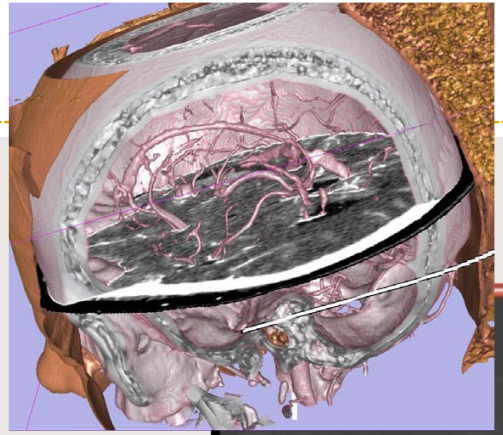
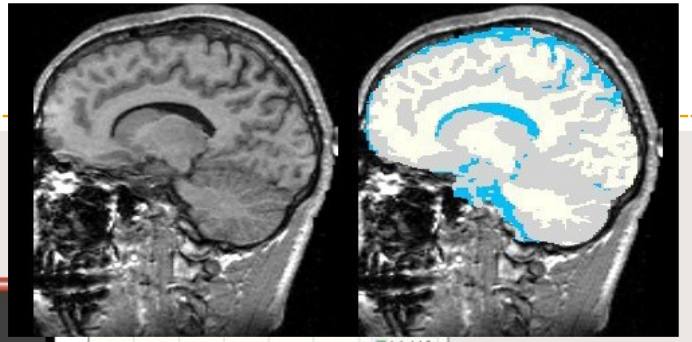
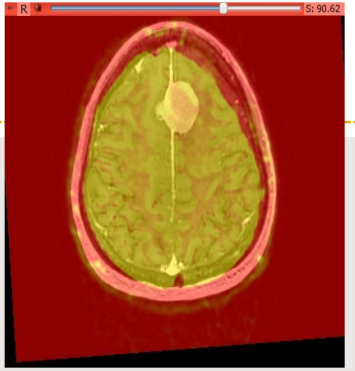
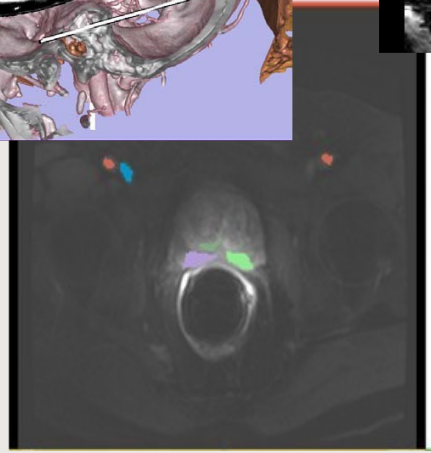
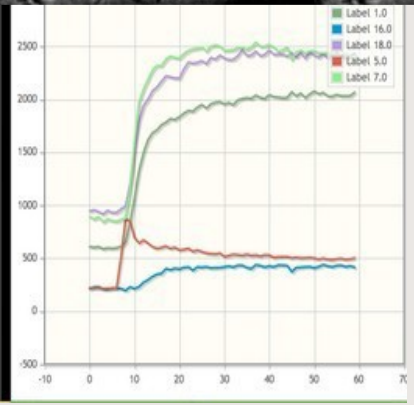
The Insight Segmentation and Registration Toolkit

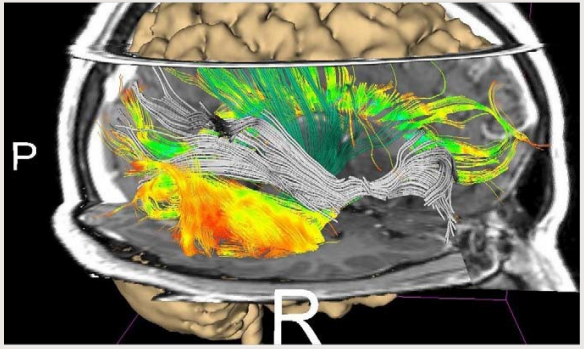












Acknowledgments

* HJ = Prepared by Dr. Hans Johnson
* JG = Prepared by Dr. John Galeotti
* LI = Prepared by Luis Ibanez
* CS= Prepared by Chuck Stewart
* WS= Prepared by William Schroeder
* DS=Prepared by Damion Shelton
* RK-Prepared by Ron Kikinis

Course Description

The Insight Segmentation and Registration Toolkit (www.itk.org) has become a standard in academia and industry for medical image analysis. In recent years, the ITK developers' community has focused on providing programming interfaces to ITK from Python, Java, and Javascript and making ITK available via leading applications such as Slicer and ImageJ. In this course we present best practices for taking advantage of ITK in your imaging research and commercial products. Via interactive examples that attendees can run on their laptops during the course, we show how ITK can speed research and commercial product development. We demonstrate how script writing and interactive GUIs can be used to access the algorithms in ITK and the multitude of ITK extensions that are freely available on the web. We also cover the opportunities and challenges with using open source software in research and in commercial applications: from prototypes that can lead to venture capital funding to applications for first in human trials and ultimately for regulatory approval.

ITK by the Numbers (WHAT?)

- Public Investment ------ >$18 Million
- March 2000 ------- First code check-in
- 33 ------- # of supported platforms each night
- 2455 ------- tests run nightly
- 1,880 ------- # of C++ classes
- 13,184 ------- # of files with code
- 2.5 Million lines ------- # of LOC (1.2 Million ITK proper)
- 84.7% --------Testing code coverage
- 209 -------- Unique developers contributing
- 326 -------- Avg. # of mailing list messages/month
- 764 -------- Pages in ITK Software Guide
- 5003 -------- Source Code Download/month

What are we doing here?

- Why does ITK exist?
  - life before ITK was sad, lonely, and often futile
  - To provide advantages over alternatives
- Where are we going
  - itk, dockers, Slicer, ImageJ, & commercial apps
- Algorithms Available
  - breadth of algorithms, and many API's to use
- Example Applications
  - breadth of uses

ITK Sponsors (WHO?)

The National Institute for Dental and Craniofacial Research

The National Science Foundation

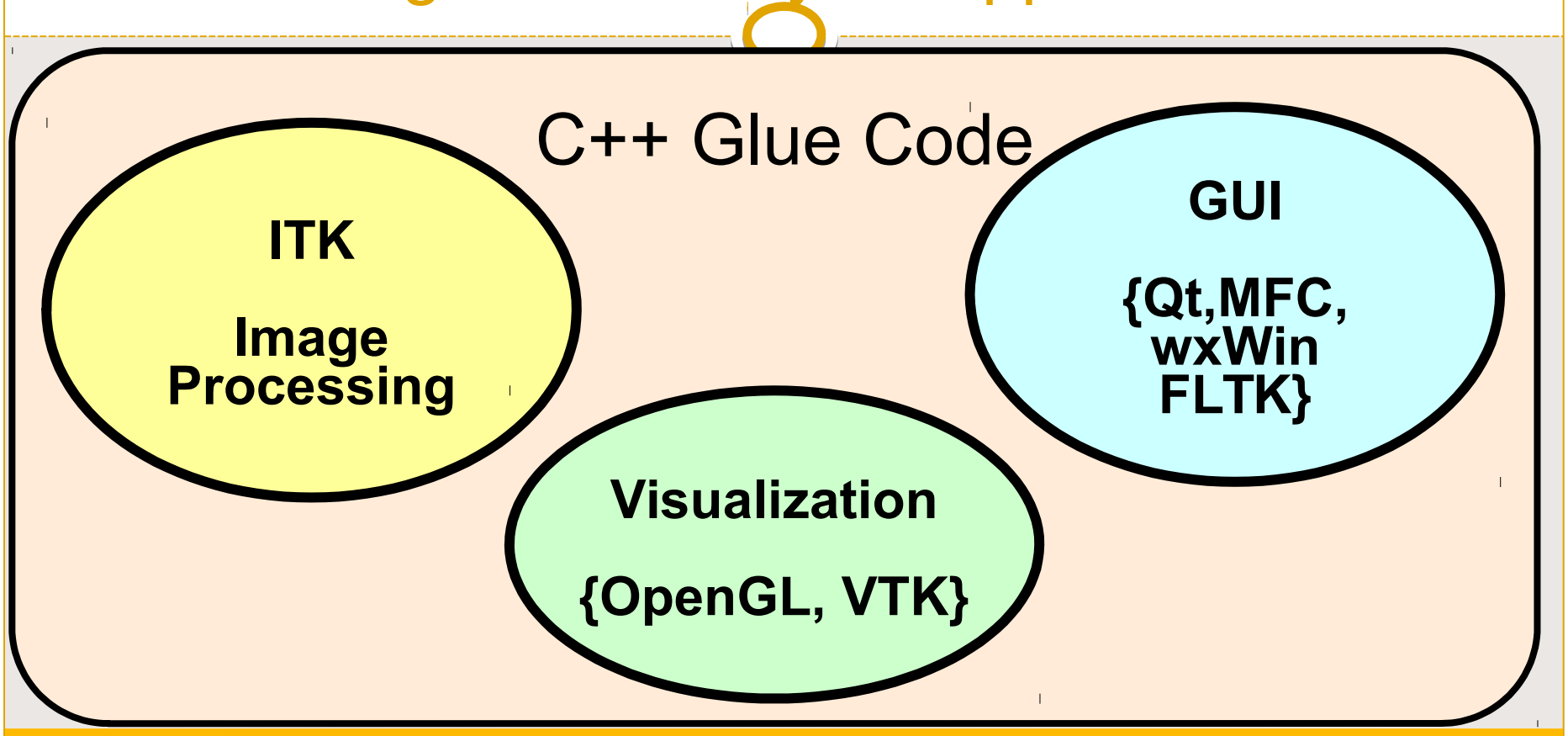
The National Institute of Neurological Disorders and Stroke

ITK Developers (HOW?/WHERE?)

