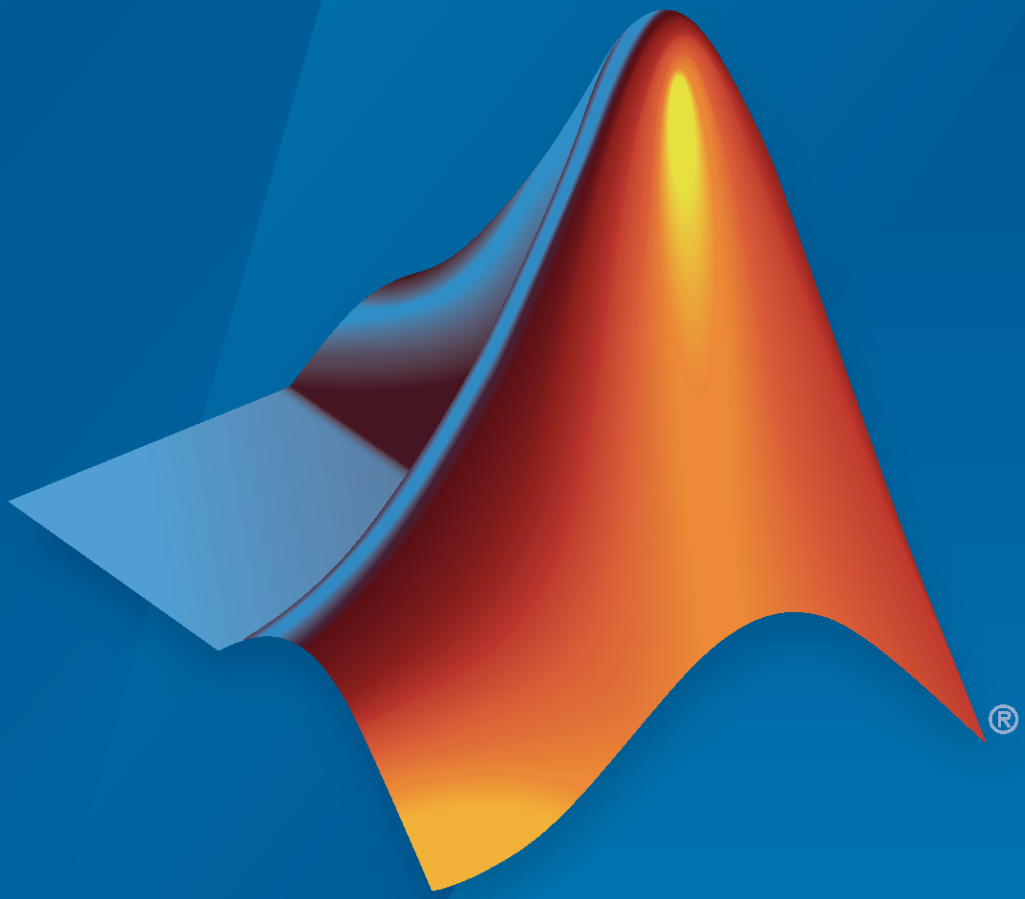
Image Processing Toolbox™ Release Notes



MATLAB®

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.
1 Apple Hill Drive
Natick, MA 01760-2098

Image Processing Toolbox™ Release Notes

© COPYRIGHT 2000–2026 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.

Contents

R2026a

New Support Package: Optical Design and Simulation Library for Image Processing Toolbox **1-2**

Circle Detection: Detect circles using YOLOX deep learning model **1-3**

SAM 2: Segment images using Segment Anything Model 2 . . . . . **1-4**

Image Display: Label ROIs **1-4**

Image Display: Draw masks using paintbrush **1-5**

Volume Display: Specify camera view and interactivity **1-6**

linkviewers Function: Synchronize camera properties of multiple viewers **1-6**

Volume Viewer App: Updated toolstrip and enhanced rendering editor **1-7**

regionprops Function: Get boundary coordinates of regions **1-7**

imresize3 Function: Resize color and multichannel volumes **1-7**

dlresize Function: Map points from input space to output space by aligning corners **1-7**

affineOutputView Function: Specify spatial referencing **1-7**

dicomCollection Function: Specify list of filenames **1-7**

DICOM Files: Read and write DICOM metadata with minimal preprocessing **1-7**

DICOM Browser App: Export volumes to Medical Volume Viewer or Medical Image Labeler **1-8**

Hyperspectral Image Processing: Hyperspectral Imaging Library for Image Processing Toolbox (Version 26.1) **1-8**

C Code Generation: Generate code from additional functions using MATLAB Coder **1-9**

GPU Code Generation Support for resize3dLayer Function **1-9**

Functionality being removed or changed **1-9**
Volume Segmenter is not recommended **1-9**
bigimage and bigimageDatastore have been removed **1-9**
imshow Function: Live Editor always uses initial magnification value "fit" **1-9**
bwskel Function: Behavior changes resulting from improvements to 2-D skeletonization algorithm **1-10**
Parent-child dependencies returned as sparse matrix in code generation for bwboundaries **1-10**

R2025b

Quality and stability improvements **2-2**

R2025a

Image Display: View pixel information and add title and scale bar **3-2**

Volume Display: View voxel information and add title and scale bar **3-3**

Volume Display: Specify volume interpolation method and isosurface transparency, new default transparency **3-5**

Label Display: Improve visual contrast of displayed labels **3-6**

Image Blending: Blend images using Poisson blend modes **3-6**

Image Demosaicing: Specify bits per sample **3-6**

Blocked Images: Preserve image properties when concatenating levels **3-7**

C Code Generation: Generate code from additional functions using MATLAB Coder **3-7**

GPU Code Generation: Generate CUDA code from improfile function using GPU Coder **3-7**

Enhanced GPU Support for bwdist Function **3-7**

Hyperspectral Image Processing: Hyperspectral Imaging Library for Image Processing Toolbox (Version 25.1) **3-7**

Functionality being removed or changed **3-8**
labelvolshow and volshow objects have been removed **3-8**
im2java2d function has been removed **3-8**

Preferences window renamed to Settings window, new location for imshow settings 3-8
Interpolation property has no effect on bigimageshow 3-8

R2024b

Image Segmentation: Automatically or interactively segment image using Segment Anything Model 4-2

Image Segmentation: Segment 2-D grayscale, RGB, and multichannel images using the ISODATA algorithm 4-2

Image Display: Efficiently display 2-D images and large images 4-3

Volume Visualization: Renamed objects and classes for image display . . 4-3

Volume Visualization: New viewer cropping options 4-3

Volume Visualization: Specify volume display range and transparency map, new default overlay colormap 4-4

Image Blending: Blend two 2-D grayscale or RGB images 4-4

Image Quality Test Charts: Measure sharpness of Dead Leaves and Spilled Coins test charts 4-5

Blocked Images: Support for reading 3-D TIFF images 4-5

montage Function: Display montage of deep learning arrays, and control thumbnail resizing method 4-5

imregcorr Function: Improved registration algorithms, including support for gradient correlation 4-5

Deep Learning: Updated functions to improve training process 4-5

C Code Generation: Generate code from additional functions using MATLAB Coder 4-6

GPU Code Generation: Generate CUDA code from additional functions using GPU Coder 4-6

GPU Acceleration for montage Function 4-6

Thread-Based Environment: Run DICOM functions in a thread-backed pool 4-6

Renamed Support Package: Image Processing Toolbox Hyperspectral Imaging Library renamed to Hyperspectral Imaging Library for Image Processing Toolbox 4-7

Hyperspectral Image Processing: Hyperspectral Imaging Library for Image Processing Toolbox (Version 24.2) **4-7**

Functionality being removed or changed **4-7**
imregcorr Function: Behavior changes resulting from improvements to registration algorithms **4-7**
Viewer3D renamed to Viewer, new default background color **4-8**
denoisingImageSource function has been removed **4-8**

R2024a

Image Segmentation: Segment objects in images using Segment Anything Model **5-2**

Image Quality: Analyze additional test charts **5-2**

Volume Visualization: Label volumes using 3-D point and line annotations **5-3**

Object Analysis: Create binary masks from circles in image **5-3**

Object Analysis: Measure circularity using Image Region Analyzer app and bwpropfilt function **5-3**

Object Analysis: Filter properties of connected components structure and display result **5-3**

NIfTI File Format: Simplified workflow for writing compressed files **5-4**

Deep Learning: Updated functions and examples to improve training process **5-4**

C Code Generation: Generate code from additional functions using MATLAB Coder **5-4**

GPU Code Generation: Generate CUDA code from dlresize function using GPU Coder **5-5**

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (Version 24.1) **5-5**

Functionality being removed or changed **5-5**
hyperSlic will become hyperslic **5-5**

R2023b

Image Display and Exploration: Enhancements to Image Viewer app . . . **6-2**

Volume Visualization: Cinematic rendering, volumetric light scattering, advanced lighting controls, and interactive cropping . . . **6-2**

Border selection and deletion: Remove or retain specified borders and border structures . . . **6-4**

imclearborder: Improved performance . . . **6-4**

C Code Generation: Generate code from additional functions using MATLAB Coder . . . **6-4**

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (Version 23.2) . . . **6-5**

Functionality being removed or changed . . . **6-5**

imtool will be removed . . . **6-5**

bigimage and bigimageDatastore will be removed . . . **6-6**

R2023a

Volume Visualization: Display blocked volumes . . . **7-2**

Blocked Images: Add levels to large images, combine block locations, and other enhancements . . . **7-2**

Image Batch Processor App: Support for image datastores and other enhancements . . . **7-2**

Color Thresholder App: Enhanced in-app image acquisition and updated graphics requirements . . . **7-3**

bwboundaries function: Trace exterior boundaries . . . **7-3**

iradon function: Additional support for data type single . . . **7-3**

Preferences in MATLAB Online: Update preferences interactively using the MATLAB Preferences window . . . **7-3**

C Code Generation: Generate code from additional functions using MATLAB Coder . . . **7-4**

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (Version 23.1) . . . **7-4**

Functionality being removed or changed **7-4**
The regionprops function uses new equation for circularity **7-4**
MATLAB Online limits maximum image display resolution **7-5**
Color Thresholder App: Updated graphics system requirements **7-5**

R2022b

Geometric Transformations: Perform geometric transformations using premultiply matrix convention **8-2**

Volume Visualization: Enhanced volume, surface, and scene rendering **8-2**

EXR Files: Read and write images stored in the EXR file format **8-3**

Color Conversion: Support for ProPhoto (ROMM RGB) color space **8-3**

DICOM: Support for reading video data **8-3**

viscircles Function: Accepts scalar radii **8-3**

locallapfilt Function: Improved performance **8-3**

C Code Generation: Generate code from additional functions using MATLAB Coder **8-4**

GPU Code Generation: Generate CUDA code from additional functions using GPU Coder **8-4**

GPU Acceleration for gray2ind Function **8-4**

Thread-Based Environment: Run functions in a thread-backed pool **8-4**

Functionality being removed or changed **8-5**
Postmultiply geometric transformation objects are not recommended **8-5**
volshow function creates Volume object **8-5**
labelvolshow and images.compatibility.volshow.R2022a.volshow will be removed **8-6**

R2022a

Deep Learning: Added examples using deep neural networks **9-2**

Image Browser App: Additional Import and Export Capabilities **9-2**

Image Region Analyzer App: Additional Export Capabilities **9-2**

imbilatfilt Function: Improved performance for uint8 and single data types **9-2**

dicomCollection Function: Improved performance **9-3**

C Code Generation: Generate code from additional functions using MATLAB Coder **9-3**

GPU Code Generation: Generate CUDA code from additional functions using GPU Coder **9-3**

Functionality being removed or changed **9-3**
The imsharpen function uses different color space conversion operations for RGB images **9-3**
The regionprops function always stores the Image, ConvexImage, and FilledImage properties as cell arrays in the output table for all inputs **9-4**

R2021b

Blocked Images: Create and display labeled blocked images **10-2**

DICOM: Find and set attributes in DICOM metadata **10-2**

Image Quality Metrics: Calculate SSIM metric of deep learning arrays and specify dimensions of computation **10-2**

Deep Learning: Added examples using deep neural networks **10-2**

medfilt3 Function: Improved performance for small neighborhood sizes **10-2**

C Code Generation: Generate code from five functions using MATLAB Coder **10-3**

C Code Generation: Generate portable C code that has improved performance for seven functions **10-3**

GPU Acceleration for ssim Function **10-4**

Thread-Based Environment: Run functions in a thread-backed pool . . . **10-4**

Functionality being removed or changed **10-4**
The Format property of the GenericImage object is not recommended . . **10-4**

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (August 2021, Version 21.1.3) **11-2**

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (May 2021, Version 21.1.2) **11-2**

Deep Learning Networks: Create and combine encoder and decoder modules for networks with repetitive structures **11-3**

Deep Learning Networks: Create generators and discriminators for GAN networks **11-3**

Deep Learning Data Processing: Rearrange data between dimensions and resize data to match feature map **11-4**

Deep Learning: Added examples using deep neural networks **11-4**

Blocked Images: Process N-D images that are too large to fit in memory **11-5**

Volume Segmenter app supports blocked images **11-5**

RAW Files: Read RAW files and color filter array data, and convert to RGB images **11-6**

Image Quality Metrics: Calculate metrics of deep learning arrays and specify dimensions of computation **11-6**

Image Quality Test Charts: Read and analyze additional versions of Imatest eSFR test charts **11-6**

Image Quality Test Charts: Measure scene illuminant using X-Rite ColorChecker test chart **11-6**

Registration Estimator App: Support for KAZE and ORB registration techniques **11-7**

imerase: Erase rectangular region of image **11-7**

randomWindow2d: Select rectangular region of image **11-7**

C Code Generation: Generate code from the adapthisteq function using MATLAB Coder **11-7**

C Code Generation: Generate portable C code with improved performance for six functions **11-7**

GPU Code Generation Support for resize2dLayer Object **11-8**

GPU Acceleration for imcrop, multissim, multissim3, psnr, and wiener2 Functions and Enhanced GPU Support for imwarp **11-8**

Functionality being removed or changed **11-8**
bigimage object and bigimageDatastore object are not recommended . . . **11-8**
randomCropWindow2d function is not recommended **11-8**
esfrChart object now refines slanted edge ROI positions **11-8**

R2026a

Version: 26.1

New Features

Bug Fixes

Compatibility Considerations

★New Support Package: Optical Design and Simulation Library for Image Processing Toolbox

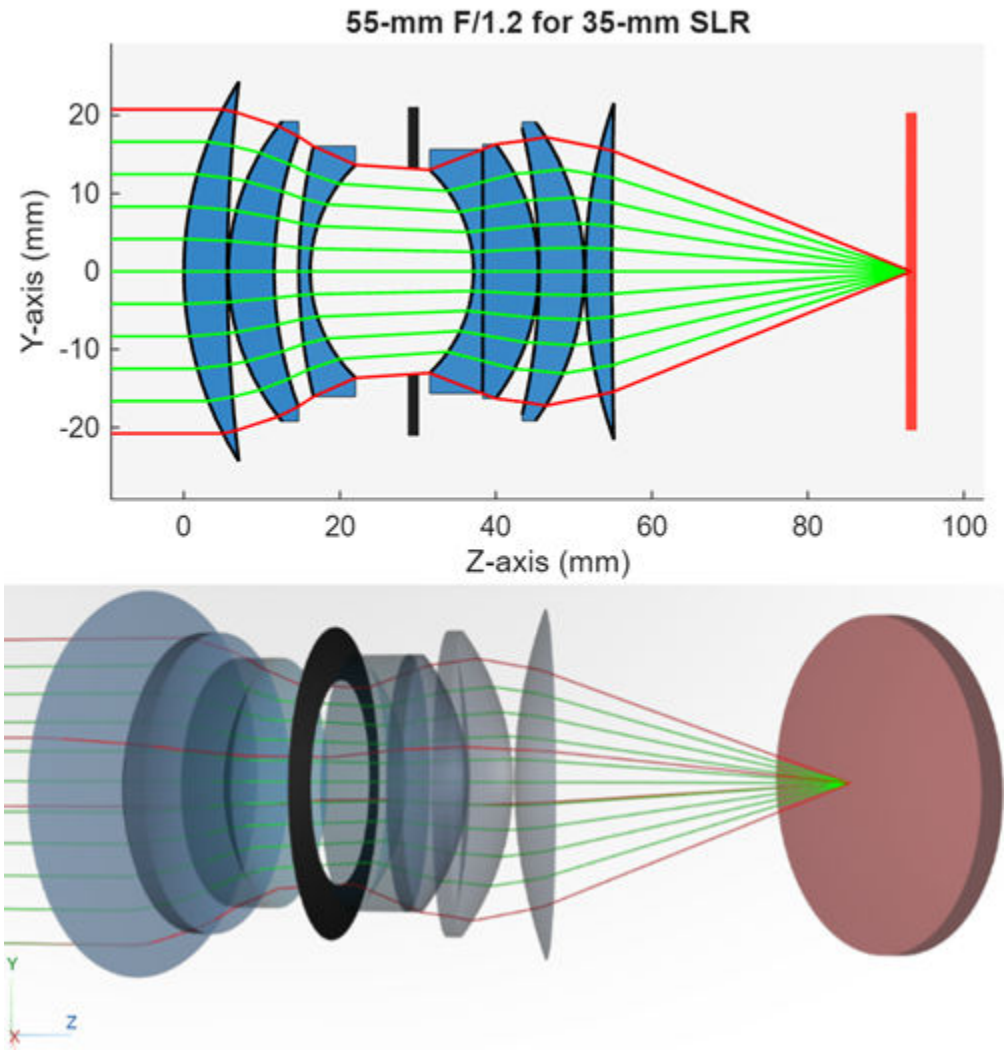
Design and simulate optical systems using the Optical Design and Simulation Library for Image Processing Toolbox. The library enables you to:

- Create and tune custom optical designs.
- Import optical designs from ZMX files.
- Visualize optical designs in 2-D and 3-D.
- Evaluate optical system performance by tracing rays, computing spot diagrams, and performing aberration, lens distortion, field curvature, and polarization analysis.
- Use bulk and coating materials from comprehensive material libraries, import bulk materials, and create custom optical materials and coatings.
- Optimize optical systems and optical coatings using optimization tools available in Optimization Toolbox™.

