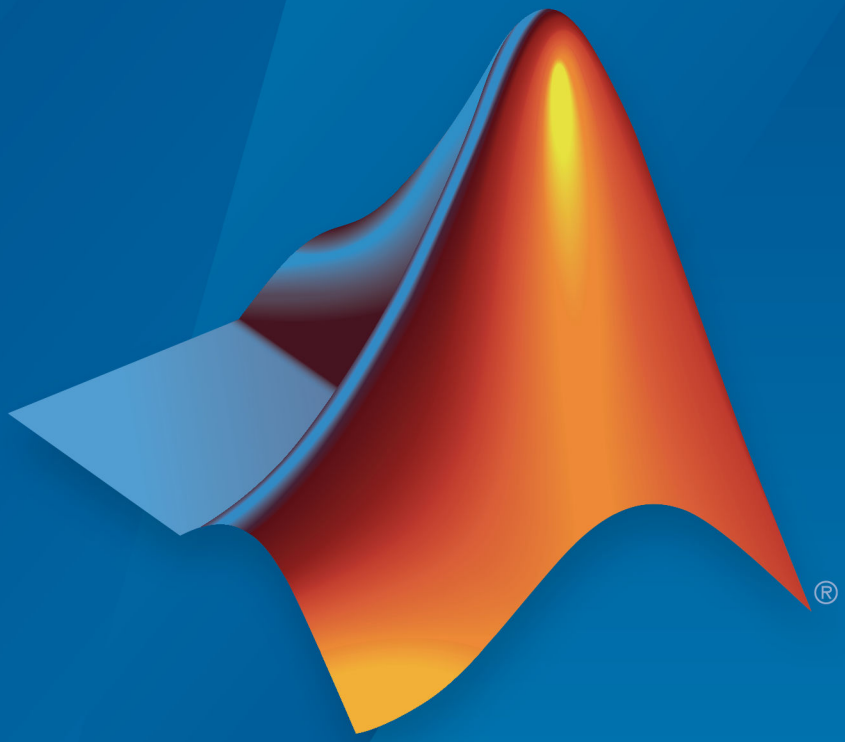
Computer Vision System Toolbox™

Getting Started Guide



MATLAB® & SIMULINK®

R2017b

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

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

Computer Vision System Toolbox™ Getting Started Guide

© COPYRIGHT 2000–2017 by The MathWorks, Inc.

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

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

Trademarks

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

Patents

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

Revision History

April 2011	Online only	Revised for Version 4.0 (Release 2011a)
September 2011	Online only	Revised for Version 4.1 (Release 2011b)
March 2012	Online only	Revised for Version 5.0 (Release 2012a)
September 2012	Online only	Revised for Version 5.1 (R2012b)
March 2013	Online only	Revised for Version 5.2 (R2013a)
September 2013	Online only	Revised for Version 5.3 (R2013b)
March 2014	Online only	Revised for Version 6.0 (R2014a)
October 2014	Online only	Revised for Version 6.1 (R2014b)
March 2015	Online only	Revised for Version 6.2 (Release R2015a)
September 2015	Online only	Revised for Version 7.0 (Release R2015b)
March 2016	Online only	Revised for Version 7.1 (Release R2016a)
September 2016	Online only	Revised for Version 7.2 (Release R2016b)
March 2017	Online only	Revised for Version 7.3 (Release R2017a)
September 2017	Online only	Revised for Version 8.0 (Release R2017b)

Contents

1 Product Overview

Computer Vision System Toolbox Product Description **1-2**
Key Features **1-2**

2 Computer Vision Algorithms and Video Processing

Computer Vision Capabilities **2-2**

Video Processing in MATLAB **2-3**

Computer Vision System Toolbox Preferences **2-4**
Parallel Computing Toolbox Support **2-4**

3 Coordinate Systems

Coordinate Systems **3-2**
Pixel Indices **3-2**
Spatial Coordinates **3-2**
3-D Coordinate Systems **3-3**

