ERC | CISST

# MATLAB interface for *cisst* libraries

Group 16

Zachary Zhou

Mentor: Anton Deguet

# Project Summary

- The cisst package is a collection of libraries designed to ease the development of computer assisted intervention systems
- Currently the cisst libraries are available only in the C/C++ programming language
- Wrap cisst in MATLAB to allow ease of access/direct manipulation of collected data

# MATLAB-ITK Interface for Medical Image Filtering, Segmentation, and Registration

Vincent Chu, Ghassan Hamarneh

Vincent Chu, Ghassan Hamarneh “MATLAB-ITK Interface for Medical Image Filtering, Segmentation, and Registration”.
<www.cs.sfu.ca/~hamarneh/ecopy/medical_showcase2005a.pdf>

* All unreferenced images are taken directly from this paper

# Stated Goals

- wrapper which will allow a greater number of researchers access to the ITK libraries
  - As ITK is limited those with a good understanding of C programming
- allow for data from ITK to be reduced on MATLAB without file IO
  - expected to result in a speed up of several orders in magnitude

# Motivation

- C is a complex language compared to MATLAB
  - Requires memory management and use of pointers
  - Understanding of fundamental programming concepts
  - Have to work with compiler
- Speed is lose when analyzing data in MATLAB
  - Must output data via file IO, accessing hard disk reduces run speed

# Background

- MATLAB (MATrix LABoratory)
  - Commonly used for matrix manipulation, numerical analysis
  - Growing popularity with scientific researchers and biomedical engineers
- ITK (Insight ToolKit)
  - C++ library
  - toolkit contains various filtering, segmentation and registration algorithms designed for medical image analysis
  - Provides algorithms which run as superior to those possible in MATLAB

