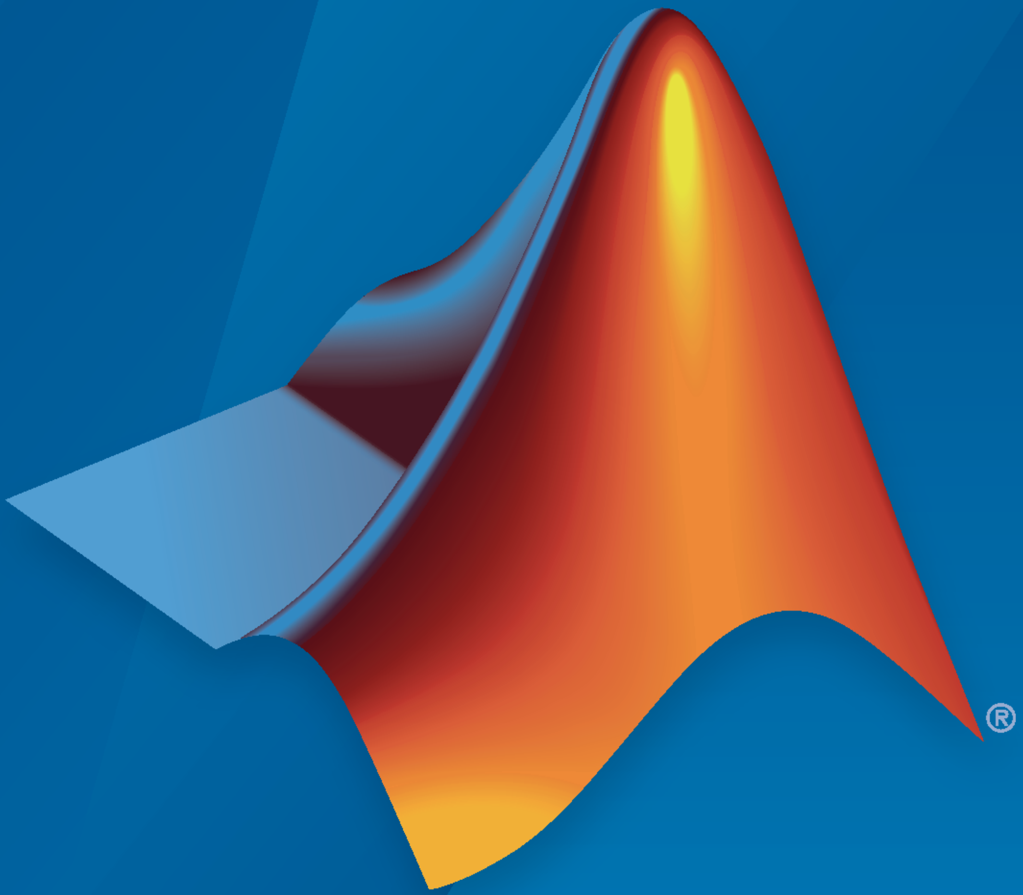
Medical Imaging Toolbox™

Reference



MATLAB®

R2022b

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

Medical Imaging Toolbox™ Reference

Revision History

September 2022	Online only	New for Version 1.0 (Release 2022b)

Functions

Medical Image Labeler

Display and label 2-D and 3-D medical images

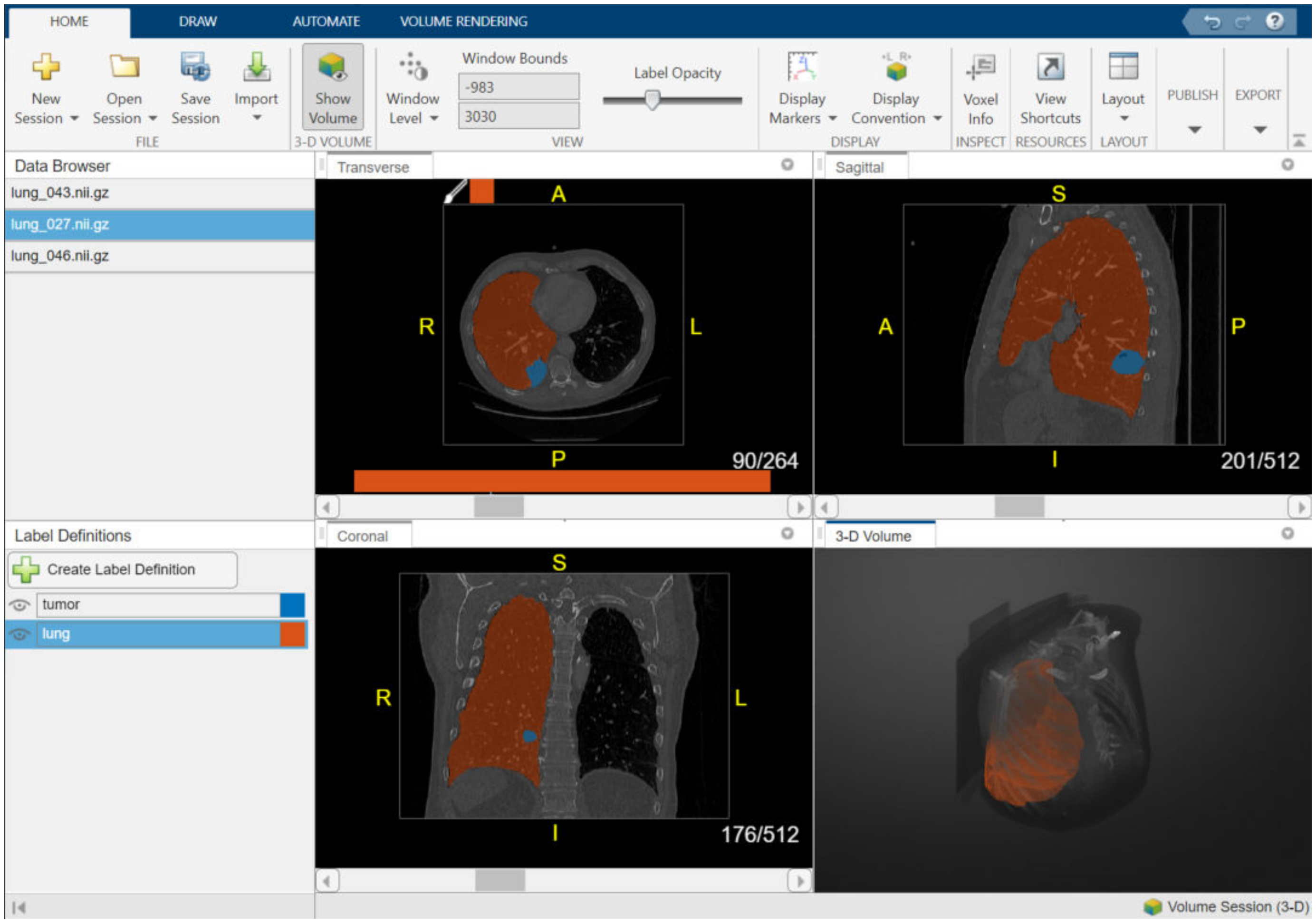
Description

The **Medical Image Labeler** app enables you to label ground truth data in medical images. Using the app, you can:

- Import multiple 2-D images or 3-D image volumes.
- View images as slice planes or volumes with anatomical orientation markers and scale bars.
- Create multiple pixel label definitions to label regions of interest. Label pixels using automatic algorithms such as flood fill, semi-automatic techniques such as interpolation, and manual techniques such as painting by superpixels.
- Write, import, and use your own custom automation algorithm to automatically label ground truth data.
- Export the labeled ground truth data as a `groundTruthMedical` object. You can use this object to share labels with colleagues or for training semantic segmentation deep learning networks.

The **Medical Image Labeler** app supports 2-D images and image sequences stored in the DICOM and NIfTI file formats. An image sequence is a series of images related by time, such as ultrasound data. The app supports 3-D image volume data stored in the DICOM (single or multifile volume), NIfTI, and NRRD file formats.

To learn more about this app, see “Get Started with Medical Image Labeler”.



Open the Medical Image Labeler App

- MATLAB® Toolstrip: On the **Apps** tab, under **Image Processing and Computer Vision**, click the **Medical Image Labeler** app icon.
- MATLAB command prompt: Enter `medicalImageLabeler`.

Examples

- "Get Started with Medical Image Labeler"
- "Visualize 3-D Medical Image Data Using Medical Image Labeler"
- "Label 2-D Ultrasound Series Using Medical Image Labeler"
- "Label 3-D Medical Image Using Medical Image Labeler"
- "Collaborate on Multi-Labeler Medical Image Labeling Projects"

Programmatic Use

`medicalImageLabeler` opens the **Medical Image Labeler** app, which enables you to start a new session to label 2-D or 3-D medical image data.

`medicalImageLabeler(gTruthFile)` opens the app and loads the image and label data stored in the file `gTruthFile` into the app. `gTruthFile` is the full path to a MAT file containing a `groundTruthMedical` object, specified as a string scalar or character vector.

`medicalImageLabeler(gTruth)` opens the app and loads the image and label data stored in the `groundTruthMedical` object `gTruth` from the workspace into the app.

`medicalImageLabeler(sessionFolder)` opens the app and loads a saved labeling session into the app, where `sessionFolder` is the full path to a session folder created using the **Medical Image Labeler** app.

`medicalImageLabeler(sessionType)` opens the app and creates a new session of the specified type, where `sessionType` is `"Volume"` or `"Image"`. For more information about session types, see “Get Started with Medical Image Labeler”.

Version History

Introduced in R2022b

See Also

`groundTruthMedical` | **Image Labeler**

Topics

“Get Started with Medical Image Labeler”
“Visualize 3-D Medical Image Data Using Medical Image Labeler”
“Label 2-D Ultrasound Series Using Medical Image Labeler”
“Label 3-D Medical Image Using Medical Image Labeler”
“Collaborate on Multi-Labeler Medical Image Labeling Projects”

extractIsosurface

Extract isosurface from volume using marching cubes algorithm

Syntax

```
[faces,verts] = extractIsosurface(V,isovalue)
```

Description

An isosurface is a 3-D surface representation of points with equal values in a 3-D intensity volume. The `extractIsosurface` function returns the face and vertex data of the isosurface extracted by connecting points of a constant value within a volume of space. The `extractIsosurface` function uses the marching cubes algorithm to extract isosurface data as arrays faster than the corresponding syntax of the `isosurface` function, without compromising resolution. For additional options, you must use the `isosurface` function.

`[faces,verts] = extractIsosurface(V,isovalue)` extracts an isosurface from the intensity volume `V` by determining where the values of `V` are equal to the specified isovalue `isovalue`. The function returns the face and vertex data of the isosurface in `faces` and `verts`, respectively.

Examples

Plot Isosurface

Load the intensity volume data into the workspace.

```
load(fullfile(toolboxdir("images"),"imdata","BrainMRILabeled","images","vol_001.mat"));
V = vol;
```

Specify the isovalue for isosurface extraction.

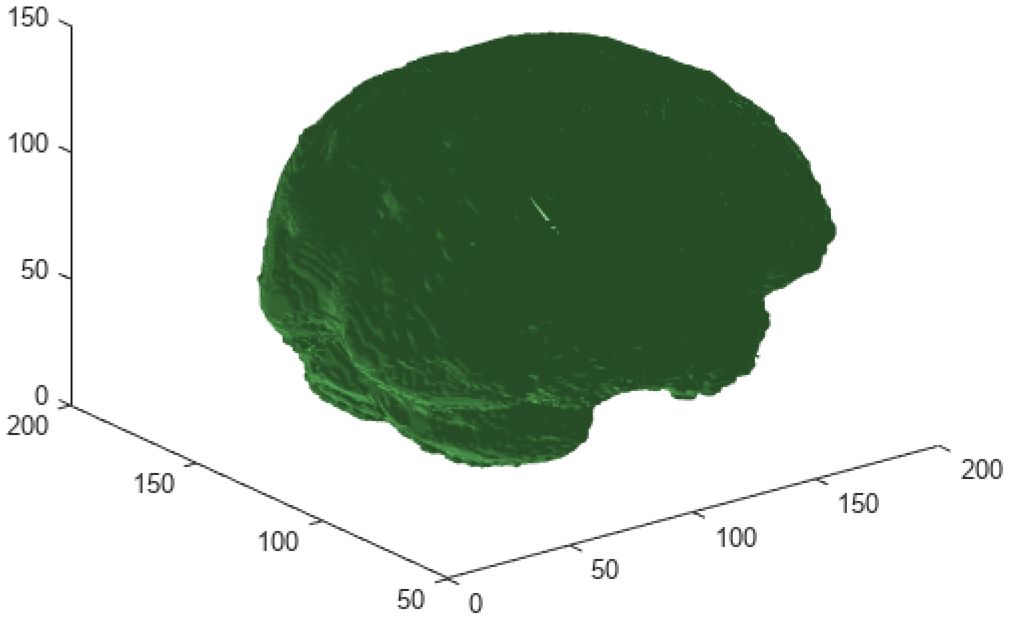
```
isovalue = 100;
```

Extract the isosurface of the input volume at the specified isovalue.

```
[faces,verts] = extractIsosurface(V,isovalue);
```

Plot the extracted isosurface.

```
figure
p = patch(Faces=faces,Vertices=verts);
isonormals(V,p)
view(3)
set(p,FaceColor=[0.5 1 0.5])
set(p,EdgeColor="none")
camlight
lighting gouraud
```



Create Point Cloud from Isosurface

Load the intensity volume data into the workspace.

```
load(fullfile(toolboxdir("images"),"imdata","BrainMRILabeled","labels","label_001.mat"));
V = label;
```

Specify the isovalue for isosurface extraction.

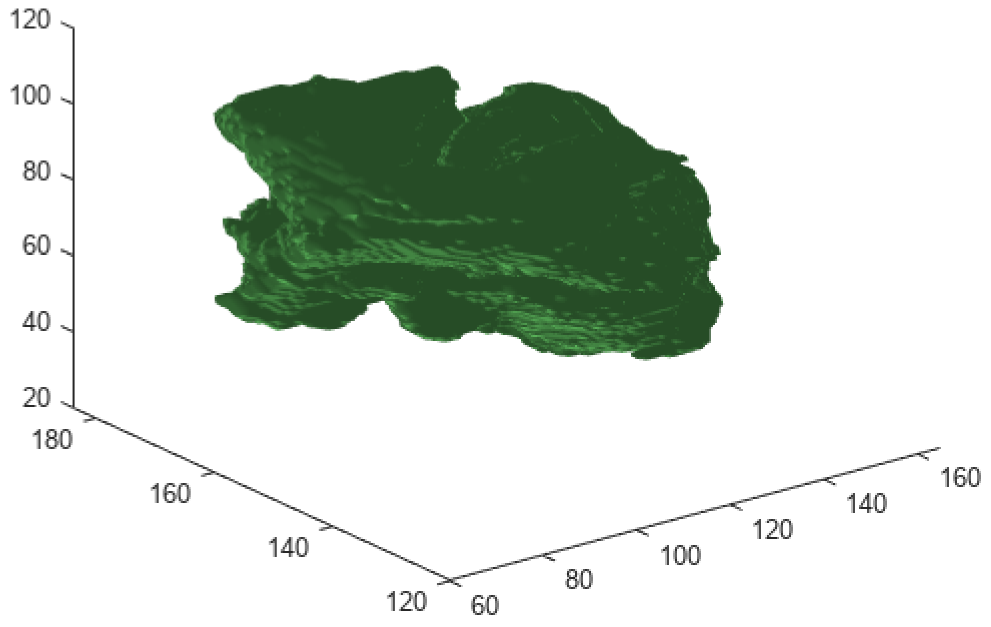
```
isovalue = 0.05;
```

Extract the isosurface of the input volume at the specified isovalue.

```
[faces,verts] = extractIsosurface(V,isovalue);
```

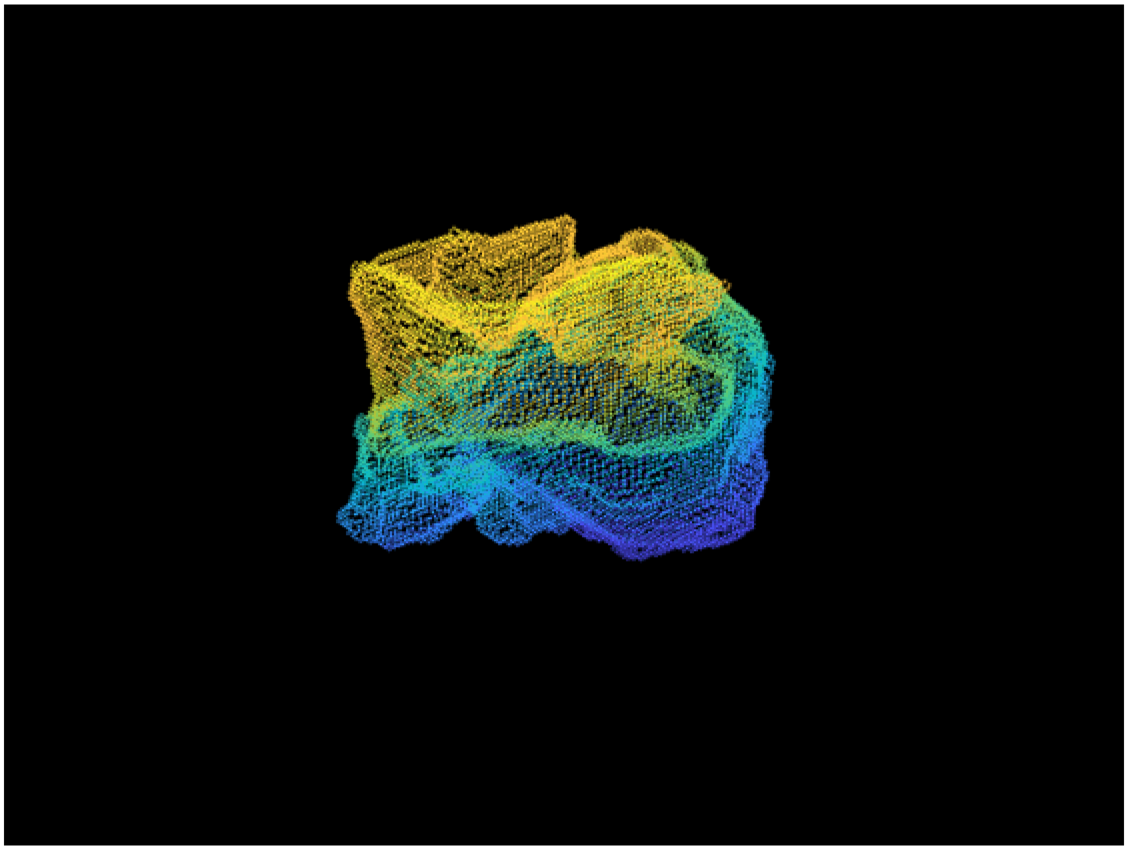
Display and inspect the extracted isosurface.

```
figure
p = patch(Faces=faces,Vertices=verts);
isonormals(V,p)
view(3)
set(p,FaceColor=[0.5 1 0.5])
set(p,EdgeColor="none")
camlight
lighting gouraud
```



Create a point cloud from the vertices of the extracted isosurface. Display the point cloud.

```
ptCloud = pointCloud(verts);
figure
pcshow(ptCloud)
```



Create STL File for 3-D Printing from Isosurface

Load the intensity volume data into the workspace.

```
load(fullfile(toolboxdir("images"),"imdata","BrainMRILabeled","labels","label_002.mat"));
V = label;
```

Specify the isovalue for isosurface extraction.

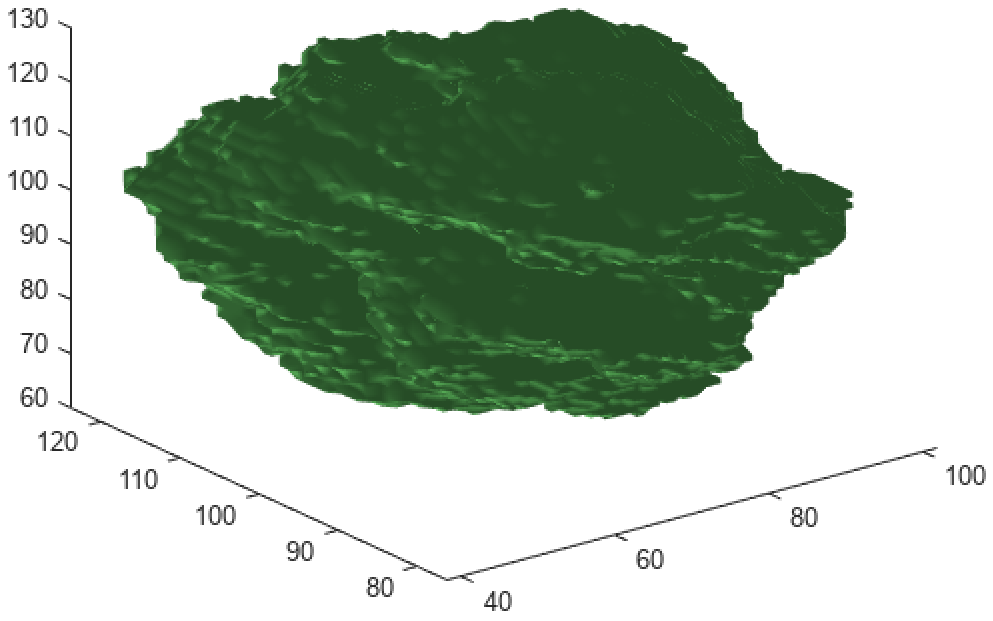
```
isovalue = 0.05;
```

Extract the isosurface of the input volume at the specified isovalue.

```
[faces,verts] = extractIsosurface(V,isovalue);
```

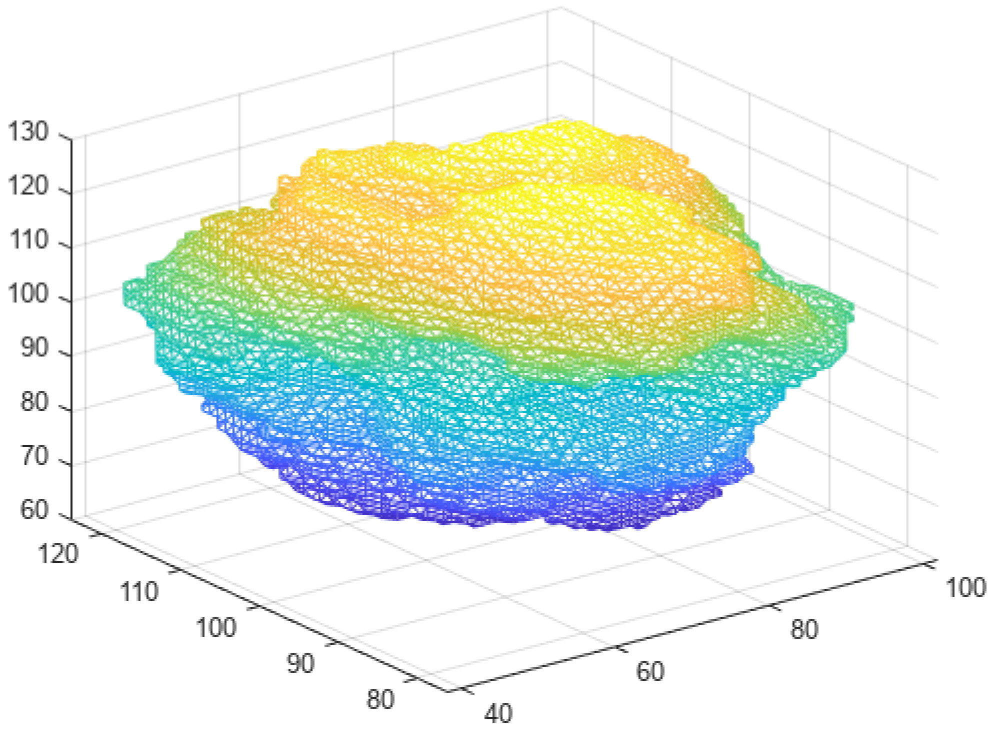
Display and inspect the extracted isosurface.

```
figure
p = patch(Faces=faces,Vertices=verts);
isonormals(V,p)
view(3)
set(p,FaceColor=[0.5 1 0.5])
set(p,EdgeColor="none")
camlight
lighting gouraud
```



Triangulate the extracted isosurface. Display the triangulation as a mesh.

```
T = triangulation(double(faces),double(verts));
figure
trimesh(T)
```



Create an STL file for 3-D printing, using the triangulation of the extracted isosurface.

```
stlwrite(T,"brain.stl")
```

Input Arguments

V — Intensity volume data
3-D numeric array | 3-D logical array

Intensity volume data, specified as a 3-D numeric or logical array.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32` | `logical`

`isovalue` — Isovalue
numeric scalar

Isovalue at which to compute the isosurface, specified as a numeric scalar.

Data Types: `single` | `double`

Output Arguments

`faces` — Face data of computed isosurface
M-by-3 matrix

Face data of the computed isosurface, returned as an *M*-by-3 matrix. *M* is the number of faces in the isosurface. Three vertices form a triangular face. The 3-D index coordinates of all the vertices of the surface are returned as the rows of `verts`. Each row of the `face` matrix contains the row indices of the three vertices from `verts` that form the triangular face.

Data Types: `single`

`verts` — Vertex data of computed isosurface
N-by-3 matrix

Vertex data of the computed isosurface, returned as an *N*-by-3 matrix. *N* is the number of vertices in the isosurface. Each row of the matrix contains the 3-D index coordinates of one vertex of the isosurface.

Data Types: `single`

Algorithms

The `extractIsosurface` function uses the marching cubes algorithm to extract the isosurface of a volume `V` at the specified isovalue `isovalue`. The marching cubes algorithm uses lookup tables to obtain information about the faces and vertices of the isosurface. The lookup tables enable the `extractIsosurface` function to extract face and vertex data as arrays faster than the `isosurface` function without compromising resolution for large intensity volumes, such as those typically used in medical imaging. Though the purposes of `extractIsosurface` and `isosurface` are similar, there are certain differences in their implementation and output.

- The behavior of `extractIsosurface` and `isosurface` differs for the edge case when an intensity value is equal to the specified isovalue. The `isosurface` function generates a surface between regions with intensities less than or equal to the isovalue and regions with intensities greater than the isovalue. The `extractIsosurface` function generates a surface between regions with intensities less than the isovalue and regions with intensities greater than or equal to the isovalue.
- Except for the edge case, both `extractIsosurface` and `isosurface` generate the same number of vertices with the index coordinates of the vertices matching within a small tolerance. However, the order of the vertices in the output `verts` can be different.
- Except for the edge case, both `extractIsosurface` and `isosurface` generate the same number of faces. However, the actual faces in the output `faces` are different because both algorithms create the same surface using different triangulations of the vertices.

Version History

Introduced in R2022b

References

[1] Lorensen, William E., and Harvey E. Cline. “Marching Cubes: A High Resolution 3D Surface Construction Algorithm.” *ACM SIGGRAPH Computer Graphics* 21, no. 4 (August 1987): 163–69. https://doi.org/10.1145/37402.37422.

See Also

`isosurface`

Topics
"Connecting Equal Values with Isosurfaces"

imregdeform

Deformable registration of grayscale images or intensity volumes using total variation method

Syntax

```
[dispField,reg] = imregdeform(moving,fixed)
[dispField,reg] = imregdeform(moving,fixed,Name=Value)
```

Description

The `imregdeform` function uses the total variation method to perform deformable registration of grayscale images or intensity volumes. You can use this function to register medical images or volumes deformed due to local transformations.

`[dispField,reg] = imregdeform(moving,fixed)` transforms the grayscale image or intensity volume `moving`, so that it is registered with the reference image or volume `fixed`, and returns the displacement field `dispField` and the registered image or volume `reg`.

`[dispField,reg] = imregdeform(moving,fixed,Name=Value)` specifies options for the total variation method using one or more optional name-value arguments.