4 Strategies for Real-Time Video Processing in Simulink

Optimizing Your Implementation 4-2

Developing Your Models 4-4

5 Data Type Support

Block Data Type Support 5-2

Fixed-Point Support for MATLAB System Objects 5-3
Getting Information About Fixed-Point System Objects 5-3
Setting System Object Fixed-Point Properties 5-4

6 Hardware Support

Computer Vision System Toolbox Supported Hardware 6-2

1

Product Overview

Computer Vision System Toolbox Product Description

Design and simulate computer vision and video processing systems

Computer Vision System Toolbox provides algorithms, functions, and apps for designing and simulating computer vision and video processing systems. You can perform feature detection, extraction, and matching, as well as object detection and tracking. For 3-D computer vision, the system toolbox supports single, stereo, and fisheye camera calibration; stereo vision; 3-D reconstruction; and 3-D point cloud processing.

Algorithms for deep learning and machine learning enable you to detect faces, pedestrians, and other common objects using pretrained detectors. You can train a custom detector using ground truth labeling with training frameworks such as Faster R-CNN and ACF. You can also classify image categories and perform semantic segmentation.

Algorithms are available as MATLAB® functions, System objects, and Simulink® blocks. For rapid prototyping and embedded system design, the system toolbox supports fixed-point arithmetic and C-code generation.

Key Features

- Deep learning for image classification, object detection, and semantic segmentation
- Machine learning using ACF, cascade object detection, and bag-of-features for object detection, object recognition, and image retrieval systems
- Object detection and tracking, including the Viola-Jones, Kanade-Lucas-Tomasi (KLT), and Kalman filtering methods
- Camera calibration and automation apps for single and stereo cameras, fisheye lens calibration, and automatic checkerboard detection
- Stereo vision, including rectification, disparity calculation, and 3-D reconstruction
- 3-D point cloud processing, including I/O, visualization, registration, denoising, and geometric shape fitting
- Support for C-code generation and fixed-point arithmetic (with MATLAB Coder™

Computer Vision Algorithms and Video Processing

- "Computer Vision Capabilities" on page 2-2
- "Video Processing in MATLAB" on page 2-3
- "Computer Vision System Toolbox Preferences" on page 2-4

Computer Vision Capabilities

Computer Vision System Toolbox provides algorithms and tools for the design and simulation of computer vision and video processing systems. The toolbox includes algorithms for feature extraction, motion detection, object detection, object tracking, stereo vision, video processing, and video analysis. Tools include video file I/O, video display, drawing graphics, and compositing. Capabilities are provided as MATLAB functions, MATLAB System objects, and Simulink blocks. For rapid prototyping and embedded system design, the system toolbox supports fixed-point arithmetic and C-code generation.

The link below provides an overview video of Computer Vision System Toolbox capabilities and applications:

Computer Vision System Toolbox capabilities

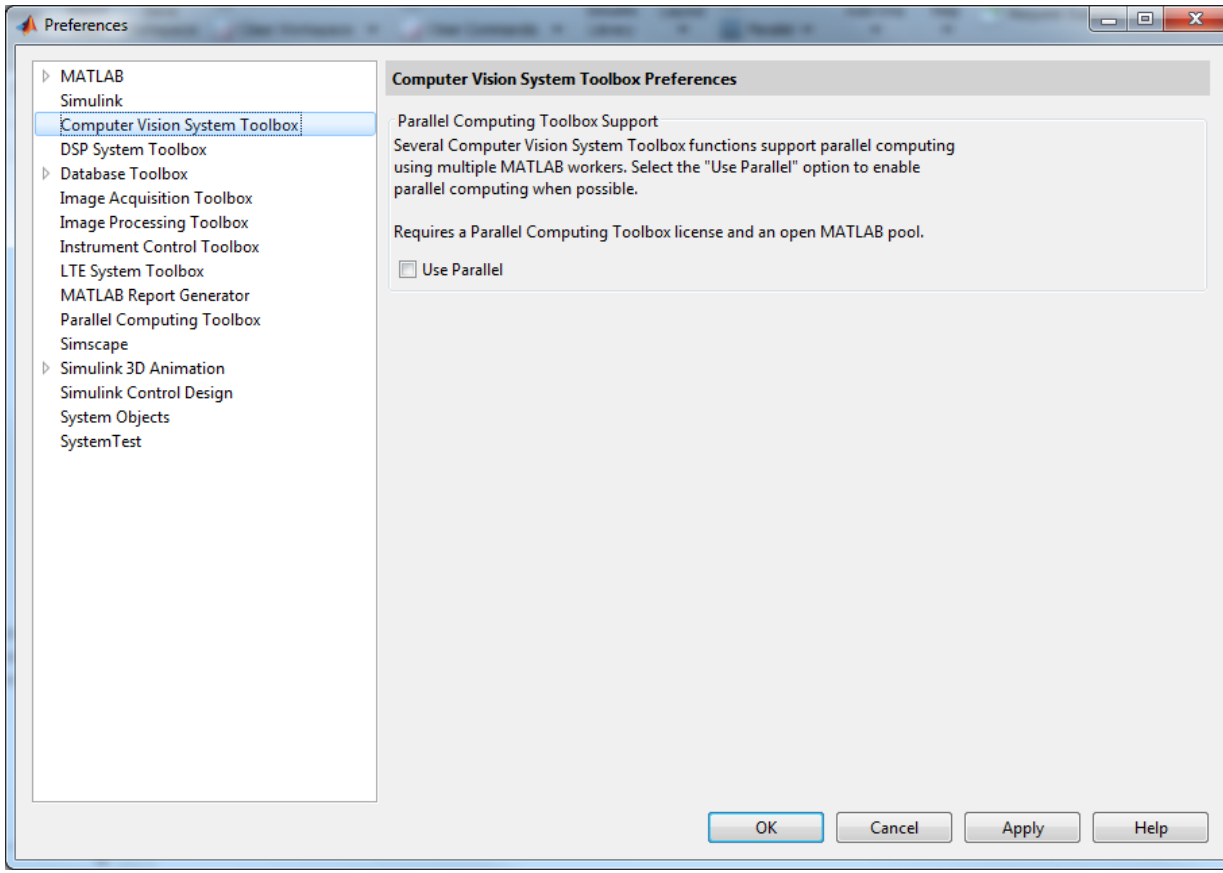
Video Processing in MATLAB

Computer Vision System Toolbox provides algorithms and tools for video processing workflows. You can read and write from common video formats, perform common video processing algorithms such as deinterlacing and chroma-resampling, and display results with text and graphics burnt in to the video. Video processing in MATLAB uses System objects, which avoids excessive memory use by streaming data to and from video files.

The link below provides an introduction video to a typical workflow for motion estimation:
Video processing in MATLAB

Computer Vision System Toolbox Preferences

To open Computer Vision System Toolbox preferences, on the **Home** tab, in the **Environment** section, click **Preferences**. Select **Computer Vision System Toolbox**.



Parallel Computing Toolbox Support

Several Computer Vision System Toolbox functions support parallel computing using multiple MATLAB workers. Select the **Use Parallel** check box to enable parallel computing when possible.

Parallel computing functionality requires a Parallel Computing Toolbox™ license and an open MATLAB pool.

The functions and methods listed below take an optional logical input parameter, `'UseParallel'` to control whether the individual function can use `parfor`. Set this logical to `'true'` to enable parallel processing for the function or method.

- `bagOfFeatures`
- `bagOfFeatures.encode`
- `trainImageCategoryClassifier`
- `imageCategoryClassifier`
- `imageCategoryClassifier.predict`
- `trainRCNNObjectDetector`

See `parpool` for details on how to create a special job on a pool of workers, and connect the MATLAB client to the parallel pool.