The Optical System Designer app provides an interactive interface for optical system design. You can use the app to design custom optical systems, simulate imported optical systems from ZMX files, and interactively visualize traced rays, spot diagrams, aberrations, distortions, field curvature, and other analyses.

To get started with the Optical Design and Simulation Library for Image Processing Toolbox, see “Get Started with Optical Design and Simulation”.

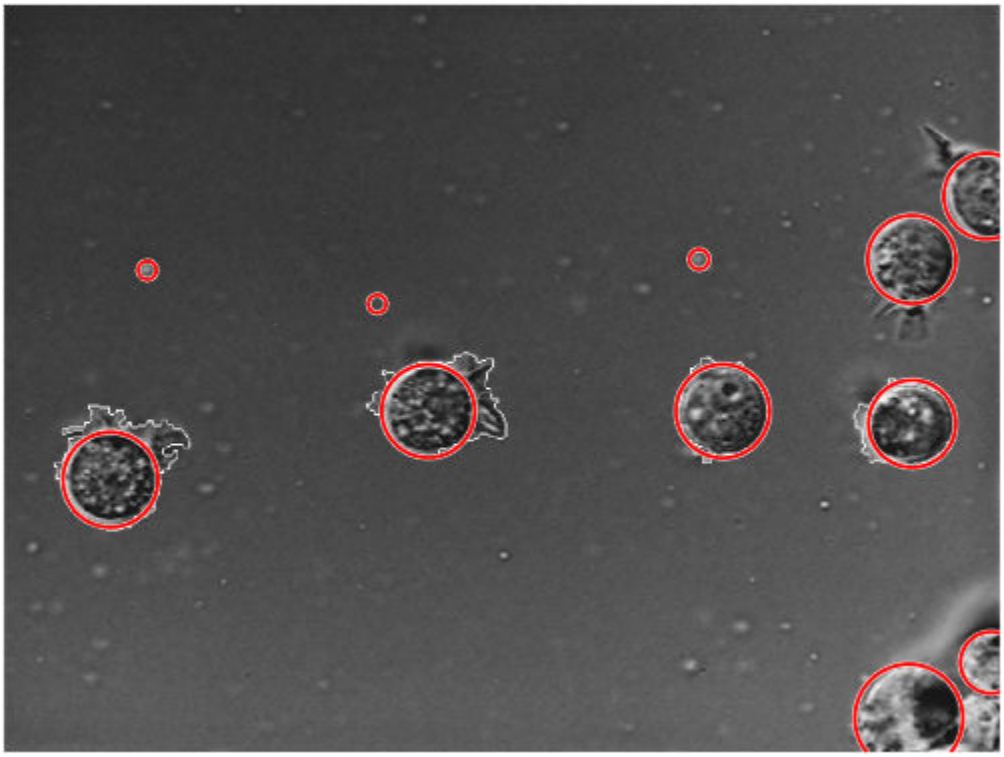
You can install the Optical Design and Simulation Library for Image Processing Toolbox from the Add-On Explorer. For more information about installing add-ons, see Get and Manage Add-Ons.



Circle Detection: Detect circles using YOLOX deep learning model

Identify circular objects of any size in images by using the `imfindcirclesYOLO` function. You can select a pretrained YOLOX model for detecting the circles by using the `Method` name-value argument of the function.

This function requires the Image Processing Toolbox Model for Circle Detection support package. You can install the support package from Add-On Explorer. For more information about installing add-ons, see “Get and Manage Add-Ons”.



⚠SAM 2: Segment images using Segment Anything Model 2

Segment images using Segment Anything Model 2 (SAM 2) by specifying the model variant using the new capabilities of the `segmentAnythingModel` object and `imsegsam` function. To use SAM 2, you require the Image Processing Toolbox Model for Segment Anything Model 2 support package. You can install the support package from Add-On Explorer. For more information about installing add-ons, see “Get and Manage Add-Ons”.

⚠Compatibility Considerations

The default model for the `segmentAnythingModel` object and `imsegsam` function is now `"sam2-large"`. To use any of the SAM 2 models, install the Image Processing Toolbox Model for Segment Anything Model 2 add-on.

★Image Display: Label ROIs

Label scenes in a `Viewer` object using new ROI annotation objects. You can create viewer-based ROIs three ways:

- From the viewer toolbar, select the **Draw annotations** icon and select a shape to draw. Click in the scene and drag to draw the ROI. To export the ROI as an object, right-click the shape and select **Export annotation to workspace**.
- Use the new `uidraw` function, which enables an interactive drawing mode in a viewer you specify.
- Use an object creation function, such as `images.ui.graphics.roi.Circle`. To display the ROI, add the object to the `Annotations` property of a `Viewer` object.

This table summarizes the options for creating each ROI shape:

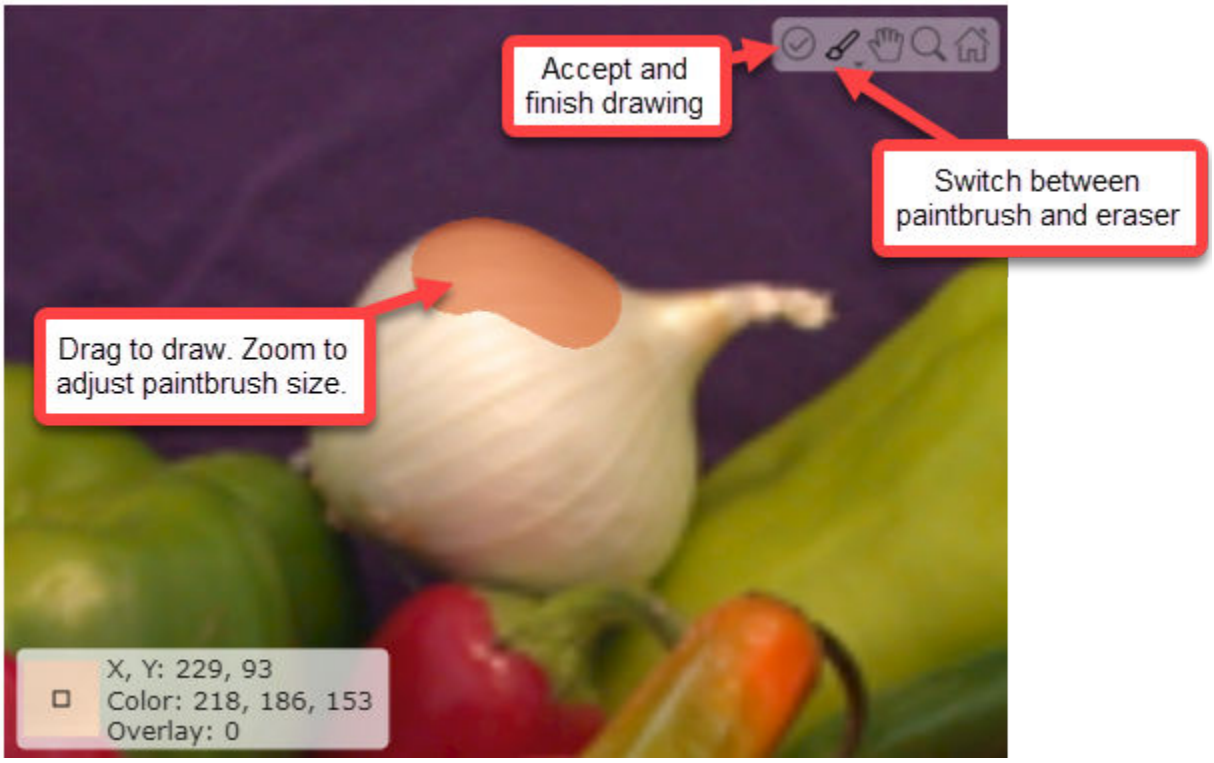
Shape	Viewer Toolbar Support	`uidraw` Syntax	Object Creation Function
`Line`	2-D and 3-D	`uidraw(viewer,"line")`	`images.ui.graphics.roi.Line`
`Point`	2-D and 3-D	`uidraw(viewer,"point")`	`images.ui.graphics.roi.Point`
`Rectangle`	2-D only	`uidraw(viewer,"rectangle")`	`images.ui.graphics.roi.Rectangle`
`Circle`	2-D only	`uidraw(viewer,"circle")`	`images.ui.graphics.roi.Circle`
`Polygon`	2-D only	`uidraw(viewer,"polygon")`	`images.ui.graphics.roi.Polygon`
`Angle`	2-D only	`uidraw(viewer,”angle”)`	`images.ui.graphics.roi.Angle`
`Polyline`	None	`uidraw(viewer,”polyline”)`	`images.ui.graphics.roi.Polyline`
`Freehand`	None	`uidraw(viewer,”freehand”)`	`images.ui.graphics.roi.Freehand`
`Ellipse`	None	`uidraw(viewer,”ellipse”)`	`images.ui.graphics.roi.Ellipse`

To convert the ROI to a binary mask for additional processing and analysis, use the `createMask` object function.

The `Line` and `Point` objects, originally introduced in R2024a, now have the `Visible`, `Interactions`, and `Parent` properties to specify their visibility, interactive behavior, and parent object, respectively. The `Line` object also has the `Distance` property, which indicates the distance between the endpoints of the line.

Image Display: Draw masks using paintbrush

The new paintbrush tool supports drawing overlay masks on images displayed in a `Viewer` object. Enable the paintbrush tool by using the `uipaint` function. After you draw the mask in the viewer window and select the accept icon in the viewer toolbar, the function returns a binary mask you can use for image processing and analysis.



Volume Display: Specify camera view and interactivity

Use a new function and several new Viewer Properties to customize the camera view and camera interactions.

- Update the camera view by specifying azimuth and elevation angles using the new `viewangle` function.
- Control the behavior of rotation, zoom, and pan interactions by specifying the new `CameraInteractionStyle` property. For example, select a style that rotates the view about the pointer location instead of the center of the volume, or that uses a dolly zoom to simulate a first-person perspective.
- View the scene using a perspective or orthographic camera by specifying the new `CameraProjection` property.
- Control the field of view by specifying the new `CameraViewAngle` property.
- Define the visible area in view of the camera by specifying the new `CameraViewport` property.

linkviewers Function: Synchronize camera properties of multiple viewers

Synchronize camera properties of multiple viewers using the new `linkviewers` function. You can synchronize camera properties such as zoom, view angle, projection, and so on for seamless comparison.

Volume Viewer App: Updated toolstrip and enhanced rendering editor

The updated Volume Viewer app now:

- Displays the voxel coordinates and intensity for a point, when you hover over it, in the bottom of the app window.
- Provides horizontal scrollbars, which make it easier to navigate slice planes, and labels that indicate the current slice and total number of slices in that plane.
- Supports new rendering styles in the Rendering Editor, including cinematic rendering, minimum intensity projection, gradient opacity, and slice planes.
- Supports visualizing and adjusting the opacity of the volume and labels simultaneously by selecting the new **View Volume** and **View Labels** buttons.
- Displays options for importing volume and label data in a single **Import** drop-down list.

regionprops Function: Get boundary coordinates of regions

The `regionprops` function can now return the coordinates of pixels on the exterior boundary of regions in a binary image. To return the coordinates of boundary pixels, include `"BoundaryCoordinates"` when specifying the `properties` input argument.

imresize3 Function: Resize color and multichannel volumes

The `imresize3` function can now resize color and multichannel volumes.

dlresize Function: Map points from input space to output space by aligning corners

The `dlresize` function can now map points from input space to output space by aligning corners. To align corners, specify the interpolation method `Method` as `"linear"` and the `GeometricTransformMode` as `"aligned-corners"`.

affineOutputView Function: Specify spatial referencing

The `affineOutputView` function now accepts a spatial referencing object for the input image. Use this argument when your input image is not in the intrinsic coordinate system.

dicomCollection Function: Specify list of filenames

The `dicomCollection` function now accepts a list of DICOM filenames specified as a vector of strings as the input source.

DICOM Files: Read and write DICOM metadata with minimal preprocessing

The `dicominfo` function now supports reading certain metadata strings with minimal preprocessing using the new `PreserveStrings` argument. When you specify `PreserveStrings` as `true`, the `dicominfo` function reads attributes with Person Name (PN), Decimal String (DS), and Integer String (IS) value representations as character vectors, preserving their original format. Reading

these attributes as character vectors can simplify searching for patient information, and preserves the precision of numeric strings. By default, `dicominfo` continues to convert PN attributes to structures with fields for each part of the patient name, and DS and IS attributes to numeric values.

When you create a new DICOM file using the `dicomwrite` and `dicomanon` functions, you can provide metadata that specifies PN attributes as structures or character vectors, and DS and IS attributes as numeric values or character vectors. Specify character vectors when you want to preserve the precision of DS and IS values from the original file.

DICOM Browser App: Export volumes to Medical Volume Viewer or Medical Image Labeler

The DICOM Browser app now supports exporting volume series to the Medical Volume Viewer and Medical Image Labeler apps, which apply file metadata to display volumes using the correct voxel spacing and patient orientation, and with anatomical display markers. You can also export 2-D images and image series to the Medical Image Labeler app. The new export options require a Medical Imaging Toolbox™ license.

Hyperspectral Image Processing: Hyperspectral Imaging Library for Image Processing Toolbox (Version 26.1)

- Label regions of interest (ROI) in hyperspectral and multispectral images by using the Spectral Image Labeler app. The app reads images stored in files or from the workspace. Using the app, you can:
  - Choose false color, RGB, CIR, or custom views of spectral images for ease of labeling.
  - Create multiple label definitions and label ROIs using built-in automatic and manual techniques.
  - Write, import, and use your own custom automation algorithm to automatically label ground truth data.
  - Export the labeled ground truth data as a `groundTruthSpectralImage` object. You can use this object to share labels with colleagues or to store ground truth labels for pixel classification or semantic segmentation.
- Manage labeled ground truth data for pixel classification and semantic segmentation applications using the `groundTruthSpectralImage` object and its functions.
  - Export a ground truth object from the Spectral Image Labeler app or create one programmatically at the command line.
  - Load existing ground truth data and labels into the Spectral Image Labeler app from `groundTruthSpectralImage` objects.
- R2026a includes these new examples:
  - “Combine Landsat 9 Multispectral Images Using Image Mosaicing”
  - “Segment Hyperspectral Image Using Segment Anything Model in Spectral Image Labeler”

To use these functions and tools, you must download the Hyperspectral Imaging Library for Image Processing Toolbox from the Add-On Explorer. For more information, see “Get and Manage Add-Ons”.

⚠★Compatibility Considerations

- The `hypercube` function will be removed in a future release. To create a `hypercube` object, use the `imhypercube` or `geohypercube` functions instead.
- The `hypercube` object no longer has the `DataCube` property.
- You can no longer read multispectral data using the `hypercube` object. Use the `multicube` object to read multispectral data.

C Code Generation: Generate code from additional functions using MATLAB Coder

This list indicates the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. For a complete list of Image Processing Toolbox functions that support code generation, see Functions Supporting Code Generation.

- `resize3dLayer`
- `ssim`

GPU Code Generation Support for resize3dLayer Function

The `resize3dLayer` function has been enabled for optimized CUDA® code generation in this release.

⚠Functionality being removed or changed

Volume Segmenter is not recommended

Still runs

The Volume Segmenter app is no longer recommended. Use the Medical Image Labeler app instead (requires Medical Imaging Toolbox). Medical Image Labeler offers several advantages, such as:

- Use of spatial referencing information found in file metadata.
- Simultaneous display of three orthogonal cross-sections.
- Improved performance.

bigimage and bigimageDatastore have been removed

Errors

The `bigimage` object and the `bigimageDatastore` object have been removed. To read and process large or multiresolution images, use the `blockedImage` object and the `blockedImageDatastore` object instead.

imshow Function: Live Editor always uses initial magnification value "fit"

Behavior change

When you use the `imshow` function in the Live Editor, the default `InitialMagnification` value is now `"fit"`. Previously, the default value was `100`, and titles could appear cropped.