Examples

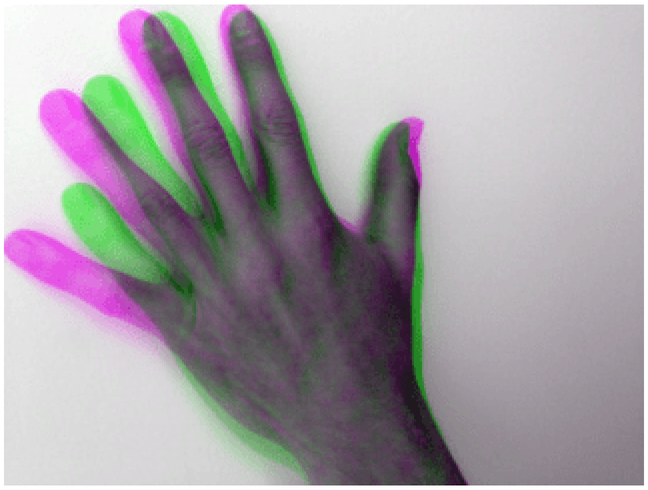
Deformable Registration of Images

Load a reference image and an image to be registered into the workspace. Convert the images to grayscale.

```
fixedImg = imread("hands1.jpg");
fixed = im2gray(fixedImg);
movingImg = imread("hands2.jpg");
moving = rgb2gray(movingImg);
```

Display the fixed image and the moving image. Observe the deformation in the alignment of the images.

```
figure
imshowpair(fixed,moving)
```



Register the moving image to the fixed image.

```
[dispField,reg] = imregdeform(moving,fixed,NumPyramidLevels=6,GridRegularization=0.6);
```

```
--------------------------- Pyramid Level = 6 --------------------------------
               Normalized     Function local                          Closeness to
 Iteration        rmse           minima           Step-size        optimal solution
     1          1.64147         37024.85            4.19               49.62783
     2          1.48633         31590.32           24.34               48.04233
     3          1.63155         30339.11           18.00               46.47121
     4          1.33748         29085.22            7.83               60.12520
     5          1.29691         28578.16           10.20               74.34988
     6          1.35689         28002.80           14.17               78.68985
     7          1.36835         27701.23            7.44               65.67400
     8          1.17573         27384.96            8.51               35.42655
     9          1.08997         26944.82           13.44               53.24290
    10          0.95153         26374.55            5.58               29.71761
    11          0.91025         26169.77            2.55               26.32110
    12          0.91195         25891.90            5.95               34.23311
    13          1.14507         25726.65           14.50               41.34727
    14          1.22759         25281.23           11.61               33.04872
    15          0.97035         24703.21            7.40               34.35227
    16          1.03194         24307.52            9.90               34.04009
    17          1.03114         24077.92            7.69               41.98692
    18          0.98282         23736.04           14.83               46.11518
    19          0.95928         23317.39           14.87               37.88543
    20          0.92435         22891.46           17.10               45.98425
    21          0.81884         22398.19           11.15               37.09230
    22          0.70854         22097.88            6.20               23.16606
    23          0.65405         21979.78            3.92               20.64745
    24          0.58632         21890.52            3.22               17.18440
    25          0.59774         21767.81            7.38               17.62879
    26          0.63929         21698.98            8.36               22.69476
    27          0.50519         21574.89            2.62               15.70961
    28          0.50145          21491.14          3.01                12.74984
    29          0.50583          21419.94          3.16                14.86162
    30          0.48495          21390.37          4.85                13.89448
    31          0.46792          21363.09          1.43                13.67136
    32          0.40962          21356.92          1.65                 9.81103
    33          0.40800          21356.83          0.18                 9.50099

--------------------------- Pyramid Level = 5 -------------------------------
              Normalized     Function local                          Closeness to
 Iteration       rmse           minima          Step-size        optimal solution
     1          0.44400          23959.26          5.30                22.96675
     2          0.39054          23254.00          9.46                18.54226
     3          0.36834          23134.24          5.02                22.51686
     4          0.33450          23051.56          1.59                12.51227
     5          0.33048          23004.59          2.58                10.00633
     6          0.32744          22960.51          3.02                 8.53760
     7          0.35590          22882.32          6.67                25.78811
     8          0.32620          22788.98          1.99                10.72627
     9          0.31109          22735.69          1.69                 7.66283
    10          0.30765          22695.63          2.81                 6.17570
    11          0.30926          22686.24          1.23                 7.16721
    12          0.31211          22682.41          0.80                 6.14514
    13          0.31531          22680.85          0.64                 4.67112
    14          0.31613          22680.72          0.19                 4.48773

--------------------------- Pyramid Level = 4 -------------------------------
              Normalized     Function local                          Closeness to
 Iteration       rmse           minima          Step-size        optimal solution
     1          0.28145          28405.52          4.99                21.00908
     2          0.26476          27597.03          9.61                12.38693
     3          0.24598          27383.91          4.02                15.36024
     4          0.23363          27324.84          1.62                 7.73944
     5          0.23838          27210.31          2.64                 5.38586
     6          0.23927          27069.79          3.11                 8.21936
     7          0.24557          26887.38          5.87                15.04406
     8          0.23496          26800.32          1.50                 8.34007
     9          0.23145          26768.71          1.57                 4.85061
    10          0.23168          26719.01          2.55                 6.54151
    11          0.23681          26642.15          5.61                15.08834
    12          0.23464          26630.58          0.24                13.59409
    13          0.23266          26610.13          0.46                10.57861
    14          0.23255          26601.49          0.22                 9.64829

--------------------------- Pyramid Level = 3 -------------------------------
              Normalized     Function local                          Closeness to
 Iteration       rmse           minima          Step-size        optimal solution
     1          0.16744          30346.55          7.61                 9.96176
     2          0.16423          30102.90          4.35                 8.32035
     3          0.16156          29980.10          2.71                 5.30050
     4          0.16202          29875.25          3.51                 5.31045
     5          0.16292          29776.72          4.77                 7.43417
     6          0.16392          29666.28          4.26                 6.45342
     7          0.16464          29634.29          3.28                 6.47225
     8          0.16468          29602.16          3.41                 9.24666
     9          0.16277          29597.64          2.84                 4.73214
    10          0.16176          29572.86          3.45                 4.35966
    11          0.16176          29572.86          0.00                 4.35966
```

```
--------------------------- Pyramid Level = 2 -------------------------------
              Normalized     Function local                          Closeness to
 Iteration       rmse            minima          Step-size        optimal solution
     1          0.12139         29164.12            9.06                 7.14654
     2          0.11765         28862.99            5.23                 7.51096
     3          0.11638         28735.95            3.36                 5.13431
     4          0.11604         28645.66            3.50                 6.77245
     5          0.11627         28571.44            4.39                 7.44123
     6          0.11643         28484.52            3.61                 6.90509
     7          0.11650         28430.70            2.00                 4.35053
     8          0.11647         28378.27            2.95                 6.94753
     9          0.11613         28331.36            3.26                 7.69636
    10          0.11584         28303.83            2.11                 6.46119
    11          0.11552         28262.96            2.64                 4.12788
    12          0.11530         28249.84            2.80                11.04358
    13          0.11536         28165.72            3.59                 7.07860
    14          0.11575         28115.74            1.89                 4.88700
    15          0.11620         28076.12            2.02                 7.58377
    16          0.11622         28074.33            0.16                 7.56145

--------------------------- Pyramid Level = 1 -------------------------------
              Normalized     Function local                          Closeness to
 Iteration       rmse            minima          Step-size        optimal solution
     1          0.08227         31990.04           11.46                 9.08693
     2          0.08242         31703.92            3.95                 6.42661
     3          0.08259         31673.28            2.83                 5.67247
     4          0.08258         31592.70            1.62                 2.86767
     5          0.08248         31472.07            3.63                 3.94959
     6          0.08236         31394.89            4.08                 8.77327
     7          0.08236         31318.72            3.15                 6.37404
     8          0.08241         31284.27            2.45                 7.80319
     9          0.08246         31261.99            2.13                 5.12846
    10          0.08250         31229.43            2.73                 9.01950
    11          0.08251         31184.45            2.76                 3.64364
    12          0.08250         31134.16            2.36                 4.62543
    13          0.08248         31093.56            2.19                 4.72333
    14          0.08242         31058.02            3.06                 4.89104
    15          0.08241         31054.33            0.65                 4.79401
    16          0.08241         31043.47            4.64                 4.55497
    17          0.08246         30993.96            1.31                 4.77195
    18          0.08250         30974.60            1.80                 2.70112
    19          0.08252         30930.28            3.55                 2.98517
    20          0.08254         30888.53            3.00                 6.74025
    21          0.08255         30859.24            0.88                 3.84303
    22          0.08255         30832.77            0.33                 4.47997
    23          0.08256         30830.20            0.29                 4.08082
    24          0.08256         30822.42            0.76                 3.60574
    25          0.08256         30821.96            0.31                 3.39634
    26          0.08256         30818.17            0.43                 3.08191
```

Display the fixed image and the registered image. Observe the alignment of the images.

```
figure
imshowpair(fixed,reg)
```

Deformable Registration of MRI Volume

Load a MAT file containing a reference volume into the workspace. Convert the reference volume to data type double.

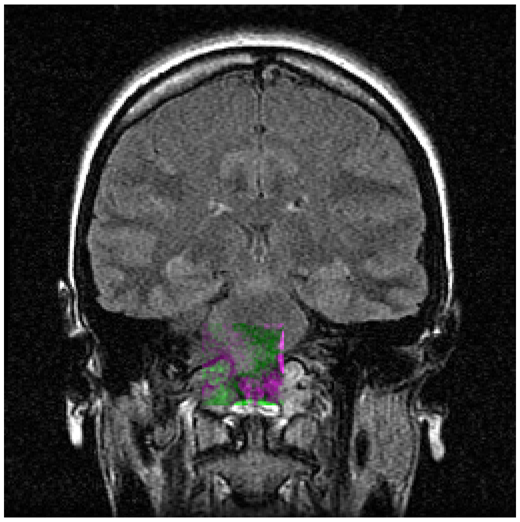
```
load mristack.mat
fixed = im2double(squeeze(mristack));
```

Create a deformed volume using local transformations.

```
local = fixed(160:200,100:140,:);
local = imrotate(local,90);
moving = fixed;
moving(160:200,100:140,:) = local;
```

Display a slice of the fixed volume and a corresponding slice of the moving volume. Observe the deformation in the alignment of the volumes.

```
figure
imshowpair(fixed(:,:,10),moving(:,:,10))
```



Register the moving volume to the fixed volume.

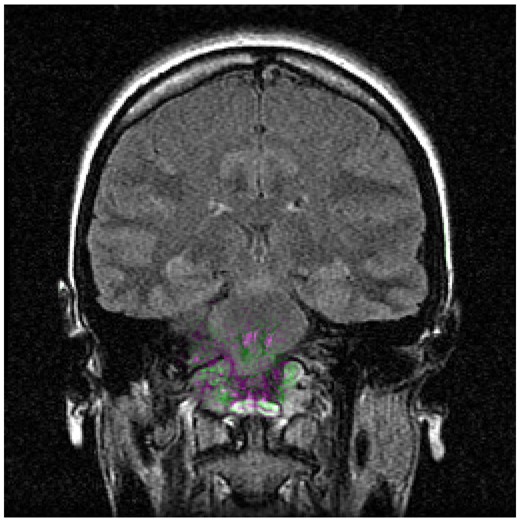
```
[dispField,reg]=imregdeform(moving,fixed,GridRegularization=0.001);
```

```
--------------------------- Pyramid Level = 3 --------------------------------
              Normalized     Function local                          Closeness to
 Iteration       rmse            minima          Step-size        optimal solution
     1         0.00016         34081.23           4.60               79.85176
     2         0.00016         27591.18          16.75               70.06876
     3         0.00019         26948.09          23.68               65.83620
     4         0.00018         26498.21           5.02               59.19717
     5         0.00018         25872.69           8.68               43.63346
     6         0.00019         25774.23           7.70               46.14477
     7         0.00019         25774.23           0.00               46.14477

--------------------------- Pyramid Level = 2 --------------------------------
              Normalized     Function local                          Closeness to
 Iteration       rmse            minima          Step-size        optimal solution
     1         0.00009         24832.60           6.00               46.80394
     2         0.00012         24491.93          29.78               58.92927
     3         0.00014         21486.74           9.76               59.40296
     4         0.00010         20100.15           7.82               36.86658
     5         0.00010         19653.02           5.40               32.11689
     6         0.00009         19344.57           6.25               27.88253
     7         0.00009         19179.94           6.32               26.61265
     8         0.00008         19105.13           6.35               25.79584
     9         0.00008         19105.13           0.00               25.79584

--------------------------- Pyramid Level = 1 --------------------------------
              Normalized     Function local                          Closeness to
 Iteration       rmse            minima          Step-size        optimal solution
     1         0.00005         22834.26           6.43               25.64158
     2         0.00006         21498.26          20.85               63.44257
```

```
       3          0.00007          20173.48          11.87                    81.18027
       4          0.00006          20039.11          21.12                    51.90494
       5          0.00006          19909.26           3.44                    51.33763
       6          0.00005          19738.00           4.92                    45.68723
       7          0.00005          19542.03           6.75                    37.99851
       8          0.00005          19430.34           7.47                    30.56819
       9          0.00005          19396.99           4.20                    26.59914
```

Display the same slice of the fixed volume and the corresponding slice of the registered volume. Observe the improved alignment of the volumes.

```
figure
imshowpair(fixed(:,:,10),reg(:,:,10))
```



Input Arguments

`moving` — Image or volume to be registered
2-D numeric matrix | 3-D numeric array

Image or volume to be registered, specified as a 2-D numeric matrix or 3-D numeric array, respectively. The size of `moving` must be the same as the size of `fixed`.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`fixed` — Reference image or volume
2-D numeric matrix | 3-D numeric array

Reference image or volume, specified as a 2-D numeric matrix or 3-D numeric array, respectively. The size of `fixed` must be the same as the size of `moving`.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1,...,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example: `[dispField,reg] = imregdeform(moving,fixed,NumPyramidLevels=6)` registers `moving` to `fixed` using six pyramid levels.

`GridSpacing` — Grid spacing
two-element numeric vector | three-element numeric vector

Grid spacing, specified as a two-element or three-element numeric vector.

If `moving` and `fixed` are 2-D grayscale images, `GridSpacing` must be a two-element vector, and its default value is `[4 4]`. If `moving` and `fixed` are 3-D intensity volumes, `GridSpacing` must be a three-element vector, and its default value is `[4 4 4]`. Smaller values of `GridSpacing` specify a finer grid resolution.

Data Types: `double`

`PixelResolution` — Pixel size
two-element numeric vector | three-element numeric vector

Pixel size, specified as a two-element or three-element numeric vector. Values are in millimeters.

If `moving` and `fixed` are 2-D grayscale images, `PixelResolution` must be a two-element vector, and its default value is `[1 1]`. If `moving` and `fixed` are 3-D intensity volumes, `PixelResolution` must be a three-element vector, and its default value is `[1 1 1]`.

Data Types: `double`

`NumPyramidLevels` — Number of multiresolution pyramid levels
3 (default) | positive integer

Number of multiresolution pyramid levels, specified as a positive integer.

- If `moving` and `fixed` are 2-D grayscale images of size *M*-by-*N*, then the value of `NumPyramidlevels` must satisfy the condition `min([M N]) > (2^NumPyramidLevels)*0.7`.
- If `moving` and `fixed` are 3-D intensity volumes of size *M*-by-*N*-by-*P*, then the value of `NumPyramidlevels` must satisfy the condition `min([M N P]) > (2^NumPyramidLevels)*0.7`.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`GridRegularization` — Weighing factor for grid displacement regularization
`0.11` (default) | nonnegative scalar

Weighing factor for grid displacement regularization, specified as a nonnegative scalar.

A large value for `GridRegularization` can create a smooth output displacement field, whereas a small value can create more localized displacements.

Data Types: `double`

`DisplayProgress` — Progress information output
`true` or `1` (default) | `false` or `0`

Progress information output, specified as a numeric or logical `1` (`true`) or `0` (`false`). Specify `DisplayProgress` as `true` to display information such as the number of iterations, normalized root mean square error (RMSE), function local minima, step size, and closeness to optimal solution.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logical`

Output Arguments

`dispField` — Displacement field
3-D numeric array | 4-D numeric array

Displacement field, returned as a 3-D or 4-D numeric array.

If `moving` and `fixed` are 2-D grayscale images of size *M*-by-*N*, the displacement field `dispField` is a 3-D numeric array of size *M*-by-*N*-by-2, where `dispField(:,:,1)` and `dispField(:,:,2)` contain the displacements in the *X*- and *Y*- directions, respectively. If `moving` and `fixed` are 3-D intensity volumes of size *M*-by-*N*-by-*P*, the displacement field `dispField` is a 4-D numeric array of size *M*-by-*N*-by-*P*-by-3, where `dispField(:,:,1)`, `dispField(:,:,2)`, and `dispField(:,:,3)` contain the displacements in the *X*-, *Y*- and *Z*- directions, respectively.

Data Types: `double`

`reg` — Registered image or volume
2-D numeric matrix | 3-D numeric array

Registered image or volume, returned with the same size and data type as `moving`.

Version History
Introduced in R2022b

References

[1] Vishnevskiy, Valery, Tobias Gass, Gabor Szekely, Christine Tanner, and Orcun Goksel. “Isotropic Total Variation Regularization of Displacements in Parametric Image Registration.” *IEEE Transactions on Medical Imaging* 36, no. 2 (February 2017): 385–95. https://doi.org/10.1109/TMI.2016.2610583.

See Also

Functions
`imregdemons` | `imregicp` | `imregmoment` | `imreggroupwise`

Topics
“Medical Image Registration”

imreggroupwise

Groupwise deformable registration

Syntax

```
[dispField,reg] = imreggroupwise(moving)
[dispField,reg] = imreggroupwise(moving,Name=Value)
```

Description

The `imreggroupwise` function uses the total variation method to perform deformable registration of slices in a series of grayscale images. You can use this function to reduce sliding motion between slices in a series of medical images, such as a timeseries. Registering all slices of the series to one of the slices using deformable registration in a `for` loop can introduce bias towards the artifacts of one slice in all the slices. In contrast, the `imreggroupwise` function reduces the overall range of sliding motion across all slices.

`[dispField,reg] = imreggroupwise(moving)` transforms the slices in the image series `moving` so that they are groupwise registered, and returns the displacement field `dispField` and the registered image series `reg`.

`[dispField,reg] = imreggroupwise(moving,Name=Value)` specifies options for the total variation method using one or more optional name-value arguments.

Examples

Groupwise Registration of MRI Slices

Load an MRI volume into the workspace.

```
imgs = load("mristack.mat");
imgs = squeeze(imgs.mristack);
```

Pad the slices of the MRI volume to allow for rotation.

```
imgs = padarray(imgs,[10 10],"both");
```

Extract the 10th slice of the MRI volume, to use to generate the moving image series.

```
paddedSlice = imgs(:,:,10);
```

Generate the moving image series by rotating the slice by different angles.

```
moving = zeros(276,276,27);
for i = 1:27
  moving(:,:,i) = imrotate(paddedSlice,i,"crop");
end
```

Perform groupwise registration of the slices of the moving image series.

```
[dispField,reg] = imreggroupwise(moving,GridRegularization=0.05);
```

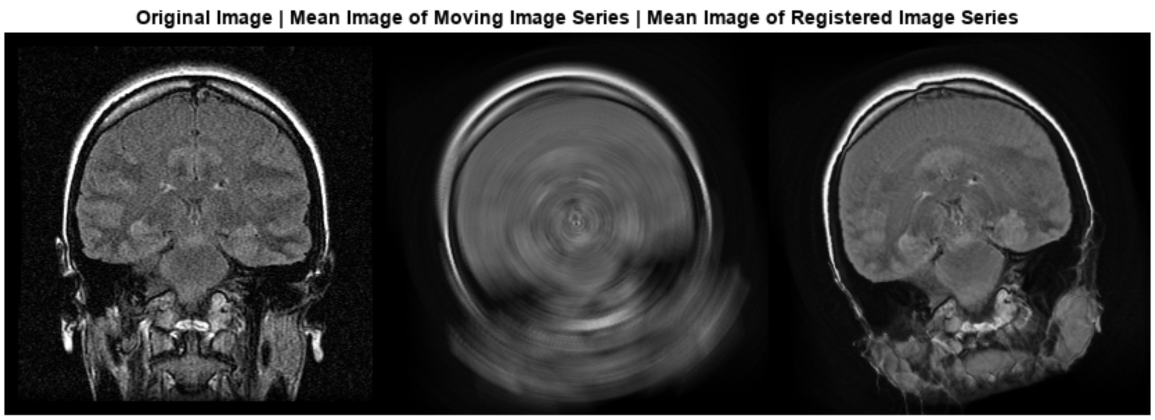
```
--------------------------- Pyramid Level = 3 --------------------------------
                                                    Closeness to
 Iteration         f(x)          Step-size         optimal solution
     1         160720.92         11.50                18.19431
     2         146622.17         41.61                31.55621
     3         137582.30         54.80                37.51630
     4         131892.89         21.32                30.22352
     5         129146.80         12.05                27.98123
     6         126260.51         18.80                34.57209
     7         123292.77         21.42                32.76271
     8         119756.72         34.44                26.53227
     9         117145.16         27.64                33.13160
    10         114257.37         28.72                39.06536
    11         111212.99         50.30                35.02401
    12         108848.93         28.47                27.27366
    13         106838.15         16.68                21.59996
    14         104141.37         48.40                29.34133
    15         103034.70         43.98                36.78617
    16         101421.24         11.33                21.98658
    17          99846.91         25.65                19.28558
    18          98950.73         20.29                22.02194
    19          97683.82         55.50                23.47823
    20          97058.47         35.79                56.52787
    21          95693.10         12.29                39.94096
    22          94308.28         31.93                50.77296
    23          93620.93         21.01                23.42839
    24          92756.49         43.04                26.44428
    25          92160.48         39.07                20.02188
    26          91649.86         20.14                23.32969
    27          90848.45         52.28                12.75790
    28          90785.24         35.18                50.98939
    29          89193.25         13.04                24.02603
    30          87913.57         26.84                23.86608
    31          87261.74         13.25                27.18752
    32          86609.40         28.99                20.23212
    33          86403.18         36.29                19.85028
    34          86171.97         12.29                14.73627
    35          85991.59         43.41                19.17823
    36          85880.43         31.97                52.82440
    37          84848.06         22.60                38.11074
    38          83896.58         24.78                26.63796
    39          83221.23         17.79                31.42078
    40          82422.75         15.26                17.38349
    41          82386.70         42.07                30.26392
    42          82118.45         10.10                20.60638
    43          81714.94         31.01                27.63559
    44          81475.85         25.61                35.72504
    45          81219.57         11.81                30.83450
    46          80900.59         42.13                21.99698
    47          80701.15         27.56                46.08761
    48          79735.79         18.84                29.77860
    49          78885.47         19.72                21.33987
    50          78443.85         14.00                33.84345
    51          78025.38         19.61                21.81821
    52          77820.46         10.80                27.58759
    53          77566.30         14.89                14.12064
    54          77356.17         16.30                24.02455
    55          77210.20         17.62                13.38399
```

```
     56        77175.39          16.57                13.65389
     57        77139.47          16.67                14.31732
     58        77103.48          16.91                14.75186
     59        77052.18          18.54                13.45865
     60        77007.15          17.58                11.45708
     61        76995.53           6.92                12.12129

--------------------------- Pyramid Level = 2 -------------------------------
                                                    Closeness to
  Iteration         f(x)         Step-size         optimal solution
      1        90397.57           9.98                16.07776
      2        88598.63          13.10                23.03916
      3        86659.87          23.18                36.97704
      4        85346.69          26.94                46.28403
      5        83821.96           9.70                20.80432
      6        82514.48          16.43                25.34133
      7        81522.62          18.21                27.43884
      8        80714.91          45.26                44.56642
      9        79515.86          16.33                17.96243
     10        78667.76          10.10                25.12929
     11        77756.18          24.19                27.73775
     12        77571.40          32.06                51.91825
     13        76629.12           7.86                22.85993
     14        75929.67          19.14                31.05180
     15        75590.59          16.23                14.57161
     16        75563.08          47.79                52.11658
     17        74964.02          19.25                34.85287
     18        74383.90           8.57                36.63964
     19        73887.61          25.26                20.34162
     20        73676.18           8.94                12.66080
     21        73360.07          26.43                19.09864
     22        73190.90          31.92                33.94272
     23        73032.36          10.59                20.21763
     24        72764.07          32.08                14.04518
     25        72626.99          24.21                27.84766
     26        72493.83          30.79                28.66531
     27        72090.89          18.12                17.14031
     28        71660.52          18.11                11.76067
     29        71269.90          15.98                24.21175
     30        71067.73          22.38                15.58385
     31        70985.12           8.50                14.58411
     32        70888.81          11.41                13.56274
     33        70796.71          14.63                12.23691
     34        70729.12          15.27                12.57831
     35        70721.71          14.16                12.59730
     36        70716.14           5.36                12.87683

--------------------------- Pyramid Level = 1 -------------------------------
                                                    Closeness to
  Iteration         f(x)         Step-size         optimal solution
      1        80123.51          13.71                13.80900
      2        78453.64          14.18                11.56824
      3        76729.32          27.24                26.34447
      4        75799.10          26.53                37.72128
      5        74508.05           9.21                19.73064
      6        73460.13          20.96                27.40239
      7        72800.90          16.67                26.18356
      8        71919.69          32.43                14.43351
```

```
    9        71545.75       30.56           32.57132
   10        70717.01       15.97           18.49375
   11        70052.45       19.04           22.48044
   12        69587.50       17.23           23.88982
   13        69094.55       28.76           22.76715
   14        68720.48       24.05           24.80507
   15        68212.79       16.41           14.46844
   16        67766.80       23.74           14.20388
   17        67589.19       23.70           29.27166
   18        67121.86       14.29           13.96609
   19        66779.83       23.35           12.63738
   20        66611.24       19.61           21.53575
   21        66390.03       23.81           15.41449
   22        66226.75        9.53           10.36505
   23        66174.37       35.33           14.71976
   24        66046.98        6.29           11.75380
   25        65777.74       20.41           21.22998
   26        65694.36       26.64           37.03862
   27        65478.70        9.46           33.20081
   28        65330.62       28.51           30.64879
   29        65193.94       14.45           30.87433
   30        64969.61       20.04           27.73873
   31        64863.63        7.47           23.45422
   32        64742.49       10.56           18.98134
   33        64655.24       11.63           19.72796
   34        64589.34       12.86           11.98109
   35        64575.96        6.52           13.75064
   36        64563.45        8.01           13.59998
   37        64558.37        3.78           12.33042
```

Visualize the output of the groupwise registration. The mean image of the moving image series shows that the slices are misaligned. In contrast, the mean image of the registered image series indicates alignment across slices.

```
figure
imshow([paddedSlice,mean(moving,3),mean(reg,3)])
title("Original Image | Mean Image of Moving Image Series | Mean Image of Registered Image Serie
```


Original Image | Mean Image of Moving Image Series | Mean Image of Registered Image Series

Input Arguments

`moving` — Image series to be registered
3-D numeric array

Image series to be registered, specified as a 3-D numeric array. The slices in the image series must capture the same anatomical slice of the body. For example, the image series can be a collection of the same slice imaged at different times.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1,...,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example: `[dispField,reg] = imreggroupwise(moving,NumPyramidLevels=6)` registers the slices of `moving` using six pyramid levels.

`GridSpacing` — Grid spacing
`[4 4]` (default) | two-element numeric vector

Grid spacing, specified as a two-element numeric vector. Smaller values of `GridSpacing` specify a finer grid resolution.

Data Types: `double`

`PixelResolution` — Pixel size
`[1 1]` (default) | two-element numeric vector

Pixel size, specified as a two-element numeric vector. Values are in millimeters.

Data Types: `double`

`NumPyramidLevels` — Number of multiresolution pyramid levels
`3` (default) | positive integer

Number of multiresolution pyramid levels, specified as a positive integer.

If `moving` is of size *M*-by-*N*-by-*P*, then the value of `NumPyramidlevels` must satisfy the condition `min([M N P]) > (2^NumPyramidLevels)*0.7`.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`GridRegularization` — Weighing factor for grid displacement regularization
`0.11` (default) | nonnegative scalar

Weighing factor for grid displacement regularization, specified as a nonnegative scalar.

A large value for `GridRegularization` can create a smooth output displacement field, whereas a small value can create more localized displacements.

Data Types: `double`

`DisplayProgress` — Progress information output
`true` or `1` (default) | `false` or `0`

Progress information output, specified as a numeric or logical `1` (`true`) or `0` (`false`). Specify `DisplayProgress` as `true` to display information such as the number of iterations, normalized root mean square error (RMSE), function local minima, step size, and closeness to optimal solution.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logical`

Output Arguments

`dispField` — Displacement field
4-D numeric array

Displacement field, returned as a 4-D numeric array.

If `moving` is of size *M*-by-*N*-by-*P*, the displacement field `dispField` is a 4-D numeric array of size *M*-by-*N*-by-2-by-*P*, where `dispField(:,:,1,:)` and `dispField(:,:,2,:)` contain the displacements for each of the slices in the *X*- and *Y*- directions, respectively.

`reg` — Registered image series
3-D numeric array

Registered image series, returned with the same size and data type as `moving`.

Version History

Introduced in R2022b

References

[1] Vishnevskiy, Valery, Tobias Gass, Gabor Szekely, Christine Tanner, and Orcun Goksel. “Isotropic Total Variation Regularization of Displacements in Parametric Image Registration.” *IEEE Transactions on Medical Imaging* 36, no. 2 (February 2017): 385–95. https://doi.org/10.1109/TMI.2016.2610583.

See Also

Functions
`imregdemons` | `imregicp` | `imregmoment` | `imregdeform`

Topics
“Medical Image Registration”

imregicp

Surface registration using iterative closest point algorithm

Syntax

```
regSurface = imregicp(movingSurface,fixedSurface)
regSurface = imregicp(movingSurface,fixedSurface,Name=Value)
[regSurface,tform] = imregicp( ___ )
[regSurface,tform,rmse] = imregicp( ___ )
```

Description

The `imregicp` function uses the iterative closest point (ICP) algorithm for rigid registration of surfaces. Use this function to register surfaces extracted from medical volumes.

`regSurface = imregicp(movingSurface,fixedSurface)` transforms the surface `movingSurface`, so that it is registered with the reference surface `fixedSurface` using the ICP algorithm. The function returns the registered surface `regSurface`.

`regSurface = imregicp(movingSurface,fixedSurface,Name=Value)` specifies options for the ICP algorithm using one or more optional name-value arguments.

`[regSurface,tform] = imregicp( ___ )` returns the transformation `tform` between the moving surface and the registered surface, in addition to any combination of input arguments from previous syntaxes.

`[regSurface,tform,rmse] = imregicp( ___ )` returns the root mean squared error (RMSE) `rmse` of the Euclidean distance between the inlier points of the aligned surfaces `regSurface` and `fixedSurface`, in addition to any combination of input arguments from previous syntaxes.

Examples

Surface Registration of Isosurfaces

Load a MAT file containing intensity volume data into the workspace. Extract the volume data.

```
load(fullfile(toolboxdir("images"),"imdata","BrainMRILabeled","images","vol_001.mat"));
V = vol;
```

Specify an isovalue for isosurface extraction.

