MATITK

Usage Guide

To use the wrapper, MATLAB must be able to locate the matitk.dll*. This usually means the current working directory of MATLAB should be set to the location of matitk.dll. Copy matitk.dll to the desired location, launch MATLAB and set search path of MATLAB or change current directory to the location of the DLL.

For help information, type matitk('?') in MATLAB's command window.

To list out the filtering, segmentation and registration methods implemented in MATITK, type matitk('f'), matitk('s') and matitk('r')** respectively. The opcodes listed are used to invoke the MATITK method.

*This assumes MATITK is being run on Windows platform. This can be .so file when MATITK is being run on Linux machines.

**Alternatively, matitk ?, matitk f, matitk s, and matitk r can be typed instead.

In MATLAB, calls to MATITK methods would generally take the following format:

matitk(operationName,[parameters],[inputArray1],[inputArray2],[seed(s)Array], [Image(s)Spacing])

Legend:

- The first argument to matitk, `operationName`, specifies the opcode of the implemented ITK method to be invoked.
- The second argument to matitk, `parameters`, specifies the required parameters of the ITK method to be invoked (specified by `operationName`). To find out what parameters are required for a particular method, type `matitk(operationName);`
- The third and fourth arguments to matitk, `inputArray1` and `inputArray2`, specify the input image volume. They must be three dimensional and contain double, float, unsigned char or signed integer data type elements. In the case where a second image volume is not required for the method being invoked, provide `[]` as the fourth argument.
- The fifth argument `seedsArray` arguments specify the seed points (in MATLAB coordinate system) in the following order: $[x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_n, y_n, z_n]$. Because it is three dimensional, the number of elements in `seedsArray` should be a multiple of three. In the case where seeding is not required for the method being invoked, provide `[]` as the fifth argument.
- The last optional argument specifies the spacing of the supplied image volume. The performance of certain ITK methods may be affected by the spacing. If this argument is omitted, an isotropic spacing of `[1,1,1]` is assumed.

Example

To demonstrate the functionality of MATITK, we first load the sample built-in 3D brain mri image from MATLAB. The loaded image will automatically be stored inside the variable `D`.

```
>> load mri;
>> D=squeeze(D);
```

We can use the following commands to visualize an axial brain slice (Figure 1):

```
subplot(131);imagesc(squeeze(D(:,:,round(end/2))));axis image; colormap gray
subplot(132);imagesc(squeeze(D(:,round(end/2),:)));colormap gray
subplot(133);imagesc(squeeze(D(round(end/2),:,:)));colormap gray
set(gcf,'position',[364  628  743  320])
```

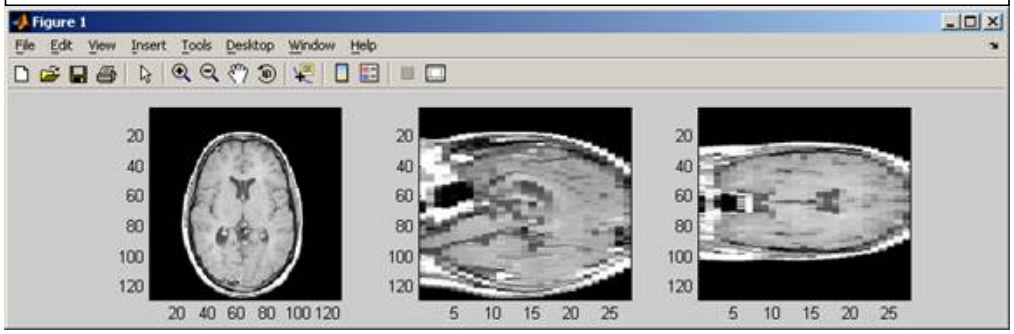

Figure 1 : (Left to right) Visualizing an axial, sagittal, and coronal brain slice of the sample image D.

The data type of the loaded image is unsigned char. As such, we would like to use the double data type version of MATITK, and we first convert the input using `double(D)`. `matitk('f')` is invoked to show the list of implemented filtering methods and the corresponding opcode.