You do not need to update your code. If you specify the `InitialMagnification` value in a live script, the function ignores it without error. However, the size of images may differ from previous releases.

bwskel Function: Behavior changes resulting from improvements to 2-D skeletonization algorithm
Behavior change

The `bwskel` function now implements an improved skeletonization algorithm for 2-D images. The new algorithm returns more symmetric and realistic skeletons. The results can differ from prior releases.

Parent-child dependencies returned as sparse matrix in code generation for bwboundaries
Behavior change

The `bwboundaries` function now returns parent-child dependencies `A` as a sparse matrix in C/C++ and GPU code generation, instead of returning a full matrix as in previous releases.

R2025b

Version: 25.2

Bug Fixes

Quality and stability improvements

R2025b delivers quality and stability improvements, building on the new features introduced in R2025a.

R2025a

Version: 25.1

New Features

Bug Fixes

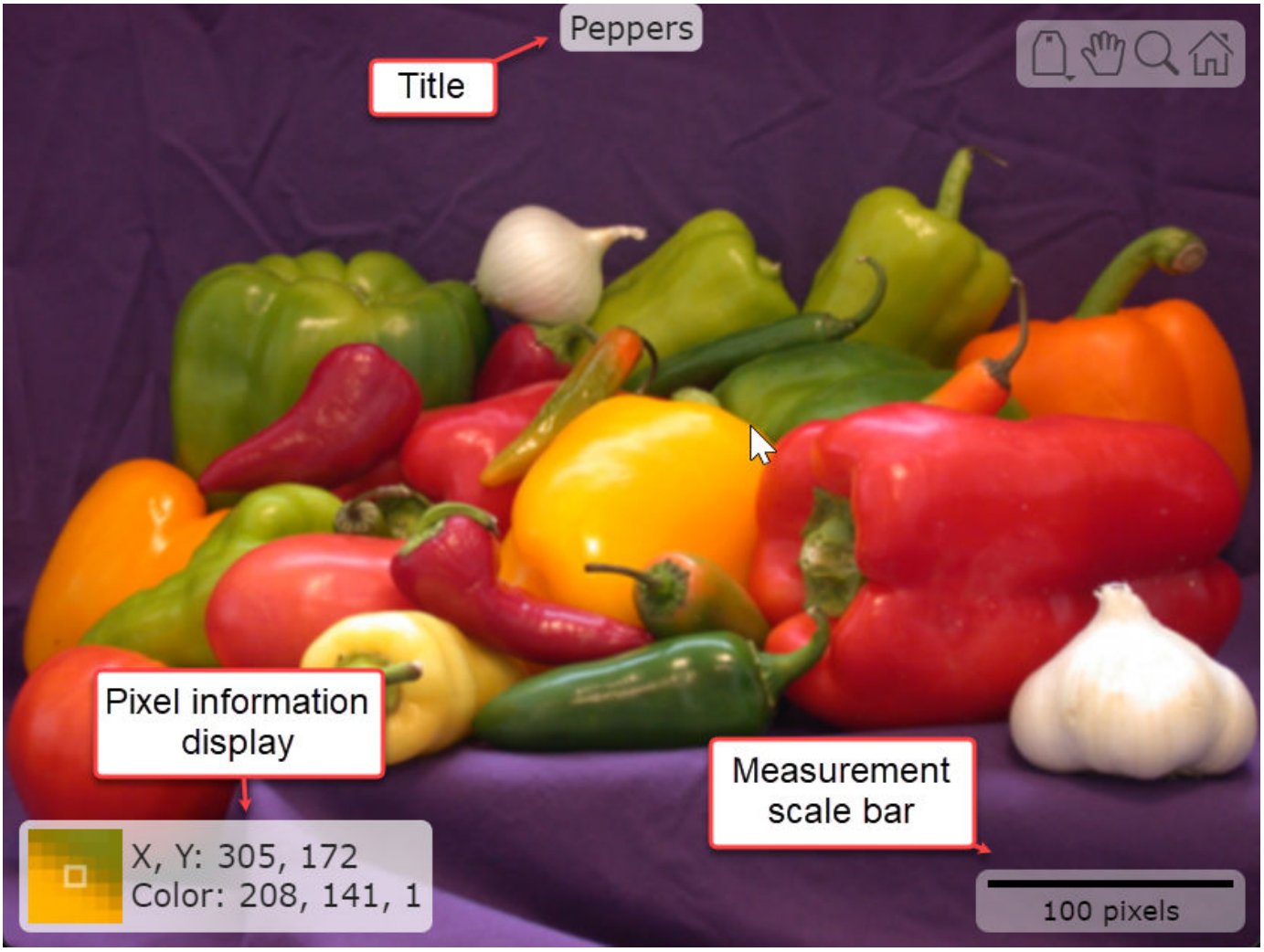
Compatibility Considerations

Image Display: View pixel information and add title and scale bar

The 2-D viewer offers new options when displaying images using `imageshow`. Specify Viewer Properties by using `viewer2d` name-value arguments, or by using dot notation after object creation. The viewer now:

- Displays the toolbar by default, regardless of whether the mouse cursor is inside or outside the viewer window.
- Supports a title. Add a title by specifying the new `Title` property.
- Supports adding a measurement scale bar. In prior releases, the scale bar only showed the zoom level of the image. Starting this release, you can add a measurement scale bar by specifying the new `ScaleBarStyle` property as `"measure"`.
- Displays information about the pixel beneath the pointer. You can interactively copy pixel coordinates and intensity or color values to the clipboard, or export pixel information to the workspace from the viewer context menu. Toggle the visibility of the pixel information display from the viewer context menu or by using the new `DisplayInfo` property.

Toggle the pixel information display for individual images by using the new `Pickable` property of the `Image` object. This option is useful when a viewer contains multiple overlaid images, and you want to view pixel values for only a certain image. Specify Image Properties by using `imageshow` name-value arguments, or by using dot notation after object creation.



Compatibility Considerations

- The viewer toolbar is now always visible by default, regardless of whether the pointer is inside or outside the viewer window. To preserve the behavior of previous releases, specify the `Toolbar` property value as `"hover"`.
- The viewer window always displays pixel information by default. To hide the pixel information, you can specify the `DisplayInfo` property as `"false"`.
- The viewer window hides the scale bar by default. To show the scale bar, you can specify the `ScaleBar` property as `"on"`.

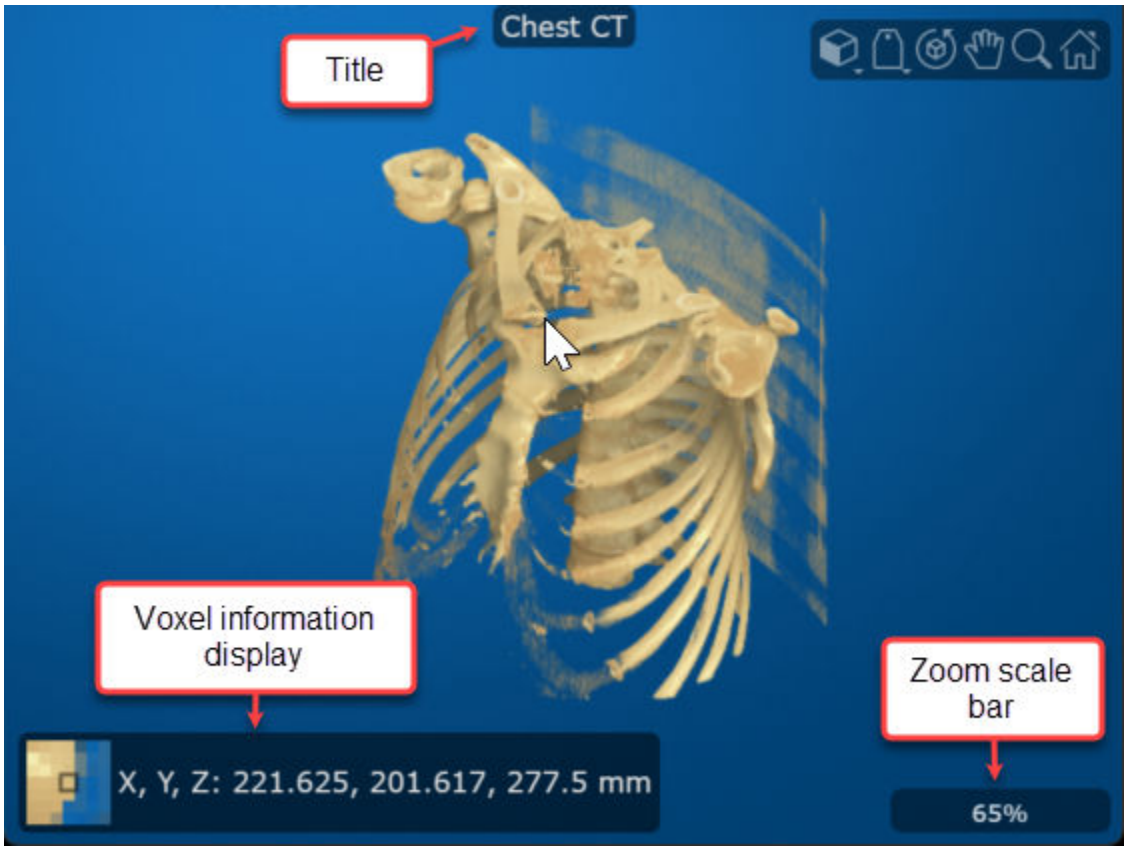
Volume Display: View voxel information and add title and scale bar

The 3-D viewer offers new options when displaying volumes, blocked volumes, and surfaces. Specify Viewer Properties by using `viewer3d` name-value arguments, or by using dot notation after object creation. The viewer now:

- Displays the toolbar by default, regardless of whether the pointer is inside or outside the viewer window.

- Supports a title. Add a title by specifying the new `Title` property.
- Supports adding a zoom scale bar by specifying the new `ScaleBarStyle` property as `"zoom"`.
- Supports displaying information about the voxel or surface point beneath the pointer. Toggle the visibility of the voxel information display from the viewer context menu or by using the new `DisplayInfo` property. You can interactively copy voxel coordinates to the clipboard, or export voxel information to the workspace from the viewer context menu.

Toggle the information display for individual volumes and surfaces by using the new `Pickable` property. This option is useful when a viewer contains multiple objects, and you want to view coordinates for only certain objects. Specify `Pickable` by using `volshow` name-value arguments, or by using dot notation after `Surface`, `Volume`, or `BlockedVolume` object creation. For details, see `Surface`, Volume Properties, and BlockedVolume Properties, respectively.



Compatibility Considerations

- The viewer toolbar is now always visible by default, regardless of whether the pointer is inside or outside the viewer window. To preserve the behavior of previous releases, specify the `Toolbar` property value as `"hover"`.

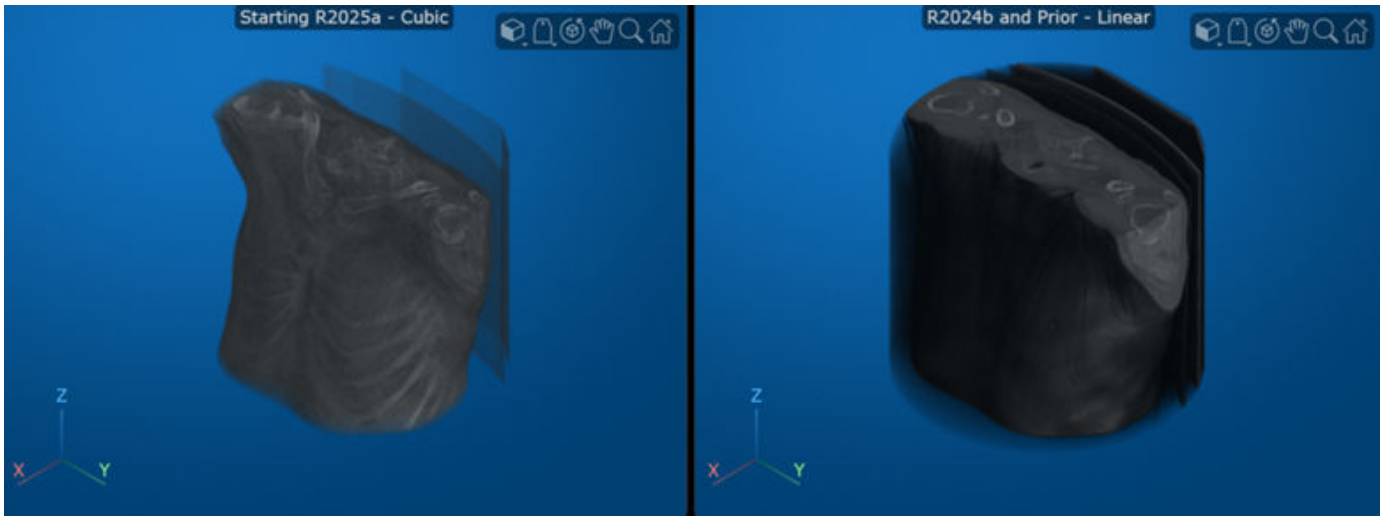
⚠Volume Display: Specify volume interpolation method and isosurface transparency, new default transparency

The toolbox offers new Volume Properties and BlockedVolume Properties to tune the appearance of volumes. Specify properties using `volshow` name-value arguments, or by using dot notation after object creation:

- Specify the interpolation method for volume rendering by using the new `Interpolation` property.
- Specify the transparency of volumes displayed using the `Isosurface` rendering style by using the new `IsosurfaceAlpha` property.

⚠Compatibility Considerations

- To improve the default appearance of most volumes displayed using `volshow`, the default `Alphamap` property value for `Volume` and `BlockedVolume` objects has changed from `"linear"` to `"cubic"`. To preserve the default appearance of previous releases, specify the `Alphamap` property value as `"linear"`. There is no requirement to update your code.



- `Surface` objects now display as fully opaque by default. To preserve the default appearance of previous releases, specify the `Alpha` property value as `0.15`. There is no requirement to update your code.

Label Display: Improve visual contrast of displayed labels

The `labeloverlay` and `label2rgb` functions offer a new method of assigning colors to labels such that adjacent labels have higher visual contrast.

To specify the color assignment using the `labeloverlay` function, specify the new `ColorAssignment` name-value argument as `"contrasting-neighbors"`.

To specify the color assignment using the `label2rgb` function, specify the existing `colorAssignment` argument as `"contrasting-neighbors"`.

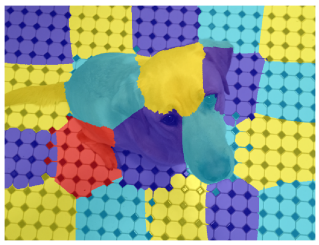


Image Blending: Blend images using Poisson blend modes

The `imblend` function now supports seamless image blending using the Poisson blend modes. The `"Poisson"` blend mode preserves the gradient of the foreground image, while the `"PoissonMixGradients"` blend mode mixes the gradients of the foreground and background images.

Image Demosaicing: Specify bits per sample

The `demosaic` function now enables you to specify the actual bit depth of the Bayer data by using the new `BitsPerSample` name-value argument.

Blocked Images: Preserve image properties when concatenating levels

When you concatenate levels of `blockedImage` objects using the `concatenateLevels` function, each level of the resulting blocked image preserves the `BlockSize`, `WorldStart`, and `WorldEnd` properties from the corresponding level of the input blocked image.

C Code Generation: Generate code from additional functions using MATLAB Coder

This list indicates the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. For a complete list of Image Processing Toolbox functions that support code generation, see Functions Supporting Code Generation.

- `imblend`
- `improfile`

GPU Code Generation: Generate CUDA code from improfile function using GPU Coder

The `improfile` function has been enabled for optimized CUDA code generation in this release.

Enhanced GPU Support for bwdist Function

The `bwdist` function, which already supports GPU acceleration, now extends that support to processing 3-D images. GPU acceleration requires a Parallel Computing Toolbox™ license and meeting the GPU computing requirements detailed here.

Hyperspectral Image Processing: Hyperspectral Imaging Library for Image Processing Toolbox (Version 25.1)

The `hypercube` object has these new capabilities and changes.

- Create a `hypercube` object for hyperspectral data with or without geospatial information by using the `geohypercube` and `imhypercube` functions, respectively. Creating a `hypercube` object with geospatial information using the `geohypercube` function requires a Mapping Toolbox™ license.
- To load the entire hyperspectral data cube into memory, use the `gather` function, as the `hypercube` object no longer has the `DataCube` property.
- Perform block processing on the hyperspectral data using built-in or custom functionality using the `apply` function.

Explore, analyze, and visualize multispectral data using the `multicube` object.