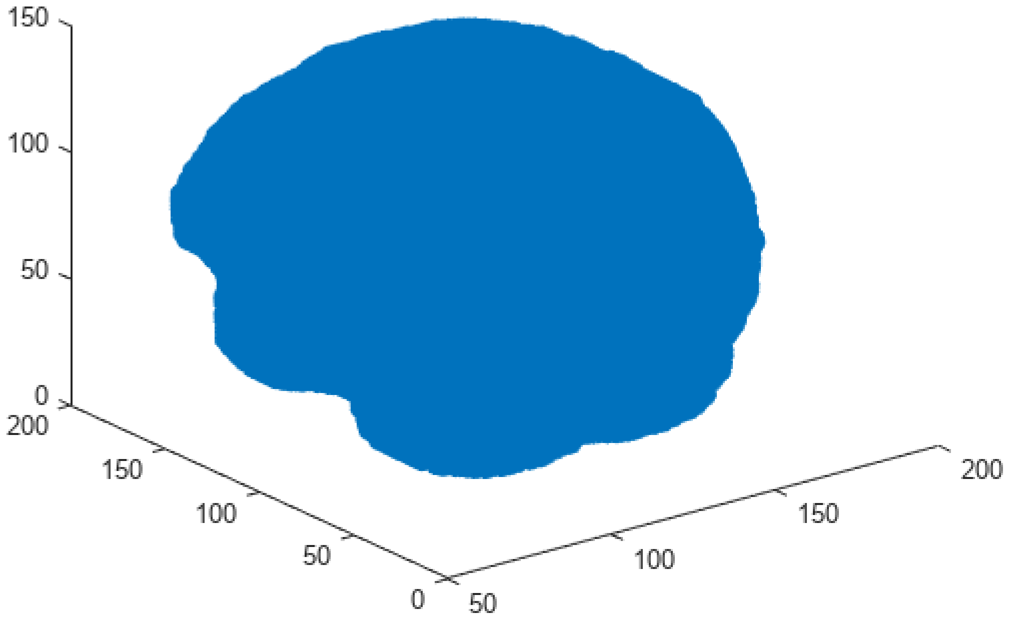
```
isovalue = 0.5;
```

Extract the isosurface of the reference volume at the specified isovalue.

```
[~,fixedSurface] = extractIsosurface(V,isovalue);
```

Display and inspect the reference isosurface.

```
figure
plot3(fixedSurface(:,2),fixedSurface(:,1),fixedSurface(:,3),"o")
```



Create a rigid transformation, as an `affinetform3d` object, for the reference volume.

```
theta = pi/18;
affineMatrix = [cos(theta)  sin(theta) 0  5; ...
                -sin(theta) cos(theta) 0  5; ...
                0           0          1  10; ...
                0           0          0  1];
tform_rigid = affinetform3d(affineMatrix);
```

Transform the reference volume using the rigid transformation.

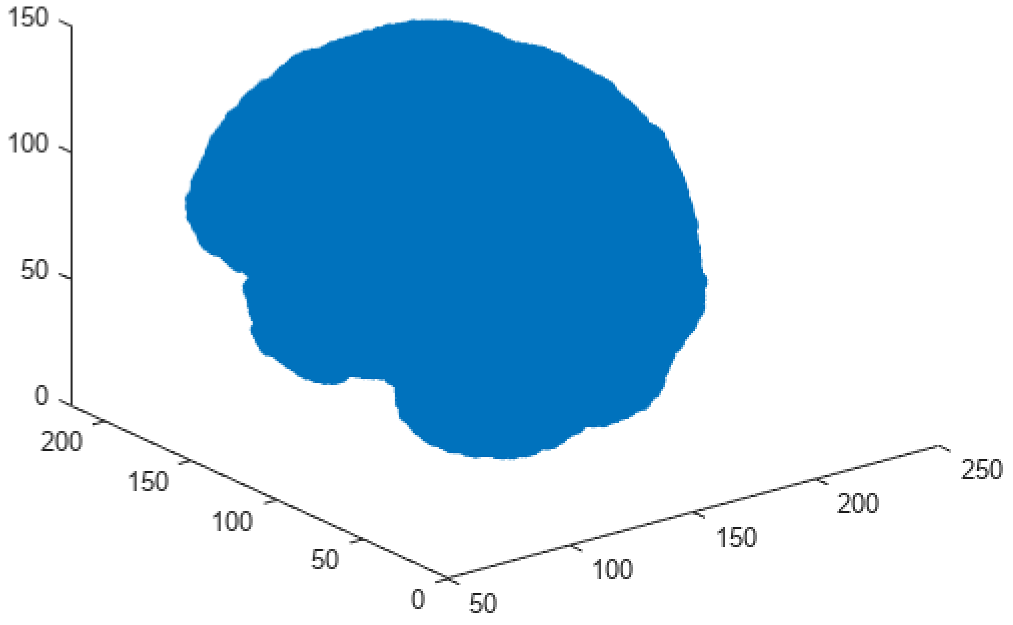
```
tformV = imwarp(V,tform_rigid);
```

Extract the isosurface of the transformed volume at the specified isovalue.

```
[~,movingSurface] = extractIsosurface(tformV,isovalue);
```

Display and inspect the transformed isosurface.

```
figure
plot3(movingSurface(:,2),movingSurface(:,1),movingSurface(:,3),"o")
```



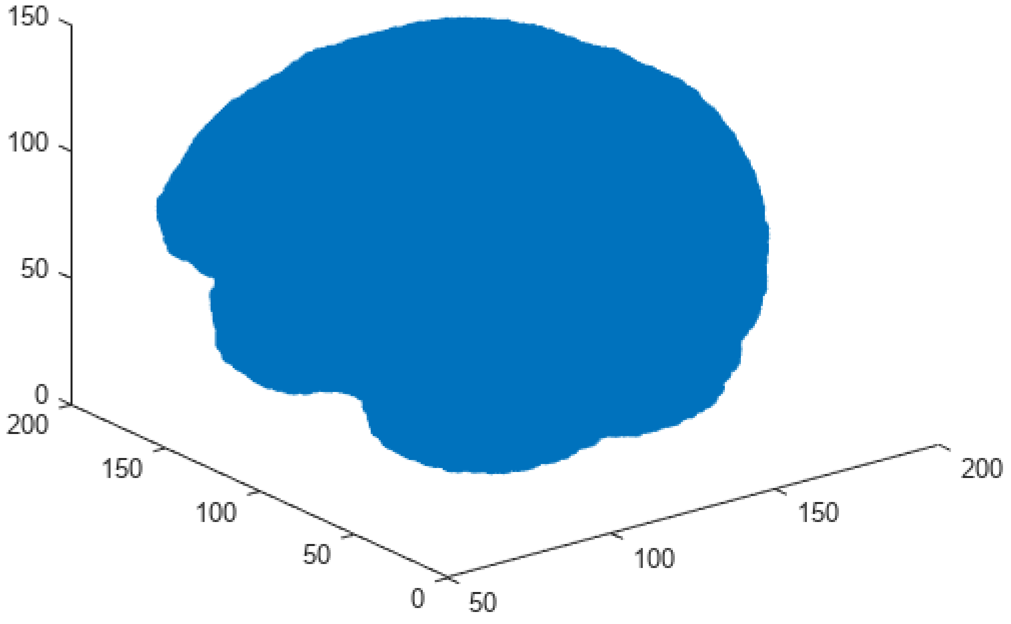
Register the transformed isosurface with respect to the reference isosurface using surface registration.

```
regSurface = imregicp(movingSurface,fixedSurface,DistanceThreshold=30);
```

```
iterations = 1    Fitness = 1.0000    inlierRmse = 11.2527
iterations = 2    Fitness = 1.0000    inlierRmse = 8.7565
iterations = 3    Fitness = 1.0000    inlierRmse = 6.9034
iterations = 4    Fitness = 1.0000    inlierRmse = 5.5111
iterations = 5    Fitness = 1.0000    inlierRmse = 4.5172
iterations = 6    Fitness = 1.0000    inlierRmse = 3.7946
iterations = 7    Fitness = 1.0000    inlierRmse = 3.2685
iterations = 8    Fitness = 1.0000    inlierRmse = 2.8860
iterations = 9    Fitness = 1.0000    inlierRmse = 2.6006
iterations = 10    Fitness = 1.0000    inlierRmse = 2.3830
iterations = 11    Fitness = 1.0000    inlierRmse = 2.2119
iterations = 12    Fitness = 1.0000    inlierRmse = 2.0698
iterations = 13    Fitness = 1.0000    inlierRmse = 1.9481
iterations = 14    Fitness = 1.0000    inlierRmse = 1.8440
iterations = 15    Fitness = 1.0000    inlierRmse = 1.7480
iterations = 16    Fitness = 1.0000    inlierRmse = 1.6617
iterations = 17    Fitness = 1.0000    inlierRmse = 1.5824
iterations = 18    Fitness = 1.0000    inlierRmse = 1.5087
iterations = 19    Fitness = 1.0000    inlierRmse = 1.4391
iterations = 20    Fitness = 1.0000    inlierRmse = 1.3745
```

Display and inspect the registered isosurface.

```
figure
plot3(regSurface(:,2),regSurface(:,1),regSurface(:,3),"o")
```



Input Arguments

`movingSurface` — Surface to be registered
M-by-3 numeric matrix

Surface to be registered, specified as an *M*-by-3 numeric matrix. *M* is the number of points in the surface. Each row contains the 3-D coordinates of a surface point.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`fixedSurface` — Reference surface
M-by-3 numeric matrix

Reference surface, specified as an *M*-by-3 numeric matrix. *M* is the number of points in the surface. Each row contains the 3-D coordinates of a surface point.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1,...,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example: `regSurface = imregicp(movingSurface,fixedSurface,Metric="pointToPlane")` registers the surfaces using the `"pointToPlane"` minimization metric for the ICP algorithm.

`Metric` — Minimization metric
`"pointToPoint"` (default) | `"pointToPlane"`

Minimization metric, specified as `"pointToPoint"` or `"pointToPlane"`.

When registering planar surfaces, you can use the `"pointToPlane"` metric to reduce the number of iterations in the algorithm, but at a higher computational complexity for each iteration.

Data Types: `char` | `string`

`MaxIterations` — Maximum number of iterations
`20` (default) | positive integer

Maximum number of iterations for the ICP algorithm, specified as a positive integer.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`InitialTransform` — Initial rigid transformation
`affinetform3d()` (default) | `affinetform3d` object

Initial rigid transformation, specified as an `affinetform3d` object.

`OutlierRemoval` — Parameters for outlier removal
`[20 2]` (default) | two-element vector

Parameters for outlier removal, specified as a two-element vector of the form [*n* *r*].

The function considers the surface points with fewer than *n* neighbors within a sphere of radius *r* to be outliers, and removes them from consideration for the registered surface. *n* must be a positive integer, and *r* must be positive. Specifying a high value for *n* with a low value for *r* causes the function to remove outliers aggressively.

Data Types: `single` | `double`

`DistanceThreshold` — Distance threshold for closest point
`0.1` (default) | positive scalar

Distance threshold for closest point, specified as a positive scalar. The `DistanceThreshold` is the maximum distance at which the k-nearest neighbor search considers a point in `movingSurface` as a possible correspondent for a point in `fixedSurface`. Increasing the distance threshold can enable you to detect a correspondence in images with sparse surface points, but can also cause the algorithm to return false positives.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`NumPoints` — Number of points used for local plane fitting
`6` (default) | positive integer greater than or equal to `3`

Number of points used for local plane fitting, specified as a positive integer greater than or equal to `3`. To specify this argument, you must specify `Metric` as `"pointToPlane"`. Smaller values of `NumPoints` can result in faster computation, but can reduce accuracy.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`Verbose` — Progress information output
`true` or `1` (default) | `false` or `0`

Progress information output, specified as a numeric or logical `1` (`true`) or `0` (`false`). Specify `Verbose` as `true` to display information such as the number of iterations, fitness score, and inlier RMSE value.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logical`

Output Arguments

`regSurface` — Registered surface
M-by-3 numeric matrix

Registered surface, returned as an *M*-by-3 numeric matrix. *M* is the number of points in the surface. Each row contains the 3-D coordinates of a surface point.

Data Types: `double` | `single`

`tform` — Rigid transformation
`affinetform3d` object

Rigid transformation, returned as an `affinetform3d` object.

`rmse` — Root mean square error
numeric scalar

Root mean square error of the Euclidean distance between the inlier points of the aligned surfaces `regSurface` and `fixedSurface`, returned as a numeric scalar.

Data Types: `double`

Limitations

- The `imregicp` function is not supported on Mac computers with Apple silicon chips.

Version History

Introduced in R2022b

References

[1] Besl, P.J., and Neil D. McKay. “A Method for Registration of 3-D Shapes.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, no. 2 (February 1992): 239–56. https://doi.org/10.1109/34.121791.

See Also

Functions
`imregister` | `imregmoment` | `imregdeform` | `imreggroupwise`

Objects
`affinetform3d`

Topics
"Medical Image Registration"

imregmoment

Fast registration of grayscale images or intensity volumes using moment of mass method

Syntax

```
[tform,reg] = imregmoment(moving,fixed)
[tform,reg] = imregmoment(moving,movingRef,fixed,fixedRef)
[tform,reg] = imregmoment(___,MedianThresholdBitmap=mtb)
```

Description

The `imregmoment` function uses the moment of mass method for fast, similarity registration of grayscale images or intensity volumes. You can use this function to register multimodal medical images or volumes as a preprocessing step before multimodal medical image fusion.

`[tform,reg] = imregmoment(moving,fixed)` transforms the grayscale image or intensity volume `moving`, so that it is registered with the reference image or volume `fixed`, and returns the transformation `tform` and the registered image or volume `reg`.

`[tform,reg] = imregmoment(moving,movingRef,fixed,fixedRef)` specifies the spatial referencing information `movingRef` and `fixedRef` for `moving` and `fixed`, respectively.

`[tform,reg] = imregmoment(___,MedianThresholdBitmap=mtb)` specifies whether to threshold `moving` and `fixed` before registration, in addition to any combination of input arguments from previous syntaxes.

Examples

Fast Registration of Multimodal Medical Images

The data used in this example is a modified version of the 3-D CT and MRI datasets provided by Dr. Michael Fitzpatrick as part of The Retrospective Image Registration Evaluation (RIRE) Dataset. The modified dataset contains the CT and MRI scans stored in the NRRD file format. The size of the entire data set is approximately 35 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","/MedicalRegistrationNRRDdata.zi
filepath = fileparts(zipFile);
unzip(zipFile,filepath);
```

Read an MRI volume into the workspace as the reference volume. Get the spatial referencing information for the MRI volume.

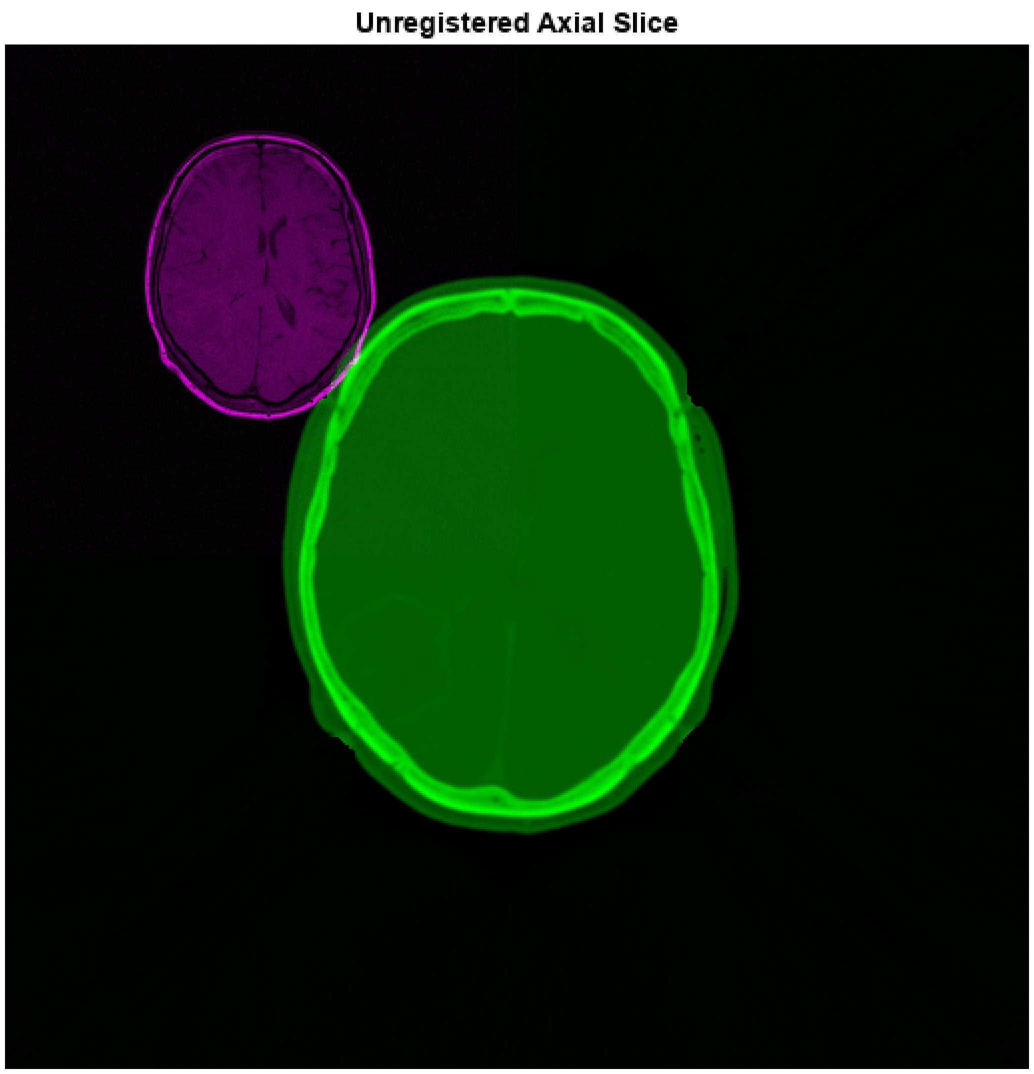
```
fixedFile = fullfile(filepath,"supportfilesNRRD/Patient007MRT1.nrrd");
fixedVol  = medicalVolume(fixedFile);
fixed = fixedVol.Voxels;
fixedRef  = imref3d(size(fixed),fixedVol.VoxelSpacing(2),fixedVol.VoxelSpacing(1),fixedVol.VoxelS
```

Read a CT volume into the workspace as the volume to be registered. Get the spatial referencing information for the CT volume.

```
movingFile = fullfile(filepath,"supportfilesNRRD/Patient007CT.nrrd");
movingVol = medicalVolume(movingFile);
moving = movingVol.Voxels;
movingRef = imref3d(size(moving),movingVol.VoxelSpacing(2),movingVol.VoxelSpacing(1),movingVol.Vo
```

Compare a slice of the unregistered volume to a corresponding slice of the reference volume.

```
centerFixed = size(fixed)/2;
centerMoving = size(moving)/2;
figure
imshowpair(moving(:,:,centerMoving(3)),fixed(:,:,centerFixed(3)))
title("Unregistered Axial Slice")
```

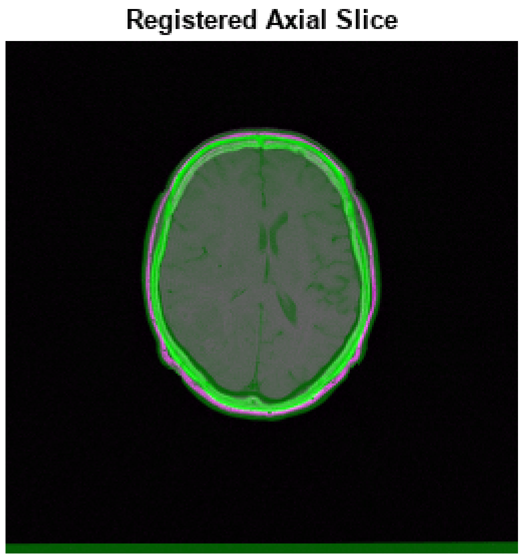


Register the volumes, and return the registered volume and the transformation between the moving and fixed volumes.

```
[tform,reg] = imregmoment(moving,movingRef,fixed,fixedRef,MedianThresholdBitmap=true);
```

Compare the same slice of the registered volume to the corresponding slice of the reference volume.

```
figure
imshowpair(reg(:,:,centerFixed(3)),fixed(:,:,centerFixed(3)))
title("Registered Axial Slice")
```



Input Arguments

`moving` — Image or volume to be registered
2-D numeric matrix | 3-D numeric array

Image or volume to be registered, specified as a 2-D numeric matrix or 3-D numeric array, respectively.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `uint8` | `uint16` | `uint32`

`fixed` — Reference image or volume
2-D numeric matrix | 3-D numeric array

Reference image or volume, specified as a 2-D numeric matrix or 3-D numeric array, respectively.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`movingRef` — Spatial referencing information for image or volume to be registered
`imref2d` object | `imref3d` object

Spatial referencing information for the image or volume to be registered, specified as an `imref2d` object or an `imref3d` object, respectively. Use the spatial referencing inputs when the images or volumes differ in size by a scaling factor.

fixedRef — Spatial referencing information for reference image or volume
imref2d object | imref3d object

Spatial referencing information for the reference image or volume, specified as an imref2d object or an imref3d object, respectively. Use the spatial referencing inputs when the images or volumes differ in size by a scaling factor.

mtb — Median threshold bitmap processing
false or 0 (default) | true or 1

Median threshold bitmap processing, specified as a numeric or logical 0 (false) or 1 (true). Specify MedianThresholdBitmap as true to threshold moving and fixed. This can improve registration if the images have been captured using different sensors or have different intensity levels.

Data Types: single | double | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64 | logical

Output Arguments

tform — Similarity transformation
affinetform2d object | affinetform3d object

Similarity transformation, returned as an affinetform2d object or an affinetform3d object.

reg — Registered image or volume
2-D numeric matrix | 3-D numeric array

Registered image or volume, returned with the same size as fixed and same data type as moving.

Version History
Introduced in R2022b

See Also

Functions
imregmtb | imregicp | imregdeform | imreggroupwise

Objects
imref2d | imref3d | affinetform2d | affinetform3d

Topics
“Medical Image Registration”

isnrrd

Check if file is valid NRRD file

Syntax

```
tf = isnrrd(filename)
```

Description

`tf = isnrrd(filename)` checks if the specified file is a valid nearly raw raster data (NRRD) file.

Examples

Check If File Is Valid NRRD File

Check whether a file is a valid NRRD format file. The file is part of a data set containing the 3-D CT and MRI scans from The Retrospective Image Registration Evaluation (RIRE) Dataset, converted to the NRRD file format. The original data set was provided by Dr. Michael Fitzpatrick. For more information, see the RIRE Project homepage. The size of the entire data set is approximately 35 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalRegistrationNRRDdata.zi
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the name of the NRRD file to check.

```
filename = fullfile(filepath,"supportfilesNRRD","Patient007CT.nrrd");
```

Check if `filename` is a valid NRRD file. The `isnrrd` function determines that the file is valid.

```
isnrrd(filename)
```

```
ans = logical
   1
```

Input Arguments

`filename` — Name of NRRD file
string scalar | character vector

Name of the NRRD file, specified as a string scalar or a character vector. The file must contain a valid NRRD file header. `filename` can contain the absolute path to the file, a relative path from the current directory, or a relative path from a directory on the MATLAB path.

Data Types: `char` | `string`

Output Arguments

tf — File is valid NRRD file
true or 1 | false or 0

File is a valid NRRD file, returned as a logical 1 (true) or 0 (false). The output value is 1 if filename contains a valid NRRD header. The function can determine the validity of NRRD files that contain image data or are detached header files.

Version History

Introduced in R2022b

See Also
nrrdinfo | nrrdread

specklefilt

Filter image using speckle-reducing anisotropic diffusion

Syntax

```
J = specklefilt(I)
J = specklefilt(I,Name=Value)
```

Description

Coherent imaging modalities, such as ultrasound images, are prone to degradation because of the interference of the transmitted waveform and its echoes. This degradation is called speckle, and is a form of multiplicative noise. The `specklefilt` function uses a speckle-reducing anisotropic diffusion (SRAD) algorithm to reduce the speckle in an image.

`J = specklefilt(I)` filters the image `I` using SRAD and returns the filtered image `J`.

`J = specklefilt(I,Name=Value)` fine-tunes the behavior of the SRAD algorithm using one or more optional name-value arguments.

Examples

Adjust Parameters for Different Smoothing Levels

Import an ultrasound image into the workspace.

```
I = imread("heartUltrasoundImage.png");
```

Filter the image using the speckle-reducing anisotropic diffusion algorithm with default parameters.

```
J1 = specklefilt(I);
```

You can increase the smoothing the filter performs on the image by increasing the degree of smoothing of the algorithm. Specify a degree of smoothing of 0.6, increased from the default value of 0.2.

```
J2 = specklefilt(I,DegreeOfSmoothing=0.6);
```

You can also increase the smoothing by increasing the number of iterations performed by the algorithm. Specify for the algorithm to perform 50 iterations, increased from the default value of 30.

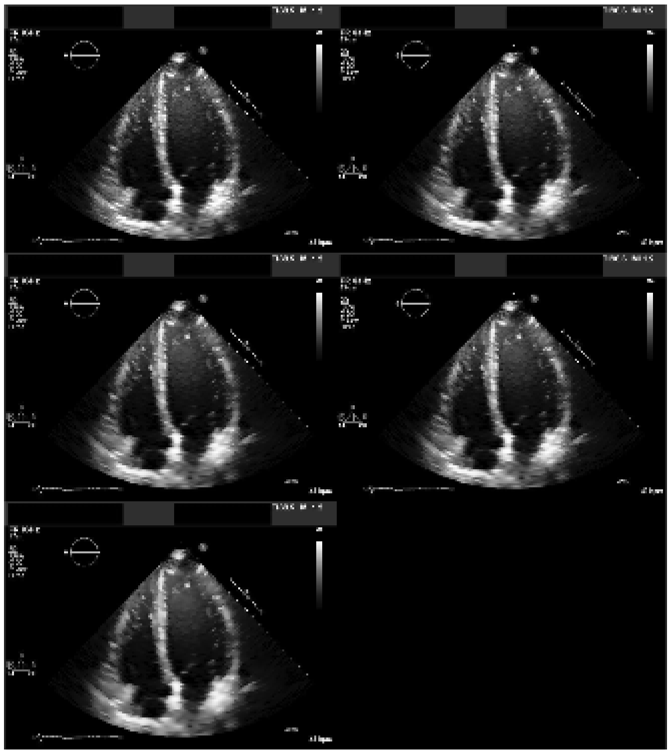
```
J3 = specklefilt(I,NumIterations=50);
```

Filter the image using the SRAD algorithm with both an increased degree of smoothing and greater number of iterations. Set the degree of smoothing to 0.6 and the number of iterations to 50.

```
J4 = specklefilt(I,DegreeOfSmoothing=0.6,NumIterations=50);
```

Display the original image alongside the denoised images, and inspect the different smoothing levels.

```
figure
montage({I J1 J2 J3 J4})
```



Filter Speckle Generated from Different Noise Models

Import an ultrasound image into the workspace.

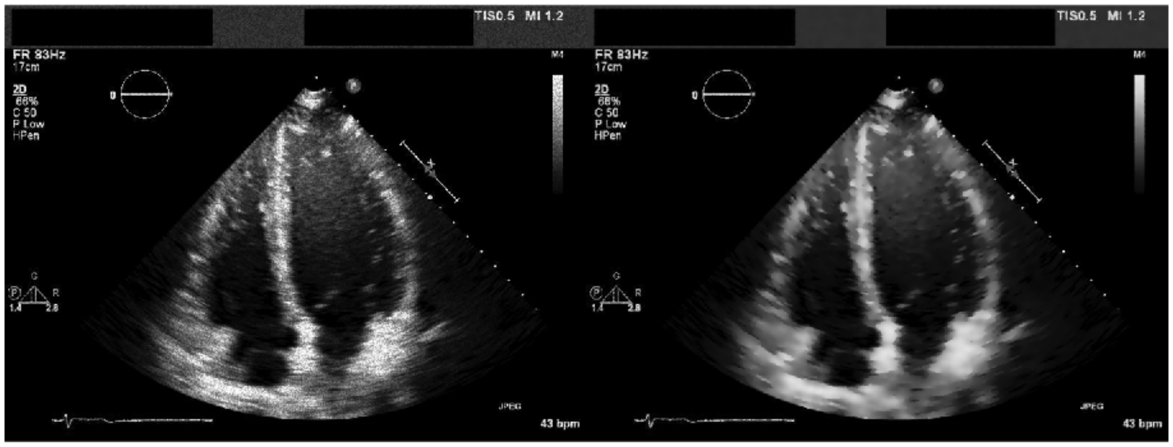
```
I = imread("heartUltrasoundImage.png");
```

Corrupt the image with multiplicative speckle drawn from a uniform distribution.

```
Inoise = imnoise(I,"speckle");
```

Filter the corrupted image to remove speckle. Display the corrupted and denoised images.

```
J = specklefilt(Inoise,DegreeOfSmoothing=0.6,NumIterations=50);
figure
montage({Inoise J})
```



Compute the structural similarity of the denoised image J with the original image I.

```
disp(ssim(I,J))
```

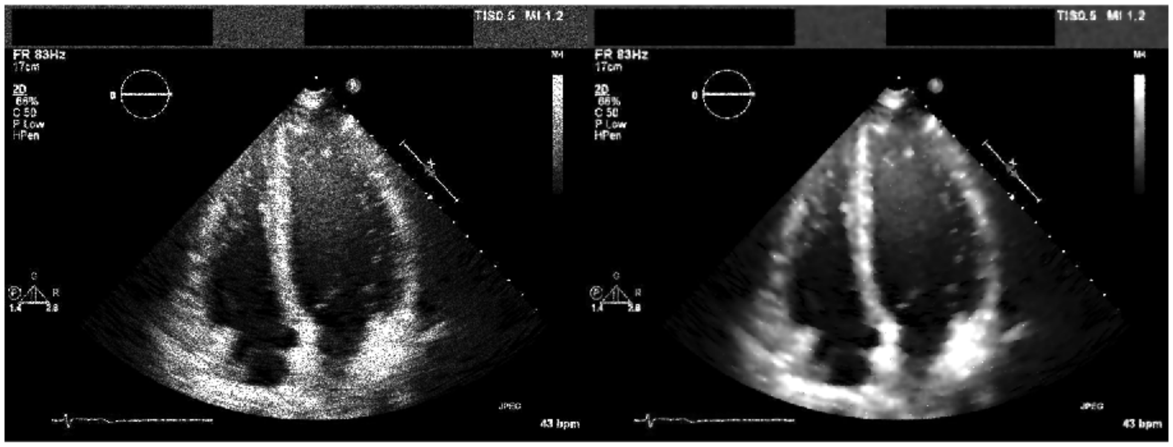
```
    0.9330
```

Corrupt the image with multiplicative speckle drawn from a Rayleigh distribution.

