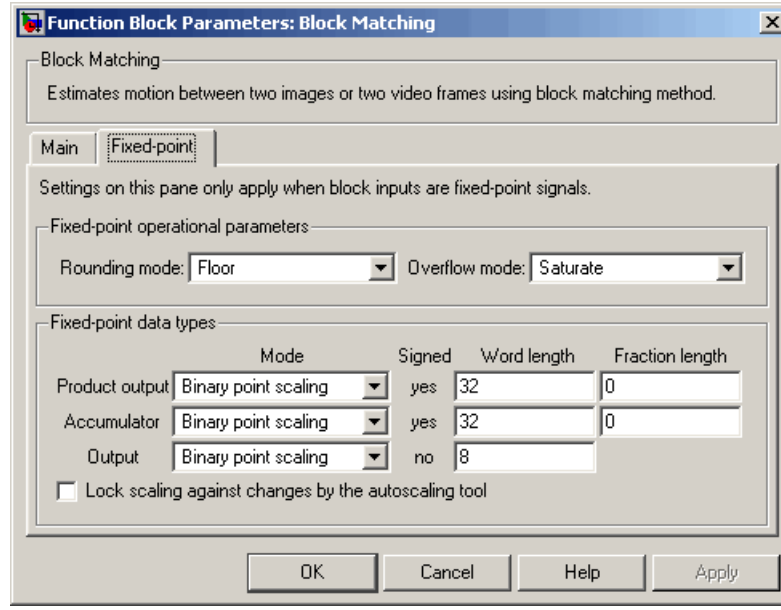
Block size [height width]
Specify the size of the block of pixels.

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.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

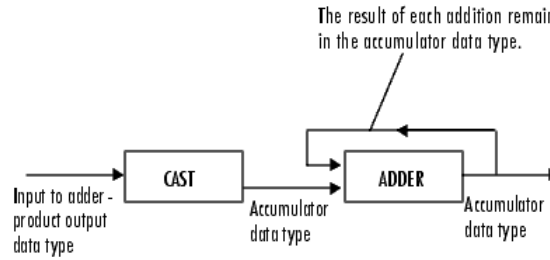
Product output



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 is 0.

Accumulator



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

- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset 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 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

See Also

Optical Flow Video and Image Processing Blockset

Block Processing

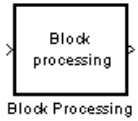
Purpose

Repeat user-specified operation on submatrices of input matrix

Library

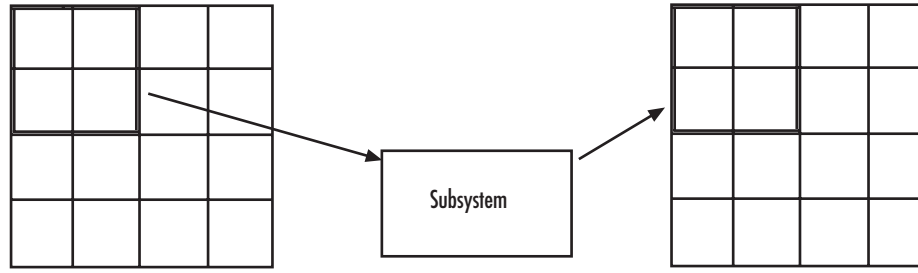
Utilities

Description



Block Processing

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



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

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

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

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

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

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

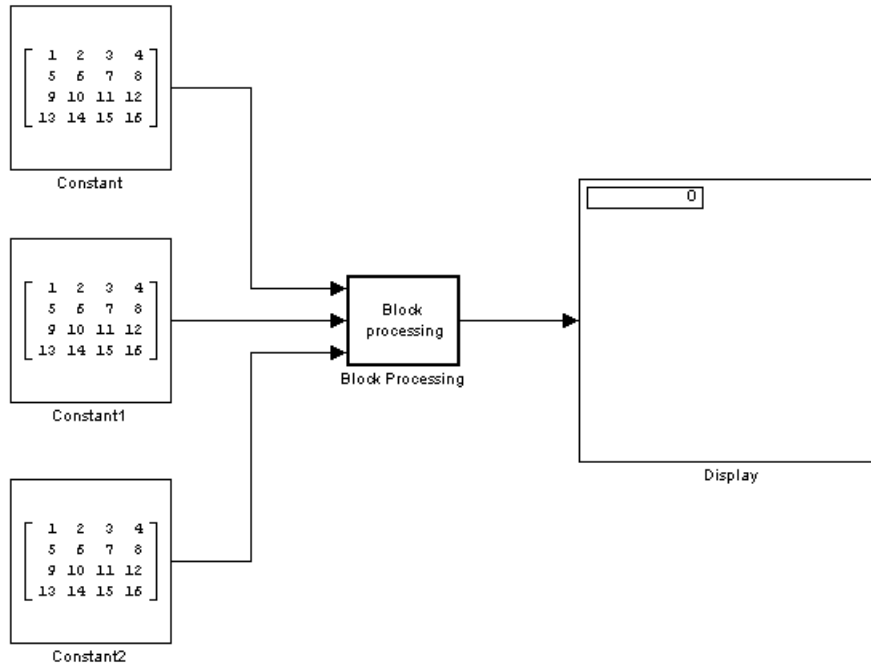
Click the **Open Subsystem** button to open the block's subsystem. Click-and-drag blocks into this subsystem to define the processing operation(s) the block performs on the submatrices. The input to this subsystem are the submatrices whose size is determined by the **Block size** parameter.

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

Example

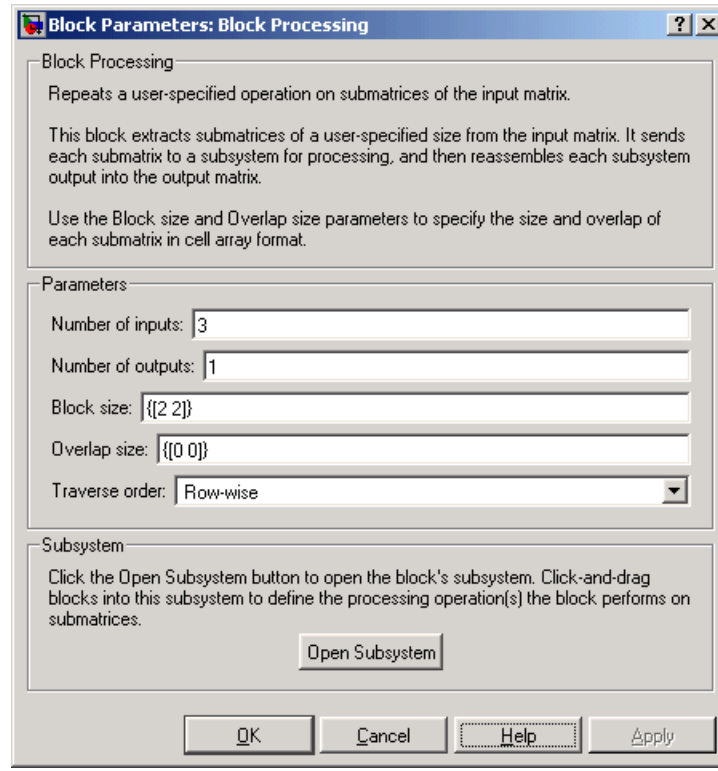
Example 1 -- Multiple Inputs

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



1 Use the Block Processing block to perform the multiplication and addition on submatrices of the three input matrices. Set the block parameters as shown in the following figure.

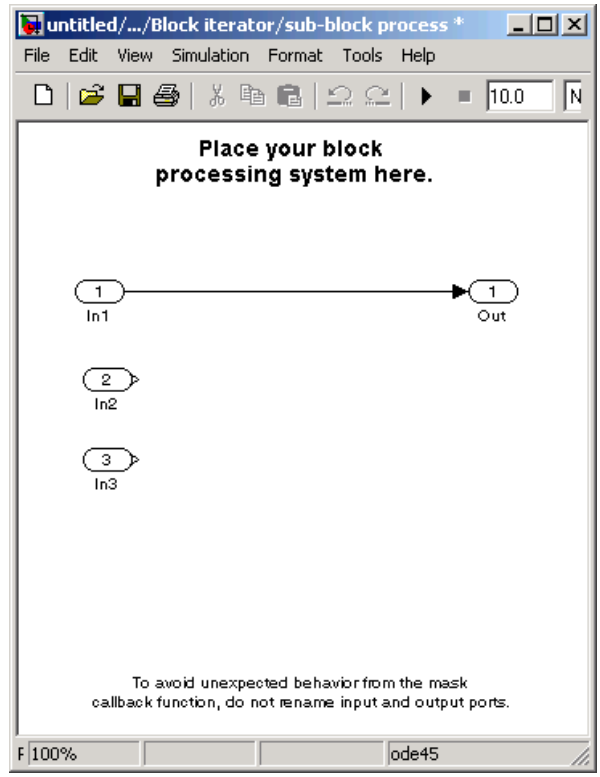
- **Number of inputs** = `3`
- **Number of outputs** = `1`
- **Block size** = `{[2 2]}`



For each iteration, the block sends a 2-by-2 submatrix from each input matrix to the Block Processing blocks' subsystem to be processed. The block calculates its total number of iterations using the dimensions of the matrix connected to the top input port. In this case, the first input is a 4-by-4 matrix. Since the block can extract four 2-by-2 submatrices from this input matrix, the block iterates four times.

2 Click **Open Subsystem**.

The block's subsystem opens.



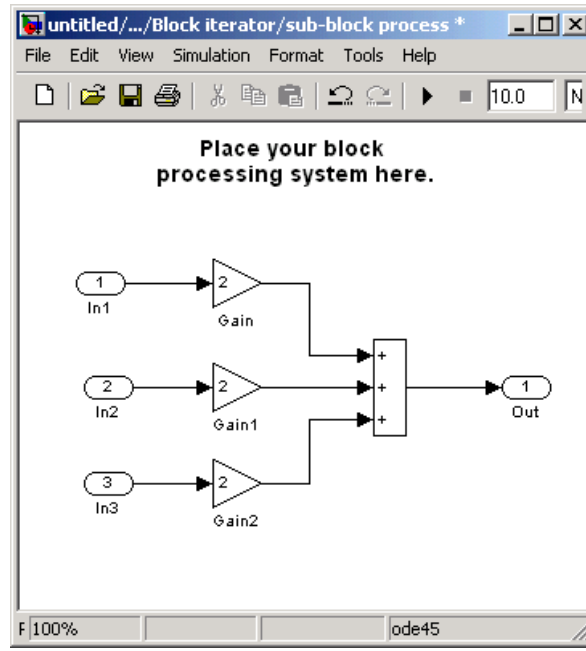
3 Click and drag the following blocks into the subsystem:

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

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

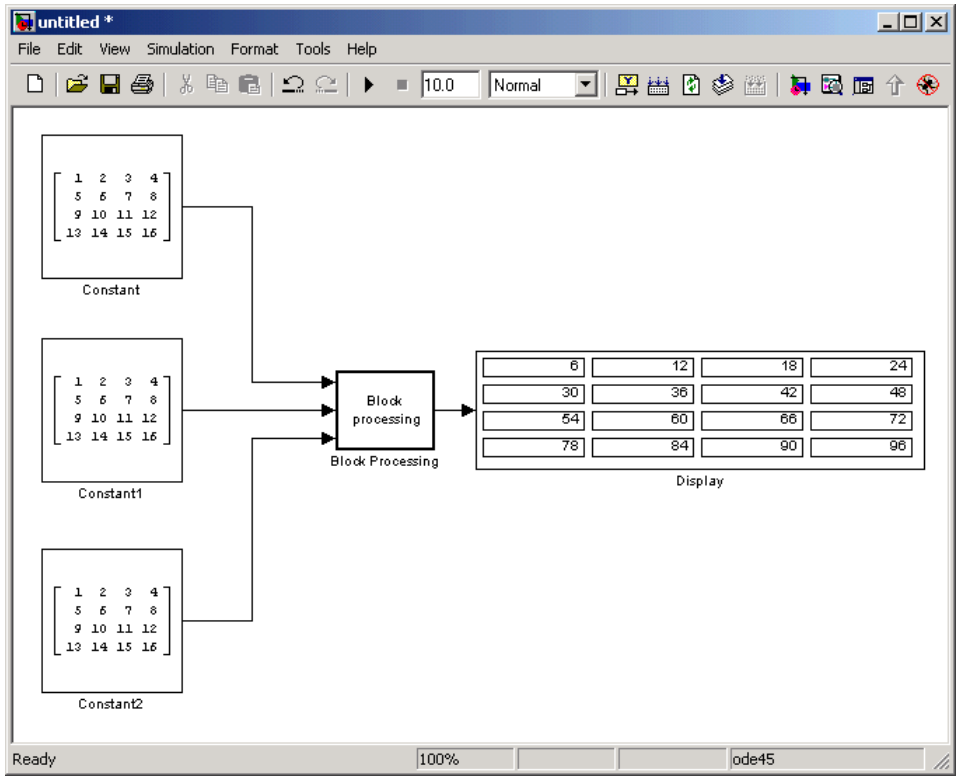
5 Use the Sum block to add the values. Set the **Icon shape** parameter to `rectangular` and the **List of signs** parameter to `+++`.

6 Connect the blocks as shown in the figure below.



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

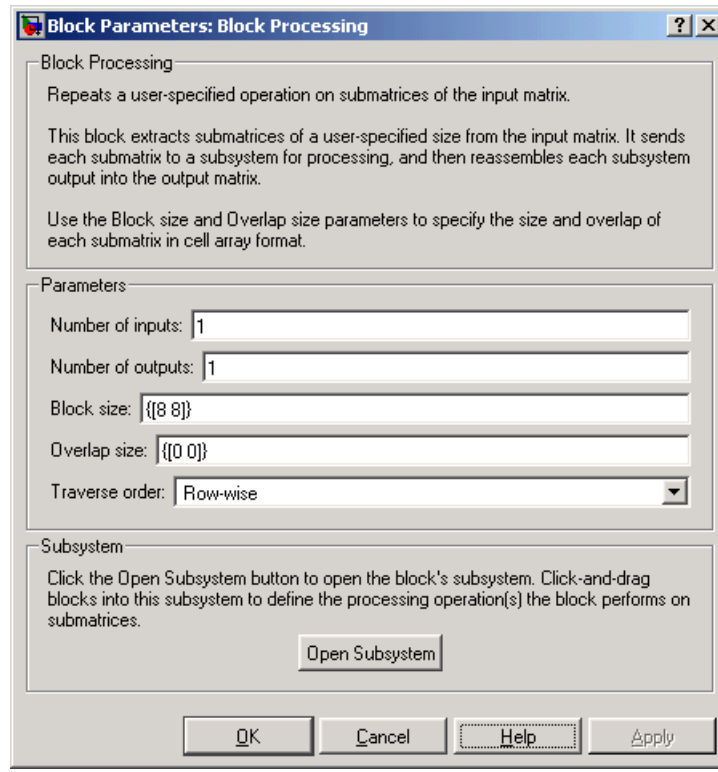
8 Run the model.



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.



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

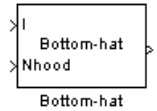
Purpose

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 result of performing a morphological closing operation on the input image from the input image itself. This block uses flat structuring elements only.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the input RGB video stream	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	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`.

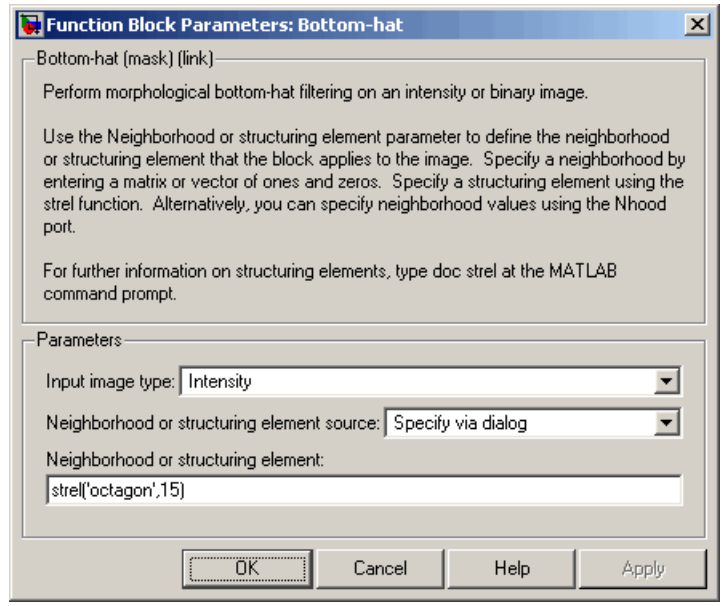
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select `Specify via dialog`, the **Neighborhood or structuring**

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.



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

Chroma Resampling

Purpose

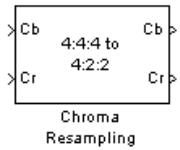
Downsample or upsample chrominance components of images

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.

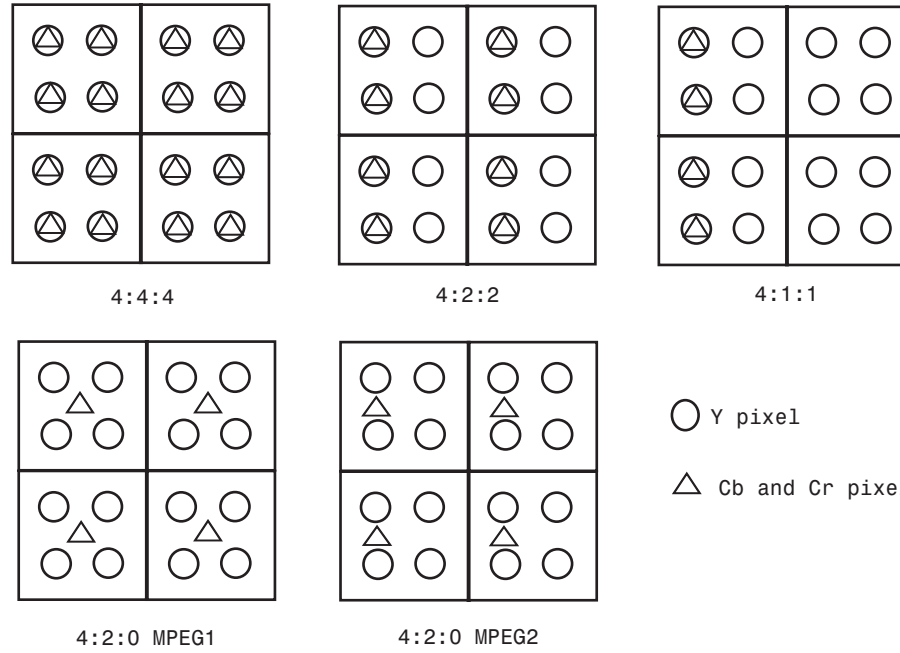


Port	Input/Output	Supported Data Types	Complex Values Supported
Cb	Matrix that represents one chrominance component of an image	• Double-precision floating point • Single-precision floating point • 8-bit unsigned integers	No
Cr	Matrix that represents one chrominance component of an image	Same as Cb port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



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. 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. 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.

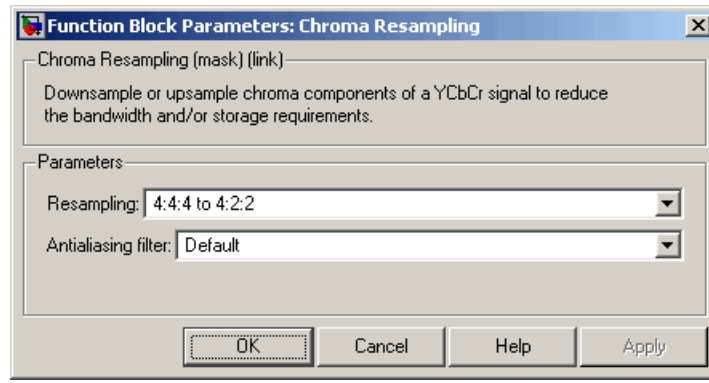
Upsampling

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.

Dialog Box

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



Resampling

Specify the resampling format.

Antialiasing filter

Specify the lowpass filter that the block uses to prevent aliasing. If you select `Default`, the block uses a built-in lowpass filter. If you select `User-defined`, the **Horizontal filter coefficients** and/or **Vertical filter coefficients** parameters appear on the dialog. 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.

References

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

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

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

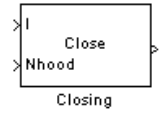
Purpose

Perform morphological closing on binary or intensity images

Library

Morphological Operations

Description



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

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Vector or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	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

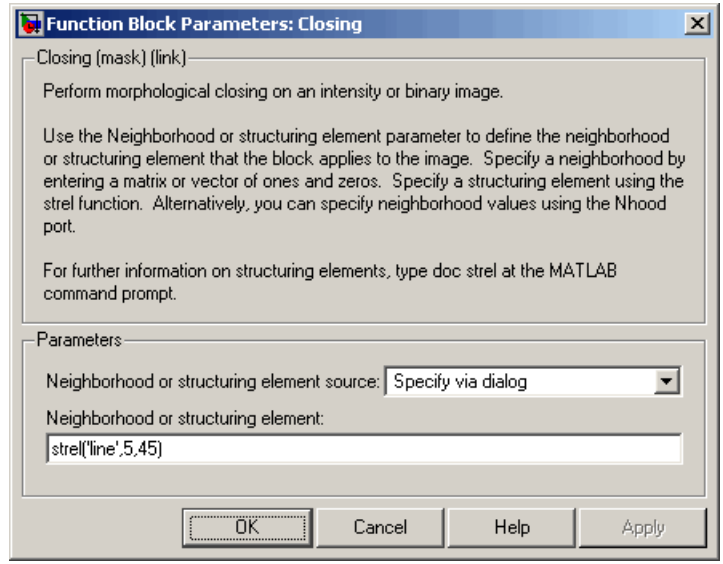
The output signal has the same data type as the input to the I port. This block supports a signal represented by a Simulink virtual bus.

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

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

Dialog Box

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



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

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

References

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

See Also

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

Color Space Conversion

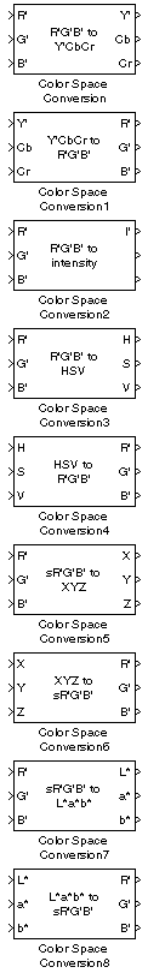
Purpose

Convert color information between color spaces

Library

Conversions

Description



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

Port	Input/Output	Supported Data Types	Complex Values Supported
R’	Matrix that represents one plane of the input RGB video stream	• Double-precision floating point • Single-precision floating point • 8-bit unsigned integers	No
G’	Matrix that represents one plane of the input RGB video stream	Same as the R’ port	No
B’	Matrix that represents one plane of the input RGB video stream	Same as the R’ port	No
Y’	Matrix that represents the luma portion of an image	Same as the R’ port	No
Cb	Matrix that represents one chrominance component of an image	Same as the R’ port	No
Cr	Matrix that represents one chrominance component of an image	Same as the R’ port	No
I’	Matrix of intensity values	Same as the R’ 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	• Double-precision floating point • Single-precision floating point	No
V	Matrix that represents the value (brightness) component of an image	• Double-precision floating point • Single-precision floating point	No
X	Matrix that represents the X component of an image	• Double-precision floating point • Single-precision floating point	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Y	Matrix that represents the Y component of an image	• Double-precision floating point • Single-precision floating point	No
Z	Matrix that represents the Z component of an image	• Double-precision floating point • Single-precision floating point	No
L*	Matrix that represents the luminance portion of an image	• Double-precision floating point • Single-precision floating point	No
a*	Matrix that represents the a* component of an image	• Double-precision floating point • Single-precision floating point	No
b*	Matrix that represents the b* component of an image	• Double-precision floating point • Single-precision floating point	No

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

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.

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

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

$$V = MAX$$

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:

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

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

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

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

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

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

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

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

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

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

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

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

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

If $R_{sRGB}, G_{sRGB}, B_{sRGB} \le 0.00304$

$$R'_{sRGB} = 12.92R_{sRGB}$$
$$G'_{sRGB} = 12.92G_{sRGB}$$
$$B'_{sRGB} = 12.92B_{sRGB}$$

otherwise, if $R_{sRGB}, G_{sRGB}, B_{sRGB} > 0.00304$

$$R'_{sRGB} = 1.055R_{sRGB}^{(1.0/2.4)} - 0.055$$
$$G'_{sRGB} = 1.055G_{sRGB}^{(1.0/2.4)} - 0.055$$
$$B'_{sRGB} = 1.055B_{sRGB}^{(1.0/2.4)} - 0.055$$

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

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

The Color Space Conversion block converts sR'G'B' values to L*a*b* values in two steps. First it converts sR'G'B' to XYZ values using the equations described in "Conversion Between sR'G'B' and XYZ Color Spaces" on page 10-239. 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:

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

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

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:

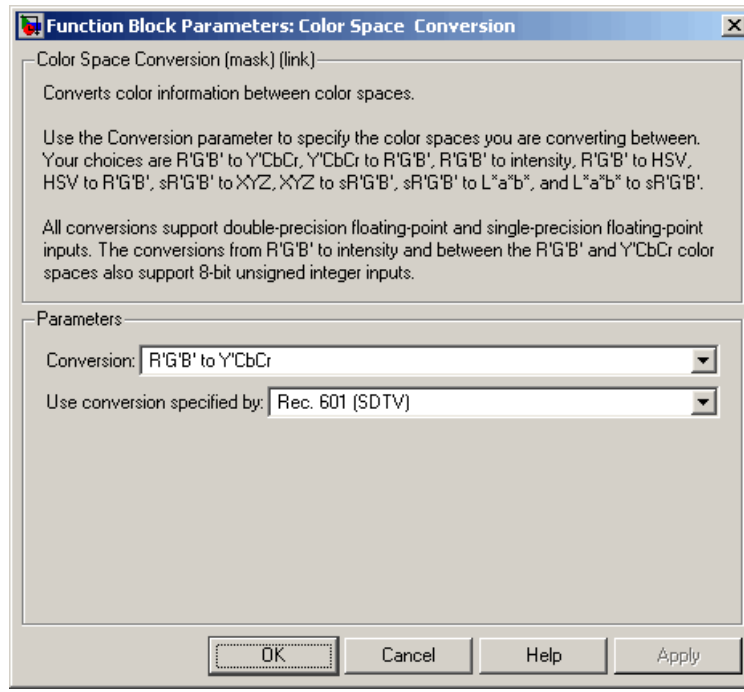
For $Y/Y_n > 0.008856$

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

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

Dialog Box

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



Conversion

Specify the color spaces you are converting between. Your choices are `R'G'B' to Y'CbCr`, `Y'CbCr to R'G'B'`, `R'G'B' to intensity`, `R'G'B' to HSV`, `HSV to R'G'B'`, `sR'G'B' to XYZ`, `XYZ to sR'G'B'`, `sR'G'B' to L*a*b*`, and `L*a*b* to sR'G'B'`.

Use conversion specified by

Specify the standard to use to convert your values between the R'G'B' and Y'CbCr color spaces. Your choices are `Rec. 601 (SDTV)` or `Rec. 709 (HDTV)`. This parameter is only available if, for the **Conversion** parameter, you select `R'G'B' to Y'CbCr` or `Y'CbCr to R'G'B'`.

Scanning standard

Specify the scanning standard to use to convert your values between the R'G'B' and Y'CbCr color spaces. Your choices are `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'`.

References

Poynton, Charles A. *A Technical Introduction to Digital Video*. New York: John Wiley & Sons, 1996.

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

Recommendation ITU-R BT.709-5. Parameter values for the HDTV standards for production and international programme exchange.

Stokes, Michael, Matthew Anderson, Srinivasan Chandrasekar, and Ricardo Motta, "A Standard Default Color Space for the Internet – sRGB." November 5, 1996. `http://www.w3.org/Graphics/Color/sRGB`

Berns, Roy S. *Principles of Color Technology, 3rd ed.* New York: John Wiley & Sons, 2000.

See Also

Chroma Resampling	Video and Image Processing Blockset
`rgb2hsv`	MATLAB
`hsv2rgb`	MATLAB
`rgb2ycbcr`	Image Processing Toolbox
`ycbcr2rgb`	Image Processing Toolbox
`rgb2gray`	Image Processing Toolbox

`makecform`	Image Processing Toolbox
`applycform`	Image Processing Toolbox

Purpose

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

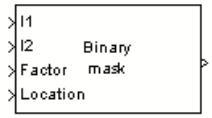
Library

Text & Graphics

Description



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 I2 over I1. The masking factor and the location determine which I1 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.

Port	Input/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 • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
I2	Scalar, vector, or matrix of intensity values	Same as I1 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 integers • 8-, 16-, and 32-bit unsigned integers	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 I1 and I2 ports is a floating-point data type.) • Single-precision floating point. (Only supported if the input to the I1 and I2 ports is a floating-point data type.) • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
Output	Scalar, vector, or matrix of intensity values	Same as I1 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. 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. Use this parameter to define the amount by which the block scales each I2 pixel value before combining them with the I1 pixel values. You can enter a scalar value used for all pixels or a matrix of values that is the same size as I2. If, for the **Opacity factor(s) source** parameter, 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 I1 and I2 ports is floating point, the input to this port must be the same floating-point data type.

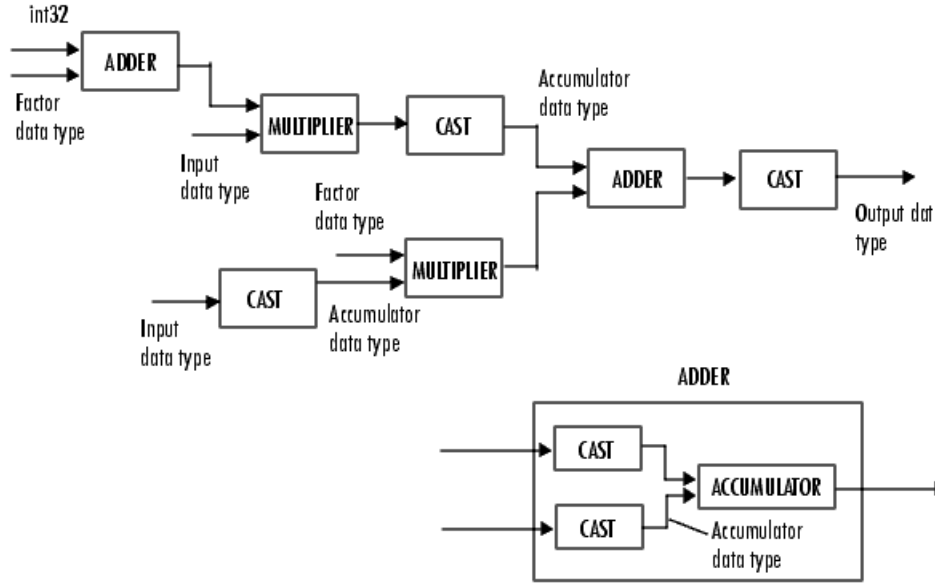
If, for the **Operation** parameter, you choose `Binary mask`, the **Mask source** parameter appears on the dialog. Use this parameter to indicate where to specify the masking factor(s). If you choose `Specify via dialog`, the **Mask** parameter appears on the dialog. Use this parameter and the location 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, for the **Mask source** parameter, 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. Enter a two-element vector that specifies the row and column position of the upper-left corner of the image input at port I2 relative to the upper-left corner of the image input at port I1. Positive values move the image down and to the right; negative images move the image up and to the left. If the first element is greater than the number of rows in the I1 matrix, the value is clipped to the total number of rows. If the second element is greater than the number of columns in the I1 matrix, the value is clipped to the total number of columns. If, for the **Location source** parameter, 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. This parameter is described in the previous paragraph.

Fixed-Point Data Types

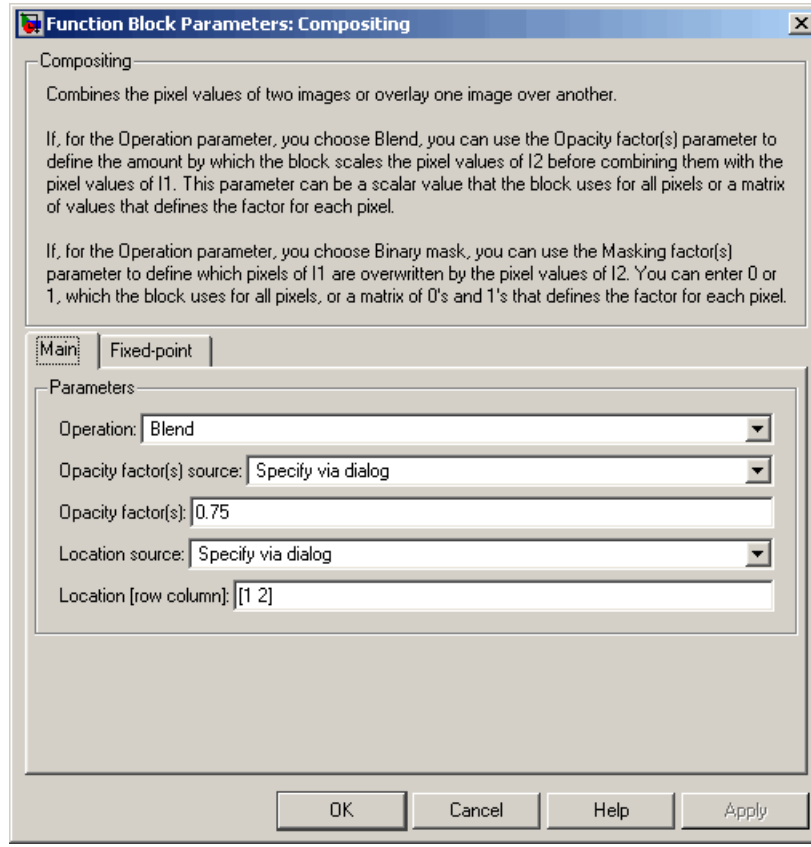
The following diagram shows the data types used in the Compositing block for fixed-point signals. It is only applicable when the **Operation** parameter is set to `Blend`.



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.



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.

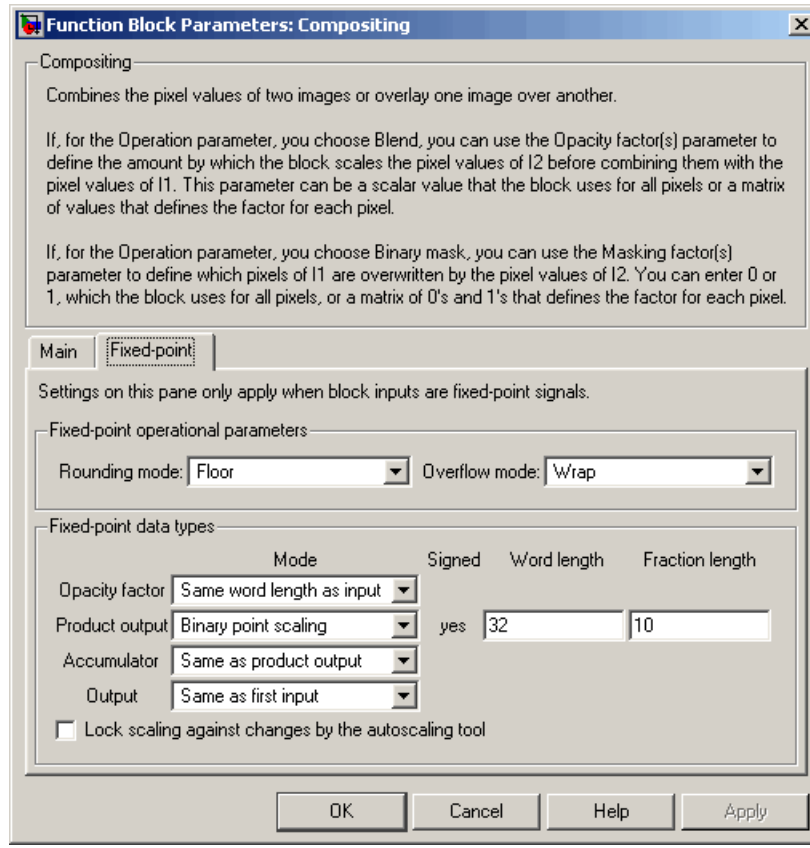
Mask source
Indicate where to specify the masking factor(s). Your choices are `Specify via dialog` and `Input port`. This parameter is visible if, for the **Operation** parameter you choose `Binary mask`.

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 I2 relative to the upper-left corner of the image input at port I1. This parameter is visible if, for the **Location source** parameter you choose `Specify via dialog`. Tunable.

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



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

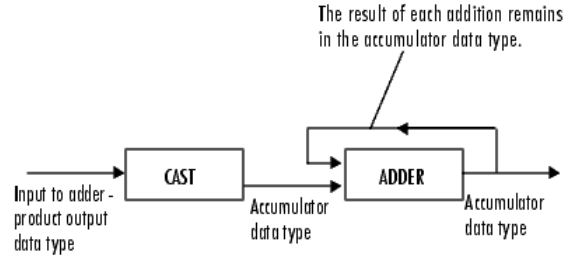
Product output



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 is 0.

Accumulator



As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder

remains in the accumulator data type as each element of the input is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths.

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

See Also

Insert Text Video and Image Processing Blockset

Purpose

Adjust image contrast by linearly scaling pixel values

Library

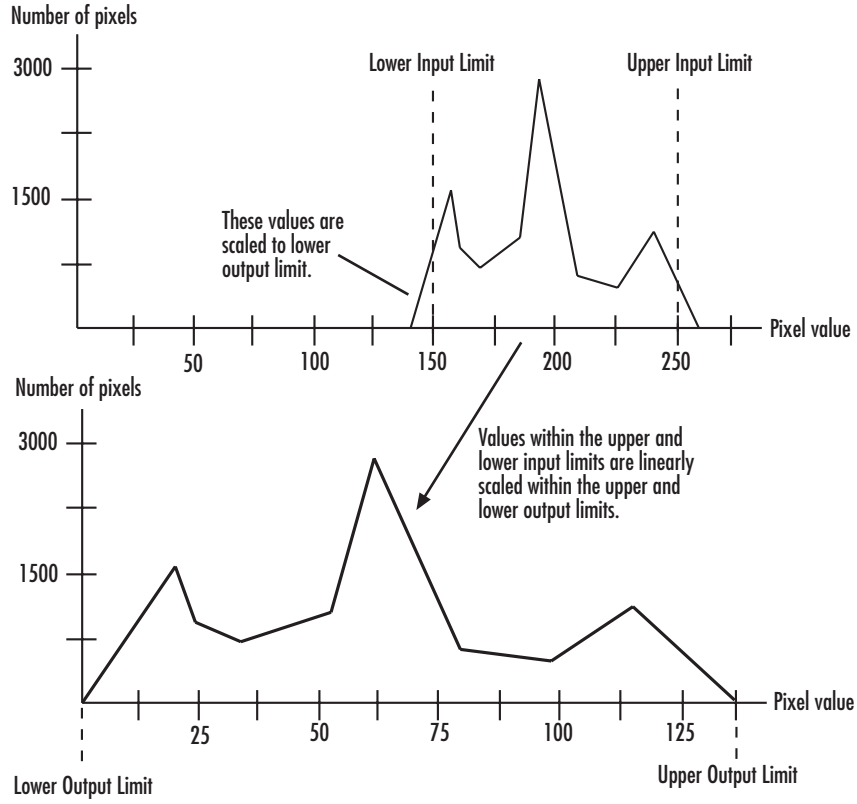
Analysis & Enhancement

Description



Contrast Adjustment

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*]:

$$Output = \begin{cases} low_out, & Input \le low_in \\ low_out + (Input - low_in)\dfrac{high_out - low_out}{high_in - low_in}, & low_in < Input < high_in \\ high_out, & Input \ge high_in \end{cases}$$

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

Use the **Adjust pixel values from** parameter to specify the upper and lower input limits. If you select `Full input data range [min max]`, the block uses the minimum input value as the lower input limit and the maximum input value as the upper input limit. If you select `User-defined`, the **Range [low high]** parameter appears on the block. Enter a two-element vector of scalar values, where the first element corresponds to the lower input limit and the second element corresponds to the upper input limit. If you select `Range determined by saturating outlier pixels`, the **Percentage of pixels to saturate [low high] (in %)**, **Specify number of histogram bins (used to**

calculate the range when outliers are eliminated), **Number of histogram bins** parameters appear on the block. The block uses these parameter values to calculate the input limits in this three-step process:

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

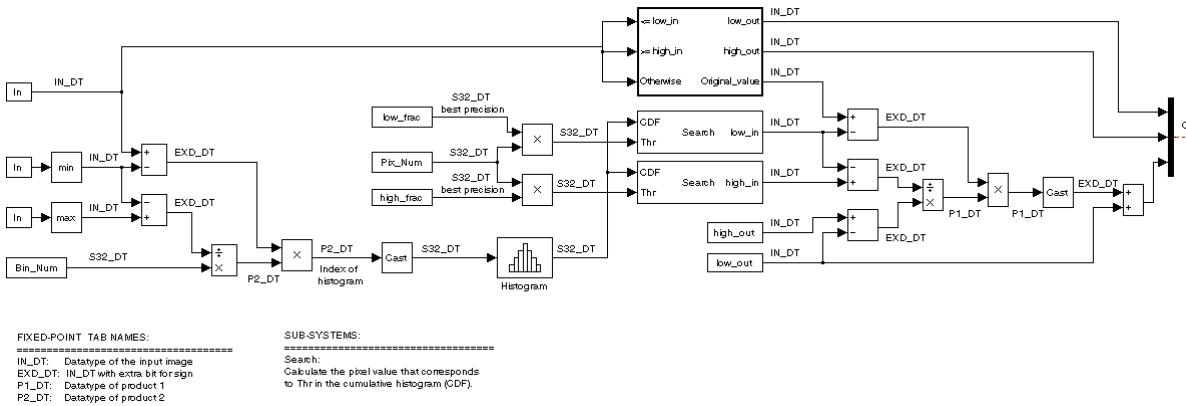
Use the **Adjust pixel values to** parameter to specify the upper and lower output limits. If you select `Full data type range`, the block uses the minimum value of the input data type as the lower output limit and the maximum value of the input data type as the upper output limit. If you select `User-defined range`, the **Range [low high]** parameter appears on the block. Enter a two-element vector of scalar values, where the first element corresponds to the lower output limit and the second element corresponds to the upper output limit.

Examples

See “Adjusting the Contrast in Intensity Images” on page 7-46 in the Video and Image Processing Blockset User’s Guide.

