An Insight Software Consortium Tutorial: Medical Image Analysis with ITK and Related Open-Source Software

http://www.InsightSoftwareConsortium.org

This course introduces attendees to select open-source efforts in the field of medical image analysis. Opportunities for users and developers are presented.

The course particularly focuses on the open-source Insight Toolkit (ITK) for medical image segmentation and registration. The course describes the procedure for downloading and installing the toolkit and covers the use of its data representation and filtering classes. Attendees are shown how ITK can be used in their research, rapid prototyping, and application development.

LEARNING OUTCOMES

After completing this course, attendees will be able to:

- contribute to and benefit from open-source software for medical image analysis
- download and install the ITK toolkit
- start their own software project based on ITK
- design and construct an image processing pipeline
- combine ITK filters for medical image segmentation
- combine ITK components for medical image registration

INTENDED AUDIENCE

This course is intended for anyone involved in medical image analysis. In particular it targets graduate students, researchers and professionals in the areas of computer science and medicine. Attendees should have an intermediate level on object oriented programming with C++ and must be familiar with the basics of medical image processing and analysis.

COURSE LEVEL

Intermediate

COURSE LENGTH

Full-day (6.5 Hours)

COURSE OUTLINE

1) The Insight Software Consortium: contributing and using open-source
2) The architecture and installation of the Insight Toolkit
3) Segmentation methods of the Insight Toolkit
4) Registration methods of the Insight Toolkit
5) Image IO using the Insight Toolkit
6) The Image-Guided Surgery Toolkit: architecture overview
7) Using the Insight Toolkit with TK/TCL
8) Applications of the Insight Toolkit:
   VTK/ITK, The Virtual Soldier, Industrial Research and NA-MIC's Slicer

CONTRIBUTORS

Many people contributed to the development of these presentations. The ISC is grateful to those peoples' support of open source software and education. Major contributors include the following:

Stephen Aylward is Chief Medical Scientist at Kitware, Inc. He joined Kitware in December 2005, after serving as a tenured Associate Professor of Radiology, Computer Science, and Surgery at the University of North Carolina at Chapel Hill. At Kitware, Stephen is working with leaders in the open-source movement on research and development for clinical image analysis software. Dr.

Aylward is also President of the Insight Software Consortium and an associate editor of IEEE Transactions on Medical Imaging. He has computer science degrees from Purdue (BS), Georgia Institute of Technology (MS), and UNC (PhD).

Luis Ibanez is a Research Engineer at Kitware, Inc. He received a BSc in Physics in 1989 and a MSc in Optics in 1994 from the Universidad Industrial de Santander (Bucaramanga, Colombia). He received a PhD in 2000 from the Universite de Rennes I (Rennes, France). From 1999 to 2001 he was Research Assistant Professor in the Division of Neurosurgery at the University of North Carolina at Chapel Hill. His current research interest the application of genomics paradigms to computation.

Josh Cates is a member of the research staff of the Scientific Computing and Imaging Institute at the University of Utah's School of Computer Science. He received his BS degree in Biology in 1995 and MS in Computer Science in 1999 from the University of Tennessee. His interests include software engineering and problems in computer vision, including image processing (differential geometry and p.d.e.-based methods), image segmentation, and computed tomography.

Lydia Ng is a member of the Informatics group at the Allen Institute for Brain Science. She received a BE in Electrical Engineering and a BSc in Computer Science in 1994 from The University of New South Wales (Sydney, Australia). She received a Ph.D. in 2000 from Macquarie University (Sydney, Australia). Her research interests include: image registration, PDE based image processing methods, motion estimation and development of software for medical image processing and analysis.

Julien Jomier is a research member at Kitware, Inc. He joined Kitware in December, 2005 after being a research faculty member in the Computer-Aided Diagnosis and Display Laboratory at the University of North Carolina at Chapel Hill. He has extensive C++ programming and software design experience as well as medical image analysis and data handling expertise. He is currently continuing his development of ITK's spatialObject library and his freely available spatial object viewer library. Julien is also contributing to the development of the Insight Journal, the Image Guided Surgery Toolkit, and numerous other open-source efforts.

William Lorensen is a Graphics Engineer in the Electronic Systems Laboratory at GE's Corporate Research and Development Center in Schenectady, NY. He has over 35 years of experience in computer graphics and software engineering. He is currently working on algorithms for 3D medical graphics and scientific visualization.

The Insight Software Consortium

http://www.InsightSoftwareConsortium.org

Educating users and developers of open-source, medical image analysis software

Outline

- History of the ISC
- Charter of the ISC
- Contributing to and benefiting from open-source…

History

- Initiated by the original developers of the National Library of Medicine's Insight Toolkit
  - Drs. Terry Yoo, Bill Lorensen, Will Schroeder, …
  - 2003 Developers' Meeting
- Motivation
  - Near term:
    - Hold the copyright of ITK
  - Long term:
    - Charter of the ISC…

Charter

ISC incorporated in New York State in 2005.
A non-profit, educational consortium.

The purpose of this Consortium is to

1. support the maintenance of,

2. guide the development of, and

3. promote the use of

open-source, medical image analysis software, data, and publications

for teaching, research, and commercial endeavors

What the ISC means to you…

- The ISC provides proven, documented, open-source software environments and data for developers and users involved in medical image analysis research.
- The haunting past of open-source: "You get what you pay for…"
  - Open-source is poorly written code
  - Open-source is inefficient
  - Open-source is poorly documented
  - Users: My task is "special." …I can do "it" better myself…
  - Developers: Users are "annoying." …weird platforms, errors, emails…

Process of ISC Open-Source

Ultimate goal: ISC certification

Steps:

1. Use Development Environment
2. Contribute to Insight Software Journal
3. Participate in Community Review
4. Seek Protection
5. Support Distribution and Education
6. Benefit from Community support

1. Development Environment

- ISC Certified
  - ITK
  - IGSTK
- Companion projects
  - CMake
  - Dart / Dashboards
  - Cable / CSwig
  - DCMTK
  - FLTK
  - Slicer
  - VTK

Safety of development environment

Policies of the ISC:

- Programming style
- Backward compatibility
- Intellectual property
- Documentation
- Open-source

- Tested
- Maintained
- Cross-platform

I can do “it” better myself…

Segmentation

- Statistical, Fuzzy Logic, Markov Random Fields, Mixture Modeling, Parzen Windows, Nearest Neighbor, K-Means, …
- Level Set, Finite Element, Region Growing, Hybrid, Watershed, Connected Components, Parameterized Models, …

Registration

- Rigid, Similarity, Affine, Vector Field, Hierarchical, Quaternion, Versor, Parameterized Deformation, Euler, 3D/2D, …
- Mutual Information, Normalized Correlation, Demons, Mean Squared, Landmark, …

- Mayo Clinic
- Harvard / Brigham and Women’s Hospital
- Cognita, Inc.
- Imperial and King’s College London
- University of Iowa
- Georgetown University
- Carnegie Mellon University
- GE Research / Harvard
- Kitware, Inc.
- Insightful / UPenn
- UNC / UPitt
- UPenn / Columbia
- University of Utah
- + Over 70 machine configurations recompile ITK every night

- 20,000+ downloads
- 30+ countries

2. Insight Software Journal
http://www.insight-journal.com

- Initiated by Dr. Luis Ibanez (Kitware) and developed by Julien Jomier, Zack Galbreath, …
- Open, online publication
  - Upload, download, and review
  - By-attribution distribution license – not copyright transfer
- Components of an open-science submission:
  1. Paper that describes and demonstrates the method
  2. Source code that implements the method
  3. The data needed to generate the demonstration

3. Community Review

Insight Journal
- Review by peers
- Review by automated compilation process

Criterion
- Code
- Documentation
- Utility
- Adherence to policy

Iterative and only a component
- Revisions, respond to reviewers
- Cite in technical journals

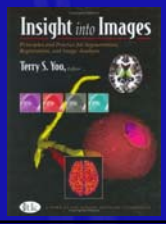
4. Protection

Once consensus builds for incorporation into an ISC project…

- Transfer copyright to the ISC
  - A level of protection for developers
  - Ensure the open-source status of the software for users
    - Right to modify and distribute
    - Teaching, research, and commercial
- Challenges
  - Acquiring copyright from developers
  - Copyright infringement (known and unknown)
  - Patented code (known and unknown)

5. Distribution and Education

- Integration into ISC development environment
- Tutorials at international conferences:
  SPIE Medical Imaging 2003-2005, MICCAI 2003-2004, IEEE Visualization 2003-2005, SIGGRAPH 2005
- Freely available courseware:
  Developed collaboratively by CMU, UPitt, Rutgers, and Kitware Inc.
- Examples, Documentation, Users Lists
- Books:



6. Community Support

www.InsightSoftwareConsortium.org

- Web portal
- Certification - not sourceforge
- Dedicated to medical image analysis
  - Jobs, News, Links
- Benefit from ISC certification and development environment…education…
- Needs
  - Contributions: Comments, Commented Code, …
  - Long term dedication
    - Good of the community vs. dissertation / proposal / …

Community Limitations

- FDA
  - FDA does not approve software libraries
  - FDA only approves applications
- GNU General Public License (GPL) software has limited distribution
  - Complicates corporate acceptance of and contribution to the open-source initiative
  - Consider instead the BSD or MIT license

Open-Science

-- Dr. Terry S. Yoo, NLM

Open Source + Open Data = Open Science

- Share data
- Share code
- Understanding methods
  - Assumptions, parameters, & outputs
- Comparison of methods
- Extension of methods

- MICCAI 2006, Copenhagen: Open-Source Workshop

Board of the ISC

- Stephen Aylward
- Joshua Cates
- Luis Ibanez
- Bill Lorensen
- Dimitris Metaxas
- Jim Miller
- Lydia Ng
- Will Schroeder
- Ross Whitaker

What is ITK

- Image Processing
- Segmentation
- Registration
- No Graphical User Interface (GUI)
- No Visualization

ITK Sponsors

The National Institute for Dental and Craniofacial Research

The National Science Foundation

The National Institute of Neurological Disorders and Stroke

ITK Developers

ITK Developers

GE CRD
Bill Lorensen

Insightful
Lydia Ng

U Penn
Dimitris Metaxas

Harvard BWH*
Ron Kikinis

U Penn*
Jim Gee

Columbia U.*
Celina Imielinska

Kitware
Will Schroeder

UNC-CH
Stephen Aylward

U Tennessee
Ross Whitaker

U Pittsburgh*
George Stetten

U Utah*
Ross Whitaker

* indicates a subcontractor.

ITK by the Numbers

- March 2000
  - First code check-in
- 1300
  - # of nightly builds
- 1062
  - tests run nightly
- 41
  - # of platforms (software + hardware)
- 700
  - # of classes
- 1600
  - # of files with code

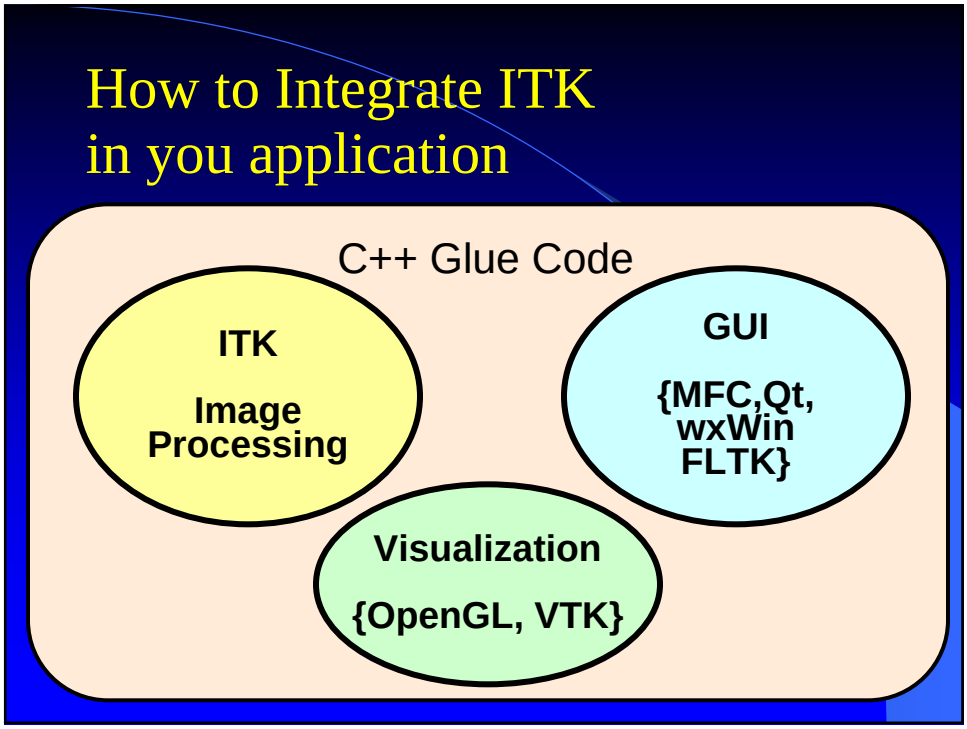
ITK by the Numbers

- 400K
  - # of lines of code
- 100K
  - # of lines of test code
- 35K
  - # of lines of examples
- 150K
  - # of lines of Applications
- 240
  - weekly t-cons
- 50
  - unique developers

ITK by the Numbers

- 1032
  - # of users subscribed to the mailing-list
- 400
  - # of emails posted monthly to the users-list
- 819
  - # of pages in the Software Guide PDF document
- 1800
  - # of monthly hits to the URL of the Software Guide PDF
- 1900
  - # of monthly hits to the URL of the Tutorial PDF
- 2400
  - # of monthly hits to the source code files (.zip + .tar.gz)

How to Integrate ITK in you application



What do I need ?

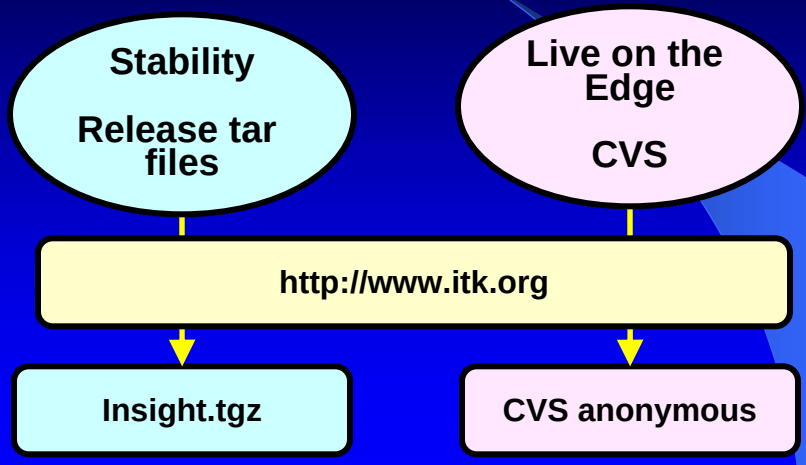
C++ Compiler

gcc 2.95 – 4.0
Visual C++ 6.0
Visual C++ 7.0
Visual C++ 7.1
Visual C++ 8.0
Intel 7.1
Intel 8.0
IRIX CC
Borland 5.5
Mac - gcc

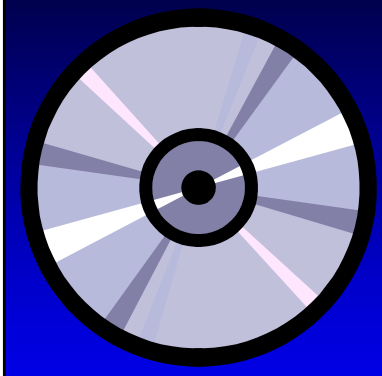
CMake

www.cmake.org

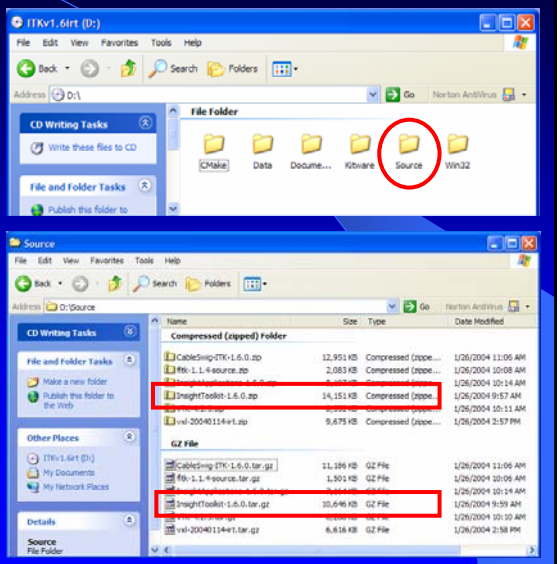
Step 1. Download ITK



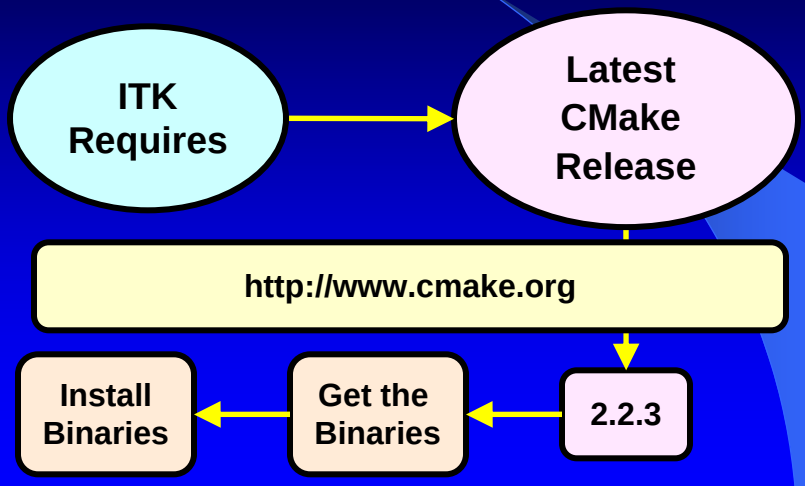
Copying ITK from the CD



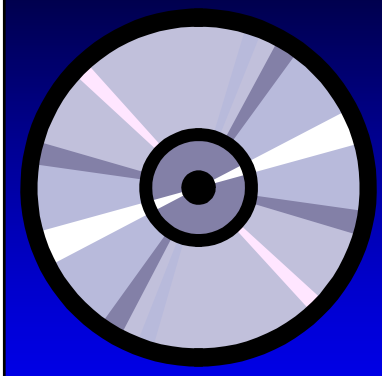
/Source/
InsightToolkit-2.4.1.zip
InsightToolkit-2.4.1.tar.gz

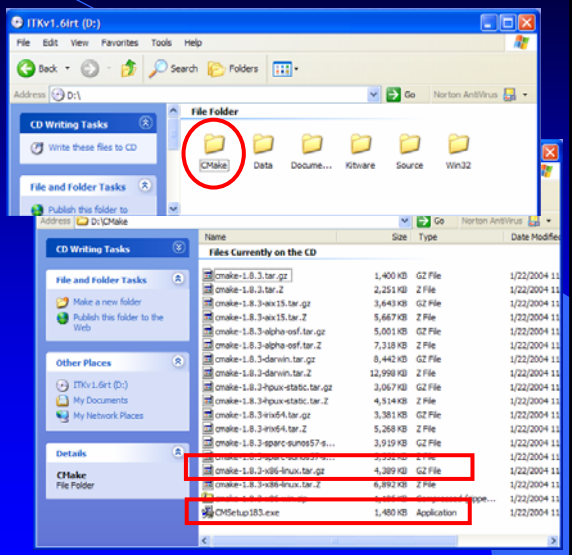


Step 2. Download CMake



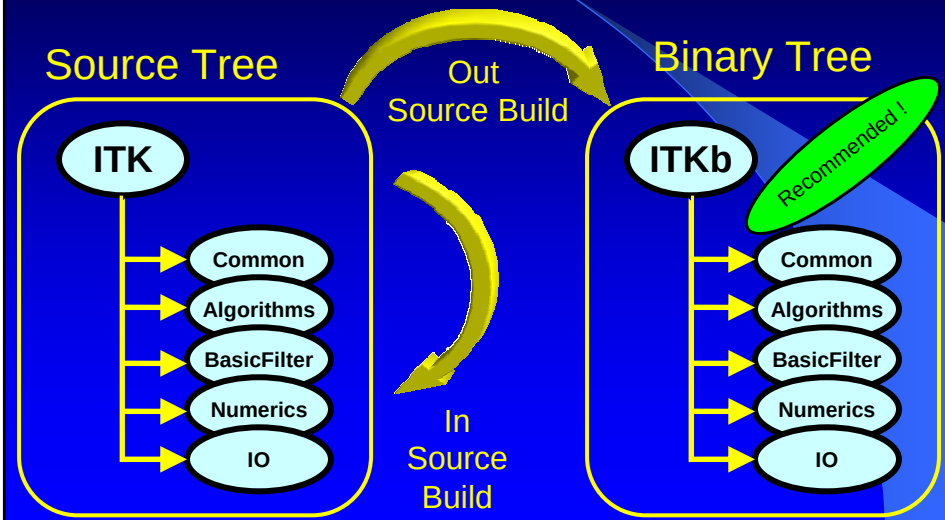
Installing CMake from the CD





/CMake/
- cmake-2.2.3.tar.gz
- cmake-2.2.3-x86-win.zip
- cmake-2.2.3-x86-linux.tar.gz
- CMakeSetup223.exe

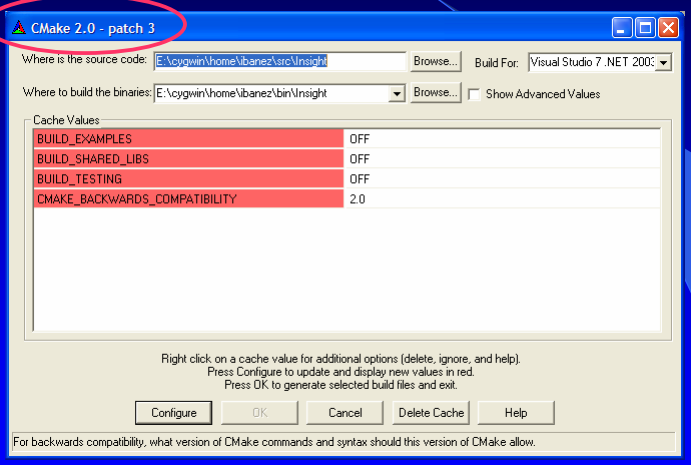
Step 3. Configure ITK



Configuring ITK – MS-Windows

- Run CMake
- Select the SOURCE directory
- Select the BINARY directory
- Select your Compiler

Configuring ITK – MS-Windows



Configuring ITK – MS-Windows

- Disable BUILD_EXAMPLES
- Disable BUILD_SHARED_LIBS
- Disable BUILD_TESTING

- Click “Configure” to configure
- Click “OK” to generate project files

Configuring ITK – Unix

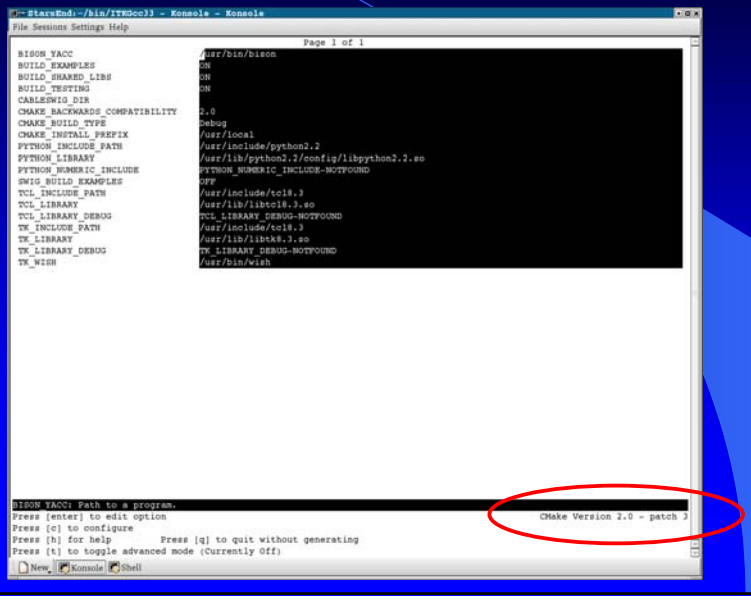
- Create the BINARY directory (mkdir)
- Change directory to the BINARY directory (cd)
- Set the environment variables CC and CXX

  setenv CC /usr/bin/gcc; setenv CXX /usr/bin/g++ OR

  export CC=/usr/bin/gcc; export CXX=/usr/bin/g++

- Type ccmake with argument the SOURCE directory

Configuring ITK – Unix

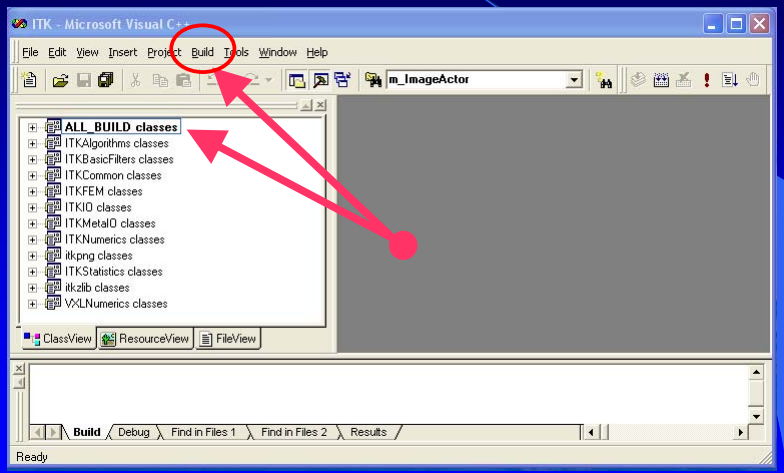


Configuring ITK – Unix

- Disable BUILD_EXAMPLES
- Disable BUILD_SHARED_LIBS
- Disable BUILD_TESTING

- Type “c” to configure
- Type “g” to generate the Makefiles
- Type “make” to start building

Building ITK

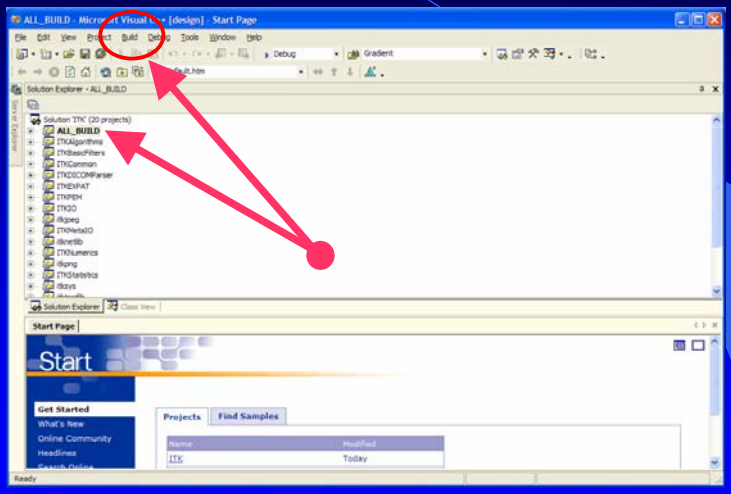


Building ITK

- Open ITK.dsw in the Binary Directory
- Select ALL_BUILD project
- Build it

  …It will take about 15 minutes …

Building ITK



Building ITK

- Open ITK.sln in the Binary Directory
- Select ALL_BUILD project
- Build it

  …It will take about 15 minutes …

Building ITK

- Most of ITK classes are C++ Templates
- Basic libraries are small

  they only contain non-templated classes
- Basic libraries are built in about 15 min

Step 5. Verify the Built

Libraries will be found in

ITK_BINARY / bin / { Debug, Release }

Step 5. Verify the Build

The following libraries should be there

- ITKCommon
- ITKBasicFilters
- ITKAlgorithms
- ITKNumerics
- ITKFEM
- ITKIO
- ITKStatistics
- ITKMetaIO
- itkpng
- itkzlib

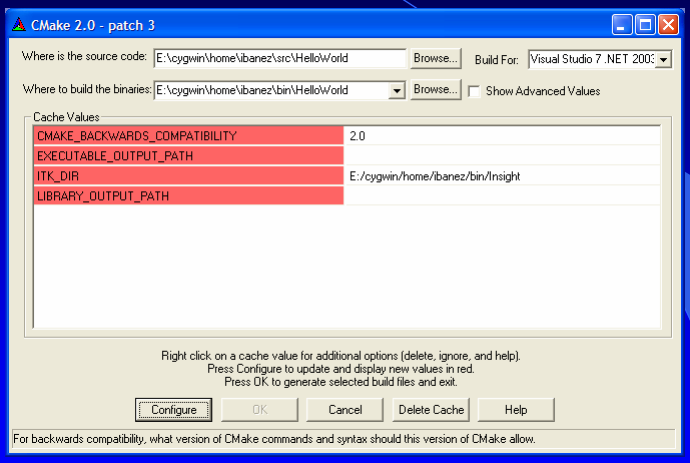
Step 6. Use ITK from an external Project

Copy
"HelloWorld.cxx"
"CMakeLists.txt"
from the
Examples/Installation
Directory
into another
directory

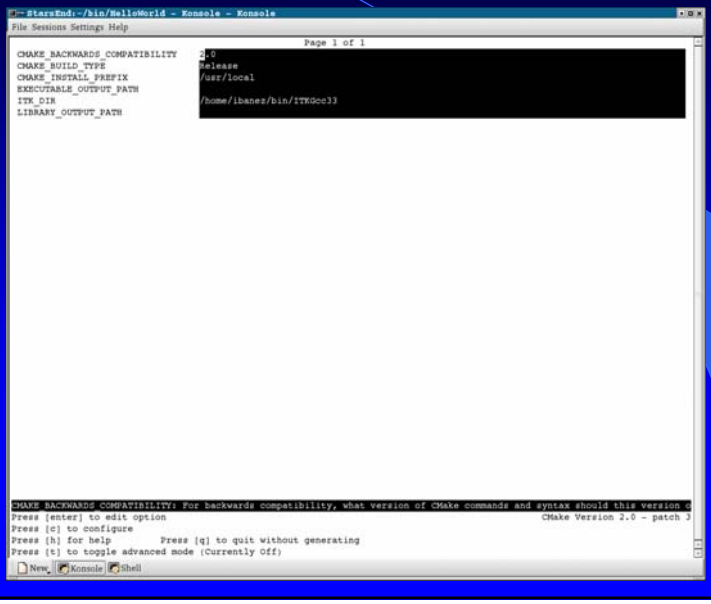
→ Run CMake →

- Select Source Dir
- Select Binary Dir

Using ITK – Hello World



Using ITK – Hello World



Step 6. Use ITK from an external Project

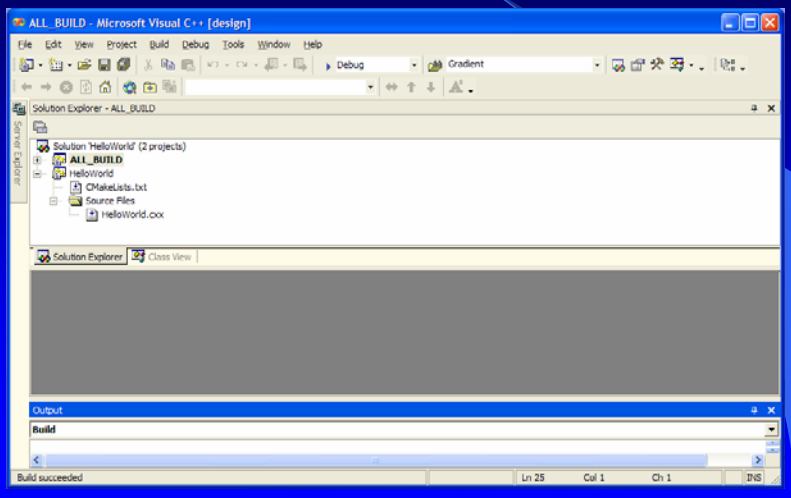
- accept the default in CMAKE_BACKBARD_COMPATIBILITY
- leave empty EXECUTABLE_OUTPUT_PATH
- leave empty LIBRARY_OUTPUT_PATH
- Set ITK_DIR to the binary directory where ITK was built

Step 7. Build Sample Project

- Open HelloWorld.dsw (or .sln) generated by CMake
- Select ALL_BUILD project
- Build it

  …It will take about 3 seconds …

Step 7. Build Sample Project



Step 8. Run the example

- Locate the file HelloWorld.exe
- Run it…
- It should produce the message:

ITK Hello World !

Starting your own project

- Create a clean new directory
- Write a CMakeLists.txt file
- Write a simple .cxx file
- Configure with CMake
- Build
- Run

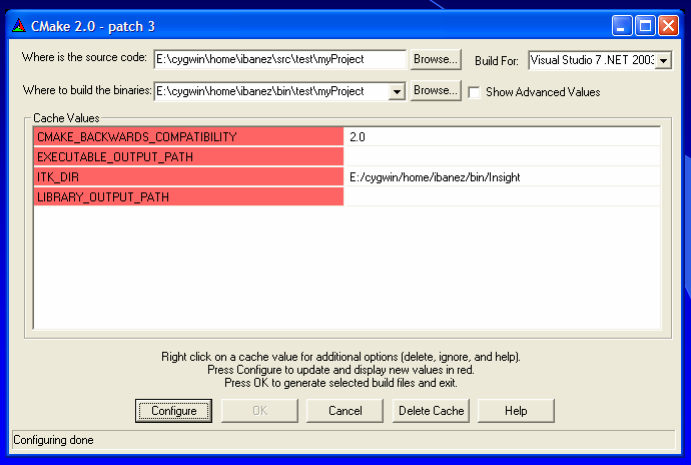
Step 9. Writing CMakeLists.txt

```
PROJECT( myProject )

FIND_PACKAGE ( ITK )
IF ( ITK_FOUND )
      INCLUDE( ${ITK_USE_FILE} )
ENDIF( ITK_FOUND )

ADD_EXECUTABLE( myProject   myProject.cxx )

TARGET_LINK_LIBRARIES ( myProject  ITKCommon   ITKIO)
```

Step 10. Writing myProject.cxx

```
#include "itkImage.h"
#include "itkImageFileReader.h"
#include "itkGradientMagnitudeImageFilter.h"

int main( int argc, char **argv ) {
  typedef itk::Image<unsigned short,2>          ImageType;
  typedef itk::ImageFileReader<ImageType>       ReaderType;
  typedef itk::GradientMagnitudeImageFilter<
                     ImageType,ImageType>       FilterType;

  ReaderType::Pointer reader = ReaderType::New();
  FilterType::Pointer filter = FilterType::New();

  reader->SetFileName( argv[1] );
  filter->SetInput( reader->GetOutput() );
  filter->Update();
  return 0;
}
```

Step 11. Run CMake

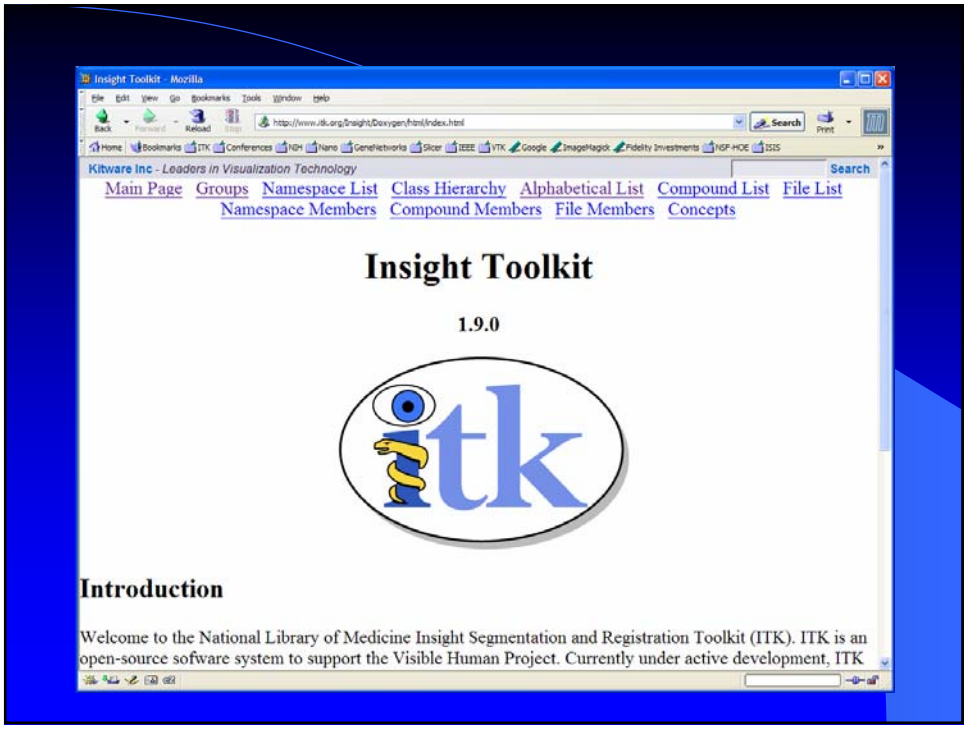


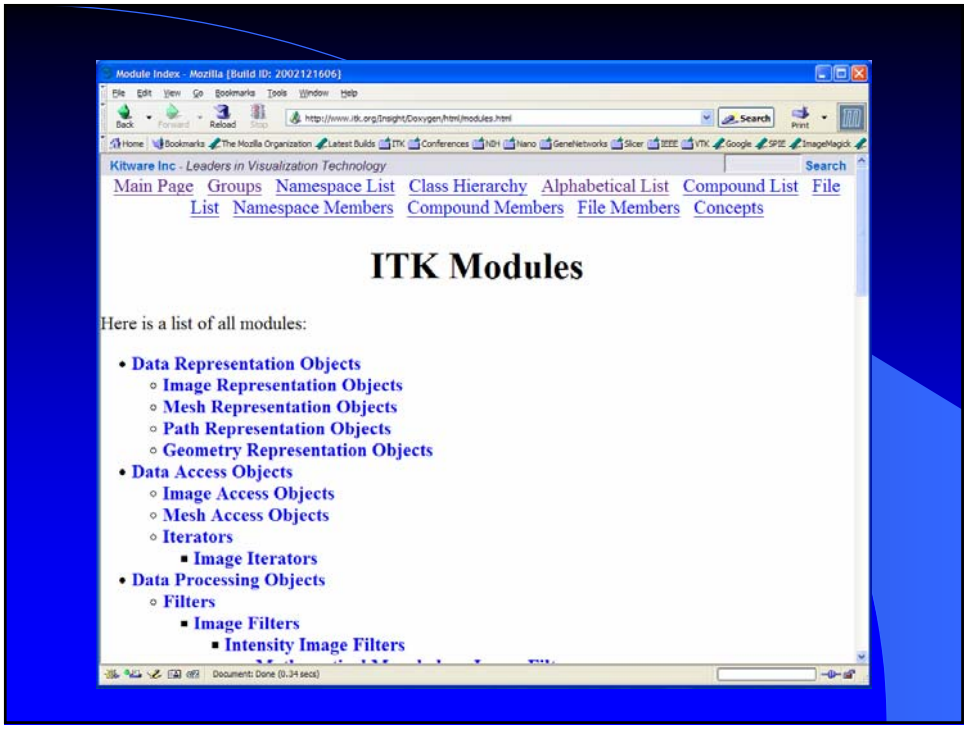
Step 12. How to find what you need

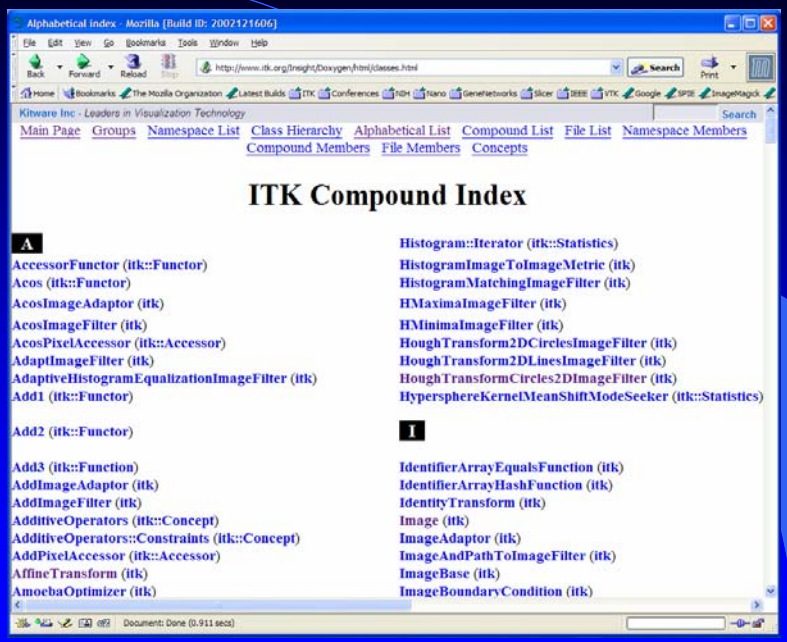
http://www.itk.org/ItkSoftwareGuide.pdf

http://www.itk.org/Doxygen/html/index.html

- Follow the link Alphabetical List
- Follow the link Groups
- Post to the insight-users mailing list


Insight Toolkit - Mozilla
Kitware Inc - Leaders in Visualization Technology
Search
Main Page Groups Namespace List Class Hierarchy Alphabetical List Compound List File List
Namespace Members Compound Members File Members Concepts
Insight Toolkit
1.9.0
itk
Introduction
Welcome to the National Library of Medicine Insight Segmentation and Registration Toolkit (ITK). ITK is an open-source sofware system to support the Visible Human Project. Currently under active development, ITK

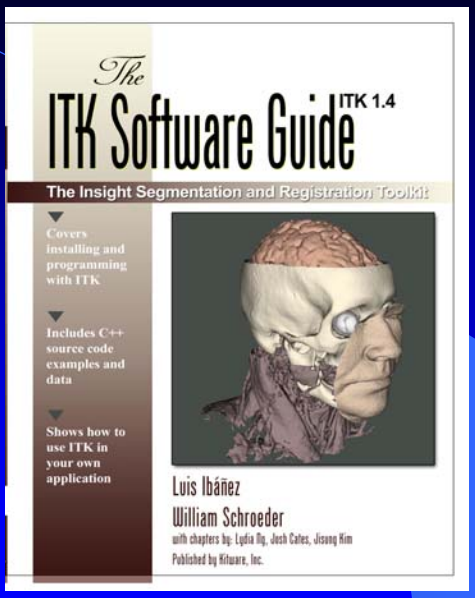

Module Index - Mozilla {Build ID: 2002121606}
Kitware Inc - Leaders in Visualization Technology
Search
Main Page Groups Namespace List Class Hierarchy Alphabetical List Compound List File List Namespace Members Compound Members File Members Concepts
ITK Modules
Here is a list of all modules:
Data Representation Objects
Image Representation Objects
Mesh Representation Objects
Path Representation Objects
Geometry Representation Objects
Data Access Objects
Image Access Objects
Mesh Access Objects
Iterators
Image Iterators
Data Processing Objects
Filters
Image Filters
Intensity Image Filters
Document: Done (0.34 secs)



The ITK Software Guide is freely available as a PDF document at

www.itk.org/ItkSoftwareGuide.pdf

Its paper version can be ordered from Amazon.com and from Kitware's e-store.




Enjoy ITK !

ITK Basics

- C++ Generic Programming
- Data Pipeline
- Multi-threading
- Streaming
- Exceptions
- Events / Observers
- Tcl, Python and Java wrapping

Generic Programming

Example: STL Standard Template Library

Abstraction of Types and Behaviors

std::vector< T >

std::vector< int >
std::vector< double >
std::vector< char * >
std::vector< Point >
std::vector< Image >

itk::Image

itk::Image< PixelType , Dimension >

itk::Image< char , 2 >
itk::Image< char , 3 >
itk::Image< char , 4 >
itk::Image< float , 2 >
itk::Image< RGB , 3 >
itk::Image< unsigned short , 2 >
itk::Image< itk::Vector<float,2> , 2 >

namespaces

Avoid naming collisions

itk::

itk::Statistics::

itk::fem::

itk::fem::itpack

itk::bio

Your favorite keyword

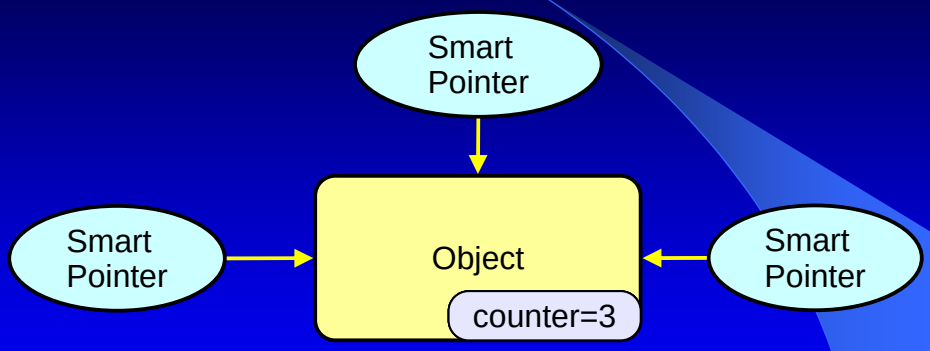
typedef

```
typedef   itk::Image< char , 2 >   ImageType
typedef   itk::ImageFilter< ImageType , ImageType >   FilterType
```

otherwise...

```
itk::ImageFilter< Image< char , 2 > ,
                  Image< char , 2 > >   FilterType
```

Smart Pointers



Self - Delete

SmartPointers

```
typedef   itk::Image< char , 2 >   ImageType
typedef   itk::ImageFilter< ImageType , ImageType >   FilterType

    FilterType::Pointer  filter = FilterType::New();

    ImageType::Pointer image = filter->GetOutput();
```

Pointer notation

```
                    filter->Update();
```

NO NEED FOR

```
                    filter->Delete();
```

Const Correctness

Knowing constancy is Insight.

Not knowing constancy leads to disaster.