```
I = im2double(I);
Inoise = I.*raylrnd(1,size(I));
```

Filter the corrupted image to remove speckle. Display the corrupted and denoised images.

```
J = specklefilt(Inoise,DegreeOfSmoothing=0.6,NumIterations=50);
figure
montage({Inoise J})
```



Compute the structural similarity between the denoised image J and the original image I.

```
disp(ssim(I,J))

    0.8868
```

Filter Image Sequence

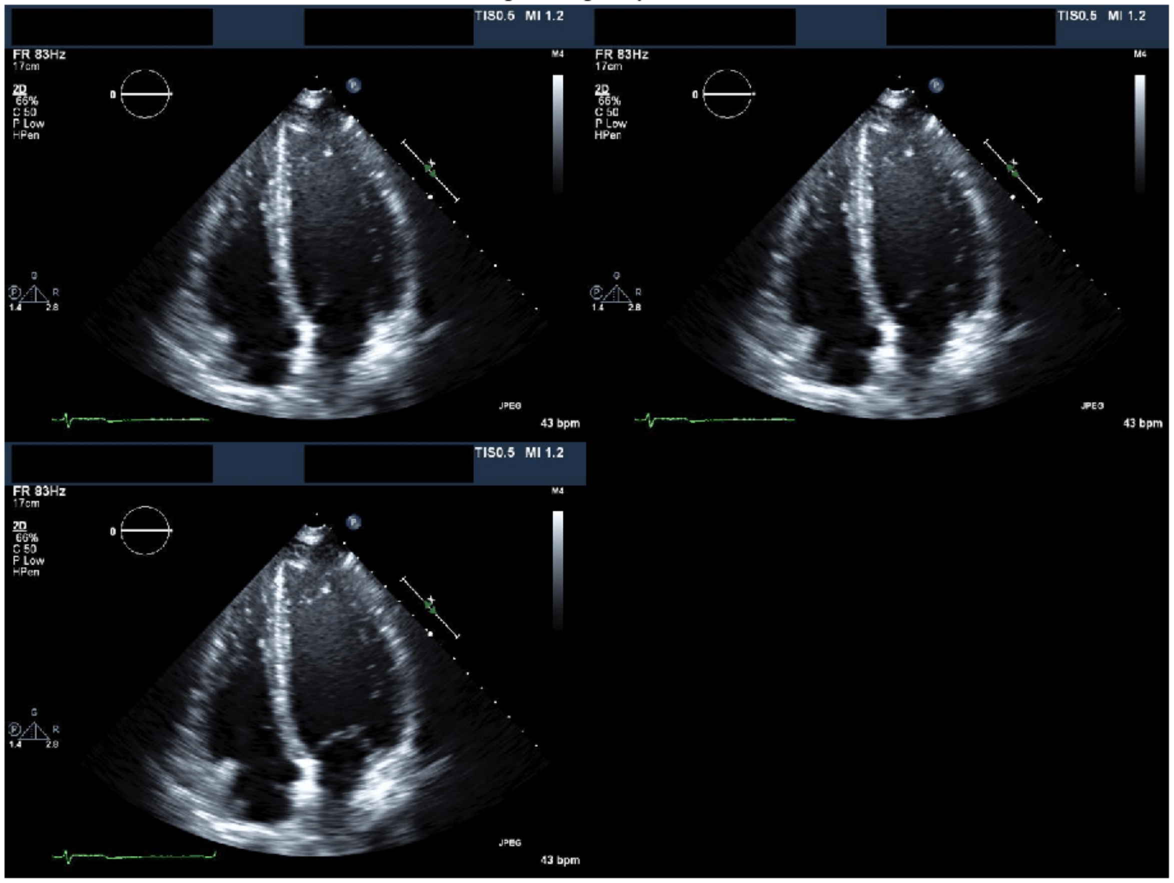
Import an ultrasound image sequence into the workspace.

```
I = dicomread("heartUltrasoundSequence.dcm");
[h,w,c,d] = size(I);
```

View the original image sequence.

```
figure
montage(I)
title("Original Image Sequence")
```

Original Image Sequence



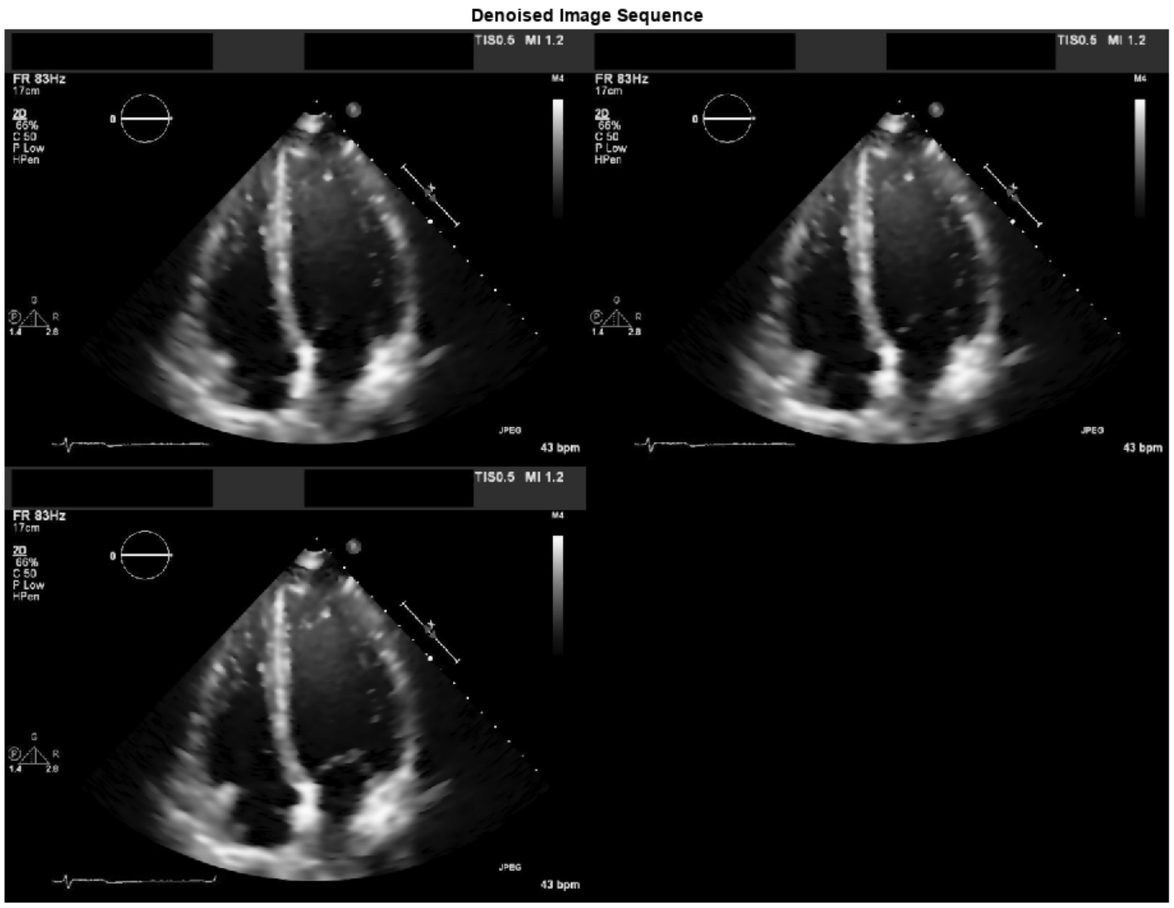
Convert each image in the sequence to grayscale. Filter the grayscale images.

```
J = zeros(h,w,d);
for i = 1:d
```

```
    Ii = im2double(im2gray(I(:,:,:,i)));
    J(:,:,i) = specklefilt(Ii,DegreeOfSmoothing=0.6,NumIterations=50);
end
```

View the denoised image sequence.

```
figure
montage(J)
title("Denoised Image Sequence")
```



Input Arguments

I — Image to be filtered
2-D numeric matrix

Image to be filtered, specified as a 2-D numeric matrix.

Data Types: single | double | int8 | int16 | int32 | int64 | uint8 | uint16 | uint32 | uint64

Name-Value Arguments

Specify optional pairs of arguments as `Name1=Value1,...,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example: `J = specklefilt(I,DegreeOfSmoothing=0.6,NumIterations=50)` applies the SRAD filter to image `I` with a degree of smoothing of 0.6 for 50 iterations, and returns the filtered image `J`.

`DegreeOfSmoothing` — Degree of smoothing
`0.2` (default) | scalar in the range (0,1]

Degree of smoothing of the SRAD filter, specified as a scalar in the range (0, 1]. Increasing the value of `DegreeOfSmoothing` denoises the image to a greater degree.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`NumIterations` — Number of iterations
`30` (default) | positive integer

Number of iterations of the SRAD filter, specified as a positive integer. Increasing the value of `NumIterations` denoises the image to a greater degree.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Output Arguments

`J` — SRAD-filtered image
2-D numeric matrix

SRAD-filtered image, returned as a 2-D numeric matrix of the same size and data type as the input image `I`.

Algorithms

The speckle-reducing anisotropic diffusion used in `specklefilt` combines elements of speckle-reducing filters with the edge-preserving anisotropic diffusion filter used in `imdiffusefilt`. This makes `specklefilt` useful for edge-preserving denoising of images corrupted with speckle.

Version History

Introduced in R2022b

References

[1] Yongjian Yu, and S.T. Acton. “Speckle Reducing Anisotropic Diffusion.” *IEEE Transactions on Image Processing* 11, no. 11 (November 2002): 1260–70. https://doi.org/10.1109/TIP.2002.804276.

[2] Aja-Fernandez, S., and C. Alberola-Lopez. “On the Estimation of the Coefficient of Variation for Anisotropic Diffusion Speckle Filtering.” *IEEE Transactions on Image Processing* 15, no. 9 (September 2006): 2694–2701. https://doi.org/10.1109/TIP.2006.877360.

See Also

`imdiffusefilt`

Topics

“Read, Process, and View Ultrasound Data”

jitterIntensity

Randomly augment intensity of grayscale image or intensity volume

Syntax

```
J = jitterIntensity(I,Name=Value)
```

Description

You can use data augmentation to increase the variety and quantity of training data in deep learning applications, especially when available training data is limited, as is typical in medical imaging. Data augmentation can be intensity augmentation, geometric augmentation, or color augmentation. The `jitterIntensity` function performs intensity augmentation of grayscale images and intensity volumes by randomly augmenting their brightness, contrast, and gamma correction.

`J = jitterIntensity(I,Name=Value)` jitters the intensity of grayscale image or intensity volume `I` by randomly selecting brightness (shifting of intensity), contrast (scaling of intensity), and gamma correction values. To specify ranges for these values, use the corresponding name-value arguments.

Examples

Jitter Intensity of 2-D Computed Tomography (CT) Image

Import a grayscale CT image into the workspace. Crop the image to retain only the object of interest.

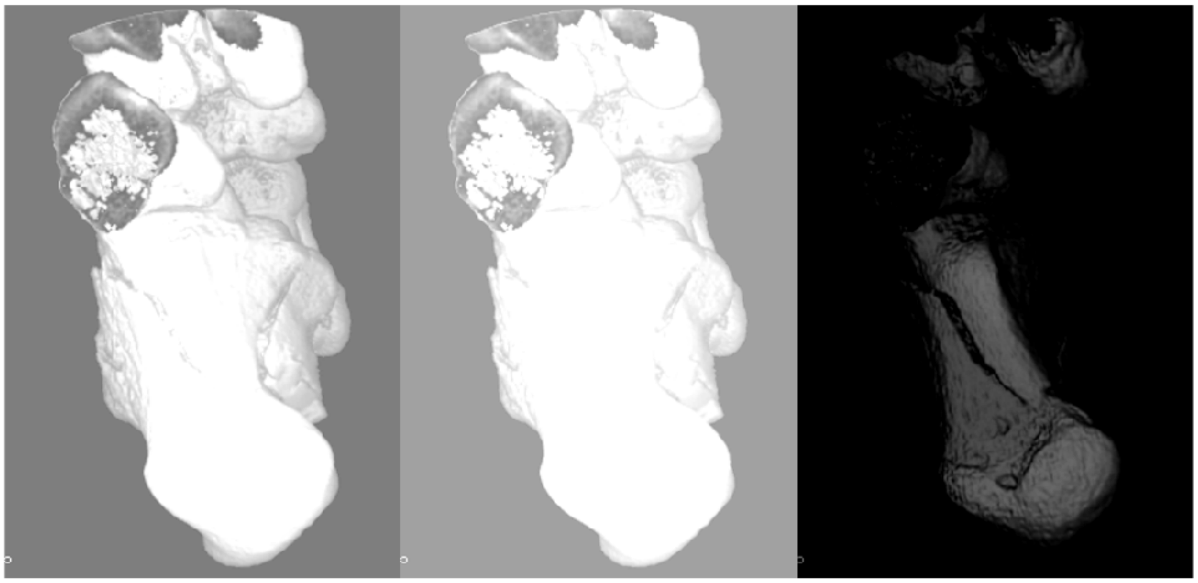
```
I = dicomread("CT-MONO2-16-ankle.dcm");
I = imcrop(I,[101 51 290 420]);
```

Randomly shift the intensity of the image `I` multiple times.

```
J1 = jitterIntensity(im2single(I),Brightness=0.75);
J2 = jitterIntensity(im2single(I),Brightness=0.75);
J3 = jitterIntensity(im2single(I),Brightness=0.75);
```

Visualize the augmented CT images to observe the impact of the randomly selected brightness value.

```
figure
montage({J1 J2 J3})
```



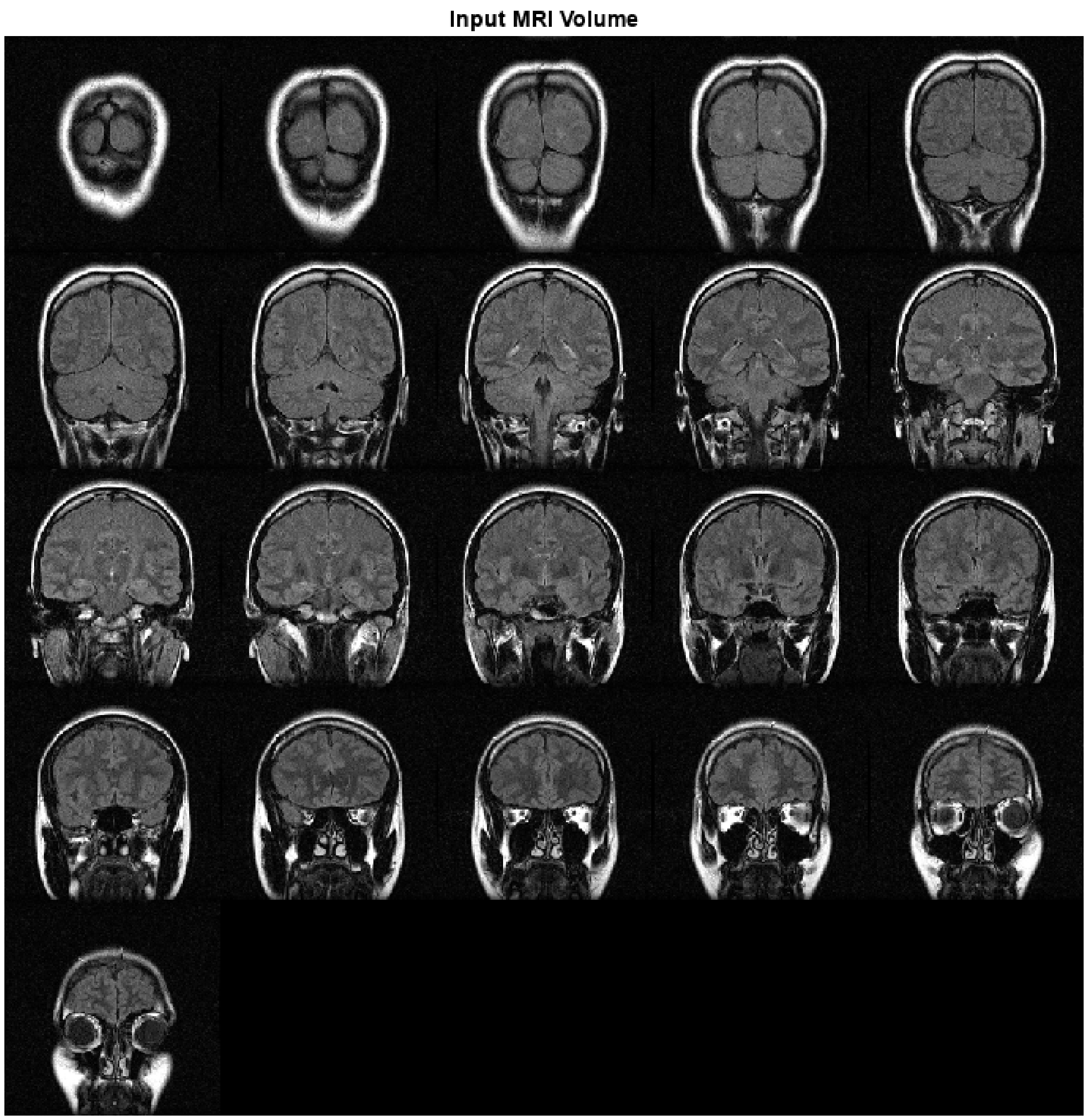
Jitter Intensity of 3-D Magnetic Resonance Imaging (MRI) Volume

Load an MRI intensity volume into the workspace.

```
load("mristack.mat","mristack");
V = mristack;
```

Visualize the MRI volume.

```
figure
montage(V)
title("Input MRI Volume")
```



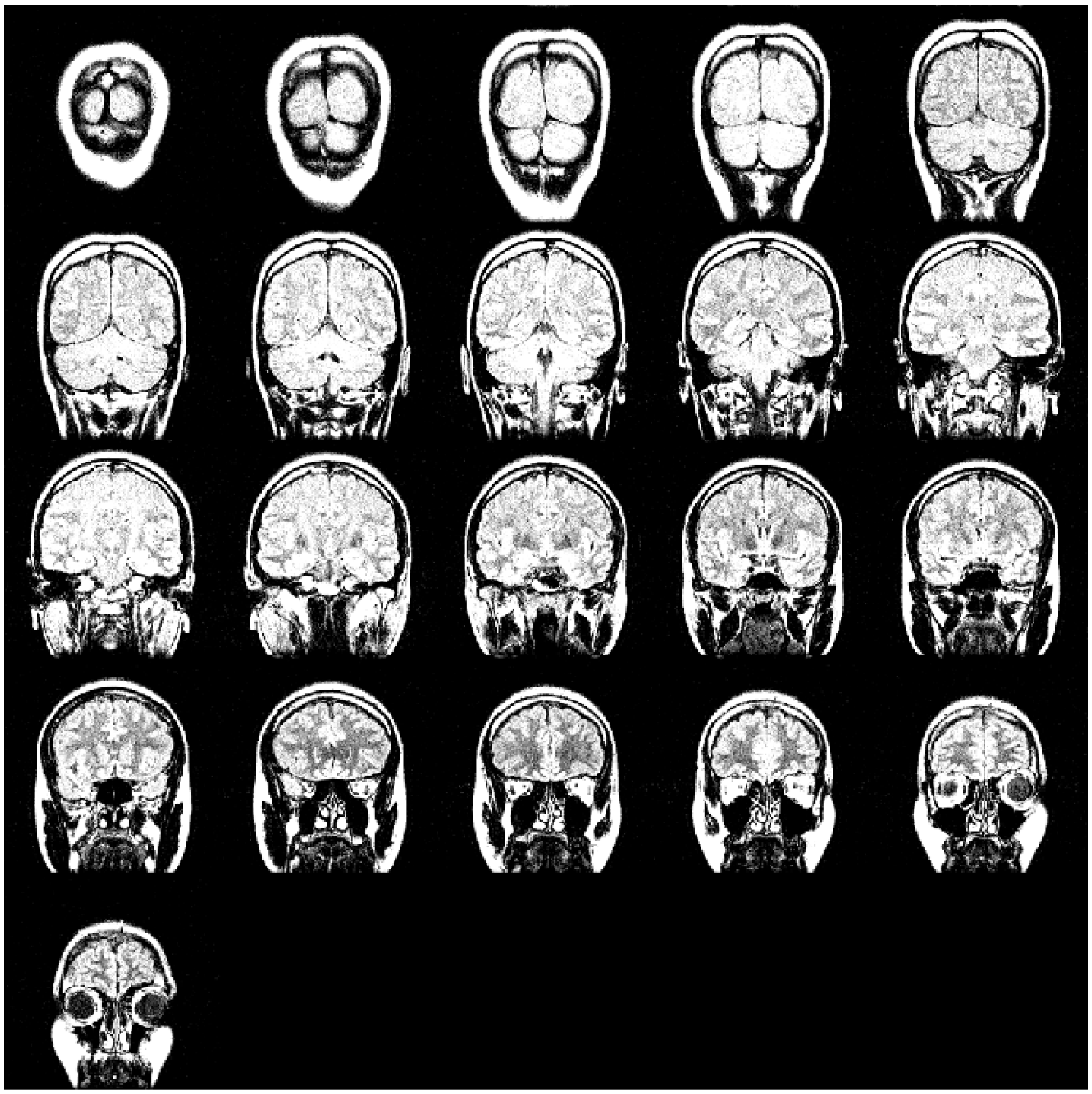
Randomly shift and scale the intensity of the volume V.

```
jitterV = jitterIntensity(V,Brightness=[-0.1 0.2],Contrast=3);
```

Visualize the augmented MRI volume.

```
figure
montage(jitterV)
title("Augmented MRI Volume")
```

Augmented MRI Volume



Augment Datastore for Deep Learning

Specify the location of a directory containing DICOM image files. Create a datastore for deep learning from the DICOM files.

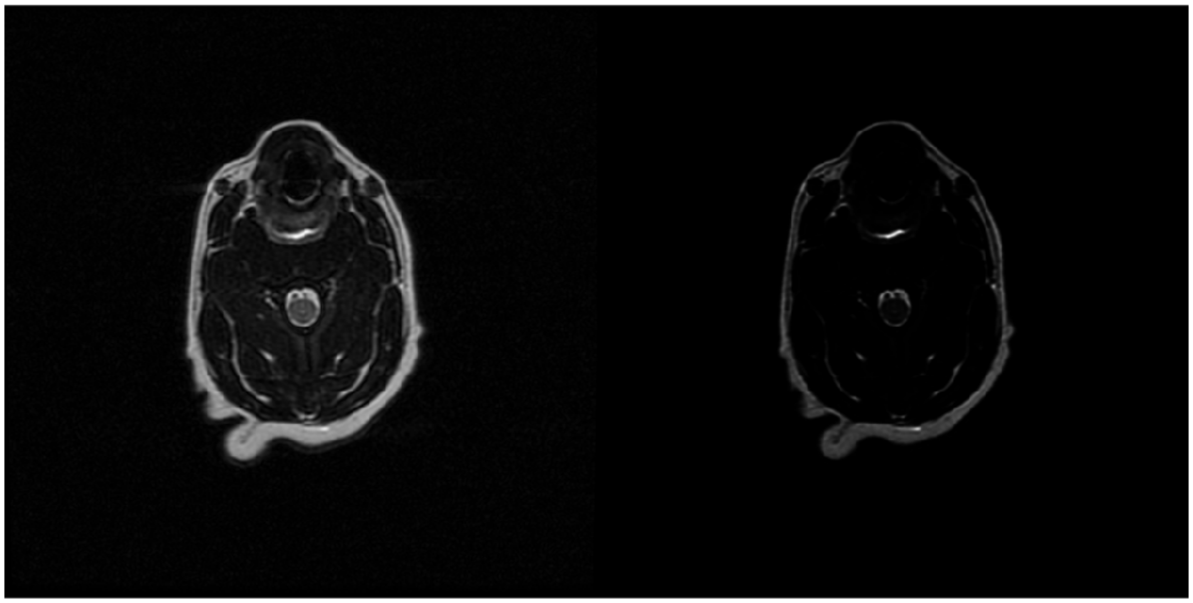
```
dicomDir = fullfile(matlabroot,"toolbox/images/imdata/dog");
dicomds = imageDatastore(dicomDir,FileExtensions=".dcm",ReadFcn=@(x)dicomread(x));
```

Transform the datastore by gamma-correcting the intensity of the images with a random gamma value.

```
jitterds = transform(dicomds,@(x)jitterIntensity(im2single(x),Gamma=[2 3]));
```

Visualize the original and augmented datastore images.

```
dicomImage = read(dicomds);
jitterImage = read(jitterds);
figure
imshowpair(dicomImage,jitterImage,"montage")
```



Input Arguments

I — Grayscale image or intensity volume
2-D numeric matrix | 3-D numeric array

Grayscale image or intensity volume, specified as a 2-D numeric matrix or 3-D numeric array, respectively.

The function does not support 3-D RGB images and 4-D RGB volumes. For color augmentation of 3-D RGB images, see `jitterColorHSV`.

Data Types: `single` | `double` | `int16` | `uint8` | `uint16`

Name-Value Arguments

Specify optional pairs of arguments as `Name1=Value1,...,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example: `J = jitterIntensity(I,Brightness=0.75,Contrast=3,Gamma=[2 3])` augments the intensity of `I` by randomly selecting parameters from the ranges specified in the name-value arguments, and returns the augmented image `J`.

`Brightness` — Brightness range
`0` (default) | scalar in the range [0,1] | two-element vector in the range [–1, 1]

Brightness range, specified as a scalar in the range [0,1] or a two-element vector with elements in the range [–1, 1]. If you specify this value as a scalar, *b*, `jitterIntensity` shifts the intensity of the image or volume by a randomly selected value from the range [–*b*, *b*]. If you specify this value as a vector, [b_1 b_2], `jitterIntensity` shifts the intensity of the image or volume by a randomly selected value from the range [b_1, b_2]. You must specify values for b_1 and b_2 such that $b_2 \geq b_1$.

Example: `Brightness=0.75` shifts the intensity by a randomly selected value from the range [–0.75, 0.75].

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`Contrast` — Contrast range
`0` (default) | nonnegative scalar | two-element nonnegative vector

Contrast range, specified as a nonnegative scalar or two-element nonnegative vector. If you specify this value as a scalar, *c*, `jitterIntensity` scales the intensity of the image or volume by a randomly selected value from the range [*min*, 1+*c*], where *min* is the higher of 1–*c* and 0. If you specify the value as a vector, [c_1 c_2], `jitterIntensity` scales the intensity of the image or volume by a randomly selected value from the range [c_1, c_2].

Example: `Contrast=3` scales the intensity by a randomly selected value from the range [0, 4].

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`Gamma` — Gamma range
`0` (default) | nonnegative scalar | two-element nonnegative vector

Gamma range, specified as a nonnegative scalar or two-element nonnegative vector. If you specify the value as a scalar, *g*, `jitterIntensity` gamma-corrects the intensity of the image or volume with a randomly selected gamma value from the range [*min*, 1+*g*], where *min* is the higher of 1-*g* and 0. If you specify the value as a vector, [g_1 g_2], `jitterIntensity` gamma-corrects the intensity of the image or volume with a randomly selected gamma value from the range [g_1, g_2].

Example: `Gamma=[2 3]` gamma-corrects the intensity with a randomly selected gamma value from the range [2, 3].

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Output Arguments

`J` — Jittered image or volume
2-D numeric matrix | 3-D numeric array

Jittered image or volume, returned as a numeric matrix or array of the same size and data type as the input image or volume `I`. If the input image or volume `I` is of the data type single or double, the function rescales the intensities in `J` to the range [0, 1].

Version History

Introduced in R2022b

See Also

`jitterColorHSV`

nrrdinfo

Read metadata from NRRD file

Syntax

```
info = nrrdinfo(filename)
```

Description

`info = nrrdinfo(filename)` reads the metadata from the nearly raw raster data (NRRD) image file `filename` and returns the metadata structure `info`.

Examples

Read Metadata from NRRD File

Read the metadata from an NRRD format file. The file is part of a data set containing the 3-D CT and MRI scans from The Retrospective Image Registration Evaluation (RIRE) Dataset, converted to the NRRD file format. The original data set was provided by Dr. Michael Fitzpatrick. For more information, see the RIRE Project homepage. The size of the entire data set is approximately 35 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalRegistrationNRRDdata.zi
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the name of the NRRD file to read.

```
filename = fullfile(filepath,"supportfilesNRRD","Patient007CT.nrrd");
```

Read the metadata from `filename`.

```
info = nrrdinfo(filename);
```

The `ImageSize` metadata field contains the image volume size, in pixels.

```
info.ImageSize
```

```
ans = 1×3

   512   512    28
```

The `PixelDimensions` metadata field contains the voxel size in each dimension, in real world units such as millimeters.

```
info.PixelDimensions
```

```
ans = 1×3

    0.6536    0.6536    4.0000
```

The `SpatialMapping` metadata field contains an `affinetform3d` object that defines the transformation between the intrinsic and patient coordinate systems for the volume.