`FCA` is the opcode for `CurvatureAnisotropicDiffusionImageFilter`

The data type of the loaded image is unsigned char. As such, we would like to use the double data type version of MATITK, and we first convert the input using `double(D)`. `matitk('f')` is invoked to show the list of implemented filtering methods and the corresponding opcode.

`FCA` is the opcode for `CurvatureAnisotropicDiffusionImageFilter`. `matitk('fca')` can be used to list out the arguments required for using

`CurvatureAnisotropicDiffusionImageFilter` (i.e. numberOfIterations, timeStep and conductance). For the example, we chose our arguments to be 5, 0.0625 and 3 respectively in this order, and supply the arguments as an array:

```
>> b=matitk('FCA',[5 0.0625 3], double(D));
Image input of type double detected, executing MATITK in double mode


FCA is being executed...
FCA has completed.
```

We can use the following commands to visualize the filtering result (Figure 2):

```
imagesc(squeeze(b(:,:,15)));
```

Figure 2 : Visualizing sample image D after “FCA” operation.

The operation is performed in double data-type mode.

We apply `ConfidenceConnectedImageFilter` to the resulting filtered image (Figure 3). The following example illustrates how the seed point (102, 82, 25) is supplied as an argument:

```
>> c= matitk('SCC',[1.4 10 255],double(b),double([]),[102 82 25]);
Image input of type double detected, executing MATITK in double mode


SCC is being executed...
SCC has completed.
```

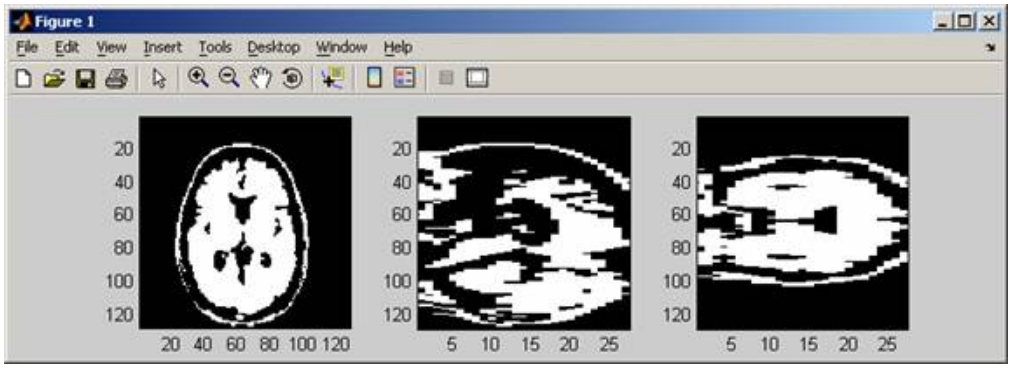

Figure 3 : Visualizing sample image D after “FCA” and “SCC” operation.

(Left to right) Axial slice, sagittal, and coronal brain slices.

The operations are performed in double data-type mode.

Instead of invoking the “double” version of `ConfidenceConnectedImageFilter`, we could first cast `b` into unsigned char first. The unsigned char version of `ConfidenceConnectedImageFilter` will be invoked as a result (Figure 4):

```
>> c=matitk('SCC',[1.4 10 255],uint8(b),uint8([]),[102 82 25]);
```

```
Image input of type unsigned char detected, executing MATITK in
unsigned char mode


SCC is being executed...

SCC has completed.
```

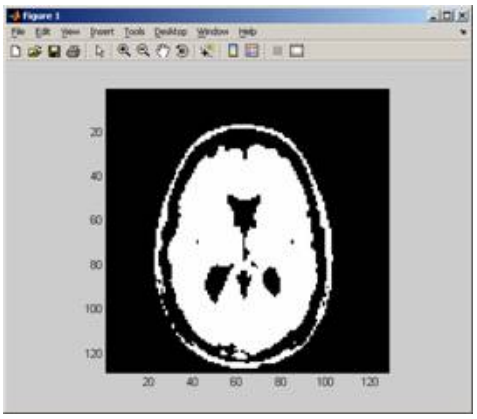

Figure 4 : Visualizing sample image D after “FCA” and “SCC” operation.
The operations are performed in unsigned char data-type mode.

Notice how casting can affect the final result.