- Create a `multicube` object for multispectral data with or without geospatial information by using the `geomulticube` and `immulticube` functions, respectively. Creating a `multicube` object with geospatial information using the `geomulticube` function requires a Mapping Toolbox license.

- Assign pixel values using the `assignData` function, crop the multispectral data using the `cropData` function, select desired spectral bands using the `selectBands` function, and resample spectral bands using the `resampleBands` function.
- Visualize the multispectral data using the `colorize` function.
- To load the entire multispectral data cube into memory, use the `gather` function.
- Perform block processing on the multispectral data using built-in or custom functionality using the `apply` function.

The Hyperspectral Viewer app has these new capabilities.

- Import multispectral data into the app directly from files or from `multicube` objects in the workspace.
- Explore the geospatial information of hyperspectral and multispectral data in the new **Raster Reference** pane of the app.

R2025a includes the new example Interactively Segment Hyperspectral Image Using Segment Anything Model..

To use these functions and tools, you must download the Hyperspectral Imaging Library for Image Processing Toolbox from the Add-On Explorer. For more information, see Get and Manage Add-Ons.

Compatibility Considerations

- The `hypercube` function will be removed in a future release. To create a `hypercube` object, use the `imhypercube` or `geohypercube` functions instead.
- The `hypercube` object no longer has the `DataCube` property.
- You can no longer read multispectral data using the `hypercube` object. Use the `multicube` object to read multispectral data.

Functionality being removed or changed

labelvolshow and volshow objects have been removed

Errors

The `labelvolshow` and `volshow` objects have been removed. Use the `viewer3d` and `volshow` functions instead.

im2java2d function has been removed

Errors

The `im2java2d` function has been removed. There is no replacement for this function.

Preferences window renamed to Settings window, new location for imshow settings

Behavior change

The Image Processing Toolbox Preferences window has been renamed to the Settings window, and preferences are now referred to as settings. See `iptprefs` for details.

Starting this release, access `imshow` settings under **MATLAB > Image Display**.

Interpolation property has no effect on bigimageshow

Behavior change

Setting the `Interpolation` property of a `bigimageshow` object no longer has an effect. All images are displayed using nearest neighbor interpolation. To display blocked images using nearest neighbor or bilinear interpolation, use the `imageshow` function.

R2024b

Version: 24.2

New Features

Bug Fixes

Compatibility Considerations

★Image Segmentation: Automatically or interactively segment image using Segment Anything Model

A new function and an enhanced app expand the support for using the Segment Anything Model (SAM).

- Use the `imsegsam` function to automatically segment all objects in an image and return masks and corresponding confidence scores.
- Use the SAM in the Image Segmenter app to interactively segment selected objects from an image, or to perform full image segmentation. For an example, see Segment Objects Using Segment Anything Model (SAM) in Image Segmenter.

To get started with the SAM for image segmentation, see Get Started with Segment Anything Model for Image Segmentation.

This functionality requires the Image Processing Toolbox Model for Segment Anything Model support package. You can install the support package from Add-On Explorer. For more information about installing add-ons, see Get and Manage Add-Ons.

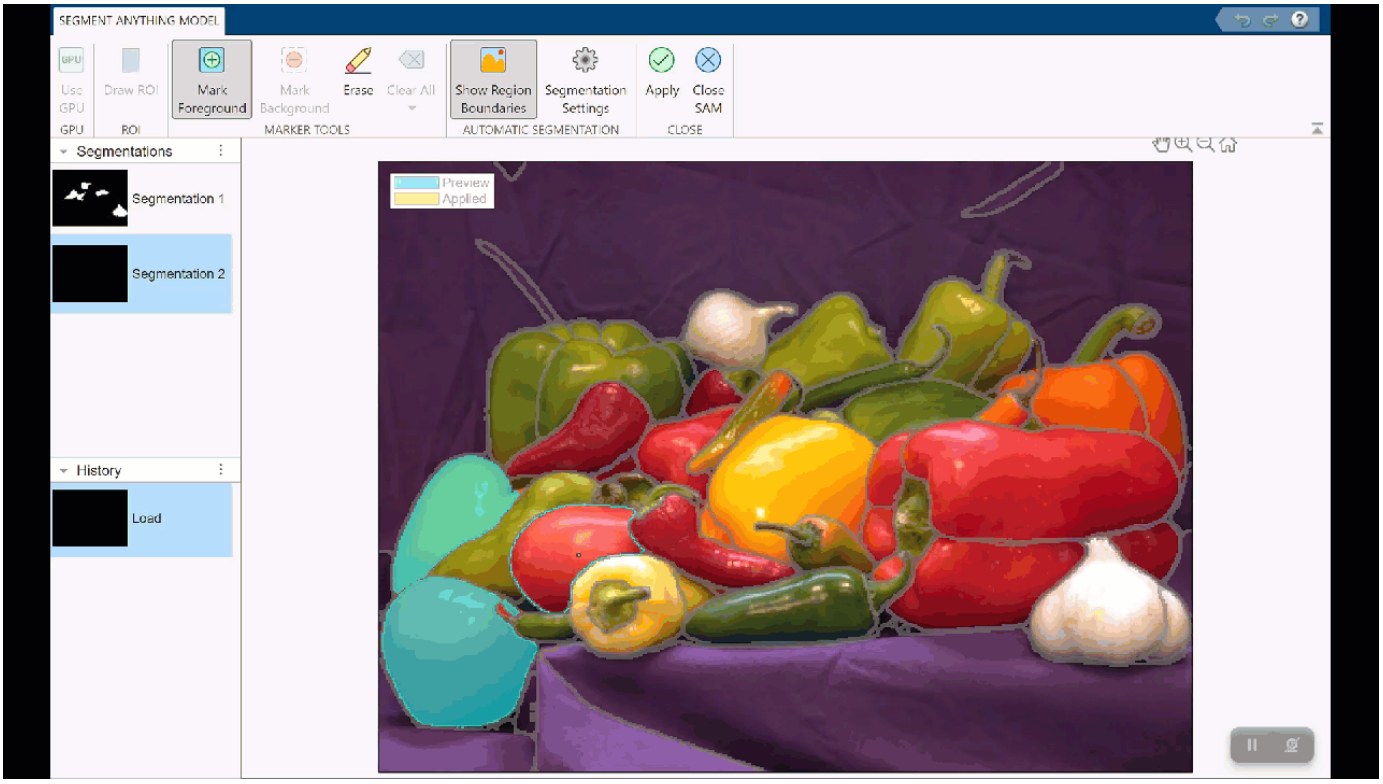


Image Segmentation: Segment 2-D grayscale, RGB, and multichannel images using the ISODATA algorithm

The `imsegisodata` function segments 2-D grayscale, RGB, and multichannel images using the ISODATA algorithm.

★Image Display: Efficiently display 2-D images and large images

New functions and objects offer enhanced performance and a unified workflow for displaying 2-D in-memory and large images.

- The `imageshow` function displays 2-D images and `blockedImage` objects. Control the appearance of the image display using `Image Properties`.
- The `viewer2d` function creates a viewer optimized for displaying 2-D images. Control the appearance of the viewer scene using `Viewer Properties`.

Volume Visualization: Renamed objects and classes for image display

To reflect support for new workflows, a group of objects and classes have been renamed.

This table shows the affected functionality and the new names. The behavior of these objects and classes remains the same, and existing instances of the old names in your code continue to work as expected. There are no plans to remove support for the old names.

Before R2024b	Starting in R2024b
`Viewer3D`	`Viewer`
`images.ui.graphics3d.Surface`	`images.ui.graphics.Surface`
`images.ui.graphics3d.Light`	`images.ui.graphics.Light`
`images.ui.graphics3d.roi.Point`	`images.ui.graphics.roi.Point`
`images.ui.graphics3d.roi.Line`	`images.ui.graphics.roi.Line`
`images.ui.graphics3d.events.AnnotationAddedEventData`	`images.ui.graphics.events.AnnotationAddedEventData`
`images.ui.graphics3d.events.AnnotationEditedEventData`	`images.ui.graphics.events.AnnotationEditedEventData`
`images.ui.graphics3d.events.AnnotationMovedEventData`	`images.ui.graphics.events.AnnotationMovedEventData`
`images.ui.graphics3d.events.AnnotationRemovedEventData`	`images.ui.graphics.events.AnnotationRemovedEventData`
`images.ui.graphics3d.events.CameraMovedEventData`	`images.ui.graphics.events.CameraMovedEventData`
`images.ui.graphics3d.events.ClippingPlanesChangedEventData`	`images.ui.graphics.events.ClippingPlanesChangedEventData`
`images.ui.graphics3d.events.SlicePlanesChangedEventData`	`images.ui.graphics.events.SlicePlanesChangedEventData`
`images.ui.graphics3d.events.CropRegionChangedEventData`	`images.ui.graphics.events.CropRegionChangedEventData`

Volume Visualization: New viewer cropping options

There are new options and `Viewer Properties` for cropping objects in a viewer.

- Crop the objects in a viewer to a spherical or cylindrical region, in addition to the rectangular crop region supported in prior releases. To add a spherical or cylindrical crop region, specify the

`CropRegion` property as a 4-element or 5-element row vector, respectively. Modify the crop region interactively in the viewer window.

- Specify whether to view the region inside or outside a crop region by specifying the new `CropMode` property.

Volume Visualization: Specify volume display range and transparency map, new default overlay colormap

Use these new `Volume Properties` and `BlockedVolume Properties` to specify the display range and transparency map for volume and overlay data:

- Control the display range used to scale the volume by using the new `DisplayRange` and `DisplayRangeMode` properties.
- Control the display range used to scale the volume overlay by using the new `OverlayDisplayRange` and `OverlayDisplayRangeMode` properties.
- Specify a uniform transparency for all nonzero label values while hiding background labels using the new `OverlayAlpha` property.
- For easier specification of transparency maps, the new options `"linear"`, `"quadratic"`, and `"cubic"` have been added for the `Alphamap` and `OverlayAlphamap` properties.

Compatibility Considerations

These `Volume` and `BlockedVolume` properties have changed in R2024b:

- To improve the display of discrete label data, the default `OverlayColormap` property value has changed from `turbo(256)` to a custom colormap that maximizes differences between adjacent colors. To apply the same colormap as in previous releases, specify the `Colormap` property as `turbo(256)`.
- To apply a uniform transparency while hiding background labels, the new `OverlayAlpha` property is recommended over the `OverlayThreshold` property. There are no plans to remove support for existing instances of `OverlayThreshold`.

The `BlockedVolume` object has these additional changes:

- To control the display range used to scale the volume, the new `DisplayRange` and `DisplayRangeMode` properties are recommended over the `DataLimits` and `DataLimitsMode` properties documented in prior releases. There are no plans to remove support for existing references to the old names.
- To control the display range used to scale the overlay, the new `OverlayDisplayRange` and `OverlayDisplayRangeMode` properties are recommended over the `OverlayDataLimits` and `OverlayDataLimitsMode` properties documented in prior releases. There are no plans to remove support for existing references to the old names.

Image Blending: Blend two 2-D grayscale or RGB images

The `imblend` function blends two 2-D grayscale or RGB images.

Image Quality Test Charts: Measure sharpness of Dead Leaves and Spilled Coins test charts

Measure the sharpness of detail in a Dead Leaves, Spilled Coins, or similar test chart using the `measureTexture` function.

Blocked Images: Support for reading 3-D TIFF images

Use the new `TIFF3D` adapter to read blocks of 3-D TIFF files in these formats:

- Image data stored in the file as individual Image File Directories (IFDs) of the same size and data type.
- Image data stored as large files greater than 4 GB, not of the BigTIFF format, created by ImageJ software.

montage Function: Display montage of deep learning arrays, and control thumbnail resizing method

The `montage` function now accepts `dlarray` input for deep learning applications.

The new `ThumbnailInterpolation` name-value argument enables you to control the interpolation method that `montage` uses to resize images for thumbnails.

⚠imregcorr Function: Improved registration algorithms, including support for gradient correlation

The `imregcorr` function now uses a new default method called gradient correlation.

The function implements an improved phase correlation algorithm that returns more accurate and consistent results. The results can differ from prior releases. For more information, see “Functionality being removed or changed” on page 4-7.

⚠Deep Learning: Updated functions to improve training process

Starting in R2024a, there is a new recommended workflow to build, train, and make predictions with neural networks using the `trainnet` (Deep Learning Toolbox) function and `dlnetwork` (Deep Learning Toolbox) objects.

To support the new deep learning workflow, these functions have been updated:

- `denoisingNetwork` now returns a network as a `dlnetwork` object.
- `dnCNNLayers` now returns an array of layers without a final regression layer.

⚠Compatibility Considerations

Before, the `denoisingNetwork` function returned the network as a `SeriesNetwork` (Deep Learning Toolbox) object, which is no longer recommended. If you use the output of the `denoisingNetwork` function as the input network for the `denoiseImage` function, then you do not need to update your code.

Before, the `dnCNNLayers` function returned a vector of layers with a `RegressionOutputLayer` (Deep Learning Toolbox) as the final layer, and you trained the custom denoising network using the `trainNetwork` (Deep Learning Toolbox) function. The `RegressionOutputLayer` object and the `trainNetwork` function are no longer recommended. To update your code, replace instances of the `trainNetwork` function with `trainnet`, and specify the third argument of `trainnet` as the loss function `"mse"`.

C Code Generation: Generate code from additional functions using MATLAB Coder

This list indicates the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. For a complete list of Image Processing Toolbox functions that support code generation, see Functions Supporting Code Generation.

- `imregcorr`
- `rgb2xyz`
- `SpaceToDepthLayer`
- `std2`
- `xyz2rgb`

GPU Code Generation: Generate CUDA code from additional functions using GPU Coder

The `imrotate3` and `montage` functions have been enabled for optimized CUDA code generation in this release.

GPU Acceleration for montage Function

R2024b adds GPU acceleration for the `montage` function. GPU acceleration requires a Parallel Computing Toolbox license and meeting the GPU computing requirements detailed here.

Thread-Based Environment: Run DICOM functions in a thread-backed pool

R2024b adds thread-based support for these Image Processing Toolbox functions and objects. Run code in the background on a single thread using MATLAB® `backgroundPool`, or accelerate code with multiple thread workers using `ThreadPool` (Parallel Computing Toolbox). Use of multiple thread workers requires Parallel Computing Toolbox.

- `dicomanon`
- `dicomCollection`
- `dicomContours` and its functions `addContour`, `convertToInfo`, `createMask`, and `deleteContour`
- `dicomdict`
- `dicomdisp`

- `dicomfind`
- `dicominfo`
- `dicomlookup`
- `dicomread`
- `dicomreadVolume`
- `dicomupdate`
- `dicomuid`
- `dicomwrite`
- `images.dicom.decodeUID`
- `images.dicom.parseDICOMDIR`

Renamed Support Package: Image Processing Toolbox Hyperspectral Imaging Library renamed to Hyperspectral Imaging Library for Image Processing Toolbox

As of R2024b, the Image Processing Toolbox Hyperspectral Imaging Library support package has been renamed to Hyperspectral Imaging Library for Image Processing Toolbox.

Hyperspectral Image Processing: Hyperspectral Imaging Library for Image Processing Toolbox (Version 24.2)

- Specify the block size to use when loading a hyperspectral data cube by using the `blockSize` name-value argument of the `hypercube` function.
- Plot endmembers and view the metadata of the hyperspectral image using the new capabilities of the Hyperspectral Viewer app.
- R2024b includes the new example Classify Land Cover Using Sentinel-2 Multispectral Images.

To use these functions and tools, you must download the Hyperspectral Imaging Library for Image Processing Toolbox from the Add-On Explorer. For more information, see Get and Manage Add-Ons.

⚠Functionality being removed or changed

imregcorr Function: Behavior changes resulting from improvements to registration algorithms

Behavior change

Prior to R2024b, the `imregcorr` function supported only the phase correlation method. Starting in R2024b, `imregcorr` uses a new default method called gradient correlation. To use the phase correlation method, specify the new `Method` name-value argument as `"phasecorr"`.

The function implements an improved phase correlation algorithm that returns more accurate and consistent results. The improvements can cause different behavior from prior releases, such as:

- A larger value for the peak correlation.
- A modified window when performing windowing in the frequency domain. The new behavior is most noticeable when the input images have a small number of rows or columns.

- More consistent windowing when estimating rigid and similarity transformations.

The function no longer issues a warning when the peak correlation value is less than 0.03.

Viewer3D renamed to Viewer, new default background color
Behavior change

- To reflect new support for 2-D image display, the `Viewer3D` object is now the `Viewer` object. The `viewer3d` function now returns a `Viewer` object. There are no plans to remove support for references to `Viewer3D`. For details, see `Viewer Properties`.
- The default background color for a viewer created using `viewer3d` in light mode has changed from `[0 0.329 0.529]` to `[0 0.251 0.451]`. The default background gradient color in light mode has changed from `[0.0 0.561 1.0]` to`[0.0667 0.4431 0.7451]`.

denoisingImageSource function has been removed

The `denoisingImageSource` function has been removed. Use the `denoisingImageDatastore` function instead.