```
info.SpatialMapping
```

```
ans = 
  affinetform3d with properties:

    Dimensionality: 3
                 A: [4×4 double]
```

The `RawAttributes` metadata field contains the raw metadata extracted from the NRRD header information.

```
info.RawAttributes
```

```
ans = struct with fields:
          dimension: '3'
              sizes: '512 512 28 '
               type: 'float'
           encoding: 'raw'
             endian: 'little'
     spacedimension: '3'
        spaceorigin: '(1.653595, 1.653595, 5.000000)'
    spacedirections: '(0.653595,0.000000,0.000000) (0.000000,0.653595,0.000000) (0.000000,0.00000
```

Input Arguments

`filename` — Name of NRRD file
string scalar | character vector

Name of the NRRD file, specified as a string scalar or a character vector. The file must contain a valid NRRD file header. The `nrrdinfo` function supports NRRD files containing header and image data (`.nrrd`) as well as detached header files (`.nhdr`). Specify `filename`as the absolute path to the file, a relative path from the current directory, or a relative path from a directory on the MATLAB path.

Data Types: `char` | `string`

Output Arguments

`info` — NRRD metadata
structure

NRRD metadata, returned as a structure.

Version History

Introduced in R2022b

See Also

`nrrdread` | `isnrrd`

nrrdread

Read NRRD image

Syntax

```
V = nrrdread(filename)
```

Description

`V = nrrdread(filename)` reads the nearly raw raster data (NRRD) image file specified by `filename`, and returns the volumetric image data `V`.

Examples

Read Image Data from NRRD File

Read the image data from an NRRD format file. The file is part of a data set containing the 3-D CT and MRI scans from The Retrospective Image Registration Evaluation (RIRE) Dataset, converted to the NRRD file format. The original data set was provided by Dr. Michael Fitzpatrick. For more information, see the RIRE Project homepage. The size of the entire data set is approximately 35 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalRegistrationNRRDdata.zi
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the name of the NRRD file to read.

```
filename = fullfile(filepath,"supportfilesNRRD","Patient007CT.nrrd");
```

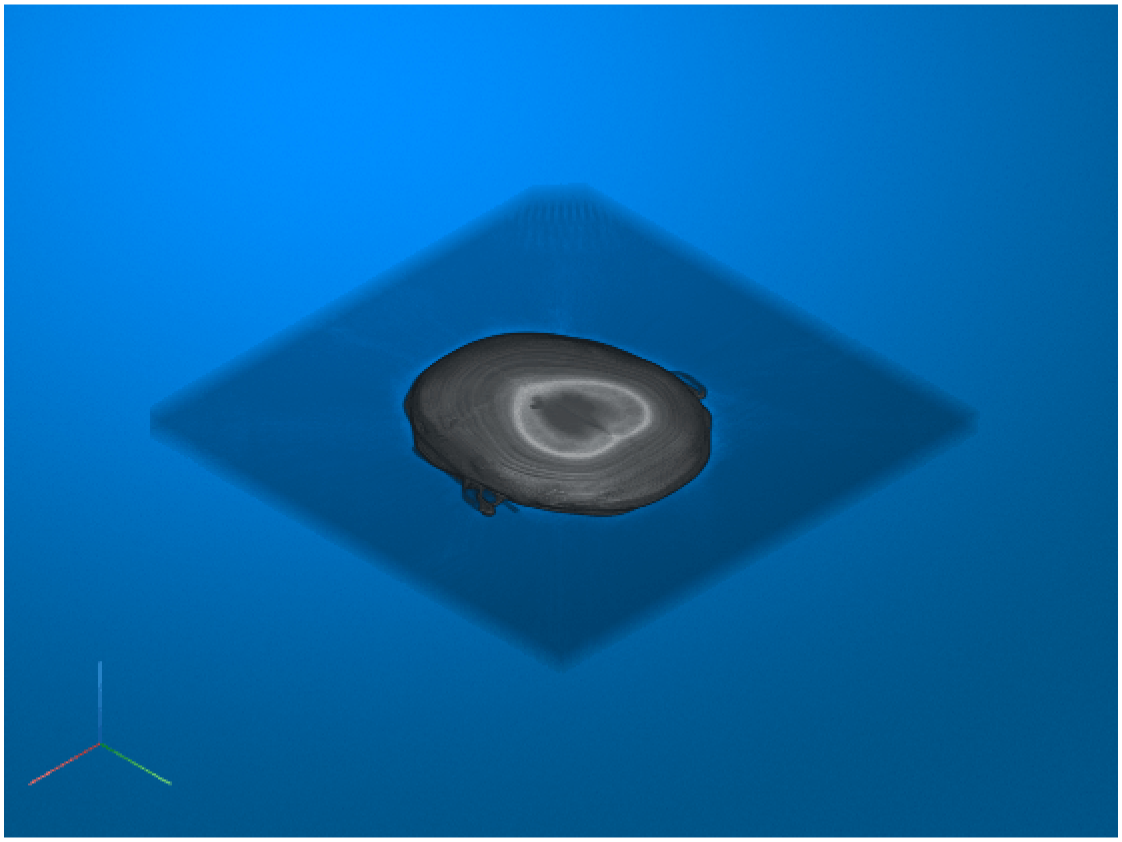
Read the image data from `filename`. The image data, `V`, is a 3-D array of intensity values.

```
V = nrrdread(filename);
whos V
```

```
  Name        Size                    Bytes  Class     Attributes

  V         512x512x28             29360128  single
```

Display the image data stored in the array `V`.

```
volshow(V);
```



Input Arguments

`filename` — Name of NRRD file
string scalar | character vector

Name of the NRRD file, specified as a string scalar or a character vector. The file must be a valid NRRD file (`.nrrd`) that contains header information and image data. Specify `filename` as the absolute path to the file, a relative path from the current directory, or a relative path from a directory on the MATLAB path.

Data Types: `char` | `string`

Output Arguments

`V` — Volumetric image data
numeric array

Volumetric image data, returned as a numeric array.

Version History

Introduced in R2022b

See Also

`nrrdinfo` | `isnrrd`

changeFilePaths

Change file paths in ground truth data for medical images

Syntax

```
unresolvedFilePaths = changeFilePaths(gTruthMed,alternateFilePaths)
unresolvedFilePaths = changeFilePaths(gTruthMed,
alternateFilePaths,ChangeProperty=propertyName)
```

Description

`unresolvedFilePaths = changeFilePaths(gTruthMed,alternateFilePaths)` changes the file paths stored in the `DataSource` and `LabelData` properties of the `groundTruthMedical` object `gTruthMed`. The `alternateFilePaths` argument specifies pairs of current paths in `gTruthMed` and alternative paths that point to the new data location on your machine. The function returns any unresolved paths in `unresolvedPaths`. An unresolved path is a current path not found in `gTruthMed` or an alternative path not found on your machine. In both cases, `unresolvedPaths` returns only the corresponding current path.

Use this function to update a `groundTruthMedical` object if you move the data source or label image files to a new folder. If you receive a `groundTruthMedical` object that was created on a different computer, use this function to point to the data location on your local machine.

`unresolvedFilePaths = changeFilePaths(gTruthMed,alternateFilePaths,ChangeProperty=propertyName)` replaces only the file paths stored in the specified property, `DataSource` or `LabelData`, of `gTruthMed`.

Examples

Change File Paths in Medical Ground Truth Data

Change the file paths in a medical ground truth data object to point to a subset of the Medical Segmentation Decathlon data set [1 on page 1-61]. The subset of data includes two CT chest volumes and corresponding label images, stored in the NIfTI file format. Download the `MedicalVolumNIfTIData.zip` file from the MathWorks® website, then unzip the file. The size of the data file is approximately 76 MB.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeNIfTIData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
dataFolder = fullfile(filepath,"MedicalVolumeNIfTIData");
```

Load a `groundTruthMedical` object containing ground truth data into the workspace. The data source and label data of the object specify file paths to the data at a different location than `dataFolder`. MATLAB® displays a warning that the path to the data source cannot be found. This can occur when a `groundTruthMedical` object is created on one computer and loaded on a different computer that stores the image data in a different location.

```
load("gTruthMed-NIfTI-Chest.MAT")
```

```
Warning: Unable to find one or more 'DataSource' or 'LabelData' files. Use the changeFilePaths ob
```

Display the current path to the data source and label data.

```
gTruthMed.DataSource.Source
```

```
ans=2×1 cell array
    {["C:\CFP\lung_027.nii.gz"]}
    {["C:\CFP\lung_043.nii.gz"]}
```

```
gTruthMed.LabelData
```

```
ans = 2×1 string
    "C:\CFP\LabelData\lung_027.nii.gz"
    "C:\CFP\LabelData\lung_043.nii.gz"
```

Specify the current path to the data source and an alternative path and store these paths in a string array `alternativePaths`.

```
currentPathDataSource = "C:\CFP";
newPathDataSource = dataFolder;
alternativePaths = [currentPathDataSource newPathDataSource];
```

Use the `changeFilePaths` object function to update the data source file paths, based on the paths in the string array. Because the function resolves all paths, it returns an empty array of unresolved paths

```
unresolvedPaths = changeFilePaths(gTruthMed,alternativePaths)
```

```
unresolvedPaths = 

  0×1 empty string array
```

Verify that the new data source and label data paths are stored in the `DataSource` and `LabelData` properties of `gTruthMed`.

```
gTruthMed.DataSource;
gTruthMed.LabelData;
```

References

[1] Medical Segmentation Decathlon. "Lung." Tasks. Accessed May 10, 2018. http://medicaldecathlon.com/.

The Medical Segmentation Decathlon data set is provided under the CC-BY-SA 4.0 license. All warranties and representations are disclaimed. See the license for details.

Input Arguments

`gTruthMed` — Ground truth data
`groundTruthMedical` object

Ground truth data, specified as a `groundTruthMedical` object. You can export a `groundTruthMedical` object from the **Medical Image Labeler** app or create one programmatically by using the `groundTruthMedical` function.

`alternateFilePaths` — Alternative file paths
n-by-2 string array

Alternative file paths, specified as an *n*-by-2 string array, where *n* is the number of paths to update. Each row specifies a current location and its corresponding new location in the format [p_{current} p_{new}].

- p_{current} is a current file path in `gTruthMed`. Specify p_{current} using the same file separators, either forward slashes or backslashes, present in `gTruthMed`.
- p_{new} is the alternative path replacing p_{current}. Specify p_{new} using either forward slashes or backslashes as the path separators.

The function updates the properties of `gTruthMed` based on the types of files specified:

- Data sources — The `DataSource` property of `gTruthMed` contains a `VolumeSource` or `ImageSource` object. The `changeFilePaths` function updates the paths in the `Source` property of these objects.
- Label data — The `changeFilePaths` function updates the paths in the `LabelData` property of `gTruthMed`.

You can specify paths as full path names to specific files, or as the beginning portion of a file path. If you specify a pair of partial paths, [$p_{\text{currentPartial}}$ $p_{\text{newPartial}}$], the function determines the full current and alternate paths using these steps:

1. The function searches `gTruthMed` for file names that start with $p_{\text{currentPartial}}$. The final list of current paths consists of all full file paths starting with $p_{\text{currentPartial}}$.
2. The final list of alternative paths is the list of current full file paths with the portion $p_{\text{currentPartial}}$ replaced with $p_{\text{newPartial}}$.

Example: `["C:\VolumeData","C:\VolumeData2"]` changes the path of the data directory. The function updates the path for all files in `gTruthMed` in the folder `C:\VolumeData` and its subfolders to begin with `C:\VolumeData2`.

Example: `["C:\VolumeData","B:\VolumeData"]` changes the drive letter in the file paths in `gTruthMed` that begin with `C:\VolumeData` from `C` to `B`.

Data Types: `string`

`propertyName` — Property for which to change file paths
`"Auto"` (default) | `"DataSource"` | `"LabelData"`

Property for which to change file paths, specified as one of these values:

- `"Auto"` — Update file paths stored in both the `DataSource` and `LabelData` properties.
- `"DataSource"` — Update file paths stored in the `DataSource` property.
- `"LabelData"` — Update file paths stored in the `LabelData` property.

Data Types: `string`

Output Arguments

`unresolvedFilePaths` — Unresolved file paths
string array

Unresolved file paths, returned as a string array. If the function cannot find either the specified current path in `gTruth` or the specified alternative path location, then it returns the corresponding current path.

If the function finds and resolves all file paths, then it returns `unresolvedFilePaths` as an empty string array.

Version History

Introduced in R2022b

See Also

`groundTruthMedical`

merge

Merge two or more `groundTruthMedical` objects

Syntax

```
gTruthMerged = merge(gTruth1,gTruth2,...,gTruthn)
```

Description

`gTruthMerged = merge(gTruth1,gTruth2,...,gTruthn)` merges two or more `groundTruthMedical` objects into a new `groundTruthMedical` object, `gTruthMerged`.

Examples

Merge Medical Ground Truth Data

Merge two compatible `groundTruthMedical` objects present in the MATLAB workspace.

The ground truth objects, `gTruthMed1` and `gTruthMed2`, specify the volumetric data source and label image for the CT lung volumes, `lung_001.nii.gz` and `lung_003.nii.gz`, respectively.

```
gTruthMed1
```

```
gTruthMed1 = 

  groundTruthMedical with properties:

          DataSource: [1×1 medical.labeler.loading.VolumeSource]
           LabelData: "C:\Merge\Label Data\lung_001.nii.gz"
    LabelDefinitions: [1×3 table]
```

```
gTruthMed2
```

```
gTruthMed2 = 

  groundTruthMedical with properties:

          DataSource: [1×1 medical.labeler.loading.VolumeSource]
           LabelData: "C:\Merge\Label Data\lung_003.nii.gz"
    LabelDefinitions: [1×3 table]
```

Merge the `groundTruthMedical` objects. The merged object specifies both lung CT volumes.

```
gTruthMerged = merge(gTruthMed1,gTruthMed2)
```

```
gTruthMerged = 

  groundTruthMedical with properties:

          DataSource: [1×1 medical.labeler.loading.VolumeSource]
          LabelData: [2×1 string]
    LabelDefinitions: [1×3 table]
```

Input Arguments

gTruth1,gTruth2,...,gTruthn — Ground truth data to merge
groundTruthMedical objects

Ground truth data to merge, specified as a comma-separated list of two or more groundTruthMedical objects. You can export the groundTruthMedical objects from the **Medical Image Labeler** app or create them programmatically by using the groundTruthMedical function.

The specified groundTruthMedical objects must all have the same data source type, specified by the DataSource property. For example, if one object has a data source of type VolumeSource, all groundTruthMedical objects you merge it with must also have a data source of type VolumeSource.

You cannot merge groundTruthMedical objects that have one or more DataSource entries pointing to the same file location. Each groundTruthMedical object must specify a unique set of image sequences or image volumes

If the LabelDefinition properties of two or more objects specify the same label name, they must also specify the same color and pixel label ID for that label name. The label name, color, and pixel label ID are specified by the LabelName, LabelColor, and PixelLabelID columns of the LabelDefinition property.

Output Arguments

gTruthMerged — Merged ground truth data
groundTruthMedical object

Merged ground truth data, returned as a groundTruthMedical object. The DataSource property of gTruthMerged is the same as that of the input groundTruthMedical objects.

Version History

Introduced in R2022b

See Also

groundTruthMedical

extractFrame

Extract pixel data for one frame of 2-D medical image series

Syntax

```
X = extractFrame(medImage,frame)
```

Description

`X = extractFrame(medImage,frame)` extracts the pixel data for the frame at index `frame` of the `medicalImage` object `medImage`.

Examples

Extract Frame from Medical Image Series

Specify the name of an echocardiogram series stored as a DICOM file.

```
fileName = "heartUltrasoundSequence.dcm";
```

Create a medical image object for the echocardiogram series.

```
medImage = medicalImage(fileName);
```

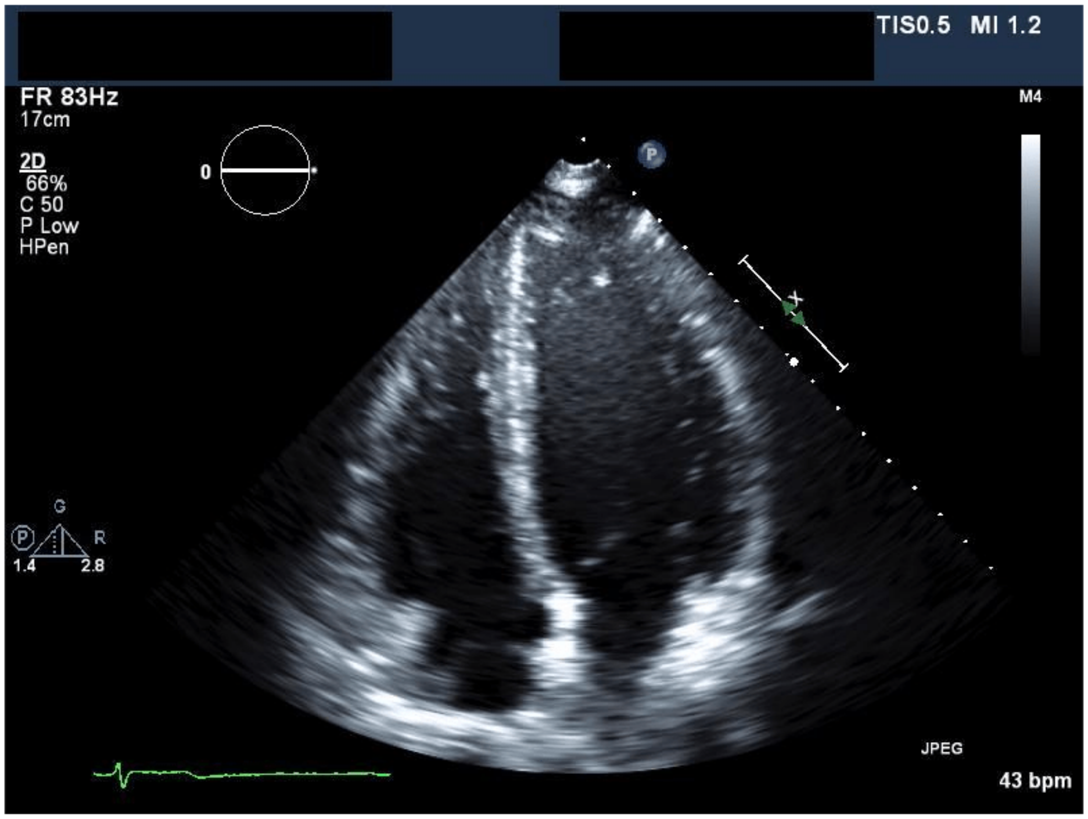
Check the `NumFrames` property value to determine the number of frames in the series. `medImage` contains three frames.

```
numFrames = medImage.NumFrames
```

```
numFrames = 3
```

Extract the second frame in the image series. Display the extracted frame image.

```
X = extractFrame(medImage,2);
imshow(X)
```



Input Arguments

`medImage` — Medical image
`medicalImage` object

Medical image, specified as a `medicalImage` object.

`frame` — Frame index
positive integer scalar in range [1, *numFrames*]

Frame index, specified as a positive integer scalar in the range [1, *numFrames*], where *numFrames* is the number of frames in the image or series of images in `medImage`. The number of frames is specified by the `NumFrames` property of `medImage`.

Data Types: `double`

Output Arguments

`X` — Pixel data of extracted frame
m-by-*n* numeric matrix

Pixel data of the extracted frame, returned as an *m*-by-*n* numeric matrix, where *m* and *n* are the first two dimensions of the image frame extracted from `medImage`. The `X` output matrix is the same data type as the `Pixels` property of `medImage`.

Version History

Introduced in R2022b

See Also

`medicalImage`

Topics

"Read, Process, and View Ultrasound Data"

intrinsicToWorldMapping

Geometric transform between intrinsic and patient coordinates of medical image volume

Syntax

```
tform = intrinsicToWorldMapping(R)
```

Description

`tform = intrinsicToWorldMapping(R)` computes the geometric transformation `tform` between the intrinsic and patient coordinate systems for the medical image volume defined by `R`. If the volume specified by `R` is non-affine, then this function computes the transformation for only the first slice along the third dimension.

Examples

Get Geometric Transform Between Intrinsic and Patient Coordinate Systems

Get the geometric transformation between the intrinsic and patient coordinate systems for a chest CT volume saved as a directory of DICOM files. The volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of the DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData","LungCT01");
```

Create a medical volume object that contains the image and spatial metadata for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

The `VolumeGeometry` property of a medical volume object contains a `medicalref3d` object that specifies the spatial referencing for the volume. Extract the `medicalref3d` object for the chest CT.

```
R = medVol.VolumeGeometry;
```

Calculate the transformation that maps between the intrinsic and patient coordinate systems by using the `intrinsicToWorldMapping` object function. The function returns an `affinetform3d` object, `tform`.

```
tform = intrinsicToWorldMapping(R)
```

```
tform = 
  affinetform3d with properties:

    Dimensionality: 3
```

```
                 A: [4×4 double]
```

The A property of the `affinetform3d` object contains a 4-by-4 geometric transformation matrix.

```
tform.A
```

```
ans = 4×4

         0    0.7285         0 -187.2285
    0.7285         0         0 -187.2285
         0         0    2.5000 -283.7500
         0         0         0    1.0000
```

Input Arguments

R — Spatial referencing information
`medicalref3d` object

Spatial referencing information, specified as a `medicalref3d` object.

Output Arguments

`tform` — Geometric transformation
`affinetform3d` object

Geometric transformation, returned as an `affinetform3d` object. If the volume specified by `R` is affine, then `tform` describes the transformation between the intrinsic and patient coordinate systems for the entire volume. If the volume is non-affine, then `tform` describes the transformation between the intrinsic and patient coordinate systems for the points in the first slice of the volume along the third dimension.

An image volume is affine if these conditions are met:

- All slices are parallel to each other.
- The spacing between slices in each dimension is uniform.
- The upper-left voxels of all slices are collinear.
- No two slices are coincident, meaning no two slices are located at the same position in space.

The A property of `tform` contains a 4-by-4 3-D transformation matrix that maps triplets of voxel indices in the order (*column*, *row*, *slice*) to (*x*, *y*, *z*) triplets of patient coordinates in real-world units.

Version History

Introduced in R2022b

See Also

`medicalref3d` | `oneSliceIntrinsicToWorldMapping`

Topics

"Display 3-D Medical Image Data in Patient Coordinate System"

“Display Labeled Medical Image Volume in Patient Coordinates”

contains

Determine if affine image volume contains points specified in patient coordinate system

Syntax

```
tf = contains(R,xyzWorld)
```

Description

`tf = contains(R,xyzWorld)` returns a logical vector, `tf`, that indicates whether each of the specified 3-D coordinates points `xyzWorld` falls within the bounds of the affine medical image volume defined by `R`.

Examples

Determine If Medical Image Volume Contains Patient Coordinates

Determine if a chest CT volume, saved as a directory of DICOM files, contains a set of coordinates specified in the patient coordinate system. The volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath);
```

Specify the directory of the DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData","LungCT01");
```

Create a medical volume object that contains the image and spatial metadata for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

The `VolumeGeometry` property of the medical volume object contains a `medicalref3d` object that specifies the spatial referencing for the volume. Extract the `medicalref3d` object for the chest CT.

```
R = medVol.VolumeGeometry;
```

Specify the patient coordinates, in millimeters, of three sample points.

```
xyzWorld = [-100 -100 -200; 0 0 -100; 300 200 -80]
```

```
xyzWorld = 3×3

  -100  -100  -200
     0     0  -100
   300   200   -80
```

Check whether the sample points are inside the image boundary. The values of `tf` indicate that the third point lies outside the image.

```
tf = contains(R,xyzWorld)
```

```
tf = 3×1 logical array

   1
   1
   0
```

Input Arguments

`R` — Spatial referencing information
`medicalref3d` object

Spatial referencing information, specified as a `medicalref3d` object. `R` must specify an affine image volume. An image volume is affine if these conditions are met:

- All slices are parallel to each other.
- The spacing between slices in each dimension is uniform.
- The upper-left voxels of all slices are collinear.
- No two slices are coincident, meaning no two slices are located at the same position in space.

`xyzWorld` — Patient coordinates of points to query
n-by-3 numeric matrix

Patient coordinates of points to query, specified as an *n*-by-3 numeric matrix, where *n* is the number of points. The patient coordinates are in real-world units defined by the patient coordinate system.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Output Arguments

`tf` — Image volume contains specified points
n-element logical vector

Image volume contains the specified points, returned as an *n*-element logical vector, where *n* is the number of points. A value of `1` (`true`) indicates that the corresponding `xyzWorld` point exists in the image volume, and a value of `0` (`false`) indicates that it does not.

Version History

Introduced in R2022b

See Also

`medicalref3d`

intrinsicToWorld

Map points from intrinsic coordinates to patient coordinates

Syntax

```
[X,Y,Z] = intrinsicToWorld(R,I,J,K)
```

Description

`[X,Y,Z] = intrinsicToWorld(R,I,J,K)` maps points from the intrinsic coordinate system to the patient coordinate system using the spatial referencing information, `R`. The intrinsic coordinates `I`, `J`, and `K` are defined by axes aligned with the row, column, and slice subscripts of the image data array, respectively. The patient coordinates `X`, `Y`, and `Z` are defined by the patient coordinate system axes.

Examples

Map 3-D Intrinsic Coordinates to Patient Coordinates

Map 3-D intrinsic coordinates from a chest CT volume, saved as a directory of DICOM files, to patient coordinates. The volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of the DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData","LungCT01");
```

Create a medical volume object that contains the image and spatial metadata for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

The `VolumeGeometry` property of the medical volume object contains a `medicalref3d` object that specifies the spatial referencing for the volume. Extract the `medicalref3d` object for the chest CT.

```
R = medVol.VolumeGeometry;
```

Select three sample points, and store their (i, j, k) intrinsic coordinates, in voxels. The points must fall within the image boundary.

```
I = [54 200 512];
J = [46 48 79];
K = [1 13 88];
```

Convert the intrinsic coordinates to patient coordinates. The output vectors provide the (x, y, z) patient coordinates, in millimeters.

```
[X,Y,Z] = intrinsicToWorld(R,I,J,K)
```

```
X = 1×3

 -147.8887  -41.5253  185.7717

Y = 1×3

 -153.7168 -152.2597 -129.6758

Z = 1×3

 -281.2500 -251.2500  -63.7500
```

Input Arguments

R — Spatial referencing information
`medicalref3d` object

Spatial referencing information, specified as a `medicalref3d` object.

I — Coordinates along *i*-dimension in intrinsic coordinate system
numeric array

Coordinates along the *i*-dimension in the intrinsic coordinate system, specified as a numeric array. The *i*-axis is aligned with the first dimension of the spatial volume specified by `R`.

`I`, `J`, and `K` must be the same size.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

J — Coordinates along *j*-dimension in intrinsic coordinate system
numeric array

Coordinates along the *j*-dimension in the intrinsic coordinate system, specified as a numeric array. The *j*-axis is aligned with the second dimension of the spatial volume specified by `R`.

`I`, `J`, and `K` must be the same size.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

K — Coordinates along *k*-dimension in intrinsic coordinate system
numeric array

Coordinates along the *k*-dimension in the intrinsic coordinate system, specified as a numeric array. The *k*-axis is aligned with the third dimension of the spatial volume specified by `R`.

`I`, `J`, and `K` must be the same size.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Output Arguments

X — Coordinates along *x*-dimension in patient coordinate system
numeric array

Coordinates along the *x*-dimension in the patient coordinate system, returned as a numeric array. X is the same size as `I`.

Data Types: `double`

Y — Coordinates along *y*-dimension in patient coordinate system
numeric array

Coordinates along the *y*-dimension in the patient coordinate system, returned as a numeric array. Y is the same size as `I`.

Data Types: `double`

Z — Coordinates along *z*-dimension in patient coordinate system
numeric array

Coordinates along the *z*-dimension in the patient coordinate system, returned as a numeric array. Z is the same size as `I`.

Data Types: `double`

Version History

Introduced in R2022b

See Also

`medicalref3d` | `worldToIntrinsic` | `worldToSubscript`

Topics

"Create STL Surface Model of Femur Bone for 3-D Printing"

oneSliceIntrinsicToWorldMapping

Geometric transform between intrinsic and patient coordinates of medical image volume slice

Syntax

```
tform = oneSliceIntrinsicToWorldMapping(R,slice)
```

Description

`tform = oneSliceIntrinsicToWorldMapping(R,slice)` computes the geometric transformation, `tform`, between the intrinsic and patient coordinate systems for one slice of the medical image volume defined by `R`. The output `tform` maps the geometric transformation for the specified slice `slice` along the third dimension.

Examples

Get Mapping Between Intrinsic and Patient Coordinate Systems for One Slice

Get the geometric transform between the intrinsic and patient coordinate systems for one slice of a chest CT volume, saved as a directory of DICOM files. The volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of the DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData","LungCT01");
```

Create a medical volume object that contains the image and spatial metadata for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

The `VolumeGeometry` property of the medical volume object contains a `medicalref3d` object that specifies the spatial referencing for the volume. Extract the `medicalref3d` object for the chest CT.

```
R = medVol.VolumeGeometry;
```

Calculate the transformation that maps between the intrinsic and patient coordinate systems for the 20th slice along the third dimension of the volume. The `oneSliceIntrinsicToWorldMapping` function returns an `affinetform3d` object, `tform`.

```
tform = oneSliceIntrinsicToWorldMapping(R,20)
```

```
tform = 
  affinetform3d with properties:

    Dimensionality: 3
```

```
                    A: [4×4 double]
```

The A property of the `affinetform3d` object contains a 4-by-4 geometric transformation matrix.