For another illustration, we apply `GradientMagnitudeImageFilter` to the original image `D` of type unsigned char (Figure 5). Notice the unsigned char version of ITK method will be used:

```
>> G=matitk('FGM',[],D);

Image input of type unsigned char detected, executing MATITK in
unsigned char mode


FGM is being executed...

FGM has completed.
```

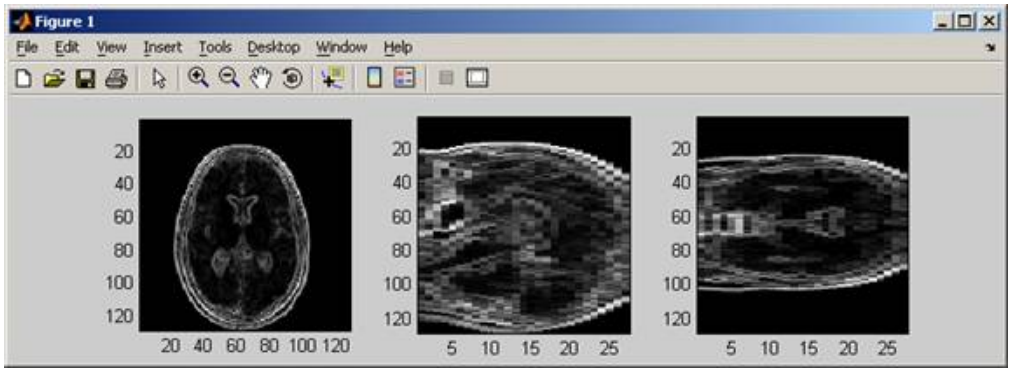

Figure 5 : Visualizing sample image D after “FGM” operation.
(Left to right) Axial, sagittal, and coronal brain slices.

The operations are performed in double data-type mode.

MATITK also supports receiving multiple outputs from ITK methods. The resulting images of invoking `OtsuMultipleThresholdImageFilter` will be stored in variables `O1`, `O2`, and `O3` (Figure 6):

```
[O1,O2,O3]=matitk ('fomt',[3,128],D);
Image input of type unsigned char detected, executing MATITK in
unsigned char mode


fomt is being executed...
fomt has completed.
```

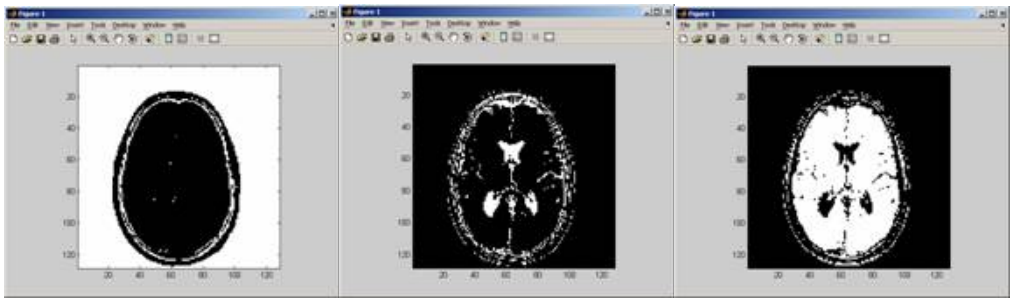

Figure 6 : Visualizing the three outputs of “FOMT” operation.

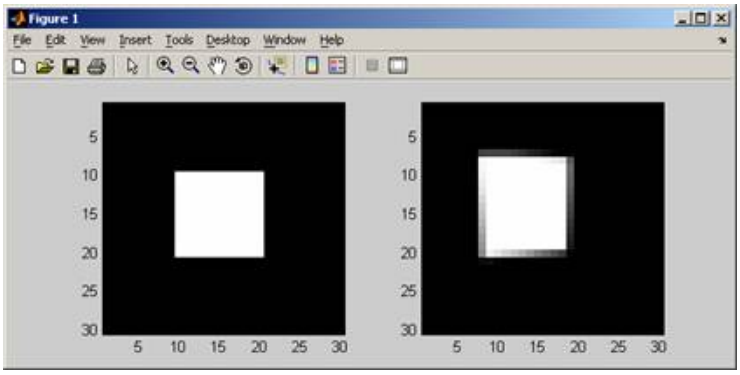

Figure 7 : A slice of the cube before (left) and after (right) applying Thin Plate Spline Warping (“RTPS”) operation.