R2024a

Version: 24.1

New Features

Bug Fixes

Compatibility Considerations

★Image Segmentation: Segment objects in images using Segment Anything Model

Interactively segment objects in images using deep learning. Create a `segmentAnythingModel` object to configure a pretrained Segment Anything Model (SAM). Generate learned feature embeddings by running the encoder on the input images using the `extractEmbeddings` object function. Segment images using the `segmentObjectsFromEmbeddings` object function. To get started, see Get Started with Segment Anything Model for Image Segmentation. For an example, see Interactively Segment Image Using Segment Anything Model.

These functions require the Image Processing Toolbox Model for Segment Anything Model support package. You can install the support package from Add-On Explorer. For more information about installing add-ons, see Get and Manage Add-Ons.

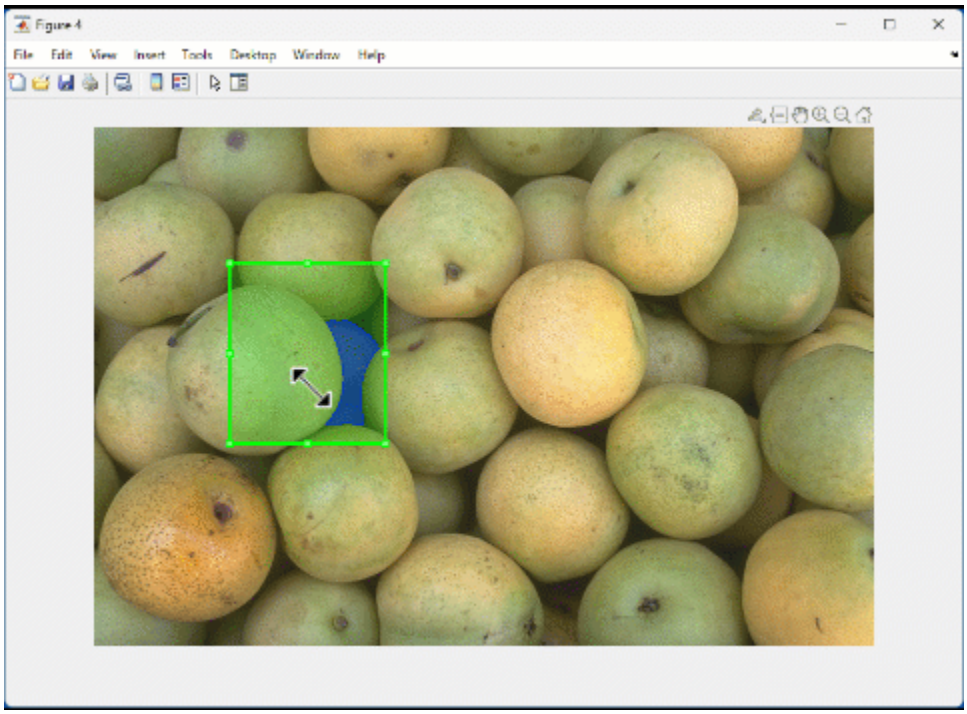


Image Quality: Analyze additional test charts

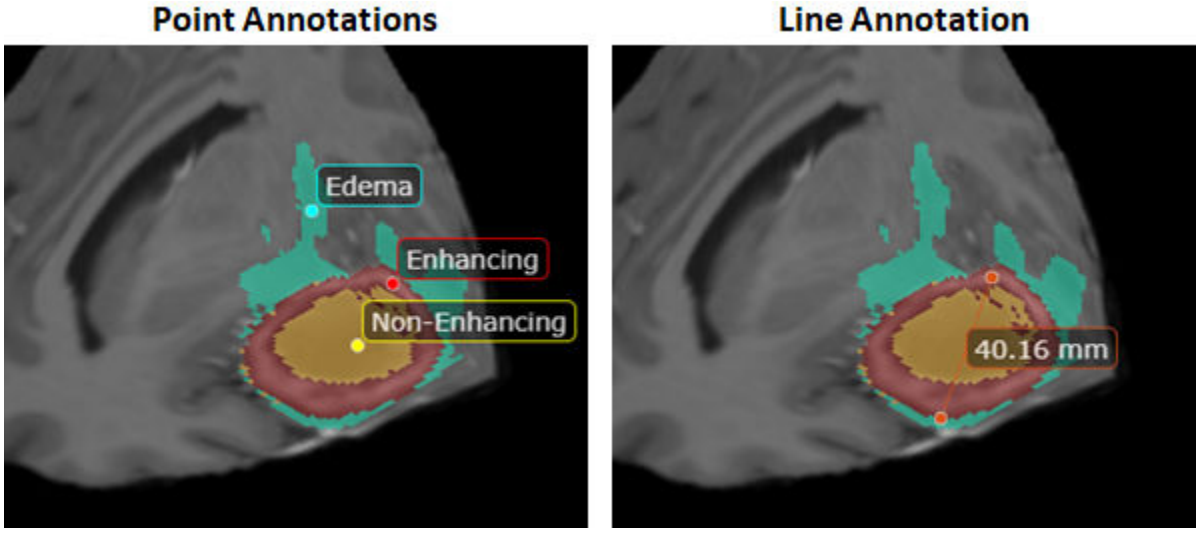
You can calculate image quality metrics, such as sharpness and color reproduction, using additional types of test charts. These functions now support a broader variety of test charts:

- `measureSharpness`
- `measureColor`
- `measureChromaticAberration`
- `measureIlluminant`
- `measureNoise`

Volume Visualization: Label volumes using 3-D point and line annotations

Label 3-D points and lines in a `Viewer3D` scene using annotations. Add annotations interactively from the `Viewer3D` toolbar. By default, the viewer labels point annotations with their 3-D coordinates and line annotations with their length. For an example, see Label Volume Display Using Point and Line Annotations.

Programmatically query or update annotations using the new `Annotations` property of `Viewer3D`. The new `Point` and `Line` objects and their properties define the position, color, and label of point and line annotations, respectively.



Object Analysis: Create binary masks from circles in image

The new `circles2mask` function creates binary mask images of circles from defined circle radii and centers. You can specify the circle radii and centers using the `radii` and `centers` outputs of the `imfindcircles` function.

Object Analysis: Measure circularity using Image Region Analyzer app and bwpropfilt function

You can measure the circularity of regions in binary images using the Image Region Analyzer app and the `bwpropfilt` function.

Image Region Analyzer includes circularity measurements by default. To measure circularity using the `bwpropfilt` function, include `"Circularity"` when specifying the `properties` argument.

Object Analysis: Filter properties of connected components structure and display result

The `bwpropfilt` function can filter objects that you specify using a connected components structure. To filter on multiple image properties in an efficient way, call `bwpropfilt` multiple times using the same connected components structure as input and output for each call.

The new `cc2bw` function converts a connected components structure to a binary image. Use the `cc2bw` function to display the result of filtering objects by properties.

NIfTI File Format: Simplified workflow for writing compressed files

Starting this release, `niftiwrite` automatically writes a compressed NIfTI file when you specify a filename with the extension `.nii.gz`, without requiring you to specify the `Compressed` name-value argument. For example, this code writes the volume data in `V` to a compressed file:

```
niftiwrite(V,"volume.nii.gz")
```

Before R2024a, you must specify the `Compressed` name-value argument as `true` to write a compressed file, even if the filename includes the extension `.nii.gz`.

Deep Learning: Updated functions and examples to improve training process

Starting in R2024a, there is a new recommended workflow to build, train, and make predictions with neural networks that uses the `trainnet` (Deep Learning Toolbox) function (introduced in R2023b) and `dlnetwork` (Deep Learning Toolbox) objects.

The `denoiseImage` function now supports networks specified as `dlnetwork` objects.

These examples were updated to train networks formatted as `dlnetwork` objects using the `trainnet` function.

- 3-D Brain Tumor Segmentation Using Deep Learning
- Brighten Extremely Dark Images Using Deep Learning
- Classify Defects on Wafer Maps Using Deep Learning
- Classify Tumors in Multiresolution Blocked Images
- Develop Camera Processing Pipeline Using Deep Learning
- Image Processing Operator Approximation Using Deep Learning
- Increase Image Resolution Using Deep Learning
- JPEG Image Deblocking Using Deep Learning
- Quantify Image Quality Using Neural Image Assessment

This example was updated to train the network formatted as a `dlnetwork` object. The example still uses a custom training loop.

- Neural Style Transfer Using Deep Learning

C Code Generation: Generate code from additional functions using MATLAB Coder

This list indicates the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. For a complete list of Image Processing Toolbox functions that support code generation, see Functions Supporting Code Generation.

- `activecontour`
- `bwareafilt`
- `cc2bw`
- `dlresize`
- `imlocalbrighten`
- `imresize3`
- `imrotate3`

The `imwarp` function now supports code generation for 3-D linear geometric transformations.

The `bwconncomp` function now supports code generation for any connectivity that is a compile-time constant.

GPU Code Generation: Generate CUDA code from dlresize function using GPU Coder

The `dlresize` function has been enabled for optimized CUDA code generation in this release.

★Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (Version 24.1)

- Load and process cropped regions of large hyperspectral images using the new capabilities of the `hypercube` object. For more information, see the Read Small Regions of Large Hyperspectral Image.
- Use the `resampleSignature` and `removeContinuum` functions to preprocess hyperspectral data for applications such as spectral matching and target detection.
- Use the `detectTarget` function to detect targets in hyperspectral images based on the spectral signature of the target and statistical distribution of the background clutter.
- Use the `hyperseganchor` function to segment hyperspectral images using fast spectral clustering with anchor graphs.
- You can now specify individual dark pixel values to subtract from each pixel of the input hyperspectral image by specifying a 3-D numeric array to the `darkPixels` argument of the `subtractDarkPixel` function.
- R2024a includes these new examples:
  - Predict Sugar Content in Grape Berries Using PLS Regression on Hyperspectral Data
  - Classify Land Cover Using Hyperspectral and Lidar Data
  - Segment Spleen in Hyperspectral Image of Porcine Tissue
  - Map Flood Areas Using Sentinel-1 SAR Imagery

To use these functions and tools, you must download the Image Processing Toolbox Hyperspectral Imaging Library from the Add-On Explorer. See Get and Manage Add-Ons.

⚠Functionality being removed or changed

hyperSlic will become hyperslic
Behavior change

Starting in R2024a, the `hyperSlic` function in the Image Processing Toolbox Hyperspectral Imaging Library is named `hyperslic`.

R2023b

Version: 23.2

New Features

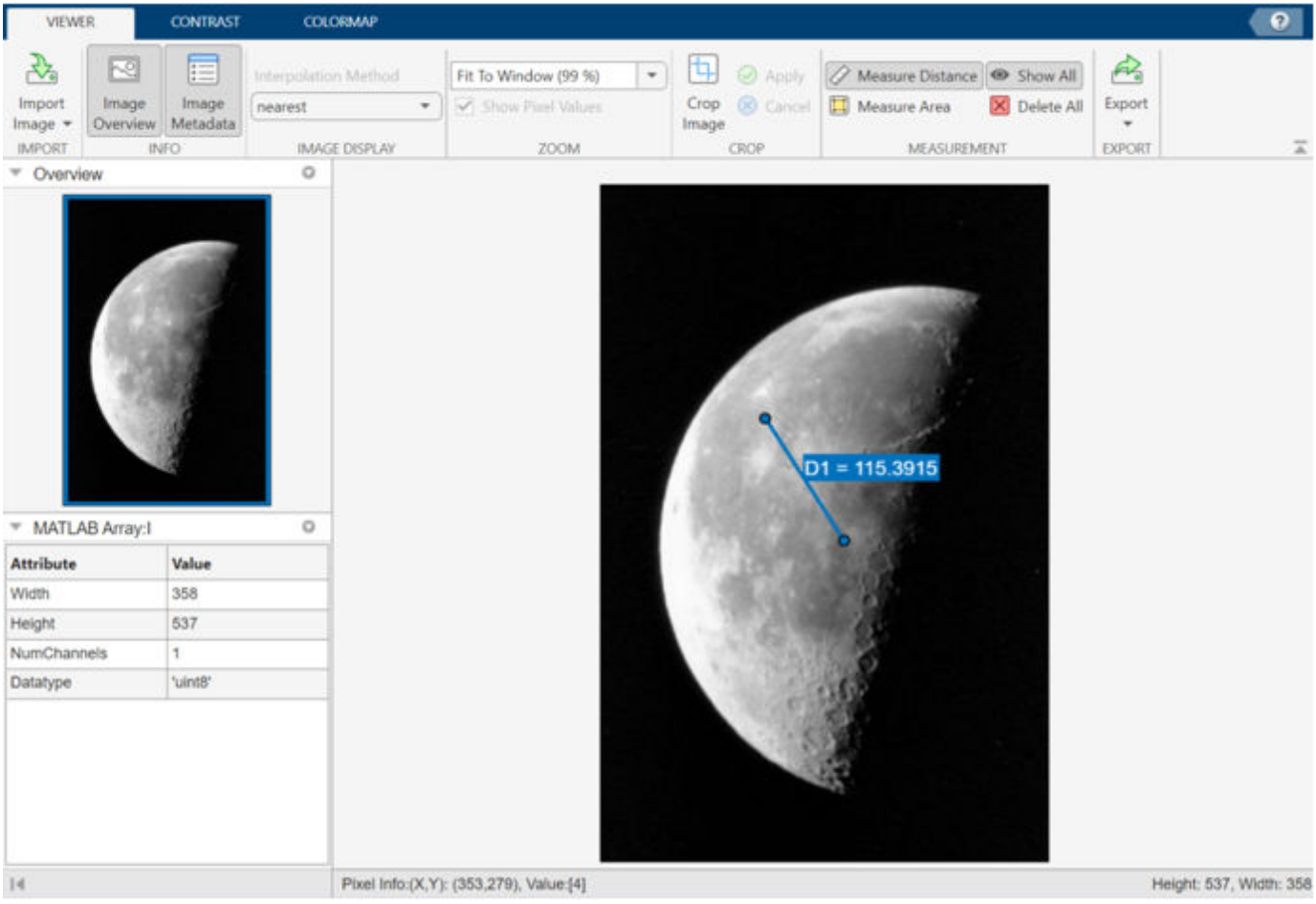
Bug Fixes

Compatibility Considerations

Image Display and Exploration: Enhancements to Image Viewer app

The enhanced Image Viewer app provides more functionality and better ease of use. For example, you can now use the app to interactively draw polygon ROIs and measure their areas. Open the app using the new `imageViewer` function.

The enhanced Image Viewer app and the `imageViewer` function are recommended over the `imtool` function, which will be removed in a future release. For more information, see “Functionality being removed or changed” on page 6-5.



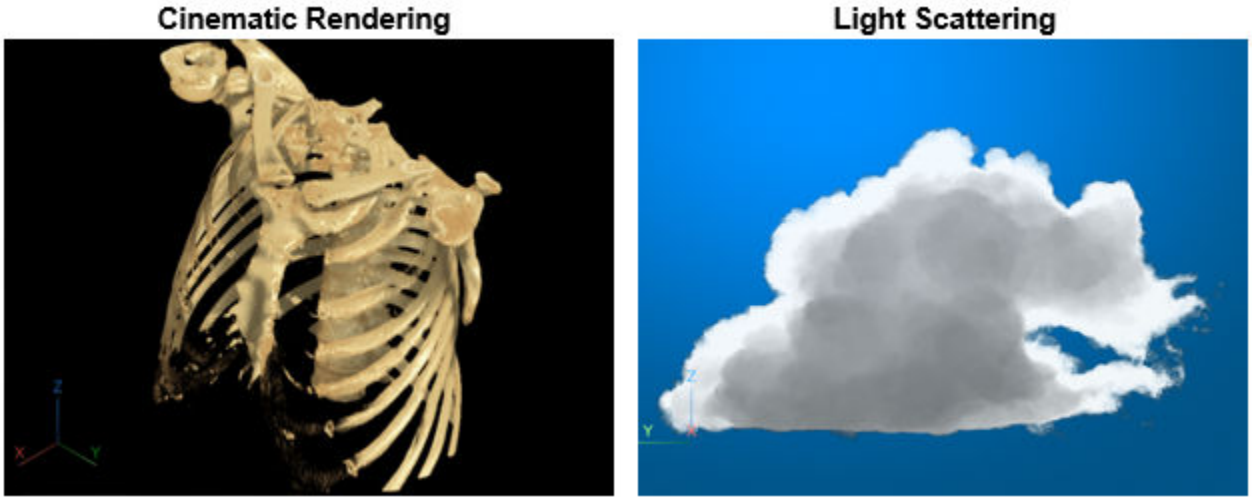
Volume Visualization: Cinematic rendering, volumetric light scattering, advanced lighting controls, and interactive cropping

The `Volume` object has new properties and rendering styles for displaying 3-D image volumes. To set property values at object creation, specify them to the `volshow` function as name-value arguments.