```
tform.A
```

```
ans = 4×4

         0    0.7285         0 -186.5000
    0.7285         0         0 -186.5000
         0         0    0.5307 -233.7500
         0         0         0    1.0000
```

Input Arguments

R — Spatial referencing information
`medicalref3d` object

Spatial referencing information, specified as a `medicalref3d` object. The volume specified by R can be affine or non-affine.

`slice` — Slice index
positive integer in range [1, *p*]

Slice index, specified as a positive integer in the range [1, *p*], where *p* is the number of slices in the image volume along the third dimension.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Output Arguments

`tform` — Geometric transformation
`affinetform3d` object

Geometric transformation, returned as an `affinetform3d` object.

The A property of `tform` contains a 4-by-4 3-D transformation matrix that maps triplets of pixel indices in the order (*column*, *row*, *slice*) to (*x*, *y*, *z*) triplets of patient coordinates in real-world units.

Version History

Introduced in R2022b

See Also

`medicalref3d` | `intrinsicToWorldMapping`

orient

Update patient coordinate system convention

Syntax

```
orientR = orient(R,targetSpace)
```

Description

`orientR = orient(R,targetSpace)` updates the patient coordinate system convention of the spatial referencing information `R` to the specified convention `targetSpace` and returns an equivalent spatial referencing information object, `orientR`, that uses the specified patient coordinate system convention. Use this function to set or update the patient coordinate system orientation, such as from LPS+ to RAS+.

Examples

Update Orientation of Medical Spatial Referencing Object

Update the orientation of the medical spatial referencing object for a chest CT volume saved as a directory of DICOM files. The volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of the DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData","LungCT01");
```

Create a medical volume object that contains the image and spatial metadata for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

The `VolumeGeometry` property of the medical volume object contains a `medicalref3d` object that specifies the spatial referencing for the volume. Extract the `medicalref3d` object for the chest CT.

```
R = medVol.VolumeGeometry
```

```
R = 
  medicalref3d with properties:

                 VolumeSize: [512 512 88]
                   Position: [88×3 double]
             VoxelDistances: {[88×3 double]  [88×3 double]  [88×3 double]}
    PatientCoordinateSystem: "LPS+"
               PixelSpacing: [88×2 double]
                   IsAffine: 1
              IsAxesAligned: 1
```

```
                     IsMixed: 0
```

The `PatientCoordinateSystem` property of the `medicalref3d` object specifies the patient coordinate system orientation. The initial orientation, based on the DICOM file metadata, is `LPS+`. An `LPS+` orientation indicates that the positive *x*-, *y*-, and *z*-axes of the patient coordinate system point in the left, posterior, and superior directions, respectively.

```
R.PatientCoordinateSystem
```

```
ans = 
"LPS+"
```

Update the patient coordinate system orientation to the `RAS+` orientation. An `RAS+` orientation indicates that the positive *x*-, *y*-, and *z*-axes of the patient coordinate system point in the right, anterior, and superior directions, respectively. The `orient` object function returns a new `medicalref3d` object, `orientR`. In addition to the `PatientCoordinateSystem` property, the `Position` and `VoxelDistances` property values reflect the new orientation of `orientR`.

```
orientR = orient(R,"RAS+")
```

```
orientR = 
  medicalref3d with properties:

                 VolumeSize: [512 512 88]
                   Position: [88×3 double]
             VoxelDistances: {[-0.7285 0 0]  [0 -0.7285 0]  [0 0 2.5000]}
    PatientCoordinateSystem: "RAS+"
               PixelSpacing: [0.7285 0.7285]
                   IsAffine: 1
              IsAxesAligned: 1
                    IsMixed: 0
```

Input Arguments

R — Spatial referencing information
`medicalref3d` object

Spatial referencing information, specified as a `medicalref3d` object.

`targetSpace` — Target orientation convention of the patient coordinate system
`"LPS+"` | `"LAS+"` | `"RAS+"`

Target orientation convention of the patient coordinate system, specified as `"LPS+"`, `"LAS+"`, or `"RAS+"`. The first three characters indicate the positive direction of the *x*-, *y*-, and *z*-axes of the patient coordinate system, respectively.

- The positive direction of the *x*-axis points left (`"L"`) or right (`"R"`).
- The positive direction of the *y*-axis points anterior (`"A"`) or posterior (`"P"`).
- The positive direction of the *z*-axis points inferior (`"I"`) or superior (`"S"`).
- `"+"` indicates that values increase in the stated direction.

For example, `"LPS+"` specifies a patient coordinate system with the *x*-, *y*-, and *z*-axes positive in the left, posterior, and superior directions, respectively.

Data Types: `char` | `string`

Output Arguments

`orientR` — Spatial referencing information with target patient coordinate system convention
`medicalref3d` object

Spatial referencing information with the target patient coordinate system convention, returned as a `medicalref3d` object. Which properties of this object the `orient` function updates from `R` depends on the `PatientCoordinateSystem` property of `R`.

- If `PatientCoordinateSystem` is `"unknown"`, then `orient` updates only the `PatientCoordinateSystem` of `orientR`.
- If `PatientCoordinateSystem` is not `"unknown"`, then `orient` updates the `VoxelDistances` and `Position` properties of `orientR`, in addition to the `PatientCoordinateSystem` property, to maintain the correct mapping between the intrinsic and patient coordinate system axes.

Version History

Introduced in R2022b

See Also

`medicalref3d`

sliceCorners

Extract patient coordinates of corner voxels for one slice

Syntax

```
xyzCorners = sliceCorners(R,slice)
```

Description

`xyzCorners = sliceCorners(R,slice)` extracts the *xyz*-coordinates of the four corner voxels for one slice of an image volume.

Examples

Extract Corner Coordinates for One Slice of Medical Spatial Referencing Object

Extract the corner coordinates for one slice of the medical spatial referencing object of a chest CT volume, saved as a directory of DICOM files. The volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of the DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData","LungCT01");
```

Create a medical volume object that contains the image and spatial metadata for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

The `VolumeGeometry` property of the medical volume object contains a `medicalref3d` object that specifies the spatial referencing for the volume. Extract the `medicalref3d` object for the chest CT.

```
R = medVol.VolumeGeometry;
```

Extract the *xyz*-coordinates, in millimeters, of the corner voxels for the first slice along the third dimension of the volume.

```
xyzCorners = sliceCorners(R,1)
```

```
xyzCorners = 4×3

 -186.5000 -186.5000 -281.2500
 -186.5000  185.7717 -281.2500
  185.7717  185.7717 -281.2500
  185.7717 -186.5000 -281.2500
```

Input Arguments

`R` — Spatial referencing information
`medicalref3d` object

Spatial referencing information, specified as a `medicalref3d` object.

`slice` — Slice index
positive integer scalar in range [1, *p*]

Slice index, specified as a positive integer scalar in the range [1, *p*], where *p* is the number of slices in the image volume along the third dimension.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Output Arguments

`xyzCorners` — Coordinates of four corner voxels in patient coordinate system
4-by-3 matrix

Coordinates of the four corner voxels in the patient coordinate system, returned as a 4-by-3 numeric matrix. Each row contains *xyz*-coordinates for one corner of the slice, returned in clockwise order:

1. Pixel in the first row and first column of the data array.
2. Pixel in the first row and last column of the data array.
3. Pixel in the last row and last column of the data array.
4. Pixel in the last row and first column of the data array.

Version History
Introduced in R2022b

See Also
`medicalref3d`

worldToIntrinsic

Map points from patient coordinates to intrinsic coordinates

Syntax

```
[I,J,K] = worldToIntrinsic(R,X,Y,Z)
```

Description

`[I,J,K] = worldToIntrinsic(R,X,Y,Z)` maps points from the patient coordinate system to the intrinsic coordinate system using the spatial referencing information, `R`. The intrinsic coordinates `I`, `J`, and `K` are defined by axes aligned with the row, column, and slice subscripts of the image data array, respectively. The patient coordinates `X`, `Y`, and `Z` are defined by the real-world patient coordinate system axes.

For a point, *n*, if the input coordinates (X_n, Y_n, Z_n) fall outside the image bounds, `worldToIntrinsic` extrapolates I_n, J_n, and K_n outside the image bounds in the intrinsic coordinate system.

Examples

Map 3-D Patient Coordinates to Intrinsic Coordinates

Map 3-D patient coordinates from a chest CT volume, saved as a directory of DICOM files, to intrinsic coordinates. The volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of the DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData","LungCT01");
```

Create a medical volume object that contains the image and spatial metadata for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

The `VolumeGeometry` property of the medical volume object contains a `medicalref3d` object that specifies the spatial referencing for the volume. Extract the `medicalref3d` object for the chest CT.

```
R = medVol.VolumeGeometry;
```

Select three sample points, and store their *(x, y, z)* patient coordinates, in millimeters. For example, the first point has patient coordinates of (100, 101, –200), in mm. The third point is outside the image boundary.

```
X = [100 100 190];
Y = [101 101.2 -190];
Z = [-200 -100 -300];
```

Convert the world coordinates to intrinsic coordinates. The `worldToIntrinsic` function extrapolates the intrinsic coordinates of the third point outside the image boundary. The output vectors provide the (*i*, *j*, *k*) intrinsic coordinates, in voxels. Note that the intrinsic coordinate system is continuous, and the intrinsic coordinates can have noninteger values.

If you receive a warning that an approximate mapping is being used, then the image volume is nearly but not perfectly affine. This might be due to small numeric precision errors in how the data was encoded in the file, or due to the discrete step sizes of motors used to move the patient through the scanner. If an approximate mapping is used, you might expect small errors, on the order of millimeters in patient coordinates.

```
[I,J,K] = worldToIntrinsic(R,X,Y,Z)
```

```
Warning: An approximate world to intrinsic mapping is being used.

I = 1×3

  394.2652  394.2652  517.8040


J = 1×3

  395.6379  395.9124   -3.8043


K = 1×3

  154.0894  342.5072  -34.3283
```

Input Arguments

R — Spatial referencing information
`medicalref3d` object

Spatial referencing information, specified as a `medicalref3d` object.

X — Coordinates along *x*-dimension in patient coordinate system
numeric array

Coordinates along the *x*-dimension in the patient coordinate system, specified as a numeric array.

X, Y, and Z must be the same size.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Y — Coordinates along *y*-dimension in patient coordinate system
numeric array

Coordinates along the *y*-dimension in the patient coordinate system, specified as a numeric array.

X, Y, and Z must be the same size.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Z — Coordinates along *z*-dimension in patient coordinate system
numeric array

Coordinates along the *z*-dimension in the patient coordinate system, specified as a numeric array.

`X`, `Y`, and `Z` must be the same size.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Output Arguments

`I` — Coordinates along *i*-dimension in intrinsic coordinate system
numeric array

Coordinates along the *i*-dimension in the intrinsic coordinate system, returned as a numeric array. The *i*-axis is aligned with the first dimension of the spatial volume specified by `R`. `I` is the same size as `X`.

Data Types: `double`

`J` — Coordinates along *j*-dimension in intrinsic coordinate system
numeric array

Coordinates along the *j*-dimension in the intrinsic coordinate system, returned as a numeric array. The *j*-axis is aligned with the second dimension of the spatial volume specified by `R`. `J` is the same size as `X`.

Data Types: `double`

`K` — Coordinates along *k*-dimension in intrinsic coordinate system
numeric array

Coordinates along the *k*-dimension in the intrinsic coordinate system, returned as a numeric array. The *k*-axis is aligned with third dimension of the spatial volume specified by `R`. `K` is the same size as `X`.

Data Types: `double`

Version History

Introduced in R2022b

See Also

`medicalref3d` | `intrinsicToWorld` | `worldToSubscript`

worldToSubscript

Convert from patient coordinates to row and column subscripts

Syntax

```
[row,col,slice] = worldToSubscript(R,X,Y,Z)
```

Description

`[row,col,slice] = worldToSubscript(R,X,Y,Z)` maps points from the patient coordinate system to the nearest subscript indices `row`, `col`, and `slice`, using the spatial referencing information, `R`.

For a point, *n*, if the input coordinates (X_n, Y_n, Z_n) fall outside the image bounds, `worldToSubscript` extrapolates row_n, col_n, and $slice_n$ outside the image bounds in the intrinsic coordinate system and rounds to the nearest integer values.

Examples

Map 3-D Patient Coordinates to Image Subscripts

Map 3-D patient coordinates from a chest CT volume, saved as a directory of DICOM files, to image subscripts. The volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of the DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData","LungCT01");
```

Create a medical volume object that contains the image and spatial metadata for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

The `VolumeGeometry` property of the medical volume object contains a `medicalref3d` object that specifies the spatial referencing for the volume. Extract the `medicalref3d` object for the chest CT.

```
R = medVol.VolumeGeometry;
```

Select three sample points, and store their (*x*,*y*,*z*) patient coordinates, in millimeters. For example, the first point has patient coordinates of (100, 101, –200), in mm. The third point is outside the image boundary.

```
X = [100 100 190];
Y = [101 101.2 -190];
Z = [-200 -100 -300];
```

Convert the world coordinates to row, column, and slice indices. The `worldToSubscript` function rounds the transformed world coordinates to integer values. The function extrapolates the subscripts of the point outside the image boundary.

If you receive a warning that an approximate mapping is being used, then the image volume is nearly but not perfectly affine. This might be due to small numeric precision errors in how the data was encoded in the file, or due to the discrete step sizes of motors used to move the patient through the scanner. If an approximate mapping is used, you might expect small errors, on the order of millimeters in patient coordinates.

```
[row,col,slice] = worldToSubscript(R,X,Y,Z)
```

```
Warning: An approximate world to intrinsic mapping is being used.

row = 1×3

   394   394   518


col = 1×3

   396   396    -4


slice = 1×3

   154   343   -34
```

Input Arguments

R — Spatial referencing information
`medicalref3d` object

Spatial referencing information, specified as a `medicalref3d` object.

X — Coordinates along *x*-dimension in patient coordinate system
numeric array

Coordinates along the *x*-dimension in the patient coordinate system, specified as a numeric array.

X, Y, and Z must be the same size.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Y — Coordinates along *y*-dimension in patient coordinate system
numeric array

Coordinates along the *y*-dimension in the patient coordinate system, specified as a numeric array.

X, Y, and Z must be the same size.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Z — Coordinates along *z*-dimension in patient coordinate system
numeric array

Coordinates along the *z*-dimension in the patient coordinate system, specified as a numeric array.

`X`, `Y`, and `Z` must be the same size.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Output Arguments

`row` — Row subscript indices
positive integer array

Row subscript indices, returned as a positive integer array. `row` is the same size as `X`.

Data Types: `double`

`col` — Column subscript indices
positive integer array

Column subscript indices, returned as a positive integer array. `col` is the same size as `X`.

Data Types: `double`

`slice` — Slice subscript indices
positive integer array

Slice subscript indices, returned as a positive integer array. `slice` is the same size as `X`.

Data Types: `double`

Version History

Introduced in R2022b

See Also

`medicalref3d` | `worldToIntrinsic` | `intrinsicToWorld`

extractSlice

Extract voxels and spatial details for one slice of medical volume

Syntax

```
[X,position,spacings] = extractSlice(medVol,slice,direction)
```

Description

`[X,position,spacings] = extractSlice(medVol,slice,direction)` extracts the voxel data and spatial information for one slice, `slice`, of the `medicalVolume` object `medVol` along the specified direction, `direction`.

Examples

Extract Slice from Medical Image Volume

Extract a slice from a medical image volume created using a chest CT volume saved as a directory of DICOM files. The CT volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData/LungCT01");
```

Create a medical volume object for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

Extract the third slice in the transverse direction.

```
[X,position,spacings] = extractSlice(medVol,3,"transverse");
```

The `X` output contains the voxel data for the extracted slice.

```
whos X
```

```
  Name         Size                Bytes  Class    Attributes

  X          512x512              524288  int16
```

The `position` output provides the patient coordinates of the first voxel in the slice, in millimeters.

```
position
```

```
position = 1×3
```

```
 -186.5000 -186.5000 -276.2500
```

The `spacings` output provides the in-plane pixel spacing within the extracted slice, in millimeters.

```
spacings
```

```
spacings = 1×2

    0.7285    0.7285
```

Input Arguments

`medVol` — Medical volume
`medicalVolume` object

Medical volume, specified as a `medicalVolume` object.

Note The `extractSlice` function is not valid if the `Orientation` property value of `medVol` is `"mixed"`, `"oblique"`, or `"unknown"`. In these cases, access the voxel data stored in the `Voxels` property directly using array indexing.

`slice` — Slice index
positive integer scalar in range [1, *numSlices*]

Slice index, specified as a positive integer scalar in the range [1, *numSlices*], where *numSlices* is the number of slices in the volume along the direction specified by `direction`.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logical`

`direction` — Direction along which to extract slice
`"coronal"` | `"sagittal"` | `"transverse"`

Direction along which to extract the slice information, specified as `"coronal"`, `"sagittal"`, or `"transverse"`.

Data Types: `char` | `string`

Output Arguments

`X` — Voxel data of extracted slice
m-by-*n* numeric matrix

Voxel data of the extracted slice, returned as an *m*-by-*n* numeric matrix, where *m* and *n* are the number of rows and columns in the image slice in the plane specified by `direction`. The `X` output array is the same data type as the `Voxels` property of `medVol`.

`position` — Position of upper-left pixel of extracted slice
3-element numeric vector

Position of the upper-left pixel of the extracted slice, returned as a 3-element numeric vector of the form [*x y z*]. The coordinates are in the patient coordinate system, in the units specified by the `SpatialUnits` property of `medVol`.

`spacings` — Spacing between voxel centers in first two dimensions of extracted slice
2-element numeric vector

Spacing between voxel centers in the first two dimensions of the extracted slice, returned as a 2-element numeric vector of the form [$dim1_{\text{spacing}}$ $dim2_{\text{spacing}}$]. The values are in the units specified by the `SpatialUnits` property of `medVol`.

Version History

Introduced in R2022b

See Also

`medicalVolume` | `medicalref3d` | `replaceSlice` | `sliceCorners` | `sliceLimits`

Topics
"Read, Process, and Write 3-D Medical Images"

replaceSlice

Replace voxel values for one slice of medical volume

Syntax

```
medVolUpdated = replaceSlice(medVol,slice,direction,sliceValues)
```

Description

`medVolUpdated = replaceSlice(medVol,slice,direction,sliceValues)` replaces the voxel values for one slice, `slice`, of the `medicalVolume` object `medVol` along the specified direction, `direction`. The `replaceSlice` function returns a new `medicalVolume` object with the updated values, specified by `sliceValues`.

Examples

Replace Slice in Medical Image Volume

Replace one slice of a medical image volume created using a chest CT volume saved as a directory of DICOM files. The CT volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData/LungCT01");
```

Create a medical volume object for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

Extract the second slice in the transverse direction. The size of the extracted slice, `X`, is 512-by-512.

```
[X,position,spacings] = extractSlice(medVol,2,"transverse");
whos X
```

```
  Name        Size              Bytes  Class    Attributes

  X         512x512            524288  int16
```

Specify a new 512-by-512 matrix to replace the extracted slice.

```
newX = ones(512);
```

Create a new `medicalVolume` object that replaces the extracted slice with the new slice.

```
medVolUpdated = replaceSlice(medVol,2,"transverse",newX);
```

Input Arguments

`medVol` — Medical volume
`medicalVolume` object

Medical volume, specified as a `medicalVolume` object.

`slice` — Slice index
positive integer scalar in range [1, *numSlices*]

Slice index, specified as a positive integer scalar in the range [1, *numSlices*], where *numSlices* is the number of slices in the volume along the direction specified by `direction`.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logical`

`direction` — Direction along which to update slice
`"coronal"` | `"sagittal"` | `"transverse"`

Direction along which to update the slice information, specified as `"coronal"`, `"sagittal"`, or `"transverse"`.

Data Types: `char` | `string`

`sliceValues` — New voxel values
numeric array

New voxel values, specified as a numeric array. The size of `sliceValues` must be the same size as the original slice in `medVol` specified by `slice`.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Output Arguments

`medVolUpdated` — Medical volume with updated slice
`medicalVolume` object

Medical volume with updated slice, returned as a `medicalVolume` object.

Version History

Introduced in R2022b

See Also

`medicalVolume` | `extractSlice`

resample

Resample medical image volume in different patient coordinate system

Syntax

```
medVolResampled = resample(medVol,R)
medVolResampled = resample(medVol,R,Name=Value)
```

Description

`medVolResampled = resample(medVol,R)` resamples the voxel data stored in the `medicalVolume` object `medVol` in the patient coordinate system specified by `R`. The updated voxel data is returned in a new `medicalVolume` object, `medVolResampled`.

`medVolResampled = resample(medVol,R,Name=Value)` specifies additional options for resampling the voxel data using name-value arguments. For example, `Method="linear"` specifies a linear method for resampling.

Examples

Resample Medical Image Volume to Target Voxel Size

Resample a chest CT volume to a target voxel size, without changing the spatial limits in the patient coordinate system. This can be useful to standardize the voxel spacing in a data set of scans acquired using different settings.

The CT volume is saved as a directory of DICOM files. The volume is part of a data set containing three CT scans. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData/LungCT01");
```

Create a medical volume object, `medVol`, for the CT volume. The voxel size is 0.7285-by-0.7825-by-2.5 mm.

```
medVol = medicalVolume(dataFolder);
medVol.VoxelSpacing
```

```
ans = 1×3

    0.7285    0.7285    2.5000
```

Resample `medVol` so that it has a voxel size of 0.5-by-0.5-by-1.25 mm. First, calculate the ratio between the original voxel size and the target voxel size, in millimeters, in each dimension.

```
targetVoxelSize = [0.5 0.5 1.25];
ratios = targetVoxelSize ./ medVol.VoxelSpacing;
```

Calculate the target dimensions, in voxels, of the resampled volume.

```
origSize = size(medVol.Voxels);
newSize = round(origSize ./ ratios);
```

Define the spatial referencing for the resampled volume. First, get the mapping between the intrinsic and patient coordinate systems of `medVol` by using the `intrinsicToWorldMapping` object function. The output, `origMapping`, is an `affinetform3d` object that contains a transformation matrix in its `A` property.

```
origRef = medVol.VolumeGeometry;
origMapping = intrinsicToWorldMapping(origRef);
tform = origMapping.A;
```

Transform the matrix `tform` so that it corresponds to the target voxel size. Create an `affinetform3d` object, `newMapping`, that contains the transformed matrix.

```
newMapping4by4 = tform.* [ratios([2 1 3]) 1];
newMapping = affinetform3d(newMapping4by4);
```

Create a `medicalref3d` object that describes the target spatial referencing for the resampled volume.

```
newRef = medicalref3d(newSize,newMapping);
```

To maintain the mapping between the patient coordinate axes and the anatomical planes, use the `orient` object function to set the `PatientCoordinateSystem` property of `newRef` to match `origRef`.

```
newRef = orient(newRef,origRef.PatientCoordinateSystem);
```

Resample `medVol` by using the `resample` object function. The voxel size of the new medical volume matches the target size.

```
newVol = resample(medVol,newRef)
```

```
newVol = 
  medicalVolume with properties:

                 Voxels: [746×746×176 int16]
         VolumeGeometry: [1×1 medicalref3d]
           SpatialUnits: "unknown"
            Orientation: "transverse"
           VoxelSpacing: [0.5000 0.5000 1.2500]
           NormalVector: [0 0 1]
       NumCoronalSlices: 746
      NumSagittalSlices: 746
    NumTransverseSlices: 176
           PlaneMapping: ["sagittal"    "coronal"    "transverse"]
               Modality: "unknown"
          WindowCenters: []
           WindowWidths: []
```

Input Arguments

`medVol` — Medical volume
`medicalVolume` object

Medical volume, specified as a `medicalVolume` object.

`R` — Spatial referencing object
`medicalref3d` object

Spatial referencing object, specified as a `medicalref3d` object. `R` defines the spatial details for the resampled volume.

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1,...,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example: `resample(medVol,R,Method="linear")` calculates the voxel values in `medVolResampled` by linearly interpolating voxel values at the corresponding locations in `medVol`.

`FillValue` — Fill value
`0` (default) | numeric scalar

Fill value, specified as a numeric scalar. The value of `FillValue` must be supported by the data type of the voxel data stored in the `Voxels` property of `R`. If an output voxel in the volume defined by `R` falls outside the input volume, `medVol`, `resample` sets the corresponding value of `medVolResampled` to `FillValue`.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

`Method` — Resampling method
`"cubic"` (default) | `"linear` | `"nearest"`

Resampling method, specified as `"cubic"`, `"linear"`, or `"nearest"`.

Data Types: `char` | `string`

Output Arguments

`medVolResampled` — Resampled medical volume
`medicalVolume` object

Resampled medical volume, returned as a `medicalVolume` object. The `VolumeGeometry` property of `medVolResampled` contains the spatial referencing information specified by `R`. The resampled voxel data is stored in the `Voxels` property of `medVolResampled`.

Algorithms

The `resample` function calculates the voxel values in the output volume, `medVolResampled`, by mapping locations in the output volume to the corresponding locations in the input volume (an inverse mapping). When the center of the voxel in the output volume does not map to the center of a voxel in the input volume, `resample` interpolates the input voxel values to calculate the output values.

`resample` performs the mapping in the patient coordinate systems of each volume. The function assumes that the patient coordinate systems of the input and output volumes have the same orientation convention, specified by the `PatientCoordinateSystem` property of a `medicalref3d` object, and that the patient coordinate systems have the same origin.

Version History

Introduced in R2022b

See Also

`medicalVolume` | `medicalref3d`

Topics

"Medical Image Coordinate Systems"

sliceCorners

Extract coordinates of corner voxels for one slice of medical volume

Syntax

```
xyzCorners = sliceCorners(medVol,slice,direction)
```

Description

`xyzCorners = sliceCorners(medVol,slice,direction)` extracts the *xyz*-coordinates of the four corner voxels for one slice, `slice`, in the specified direction `direction` of the `medicalVolume` object `medVol`. The function returns the corner coordinates in the patient coordinate system.

Examples

Extract Corner Coordinates for One Slice of Medical Volume

Extract corner coordinates for a slice of a medical volume created using a chest CT volume saved as a directory of DICOM files. The CT volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData/LungCT01");
```

Create a medical volume object for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

Extract the *xyz*-coordinates, in millimeters, of the corner voxels for the second slice in the coronal direction.

```
xyzCorners = sliceCorners(medVol,2,"coronal")
```

```
xyzCorners = 4×3

 -186.5000 -185.7715 -281.2500
 -186.5000 -185.7715  -63.7500
  185.7717 -185.7715  -63.7500
  185.7717 -185.7715 -281.2500
```

Input Arguments

`medVol` — Medical volume
`medicalVolume` object

Medical volume, specified as a `medicalVolume` object.

`slice` — Slice index
positive integer scalar in range [1, *numSlices*]

Slice index, specified as a positive integer scalar in the range [1, *numSlices*], where *numSlices* is the number of slices in the volume along the direction specified by `direction`.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logical`

`direction` — Direction along which to extract slice limits
`"coronal"` | `"sagittal"` | `"transverse"`

Direction along which to extract the slice limits, specified as `"coronal"`, `"sagittal"`, or `"transverse"`.

Data Types: `char` | `string`

Output Arguments

`xyzCorners` — Coordinates of four corner voxels in patient coordinate system
4-by-3 numeric matrix

Coordinates of the four corner voxels in the patient coordinate system, returned as a 4-by-3 numeric matrix. Each row contains the *xyz*-coordinates for one corner, returned in clockwise order from the upper-left corner.

Version History

Introduced in R2022b

See Also

`medicalVolume` | `sliceLimits`

sliceLimits

Extract *X*-, *Y*-, *Z*-limits for one slice of medical volume

Syntax

```
[XLim,YLim,ZLim] = sliceLimits(medVol,slice,direction)
```

Description

`[XLim,YLim,ZLim] = sliceLimits(medVol,slice,direction)` extracts the *X*-, *Y*-, and *Z*-limits for one slice, `slice`, in the specified direction `direction` of the `medicalVolume` object `medVol`. The slice limits are in the patient coordinate system.

Examples

Get Limits for One Slice of Medical Volume

Get the limits for one slice of a medical volume created using a chest CT volume saved as a directory of DICOM files. The CT volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData/LungCT01");
```

Create a medical volume object for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

Extract the *x*-, *y*-, and *z*-axis limits, in millimeters, of the second slice in the coronal direction.

```
[XLim,YLim,ZLim] = sliceLimits(medVol,2,"coronal")
```

```
XLim = 1×2

 -186.5000  185.7717


YLim = 1×2

 -185.7715 -185.7715


ZLim = 1×2

 -281.2500  -63.7500
```

Input Arguments

`medVol` — Medical volume
`medicalVolume` object

Medical volume, specified as a `medicalVolume` object.

`slice` — Slice index
positive integer scalar in range [1, *numSlices*]

Slice index, specified as a positive integer scalar in the range [1, *numSlices*], where *numSlices* is the number of slices in the volume along the direction specified by `direction`.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logical`

`direction` — Direction along which to extract slice limits
`"coronal"` | `"sagittal"` | `"transverse"`

Direction along which to extract the slice limits, specified as `"coronal"`, `"sagittal"`, or `"transverse"`.

Data Types: `char` | `string`

Output Arguments

`XLim` — Limits of volume in *x*-dimension of patient coordinate system
2-element numeric vector

Limits of the volume in the *x*-dimension of the patient coordinate system, returned as a 2-element numeric vector of the form `[xMin xMax]`. The limits are in the units specified by the `SpatialUnits` property of `medVol`.

`YLim` — Limits of volume in *y*-dimension of patient coordinate system
2-element numeric vector

Limits of the volume in the *y*-dimension of the patient coordinate system, returned as a 2-element numeric vector of the form `[yMin yMax]`. The limits are in the units specified by the `SpatialUnits` property of `medVol`.

`ZLim` — Limits of volume in *z*-dimension of patient coordinate system
2-element numeric vector

Limits of the volume in the *z*-dimension of the patient coordinate system, returned as a 2-element numeric vector of the form `[zMin zMax]`. The limits are in the units specified by the `SpatialUnits` property of `medVol`.