To illustrate how warping can be applied using MATITK, consider warping a cube according to the following landmarks:

Source			Target		
x	y	z	x	y	z
10	10	10	12	12	13
10	10	20	11	13	22
10	20	10	12	23	11
10	20	20	12	21	21
20	10	10	20	11	12
20	10	20	22	10	23
20	20	10	20	21	11
20	20	20	20	23	21

The source landmarks correspond to the edges of the cube, and the target landmarks correspond to randomly generated points close to the original edges.

```
A=zeros(30,30,30);

A(10:20,10:20,10:20)=1;

output=matitk ('rtps',[],A,A,[10 10 10 12 12 13 10 10 20 11 13 22 10
20 10 12 23 11 10 20 20 12 21 21 20 10 10 20 11 12 20 10 20 22 10 23
20 20 10 20 21 11 20 20 20 20 23 21]);

Image input of type double detected, executing MATITK in double mode


rtps is being executed...

rtps has completed.
```

The result of execution is shown in (Figure 7).

Available Opcodes

As of version 2.4.04 released on Aug 24 2006, the following table lists the operations are available.

Opcode	Method
FAAB	AntiAliasBinaryImageFilter
FBB	BinomialBlurImageFilter
FBD	BinaryDilateFilter
FBE	BinaryErodeFilter
FBL	BilateralFilter
FBT	BinaryThresholdImageFilter
FCA	CurvatureAnsioFilter
FCF	CurvatureFlowFilter
FD	DerivativeImageFilter
FDG	DiscreteGaussianImageFilter
FDM	DanielssonDistanceMapImageFilter
FDMV	DanielssonDistanceMapImageFilterGetVoronoiMap
FF	FlipImageFilter
FFFT	FFTImageFilter
FGA	GaussianFilter
FGAD	GradientAnisotropicDiffusionImageFilter
FGM	GradientMagnitudeFilter
FGMRG	GradientMagnitudeRecursiveGaussianImageFilter
FGMS	GradientMagnitudeWithSmoothingFilter
FLS	LaplacianRecursiveGaussianImageFilter
FMEAN	MeanImageFilter
FMEDIAN	MedianImageFilter
FMMCF	MinMaxCurvatureFlowFilter

FOMT	OtsuMultipleThresholdImageFilter
FSN	SigmoidNonlinearMappingFilter
FVBIH	VotingBinaryIterativeHoleFillingImageFilter
FVMI	VesselnessMeasureImageFilter
FRG	RecursiveGaussianImageFilter

The following segmentation functions are implemented:

Opcode	**Method**
SCC	ConfidenceConnectedSegmentation
SCSS	CellularSegmentationSegmentation(Debug)
SCT	ConnectedThresholdSegmentation
SFM	FastMarchSegmentation
SGAC	GeodesicActiveContourLevelSetSegmentation
SIC	IsolatedConnectedSegmentation
SLLS	LaplacianLevelSetLevelSetSegmentation
SNC	NeighbourhoodConnectedSegmentation
SOT	OtsuThresholdSegmentation
SSDLS	ShapeDetectionLevelSetFilter
SWS	WatershedSegmentation

The following registration functions are implemented:

Opcode	**Method**
RD	registerDemon
RTPS	registerThinPlateSpline

For documentation on each method, simply types its corresponding opcode in MATITK prompt.

e.g. `matitk ('rtps')` would give:

```
rtps is being executed...

**********Begin description of registerThinPlateSpline(rtps)**********

 ThinPlateSplineKernelTransform
 This class defines the thin plate spline (TPS) transformation.
 It is implemented in as straightforward a manner as possible from
 the IEEE TMI paper by Davis, Khotanzad, Flamig, and Harms,
 Vol. 16 No. 3 June 1997

  Transforms
**************************End description**************************
```

You must supply parameters for this function in an array, with the elements in this order:

0 parameters must be supplied. You supplied 0.

??? Correct number of parameters must be supplied. At least one image volume has to be supplied.