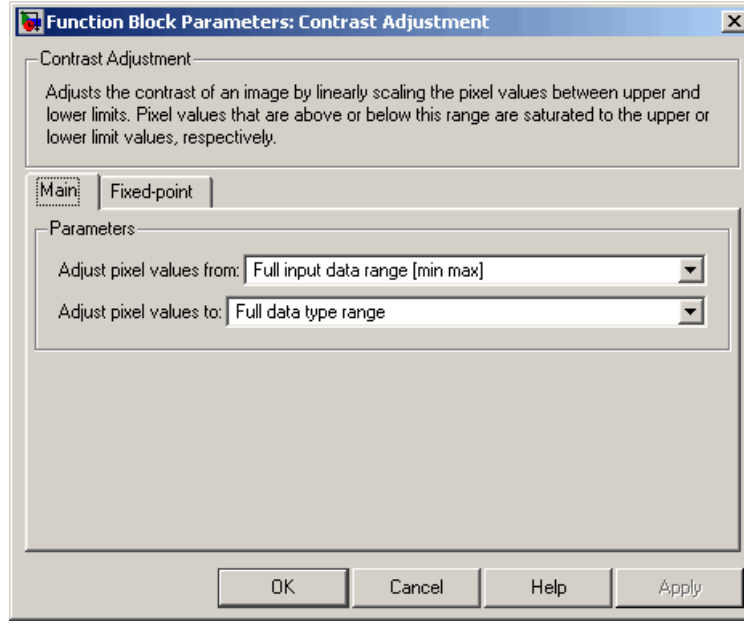
Fixed-Point Data Types

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



Dialog Box

The Contrast Adjustment dialog box appears as shown in the following figure.



Adjust pixel values from
Specify how to enter the upper and lower input limits. Your choices are `Full input data range [min max]`, `User-defined`, and `Range determined by saturating outlier pixels`.

Range [low high]
Enter a two-element vector of scalar values. The first element corresponds to the lower input limit, and the second element corresponds to the upper input limit. This parameter is visible if, for the **Adjust pixel values from** parameter, you select `User-defined`.

Percentage of pixels to saturate [low high] (in %)
Enter a two-element vector. The block calculates the lower input limit such that the percentage of pixels with values smaller than the lower limit is at most the value of the first element. It calculates the upper input limit similarly. This parameter is visible if, for the **Adjust pixel values from** parameter, you select `Range determined by saturating outlier pixels`.

Specify number of histogram bins (used to calculate the range when outliers are eliminated)
Select this check box to change the number of histogram bins. This parameter is editable if, for the **Adjust pixel values from** parameter, you select `Range determined by saturating outlier pixels`.

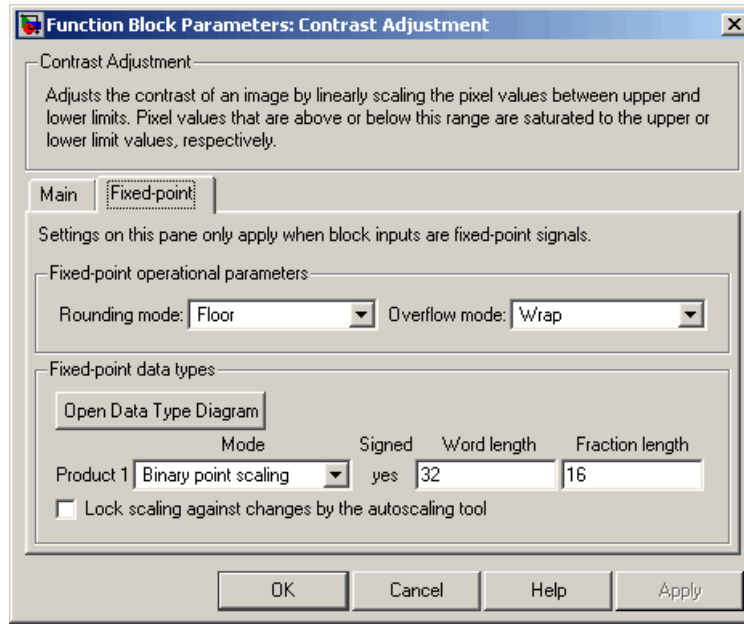
Number of histogram bins
Enter the number of histograms to use to calculate the scaled input values. This parameter is available if you select the **Specify number of histogram bins (used to calculate the range when outliers are eliminated)** check box.

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

Range [low high]

Enter a two-element vector of scalar values. The first element corresponds to the lower output limit and the second element corresponds to the upper output limit. This parameter is visible if, for the **Adjust pixel values to** parameter, you select `User-defined range`

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



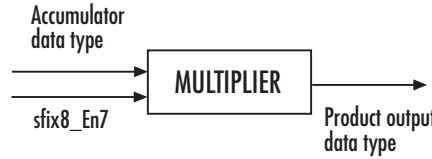
Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product 1

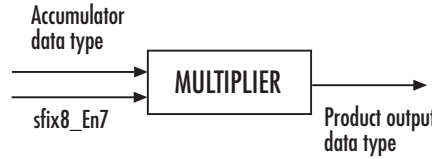


As shown in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.

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

Product 2



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

When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.

When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the product output. The bias of all signals in the Video and Image Processing Blockset 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

See Also

2-D Histogram	Video and Image Processing Blockset
Histogram Equalization	Video and Image Processing Blockset

Purpose

Remove motion artifacts by deinterlacing input video signal

Library

Analysis & Enhancement

Description

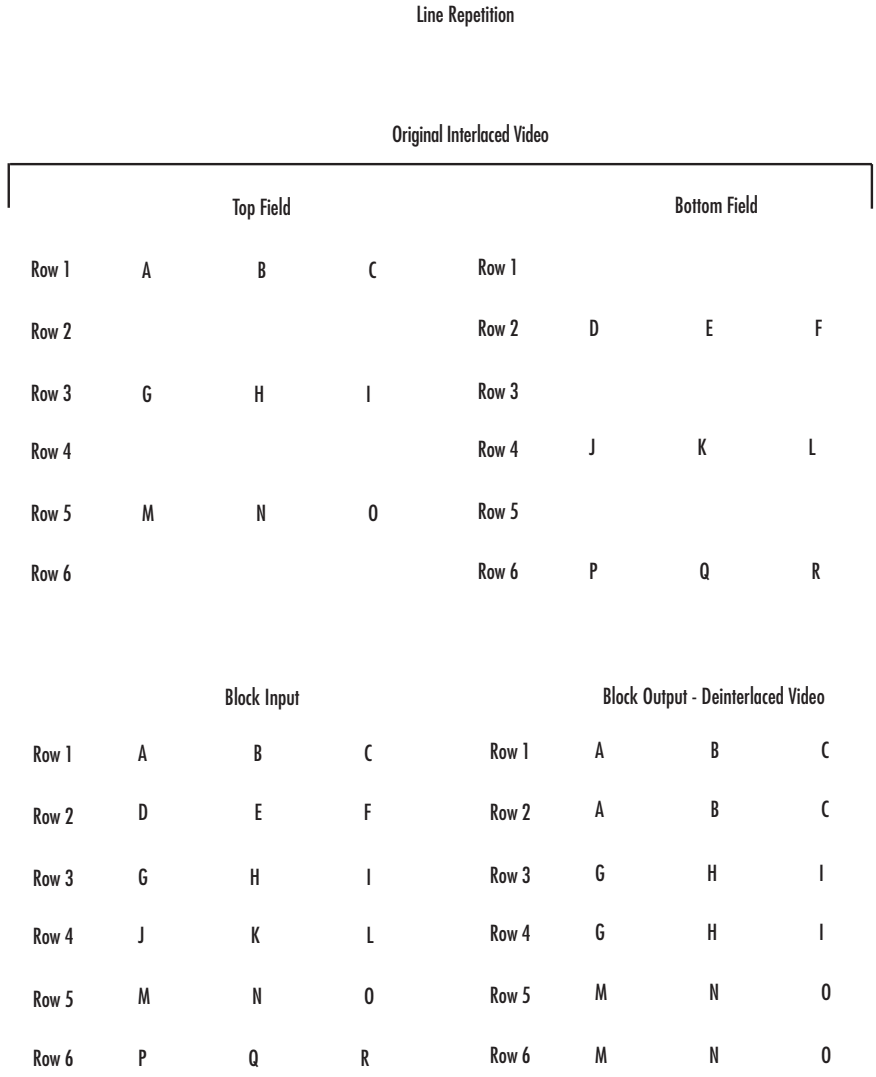


The Deinterlacing block takes the input signal, which is the combination of the top and bottom fields of the interlaced video, and converts it into deinterlaced video using line repetition, linear interpolation, or vertical temporal median filtering.

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 integers • 8-, 16-, and 32-bit unsigned integers	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

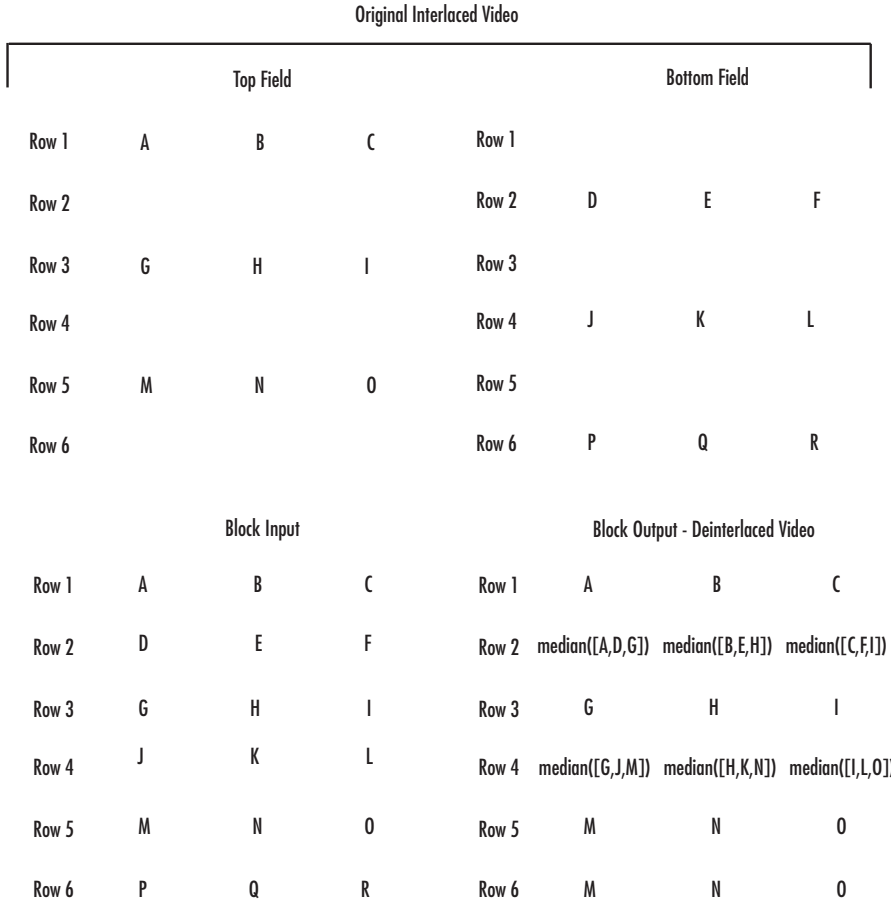
Original Interlaced Video

	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`.

Vertical Temporal Median Filtering



Example

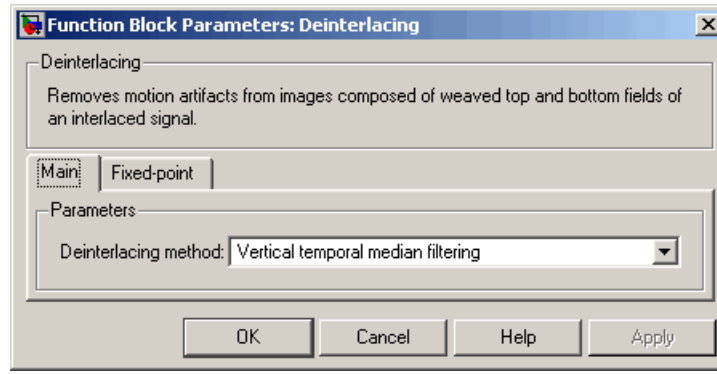
The following example shows you how to use the Deinterlacing block to remove motion artifacts from an image.

1 Open the example model by typing

```
doc_deinterlace
```

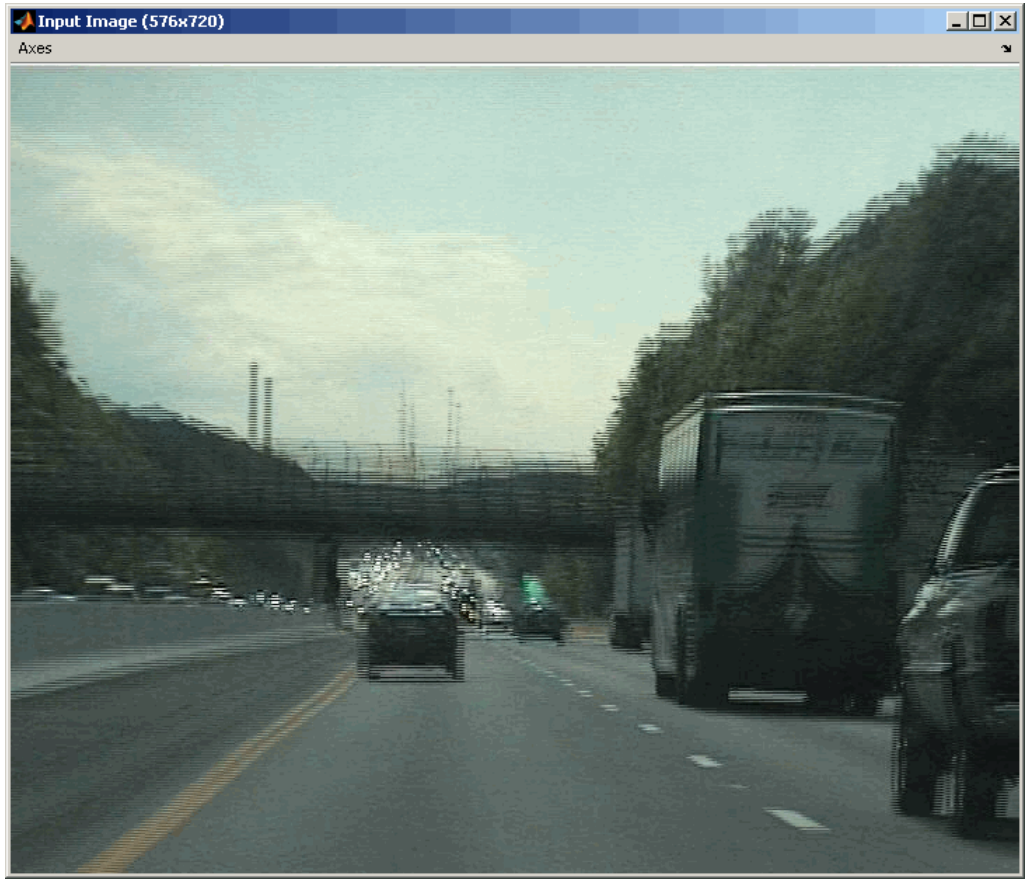
at the MATLAB command prompt.

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

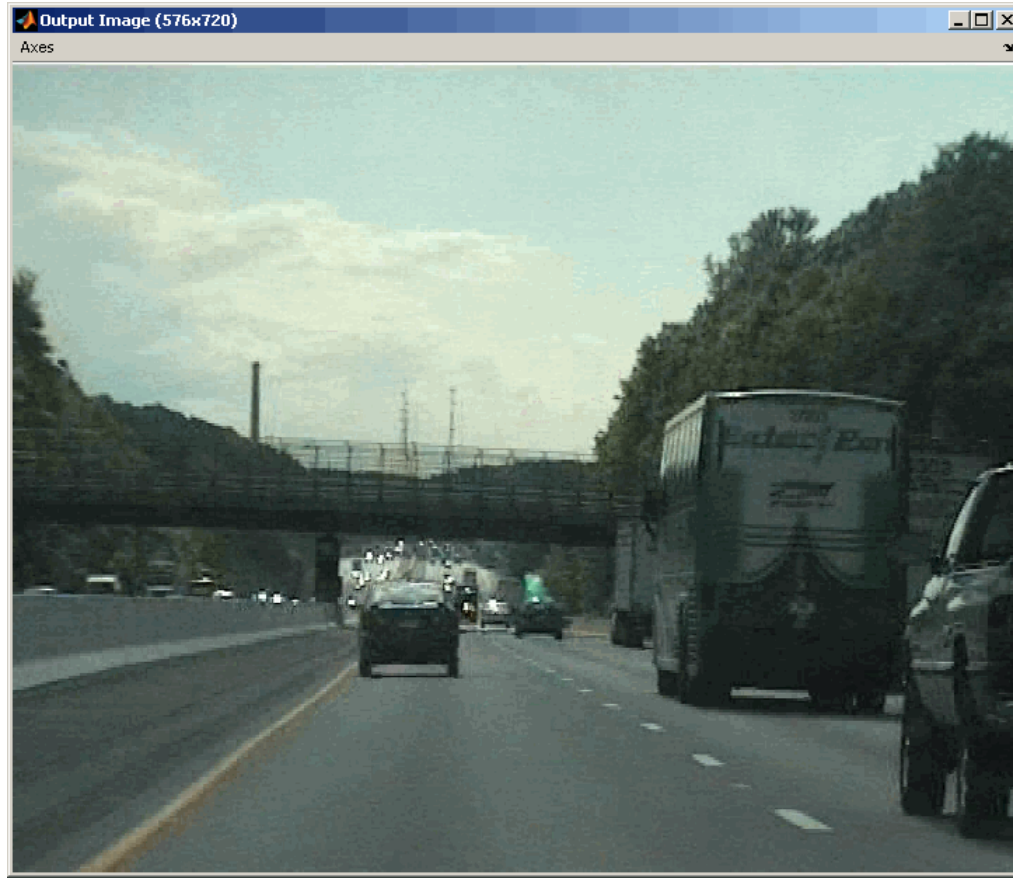


3 Run the model.

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

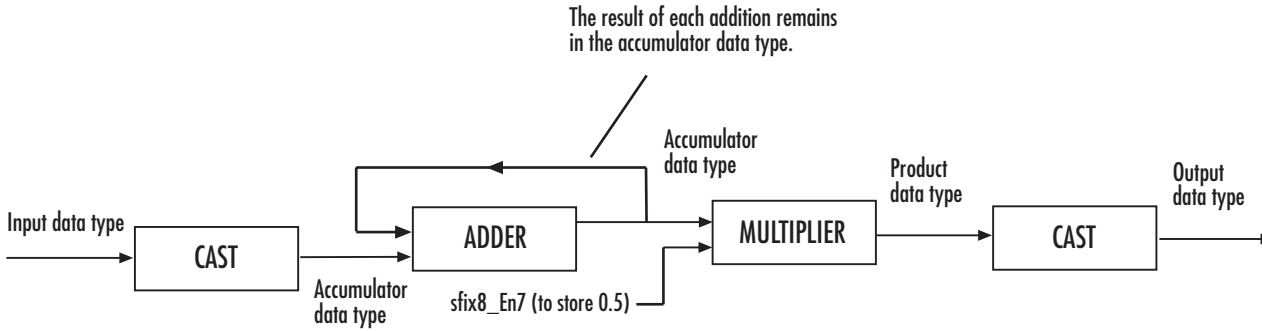


The clearer output image appears in the Output Image window.



Fixed-Point Data Types

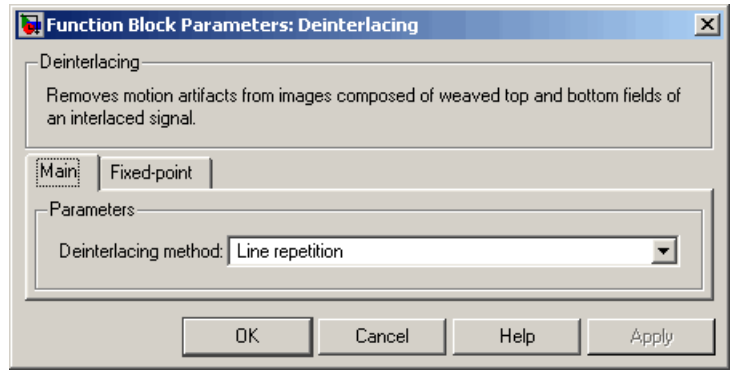
The following diagram shows the data types used in the Deinterlacing block for fixed-point signals.



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

Dialog Box

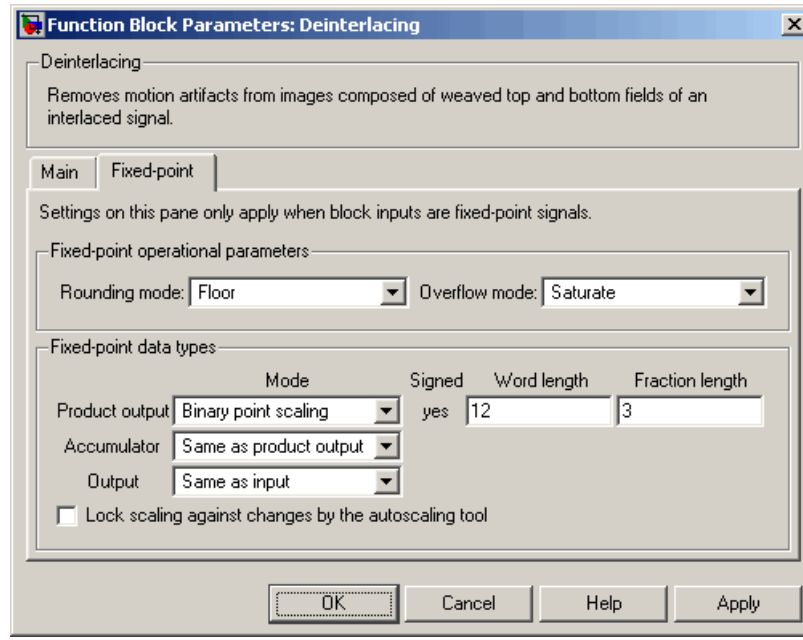
The **Main** pane of the Deinterlacing dialog box appears as shown in the following figure.



Deinterlacing method

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

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



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

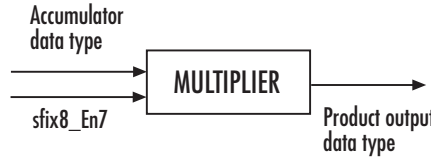
Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output



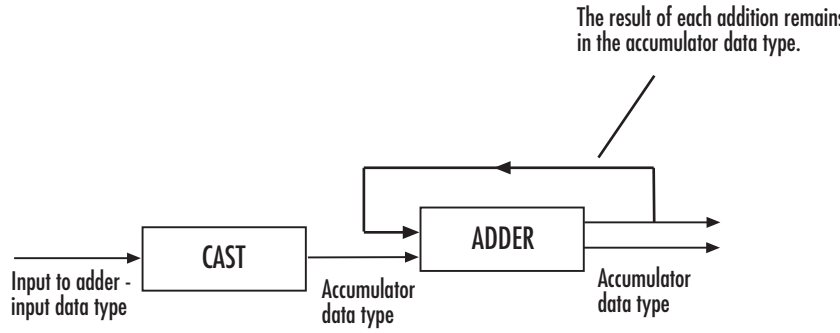
As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

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 is 0.

Accumulator



As depicted in the previous figure, inputs to the accumulator are cast to the accumulator data type. The output of the adder remains in the accumulator data type as each element of the input

is added to it. Use this parameter to specify how to designate this accumulator word and fraction lengths:

- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as input`, these characteristics match those of the input.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset 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 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.	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
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

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].

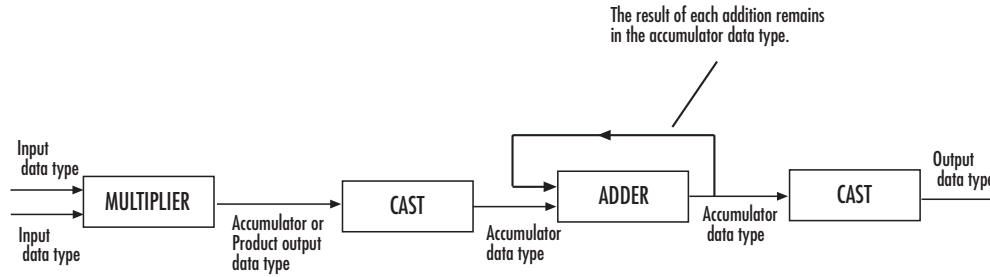
Use the **Sensor alignment** parameter to specify the alignment of the input image. Select the sequence of R, G and B pixels that correspond to the 2-by-2 block of pixels in the top-left corner of the image. You specify

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`.

Note Both methods use symmetric padding at the image boundaries. For more information, see the 2-D Pad block reference page.

Fixed-Point Data Types

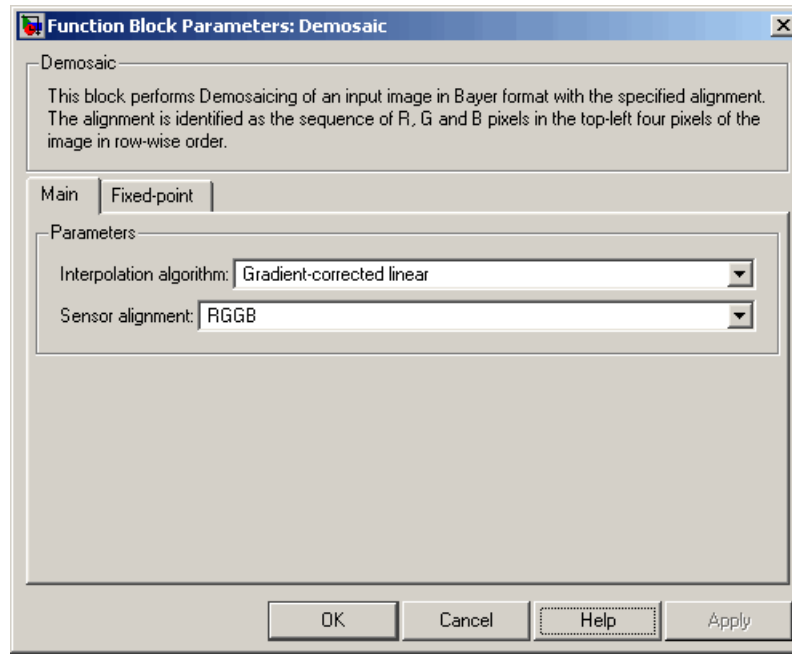
The following diagram shows the data types used in the Demosaic block for fixed-point signals.



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.



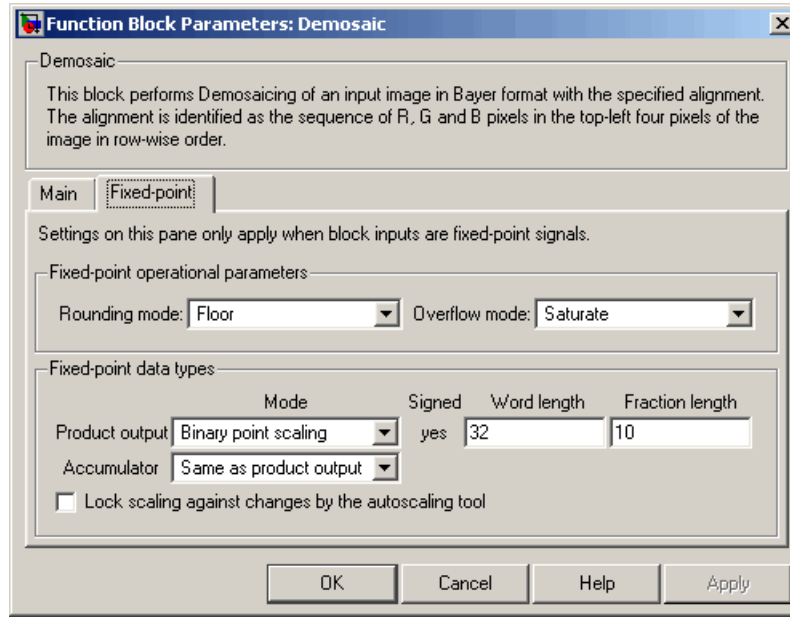
Interpolation algorithm

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.

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



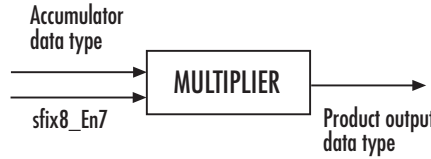
Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Product output



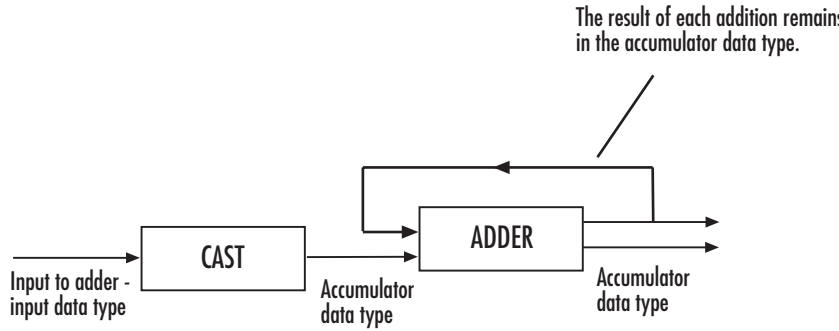
As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

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

When you select `Binary point scaling`, you can enter the word length and the fraction length of the product output, in bits.

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

Accumulator



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

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

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

Lock scaling against changes by the autoscaling tool
Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset 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

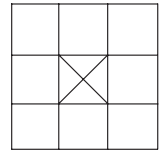
Morphological Operations

Description

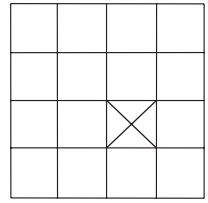


Dilation

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 below.



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 below.



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 integers • 8-, 16-, and 32-bit unsigned integers	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. This block supports a signal represented by a Simulink virtual bus.

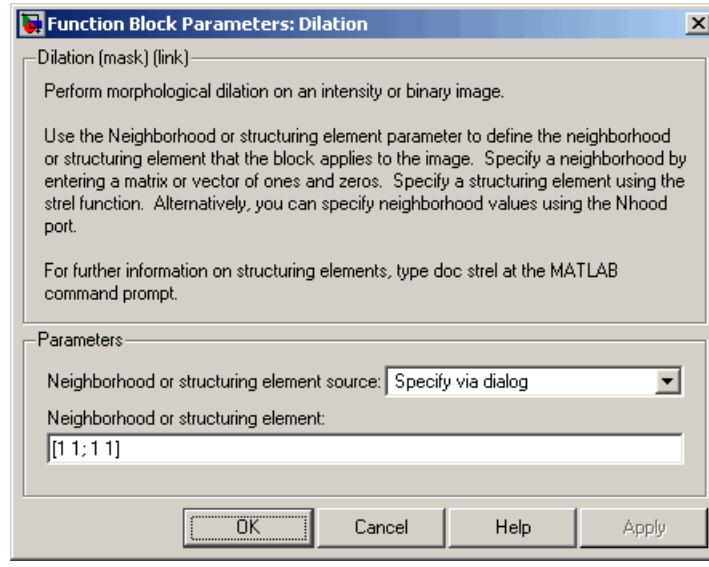
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select `Specify via dialog`, the **Neighborhood or structuring element** parameter appears in the dialog box. If you select `Input port`, the Nhood port appears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

Use the **Neighborhood or structuring element** parameter to define the neighborhood or structuring element that the block applies to the image. Specify a neighborhood by entering a matrix or vector of 1s and 0s. Specify a structuring element with the `strel` function from the Image Processing Toolbox. If the structuring element is decomposable

into smaller elements, the block executes at higher speeds due to the use of a more efficient algorithm. If you enter an array of STREL objects, the block applies each object to the entire matrix in turn.

Dialog Box

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



Neighborhood or structuring element source

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

Neighborhood or structuring element

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

structuring element source parameter, you select `Specify via dialog`.

References

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

See Also

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

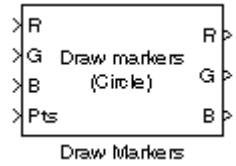
Purpose

Draw markers by embedding predefined shapes on output image

Library

Text & Graphics

Description



Draw Markers

The Draw Markers block can draw multiple circles, x-marks, plus signs, stars, or squares on images by overwriting pixel values. As a result, the shapes are embedded on the output image.

This block uses Bresenham's circle drawing algorithm to draw circles and Bresenham's line drawing algorithm to draw all other markers.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
R, G, B	Scalar, vector, or matrix that represents one plane of the input RGB video stream. Inputs to the R, G, and B ports must have the same dimensions and data type.	Same as I port	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Pts	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 integers • 8-, 16-, and 32-bit unsigned integers	No
ROI	Four-element vector of integers that define a rectangular area in which to draw the shapes. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the area. The second two elements represent the height and width of the area.	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
Output	Scalar, vector, or matrix of pixel values that contain the shape(s)	Same as I port	No

The output signal is the same size and data type as the inputs to the I, R, G, and B ports.

Use the **Input image type** parameter to specify the type of image input to the block. Your choices are `Intensity` or `RGB`.

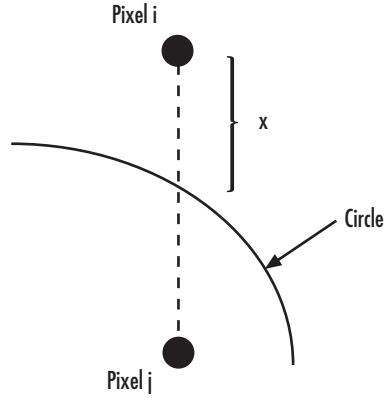
Use the **Marker shape** parameter to specify the shape of the markers. Your choices are `Circle`, `X-mark`, `Plus`, `Star`, or `Square`.

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

Use the **Draw markers in** parameter to define the area in which to draw the markers. If you select `Entire image`, you can draw markers in the entire image. If you select `Specify region of interest via port`, the ROI port appears on the block. Enter a four-element vector of integer values, `[r c height width]`, where `r` and `c` are the row and column coordinates of the upper-left corner of the area, and `height` and `width` represent the height (in rows) and width (in columns) of the area. If you specify values that are outside the image, the block clips the values to the image boundaries.

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

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



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

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

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

Intensity Images

Use the **Intensity** parameter to specify the appearance of the markers. If you select `Black`, the marker border is black. If you select `White`, the marker border is white. If you select `User-specified value`, the **Intensity value** parameter appears in the dialog box. Enter a scalar intensity value between the minimum and maximum values that can be represented by the data type of the input image. The minimum value corresponds to black and the maximum value corresponds to white. If you enter a value outside this range, the block produces an error. You can specify a separate intensity value for each marker by entering a vector of intensities that is the same length as the total number of markers.

If you select the **Filled** check box, the **Intensity** parameter controls the appearance of the border and interior of the marker. Use the **Opacity factor (between 0 and 1)** parameter to specify the opacity of the shading inside the marker, where 0 is transparent and 1 is opaque.

Note If you are generating code and you select the **Filled** check box, the word length of the block input(s) must be less than or equal to 16 bits.

Color Images

Use the **Intensity** parameter to specify the appearance of the markers. If you select `Black`, the marker border is black. If you select `White`, the marker border is white. If you select `User-specified value`, the **RGB values** parameter appears in the dialog box. Enter a three-element vector of values between the minimum and maximum values that can be represented by the data type of the input image. The minimum value corresponds to an absence of color and the maximum value corresponds

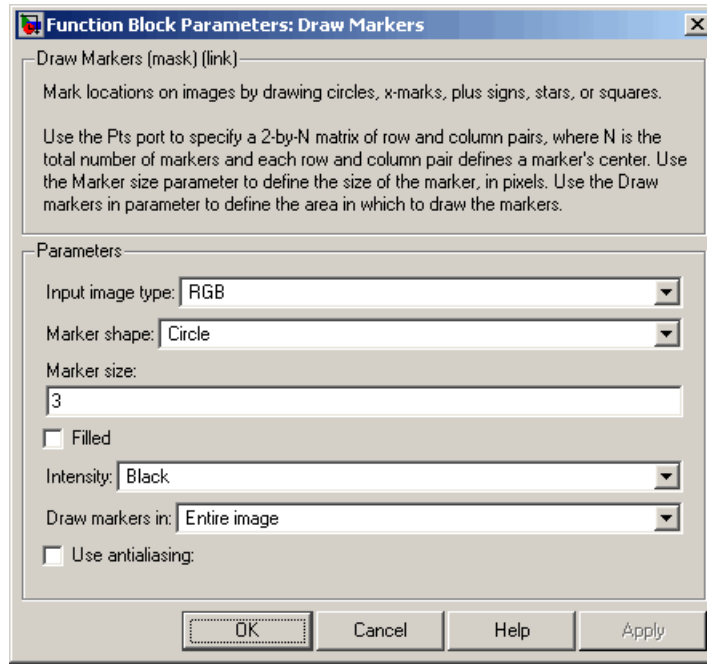
to the saturation of color. If you enter a value outside the data type range, the block produces an error. You can specify a separate RGB triplet for each shape by entering a 3-by-N matrix, where N is the total number of shapes.

If you select the **Filled** check box, the **Intensity** parameter controls the appearance of the border and interior of the marker. Use the **Opacity factor (between 0 and 1)** parameter to specify the opacity of the shading inside the marker, where 0 is transparent and 1 is opaque.

Note If you are generating code and you select the **Filled** check box, the word length of the block input(s) must be less than or equal to 16 bits.

Dialog Box

The Draw Markers dialog box appears as shown in the following figure.



Input image type

Specify the type of image input to the block. Your choices are `Intensity` or `RGB`.

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`.

Intensity

Specify the appearance of the marker. If you select `Black`, the marker border is black. If you select `White`, the marker border is white. If you select `User-specified value`, either the **Intensity value** or the **RGB values** parameter appears in the dialog box. If you select the **Filled** check box, this parameter controls the appearance of the border and interior of each marker.

Intensity value

Enter a scalar intensity value for the interior of the marker. This value must be between the minimum and maximum values that can be represented by the data type of the input image. The minimum value corresponds to black and the maximum value corresponds to white. This parameter is visible if, for the **Input image type** parameter, you select `Intensity` and, for the **Intensity** parameter, you select `User-specified value`. Tunable.

RGB values

Enter a three-element vector of values between the minimum and maximum values that can be represented by the data type of the input image. The minimum value corresponds to an absence of color and the maximum value corresponds to the saturation of color. This parameter is visible if, for the **Input image type** parameter, you select `RGB` and, for the **Intensity** 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. Tunable.

Draw markers in

Define the area in which to draw the markers. If you select `Entire image`, you can draw markers in the entire image. If you select `Specify region of interest via port`, the ROI port appears on the block. Enter a four-element vector, `[r c height width]`, where `r` and `c` are the row and column coordinates of the upper-left corner of the area, and `height` and `width` represent the height (in rows) and width (in columns) of the area.

Use antialiasing

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

References

[1] Gupta, Satish and Robert F. Sproull,*Filtering Edges for Gray-Scale Displays*, Computer Graphics, vol. 15, no. 3, August 1981.

See Also

Draw Shapes	Video and Image Processing Blockset
Insert Text	Video and Image Processing Blockset

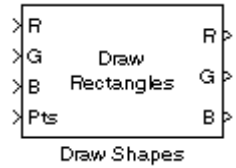
Purpose

Draw rectangle around region of interest (ROI)

Library

`vipobslib`

Description



The Draw Shape block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset. 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	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the input RGB video stream	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean	No
ROI	Four-element vector of integers. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the ROI. The second two elements represent the height and width of the ROI.	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
Output	Scalar, vector, or matrix of pixel values that contains the region of interest	Same as I port	No

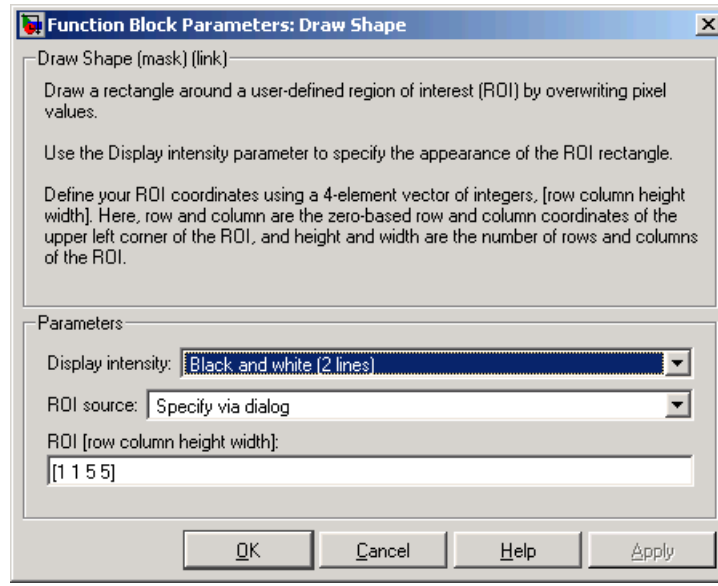
The output signal is the same size and data type as the input to the I port. This block supports a signal represented by the Simulink virtual bus.

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

Dialog Box

The Draw Shape dialog box appears as shown in the following figure.



Display intensity

Specify the appearance of the ROI rectangle. If you select `Black` or `White`, the rectangle is black or white. If you select `Black and white (2 lines)`, the rectangle is created by a black line on the outside and a white line on the inside. If you select `User-specified intensity`, the **Intensity value (0 to 1)** parameter appears in the dialog box.

Intensity value

Enter a scalar intensity value from 0 to 1, where 0 corresponds to black and 1 corresponds to white. This parameter is visible if, for the **Display intensity** parameter, you select `User-specified intensity`. Tunable.

ROI source

Specify how to enter your ROI coordinates. If you select `Specify via dialog`, the **ROI [row column height width]** parameter

appears on the dialog. If you select `Input port`, the ROI port appears on the dialog. 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.

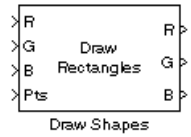
Purpose

Draw rectangles, lines, polygons, or circles on images

Library

Text & Graphics

Description



Draw Shapes

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
I	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
R, G, B	Scalar, vector, or matrix that represents one plane of the input RGB video stream. Inputs to the R, G, and B ports must have the same dimensions and data type.	Same as I port	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Pts	Use integer values to define zero-based shape coordinates. If you enter noninteger values, the block rounds them to the nearest integer.	• 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 integers • 8-, 16-, and 32-bit unsigned integers	No
ROI	Four-element vector of integers that defines a rectangular area in which to draw the shapes. The first two elements represent the zero-based row and column coordinates of the upper-left corner of the area. The second two elements represent the height and width of the area.	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
Output	Scalar, vector, or matrix of pixel values that contain the shape(s)	Same as I port	No

The output signal is the same size and data type as the inputs to the I, R, G, and B ports.

Use the **Input image type** parameter to specify the type of image input to the block. Your choices are `Intensity` or `RGB`.

Use the **Shape** parameter to specify the type of shape(s) to draw. Your choices are `Rectangles`, `Lines`, `Polygons`, or `Circles`.

Use the **Draw shapes in** parameter to define the area in which to draw the shapes. If you select `Entire image`, you can draw shapes in the entire image. If you select `Specify region of interest via port`, the ROI port appears on the block. Enter a four-element vector of integer values, `[r c height width]`, where `r` and `c` are the row and column coordinates of the upper-left corner of the area, and `height` and `width` represent the height (in rows) and width (in columns) of the area. If you specify values that are outside the image, the block sets the values to the image boundaries.

Working with Intensity Images

If, for the **Input image type** parameter you choose `Intensity`, the **Border intensity** parameter appears on the block. Use this parameter to determine the appearance of the rectangle(s), line(s), polygon(s), or circle(s). If you select `Black`, the border is black. If you select `White`, the border is white. If you select `User-specified value`, the **Intensity value** parameter appears in the dialog box. Enter a scalar intensity value between the minimum and maximum values that can be represented by the data type of the input image. The minimum value corresponds to black and the maximum value corresponds to white. If you enter a value outside this range, the block produces an error message. You can specify a separate intensity value for each shape's border by entering a vector of intensities that is the same length as the total number of shapes.

