MATITK:
Extending MATLAB with ITK

Vincent Chu (vwchu@sfu.ca)
Ghassan Hamarneh (hamarneh@cs.sfu.ca)
Simon Fraser University, British Columbia, Canada.

I. DESCRIPTION

To facilitate the analysis of medical image data in research environment, MATITK is developed to allow ITK algorithms to be called in MATLAB. ITK is a powerful open-source image analysis toolkit, but it requires the knowledge of C++ to use it. With the help of MATITK, researchers familiar with MATLAB can harness the power of ITK without learning C++ and worrying about low-level programming issues. A common set of C++ classes have also been produced to allow future ITK methods to be added to MATITK – therefore callable in MATLAB – without the bothersome translation between MATLAB and ITK.

II. HISTORY

May 2005.	First implementation of MATITK was showcased in BC Medical Technology Research Showcase held by BC Medical Technology Industry Association [1].
July 2005.	Technical details of MATITK based on ITK 1.8 were submitted to SPIE Medical Imaging 2006 (accepted for publication in Oct 2006) [2]
Dec 2005.	New methods from ITK 2.4 are added. Other data types besides double are now supported. MATITK is available for download for the general public[1].

III. IMPLEMENTATION

For details of MATITK's software design and architecture, the reader is referred to [1,2], (see section VIII Supplementary Material). The reader who is interested in *using* MATITK and not *building* it, should skip to section IV Usage. Following is an overview of the implementation of MATITK and the steps required for building it.

A set of C++ reusable template classes have been written to translate between MATLAB's and ITK's representations of images, connect the pipeline, and return the results or error messages appropriately.

---

[1] MATITK can be obtained at http://mial.fas.sfu.ca/researchProject.php?s=308

Based on the ITK example files, a Perl script has been created to automatically generate C++ code that can be added into MATITK with little human intervention. After compilation, the included ITK methods can be called in MATLAB.

The current version[2] (MATITK 2.4.02) accepts 3D images of types double, single (float), unsigned char (uint8) and integer (int32), and the included methods are based on the example files from ITK version 2.4. For this submission, the source code is compiled on Windows with Visual Studio 2003 and MATLAB 7.0 to generate matitk.dll.

To build MATITK, the following steps are suggested:

1. Type `mex -setup` in MATLAB to configure the default options file for your compiler. This assumes that a MATLAB compatible compiler is present on your system such as VS 6 and VS 2003 on Windows.

2. Modify the automatically generated options file (mexopts.bat on Windows, with its path shown in the last line of `mex -setup` execution) to include the header files paths and the linking paths of ITK. Alternatively, set the Windows environment variables `%ITKSRC%` and `%ITKBIN%` to ITK source and binary directories respectively, and replace the generated mexopts.bat with the one included in the submission.

3. In MATLAB, change the current working directory to the directory containing all the MATITK source files (.cxx, .h and .inl files) included in this submission. Type `mex matitk.cxx` to build matitk.dll.

IV. USAGE

To use the wrapper, MATLAB must be able to locate the `matitk.dll`[3]. This usually means the current working directory of MATLAB should be set to the location of `matitk.dll`. Copy matitk.dll (see section VIII Supplementary Material) 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')`[4] respectively. The opcodes listed are used to invoke the MATITK method.

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

```
matitk(operationName,[parameters],[inputArray1],[inputArray2],[seed(s)Array],[I
mage(s)Spacing])
```

---

[2] For MATITK, the first two major version numbers will follow the version number of ITK that it is based on. The last number is the stable minor build number. Unstable versions will be labelled as betas.
[3] This assumes MATITK is being run on Windows platform. This can be .so file when MATITK is being run on Linux machines.
[4] Alternatively, `matitk ?`, `matitk f`, `matitk s`, and `matitk r` can be typed instead.

Legend:

1. The first argument to `matitk`, `operationName`, specifies the opcode of the implemented ITK method to be invoked.

2. 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);`

3. 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.

4. 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.

5. 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.

V. 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])
```

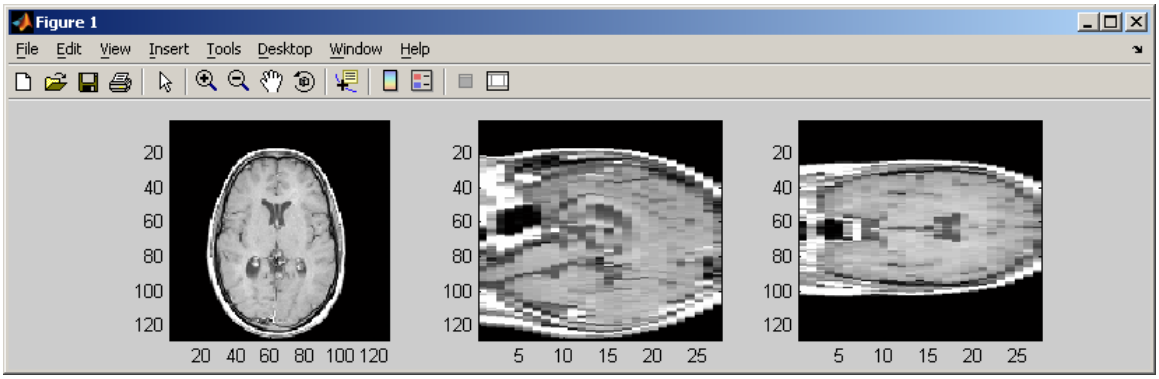

Figure 1: (Left to right) Visualizing an axial, sagittal, and coronal brain slice of the sample image D.

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)));
```