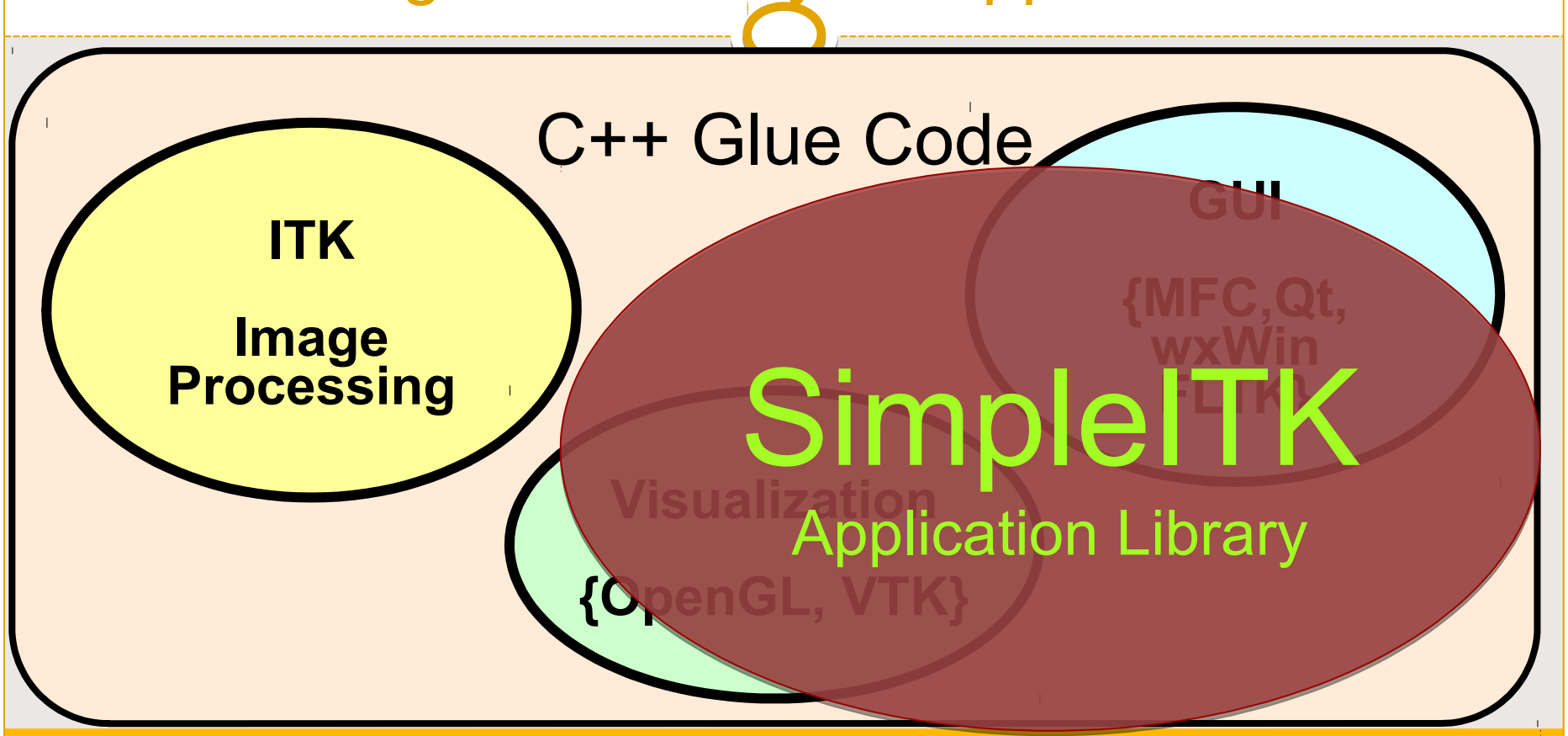
What is ITK

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

How to Integrate ITK in your application



How to Integrate ITK in your application



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

Step 0 – Don’t panic!

- There is *substantial* documentation on everything we are going to present today, and vastly more about things that we will never cover in this course
- http://www.itk.org/ITK/help/documentation.html
- Download a copy of the *ITK Software Guide*

Why all this effort?

- There are many packages that do multi-dimensional signal analysis (i.e. image processing/computer vision)

- What problem domain benefits from these efforts?
  - ITK algorithm implementations are particularly well suited for “Medical Image Analysis”

Why Is *Medical* Image Analysis Special?

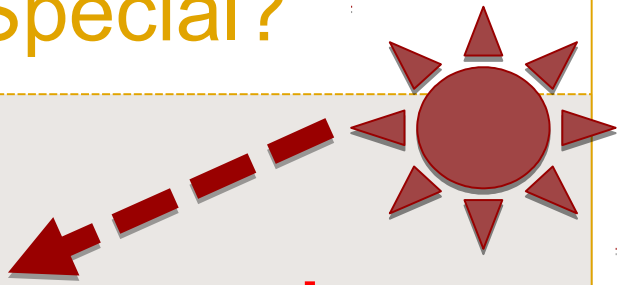
- Because of the *patient*
- Computer Vision:
  - Good at detecting irregulars, e.g. on the factory floor
  - But no two patients are alike—everyone is “irregular”

Why Is *Medical* Image Analysis Special?

- Because of the *patient*

- Medicine is war … collaboration is key to success
  - Radiology is primarily for reconnaissance
  - Surgeons are the marines (knife's edge, frontal attack)
  - Life/death decisions are made on insufficient information

Why Is *Medical* Image Analysis Special?



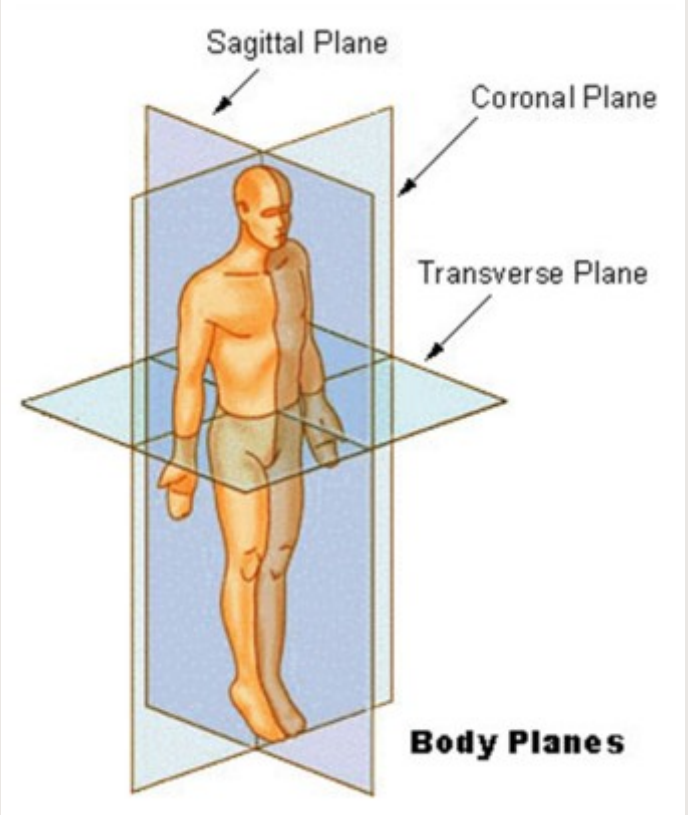
- Because of the *patient*
  - You're not in "*theory land*" anymore!
  - In addition to "the math" working, the implementation also needs to be *accessible* and produce *useful* results.
  - Success measured by patient recovery/health (often not quantifiable)

What Do I Mean by *Analysis*?

- Different from “Image Processing”
- Results in identification, measurement, &/or judgment
- Produces numbers, words, & actions
- Holy Grail: *complete image understanding* automated within a computer to perform diagnosis & control robotic intervention
- State of the art: segmentation & registration

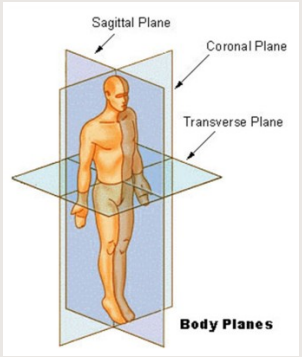
Anatomical Axes

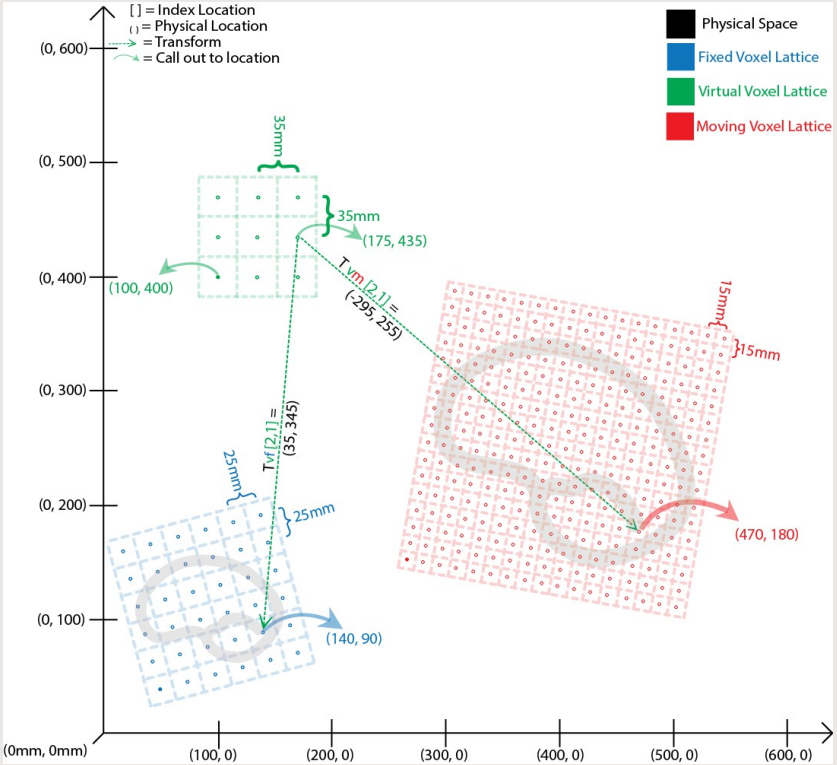
- Biological organisms are not constrained to [slices][columns][rows]
  - Superior = head
  - Inferior = feet
  - Anterior = front
  - Posterior = back
  - Proximal = central
  - Distal = peripheral



Registration

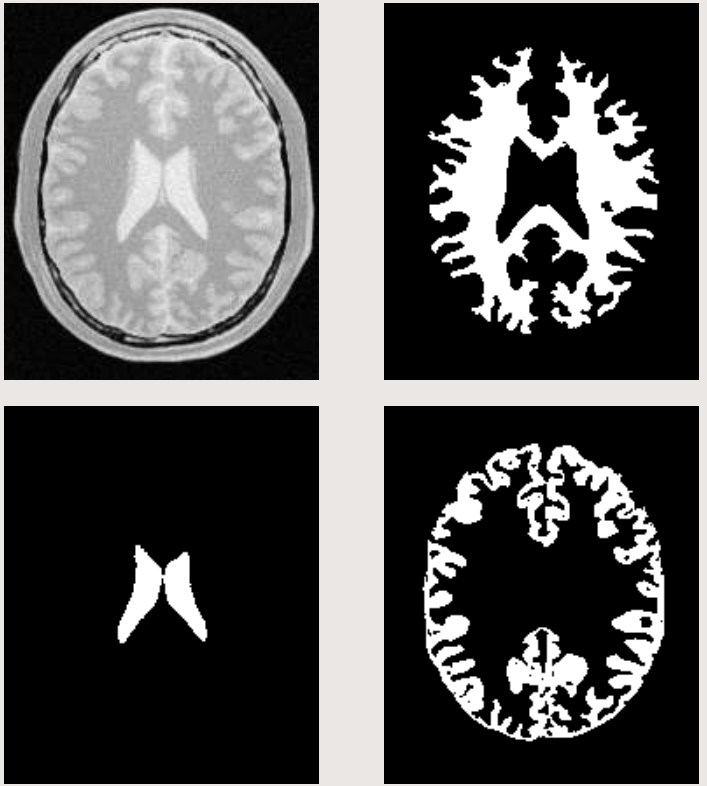
- Image to Image
  - same vs. different imaging modality
  - same vs. different patient
  - topological variation
- Image to Model
  - deformable models
- Model to Model
  - matching graphs





Segmentation

- Labeling every voxel
- Discrete vs. fuzzy
- How good are such labels?
  - Gray matter (ProcesingUnits) vs. white matter (interconnecting wires).
  - Tremendous oversimplification
- Requires a model



What is Needed Medical Image Analysis

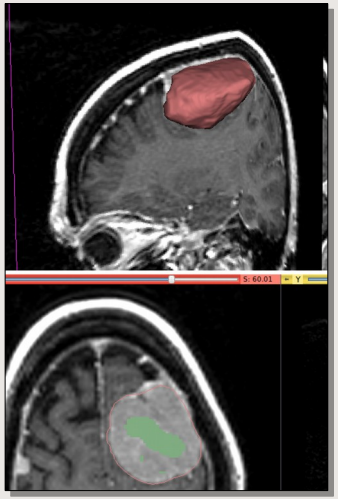
http://wiki.slicer.org/slicerWiki/index.php/Documentation/Nightly/Modules/RobustStatisticsSegmenter

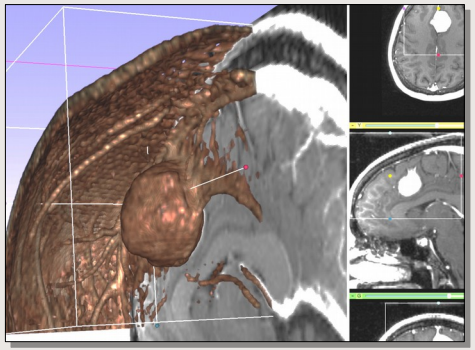
The extraction of information and knowledge from medical images using computational methods