If you select the **Fill shapes** check box, the **Fill value intensity** and **Opacity factor (between 0 and 1)** parameters appear in the dialog box. Use the **Fill value intensity** parameter to specify the intensity of the shading inside the shape. The word length of this value cannot exceed 32 bits. 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.

Working with Color Images

If, for the **Input image type** parameter you choose `RGB`, the **Border color** parameter appears on the block. Use this parameter to determine the appearance of the rectangle(s), line(s), polygon(s), or circle(s). If you select `Black`, the shape is black. If you select `White`, the shape is white. If you select `User-specified value`, the **RGB values** parameter appears in the dialog box. Enter a three-element vector of values between the minimum and maximum values that can be represented by the data type of the input image. The minimum value corresponds to the absence of color and the maximum value corresponds to the saturation of color. If you enter a value outside the data type range, the block produces an error message. You can specify a separate RGB triplet for each shape by entering a 3-by-N matrix, where N is the total number of shapes.

If you select the **Fill shapes** check box, the **Fill value color** and **Opacity factor (between 0 and 1)** parameters appear in the dialog box. Use the **Fill value color** parameter to specify the color of the shading inside the shape. For simulation, the word length of these values cannot exceed 32 bits. 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.

Drawing Rectangles

If to draw one rectangle, for the **Shape** parameter, choose `Rectangles`. The input to the Pts port must be a four-element vector of the form `[r`

`c height width]`, where `r` and `c` are the zero-based row and column coordinates of the upper-left corner of the rectangle, and `height` and `width` represent the height, in rows, and width, in columns, of the rectangle. Here, `height` and `width` must be greater than 0.

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

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

Drawing Lines and Polylines

If you want to draw one line, for the **Shape** parameter, choose `Lines`. The input to the Pts port must be a four-element vector of the form `[r1 c1 r2 c2]`, where `r1` and `c1` are the row and column coordinates of the beginning of the line and `r2` and `c2` are the row and column coordinates of the end of the line.

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

If you want to draw N polylines, for the **Shape** parameter, choose `Lines`. If the biggest polyline has L-1 line segments, the input to the Pts port must be a 2L-by-N matrix, where each column of the matrix corresponds to a different polyline, as described in the previous paragraph. If some polylines are shorter than others, repeat the ending coordinates to fill

the polyline matrix. The block produces an error message if the number of rows is less than two or is not a multiple of two.

If you select the **Use antialiasing** check box, the block performs the smoothing algorithm described in "Filtering Edges for Gray-Scale Displays" by Satish Gupta and Robert Sproull.

Drawing Polygons

If you want to draw one polygon, for the **Shape** parameter, choose `Polygons`. For a polygon with L line segments, the input to the Pts port must be a vector of size 2L, `[r1 c1 r2 c2 ... rL cL]`. Here, `r1` and `c1` are the row and column coordinates of the beginning of the first line segment, `r2` and `c2` are the row and column coordinates of the end of the first line segment and the beginning of the second line segment, etc. The block connects `[r1 c1]` to `[rL cL]` to complete the polygon. The block produces an error if the number of rows is negative or not a multiple of two.

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

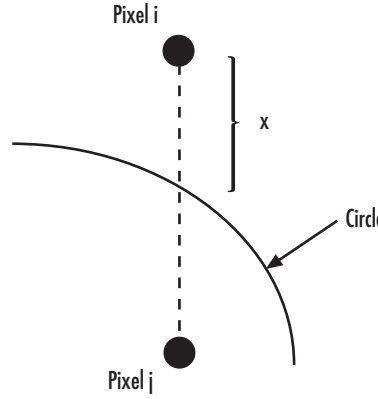
If you select the **Use antialiasing** check box, the block performs the smoothing algorithm described in "Filtering Edges for Gray-Scale Displays" by Satish Gupta and Robert F. Sproull.

Drawing Circles

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

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

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

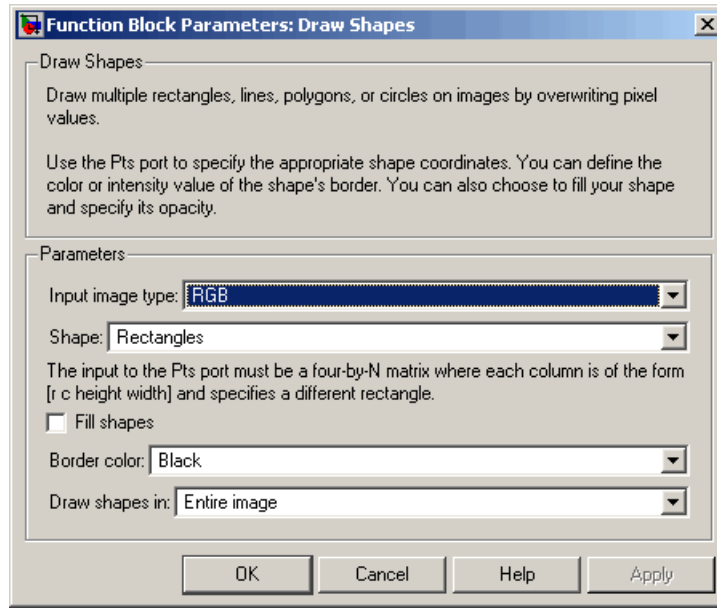


Then it uses the following equations to calculate the intensity at pixels i and j:

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

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

Dialog Box



Input image type

Specify the type of image input to the block. Your choices are `Intensity` or `RGB`.

Shape

Specify the type of shape(s) to draw. Your choices are `Rectangles`, `Lines`, `Polygons`, or `Circles`.

Fill shapes

Fill the shape with an intensity value or a color.

Fill value intensity

Specify the intensity of the shading inside the shape. This parameter is visible if, for the **Input image type** parameter, you select `Intensity` and you select the **Fill shapes** check box.

Fill value color

Specify the color of the shading inside the shape. This parameter is visible if, for the **Input image type** parameter, you select `RGB` and you select the **Fill shapes** check box.

Border intensity

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 **Intensity value** parameter appears in the dialog box. This parameter is visible if, for the **Input image type** parameter, you select `Intensity`.

Intensity value

Enter a scalar intensity value for the border or shading of the shape. This value must be between the minimum and maximum values that can be represented by the data type of the input image. The minimum value corresponds to black and the maximum value corresponds to white. This parameter is visible if, for the **Input image type** parameter, you select `Intensity` and, for the **Border intensity** parameter, you select `User-specified value`. 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.

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 **RGB values** parameter appears in the dialog box. This parameter is visible if, for the **Input image type** parameter, you select `RGB`.

RGB values

Enter a three-element vector of values between the minimum and maximum values that can be represented by the data type of the input image. The minimum value corresponds to the absence of color and the maximum value corresponds to the saturation

of color. This parameter is visible if, for the **Input image type** parameter, you select `RGB` and, for the **Border color** parameter, you select `User-specified value`. Tunable.

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`.

References

Gupta, Satish and Robert F. Sproull, “Filtering Edges for Gray-Scale Displays”, Computer Graphics, vol. 15, no. 3, August 1981.

See Also

Draw Markers	Video and Image Processing Blockset
Insert Text	Video and Image Processing Blockset

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 integers (not supported for the Canny method) • 8-, 16-, 32-bit unsigned integers (not supported for the Canny method)	No
Th	Matrix of intensity values	Same as I port	No
Edge	Matrix that represents a binary image	Boolean	No
Gv	Matrix of gradient values in the vertical direction	Same as I port	No
Gh	Matrix of gradient values in the horizontal direction	Same as I port	No
G45	Matrix of gradient values	Same as I port	No
G135	Matrix of gradient values	Same as I port	No

The output of the Gv, Gh, G45, and G135 ports is the same data type as the input to the I port. The input to the Th port must be the same data type as the input to the I port.

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.

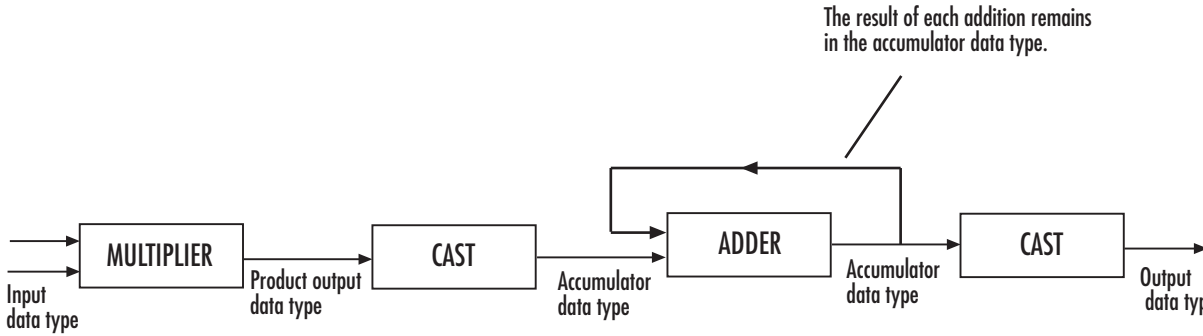
Use the **Threshold source** parameter to specify how to enter your threshold values. If you select `Specify via dialog`, the **Threshold [low high]** parameter appears in the dialog box. Enter the threshold values. If a pixel's magnitude in the gradient image, which is formed by convolving the input 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.

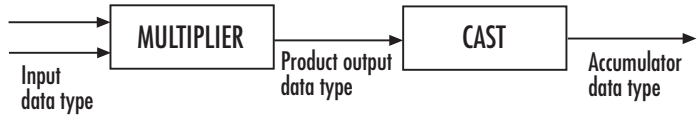
Fixed-Point Data Types

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

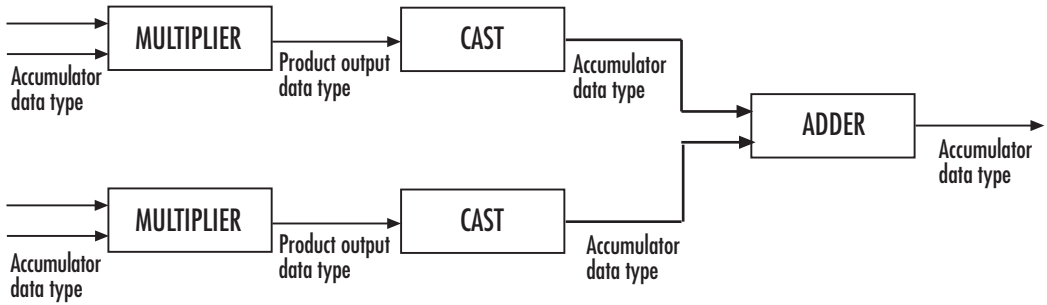


The block squares the threshold and compares it to the sum of the squared gradients to avoid using square roots.

Threshold:



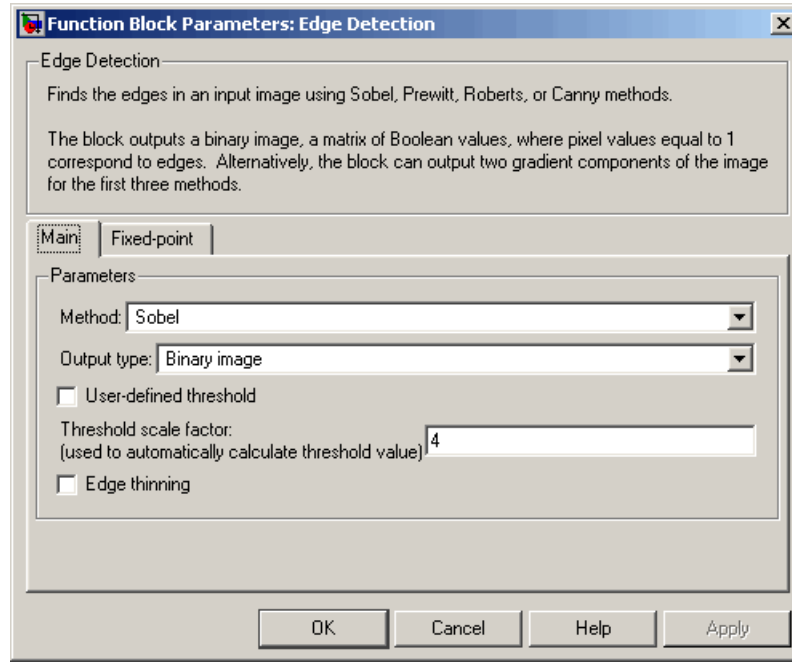
Gradients:



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.



Method

Select the method by which to perform edge detection. Your choices are `Sobel`, `Prewitt`, `Roberts`, or `Canny`.

Output type

Select the desired form of the output. If you select `Binary image`, the block outputs a matrix that is filled with ones, which correspond to edges, and zeros, which correspond to the background. If you select `Gradient components` and, for the **Method** parameter, you select `Sobel` or `Prewitt`, the block outputs the gradient components that correspond to the horizontal and vertical edge responses. If you select `Gradient components` and, for the **Method** parameter, you select `Roberts`, the block

outputs the gradient components that correspond to the 45 and 135 degree edge responses. If you select `Binary image and gradient components`, the block outputs both the binary image and the gradient components of the image. This parameter is visible if, for the **Method** parameter, you select `Sobel`, `Prewitt`, or `Roberts`.

User-defined threshold

If you select this check box, you can enter a desired threshold value. If you clear this check box, the block computes the threshold for you. This parameter is visible if, for the **Method** parameter, you select `Sobel`, `Prewitt`, or `Roberts`, and, for the **Output type** parameter, you select `Binary image` or `Binary image and gradient components`. This parameter is also visible if, for the **Method** parameter, you select `Canny`.

Threshold source

If you select `Specify via dialog`, enter your threshold value in the dialog box. If you choose `Input port`, use the Th input port to specify a threshold value that is the same data type as the input data. This parameter is visible if you select the **User-defined threshold** check box.

Threshold

Enter a threshold value that is within the range of your input data. This parameter is visible if, for the **Method** parameter, you select `Sobel`, `Prewitt`, or `Roberts`, you select the **User-defined threshold** check box, and, for **Threshold source** parameter, you select `Specify via dialog`. Tunable.

Threshold [low high]

Enter the low and high threshold values that define the weak and strong edges. This parameter is visible if, for the **Method** parameter, you select `Canny`. Then you select the **User-defined threshold** check box, and, for **Threshold source** parameter, you select `Specify via dialog`. Tunable.

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.

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. Tunable.

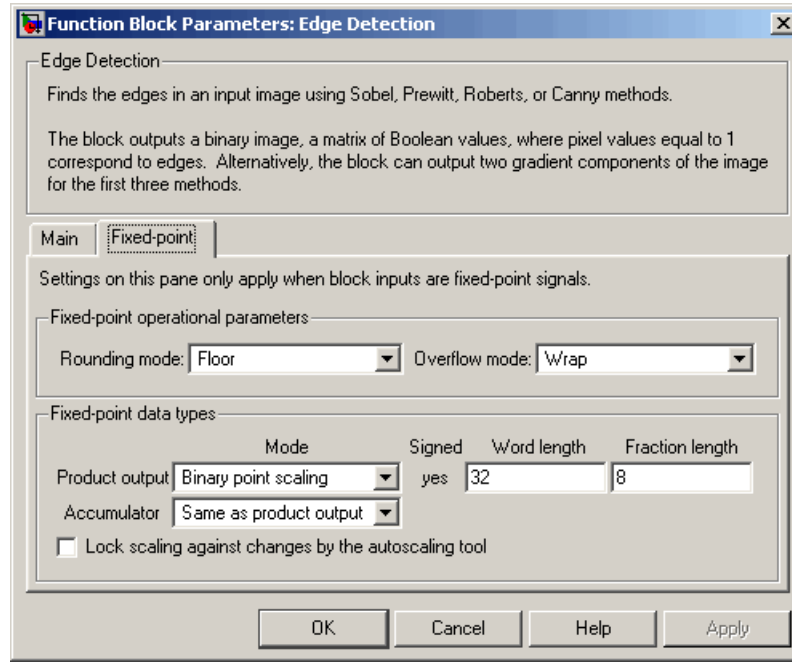
Edge thinning

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`.

Standard deviation of Gaussian filter

Enter the standard deviation of the Gaussian filter whose derivative is convolved with the input image. This parameter is visible if, for the **Method** parameter, you select `Canny`.

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



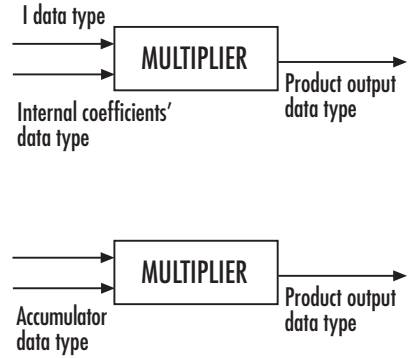
Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

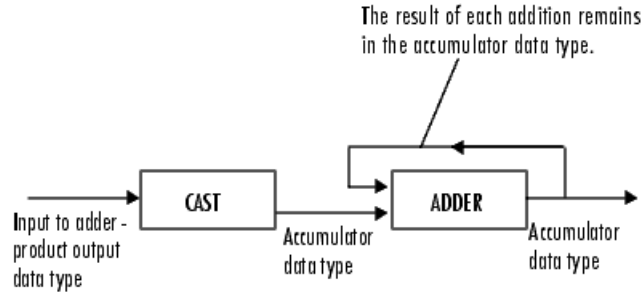
Product output



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 is 0.

Accumulator



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

- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as first input`, these characteristics match those of the first input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset 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 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset 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

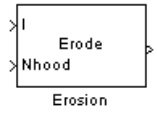
Purpose

Find local minima in binary or intensity images

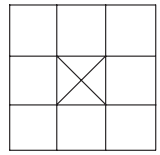
Library

Morphological Operations

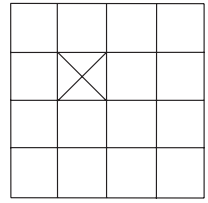
Description



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 below.



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 below.



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 integers • 8-, 16-, and 32-bit unsigned integers	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. This block supports a signal represented by a Simulink virtual bus.

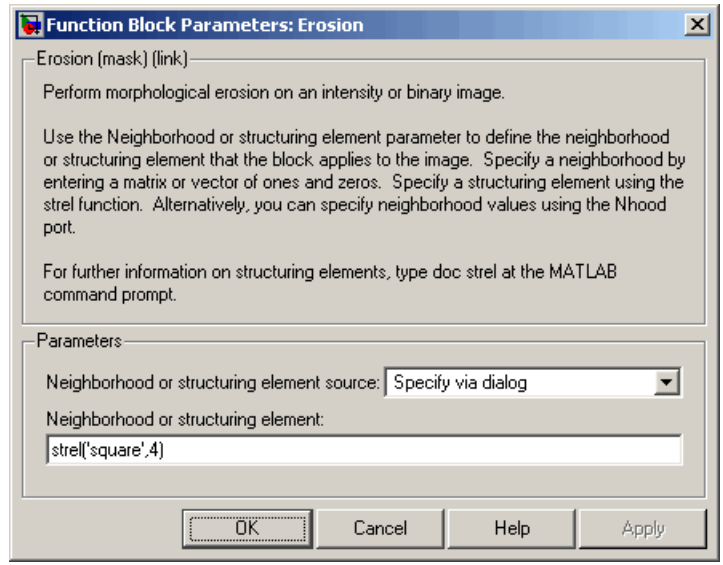
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select `Specify via dialog`, the **Neighborhood or structuring element** parameter appears in the dialog box. If you select `Input port`, the Nhood port appears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

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

use of a more efficient algorithm. If you enter an array of STREL objects, the block applies each object to the entire matrix in turn.

Dialog Box

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



Neighborhood or structuring element source

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

Neighborhood or structuring element

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

References

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

See Also

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

Find Local Maxima

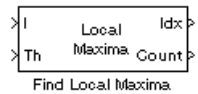
Purpose

Find local maxima in matrices

Library

Statistics

Description



Find Local Maxima

The Find Local Maxima block finds the local maxima in each input matrix. The block slides a neighborhood over the matrix and finds the maximum value within the neighborhood. If this maximum value is greater than or equal to a user-specified threshold, the block considers the value a valid local maximum. Then, it sets all the matrix values in the neighborhood to 0. The block repeats this 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, 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 integers • 8-, 16-, 32-bit unsigned integers	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 column represents the row coordinates and the second column represents the column coordinates.	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit unsigned integers	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. This block supports Simulink virtual buses.

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`.

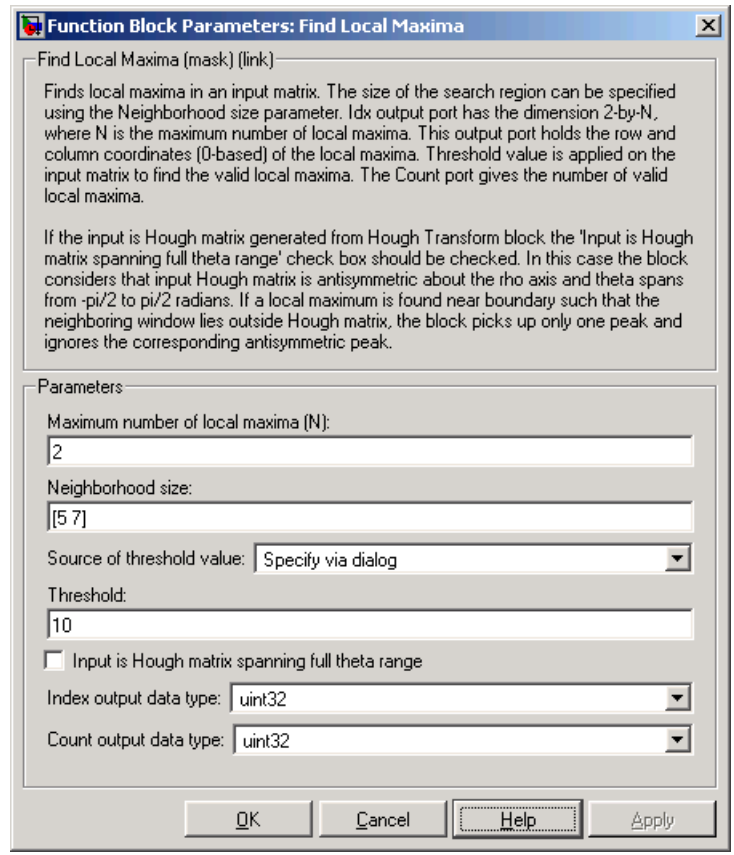
Use the **Count output data type** parameter to specify the data type of the Count port output. Your choices are `double`, `single`, `uint8`, `uint16`, or `uint32`.

Examples

See "Finding Lines in Images" on page 7-9 and "Measuring an Angle Between Lines" on page 7-17 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.



Maximum number of local maxima (N)

Specify the maximum number of maxima you want the block to find.

Neighborhood size
Specify the size of the neighborhood around the maxima over which the block zeros out the values. Enter a two-element vector of positive odd integers, [r c].

Source of threshold value
Specify how to enter the threshold value. If you select `Input port`, the Th port appears on the block. If you select `Specify via dialog`, the **Threshold** parameter appears in the dialog box.

Threshold
Enter a scalar value that represents the value all maxima should meet or exceed. This parameter is visible if, for the **Source of threshold value** parameter, you choose `Specify via dialog`.

Input is Hough matrix spanning full theta range
If you select this check box, the block assumes that the Hough port input is antisymmetric about the rho axis and theta ranges from -pi/2 to pi/2 radians.

Index output data type
Specify the data type of the Peaks port output. Your choices are `double`, `single`, `uint8`, `uint16`, or `uint32`.

Count output data types
Specify the data type of the Count port output. Your choices are `double`, `single`, `uint8`, `uint16`, or `uint32`.

See Also

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

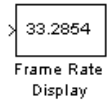
Purpose

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.

Port	Input	Supported Data Types	Complex Values Supported
Input	Signal whose frame rate you want to monitor	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No

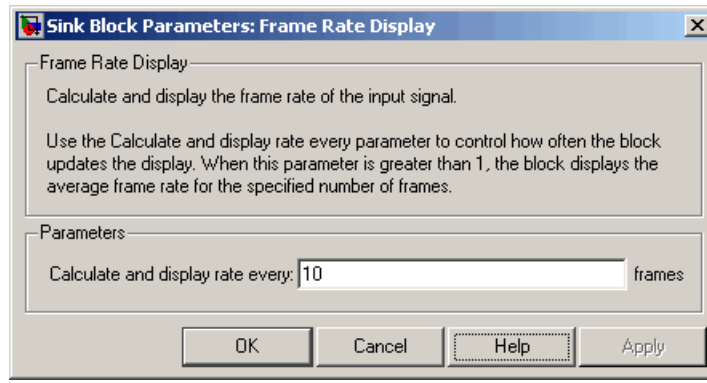
Use the **Calculate and display rate every** parameter to control how often the block updates the display. When this parameter is greater than 1, the block displays the average update rate for the specified number of video frames. For example, if you enter `10`, the block calculates the amount of time it takes for the model to pass 10 video frames to the block. It divides this time by 10 and displays this average video frame rate on the block.

Note If you do not connect the Frame Rate Display block to a signal line, the block displays the base (fastest) rate of the Simulink model.

Frame Rate Display

Dialog Box

The Frame Rate Display dialog box appears as shown in the following figure.



Calculate and display rate every

Use this parameter to control how often the block updates the display.

See Also

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

Purpose

Read video frames and audio samples from compressed multimedia file

Library

Sources

Description

The From Multimedia File block is a Signal Processing Blockset block. For more information, see the From Multimedia File block reference page in the Signal Processing Blockset documentation.

Gamma Correction

Purpose

Apply or remove gamma correction from images or video streams

Library

Conversions

Description





Use the Gamma Correction block to apply or remove gamma correction from an image or video stream. For input signals normalized between 0 and 1, the block performs gamma correction as defined by the following equations. Note that for integers and fixed-point data types, these equations are generalized by applying scaling and offset values specific to the data type:

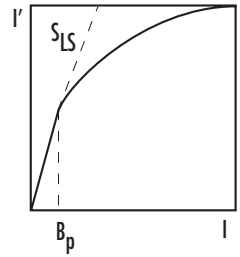
$$S_{LS} = \frac{1}{\frac{\gamma}{B_P^{(1/\gamma - 1)}} - \gamma B_P + B_P}$$

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

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

$$I' = \begin{cases} S_{LS} I, & I \le B_P \\ F_S I^{1/\gamma} - C_O, & I > B_P \end{cases}$$

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:

$$I = \begin{cases} I' / S_{LS}, & I' \le B_P \\ \left(\frac{I' + C_O}{F_S} \right)^{\gamma}, & I' > B_P \end{cases}$$

Typical gamma values range from 1 to 3. Most monitor gamma values range from 1.8 to 2.2. Check with the manufacturer of your hardware to obtain the exact gamma value. Gamma function parameters for some common standards are shown in the following table:

Standard	Slope	Break Point	Gamma
CIE L*	9.033	0.008856	3
Recommendation ITU-R BT.709-3, Parameter Values for the HDTV Standards for Production and International Programme Exchange	4.5	0.018	$20/9$
sRGB	12.92	0.00304	2.4

The properties of the input and output ports are summarized in the following table:

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the RGB video stream	• Double-precision floating point • Single-precision floating point • Fixed point (up to 16-bit word length) • 8- and 16-bit signed integers • 8- and 16-bit unsigned integers	No
I'	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the RGB video stream	Same as I port	No

This block supports Simulink virtual buses.

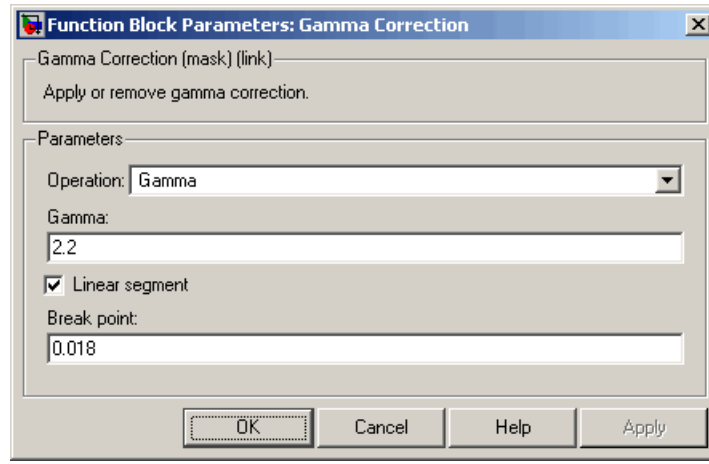
Use the **Operation** parameter to specify the block's operation. If you want to perform gamma correction, select `Gamma`. If you want to linearize the input signal, select `De-gamma`.

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. 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.



Operation

Specify the block's operation. Your choices are `Gamma` or `De-gamma`.

Gamma

If, for the **Operation** parameter, you select `Gamma`, enter the desired gamma value of the output video stream. This value must be greater than or equal to 1. If, for the **Operation** parameter, you select `De-gamma`, enter the gamma value of the input video stream. Tunable.

Linear segment

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

Break point

Enter a scalar value that indicates the *I*-axis value of the end of the linear segment. This parameter is visible if you select the **Linear segment** check box. Tunable.

References

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

`http://www.srgb.com/hpsrgbprof/`

See Also

Color Space Conversion	Video and Image Processing Blockset
`imadjust`	Image Processing Toolbox

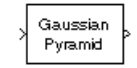
Purpose

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 10-338

Port	Output	Supported Data Types	Complex Values Supported
Input	In `Reduce` mode, the input can be a matrix of intensity values. In `Expand` mode, the input can be a scalar, vector, or matrix of intensity values.	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	No
Output	In `Reduce` mode, the output can be a scalar, vector, or matrix that represents one level of a Gaussian pyramid. In `Expand` mode, the output can be a matrix that represents one level of a Gaussian pyramid.	Same as Input port	No

Use the **Operation** parameter to specify whether to reduce or expand the input image. If you select `Reduce`, the block applies a lowpass filter and then downsamples the input image. If you select `Expand`, the block upsamples and then applies a lowpass filter to the input image.

Use the **Pyramid level** parameter to specify the number of times the block upsamples or downsamples each dimension of the image by a factor of 2. For example, suppose you have a 4-by-4 input image. You set the **Operation** parameter to `Reduce` and the **Pyramid level** to `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.

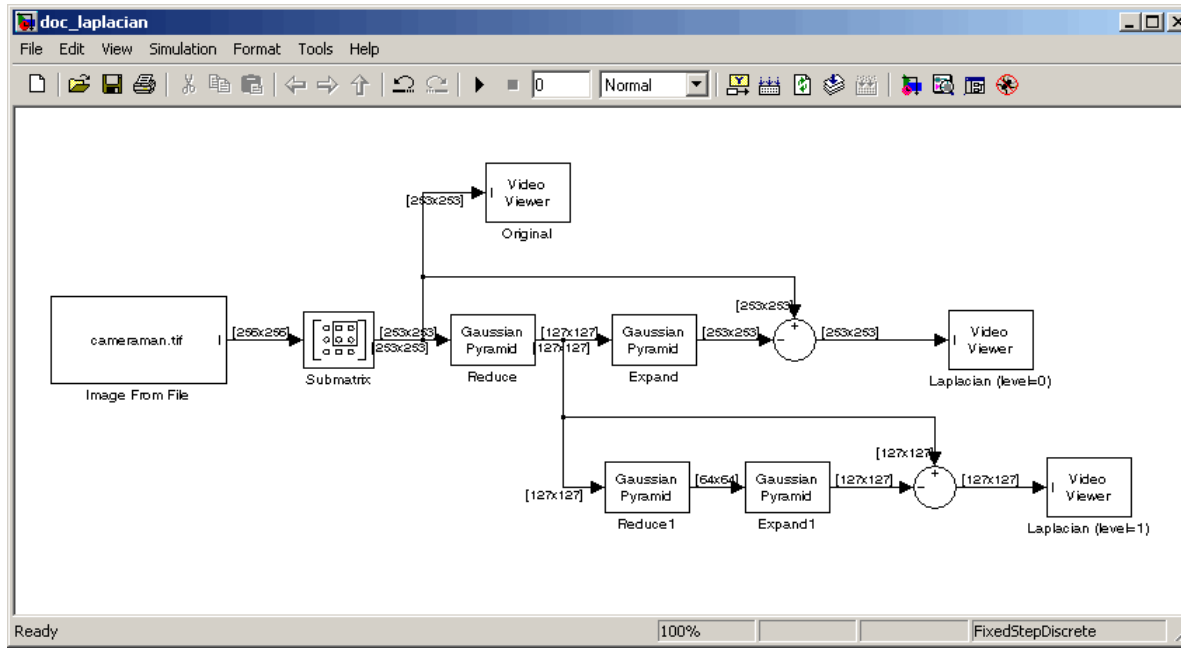
Examples

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.


Original (253x253)
Axes

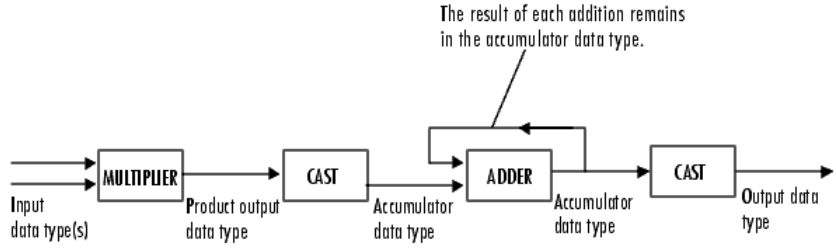

Laplacian (level=0) (253x253)
Axes



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

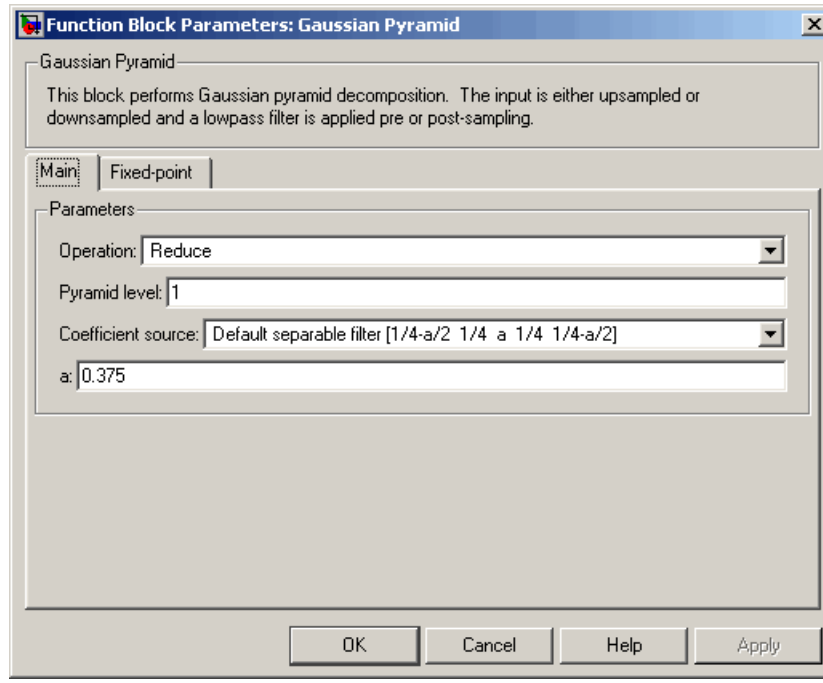
The following diagram shows the data types used in the Gaussian Pyramid block for fixed-point signals:



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.



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`.

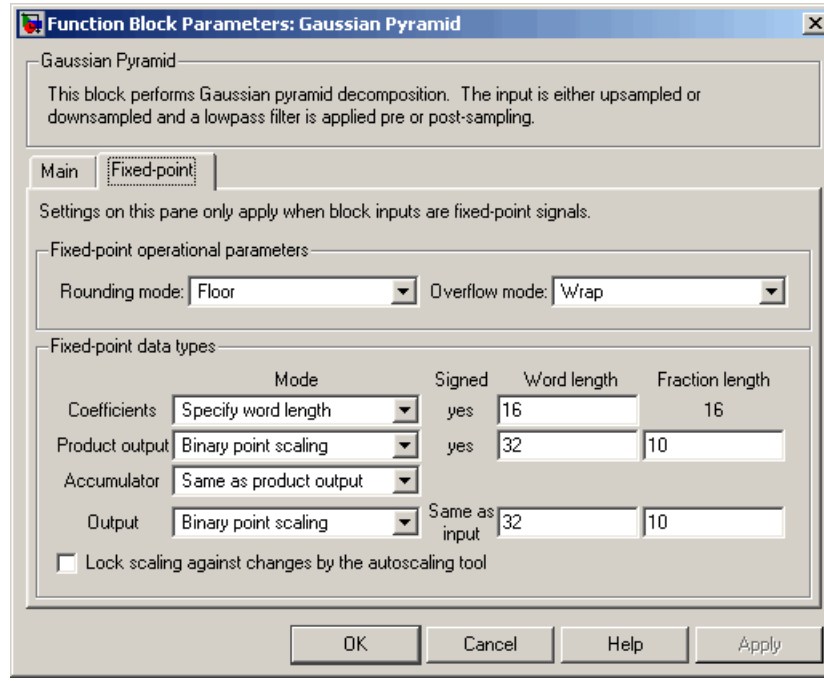
a

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.



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 is 0.

Product output

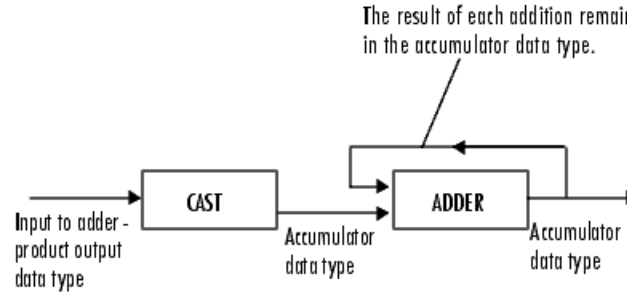


As shown in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate the product output word and fraction lengths.

- 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 is 0.

Accumulator



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

- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as input`, these characteristics match those of the input to the block.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset 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 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

See Also

Resize Video and Image Processing Blockset

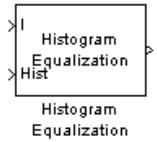
Purpose

Enhance contrast of images using histogram equalization

Library

Analysis & Enhancement

Description



The Histogram Equalization block enhances the contrast of images by transforming the values in an intensity image so that the histogram of the output image approximately matches a specified histogram.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	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 integers • 8-, 16-, 32-bit unsigned integers	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. This block supports a signal represented by a Simulink virtual bus.

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. 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.

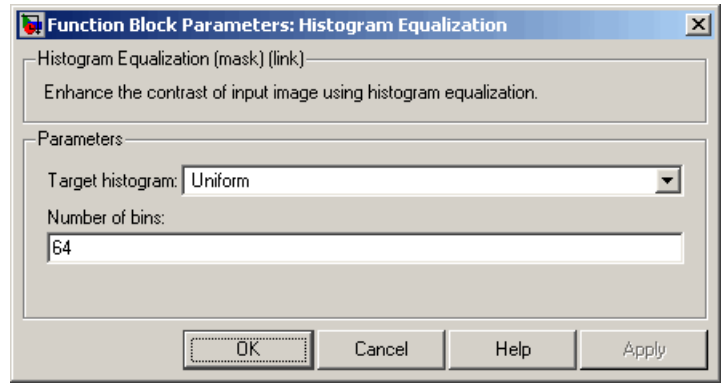
Note The vector input to the Hist port must be normalized such that the sum of the values in all the bins is equal to the number of pixels in the input image. The block does not error if the histogram is not normalized.

Examples

See “Adjusting the Contrast in Intensity Images” on page 7-46 and “Adjusting the Contrast in Color Images” on page 7-52 in the Video and Image Processing Blockset User’s Guide.

Dialog Box

The Histogram Equalization dialog box appears as shown in the following figure.



Target histogram

Designate the histogram you want the output image to have. If you select `Uniform`, the block transforms the input image so that the histogram of the output image is approximately flat. If you select `User-defined`, you can specify the histogram of your output image.

Number of bins

Enter the number of equally spaced bins you want the uniform histogram to have. This parameter is visible if, for the **Target histogram** parameter, you select `Uniform`.

Histogram source

Select how to specify your histogram. Your choices are `Specify via dialog` and `Input port`. This parameter is visible if, for the **Target histogram** parameter, you select `User-defined`.

Histogram

Enter the desired histogram of the output image. This parameter is visible if, for the **Target histogram** parameter, you select `User-defined`. Tunable.

See Also

`imadjust`	Image Processing Toolbox
`histeq`	Image Processing Toolbox

Purpose

Find Cartesian coordinates of lines described by rho and theta pairs

Library

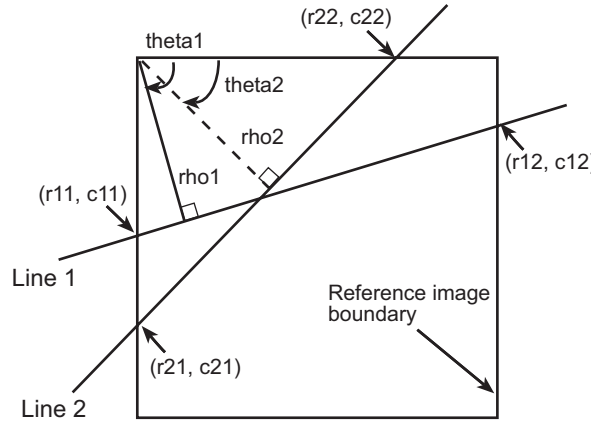
Transforms

Description

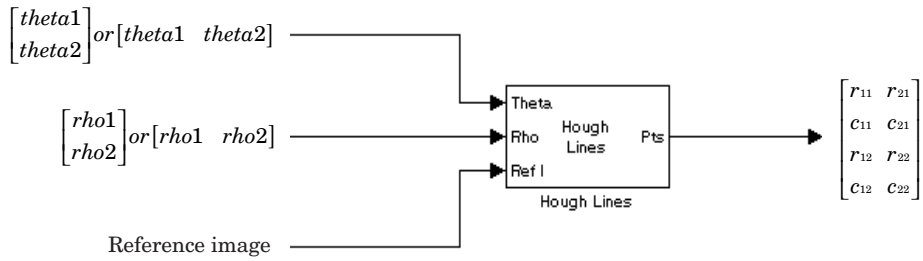


The Hough Lines block inputs are the theta and rho values of lines and a reference image. The block outputs the zero-based row and column positions of the intersections between the lines and two of the reference image boundary lines. The boundary lines are the left and right vertical boundaries and the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

Suppose that Line1 and Line2 intersect the boundaries of the reference image as shown below.



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.



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 integers	No
Rho	Vector of rho values that represent input line(s)	Same as Theta port	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Ref I	Matrix that represents a binary or intensity image or matrix that represents one plane of an RGB image	• Double-precision floating point • Single-precision floating point • Fixed point (signed and unsigned) • Custom data types • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
Pts	4-by-N matrix of intersection values, where N is the number of input lines	• 32-bit signed integers	No

This block supports Simulink virtual buses.

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`.