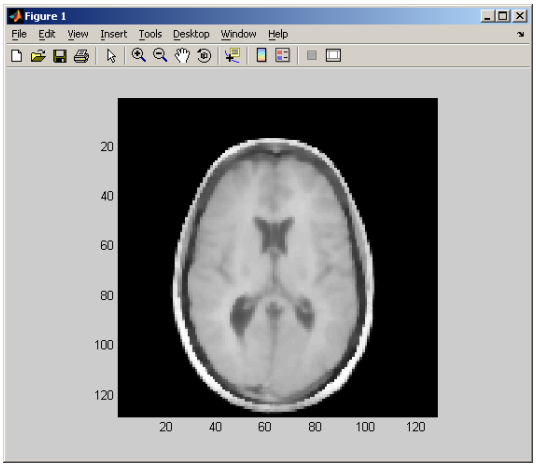

Figure 2: Visualizing sample image D after "FCA" operation.
The operation is performed in double data-type mode.

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.
```

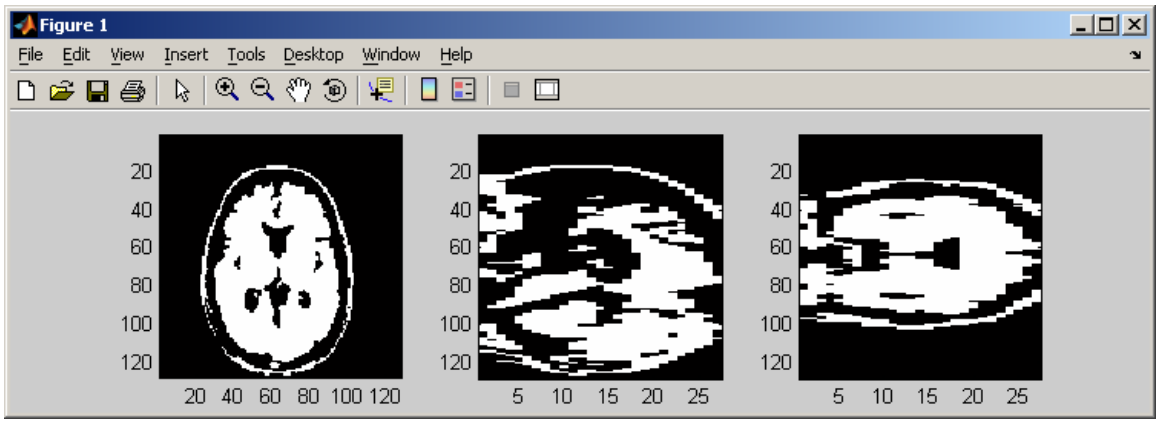

Figure 3: Visualizing sample image D after "FCA" and "SCC" operation.
(Left to right) Axial slice, sagittal, and coronal brain slices.
The operations are performed in double data-type mode.

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.
```

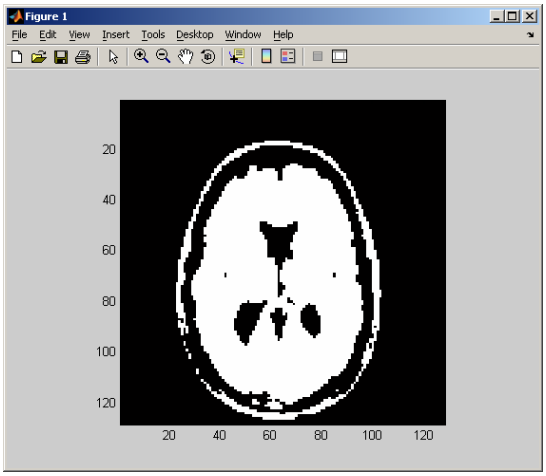

Figure 4: Visualizing sample image D after "FCA" and "SCC" operation. The operations are performed in unsigned char data-type mode.