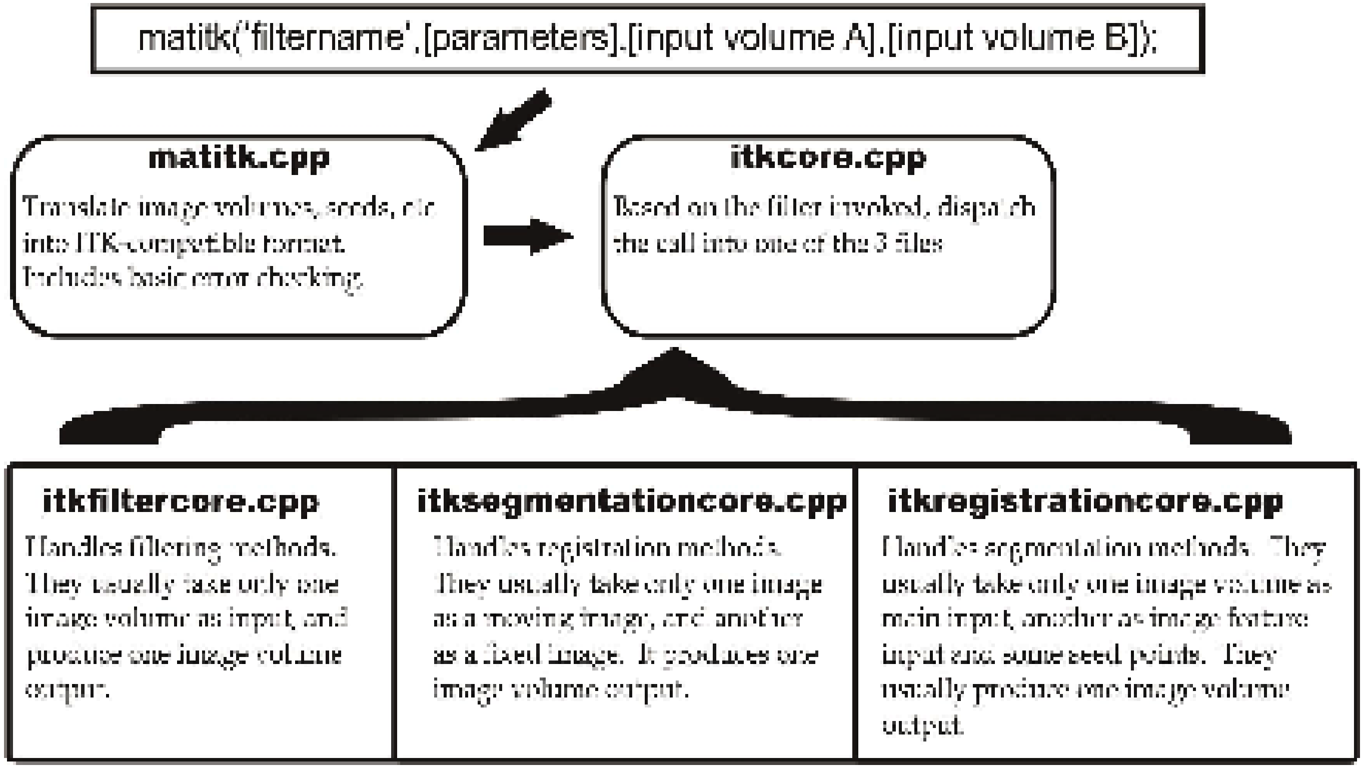
# Background

- MEX files
  - Compiled from C source code by MATLAB compiler
  - Allows for manipulation of mxArrays (MATLAB arrays)
  - One point of entry (mexfunction)

# Relevance

- MATLAB wrapper for a C library
- Goal is to allow for ease of access by wrapping in a simpler to understand language
  - Additional goal is to allow for direct reduction of collected data

# Design Architecture



matitk('filtername',[parameters].[input volume A],[input volume B]);

**matitk.cpp**

Translate image volumes, seeds, etc into ITK-compatible format. Includes basic error checking.

**itkcore.cpp**

Based on the filter invoked, dispatch the call into one of the 3 files

**itkfiltercore.cpp**

Handles filtering methods. They usually take only one image volume as input, and produce one image volume output.

**itksegmentationcore.cpp**

Handles registration methods. They usually take only one image as a moving image, and another as a fixed image. It produces one image volume output.

**itkregistrationcore.cpp**

Handles segmentation methods. They usually take only one image volume as main input, another as image feature input and some seed points. They usually produce one image volume output.

**Figure 1.** MATITK Execution Flowchart.

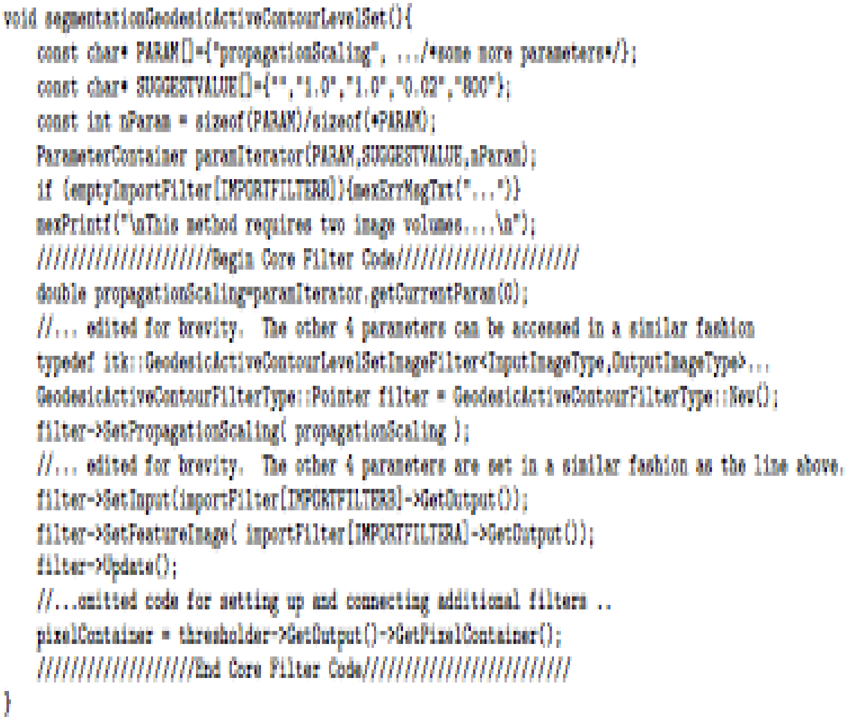
# Design Architecture

- Utilizes CMake to compile standard ITK files into MEX files. Automatically generates mexopt.bat file which contains header and library paths for ITK files.
- Matitk() is the only function which will be called from the MATLAB console
- Matitk will then parce the input string values and pass results to itkcore.cpp