- Technologies:
  - image segmentation
  - image registration
  - image-based physiological modeling
  - visualization

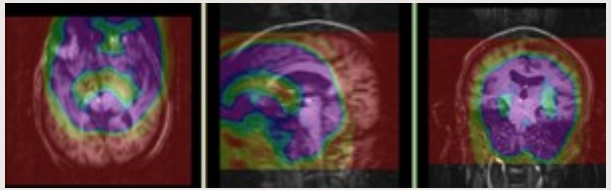
result

initialization





Volume rendering

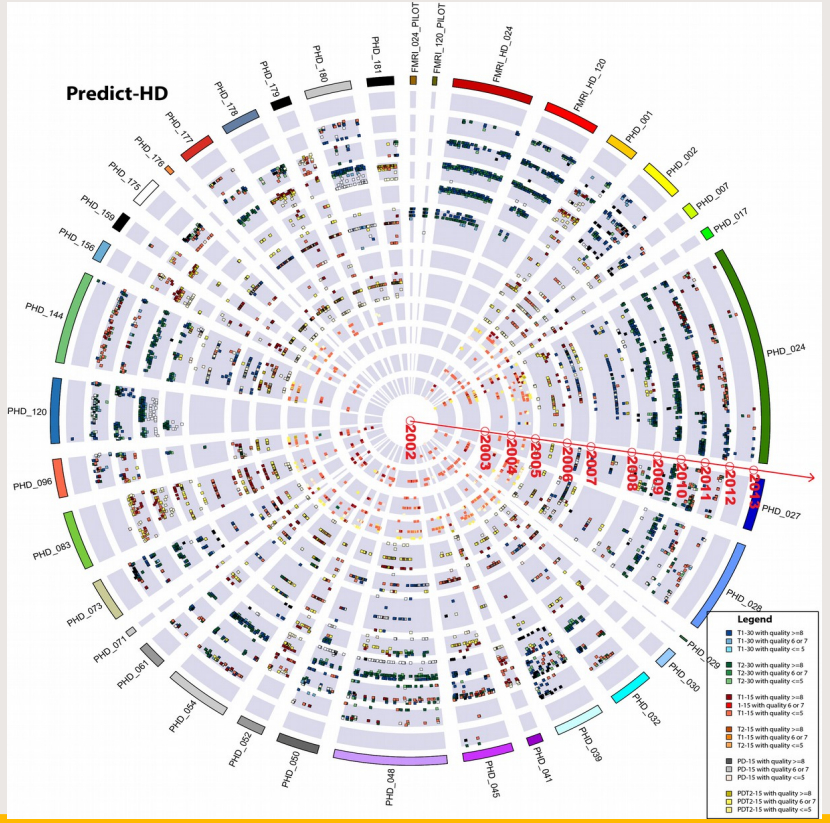


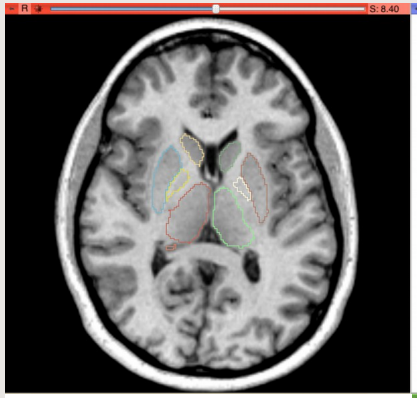
before

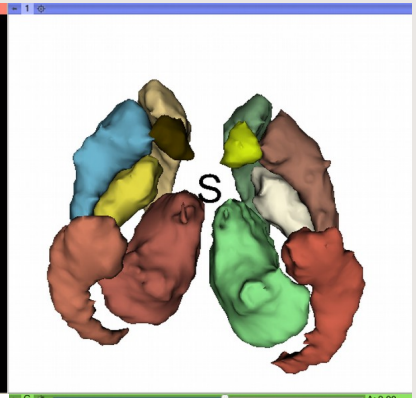
after

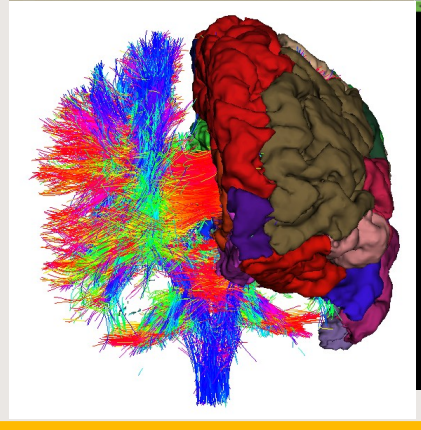
https://en.wikipedia.org/wiki/Medical_image_computing

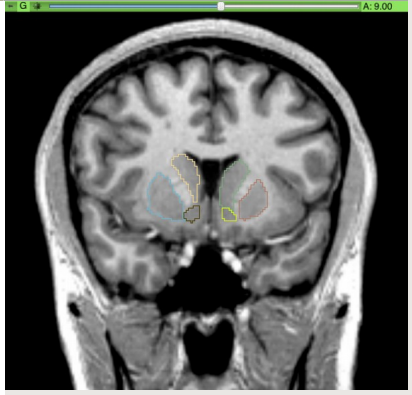
Improved Understanding



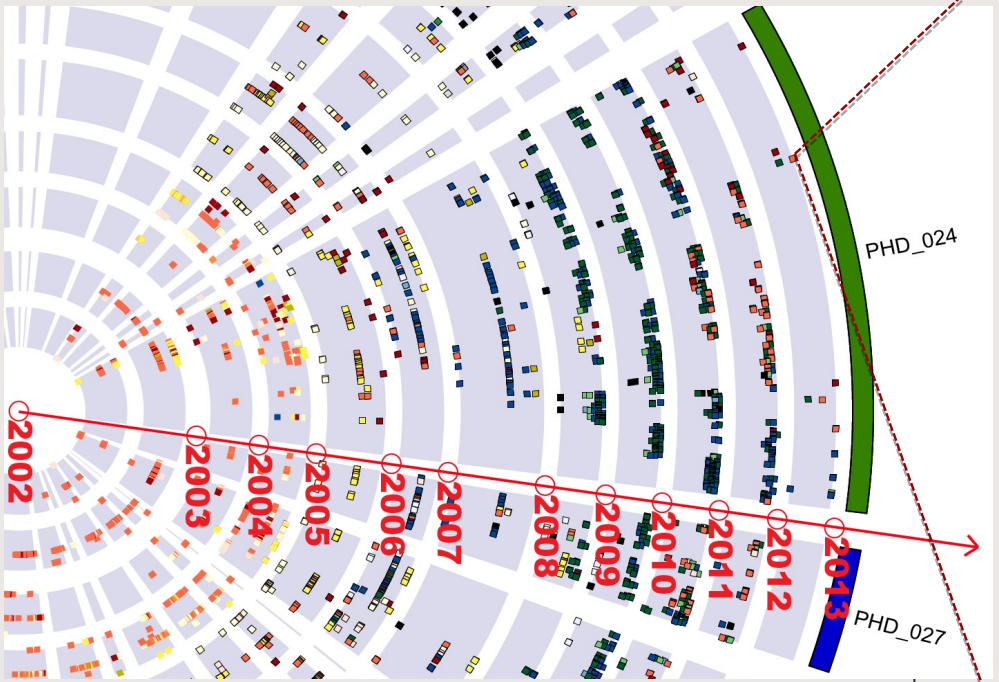


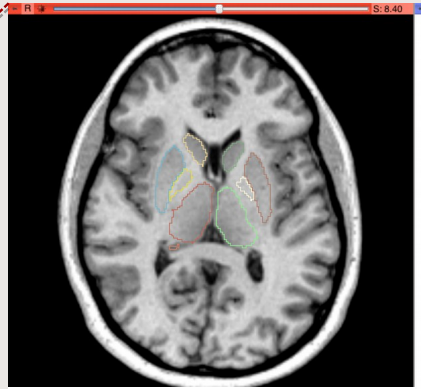


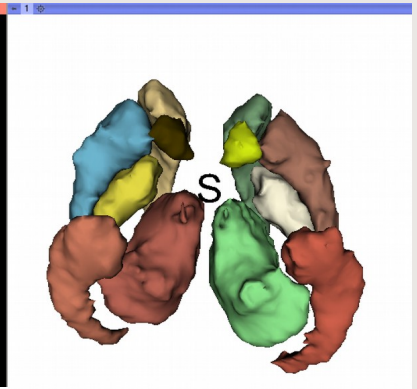


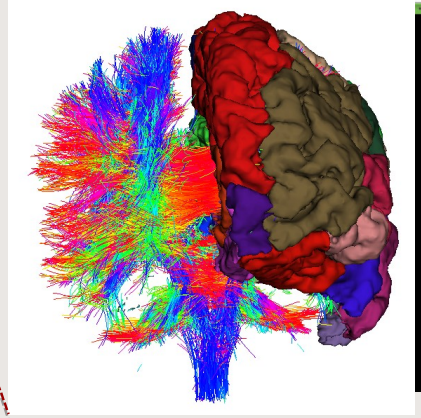


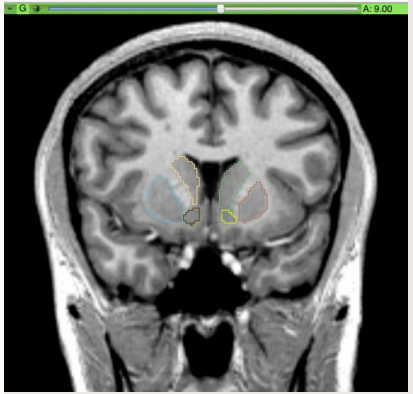
Improved Understanding











Tools & Techniques

- Tools --- Allow us to see or measure biological processes

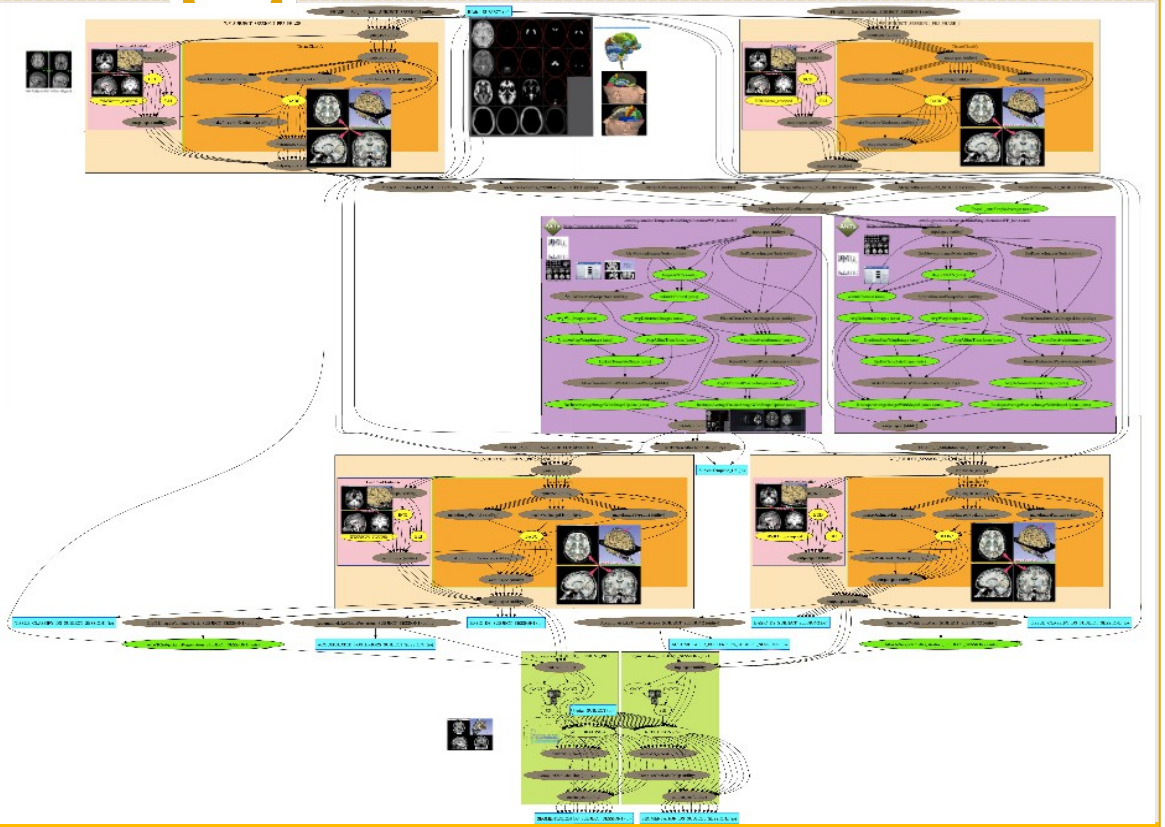
- Techniques --- Provide frameworks for managing complexity