Tao Te Ching, XVI. Lao Tsu

Const Smart Pointers

```
typedef   itk::Image< char , 2 >   ImageType
typedef   itk::ImageFilter< ImageType , ImageType >   FilterType

FilterType::Pointer  filter = FilterType::New();

ImageType::ConstPointer image = filter->GetOutput();
```

Can only invoke “const” methods

```
image->GetSpacing ();
```

Compiler error for “non-const” methods

```
image->SetSpacing ( spacing );
```

Creating an Image

```
typedef   itk::Image< char , 3 >   ImageType

ImageType::Pointer image = ImageType::New();

ImageType::SizeType size;
size[ 0 ] = 512;   // x direction
size[ 1 ] = 512;   // y direction
size[ 2 ] =   50;   // z direction

ImageType::IndexType start;
start[ 0 ] = 0;      // x direction
start[ 1 ] = 0;      // y direction
start[ 2 ] = 0;      // z direction
```

Creating an Image

```
ImageType::RegionType region;
region.SetSize( size );
region.SetIndex( start );

image->SetRegions( region );
image->Allocate();
image->FillBuffer( 0 );

ImageType::SpacingType spacing;
spacing[ 0 ] = 0.83;        // x direction
spacing[ 1 ] = 0.83;        // y direction
spacing[ 2 ] = 2.15;        // z direction

image->SetSpacing( spacing );
```

Exercise 3

Streaming

Processing Large Images

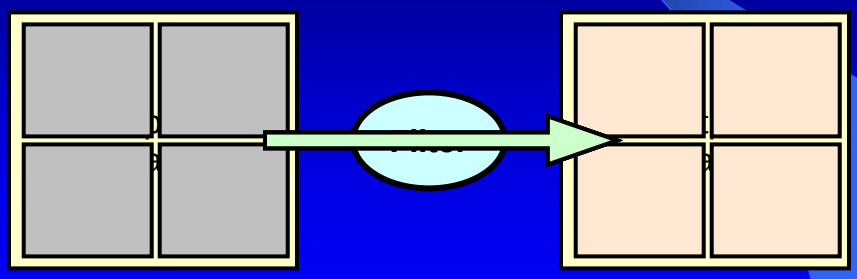
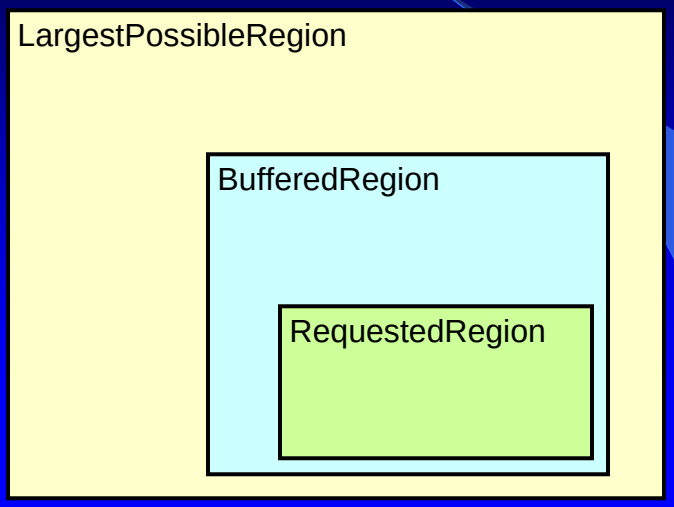
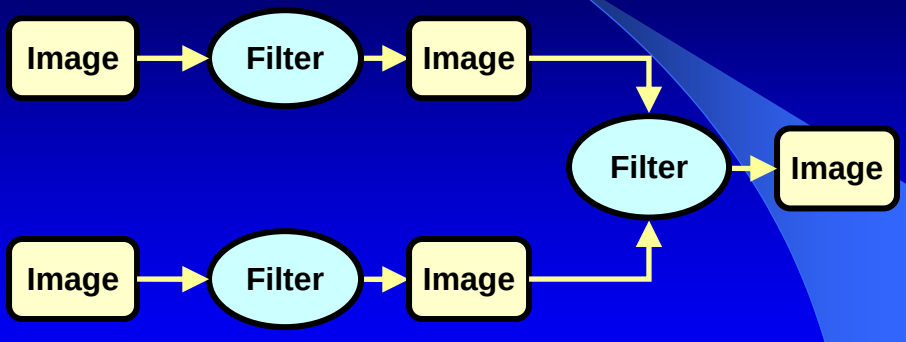
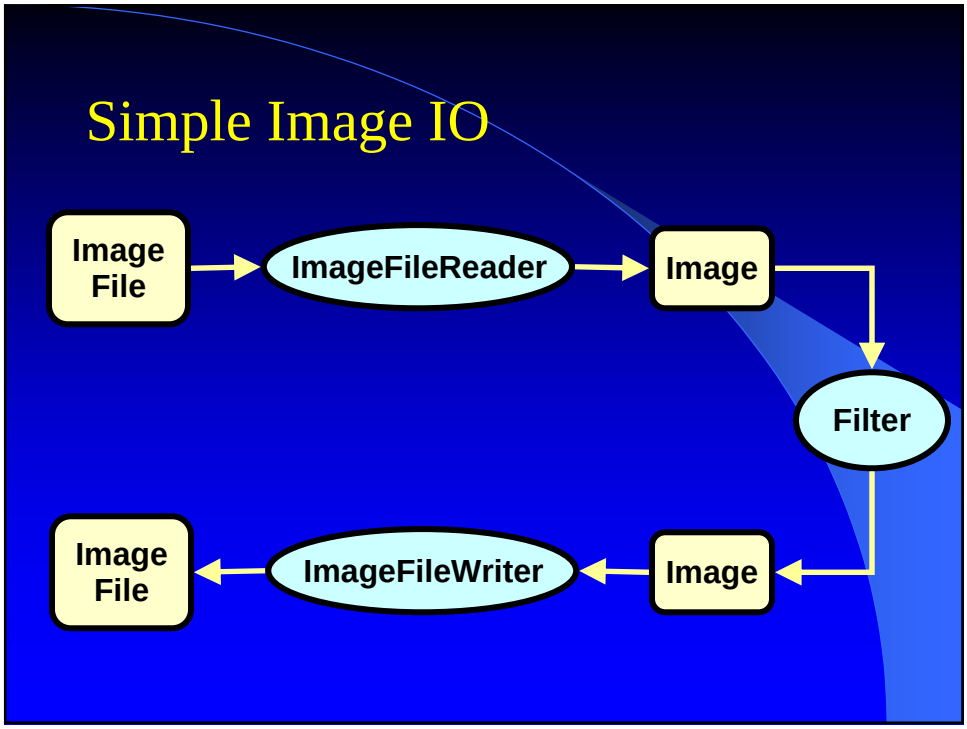


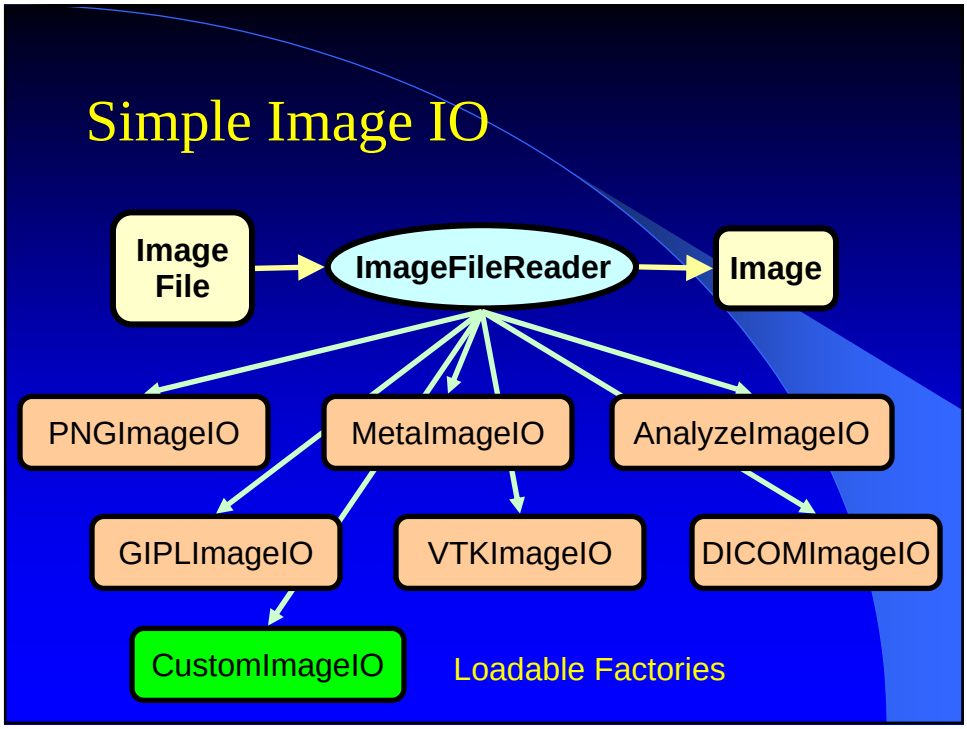
Image Regions



Data Pipeline




Simple Image IO
Image File
ImageFileReader
Image
Filter
Image File
ImageFileWriter
Image

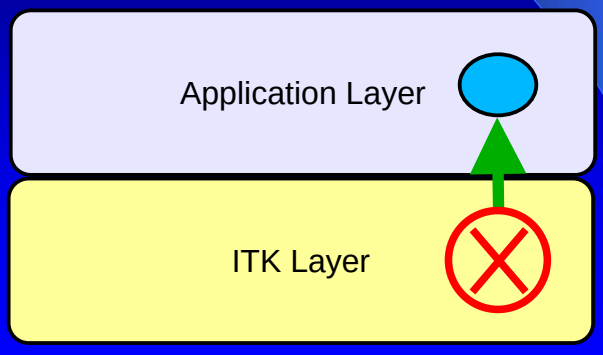

Simple Image IO
Image File
ImageFileReader
Image
PNGImageIO
MetaImageIO
AnalyzeImageIO
GIPLImageIO
VTKImageIO
DICOMImageIO
CustomImageIO
Loadable Factories

Simple Image IO

```
#include “itkImage.h”
#include “itkImageFileReader.h”
#include “itkImageFileWriter.h”

typedef   itk::Image< char , 2 >   ImageType;
typedef   itk::ImageFileReader< ImageType >   ReaderType;
typedef   itk::ImageFileWriter<   ImageType >   WriterType;

ReaderType::Pointer reader = ReaderType::New();
WriterType::Pointer    writer  = WriterType::New();

reader->SetFileName( “inputImage.dcm” );      // DICOM
writer->SetFileName( “outputImage.hdr” );      // Analyze

writer->SetInput(   reader->GetOutput()   );
writer->Update();
```

Exceptions

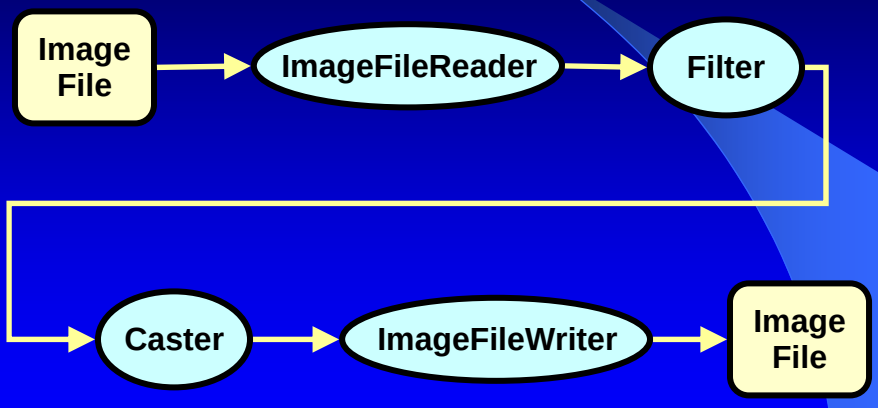
Error Management



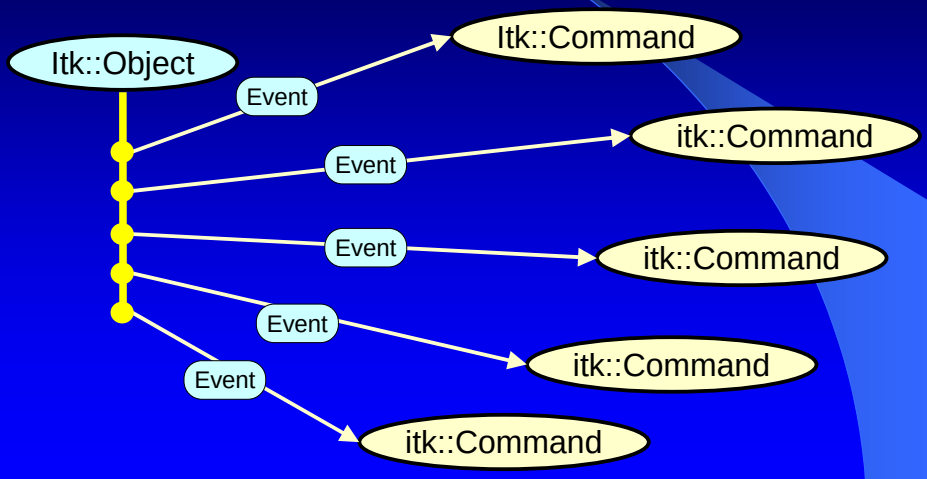
Exceptions

```
try
  {
      filter->Update();
  }
catch(  itk::ExceptionObject & exp )
  {
      std::cerr << exp << std::endl;
  }
```

Exercise 4



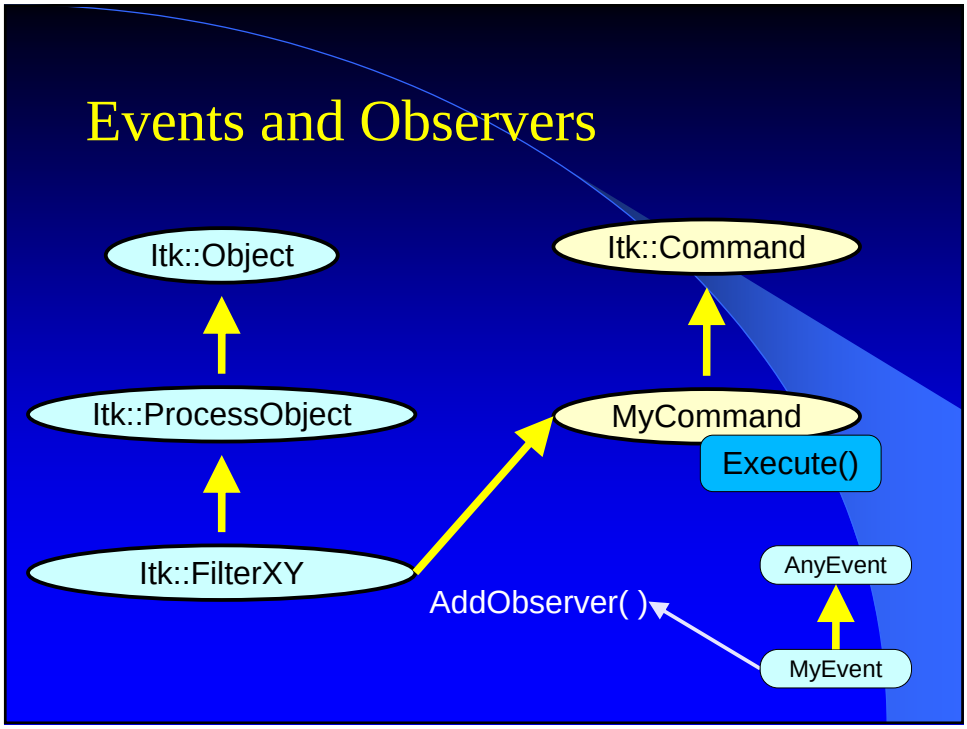
Events and Observers

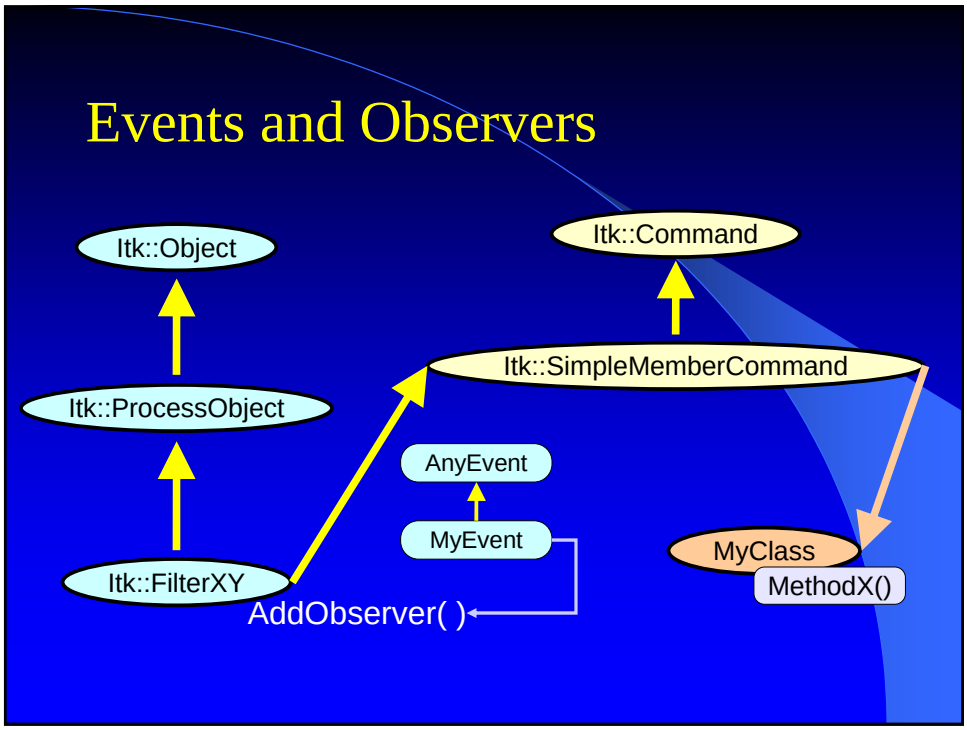


Events and Observers

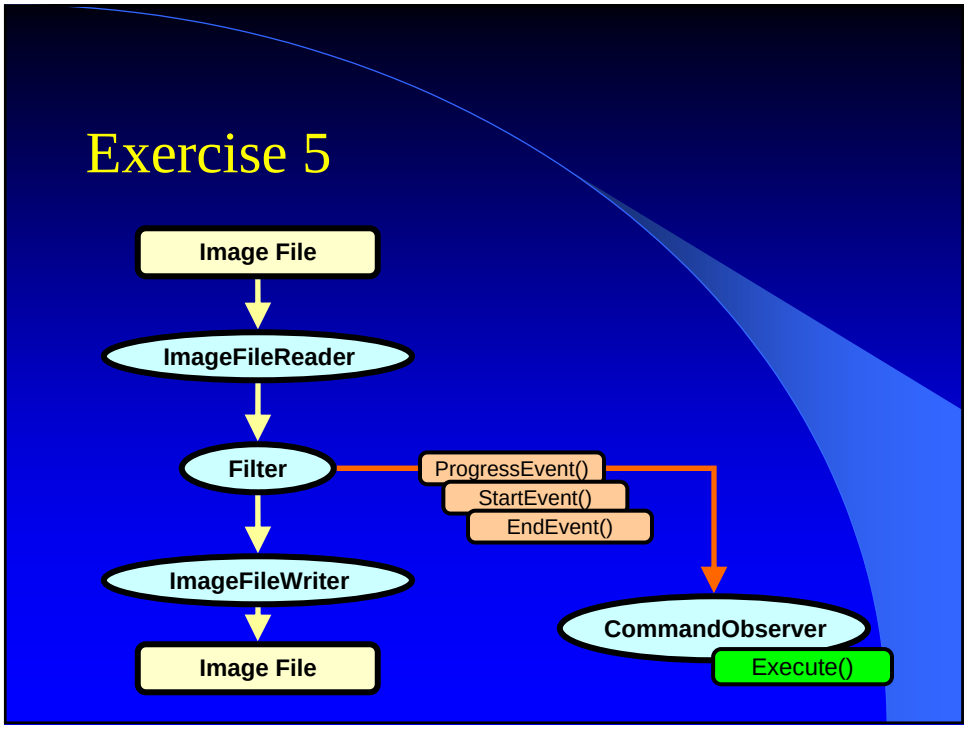
Common Events

AnyEvent()
StartEvent()
EndEvent()
ProgressEvent()
IterationEvent()

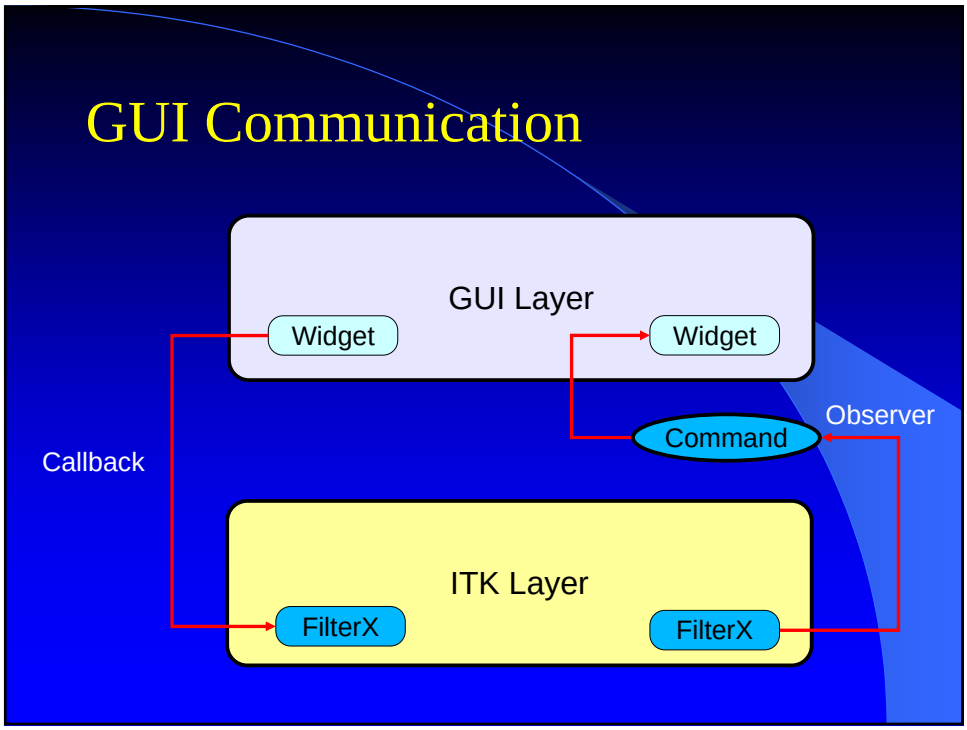

Events and Observers
Itk::Object
Itk::Command
Itk::ProcessObject
MyCommand
Execute()
Itk::FilterXY
AddObserver( )
AnyEvent
MyEvent


Events and Observers
Itk::Object
Itk::Command
Itk::SimpleMemberCommand
Itk::ProcessObject
AnyEvent
MyEvent
MyClass
MethodX()
Itk::FilterXY
AddObserver( )

Exercise 5



GUI Communication




Enjoy ITK !

Introduction to ITK Segmentation

The Insight Consortium
presented by
Josh Cates
Scientific Computing and Imaging Institute
University of Utah

Session Objectives

- Review some important considerations when using ITK filters
- Overview of low-level image processing filters
  - –denoising / scale space
  - –feature extraction
- Overview some common segmentation filters
  - –connected component
  - –pixel classification
  - –watersheds
  - –level-set methods

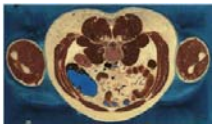
- Brief introduction to image processing “frameworks” in ITK
- See examples
  - –simple command line
  - –full applications with user interaction

Important Facts about ITK Filters

1. Most filters are N dimensional
2. Many filters run multi-threaded
3. Filters are implemented within frameworks
4. Most filters are documented
5. Filters are regression tested
6. Data type matters
7. Pixel spacing matters
8. Filters are not full applications
9. Filters require parameter tuning

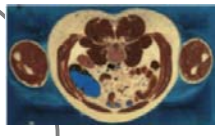
Big Picture: The Role of ITK Filters

Raw Data



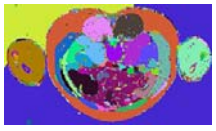
Preprocessing

Filtering
linear
nonlinear



Feature Extraction
differential
geom.
edge detection

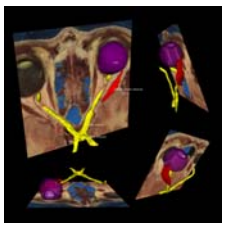
Segmentation
region growing
watersheds
level-sets



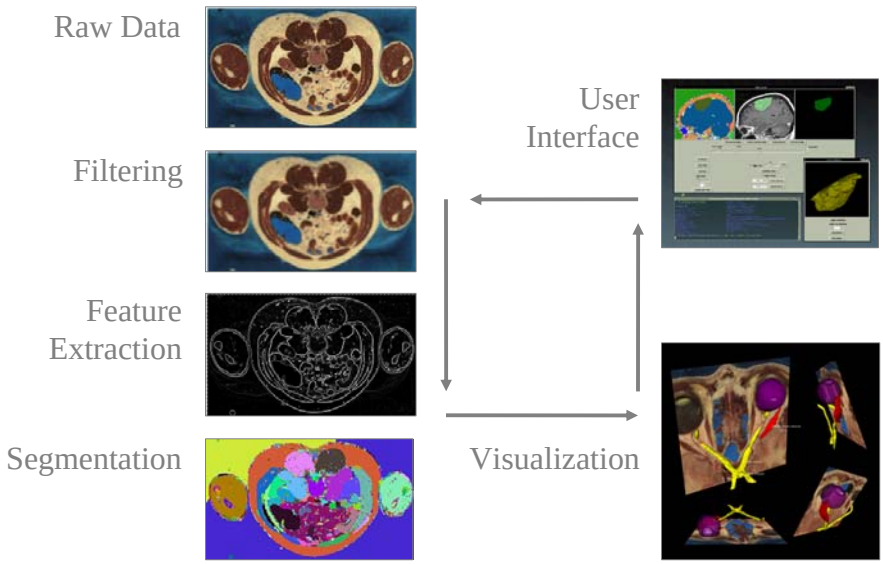
Dataflow architecture: ITK filters fit together to produce segmentation applications.

Moral: ITK segmentation filters are not complete applications – components in a pipeline.

Visualization
binary volume
meshes
labeled image
implicit surfaces



Big Picture: The Role of ITK Filters

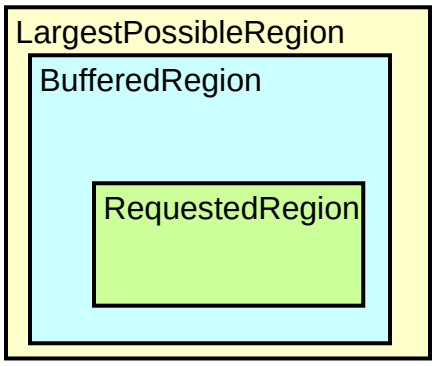


Where to go to *really* learn to use the filters

- http://www.itk.org
- Doxygen Manual Pages
- Software Guide
  - Algorithm descriptions
  - Tutorials for using algorithms
- Theory Book
  - *Insight into Images*, Terry Yoo, ed.
- Users Mailing List
- CVS Repositories:
  - Insight/Examples
  - InsightApplications

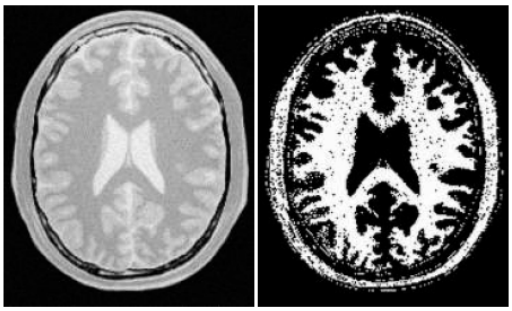
What is an ITK Image?

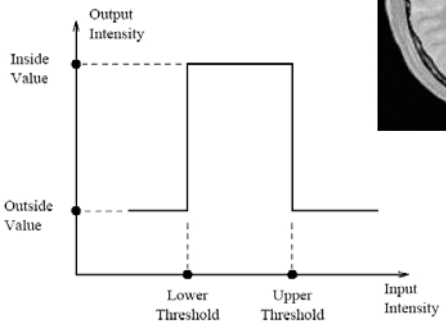
- Templated over data type
- May be vector-valued
- Templated over dimensionality
- Spacing, origin & orientation information
- Pixels may be addressed directly or through an assortment of iterators
- Interpolators may also be used
- In general, implemented as a flat, C++ array
- Buffers may be imported & exported



Thresholding

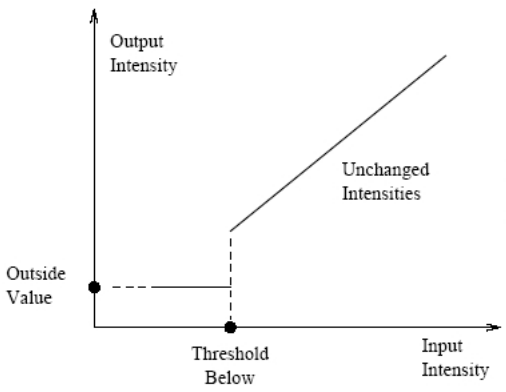
BinaryThresholdImageFilter





Thresholding

ThresholdImageFilter



Intensity Transformations

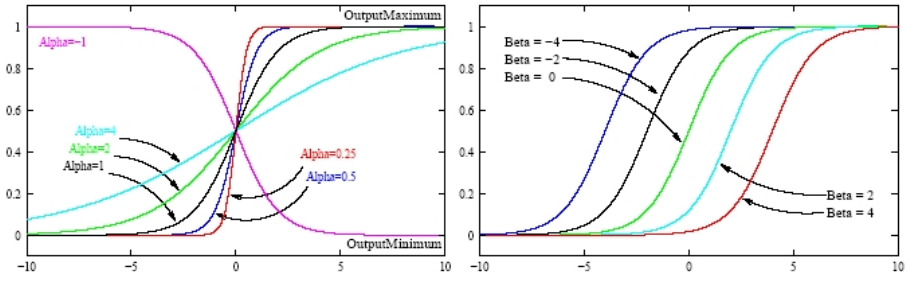
CastImageFilter

RescaleIntensityImageFilter

ShiftScaleImageFilter

NormalizeImageFilter

SigmoidImageFilter



Intensity Transformations

SigmoidImageFilter

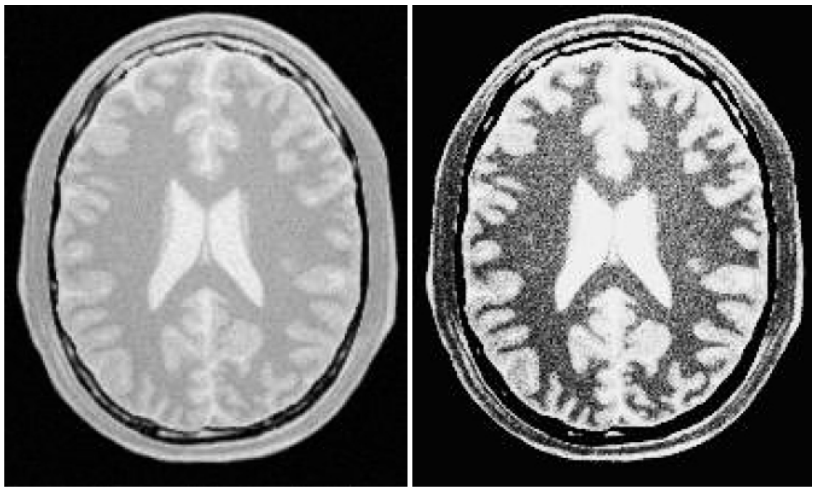


Image Morphology

BinaryErodeImageFilter

BinaryDilateImageFilter

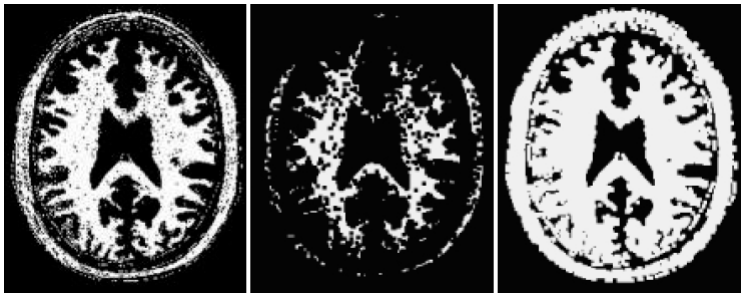
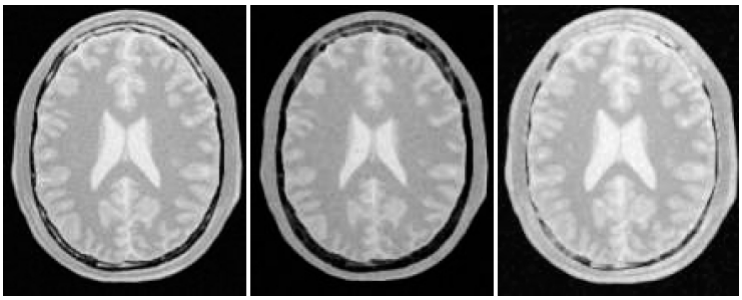


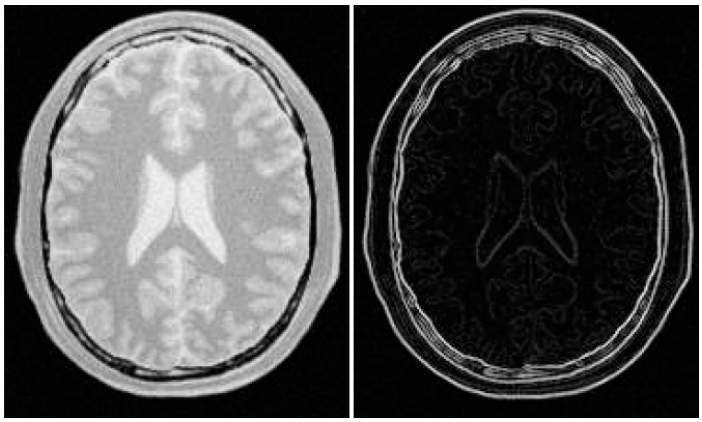
Image Morphology

GrayscaleErodeImageFilter
GrayscaleDilateImageFilter



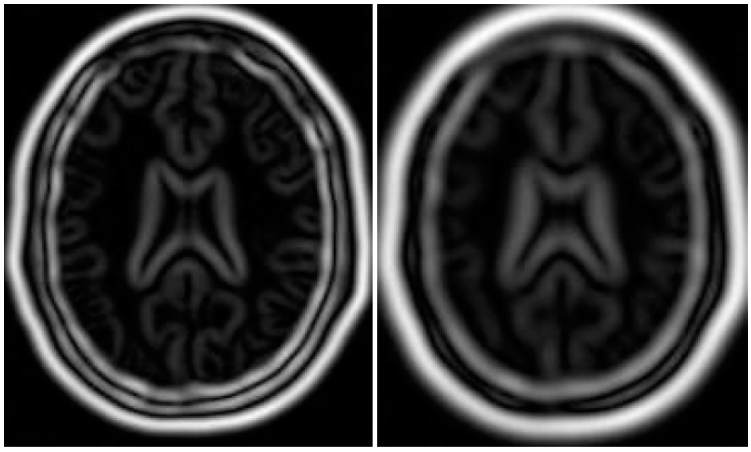
Edge Detection & Feature Extraction

CannyEdgeDetectionImageFilter
LaplacianImageFilter
GradientMagnitudeImageFilter



Edge Detection & Feature Extraction

GradientMagnitudeRecursiveGaussianImageFilter



Edge Detection & Feature Extraction

DerivativeImageFilter

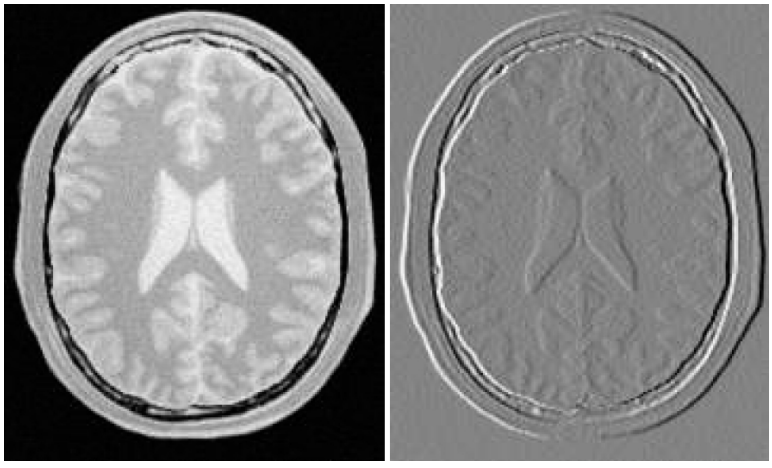


Image Denoising: Linear

MeanImageFilter

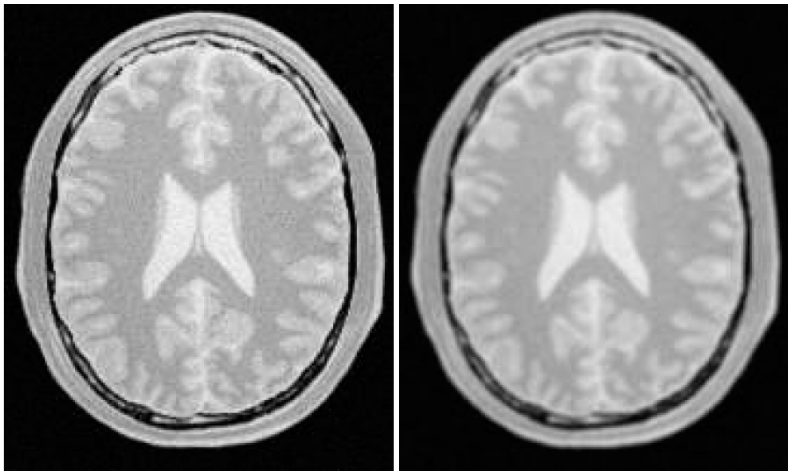


Image Denoising: Linear

MedianImageFilter

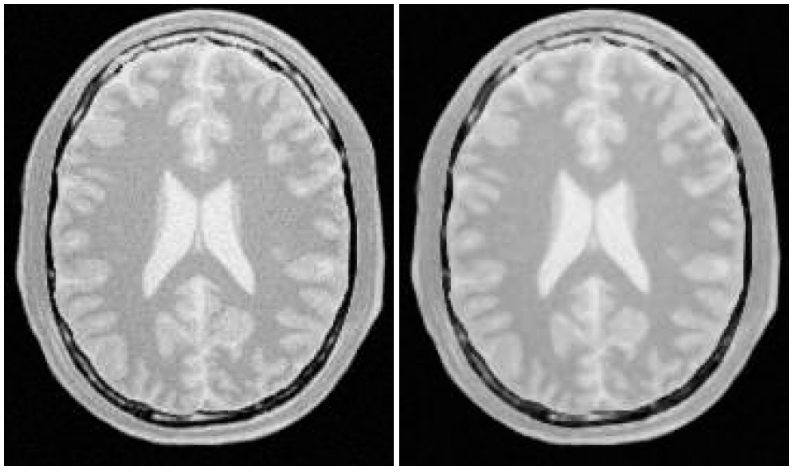


Image Denoising: Linear

BinomialBlurImageFilter
RecursiveGaussianImageFilter
DiscreteGaussianImageFilter

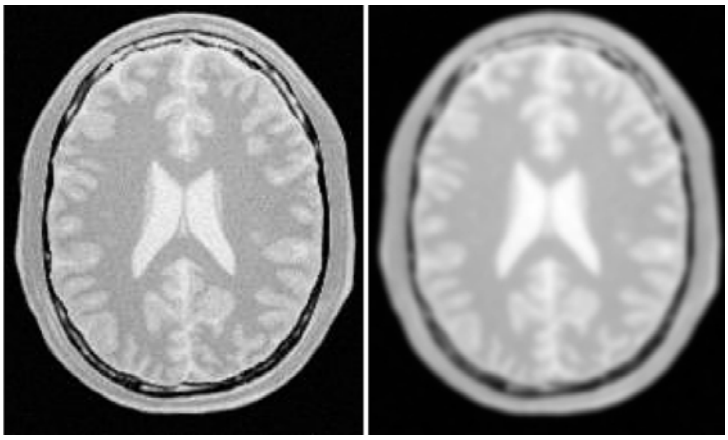


Image Denoising: Nonlinear

BilateralImageFilter
CurvatureFlowImageFilter
GradientAnisotropicDiffusionImageFilter
CurvatureAnisotropicDiffusionImageFilter

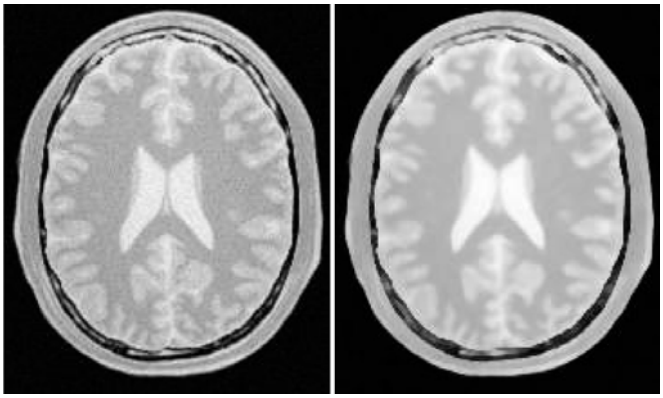
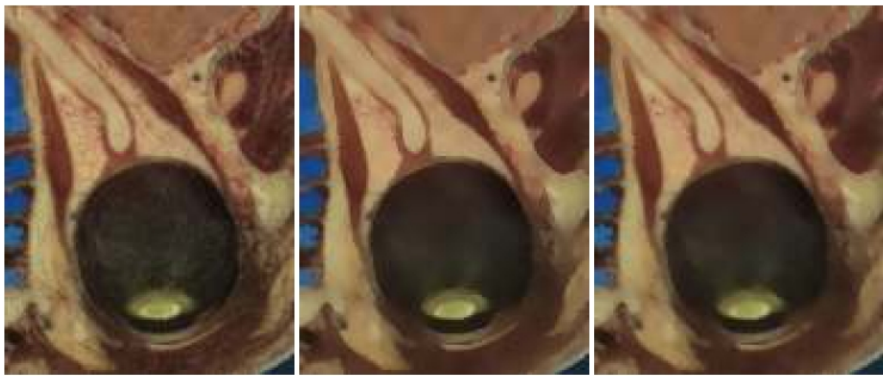


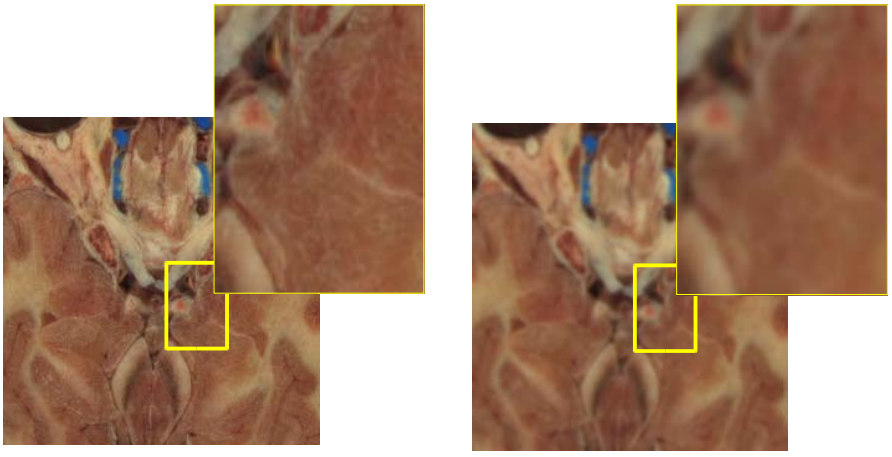
Image Denoising: Nonlinear

VectorGradientAnisotropicDiffusionImageFilter
VectorCurvatureAnisotropicDiffusionImageFilter



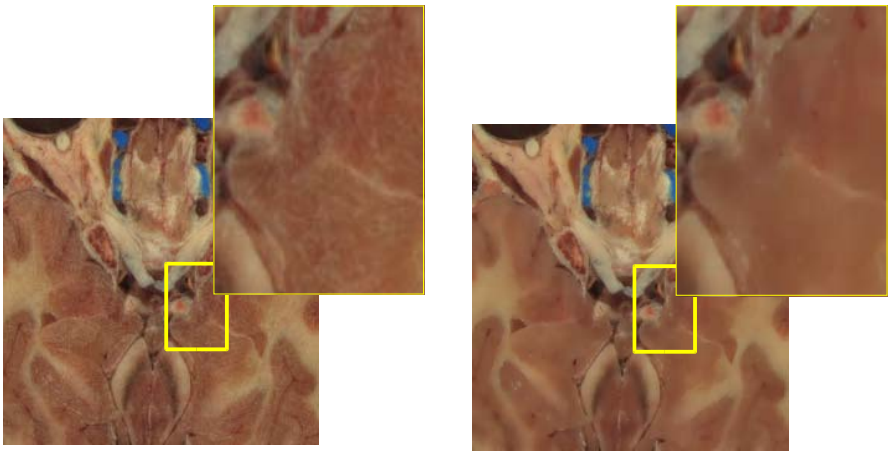
Linear Diffusion

Destroys and moves edges



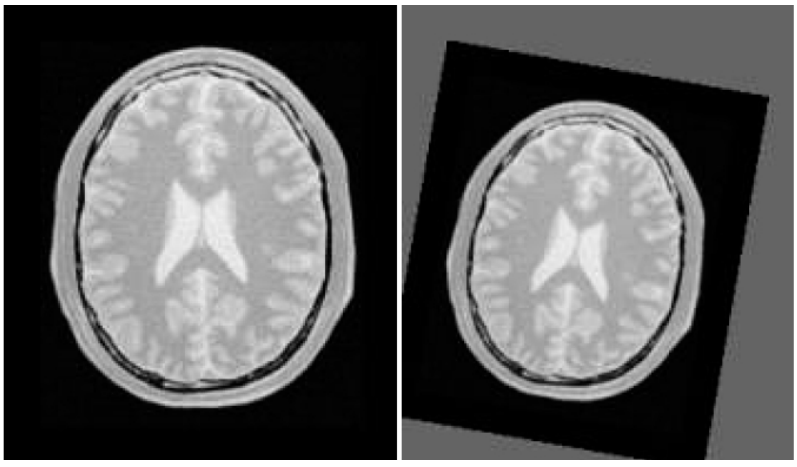
Nonlinear Diffusion

Preserves Edges



Geometric Transformations

TransformImageFilter



Remember

Filters generalize to N dimensions

Filters “work” on arbitrary data types, but only some data types make sense

Filters are implemented within *extensible* frameworks

Image Neighborhood Framework

Neighborhood is a set of pixels local adjacent to one another

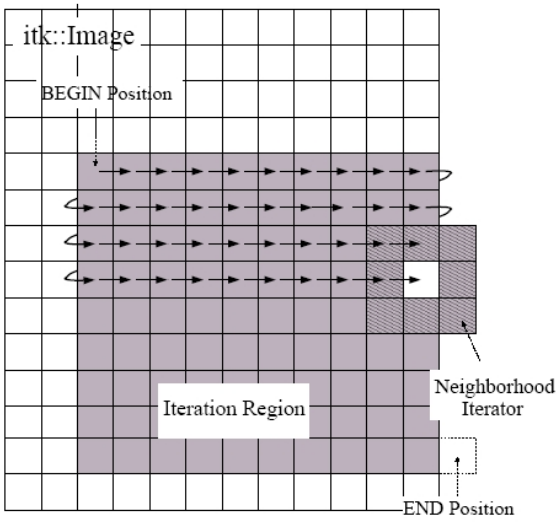
Used to implement algorithms where result at pixel *i* is based on some computation within the neighborhood of *i*

Iterators abstract the API for working with a neighborhood of pixels in arbitrary dimensions

Operators are applied to image neighborhood to perform various calculations

Boundary conditions are handled automatically

Neighborhood Iterator Framework



Neighborhood Iterators

0 (−1, −1)	1 (0, −1)	2 (1,−1)
3 (−1,0)	4 (0,0)	5 (1,0)
6 (−1,1)	7 (0,1)	8 (1,1)

radius = [1,1]
size = [3,3]

0 (−1,−2)	1 (0,−2)	2 (1,−2)
3 (−1,−1)	4 (0,−1)	5 (1,−1)
6 (−1,0)	7 (0,0)	8 (1,0)
9 (−1,1)	10 (0,1)	11 (1,1)
12 (−1,2)	13 (0,2)	14 (1,2)

radius = [1,2]
size = [3,5]

0 (−1,0)	1 (0,0)	2 (1,0)

radius = [1,0]
size = [3,1]

0 (−3,−1)	1 (−2,−1)	2 (−1,−1)	3 (0,−1)	4 (1,−1)	5 (2,−1)	6 (3,−1)
7 (−3,0)	8 (−2,0)	9 (−1,0)	10 (0,0)	11 (1,0)	12 (2,0)	13 (3,0)
14 (−3,1)	15 (−2,1)	16 (−1,1)	17 (0,1)	18 (1,1)	19 (2,1)	20 (3,1)

radius = [3,1]
size = [7,3]

0 (0,−2)
1 (0,−1)
2 (0,0)
3 (0,1)
4 (0,2)

radius = [0,2]
size = [1,5]

ITK Segmentation Algorithms

Classification / Thresholding

Region Growing

Watersheds

Level-set Methods

“Hybrid” Methods

Statistical Pattern Classification

Idea

- Find disjoint regions in a feature space
- Classify image pixels according to feature vectors

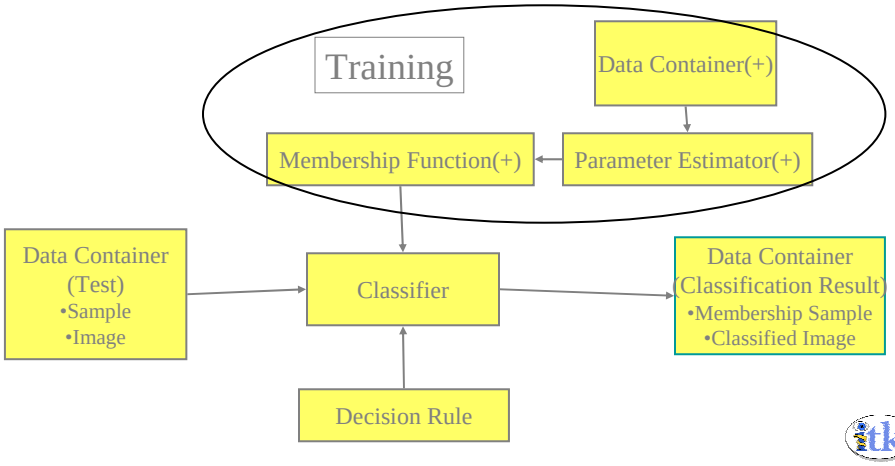
Classifier

- Multiple membership functions (each represents one possible class) return scores from feature vectors
- Decision rule based on scores

Implemented using the ITK statistics subsystem (framework)

Statistical Pattern Classification Framework

1. Measurement vectors are input to membership functions
2. Membership functions feed scores to decision rule
3. Decision rule compares scores and returns a class label



Classifier Framework Example

Segmentation of gene expression images of the mouse brain at the Allen Institute for Brain Science, Seattle WA*

Designed with ITK components

- Itk::ImageModelEstimatorBase & related classes
- itk::Statistics::MembershipFunctionBase & related classes
- itk::DecisionRuleBase & related classes
- Other filters: morphological, connected component labeling, threshold filters etc.

Requirements

- Modular design / code reuse
- Careful memory usage (each 2D image is on the order of 150 MPixels)
- Robust and adaptable to change in image quality

*Courtesy of Dr. Lydia Ng, Allen Institute for Brain Science, www.brainatlas.org.

Classifier Framework Example

(a) (b) (c)

(a) Original ISH stained image; (b) heat map representing the membership values of each pixel representing expression, and (c) the final threshold mask generated from the heat map

Region Growing

Idea
- Start with set of ***seed pixels*** – *region*
- Iteratively include neighboring pixels that satisfy ***membership criteria***

Membership criteria – similarity based metrics
- Intensity interval, Regional statistics

Algorithms
- Simple to complex variations
- Easy to write using ND neighborhood tools

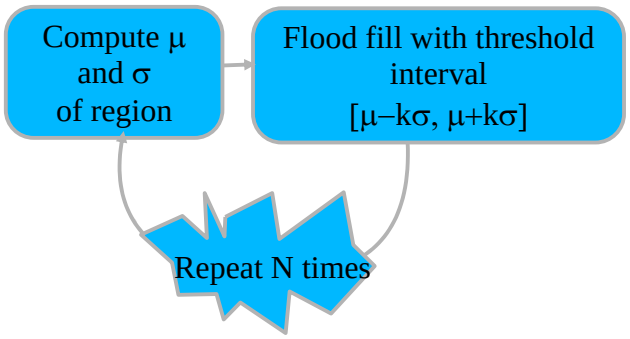
Several strategies
- Connected Threshold, Otsu Threshold, Neighborhood Connected, (Vector) Confidence Connected, Isolated Connected

Confidence Connected Filter

Threshold based region growing

Mean and standard deviation of region determine upper and lower thresholds

Recomputes thresholds at intervals



Region Growing Segmentation

ConfidenceConnectedImageFilter

original | white matter (60,116) | ventricle (81,112) | gray matter (107,69)

smoothing iterations 5
smoothing time step 0.125
C.C. multiplier 2.5
C.C. iterations 5

Region Growing Segmentation

VectorConfidenceConnectedImageFilter

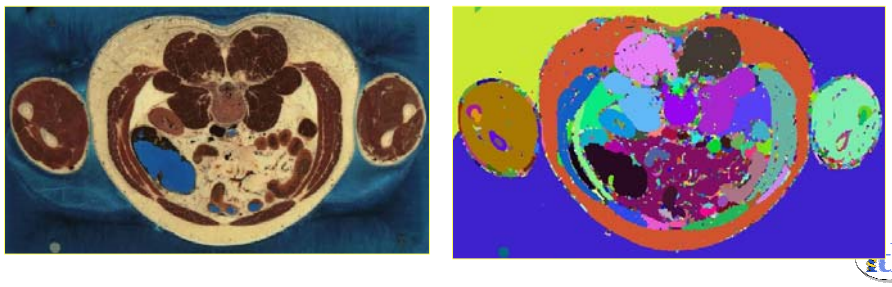
Watershed Segmentation

Image treated as a topological relief map – intensity represents height

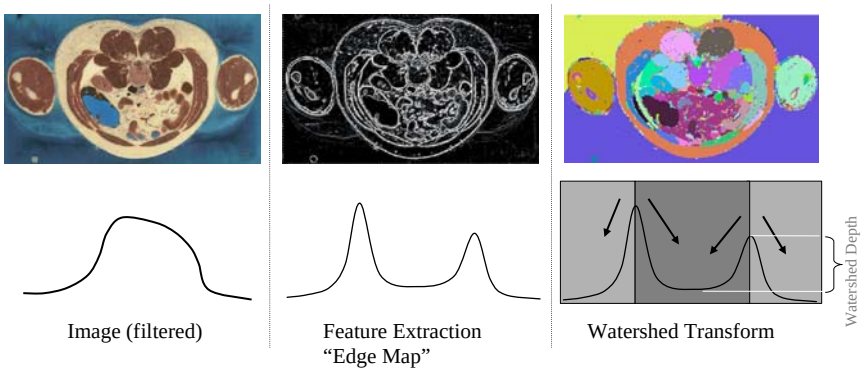
Gradient descent defines ***segmented regions***

- Set of all pixels whose paths of steepest descent terminate in same local minimum
- Bounded by image features

"No parameters"



ITK Watershed Transform



The Oversegmentation Problem

Watershed transform produces too many regions

- One per local minimum
- Especially in noisy or highly detailed data

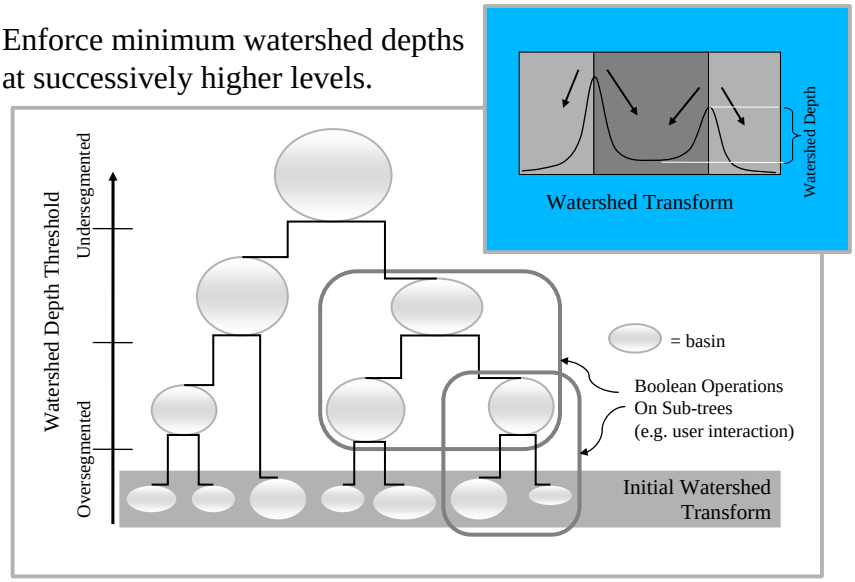
To alleviate oversegmentation

- Hierarchical approach – merge adjacent regions according to increasing *watershed depth*

A. P. Mangan, R. T. Whitaker, Partitioning 3D surface meshes using watershed segmentation, IEEE Transactions on Visualization and Computer Graphics 5 (4) (1999) 308–321.