# Design Architecture

- itkcore calls:
  - – Itkfiltercore.cpp
  - – Itksegmentationcore.cpp
  - – Itkregistrationcore.cpp

# Filtering, Segmentation, Registration



```
void segmentationGeodesicActiveContourLevelSet(){
    const char* PARAM[]={"propagationScaling", .../*some more parameters*/};
    const char* SUGGESTVALUE[]={"","1.0","1.0","0.02","800"};
    const int nParam = sizeof(PARAM)/sizeof(*PARAM);
    ParameterContainer paramIterator(PARAM,SUGGESTVALUE,nParam);
    if (emptyImportFilter[IMPORTFILTERB]){mexErrMsgTxt("...")}
    mexPrintf("\nThis method requires two image volumes....\n");
    ////////////////////Begin Core Filter Code////////////////////
    double propagationScaling=paramIterator.getCurrentParam(0);
    //... edited for brevity.  The other 4 parameters can be accessed in a similar fashion
    typedef itk::GeodesicActiveContourLevelSetImageFilter<InputImageType,OutputImageType>...
    GeodesicActiveContourFilterType::Pointer filter = GeodesicActiveContourFilterType::New();
    filter->SetPropagationScaling( propagationScaling );
    //... edited for brevity.  The other 4 parameters are set in a similar fashion as the line above.
    filter->SetInput(importFilter[IMPORTFILTERB]->GetOutput());
    filter->SetFeatureImage( importFilter[IMPORTFILTERA]->GetOutput());
    filter->Update();
    //...omitted code for setting up and connecting additional filters ..
    pixelContainer = thresholder->GetOutput()->GetPixelContainer();
    //////////////////End Core Filter Code//////////////////////
}
```

- First 3 lines defines parameters
- Other lines call ITK functions which modify variables stored in itkcore

# Filtering, Segmentation, Registration

- The first letter of MATLAB function call dictates which process is being called:
  - Ex: calls starting with letter ‘r’ will result in a call to itkregistrationcore
  - Calls starting with letter ‘f’ will result in call to itkfiltercore
  - Etc.
- There is no return type, itkcore contains variables which are modified by the routines and then accesed from matitk

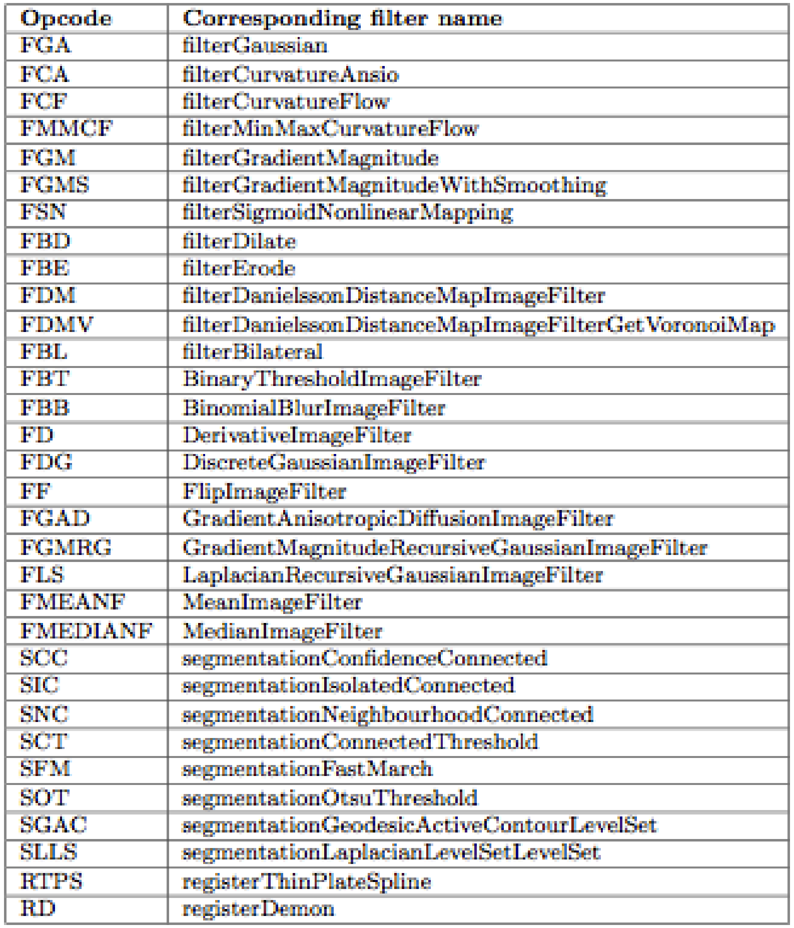
# Automated Generation of Filtering Script