The Practice of Automated Medical Image Analysis

- A collection of recipes, a box of tools
  - Equations that function: crafting human thought.
  - ITK is a library, not a program.
- Solutions:
  - Computer programs (fully- and semi-automated).
  - Very application-specific, no general solution.
  - Supervision / apprenticeship of machines

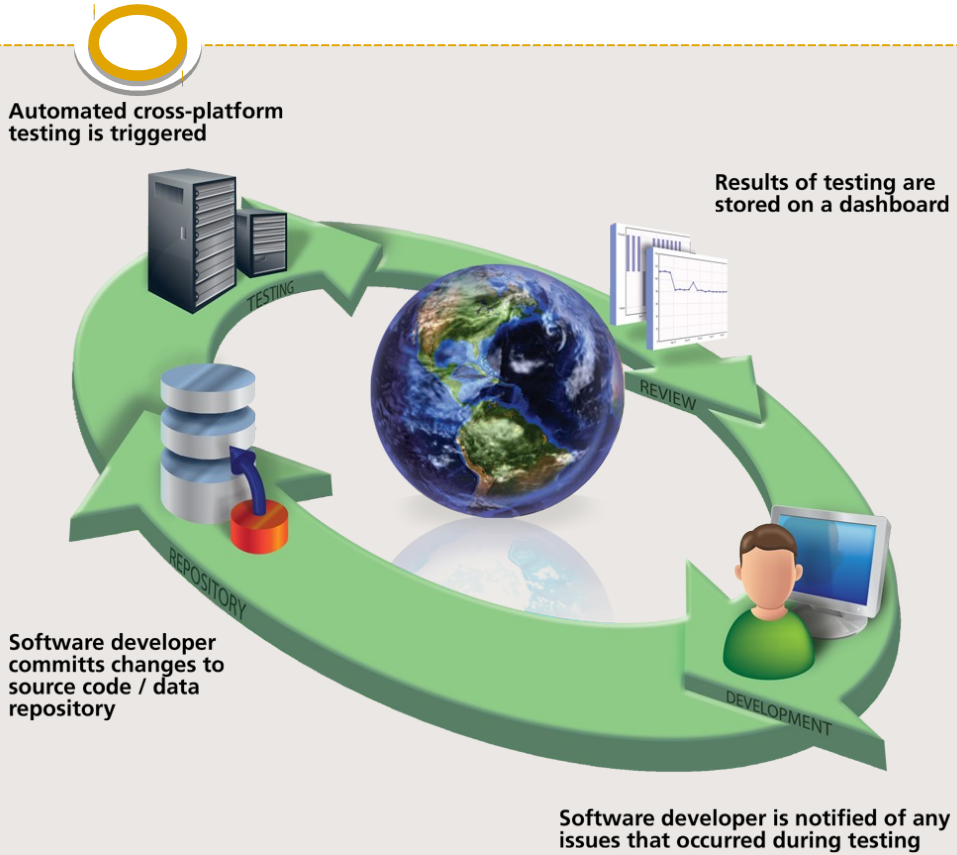
Diagram of Longitudinal Processing Pipeline

- Solutions require complex interaction of tools
  - Computational
  - Knowledge Database
  - Image Processing
    - Registration
    - Segmentation
    - Filtering
  - Image Analysis
    - Machine Learning
    - Interpretation



Software Engineering Process

- Community driven engineering
- Quality Assurance:
- Extreme programming: publish early, publish often



Graphic courtesy of JC Fillion-Robin, Kitware

The Process of Translation

Can it be done?

Innovative, not robust, usually single developer

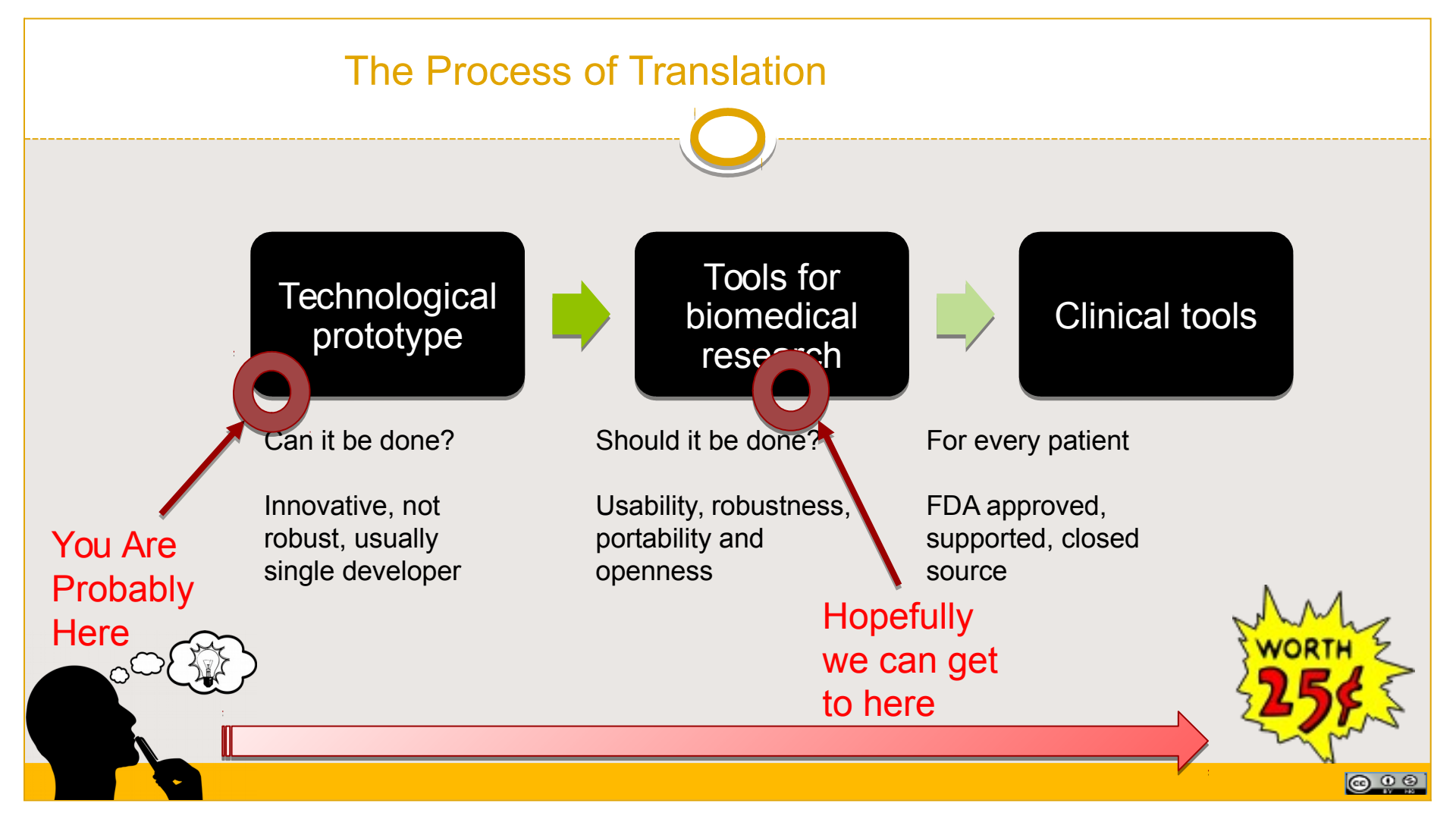
Should it be done?

Usability, robustness, portability and openness

For every patient

FDA approved, supported, closed source

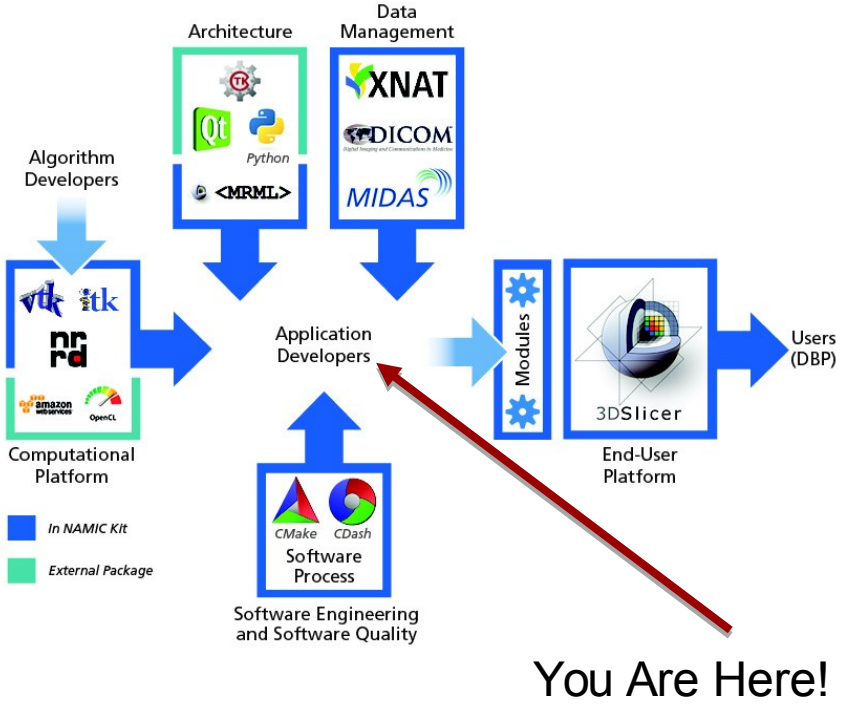
The Process of Translation



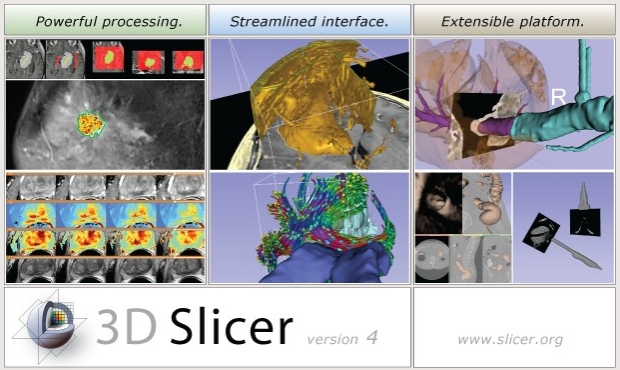
Ron's Rules For Tools

- You make it, I break It!
  - If it's not tested, it's broken
- Your tool does not exist until it works on my laptop with my data
- I am lazy/busy/overworked
  - I do not like to move the mouse or type
- Make your algorithm simple by default
  - No more than one simple parameter
- I have ADD (attention deficit disorder)
  - Make your algorithm fast