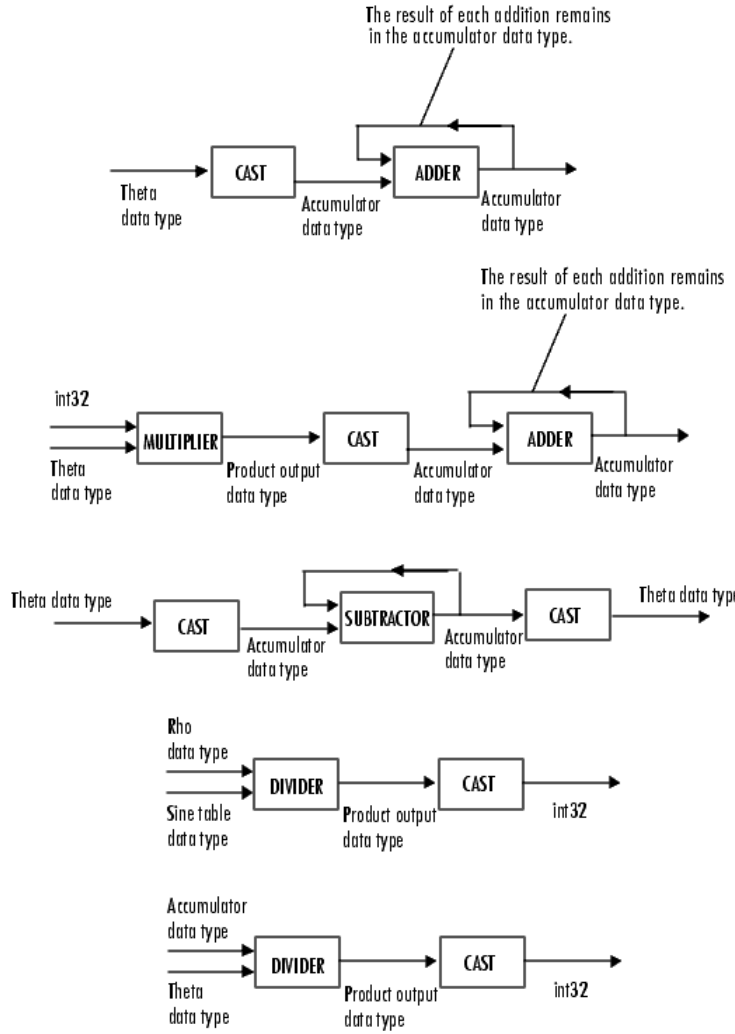
If, for the **Sine value computation method** parameter, you select `Table lookup`, the **Theta resolution (radians)** parameter appears in the dialog box. Use this parameter to specify the spacing of the theta-axis.

Examples

See “Finding Lines in Images” on page 7-9 and “Measuring an Angle Between Lines” on page 7-17 in the Video and Image Processing Blockset User’s Guide.

Fixed-Point Data Types

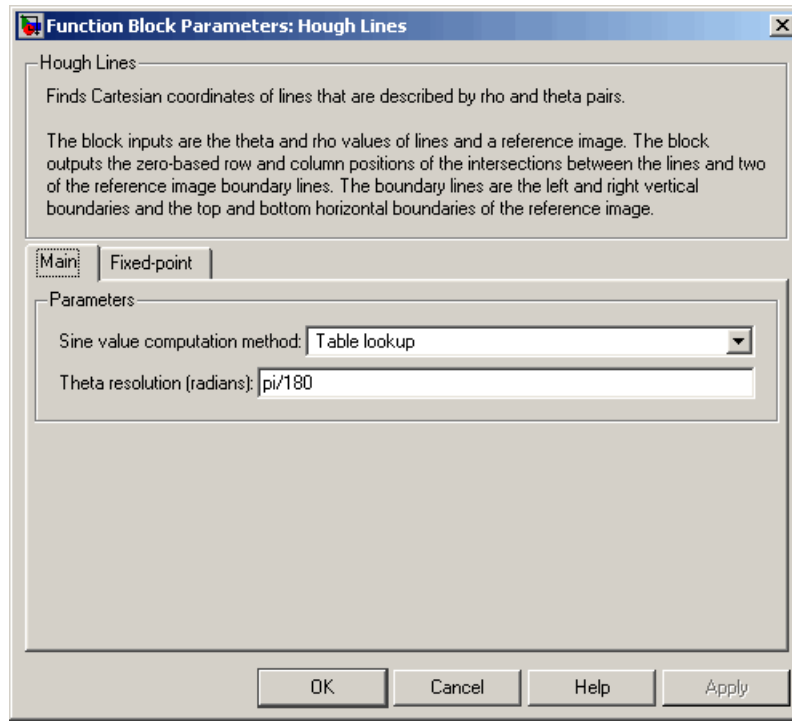
The following diagram shows the data types used in the Hough Lines block for fixed-point signals.



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.



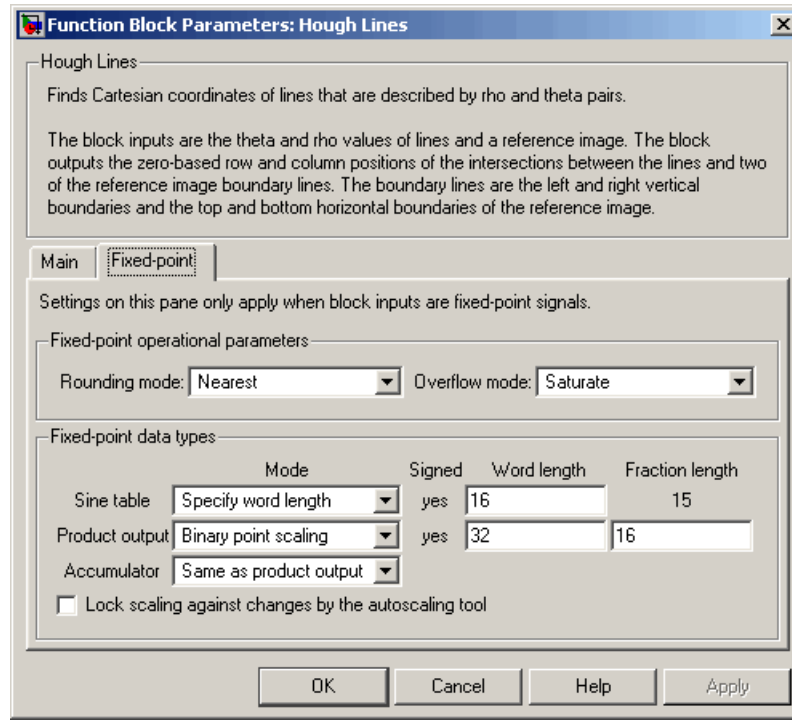
Sine value computation method

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

Theta resolution (radians)

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

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



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

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.

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



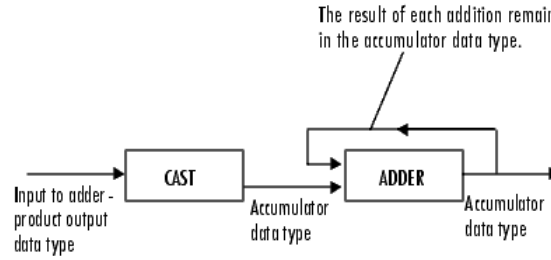
As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths:

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 is 0.

Accumulator



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

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

See Also

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

Find Local Maxima	Video and Image Processing Blockset
Hough Transform	Video and Image Processing Blockset

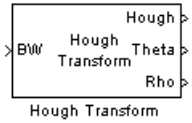
Purpose

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 integers	No
Theta	Vector of theta values	• Double-precision floating point • Single-precision floating point • Fixed point (signed) • 8-, 16-, 32-bit signed integers	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. This block supports Simulink virtual buses.

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.

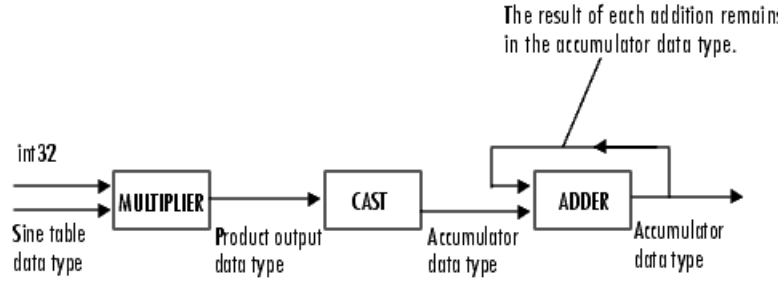
Use the **Output data type** parameter to specify the data type of your output signal.

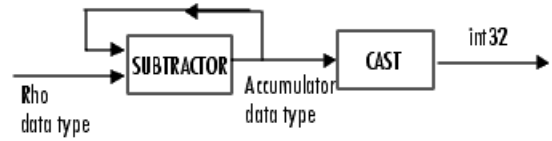
Examples

See “Finding Lines in Images” on page 7-9 and “Measuring an Angle Between Lines” on page 7-17 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.

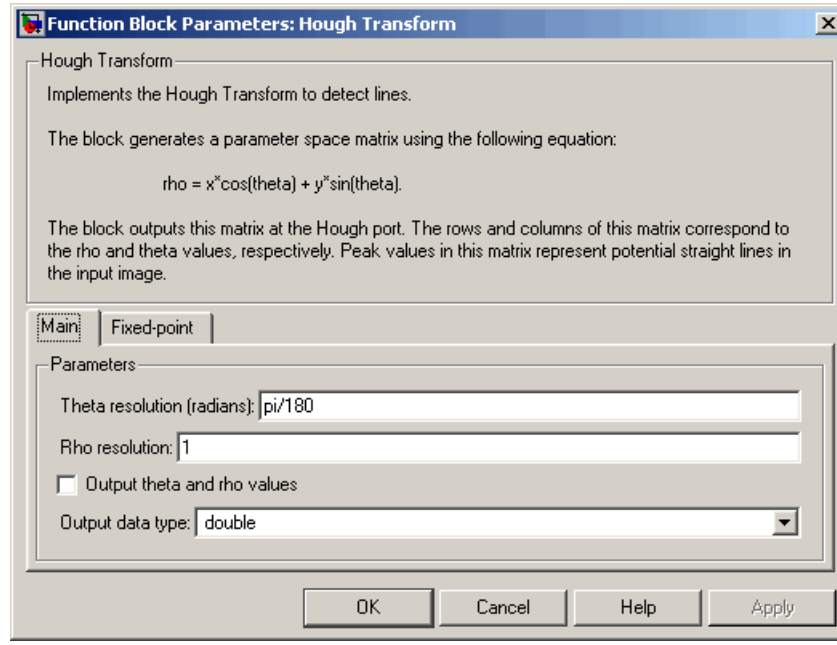




You can set the sine table, rho, product output, accumulator, Hough output, and Theta output data types in the block mask as discussed in the next section.

Dialog Box

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



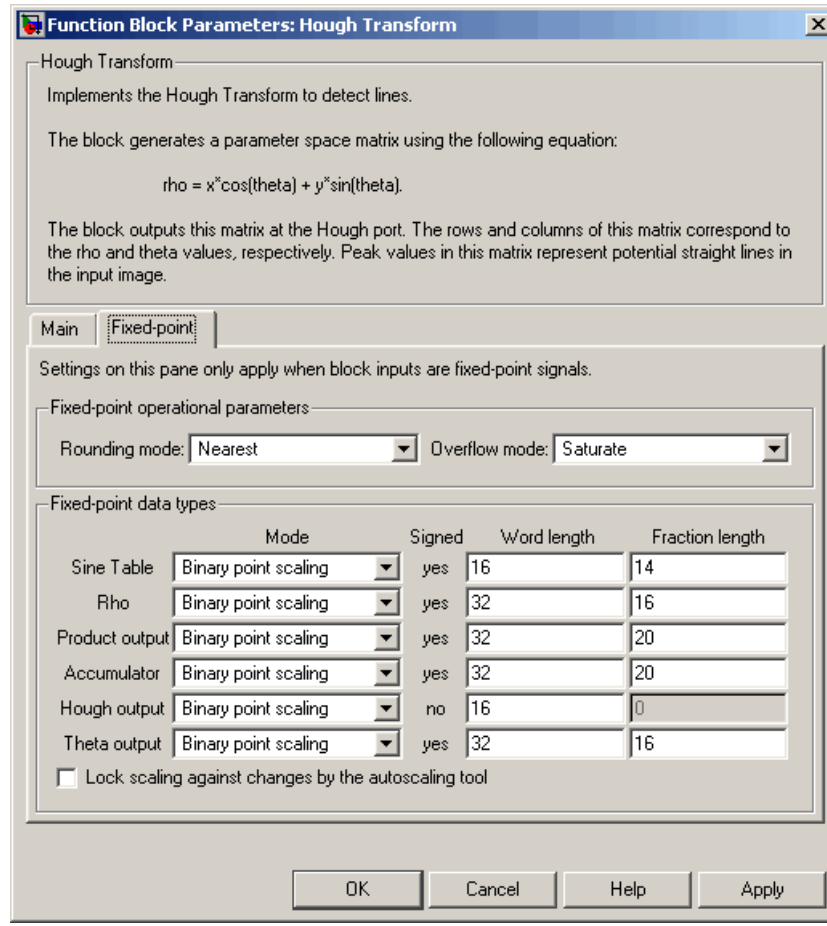
Theta resolution (radians)
Specify the spacing of the Hough transform bins along the *theta*-axis.

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.



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 the Video and Image Processing Blockset is 0.

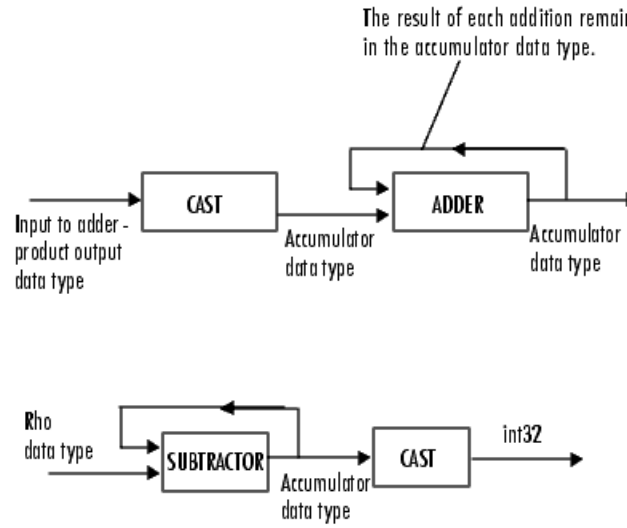
Product output



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

- When you select `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 is 0.

Accumulator



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

- When you select `Same as product output`, these characteristics match those of the product output.

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

Hough output

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

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

Theta output

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

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

See Also

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

Purpose

Compute complement of pixel values in binary, intensity, or RGB images

Library

Conversions

Description



Image Complement

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 $y(i) = 255 - x(i)$.

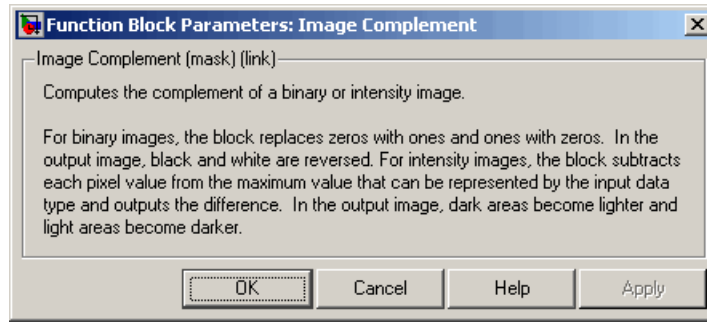
Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the input RGB video stream	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	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. This block supports a signal represented by a Simulink virtual bus.

Image Complement

Dialog Box

The Image Complement dialog box appears as shown in the following figure.



See Also

Autothreshold	Video and Image Processing Blockset
Chroma Resampling	Video and Image Processing Blockset
Color Space Conversion	Video and Image Processing Blockset
`imcomplement`	Image Processing Toolbox

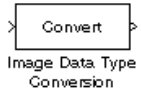
Purpose

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.

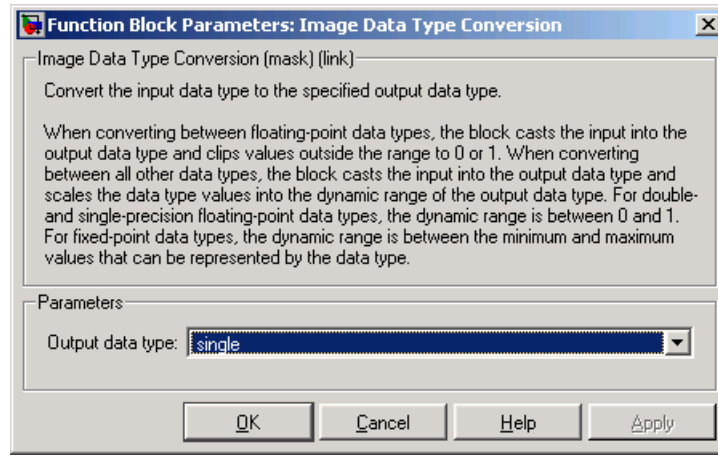
Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the input RGB video stream	• Double-precision floating point • Single-precision floating point • Fixed point (Word length less than or equal to 16) • Boolean • 8-, 16-bit signed integers • 8-, 16-bit unsigned integers	No
Output	Scalar, vector, or matrix with values that are the specified data type	Same as Input port	No

The dimensions, complexity, and frame status of the input and output signals are the same. This block supports a signal represented by a Simulink virtual bus.

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

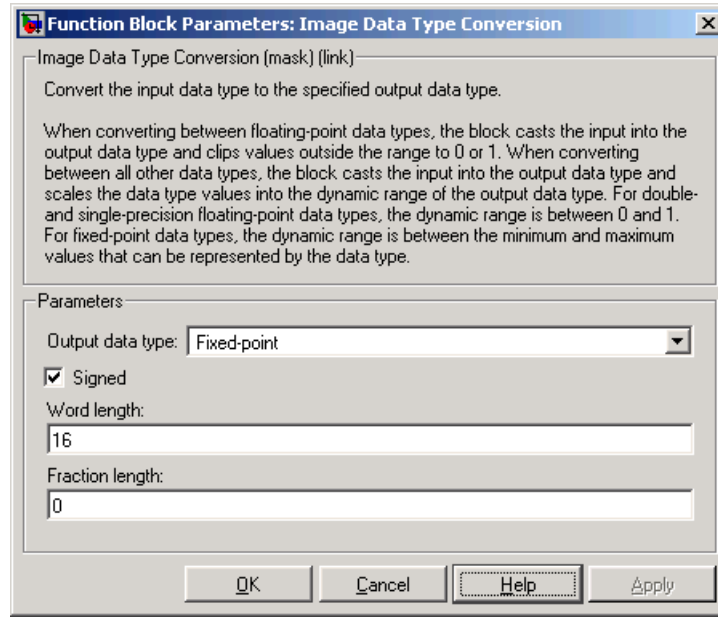
Dialog Box

The Image Data Type Conversion dialog box appears as shown in the following figure.



Output data type

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



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

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 Image Processing Toolbox 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-3 array, the block outputs a color image, where M and N are the number of rows and columns in each color plane.

Port	Output	Supported Data Types	Complex Values Supported
I	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	Yes
R, G, B	Scalar, vector, or matrix that represents one plane of the input RGB video stream. Outputs from the R, G, or B ports have the same dimensions.	Same as I port	Yes

For Video and Image Processing Blockset blocks to display video data properly, double- and single-precision floating-point pixel values must be between 0 and 1. If the input pixel values have a different data type than the one you select using the **Output data type** parameter, the

block scales and/or adds an offset to the pixel values so that they are within the dynamic range of their new data type.

Use the **File name** parameter to specify the name of the graphics file that contains the image to import into Simulink. If the file is not on the MATLAB path, use the **Browse** button to locate the file. This parameter supports URL paths.

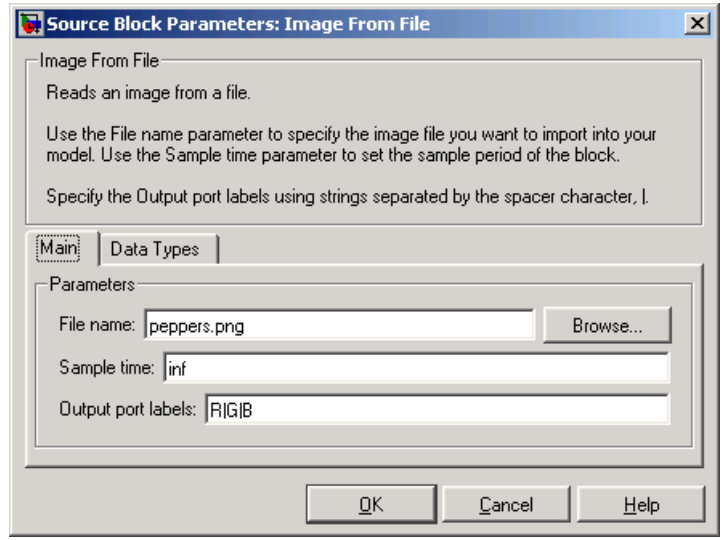
Use the **Sample time** parameter to set the sample period of the output signal.

Use the **Output port labels** parameter to label your output ports. Use the spacer character, |, as the delimiter.

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.

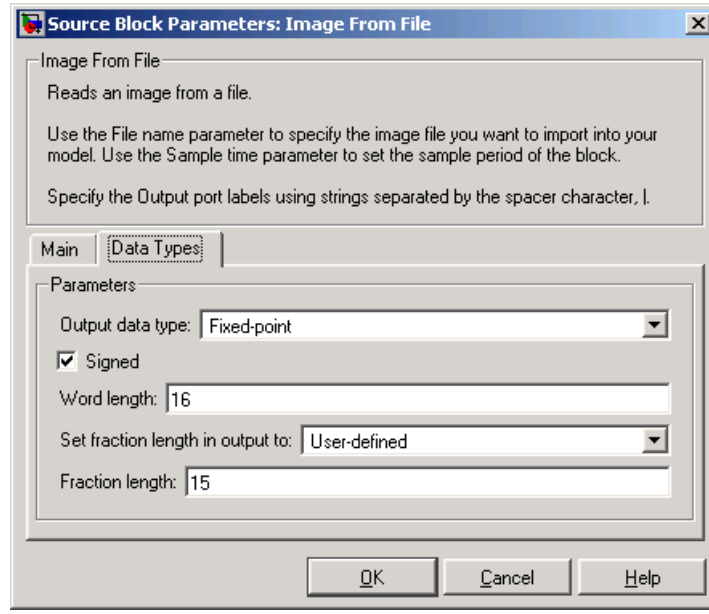


File name
Specify the name of the graphics file that contains the image to import into Simulink.

Sample time
Enter the sample period of the output signal.

Output port labels
Enter the labels for your output ports using the spacer character, |, as the delimiter.

The **Data Types** pane of the Image From File dialog box appears as shown in the following figure.



Output data type
Specify the data type of your output signal.

Signed

Select to output a signed fixed-point signal. Otherwise, the signal will be unsigned. This parameter is only visible if, from the **Output data type** list, you select `Fixed-point`.

Word length

Specify the word length, in bits, of the fixed-point output data type. This parameter is only visible if, from the **Output data type** list, you select `Fixed-point`.

Set fraction length in output to

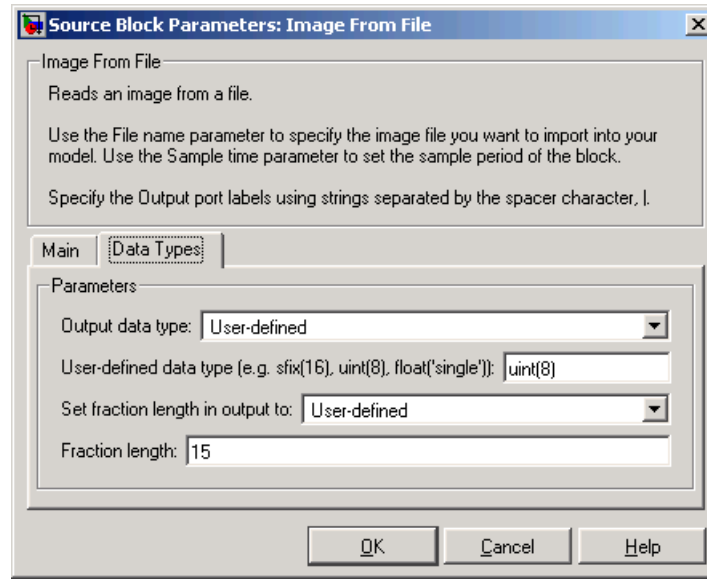
Specify the scaling of the fixed-point output by either of the following two methods:

- Choose `Best precision` to have the output scaling automatically set such that the output signal has the best possible precision.
- Choose `User-defined` to specify the output scaling in the **Fraction length** parameter.

This parameter is only visible if, from the **Output data type** list, you select `Fixed-point` or when you select `User-defined`.

Fraction length

For fixed-point output data types, specify the number of fractional bits, or bits to the right of the binary point. This parameter is only visible when you select `Fixed-point` or `User-defined` for the **Output data type** parameter and `User-defined` for the **Set fraction length in output to** parameter.



User-defined data type

Specify any built-in or fixed-point data type. You can specify fixed-point data types using the `sfix`, `ufix`, `sint`, `uint`, `sfrac`, and `ufrac` functions from Simulink Fixed Point. This parameter is only visible when you select `User-defined` for the **Output data type** parameter.

See Also

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

`im2double`	Image Processing Toolbox
`im2uint8`	Image Processing Toolbox
`imread`	Image Processing Toolbox

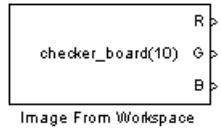
Purpose

Import image from MATLAB workspace

Library

Sources

Description



Use the Image From Workspace block to import an image from the MATLAB workspace. If the image is a M-by-N workspace array, the block outputs a binary or intensity image, where M and N are the number of rows and columns in the image. If the image is a M-by-N-by-3 workspace array, the block outputs a color image, where M and N are the number of rows and columns in each color plane.

Port	Output	Supported Data Types	Complex Values Supported
I	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	No
R, G, B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B ports have the same dimensions.	Same as I port	No

For Video and Image Processing Blockset blocks to display video data properly, double- and single-precision floating-point pixel values must be between 0 and 1. If the input pixel values have a different data type than the one you select using the **Output data type** parameter, the block scales and/or adds an offset to the pixel values so that they are within the dynamic range of their new data type.

Use the **Value** parameter to specify the MATLAB workspace variable that contains the image you want to import into Simulink.

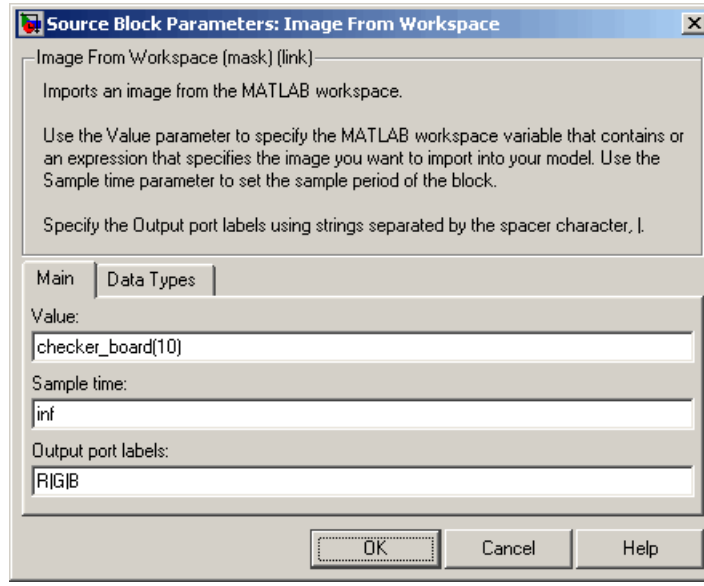
Use the **Sample time** parameter to set the sample period of the output signal.

Use the **Output port labels** parameter to label your output ports. Use the spacer character, |, as the delimiter.

On the **Data Types** pane, use the **Output data type** parameter to specify the data type of your output signal.

Dialog Box

The **Main** pane of the Image From Workspace dialog box appears as shown in the following figure.



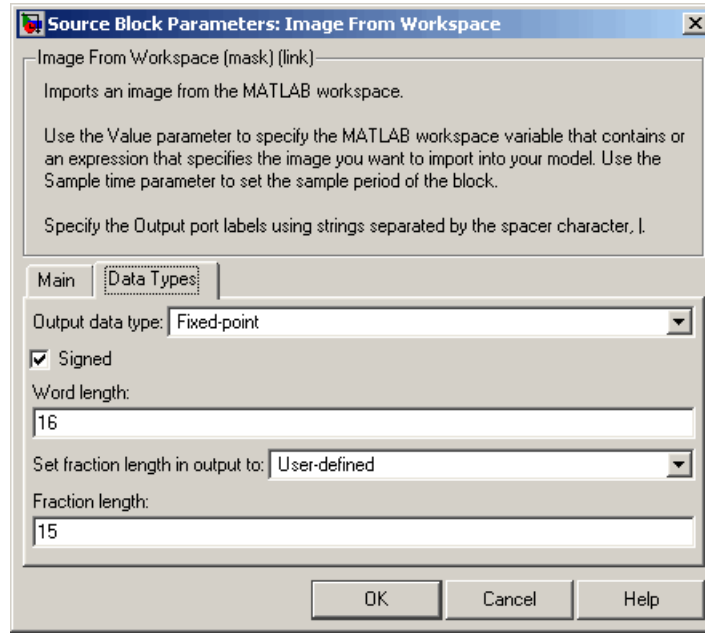
Value

Specify the MATLAB workspace variable that you want to import into Simulink.

Sample time
Enter the sample period of the output signal.

Output port labels
Enter the labels for your output ports using the spacer character, `|`, as the delimiter.

The **Data Types** pane of the Image From Workspace dialog box appears as shown in the following figure.



Output data type
Specify the data type of your output signal.

Signed
Select to output a signed fixed-point signal. Otherwise, the signal will be unsigned. This parameter is only visible if, from the **Output data type** list, you select `Fixed-point`.

Word length

Specify the word length, in bits, of the fixed-point output data type. This parameter is only visible if, from the **Output data type** list, you select `Fixed-point`.

Set fraction length in output to

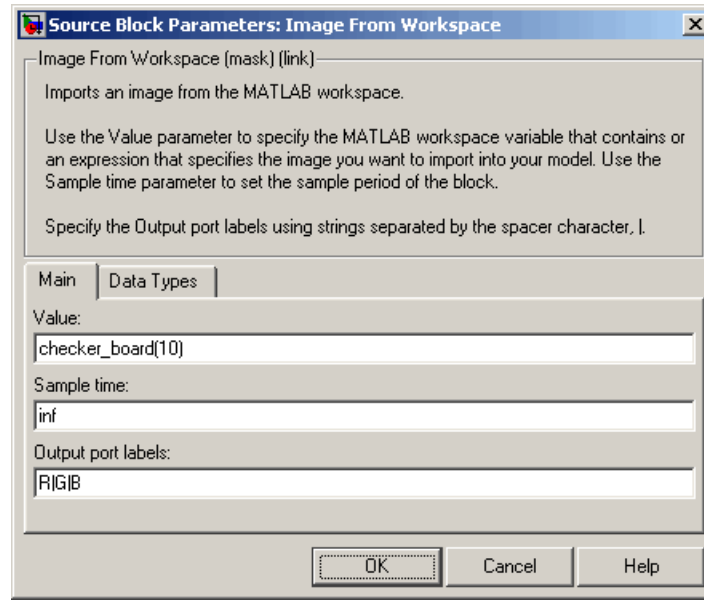
Specify the scaling of the fixed-point output by either of the following two methods:

- Choose `Best precision` to have the output scaling automatically set such that the output signal has the best possible precision.
- Choose `User-defined` to specify the output scaling in the **Fraction length** parameter.

This parameter is only visible if, from the **Output data type** list, you select `Fixed-point` or when you select `User-defined`.

Fraction length

For fixed-point output data types, specify the number of fractional bits, or bits to the right of the binary point. This parameter is only visible when you select `Fixed-point` or `User-defined` for the **Output data type** parameter and `User-defined` for the **Set fraction length in output to** parameter.



User-defined data type

Specify any built-in or fixed-point data type. You can specify fixed-point data types using the `sfix`, `ufix`, `sint`, `uint`, `sfrac`, and `ufrac` functions from Simulink Fixed Point. This parameter is only visible when you select `User-defined` for the **Output data type** parameter.

See Also

From Multimedia File	Video and Image Processing Blockset
To Video Display	Video and Image Processing Blockset
Video From Workspace	Video and Image Processing Blockset
Video Viewer	Video and Image Processing Blockset
`im2double`	Image Processing Toolbox
`im2uint8`	Image Processing Toolbox

Insert Text

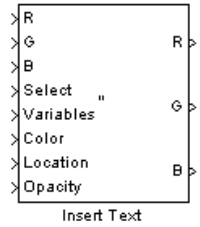
Purpose

Draw text on image or video stream

Library

Text & Graphics

Description



The Insert Text block draws formatted text or numbers on an image or video stream. The block uses the FreeType library, an open-source font engine, to produce stylized text bitmaps. To learn more about the FreeType Project, visit `http://www.freetype.org/`.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point. (Word length must be less than or equal to 32.) • Boolean • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	No
R, G, B	Matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B ports have the same dimensions and data type.	Same as I port	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Select	Scalar value that indicates which text string to display	• 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 integers • 8-, 16-, 32-bit unsigned integers	No
Variables	Vector whose values are used to replace ANSI C printf-style format specifications	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Location	Two-element vector of the form [row column], which represents the top-left corner of the text bounding box	• 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 integers • 8-, 16-, 32-bit unsigned integers	No
Intensity	Scalar value that represents the intensity value of the text. Input must be the same data type as port I.	Same as I port	No

Port	Input/Output	Supported Data Types	Complex Values Supported
Color	Three-element vector that specifies the RGB triplet value for the text. Input must be the same data type as ports R, G, and B.	Same as ports R, G, and B. (The input to this port must be the same data type as the input to the R, G, and B ports.)	No
Opacity	Scalar value that indicates the text's opaqueness	• 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.) • `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.)	No

Use the **Input type** parameter to specify the input to the block. If the block input is an RGB image or video stream, select `RGB`. If the block input is an intensity image or video stream, select `Intensity`.

Use the **Text** parameter to specify the text string to be drawn on the image or video frame. This parameter can be a single text string, such as `'Figure1'`, or a cell array of strings, such as `{'Figure1','Figure2'}`.

If, for the **Text** parameter, you enter a cell array of strings, the Select port appears on the block. The input to this port must be a scalar value that indicates which text string to display, where 1 indicates the first

string. If the input is less than 1 or greater 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. If you use the %s format specification, the input to the Variables port must be a vector of 8-bit unsigned integers. The block treats this input as one null-terminated C-style character string. You cannot enter more than one type of ANSI C printf-style format specification in the **Text** parameter.

Use the **Location source** parameter to indicate where to specify the text location. If you select `Specify via dialog`, the **Location [row column]** parameter appears on the dialog. Enter a two-element vector of the form [row column], which indicates the top-left corner of the text bounding box. 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. If, for the **Location source** parameter, you select `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 **Input type** parameter, you select `RGB`, the **Color source** parameter appears on the dialog. Use this parameter to indicate where to specify the text color. If you select `Specify via dialog`, the **Color [R G B]** parameter appears on the dialog. Enter an RGB triplet that specifies the color of the text. If the input RGB image is floating point, the values must be between 0 and 1. If the input RGB image is composed of 8-bit unsigned integers, the values must range between 0 and 255. If, for the **Color source** parameter, you select `Input port`, the Color port appears on the block. The input to this port must be a three-element vector that specifies the RGB triplet value for the text.

If, for the **Input type** parameter, you select `Intensity`, the **Intensity source** parameter appears on the dialog. Use this parameter to indicate where to specify the text intensity. If you select `Specify via dialog`, the **Intensity** parameter appears on the dialog. Enter a

scalar value that represents the intensity value of the text. If the input intensity image is floating point, the value must be between 0 and 1. If the input intensity image is composed of 8-bit unsigned integers, the value must range between 0 and 255. If, for the **Intensity source** parameter, you select `Input port`, the Intensity port appears on the block. The input to this port must be a scalar value as described for the **Intensity** parameter.

Use the **Opacity source** parameter to indicate where to specify the text's opaqueness. If you select `Specify via dialog`, the **Opacity** parameter appears on the dialog. Enter a scalar value between 0 and 1, where 0 is translucent and 1 is opaque. If, for the **Opacity source** parameter, you select `Input port`, the Opacity port appears on the block. The input to this port must be a scalar value as described for the **Opacity** parameter.

On the **Font** pane, use the **Font face** parameter to specify the font of your text. The block populates this list with the TrueType fonts installed on your system. On Windows, the block searches the system registry for font files. On UNIX, the block searches the X Server's font path for font files.

Use the **Font size (points)** parameter to specify the font size.

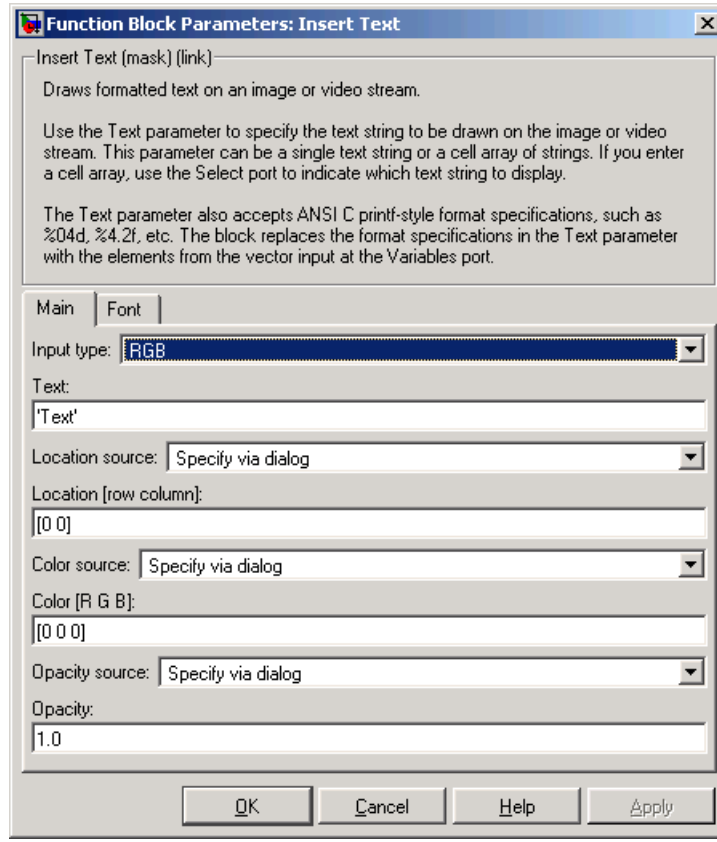
If you select the **Anti-aliased** check box, the block smoothes the edges of the text, which can be computationally expensive. If you want your model to run faster, clear this check box.

Examples

See "Annotating AVI Files" on page 2-9 in the Video and Image Processing Blockset User's Guide.

Dialog Box

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



Input type

Specify the input to the block. If the block input is an RGB image or video stream, select `RGB`. If the block input is an intensity image or video stream, select `Intensity`.

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.

Location source

Indicate where you want to specify the text location. Your choices are `Specify via dialog` or `Input port`.

Location [row column]

Enter a two-element vector of the form [row column], which indicates the top-left corner of the text bounding box. This parameter is visible if, for the **Location source** parameter, you select `Specify via dialog`. Tunable.

Color source

Indicate where you want to specify the text color. Your choices are `Specify via dialog` or `Input port`. This parameter is visible if, for the **Input type** parameter, you select `RGB`.

Color [R G B]

Enter an RGB triplet that specifies the color of the text. If the input RGB image is floating point, the values must be between 0 and 1. If the input RGB image is composed of 8-bit unsigned integers, the values must range between 0 and 255. This parameter is visible if, for the **Color source** parameter, you select `Specify via dialog`. Tunable.

Intensity source

Indicate where you want to specify the text intensity. Your choices are `Specify via dialog` or `Input port`. This parameter is visible if, for the **Input type** parameter, you select `Intensity`.

Intensity

Enter a scalar value that represents the intensity value of the text. If the input intensity image is floating point, the value must be between 0 and 1. If the input intensity image is composed of 8-bit unsigned integers, the value must range between 0 and 255. This parameter is visible if, for the **Intensity source** parameter, you select `Specify via dialog`. Tunable.

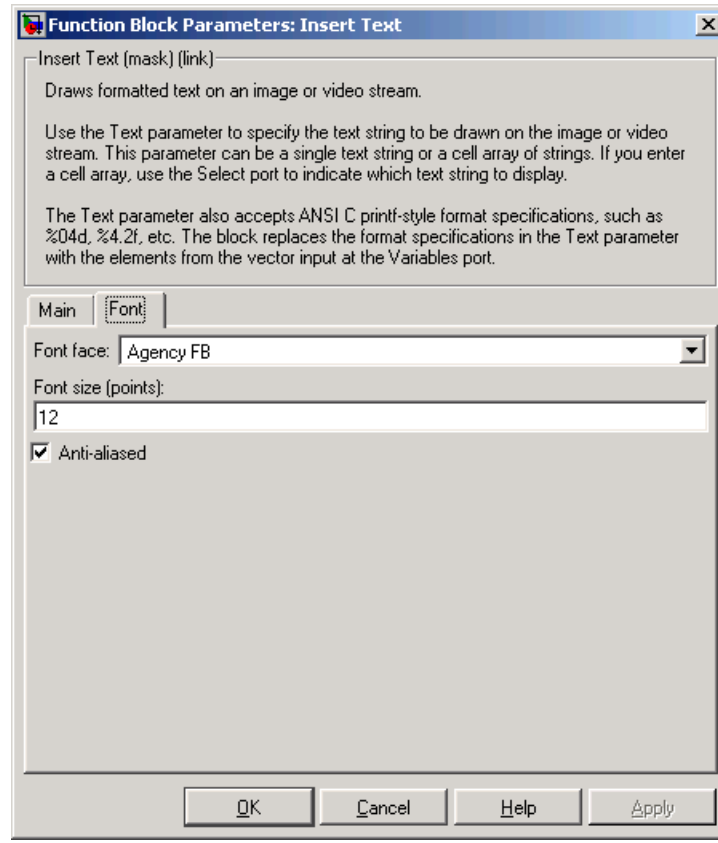
Opacity source

Indicate where you want to specify the text's opaqueness. Your choices are `Specify via dialog` or `Input port`.

Opacity

Enter a scalar value between 0 and 1, where 0 is translucent and 1 is opaque. This parameter is visible if, for the **Opacity source** parameter, you select `Specify via dialog`. Tunable.

The **Font** pane of the Insert Text dialog box appears as shown in the following figure.



Font face

Specify the font of your text. The block populates this list with the fonts installed on your system.

Font size (points)

Specify the font size.

Anti-aliased

Select this check box if you want the block to smooth the edges of the text.

See Also

Draw Shapes

Video and Image Processing Blockset

Purpose

Label connected components in binary images

Library

Morphological Operations

Description

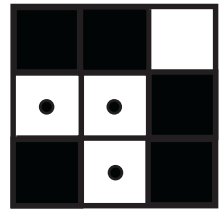


The Label block labels the objects in a binary image, BW, where the background is represented by pixels equal to 0 (black) and objects are represented by pixels equal to 1 (white). At the Label port, the block outputs a label matrix that is the same size as the input matrix. In the label matrix, pixels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

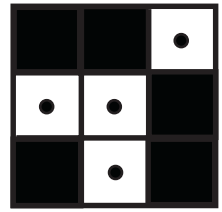
Port	Input/Output	Supported Data Types	Complex Values Supported
BW	Scalar, vector, or matrix that represents a binary image	Boolean	No
Label	Label matrix	• 8-, 16-, and 32-bit unsigned integers	No
Count	Scalar that represents the number of labeled objects	Same as Label port	No

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 3-by-3 image shown below. 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.