- Control the amount of light reflected by a volume using the new `SpecularReflectance` property. Increase the specular reflectance to make a volume appear shinier.
- Specify the `RenderingStyle` property as `"CinematicRendering"` to display a photorealistic volume using iterative postprocessing. Specify the number of postprocessing iterations using the

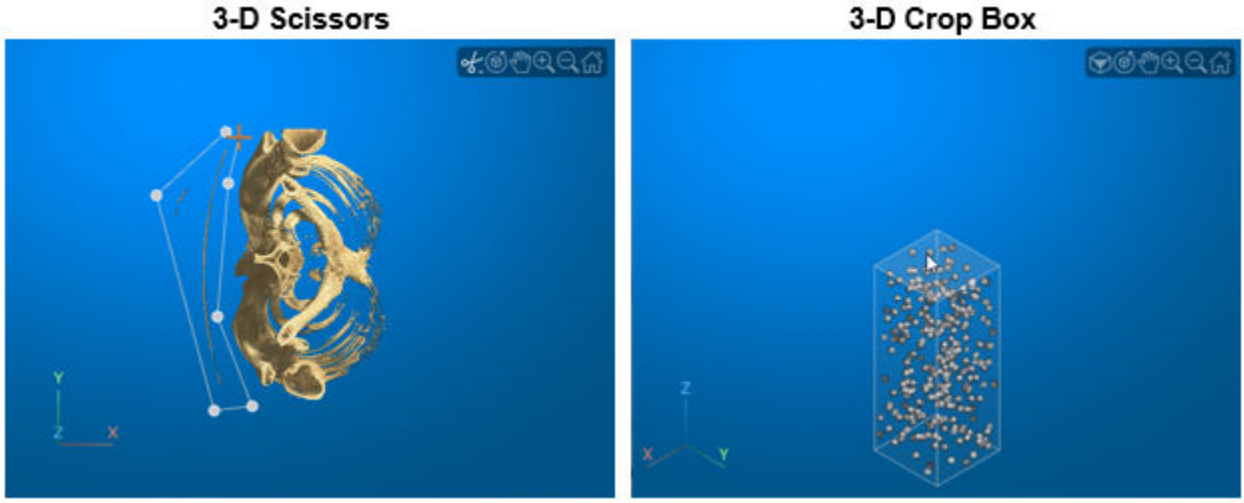
new `CinematicNumIterations` property. For an example, see Display Volume Using Cinematic Rendering.

- Specify the `RenderingStyle` property as `"LightScattering"` to display the object with volumetric light scattering, including light absorption, inscattering, and outscattering. Specify the balance between rendering quality and speed using the new `LightScatteringQuality` property. For an example, see Display Translucent Volume with Advanced Light Scattering.



The `Viewer3D` object has new properties and tools to control the lighting and remove objects in a scene. To set properties at object creation, specify them to the `viewer3d` function as name-value arguments.

- Control the strength of ambient and diffuse light within a viewer using the new `AmbientLight` and `DiffuseLight` properties, respectively.
- Apply denoising to the objects in a viewer using the new `Denoising` property. Specify the amount of smoothing and standard deviation of the Gaussian smoothing kernel using the new `DenoisingDegreeOfSmoothing` and `DenoisingSigma` properties, respectively.
- Use a crop box to crop the objects in a viewer to a rectangular subregion. For an example, see the Crop Blocked Volume section of Display Large 3-D Images Using Blocked Volume Visualization.
- Use the 3-D scissors tool to remove regions from of a viewer. For an example, see Remove Objects from Volume Display Using 3-D Scissors.



Border selection and deletion: Remove or retain specified borders and border structures

The `imclearborder` function has a new `Borders` name-value argument that enables you to specify which image borders to remove or from which to remove connected structures.

The `imkeepborder` function retains only the light structures in an image that are connected to the image borders. You can specify a subset of image borders which to retain or for which to retain structures.

imclearborder: Improved performance

The `imclearborder` function shows improved performance. For example, in this code, the call to `imclearborder` is about 1.7x faster than in the previous release.

```
function t = imclearborderTimingTest
    A = imbinarize(imread("rice.png"));
    f = @() imclearborder(A);
    t = timeit(f);
end
```

The approximate execution times are:

R2023a: 0.85 ms

R2023b: 0.51 ms

The code was timed on a macOS 12.5.1, Intel® Core i9 CPU @ 3.6 GHz test system.

C Code Generation: Generate code from additional functions using MATLAB Coder

This list indicates the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. For a complete list of Image

Processing Toolbox functions that support code generation, see Functions Supporting Code Generation.

- `axes2pix`
- `dct2`
- `decorrstretch`
- `idct2`
- `imimposemin`
- `roifilt2`
- `roipoly`

The `edge` function now supports code generation for the `"approxcanny"` edge detection method.

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (Version 23.2)

- Read hyperspectral data from `.bsq`, `.bil`, `.bip`, and multipage TIFF files using the new capabilities of the `hypercube` object.
- View statistical data of spectral signatures, and export the spectral signatures using the new capabilities in the Hyperspectral Viewer app.
- Use the `hyperSlic` function to perform 2-D superpixel oversegmentation of hyperspectral cubes using the simple linear iterative clustering (SLIC) algorithm.
- The Change Detection in Hyperspectral Images example shows how to detect changes in land cover from hyperspectral images taken at different times.
- The Ship Detection from Sentinel-1 C Band SAR Data Using YOLO v2 Object Detection example shows how to detect ships from Sentinel-1 C Band SAR Data using YOLO v2 object detection.
- The Automate Pixel Labeling of Hyperspectral Images Using ECOSTRESS Spectral Signatures in Image Labeler example shows how to load hyperspectral images into the Image Labeler (Computer Vision Toolbox) and automatically label pixels.

To use these functions and tools, you must download the Image Processing Toolbox Hyperspectral Imaging Library from the Add-On Explorer. See Get and Manage Add-Ons.

Functionality being removed or changed

imtool will be removed

Still runs

The `imtool` function will be removed in a future release. In most situations, use the Image Viewer app instead. The app has more functionality and is easier to use than the Image Tool. To update your code, replace instances of `imtool` with `imageViewer`.

Unlike Image Tool, the Image Viewer app does not return a figure. If you want to display an image in Image Tool and return the figure that contains the tool, use the `images.compatibility.imtool.r2023b.imtool` function instead. To update your code that displays an image and returns the Image Tool figure, replace instances of `imtool` with `images.compatibility.imtool.r2023b.imtool`.

Discouraged Usage	Recommended Replacement

This example uses the `imtool` function to display a grayscale image in the Image Tool.
 	
`I = imread("cameraman.tif");`
`imtool(I)`
	Here is equivalent code that uses the `imageViewer` function to display the grayscale image in the Image Viewer app.
	
	`I = imread("cameraman.tif");`
	`imageViewer(I)`

This example uses the `imtool` function to display a grayscale image in the Image Tool, setting the initial display range and interpolation method.
 	
`I = imread("cameraman.tif");`
`imtool(I,[5 250],Interpolation="bilinear")`
	Here is equivalent code that uses the `imageViewer` function to display the grayscale image and set the initial display range and interpolation method. To set the display range, you must use the `DisplayRange` name-value argument.
	
	`I = imread("cameraman.tif");`
	`imageViewer(I,DisplayRange=[5 250],Interpolation="bil`

This example uses the `imtool` function to display an indexed image in the Image Tool, specifying the colormap.
 	
`[I,map] = imread("trees.tif");`
`imtool(I,map)`
	Here is equivalent code that uses the `imageViewer` function to display the indexed image. To specify the colormap, you must use the `Colormap` name-value argument.
	
	`[I,map] = imread("trees.tif");`
	`imageViewer(I,Colormap=map)`

This example uses the `imtool` function to display a grayscale image in the Image Tool and returns the figure containing the Image Tool.
 	
`I = imread("cameraman.tif");`
`hTool = imtool(I);`
	Here is equivalent code that uses the `images.compatibility.imtool.r2023b.imtool` function to display a grayscale image in the Image Tool and return the figure containing the Image Tool.
	
	`I = imread("cameraman.tif");`
	`hTool = images.compatibility.imtool.r2023b.imtool(I);`

bigimage and bigimageDatastore will be removed
Still runs

The `bigimage` object and the `bigimageDatastore` object will be removed in a future release. To read and process large or multiresolution images, use the `blockedImage` object and the `blockedImageDatastore` object instead.

Discouraged Usage	Recommended Replacement
This example creates a `bigimage` object. `bigIm = bigimage("tumor_091R.tif");`	Here is equivalent code, replacing the `bigimage` object with the `blockedImage` object. `blockedIm = blockedImage("tumor_091R.tif");`
This example creates a `bigimageDatastore` object from the `bigimage` object `bigIm`. `bigImds = bigimageDatastore(bigIm);`	Here is equivalent code that creates a `blockedImageDatastore` object from the `blockedImage` object `blockedIm`. `blockImds = blockedImageDatastore(blockedIm);`

Continue to display large or multiresolution image data using the `bigimageshow` function.

R2023a

Version: 11.7

New Features

Bug Fixes

Compatibility Considerations

Volume Visualization: Display blocked volumes

The `volshow` function now supports displaying large image volumes stored as a `blockedImage` object. When you pass a `blockedImage` object as input, `volshow` displays the volume and creates a `BlockedVolume` object. The `BlockedVolume` object controls the appearance and behavior of the blocked image volume within a 3-D scene. To get started, see Display Large 3-D Images Using Blocked Volume Visualization.

Blocked Images: Add levels to large images, combine block locations, and other enhancements

Several enhancements to blocked images enable streamlined workflows when working with blocked images.

- Create multilevel blocked images using the `makeMultiLevel2D` and `makeMultiLevel3D` functions. These functions are particularly useful when you want to create overview levels for a very large image.
- Combine the levels of two or more blocked images using the `concatenateLevels` function. This function supports combining data in different formats and in different locations. To efficiently add overview levels to a very large image on disk, you can create the overview levels in memory, and then combine the original image and the overview levels using `concatenateLevels`.
- Merge the locations of selected blocks from two arrays of blocked images using the `mergeBlockLocationSets` function. Use this function when you want to combine block locations selected from a set of images using different criteria, or when you want to combine block locations selected from two different sets of images.
- Use the `JPEG2000` adapter to read blocks of a single JPEG2000 file.
- Write all blocks of a blocked image datastore to files using the `writeall` function.
- When you display 2-D blocked images using the `bigimageshow` object, you can add listeners for the new `DataReadStarted` and `DataReadFinished` events.
- When you get or specify pixel coordinates using the `getRegion`, `sub2world`, or `world2sub` functions, you can now specify the pixel or world coordinates using a lower dimensionality than that of the blocked image.

Image Batch Processor App: Support for image datastores and other enhancements

The Image Batch Processor app includes several new capabilities:

- Image datastore support — Load a batch of images by importing an `imageDatastore` object from the MATLAB workspace.
- Support for additional file formats — Load images from file formats including `RAW` and `EXR`.
- Additional command line syntaxes — Open the Image Batch Processor app and load images using two new syntaxes.

  This syntax opens the app and loads the batch of images stored in the specified folder `folderLocation`.

  ```
  imageBatchProcessor(folderLocation)
  ```

  This syntax opens the app and loads the images in the `imageDatatore` object `imds`.

```
imageBatchprocessor(imds)
```

- You can now optionally pass a second input argument, `info`, to your batch processing function. The `info` argument contains information about the image source, including the image filename. For an example that uses the `info` argument, see Process Images Using Image Batch Processor App with File Metadata.

⚠Color Thresholder App: Enhanced in-app image acquisition and updated graphics requirements

Starting in R2023a, you can more easily retake images acquired in the Color Thresholder using the new Import From Camera window. To open the window, on the app toolstrip, select **Load Image > Acquire Image From Camera**. In the window, you can adjust camera settings, capture an image by clicking **Capture**, and then choose to **Retake** or **Accept** the image. Once you accept an image, the window closes and the accepted image is loaded into the **Choose a Color Space** tab. Previously, you needed to reopen the Image Capture tab each time you wanted to retake an image.

⚠Compatibility Considerations

The app now requires hardware-accelerated graphics rendering for some features. For more information, see “Functionality being removed or changed” on page 7-4.

bwboundaries function: Trace exterior boundaries

The `bwboundaries` function has two new name-value arguments.

- The `TraceStyle` name-value argument adds support for tracing the boundary along the exterior edge of boundary pixels. Before, the function traced the boundary through only the centers of boundary pixels.
- The `CoordinateOrder` name-value argument adds support for returning boundary coordinates in (*x*, *y*) order. Before, the function returned boundary coordinates in only (*y*, *x*) order.

iradon function: Additional support for data type single

The `iradon` function now supports projection angles `theta` of data type `single`. When you specify `theta` as data type `single`, the `iradon` function returns the frequency response `H` as data type `single`.

Preferences in MATLAB Online: Update preferences interactively using the MATLAB Preferences window

Before R2023a, in MATLAB Online™, you could only access and update toolbox preferences programmatically using the `iptgetpref` and `iptsetpref` functions, respectively. You could not set toolbox preferences interactively.

Starting in R2023a, in MATLAB Online, you can additionally access and set toolbox preferences interactively using the MATLAB Preferences window. Open the Preferences window using the `iptprefs` function.

- Access display settings for the `imshow` function under **MATLAB>Image Display**.

- Access display settings for the Image Viewer app, as well as an option for enabling hardware optimizations, under **Image Processing Toolbox**.

There is no change to the MATLAB Preferences window in MATLAB desktop. For details, see `iptprefs`.

C Code Generation: Generate code from additional functions using MATLAB Coder

This list indicates the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. For a complete list of Image Processing Toolbox functions that support code generation, see Functions Supporting Code Generation.

- `corr2`
- `deconvlucy`
- `entropy`
- `entropyfilt`
- `rangefilt`
- `regionprops3`

The `bwconncomp` function now supports input images of any dimension. The function can now generate code that returns the `PixelIdxList` field in the `CC` structure.

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (Version 23.1)

- Use the `customSpectralIndex` function to define and compute an index specific to your application. You can use a custom spectral index to ease a workflow where standard hyperspectral indices are not applicable. You can also use a custom spectral index to fine-tune commonly used spectral indices to your specific hyperspectral data.
- Visualize standard and custom spectral indices of hyperspectral data and export color bands, spectral indices and masks to the workspace using the new capabilities in the Hyperspectral Viewer app.
- Support for hyperspectral data from the AVIRIS sensor.
- The Manually Label ROIs in Multispectral Image example shows how to manually select regions of interest (ROIs) from a multispectral image and save them in a shapefile using the Mapping Toolbox.

To use these functions and tools, you must download Image Processing Toolbox Hyperspectral Imaging Library from the Add-On Explorer. See Get and Manage Add-Ons.

⚠Functionality being removed or changed

The regionprops function uses new equation for circularity
Behavior change

Starting in R2023a, the `regionprops` function uses a new equation to calculate circularity. The new equation removes a bias that caused the computed circularity to be too high for relatively small objects.

If you wants to reproduce circularity measurements using the old equation, then you can use these commands:

```
props = regionprops(I,["Area" "Perimeter"]);
oldCircularity = 4*pi*props.Area ./ props.Perimeter.^2;
```

MATLAB Online limits maximum image display resolution
Behavior change

Starting in R2023a, MATLAB Online limits image display resolution to improve rendering speed for large images. This behavior affects the onscreen display, but it does not affect the image data. Before displaying large images, MATLAB Online resizes the largest dimension to a maximum size of 512 pixels for 2-D images and 256 voxels for 3-D image volumes. The smaller dimensions adjust to preserve the aspect ratio of the image or volume. To view images at their full resolution, use MATLAB desktop.

For the `imshow` function only, you can display images at their full resolution online by setting the `MaxRenderedResolution` property of the output `Image` object to `"none"`. For details about the `MaxRenderedResolution` property, see Image Properties.

Color Thresholder App: Updated graphics system requirements
Behavior change

Starting in R2023a, some features within the Color Thresholder app require hardware-accelerated graphics rendering. If MATLAB is using a software-rendering implementation, the app disables these features:

- Visualization of color space values as point clouds.
- Interactive drawing using the freehand ROI and polygon ROI tools.

To determine which implementation MATLAB is using, call the `rendererinfo` function. For details about graphics rendering in MATLAB, see System Requirements for Graphics.

R2022b

Version: 11.6

New Features

Bug Fixes

Compatibility Considerations

Geometric Transformations: Perform geometric transformations using premultiply matrix convention

Starting in R2022b, Image Processing Toolbox functions use a premultiply convention to create and perform geometric transformations.

A new set of objects enables geometric transformations using a premultiply convention.

Transformation Type	2-D Object	3-D Object
Affine transformation	`affinetform2d`	`affinetform3d`
Rigid transformation	`rigidtform2d`	`rigidtform3d`
Similarity transformation	`simtform2d`	`simtform3d`
Translation transformation	`transltform2d`	`transltform3d`
Projective transformation	`projtform2d`	Not applicable

The new `fitgeotform2d` function estimates a 2-D geometric transformation that maps pairs of control points between two images.