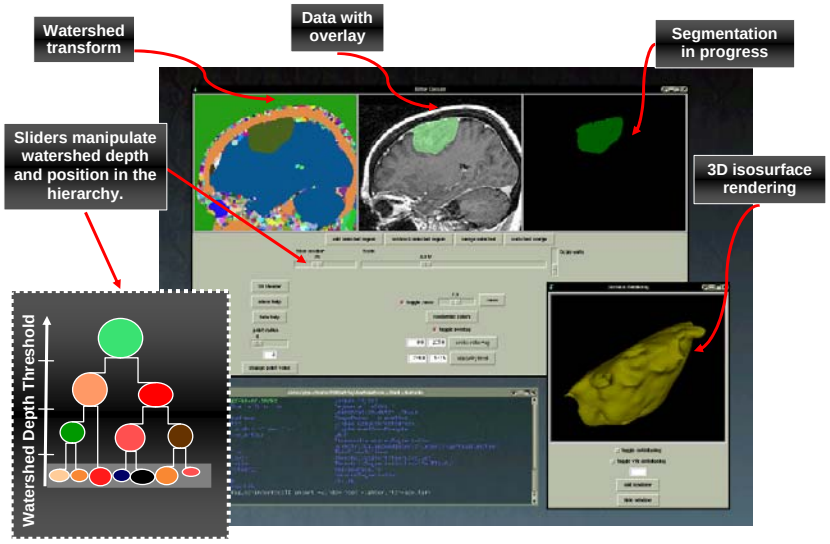
Watersheds Hierarchy

Enforce minimum watershed depths at successively higher levels.

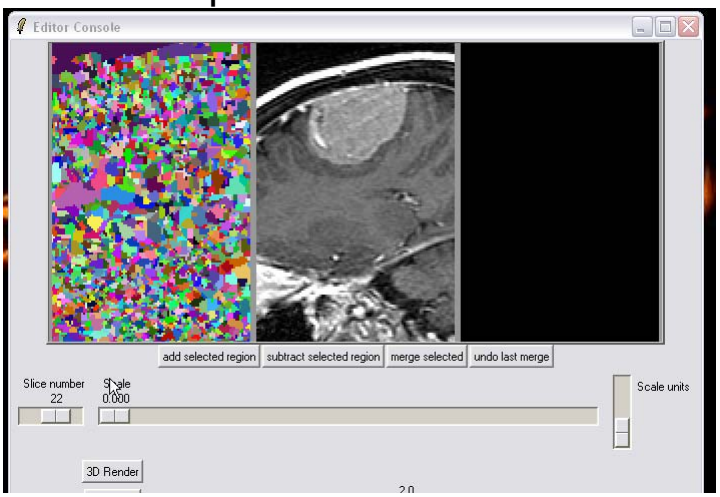


Example: Watersheds GUI

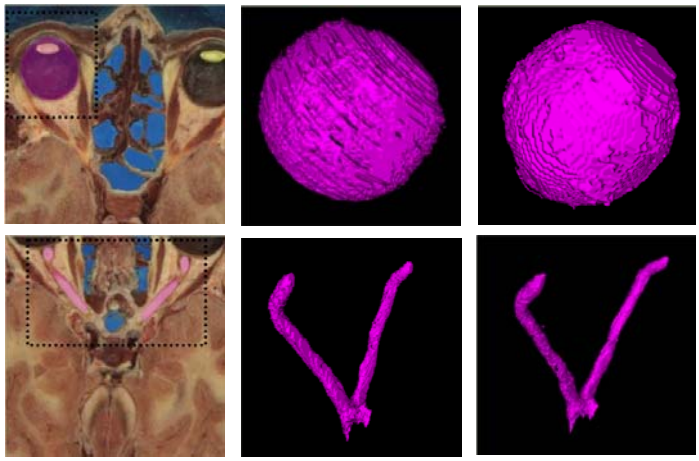
InsightApplications/SegmentationEditor



Example: Watersheds GUI

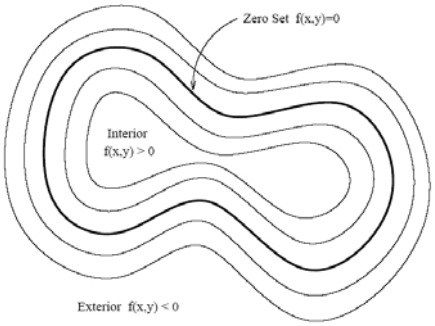


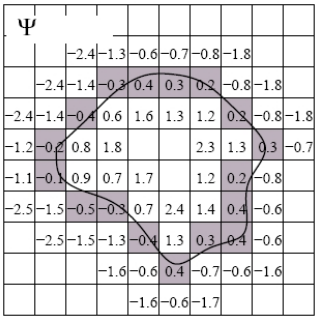
Example: Watersheds GUI



LevelSet Surface Modeling Theory

- k^{th} Level Set: set of all points of value k
- Embed N dimensional surface as ZERO level set of N+1 dim. volume
- Model N dim. surface movement as evolving wavefront – forward differences solution to PDE





Segmentation Using Level Sets

- Define speed term(s) to go to zero at edges – data fitting term
- Surface motion/speed based on intensity-based features
- Solve the level-set equation where

$$\Gamma((\mathbf{X}),\mathbf{t}) = \{\psi(\mathbf{X},\mathbf{t}) = \mathbf{0}\}$$

$$\frac{d}{dt}\psi = -\alpha\mathbf{A}(\mathbf{x})\cdot\nabla\psi - \beta P(\mathbf{x}) \mid \nabla\psi \mid + \gamma Z(\mathbf{x})\kappa \mid \nabla\psi \mid$$

PDE Solver Framework

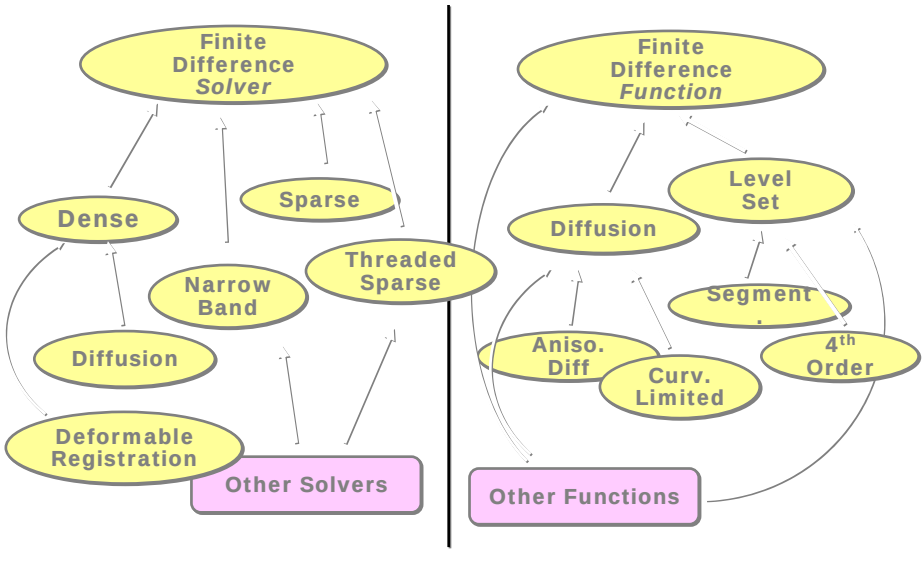
Purpose

- Nonlinear image processing – e.g. anisotropic diffusion
- Moving wave fronts – level set models
- Deformable registration

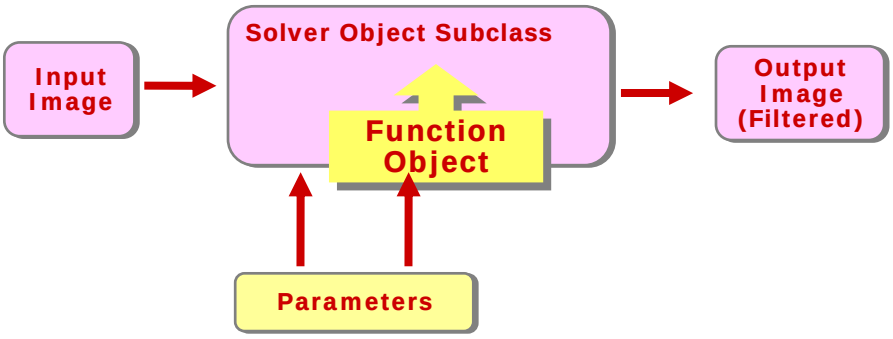
Generic framework

- Separate solvers from equations – *interchangeable* code objects

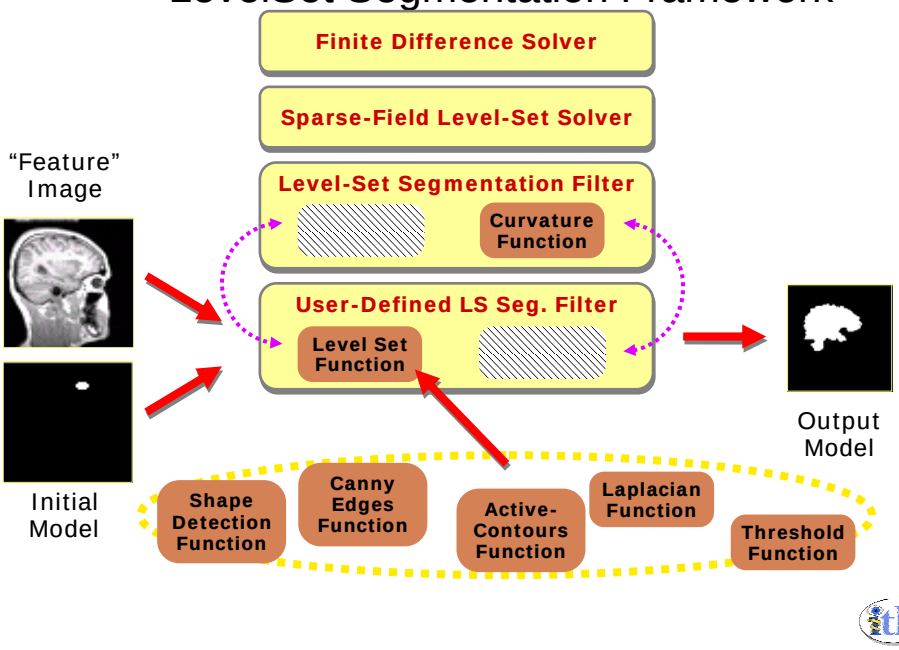
PDE Solver Hierarchy



Constructing a PDE Filter



LevelSet Segmentation Framework



LevelSet Segmentation Algorithms in ITK

Fast marching

Geodesic active contours

CURVES (vessel segmentation)

Intensity interval (scalar and vector)

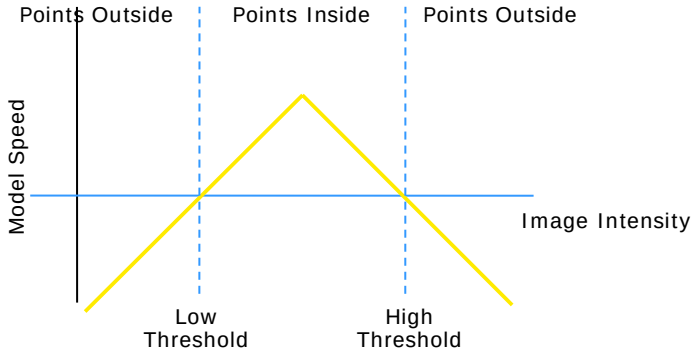
Canny edge distance

Laplacian edges

… and more.

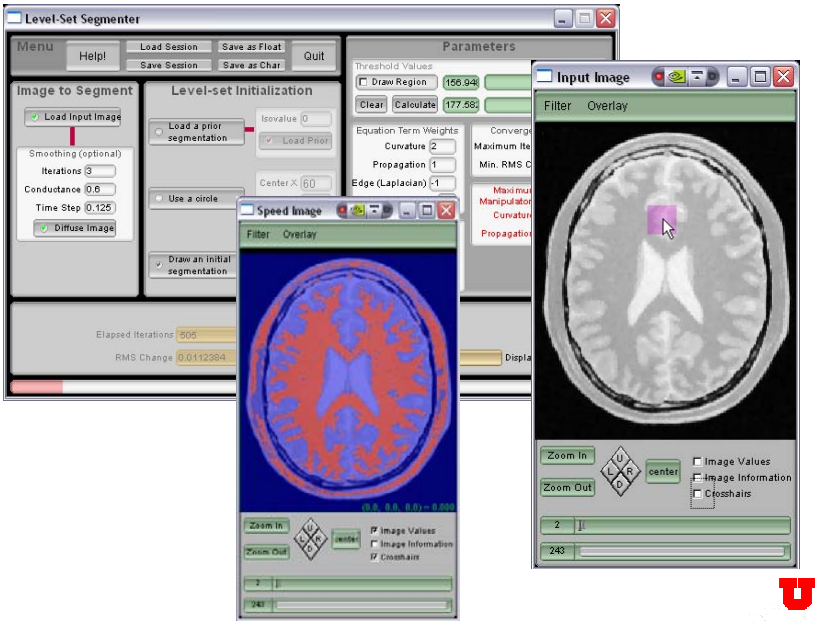
Example: Threshold based LS Segmentation

Speed function (positive inside object)
Similar to confidence connected filter

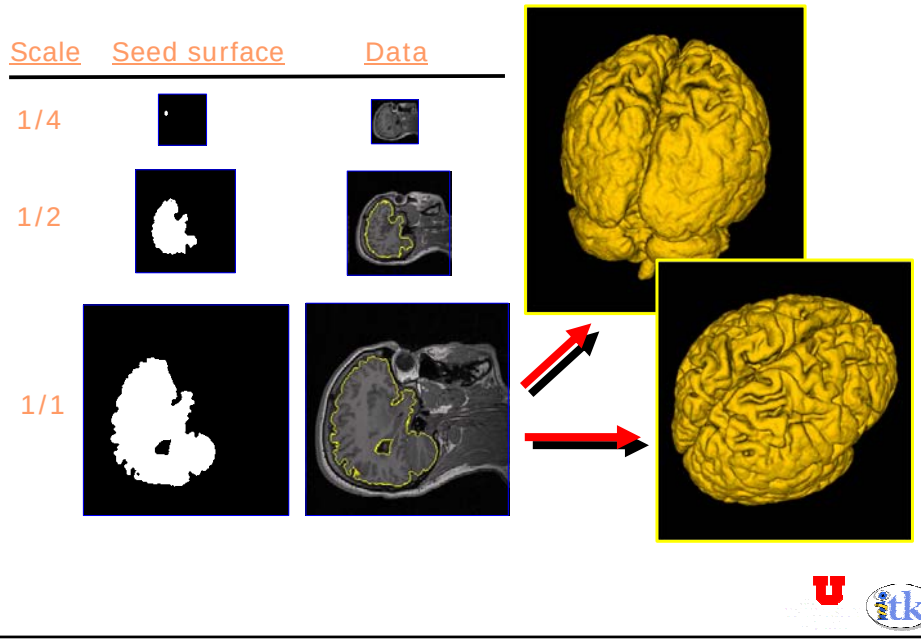


- Lefohn, Kniss, Hansen, Whitaker, "GPU-Based, Interactive, Level-Set Models for Volume Visualization and Analysis", IEEE Vis 2003
- Lefohn, Whitaker, Cates, "Interactive Level-Set Models for Brain Tumor Segmentation", MICCAI 2003

Example: LevelSet Segmentation GUI

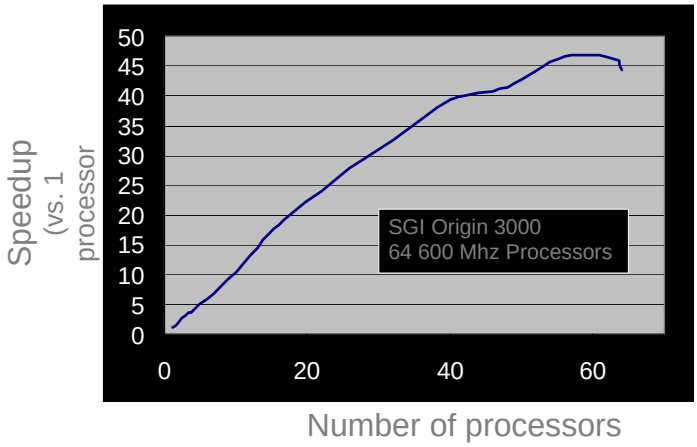


Multiscale LevelSet 3D Segmentation

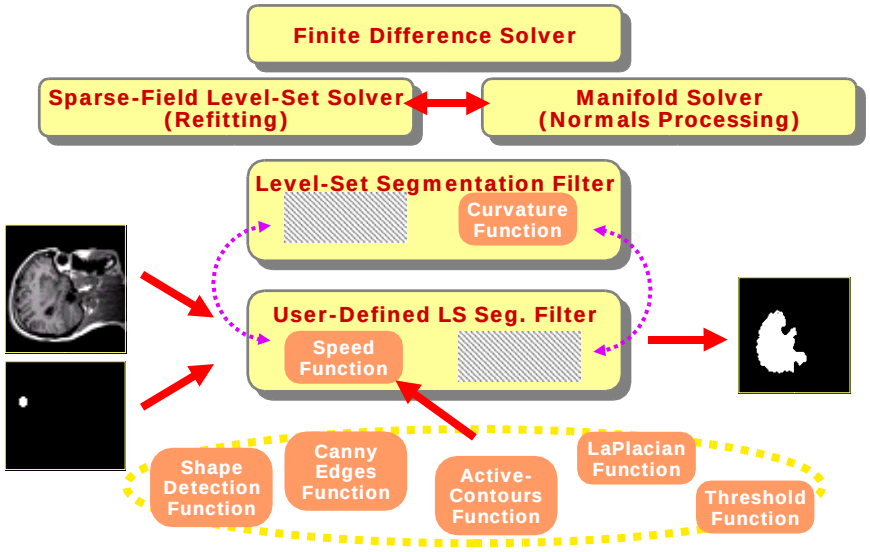


Advanced Features in the PDE Framework

Parallel Solvers – Narrowband,
Sparse field

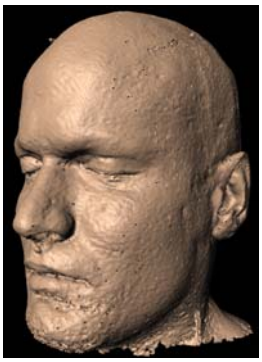


4th Order Flow Segmentation Framework

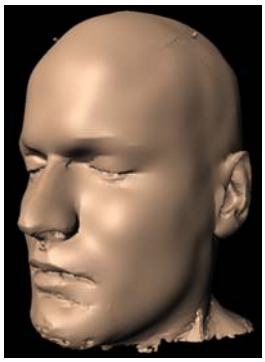


Segmentation Using 4th Order Flows

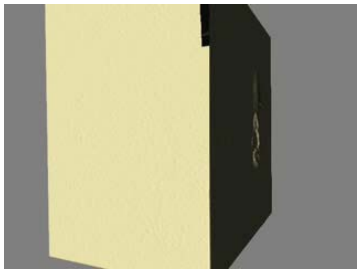
Special 4th order solver plugs into LS segmentation framework – no change in function objects.



Speed term only



Speed + Anisotropic 4th order terms



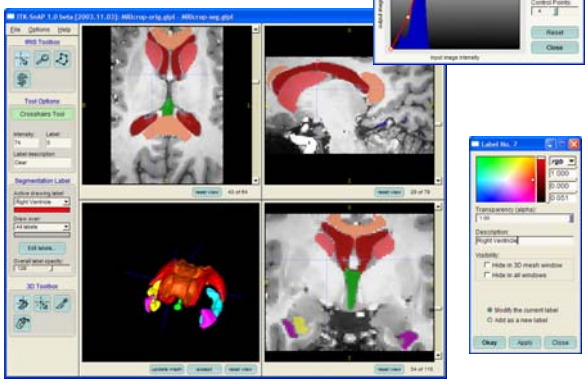
(not real-time)

SNAP Tool

Aimed at clinical users – easy to learn and use

Implements various ITK level set algorithms

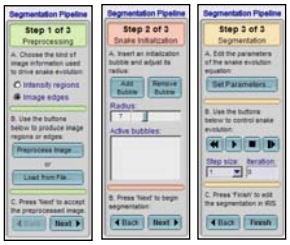
Implements both edge-based and region competition geodesic snake methodology



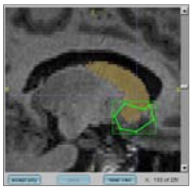
InsightApplications/Snap

SNAP User Interface

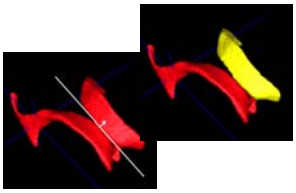
SNAP Segmentation of the Caudate Nuclei



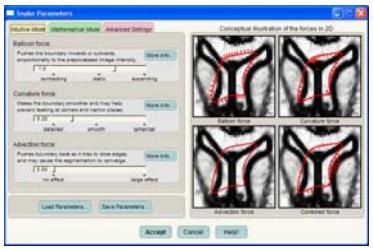
SNAP Segmentation Wizard with VCR Controls



Manual Segmentation



Scalpel tool for editing segmentations



Dialog for setting evolution parameters

InsightApplications/Snap

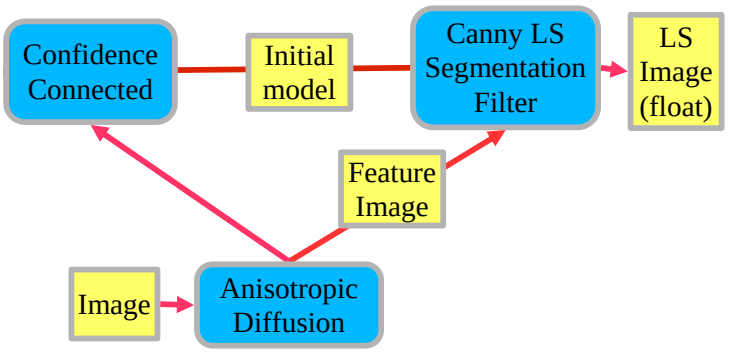
"Hybrid" Segmentation Methods

- Apply several algorithms in sequence
- Utilize strengths of each
  - Speed / accuracy tradeoffs — e.g. connected component vs. level-sets
  - Maximize use of information in data — e.g. region based plus boundary based

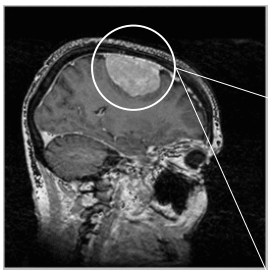
Hybrid Method: Region Growing + Level Sets

Generate initial model using confidence connected filter

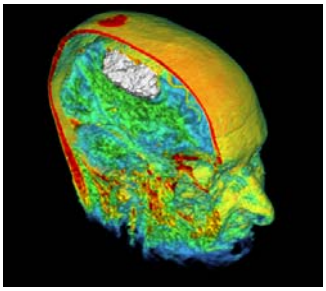
Fit to data using level-set methods – minimize distance to Canny edges

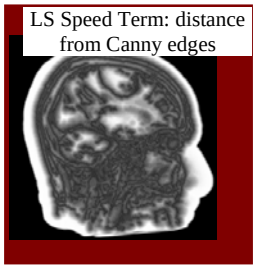


Confidence Connected + LevelSet Result

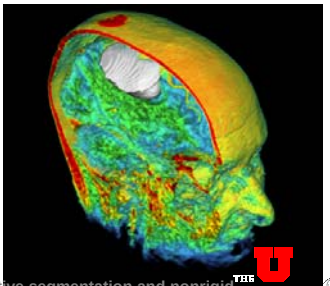


Initial confidence-connected result





Post-processing with Canny LS segmenter



Data: Warfield, Nabavi, Butz, Tuncali, Silverman, "Intraoperative segmentation and nonrigid registration for image guided therapy, in: MICCAI'2000, SpringerVerlag, 2000, pp.176-185.

enjoy ITK!

http://www.itk.org

Overview: ITK Registration Methods

Lydia Ng
Allen Institute for Brain Science

Overview

- Introduction
- ITK Registration Framework
  - Component Overview
  - Registration Strategies
  - GUI Communication
- Deformable Registration in ITK
- Resources
- Application: Allen Brain Atlas

Introduction

- Image registration
  - Process of finding the spatial transform that maps all points from one image to their homologous points in another image
- Medical applications
  - Monitoring change in an individual
  - Fuse information from multiple sources to aid clinical interpretation
  - Compare one subject to another

Intra-subject Registration

- Repeated image scans of the same subject can be used to capture the effect of disease development, treatment progress, contrast bolus propagation etc.
- Registration can be used to compensate for differences in patient placement
- Subtraction/overlay of registered images can be used for visualization and quantification

3D Breast MR Contrast Uptake Deformable Registration

Image data courtesy of University of Washington.

Unregistered:

Registered:

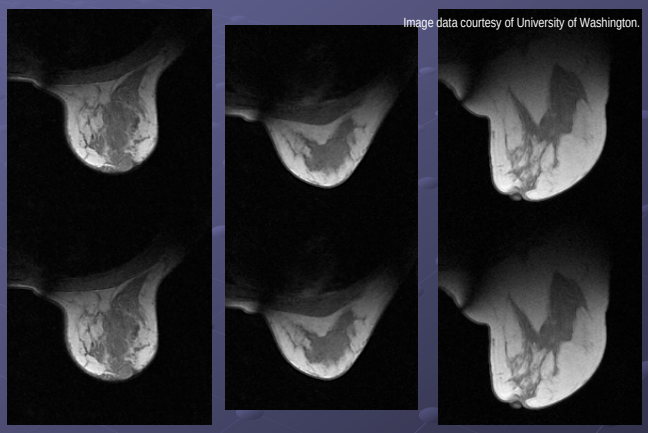


Image Data: 192 x 192 x 13 pixels, 0.94 x 0.94 x 8.00 mm
Registration: 2 levels, MI, LBFGS, B-spline deformation

Inter-subject Registration

- Creation of mean images and atlases
  - Useful for objective and quantitative assessment of abnormalities
  - Computed deformation field used to encode pattern of anatomic variability within a population
- Atlas based segmentation
  - Automatic segmentation by mapping to a labeled atlas

3D Inter-subject MR-PD Deformable Registration

Image data courtesy of N. C. Andreasen, University of Iowa, Psychiatry Dept.

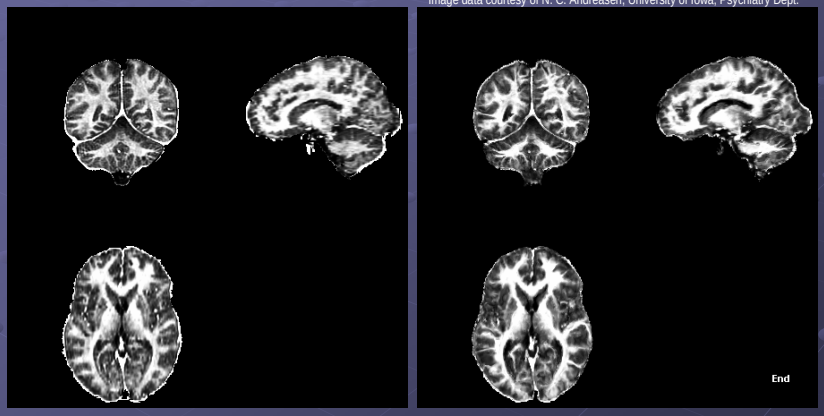
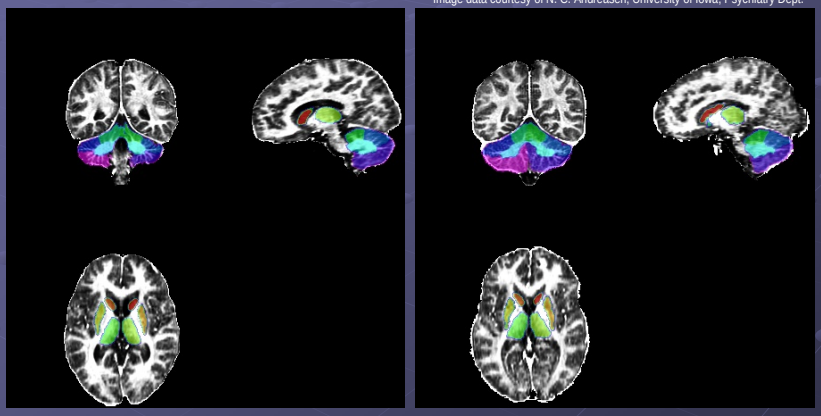


Image Data: 256 x 256 x 192 pixels, 1.02 x 1.02 x 1.02 mm
Registration: 3 levels, Demons algorithm, histogram matching

3D Atlas-based Segmentation

Image data courtesy of N. C. Andreasen, University of Iowa, Psychiatry Dept.



Multi-modality Registration

- Combine or fuse information from multiple sources to aid clinical interpretation
- Some modalities
  - MR: soft tissue discrimination; lesion identification
  - CT: bone localization; surgical guidance
  - PET/SPECT: functional information; tumor localization

PET/CT Fusion

Image data courtesy of University of Washington.

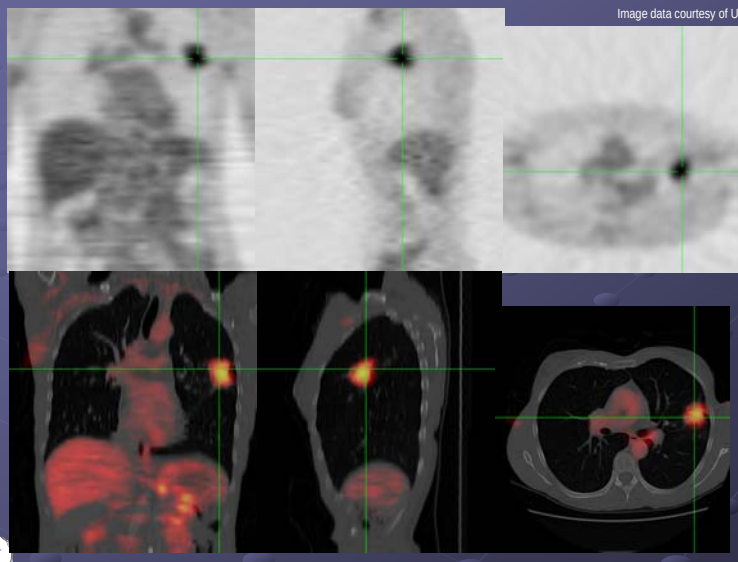


Image Registration Classification

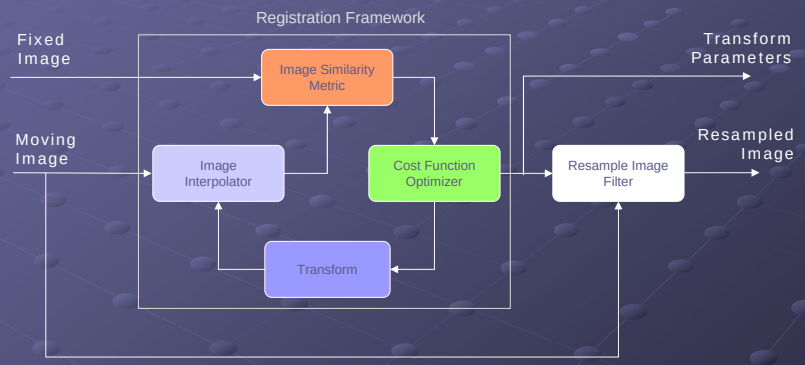
- Registration criteria
  - Quantitative measure of a “good match”
  - Focus on intensity based measures
- Spatial transform type
  - Allowable mapping from one image to another
  - Rigid versus non-rigid
- Optimization algorithm used
  - Optimize transform parameters with respect to match criteria
- Image interpolation method
  - Value of image at non-grid position

ITK Registration Framework

ITK Registration Framework

- Generic framework for building intensity based registration algorithms
- Each functionality encapsulated as components
- Components are inter-changeable allowing a combinatorial variety of registration methods
- Components are generic
  - Can be used outside the registration framework

Registration Framework Components



itk::ImageRegistrationMethod

- Simple helper driver class that connects up the components and starts the optimizer
- Examples in Software Guide

Examples/Registration

- Declare types
- Instantiate objects via New()
- Connect components and images to the driver using Set methods
- Set initial transform parameters
  - Don't forget!
- Setup each component
  - optimization parameters: step length, convergence …
  - MI parameters: number of samples,
- Connect up observers
- StartRegistration()
  - Should do this inside try/catch block
- Get the last transform parameters
- Create registered image using ResampledImageFilter

Component Overview

- Transforms
  - Forward versus inverse mapping
  - Rigid and non-rigid
- Interpolators
- Image to image similarity metrics
  - Intra-modality
  - Inter-modality (information-theoretic methods)
- (Optimizers)

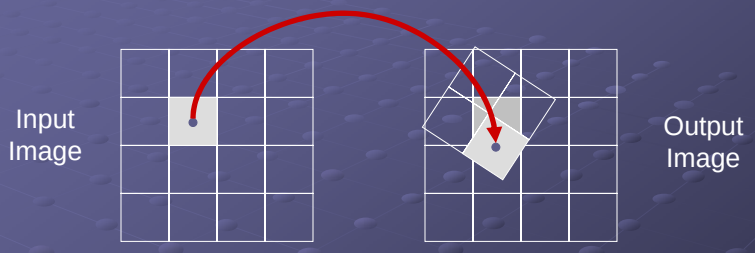
itk::Transfrom

- Encapsulates the mapping of points and vectors from an “input” space to an “output” space
- ITK provides a variety of transforms from simple translation, rotation and scaling to general affine, kernel and B-spline transfoms
- Forward versus inverse mapping
- Parameters representation
- “Transform Jacobians”

Forward and Inverse Mappings

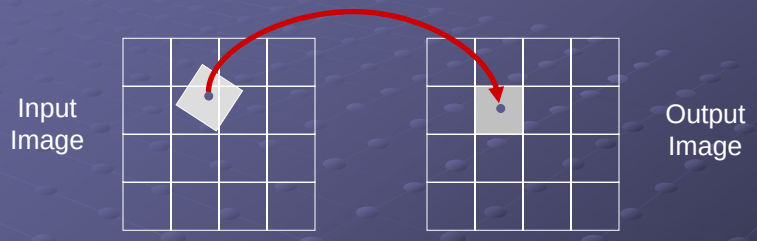
- Relationship between points of two images can be expressed in two ways:
  - Forward: pixel of input (moving) image mapped onto the output (fixed) image
  - Inverse: output (fixed image) pixels are mapped back onto the input (moving) image

Forward Mapping



- Input image pixel is mapped onto the output image
- Output pixels with more than one hit: overlap
  - Value must be accumulated from overlapping pixels
- Output pixels with no hits: hole

Inverse Mapping



- Output pixels are mapped back onto the input image
- Output pixel value must be interpolated from a neighborhood in the input image
- Scheme avoids any holes and overlaps in the output image because all pixels are scanned sequentially

Registration and Inverse Mapping

- The registration framework uses **inverse** mapping
- The transform component maps points from the **fixed** image space to the **moving** image space

$$\mathbf{x}' = \mathbf{T}(\mathbf{x}|\mathbf{p})$$

Point in moving image space

Point in fixed image space

Transform Parameters

- Each transform class is defined by a set of parameters
  - Translation, angle of rotation etc
- Represented as a flat array of doubles to facilitate communication with generic ITK Optimizers
- Transform parameters define the search space for the optimizer
- Goal of registration
  - Find the set of transform parameters that result in the best value of an image similarity metric

Transform Jacobian

- Some metrics require the knowledge of the "transform Jacobian" in order to compute metric derivatives
- The "transform Jacobian" is a matrix whose elements are the partial derivatives of the output point with respect to the transform parameters

$$J_{ij} = \frac{\partial x_i'}{\partial p_j}$$

itk::TranslationTransform

- Maps all points by adding a constant vector:

$$\mathbf{x}' = \mathbf{x} + \mathbf{t}$$

- Parameters:
  - i-th parameter represent the translation in the i-th dimension $\mathbf{p} = \mathbf{t}$
- Jacobian in 2D:

$$\mathbf{J} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

itk::Euler2DTranform

- Represents a rotation and translation in 2D

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

- Parameters: $\mathbf{p} = \{\theta, t_x, t_y\}$
- Jacobian:

$$\mathbf{J} = \begin{bmatrix} -(\sin\theta)x - (\cos\theta)y & 1 & 0 \\ +(\cos\theta)x - (\sin\theta)y & 0 & 1 \end{bmatrix}$$

WARNING Don't forget TRANSFORM CENTERING and PARAMETER SCALING!

Center of Transformation/Rotation

- Transformation/rotation is about the origin {0,0}
- Can represent rotation at arbitrary rotation center but at the expense of optimization performance, especially for large auxiliary translation

$$\mathbf{x}' - \mathbf{c}_m = \mathbf{M}(\mathbf{x} - \mathbf{c}_f) + \mathbf{t}$$
$$\mathbf{x}' = \mathbf{M}\mathbf{x} + \underbrace{(\mathbf{t} + \mathbf{c}_m - \mathbf{M}\mathbf{c}_f)}_{\mathbf{t}'}$$

- Alternatives:
  - center of image
  - center of mass of an object

Parameter Scaling

- Different type of parameters (translation vs. angle vs. scaling) have different dynamic ranges
  - E.g. a unit change in rotation has a much larger impact than a unit change in translation
- Differences in scale appears as long narrow valleys in the search space making optimization difficult
- Rescaling the parameters can help fix this problem

itk::Euler3DTranform

- Represents 3D rotation and translation
  - Rotation about each coordinate axis
  - Ordering matters!

$$\mathbf{x}' = \mathbf{R}_Z \mathbf{R}_Y \mathbf{R}_X \mathbf{x} + \mathbf{t}$$

or

$$\mathbf{x}' = \mathbf{R}_Z \mathbf{R}_X \mathbf{R}_Y \mathbf{x} + \mathbf{t}$$

- Parameters:

$$\mathbf{p} = \{\theta_x, \theta_y, \theta_z, t_x, t_y, t_z\}$$

Alternative 3D Rigid Transforms

- itk::QuaternionRigidTransform
  - 3D rotation represented by a quaternion
  - Represented by 4 numbers
  - Similar to axis/angle representation
  - Unit quaternion is equivalent to pure rigid
  - Does not suffer from “Gimbal lock”
- itk::VersorRigidTransform
  - Strictly the rotational part of a quaternion (always rigid)
  - Represented by 3 numbers
- Both quaternion and versor components do not form vector spaces
  - Specialized optimizers required!

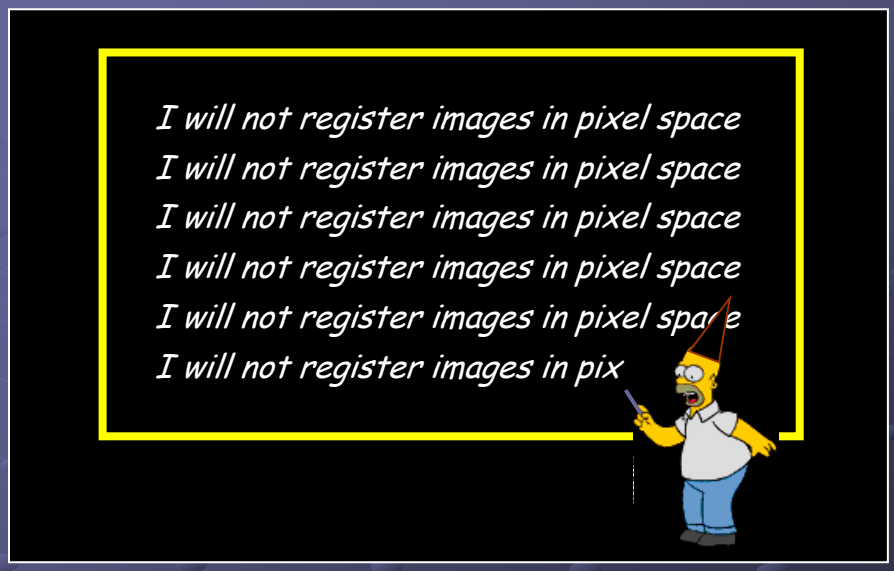


Image Spacing and Origin

- Medical image volume are typically anisotropic
  - In-plane pixel size smaller than inter-slice spacing
- A transform is rigid only with respect to physical coordinates and not pixel coordinates
  - PhysCoord = PixelCoord x ImageSpacing + ImageOrigin
- The registration is always with respect to physical coordinates
- Make sure spacing and origin information is set correctly in the images!

itk::AffineTransform

- General affine transform can represent rotation, scaling, shearing and translation
- Parameters: matrix coefficient + translation (6 in 2D, 12 in 3D)
- Parallel lines are preserved

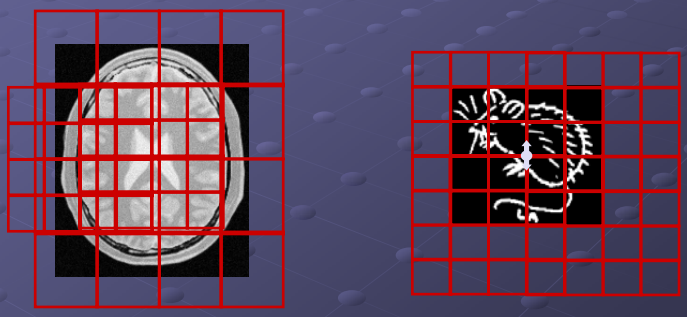
Don't forget TRANSFORM CENTERING and PARAMETER SCALING!

itk::BSplineDeformableTransform

- Represent a low-dimensional smooth deformable warp
- Deformation field represented by B-splines coefficient on a regular grid
- Parameters: B-spline coefficient for each dimension at each grid position

B-Spline Grid Placement

- Grid defined by origin, spacing, size



SplineOrder = 3

itk::InterpolateImageFunction

- When a point is mapped from one image space to another image space, it will generally be mapped to a non-grid position
- Interpolation is needed to compute the intensity value at the mapped position

Choice of Interpolation Method

- Interpolation affects smoothness of metric space
- Interpolations computed 1000's of times in a single optimization cycle
- Trade-off efficiency with ease of optimization

Interpolation Schemes

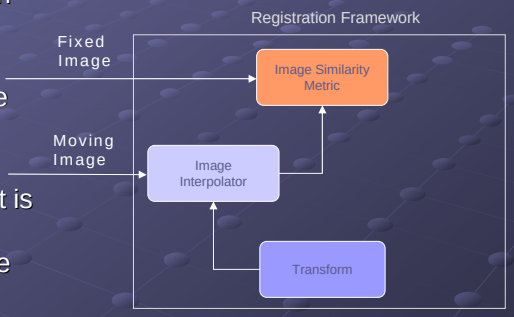
- itk::NearestNeighborInterpolateFunction
  - Assumes image is piecewise constant
- itk::LinearInterpolateFunction
  - Assumes image is piecewise linear
- itk::BSplineInterpolateFunction
  - Underlying image represented using B-spline basis functions
  - On connection, image of B-spline coefficients is computed

itk::ImageToImageMetric

- Measures how well a transformed moving image “matches” the fixed image
- The most critical component
- Scalar function of the transform parameters
  - For given, fixed image, moving image, transformation type and interpolation type

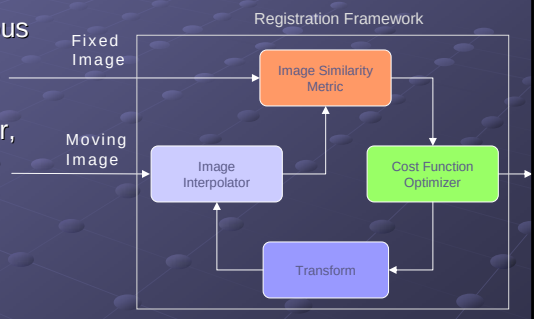
Metrics, Transforms and Interpolators

- The **metric** typically samples points within a defined region of the fixed image
- For each point, the corresponding moving image point is obtained using the **transform** component
- The **interpolator** component is then used to compute the moving image intensity at the mapped position



Metrics and Optimizers

- The **metric** is used by the **optimizer** to evaluate the quantitative criterion at various positions in the transform parameter search space
- For gradient-based optimizer, the metric must also provide the metric derivatives w.r.t each transform parameter
  - Use chain rule with moving image derivatives and transform Jacobian



Choice of Metric

- Highly dependent on the registration problem to be solved
  - Single-modality
  - Multi-modality
  - Large capture range
  - Requires close initialization
- Examples:
  - Mean squares, normalized correlation and mutual information

Mean Squares Metric

Interpolate the moving image

$$S(\mathbf{p}|F,M,\mathbf{T}) = \frac{1}{N}\sum_{i}^{N}\left(F(\mathbf{x}_i) - M(\mathbf{x}'_i)\right)^2$$

where

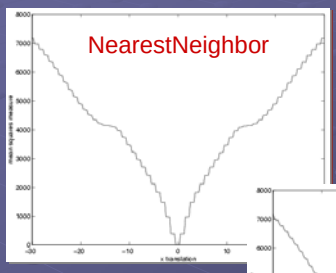
$$\mathbf{x}'_i = \mathbf{T}(\mathbf{x}_i, \mathbf{p})$$

Over a user specified fixed image region

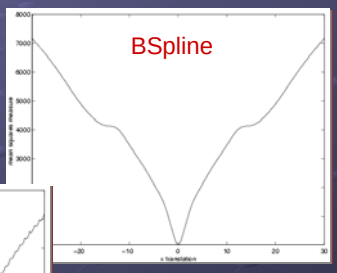
Transform from fixed image point to moving image point

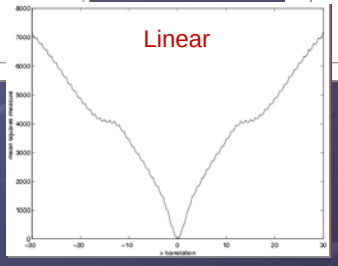
- Simple to compute
- Large capture range
- Restricted to mono-modality applications
- Linear differences in intensity results in poor similarity measure

Mean Squares Metric

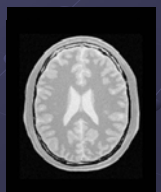


Optimal Value at Zero





Metric range: image dependent



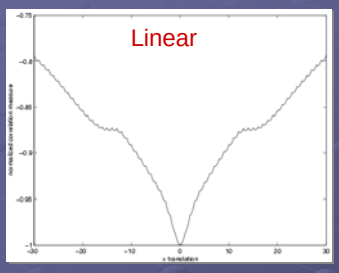
Translations

Normalized Correlation Metric

$$S(\mathbf{p}|F,M,\mathbf{T}) = -1 \times \frac{\sum_i^N F(\mathbf{x}_i) M(\mathbf{x}'_i)}{\sqrt{\sum_i^N F^2(\mathbf{x}_i) \sum_i^N M^2(\mathbf{x}'_i)}}$$

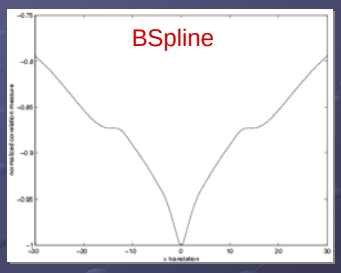
- Pixel-wise cross-correlation between fixed and moving image intensity
- Factor -1 added to work with minimum seeking generic optimizers
- Insensitive to multiplicative difference

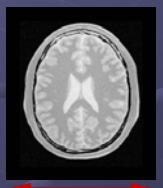
Normalized Correlation Metric



Optimal Value at -1

Metric range: 1 to -1





Translations

Mutual Information Metric

- Information theoretic entity that qualitatively measures how much information is gained about one RV (intensity in one image) by the knowledge of another RV (intensity in another image)

Mutual Information Metric

- Introduced in the context of multi-modality registration by two different groups: Viola and Wells (1997) and Collignon et al. (1995)
- MI well suited since actual form of dependency between the two RVs does not have to be specified
- MI defined in terms of entropy
  - Measure of information contained in a piece of data

Entropy

- Two RVs: A and B
- Marginal entropies:

$$H(A) = -\int p_A(a) \log p_A(a)\, da$$

$$H(B) = -\int p_B(b) \log p_B(b)\, db$$

- Joint entropy:

$$H(A,B) = -\int p_{AB}(a,b) \log p_{AB}(a,b)\, da\, db$$

Mutual Information

- If A and B independent:

$$H(A,B) = H(A) + H(B)$$

- If A and B not independent:

$$H(A,B) < H(A) + H(B)$$

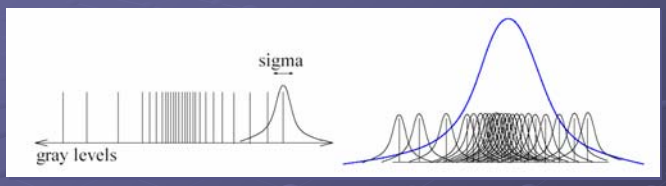
- The difference is MI:

$$I(A,B) = H(A) + H(B) - H(A,B)$$

Estimating the Probabilistic Models

- Entropy is defined in context of a probabilistic model of the data source
  - Typically there is no direct access to these models
- Marginal and joint densities has to be estimated from the image data
  - Parzen windowing
  - Kernel density estimate
  - Histogram binning

Parzen Windowing



- Density function is constructed by superimposing kernel functions centered on the intensity samples obtained from the image
- Kernel type
  - Gaussian, boxcar, B-Spline
- Kernel width (crucial)
  - Depend on dynamic range of data
  - Normalize data
- Number of samples

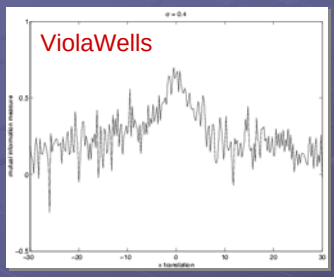
Estimating Entropy

- Using the density estimate, approximate entropy integral by a sum
  - Evaluate at discrete positions/bins uniformly spread within dynamic range
  - Computing a sampled mean using another set of intensity samples randomly drawn from image

Flavors of Mutual Information

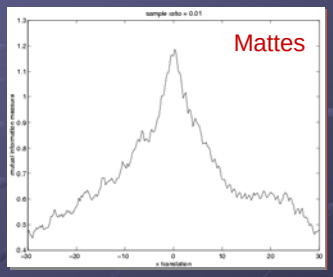
- itk::MutualInformationImageToImageMetric
  - Viola and Wells
  - New samples (50-100) each iteration
- itk::MattesMutualInformationImageToImageMetric
  - Mattes et al, Thevenaz and Unser
  - One set of samples (5-10% of image)
- itk::MutualInformationHistogramImageToImageMetric
  - Use all pixels

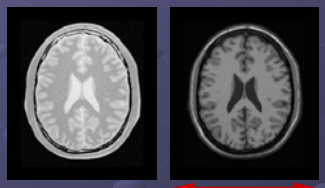
Mutual Information Metric



Optimal Value at maximum

Metric range: Image dependent



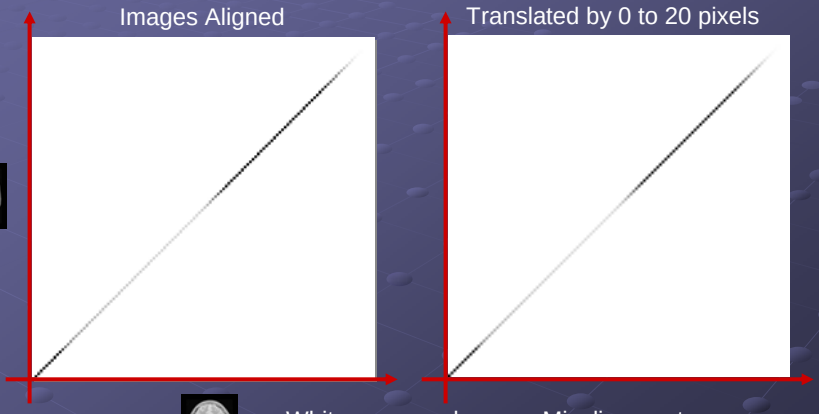


Translations

Joint Histograms: Mono-modality

Images Aligned

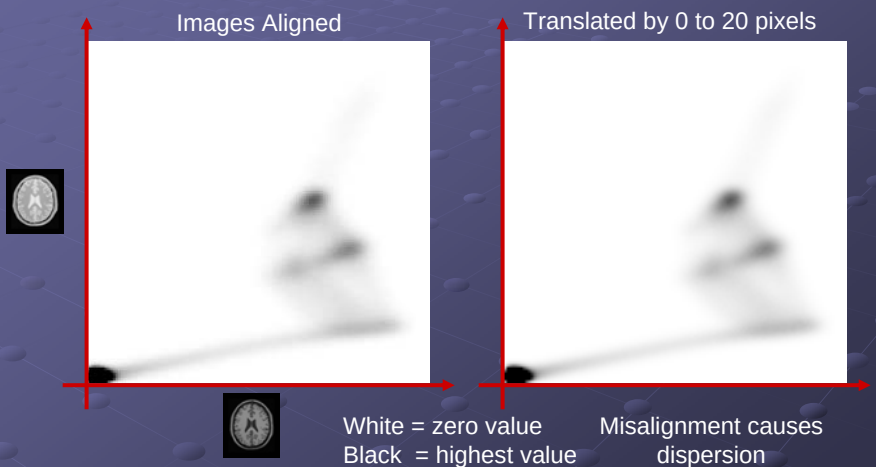
Translated by 0 to 20 pixels



White = zero value
Black = highest value

Misalignment causes dispersion

Joint Histograms: Multi-modality



Joint Histograms and Registration

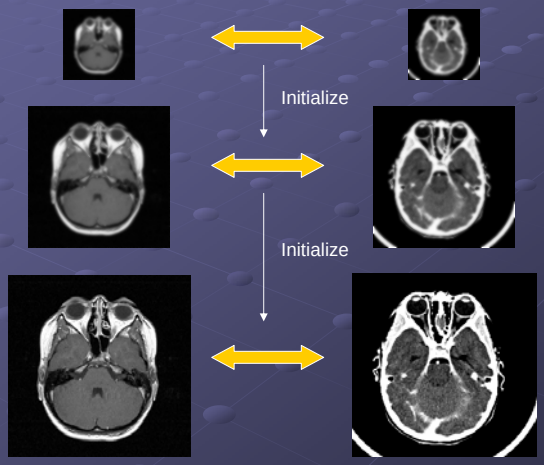
- Seek transform that produces a small number of high value bins and as many zero bins as possible == minimizing joint entropy
- Joint entropy metric favors transforms which causes the images to be far apart as possible (i.e. minimum overlap)
- MI overcomes this problem by also trying to maximize the information contributed by each image in the overlap region

Registration Strategies

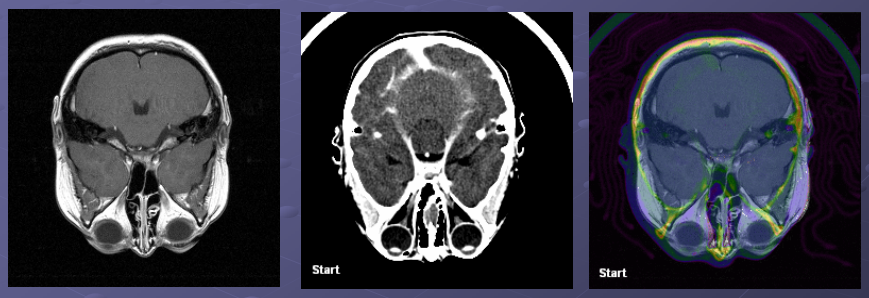
- Naive application of registration may not work
- Coarse to fine strategy
  - Improves computational speed and robustness
- Low to high dimensional transform
  - Translation to rigid to deformable
  - Sparse to fine grid

itk::MultiResolutionImageRegistrationMethod

- Helper driver class to perform multi-resolution registration



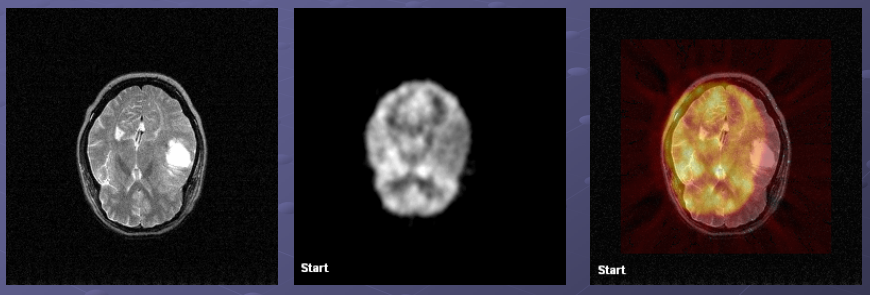
3D CT to MR-T1 Rigid Registration



Fixed Image: MR-T1, 256 x 256 x 52 pixels, 0.78 x 0.78 x 3.00 mm
Moving Image: CT, 512 x 512 x 44, 0.41 x 0.41 x 3.00 mm
Registration: 4 levels, MI, gradient descent, quaternion rigid

Images provided as part of the project: "Retrospective Image Registration Evaluation",
NIH, Project No. 8R01EB002124-03, Principal Investigator, J. Michael Fitzpatrick, Vanderbilt University, Nashville, TN.