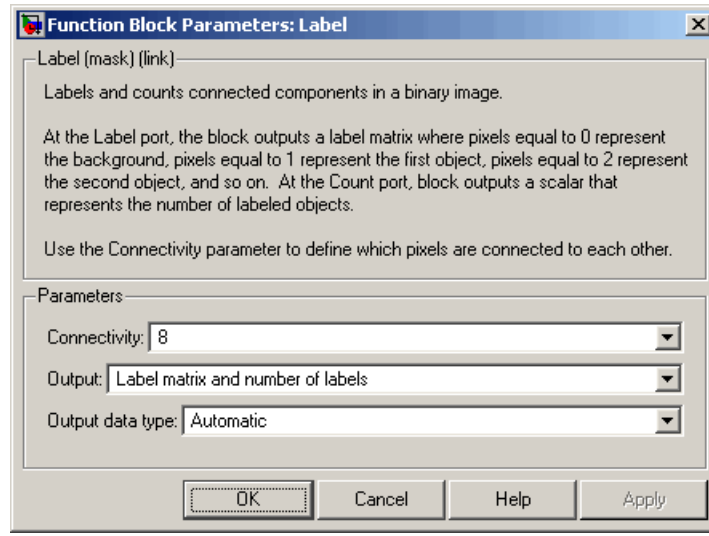
Use the **Output** parameter to determine the block's output. If you select `Label matrix and number of labels`, ports Label and Count appear on the block. The block outputs the label matrix at the Label port and the number of labeled objects at the Count port. If you select `Label matrix`, the Label 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.



Connectivity

Specify which pixels are connected to each other. If you want a pixel to be connected to the pixels on the top, bottom, left, and right, select `4`. If you want a pixel to be connected to the pixels on the top, bottom, left, right, and diagonally, select `8`.

Output

Determine the block's output. If you select `Label matrix and number of labels`, the Label and Count ports appear on the block. The block outputs the label matrix at the Label port and the number of labeled objects at the Count port. If you select

`Label matrix`, the Label port appears on the block. If you select `Number of labels`, the Count port appears on the block.

Output data type
Set the data type of the outputs at the Label and Count ports. If you select `Automatic`, the block determines the appropriate data type for the output. If you select `uint32`, `uint16`, or `uint8`, the data type of the output is 32-, 16-, or 8-bit unsigned integers, respectively.

If label exceeds data type size, mark remaining regions using
Use this parameter to specify the block's behavior if the number of found objects exceeds the maximum number that can be represented by the output data type. If you select `Maximum value of the output data type`, the remaining regions are labeled with the maximum value of the output data type. If you select `Zero`, the remaining regions are labeled with zeroes. This parameter is visible if, for the **Output data type** parameter, you choose `uint16` or `uint8`.

See Also

Bottom-hat	Video and Image Processing Blockset
Closing	Video and Image Processing Blockset
Dilation	Video and Image Processing Blockset
Erosion	Video and Image Processing Blockset
Opening	Video and Image Processing Blockset
Top-hat	Video and Image Processing Blockset
`bwlabel`	Image Processing Toolbox
`bwlabeln`	Image Processing Toolbox

Purpose

Find maximum values in input or sequence of inputs

Library

Statistics

Description

The Maximum block is a Signal Processing Blockset block. For more information, see the Maximum block reference page in the Signal Processing Blockset documentation.

Median Filter

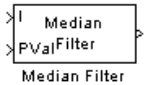
Purpose

Perform 2-D median filtering

Library

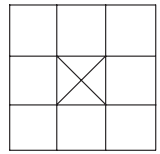
Filtering / Analysis & Enhancement

Description

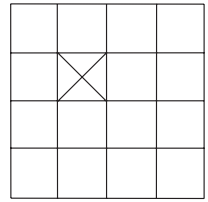


Median Filter

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 below.



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 below.



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 integers • 8-, 16-, 32-bit unsigned integers	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. This block supports a signal represented by a Simulink virtual bus.

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 2-D 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 10-406. 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

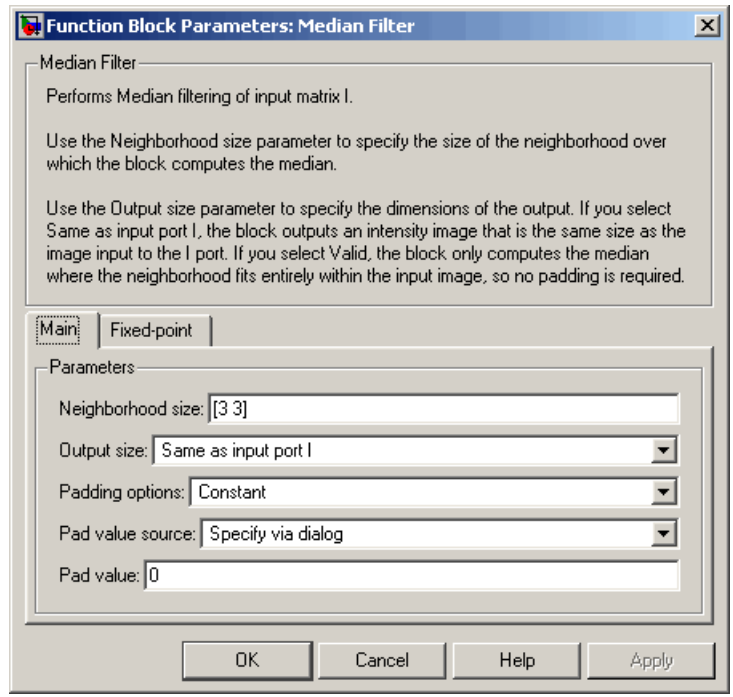
The accumulator and output data types and scalings are used for fixed-point signals when `M` is even. The result of the sum performed while calculating the average of the two central rows of the input matrix is stored in the accumulator data type and scaling. The total result of the average is then put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



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`.

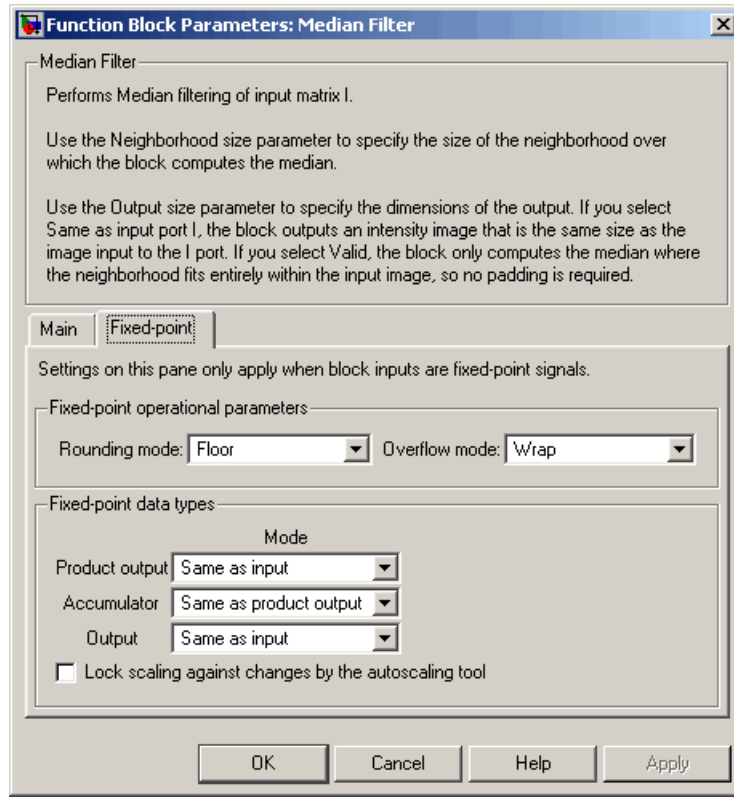
Pad value source

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 are only visible when the dimensions of the neighborhood are even.



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Note The product output, accumulator, and output parameters listed below are only used in certain cases. Refer to “Fixed-Point Data Types” on page 10-404 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

References

Gonzales, Rafael C. and Richard E. Woods. *Digital Image Processing. 2nd ed*. Englewood Cliffs, NJ: Prentice-Hall, 2002.

See Also

2-D Convolution	Video and Image Processing Blockset
2-D FIR Filter	Video and Image Processing Blockset
`medfilt2`	Image Processing Toolbox

Purpose

Find minimum values in input or sequence of inputs

Library

Statistics

Description

The Minimum block is a Signal Processing Blockset block. For more information, see the Minimum block reference page in the Signal Processing Blockset documentation.

Opening

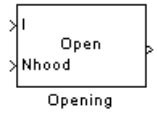
Purpose

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	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	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. This block supports a signal represented by a Simulink virtual bus.

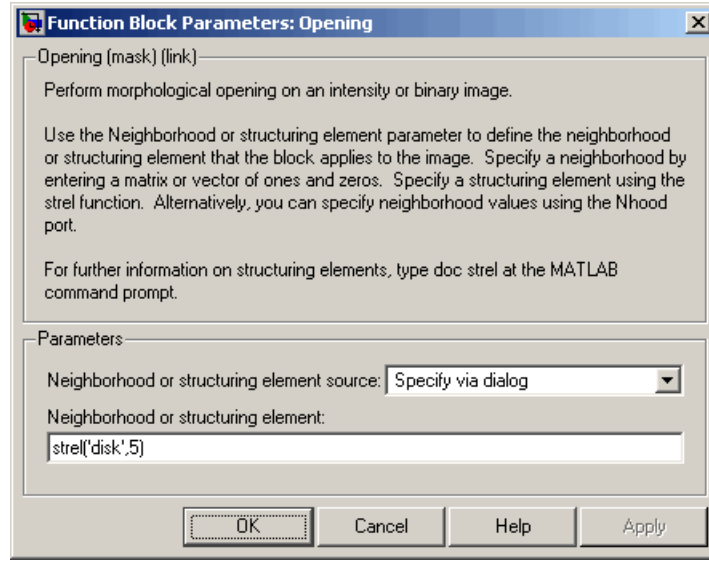
Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values.

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

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

Dialog Box

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



Neighborhood or structuring element source

Specify how to enter your neighborhood or structuring element values. Select `Specify via dialog` to enter the values in the

dialog box. Select `Input port` to use the Nhood port to specify the neighborhood values. You can only specify a structuring element using the dialog box.

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

References

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

See Also

Bottom-hat	Video and Image Processing Blockset
Closing	Video and Image Processing Blockset
Dilation	Video and Image Processing Blockset
Erosion	Video and Image Processing Blockset
Label	Video and Image Processing Blockset
Top-hat	Video and Image Processing Blockset
`imopen`	Image Processing Toolbox
`strel`	Image Processing Toolbox

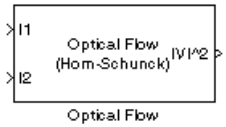
Purpose

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	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, I_x, I_y and I_t are the spatiotemporal image brightness derivatives, u is the horizontal optical flow, and v is the vertical optical flow. Because this equation is underconstrained, there are several methods to solve for u and v. Two such methods, used by the Optical Flow block, are described next.

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, $\left[u_{x,y}^k \quad v_{x,y}^k \right]$ is the velocity estimate for the pixel at (x,y), and $\left[\bar{u}_{x,y}^{-k} \quad \bar{v}_{x,y}^{-k} \right]$ is the neighborhood average of $\left[u_{x,y}^k \quad v_{x,y}^k \right]$. For $k=0$, the initial velocity is 0.

If, for the **Method** parameter, you select `Horn-Schunck`, the block solves for u and v as follows:

1 Compute I_x and I_y using the Sobel convolution kernel,

$[-1 \quad -2 \quad -1;\ 0 \quad 0 \quad 0;\ 1 \quad 2 \quad 1]$, and its transposed form for each pixel in the first image.

2. Compute I_t between images 1 and 2 using the $[-1 \quad 1]$ kernel.
3. Assume the previous velocity to be 0, and compute the average velocity for each pixel using $[0 \quad 1 \quad 0; \quad 1 \quad 0 \quad 1; \quad 0 \quad 1 \quad 0]$ 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 real part of each value is the horizontal velocity component and the imaginary part of each value is the vertical velocity component.

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 divided 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, for the **Method** parameter, you select `Lucas-Kanade`, the block solves for u and v as follows:

1. Compute I_x and I_y using the kernel $\frac{1}{12}[-1 \quad 8 \quad 0 \quad -8 \quad 1]$ and its transposed form.

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 $\frac{1}{16}[1 \quad 4 \quad 6 \quad 4 \quad 1]$.

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 $\lambda_i = \frac{a+c}{2} \pm \frac{\sqrt{4b^2 + (a-c)^2}}{2}; i = 1,2$
- Once the block finds the eigenvalues, it compares them to the threshold, τ, that corresponds to the value you enter for the **Noise reduction threshold** 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.

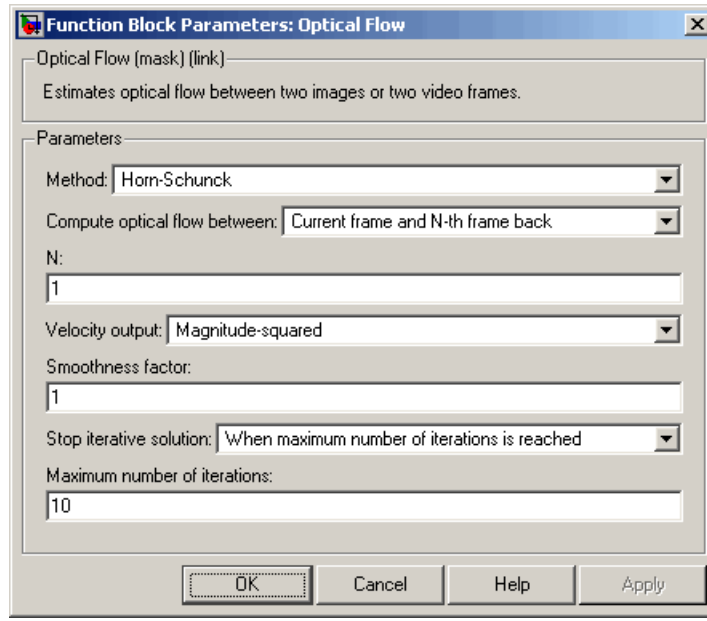
The **Compute optical flow between**, **N**, and **Velocity output** parameters are described in "Horn-Schunck Method" on page 10-416.

Use the **Noise reduction threshold** 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.

Dialog Box

The Optical Flow dialog box appears as shown in the following figure.



Method

Select the method the block uses to calculate the optical flow. Your choices are `Horn-Schunck` or `Lucas-Kanade`.

Compute optical flow between

Select `Two images` to compute the optical flow between two images. Select `Current frame and N-th frame back` to compute the optical flow between two video frames that are N frames apart.

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 **Compute optical flow between** parameter, you select `Current frame and N-th frame back`.

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 $u + jv$.

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 is only visible if, for the **Method** parameter, you select `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 is only visible if, for the **Method** parameter, you select `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 is only visible if, for the **Method** parameter, you select `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 is only visible if, for the **Method** parameter, you select `Horn-Schunck`.

Noise reduction threshold

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 is only visible if, for the **Method** parameter, you select `Lucas-Kanade`.

References

Barron, J.L., D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt. *Performance of optical flow techniques*. CVPR, 1992.

See Also

Block Matching	Video and Image Processing Blockset
Gaussian Pyramid	Video and Image Processing Blockset

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
I (input and output)	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	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 integers • 8-, 16-, 32-bit unsigned integers 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 **Input image type** parameter to specify whether the block input is an intensity or RGB image.

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

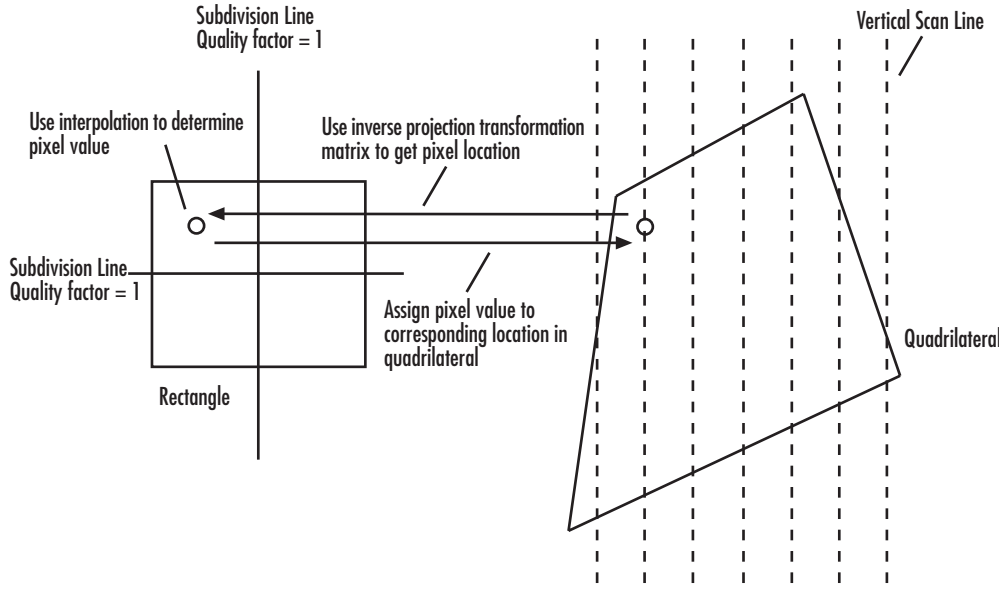
Rectangle to Quadrilateral Mode

Use the **Inverse mapping method** parameter to specify the algorithm the block uses to implement the projective transformation. If you choose

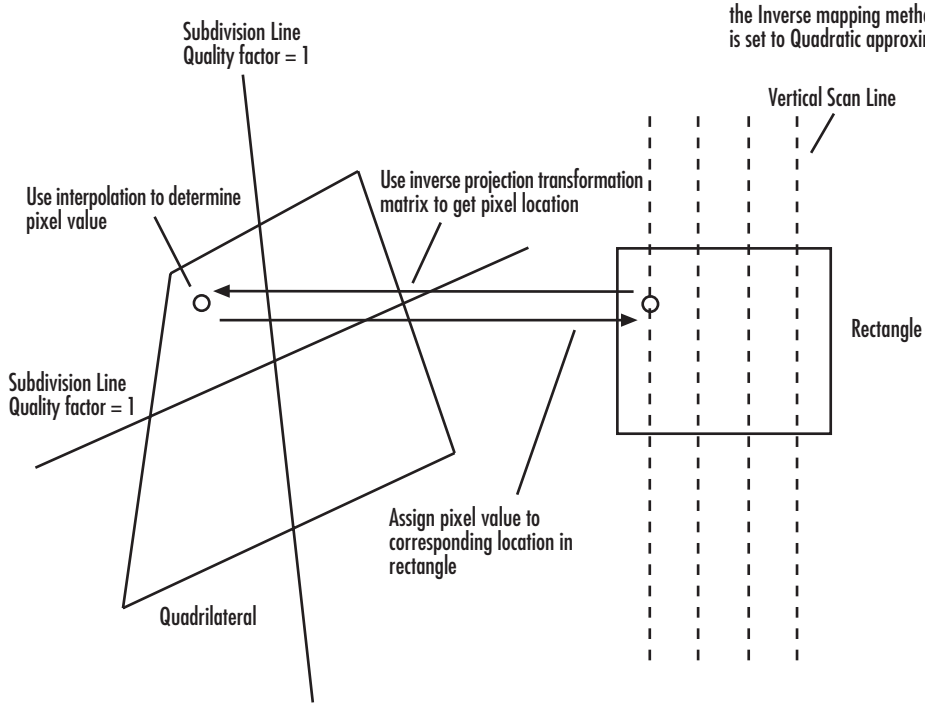
`Exact solution`, the block divides the output shape using vertical scan lines. For each pixel location on a scan line, it uses an inverse projection transformation matrix to find the corresponding pixel location in the input image. When this pixel location is not located directly on a pixel in the input 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.



*The subdivision lines are only used when the Inverse mapping method parameter is set to Quadratic approximation.



Use the **Quality factor (number of subdivisions)** parameter to specify the number of pairs of horizontal and vertical lines (subdivision lines) the block uses to subdivide the output shape. Enter a scalar integer value that is greater than or equal to 0 and less than or equal to the height or width of the input image, whichever is smaller. The larger the quality factor, the closer the approximate solution is to the exact solution. Experiment with this parameter to find the value that best suits your application.

Use the **Background fill value(s)** parameter to specify the background of the output image. If the block outputs an intensity image, enter a scalar value. If the block outputs an RGB image, enter a scalar value that the block uses as each of the R, G, and B values or a three-element vector that specifies an RGB triplet.

Use the **Interpolation method** parameter to specify which 2-D interpolation method the block uses to calculate the pixel values in the output image. If you select `Nearest neighbor`, the block uses the value of the nearest pixel for the new pixel intensity. If you select `Bilinear`, the new pixel value is the weighted average of the four nearest pixel intensities. If you select `Bicubic`, the new pixel value is the weighted average of the 16 nearest pixel intensities.

Input image parameters — Use the **Rectangular ROI** parameter to define the portion of the input image that the block transforms into a quadrilateral. Your choices are `Full image` or `User-defined`. If you select `User-defined`, the **Rectangular ROI source** parameter appears in the dialog box. Use this parameter to specify whether you want to define the ROI using the block dialog box or an input port. If you select `Specify via dialog`, use the **ROI [r,c,height,width]** parameter to enter the row and column coordinates of the upper-left corner of the ROI as well as its height and width. If you select `Input port`, the **If ROI is invalid** parameter appears in the dialog box. Use it to specify the block's behavior if the four-element vector input to the InROI port contains values that are outside the input image. Your choices are `Clip`, `Clip and warn`, or `Error`. If you select `Clip`, the block changes the row, column, height, or width values so the ROI fits entirely within the input image.

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 10-425.

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 10-425.

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

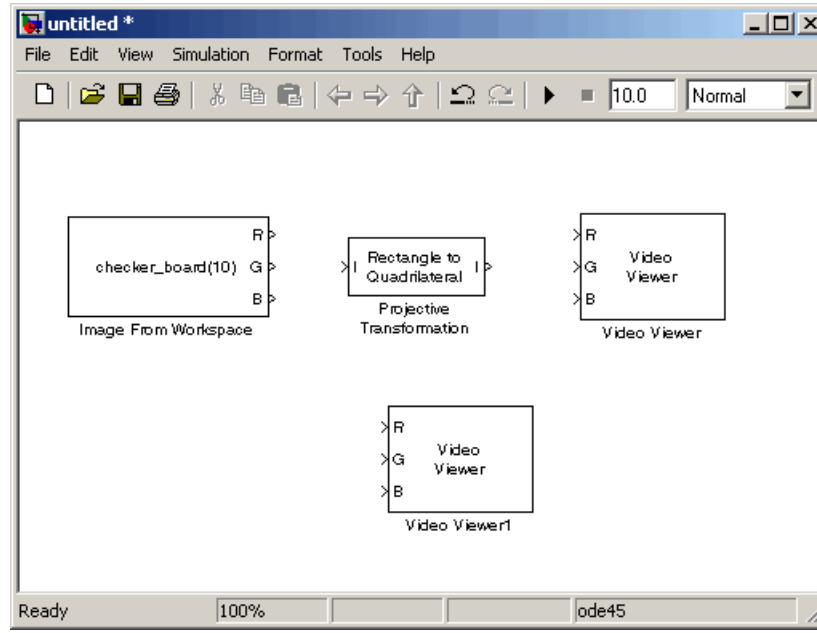
The following example shows you how to convert a rectangular image into a quadrilateral. It also shows you how to change the sizes of the input and output images.

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 / Sources	1
Projective Transformation	Video and Image Processing Blockset / Geometric Transformations	1
Video Viewer	Video and Image Processing Blockset / Sinks	2

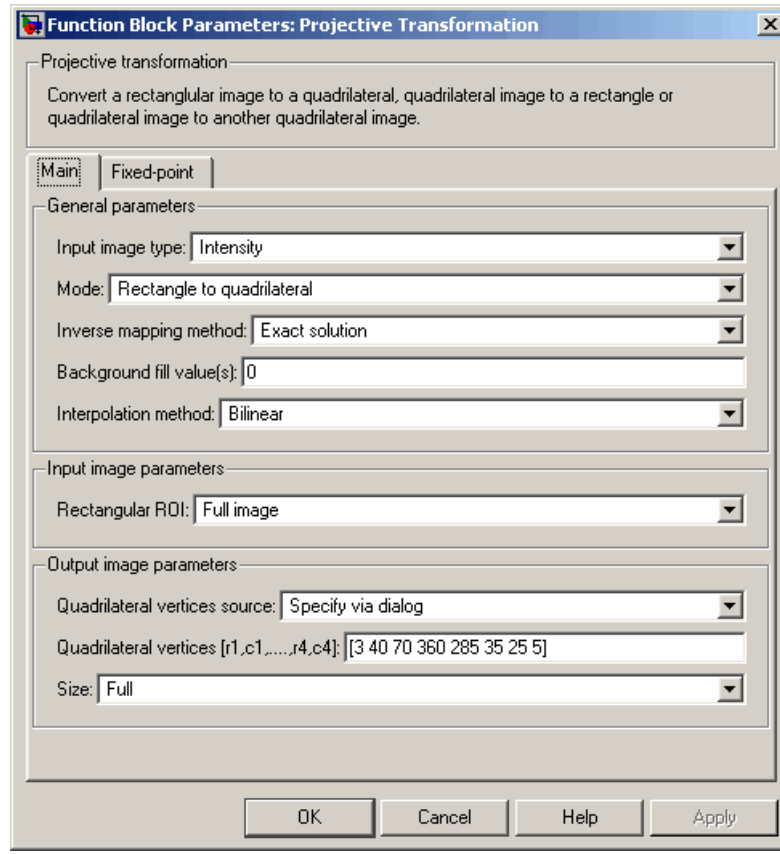
3. Place the blocks so your model looks similar to the figure below.



4 Use the Image From Workspace block to import an image into your model. Set the block parameters as follows:

- **Value** = `imread('cameraman.tif')`
- **Output port labels** = `I`

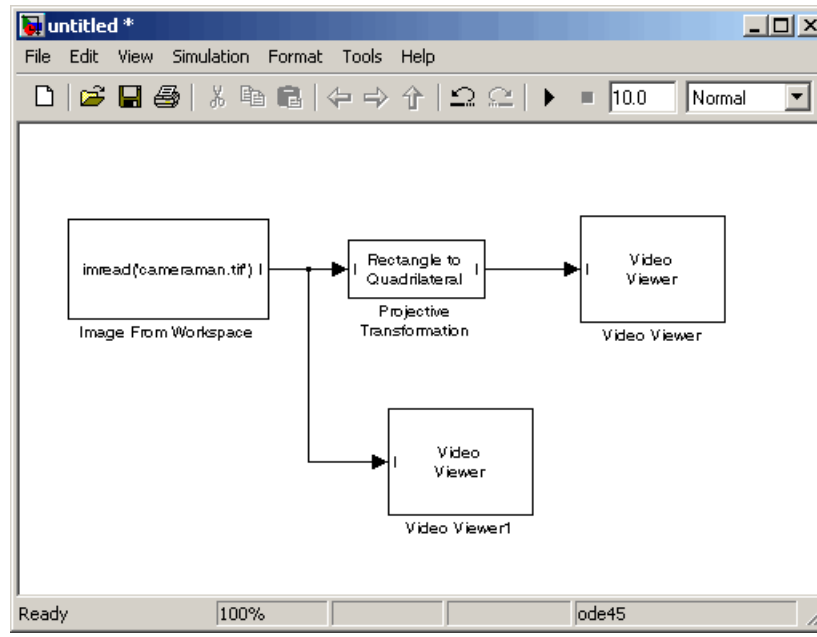
5 Use the Projective Transformation block to transform your rectangular image into a quadrilateral. Set the **Quadrilateral vertices [r1,c1,...,r4,c4]** parameter to `[3 40 70 360 285 35 25 5]`.



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 Viewer1 and Video Viewer blocks to view the rectangular and quadrilateral images, respectively. Set the **Input image type** parameters to `Intensity`.

7 Connect the blocks so that your model resembles the following figure.

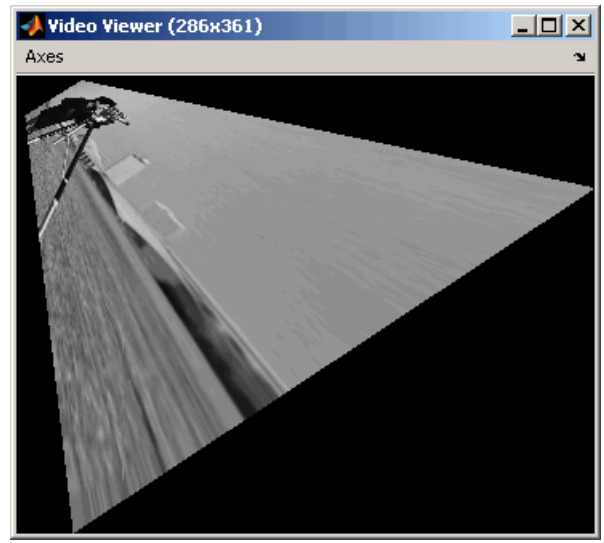


8 Run the model.

The original rectangular image appears in the Video Viewer1 window.



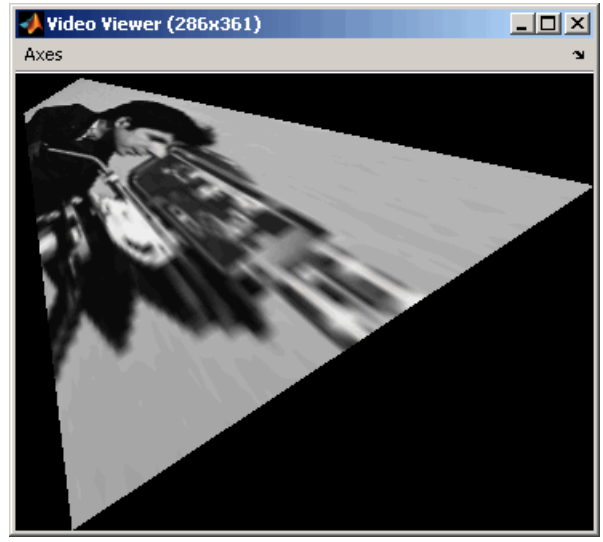
The quadrilateral image appears in the Video Viewer window.



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]`

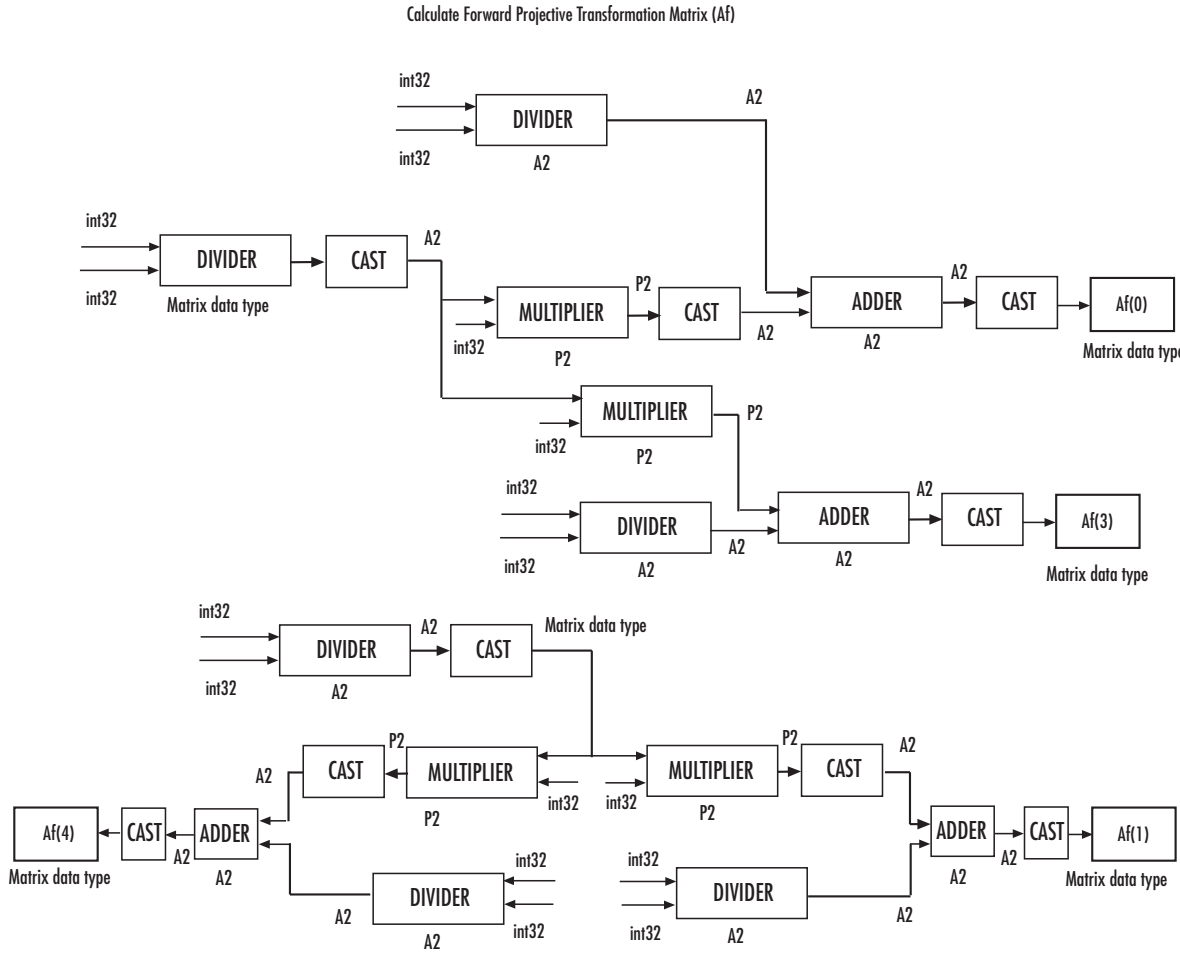
The **Location and size [r,c,height,width]** parameter defines the row and column coordinates as well as the height and width of the output image.

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.



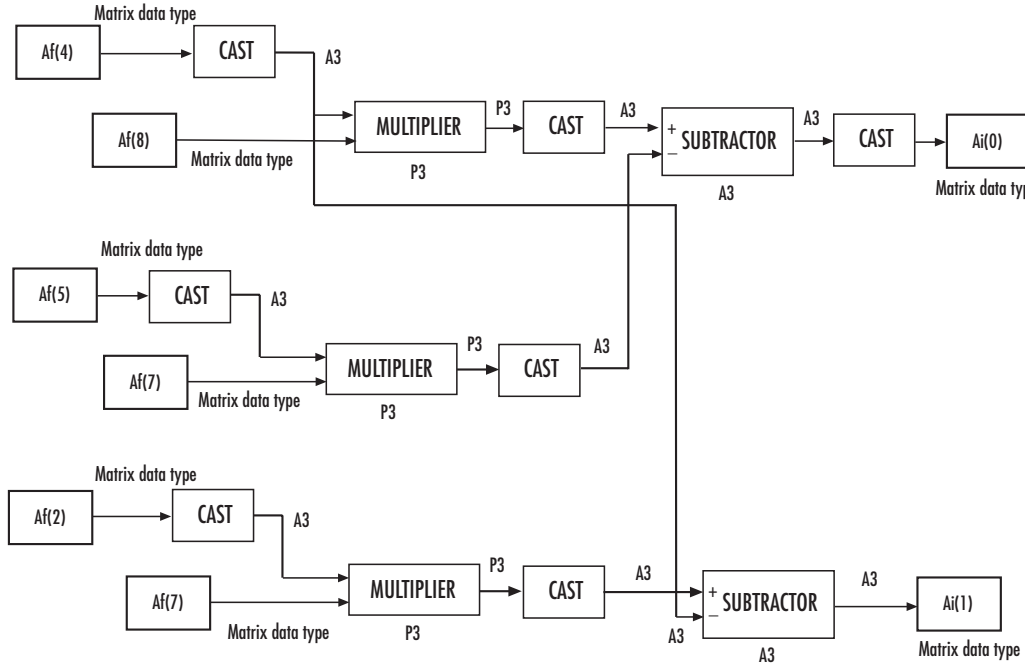
Fixed-Point Data Types

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

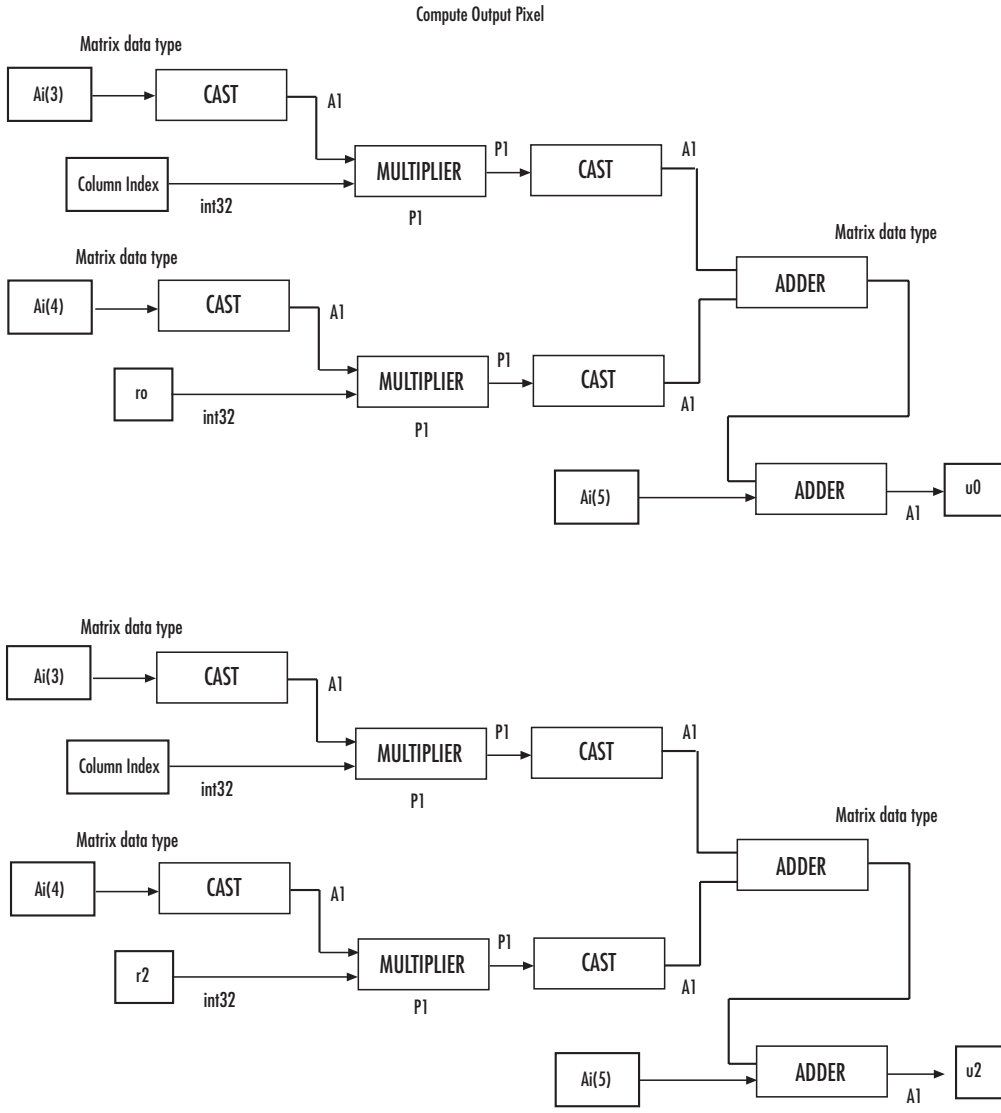


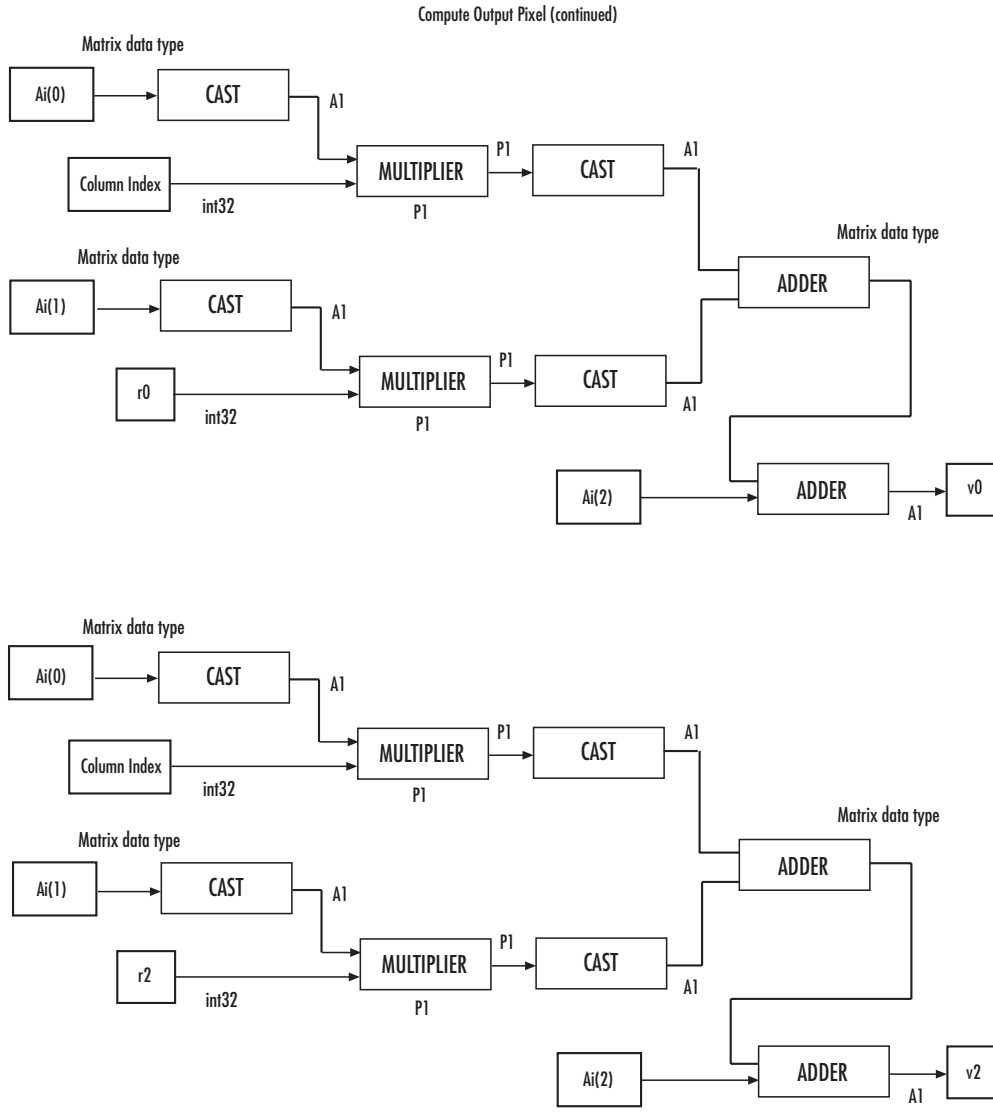
The block uses a similar computation to calculate Af(2), Af(4), Af(5), Af(6), Af(7), and Af(8).