The Slicer Ecosystem



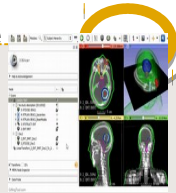
- **Major building blocks**: VTK, ITK, CTK, Qt, Python, Teem, PythonQt, SimpleITK...
- **Key infrastructure**: CMake, CTest, CDash, MediaWiki, GitHub….
- **Distribution mechanisms**: Extension manager for software distribution, data store for data distribution
- **Slicer for end-users**



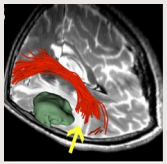
Graphic courtesy of JC Fillion-Robin, Kitware

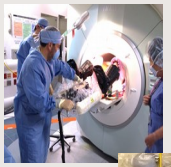
Examples of Clinical Projects

Radiation dose calculations



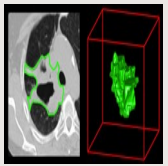
Tracking peritumoral white matter fibers



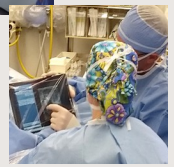


Prostate procedures

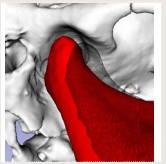
Diagnosis of Different Tumors in Lung Cancer



Clinical users drive creation of technology

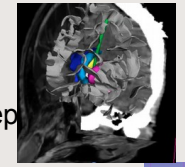


Breast cancer surgery guidance

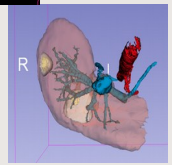


Diagnosis of Osteoarthritis Degeneration

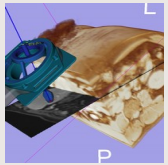
Model-Guided Deep Brain Simulation



Quantitative assessment of COPD



Surgical navigation



Liver procedures

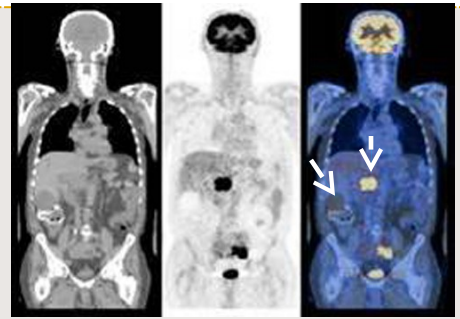
Slide courtesy of L. Cevidanes

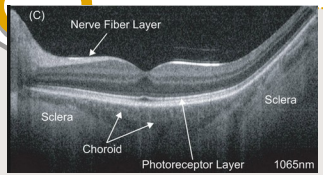
Visualization

- *Visualization* is used to mean *to picture in the mind*.
- Retina is a 2D device
- Visualization is required to reach visual cortex
- Computers have an advantage over humans in 3D (or more) dimensions
- Doctors prefer 2D pictures to volumetric data arrays

Advancement Continues

PET CT: http://nucmed.wikispaces.com/Wendt+Talk+6



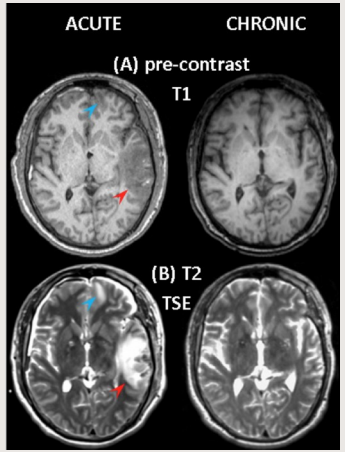


OCT: http://spie.org/x88950.xml?pf=true&ArticleID=x88950

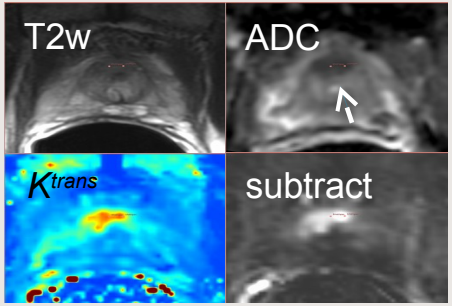
MR Images:

P. Vespa, J. Alger,

UCLA



Complex MRI: Fedorov et al. 2012. JMRI. 36(4):987–992.

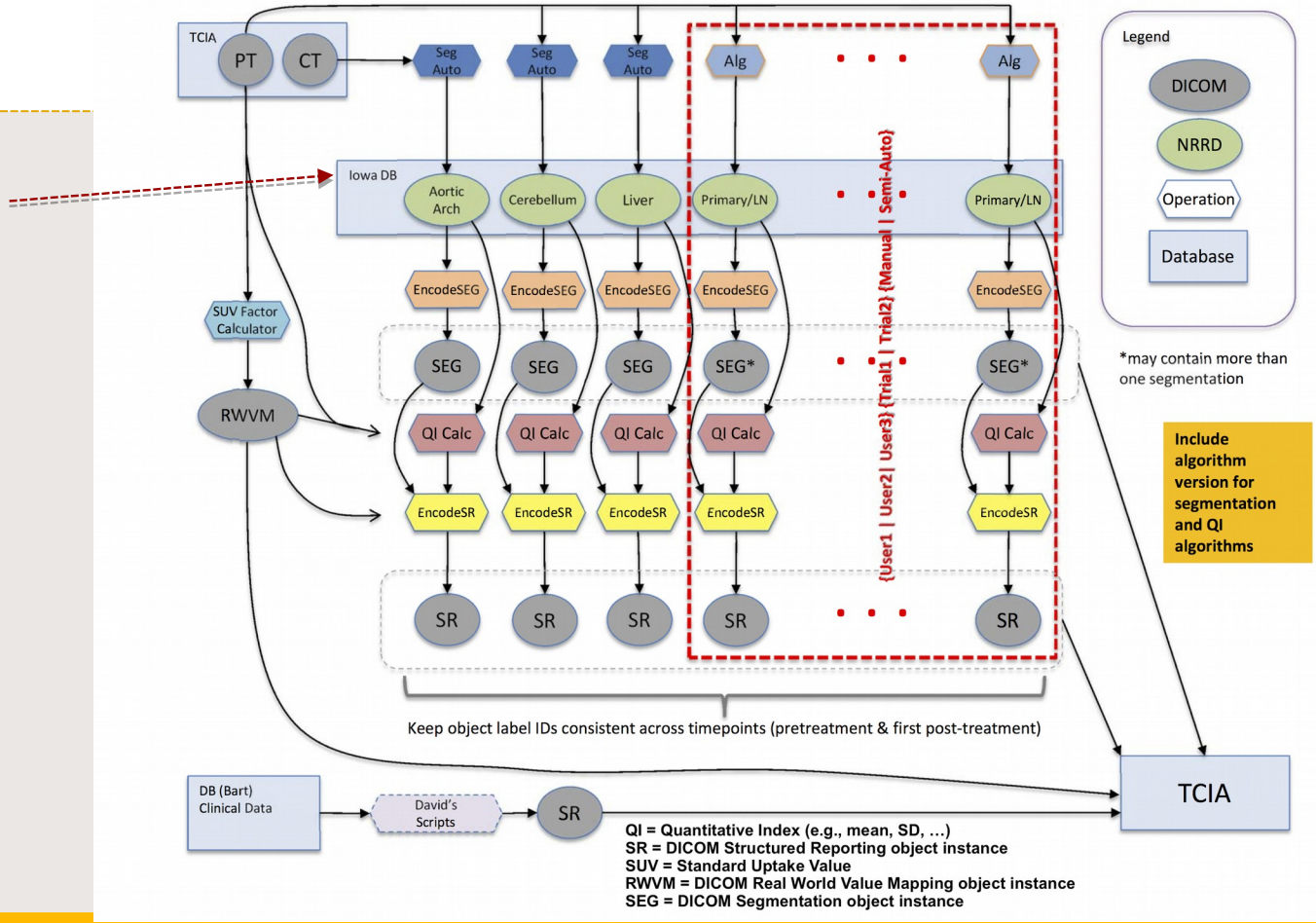


Many modalities

- X ray radiography
- Magnetic resonance imaging (MRI)
- Nuclear medicine
- Computed Tomography
- Tomography
- Ultrasound
- Optical Coherence Tomography
- Photoacoustic imaging
- Thermography
- Light Microscopy
  - Bright, dark field
  - Phase contrast
  - Fluorescence
  - Confocal

4D Ultrasound: http://ultrasoundcarespecialist.com/html/anib_3d4d.html

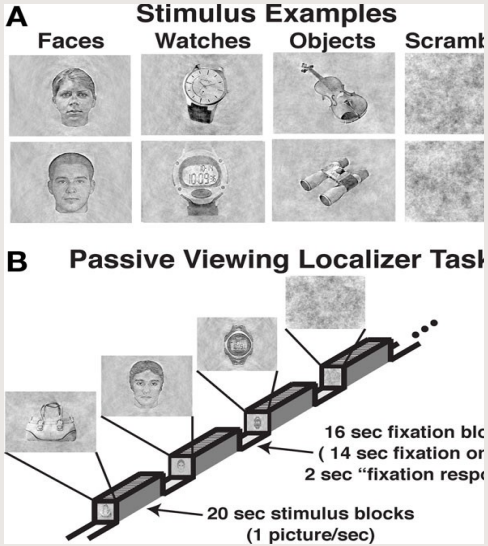
Chubby Cheeks


TCIA
PT
CT
Seg Auto
Seg Auto
Seg Auto
Alg
Alg
Legend
DICOM
NRRD
Operation
Database
Iowa DB
Aortic Arch
Cerebellum
Liver
Primary/LN
Primary/LN
{User1 | User2| User3} {Trial1 | Trial2} {Manual | Semi-Auto}
EncodeSEG
EncodeSEG
EncodeSEG
EncodeSEG
EncodeSEG
SUV Factor Calculator
SEG
SEG
SEG
SEG*
SEG*
*may contain more than one segmentation
RWVM
QI Calc
QI Calc
QI Calc
QI Calc
QI Calc
Include algorithm version for segmentation and QI algorithms
EncodeSR
EncodeSR
EncodeSR
EncodeSR
EncodeSR
SR
SR
SR
SR
SR
Keep object label IDs consistent across timepoints (pretreatment & first post-treatment)
DB (Bart) Clinical Data
David's Scripts
SR
TCIA
QI = Quantitative Index (e.g., mean, SD, …)
SR = DICOM Structured Reporting object instance
SUV = Standard Uptake Value
RWVM = DICOM Real World Value Mapping object instance
SEG = DICOM Segmentation object instance