3D PET to MR-T2 Rigid Registration



Fixed Image: MR-T2, 256 x 256 x 26 pixels, 1.25 x 1.25 x 4.00 mm
Moving Image: PET, 128 x 128 x 15, 1.94 x 1.94 x 8.00 mm
Registration: 3 levels, MI, gradient descent, quaternion rigid

Images provided as part of the project: "Retrospective Image Registration Evaluation",
NIH, Project No. 8R01EB002124-03, Principal Investigator, J. Michael Fitzpatrick, Vanderbilt University, Nashville, TN.

Registration To GUI Communication

- Drive progress bar
- Observe registration progress
  - Interrogate metric values
  - Interrogate current parameters
  - Display current results
    - Resample and display ROI
- Change component parameters
- Terminate registration

ITK Observers/Commands

- itk::Object can invoke events
  - Start, End, Progress, Iteration
- itk::Object maintains a linked list of event observers
- Observers register themselves to an object and declare the type of events they are interested in

Observing Registration

- ITK Optimizers typically execute an iterative process
- Most Optimizers invoke an IterationEvent at the end of each iteration
- Observing IterationEvents provide periodic communication facilitating progress feedback and opportunity to control the registration
- itk::MultiResolutionImageRegistrationMethod also invokes an IterationEvent between levels to give the GUI an opportunity to change components and/or parameters

ITK Registration Examples

- ITK Software Guide
  - 9+ examples
- InsightApplications
  - ImageRegistration
  - MRIRegistration
  - MultiResMIRegistration
  - MutualInformationEuler2DRegistration
- Landmark Initialized 3D Registration Tool
  - CADD Lab, UNC

Deformable Registration in ITK

- Low dimensional transformations can be handled in the basic registration framework
- For high dimensional transformations:
  - Finite Element Methods (FEM)
  - Finite Different Methods
- Metric optimization
  - Mean squares, normalized correlation, MI
- FEM framework additionally supports regularization and landmark constraints:
  - Diffeomorphic constraints
  - Linear elastic and large deformation (fluid and transient-quadratic) models
- Simple examples in Software Guide

Resources

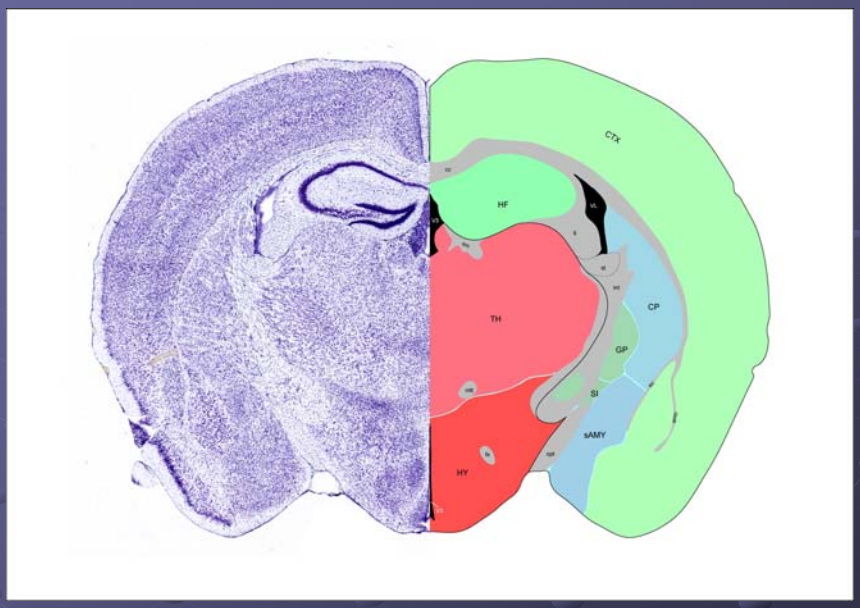
- ITK Software Guide
- "Insight into Images",
  - Yoo (ed), 2004
- "A Survey of Medical Image Registration"
  - Maintz and Viergever, Medical Image Analysis, 1998
- "Handbook of Medical Imaging: Medical Image Processing and Analysis"
  - Fitzpatrick et al (eds), SPIE, 2000
- "Handbook of Medical Imaging: Processing and Analysis"
  - Bankman (ed.), Academic Press, 2000
- "Medical Image Registration"
  - Hajnal et al (eds), CRC Press, 2001
- "Mutual-Information-Based Registration of Medical Images: A Survey"
  - Pluim et al, IEEE-TMI, 22(8), 2003
- IEEE Transaction of Medical Imaging Special Issue
  - November 2003

Data for Registration Testing

- Retrospective Registration Project
  - http://www.vuse.vanderbilt.edu/~image/registration/
  - Multi-modality images of the brain
- Internet Brain Segmentation Repository
  - http://www.cma.mgh.harvard.edu/ibsr/
  - MR-TI images of brain with segmentations
- BrainWeb
  - http://www.bic.mni.mcgill.ca/brainweb/
  - Simulated brain database
- International Consortium For Brain Mapping (ICBM)
  - http://www.loni.ucla.edu/ICBM/ICBM_Databases.html

Application: Allen Brain Atlas

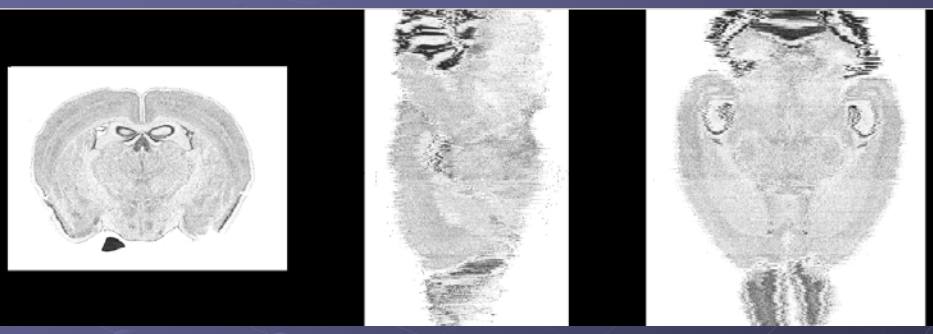
- Detailed, cellular-resolution, genome-wide map of gene expression in the mouse brain
  - http://www.brain-map.org
- The Allen Reference Atlas is being developed to allow comparison of the gene expression data in a common anatomic framework
  - Created from 528 coronal sections from fresh frozen tissue
  - 3D reconstruction is needed to take into account differences between specimens and slicing orientation



3D Atlas Reconstruction

- Inherently 2D data
  - Spatial relationship between sections is lost
  - Artifacts (distortions, tears, smears, inhomogeneity etc) appear independently on per section basis
- Blind application of section-to-section registration could result in non-realistic 3D reconstructions requiring post-processing or constraints

Initialization



- 528 coronal images were converted to grayscale, downsampled, cropped and padded to same size
- Brain tissue mask were created to direct from where registration samples information
- Badly damaged sections were removed

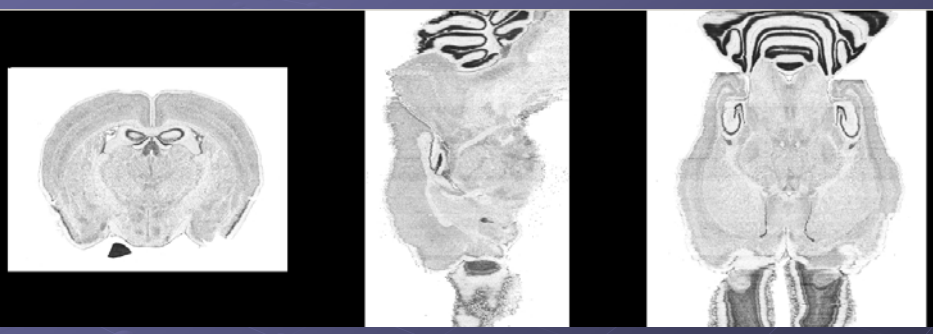
Rigid Registration

- Section to section rigid registration results in a 3D volume which is fairly self-consistent
- Volume appears bent as registration tend to straighten high gradient features (e.g hippocampus, cerebellum)

Unbending Rigid Registration

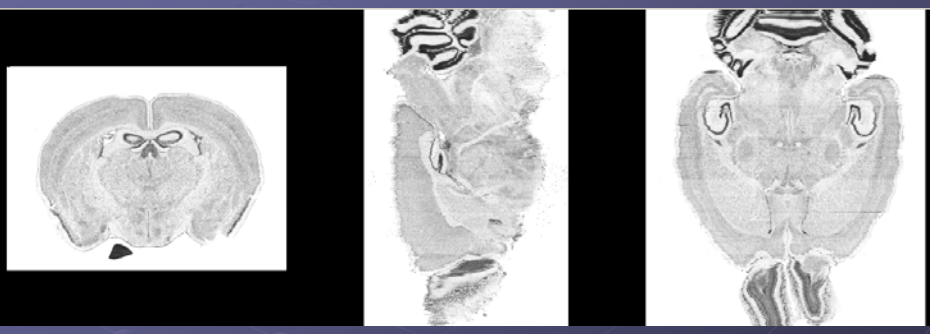
- Unbend the volume by applying the inverse of the smoothed transform series
- Result is then used as initialization for the next step: affine registration

Affine Registration



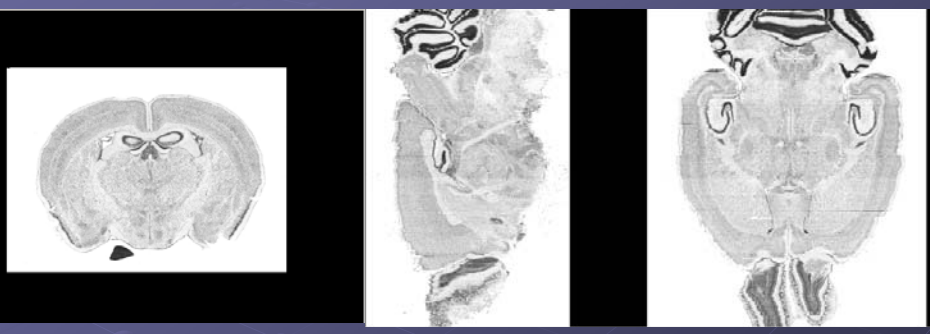
- Section to section affine registration results in very smooth structures but causes severe warping
  - Due to straightening out of the hippocampus and cerebellum

De-warping Affine Registration



- De-warp the volume by applying the inverse of the smoothed transform series
- Result is then used as initialization for the next step: deformable registration

Deformable Registration



- Section to section deformable registration
  - Constraint movement to less than 125 microns
  - Cleaned up jags in hippocampus and cerebellum
  - Appearance of very fine line structures

Hippocampus (Azimuth)

Hippocampus (Elevation)

Cerebellum (Elevation)

Ventricles (Elevation)

ITK Input/Output

Kitware Inc.

Overview

- IO Factory Mechanism
- Image File IO
- Transform File IO
- SpatialObject File IO
- Logger

IO Factory Mechanism

Why do we need a Factory?

How many file formats can you list?

…

(and many more exist…)

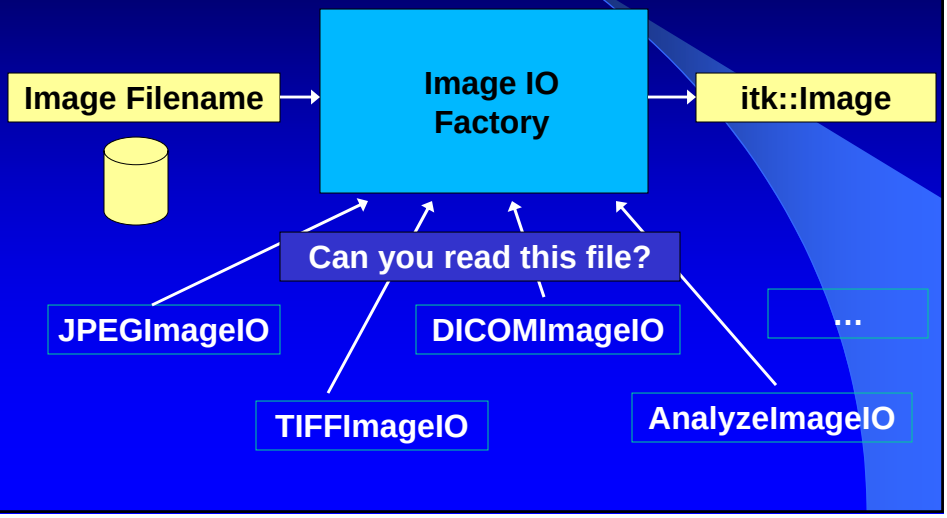
Supported file formats

2D Only
- JPEG (.jpg/.jpeg)
- Bitmap (.bmp)
- PNG (.png)

2D/3D
- Analyze 3.5
- GIPL (.gipl)
- RAW (.raw)
- DICOM
- GE 4x,5x
- IPLCommon
- MetaImage (.mha/.mhd)
- Siemens
- Stimulate
- TIFF
- VTKImage
- NRRD
- LSM
- NIFTI

How does the Factory work?

reading



How does the Factory work?

writing

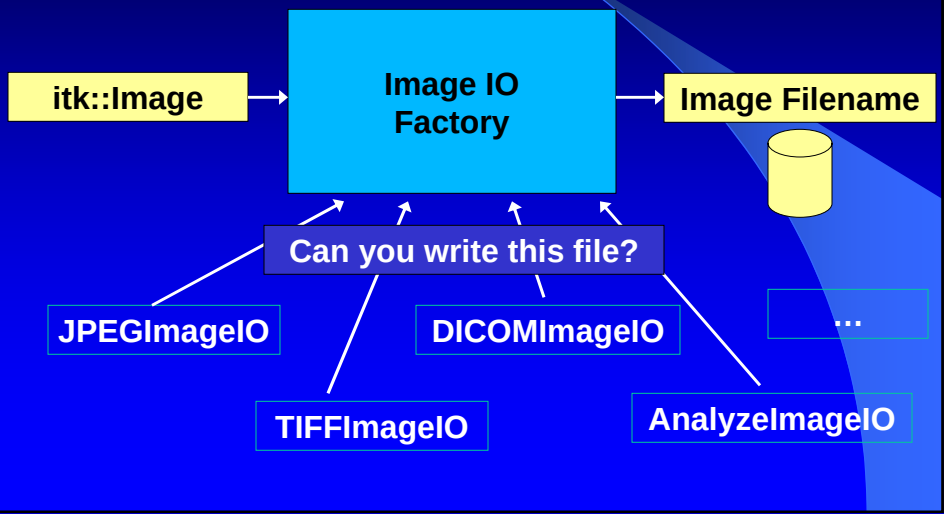


Image File IO

Reading my first image

```
#include <itkImageFileReader.h>

typedef unsigned char PixelType;
typedef itk::Image<PixelType, 2> ImageType;

itk::ImageFileReader<ImageType>::Pointer reader
                        = itk::ImageFileReader<ImageType>::New();

reader->SetFileName("lena.jpg");

try
  {
  reader->Update();
  }
catch (itk::ExceptionObject & e)
  {
  std::cerr << e.GetDescription() << std::endl;
  return EXIT_FAILURE;
  }

ImageType::Pointer image = reader->GetOutput();
```

Writing my first image

```
#include <itkImageFileWriter.h>

typedef unsigned char PixelType;
typedef itk::Image<PixelType, 2> ImageType;

itk::ImageFileWriter<ImageType>::Pointer writer
                        = itk::ImageFileWriter<ImageType>::New();

writer->SetFileName("lena.jpg");
writer->SetInput(image);

try
  {
  writer- >Update();
  }
catch (itk::ExceptionObject & e)
  {
  std::cerr << e.GetDescription() << std::endl;
  return EXIT_FAILURE;
  }
```

My file format converter

```
itk::ImageFileReader<ImageType>::Pointer reader
                        = itk::ImageFileReader<ImageType>::New();

itk::ImageFileWriter<ImageType>::Pointer writer
                        = itk::ImageFileWriter<ImageType>::New();

 reader->SetFileName("myImage.jpg")
 writer->SetFileName("myImage.tiff");
 writer->SetInput(reader->GetOutput());

 try
  {
  writer->Update();
  }
 catch (itk::ExceptionObject & e)
  {
  std::cerr << e.GetDescription() << std::endl;
  return EXIT_FAILURE;
  }
```

How to avoid using the Factory?

- I know the type of images I want to read/write
- Factory can be slow

```
#include <itkTIFFImageIO.h>

itk::TIFFImageIO::Pointer tiffImageIO = itk::TIFFImageIO::New();

reader->SetFilename("myimage.tiff");
reader->SetImageIO(tiffImageIO);
reader->Update();
```

Reading RAW images

```
#include  <itkRawImageIO.h>

itk::RawImageIO<unsigned short,2>::Pointer io;
io = itk::RawImageIO<unsigned short,2>::New();

io->SetFileName("myimage.raw");
unsigned int dim[2] = {800,60};
double spacing[2] = {0.8, 0.8};
double origin[2] = {0.0,0.0};
for(unsigned int i=0; i<2; i++)
  {
  io->SetDimensions(i,dim[i]);
  io->SetSpacing(i,spacing[i]);
  io->SetOrigin(i,origin[i]);
  }
io->SetHeaderSize(0);
io->SetByteOrderToLittleEndian();
io->SetPixelType(itk::ImageIOBase::SCALAR);
io->SetNumberOfComponents(1);
```

Reading RAW images (2)

```
itk::ImageFileReader<ImageType>::Pointer reader
                        = itk::ImageFileReader<ImageType>::New();

reader->SetFileName("myImage.raw");
reader->SetImageIO(io);

try
  {
  reader->Update();
  }
catch (itk::ExceptionObject & e)
  {
  std::cerr << e.GetDescription() << std::endl;
  return EXIT_FAILURE;
  }
```

Creating a MetaImage Header

- Create a simple text file with the .mhd extension
- Set the appropriate MetaData
- Example:

```
NDims = 3
DimSize = 100 200 300
ElementSpacing = 1.2 1.2 1.0
ElementType = MET_UCHAR
ElementByteOrderMSB = False
ElementDataFile = HeadMRVolume.raw
OR
ElementDataFile = HeadMRVolume%04d.raw 0 10 1
```

Dealing with DICOM images

- Often consists of several files
- Uses the GDCM library
- GDCMSeriesFileNames to construct the serie:

1. Image Orientation & Image Position
2. 'Image Number'
3. Lexicographical order

Reading DICOM images

```
// Select the correct files from the directory
typedef itk::ImageSeriesReader< ImageType >        ReaderType;
ReaderType::Pointer reader = ReaderType::New();

typedef itk::GDCMSeriesFileNames NamesGeneratorType;
NamesGeneratorType::Pointer nameGenerator = NamesGeneratorType::New();
nameGenerator->SetUseSeriesDetails( true );

nameGenerator->SetDirectory( "./MyDirectory/" );
typedef std::vector< std::string >   FileNamesContainer;
FileNamesContainer fileNames;
fileNames = nameGenerator->GetFileNames( seriesIdentifier );

reader->SetFileNames( fileNames );

// Set the DicomIO
typedef itk::GDCMImageIO       ImageIOType;
ImageIOType::Pointer dicomIO = ImageIOType::New();
reader->SetImageIO( dicomIO );
reader->Update();
```

More on Series Filenames

- NumericSeriesFilenames
  - ordered sequence of filenames
  - unique, non-negative, integral value
  - Set(Start/End)Index and SetIncrementIndex()
- RegularExpressionSeriesFilenames
  - ordered sequence that matches RegEx
  - ordered by submatch or numerically

Metadata

MetaData

- Extra information attached to an image
- itk::Image defines:
  - Spacing (size of the voxels in mm)
  - Origin (physical location of (0,0,0))
  - Size (size of the largest region of the image)
  - Orientation (direction cosines)

MetaData Dictionary

- itkMetaDataDictionary
- Filled in by the reader (when available)
- Not passed through the filters
- Accessing the dictionary:

```
ImageType::Pointer image = reader->GetOutput();
typedef itk::MetaDataDictionary   DictionaryType;
DictionaryType & dictionary = image->GetMetaDataDictionary();
```

Transform File IO

Transform IO

- Special ITK File format
- Uses IO Factory
- Write Parameters of the transforms as well as Fixed Parameters
- Concatenation of transforms in a single file

Write Transform Example

```
#include "itkTransformFileWriter.h”
#include "itkAffineTransform.h”

typedef itk::AffineTransform<double,3> AffineTransformType;
AffineTransformType::Pointer affine = AffineTransformType::New();
itk::TransformFileWriter::Pointer writer;
writer = itk::TransformFileWriter::New();

writer->AddTransform(affine);
writer->SetFileName( “AffineTransform.txt" );
try
  {
  writer->Update();
  }
 catch( itk::ExceptionObject & excp )
  {
  std::cerr << excp << std::endl;
  return EXIT_FAILURE;
  }
```

Read Transform Example

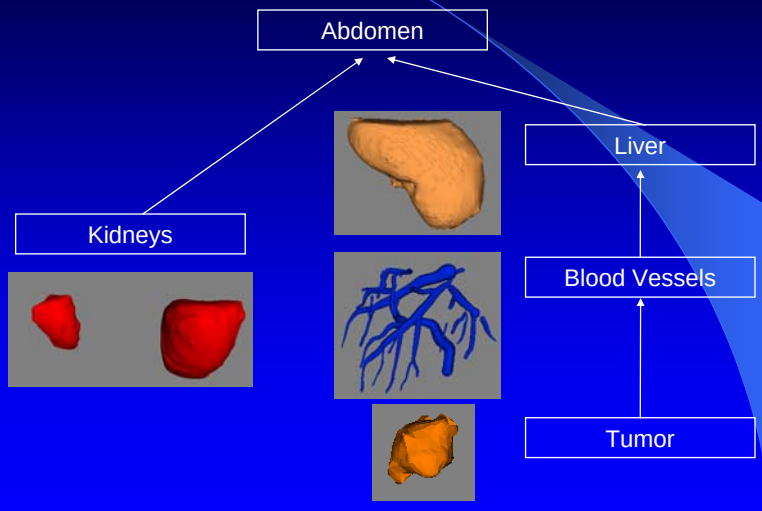
```
#include "itkTransformFileReader.h”

itk::TransformFileReader::Pointer reader;
reader = itk::TransformFileReader::New();

reader->SetFileName( “AffineTransform.txt" );
reader->Update();

typedef itk::TransformFileReader::TransformListType * TransformListType;
TransformListType transforms = reader->GetTransformList();
itk::TransformFileReader::TransformListType::const_iterator it;
it  = transforms->begin();
if(!strcmp((*it)->GetNameOfClass(),"AffineTransform"))
 {
 AffineTransformType::Pointer affineTransform =
          static_cast<AffineTransformType*>((*it).GetPointer());
 }
```

Object File IO

Spatial Objects

- Represents Physical Objects (not only images)
- Scene Graph Concept
- Common coordinate frame
- Support a common IO framework through MetaIO

Spatial Object Example



Spatial Objects IO

- Uses the IO Factory
- Uses MetaIO file format
- Conversion is done internally
- Concatenation of objects in a single file
- Easy to extend

Writing Spatial Objects

```
#include "itkSpatialObjectWriter.h"
#include "itkEllipseSpatialObject.h"

typedef itk::EllipseSpatialObject<3> SphereType;
SphereType::Pointer sphere = SphereType::New();
sphere->SetRadius(2);

typedef itk::SpatialObjectWriter<3> WriterType;
WriterType::Pointer writer = WriterType::New();
writer->SetInput(sphere);
writer->SetFileName("ellipse.meta");
writer->Update();
```

Reading Spatial Objects

```
#include "itkSpatialObjectReader.h"

 typedef itk::SpatialObjectReader<3> ReaderType;
 ReaderType::Pointer reader = ReaderType::New();
 reader->SetFileName("ellipse.meta");
 reader->Update();

// Return an itk::SceneSpatialObject with all the objects in the file
ReaderType::ScenePointer scene = reader->GetScene();

// Return an itk::GroupSpatialObject with all the objects in the file
ReaderType::GroupPointer group = reader->GetGroup();

 ReaderType::SceneType::ObjectListType * sceneChildren = scene->GetObjects(999);
 ReaderType::SceneType::ObjectListType::const_iterator objectIterator;
 objectIterator = sceneChildren->begin();
 if(!strcmp((* objectIterator)->GetTypeName(),"EllipseSpatialObject"))
   {
   SphereType::Pointer sphere = dynamic_cast<SphereType*>((*objectIterator).GetPointer());
   }
```

Writing an itkMesh

```
#include "itkSpatialObjectWriter.h"
#include "itkMeshSpatialObject.h"

typedef itk::DefaultDynamicMeshTraits< float , 3, 3 > MeshTrait;
typedef itk::Mesh<float,3,MeshTrait>                  MeshType;

MeshType::Pointer mesh = MeshType::New();

// Create the mesh Spatial Object
MeshSpatialObjectType::Pointer meshSO =  MeshSpatialObjectType::New();
meshSO->SetMesh(mesh);

// Writing the file
typedef itk::SpatialObjectWriter<3,float,MeshTrait> WriterType;
WriterType::Pointer writer = WriterType::New();
writer->SetInput(meshSO);
writer->SetFileName("metamesh.txt");
writer->Update();
```

Logging Capabilities

Logger

- Record output information and send the output to a stream (or multiple streams)
- Priority Level (Fatal, Critical, Warning, Info, Debug…)
- AddLogOutput() : attach an output stream to the logger
- Write() : send information to the logger

Logger in use

```
#include <itkLogger.h>
#include <itkStdStreamLogOutput.h>

itk::Logger::Pointer logger = itk::Logger::New();
itk::StdStreamLogOutput::Pointer output =
                                        itk::StdStreamLogOutput::New();
output->SetStream(std::cout);

logger->SetName("MyLogger");
logger->SetPriorityLevel(itk::LoggerBase::INFO);
logger->SetLevelForFlushing(itk::LoggerBase::CRITICAL);

logger->AddLogOutput(output);

logger->Write(itk::LoggerBase::INFO, "This is the INFO message.\n");
```

Logger Manager

- Centralize several loggers
- AddLogger()
- CreateLogger() / CreateThreadLogger
- Write()
- Flush()

References

- Culp, Timothy R. "Industrial Strength Pluggable Object Factories". C++ Report Online. http://www.creport.com/html/from_pages/view_recent_articles_c.cfm?ArticleID=1520
- P. Chandra, L. Ibanez, "ImageIO: Design of an Extensible Image Input/Output Library", ACM Crossroad online magazine, April 2001. Available online at http://www.acm.org/crossroads/xrds7-4/imageIO.html
- E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Patterns", Addison Wesley, 1995.

Enjoy ITK !

IGSTK
The Image-Guided Surgery Toolkit

Kitware, Inc

Overview

- History
- Infrastructure
- Components
- Quality Control
- Software Process
- Applications

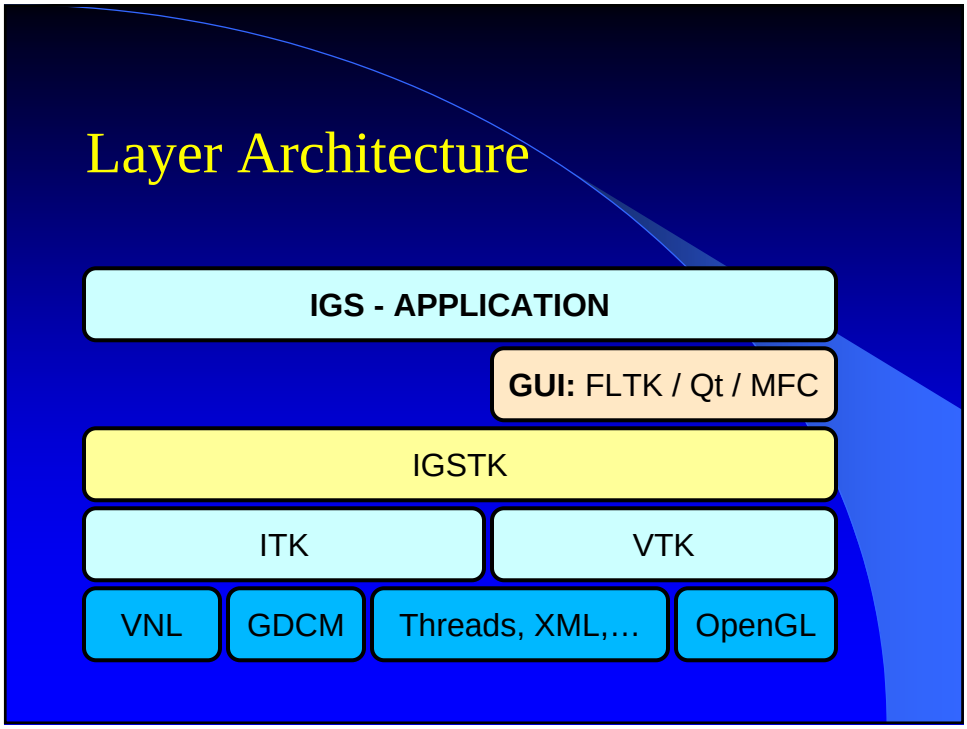
The Toolkit in a Nutshell

- Open Source Toolkit
- BSD-like License
- Written in C++
- Cross Platform
- Based on ITK and VTK
- GUI based on FLTK
- Designed for a Critical Application

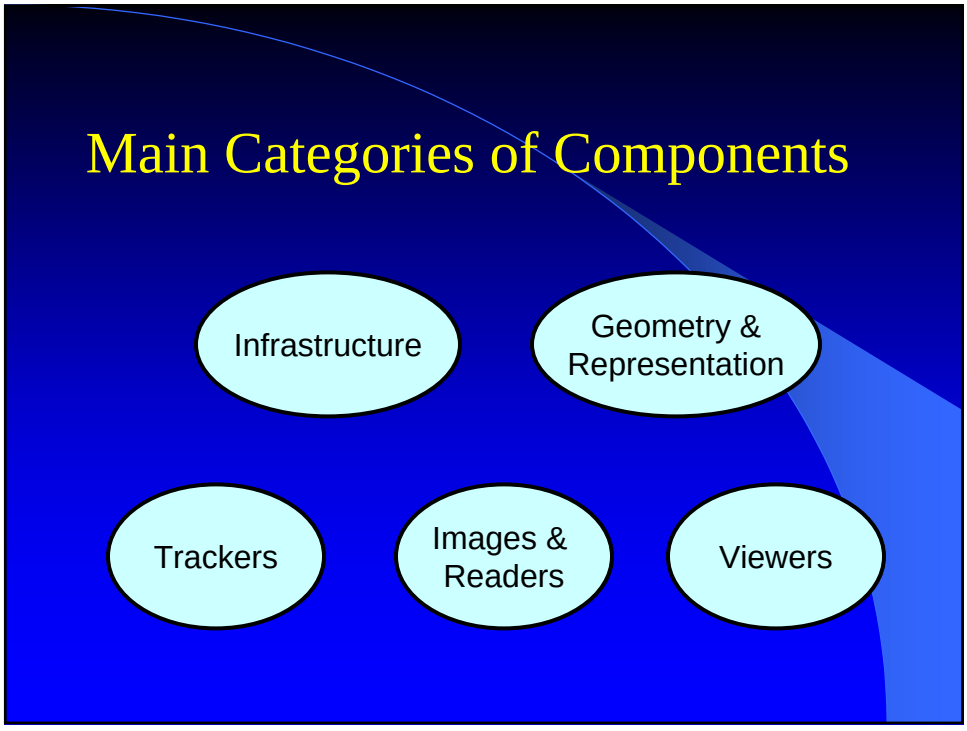
History

- STTR Funded by NIBIB/NIH (Georgetown-Kitware)
- Phase I from May to Sept 2003. Used in example applications.
- Phase II started on October 2004
- CADDLab UNC Joined the project and contributed Spatial Objects & RF ablation Application.
- Atamai joined on 2005 and contributed Tracker code.
- Releases done about every 6 months

Layer Architecture



Main Categories of Components



Infrastructure

- State Machine
- Time Stamp
- Pulse Generator
- Events
- Transforms
- Logger

Trackers

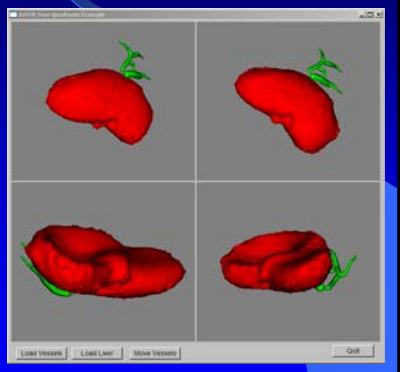
- Tracker (base class)
  - PolarisTracker
  - AuroraTracker
  - Flock Of Birds
- Support
  - SerialCommunication
    - SerialCommunicationForWindows
    - SerialCommunicationForPosix
  - NDICommandInterpreter

Object Representation

- SpatialObject
  - Ellipsoid
  - Cylinder
  - Images
  - Mesh
  - Tube
  - TubeGroup
  - Group
  - Box
  - Cone
- SpatialObjectRepresentation
  - Ellipsoid
  - Cylinder
  - Images
  - Mesh
  - Tube
  - Box
  - Cone
  - VascularNetwork
  - AirwaysNetwork
  - Surgical Tools

Viewers

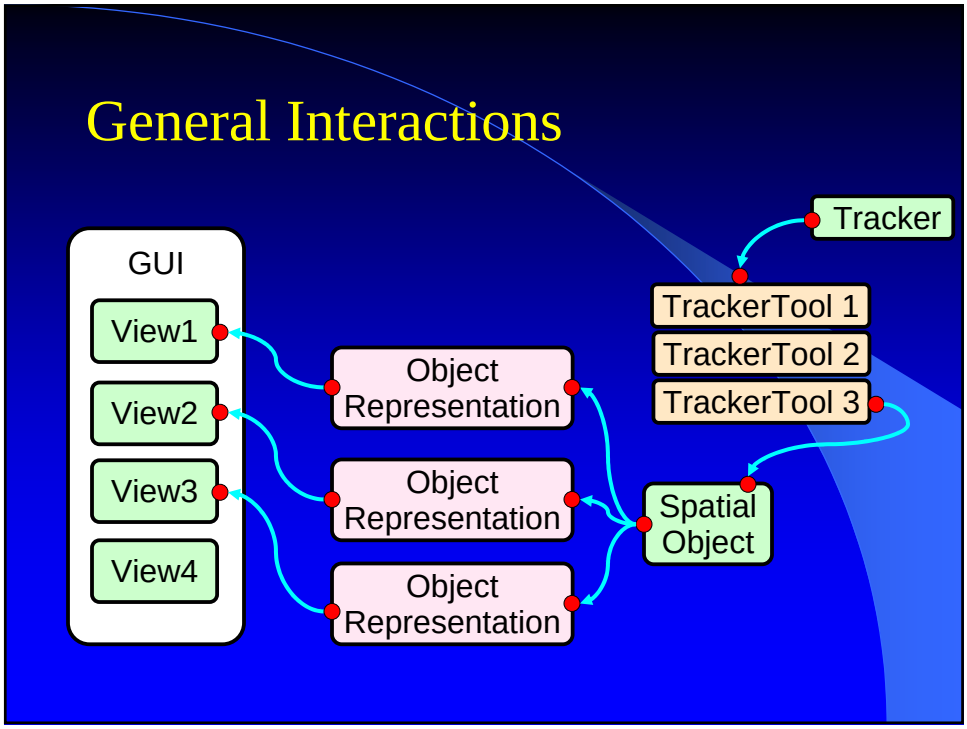
- View
  - View2D
  - View3D
- VTK Based
- GUI / VTK hybrid class

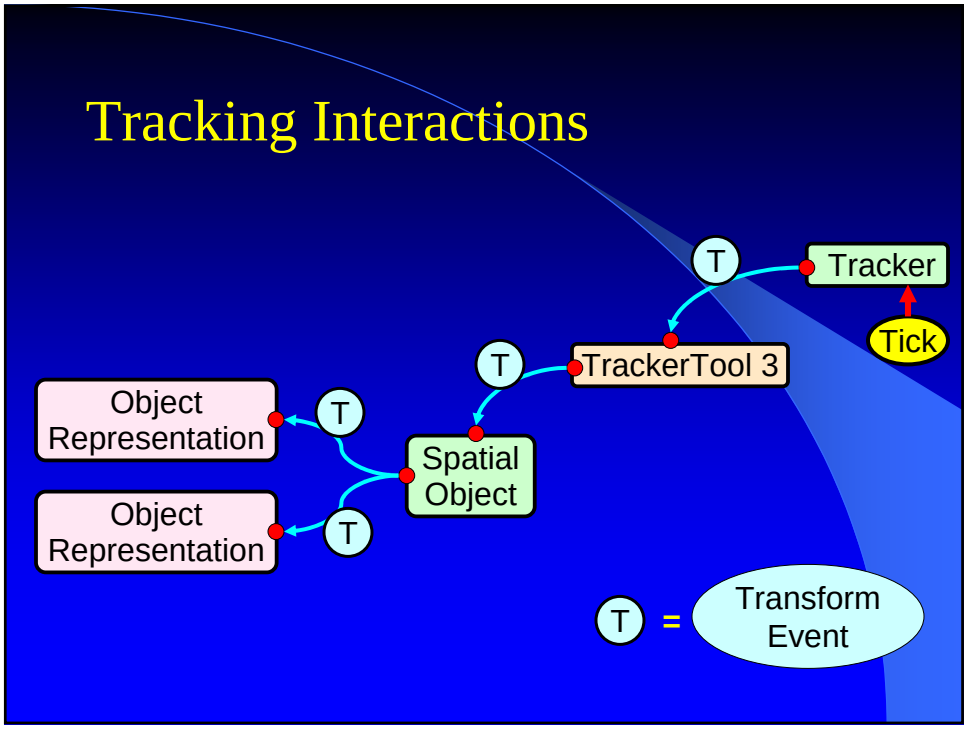


Images and Readers

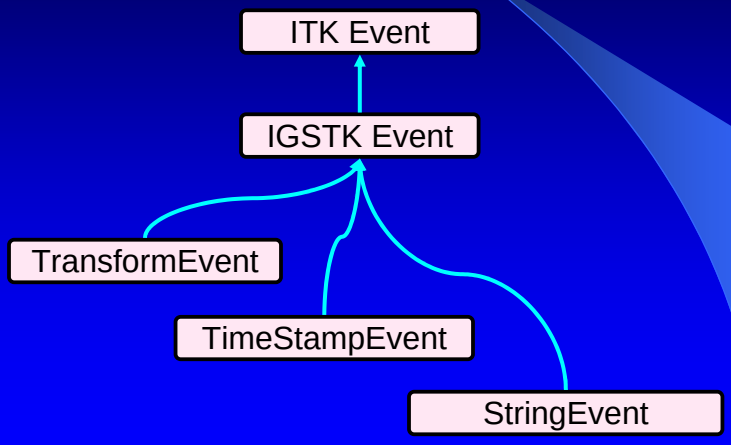
- Image
  - CTImage
  - MRImage
  - FluoroscopicImage
- Readers (DICOM)
  - CTImageReader
  - MRImageReader
  - Fluoroscopy (video input)

Dynamics Architecture


General Interactions
Tracker
GUI
View1
View2
View3
View4
TrackerTool 1
TrackerTool 2
TrackerTool 3
Object Representation
Object Representation
Object Representation
Spatial Object


Tracking Interactions
T
Tracker
Tick
T
TrackerTool 3
Object Representation
T
Spatial Object
Object Representation
T
T = Transform Event

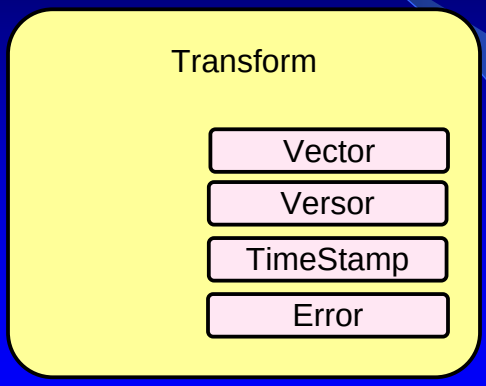
Event Class



Event Class

```
namespace igstk
{

itkEventMacro( IGSTKEvent,               itk::UserEvent );
itkEventMacro( PositionModifiedEvent,    IGSTKEvent );
itkEventMacro( OrientationModifiedEvent, IGSTKEvent );
itkEventMacro( GeometryModifiedEvent,    IGSTKEvent );
itkEventMacro( PropertyModifiedEvent,    IGSTKEvent );
itkEventMacro( PulseEvent,               IGSTKEvent );
itkEventMacro( RefreshEvent,             IGSTKEvent );

}
```

Transform Class



Transform Class

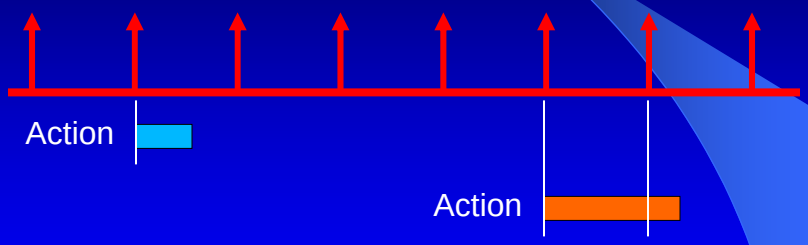
```
class Transform
{
public:

  typedef ::itk::Vector<double, 3>    VectorType;
  typedef ::itk::Versor<double>       VersorType;
  typedef double                      ErrorType;
  typedef TimeStamp::TimePeriodType   TimePeriodType;
```

```
private:

  TimeStamp       m_TimeStamp;
  VectorType      m_Translation;
  VersorType      m_Rotation;
  ErrorType       m_Error;
```

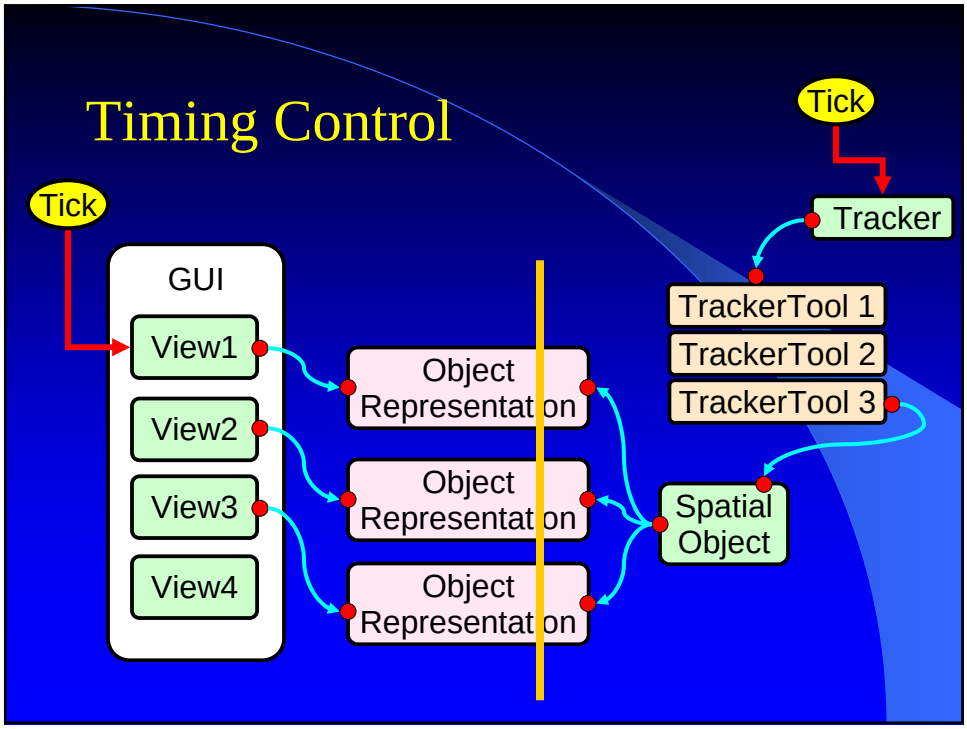
Transform Event Class

```
class  TransformModifiedEvent : public IGSTKEvent
{
public:
  typedef TransformModifiedEvent Self;
  typedef IGSTKEvent Superclass;

  TransformModifiedEvent();
  virtual ~TransformModifiedEvent();
  virtual const char * GetEventName();

  virtual bool CheckEvent(const ::itk::EventObject* e) const
  virtual ::itk::EventObject* MakeObject() const

  TransformModifiedEvent(const Self&s);

  const Transform & GetTransform() const
  void SetTransform( const Transform & transform )

private:

  void operator=(const Self&);

  // Payload of this event.
  Transform  m_Transform;
};
```

Pulse Generator



How not to miss pulses ?