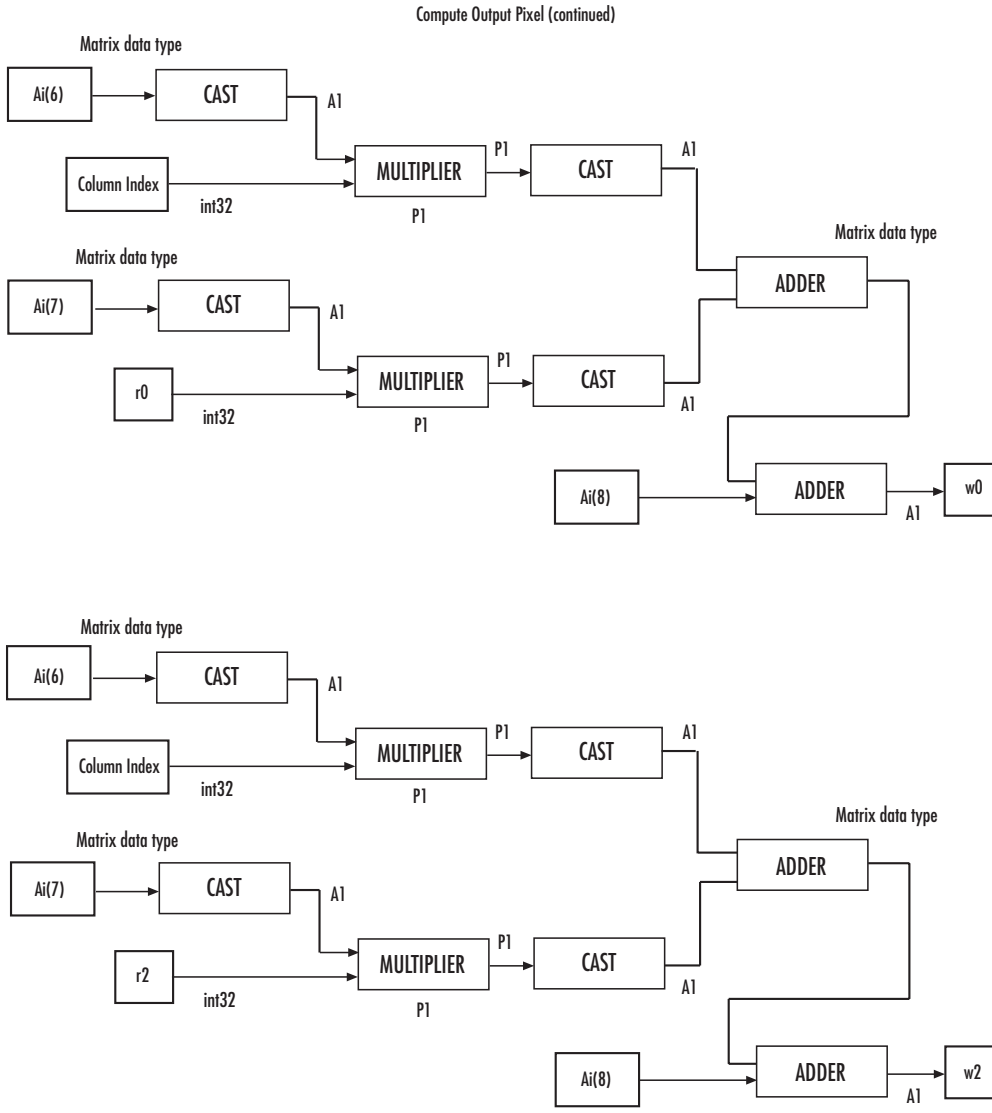
Calculate Inverse Projective Transformation Matrix (Ai)

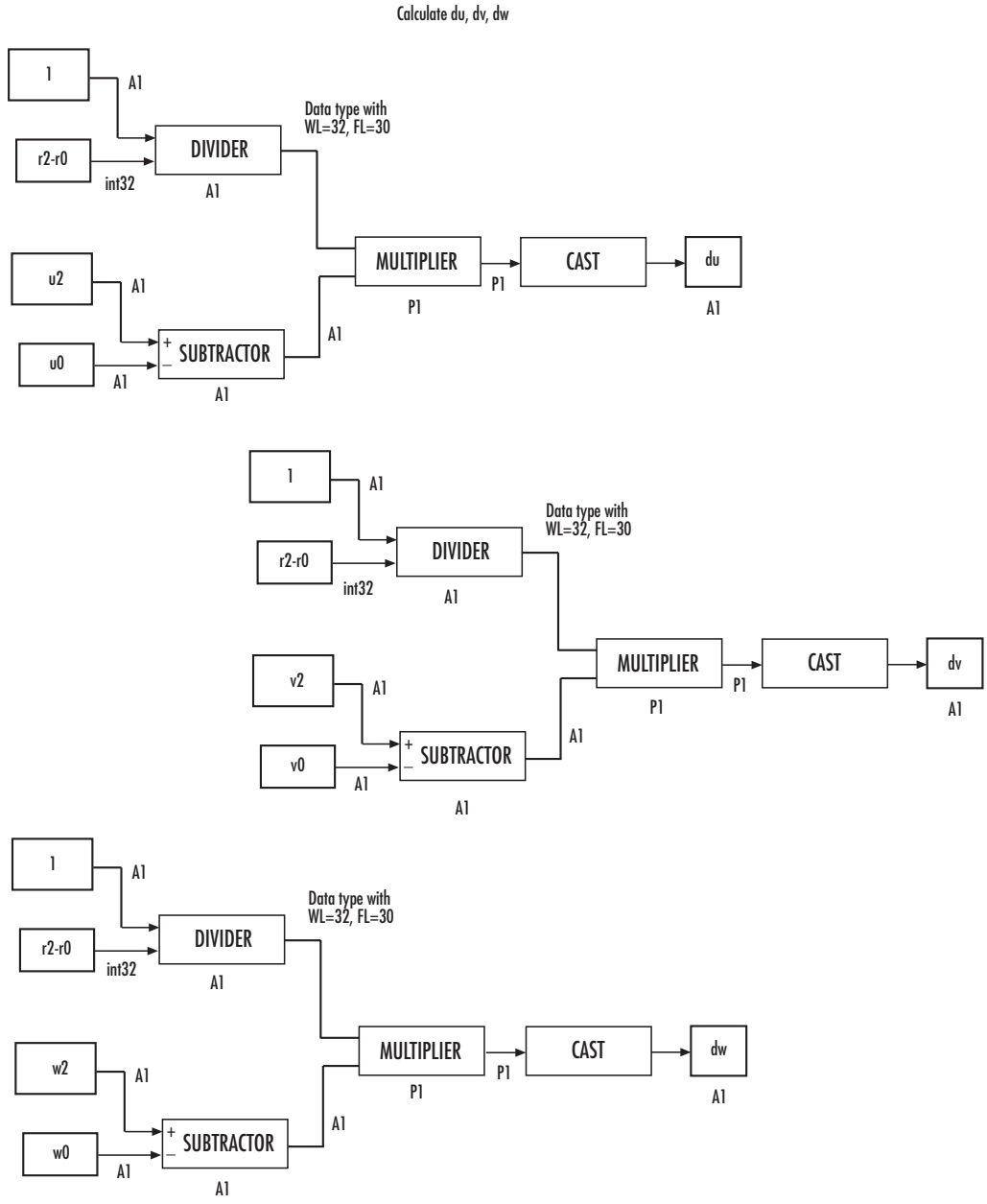


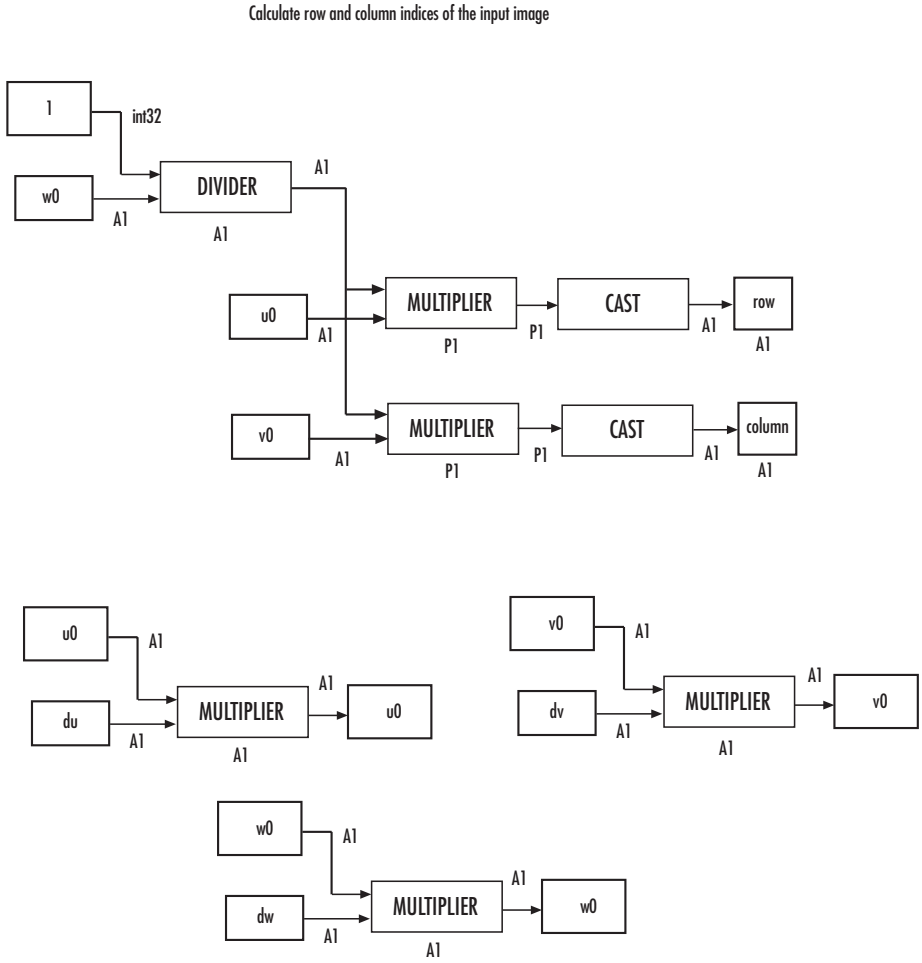
The block uses a similar computation to calculate Ai(2), Ai(3), Ai(4), Ai(5), Ai(6), Ai(7), and Ai(8).

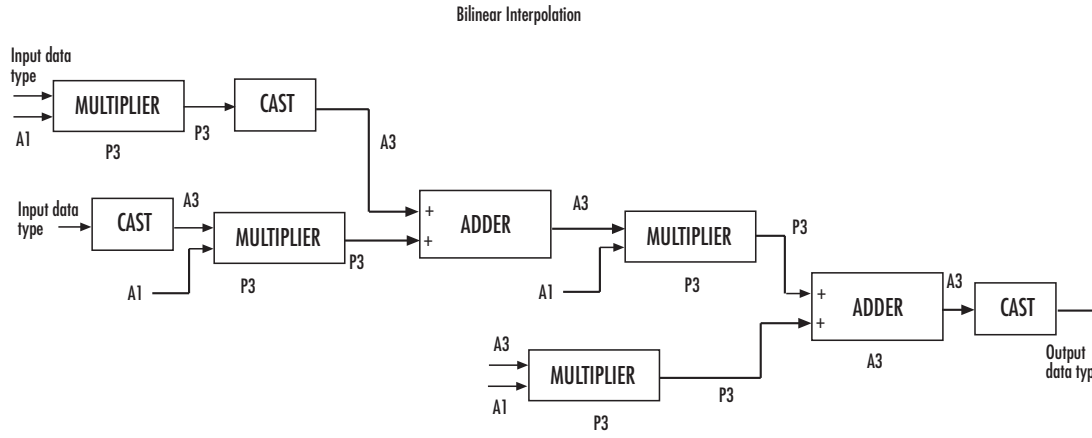

Compute Output Pixel
Matrix data type
Ai(3)
CAST
A1
Column Index
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
ADDER
Matrix data type
Ai(4)
CAST
A1
ro
int32
MULTIPLIER
P1
P1
CAST
A1
Ai(5)
ADDER
u0
A1
Matrix data type
Ai(3)
CAST
A1
Column Index
int32
MULTIPLIER
P1
P1
CAST
A1
Matrix data type
ADDER
Matrix 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
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
w0
A1
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
w2
A1

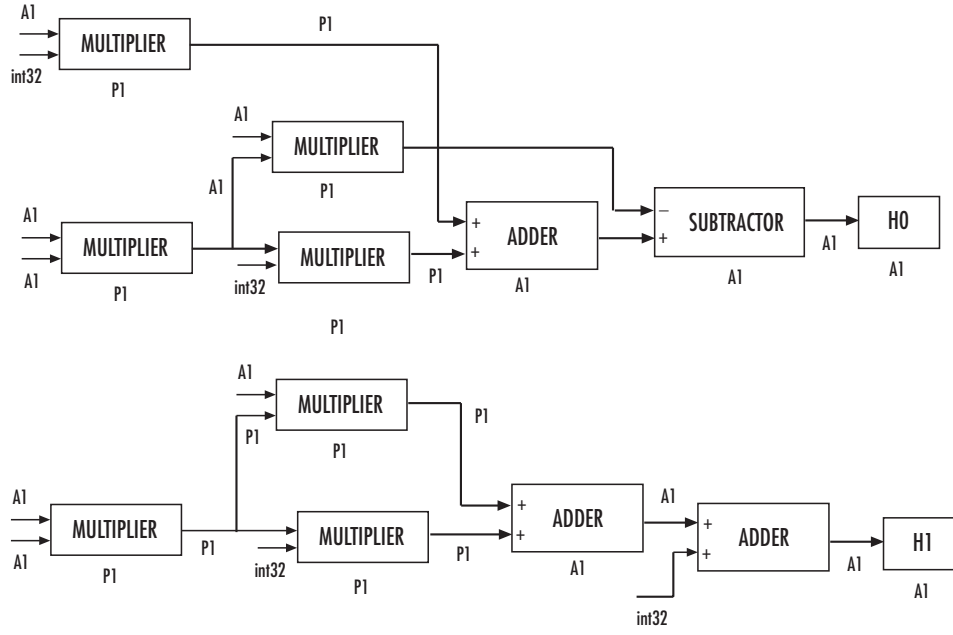

Calculate du, dv, dw
1
A1
r2-r0
int32
DIVIDER
A1
Data type with
WL=32, FL=30
u2
u0
SUBTRACTOR
MULTIPLIER
P1
CAST
du
v2
v0
dv
w2
w0
dw


Calculate row and column indices of the input image
1
int32
w0
A1
DIVIDER
A1
A1
u0
A1
MULTIPLIER
P1
CAST
P1
A1
row
A1
v0
A1
MULTIPLIER
P1
CAST
P1
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
MULTIPLIER
A1
P3
CAST
P3
A3
Input data type
CAST
A3
MULTIPLIER
A1
P3
P3
+
+
ADDER
A3
MULTIPLIER
A1
P3
P3
+
+
ADDER
A3
CAST
Output data type
A3
MULTIPLIER
A1
P3
P3

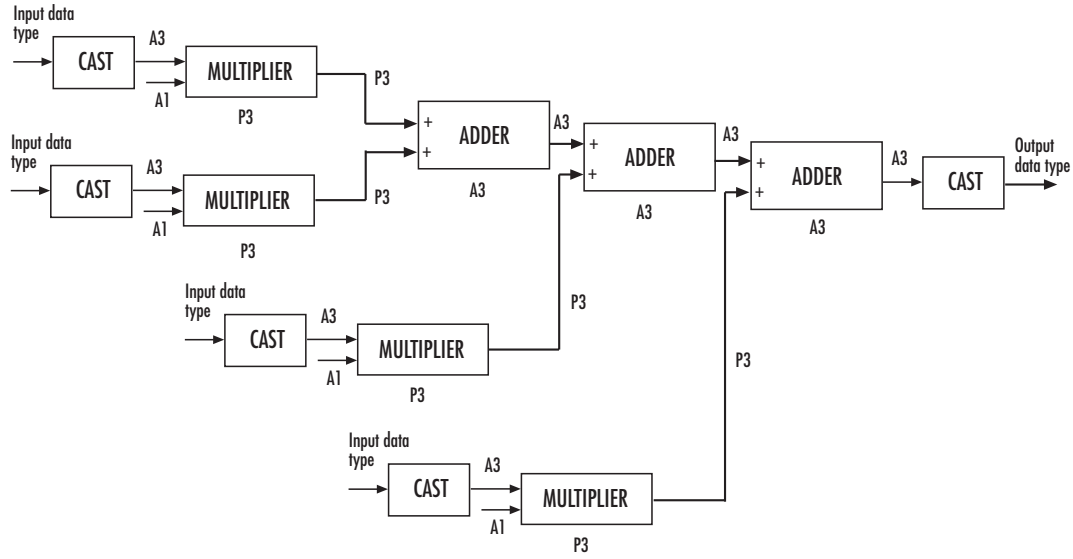
Bicubic Interpolation

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



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

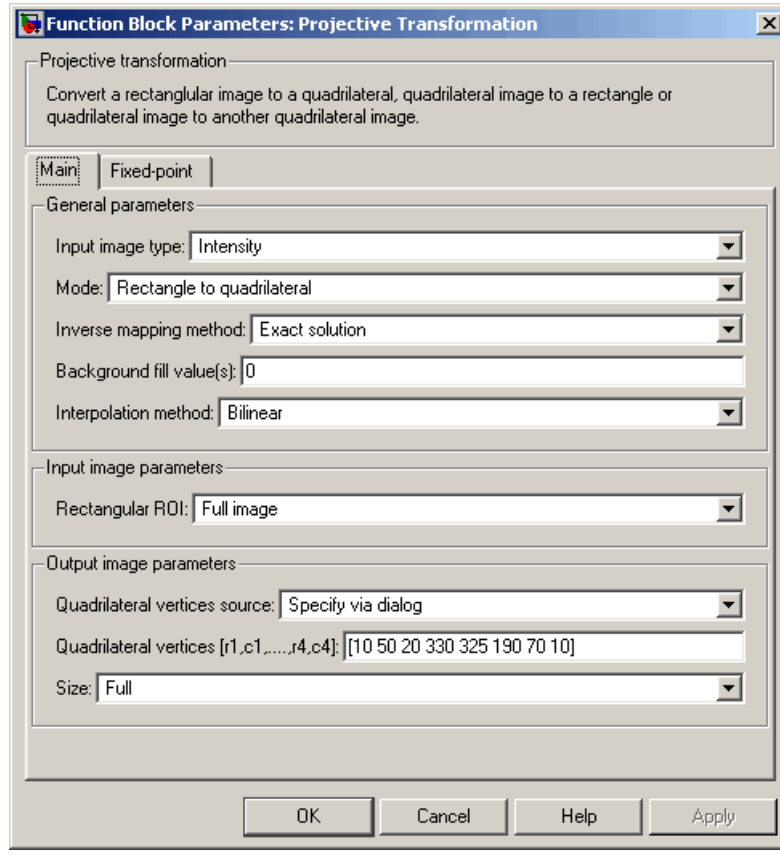
Bicubic Interpolation (continued)



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.



Input image type

Specify whether the block input is an intensity or RGB image.

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.

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.

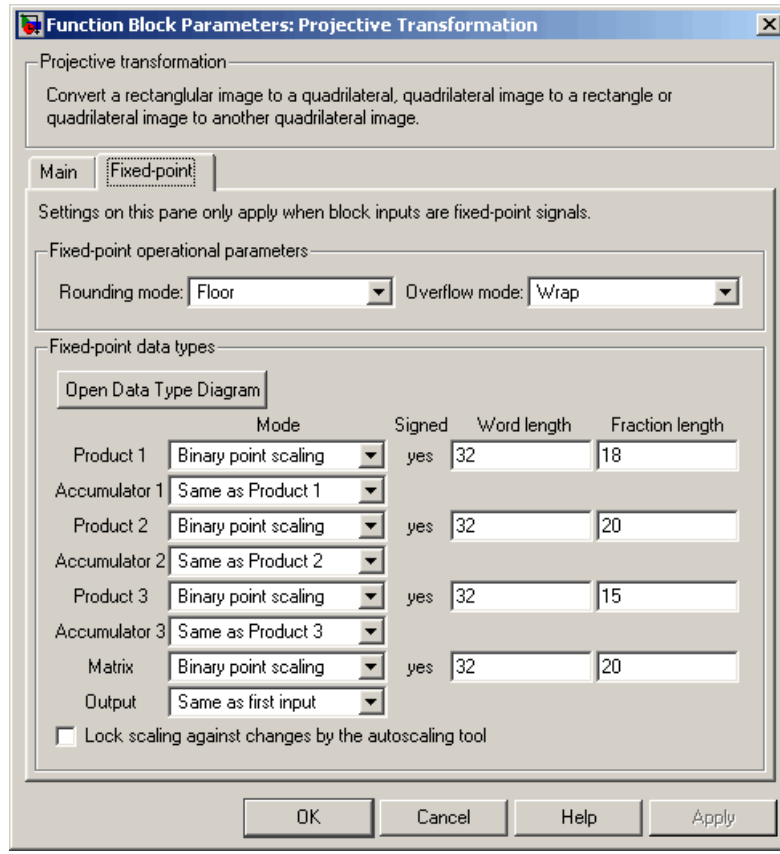
Rectangle size source

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 3Rounding mode** parameter is always set to `Nearest`.

Overflow mode

Select the overflow mode for fixed-point operations.

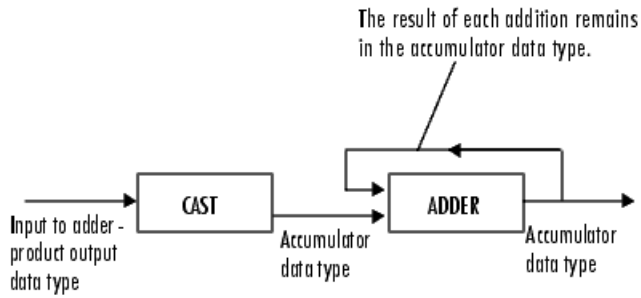
Product 1, 2, 3



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 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 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 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 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 Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

References

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

See Also

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

PSNR

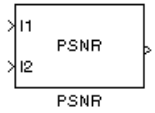
Purpose

Compute peak signal-to-noise ratio (PSNR) between images

Library

Statistics

Description



The PSNR block computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is often used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed image.

To compute the PSNR, the block first calculates the mean-squared error using the following equation:

$$MSE = \frac{\sum_{M,N} [I_1(m,n) - I_2(m,n)]^2}{M * N}$$

In the previous equation, M and N are the number of rows and columns in the input images, respectively. Then the block computes the PSNR using the following equation:

$$PSNR = 10 \log_{10} \left(\frac{R^2}{MSE} \right)$$

In the previous equation, R is the maximum fluctuation in the input image data type. For example, if the input image has a double-precision floating-point data type, then R is 1. If it has an 8-bit unsigned integer data type, R is 255, etc.

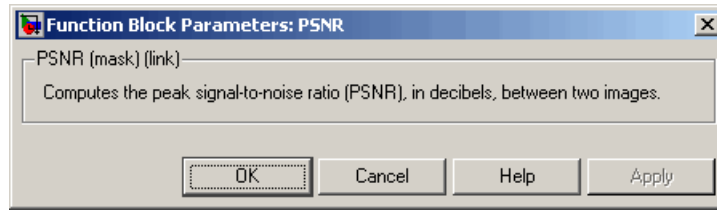
Note To compute the PSNR for color images, convert the images to a color space that separates intensity portion of the image from color information, such as YCbCr. Then perform the PSNR computation only on the intensity portion of the images.

Port	Output	Supported Data Types	Complex Values Supported
I1	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	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

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

Dialog Box

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



Purpose

Read uncompressed video frames from AVI file

Library

`vipobslib`

Description



The Read AVI File block is obsolete. It may be removed in a future version of the Video and Image Processing Blockset. Use the replacement block From Multimedia File.

The Read AVI File block reads video frames from an uncompressed AVI file and import them into a Simulink model. You can view the video frames using a To Video Display block or Video Viewer block. This block does not support audio samples. Also, this block is supported for simulation only. It produces an error during Real-Time Workshop code generation.

The output ports of the Read AVI File block change according the content of the AVI file. If the file contains RGB video frames, the R, G, and B ports appear on the block. If the file contains intensity video frames, the I port appears on the block.

Port	Output	Supported Data Types	Complex Values Supported
I	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • 8-, 16- 32-bit signed integers • 8-, 16- 32-bit unsigned integers	No
R, G, B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B ports have the same dimensions.	Same as I port	No
EOF	Scalar value	Boolean	No

Use the **File name** parameter to specify the name of the AVI file from which to read. If the location of this file is on your MATLAB path, enter the filename. If the location of this file is not on your MATLAB path, use the **Browse** button to specify the full path to the file as well as the filename.

If `filename.avi` has a colormap associated with it, the AVI file must satisfy the following conditions or the block produces an error:

- The colormap must be empty or have 256 values.
- The data must represent an intensity image.
- The pixel values must be 8-bit.

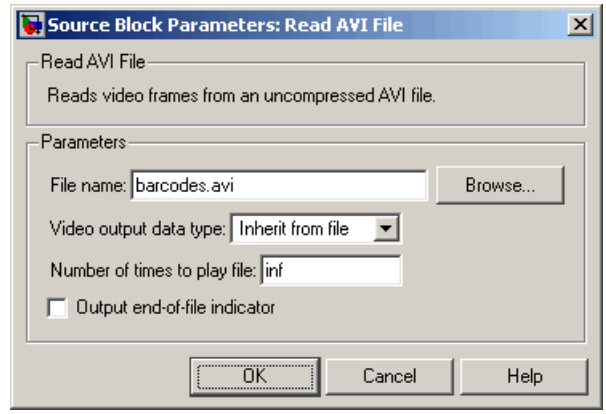
Use the **Video output data type** parameter to set the data type of the values output from the block. You can choose `double`, `single`, `int8`, `uint8`, `int16`, `uint16`, `int32`, `uint32`, and `Inherit from file`. If you choose `double` or `single`, the block scales the input pixels values and outputs values between 0 and 1. If you choose `int8`, `uint8`, `int16`, `uint16`, `int32`, or `uint32`, the blocks scales the input pixel values and outputs values between the minimum and maximum values supported by the chosen data type. If you choose `Inherit from file`, the block does not scale the input pixel values.

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

Use the **Output end-of-file indicator** parameter to determine when the last video frame in the AVI file is output from the block. When you select this check box, a Boolean output port labeled EOF appears on the block. The output from the EOF port is 1 when the last video frame is output from the block. Otherwise, the output from the EOF port is 0.

Dialog Box

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



File name
Specify the name of the AVI file from which to read.

Video output data type
Set the data type of the video data output from the block.

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

Output end-of-file indicator
Use this check box to determine whether the output is the last video frame in the AVI file.

See Also

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

To Video Display	Video and Image Processing Blockset
Video From Workspace	Video and Image Processing Blockset
Video Viewer	Video and Image Processing Blockset
Write AVI File	Video and Image Processing Blockset

Purpose

Read binary video data from files

Library

Sources

Description

The Read Binary File block reads video data from a binary file and imports it into a Simulink model.



Port	Output	Supported Data Types	Complex Values Supported
Output	Scalar, vector, or matrix of integer values	• 8-, 16- 32-bit signed integers • 8-, 16- 32-bit unsigned integers	No
EOF	Scalar value	Boolean	No

Use the **File name** parameter to specify the name of the binary file from which to read. If the location of this file is on your MATLAB path, enter the filename. If the location of this file is not on your MATLAB path, use the **Browse** button to specify the full path to the file as well as the filename.

Use the **Video format** parameter to specify the format of the binary video data. Your choices are `Four character codes` or `Custom`. See "Four Character Code Video Formats" on page 10-466 or "Custom Video Formats" on page 10-466 for more details.

Use the **Line ordering** parameter to determine how the block fills the output matrix. If you select `Top line first`, the block first fills the first row of the output matrix with the contents of the binary file. It fills the other rows in increasing order. If you select `Bottom line first`, the block first fills the last row of the output matrix. It fills the other rows in decreasing order.

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

Use the **Output end-of-file indicator** parameter to determine when the last video frame in the binary file is output from the block. When you select this check box, a Boolean output port labeled EOF appears on the block. The output from the EOF port is 1 when the last video frame in the binary file is output from the block. Otherwise, the output from the EOF port is 0.

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

Four Character Code Video Formats

Four Character Codes (FOURCC) are used to identify video formats. For more information about these codes, see `http://www.fourcc.org`.

Use the **Four character code** parameter to identify the binary file format. Then use the **Rows** and **Cols** parameters to define the size of the output matrix. These dimensions should match the matrix dimensions of the data inside the file.

Custom Video Formats

If your binary file contains data that is not in FOURCC format, you can configure the Read Binary File block to understand a custom format.

Use the **Bit stream format** parameter to specify whether your data is planar or packed. If your data is packed, use the **Rows** and **Cols** parameters to define the size of the output matrix.

Use the **Number of output components** parameter to specify the number of components in the binary file. This number corresponds to the number of block output ports.

Use the **Component**, **Bits**, **Rows**, and **Cols** parameters to specify the component name, bit size, and size of the output matrices, respectively. The block uses the **Component** parameter to label the output ports.

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

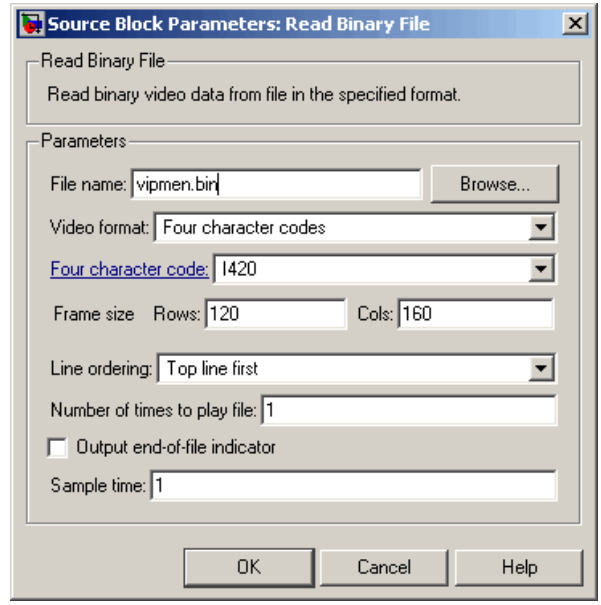
Select the **Interlaced video** check box if the binary file contains interlaced video data.

Select the **Input file has signed data** check box if the binary file contains signed integers.

Use the **Byte order in binary file** to indicate whether your binary file has little endian or big endian byte ordering.

Dialog Box

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



File name

Specify the name of the binary file.

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

Four character code
From the list, select the binary file format.

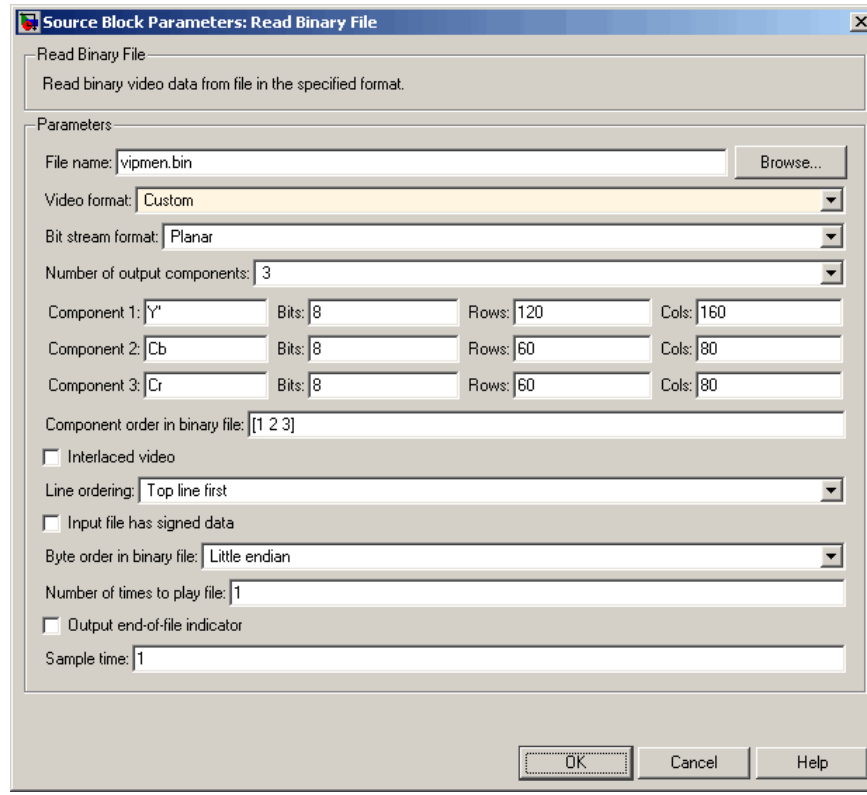
Frame size: Rows, Cols
Define the size of the output matrix. These dimensions should match the matrix dimensions of the data inside the file.

Line ordering
Specify how the block fills the output matrix. If you select `Top line first`, the block first fills the first row of the output matrix with the contents of the binary file. If you select `Bottom line first`, the block first fills the last row of the output matrix.

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

Output end-of-file indicator
Use this check box to determine whether the output is the last video frame in the binary file.

Sample time
Enter the sample period of the output signal.



Bit stream format

Specify whether your data is planar or packed.

Frame size: Rows, Cols

Define the size of the output matrix. This parameter is visible if, for the **Bit stream format** parameter, you select `Packed`.

Number of output components

Specify the number of components in the binary file.

Component, Bits, Rows, Cols

Specify the component name, bit size, and size of the output matrices, respectively.

Component order in binary file
Specify how the components are arranged within the binary file.

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

Input file has signed data
Select this check box if the binary file contains signed integers.

Byte order in binary file
Use this parameter to indicate whether your binary file has little endian or big endian byte ordering

See Also

From Multimedia File — Video and Image Processing Blockset

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).

Port	Input/Output	Supported Data Types	Complex Values Supported
Input	Matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	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 integers • 8-, 16-, 32-bit unsigned integers	No
Output	Matrix of intensity values	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. This block supports a signal represented by a Simulink virtual bus.

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 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.

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)

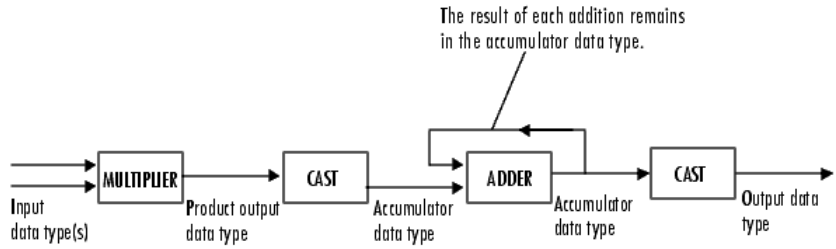
in the input matrix, I, that you want to resize. The input to this port must be a four-element vector, [row column height width]. The first two elements define the upper-left corner of the ROI, and the second two elements define the height and width of the ROI.

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

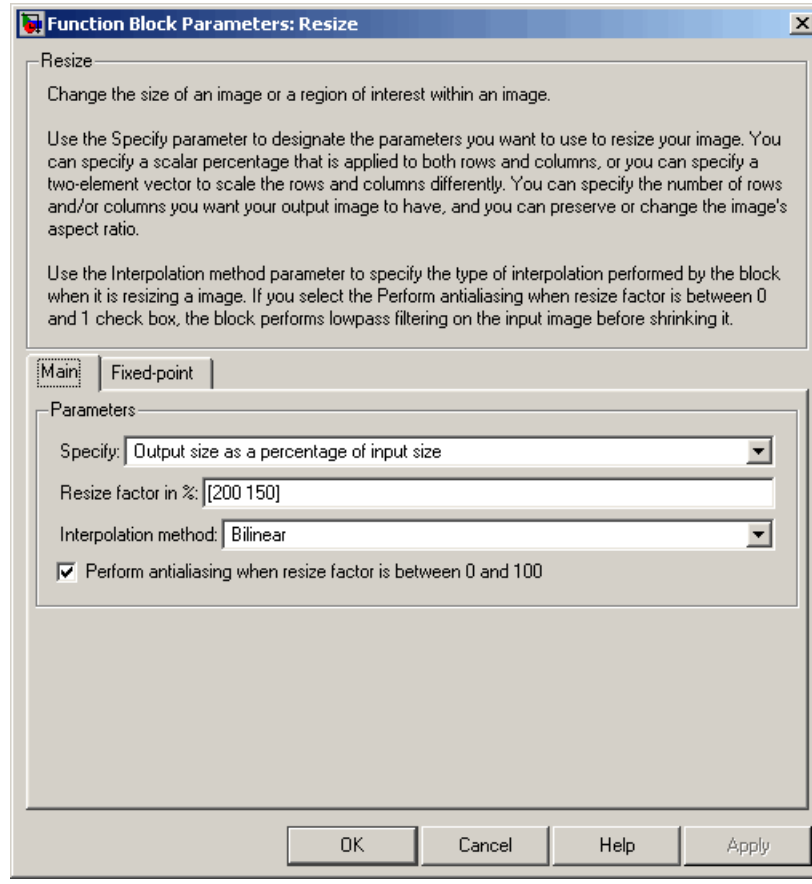
The following diagram shows the data types used in the Resize block for fixed-point signals.



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:



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`.

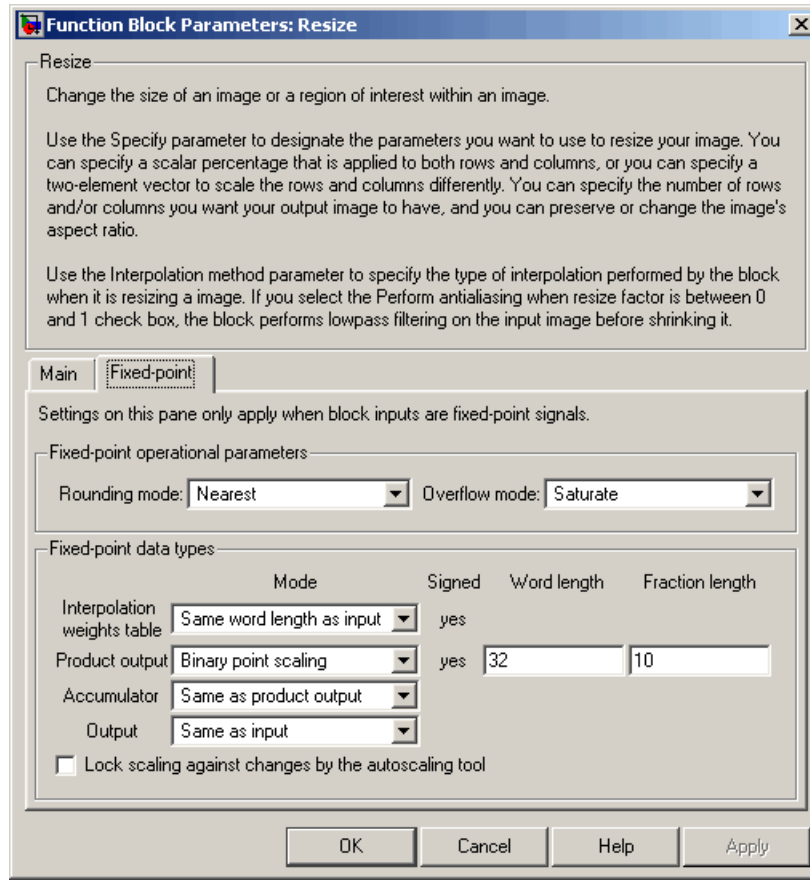
Interpolation method
Determine which interpolation method the block uses to resize the image. If you select `Nearest neighbor`, the block uses one nearby pixel to interpolate the pixel value. If you select `Bilinear`, the block uses two nearby pixels to interpolate the pixel value. If you select `Bicubic` 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 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.



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.

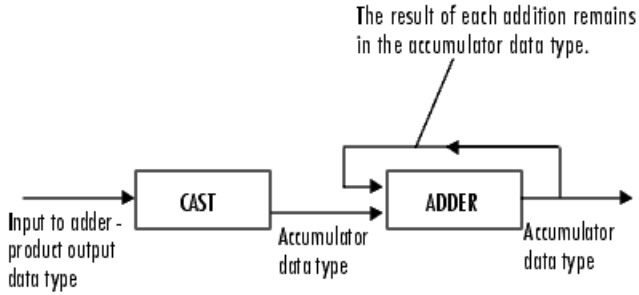
Product output



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 is 0.

Accumulator



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

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

References

[1] Ward, Joseph and David R. Cok. "Resampling Algorithms for Image Resizing and Rotation", *Proc. SPIE Digital Image Processing Applications*, vol. 1075, pp. 260-269, 1989.

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

See Also

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

Rotate

Purpose

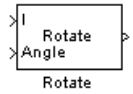
Rotate image by specified angle

Library

Geometric Transformations

Description

Use the Rotate block to rotate an image by an angle specified in radians.



Port	Input/Output	Supported Data Types	Complex Values Supported
I	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 integers • 8-, 16-, 32-bit unsigned integers	No
Angle	Rotation angle	Same as I port	No
Output	Rotated matrix	Same as I 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. This block supports a signal represented by a Simulink virtual bus.

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. 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)**, **Display rotated image in**, **Sine value computation method**, and **Interpolation method** 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)** , **Sine value computation method**, and **Interpolation method** 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,

$$0 < \max angle \le \pi$$

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

$0 \le angle \le \max angle$, 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.

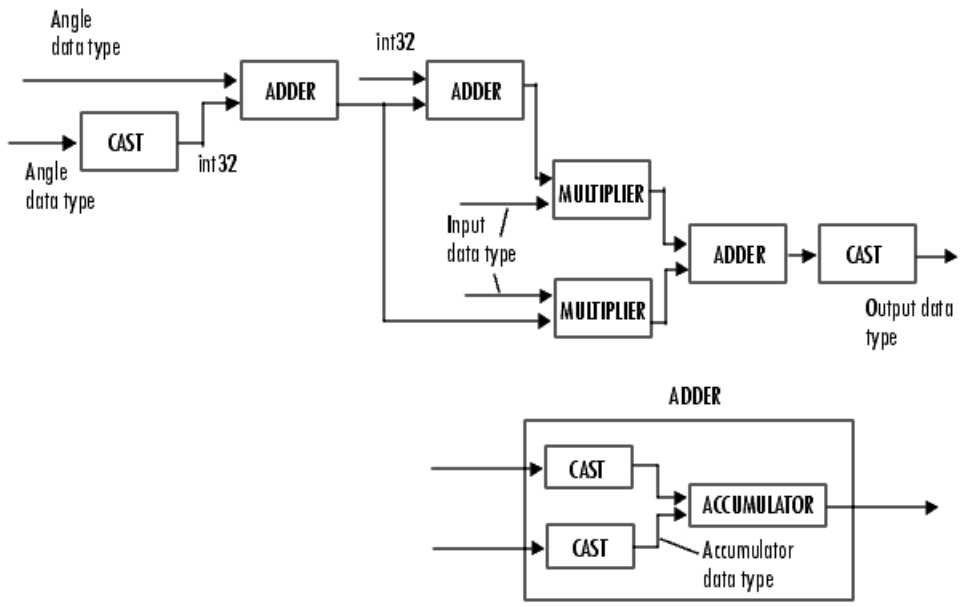
Use the **Sine value computation method** parameter to specify how much memory the Rotate block requires. If you select `Trigonometric function`, the block computes sine and cosine values it needs to calculate the rotation of your image during the simulation. If you select `Table lookup`, the block computes 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 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 “Interpolation Overview” on page 5-2.