- PERL script
    - Generates C source code
    - Follows pseudocode for other filtering fucntions
    - May result in error

# Using the Wrapper

- Load compiled MEX library (matidk.dll) into current MATLAB directory
- Following call is made in MATLAB console
  - matitk(operationName,[parameters],[inputArray1] ,[inputArray2],[seed(s)Array],[Image(s)Spacing]);

# Results



| Opcode | Corresponding filter name |
|---|---|
| FGA | filterGaussian |
| FCA | filterCurvatureAnsio |
| FCF | filterCurvatureFlow |
| FMMCF | filterMinMaxCurvatureFlow |
| FGM | filterGradientMagnitude |
| FGMS | filterGradientMagnitudeWithSmoothing |
| FSN | filterSigmoidNonlinearMapping |
| FBD | filterDilate |
| FBE | filterErode |
| FDM | filterDanielssonDistanceMapImageFilter |
| FDMV | filterDanielssonDistanceMapImageFilterGetVoronoiMap |
| FBL | filterBilateral |
| FBT | BinaryThresholdImageFilter |
| FBB | BinomialBlurImageFilter |
| FD | DerivativeImageFilter |
| FDG | DiscreteGaussianImageFilter |
| FF | FlipImageFilter |
| FGAD | GradientAnisotropicDiffusionImageFilter |
| FGMRG | GradientMagnitudeRecursiveGaussianImageFilter |
| FLS | LaplacianRecursiveGaussianImageFilter |
| FMEANF | MeanImageFilter |
| FMEDIANF | MedianImageFilter |
| SCC | segmentationConfidenceConnected |
| SIC | segmentationIsolatedConnected |
| SNC | segmentationNeighbourhoodConnected |
| SCT | segmentationConnectedThreshold |
| SFM | segmentationFastMarch |
| SOT | segmentationOtsuThreshold |
| SGAC | segmentationGeodesicActiveContourLevelSet |
| SLLS | segmentationLaplacianLevelSetLevelSet |
| RTPS | registerThinPlateSpline |
| RD | registerDemon |

**Table 1.** MATITK available opcodes and the corresponding opnames.

# Personal Critique

- Good example of how to wrap a C library in MATLAB
- Provides solution to single entry point weakness of MEX files
- Offers another potential solution
  - Generate MEX files for each ITK function
- Provides wrapper for dynamically adding new filtering functions to library

# Personal Critique

- Does not analyze one of stated goals
  - No comparison of run speed utilizing the wrapper vs using File IO to analyze data collected by C
- MEX function call is very awkward to use
  - matitk(operationName,[parameters],[inputArray1] ,[inputArray2],[seed(s)Array],[Image(s)Spacing]);
  - componentA=componentAClass
  - Result=componentA.function();

# Personal Critique

- Not usable for the cisst wrapper
  - Cisst is object oriented
  - Not possible to simply use string manipulation in creation of all cisst library objects
- Design requires a lot of String manipulation
  - Would be a more elegant solution to not try to pipe everything through one MEX function

# Personal Critique

- Auto generator for filters is a PERL script
  - Generates new C source code
  - Can possibly error
  - Needs to generate new code for every additional function to add in

# Questions?

Thank you