Timing Control
Tick
Tick
Tracker
GUI
View1
View2
View3
View4
Object Representation
Object Representation
Object Representation
TrackerTool 1
TrackerTool 2
TrackerTool 3
Spatial Object


Quality Control

Testing Platforms

- Windows
  - Visual Studio 6.0 / 7.1 / 8.0
- Linux
  - gcc 3.2 / 3.3 / 3.4 / 4.0
  - icc 7.1 / 8.0
- Cygwin – Windows
  - gcc 3.4
  - gcc 3.3

Testing

- Goal
  - 100 % Code Coverage
  - 100 % State Machine Transitions Testing
- Current Code Coverage
  - IGSTK 93.6 % (untested 321 / 4714 lines)
  - Sandbox 92.4 % (untested 4 / 49 lines)

Dynamic Testing Platforms

- Valgrind on Linux
  - gcc 3.3
  - gcc 3.4
- Valgrind on Cygwin
- Purify (too expensive)

Software Process

Software Process

- Sandbox
- Extreme programming
- Release early, release often
- Code reviews
- Requirements
- Bug tracking

Typical Agenda for a release

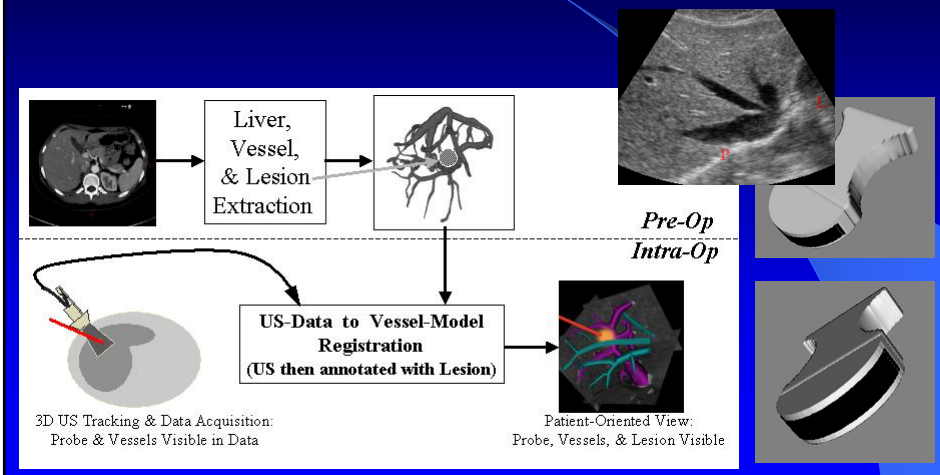
- Create feature list
- Write requirements
- Develop code
- Tag Sandbox, create code review list
- Code Review
- Move code and tag
- Fix Documentation
- Fix remaining bugs
- Increase code coverage
- Create and release package

Applications

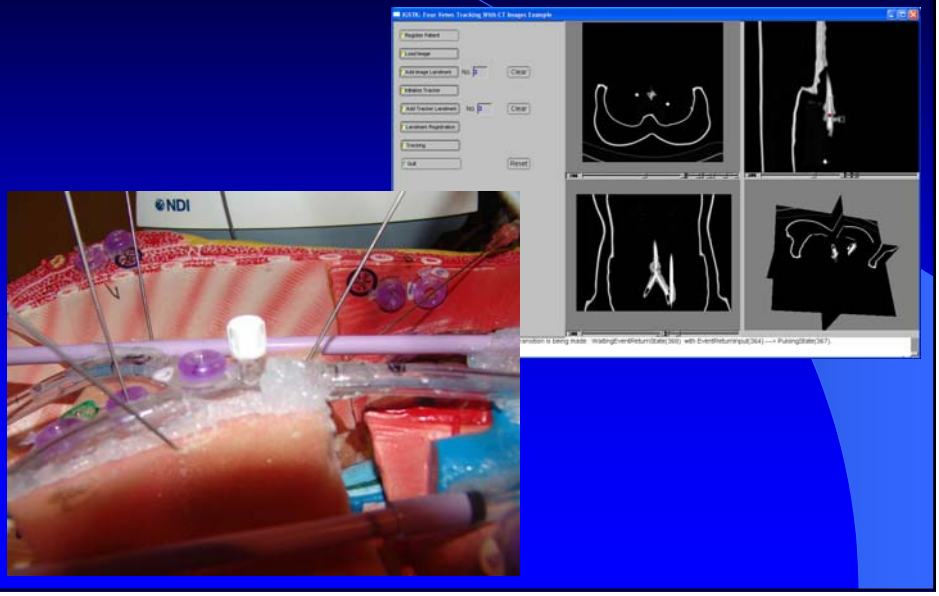
Applications

- Ultrasound-guided percutaneous liver lesion RFA
- Needle Biopsy
- Guidewire Tracking

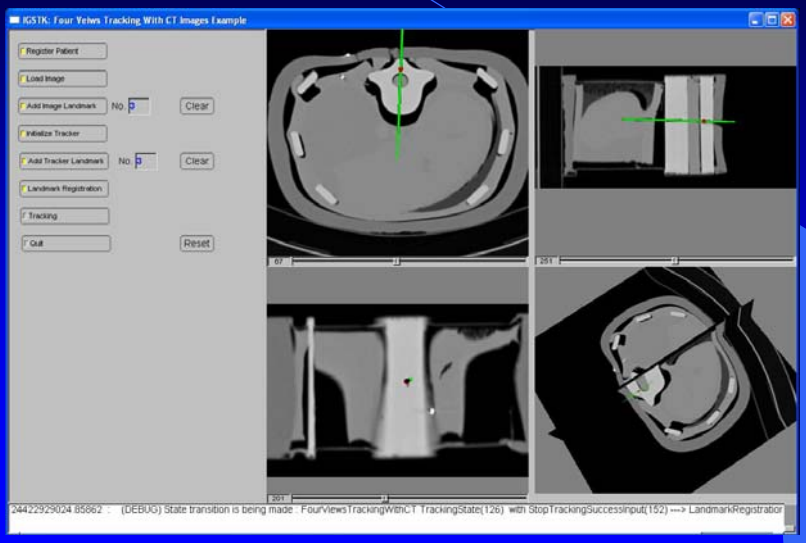
Ultrasound-Guided Liver RFA



GuideWire Tracking



Needle Biopsy



Download IGSTK and Try it!

- http://www.igstk.org
- Last stable release “Iteration 7”
- Demo: Sunday Feb 12 from 8:30 to 10:30 pm

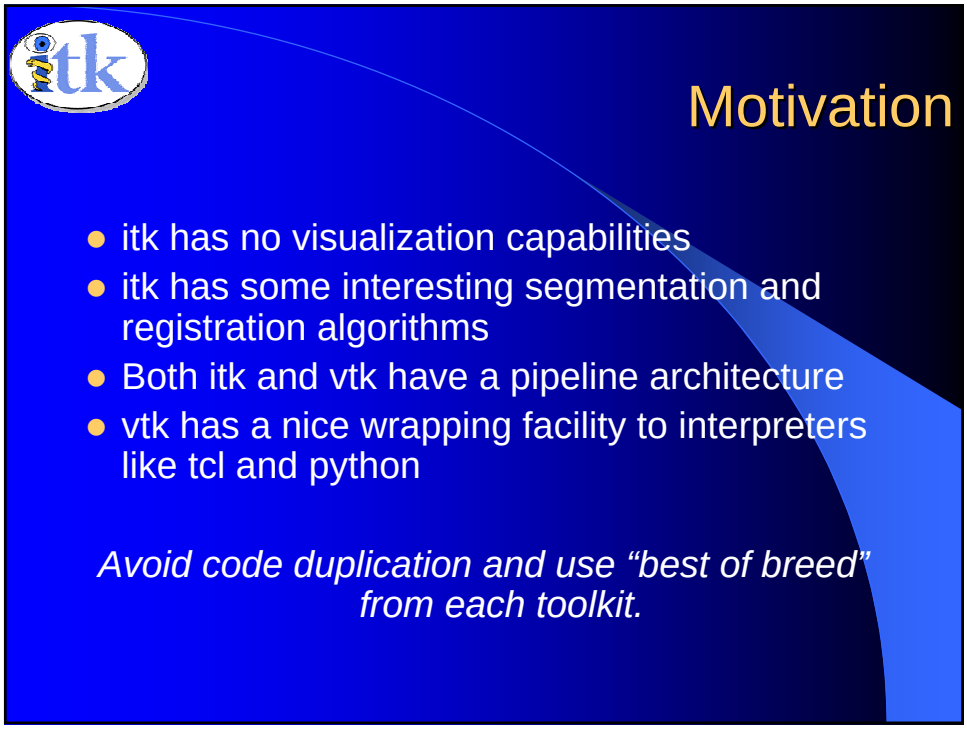
Enjoy IGSTK!

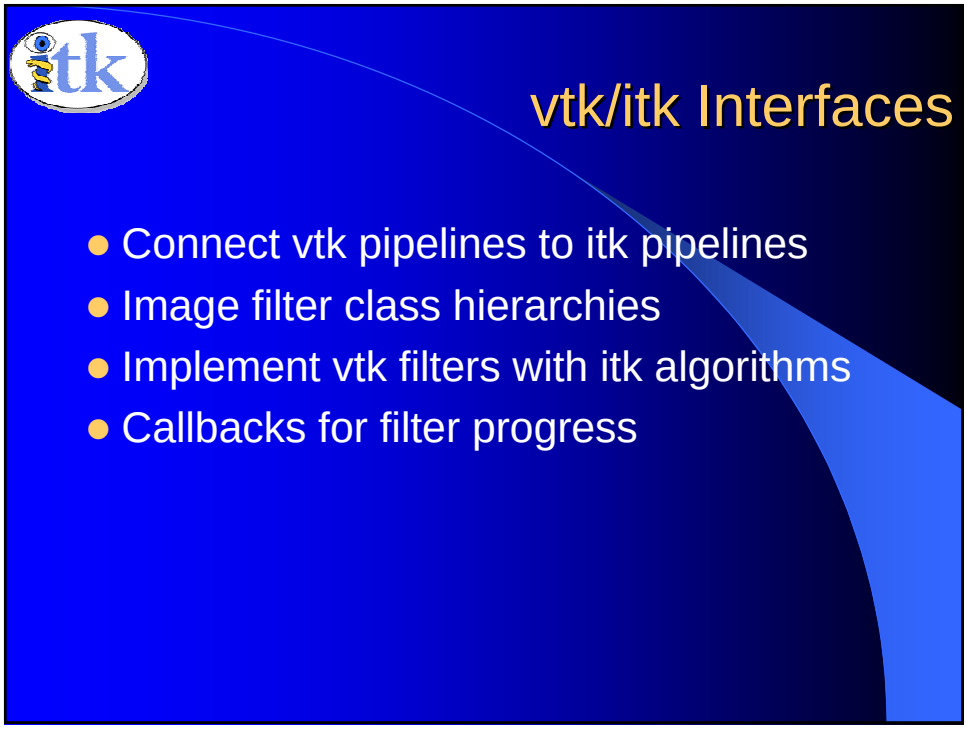
Applications:
Using Insight(itk) and
The Visualization Toolkit(vtk)

Bill Lorensen
Jim Miller
GE Research
Niskayuna, NY

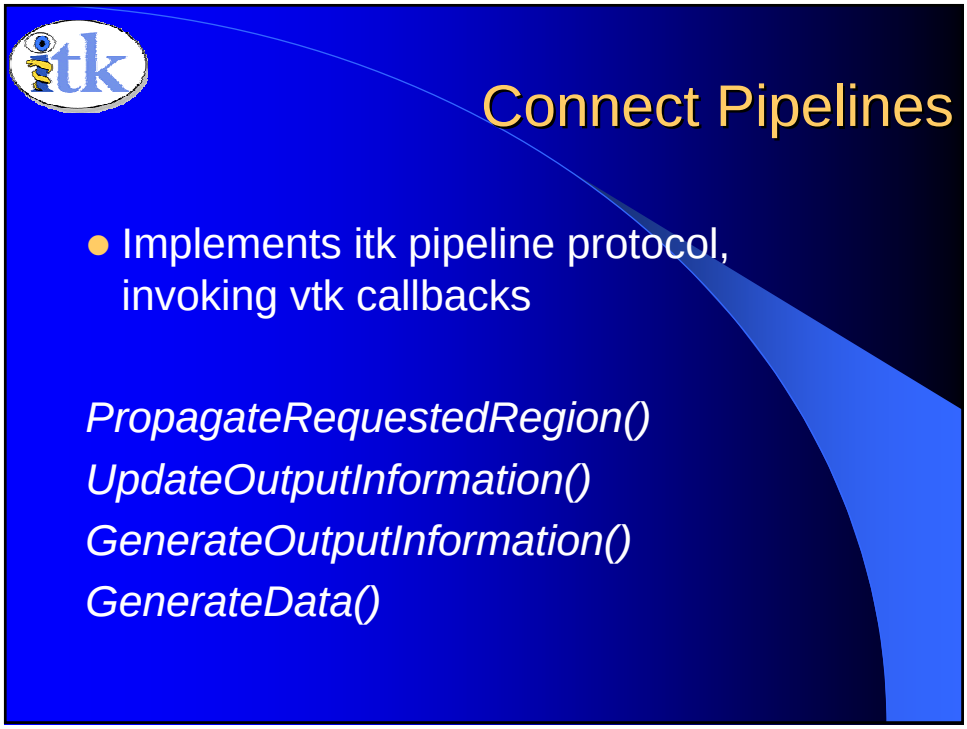
Agenda

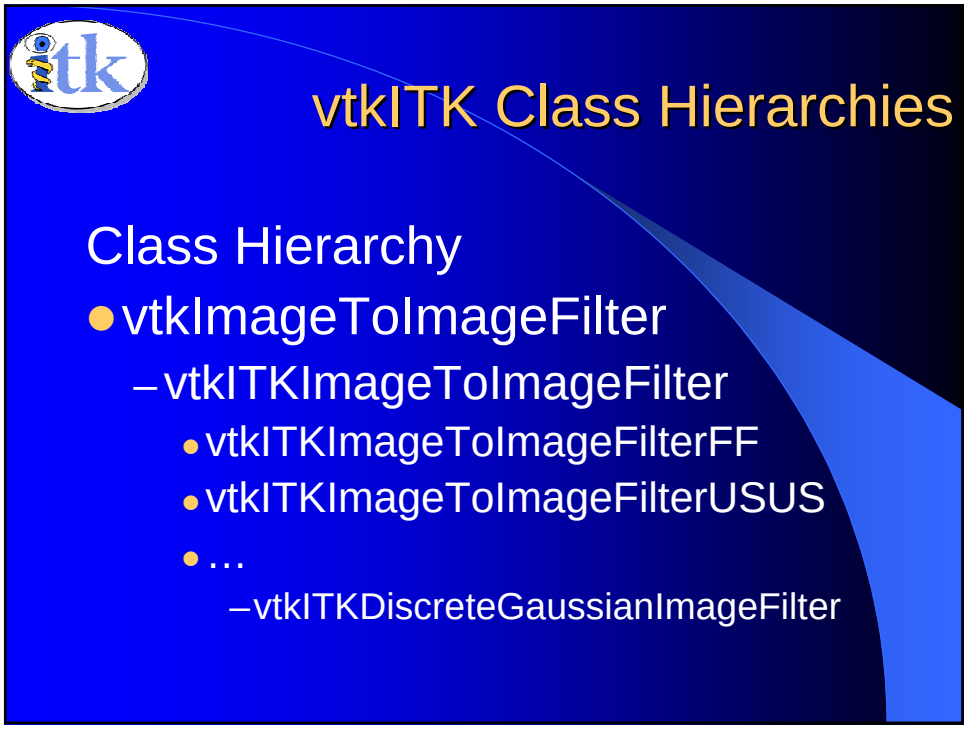
- Approaches to interfacing vtk and itk
- Case Studies / Demos
  - Registration
    - MR/MR
    - CT/MR
  - Segmentation
    - Watershed

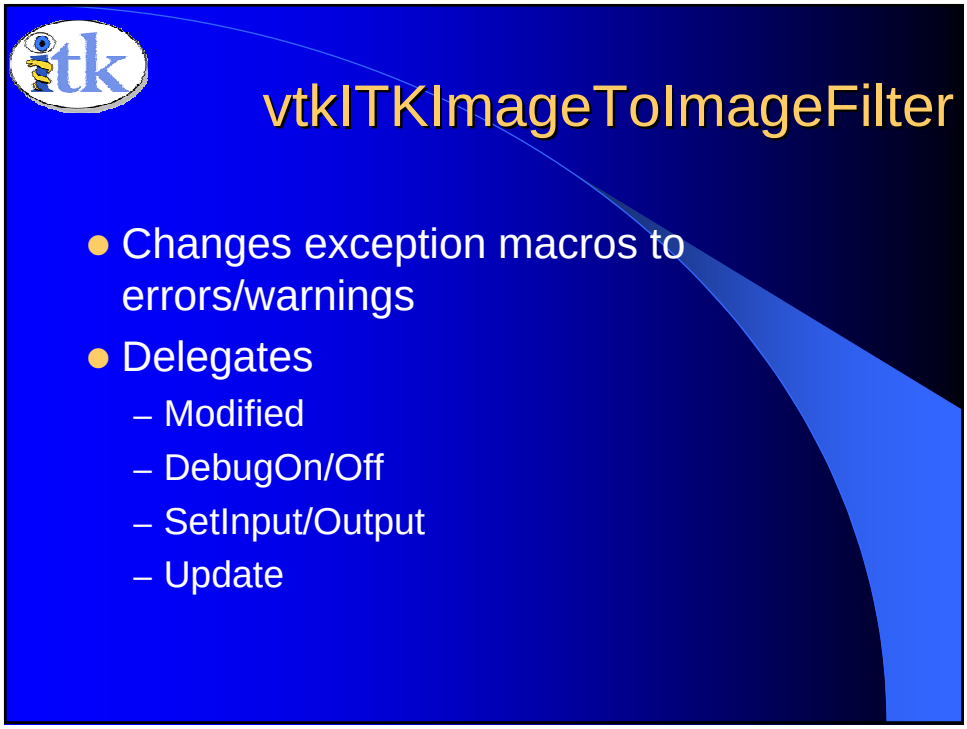

Motivation
itk has no visualization capabilities
itk has some interesting segmentation and registration algorithms
Both itk and vtk have a pipeline architecture
vtk has a nice wrapping facility to interpreters like tcl and python
Avoid code duplication and use “best of breed” from each toolkit.

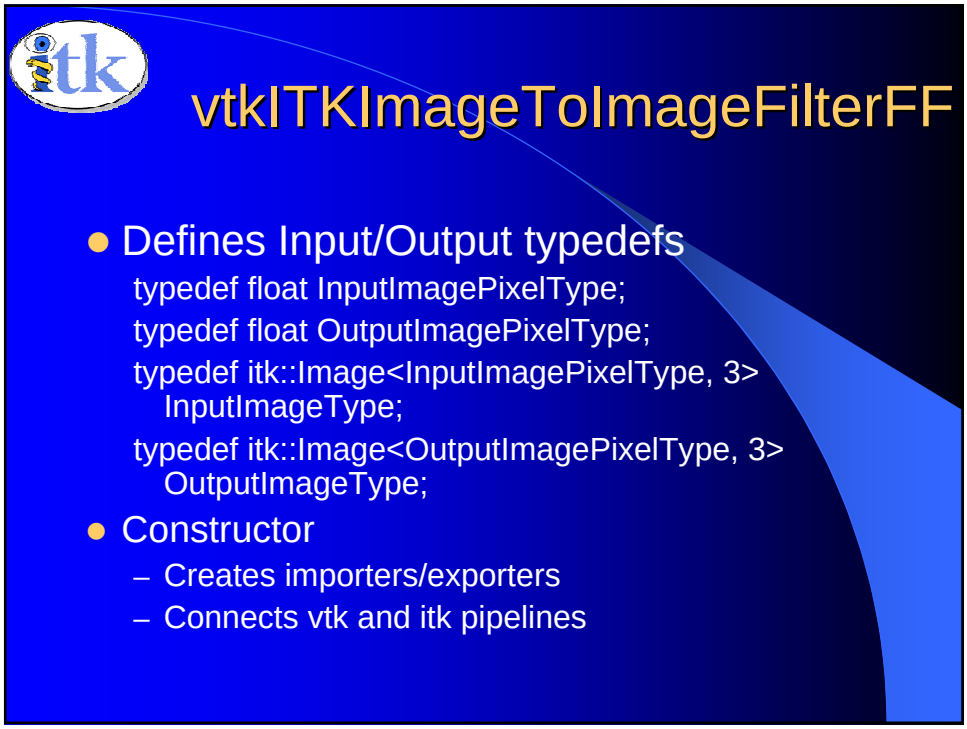

vtk/itk Interfaces
Connect vtk pipelines to itk pipelines
Image filter class hierarchies
Implement vtk filters with itk algorithms
Callbacks for filter progress

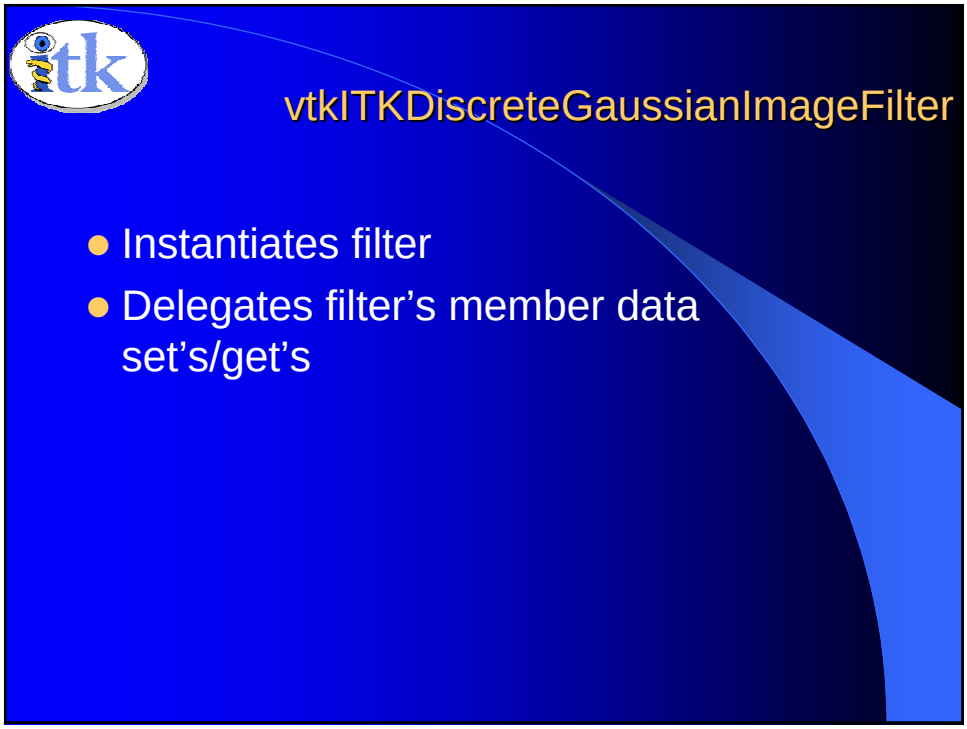

Connect pipelines - vtk to itk
vtkImageExport *movingExporter = vtkImageExport::New();
movingExporter->SetInput(movingReader->GetOutput());
typedef itk::VTKImageImport<InputType> ImageImportType;
ImageImportType::Pointer movingImporter =
ImageImportType::New();
ConnectPipelines(movingExporter, movingImporter);
InsightApplications/vtkITK/Common/vtkITKUtility.h


Connect Pipelines
Implements itk pipeline protocol, invoking vtk callbacks
PropagateRequestedRegion()
UpdateOutputInformation()
GenerateOutputInformation()
GenerateData()


vtkITK Class Hierarchies
Class Hierarchy
vtkImageToImageFilter
vtkITKImageToImageFilter
vtkITKImageToImageFilterFF
vtkITKImageToImageFilterUSUS
…
vtkITKDiscreteGaussianImageFilter

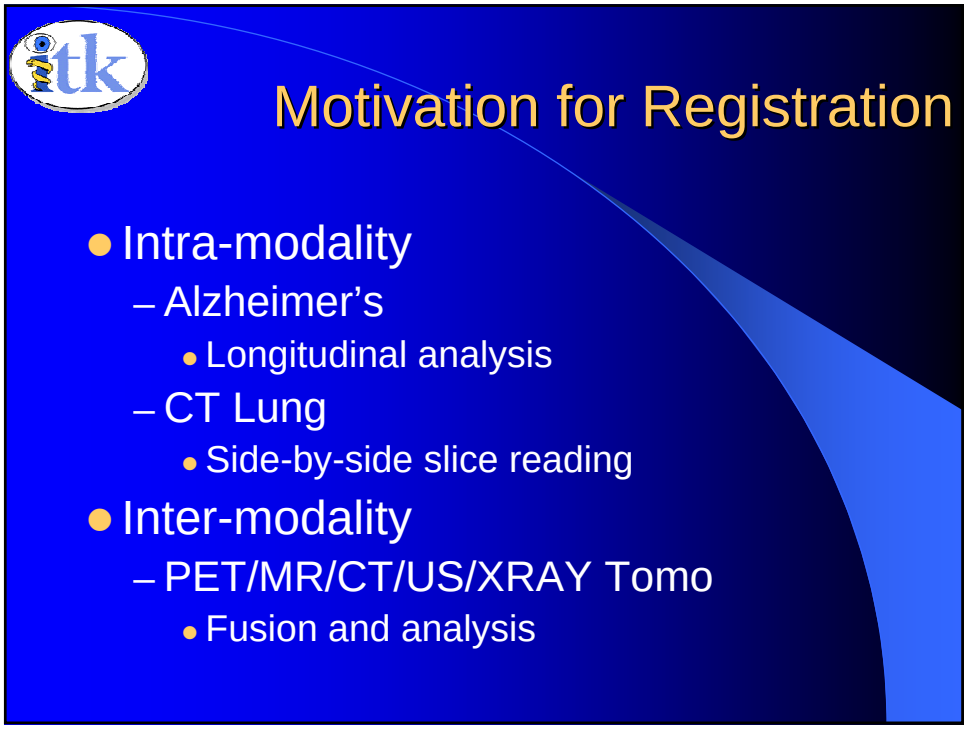

vtkITKImageToImageFilter
Changes exception macros to errors/warnings
Delegates
Modified
DebugOn/Off
SetInput/Output
Update

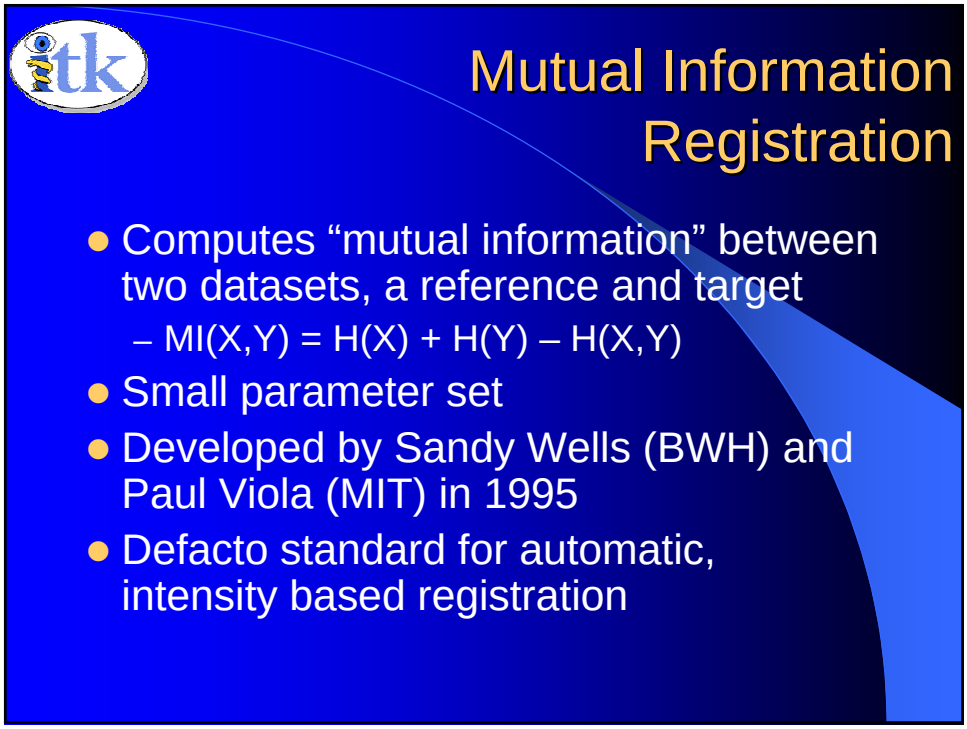

vtkITKImageToImageFilterFF
Defines Input/Output typedefs
typedef float InputImagePixelType;
typedef float OutputImagePixelType;
typedef itk::Image<InputImagePixelType, 3> InputImageType;
typedef itk::Image<OutputImagePixelType, 3> OutputImageType;
Constructor
– Creates importers/exporters
– Connects vtk and itk pipelines

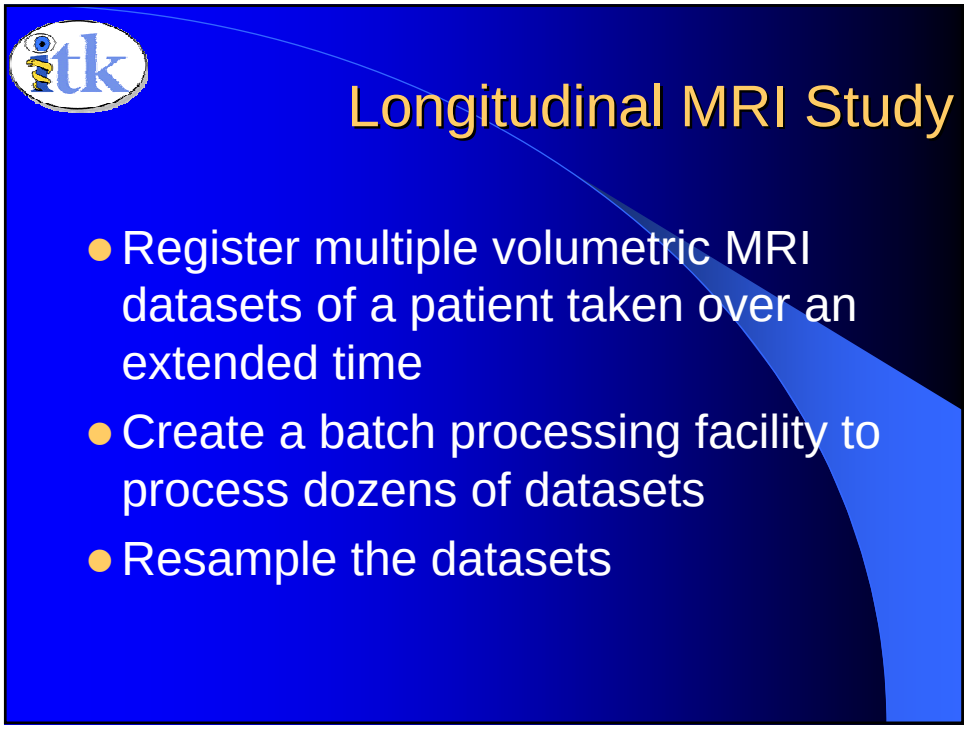

vtkITKDiscreteGaussianImageFilter
Instantiates filter
Delegates filter's member data set's/get's

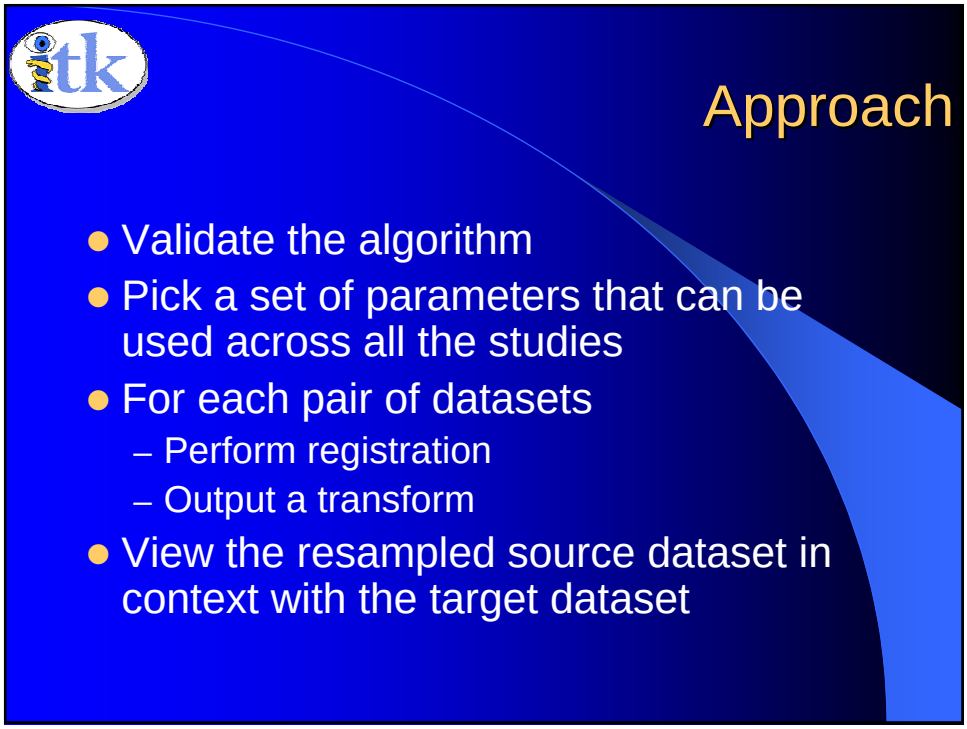

vtk Filters Implemented with itk algorithms
vtkITKMutualInformationTransform
Implement InternalUpdate()
vtkImageExport
itk::VTKImageImport
ConnectPipelines()
Convert vtk transform to quaternion
itk::RegistrationMethod
Convert itk quaternion to vtk transform


Illustrative Examples
Registration


Motivation for Registration
Intra-modality
Alzheimer's
Longitudinal analysis
CT Lung
Side-by-side slice reading
Inter-modality
PET/MR/CT/US/XRAY Tomo
Fusion and analysis


Mutual Information Registration
Computes "mutual information" between two datasets, a reference and target
MI(X,Y) = H(X) + H(Y) – H(X,Y)
Small parameter set
Developed by Sandy Wells (BWH) and Paul Viola (MIT) in 1995
Defacto standard for automatic, intensity based registration

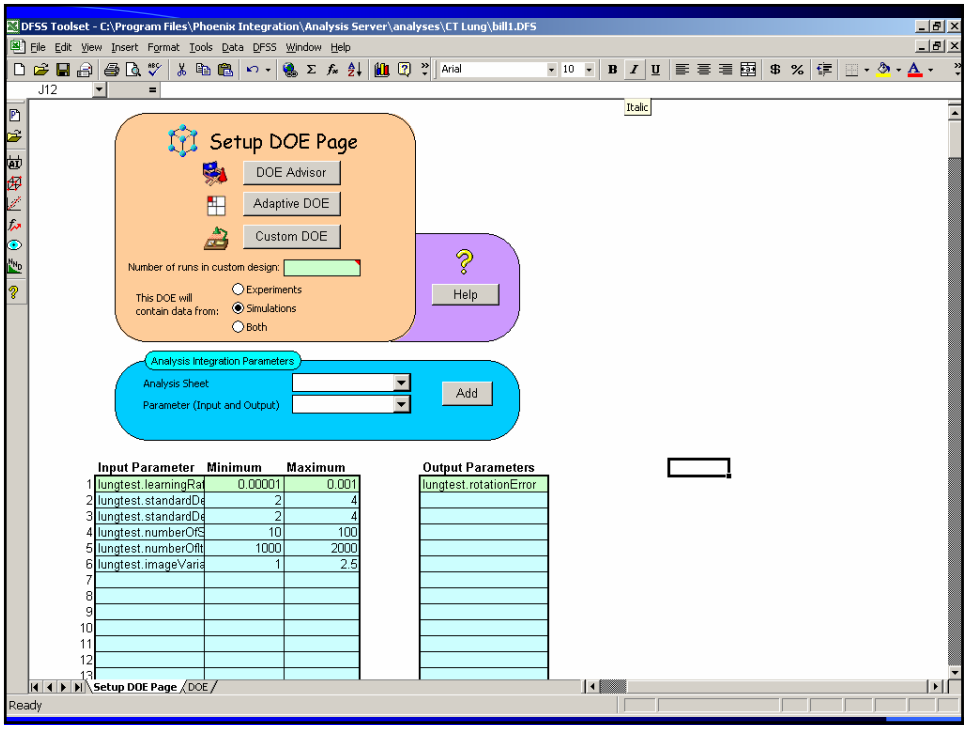

Longitudinal MRI Study
Register multiple volumetric MRI datasets of a patient taken over an extended time
Create a batch processing facility to process dozens of datasets
Resample the datasets

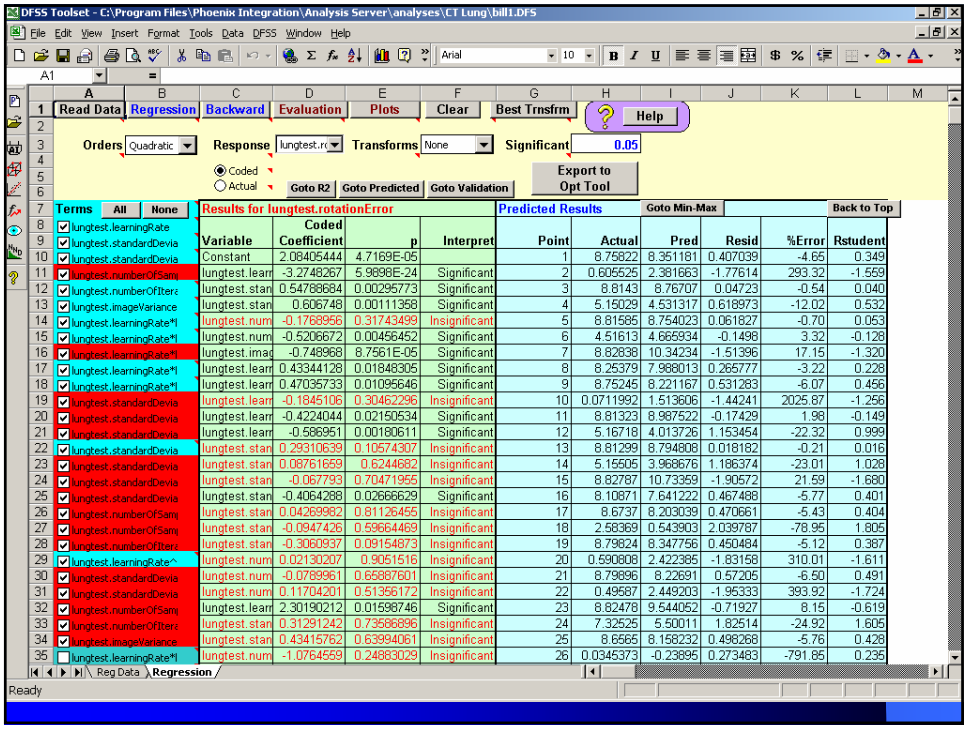

Approach
Validate the algorithm
Pick a set of parameters that can be used across all the studies
For each pair of datasets
– Perform registration
– Output a transform
View the resampled source dataset in context with the target dataset

The MI Algorithm Parameter Space

- Learning Rate
- Standard Deviation of Parzen Window estimate
- Number of Samples for density estimate
- Number of Iterations
- Image variance
- Translation scale factors

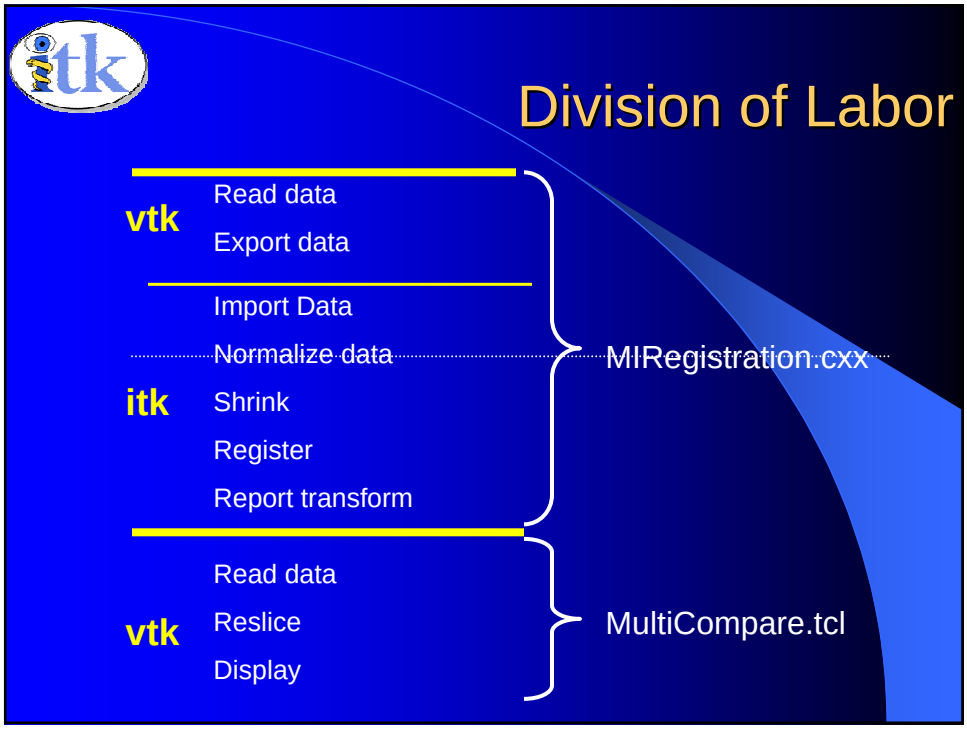
GE's Design for Six Sigma (DFSS) Design of Experiments Tool





Variable	Coded Coefficient	p	Interpret
Constant	2.08405444	4.7169E-05	
lungtest.learningRate	-3.2748267	5.9898E-24	Significant
lungtest.standardDeviationA	0.54788684	0.00295773	Significant
lungtest.standardDeviationB	0.606748	0.00111358	Significant
lungtest.numberOfSamples	-0.1768956	0.31743499	Insignificant
lungtest.numberOfIterations	-0.5206672	0.00456452	Significant
lungtest.imageVariance	-0.748968	8.7561E-05	Significant

Point	Actual	Pred	Resid	%Error	Rstudent
1	8.75822	8.351181	0.407039	-4.65	0.349
2	0.605525	2.381663	-1.77614	293.32	-1.559
3	8.8143	8.76707	0.04723	-0.54	0.040
4	5.15029	4.531317	0.618973	-12.02	0.532
5	8.81585	8.754023	0.061827	-0.70	0.053
6	4.51613	4.665934	-0.1498	3.32	-0.128
7	8.82838	10.34234	-1.51396	17.15	-1.320
8	8.25379	7.988013	0.265777	-3.22	0.228
9	8.75245	8.221167	0.531283	-6.07	0.466
10	0.0711992	1.513606	-1.44241	2025.87	-1.256
11	8.81323	8.987522	-0.17429	1.98	-0.149
12	5.16718	4.013726	1.153454	-22.32	0.999
13	8.81299	8.794808	0.018182	-0.21	0.016
14	5.15505	3.968676	1.186374	-23.01	1.028
15	8.82787	10.73359	-1.90572	21.59	-1.680
16	8.10871	7.641222	0.467488	-5.77	0.401
17	8.6737	8.203039	0.470661	-5.43	0.404
18	2.58369	0.543903	2.039787	-78.95	1.805
19	8.79824	8.347756	0.450484	-5.12	0.387
20	0.590808	2.422385	-1.83158	310.01	-1.611
21	8.79896	8.22691	0.57205	-6.50	0.491
22	0.49587	2.449203	-1.95333	393.92	-1.724
23	8.82478	9.544052	-0.71927	8.15	-0.619
24	7.32525	5.50011	1.82514	-24.92	1.605
25	8.6565	8.158232	0.498268	-5.76	0.428
26	0.0345373	-0.23895	0.273483	-791.85	0.235

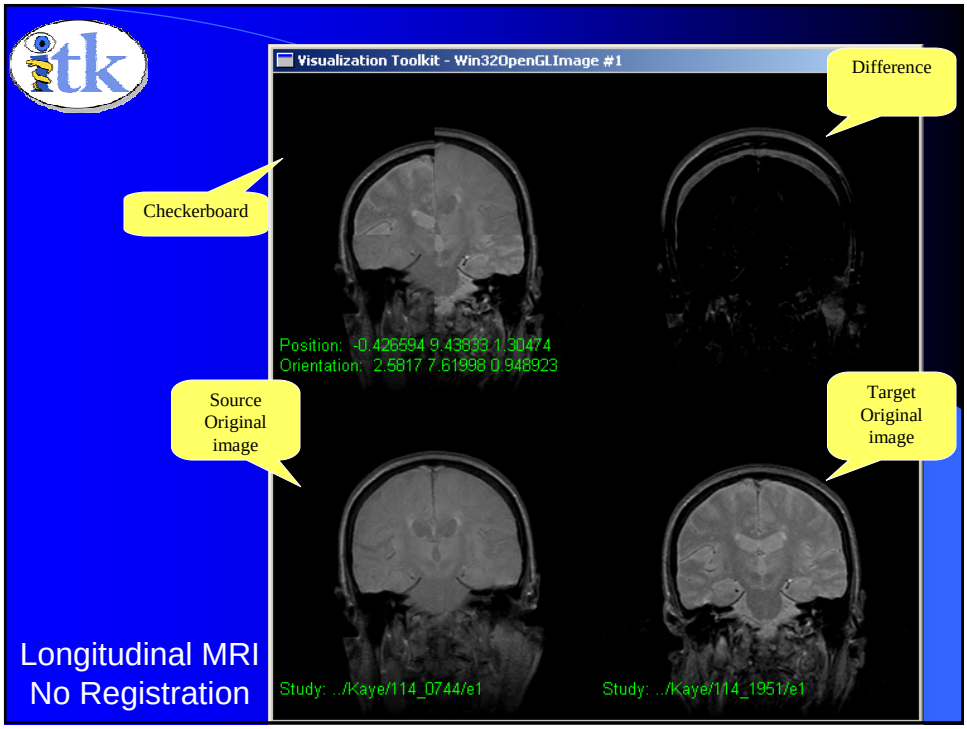

Division of Labor
vtk
Read data
Export data
Import Data
Normalize data
itk
Shrink
Register
Report transform
MIRegistration.cxx
vtk
Read data
Reslice
Display
MultiCompare.tcl

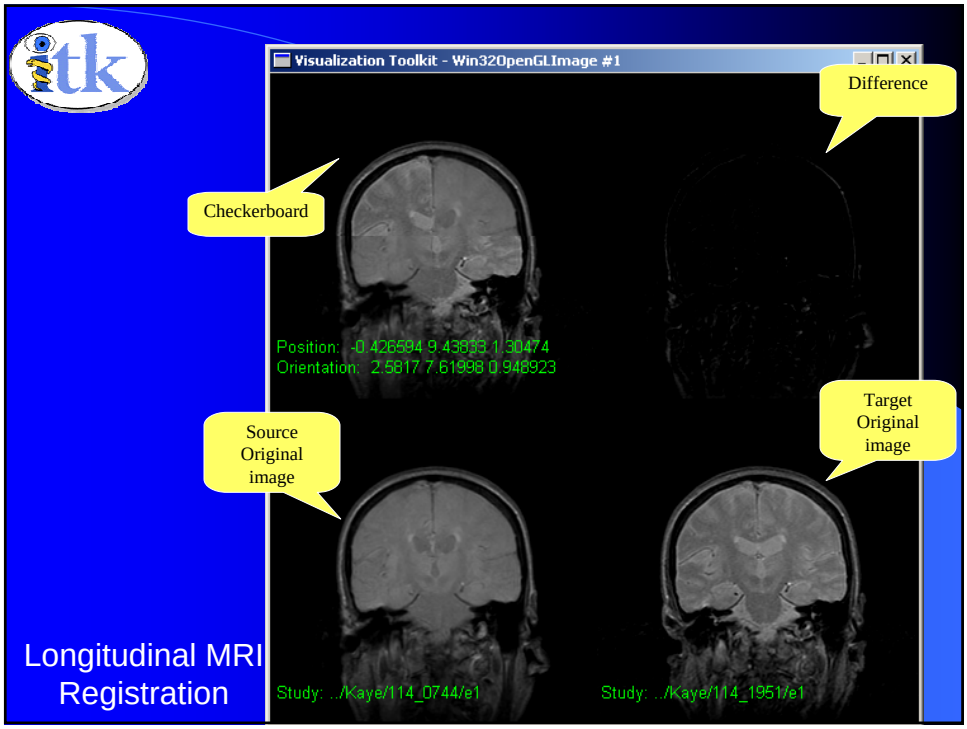

The Pipeline
vtkImageReader
vtkImageExport
itk::vtkImporter
itk::NormalizeImageFilter
itk::ShrinkImageFilter
itk::ImageRegistrationMethod
vtkTransform

Oregon Data

- 25 Registrations
- 13 Subjects, some with AD
- Qualitative comparison
- One set of parameters for all studies