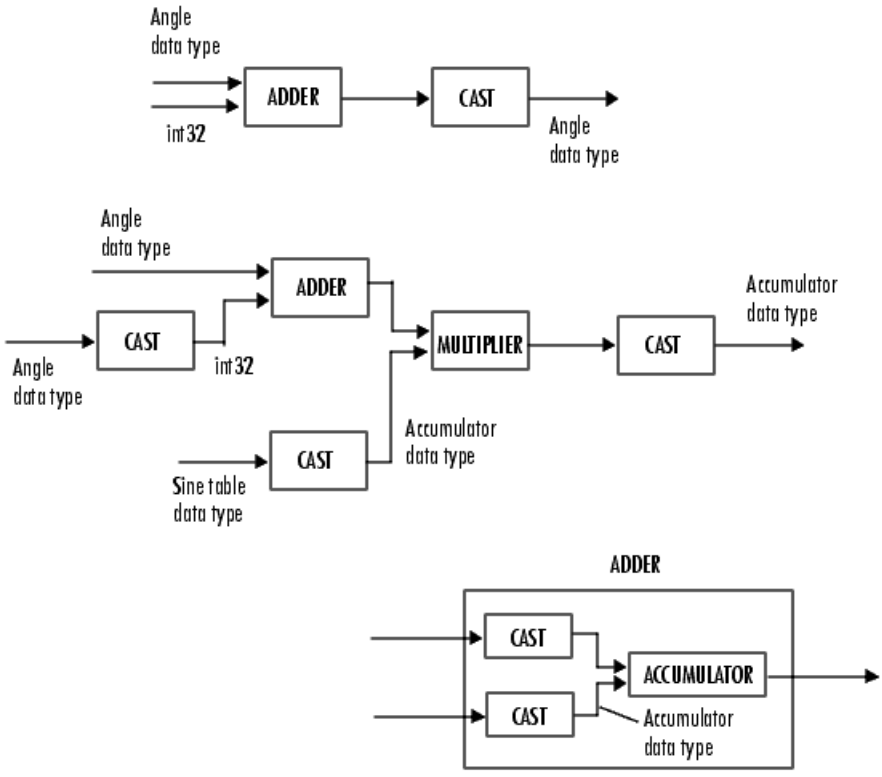
Fixed-Point Data Types

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.

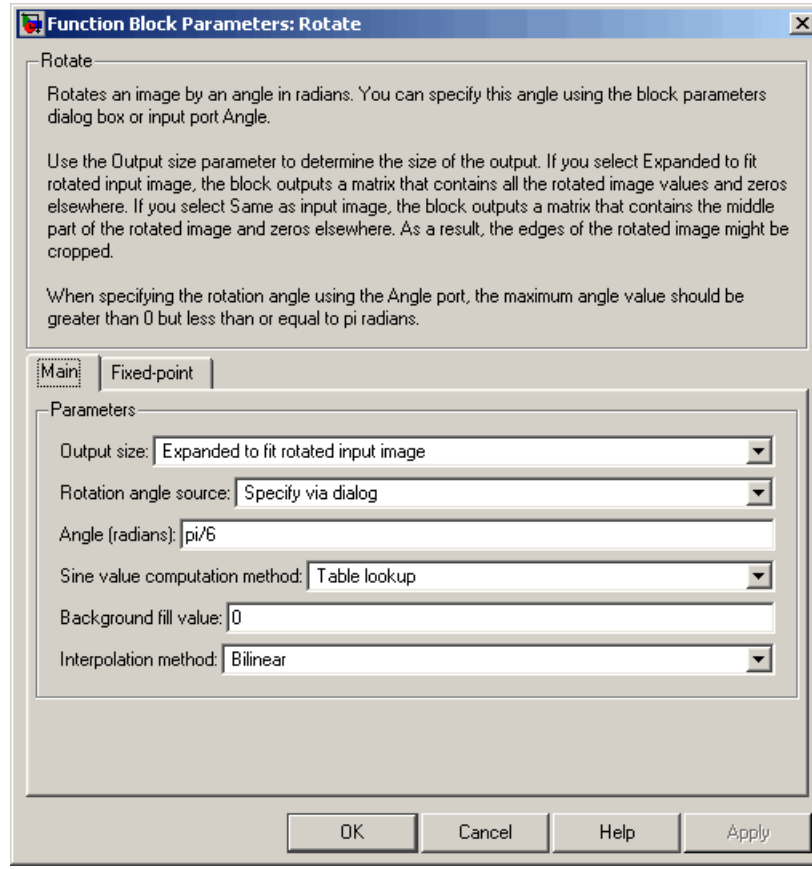
The Rotate block requires additional data types. The Sine table value has the same word length as the angle data type and a fraction length that is equal to its word length minus one. The following diagram shows how these data types are used inside the block.



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.



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, for the **Output size** parameter, you select `Expanded to fit rotated input image`, and, for the **Rotation angle source** parameter, you select `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`.

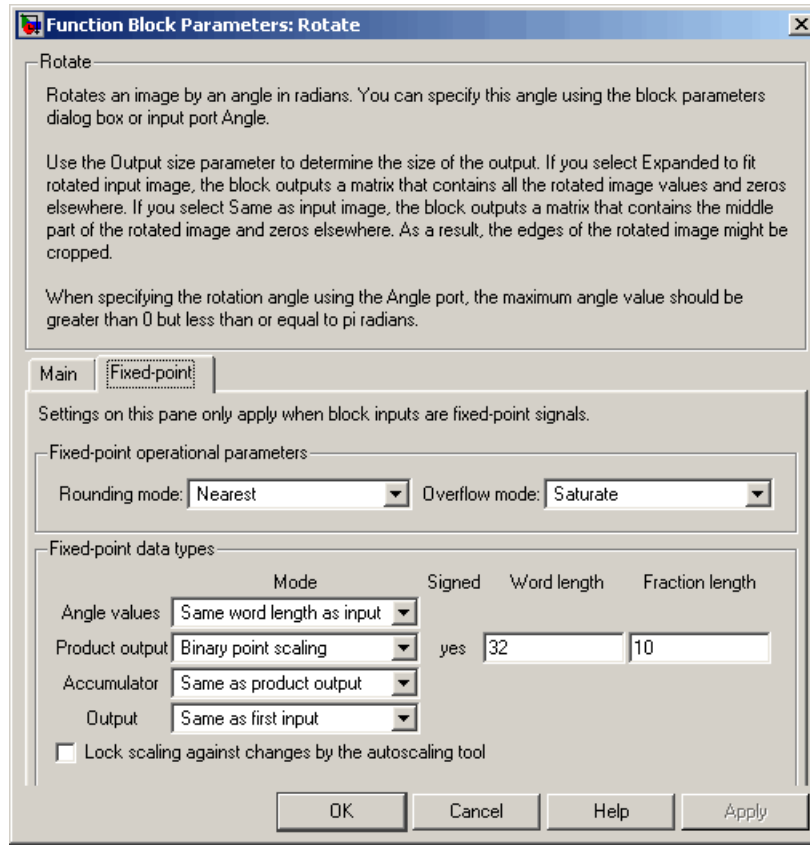
Sine value computation method

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. This parameter is visible if, for the **Rotation angle 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 rotate the image. If you select `Nearest neighbor`, the block uses the value of one nearby pixel for the new pixel value. If you select `Bilinear`, the new pixel value is the weighted average of the two nearest pixel values. If you select `Bicubic`, the new pixel value is the weighted average of the four nearest pixel values.

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



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 is 0.

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

Product output

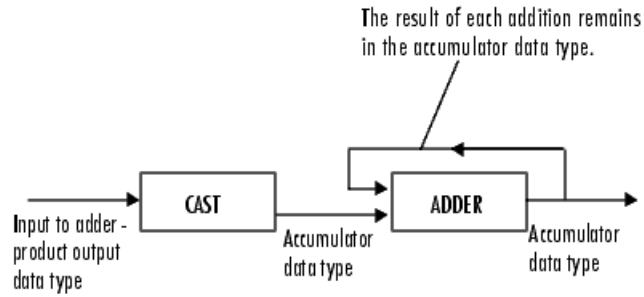


As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths.

- 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 is 0.

Accumulator



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

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

References

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

See Also

Resize	Video and Image Processing Blockset
Translate	Video and Image Processing Blockset
Shear	Video and Image Processing Blockset
`imrotate`	Image Processing Toolbox

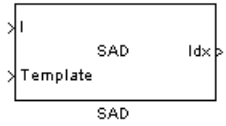
Purpose

Perform 2-D sum of absolute differences (SAD)

Library

Analysis & Enhancement

Description



The SAD block finds the similarity between two input images by performing the sum of absolute differences. The greater the similarity between the two matrices, the smaller the SAD values that result. Assume that input matrix I has dimensions (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 integers • 8-, 16-, and 32-bit unsigned integers	No
Template	Matrix of intensity values	Same as I port	No
ROI	Four-element vector that defines the ROI	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
Val	Matrix of SAD values	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
Idx	Scalar value that represents the zero-based index location of the minimum SAD value	• 32-bit signed integers	No

Port	Input/Output	Supported Data Types	Complex Values Supported
NVals	N-by-N matrix of SAD values centered around the minimum SAD value	Same as Val port	No
NValid	Boolean 0 or 1 that represents whether or not the block went beyond the dimensions of the SAD value matrix to construct an N-by-N matrix around the minimum SAD value	Boolean	No

The data type of the two input signals must be the same. The output signal is the same data type as the input signals.

The dimensions of the output at the Val port are determined by the sizes of the inputs at ports I and Template. If the input at port I has dimensions (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

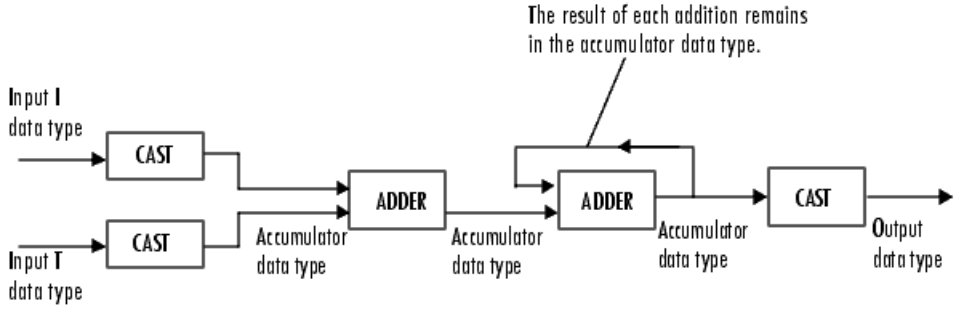
- `Ignore` -- Proceed with the computation and do not issue an alert. The output is not valid.
- `Warn` -- Display a warning message in the MATLAB Command Window, and continue the simulation. The output is not valid.
- `Error` -- Display an error dialog box and terminate the simulation.

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

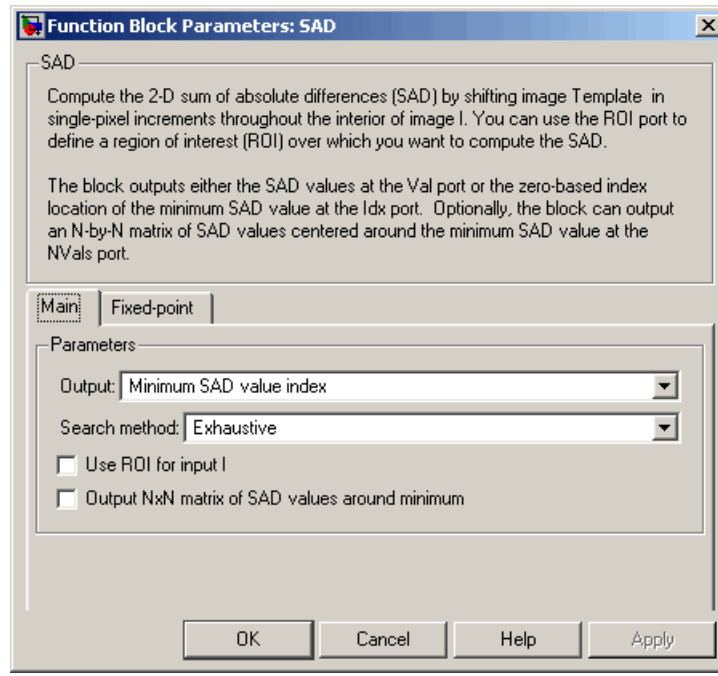
The following diagram shows the data types used in the SAD block for fixed-point signals.



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.



Output

Specify the output of the block. Your choices are `SAD values` or `Minimum SAD value index`. If you select `Minimum SAD value index`, the block outputs the zero-based index location of the minimum SAD value.

Search method

Specify how the block searches for the minimum difference between the two input matrices. If you select `Exhaustive`, the block searches for the minimum difference pixel-by-pixel. If you select `Three-step`, the block searches for the minimum difference

using a steadily decreasing step size. This parameter is visible if, for the **Output** parameter, you select `Minimum SAD value index`.

Use ROI for input I

If you select this check box, the ROI port appears on the block. Use this port to define a region of interest (ROI) in the input matrix, I, over which you want to compute the SAD. This parameter is visible if, for the **Output** parameter, you select `Minimum SAD value index`.

Invalid ROI

Specify the block's behavior if you enter a ROI that is outside the bounds of the input matrix, I. The options are `Ignore`, `Warn`, or `Error`.

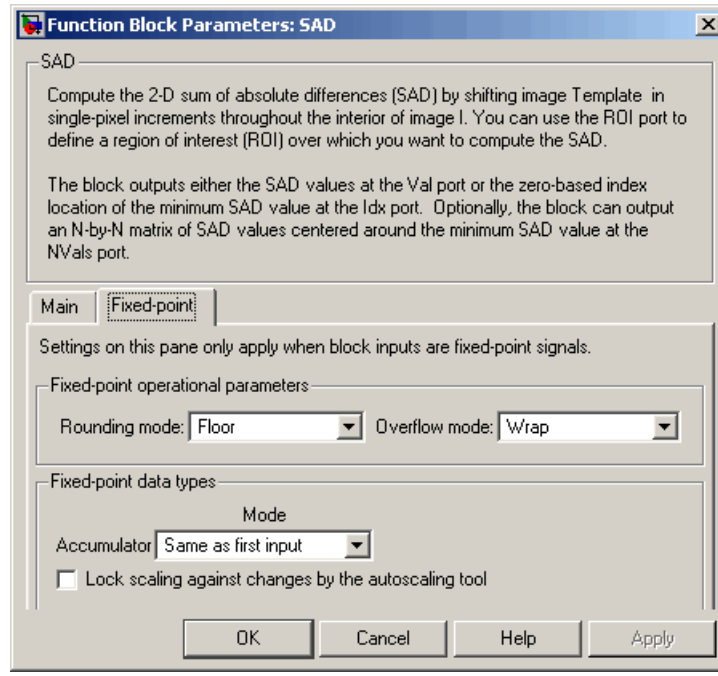
Output NxN matrix of SAD values around minimum

If you select this check box, the NVals and NValid ports appear on the block. The block outputs an N-by-N matrix of SAD values centered around the minimum SAD value at the NVals port. If the block must go beyond the dimensions of the SAD value matrix to construct the N-by-N output matrix, the block outputs a Boolean 0 at the NValid port. Otherwise, the block outputs a Boolean 1 at the NValid port. This parameter is visible if, for the **Output** parameter, you select `Minimum SAD value index`.

Size (N) of square matrix

Enter an odd number that determines the size of the N-by-N matrix of SAD values. This parameter is visible if you select the **Output NxN matrix of SAD values around minimum** check box.

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



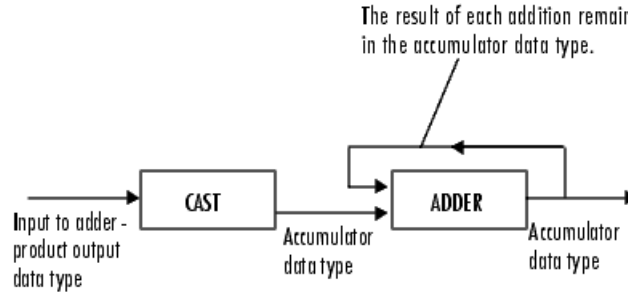
Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Accumulator



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

- When you select `Same as first input`, these characteristics match those of the input to the block. When you have a Boolean input, you cannot select this choice.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you can enter the word length, in bits, and the slope of the accumulator. The bias of all signals in the Video and Image Processing Blockset is 0.

Output

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

- When you select `Same as first input`, these characteristics match those of the first input to the block. When you have a Boolean input, you cannot select this choice.
- When you select `Binary point scaling`, you can enter the word length and the fraction length of the output, in bits.

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

This parameter is not visible if, for the **Output** parameter you select `Minimum SAD value index`, and you clear the **Output NxN matrix of SAD values around minimum** check box.

Lock scaling against changes by the autoscaling tool
Select this parameter to prevent any fixed-point scaling you specify in this block mask from being overridden by the autoscaling tool in the Fixed-Point Settings interface. For more information about the autoscaling tool, refer to in the Signal Processing Blockset documentation.

References

Koga, T., et al. *Motion-compensated interframe coding for video conferencing*. In Nat. Telecommun. Conf., Nov. 1981, G5.3.1-5, New Orleans, LA.

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

Shear

Purpose

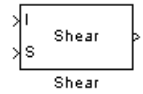
Shift rows or columns of image by linearly varying offset

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.



Port	Input/Output	Supported Data Types	Complex Values Supported
I	Matrix of intensity values 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 integers • 8-, 16-, 32-bit unsigned integers	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	Matrix of shifted values	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. This block supports a signal represented by a Simulink virtual bus.

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.

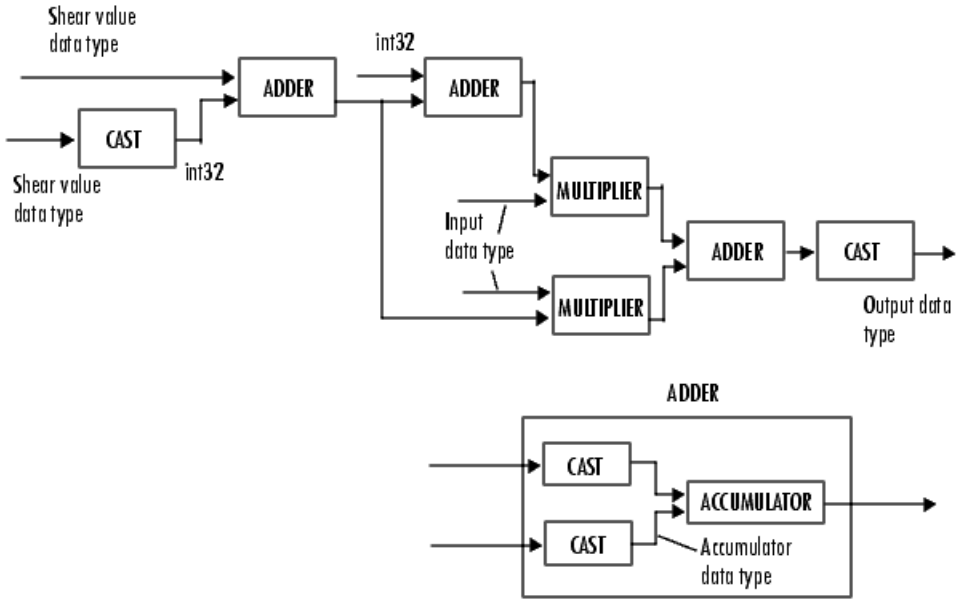
The number of pixels the block considers affects the complexity of the computation. Therefore, the nearest-neighbor interpolation is the most computationally efficient. However, because the accuracy of the method is proportional to the number of pixels considered, the bicubic method is the most accurate. For more information, see “Interpolation Overview” on page 5-2.

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

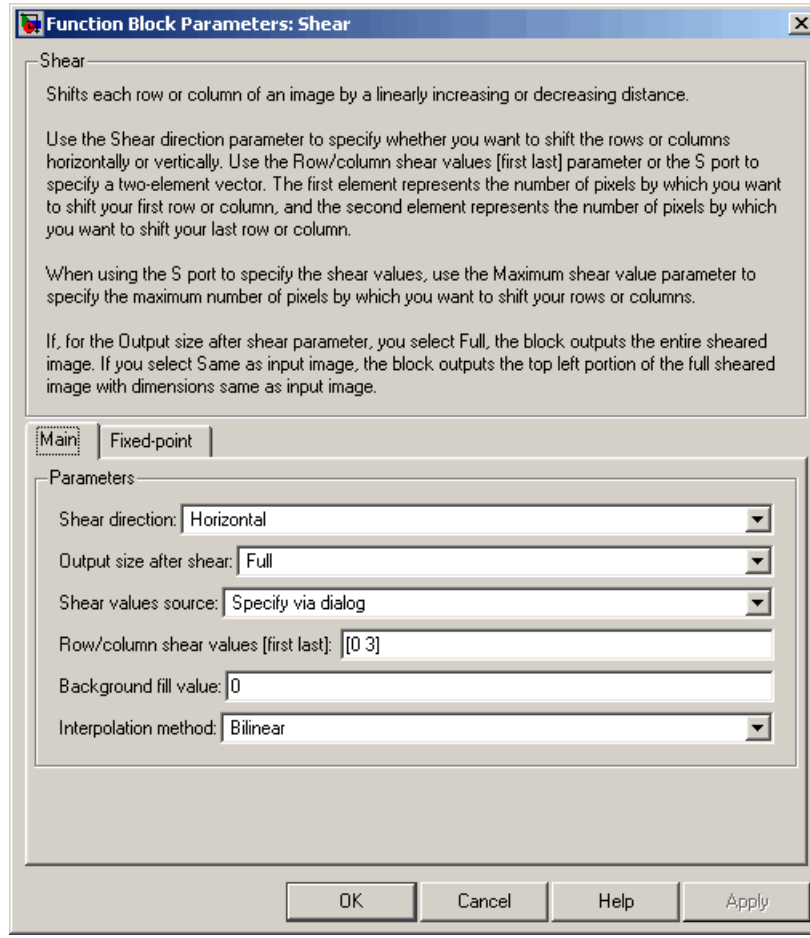
The following diagram shows the data types used in the Shear block for bilinear interpolation of fixed-point signals.



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.



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.

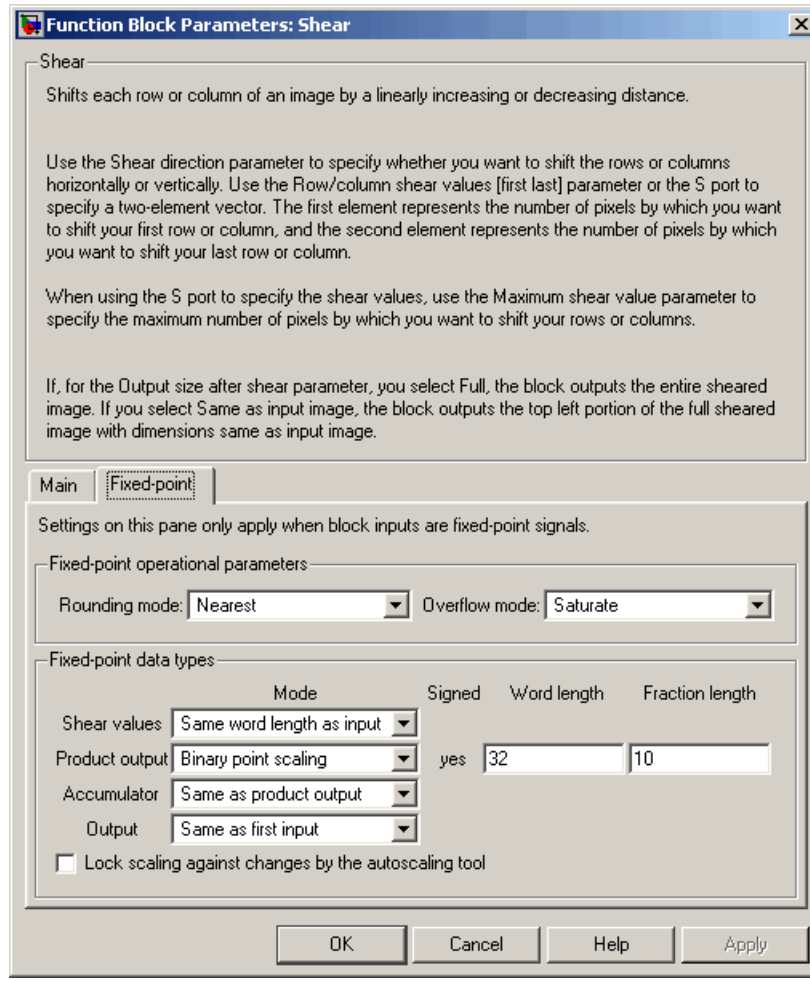
Row/column shear values [first last]
Enter a two-element vector that represents the number of pixels by which to shift your first and last rows or columns. This parameter is visible if, for the **Shear values source** parameter, you select `Specify via dialog`.

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.



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 is 0.

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

Product output

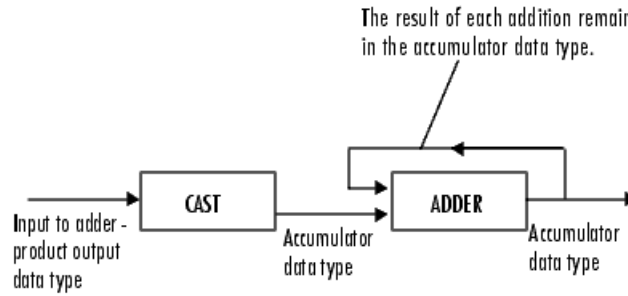


As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this

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 is 0.

Accumulator



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

- When you select `Same as product output`, these characteristics match those of the product output.
- When you select `Same as first input`, these characteristics match those of the first input to the block at the I port.

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

References

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

See Also

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

To Multimedia File

Purpose

Write video frames and audio samples to multimedia file

Library

Sinks

Description

The To Multimedia File block is a Signal Processing Blockset block. For more information, see the To Multimedia File block reference page in the Signal Processing Blockset documentation.

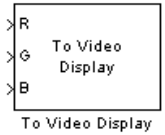
Purpose

Send video data to display devices

Library

Sinks

Description



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

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

Port	Input	Supported Data Types	Complex Values Supported
I	Matrix of intensity values	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16, and 32-bit signed integers • 8-, 16, and 32-bit unsigned integers	No
R, G, B	Matrix that represents one plane of the RGB video stream. Inputs to the R, G, or B ports must have the same dimensions and data type.	Same as I port	No

For the block to display video data properly, double- and single-precision floating-point pixel values must be from 0 to 1. For any other data type,

the pixel values must be between the minimum and maximum values supported by their data type.

Use the **Input image type** parameter to specify the type of image or video stream to view. If you select `RGB`, the R, G, and B ports appear on the block. Use this option for RGB images and video. If you select `Intensity`, the I port appears on the block. Use this option for binary and intensity images and video.

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

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

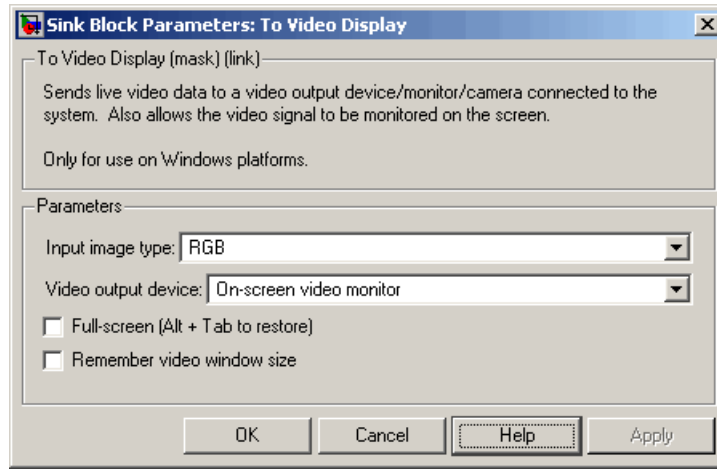
Select the **Remember video window size** check box if you want the block to save changes you make to the size of the video window.

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

1. Replace the To Video Display block with a Video Viewer block.
2. Disable the DirectDraw Acceleration, Direct3D Acceleration, and AGP Texture Acceleration on your system.
   a. **Start > Run**.
   b. For the **Open** parameter, type `dxdiag`. Click **OK**. The DirectX Diagnostic Tool opens.
   c. On the **Display** tab, click the **Disable** buttons that are next to DirectDraw Acceleration, Direct3D Acceleration, and AGP Texture Acceleration.
   d. Click **Exit**.

Dialog Box

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



Input image type

Specify the type of image or video stream to view. Your choices are `RGB` or `Intensity`.

Video output device

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

Full-screen

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

Remember video window size

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

See Also

Frame Rate Display	Video and Image Processing Blockset
From Multimedia File	Video and Image Processing Blockset
To Multimedia File	Video and Image Processing Blockset
Video To Workspace	Video and Image Processing Blockset
Video Viewer	Video and Image Processing Blockset

Top-hat

Purpose

Perform top-hat filtering on intensity or binary images

Library

Morphological Operations

Description



The Top-hat block performs top-hat filtering on an intensity or binary image using a predefined neighborhood or structuring element. Top-hat filtering is the equivalent of subtracting the result of performing a morphological opening operation on the input image from the input image itself. This block uses flat structuring elements only.

Port	Input/Output	Supported Data Types	Complex Values Supported
I	Scalar, vector, or matrix of intensity values or scalar, vector, or matrix that represents one plane of the input RGB video stream	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
Nhood	Matrix or vector of 1s and 0s that represents the neighborhood values	Boolean	No
Output	Scalar, vector, or matrix that represents the filtered image	Same as I port	No

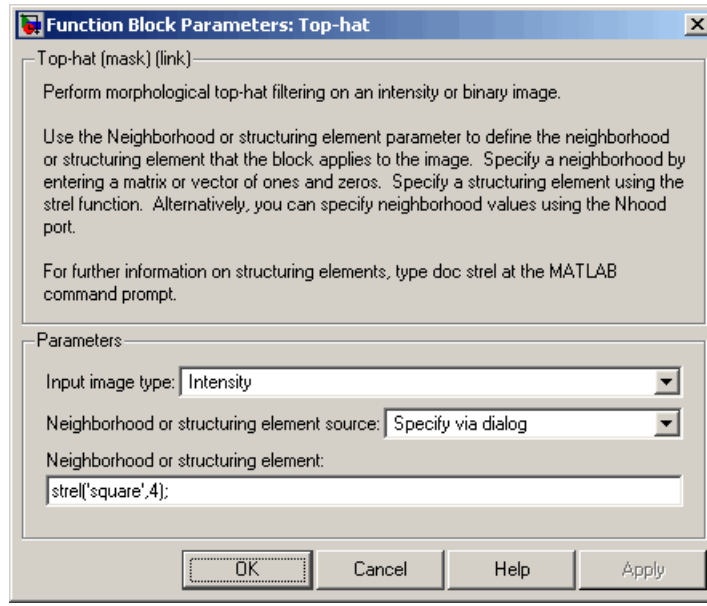
If your input image is a binary image, for the **Input image type** parameter, select `Binary`. If your input image is an intensity image, select `Intensity`.

Use the **Neighborhood or structuring element source** parameter to specify how to enter your neighborhood or structuring element values. If you select `Specify via dialog`, the **Neighborhood or structuring element** parameter appears in the dialog box. If you select `Input port`, the Nhood port appears on the block. Use this port to enter your neighborhood values as a matrix or vector of 1s and 0s. Choose your structuring element so that it matches the shapes you want to remove from your image. You can only specify a it using the dialog box.

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

Dialog Box

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



Input image type

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

Neighborhood or structuring element source

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

Neighborhood or structuring element

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

structuring element source parameter, you select `Specify via dialog`.

See Also

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

Trace Boundaries

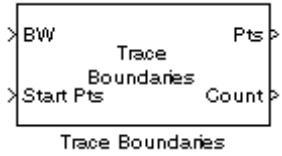
Purpose

Trace object boundaries in binary images

Library

Analysis & Enhancement

Description



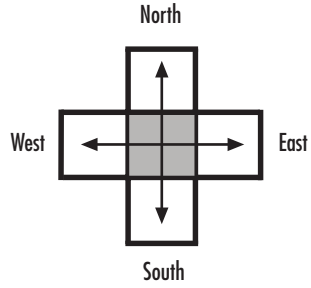
The Trace Boundaries block traces object boundaries in binary images, where nonzero pixels represent objects and 0 pixels represent the background.

Port	Input/Output	Supported Data Types	Complex Values Supported
BW	Scalar, vector, or matrix that represents a binary image	Boolean	No
Start Pts	2-by-N matrix where each column represents the coordinates of the boundary starting point, and N is the total number of starting points: $\begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ y_1 & y_2 & \cdots & y_N \end{bmatrix}$	• Double-precision floating point • Single-precision floating point • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No

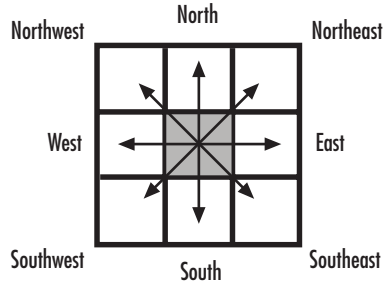
Port	Input/Output	Supported Data Types	Complex Values Supported
Pts	2M-by-N matrix where each column contains the locations of the boundary pixels, M is the maximum number of boundary pixels, and N is the total number of starting points: $\begin{bmatrix} x_{11} & \cdots & x_{N1} \\ y_{11} & \cdots & y_{N1} \\ x_{12} & \cdots & x_{N2} \\ y_{12} & \cdots & y_{N2} \\ \vdots & \ddots & \vdots \\ x_{1M} & \cdots & x_{NM} \\ y_{1M} & \cdots & y_{NM} \end{bmatrix}$	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 integers	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.

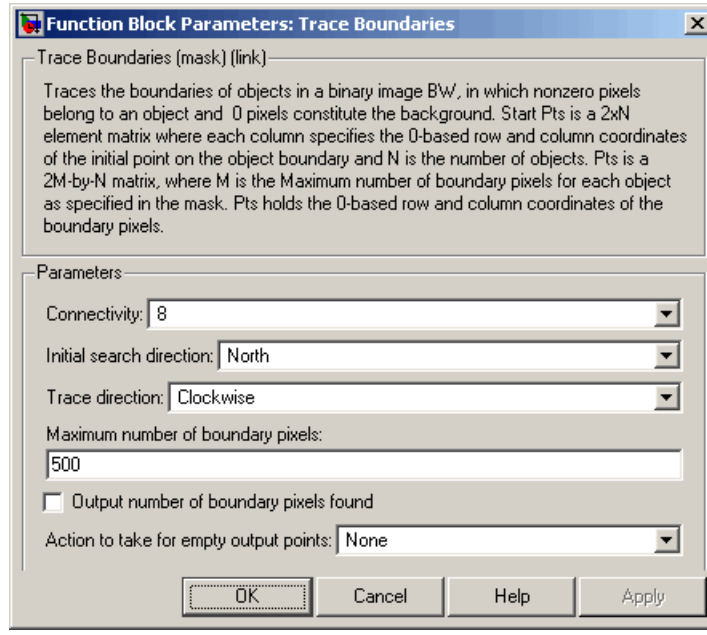
Because you specify the number of rows of the Pts port output matrix using the **Maximum number of boundary pixels** parameter, use the **Action to take for empty output points** parameter to specify what happens to the empty elements in this vector when the number of boundary pixels is less than the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



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

Translate

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.

Port	Input/Output	Supported Data Types	Complex Values Supported
I/Input	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	No
Offset	Scalar, vector, or matrix of intensity values	Same as I port	No
Output	Scalar, vector, or matrix of intensity values	Same as I port	No

The input to the Offset port must be the same data type as the input to the I port. The output is the same data type as the input to the I port. This block supports signals represented by Simulink virtual buses.

Use the **Output size after translation** parameter to specify the size of the translated image. If you select `Full`, the block outputs a matrix that contains the entire translated image. If you select `Same as input image`, the block outputs a matrix that is the same size as the

input image and contains a portion of the translated image. Use the **Background fill value** parameter to specify the pixel values outside the image.

Use the **Translation values source** parameter to specify how to enter your displacement vector. If you select `Specify via dialog`, the **Offset** parameter appears in the dialog box. Use it to enter your displacement vector, a two-element vector, `[r c]`, of real, integer values that represent the number of pixels by which you want to translate your image. The `r` value represents how many pixels up or down to shift your image. The `c` value represents how many pixels left or right to shift your image. The axis origin is the top-left corner of your image. For example, if you enter `[2.5 3.2]`, the block moves the image 2.5 pixels downward and 3.2 pixels to the right of its original location. When the displacement vector contains fractional values, the block uses interpolation to compute the output.

Use the **Interpolation method** parameter to specify which interpolation method the block uses to translate the image. If you translate your image in either the horizontal or vertical direction and you select `Nearest neighbor`, the block uses the value of the nearest pixel for the new pixel value. If you translate your image in either the horizontal or vertical direction and you select `Bilinear`, the new pixel value is the weighted average of the two nearest pixel values. If you translate your image in either the horizontal or vertical direction and you select `Bicubic`, the new pixel value is the weighted average of the four nearest pixel values.

The number of pixels the block considers affects the complexity of the computation. Therefore, the nearest-neighbor interpolation is the most computationally efficient. However, because the accuracy of the method is roughly proportional to the number of pixels considered, the bicubic method is the most accurate. For more information, see “Interpolation Overview” on page 5-2.

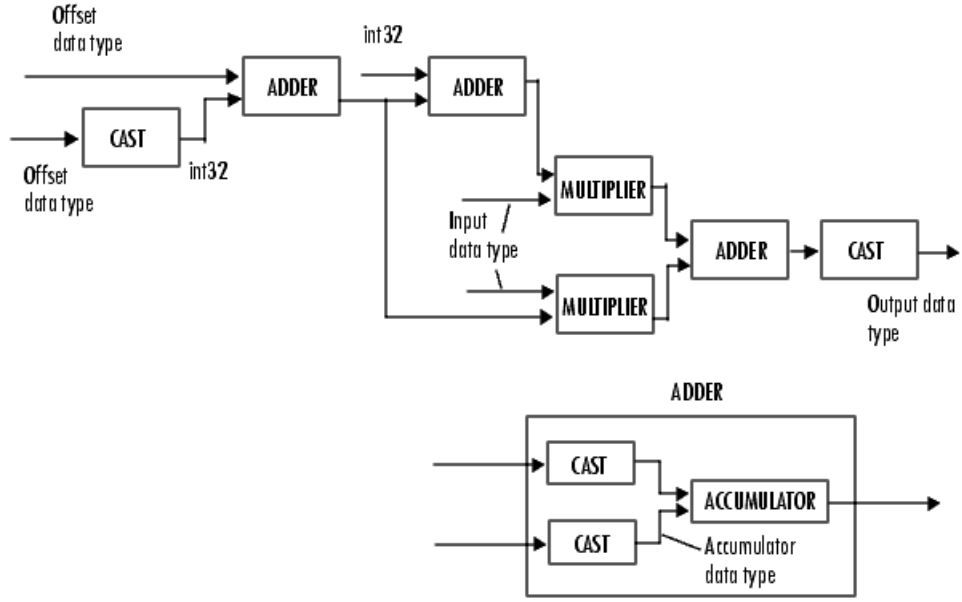
If, for the **Output size after translation** parameter, you select `Full`, and for the **Translation values source** parameter, you select `Input port`, the **Maximum offset** parameter appears in the dialog box. Use the **Maximum offset** parameter to enter a two-element vector of real,

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

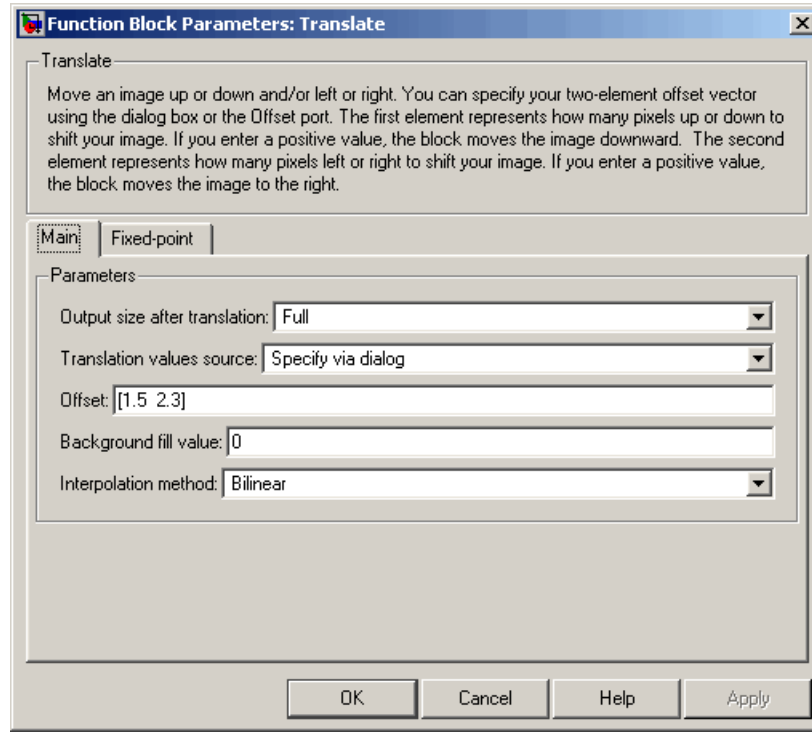
The following diagram shows the data types used in the Translate block for bilinear interpolation of fixed-point signals.



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.



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.

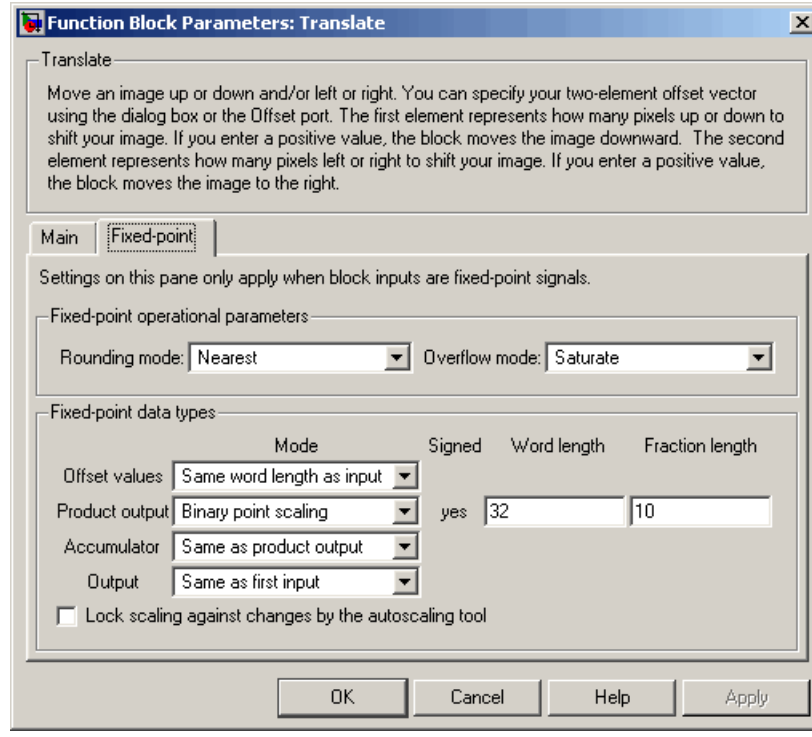
Offset
Enter a vector of real, scalar values that represent the number of pixels by which to translate your image.