Several functions are updated to use the premultiply convention. These functions accept the new geometric transformation objects as input, or return the objects as output:

- `affineOutputView`
- `imregcorr`
- `imregister`
- `imregtform`
- `imwarp`
- `randomAffine2d`
- `randomAffine3d`
- `Warper`

Compatibility Considerations

The old geometric transformation objects that use a postmultiply convention, such as `affine2d`, are not recommended. You can streamline your geometric transformation workflows by switching to the new objects that use the premultiply convention. For more information, see Compatibility Considerations on page 8-5.

Volume Visualization: Enhanced volume, surface, and scene rendering

New functions and objects offer enhanced visual quality and interactive performance, as compared to functions and objects in previous releases, when displaying and interacting with 3-D volumes and surfaces.

- A `Viewer3D` object controls the appearance and behavior of a 3-D scene. Create a `Viewer3D` object using the `viewer3d` function.

- A `Volume` object controls the appearance and behavior of a volume within a 3-D scene. Create a `Volume` object using the `volshow` function.
- A `Surface` object controls the appearance and behavior of a 3-D surface within a 3-D scene.

⚠Compatibility Considerations

The `volshow` function now creates a `Volume` object instead of a `volshow` object. For more information, see Compatibility Considerations on page 8-5.

EXR Files: Read and write images stored in the EXR file format

The EXR format stores image data according to the OpenEXR file specification. To work with EXR files, use these functions:

- `isexr` — Check whether a file is a valid EXR file.
- `exrinfo` — Read information about image data from EXR files.
- `exrread` — Read image data from EXR files.
- `exrwrite` — Write image data to EXR file.
- `exrHalfAsSingle` — Convert numeric values into half-precision values.

Color Conversion: Support for ProPhoto (ROMM RGB) color space

These functions now support the ProPhoto color space: `lin2rgb`, `rgb2lin`, `lab2rgb`, `rgb2lab`, `rgb2xyz`, `xyz2rgb`, and `chromadapt`. To use the ProPhoto color space, specify the `ColorSpace` name-value argument as `"prophoto-rgb"`. The ProPhoto color space is also known as the Reference Output Medium Metric (ROMM) RGB color space, and has a larger gamut than the sRGB and Adobe 1998 RGB color spaces.

DICOM: Support for reading video data

The `dicomread` function now supports reading DICOM files that contain video data with MPEG-2, MPEG-4/H.264 and HEVC/H.265 encoding. Because MATLAB decodes video data using the codecs installed on your computer, some files might not able to be read on all platforms. For more details, see Supported Video and Audio File Formats.

viscircles Function: Accepts scalar radii

`viscircles` accepts a scalar value for the `radii` argument. When `radii` is a scalar, `viscircles` draws all circles with the same radius.

locallapfilt Function: Improved performance

The `locallapfilt` function shows improved performance. For example, this code is about 6.8x faster than in the previous release.

```
function locallapfiltTimingTest
      A = imread("peppers.png");
      locallapfilt(A,0.4,0.5);
end
```

The approximate execution times are:

R2022a: 0.254 s

R2022b: 0.037 s

The code was timed on a Windows® 10, Intel Xeon® E5-2683 v4 CPU @ 2.10 GHz test system (two processors) using the `timeit` function:

```
timeit(@locallapfiltTimingTest)
```

C Code Generation: Generate code from additional functions using MATLAB Coder

The list indicates the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. For a complete list of Image Processing Toolbox functions that support code generation, see Functions Supporting Code Generation.

- `bwpropfilt`
- `iradon`
- `psf2otf`
- `radon`

The `edge` function now supports code generation for the directional gradient output arguments, `Gx` and `Gh`.

GPU Code Generation: Generate CUDA code from additional functions using GPU Coder

The list indicates the Image Processing Toolbox functions that have been enabled for optimized CUDA code generation in this release.

- `adaptthresh`
- `histeq`
- `label2rgb`

GPU Acceleration for gray2ind Function

R2022b adds GPU acceleration for the `gray2ind` function. GPU acceleration requires a Parallel Computing Toolbox license and meeting the GPU computing requirements detailed here.

Thread-Based Environment: Run functions in a thread-backed pool

R2022b adds thread-based support for the Image Processing Toolbox functions listed in the table. Run code in the background on a single thread using MATLAB `backgroundPool`, or accelerate code with multiple thread workers using `ThreadPool`. Use of multiple thread workers requires Parallel Computing Toolbox.

adapthisteq	bfscore	bwconncomp	bwdist
bwferet	bwlabel	bwlabeln	bwlookup
bwmorph3	bwpack	bwskel	bwtraceboundary
bwunpack	demosaic	edge	edge3
entropyfilt	exrHalfAsSingle	exrinfo	exrread
exrwrite	fibermetric	grabcut	histeq
hough	imabsdiff	imapplymatrix	imbothat
imboxfilt	imboxfilt3	imclose	imdiffusefilt
imdilate	imerode	imfill	imfilter
imgaussfilt	imgaussfilt3	imgradient	imgradient3
imhist	imlincomb	imnlmfilt	imopen
imreconstruct	imregionalmax	imregionalmin	imsegfmm
imtophat	inpaintCoherent	inpaintExemplar	integralImage
integralImage3	intlut	iradon	isexr
label2idx	medfilt2	medfilt3	modefilt
multissim	multissim3	nitfread	obliqueslice
ordfilt2	planar2raw	poly2mask	qtdecomp
radon	raw2planar	raw2rgb	rawread
regionprops	regionprops3	rgb2lab	rgb2lightness
superpixels	superpixels3	watershed	

Functionality being removed or changed

Postmultiply geometric transformation objects are not recommended
Still runs

These geometric transformation objects are not recommended because they use the postmultiply matrix convention: `affine2d`, `affine3d`, `rigid2d`, `rigid3d`, and `projective2d`. There are no plans to remove the old geometric transformation objects at this time. However, you can streamline your geometric transformation workflows by switching to new objects that use the premultiply convention. For more information, see Migrate Geometric Transformations to Premultiply Convention.

volshow function creates Volume object
Behavior change

The `volshow` function now creates a `Volume` object instead of a `volshow` object. The `Volume` object offers more rendering styles and integrates with a `Viewer3D` object to offer easier control of the volume visualization. The `Volume` object also supports web graphics.

The `volshow` function accepts a different set of name-value arguments based on the properties of the `Volume` object. For a list of these properties, see `Volume`.

When you call `volshow` without specifying a parent object, the function now creates a new `Viewer3D` object and sets that object as the parent. Before, the function identified the current figure using the `gcf` function and set that figure as the parent.

If you want to reproduce the prior behavior, then use the `images.compatibility.volshow.R2022a.volshow` function to create a `volshow` object.

`labelvolshow` and `images.compatibility.volshow.R2022a.volshow` will be removed
Still runs

The `labelvolshow` object and the `images.compatibility.volshow.R2022a.volshow` function will be removed in a future release. Use the `viewer3d` and `volshow` functions instead. The `viewer3d` function creates a `Viewer3D` object that you can use to modify the scene, such as the camera, background colors, and lighting. The `volshow` object creates a `Volume` object that you can use to modify the appearance of the volumetric data, such as the rendering style, colormap, and transparency map. Specify and change the labels and intensity data by setting the `OverlayData` and `Data` properties of the `Volume` object, respectively.

Some of the properties of the `labelvolshow` object and the `volshow` object created using the `images.compatibility.volshow.R2022a.volshow` function have a different name with the `Viewer3D` and `Volume` objects. In particular, use the overlay properties, of the `Volume` object, such as `OverlayColormap` and `OverlayAlphamap`, to adjust the appearance of the volume labels. For more information, see `Viewer3D` and `Volume`.

Discouraged Usage	Recommended Replacement
This example displays volume data using the `images.compatibility.volshow.R2022a.volshow` function. `vol = images.compatibility.volshow.R2022a.volshow(V);`	Here is equivalent code that displays volume data in a `Volume` object using the `volshow` function. `vol = volshow(V);`

This example displays a volume on a blue background using an isosurface rendering style.
 	
`vol = images.compatibility.volshow.R2022a.volshow(V, ...`
`    BackgroundColor="b",Renderer="Isosurface");`
 	
Change the background color and the rendering style by setting properties of the object.
 	
`vol.BackgroundColor = "g";`
`vol.Renderer = "VolumeRendering");`
	Here is equivalent code that sets the background color using a `Viewer3D` object and the rendering style using a `Volume` object.
	
	`viewer = viewer3d(BackgroundColor="b");`
	`vol = volshow(V,Parent=viewer,RenderingStyle="Isosurf`
	
	Change the background color by setting properties of the `Viewer3D` object and change the rendering style by setting properties of the `Volume` object.
	
	`viewer.BackgroundColor = "g";`
	`vol.RenderingStyle = "VolumeRendering");`

This example displays labeled volume data using the `labelvolshow` function. `vol = labelvolshow(labels,volume);`	Here is equivalent code that displays labeled volume data in a `Volume` object using the `volshow` function. `vol = volshow(volume,OverlayData=labels);`

Discouraged Usage	Recommended Replacement

This example displays a labeled volume on a blue background and specifies the label colors.
 	
`vol = labelvolshow(labels,volume, ...`
`    BackgroundColor="b",LabelColor=cmap);`
 	
Change the background color and label colors by setting properties of the `labelvolshow` object.
 	
`vol.BackgroundColor = "g";`
`vol.LabelColor = cmap2;`
	Here is equivalent code that sets the background color using a `Viewer3D` object and the label colors using a `Volume` object.
	
	`viewer = viewer3d(BackgroundColor="b");`
	`vol = volshow(V,OverlayData=labels, ...`
	`    OverlayColormap=cmap,Parent=viewer);`
	
	Change the background color by setting properties of the `Viewer3D` object and change the label colors by setting properties of the `Volume` object.
	
	`viewer.BackgroundColor = "g";`
	`vol.OverlayColormap = cmap2;`

R2022a

Version: 11.5

New Features

Bug Fixes

Compatibility Considerations

Deep Learning: Added examples using deep neural networks

These examples show how to solve image processing problems by using deep neural networks. These examples require Deep Learning Toolbox™.

- The Detect Image Anomalies Using Explainable One-Class Classification Neural Network example has been updated to use a new data set. The new version shows how to train an anomaly detector for visual inspection of pill images.
- The Detect Image Anomalies Using Pretrained ResNet-18 Feature Embeddings example shows how to train a similarity-based anomaly detector using one-class learning of feature embeddings extracted from a pretrained ResNet-18 convolutional neural network.
- The Classify Defects on Wafer Maps Using Deep Learning example shows how to classify eight types of defects on silicon wafer maps using a simple convolutional neural network (CNN).

Image Browser App: Additional Import and Export Capabilities

The Image Browser app supports a greater set of options for curating a collection of images:

- Add images in directories or datastores to an already opened collection of images
- Remove a subset of images from a collection
- Export a subset of images to an image datastore
- Export images from a datastore to another datastore

Image Region Analyzer App: Additional Export Capabilities

The Image Region Analyzer app supports a greater set of export options:

- Export region properties as a structure or table
- Export a function that enables you to perform identical filtering and measurements on other binary images

imbilatfilt Function: Improved performance for uint8 and single data types

The `imbilatfilt` function shows improved performance for inputs of data type `uint8` and `single`.

For example, this code is about 2.5x faster than in the previous release.

```
function imbilatfiltTimingTest
      im = imread("cameraman.tif");
      im4k = imresize(im,[3840 2160]);
      tic
      out = imbilatfilt(im4k);
      toc
end
```

The approximate execution times are:

R2021b: 0.007 s

R2022a: 0.003 s

The code was timed on a Windows 10, Intel Xeon E5-2683 v4 CPU @ 2.10 GHz test system (two processors) by calling the function `imbilatfiltTimingTest`.

dicomCollection Function: Improved performance

The performance of the `dicomCollection` function has been improved. The function is about 30–50% faster than in the previous release.

C Code Generation: Generate code from additional functions using MATLAB Coder

The list indicates the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, these functions generate C code. For a complete list of Image Processing Toolbox functions that support code generation, see Functions Supporting Code Generation.

- `graythresh`
- `imguidedfilter`
- `imreducehaze`
- `imsharpen`
- `normxcorr2`
- `stdfilt`
- `wiener2`

GPU Code Generation: Generate CUDA code from additional functions using GPU Coder

The list indicates the Image Processing Toolbox functions that have been enabled for optimized CUDA code generation in this release.

- `adapthisteq`
- `imfindcircles`
- `mat2gray`
- `regionfill`

⚠ Functionality being removed or changed

The imsharpen function uses different color space conversion operations for RGB images

Behavior change

Starting in R2022a, the `imsharpen` function uses different color space conversion operations to sharpen RGB images. In R2021b and earlier, the `imsharpen` function performed color space conversions using the `makecform` and `applycform` functions. Starting in R2022a, the `imsharpen` function performs color space conversions using the `rgb2lab` and `lab2rgb` functions.

The new operations yield different results for sharpened RGB images. If you need to reproduce the old behavior, then you can replace the call to `imsharpen` with a call to the

`images.compatibility.imsharpen.r2021b.imsharpen` function instead. You do not need to change the input arguments.

The regionprops function always stores the Image, ConvexImage, and FilledImage properties as cell arrays in the output table for all inputs
Behavior change

Starting in R2022a, when you specify a table output format, the `regionprops` function stores the `Image`, `ConvexImage`, and `FilledImage` property values as cell arrays, regardless of the size of the image objects. In previous releases, if the size of the bounding box of an object was 1-by-1 or 1-by-*n*, these properties were stored in the output table as a numeric scalar or row vector, respectively.

To update your code, access the value of the `Image`, `ConvexImage`, and `FilledImage` properties using dot notation with curly braces, `{}`. For example, use this code to access the `Image` property for the first object in the input image `BW`. In previous releases, curly braces were not required to access values stored as a numeric scalar or row vector.

```
stats = regionprops("table",BW,"Image");
imdata = stats.Image{1};
```

R2021b

Version: 11.4

New Features

Bug Fixes

Compatibility Considerations

Blocked Images: Create and display labeled blocked images

The `polyToBlockedImage` function creates a labeled `blockedImage` from a set of coordinates that specify the vertices of one or many ROIs.

The `showlabels` function shows the label data in a `blockedImage` over the image data in a `bigimageshow` object. The `hidelabels` function hides the labels.

DICOM: Find and set attributes in DICOM metadata

The `dicomfind` function finds the location and value of a target attribute in DICOM metadata. The `dicomupdate` function enables you to set the value of a target attribute using the location of the attribute. These functions support nested attributes.

Image Quality Metrics: Calculate SSIM metric of deep learning arrays and specify dimensions of computation

The `ssim` function now accepts `dlarray` input for deep learning applications.

This function also supports formatted data with dimension labels of `'S'` (spatial), `'C'` (channel), and `'B'` (batch). The function returns a separate result for each index along the channel and batch dimensions.

Deep Learning: Added examples using deep neural networks

These examples show how to solve image processing problems by using deep neural networks. These examples require Deep Learning Toolbox.

- The Detect Image Anomalies Using Explainable One-Class Classification Neural Network example shows how to detect anomalies such as cracks in concrete using single-class classification.
- Two examples show how to generate a high-quality X-ray computed tomography (CT) image from a noisy CT image. The Unsupervised Medical Image Denoising Using CycleGAN example uses a cycle-consistent generative adversarial network (CycleGAN) and trains on patches of image data from a large collection of chest scans. The Unsupervised Medical Image Denoising Using UNIT example uses a UNIT network and trains using full images from a single chest scan.
- The Recover Images from Extreme Low-Light Conditions Using Deep Learning example shows how to recover brightened RGB images from RAW camera data collected in extreme low-light conditions using a U-Net.
- The Preprocess Multiresolution Images for Training Classification Network example shows how to create datastores that read and preprocess multiresolution whole slide images (WSIs) for the purpose of training a classification network. The Classify Tumors in Multiresolution Blocked Images example shows how to use the datastores to train and evaluate the performance of an Inception-v3 deep neural network.

medfilt3 Function: Improved performance for small neighborhood sizes

The `medfilt3` function shows improved performance for neighborhood sizes from [3, 3, 3] up to [31, 31, 31].

For example, this code is about 3x faster than in the previous release.

```
function timingTestMedfilt3
      load mristack;
      noisyV = imnoise(mristack,'salt & pepper',0.2);
      tic
      filteredV = medfilt3(noisyV);
      toc
end
```

The approximate execution times are:

R2021a: 0.24 s

R2021b: 0.08 s

The code was timed on a Windows 10, Intel Xeon Gold 5220 CPU @ 2.2 GHz test system by calling the function `timingTestMedfilt3`.

C Code Generation: Generate code from five functions using MATLAB Coder

The list indicates the Image Processing Toolbox functions that have been enabled for code generation in this release. For all target platforms, this function generates C code. For a complete list of Image Processing Toolbox functions that support code generation, see Functions Supporting Code Generation.