Data Courtesy of Jeffrey Kaye, Oregon Health Sciences Center



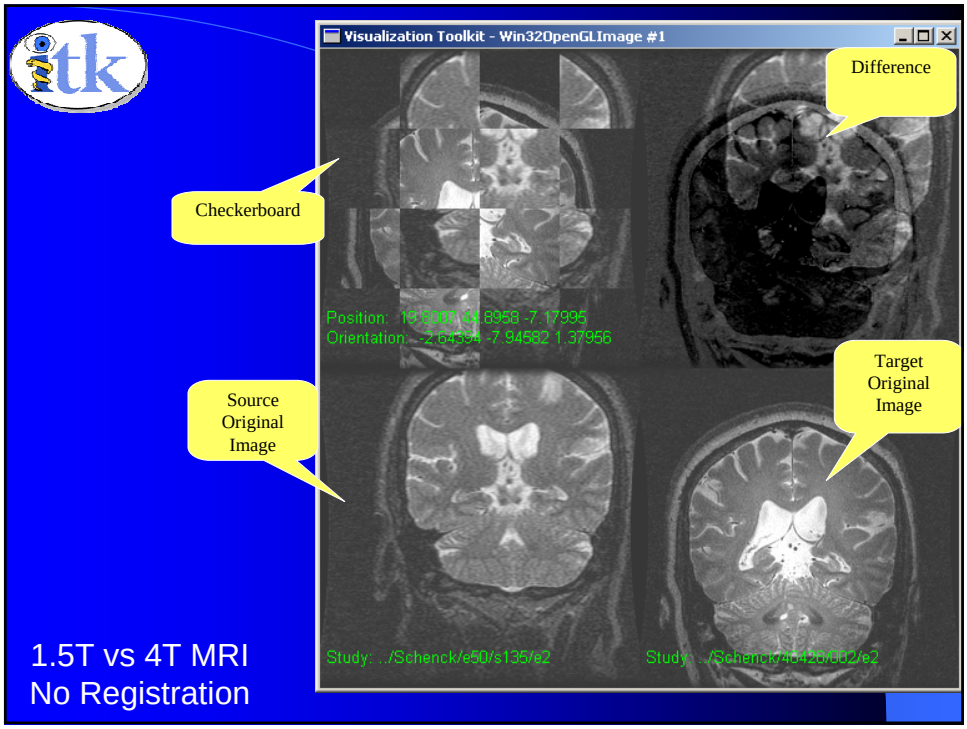


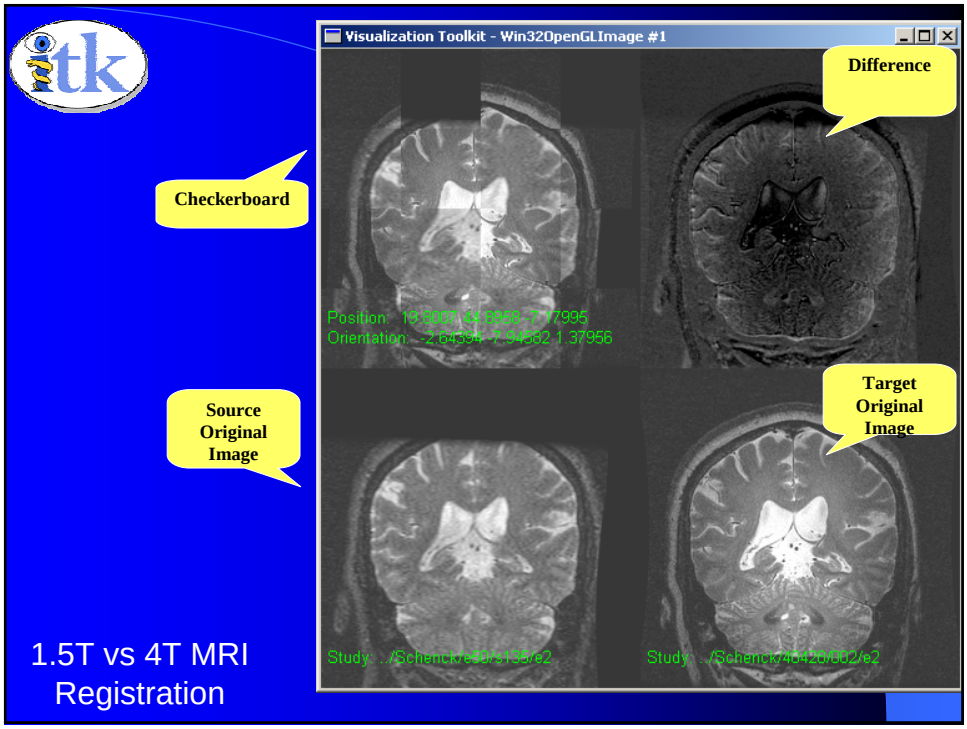
Longitudinal MRI Registration

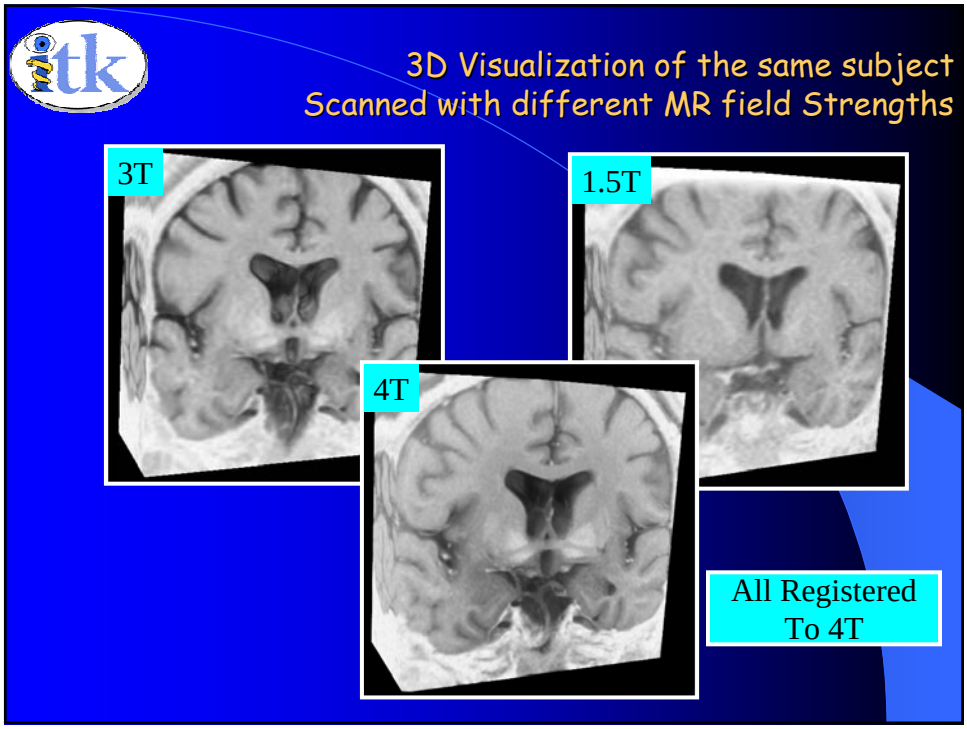
Multi Field MRI Data

- Register 1.5T and 3T to 4T data
- Resampled 1.5T and 3T to correspond to the 4T sampling
- Volume rendering of the 3 datasets from the same view
- Demo mri1.bat, MultiCompareAD.tcl

InsightApplications/MRIRegistration

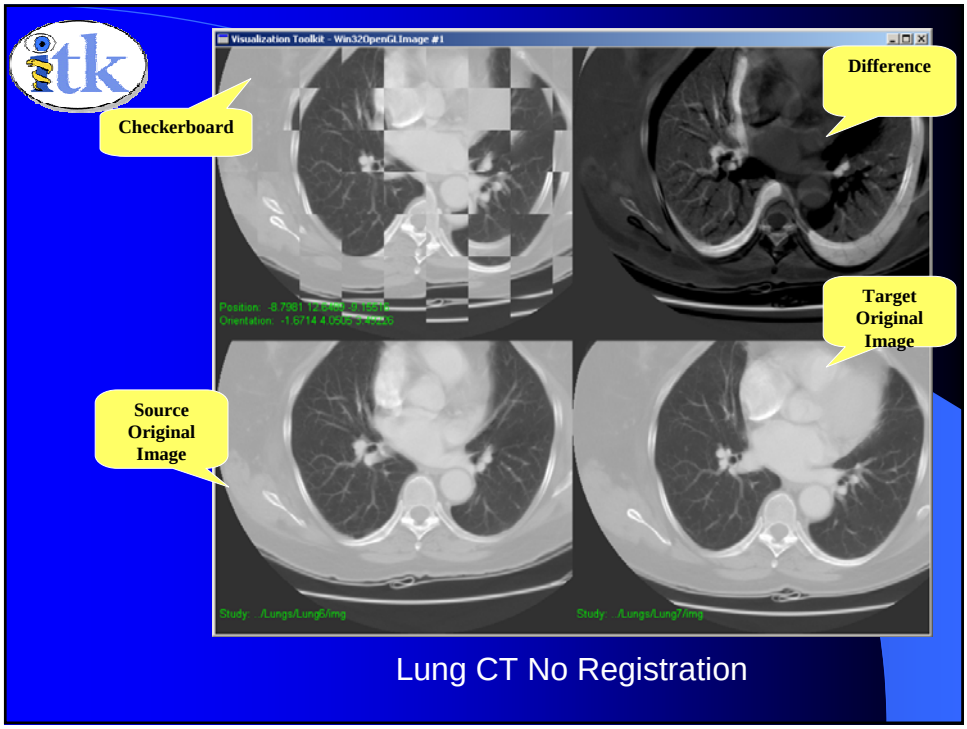

Visualization Toolkit - Win32OpenGLImage #1
Difference
Checkerboard
Source Original Image
Target Original Image
Position: 18.9007 44.8958 -7.17995
Orientation: -2.64394 -7.94582 1.37956
Study: ../Schenck/e50/s135/e2
Study: ../Schenck/40428/662/e2
1.5T vs 4T MRI
No Registration

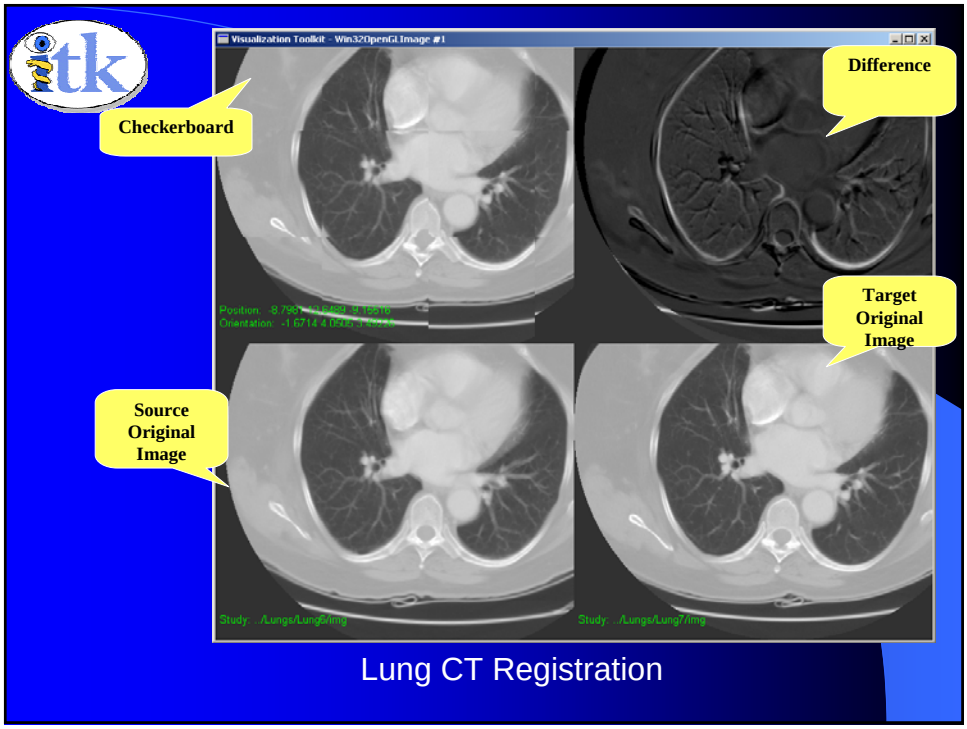

Visualization Toolkit - Win32OpenGLImage #1
Difference
Checkerboard
Source Original Image
Target Original Image
Position: 18.9007 44.8958 -7.17995
Orientation: -2.64394 -7.94582 1.37956
Study: ../Schenck/e50/s135/e2
Study: ../Schenck/40428/662/e2
1.5T vs 4T MRI
Registration

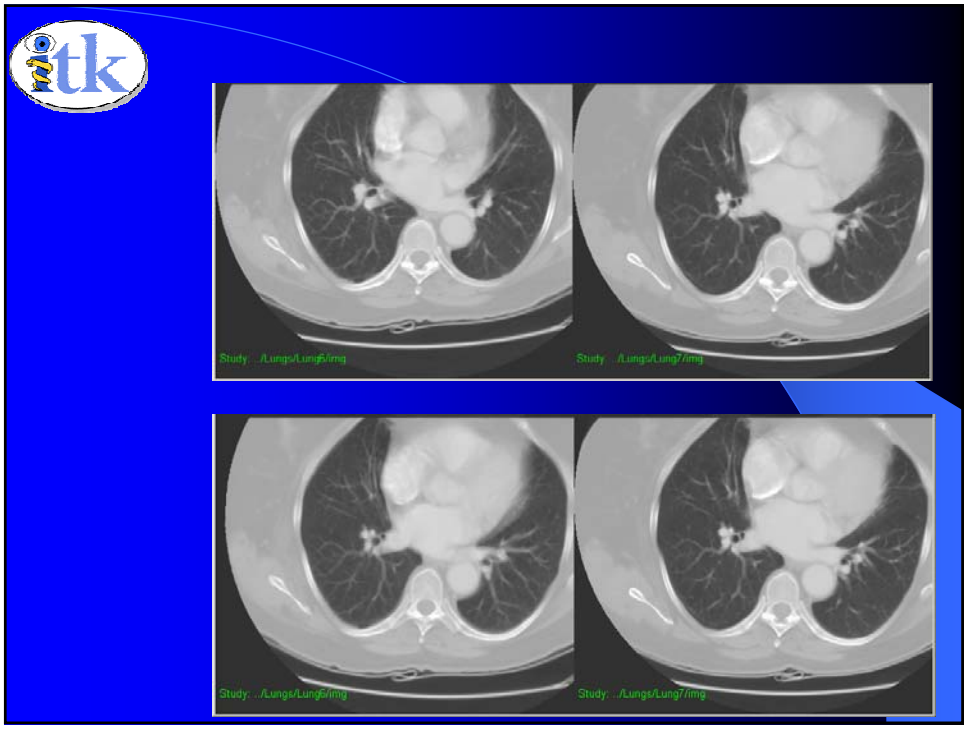


CT Lung Longitudinal Study

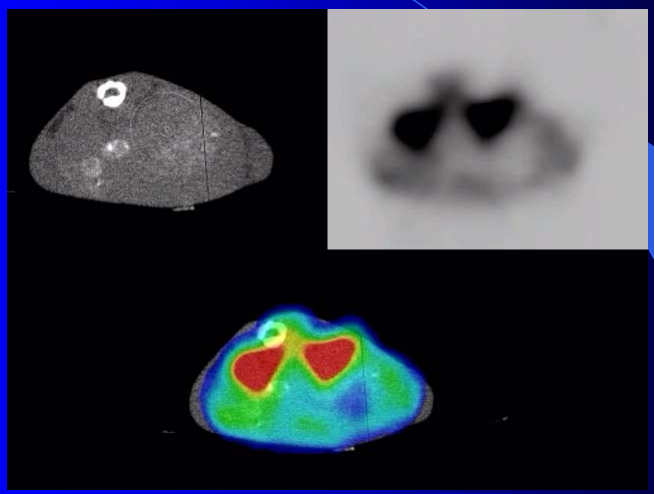
- Register two CT exams of the same patient taken at two different times
- Side-by-side synchronized view for visual comparison

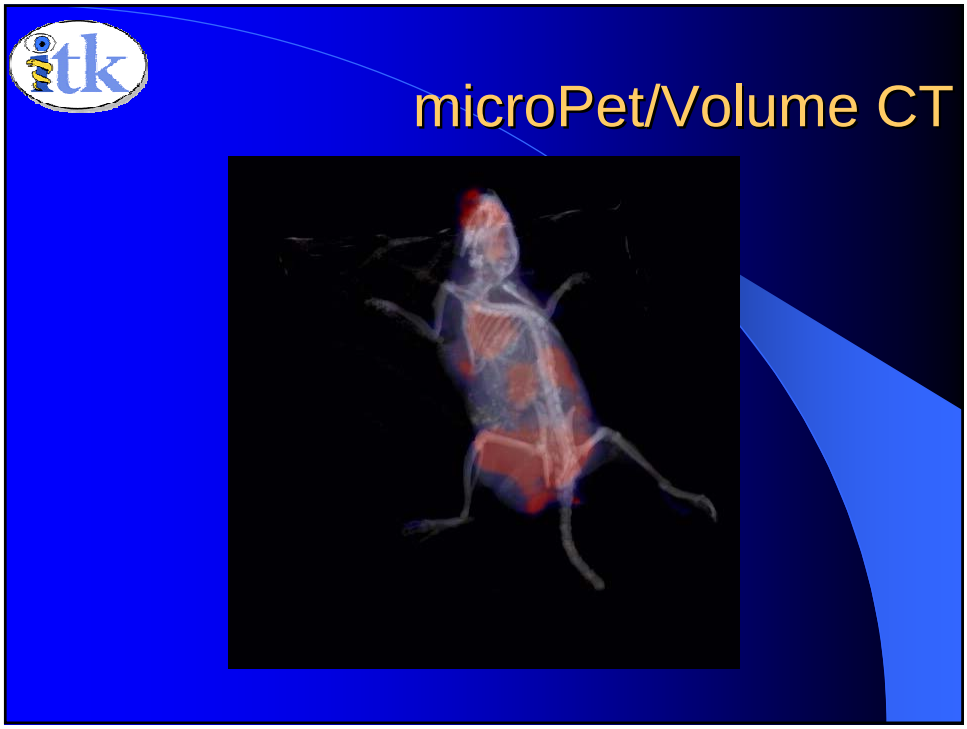

Visualization Toolkit - Win32OpenGLImage #1
Checkerboard
Difference
Target Original Image
Source Original Image
Lung CT No Registration

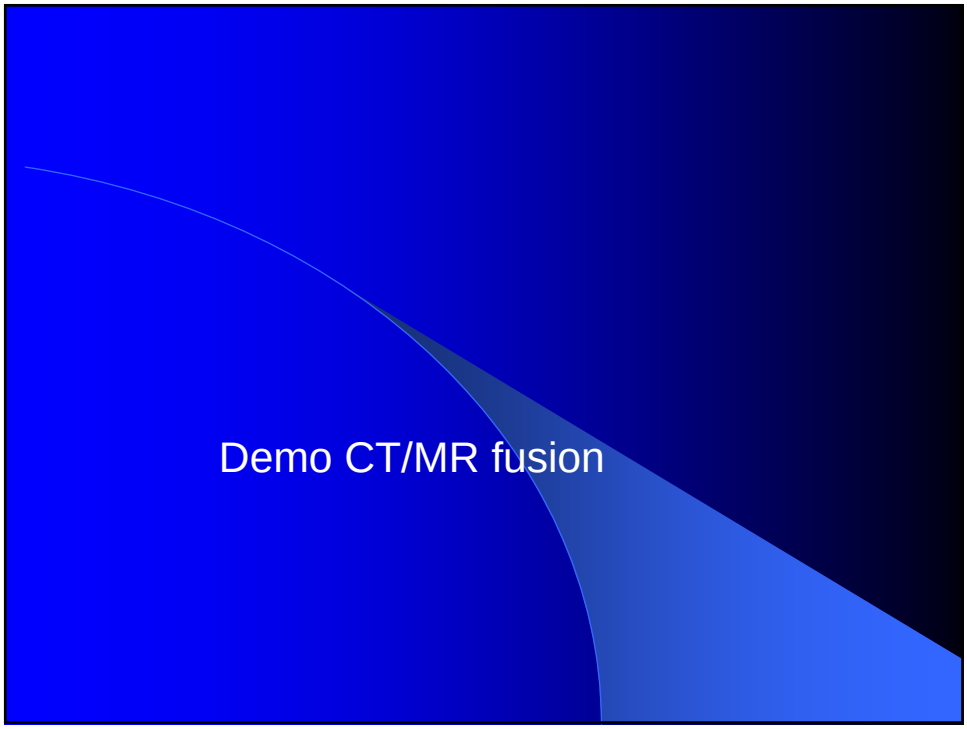

Visualization Toolkit - Win32OpenGLImage #1
Checkerboard
Difference
Target Original Image
Source Original Image
Lung CT Registration



microPet/Volume CT



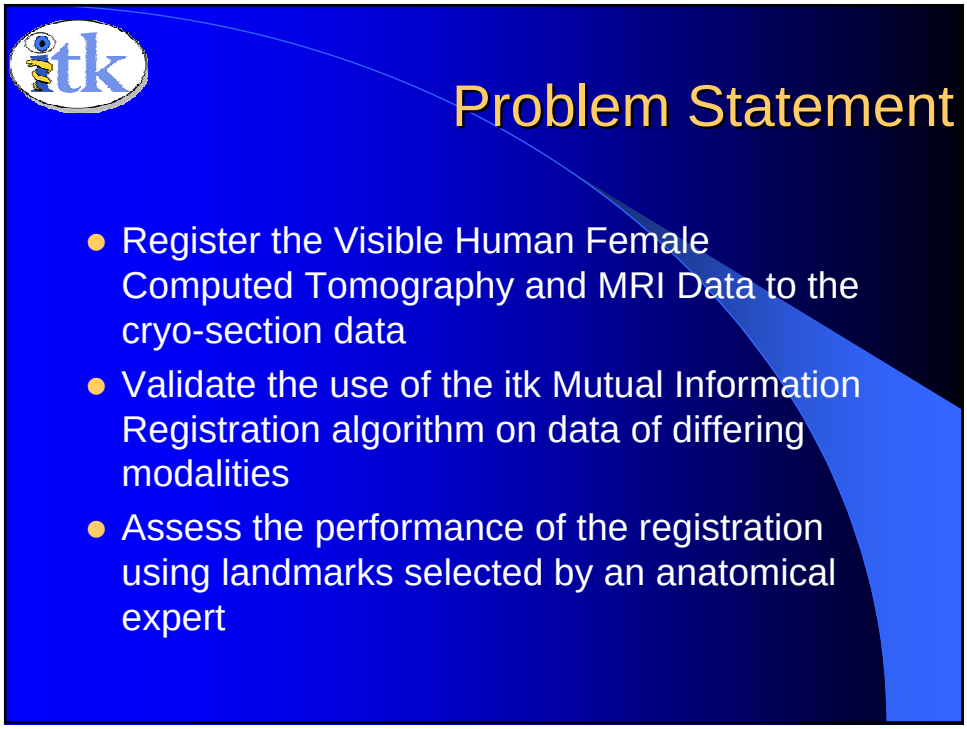

microPet/Volume CT

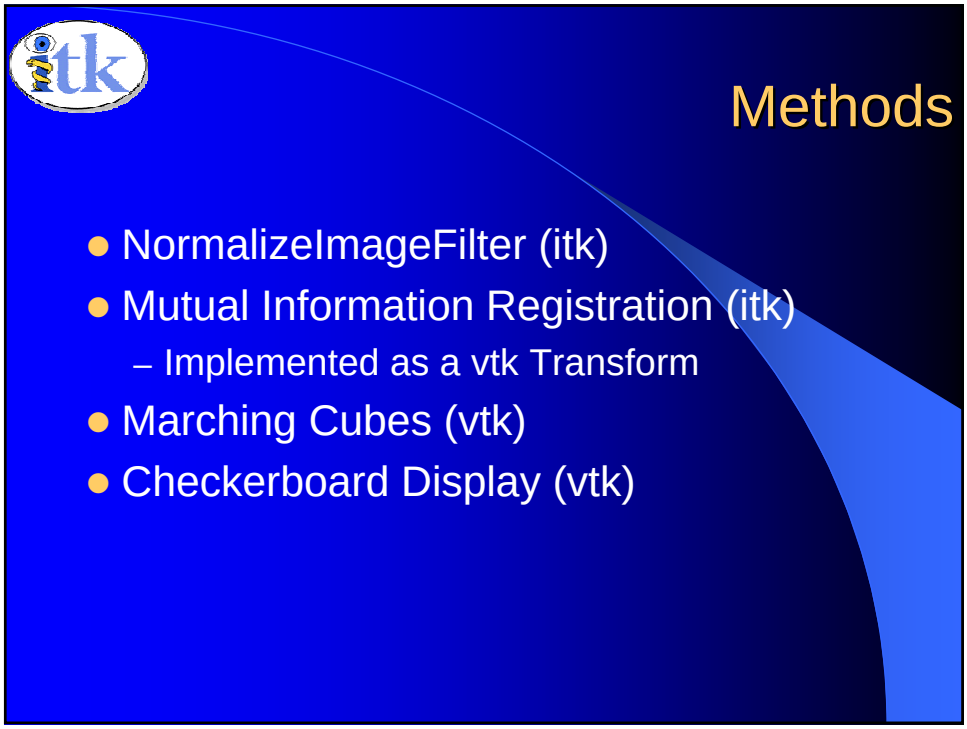

Demo CT/MR fusion

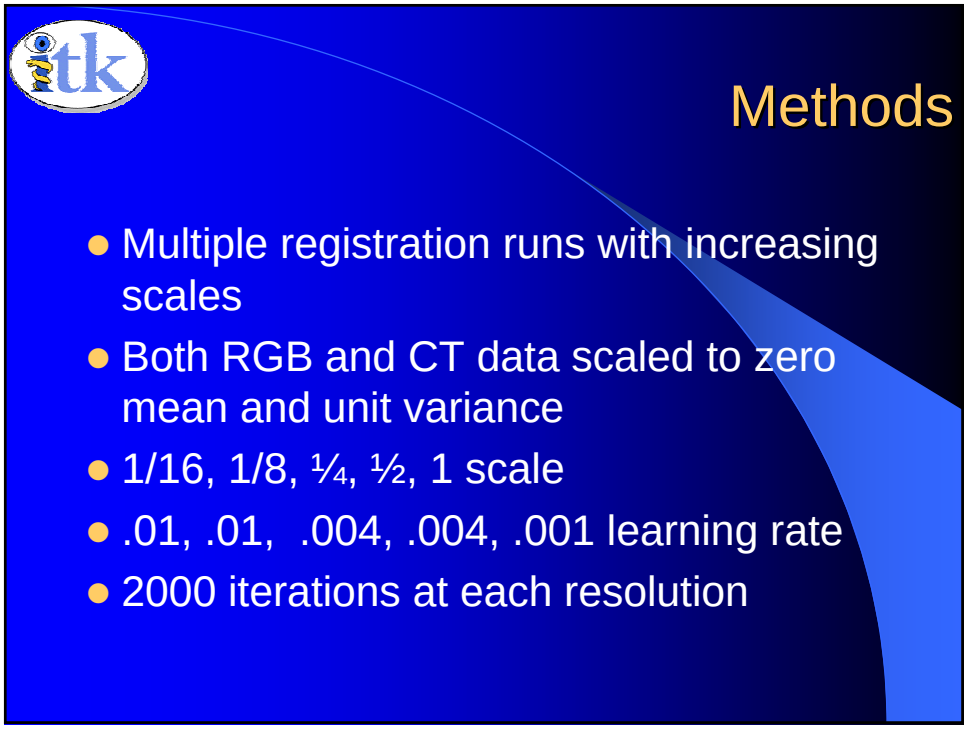

Implementing a vtk Transform with itk
vtkITKMutualInformationTransform
DemoMutual.tcl

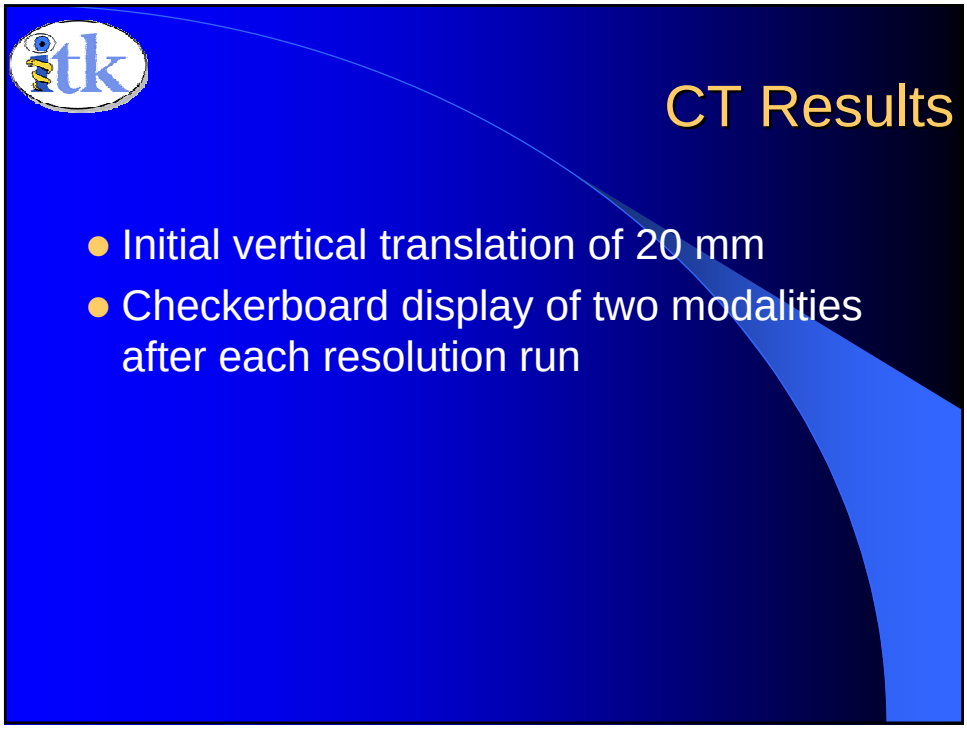

Visible Woman Registration Case Study
Bill Lorensen, GE Research
Peter Ratiu, Brigham and Womens Hospital

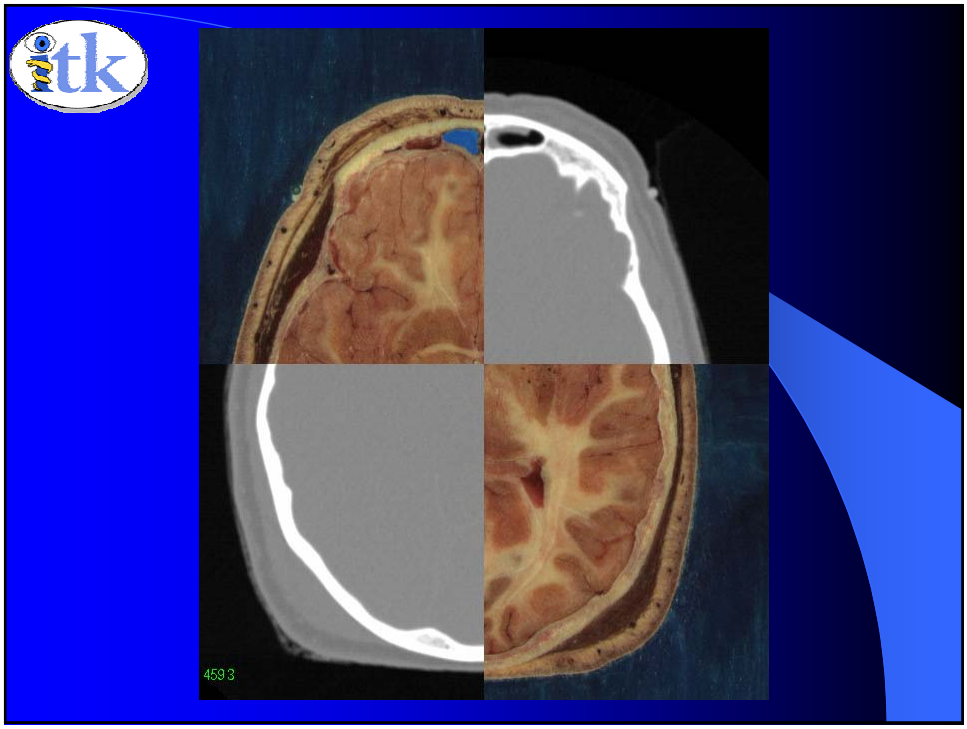

Agenda
Problem Statement
Methods
Results
Discussion

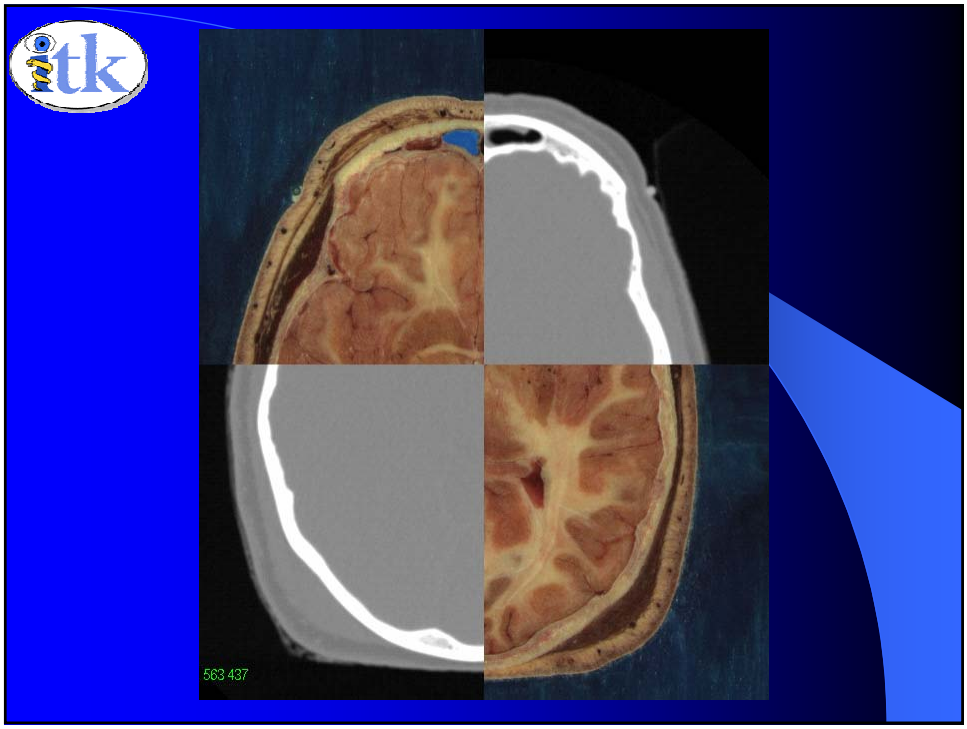

Problem Statement
Register the Visible Human Female Computed Tomography and MRI Data to the cryo-section data
Validate the use of the itk Mutual Information Registration algorithm on data of differing modalities
Assess the performance of the registration using landmarks selected by an anatomical expert

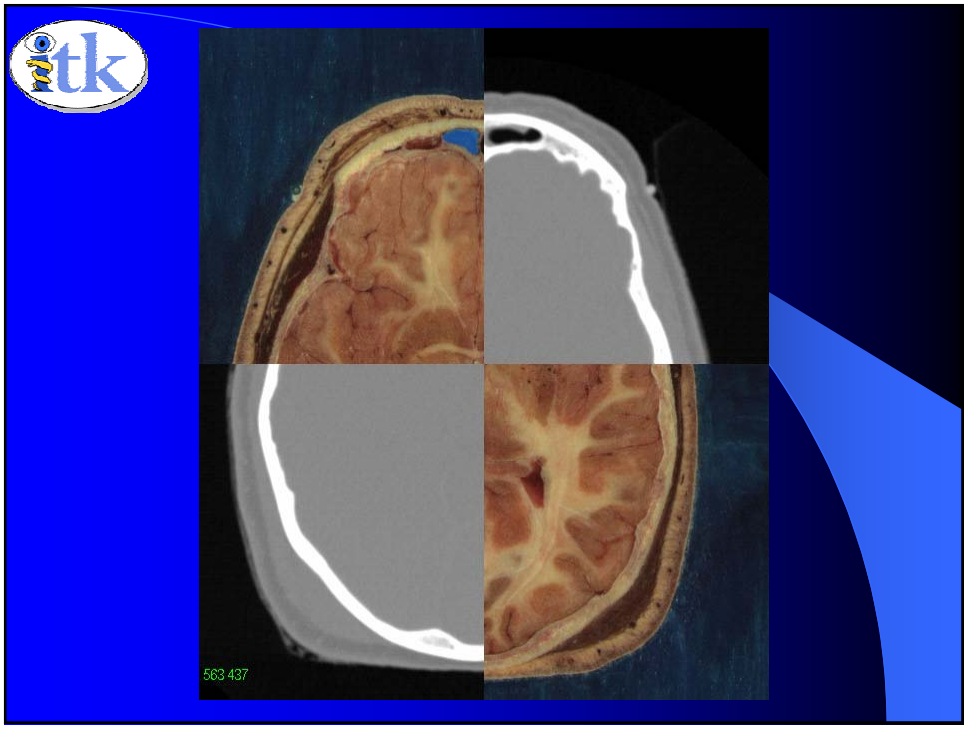

Methods
NormalizeImageFilter (itk)
Mutual Information Registration (itk)
– Implemented as a vtk Transform
Marching Cubes (vtk)
Checkerboard Display (vtk)

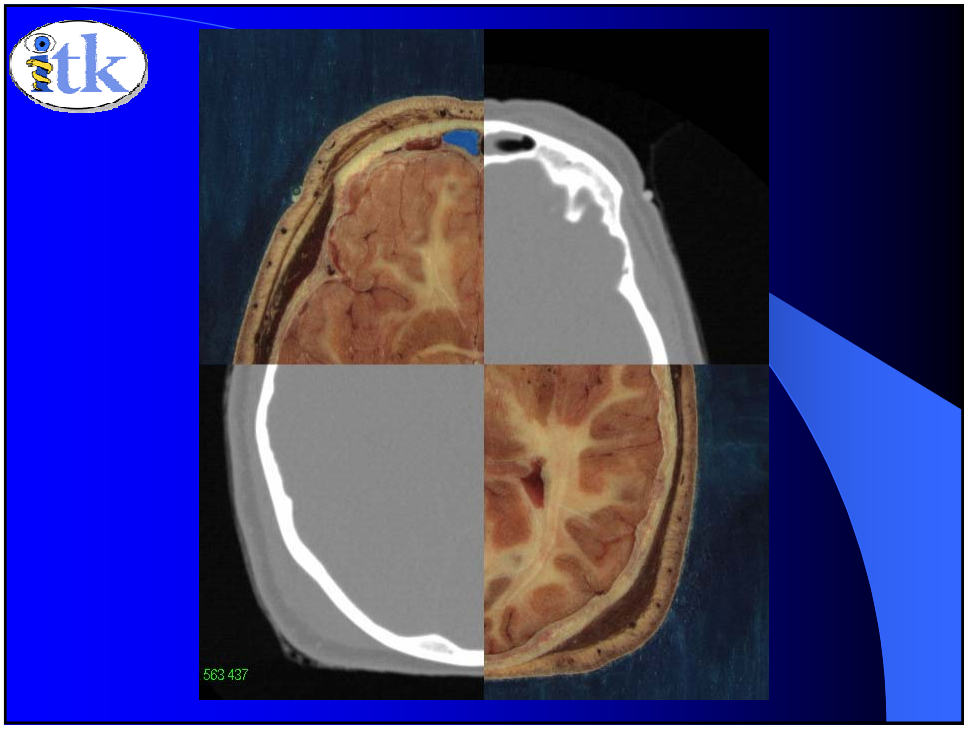

Methods
Multiple registration runs with increasing scales
Both RGB and CT data scaled to zero mean and unit variance
1/16, 1/8, ¼, ½, 1 scale
.01, .01, .004, .004, .001 learning rate
2000 iterations at each resolution

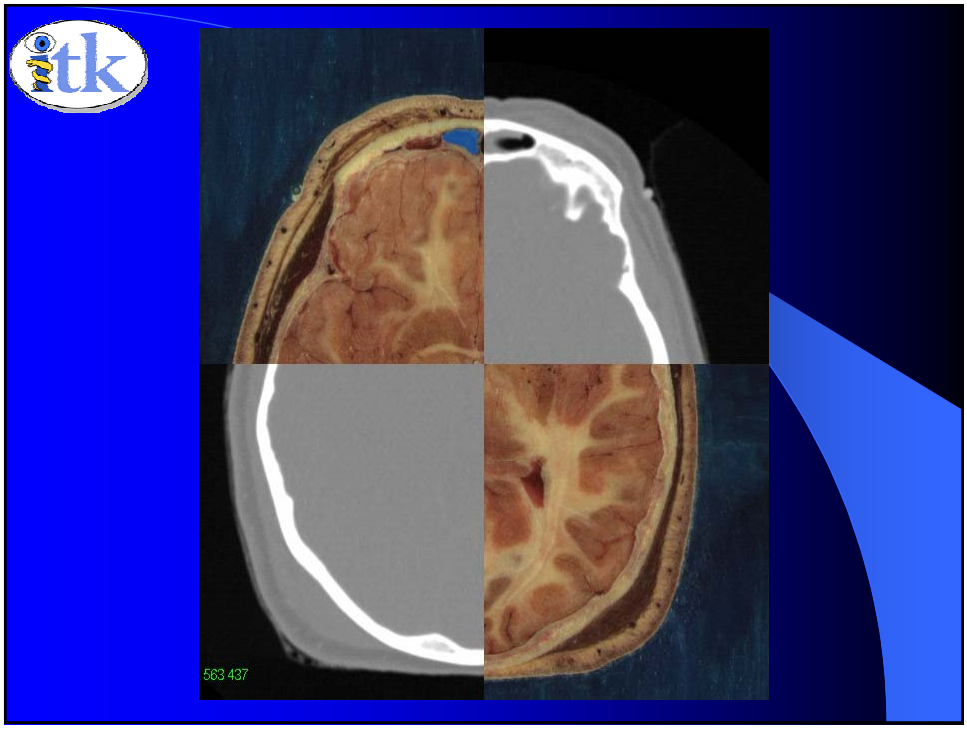

CT Results
Initial vertical translation of 20 mm
Checkerboard display of two modalities after each resolution run

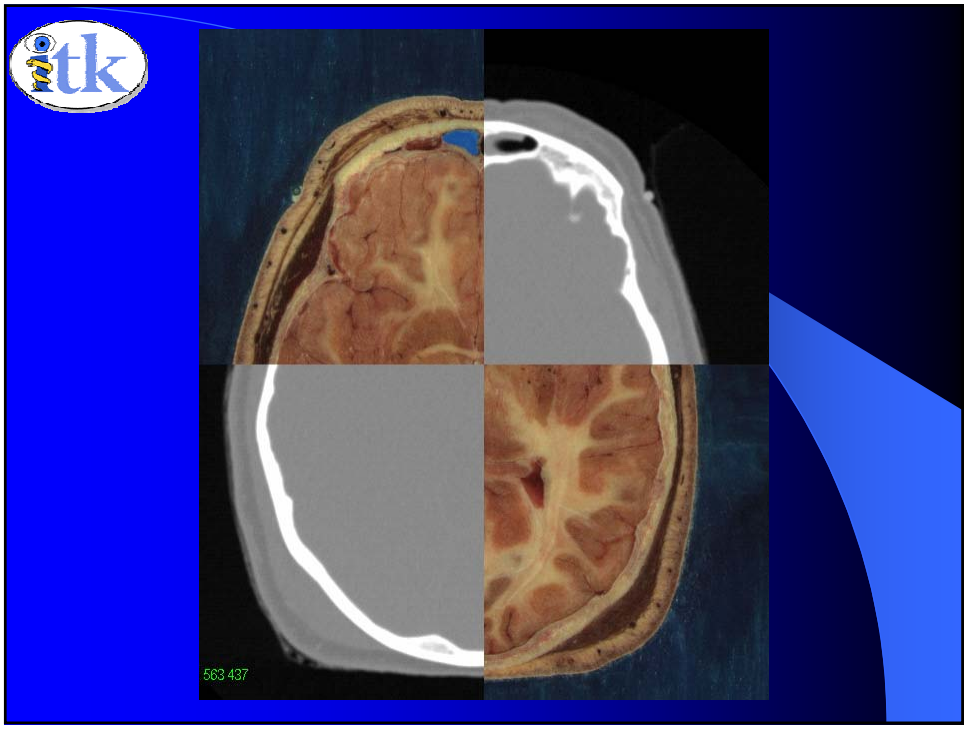

4593

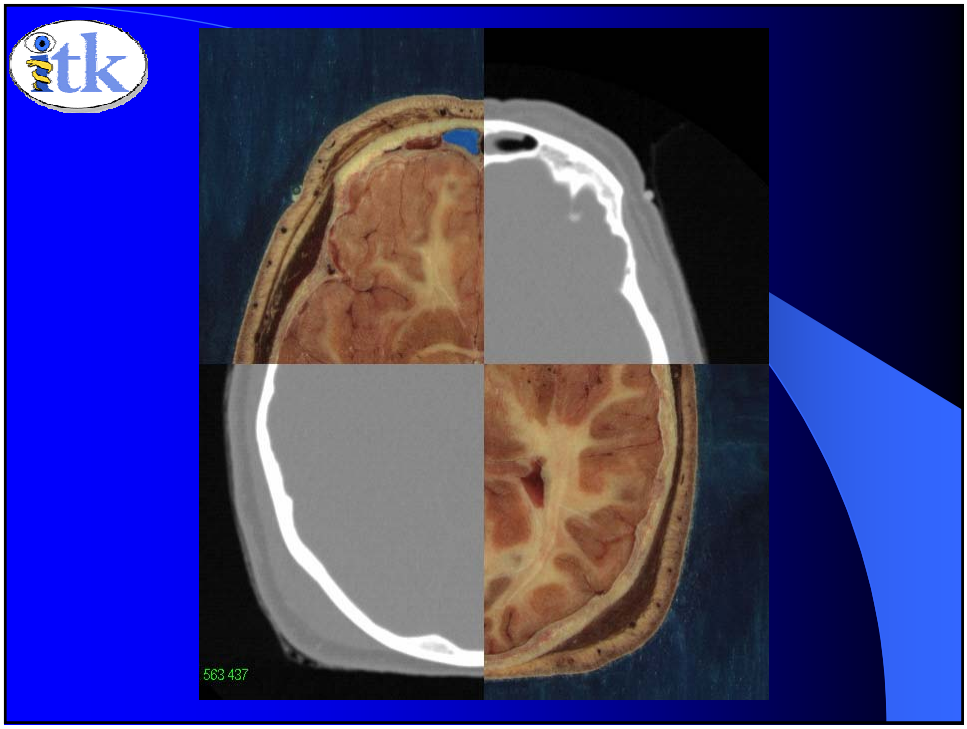

563 437

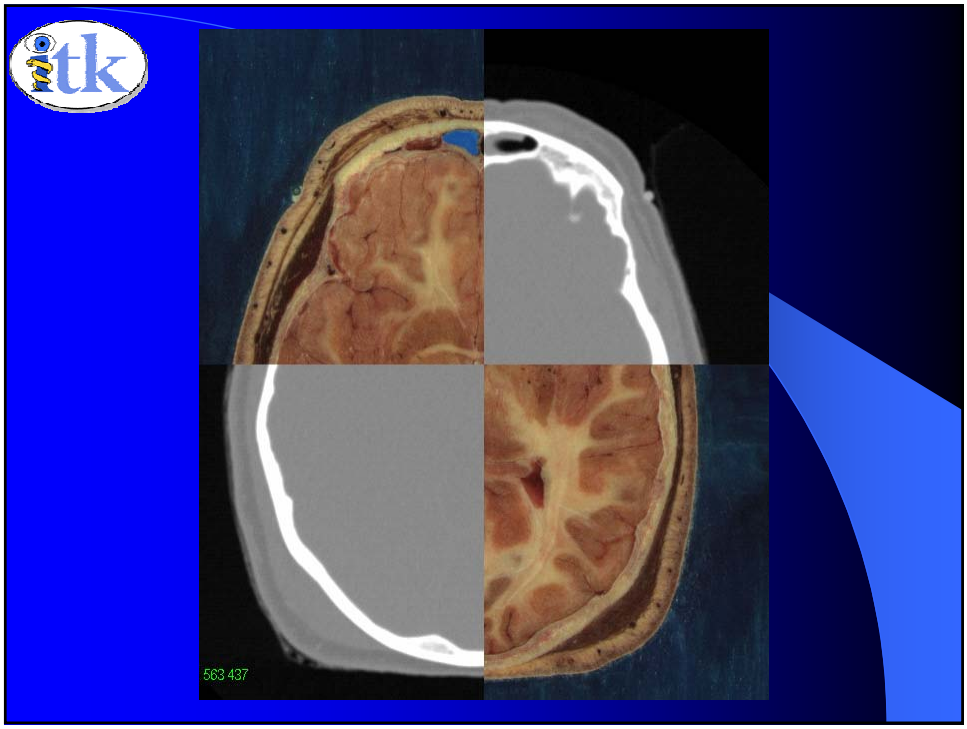

563 437

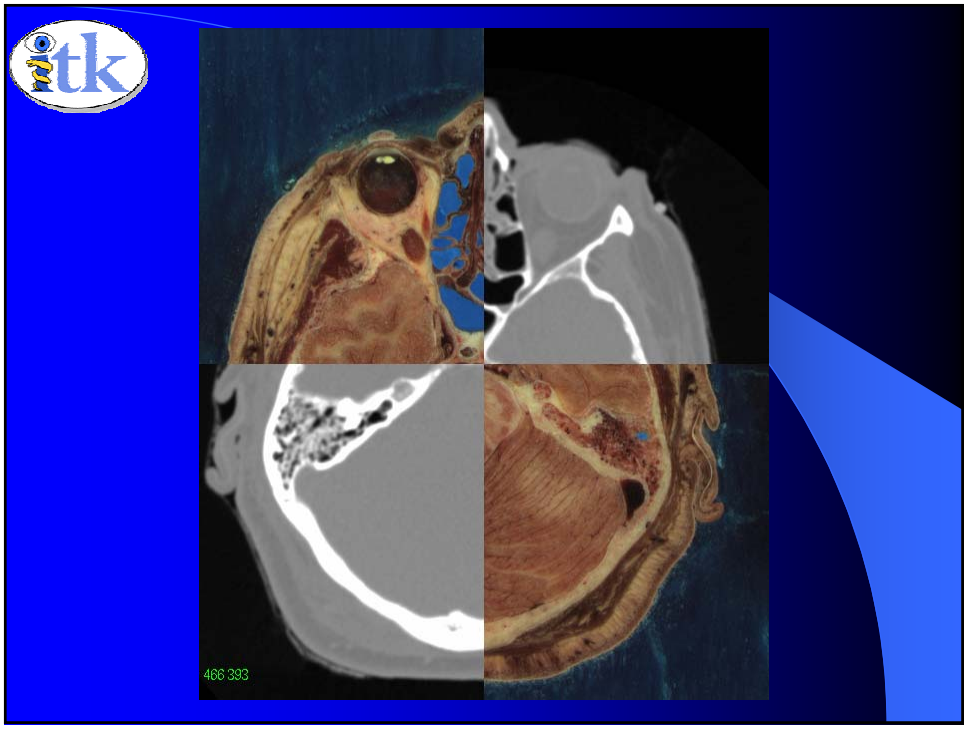

563 437

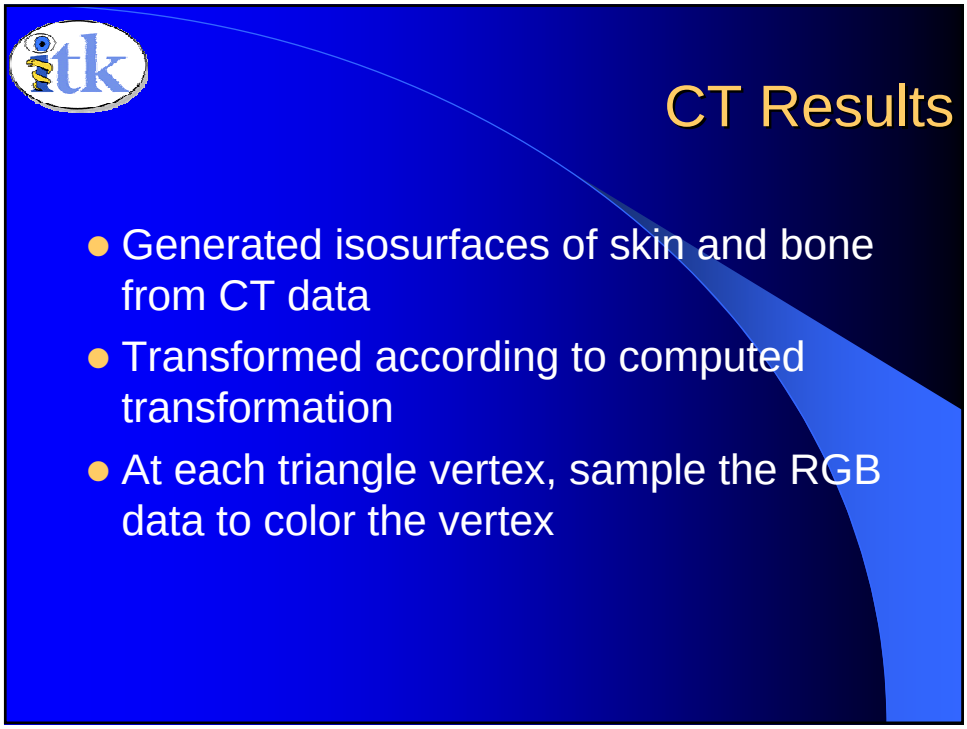

563 437


563 437


563 437

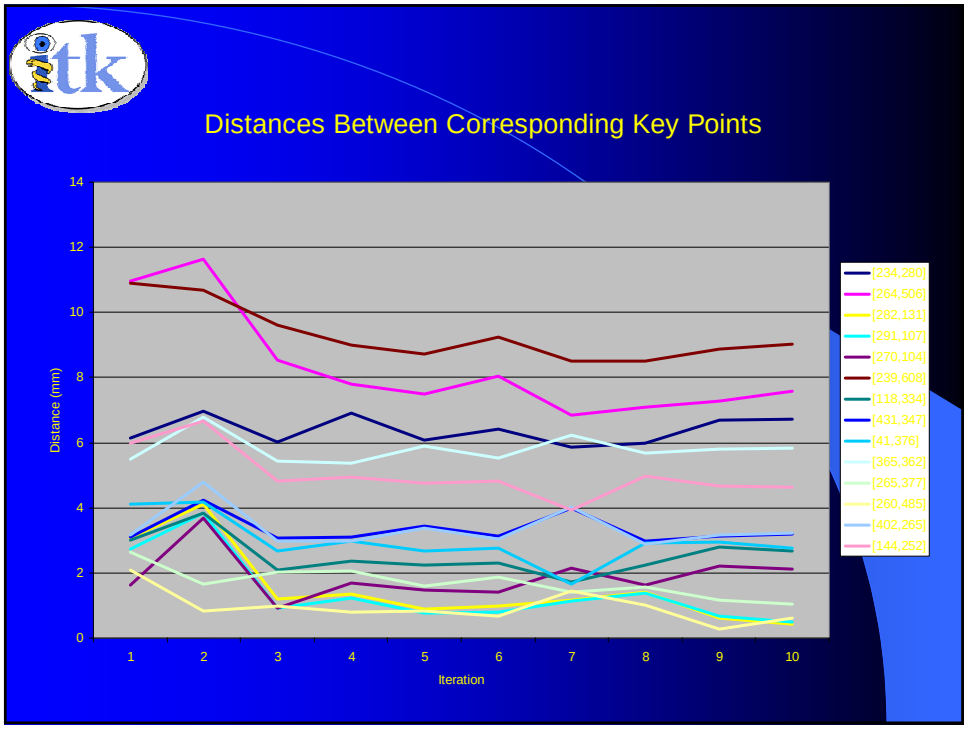

563 437


466 393


CT Results
Generated isosurfaces of skin and bone from CT data
Transformed according to computed transformation
At each triangle vertex, sample the RGB data to color the vertex