Version History

Introduced in R2022b

See Also

`medicalVolume` | `sliceCorners`

write

Write affine medical volume data to NIfTI file

Syntax

```
write(medVol,filename)
write(medVol,filename,info)
```

Description

`write(medVol,filename)` writes the voxel data and spatial information of the affine `medicalVolume` object `medVol` to the file `filename` in the Neuroimaging Informatics Technology Initiative (NIfTI) file format. The `write` function creates a combined NIfTI file that contains both metadata and volumetric data. The object function populates the metadata using appropriate default values and volume properties, such as size and data type.

`write(medVol,filename,info)` sets metadata attributes by using the metadata structure `info`. If the specified metadata structure does not match the image contents and size, then `write` returns an error.

Examples

Write Medical Volume Object Data to NIfTI File

Write data from a medical volume object, created using a chest CT volume saved as a directory of DICOM files. The CT volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData/LungCT01");
```

Create a medical volume object for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

The `Voxels` property contains the intensity values of each voxel. The `VolumeGeometry` property contains a `medicalref3d` object defining the spatial referencing for the image volume.

```
V = medVol.Voxels;
R = medVol.VolumeGeometry;
```

Modify the voxel data by applying a 3-D Gaussian filter.

```
sigma = 2;
filterV = imgaussfilt3(V,sigma);
```

Create a new `medicalVolume` object that contains the smoothed voxel values. To maintain the same spatial referencing as the original volume, specify the original `medicalref3d` object `R`.

```
medVolSmooth = medicalVolume(filterV,R);
```

Write the smoothed image data to a new NIfTI file.

```
niftiFilename = "LungCT01_smoothed.nii";
write(medVolSmooth,niftiFilename)
```

Input Arguments

`medVol` — Medical volume

`medicalVolume` object

Medical volume, specified as a `medicalVolume` object. `medVol` must be an affine image volume.

An image volume is affine if these conditions are met:

- All slices are parallel to each other.
- The spacing between slices in each dimension is uniform.
- The upper-left voxels of all slices are collinear.
- No two slices are coincident, meaning no two slices are located at the same position in space.

`filename` — Name of NIfTI file

string scalar | character vector

Name of the NIfTI file, specified as a string scalar or a character vector. The `write` function creates a combined NIfTI file that contains both metadata and volumetric data and has the file extension `.nii`.

`info` — NIfTI file metadata

structure

NIfTI file metadata, specified as a structure in the format returned by the `niftiinfo` function.

Version History

Introduced in R2022b

See Also

`medicalVolume` | `niftiinfo` | `niftiwrite`

groundTruthMedical

Ground truth label data for medical images

Description

The `groundTruthMedical` object contains information about the data sources, label definitions, and label data for a collection of medical image data. You can import or export a `groundTruthMedical` object to or from the **Medical Image Labeler** app.

Creation

The **Medical Image Labeler** app automatically creates a `groundTruthMedical` object in the Session folder for an app session. The app saves the object as a MAT file. To manually export a `groundTruthMedical` object to a specific file location, on the **Home** tab of the app toolstrip, select **Export** and, under **Ground Truth**, select **To File**. To create a `groundTruthMedical` object programmatically, use the `groundTruthMedical` function.

Syntax

```
gTruthMed = groundTruthMedical(dataSource,labelDefinitions,labelData)
```

Description

`gTruthMed = groundTruthMedical(dataSource,labelDefinitions,labelData)` creates a `groundTruthMedical` object, `gTruthMed`, that you can import into the **Medical Image Labeler** app.

- `dataSource` specifies the source of the unlabeled ground truth images and sets the `DataSource` property.
- `labelDefs` specifies the label definitions of the ground truth data and sets the `LabelDefinitions` property.
- `labelData` specifies the location of image label data and sets the `LabelData` property.

Properties

`DataSource` — Source of ground truth data
`VolumeSource` object | `ImageSource` object

This property is read-only.

Source of the ground truth data, specified as a `VolumeSource` object or an `ImageSource` object. The data source object specifies the locations of the unlabeled medical image files from which the ground truth data is labeled. You must set this property at object creation by using the `dataSource` input argument.

`LabelDefinitions` — Label definitions
table

This property is read-only.

Label definitions, specified as a table. To create this table, use one of these options:

- In the **Medical Image Labeler** app, create label definitions, and then export them as a MAT file by clicking **Export** and, under **Label Definitions**, selecting **To File**.
- Manually create the label definitions table at the MATLAB command line.

This table describes the required and optional columns of the `LabelDefinitions` table.

Column	Description	Required or Optional
`Name`	String scalars or character vectors, each specifying the name of a label definition.	Required All names must be unique and valid MATLAB variable names. For more details about valid variable names, see "Variable Names".
`PixelLabelID`	Numeric integer scalars in the range [1, 255], each specifying a numeric label ID.	Required All pixel label IDs must be unique.
`LabelColor`	RGB triplets that specify the label colors. Values must be in the range [0, 1].	Optional When you define labels in the **Medical Image Labeler** app, you must specify a color. Therefore, an exported label definitions table always includes this column. When you create label definitions programmatically, you can exclude the `LabelColor` column. When you create a `groundTruthMedical` object, this column is automatically added with default values. The default colors are assigned in the same order as in the **Medical Image Labeler** app.

You must set this property at object creation by using the `labelDefs` input argument.

Data Types: `table`

`LabelData` — Label data file names
n-by-1 string array

This property is read-only.

Label data file names, specified as an *n*-by-1 string array, where *n* is the number of images or image volumes specified by `DataSource`. Each element of `LabelData` contains the name of the label data file for the corresponding image or volume in the data source.

- If the data source is a `VolumeSource` object, then the label data must be stored as NIfTI files.
- If the data source is an `ImageSource` object, then the label data must be stored as MAT files.

- If no labels exist for an image or image volume, specify the corresponding element of `LabelData` as an empty string, `""`.

You must set this property at object creation by using the `labelData` input argument.

Data Types: `string`

Object Functions

changeFilePaths	Change file paths in ground truth data for medical images
merge	Merge two or more groundTruthMedical objects

Examples

Create Ground Truth for Medical Image Data

Create ground truth data for three 2-D X-rays of foreams, stored as DICOM files, and their corresponding label images, stored as MAT files. The images are attached to this example as supporting files. Specify the data directory as the current example directory.

```
dataFolder = pwd;
```

Create a table of the X-ray image file information by using the `dicomCollection` function, and specify the table as the `sourceTable` for an `ImageSource` object.

```
sourceTable = dicomCollection(dataFolder);
dataSource = medical.labeler.loading.ImageSource(sourceTable);
```

Define the labels used to specify the ground truth. Specify the name, display color, and numeric label ID for the bone label data.

```
labelName = "radiusBone";
labelColor = [1 0 0];
labelId = 1;
variableNames = ["Name","LabelColor","PixelLabelID"];
labelDefs = table(labelName,labelColor,labelId,VariableNames=variableNames)
```

```
labelDefs=1×3 table
        Name         LabelColor     PixelLabelID
    ____________    __________    ____________

    "radiusBone"    1    0    0          1
```

Specify the file paths to the label images for each of the images in the data source as a string array. For ground truth data containing an `ImageSource` object, the label images must be in the MAT file format.

```
labelData = [fullfile(dataFolder,"forearmXrayLabels1.mat");fullfile(dataFolder,"forearmXrayLabel
```

Create a `groundTruthMedical` object.

```
gTruthMed = groundTruthMedical(dataSource,labelDefs,labelData)
```

```
gTruthMed = 
  groundTruthMedical with properties:
          DataSource: [1x1 medical.labeler.loading.ImageSource]
           LabelData: [3x1 string]
    LabelDefinitions: [1x3 table]
```

Create Ground Truth for Medical Image Volume Data

Create ground truth data for two chest CT volumes and their corresponding label images, stored in the NIfTI file format. The files are a subset of the Medical Segmentation Decathlon data set [1 on page 1-110]. Download the `MedicalVolumNIfTIData.zip` file from the MathWorks® website, then unzip the file. The size of the data file is approximately 76 MB.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeNIfTIData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
dataFolder = fullfile(filepath,"MedicalVolumeNIfTIData");
```

Create a `VolumeSource` object specifying the two CT volumes.

```
filePath1 = fullfile(dataFolder,"lung_027.nii.gz");
filePath2 = fullfile(dataFolder,"lung_043.nii.gz");
source = {filePath1; filePath2};
dataSource = medical.labeler.loading.VolumeSource(source);
```

Define the labels used to specify the ground truth. Specify the name, display color, and numeric label ID for the tumor label data.

```
labelName = "tumor";
labelColor = [1 0 0];
labelId = 1;
variableNames = ["Name","LabelColor","PixelLabelID"];
labelDefs = table(labelName,labelColor,labelId,VariableNames=variableNames)
```

```
labelDefs=1×3 table
     Name      LabelColor     PixelLabelID
    _______    ___________    ____________

    "tumor"    1    0    0         1
```

Specify the file paths to the label images for each of the images in the data source as a string array. If a data source image does not have label data, specify the corresponding element of the label data array as an empty string.

```
labelDataFile = fullfile(dataFolder,"LabelData","lung_027.nii.gz");
labelData = [labelDataFile ""];
```

Create a `groundTruthMedical` object.

```
gTruthMed = groundTruthMedical(dataSource,labelDefs,labelData)
```

```
gTruthMed = 
  groundTruthMedical with properties:

          DataSource: [1×1 medical.labeler.loading.VolumeSource]
          LabelData: [2×1 string]
    LabelDefinitions: [1×3 table]
```

[1] Medical Segmentation Decathlon. "Lung." Tasks. Accessed May 10, 2018. http://medicaldecathlon.com/.

The Medical Segmentation Decathlon data set is provided under the CC-BY-SA 4.0 license. All warranties and representations are disclaimed. See the license for details.

Version History

Introduced in R2022b

See Also

`ImageSource` | `VolumeSource`

ImageSource

Source of 2-D medical image data for `groundTruthMedical` object

Description

An `ImageSource` object defines the source of ground truth data for medical images or series of images related by time, such as ultrasound data. Use this object to specify the data sources for a `groundTruthMedical` object. Each source must be a single DICOM or NIfTI file.

Creation

When you export labels from a **Medical Image Labeler** app image session, the `DataSource` property of the exported `groundTruthMedical` object contains an `ImageSource` object.

To create an `ImageSource` object programmatically, such as when programmatically creating a `groundTruthMedical` object, use the `medical.labeler.loading.ImageSource` function.

Syntax

```
imgSource = medical.labeler.loading.ImageSource(source)
imgSeqSource = medical.labeler.loading.ImageSource(sourceTable)
```

Description

`imgSource = medical.labeler.loading.ImageSource(source)` creates an `ImageSource` object for loading the 2-D medical image data stored in the files specified by `source`.

`imgSeqSource = medical.labeler.loading.ImageSource(sourceTable)` creates an `ImageSource` object for loading the image files specified in a table, `sourceTable`, returned by the `dicomCollection` function.

Input Arguments

`source` — Source file names
n-by-1 string array | *n*-by-1 cell array of character vectors

Source file names, specified as an *n*-by-1 string array or an *n*-by-1 cell array of character vectors. *n* is the total number of images or series of images related by time to store in the `groundTruthMedical` object. Each name specifies a single DICOM or NIfTI file containing 2-D image data that is readable by a `medicalImage` object.

Data Types: `string` | `cell`

`sourceTable` — Table of source file names
table

Table of source file names, specified as a table returned by the `dicomCollection` function. Each row in `sourceTable` must specify a valid DICOM series that contains a 2-D image or series of images related by time that is readable by a `medicalImage` object. The file name for each DICOM series is stored in the `Filenames` column of the table returned by `dicomCollection`.

Data Types: `table`

Properties

`Source` — Source of ground truth data
n-by-1 string array

This property is read-only.

Source of the ground truth data, specified as an *n*-by-1 string array, where *n* is the total number of images or series of images related by time to store in the `groundTruthMedical` object. Each element in the string array specifies the file name for one image or image series.

Data Types: `string`

Examples

Create Medical Image Data Source from File

Specify the name of a 2-D X-ray image. The file is attached to this example as a supporting file.

```
filename = fullfile(pwd,"forearmXrayImage1.dcm");
```

Create an `ImageSource` object that specifies the X-ray image.

```
imgSource = medical.labeler.loading.ImageSource(filename);
```

Verify that the filename is stored in the `Source` property of the data source object.

```
imgSource.Source;
```

Create Image Data Source from DICOM Collection

Specify the name of the directory containing this example, which includes three 2-D X-ray images of a forearm stored as DICOM files.

```
dataFolder = pwd;
```

Gather the details about the DICOM files in the directory into a table by using the `dicomCollection` function.

```
sourceTable = dicomCollection(dataFolder);
```

Create an `ImageSource` object that specifies the files in `sourceTable`.

```
imgSource = medical.labeler.loading.ImageSource(sourceTable);
```

Verify that each element of the `Source` property contains a filename from the `Filenames` column of `sourceTable`.

```
imgSource.Source;
```

Version History

Introduced in R2022b

See Also

`groundTruthMedical` | `VolumeSource`

medicalImage

2-D medical image pixel data and file metadata

Description

A `medicalImage` object stores the pixel data and metadata for a 2-D medical image or image sequence contained in a single DICOM file. An image sequence is a series of 2-D image frames related by time, such as an ultrasound video.

Creation

Syntax

```
medImage = medicalImage(dirname)
medImage = medicalImage(filename)
medImage = medicalImage(sourceTable)
medImage = medicalImage(sourceTable,rowname)
medImage = medicalImage(imds)
medImage = medicalImage(pixels,info)
```

Description

`medImage = medicalImage(dirname)` creates a `medicalImage` object for the single DICOM file in the directory `dirname`.

`medImage = medicalImage(filename)` creates a `medicalImage` object for the image contained in the single DICOM file `filename`.

`medImage = medicalImage(sourceTable)` creates a `medicalImage` object for the image listed in `sourceTable`. The table must contain only one row that specifies the metadata for a 2-D DICOM image.

`medImage = medicalImage(sourceTable,rowname)` creates a `medicalImage` object for the image listed in the row `rowname` of `sourceTable`. Use this syntax to specify one row of a multirow table by name or by its numeric index.

`medImage = medicalImage(imds)` creates a `medicalImage` object for the image specified by the image datastore object `imds`.

`medImage = medicalImage(pixels,info)` creates a `medicalImage` object by specifying the image data `pixels` and `info`, a metadata structure returned by the `dicominfo` function. `pixels` sets value of the `Pixels` property.

Input Arguments

`dirname` — Name of directory containing medical image data
string scalar | character vector

Name of the directory containing the medical image data, specified as a string scalar or a character vector. `dirname` is the name of a directory that contains one DICOM file that specifies one 2-D image or sequence of 2-D images related by time, such as an ultrasound video.

`filename` — Name of file containing medical image data
string scalar | character vector

Name of the file containing the medical image data, specified as a string scalar or a character vector. `filename` can specify a single DICOM file that contains a 2-D image or series of 2-D images related by time, such as an ultrasound video.

`sourceTable` — Collection of DICOM file metadata
table

Collection of DICOM file metadata, specified as a table returned by the `dicomCollection` function.

`rowname` — Name or index of table row
string scalar | character vector | positive integer

Name or index of the table row to read from `sourceTable`, specified as a string scalar, character vector, or positive integer scalar. Specify `rowname` as a string scalar or character vector to specify the row by name. Specify `rowname` as a positive integer scalar to specify the row by its numeric index.

`imds` — Datastore containing one DICOM file
`ImageDatastore` object

Datastore containing one DICOM file, specified as an `imageDatastore` object.

`info` — DICOM file metadata
structure

DICOM file metadata, specified as a structure returned by the `dicominfo` function.

Properties

`Pixels` — Image pixel values
m-by-*n*-by-*t*-by-*c* numeric array

Image pixel values, specified as an *m*-by-*n*-by-*t*-by-*c* numeric array, where *m* and *n* are the spatial dimensions of the 2-D image or image frames, *t* is the number of image frames related by time, and *c* is the number of color channels in each frame. If the file contains the `RescaleIntercept` and `RescaleSlope` metadata attributes, then `medicalImage` rescales the intensity values based on them.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logical`

`Colormap` — Colormap
j-by-3 numeric matrix | `[]`

Colormap, specified as one of these options:

- If the source is an indexed image, then specify `Colormap` as a *j*-by-3 numeric matrix with values in the range [0, 1]. Each row is a three-element RGB triplet that specifies the red, green, and blue components of a single color of the colormap.

- If the source is a grayscale or RBG image, then `Colormap` is empty (`[]`).

Data Types: `double`

`SpatialUnits` — Real-world spatial units
`"mm"` | `"unknown"`

This property is read-only.

Real-world spatial units, specified as one of these options:

- If the pixel spacing is specified in the file metadata, then the spatial units value is `"mm"`.
- If the metadata does not specify the pixel spacing information, then `SpatialUnits` is `"unknown"`.

Data Types: `string`

`FrameTime` — Number of milliseconds between frames
numeric scalar | `[]`

This property is read-only.

Number of milliseconds between frames, specified as one of these options:

- If the `Frame Time` metadata attribute is specified in the file, then the frame time value is specified as a numeric scalar extracted from the metadata.
- If the metadata does not specify the frame time, then `FrameTime` is empty (`[]`).

Data Types: `double`

`NumFrames` — Number of frames
numeric scalar

This property is read-only.

Number of frames in the image data, specified as a numeric scalar.

Data Types: `double`

`PixelSpacing` — Spacing between pixel centers
`[1 1]` (default) | 2-element row vector

This property is read-only.

Spacing between pixel centers, specified as a 2-element vector of the form [Δx Δy]. The Δx and Δy values are the distances between adjacent pixels in the *x*- and *y*-directions, respectively, in the units specified by the `SpatialUnits` property. If the file metadata does not specify the pixel spacing, the default value is `[1 1]`.

Data Types: `double`

`Modality` — Imaging modality used to capture image data
`"unknown"` (default) | string scalar

This property is read-only.

Imaging modality used to capture image data, specified as a string scalar. The modality is extracted from the file metadata if it is present. Common values include, but are not limited to, "DX" for digital radiography, "US" for ultrasound, "XA" for X-ray angiography, and "MG" for mammography. If the file metadata does not specify the modality, then the default value is "unknown".

Data Types: string

WindowCenter — Center of display range window
numeric scalar | [ ]

This property is read-only.

Center of the display range window, specified as one of these options:

- If the Window Center attribute is specified in the file metadata, then the WindowCenter value is specified as a numeric scalar extracted from the metadata.
- If the metadata does not specify the window center, then WindowCenter is empty ([]).

Data Types: double

WindowWidth — Width of display range window
numeric scalar | [ ]

This property is read-only.

Width of the of display range window, specified as one of these options:

- If the Window Width attribute is specified in the file metadata, then the WindowWidth value is specified as a numeric scalar extracted from the metadata.
- If the metadata does not specify the window width, then WindowWidth is empty ([]).

Data Types: double

Object Functions

extractFrame	Extract pixel data for one frame of 2-D medical image series

Examples

Create Medical Image Object from Filename

Specify the name of a 2-D X-ray image. The image is attached to this example as a supporting file.

```
filename = "forearmXrayImage1.dcm";
```

Create a medicalImage object for the file.

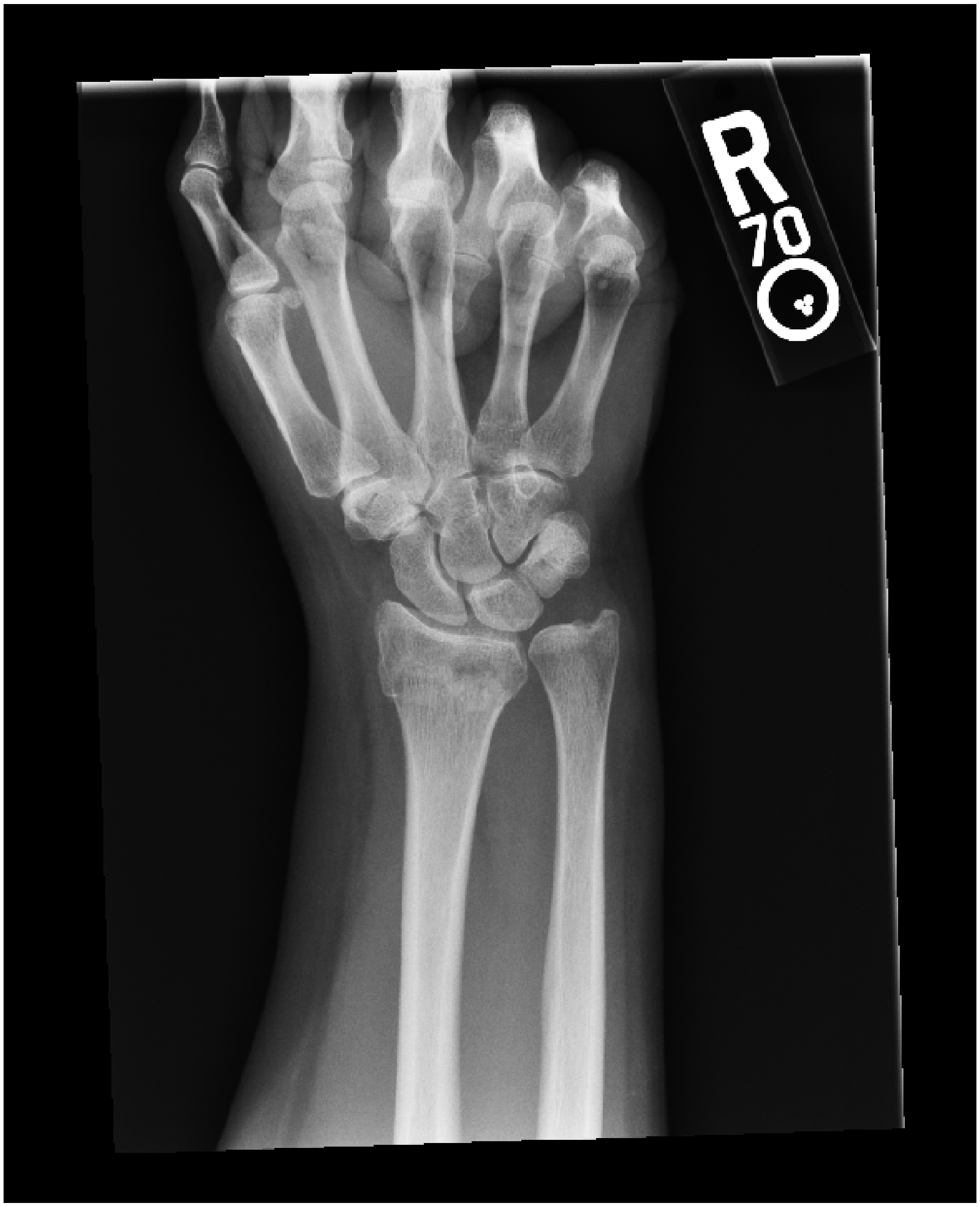
```
medImage = medicalImage(filename)
```

```
medImage = 
  medicalImage with properties:

          Pixels: [1540x1250 uint16]
        Colormap: []
    SpatialUnits: "mm"
```

```
       FrameTime: []
       NumFrames: 1
    PixelSpacing: [0.1390 0.1390]
        Modality: 'DX'
    WindowCenter: 2048
     WindowWidth: 4096
```

The image pixel data is stored in the `Pixels` property. Display the image stored in `medImage`.

```
imshow(medImage.Pixels,[])
```


R
70

Create Medical Image Object from DICOM Collection

Specify the name of the directory containing this example, which includes an echocardiogram image series stored as a DICOM file.

```
dirname = pwd;
```

Gather the details about the DICOM files in the directory into a table by using the `dicomCollection` function.

```
sourceTable = dicomCollection(dirname);
```

Create a `medicalImage` object for the echocardiogram sequence.

```
medImage = medicalImage(sourceTable)
```

```
medImage = 
  medicalImage with properties:

          Pixels: [600x800x3x3 uint8]
        Colormap: []
    SpatialUnits: "unknown"
       FrameTime: []
       NumFrames: 3
    PixelSpacing: [1 1]
        Modality: 'OT'
    WindowCenter: []
     WindowWidth: []
```

Create Medical Image Object from Specified Row of DICOM Collection

Specify the name of the directory containing this example, which includes three 2-D X-ray images of a forearm.

```
dataFolder = pwd;
```

Gather the details about the DICOM files in the directory into a table by using the `dicomCollection` function. The table contains three rows corresponding to the three DICOM series.

```
sourceTable = dicomCollection(dataFolder);
```

Create a medical image object for the second X-ray image by specifying the DICOM collection table and a numeric row index.

```
medImage = medicalImage(sourceTable,2)
```

```
medImage = 
  medicalImage with properties:

          Pixels: [1932x1492 uint16]
        Colormap: []
    SpatialUnits: "mm"
       FrameTime: []
       NumFrames: 1
    PixelSpacing: [0.1390 0.1390]
```

```
        Modality: 'DX'
    WindowCenter: 2048
     WindowWidth: 4096
```

Create Medical Image Object from Image Datastore

Specify the name of the directory containing this example, which includes an echocardiogram image series stored as a single DICOM file.

```
dataFolder = pwd;
```

Create an image datastore containing the DICOM file in the `dataFolder` directory. Specify a custom read function to read the DICOM file.

```
dicomds = imageDatastore(dataFolder, ...
    FileExtensions=".dcm",ReadFcn=@(x) dicomread(x));
```

Create a medical image object for the echocardiogram series.

```
medImage = medicalImage(dicomds)
```

```
medImage = 
  medicalImage with properties:

          Pixels: [600x800x3x3 uint8]
        Colormap: []
    SpatialUnits: "unknown"
       FrameTime: []
       NumFrames: 3
    PixelSpacing: [1 1]
        Modality: 'OT'
    WindowCenter: []
     WindowWidth: []
```

Create Medical Image Object from Pixels and Metadata

Specify the name of a 2-D X-ray image. The file is attached to this example as a supporting file.

```
filename = "forearmXrayImage1.dcm";
```

Read the metadata from the DICOM file by using the `dicominfo` function.

```
info = dicominfo(filename);
```

Create a `medicalImage` object for the file.

```
medImage = medicalImage(filename)
```

```
medImage = 
  medicalImage with properties:
          Pixels: [1540x1250 uint16]
        Colormap: []
    SpatialUnits: "mm"
       FrameTime: []
       NumFrames: 1
    PixelSpacing: [0.1390 0.1390]
        Modality: 'DX'
    WindowCenter: 2048
     WindowWidth: 4096
```

The image pixel data is stored in the `Pixels` property. Apply a smoothing filter to the pixel values and save the smoothed values as a new image array, `imSmooth`.

```
sigma = 2;
imSmooth = imgaussfilt(medImage.Pixels,sigma);
```

Create a new `medicalImage` object that contains the smoothed pixel values. To maintain the same spatial information as the original image, specify the metadata structure `info`.

```
medImageSmoothed = medicalImage(imSmooth,info)
```

```
medImageSmoothed = 
  medicalImage with properties:

          Pixels: [1540x1250 uint16]
        Colormap: []
    SpatialUnits: "mm"
       FrameTime: []
       NumFrames: 1
    PixelSpacing: [0.1390 0.1390]
        Modality: 'DX'
    WindowCenter: 2048
     WindowWidth: 4096
```

Version History
Introduced in R2022b

See Also
`ImageSource` | `medicalVolume`

Topics
"Read, Process, and View Ultrasound Data"

medicalref3d

Spatial referencing information for 3-D medical image volumes

Description

A `medicalref3d` object stores the relationship between the intrinsic coordinate system anchored to the columns, rows, and planes of a medical image volume and the real-world patient coordinate system.

The `medicalref3d` object handles affine and non-affine medical image volumes. For non-affine volumes, the object stores information about the spacing and orientation of each slice in each dimension. You can also use the `medicalref3d` object to access and update the mapping between the patient coordinate system axes and the left-right, superior-inferior, and anterior-posterior anatomical axes.

Creation

You can create a `medicalred3d` object in these ways:

- `medicalVolume` — Creates a `medicalref3d` object, stored in the `VolumeGeometry` property.
- The `medicalref3d` function described here.

Syntax

```
R = medicalref3d(volumeSize,tform)
R = medicalref3d(volumeSize,position,pixelSpacing,cosines)
R = medicalref3d(volumeSize,position,voxelDistances)
```

Description

`R = medicalref3d(volumeSize,tform)` creates a `medicalref3d` object and sets the `VolumeSize` property. `tform` is an `affinetform3d` object that specifies an affine image volume.

`R = medicalref3d(volumeSize,position,pixelSpacing,cosines)` specifies the spatial referencing information for an affine or non-affine image volume by setting the `Position` and `PixelSpacing` properties, and specifies the orientation of each slice using the `cosines` argument.

`R = medicalref3d(volumeSize,position,voxelDistances)` specifies the spatial referencing information for an affine or non-affine image volume by setting the `Position` and `VoxelDistances` properties.

Input Arguments

`tform` — Geometric transformation between image data array and patient coordinate system
`affinetform3d` object

Geometric transformation between the image data array and the patient coordinate system, specified as an `affinetform3d` object. The `A` property of `tform` contains a 4-by-4 3-D transformation matrix that maps triplets of pixel indices in the order (*column*, *row*, *slice*) to (*x*, *y*, *z*) triplets of patient coordinates in real-world units.