3

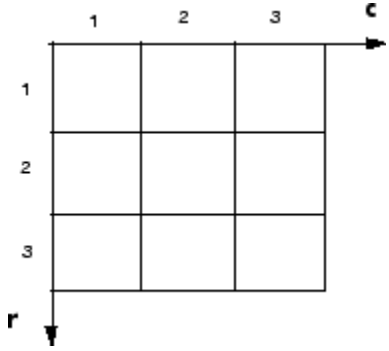
Coordinate Systems

Coordinate Systems

You can specify locations in images using various coordinate systems. Coordinate systems are used to place elements in relation to each other. Coordinates in pixel and spatial coordinate systems relate to locations in an image. Coordinates in 3-D coordinate systems describe the 3-D positioning and origin of the system.

Pixel Indices

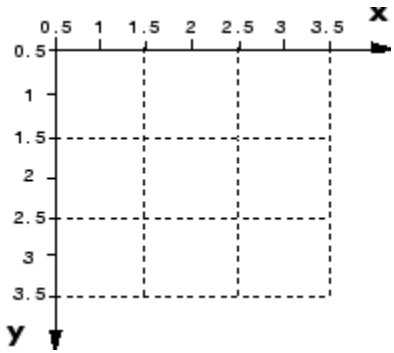
Pixel coordinates enable you to specify locations in images. In the pixel coordinate system, the image is treated as a grid of discrete elements, ordered from top to bottom and left to right.



For pixel coordinates, the number of rows, *r*, downward, while the number of columns, *c*, increase to the right. Pixel coordinates are integer values and range from 1 to the length of the row or column. The pixel coordinates used in Computer Vision System Toolbox software are one-based, consistent with the pixel coordinates used by Image Processing Toolbox™ and MATLAB. For more information on the pixel coordinate system, see “Pixel Indices” (Image Processing Toolbox).

Spatial Coordinates

Spatial coordinates enable you to specify a location in an image with greater granularity than pixel coordinates. Such as, 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 the spatial coordinate system, locations in an image are represented in terms of partial pixels, such as (3.3, 4.7).



For more information on the spatial coordinate system, see “Spatial Coordinates” (Image Processing Toolbox).

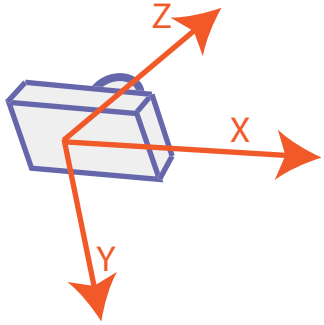
3-D Coordinate Systems

When you reconstruct a 3-D scene, you can define the resulting 3-D points in one of two coordinate systems. In a camera-based coordinate system, the points are defined relative to the center of the camera. In a calibration pattern-based coordinate system, the points are defined relative to a point in the scene.

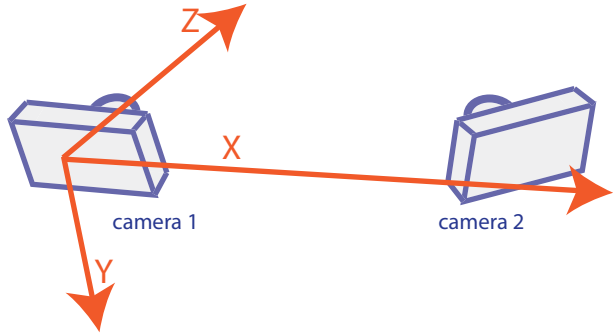
The Computer Vision System Toolbox functions use the right-handed world coordinate system. In this system, the *x*-axis points to the right, the *y*-axis points down, and the *z*-axis points away from the camera. To display 3-D points, use `pcshow`.

Camera-Based Coordinate System

Points represented in a camera-based coordinate system are described with the origin located at the optical center of the camera.