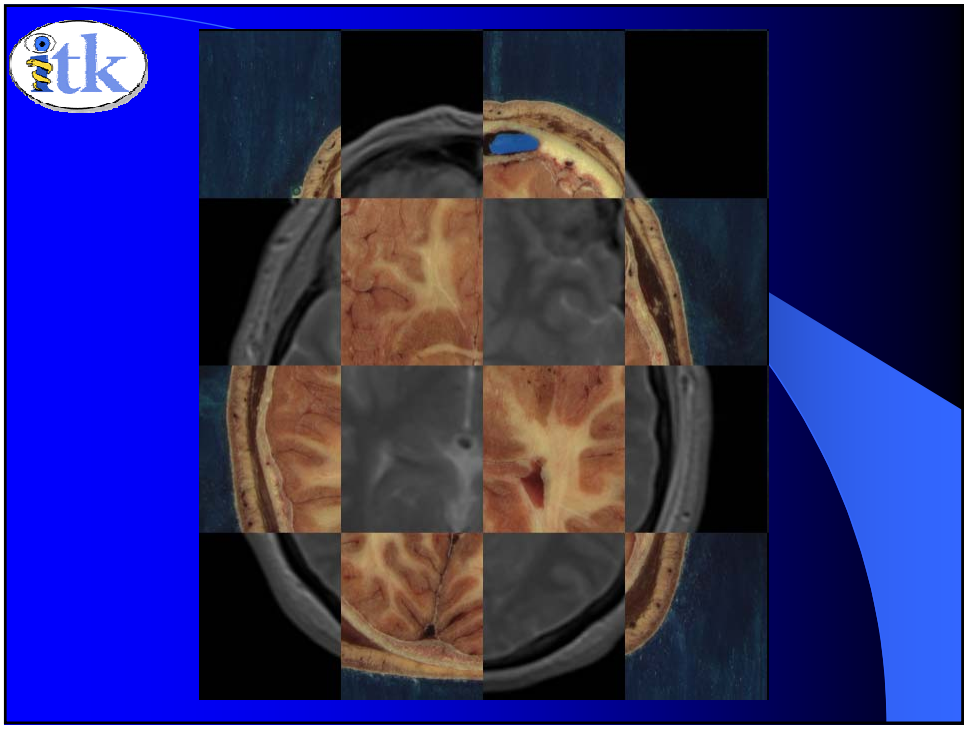
CT Results

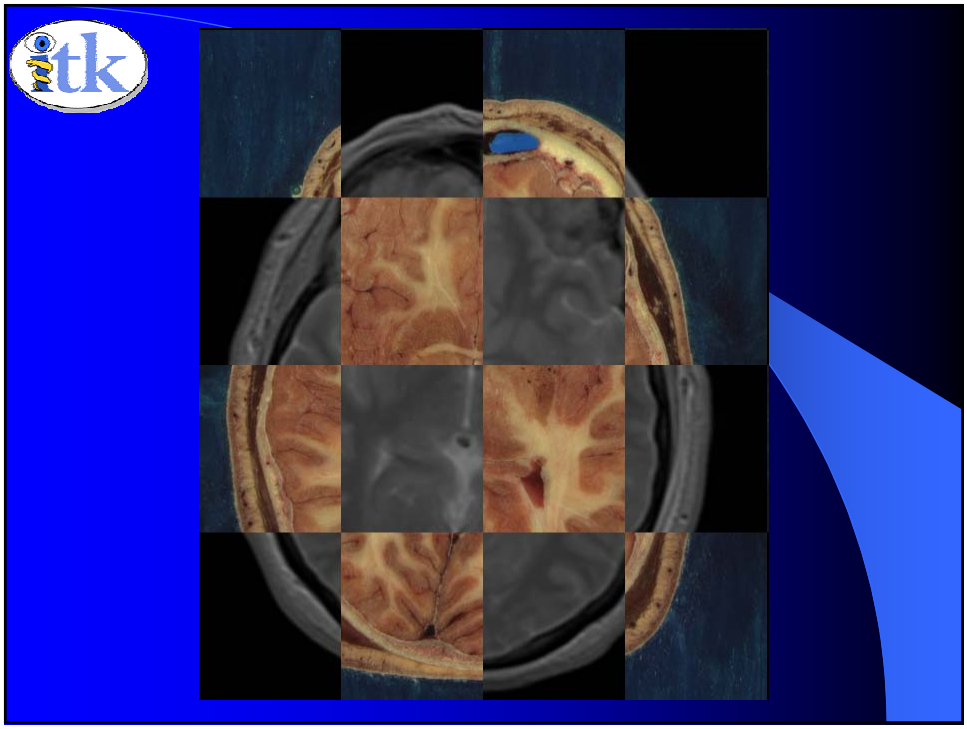
- CT Landmarks transformed by Mutual Information matrix into RGB coordinate system
- Compute distance between CT and RGB points at each iteration

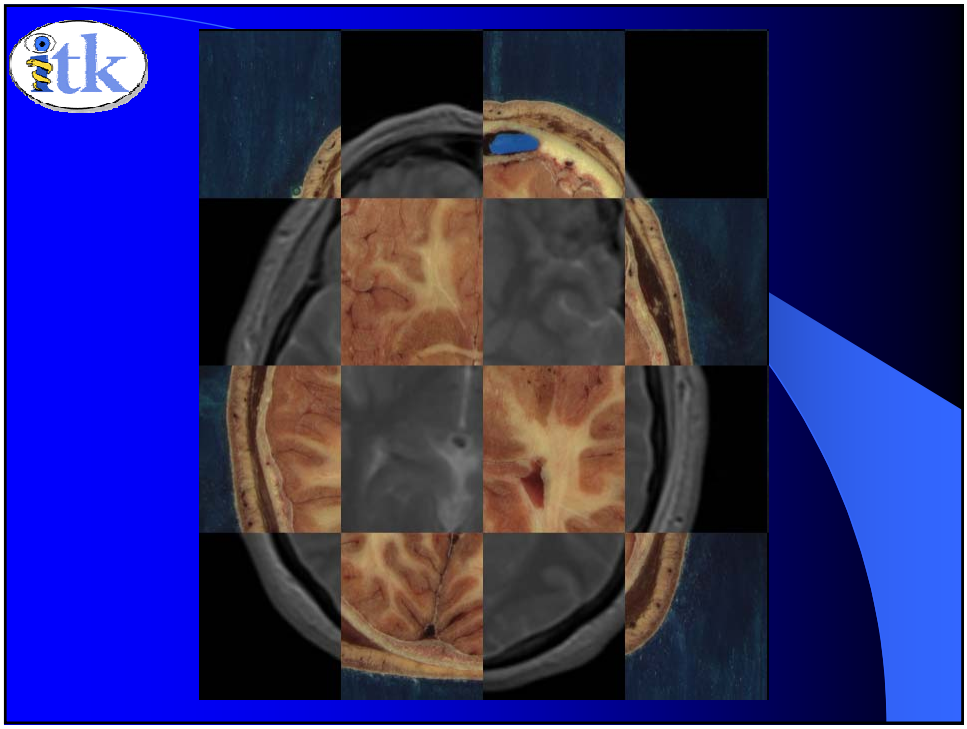


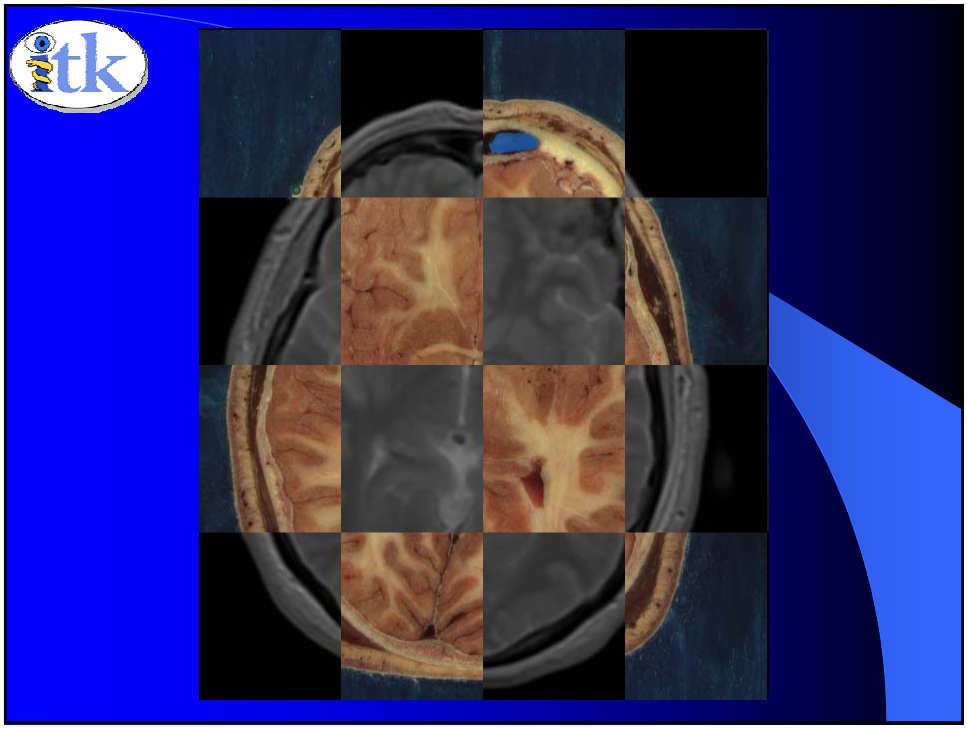
MR Results

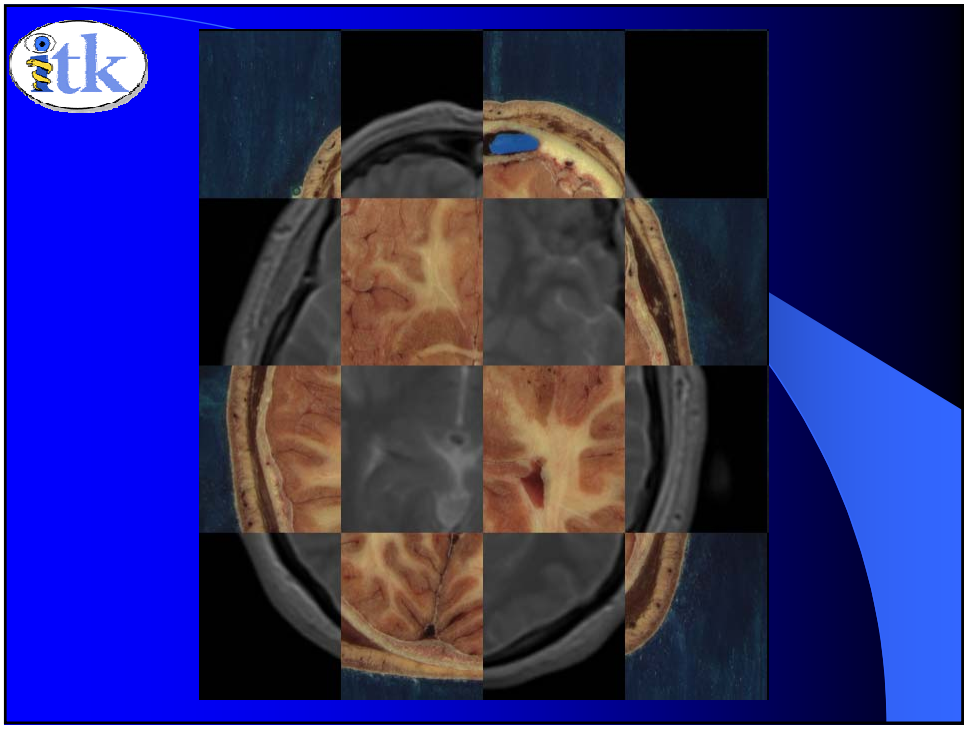
- Initial transform computed using experts anatomical landmarks as correspondences
- Same parameter schedule as CT/RGB

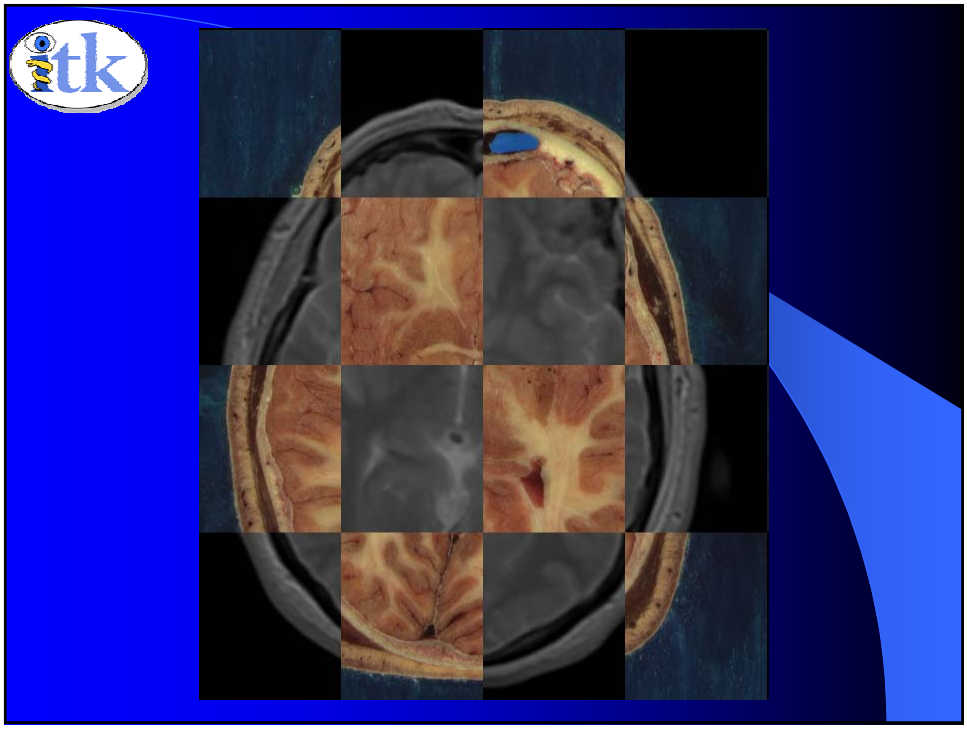


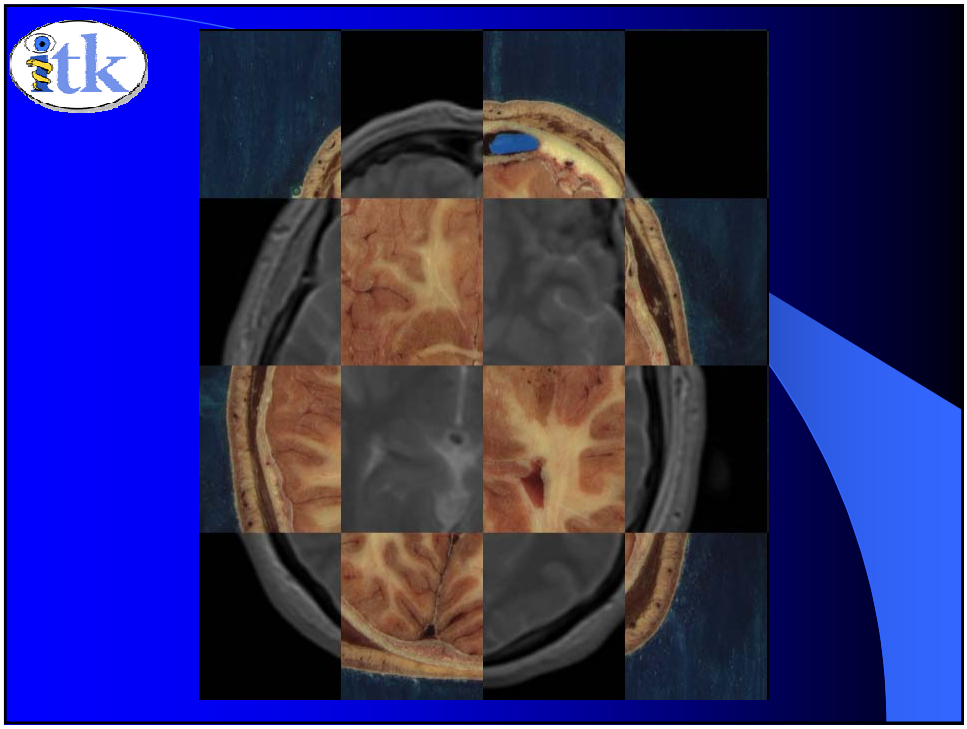


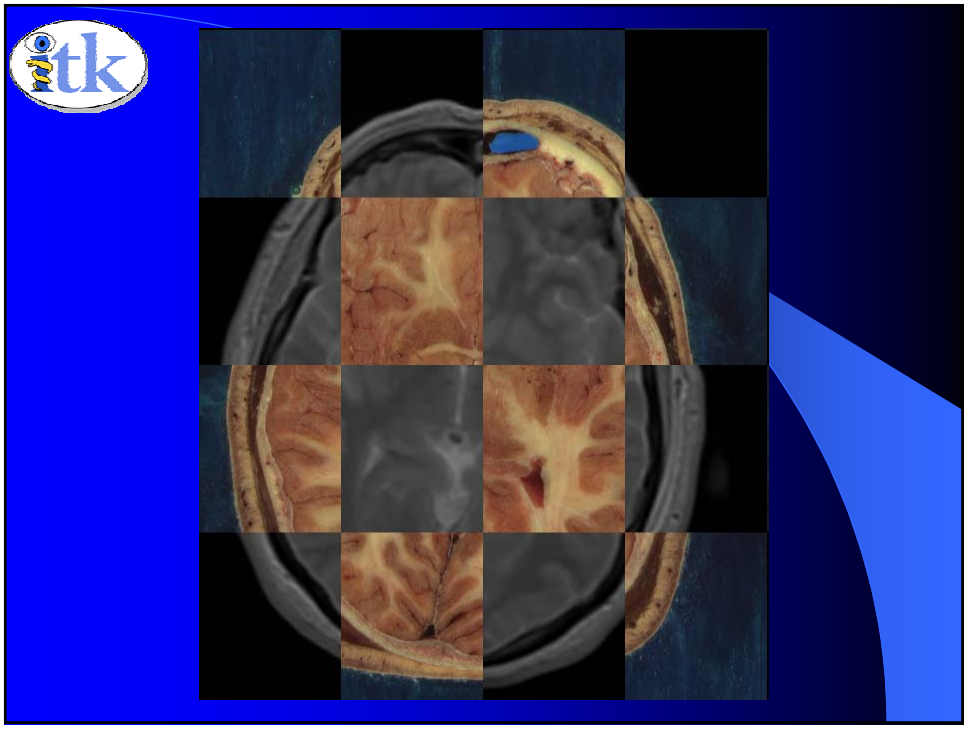


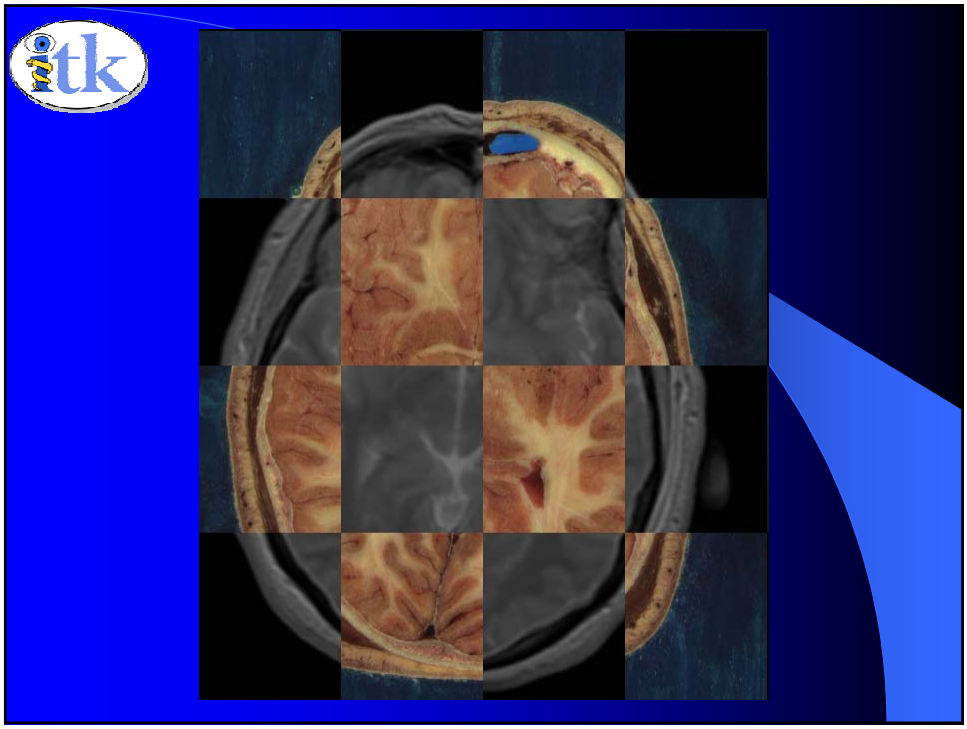






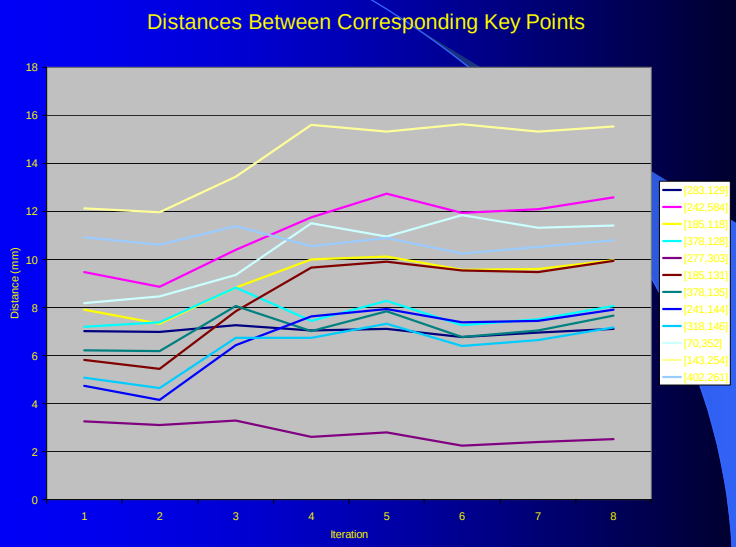






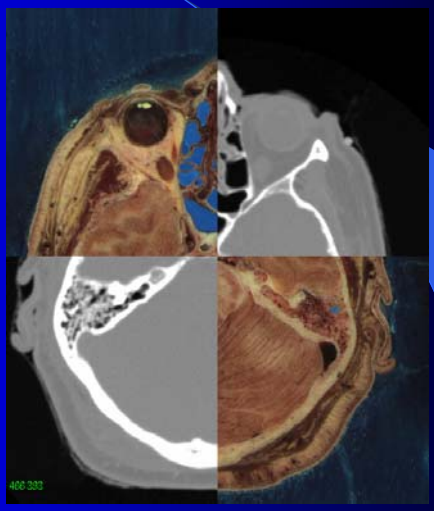
MRI Results

- MRI Landmarks transformed by Mutual Information matrix into RGB coordinate system
- Compute distance between MRI and RGB points at each iteration



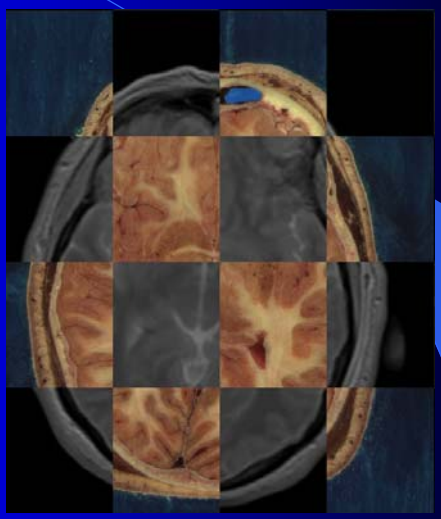
CT Discussion

- CT to RGB rigid body registration is a good first step for subsequent deformable registration



MRI Discussion

- MRI to RGB registration was not effective
  - Too much anatomical distortion occurred between the MR and RGB acquisitions



Examples

Segmentation

Watershed Segmentation

- I/O and Visualization with vtk
- Image processing and segmentation with itk
- User interface with tk
- Prototype in tcl

Demo Watershed

Applications:
Using Insight(itk) and
The Visualization Toolkit(vtk)

Bill Lorensen
Jim Miller
GE Research
Niskayuna, NY

ITK Meets the Virtual Soldier

GE Global Research
Bill Lorensen, Jim Miller
Dirk Padfield, James Ross
Wes Turner
lorensen@crd.ge.com

Outline

- Atlas Quality Control
- Model Generation
- Implicit Anatomical Modeling
- Registration
- Model Refinement Usinig Level Sets

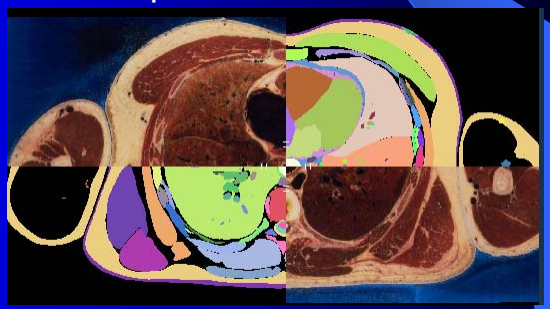
Activities

- Atlas from NLM
- Reconciled labels
- Registered slices
- Computed bounding boxes
- Calculated adjacencies
- Generated models using vtkDiscreteMarchingCubes
- Colored models
- Registered atlas to CT
- Atlas as seed for level sets
- Deformation of man to woman

Atlas Quality Control

- Thorax Atlas consists of 411 70mm camera cryo-section slices and corresponding segmentation label maps

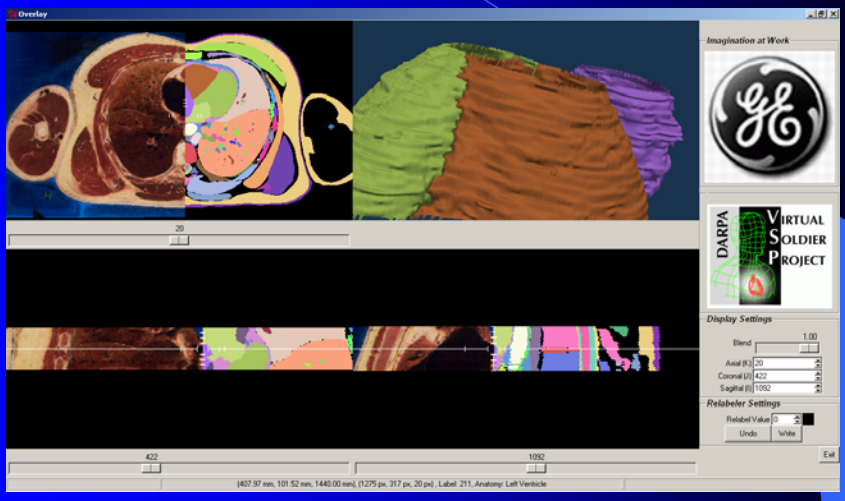
Slice Resolution: 2048 x 1350



Atlas Quality Control

- Labeled 407 structures
- Registered 70mm film images
- Reconciled labels throughout entire volume
- Computed bounding boxes for all labels
- Removed some “dusty” structures
  - Automatic reassignment

Overlay.tcl

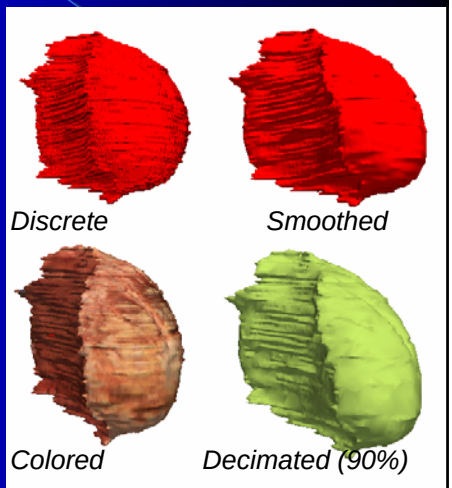


Atlas Quality Control

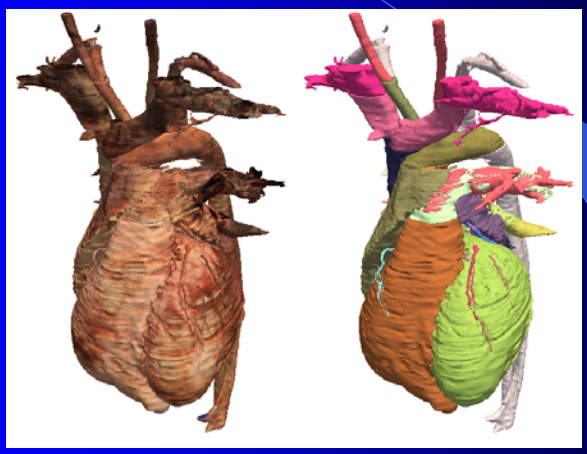
Segmentation Dust removed via "voting"

Model Generation

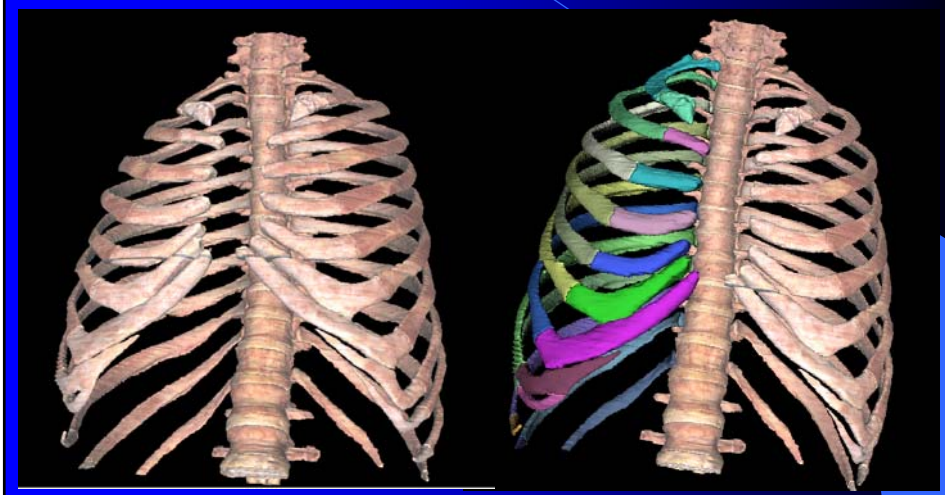
- Generated four types of models from segmentation labels



Model Generation



Model Generation



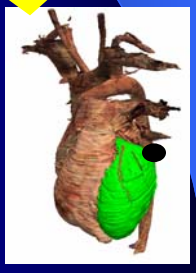
February 2005 SPIE Tutorial

VHM Thorax Models

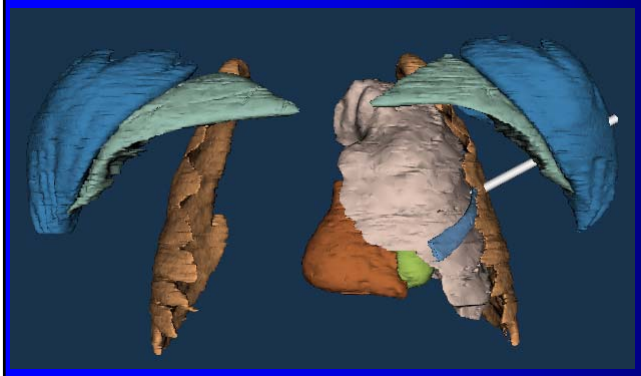
Implicit Anatomical Modeling

- Label = f(X,Y,Z)

X,Y,Z → I,J,K → Label

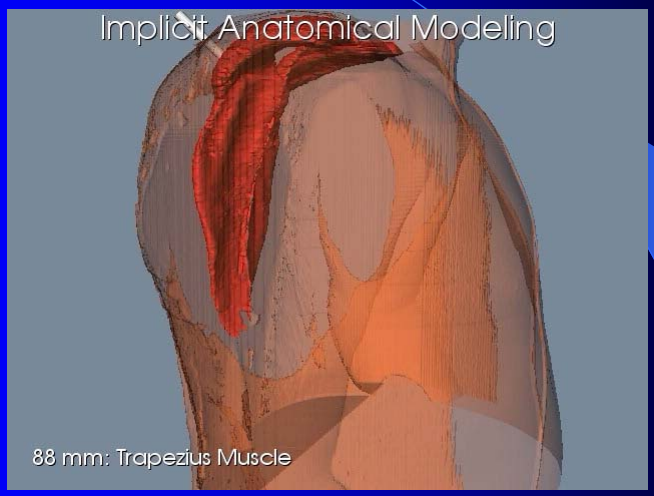


Implicit Anatomical Modeling

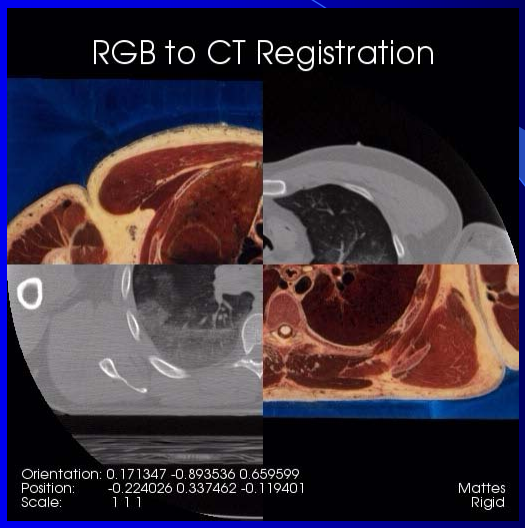


0 mm: right_ventricle
2 mm: pericardium
3 mm: left_ventricle
52 mm: mediastinum
53 mm: left_upper_lobe_of_lung
66 mm: superior_lingular_branch_left_pulmonary
68 mm: left_upper_lobe_of_lung
107 mm: unknown
112 mm: left_fourth_rib
117 mm: serratus_anterior_muscle
123 mm: unknown
186 mm: pectoralis_minor_muscle
191 mm: deltoid_muscle
259 mm: unknown
261 mm: subcutaneous_tissue_of_trunk
271 mm: skin_of_trunk

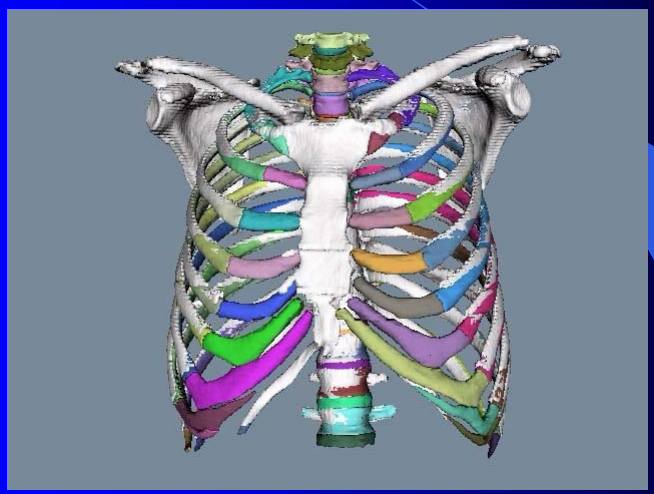
Implicit Anatomical Modeling

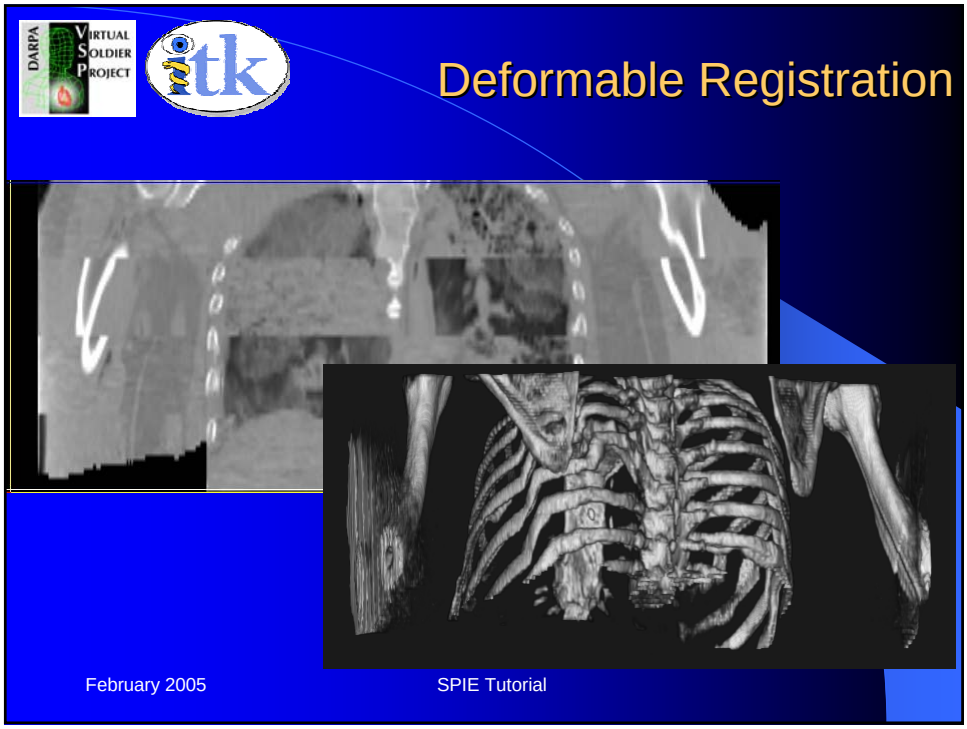


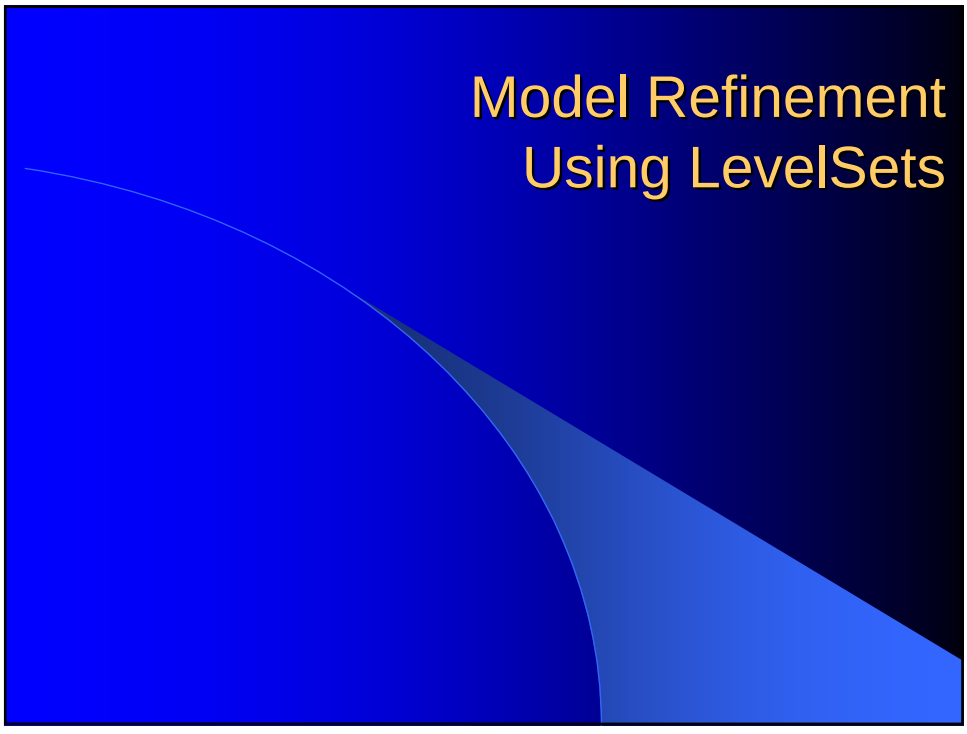
RGB to CT Registration

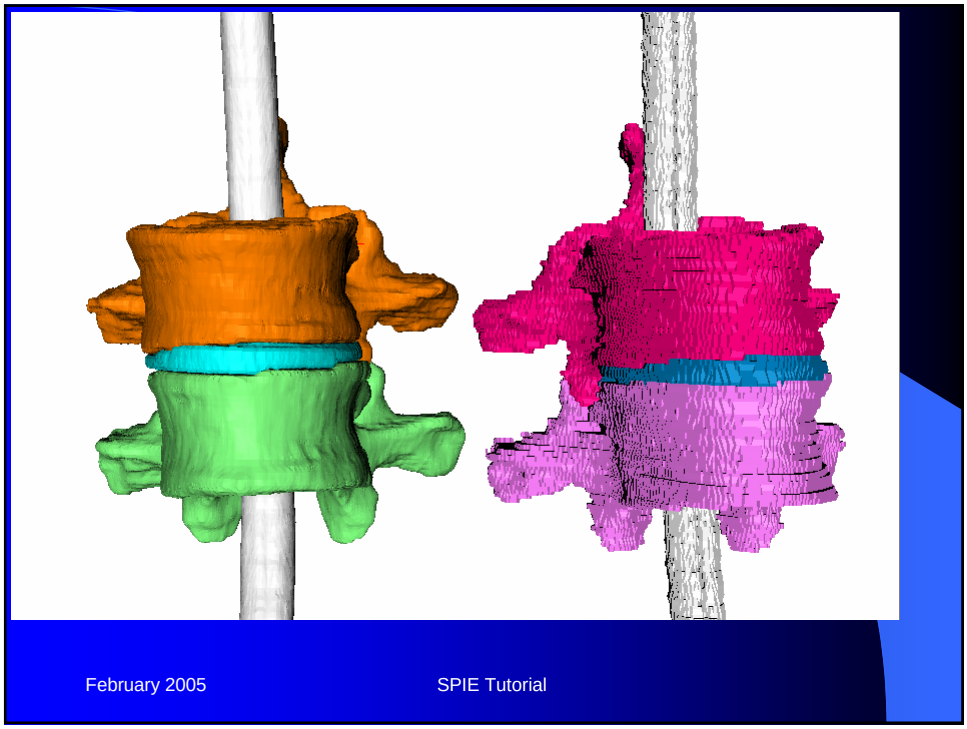


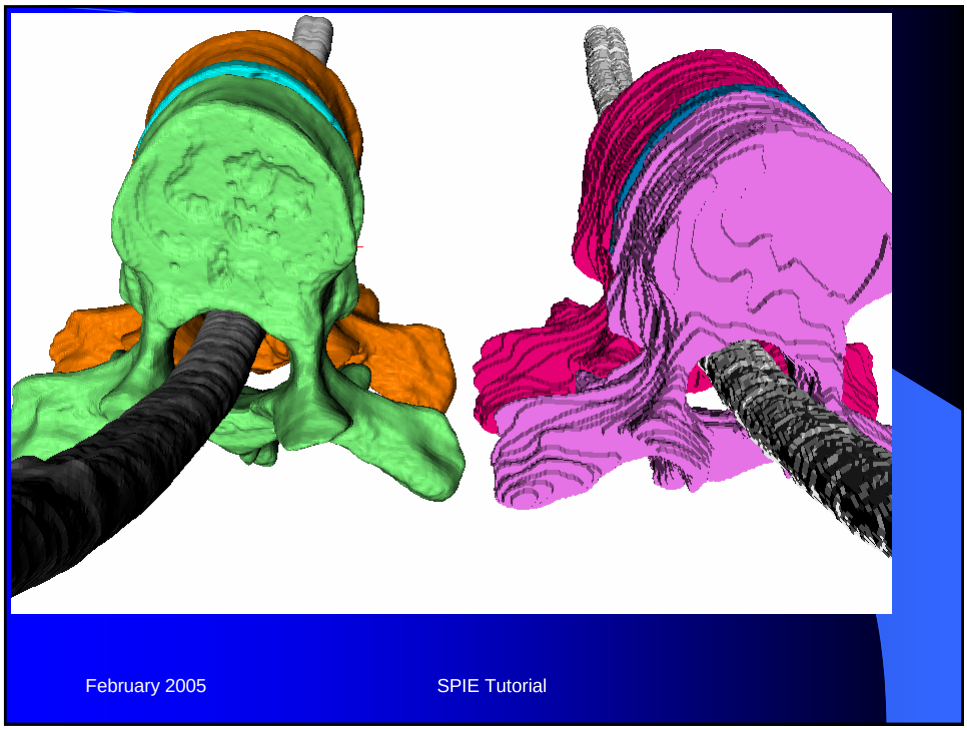
RGB to CT Registration




Deformable Registration
February 2005
SPIE Tutorial


Model Refinement
Using LevelSets





LevelSet Model Refinement

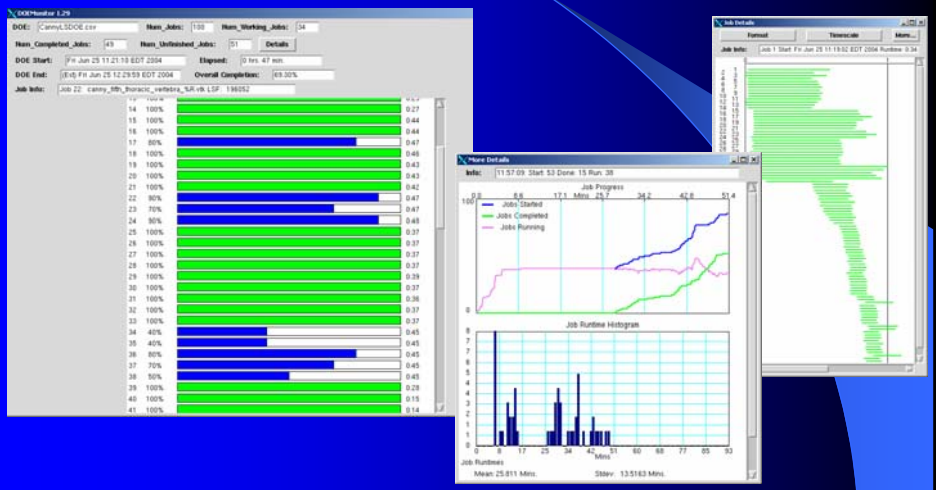
- Powerful technique to track the evolution of surfaces using image based constraints and forces
- Three techniques investigated
  - Threshold Level Set
  - Laplacian Level Set
  - Canny Level Set
- All techniques are part of Insight Toolkit (itk)

But: Algorithm parameter space is high!