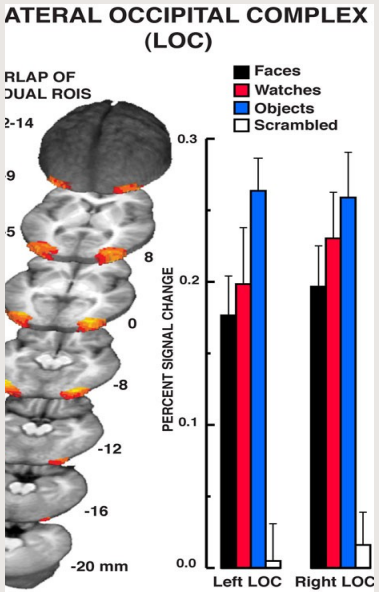
Complexity Increases

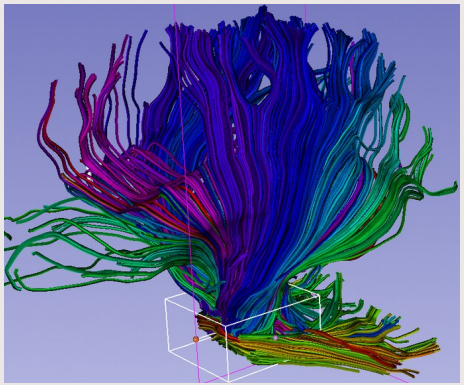
fMRI

dMRI

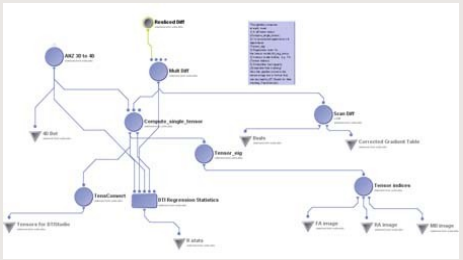


http://www.frontiersin.org/human_neuroscience/10.3389/fnhum.2010.00181/full





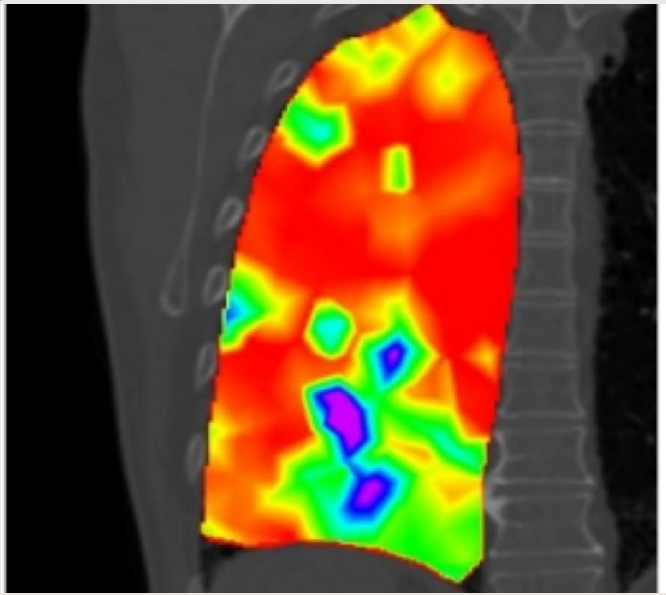
DTI streamlines:
http://wiki.slicer.org/slicerWiki/index.php/Slicer4:VisualBlog



DTI processing:
http://www.loni.ucla.edu/~ophillip/DTIPipelines.html

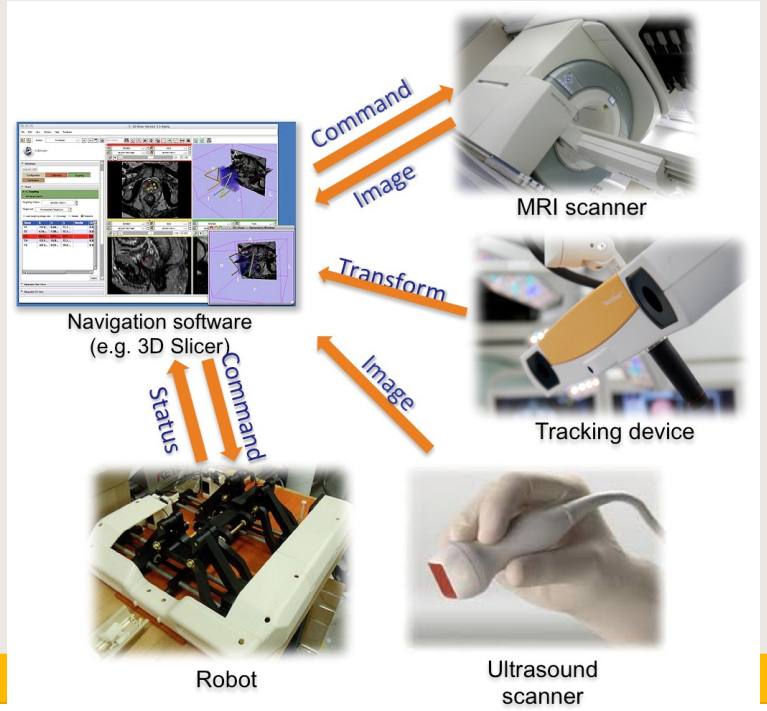
Increasing Importance of MIC

- More applications
  - Discovery, Diagnosis
  - Therapy monitoring
- More data and complexity
  - Gigabytes to terabytes
  - fMRI, molecular imaging, dMRI, 4DUS
- Translational infrastructure is needed



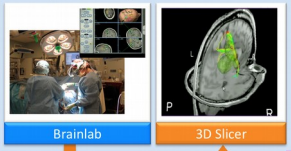
Risholm P., Ross J., Washko G.R., Wells III W.M. *Probabilistic Elastography: Estimating Lung Elasticity*. Inf Process Med Imaging. 2011;22:699-710. PMID: 21761697. PMCID: PMC3249413.

Real-time Communication In The OR



NCIGT (2009-)

- Interactive DTI tractography
- Intraoperative mass spectrometry

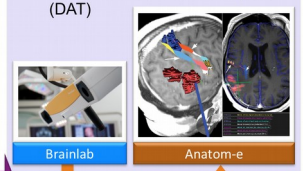


Elhawary *et al.* Neurosurgery 2011; 68(2):506-16
Calligaris *et al.* J Mass Spectrom 2013; 48:1178-87

MD Anderson (2013-)

Dr. Prabhu (UT, MD Anderson)

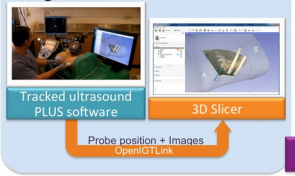
- Atlas-based neuronavigation
- Deformable Anatomic Template (DAT)



Vabulas *et al.* Neurosurgery 2014; 74(1):128-34

NCIGT+OCAIRO (2011-)

- Real-time tracked ultrasound image guidance with 3D Slicer

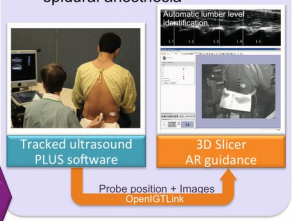


Ungi T, Tokuda J, et al. NCIGT Workshop, 2013

UBC, Canada (2012-)

Dr. Abolmaesumi (UBC)

- Augmented reality (AR) to guide epidural anesthesia



Al-Deen AH, et al. IEEE Trans Biomed Eng 2013;60(9):2636–44.

NCIGT+BRP+WPI (2008-)

- Image-guided robotic intervention for prostate

Tokuda J, et al. Comput Med Imaging Graph, 2010; 34(1):3-8.
Tokuda J, *et al.* NCIGT Workshop, 2014

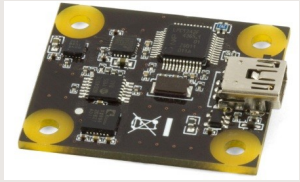
KUKA, Germany (2013-)

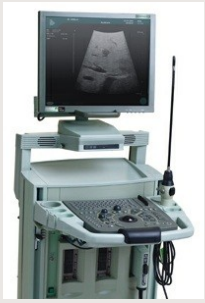
Mr. Tauscher (U of Hannover), Dr. Neff (KUKA)

- KUKA LBR robot for surgical interventions

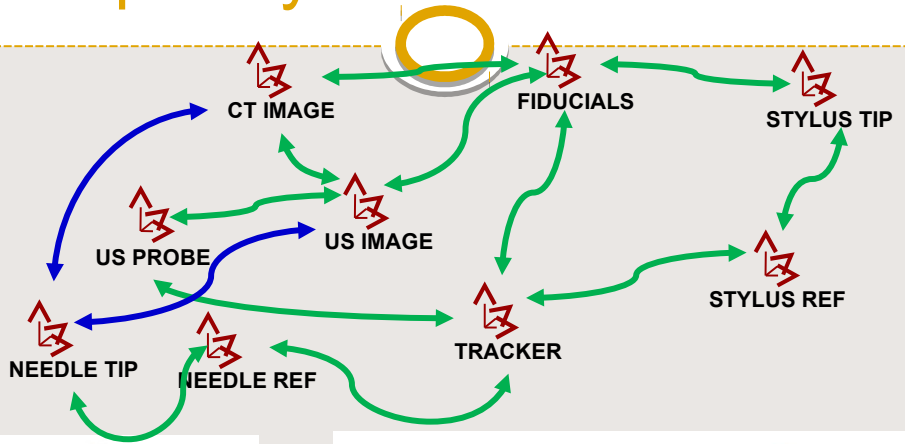
Tauscher S, *et al.* Int J Comput Assist Radiol Surg 2014

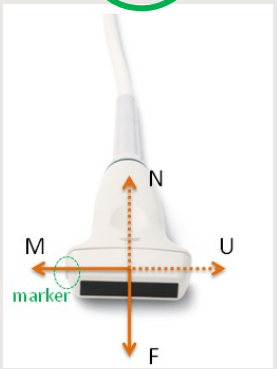
Plurality of devices

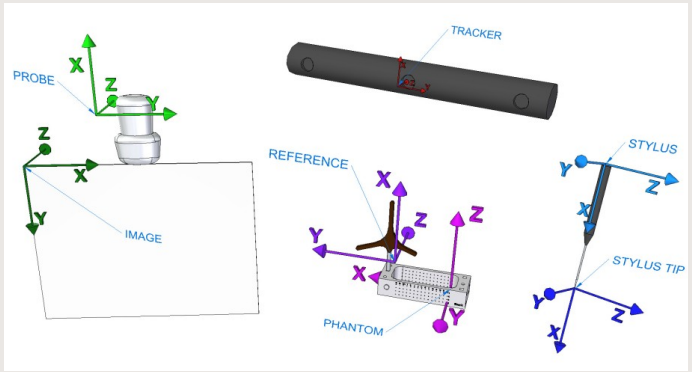


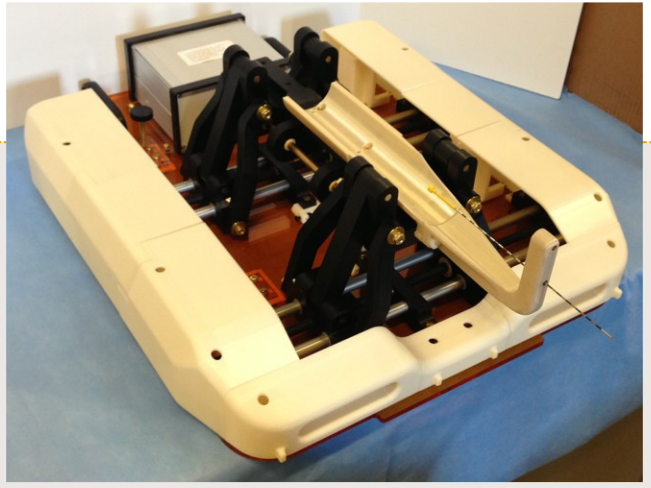


IGT: Complexity without standards

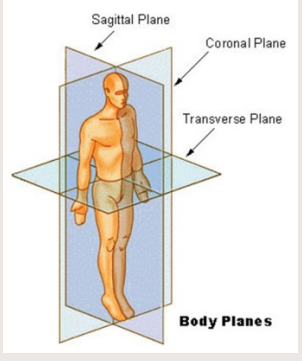








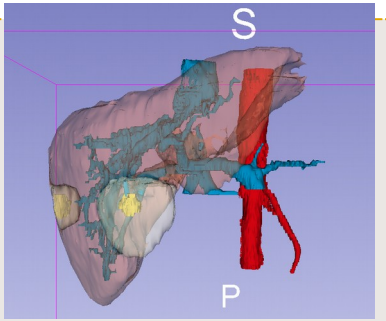
Prostate Biopsy robot with higher degree of freedom