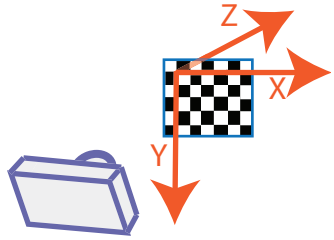
In a stereo system, the origin is located at the optical center of Camera 1.



When you reconstruct a 3-D scene using a calibrated stereo camera, the `reconstructScene` and `triangulate` functions return 3-D points with the origin at the optical center of Camera 1. When you use Kinect® images, the `pcfromkinect` function returns 3-D points with the origin at the center of the RGB camera.

Calibration Pattern-Based Coordinate System

Points represented in a calibration pattern-based coordinate system are described with the origin located at the (0,0) location of the calibration pattern.



When you reconstruct a 3-D scene from multiple views containing a calibration pattern, the resulting 3-D points are defined in the pattern-based coordinate system. The "Structure From Motion From Two Views" example shows how to reconstruct a 3-D scene from a pair of 2-D images containing a checkerboard pattern.

See Also

Related Examples

- “Measuring Planar Objects with a Calibrated Camera”
- “Structure From Motion From Two Views”
- “Structure From Motion From Multiple Views”
- “Depth Estimation From Stereo Video”

Strategies for Real-Time Video Processing in Simulink

- "Optimizing Your Implementation" on page 4-2
- "Developing Your Models" on page 4-4

Optimizing Your Implementation

Video processing is computationally intensive, and the ability to perform real-time video processing is affected by the following factors:

- Hardware capability
- Model complexity
- Model implementation
- Input data size

Optimizing your implementation is a crucial step toward real-time video processing. The following tips can help improve the performance of your model:

- Minimize the number of blocks in your model.
- Process only the regions of interest to reduce the input data size.
- Use efficient algorithms or the simplest version of an algorithm that achieves the desired result.
- Use efficient block parameter settings. However, you need to decide whether these settings best suit your algorithm. For example, the most efficient block parameter settings might not yield the most accurate results. You can find out more about individual block parameters and their effect on performance by reviewing specific block reference pages.

  The two following examples show settings that make each block's operation the least computationally expensive:

  - Resize block — **Interpolation method** = `Nearest neighbor`
  - Blocks that support fixed point — On the **Fixed-Point** tab, **Overflow mode** = `Wrap`
- Choose data types carefully.

  - Avoid data type conversions.
  - Use the smallest data type necessary to represent your data to reduce memory usage and accelerate data processing.

    In simulation mode, models with floating-point data types run faster than models with fixed-point data types. To speed up fixed-point models, you must run them in accelerator mode. Simulink contains additional code to process all fixed-point data

types. This code affects simulation performance. After you run your model in accelerator mode or generate code for your target using the Simulink Coder, the fixed-point data types are specific to the choices you made for the fixed-point parameters. Therefore, the fixed-point model and generated code run faster.

Developing Your Models

Use the following general process guidelines to develop real-time video processing models to run on embedded targets. By optimizing the model at each step, you improve its final performance.

1. Create the initial model and optimize the implementation algorithm. Use floating-point data types so that the model runs faster in simulation mode. If you are working with a floating-point processor, go to step 3.
2. If you are working with a fixed-point processor, gradually change the model data types to fixed point, and run the model after every modification.