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

Maximum offset
Enter a vector of real, scalar values that represent the maximum number of pixels by which you want to translate your image. This parameter must have the same data type as the input to the Offset port. This parameter is visible if, for the **Output size after translation** parameter, you select `Full` and, for the **Translation values source** parameter, you select `Input port`.

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



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

Offset values

Choose how to specify the word length and the fraction length of the offset values.

- When you select `Same word length as input`, the word length of the offset values match that of the input to the block. In this mode, the fraction length of the offset values is automatically set to the binary-point only scaling that provides you with the

best precision possible given the value and word length of the offset values.

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

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

Product output

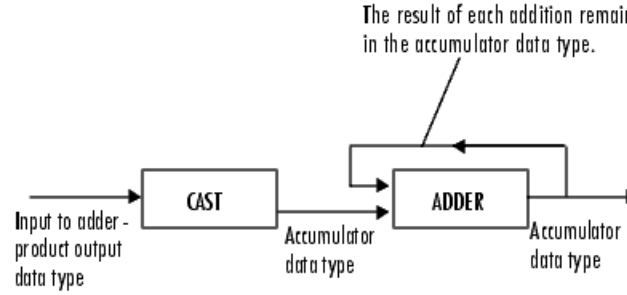


As depicted in the previous figure, the output of the multiplier is placed into the product output data type and scaling. Use this parameter to specify how to designate this product output word and fraction lengths.

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

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

Accumulator



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

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

Output

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

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

References

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

See Also

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

Purpose

Specify subset of rows or columns from input

Library

Utilities

Description

The Variable Selector block is a Signal Processing Blockset block. For more information, see the Variable Selector block reference page in the Signal Processing Blockset documentation.

Video From Workspace

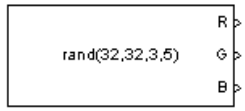
Purpose

Import video signal from MATLAB workspace

Library

Sources

Description



Video From Workspace

The Video From Workspace block imports a video signal from the MATLAB workspace. If the video signal is a M-by-N-by-T workspace array, the block outputs an intensity video signal, where M and N are the number of rows and columns in a single video frame, and T is the number of frames in the video signal. If the video signal is a M-by-N-by-C-by-T workspace array, the block outputs a color video signal, where M and N are the number of rows and columns in a single video frame, C is the number of color channels, and T is the number of frames in the video stream. In addition to the video signals previously described, this block supports fi objects and variables that are in the structure format returned by the MATLAB `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, we recommend converting your video data to a format supported by the From Multimedia File block or the Read Binary File block. For more information about these blocks, see the From Multimedia File andRead Binary File block reference pages.

Port	Output	Supported Data Types	Complex Values Supported
I	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	No
R, G, B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B ports have the same dimensions.	Same as I port	No

For Video and Image Processing Blockset blocks to display video data properly, double- and single-precision floating-point pixel values must be from 0 to 1. This block does not scale pixel values.

Use the **Signal** parameter to specify the MATLAB workspace variable from which to read. For example, to read an AVI file, use the following syntax:

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

The `aviread` function reads the AVI file into the MATLAB movie structure `mov`. Note that 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.

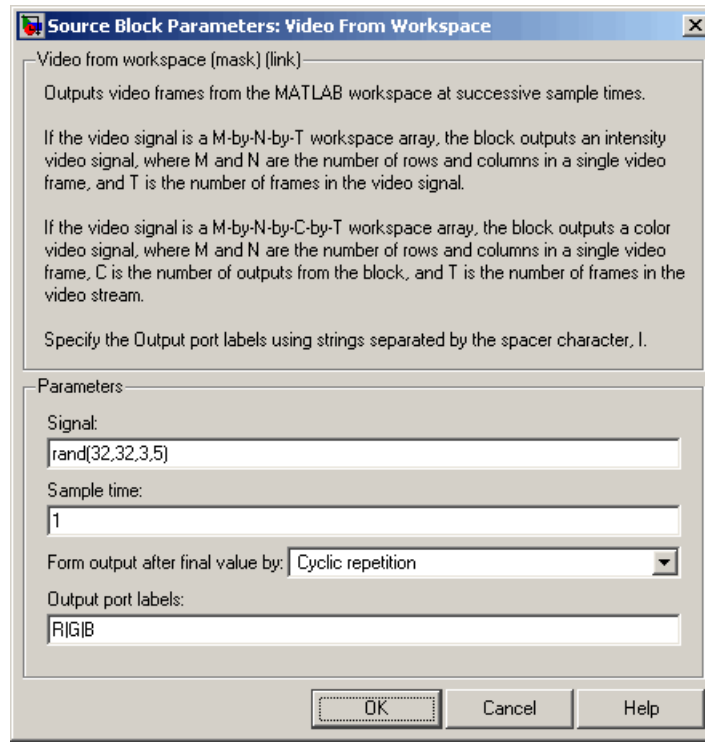
When the block has output all of the available signal samples, it can start again at the beginning of the signal, repeat the final value, or generate 0s until the end of the simulation. The **Form output after final value by** parameter controls this behavior:

- 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 **Output port labels** parameter to label your output ports. Use the spacer character, `|`, as the delimiter.

Dialog Box

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



Signal

Specify the MATLAB workspace variable that contains the video signal, or use the `aviread` function to specify an AVI filename.

Sample time

Enter the sample period of the output.

Form output after final value by

Specify the output of the block after all of the specified signal samples have been generated. The block can output zeros for the duration of the simulation (`Setting to zero`), repeat the final

value (`Holding Final Value`) or repeat the entire signal from the beginning (`Cyclic Repetition`).

Output port labels
Enter the labels for your output ports using the spacer character, `|`, as the delimiter.

See Also

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

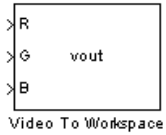
Purpose

Export video signal to MATLAB workspace

Library

Sinks

Description



The Video To Workspace block exports a video signal to the MATLAB workspace. If the video signal is represented by intensity values, it appears in the workspace as a three-dimensional M-by-N-by-T array, where M and N are the number of rows and columns in a single video frame, and T is the number of frames in the video signal. If it is a color video signal, it appears in the workspace as a four-dimensional M-by-N-by-C-by-T array, where M and N are the number of rows and columns in a single video frame, C is the number of inputs to the block, and T is the number of frames in the video stream. During code generation, Real-Time Workshop does not generate code for this block.

Port	Input	Supported Data Types	Complex Values Supported
I	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Fixed point • Boolean • 8-, 16-, 32-bit signed integers • 8-, 16-, 32-bit unsigned integers	No
R, G, B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Outputs from the R, G, or B ports have the same dimensions.	Same as I port	No

Use the **Variable name** parameter to specify the MATLAB workspace variable to which to write the video signal.

Use the **Number of inputs** parameter to determine the number of inputs to the block. If the video signal is represented by intensity values, enter `1`. If it is a color (R, G, B) video signal, enter `3`.

Use the **Limit video frames to last** parameter to determine the number of video frames, T, you want to export to the MATLAB workspace.

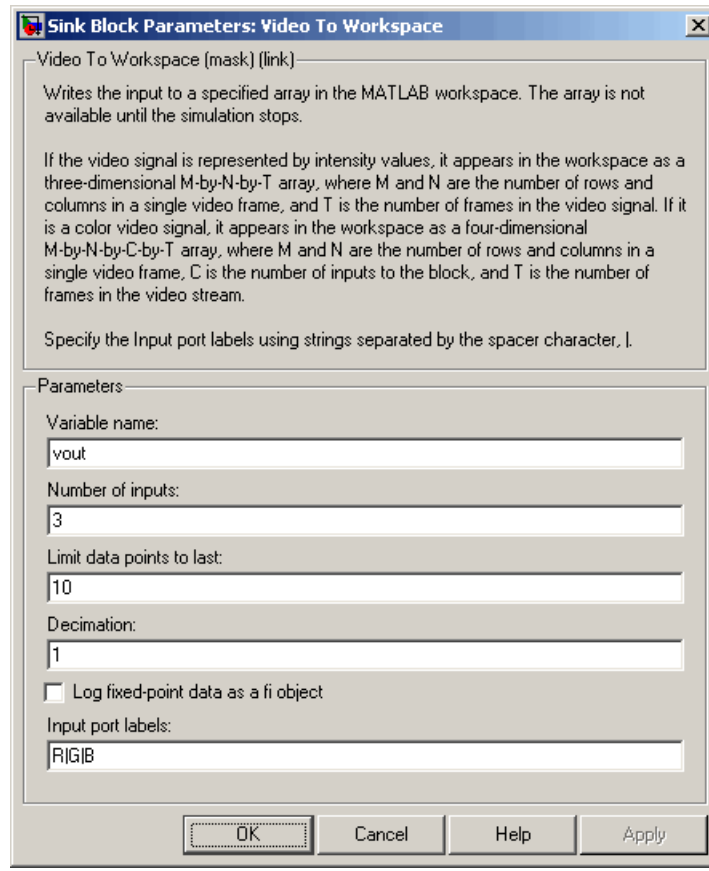
If you want to downsample your video signal, use the **Decimation** parameter to enter your decimation factor.

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.

Dialog Box

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



Variable name

Specify the MATLAB workspace variable to which to write the video signal.

Number of inputs
Enter the number of inputs to the block. If the video signal is black and white, enter `1`. If it is a color (R, G, B) video signal, enter `3`.

Limit data points to last
Enter the number of video frames to export to the MATLAB workspace.

Decimation
Enter your decimation factor.

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.

See Also

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

Purpose

Display binary, intensity, or RGB images or video streams

Library

Sinks

Description



Video Viewer

The Video Viewer block enables you to view a binary, intensity, or RGB image or a video stream. During code generation, Real-Time Workshop does not generate code for this block.

Port	Output	Supported Data Types	Complex Values Supported
I	Scalar, vector, or matrix of intensity values	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16-, and 32-bit signed integers • 8-, 16-, and 32-bit unsigned integers	No
R, G, B	Scalar, vector, or matrix that represents one plane of the RGB video stream. Inputs to the R, G, or B ports must have the same dimensions and data type.	Same as I port	No

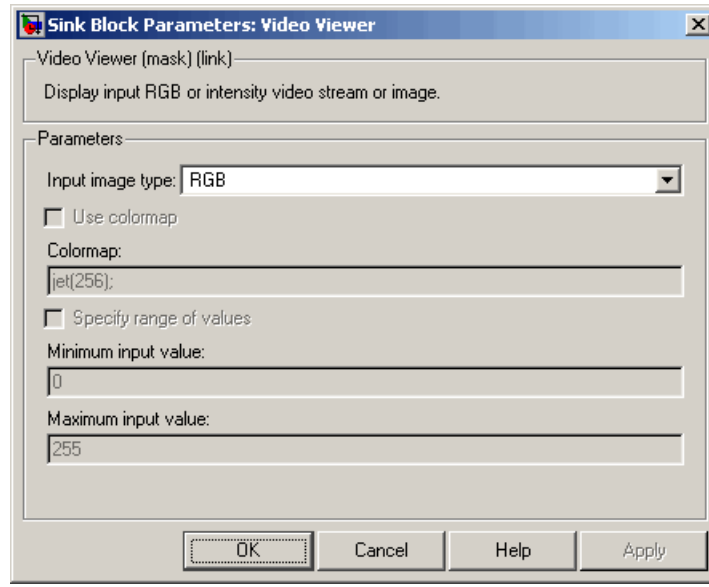
Use the **Input image type** parameter to specify the type of image or video stream to view. If you select `RGB`, the R, G, and B ports appear on the block. Use this option for RGB images and video. If you select `Intensity`, the I port appears on the block. Use this option for binary and intensity images and video.

If, for the **Input image type** parameter, you select `Intensity`, you can set additional block parameters. If you select the **Use colormap** check box, you can use the **Colormap** parameter to specify a colormap or a call to a colormap-generating function. MATLAB provides a number of functions for generating predefined colormaps, such as `hot`, `cool`, `bone`, and `autumn`. Each of these functions accepts the colormap size as an argument, and can be used in this parameter. For example, if you enter `gray(128)`, the image is displayed in 128 shades of gray. The color in the first row of the colormap matrix represents the minimum input value and the color in the last row represents the maximum input value. Values between the minimum and maximum are quantized and mapped to the intermediate rows of the colormap matrix. For more information, see the MATLAB `colormap` function documentation.

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

Dialog Box

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



Input image type

Specify the type of image or video stream to view. Your choices are `RGB` or `Intensity`.

Use colormap

Select this check box to specify a colormap. This parameter is available if, for the **Input image type** parameter, you select `Intensity`.

Colormap

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

Specify range of values
Select this check box if you want to define the range for the input. This parameter is available if, for the **Input image type** parameter, you select `Intensity`.

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

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

See Also

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

Purpose

Write video frames to uncompressed AVI file

Library

Sinks

Description



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.

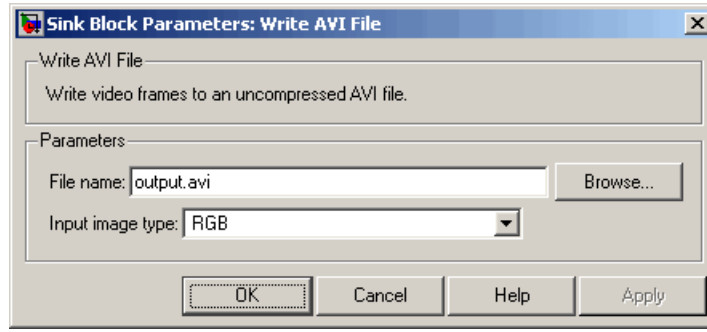
Port	Input	Supported Data Types	Complex Values Supported
I	Matrix of intensity values	• Double-precision floating point • Single-precision floating point • Boolean • 8-, 16- 32-bit signed integers • 8-, 16- 32-bit unsigned integers	No
R, G, B	Matrix that represents one plane of the RGB video stream. 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 **Input image type** parameter to specify the type of video input into the block. If the input contains RGB video frames, select `RGB`. If the input contains intensity video frames, select `Intensity`.

Dialog Box

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



File name

Specify the name of the AVI file to which to write.

Input image types

Specify the type of video input into the block. If the input contains RGB video frames, select `RGB`. If the input contains intensity video frames, select `Intensity`.

See Also

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

Purpose

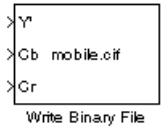
Write binary video data to files

Library

Sinks

Description

The Write Binary File block takes video data from a Simulink model and exports it to a binary file.



Port	Input	Supported Data Types	Complex Values Supported
Input	Matrix that represents the luma (Y') and chroma (Cb and Cr) components of a video stream	• 8-, 16- 32-bit signed integers • 8-, 16- 32-bit unsigned integers	No

Use the **File name** parameter to specify the name of the binary file. If the location of this file is on your MATLAB path, enter the filename. If the location of this file is not on your MATLAB path, use the **Browse** button to specify the full path to the file as well as the filename.

Use the **Video format** parameter to specify the format of the binary video data. Your choices are `Four character codes` or `Custom`. See "Four Character Code Video Formats" on page 10-556 or "Custom Video Formats" on page 10-556 for more details.

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

Four Character Code Video Formats

Four Character Codes (FOURCC) are used to identify video formats. For more information about these codes, see `http://www.fourcc.org`.

Use the **Four character code** parameter to identify the video format.

Custom Video Formats

You can use the Write Binary File block to create a binary file that contains video data in a custom format.

Use the **Bit stream format** parameter to specify whether you want your data in planar or packed format.

Use the **Number of input components** parameter to specify the number of components in the video stream. This number corresponds to the number of block input ports.

Select the **Inherit size of components from input data type** check box if you want each component to have the same number of bits as the input data type. If you clear this check box, you must specify the number of bits for each component.

Use the **Component** parameters to specify the component names.

Use the **Component order in binary file** parameter to specify how to arrange the components in the binary file.

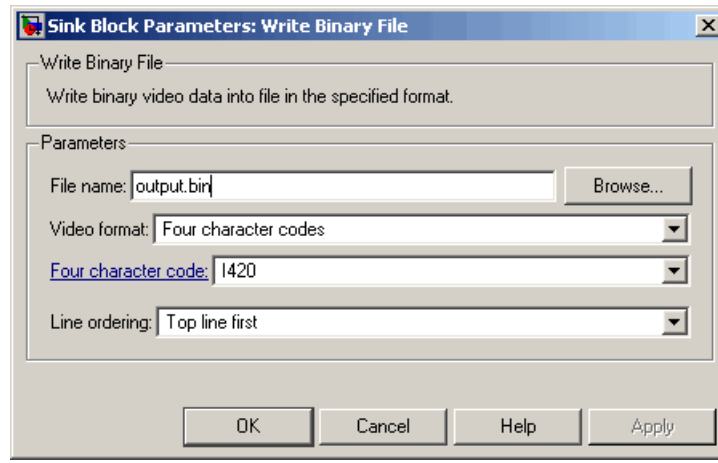
Select the **Interlaced video** check box if the video stream represents interlaced video data.

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

Use the **Byte order in binary file** parameter to specify whether the byte ordering in the output binary file is little endian or big endian.

Dialog Box

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



File name

Specify the name of the binary file.

Video format

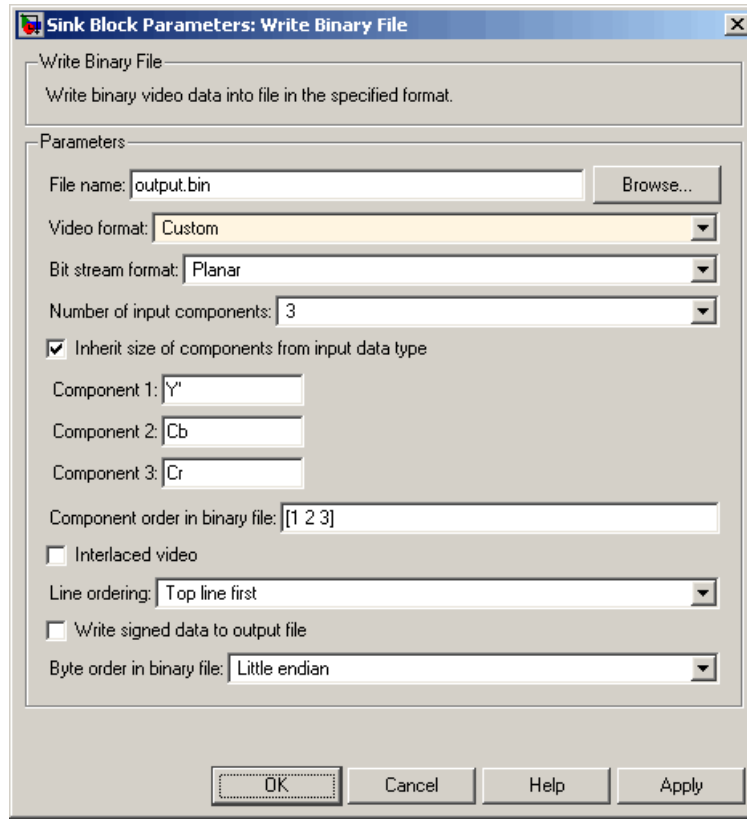
Specify the format of the binary video data. Your choices are `Four character codes` or `Custom`.

Four character code

From the list, select the binary file format.

Line ordering

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



Bit stream format

Specify whether you want your data in planar or packed format.

Number of input components

Specify the number of components in the video stream. This number corresponds to the number of block input ports

Inherit size of components from input data type

Select this check box if you want each component to have the same number of bits as the input data type. If you clear this check box, you must specify the number of bits for each component.

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

To Multimedia File	Video and Image Processing Blockset
Write AVI File	Video and Image Processing Blockset

11

Functions — Alphabetical List

isfilterseparable

Purpose

Determine whether filter coefficients are separable

Syntax

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

Description

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

See Also

2-D FIR Filter Video and Image Processing Blockset

Purpose

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

Library

Sinks

Description

Use the MPlay GUI to view video from files or the MATLAB workspace. You can also use it to view video signals in Simulink models. If the video contains audio, the GUI ignores it and plays only the video frames.

This reference page contains the following sections:

- "MPlay Preferences Dialog Box" on page 11-7
- "Video Information Dialog Box" on page 11-9
- "Colormap Dialog Box" on page 11-9
- "Frame Rate Dialog Box" on page 11-10
- "Saving the Settings of Multiple MPlay GUIs" on page 11-11
- "Command Line Syntax" on page 11-11

MPlay GUI Buttons

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

Button	Menu Equivalent	Shortcut Keys and Accelerators	Description
	**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.
	**Tools > Video Information**	**I**	View information about the video data source.
	**Tools > Pixel Region**	N/A	Open the Pixel Region tool. For more information about this tool, see the Image Processing Toolbox documentation.
	**Tools > Zoom In**	N/A	Zoom in on the video display.
	**Tools > Zoom Out**	N/A	Zoom out of the video display.
	**Tools > Pan**	N/A	Move the image displayed in the GUI.
	**Tools > Maintain Fit to Window**	N/A	Scale video to fit GUI size automatically. Toggle the button on or off.
	N/A	N/A	Enlarge or shrink the video. This option is available if the **Maintain Fit to Window** button is not selected.
For workspace and file sources:			

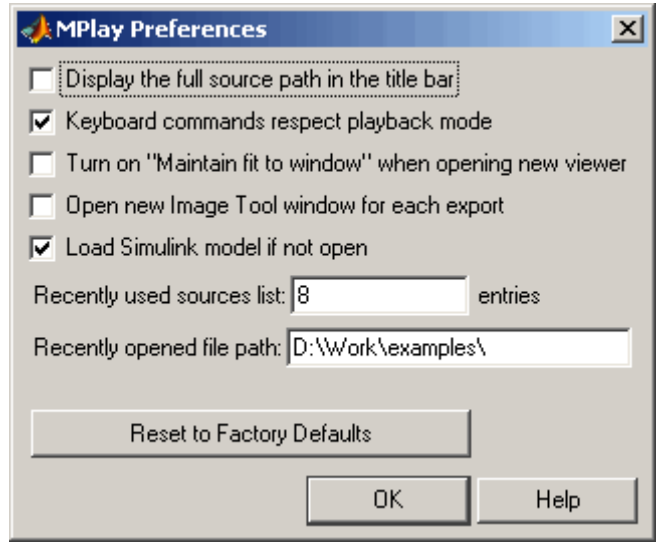
Button	Menu Equivalent	Shortcut Keys and Accelerators	Description
	**Playback > Go to First**	**F**, **Home**	Go to the first frame of the video.
	**Playback > Rewind**	Up arrow	Jump back 10 frames.
	**Playback > Step Back**	Left arrow, **Page Up**	Step back one frame.
	**Playback > Stop**	**S**	Stop the video.
	**Playback > Play**	**P**, Space bar	Play the video.
	**Playback > Pause**	**P**, Space bar	Pause the video. This button is visible only when the video is playing.
	**Playback > Step Forward**	Right arrow, **Page Down**	Step forward one frame.
	**Playback > Fast Forward**	Down arrow	Jump forward 10 frames.
	**Playback > Go to Last**	**L**, **End**	Go to the last frame of the video.
	**Playback > Jump to**	**J**	Jump to a specific frame.
	**Playback > Playback Modes > Repeat**	**R**	Repeated video playback.
	**Playback > Playback Modes > Forward play**	**A**	Play the video forward.

Button	Menu Equivalent	Shortcut Keys and Accelerators	Description
	**Playback > Playback Modes > Backwardplay**	**A**	Play the video backward.
	**Playback > Playback Modes > AutoReverse play**	**A**	Play the video forward and backward.
For Simulink sources:			
	**Playback > Stop**	**S**	Stop the video. This button also controls the Simulink model.
	**Playback > Start**	**P**, Space bar	Play the video. This button also controls the Simulink model.
	**Playback > Pause**	**P**, Space bar	Pause the video. This button also controls the Simulink model and is visible only when the video is playing.
	**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.

Button	Menu Equivalent	Shortcut Keys and Accelerators	Description
	**Playback > Floating Signal Connection (not selected)**	N/A	Indicates persistent Simulink connection. In this mode, the MPlay GUI is always associated with the Simulink signal you selected before you clicked the **Connect to Simulink Signal** button.
	**Playback > Floating Signal Connection (selected)**	N/A	Indicates floating Simulink connection. In this mode, you can click different signals in the model, and the MPlay GUI displays them. Only one MPlay GUI can be in floating-scope mode at one time.

MPlay Preferences Dialog Box

The MPlay Preferences dialog box enables you to change the behavior and appearance of the GUI and the behavior of the playback shortcut keys. To open this dialog box, click **File** > **Preferences** or press **N**.



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

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

If you select the **Turn on “Maintain fit to window” when opening new viewer** check box, each new instance of the MPlay GUI automatically changes size according to the size of the video frames it is displaying.

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.

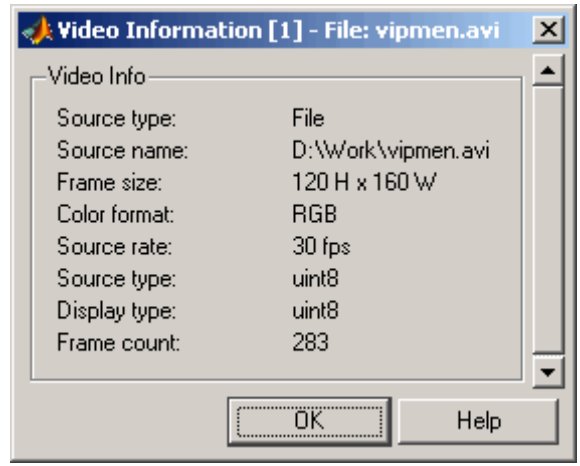
Select the **Load Simulink model if not open** check box if you want the Simulink model associated with an MPlay GUI to open when you open the GUI.

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

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

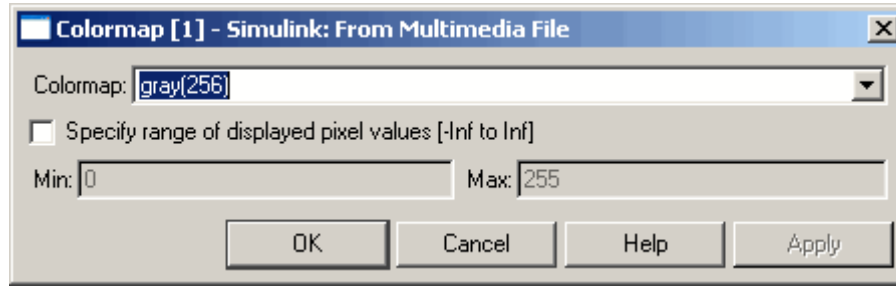
Video Information Dialog Box

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



Colormap Dialog Box

The Colormap dialog box lets you change the colormap of an intensity video. The parameters on this dialog box are not available when the GUI is displaying RGB video data. To open this dialog box, click **Tools > Colormap** or press **C**.

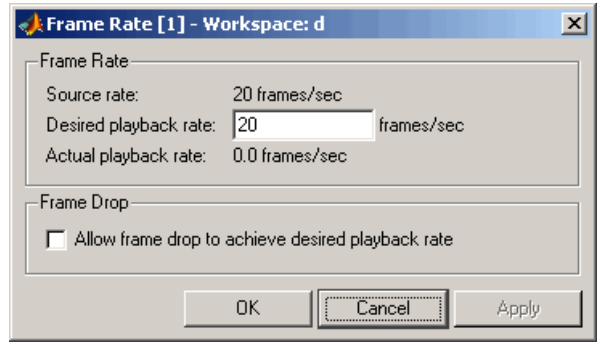


Use the **Colormap** parameter to specify the colormap to apply to the intensity video.

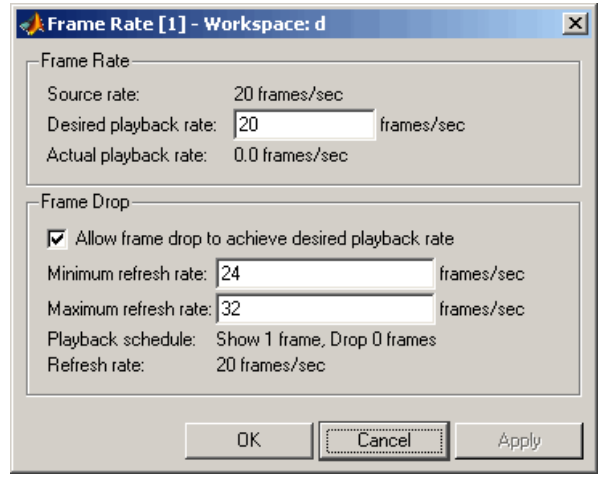
If you know that the pixel values do not use the entire data type range, you can select the **Specify range of displayed pixel values** check box and enter the range for your data. The dialog box automatically displays the range based on the data type of the pixel values.

Frame Rate Dialog Box

The Frame Rate dialog box displays the frame rate of the source, lets you change the rate at which the MPlay GUI plays the video, and displays the actual playback rate. You can use the **Desired playback rate** parameter to increase the playback rate. This dialog box is not available if you are using the MPlay GUI to view a video signal in a Simulink model. To open this dialog box, click **Playback** > **Frame Rate** or press **T**.



If you want to improve the actual playback rate, select the **Allow frame drop to achieve desired playback rate** check box.



Experiment with the **Minimum refresh rate** and **Maximum refresh rate** parameters to achieve your desired frame rate.

Saving the Settings of Multiple MPlay GUIs

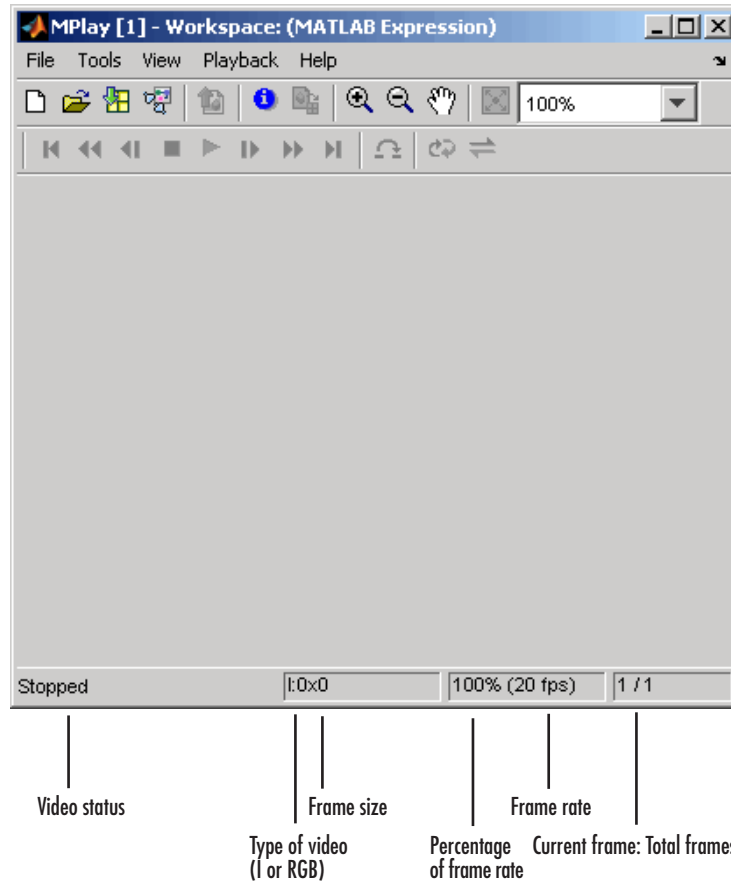
The MPlay GUI enables you to save and load the settings of multiple GUI instances. That way, you only need to configure the MPlay GUIs that are associated with your model once. To save the GUI settings, click **File > Instrumentation Sets > Save Set**. To open the preconfigured MPlay GUIs, click **File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

Command Line Syntax

Syntax	Description
`mplay`	Open an MPlay GUI.
`mplay('filename.avi')`	Connect the MPlay GUI to the specified AVI file.

Syntax	Description
`mplay(a)`	Connect the MPlay GUI to the variable in the MATLAB workspace, `a`.
`mplay(a,fps)`	Connect 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 GUI



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

For more information about the MPlay GUI, see “Viewing Videos from the MATLAB Workspace” on page 3-2, “Viewing Video Files” on page

3-6, and “Viewing Video Signals in Simulink” on page 3-8. You can also see the “MPlay Simulink Tutorial” in the “Video Playback” section of the Video and Image Processing Blockset demos.

See Also

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

Index

2-D Autocorrelation block 10-2
2-D Convolution block 10-8
2-D Correlation block 10-19
2-D DCT block 10-31
2-D FFT block 10-40
2-D FIR Filter block 10-54
2-D Histogram block 10-66
2-D IDCT block 10-76
2-D IFFT block 10-85
2-D Maximum block 10-98
2-D Mean block 10-109
2-D Median block 10-122
2-D Minimum block 10-128
2-D Pad 10-139
2-D Standard Deviation block 10-149
2-D Variance block 10-161

A

`Accelerator` mode 1-18
adding periodic noise to a signal 7-39
adjusting
    intensity image contrast 7-46
    RGB image contrast 7-52
Adobe Acrobat Reader 1-8
algorithms
    bicubic interpolation 5-4
    bilinear interpolation 5-3
    nearest neighbor interpolation 5-2
angles
    rotation 5-6
annotating
    AVI files 2-9
arrays
    interpretation of 1-11
artifacts
    in an image 7-39
audio
    exporting to multimedia file 2-20
Autothreshold block 10-177
    to perform thresholding 4-7
AVI files
    annotating 2-9
    exporting 2-5
    importing 2-2
    viewing 2-2

B

background
    estimation 6-11
    pixels 7-2
    user's expected 1-8
bicubic interpolation 5-4
bilinear interpolation 5-3
binary
    conversion from intensity 4-2
    images 1-12
Blob Analysis block 10-187
Block Matching block 10-203
Block Processing block 10-214
blurring images 7-26
Boolean matrices 1-12
Bottom-hat block 10-223
boundaries
    of objects 7-2
boundary artifacts 7-39
brightening images 6-11

C

capabilities of
    Video and Image Processing Blockset 1-2
CD installation 1-3
changing
    image size 5-14
    intensity image contrast 7-46
    RGB image contrast 7-52
chapter descriptions 1-9
chroma components

of images 4-19
chroma resampling 4-19
Chroma Resampling block 10-226
chrominance resampling 4-19
Closing block 10-231
codecs
supported by Microsoft Windows Media Player 2-14
color
definition of 1-12
color space conversion 4-14
Color Space Conversion block 10-234
colormaps 1-12
column-major format 1-15
compensation
for motion 8-9
Compositing block 10-245
compression
of images 8-11
of video 8-9
concepts
description of 1-11
Configuration dialog box 1-16
continuous rotation 5-6
contrast
increasing 2-17
Contrast Adjustment block 10-255
controlling video duration 1-17
conventions
column-major format 1-15
conversion
color space 4-14
intensity to binary 4-2
R'G'B' to intensity 4-14
correction
of uneven lighting 6-11
correlation
used in object tracking 8-2
counting objects 6-3
cropping
images 5-21

D

data types 1-12
definition of
intensity and color 1-12
Deinterlacing block 10-263
demos
in the Help browser 1-5
on MATLAB Central 1-6
on the Web 1-6
Periodic noise reduction 7-39
Video compression 8-9
Video stabilization 8-9
Demosaic block 10-274
dependencies
on Windows dynamic libraries 1-19
detection of
edges 7-2
lines 7-9
dilation 6-2
Dilation block 10-281
DirectX 2-14
dlls
dependencies on 1-19
documentation
on the Web 1-7
on your system 1-7
PDF 1-8
printing 1-8
viewing 1-7
downsampling
chroma components 4-19
Draw Markers block 10-285
Draw Shapes block 10-297
dynamic range 1-12

E

edge
  pixels 7-2
  thinning 7-2
edge detection 7-2
Edge Detection block 10-307
electrical interference 7-39
erosion 6-2
Erosion block 10-320
estimation
  of image background 6-11
executables
  running 1-19
exporting
  AVI files 2-5
  multimedia files 2-17

F

feature extraction
  finding angles between lines 7-17
  finding edges 7-2
  finding lines 7-9
filtering
  median 7-33
  operations 6-2
Find Local Maxima 10-324
finding
  angles between lines 7-17
  edges of objects 7-2
  histograms of images 7-58
  lines in images 7-9
form of objects 6-2
Frame Rate Display block 10-329
frequency distribution
  of elements in an image 7-58
From Multimedia File block 10-331
`fspecial` function 7-26
function
  isfilterseparable 11-2
  mplay 11-3

G

gamma correction 4-14
Gamma Correction block 10-332
Gaussian Pyramid block 10-337
geometric transformation 5-1
gradient components
  of images 7-2

H

Help browser
  demos 1-5
  documentation 1-7
Histogram Equalization block 10-347
histograms
  of images 7-58
Hough Lines block 10-351
Hough Transform block 10-361

I

Image Complement block 10-369
image compression 8-11
image credits 1-26
image data
  storage of 1-15
Image Data Type Conversion block 10-371
Image From File block 10-375
Image From Workspace block 10-381
image rotation 5-6
image types 1-11
images
  binary 1-12
  boundary artifacts 7-39
  brightening 6-11
  correcting for uneven lighting 6-11
  counting objects in 6-3
  cropping 5-21

filtering of 6-2
finding angles between lines 7-17
finding edges in 7-2
finding histograms of 7-58
finding lines in 7-9
gradient components 7-2
intensity 1-12
intensity to binary conversion 4-2
labeling objects in 6-3
lightening 6-11
noisy 7-33
periodic noise removal 7-39
removing salt and pepper noise 7-33
resizing of 5-14
RGB 1-12
rotation of 5-6
segmentation of 6-2
sharpening and blurring 7-26
true-color 1-12
types of 1-11
importing
AVI files 2-2
multimedia files 2-14
improvement
of performance 1-18
increasing video contrast 2-17
Insert Text block 10-386
installation
CD 1-3
Video and Image Processing Blockset 1-3
Web download 1-3
intensity
conversion from R'G'B' 4-14
conversion to binary 4-2
definition of 1-12
images 1-12
intensity images
adjusting the contrast of 7-46
interference
electrical 7-39
interpolation
bicubic 5-4
bilinear 5-3
examples 5-2
nearest neighbor 5-2
overview 5-2
interpretation of
matrices 1-11
irregular illumination 6-11
isfilterseparable function 11-2

K

key blockset concepts 1-11
knowledge
user's expected 1-8

L

Label block 10-397
labeling objects 6-3
Laplacian pyramid 10-337
lightening images 6-11
location of
lines 7-9
object edges 7-2
objects in an image 8-2
luma components
applying highpass filter 7-26
applying lowpass filter 7-26
of images 4-19
luminance 4-19

M

matching
patterns in an image 8-2
MATLAB Central
demos 1-6
matrices
interpretation of 1-11

Maximum block 10-401
measurement operations 6-2
Median Filter block 10-402
median filtering 7-33
methods
    interpolation 5-2
    sum of absolute differences (SAD) 8-9
    thresholding 6-11
Microsoft Windows Media Player 2-14
Minimum block 10-411
modes
    `Normal` and `Accelerator` 1-18
morphology 6-1
    opening 6-3
    overview 6-2
    STREL object 6-3
motion compensation 8-9
motion detection 8-9
MPlay GUI 11-3
multimedia files
    exporting 2-17
    exporting audio and video 2-20
    importing 2-14
    viewing 2-14

N

nearest neighbor interpolation 5-2
noise
    adding to a signal 7-39
noise removal
    periodic 7-39
    salt and pepper 7-33
nonuniform illumination
    correcting for 6-11
`Normal` mode 1-18

O

object boundaries 7-2
object extraction 6-2
object tracking
    using correlation 8-2
objects
    delineating 6-11
    location of 8-2
opening 6-3
Opening block 10-412
operations
    morphological 6-1
    thresholding 4-2
Optical Flow block 10-415
organization of the chapters 1-9
overview of
    documentation 1-9
    interpolation 5-2
    morphology 6-2
    Video and Image Processing Blockset 1-2

P

padding 7-39
pattern matching 8-2
performance
    improving 1-18
periodic noise
    removal 7-39
printing
    PDF documentation 1-8
product demos 1-5
products
    related 1-4
    required 1-4
Projective Transformation block 10-423
PSNR block 10-458

R

R'B'G'
    conversion to intensity 4-14

Read AVI File block 10-461
Read Binary File block 10-465
reception
    of an RGB image 4-19
reconstruction
    of images 8-11
reduction
    of image size 5-14
region of interest
    cropping to 5-21
    visualizing 8-2
related products 1-4
relational operators
    to perform thresholding 4-2
removal of
    periodic noise 7-39
    salt and pepper noise 7-33
required products 1-4
resampling
    chroma 4-19
Resize block 10-471
resizing
    images 5-14
RGB images 1-12
    adjusting the contrast of 7-52
Rotate block 10-482
rotation
    continual 5-6
    of an image 5-6

S

SAD block 10-494
salt and pepper noise removal 7-33
sample time 1-16
scaling 1-12
    data types 7-2
segmentation operations 6-2
setting
    configuration parameters 1-16
    simulation time 1-17
shape of objects 6-2
sharpening images 7-26
Shear block 10-504
shrinking
    image size 5-14
simulation time 1-17
Simulink Solver 1-16
Sobel kernel 7-2
stabilization
    of video 8-9
storage of image data 1-15
STREL object 6-3
sum of absolute differences (SAD) method 8-9
summary of morphology 6-2

T

techniques
    motion compensation 8-9
    sum of absolute differences (SAD) 8-9
    thresholding 6-11
thresholding operation 4-2
    with uneven lighting 4-7
thresholding techniques 6-11
To Video Display block 10-515
Top-hat block 10-520
Trace Boundaries block 10-524
tracking
    of an object 8-2
transformation
    geometric 5-1
Translate block 10-530
transmission
    of an RGB image 4-19
trimming
    images 5-21
true size 2-2
true-color images 1-12
tutorials 1-9

types of images 1-11

U

uneven lighting
    correcting for 6-11

V

Variable Selector 10-539
vectors
    motion 8-9
video
    adjusting display size 2-2
    annotating AVI files 2-9
    duration 1-17
    exporting from AVI file 2-5
    exporting from multimedia file 2-17
    importing from AVI file 2-2
    importing from multimedia file 2-14
    increasing the contrast of 2-17
    interpretation of 1-12
    speed of 2-14
    stabilization 8-9
video compression and stabilization 8-9
Video From Workspace block 10-540
Video To Workspace block 10-545
Video Viewer block 10-549
viewing
    AVI files 2-2
    compressed images 8-18
    demos 1-5
    documentation 1-7
    multimedia files 2-14
vip_rt.dll 1-19

W

Web
    demos 1-6
    documentation 1-7
    download 1-3
Windows dynamic libraries
    dependencies on 1-19
Windows platforms 2-14
Write AVI File block 10-553
Write Binary File 10-555