- `imgradient`[1]
- `imgradientxy`[1]
- `imnoise`
- `poly2mask`
- `regionfill`

[1] If you choose the generic `MATLAB Host Computer` option in the MATLAB Coder™ configuration settings, this function generates C code that uses a precompiled, platform-specific shared library. Using a shared library preserves performance optimizations in this function but limits the target to only those platforms that support MATLAB (see system requirements).

C Code Generation: Generate portable C code that has improved performance for seven functions

You can generate portable C code that has faster execution speed than in previous releases for these functions:

- `histeq`
- `imadjust`
- `imfill`
- `imfilter`
- `imreconstruct`
- `label2rgb`

- `medfilt2`

The optimizations include multithreading and algorithm improvements. Generating portable C code requires MATLAB Coder.

GPU Acceleration for ssim Function

R2021b adds GPU acceleration for the `ssim` function. GPU acceleration requires Parallel Computing Toolbox software and meeting the GPU computing requirements detailed here.

Thread-Based Environment: Run functions in a thread-backed pool

R2021b adds thread-based support for the Image Processing Toolbox functions listed in the table. Run code in the background on a single thread using MATLAB `backgroundPool` or accelerate code with multiple thread workers using `ThreadPool`. Use of multiple thread workers requires Parallel Computing Toolbox.

`adaptthresh`	`affine2d`	`affine3d`	`affineOutputView`
`cpcorr`	`cpstruct2pairs`	`findbounds`	`fitgeotrans`
`fliptform`	`gray2ind`	`grayslice`	`im2int16`
`im2single`	`im2uint16`	`im2uint8`	`imcrop`
`imcrop3`	`impyramid`	`imquantize`	`imref2d`
`imref3d`	`imregconfig`	`imregcorr`	`imregdemons`
`imregmtb`	`imresize3`	`imrotate`	`imrotate3`
`imsplit`	`imtranslate`	`imwarp`	`ind2gray`
`label2rgb`	`makeresampler`	`maketform`	`mat2gray`
`MattesMutualInformation`	`MeanSquares`	`normxcorr2`	`OnePlusOneEvolutionary`
`otsuthresh`	`projective2d`	`RegularStepGradientDescent`	`tformarray`
`tformfwd`	`tforminv`		

⚠ Functionality being removed or changed

The Format property of the GenericImage object is not recommended
Behavior change

Starting in R2021b, the `Format` property of the `GenericImage` object is not recommended. Use the `Extension` property instead. If you specify the `Format` property using dot notation, then the value is assigned to the `Extension` property.

R2021a

Version: 11.3

New Features

Bug Fixes

Compatibility Considerations

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (August 2021, Version 21.1.3)

Image Processing Toolbox Hyperspectral Imaging Library now supports singleton dimensions in hyperspectral data cubes.

To use 1-D or 2-D spectral data as the hypercube input to the supported hyperspectral functions, you must reshape it to 3-D volume data:

- Reshape 1-D spectral data of size 1-by-*P* into a 3-D hypercube of size 1-by-1-by-*P*.
- Reshape 2-D spectral data of size *M*-by-*P* into a 3-D hypercube of either size *M*-by-1-by-*P* or 1-by-*M*-by-*P*.

These functions in Image Processing Toolbox Hyperspectral Imaging Library support singleton dimensions for hyperspectral data.

Explore, Analyze, and Visualize	**Data Correction**	**Dimensionality Reduction**	**Spectral Unmixing**	**Spectral Matching and Target Detection**

`hypercube`	`dn2radiance`
	`dn2reflectance`
	`radiance2Reflectance`
	`empiricalLine`
	`flatField`
	`iarr`
	`logResiduals`
	`subtractDarkPixel`
		`hyperpca`
		`hypermnf`
		`inverseProjection`
			`ppi`
			`fippi`
			`nfindr`
			`estimateAbundanceLS`
				`sam`
				`sid`
				`jmsam`
				`sidsam`
				`ns3`
				`spectralMatch`
				`ndvi`
				`anomalyRX`

To use these functions and tools, you must download Image Processing Toolbox Hyperspectral Imaging Library from the Add-On Explorer. See Get and Manage Add-Ons.

Hyperspectral Image Processing: Image Processing Toolbox Hyperspectral Imaging Library (May 2021, Version 21.1.2)

This release introduces new features for smile detection and advanced smile correction. It also adds support for block processing of large-sized hyperspectral data cubes.

- Use the `smileMetric` function to detect spectral smile effect in a hyperspectral data cube.
- The `reduceSmile` function now supports the maximum noise fraction (MNF) transform-based method for smile correction. To select a smile correction method, specify the name-value argument `'Method'` with a value of either `'SpectralSmoothing'` or `'MNF'` for the moving average or MNF method, respectively.

- To process a large-sized hyperspectral data cube without running out of memory, you can use block processing. Enable block processing for the functions that support it by specifying the `'BlockSize'` name-value argument.

  For example, to enable block processing while using the `correctOOB` function, specify the name-value argument `'BlockSize',[20 20]`.

  ```
  newhcube = correctOOB(hcube,spectralResponse,'BlockSize',[20 20]);
  ```

  These functions in the Image Processing Toolbox Hyperspectral Imaging Library support block processing.

  - `correctOOB`
  - `dn2radiance`
  - `dn2reflectance`
  - `radiance2Reflectance`
  - `subtractDarkPixel`
  - `reduceSmile`
  - `spectralIndices`
  - `ndvi`

  To perform block processing by specifying the `'BlockSize'` name-value argument, you must have MATLAB R2021a or a later release.

To use these functions and tools, you must download Image Processing Toolbox Hyperspectral Imaging Library from the Add-On Explorer. See Get and Manage Add-Ons.

Deep Learning Networks: Create and combine encoder and decoder modules for networks with repetitive structures

Create a network with repetitive blocks of layers, such as an encoder module that performs a series of downsampling operations or a decoder module that performs a series of upsampling operations, using the `blockedNetwork` function.

Create an encoder module directly from a pretrained network, such as VGG-19 or Inception-v3, using the `pretrainedEncoderNetwork` function.

Connect an encoder module and a decoder module with an optional bridge, final network, and skip connections using the `encoderDecoderNetwork` function.

These functions require Deep Learning Toolbox.

Deep Learning Networks: Create generators and discriminators for GAN networks

You can create popular GAN generator and discriminator networks: CycleGAN, PatchGAN, pix2pixHD, and UNIT. You can optionally modify the networks by changing aspects such as the number of downsampling operations and the type of activation and normalization operations. The table describes the functions that enable you to create and modify GAN networks.

Network	Creation and Modification Functions
CycleGAN generator network	Create a CycleGAN generator network using the `cycleGANGenerator` function.
pix2pixHD generator network	Create a pix2pixHD generator network using the `pix2pixHDGlobalGenerator`. Add a local enhancer to a pix2pixHD generator network using the `addPix2PixHDLocalEnhancer` function.
UNIT generator network	Create an unsupervised image-to-image translation (UNIT) generator network using the `unitGenerator` function. Perform image-to-image translation on a trained UNIT network using the `unitPredict` function.
PatchGAN discriminator network	Create a PatchGAN discriminator network or a pixel discriminator network using the `patchGANDiscriminator` function.

These functions require Deep Learning Toolbox.

Deep Learning Data Processing: Rearrange data between dimensions and resize data to match feature map

The `DepthToSpace2DLayer` object and `depthToSpace` function rearrange data from the depth dimension to spatial dimensions.

The `SpaceToDepthLayer` object, which rearranges spatial blocks of data into the depth dimension, no longer requires Computer Vision Toolbox™ as of R2021a. The new `spaceToDepth` function also rearranges spatial blocks of data into the depth dimension.

The `resize2dLayer` and `resize3dLayer` objects now support resizing input to the size of a second input feature map connected to the layer. To use this resizing mode, specify the `EnableReferenceInput` property of the layer as `true`.

These functions require Deep Learning Toolbox.

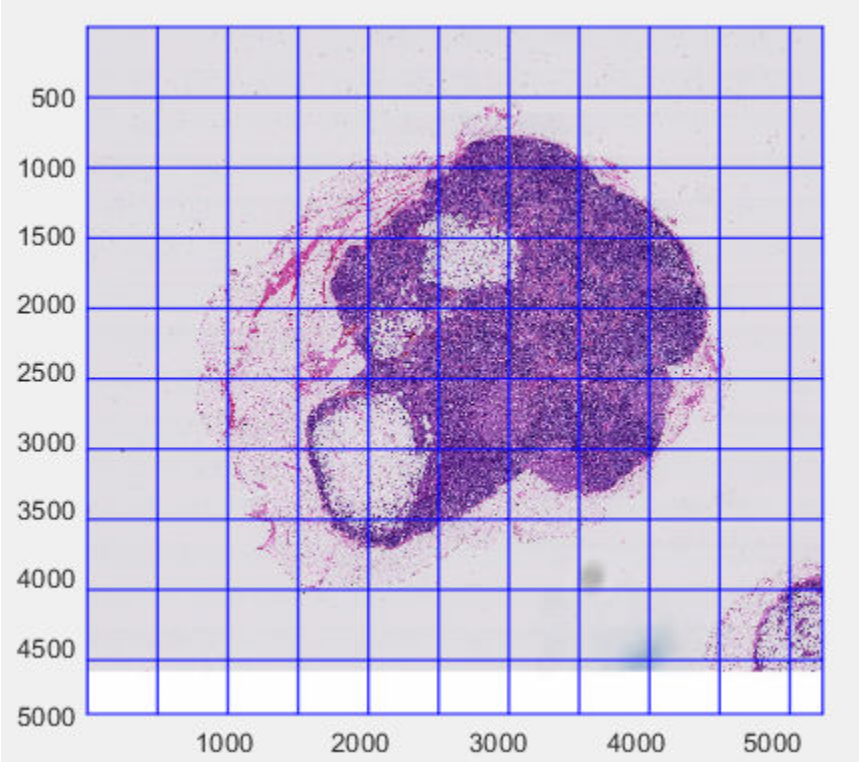
Deep Learning: Added examples using deep neural networks

These examples show how to solve image processing problems by using deep neural networks. These examples require Deep Learning Toolbox.

- The Unsupervised Day-To-Dusk Image Translation Using UNIT example shows how to perform image-to-image translation of images captured during daytime and dusk conditions using a UNIT network.
- The Quantify Image Quality Using Neural Image Assessment example shows how to quantify image quality using the no-reference NIMA model.
- The Classify Large Multiresolution Images Using `blockedImage` and Deep Learning example now demonstrates the new recommended workflow using `blockedImage` and `blockedImageDatastore` objects.

⚠Blocked Images: Process N-D images that are too large to fit in memory

Read and process N-D images that are too large to fit in memory by using a `blockedImage` object. A `blockedImage` object supports images with one or multiple resolution levels. This illustration shows a large, multiresolution image, converted to a blocked image, displayed with block boundaries visible. Each block is 512-by-512.



To manage a collection of image blocks that belong to one or more `blockedImage` objects, use a `blockedImageDatastore` object.

The `blockedImage` and `blockedImageDatastore` objects replace the `bigimage` and `bigimageDatastore` objects. For more information, see Compatibility Considerations on page 11-8.

Volume Segmenter app supports blocked images

The Volume Segmenter app now accepts `blockedImage` objects as input for segmentation. Use blocked images when a volume is too large to fit into memory. By using blocked images, you can segment volumes without running out of memory.

After loading the blocked image into Volume Segmenter, you can use the same set of tools the app offers for nonblocked images to segment regions of the volume. This includes the use of automated methods, working block-by-block through each slice. Use the **Blocked Image** tab in Volume Segmenter to specify the block you want to work on, and to mark that block completed. Volume Segmenter automatically calculates the percent of blocks that are done. Volume Segmenter combines

your work across blocks to create masks. For more information, see Work with Blocked Images Using the Volume Segmenter.

RAW Files: Read RAW files and color filter array data, and convert to RGB images

You can read and work with images in the RAW file format. The RAW file format preserves image data in its most unedited state, recorded directly from the camera sensor. Most camera manufacturers define their own proprietary RAW file format, such as the Nikon NEF file format and the Canon® CRW format. Adobe® has also defined a RAW file format, DNG (digital negative), which is supported by several cameras. To work with RAW files, use these functions:

- `rawinfo` – Read information about color filter array (CFA) images in RAW files
- `rawread` – Read CFA images from RAW files
- `raw2rgb` – Convert a RAW file into an RGB file
- `raw2planar` – Separate a Bayer-patterned CFA image into individual, sensor-element images
- `planar2raw` – Combine planar sensor images into a full Bayer-pattern CFA image

Image Quality Metrics: Calculate metrics of deep learning arrays and specify dimensions of computation

The `psnr`, `multissim`, and `multissim3` functions now accept `dlarray` input for deep learning applications.

These functions also support formatted data with dimension labels of `'S'` (spatial), `'C'` (channel), and `'B'` (batch). For data with a batch dimension, the functions return a separate result for each index along the batch dimension.

⚠Image Quality Test Charts: Read and analyze additional versions of Imatest eSFR test charts

The `esfrChart` object can detect and analyze the Enhanced, Wedge Enhanced, and Wedge Extended versions of the Imatest® eSFR test chart.

You can specify camera parameters to compensate for distortion using the `'CameraParameters'` name-value argument.

By default, the `esfrChart` object now refines the position of slanted-edge ROIs based on localized image content. For more information, see Compatibility Considerations on page 11-8.

Image Quality Test Charts: Measure scene illuminant using X-Rite ColorChecker test chart

You can now measure the scene illuminant of an image of an X-Rite® ColorChecker® test chart by using the `measureIlluminant` function.

Registration Estimator App: Support for KAZE and ORB registration techniques

The Registration Estimator app now supports two additional registration techniques: KAZE and ORB.

The KAZE technique is a multiscale 2-D feature detection and description algorithm in nonlinear scale spaces.

The ORB technique detects corners in images with changes in scale, rotations, or both.

imerase: Erase rectangular region of image

Erase a rectangular region from an image using the `imerase` function. You can specify the fill value of the erased region.

randomWindow2d: Select rectangular region of image

Select a rectangular region of an image using the `randomWindow2d` function. You can specify the dimensions of the region or the aspect ratio and area of the region as a fraction of the total image area.

C Code Generation: Generate code from the adapthisteq function using MATLAB Coder

You can generate C code from the `adapthisteq` function by using MATLAB Coder. If you choose the generic `MATLAB Host Computer` target platform, the function generates code that uses a precompiled, platform-specific shared library.

C Code Generation: Generate portable C code with improved performance for six functions

For these functions, you can generate portable C code that has faster execution speed than in previous releases.

- `bwareaopen`
- `bwboundaries`
- `bwlabel`
- `houghpeaks`
- `imbilatfilt`
- `otsuthresh`

The optimizations include the use of multithreading, data parallelization, and SIMD code generation. Generating portable C code requires MATLAB Coder.

GPU Code Generation Support for resize2dLayer Object

You can generate optimized CUDA code from the `resize2dLayer` layer. The generated code includes CUDA kernels for parallelizable parts of your deep learning algorithms. To generate this code, you must have a GPU Coder™ license.

GPU Acceleration for imcrop, multissim, multissim3, psnr, and wiener2 Functions and Enhanced GPU Support for imwarp

R2021a adds GPU acceleration for the `imcrop`, `multissim`, `multissim3`, `psnr`, and `wiener2` functions. GPU acceleration requires Parallel Computing Toolbox software and meeting the GPU computing requirements detailed here.

The `imwarp` function, which already supports GPU acceleration, now extends that support to include the use of displacement fields.

⚠Functionality being removed or changed

bigimage object and bigimageDatastore object are not recommended
Still runs

The `bigimage` object and the `bigimageDatastore` object are not recommended. To work with large images, use the `blockedImage` object and the `blockedImageDatastore` object instead.

The new blocked image technology offers these advantages:

- Extension to N-D processing
- Introduction of a documented adapter interface to enable custom support for any data source that can be chunked into blocks
- Simpler interface because of use of direct pixel subscripts (`bigimage` uses world coordinates)
- Simpler interface for single-resolution level images (default resolution level is 1)

To display blocked image data, use the `bigimageshow` function.

randomCropWindow2d function is not recommended
Still runs

The `randomCropWindow2d` function is not recommended. To select random crop windows from an image, use the `randomWindow2d` function instead.

To update your code, change instances of the function name `randomCropWindow2d` to `randomWindow2d`. You do not need to change the input arguments.

esfrChart object now refines slanted edge ROI positions
Behavior change

Starting in R2021a, the `esfrChart` object refines the position of slanted edge ROIs based on localized image content by default. In previous releases, the `esfrChart` object did not refine the position of the slanted edge ROIs after the initial estimate of the position. The refinement results in more precise ROI positions.

To replicate the previous behavior, specify the '`RefinePoints`' name-value argument as `false` when creating an `esfrChart` object.