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.
```

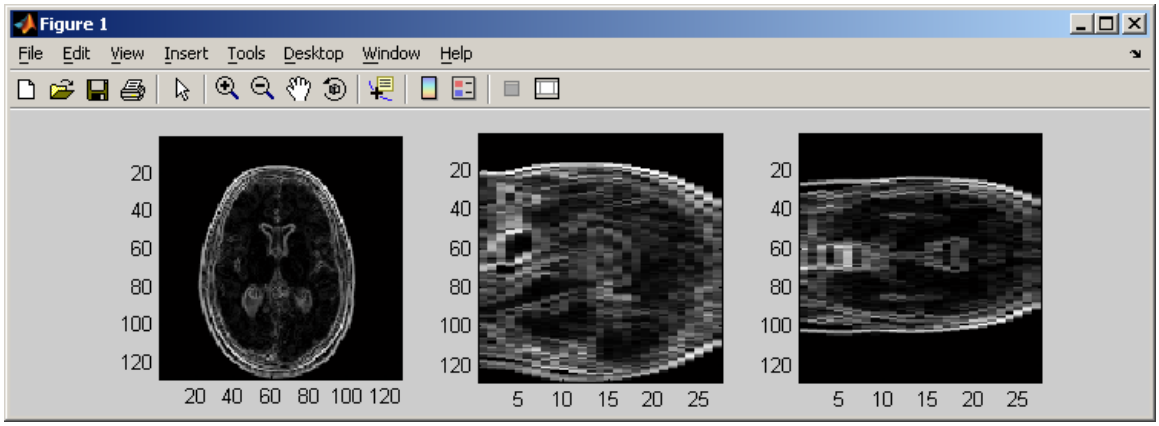

Figure 5: Visualizing sample image D after "FGM" operation.
(Left to right) Axial, sagittal, and coronal brain slices.
The operations are performed in double data-type mode.

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.
```

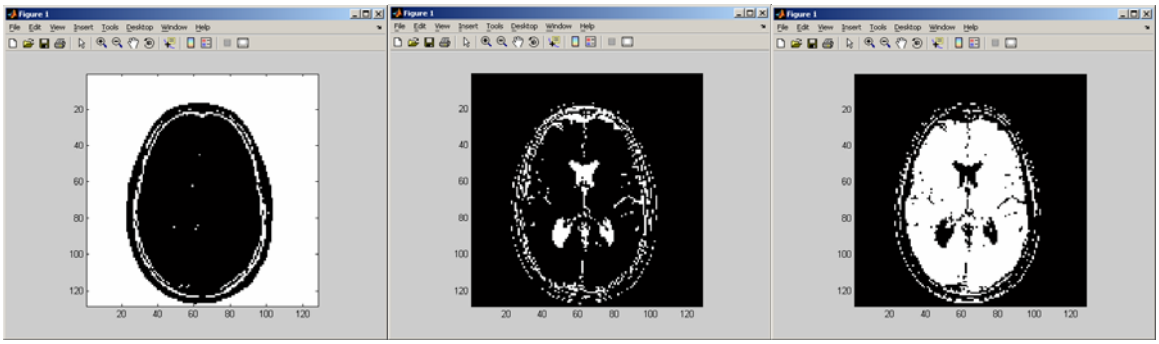

Figure 6: Visualizing the three outputs of "FOMT" operation.

VI. FUTURE WORK

We plan to continue updating MATITK with new and enhanced methods of future ITK releases.

Future work also includes the addition of methods related to deformable meshes, more image registration support, ability to specify pipelines of filters from within MATLAB, and options to return spatial transformations from registration or surfaces from deformable model segmentation.

The ability to provide physical dimensions of image data is inherently unsupported in MATLAB. Voxel spacing can currently be passed as an additional argument to MATITK. Other meta-data such as origin location and direction cosines will be incorporated as well.

The only MATITK binary distribution we provide is for Windows XP, Linux support will be added in the near future.

VII. FREQUENTLY ASKED QUESTIONS

Q1. When I type matitk in MATLAB, I am receiving "??? Invalid MEX-file 'matitk.dll': The specified module could not be found."

A1. It has come to our attention that the original distribution (v.2.4.01 Dec 28 2005) had a redundant dependency on a Windows DLL that not every Windows machine has. We have now removed the dependency. Also, we repackage the distribution and it now includes the Windows DLLs that matitk.dll depends on. If you are receiving this error, please re-download the latest binary distribution.

Q2. Do I need to install ITK first? What does it mean when you say MATITK is based on ITK 2.4?

A2. No, you do not need to have or install ITK, Visual Studio or any developer tools to use MATITK; all you need is MATLAB. MATITK is based on ITK 2.4 means that the algorithms that are made available through MATITK are derived from ITK version 2.4. In other words, with MATITK, you get to access many of the algorithms available in ITK 2.4.

VIII. SUPPLEMENTARY MATERIAL

- mexopts.bat : MATLAB's compiler options file modified to include necessary paths
- matitk.dll: MATITK built into a shared library on Windows
- cxx, h, and inl files: C++ source code files

- matitkcode.pl: Perl Script for generating MATITK source code from ITK examples
- PDF files: Related poster and paper [1, 2] with further details about MATITK.

IX. REFERENCES

[1] V. Chu, G. Hamarneh, "MATITK: MATLAB-ITK Interface for Medical Image Processing", 1st Annual Medical Technology Research Showcase, Vancouver, BC, 2005.

Download: http://www.cs.sfu.ca/~hamarneh/ecopy/medical_showcase2005a.pdf

[2] V. Chu, G. Hamarneh, " MATLAB-ITK Interface for Medical Image Filtering, Segmentation, and Registration", Proceedings of SPIE Medical Imaging: Image Processing 2006 (accepted), Paper/Code Number: 6144-135.

Download (not final paper): http://www.cs.sfu.ca/~hamarneh/ecopy/spiemi2006a.pdf