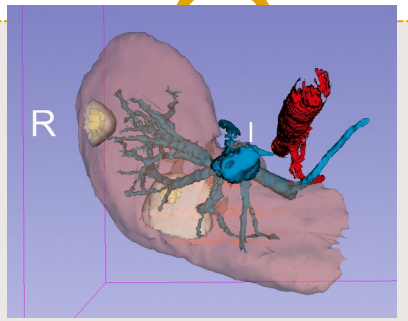


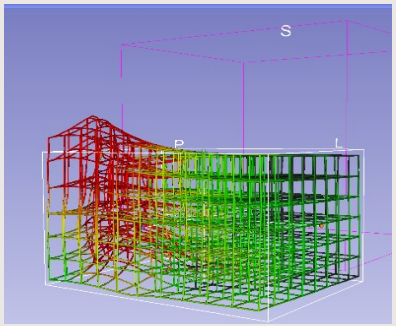
R01CA 111288 (PI Tempany/BWH) Enabling Technologies for MRI-Guided Prostate Interventions

Collaboration with Fischer (WPI), Iordachita (JHU), Burdette (Acoustic Med)

Liver surgery navigation





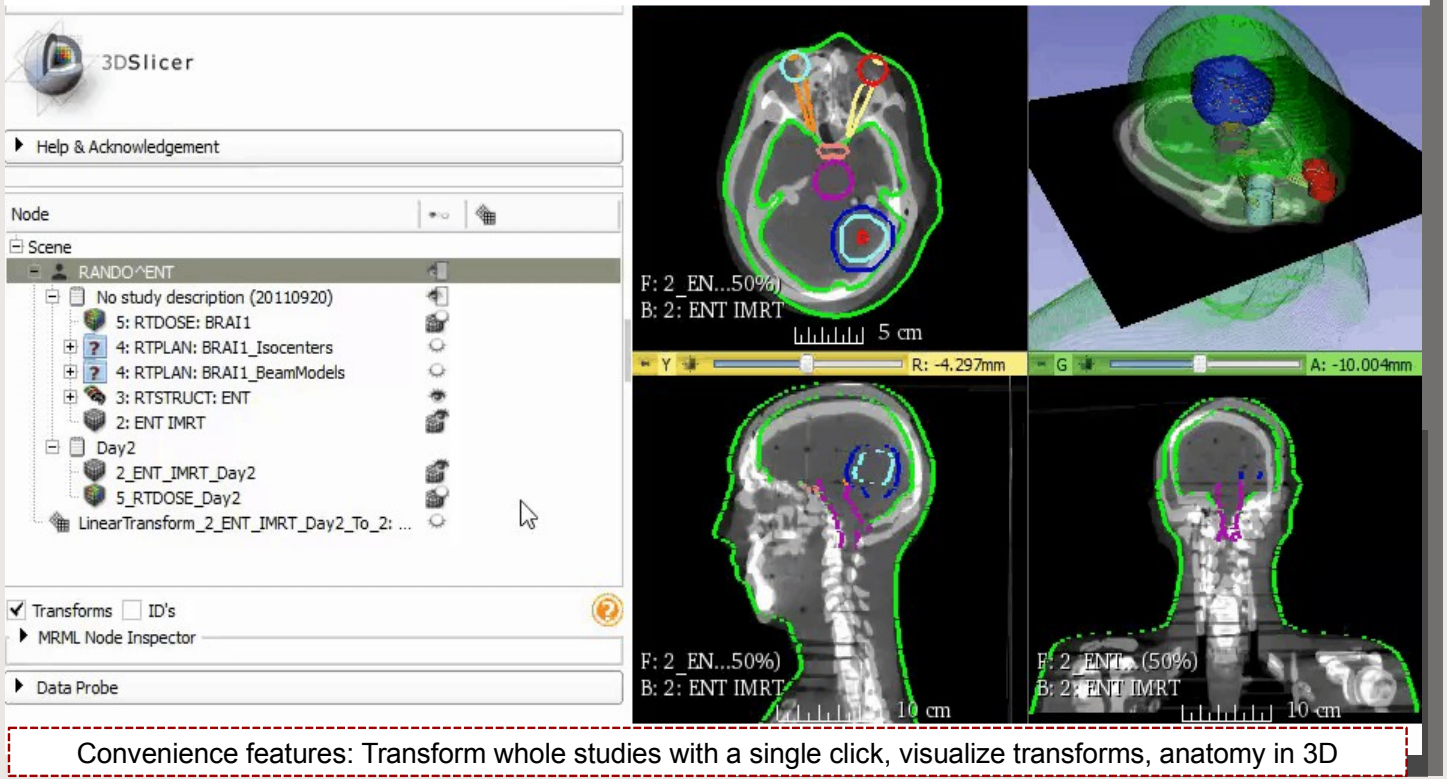


- 3D models for liver, vessels, tumor, resection margins
- Volumetric computations
- Integrated with SlicerIGT navigation system

Intra-op measurement and visualization of EM tracking error by Thompson retractor

V Harish, T Ungi, A Lasso, A Macdonald, S Nanji, G Fichtinger, Intraoperative visualization and assessment of electromagnetic tracking error, SPIE Medical Imaging 2015, Orlando, FL, USA, Feb 21-26, 2015, Proc. SPIE 9415, Medical Imaging 2015: Image-Guided Procedures, Robotic Interventions, and Modeling, 94152H

Use case: Dose accumulation



Convenience features: Transform whole studies with a single click, visualize transforms, anatomy in 3D

Companies Use Technologies of this Course










Known commercial activities range from use “as is” to full blown product development:

- Xstrahl (small animal radiation product)
- mebio (radiology product, prostate guidance)
- SonoVol (ultrasound product) (R43CA192482...)
- Novartis (quantitative imaging clinical trials)
- New Frontier (navigation system)
- KUKA (surgical robotics)
- Siemens (diagnostic and interventional research)
- Canon (robotic interventions)
- GE (research and products)
- NDI (trackers for surgical navigation)
- Isomics (research, consulting)
- Kitware (research, consulting)
  - 10+ Slicer based projects in the past two years
  - 5 commercial products being launched

Slide courtesy of Ron Kikinis

Take-home message

- Real problems are HARD
- Promote reproducible science by using open platforms
- Invest in your science not in duplication of infrastructure
- Invest in learning open platforms

This is FUN!

- No one is making you do this
- The skills and knowledge presented today will be incredibly useful
- Nothing worth having comes easy…

You have to learn the rules of the game. And then you have to play better than anyone else.

Albert Einstein

The only source of knowledge is experience.

Albert Einstein

A person who never made a mistake never tried anything new.

Albert Einstein

3_a_Docker_Jupyter_Notebook_and_Reproducible_Research

February 28, 2016

1 Docker and Jupyter Notebooks for Reproducible Research

Goal: To understand what Docker is and how it can be used with Jupyter notebooks for reproducible research.

Docker is technological tool that creates high performance, shareable, reproducible computational environments. Jupyter notebooks are tools for interactive analysis that interweave prose, code, and results. Together, Docker and Jupyter notebooks are best-of-breed methods to create research that is reproducible.

In [ ]:
```
#Imports for running this presentation live

from ipywidgets import interact, interactive
from IPython.display import clear_output, display, HTML, YouTubeVideo

import numpy as np

from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.colors import cnames
from matplotlib import animation

%matplotlib inline

!docker info
!docker load -i busybox.dockerarchive.tar
```

2 The Problem

Even though computers are often considered deterministic, **computational software is a rapidly evolving and changing landscape**. Libraries are constantly adding new features and fixing issues.

Image source: http://www.michaelogawa.com/research/storylines/

Even libraries with the strictest backwards-compatibility policies can **change in significant ways**.

Image source: http://www.bonkersworld.net/backwards-compatibility/

A **reproducible computational environment** has a sufficiently consistent state for the computational task at hand.

For example, this can consist of

- a similar CPU instruction set
- libraries and executables available with a specific version and configuration options
- a specific version of a given compiler
- a specific version of a libc implementation
- a specific version of the C++ standard library

2.1 Close But Not Good Enough

2.1.1 Source code

Does not include:

- Compiler
- Hardware it was built on
- How it is configured
- Package dependencies
- Run-time environment
- How to run it

Image source: https://www.youtube.com/watch?v=g1LgVfV5_ZQ

2.1.2 Package managers and distributions

- There is not a consensus on the package manager
- Packages become unsupported over time
- What to do if a required library is not packaged?

2.1.3 Virtual machines (VMs)

- Inefficient utilization of computational resources

Image source: http://time-az.com/images/2014/02/20140203carjam.jpg

3 Enter Linux Containers

Figure 1: Docker logo

Linux container systems , like Docker, are new type of tool to easily build, ship, and run reproducible, binary applications.

It is "good enough" for a reproducible computational environment.

In this talk, we will introduce Docker from the perspective a scientific research software engineer. We will

- Generate an understanding of what Docker is by comparing it to existing technologies.
- Give an introduction to basic Docker concepts.
- Describe how Docker fits into the scientific analysis workflow with Jupyter notebooks.

4 Understanding Docker

4.0.4 Not just this cute whale thing

Docker is an open-source engine that automates the deployment of any application as a **lightweight**, **portable**, **self-sufficient container** that will run virtually anywhere.

```
In [2]: !docker run --rm busybox sh -c 'echo "Hello Docker World!"'
```

```
Hello Docker World!
```

4.0.5 Docker is a combination of a:

1. **Sandboxed chroot**
2. **Copy on write filesystem**
3. **Distributed VCS for binaries**

4.1 Sandboxed chroot

Docker works with images that **consume minimal disk space**, **versioned**, **archiveable**, and **shareable**. Executing applications in these images does not require dedicated resources and is **high performance**.

It works with **containers** as opposed to **virtual machines** (VM's).

```
In [4]: %time !docker run --rm busybox sh -c 'echo "Hello Docker World!"'
```

```
Hello Docker World!
CPU times: user 8 ms, sys: 0 ns, total: 8 ms
Wall time: 950 ms
```

A Docker container is similar to a running an application in a chroot, but it sandboxes processes and the network stack with Linux kernel:

- **Namespaces**: isolated processes, networking, messaging, file systems, hostname's
- **CGroups**: groups together cpu, memory, and IO resources

4.2 Copy on Write Filesystem

Union file systems, or UnionFS, are file systems that operate by **creating layers**, making them very **lightweight** and **fast** while **saving disk space**.

Docker can make use of several union file system variants including:

- AUFS
- btrfs
- vfs
- DeviceMapper

4.3 Distributed VCS for binaries

4.3.1 Docker is like Git for binaries

```
In [5]: !docker search itk
```

```
NAME                                DESCRIPTION                                     STARS     O
pitkley/samba-ad-dc                 Samba4 Active Directory Domain Controller ...   1
insighttoolkit/itk-bin-testing                                                      1
insighttoolkit/itk-bin-examples                                                     1
businessdecision/itkg               Interakting Docker base images                  1
```

```
insighttoolkit/itk-bin                                                    1
itkj/cloud-sdk-appengine-go-godep          google/cloud-sdk + GAE/Go SDK + Godep          1
insighttoolkit/itk-dashboard                                              0
mdavezac/itk                               Minimal ensight toolkit                        0
insighttoolkit/itksoftwareguide-base                                      0
insighttoolkit/itksoftwareguide-dashboard                                 0
insighttoolkit/itksoftwareguide-src                                       0
jamespshields/agitk-sample-base            This sample base image includes replicatio...  0
insighttoolkit/itksoftwareguide-edit                                      0
insighttoolkit/itksoftwareguide-html                                      0
itkj/nginx                                 nginx + http2 + lua                            0
rogerkerse/itk-smartcare-build                                            0
insighttoolkit/itksoftwareguide-bin                                       0
bpierre/ubuntu-uitk-gallery                Ubuntu UITK Gallery                            0
pypi/witkets                               witkets PyPi package based on python:3         0
insighttoolkit/itk-base                                                   0
insighttoolkit/itk-data                                                   0
insighttoolkit/itk-src                                                    0
pitkley/jenkins-slave-texlive-personal     Jenkins-slave (either regular slave or Swa...  0
pitkley/python-swarm                       Python on a Jenkins Swarm slave                0
```

- Docker images are identified with hex string or tags
- Interface is `docker <subcommand>`
- `docker push`, `docker pull`, `docker tag`
- `docker export` will create a archiveable tarball of an image's filesystem.
- DockerHub is like GitHub