`cosines` — Orientation of each slice in patient coordinate system
2-by-3-by-*p* numeric array

Orientation of each slice in the patient coordinate system, specified as a 2-by-3-by-*p* numeric array, where *p* is the number of slices in the image volume along the third dimension. Each 2-by-3 matrix specifies the orientation of one slice:

- The first row specifies the direction cosines of the first dimension, which is aligned with the rows of the data array, relative to the *x*-, *y*-, and *z*-axes of the patient coordinate system.
- The second row specifies the direction cosines of the second dimension, which is aligned with the columns of the data array, relative to the *x*-, *y*-, and *z*-axes of the patient coordinate system.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64`

Properties

`VolumeSize` — Number of voxels in each dimension of data array
3-element row vector of positive integers

Number of voxels in each dimension of the data array, specified as a 3-element row vector of positive integers. The three elements of `VolumeSize` specify the number of rows, columns, and slices, respectively, in the image data. You must set this property at object creation by using the `volumeSize` input argument. Otherwise, this property is read-only.

Data Types: `double`

`Position` — Patient coordinates of pixel in first index of each slice in data array
p-by-3 numeric matrix

Patient coordinates of the pixel in the first index of each slice in the data array, specified as a *p*-by-3 numeric matrix. *p* is the number of slices in the image volume along the third dimension. Each row specifies the *xyz*-coordinates, in real-world units, of the center of the first pixel of the corresponding slice. The first pixel is the pixel in the first row and column of the data array. You can set this property at object creation directly, by using the `position` input argument, or indirectly, by using the `tform` input argument. Otherwise, this property is read-only.

Data Types: `double`

`VoxelDistances` — 3-D displacement vectors between adjacent voxels
3-element cell array of *p*-by-3 numeric matrices | 3-element cell array of 1-by-3 numeric matrices

3-D displacement vectors between adjacent voxels, specified as a 3-element cell array of *p*-by-3 numeric matrices or 1-by-3 numeric matrices. *p* is the number of slices in the image volume along the third dimension. `VoxelDistances` defines the 3-D mapping between the intrinsic coordinate system, in voxels, and the patient coordinate system, in real-world units. Displacements can be positive or negative.

- `VoxelDistances{1}` specifies the real-world displacements [Δx Δy Δz] corresponding to a [1 0 0] voxel displacement.
- `VoxelDistances{2}` specifies the real-world displacements [Δx Δy Δz] corresponding to a [0 1 0] voxel displacement.

- `VoxelDistances{3}` specifies the real-world displacements [*Δx Δy Δz*] corresponding to a [0 0 1] voxel displacement.

For example, `VoxelDistances{2}(1,:)` specifies the vector between voxels of the first slice when moving along the second dimension of the data array.

If the image volume is non-affine, or if the `medicalref3d` object was created as the `VolumeGeometry` property value of a `medicalVolume` object for a DICOM file, then `VoxelDistances` specifies displacement vectors for each slice in a separate row. Otherwise, each element of `VoxelDistances` simplifies to a 1-by-3 matrix.

You can set this property at object creation directly, by using the `voxelDistances` input argument, or indirectly, by using the `tform`, `position`, `pixelSpacing`, or `cosines` input argument. Otherwise, this property is read-only.

Data Types: `cell`

`PatientCoordinateSystem` — Orientation convention of patient coordinate system relative to anatomical axes
`"unknown"` (default) | `"LPS+"` | `"LAS+"` | `"RAS+"`

Orientation convention of the patient coordinate system relative to the anatomical axes, specified as `"LPS+"`, `"LAS+"`, `"RAS+"`, or `"unknown"`. For `"LPS+"`, `"LAS+"`, and `"RAS+"`, the first three characters indicate the positive direction of the *x*-, *y*-, and *z*-axes of the patient coordinate system, respectively.

- The positive direction of the *x*-axis points left (`"L"`) or right (`"R"`).
- The positive direction of the *y*-axis points anterior (`"A"`) or posterior (`"P"`).
- The positive direction of the *z*-axis points inferior (`"I"`) or superior (`"S"`).
- `"+"` indicates that values increase in the stated direction.

For example, `"LPS+"` specifies a patient coordinate system with the *x*-, *y*-, and *z*-axes positive in the left, posterior, and superior directions, respectively.

The default value is `"unknown"`. For a `medicalref3d` object created by the `medicalVolume` function, the `PatientCoordinateSystem` is set based on the corresponding attribute of the file metadata, if present.

Data Types: `char` | `string`

`PixelSpacing` — Spacing between voxel centers in first two dimensions of each slice
p-by-2 matrix of positive values | 2-element vector of positive values

Spacing between voxel centers in the first two dimensions of each slice, specified as a *p*-by-2 matrix or 2-element vector of positive values. *p* is the number of slices in the image volume along the third dimension. Each row specifies the distances between voxel centers, for one slice, in the form `[xSpacing ySpacing]`. Distances are in real-world units such as millimeters.

If the image volume is non-affine, or if the `medicalref3d` object was created as the `VolumeGeometry` property value of a `medicalVolume` object for a DICOM file, then the `PixelSpacing` property specifies spacing for each slice in a separate row. Otherwise, `PixelSpacing` simplifies to a 2-element vector.

You can set this property at object creation directly, by using the `pixelSpacing` input argument, or indirectly, by using the `tform` or `voxelDistances` input argument. Otherwise, this property is read-only.

Data Types: `double`

`IsAffine` — Spatial referencing system is affine
`true` or `1` | `false` or `0`

This property is read-only.

Spatial referencing system is affine, specified as a logical `1` (`true`) or `0` (`false`). An image volume is affine if these conditions are met:

- All slices are parallel to each other.
- The spacing between slices in each dimension is uniform.
- The upper-left voxels of all slices are collinear.
- No two slices are coincident, meaning no two slices are located at the same position in space.

Data Types: `logical`

`IsAxesAligned` — Image data array is aligned with patient coordinate system
`true` or `1` | `false` or `0`

This property is read-only.

Image data array is aligned with the patient coordinate system, specified as a logical `1` (`true`) or `0` (`false`). A `true` value indicates that the directions along which the row, column, and slice indices change are aligned with the *x*-, *y*-, and *z*-axes of the patient coordinate system.

Data Types: `logical`

`IsMixed` — Orientation of image slices is mixed
`true` or `1` | `false` or `0`

This property is read-only.

Orientation of the image slices is mixed, specified as a logical `1` (`true`) or `0` (`false`). A `false` value indicates that all of the slices in the image volume are parallel and have a similar orientation. A `true` value indicates that one or more slices are not parallel and similarly oriented to other slices.

Data Types: `logical`

Object Functions

intrinsicToWorldMapping	Geometric transform between intrinsic and patient coordinates of medical image volume
contains	Determine if affine image volume contains points specified in patient coordinate system
intrinsicToWorld	Map points from intrinsic coordinates to patient coordinates
oneSliceIntrinsicToWorldMapping	Geometric transform between intrinsic and patient coordinates of medical image volume slice
orient	Update patient coordinate system convention
sliceCorners	Extract patient coordinates of corner voxels for one slice
worldToIntrinsic	Map points from patient coordinates to intrinsic coordinates

worldToSubscript	Convert from patient coordinates to row and column subscripts

Examples

Create `medicalref3d` Object Using Volume Size and Geometric Transformation

Specify the size of the target image volume array, in voxels.

```
volumeSize = [256 256 100];
```

Specify the geometric transformation that describes the mapping between the intrinsic and patient coordinates for the target image volume. Assume the patient coordinate system has units of millimeters.

Specify the scale factors that correspond to voxel dimensions of 0.5 mm, 0.5 mm, and 1.0 mm.

```
sx = 0.5;
sy = 0.5;
sz = 1;
```

Specify the position of the first voxel of the first transverse slice, in millimeters. The position defines the translation between the intrinsic coordinate origin and the patient coordinate origin.

```
tx = 1000;
ty = 50;
tz = 75;
```

Define a 4-by-4 geometric transformation matrix that performs the voxel scaling and translation. The first column corresponds to the *y*-dimension of the image, and the second column corresponds to the *x*-dimension.

```
A = [0 sx 0 tx;
    sy 0 0 ty;
    0 0 sz tz;
    0 0 0 1];
```

Create an `affinetform3d` object that performs the transformation.

```
tform = affinetform3d(A);
```

Create a `medicalref3d` object with the specified array size and geometric transformation.

```
R = medicalref3d(volumeSize,tform)
```

```
R = 
  medicalref3d with properties:

                 VolumeSize: [256 256 100]
                   Position: [100x3 double]
             VoxelDistances: {[0.5000 0 0]  [0 0.5000 0]  [0 0 1]}
    PatientCoordinateSystem: "Unknown"
               PixelSpacing: [0.5000 0.5000]
                   IsAffine: 1
              IsAxesAligned: 1
                    IsMixed: 0
```

Version History

Introduced in R2022b

See Also

Topics

“Read, Process, and Write 3-D Medical Images”
“Display 3-D Medical Image Data in Patient Coordinate System”
“Display Labeled Medical Image Volume in Patient Coordinates”

medicalVolume

3-D medical image voxel data and spatial referencing information

Description

A `medicalVolume` object stores the voxel data and spatial referencing information for the medical image volume contained in a single DICOM, NIfTI, or NRRD file, or in a directory of DICOM files. The `medicalVolume` object specifies the mapping between the intrinsic image coordinate system, the patient coordinate system, and the anatomical planes. The `medicalVolume` object and its object functions provide a standardized interface for accessing voxel data, spatial referencing, and intensity scaling information.

Creation

Syntax

```
medVol = medicalVolume(dirname)
medVol = medicalVolume(filenames)
medVol = medicalVolume(sourceTable)
medVol = medicalVolume(sourceTable,rowname)
medVol = medicalVolume(imds)
medVol = medicalVolume(voxels,VolumeGeometry)
```

Description

`medVol = medicalVolume(dirname)` creates a `medicalVolume` object for the image volume contained in a multifile DICOM series in the directory `dirname`.

`medVol = medicalVolume(filenames)` creates a `medicalVolume` object for the image volume stored in the single DICOM, NIfTI, or NRRD file or list of DICOM files specified by `filenames`.

`medVol = medicalVolume(sourceTable)` creates a `medicalVolume` object for the image volume listed in `sourceTable`. The table must contain only one row that specifies the metadata for a DICOM volume.

`medVol = medicalVolume(sourceTable,rowname)` creates a `medicalVolume` object for the image volume listed in the row `rowname` of `sourceTable`. Use this syntax to specify one row of a multirow table by name or by numeric index.

`medVol = medicalVolume(imds)` creates a `medicalVolume` object for the image volume specified by the image datastore object `imds`.

`medVol = medicalVolume(voxels,VolumeGeometry)` creates a `medicalVolume` by directly specifying the Voxels and “VolumeGeometry” on page 1-0 properties.

Input Arguments

`dirname` — Name of directory containing medical image volume data
string scalar | character vector

Name of the directory containing the medical image volume data, specified as a string scalar or a character vector. Specify `dirname` as the name of a directory containing multiple DICOM files corresponding to one image volume.

`filenames` — Name of file or files containing medical image volume data
string scalar | character vector | string array

Name of the file or files containing the medical image volume data, specified as a string scalar, character vector, or string array. Specify `filenames` as a single DICOM, NIfTI, or NRRD file or as a list of DICOM files corresponding to one image volume.

`sourceTable` — Collection of DICOM file metadata
table

Collection of DICOM file metadata, specified as a table returned by the `dicomCollection` function.

`rowname` — Name or index of table row
string scalar | character vector | positive integer

Name or index of table row, specified as a string scalar, character vector, or positive integer. Specify `rowname` as a string scalar or character vector to specify a row of `sourceTable` by name. Specify `rowname` as a positive integer to specify a row by its numeric index.

`imds` — Datastore specifying list of DICOM files containing medical image volume data
`ImageDatastore` object

Datastore specifying a list of DICOM files containing medical image volume data, specified as an `ImageDataStore` object. The list of files must correspond to one medical image volume.

Properties

`Voxels` — Image voxel values
m-by-*n*-by-*p* numeric array | *N*-D numeric array

Image voxel values, specified as an *m*-by-*n*-by-*p* numeric array or an *N*-D numeric array. The first three dimensions of `Voxels` correspond to the spatial dimensions of the patient coordinate system. For DICOM files, `medicalVolume` rescales the intensity values using the `RescaleIntercept` and `RescaleSlope` metadata attributes, if they are present in the file.

Data Types: `single` | `double` | `int8` | `int16` | `int32` | `int64` | `uint8` | `uint16` | `uint32` | `uint64` | `logical`

`VolumeGeometry` — Spatial referencing information
`medicalref3d` object

Spatial referencing information, specified as a `medicalref3d` object. The coordinates and distances specified by `VolumeGeometry` are in the units specified by the `SpatialUnits` property, if available in the file metadata.

`SpatialUnits` — Real-world spatial units
`"unknown"` (default) | string scalar

Real-world spatial units, specified as a string scalar or a character vector. If the data source is a DICOM file, then the `SpatialUnits` value is `"mm"` (millimeters). If the data source is a NIfTI or NRRD file, the object extracts the spatial units from the `SpaceUnits` or `spaceunits` metadata attribute, respectively, if present in the file.

Data Types: `string`

`VoxelSpacing` — Distances between voxel centers in each dimension
1-by-3 numeric vector

This property is read-only.

Distances between voxel centers in each dimension, specified as a 1-by-3 numeric vector. The values of `VoxelSpacing` are in the units specified by the `SpatialUnits` property. If `VolumeGeometry.IsAffine` is `false`, then the object calculates the spacing for each dimension as the average spacing across all slices.

Data Types: `double`

`Orientation` — Slice plane with greatest spatial resolution
`"coronal"` | `"mixed"` | `"oblique"` | `"sagittal"` | `"tranverse"` | `"unknown"`

This property is read-only.

Slice plane with the greatest spatial resolution, specified as one of these values:

- `"coronal"` — Coronal plane.
- `"sagittal"` — Sagittal plane.
- `"transverse"` — Transverse plane.
- `"mixed"` — Image volume slices are not parallel. This value occurs when the `IsMixed` property of the `VolumeGeometry` object is `true`.
- `"oblique"` — Image slices are not aligned with the anatomical axes.
- `"unknown"` — Mapping between image coordinate system and anatomical axes is unknown. This value occurs when the `PatientCoordinateSystem` property of the `VolumeGeometry` object is `"unknown"`.

Data Types: `string`

`NumCoronalSlices` — Number of slices in coronal direction
numeric scalar | `[]`

This property is read-only.

Number of slices in the coronal direction, specified as a numeric scalar or empty array. `NumCoronalSlices` is empty in these cases:

- `PatientCoordinateSystem` property of the `VolumeGeometry` object is `"unknown"`.
- Image volume is oblique.
- Image volume contains a temporal dimension with multiple slices at each spatial location.

Data Types: `double`

`NumSaggitalSlices` — Number of slices in sagittal direction
numeric scalar | `[]`

This property is read-only.

Number of slices in the sagittal direction, specified as a numeric scalar or empty array. `NumCoronalSlices` is empty in these cases:

- `PatientCoordinateSystem` property of the `VolumeGeometry` object is `"unknown"`.
- Image volume is oblique.
- Image volume contains a temporal dimension with multiple slices at each spatial location.

Data Types: `double`

`NumTransverseSlices` — Number of slices in transverse direction
numeric scalar | `[]`

This property is read-only.

Number of slices in the transverse direction, specified as a numeric scalar or empty array. `NumCoronalSlices` is empty in these cases:

- `PatientCoordinateSystem` property of the `VolumeGeometry` object is `"unknown"`.
- Image volume is oblique.
- Image volume contains a temporal dimension with multiple slices at each spatial location.

Data Types: `double`

`PlaneMapping` — Mapping between image data dimensions and anatomical planes
1-by-3 string array

This property is read-only.

Mapping between image data dimensions and anatomical planes, specified as a 1-by-3 string array. If the spatial referencing information is available in the source file, `PlaneMapping` contains the strings `"transverse"`, `"coronal"`, and `"saggital"`. For example, if the first element of `PlaneMapping` is `"coronal"`, then moving along the first dimension of `Voxels` corresponds to moving between slices in the coronal plane.

If the `PatientCoordinateSystem` property of the `VolumeGeometry` object is `"unknown"`, or if the volume is oblique, the values of `PlaneMapping` are `"unknown"` or `"oblique"`, respectively.

Example: `["transverse","coronal","sagittal"]`

Data Types: `string`

`NormalVector` — Unit vector normal to first slice of image volume
1-by-3 numeric vector

This property is read-only.

Unit vector normal to the first slice of the image volume, specified as a 1-by-3 numeric vector. The first slice of the volume corresponds to the points stored in `Voxels(:,:,1)`.

Data Types: `double`

`Modality` — Imaging modality used to capture image volume data
`"unknown"` (default) | string scalar

This property is read-only.

Imaging modality used to capture the image volume data, specified as a string scalar. The modality is extracted from the file metadata, if present. Common values include, but are not limited to, `"CT"` for computed tomography, `"MR"` for magnetic resonance, `"NM"` for nuclear medicine, and `"US"` for ultrasound. If the modality is not specified in the file metadata, the default value is `"unknown"`.

Data Types: `string`

`WindowCenters` — Center of display range window of each slice
`[]` (default) | *p*-by-1 numeric vector

This property is read-only.

Center of the display range window for each slice, specified as a *p*-by-1 numeric vector, where *p* is the number of slices in the image volume along the third dimension. The object extracts the `WindowCenters` property value from the file metadata, if available. If the display window metadata is not available, then `WindowCenters` is empty.

Data Types: `double`

`WindowWidths` — Width of display window of each slice
`[]` (default) | *p*-by-1 numeric vector

This property is read-only.

Width of display window of each slice, specified as a *p*-by-1 numeric vector, where *p* is the number of slices in the image volume along the third dimension. The object extracts the `WindowWidths` property value from the file metadata, if available. If the display window metadata is not available, then `WindowWidths` is empty.

Data Types: `double`

Object Functions

extractSlice	Extract voxels and spatial details for one slice of medical volume
replaceSlice	Replace voxel values for one slice of medical volume
resample	Resample medical image volume in different patient coordinate system
sliceCorners	Extract coordinates of corner voxels for one slice of medical volume
sliceLimits	Extract X-, Y-, Z-limits for one slice of medical volume
write	Write affine medical volume data to NIfTI file

Examples

Create Medical Volume Object for Multifile DICOM File

Create a medical volume object using a chest CT volume saved as a directory of DICOM files. The volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks® website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData/LungCT01");
```

Create a medical volume object for the CT volume.

```
medVol = medicalVolume(dataFolder)
```

```
medVol = 
  medicalVolume with properties:

                 Voxels: [512×512×88 int16]
         VolumeGeometry: [1×1 medicalref3d]
           SpatialUnits: "mm"
            Orientation: "transverse"
           VoxelSpacing: [0.7285 0.7285 2.5000]
           NormalVector: [0 0 1]
       NumCoronalSlices: 512
      NumSagittalSlices: 512
    NumTransverseSlices: 88
           PlaneMapping: ["sagittal"    "coronal"    "transverse"]
               Modality: "CT"
          WindowCenters: [88×1 double]
           WindowWidths: [88×1 double]
```

Create Medical Volume Object from Filename

Create a medical volume object using a CT chest volume from the Medical Segmentation Decathlon data set [1 on page 1-135]. Download the `MedicalVolumNIfTIData.zip` file from the MathWorks website, then unzip the file. The file contains two CT chest volumes and corresponding label images, stored in the NIfTI file format. The size of the data file is approximately 76 MB.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeNIfTIData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
dataFolder = fullfile(filepath,"MedicalVolumeNIfTIData");
```

Specify the file name of the first CT volume.

```
filePath = fullfile(dataFolder,"lung_027.nii.gz");
```

Create a medical volume object for the CT volume.

```
medVol = medicalVolume(filePath)
```

```
medVol = 
  medicalVolume with properties:

                 Voxels: [512×512×264 single]
         VolumeGeometry: [1×1 medicalref3d]
           SpatialUnits: "mm"
```

```
            Orientation: "transverse"
           VoxelSpacing: [0.8594 0.8594 1.2453]
           NormalVector: [0 0 -1]
       NumCoronalSlices: 512
      NumSagittalSlices: 512
    NumTransverseSlices: 264
           PlaneMapping: ["sagittal"    "coronal"    "transverse"]
               Modality: "unknown"
          WindowCenters: 0
           WindowWidths: 0
```

[1] Medical Segmentation Decathlon. "Lung." Tasks. Accessed May 10, 2018. http://medicaldecathlon.com/.

The Medical Segmentation Decathlon data set is provided under the CC-BY-SA 4.0 license. All warranties and representations are disclaimed. See the license for details.

Create Medical Volume Object from DICOM Collection

Create a medical volume object using a chest CT volume saved as a directory of DICOM files. The volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData/LungCT01");
```

Gather the details about the DICOM files in the `dataFolder` directory into a table by using the `dicomCollection` function. The files belong to one CT volume series, so the table has one row.

```
sourceTable = dicomCollection(dataFolder);
```

Create a medical volume object for the CT volume by specifying the single-row DICOM collection table.

```
medVol = medicalVolume(sourceTable)
```

```
medVol = 
  medicalVolume with properties:

                 Voxels: [512×512×88 int16]
         VolumeGeometry: [1×1 medicalref3d]
           SpatialUnits: "mm"
            Orientation: "transverse"
           VoxelSpacing: [0.7285 0.7285 2.5000]
           NormalVector: [0 0 1]
       NumCoronalSlices: 512
      NumSagittalSlices: 512
```

```
    NumTransverseSlices: 88
           PlaneMapping: ["sagittal"    "coronal"    "transverse"]
               Modality: "CT"
          WindowCenters: [88×1 double]
           WindowWidths: [88×1 double]
```

Create Medical Volume Object from Specified Row of DICOM Collection

Create a medical volume object using a data set containing three chest CT scans. Each CT scan is saved as a directory of DICOM files. The size of the data set is approximately 81 MB. Download the data set from the MathWorks website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData");
```

Gather the details about the DICOM files in the `dataFolder` directory into a table by using the `dicomCollection` function. The table contains three rows, each corresponding to one of three CT DICOM series.

```
sourceTable = dicomCollection(dataFolder);
```

Create a medical volume object for the second CT volume by specifying the DICOM collection table and a numeric row index.

```
medVol = medicalVolume(sourceTable,2)
```

```
medVol = 
  medicalVolume with properties:

                 Voxels: [512×512×88 int16]
         VolumeGeometry: [1×1 medicalref3d]
           SpatialUnits: "mm"
            Orientation: "transverse"
           VoxelSpacing: [0.7617 0.7617 2.5000]
           NormalVector: [0 0 1]
      NumCoronalSlices: 512
     NumSagittalSlices: 512
    NumTransverseSlices: 88
           PlaneMapping: ["sagittal"    "coronal"    "transverse"]
               Modality: "CT"
          WindowCenters: [88×1 double]
           WindowWidths: [88×1 double]
```

Create Medical Volume Object from Image Datastore

Create a medical volume object using a chest CT volume saved as a directory of DICOM files. The volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData/LungCT01");
```

Create an image datastore containing the DICOM files in the `dataFolder` directory. Specify a custom read function to read the DICOM files.

```
dicomds = imageDatastore(dataFolder, ...
    FileExtensions=".dcm",ReadFcn=@(x) dicomread(x));
```

Create a medical volume object for the CT volume.

```
medVol = medicalVolume(dicomds)
```

```
medVol = 
  medicalVolume with properties:

                 Voxels: [512×512×88 int16]
         VolumeGeometry: [1×1 medicalref3d]
           SpatialUnits: "mm"
            Orientation: "transverse"
           VoxelSpacing: [0.7285 0.7285 2.5000]
           NormalVector: [0 0 1]
       NumCoronalSlices: 512
      NumSagittalSlices: 512
    NumTransverseSlices: 88
           PlaneMapping: ["sagittal"    "coronal"    "transverse"]
               Modality: "CT"
          WindowCenters: [88×1 double]
           WindowWidths: [88×1 double]
```

Create Medical Volume Object from Voxels and Spatial Details

Create a medical volume object using a chest CT volume saved as a directory of DICOM files. The volume is part of a data set containing three CT volumes. The size of the entire data set is approximately 81 MB. Download the data set from the MathWorks website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
```

Specify the directory of DICOM files for the first CT volume in the data set.

```
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData/LungCT01");
```

Create a medical volume object for the CT volume.

```
medVol = medicalVolume(dataFolder);
```

The `Voxels` property contains the intensity values of each voxel. The `VolumeGeometry` property contains a `medicalref3d` object defining the spatial referencing for the image volume.

```
V = medVol.Voxels;
R = medVol.VolumeGeometry;
```

Modify the voxel data by applying a 3-D Gaussian filter.

```
sigma = 2;
filterV = imgaussfilt3(V,sigma);
```

Create a new `medicalVolume` object that contains the smoothed voxel values. To maintain the same spatial referencing as the original volume, specify the original `medicalref3d` object `R`.

```
medVolFiltered = medicalVolume(filterV,R);
```

Version History

Introduced in R2022b

See Also

`medicalref3d` | `medicalImage`

Topics

"Display 3-D Medical Image Data in Patient Coordinate System"
"Display Labeled Medical Image Volume in Patient Coordinates"
"Create STL Surface Model of Femur Bone for 3-D Printing"

VolumeSource

Source of 3-D medical image data for `groundTruthMedical` object

Description

A `VolumeSource` object defines the source of ground truth data for 3-D medical image volumes. Use this object to specify the volume data sources for a `groundTruthMedical` object. Each source must be a single DICOM, NIfTI, or NRRD file or multiple DICOM files comprising one 3-D volume.

Creation

When you export labels from a **Medical Image Labeler** app volume session, the `DataSource` property of the exported `groundTruthMedical` object contains a `VolumeSource` object.

To create a `VolumeSource` object programmatically, such as when programmatically creating a `groundTruthMedical` object, use the `medical.labeler.loading.VolumeSource` function.

Syntax

```
volSrc = medical.labeler.loading.VolumeSource(source)
volSource = medical.labeler.loading.VolumeSource(sourceTable)
```

Description

`volSrc = medical.labeler.loading.VolumeSource(source)` creates a `VolumeSource` object for loading the 3-D medical image data stored in the files or directory specified by `source`.

The data source for a `VolumeSource` object must be readable by a `medicalVolume` object and have a primary slice direction of `"sagittal"`, `"coronal"`, `"transverse"`, or `"oblique"`, which indicates that all slices are parallel. The `Orientation` property of the `medicalVolume` object specifies the primary slice direction. `VolumeSource` does not support volumes with a primary orientation that is `"mixed"` or `"unknown"`.

`VolumeSource` does not support volumes with more than three dimensions.

`volSource = medical.labeler.loading.VolumeSource(sourceTable)` creates a `VolumeSource` object for loading the image data specified in a table, `sourceTable`, returned by the `dicomCollection` function.

Input Arguments

`source` — Source file names
n-by-1 cell array

Source file names, specified as an *n*-by-1 cell array, where *n* is the total number of image volumes to store in the `groundTruthMedical` object.

Specify each element of `source` as one of these options:

- Name of a single DICOM, NIfTI, or NRRD file defining one image volume, specified as a string scalar.
- List of file names defining one multi-file DICOM image volume, specified as a string array.
- Name of directory containing files defining one multi-file DICOM image volume, specified as a string scalar.

Data Types: `cell`

`sourceTable` — Table of source file names
table

Table of source file names, specified as a table returned by the `dicomCollection` function. Each row in `sourceTable` must specify a valid DICOM series that contains a 3-D image volume readable by a `medicalVolume` object. The file name or list of file names for each DICOM series is stored in the `Filenames` column of the table returned by `dicomCollection`.

Data Types: `table`

Properties

`Source` — Source of ground truth data
n-by-1 cell array

This property is read-only.

Source of the ground truth data, specified as an *n*-by-1 cell array, where *n* is the total number of image volumes to store in the `groundTruthMedical` object. Each element contains the file path or file paths corresponding to one image volume, stored as a string scalar for single files or a string array for multi-file DICOM volumes.

Data Types: `cell`

Examples

Create Volume Data Source from Medical Image Files

Create a volume data source using a subset of the Medical Segmentation Decathlon data set [1 on page 1-141]. The subset of data includes two CT chest volumes and corresponding label images stored in the NIfTI file format. Download the `MedicalVolumNIfTIData.zip` file from the MathWorks® website, then unzip the file. The size of the data file is approximately 76 MB.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeNIfTIData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
dataFolder = fullfile(filepath,"MedicalVolumeNIfTIData");
```

Create a `VolumeSource` object specifying the two CT volumes.

```
filePath1 = fullfile(dataFolder,"lung_027.nii.gz");
filePath2 = fullfile(dataFolder,"lung_043.nii.gz");
source = {filePath1; filePath2};
dataSource = medical.labeler.loading.VolumeSource(source);
```

Verify that the filenames are stored in the `Source` property of the data source object.

```
dataSource.Source;
```

[1] Medical Segmentation Decathlon. "Lung." Tasks. Accessed May 10, 2018. http://medicaldecathlon.com/.

The Medical Segmentation Decathlon data set is provided under the CC-BY-SA 4.0 license. All warranties and representations are disclaimed. See the license for details.

Create Volume Data Source from DICOM Collection

Create a volume data source using a data set containing three chest CT scans. Each CT scan is saved as a directory of DICOM files. The size of the data set is approximately 81 MB. Download the data set from the MathWorks website, then unzip the folder.

```
zipFile = matlab.internal.examples.downloadSupportFile("medical","MedicalVolumeDICOMData.zip");
filepath = fileparts(zipFile);
unzip(zipFile,filepath)
dataFolder = fullfile(filepath,"MedicalVolumeDICOMData");
```

Gather the details about the DICOM files in the `dataFolder` directory into a table by using the `dicomCollection` function.

```
sourceTable = dicomCollection(dataFolder);
```

Create a volume data source specifying the files in `sourceTable`.

```
dataSource = medical.labeler.loading.VolumeSource(sourceTable);
```

Verify that each element of the `Source` property value contains a string array of filenames for one CT volume. The file names are extracted from the `Filenames` column of `sourceTable`.

```
dataSource.Source;
```

Version History

Introduced in R2022b

See Also

`groundTruthMedical` | `ImageSource`