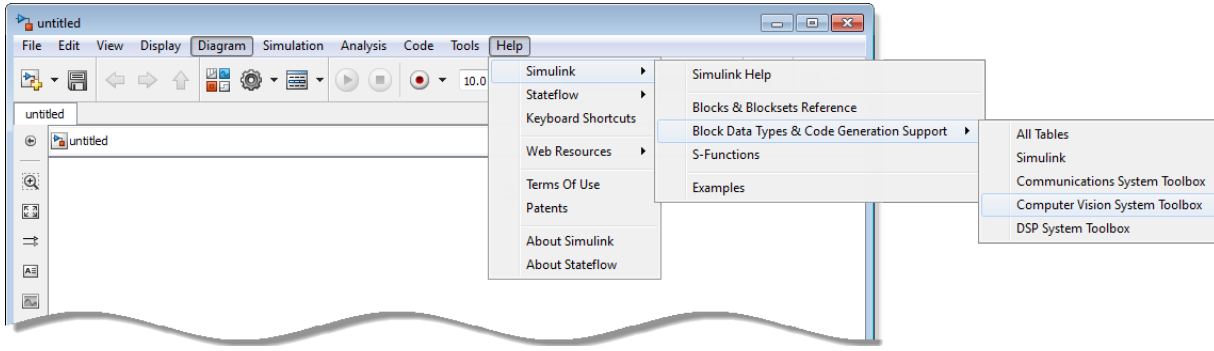
   During this process, you can use data type conversion blocks to isolate the floating point sections of the model from the fixed-point sections. You should see a performance improvement if you run the model in accelerator mode.
3. Remove unnecessary sink blocks, including scopes, and blocks that log data to files.
4. Compile the model for deployment on the embedded target.

5

Data Type Support

Block Data Type Support

The Computer Vision System Toolbox Data Type Support Table is available through the Simulink model Help menu. The table provides information about data type support and code generation coverage for all Computer Vision System Toolbox blocks. Select **Help > Simulink> Block Data Types & Code Generation Support > Computer Vision System Toolbox**.



Fixed-Point Support for MATLAB System Objects

In this section...
“Getting Information About Fixed-Point System Objects” on page 5-3 “Setting System Object Fixed-Point Properties” on page 5-4

For information on working with Fixed-Point features, refer to the “Fixed-Point”topic.

Getting Information About Fixed-Point System Objects

System objects that support fixed-point data processing have fixed-point properties. When you display the properties of a System object™, click `Show all properties` at the end of the property list to display the fixed-point properties for that object. You can also display the fixed-point properties for a particular object by typing `vision.<ObjectName>.helpFixedPoint` at the command line.

The following Computer Vision System Toolbox objects support fixed-point data processing.

Fixed-Point Data Processing Support

```
vision.AlphaBlender
vision.Autocorrelator
vision.BlobAnalysis
vision.BlockMatcher
vision.Convolver
vision.Crosscorrelator
vision.DCT
vision.Deinterlacer
vision.DemosaicInterpolator
vision.FFT
vision.HoughLines
vision.IDCT
vision.IFFT
vision.Maximum
vision.Mean
vision.Median
vision.Minimum
vision.Pyramid
```

`vision.Variance`

Setting System Object Fixed-Point Properties

Several properties affect the fixed-point data processing used by a System object. Objects perform fixed-point processing and use the current fixed-point property settings when they receive fixed-point input.

You change the values of fixed-point properties in the same way as you change any System object property value. You also use the Fixed-Point Designer™ `numerictype` object to specify the desired data type as fixed point, the signedness, and the word- and fraction-lengths.

In the same way as for blocks, the data type properties of many System objects can set the appropriate word lengths and scalings automatically by using full precision. System objects assume that the target specified on the Configuration Parameters Hardware Implementation target is `ASIC/FPGA`.

If you have not set the property that activates a dependent property and you attempt to change that dependent property, you will get a warning message.

You must set the property that activates a dependent property before attempting to change the dependent property. If you do not set the activating property, you will get a warning message.

Note System objects do not support fixed-point word lengths greater than 128 bits.

For any System object provided in the Toolbox, the fimath settings for any fimath attached to a fi input or a fi property are ignored. Outputs from a System object never have an attached fimath.

Hardware Support

Computer Vision System Toolbox Supported Hardware

As of this release, Computer Vision System Toolbox supports the following hardware.

Support Package	Vendor	Earliest Release Available	Last Release Available
Xilinx Zynq-Based Hardware	Xilinx®	R2016a	Current

For a complete list of supported hardware, see Hardware Support.