Load Sharing Facility

- 256 3.2 Ghz Dual Processor Xeon
- 2 gig of memory each
- Jobs scheduled with Load Sharing Facility (LSF)
  - Platform Computing Corporation
- LSF Toolkit
  - GE Research front-end and monitoring

LSF Toolkit

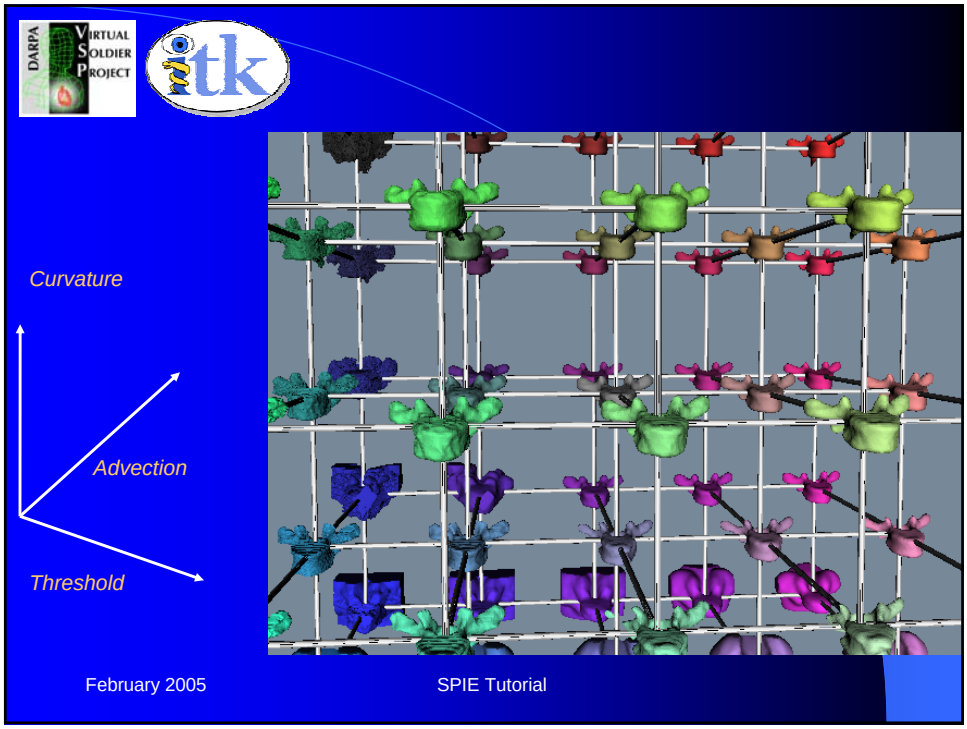


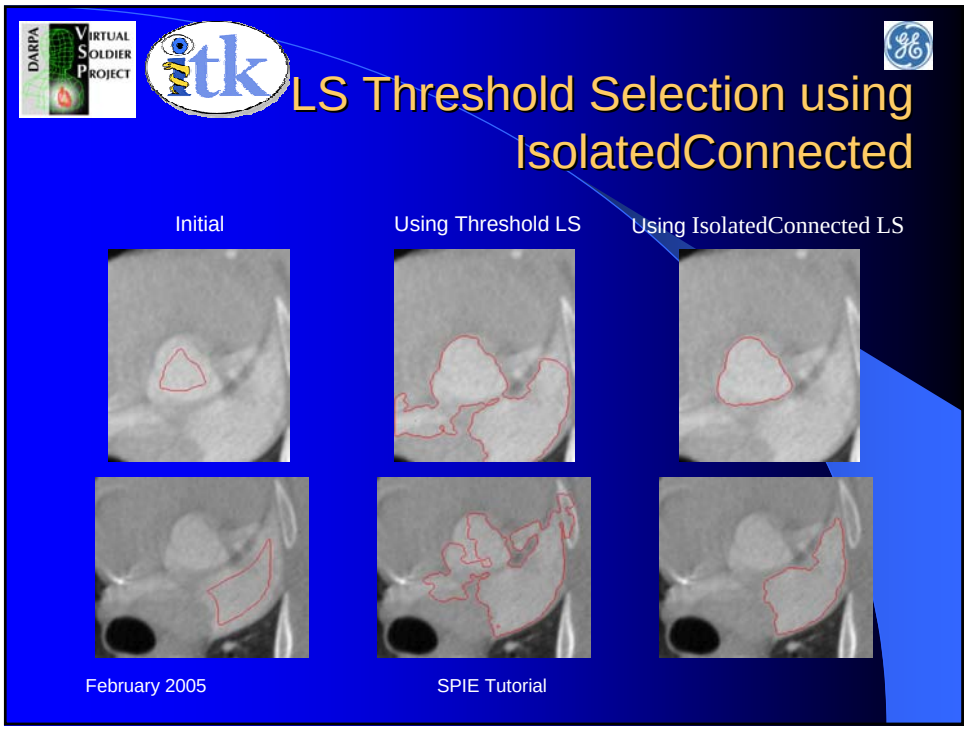
Laplacian LevelSet DOE

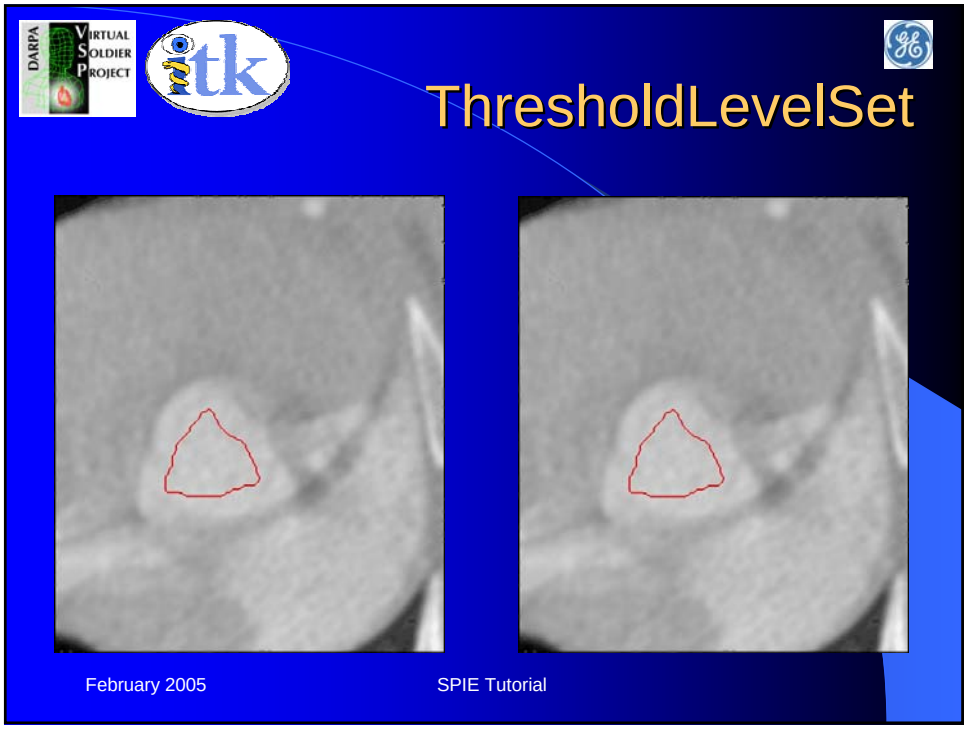
Curvature

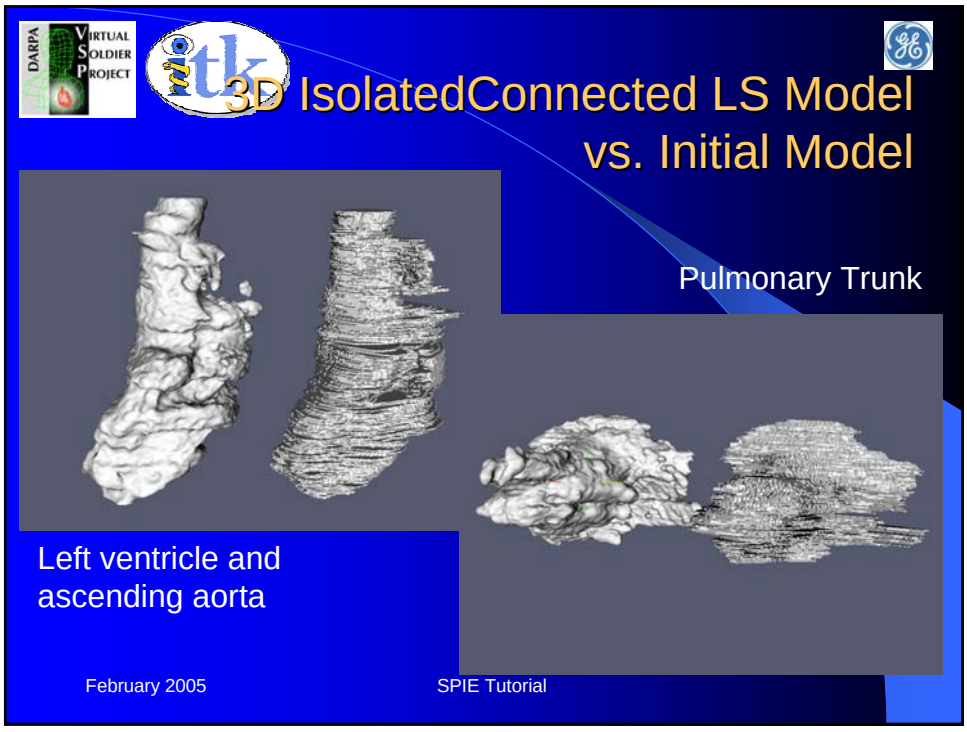
Advection

Threshold


Curvature
Advection
Threshold
February 2005
SPIE Tutorial


LS Threshold Selection using IsolatedConnected
Initial
Using Threshold LS
Using IsolatedConnected LS
February 2005
SPIE Tutorial


ThresholdLevelSet
February 2005
SPIE Tutorial


3D IsolatedConnected LS Model vs. Initial Model
Pulmonary Trunk
Left ventricle and ascending aorta
February 2005
SPIE Tutorial

ITK Meets the Virtual Soldier

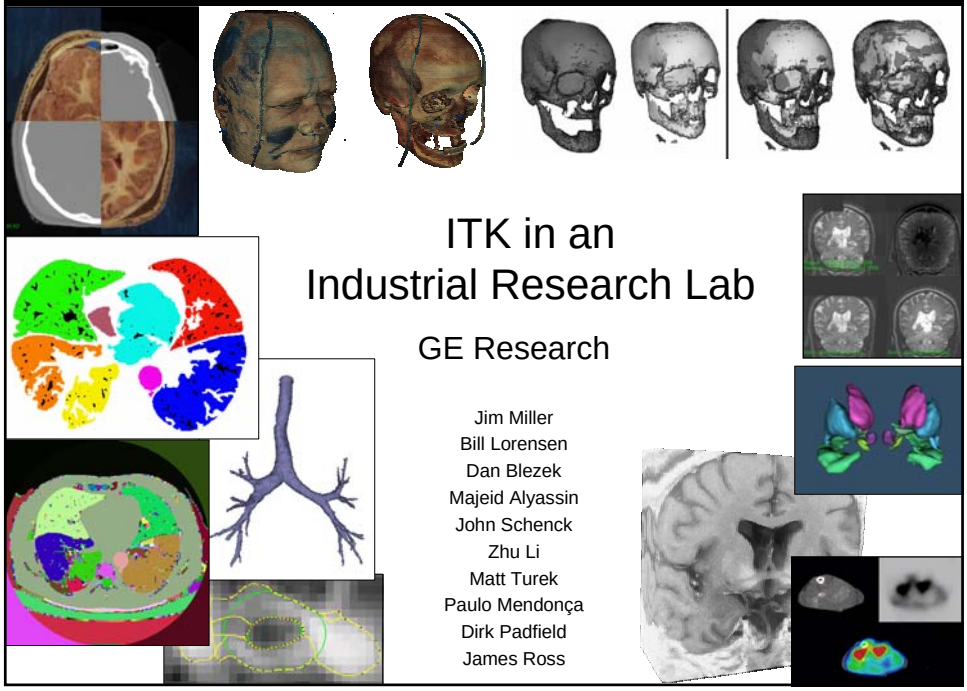
GE Global Research

Bill Lorensen, Jim Miller

Dirk Padfield, James Ross

Wes Turner

lorensen@crd.ge.com

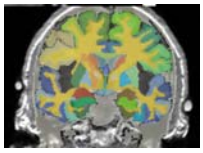


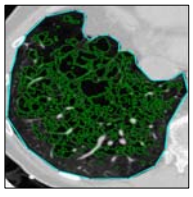
ITK in an Industrial Research Lab

GE Research

Jim Miller
Bill Lorensen
Dan Blezek
Majeid Alyassin
John Schenck
Zhu Li
Matt Turek
Paulo Mendonça
Dirk Padfield
James Ross

ITK at GE Research

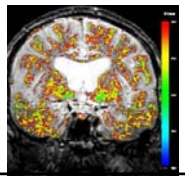


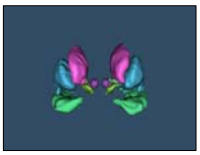


- Alzheimer's Research
  - Exploratory Research
  - Molecular Imaging
  - Diagnostic Pharmaceuticals
- Chronic Obstructive Pulmonary Disease
  - Research Tool Development/Product Development
  - Image Guided Clinical Trials
- Both projects are quantitative imaging projects
  - Reduce images to numbers
    - Volumes, Areas, Percent change
- Both projects are studying a disease from an imaging perspective
- Both projects use ITK along with other toolkits and applications

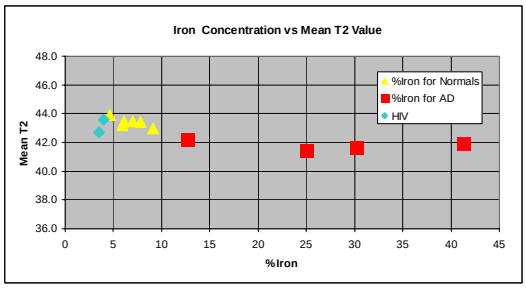
Alzheimer's Research

- Hypotheses:
  - Alzheimer's patients have elevated brain iron concentrations
    - B Drayer, et al., Am. J. Neuroradiol. 7, 373-380, 1986.
    - JF Schenck, J. Neurol. Sci 134 (Suppl.), 10-18, 1995.
  - Alzheimer's patients have volumetric differences in key brain structures
    - *"A discriminant function analysis demonstrated that a linear combination of the volumes of the hippocampus and the temporal horn of the lateral ventricles differentiated 100% of the patients and controls from one another."*
    - Killiany, et al., "Temporal Lobe Regions on Magnetic Resonance Imaging Identify Patients with Early Alzheimer's Disease". *Archives of Neurol*ogy. 50:949-954, 1993

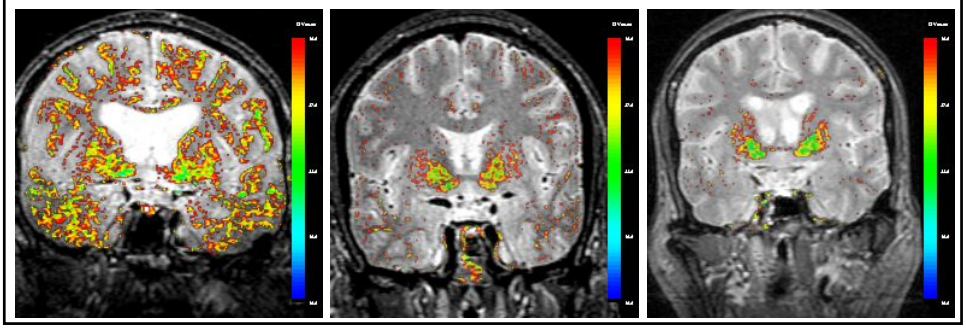




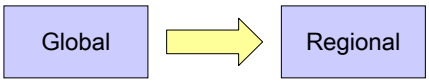
Brain Iron as AD Marker



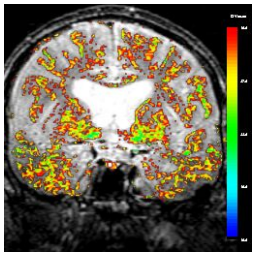
Alzheimer's Disease Normal HIV



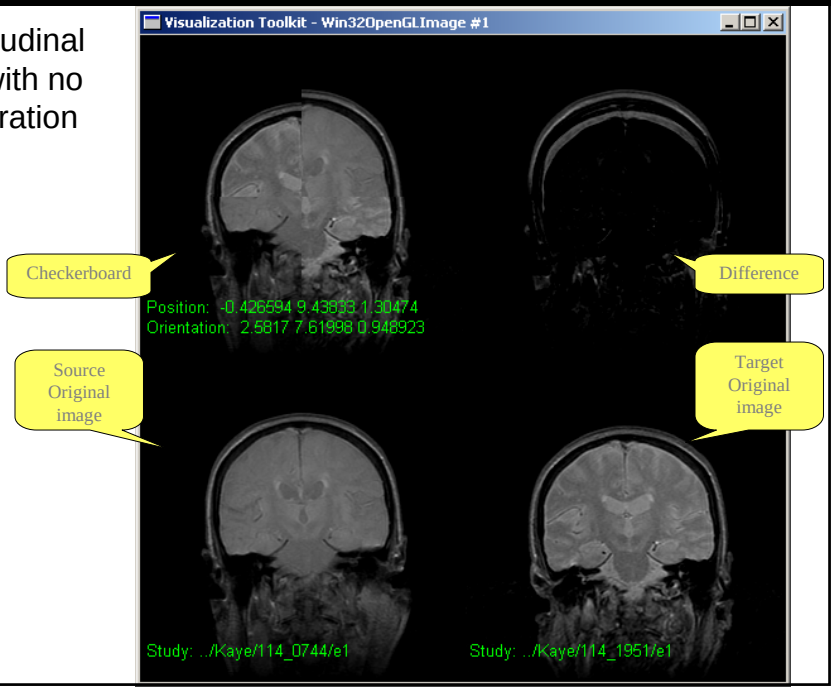
Brain Iron Regional Quantification



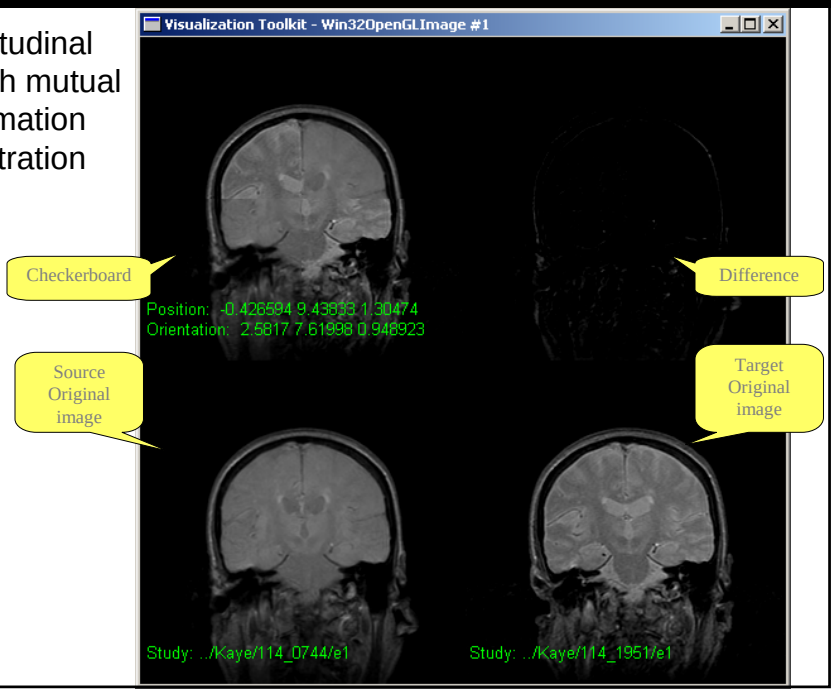
- Overall brain iron concentration can be used to assess Alzheimer's and monitor disease progression
  - Global assessment
- Regional quantification and analysis will lead to:
  - Increased understanding of disease locality
  - Monitoring patient specific disease progression
- Requires longitudinal analysis
- Requires registration of patient data
- Change detection, change monitoring



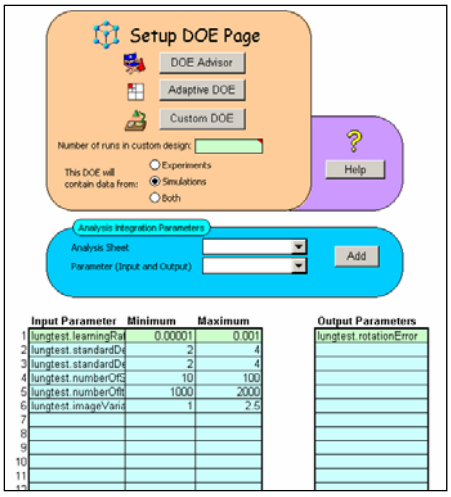
Longitudinal MRI with no registration

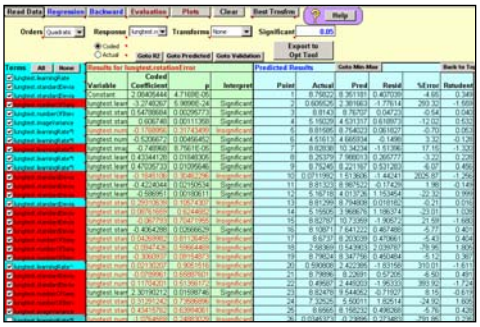


Longitudinal MRI with mutual information registration



Design of Experiments



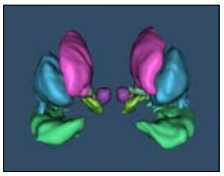


Results for lungtest.rotationError

Variable	Coded Coefficient	p	Interpret
Constant	2.08405444	4.7169E-05	
lungtest.learningRate	-3.2748267	5.9898E-24	Significant
lungtest.standardDeviationA	0.54788684	0.00295773	Significant
lungtest.standardDeviationB	0.606748	0.00111358	Significant
lungtest.numberOfSamples	-0.1768956	0.31743499	Insignificant
lungtest.numberOfIterations	-0.5206672	0.00456452	Significant
lungtest.imageVariance	-0.748968	8.7561E-05	Significant

Brain Iron Anatomical Quantification

- 40 normals, 40 Alzheimer's patients
- 8 structures per case
  - Hippocampus, Caudate, Red Nucleus, Globus Pollidus, Putamen, Substantia Nigra, Nucleus Basalis, Subthalamic Nucleus
- Structures with subtle boundaries
- Hand segmented
  - Some on T1, some on T2
  - 1 case per day

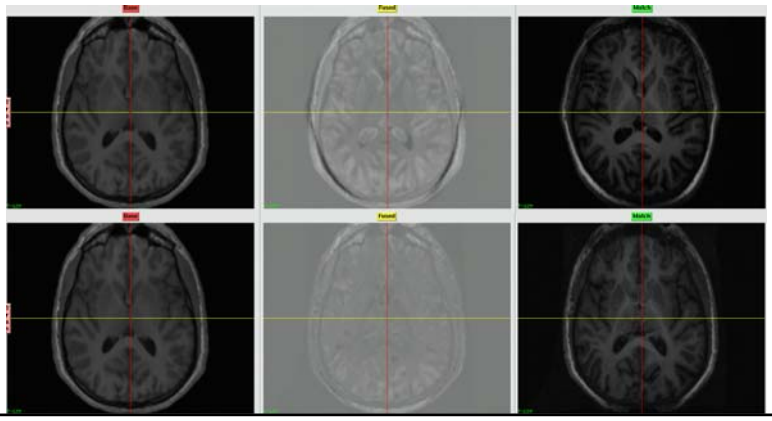


- Analyze brain iron concentrations on structure by structure basis
- Hand segmentations provide a rich set of segmentation priors

GE Research ITK in an Industrial Research Lab

Automatic Brain Structure Segmentation

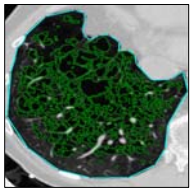
- Just starting this effort
- Demon's-based atlas segmentation
- Statistical shape models

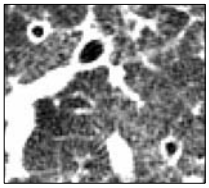


Chronic Obstructive Pulmonary Disease

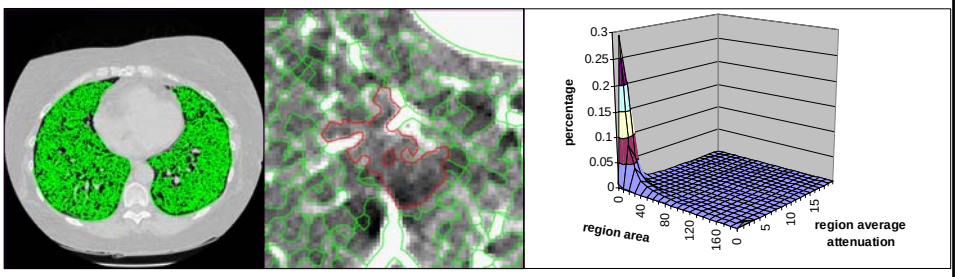
COPD is the 4th leading cause of death in the US.

- Emphysema
  - Parenchyma disease
  - Loss of lung tissue
    - Reduced capability to exchange gases
  - Loss of elastic recoil
    - Reduce capability to exhale
- Chronic Bronchitis
  - Airway disease
  - Thickening of airway wall
    - Restriction of airflow
- Quantitative analysis from CT images





Emphysema

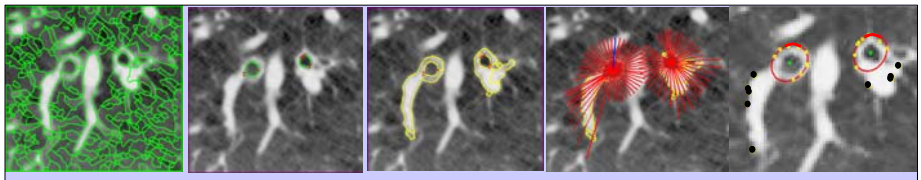


- Partition lung field into homogeneous regions
- Calculate features for each region
  - Intensity mean, intensity variance, region area, number of adjacent regions
- Compare multi-dimensional histograms to canonical cases
- Score severity on a 1-5 scale
- Legacy computer vision system

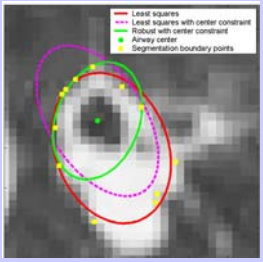
Class. rate (%) - Müller		
Chi-Square	63.16	92.11
Jeffrey	57.89	94.74
Match	50.00	92.11
EMD	52.63	94.74

5 class and 3 class performance

Chronic Bronchitis

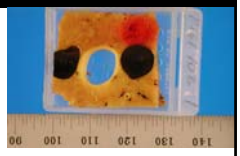


Strong model, robust fits



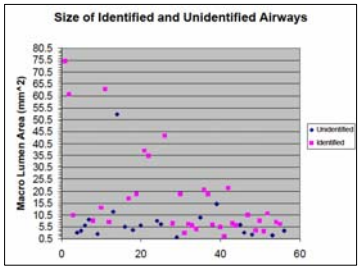
Higher order shape models

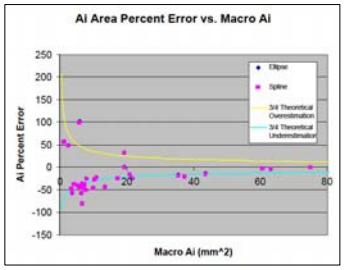
Airway wall study



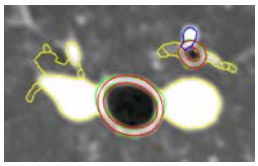
- Compare automated measurements derived from CT to manual pathology measurements from resected lung
- Pathology measurements are based on high resolution digital camera images
  - 6 times the resolution of CT
  - 2X read

Gage R&R
- Systematic bias on radii of ¾ pixel
- Similar to other techniques
- Due to scale and curvature interactions

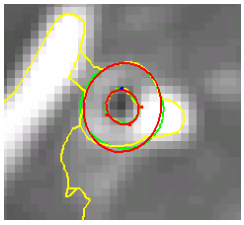




120-3
Lumen Error: 38.3%
Wall Error: 54.1%

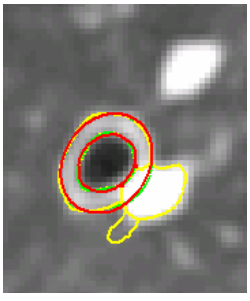


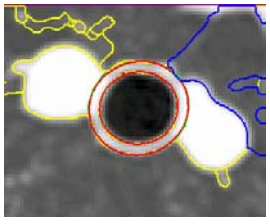
120-2
Lumen Error: 4.2%
Wall Error: 38.3%



177-3
Lumen Error: 57.7%
Wall Error: 410.9%

12-2
Lumen Error: 24.6%
Wall Error: 25.7%

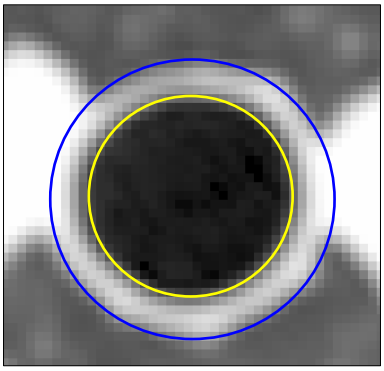




103-1
Lumen Error: 2.9%
Wall Error: 6.2%

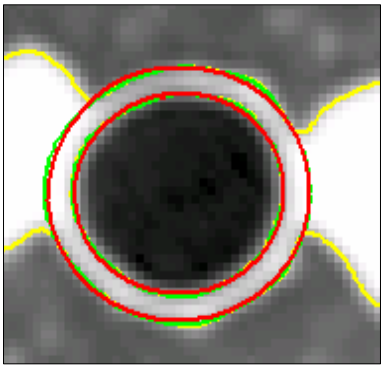
Pig Airway 102-1

(9 mm Lumen Diameter)



Pathology Measurements

Lumen Area: 75.02 mm^2
Wall Area: 101.15 mm^2

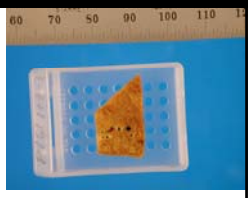


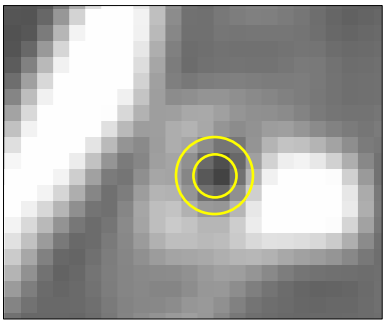
Automatic CT Measurements

Lumen Area: 75.27 mm^2 (0.32 % Error)
Wall Area: 107.38 mm^2 (6.2 % Error)

Pig Airway 177-3

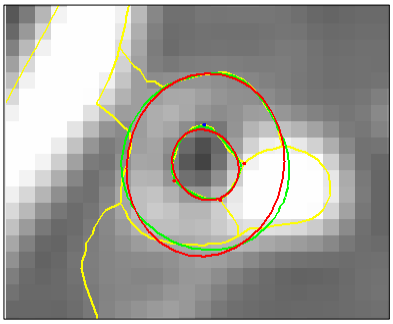
(~1 mm Lumen Diameter)





Pathology Measurements

Lumen Area: 1.36 mm^2
Wall Area: 2.54 mm^2



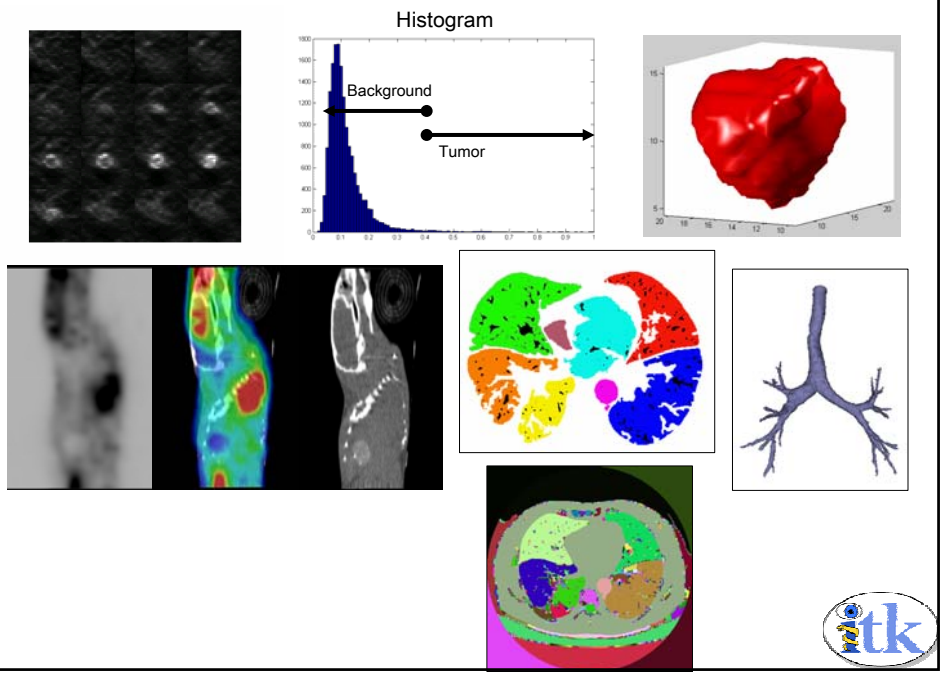
Automatic CT Measurements

Lumen Area: 2.14 mm^2 (57.7 % Error)
Wall Area: 12.96 mm^2 (410.9 % Error)

Other ITK Applications

- PET Segmentation
  - Otsu, Expectation-Maximization, Markov Random Field
- Small Animal Research
  - CT + PET Mutual Information Registration
- Surgical Navigation
  - CT + MR Mutual Information Registration
- Computer Aided Detection
  - Watersheds, Level Sets, Connected Confidence, Bayesian Decisions
- Image Quality
  - Edge preserving smoothing, Histogram matching
- Facial Reconstruction
  - Geodesic morphology
- Virtual Soldier
  - Atlas-based segmentation
- Product Teams
  - Looking at reference implementations for registration, level sets, etc.

GE Research ITK in an Industrial Research Lab



Life at an Industrial Research Lab

Academia	Industrial Research	Industry
Publish	Evaluate	*Product*
Publish	Accelerate	*Product*
Publish	Harden	*Product*
	Extend	
How	Invent	What
Why		When

NA-MIC
National Alliance for Medical Image Computing
http://na-mic.org

Slicer Overview

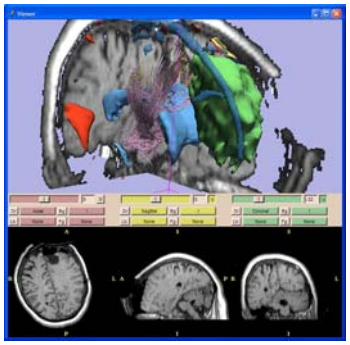
Steve Pieper, PhD

Outline

- **Slicer**
  - Overview and History
  - Architecture / Implementation
  - Image I/O and Management
  - Segmentation Tools (EM, Editor)
  - DTMRI Tools
  - fMRI Tools
  - Applications (IGT, Clinical Research)

What is 3D Slicer?

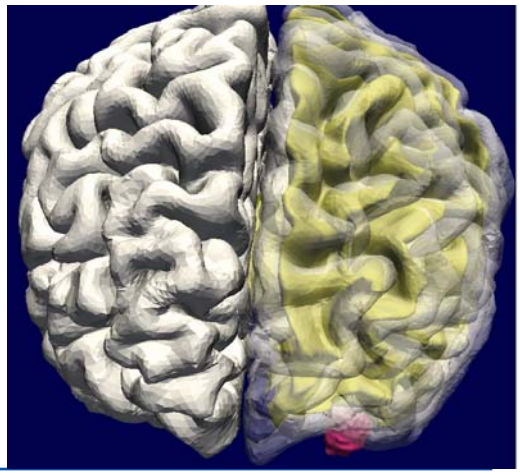
- 3D Slicer is…
  - An end-user application for 3D medical image computing research and Image Guided Therapy
  - A platform for research where new techniques can be plugged into a useful framework
  - A freely-downloadable program with source and binaries for Windows, Linux, Solaris and (sort-of) Mac OSX
  - NOT an FDA approved medical device and CANNOT be used clinically without proper research controls (IRB etc.)
  - NOT finished – some parts will work better than others



Surgical Planning Example

Dr. Jose Miguel Selman, Clinica Las Condes, Santiago, CHILE

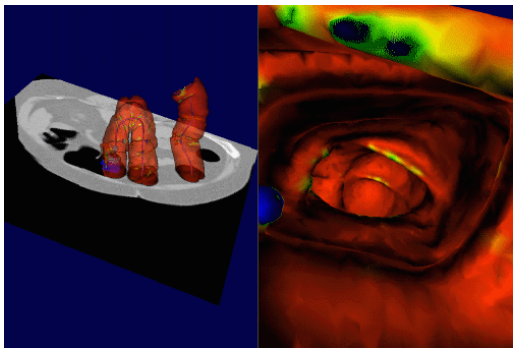
- Temporal Lobe Cavernoma
- MR Cortex and White Matter Extracted by FreeSurfer (MGH software, interface created for BIRN)
- Registration to CT and Visualization in Slicer



Virtual Endoscopy Example

Delphine Nain, MIT AI Lab, now at Georgia Tech

- Automatic or Manual Path Planning
- Animated Camera and Controls



http://www.ai.mit.edu/projects/medical-vision/virtual-endoscopy/

Slicer Background

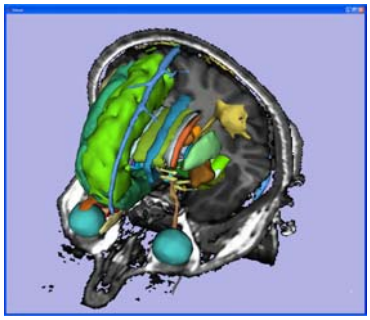
- SPL Image Guided Surgery and Visualization (Kikinis, Westin, Hata, Halle, others)
- Slicer Application Pulled Together by Dave Gering 1997-1999 with VTK and Tcl
- Further Development and Architecture by Lauren O'Donnell 1999-2001
- Ongoing Development of Slicer's Base Primarily by Steve Pieper and Nicole Aucoin
- Many Modules and Contributions by Various Authors
  - BWH, MIT, MGH, Georgia Tech, UCSD, JHU…

Why Develop with Slicer?

- **Start with a Powerful Platform**
- **Remove Obstacles to Problem Solving**
  - **Access to Every Layer of Source Code: Numeric, Graphic, Network, etc.**
- **Commit to an Environment that will Always be Available**
  - **Not Tied Up in IP of Old Institution**
  - **Not Tied to Proprietary Platform with License Fees**
- **Keep Your *Own* Work Available to *You***

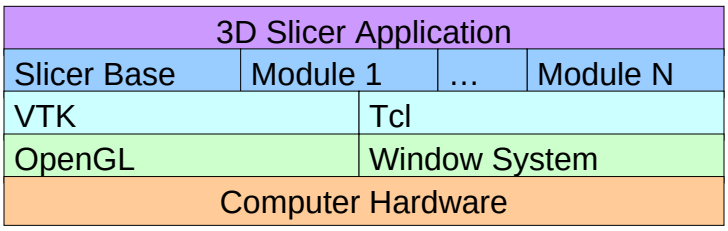
Slicer Today

- 300K Lines of Code
  - Cross-Platform Tcl/Tk GUI
  - VTK/ITK Based C++ Computing
- www.slicer.org
  - 166 on slicer-users
  - 117 on slicer-devel
  - 4000 Registered Downloads



Architecture

- Plug-in Modules consist of Tcl and C++ code with cross-platform file layout for building and loading
- Standard naming convention and hooks to add GUI and processing components
- Personally, I find the speed of C++ and the interactive development of Tcl to be a near-perfect environment



Features

- Load Medical Image Data: MR, CT in DICOM, GE, Analyze…
- XML-based File Format: MRML (Medical Reality Markup Language)
- Interactive Editor: Draw, Threshold, Math Morphology…
- Automated Segmenters: EM Segmentation, Fast Marching, Level Sets…
- Visualization: Model Building, Stereo Rendering, Animation…
- Registration: Manual, ITK, CNI
- Measurement: Fiducial-Based, Volumetric, Polyhedral Intersection, Vessel Cross-Section, Osteotomy Planning
- IGT: Tracked Probes, Real-Time Images, Robot Control
- Additional Application-Specific Features in Modules…

Administration

- **Project Housed at Surgical Planning Lab, Brigham and Women's Hospital / Harvard Medical School, Many MIT CSAIL (formerly AI Lab) Students and Faculty Involved**
- **CVS, Mailing Lists, etc at the SPL**
- **Funded Projects Supporting Slicer Base Development**
  - **Neuroimage Analysis Center (NAC): NIH Center at SPL**
  - **Biomedical Informatics Research Network (BIRN): NIH Supported Neuroimaging Collaboration**
  - **Computer Integrated Surgical Systems and Technology (CISST): NSF Supported Robotics Collaboration Headed by JHU**
  - **Virtual Soldier: DARPA Organ Simulation Collaboration**
  - **National Alliance for Medical Image Processing**
- **Module Development Supported by Application-Oriented Grants**

Image/Scene Management

- XML-Based MRML File Stores Scene Description
  - Volumes (Images, Label Maps)
  - Models
  - Hierarchical Affine Transforms
  - Scene Data (Cameras, Colors, Fiducials, etc).
- Manipulated in World Coordinates based on Patient RAS
  - Automatically Extracted from DICOM or GE Files

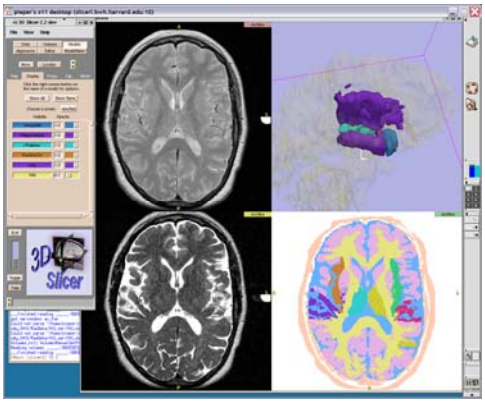
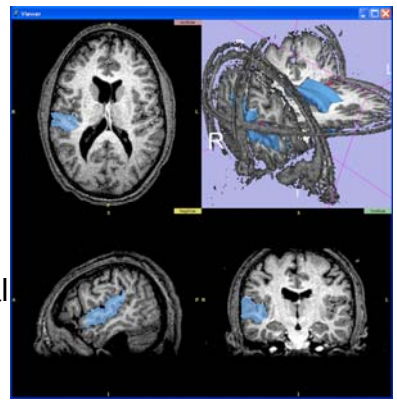


Image Formats

- DICOM, GE, Headerless, Analyze
  - Best Support for MR
  - CT and RGB Support limited
  - Real-Time from OpenMR and US
- Time Series
  - Analyze Sequence
  - DICOM Sequence
  - BXH Files
    - BIAC XML Header (Brian Image Analysis Center, Duke)
    - Like MRML for fMRI; Integration Work Ongoing

User Segmentation Tools

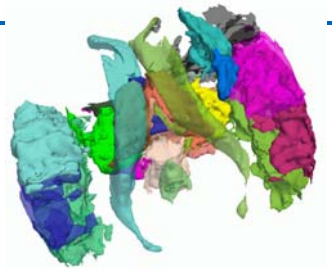
- Label Map Editor
  - Draw on Orthogonal Planes
  - Connected Component "Island" Tools
  - Math Morphology
  - Image Masking and Logical Operations
  - Level Set, Fast Marching

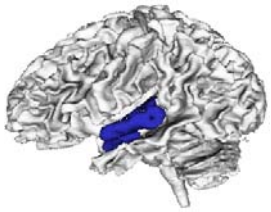


EM Segmenter

Segmentation tool designed for fully automatic, high-quality parcellation of the brain*:

- Segmentation of cortical and ventricle substructures
- Multi channel input





- Hierarchical segmentation based on anatomy
- Multi threaded
- User friendly interface

* For further information see Pohl et al. “Incorporating Non-Rigid Registration into Expectation Maximization Algorithm to Segment MR Images”, MICCAI 2002, pp. 564-572

Models

- Triangle Meshes from Label Maps
  - Marching Cubes, Decimation, Smoothing
- Model Hierarchies
- Clipping By Slice Planes

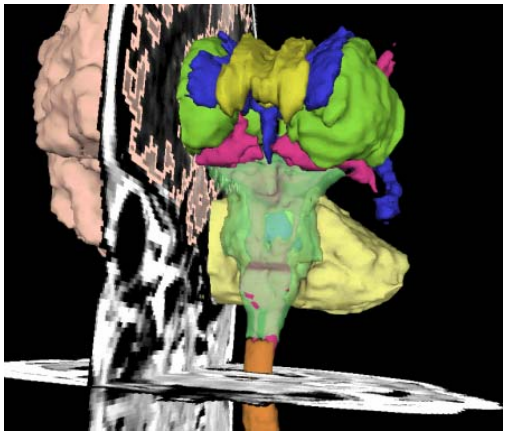
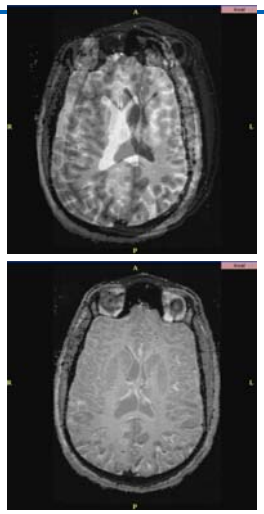


Image: Developmental Neuroinformatics, Simon Warfield

Registration

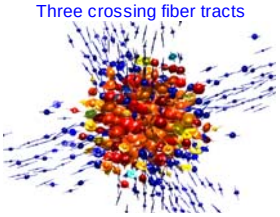
- Interactive Manual Transform Editing
- Landmark Based Alignment
- Rigid Intensity Registration
  - Mutual Information Metric
  - ITK Implementation
- Non-Rigid Registration
  - Demon's Method
  - Available by Request to CNI

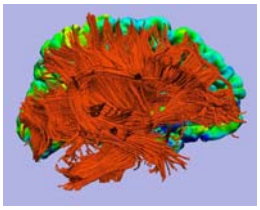


Diffusion Tensor Tractography

- Multiple MR Gradient Acquisitions
  - Sensitive to Brownian Diffusion of Water
  - Cell Membranes Restrict Diffusion
- Post Processing to Extract Probable White Matter Tracts
  - Actual Tracts are Far Below the Resolution of the Scan

Three crossing fiber tracts

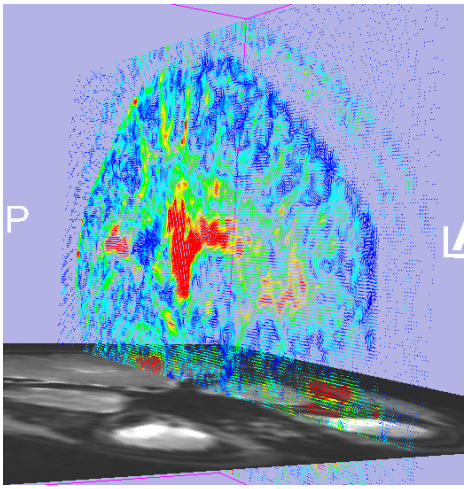




Images Provided by Westin, Park, O'Donnell et al

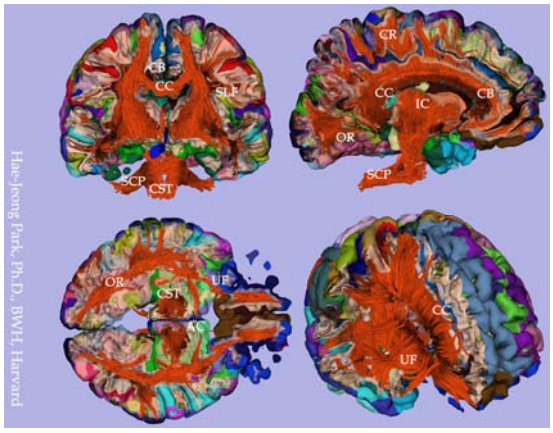
DTMRI Tools

- Convert Gradient Images to Tensors
- Generate Scalars
  - ADC, FA, etc
- Visualize Glyphs
- Tractography
  - User Guided
  - From ROI



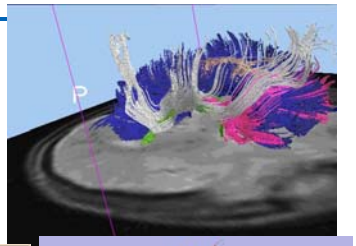
Segmentation and Tractography

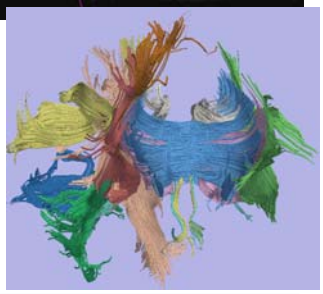
- Parcellation
  - Freesurfer (MGH)
- Tractography
  - DoDTI (H.J. Park)
- Visualization
  - Slicer
- Full Integration with Slicer Planned



DTMRI Examples

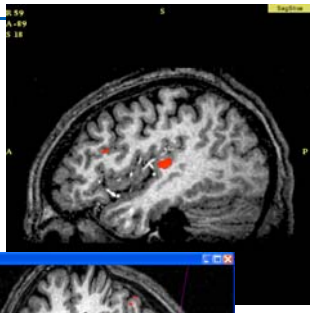
- White Matter Atlas
  - Under Development in Slicer by James Fallon, UCI

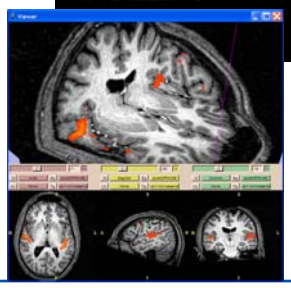




Slicer fMRI Goals

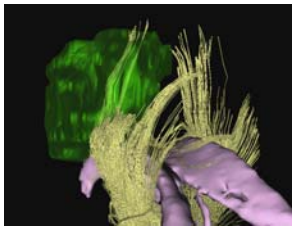
- Advanced GUI and Interactive Visualization Environment
  - iBrowser
- Platform for Activation Detector Research
  - fMRIEngine
- Framework for fMRI Integration with Other Modalities

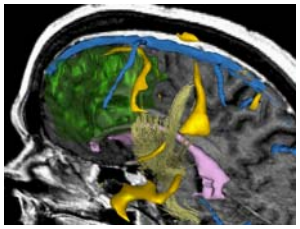




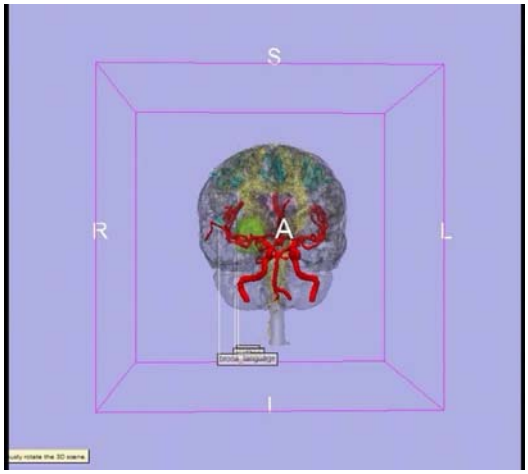
Application: Pre-Operative Map

- Structural
  - MRI Tumor Segmentation
- DTI
  - Diffusion Tensor Imaging
- fMRI
  - Functional MRI
- MEG
  - Magneto Encephlogram
- Anatomy Atlas
  - “Textbook” Information





Pre-Op Map Example



Medical Image Analysis with ITK and Related Open-Source Software

This course introduces attendees to select open-source efforts in the field of medical image analysis. Opportunities for users and developers are presented.

The course particularly focuses on the open-source Insight Toolkit (ITK) for medical image segmentation and registration. The course describes the procedure for downloading and installing the toolkit and covers the use of its data representation and filtering classes. Attendees are shown how ITK can be used in their research, rapid prototyping, and application development.

LEARNING OUTCOMES

After completing this course, attendees will be able to:

- contribute to and benefit from open-source software for medical image analysis
- download and install the ITK toolkit
- start their own software project based on ITK
- design and construct an image processing pipeline
- combine ITK filters for medical image segmentation
- combine ITK components for medical image registration

INTENDED AUDIENCE

This course is intended for anyone involved in medical image analysis. In particular it targets graduate students, researchers and professionals in the areas of computer science and medicine. Attendees should have an intermediate level on object oriented programming with C++ and must be familiar with the basics of medical image processing and analysis.

COURSE LEVEL

Intermediate

COURSE LENGTH

Full-day (6.5 hours)

COURSE SCHEDULE

Luis Ibanez, Kitware Inc.
1) The Insight Software Consortium: contributing and using open-source
2) The architecture and installation of the Insight Toolkit

Josh Cates, Univ. of Utah
1) Segmentation methods of the Insight Toolkit

Lydia Ng, Allen Brain Institute
1) Registration methods of the Insight Toolkit

Julien Jomier, CADDLab, Univ of North Carolina
1) Image IO using the Insight Toolkit
2) The Image-Guided Surgery Toolkit: architecture overview

Bill Lorensen, GE Research
1) Using the Insight Toolkit with TK/TCL
2) Applications of the Insight Toolkit

INSTRUCTORS

Luis Ibanez is a Research Engineer at Kitware, Inc. He received a BSc in Physics in 1989 and a MSc in Optics in 1994 from the Universidad Industrial de Santander (Bucaramanga, Colombia). He received a PhD in 2000 from the Universite de Rennes I (Rennes, France). From 1999 to 2001 he was Research Assistant Professor in the Division of Neurosurgery at the University of North Carolina at Chapel Hill. His current research interest the application of genomics paradigms to computation.

Josh Cates is a member of the research staff of the Scientific Computing and Imaging Institute at the University of Utah's School of Computer Science. He received his BS degree in Biology in 1995 and MS in Computer Science in 1999 from the University of Tennessee. His interests include

software engineering and problems in computer vision, including image processing (differential geometry and p.d.e.-based methods), image segmentation, and computed tomography.

Lydia Ng is a member of the Informatics group at the Allen Institute for Brain Science. She received a BE in Electrical Engineering and a BSc in Computer Science in 1994 from The University of New South Wales (Sydney, Australia). She received a Ph.D. in 2000 from Macquarie University (Sydney, Australia). Her research interests include: image registration, PDE based image processing methods, motion estimation and development of software for medical image processing and analysis.

Julien Jomier is a research faculty member in the Computer-Aided Diagnosis and Display Laboratory at the University of North Carolina at Chapel Hill. He has extensive C++ programming and software design experience as well as medical image analysis and data handling expertise. He is currently continuing his development of ITK's spatialObject library and his freely available spatial object viewer library. Julien is also contributing to the development of the Insight Journal, the Image Guided Surgery Toolkit, and numerous other open-source efforts.

William Lorensen is a Graphics Engineer in the Electronic Systems Laboratory at GE's Corporate Research and Development Center in Schenectady, NY. He has over 35 years of experience in computer graphics and software engineering. He is currently working on algorithms for 3D medical graphics and scientific visualization.

Stephen Aylward is the director of the Computer-Aided Diagnosis and Display Laboratory and an Associate Professor of Radiology, Computer Science, and Surgery at the University of North Carolina at Chapel Hill. The CADDLab's research spans medical image analysis research for disease detection, diagnosis, and treatment guidance. Dr. Aylward's research focuses on segmenting and registering medical images using model-to-image registration strategies.