4.3.2 Installing

Here's what you need:

- Linux kernel with control groups and namespaces
- Support for a layered filesystem (like AUFS)
- Docker Daemon / Server (written in Go)

|Linux

- Ubuntu 14.04 or
- See Docker installation instructions for distributions with Kernel 3.8 + later or
- Kernel configuration instructions

Windows and Mac Docker Machine

- easy install of
- Git Bash
- VirtualBox
- Lightweight Linux distribution
- Docker
- Mac OSX users can use the Docker client from the Mac bash shell
- Comes with busybox shell -> Write your Docker build.sh and run.sh in Bourne shell

5 Docker Concepts

5.1 Image

5.1.1 A read-only file system layer

```
In [6]: !docker images
```

```
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
busybox             latest              65e4158d9625        10 days ago         1.114 MB
```

5.2 Container

5.2.1 An modifiable image with processes running in memory, or an exited container with a modified filesystem

```
In [7]: !docker ps
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              POR
```

```
In [8]: !docker run -d busybox sh -c 'sleep 3'
```

```
06fcb48ae8f17f18b1727c998c003c697cd2f4d292ca096624485291b617571b
```

```
In [10]: !docker ps
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              POR
```

```
In [11]: !docker ps -a
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
06fcb48ae8f1        busybox             "sh -c 'sleep 3'"   23 seconds ago      Exited (0) 19 seconds a
```

5.3 Volume

5.3.1 A directory within one or more containers that bypasses the Union File System

- Data volumes are initialized when a container is created
- Volumes can be shared and reused between containers
- Changes to a data volume are made directly
- Changes to a data volume will not be included when you update an image
- Volume persist until no containers use them
- Host directories can also be mounted as data volumes

5.3.2 Why use a data volume?

- Store and share data
- Expose data or code from the host to the Docker computational environment

5.4 Dockerfile

5.4.1 A sequence of instructions to generate a Docker image

```
In [12]: !mkdir -p docker-ls-data
         !cp $PWD/Data/*.png docker-ls-data/
```

```
In [13]: %%writefile docker-ls-data/Dockerfile

         FROM busybox
         MAINTAINER Matt McCormick <matt.mccormick@kitware.com>
         RUN mkdir -p /Data
         ADD *.png /Data/
         VOLUME /Data
         CMD ["/bin/sh", "-c", "ls /Data"]

Overwriting docker-ls-data/Dockerfile

In [14]: !docker build -t ls-data ./docker-ls-data

Sending build context to Docker daemon 1.252 MB
Step 1 : FROM busybox
 ---> 65e4158d9625
Step 2 : MAINTAINER Matt McCormick <matt.mccormick@kitware.com>
 ---> Running in 38c48dc2d406
 ---> b96fb10c253e
Removing intermediate container 38c48dc2d406
Step 3 : RUN mkdir -p /Data
 ---> Running in cdd5148e8620
 ---> 218e55546268
Removing intermediate container cdd5148e8620
Step 4 : ADD *.png /Data/
 ---> 8995b0829c1c
Removing intermediate container eab99e441323
Step 5 : VOLUME /Data
 ---> Running in a34db940bc7a
 ---> 85d12a9b4a39
Removing intermediate container a34db940bc7a
Step 6 : CMD /bin/sh -c ls /Data
 ---> Running in 071760d6bcae
 ---> b221b5c1a4fe
Removing intermediate container 071760d6bcae
Successfully built b221b5c1a4fe

In [15]: !docker run --rm ls-data

BackwardsCompatibility.png
Chroot.png
Debian.png
DockerFilesystemsBusybox.png
DockerHub.png
DockerLogo.png
Jupyter.png
Liar.png
```

5.5 Scientific Research with Docker Notebook

5.6 Graphical Applications and Docker

A **portable Docker image** will only assume standard CPU/memory/disk/network resources are available. If local USB devices and **video card devices** are used the images will **not be runnable anywhere**.

- Use IPython / Jupyter Notebooks
- The docker-opengl image offers CPU-based OpenGL rendering viewable via an HTML5 VNC client.

5.7 Choosing a base image

- debian - Most common lightweight image with many packages available
- alpine Very small image
- ipython/notebook - Launcher SSL / password enabled IPython notebook
- jupyter/tmpnb - Launches "temporary" Jupyter notebook servers
- continuumio/miniconda miniconda installed
- nixos/nix Nix package manager installed
- ...
- Make your own

6 Recap and Next Steps

6.1 Docker is

- Sandboxed chroot +
- Incremental, copy on write filesystem +
- Distributed VCS for binaries +

6.2 Concepts

- Image: A read-only file system layer
- Container: A writable image with processes running in memory, or an exited container with a modified filesystem
- Volume: A mounted directory that is not tracked as a filesystem layer
- Dockerfile: A sequence of instructions to generate a Docker image

6.3 Scientific Python and Docker

- Not for graphical applications, especially OpenGL
- Reproducible computational environment for IPython notebook
- Use with Linux-based packaging system of your choice

6.4 Learn more!

- Interactive Brower-Based Docker Tutorial
- Docker Documentation
- Reproducible Research: Walking the Walk Tutorial
- IPython DockerHub Repositories

6.5 Docker vs. LXC

- LXC is a set of tools and API to interact with Linux kernel namespaces, cgroups, etc.
- LXC used to be the default execution enviroment for Docker
- Docker provides LXC function, plus:
- Portable deployment across machines
- Application-centric
- Automatic builds

- Versioning
- Component re-use
- Sharing
- Tool echosystem

6.6 Docker vs Rocket

- Rocket is a container system like Docker developed by CoreOS
- Rocket is not as mature
- Rocket does not use a daemon/client system

3_b_SimpleITK_Prototyping_Introduction

February 28, 2016

1 ITK Prototyping with SimpleITK in Jupyter Notebooks

Goal: Introduce SimpleITK and the SimpleITK style of interface to ITK classes interactively the Notebook environment.

SimpleITK is a wrapping of the Insight Segmentation and Registration Toolkit (ITK) designed to facilitate rapid prototyping, education as well as to be easily used and improve the accessibility to ITK's algorithms.

It was designed from the ground up to have any easy to use interface which follows modern scripting language conventions.

1.1 SimpleITK Introduction

- Simplify ITK by not exposing the algorithm type dependencies on the image type (and many other hidden simplifications)
- Binary built distributions
- Procedural and object oriented interfaces
- Supports 2D and 3D image, along with multi-component images
- Overload operators for image types
- Easy importing and exporting bulk data through numpy

Additional information about the design and architecture of SimpleITK can be found in the following publication:

- Lowekamp BC, Chen DT, Ibáñez L and Blezek D (2013) The Design of SimpleITK. Front. Neuroinform. 7:45. doi: 10.3389/fninf.2013.00045

1.2 Introduction to the ITK Image in SimpleITK

Need to do some standard importing of Python modules first

```
In [1]: from __future__ import print_function

        import SimpleITK as sitk

        import matplotlib.pyplot as plt
        %matplotlib inline

        # Utility method that either downloads data from the MIDAS repository or
        # if already downloaded returns the file name for reading from disk (cached data).
        from downloaddata import fetch_data as fdata
```

1.2.1 Image Fundamentals

The ITK Image class is more that just a multi-dimensional array of values. It contains important meta-data that affects the how algorithms process the images, and is fundamental to bio-medical image analysis.

Image Geometry A feature of ITK as a toolkit for image manipulation and analysis is that it views images as physical objects occupying a bounded region in physical space. In addition images can have different spacing between pixels along each axis, and the axes are not necessarily orthogonal. The following figure illustrates these concepts.

Pixel Types ITK support more that just floating point images. An image can be one of many numerical representations, based on what efficient or the source of the image.

The pixel type is represented as an enumerated type. The following is a table of the enumerated list.

sitkUInt8
Unsigned 8 bit integer
sitkInt8
Signed 8 bit integer
sitkUInt16
Unsigned 16 bit integer
sitkInt16
Signed 16 bit integer
sitkUInt32
Unsigned 32 bit integer
sitkInt32
Signed 32 bit integer
sitkUInt64
Unsigned 64 bit integer
sitkInt64
Signed 64 bit integer
sitkFloat32
32 bit float
sitkFloat64
64 bit float
sitkComplexFloat32
complex number of 32 bit float
sitkComplexFloat64
complex number of 64 bit float
sitkVectorUInt8
Multi-component of unsigned 8 bit integer
sitkVectorInt8
Multi-component of signed 8 bit integer
sitkVectorUInt16
Multi-component of unsigned 16 bit integer
sitkVectorInt16
Multi-component of signed 16 bit integer
sitkVectorUInt32
Multi-component of unsigned 32 bit integer
sitkVectorInt32
Multi-component of signed 32 bit integer
sitkVectorUInt64
Multi-component of unsigned 64 bit integer
sitkVectorInt64
Multi-component of signed 64 bit integer
sitkVectorFloat32
Multi-component of 32 bit float
sitkVectorFloat64
Multi-component of 64 bit float
sitkLabelUInt8
RLE label of unsigned 8 bit integers

sitkLabelUInt16
RLE label of unsigned 16 bit integers
sitkLabelUInt32
RLE label of unsigned 32 bit integers
sitkLabelUInt64
RLE label of unsigned 64 bit integers
There is also `sitkUnknown`, which is used for undefined or erroneous pixel ID's. It has a value of -1.
The 64-bit integer types are not available on all distributions. When not available the value is `sitkUnknown`.

1.2.2 Load and Display

Load your first image and display it!

```
In [2]: logo = sitk.ReadImage(fdata('SimpleITKLogo.png'))

        plt.imshow(sitk.GetArrayFromImage(logo))
        plt.axis('off');
```

```
Fetching SimpleITKLogo.png
```

1.2.3 Image Contruction

There are a variety of ways to create an image.
The following components are required for a complete definition of an image:
Pixel type [fixed on creation, no default]: unsigned 32 bit integer, sitkVectorUInt8, etc., see list above.
Sizes [fixed on creation, no default]: number of pixels/voxels in each dimension. This quantity implicitly defines the image dimension.
Origin [default is zero]: coordinates of the pixel/voxel with index (0,0,0) in physical units (i.e. mm).
Spacing [default is one]: Distance between adjacent pixels/voxels in each dimension given in physical units.
Direction matrix [default is identity]: mapping, rotation, between direction of the pixel/voxel axes and physical directions.
Initial pixel/voxel values are set to zero.

```
In [3]: image_3D = sitk.Image(256, 128, 64, sitk.sitkInt16)
        image_2D = sitk.Image(64, 64, sitk.sitkFloat32)
        image_2D = sitk.Image([32,32], sitk.sitkUInt32)
        image_RGB = sitk.Image([128,64], sitk.sitkVectorUInt8, 3)
```

1.2.4 Image Attrributes

You can change the image origin, spacing and direction. Making such changes to an image already containing data should be done cautiously.

```
In [4]: image_3D.SetOrigin((78.0, 76.0, 77.0))
        image_3D.SetSpacing([0.5,0.5,3.0])

        print(image_3D.GetOrigin())
        print(image_3D.GetSize())
        print(image_3D.GetSpacing())
        print(image_3D.GetDirection())
```

```
(78.0, 76.0, 77.0)
(256, 128, 64)
(0.5, 0.5, 3.0)
(1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0)
```

Image dimension queries

```
In [5]: print(image_3D.GetDimension())
        print(image_3D.GetWidth())
        print(image_3D.GetHeight())
        print(image_3D.GetDepth())
```

```
3
256
128
64
```

Pixel/voxel type queries:

```
In [6]: print(image_3D.GetPixelIDValue())
        print(image_3D.GetPixelIDTypeAsString())
        print(image_3D.GetNumberOfComponentsPerPixel())
```

```
2
16-bit signed integer
1
```

1.2.5 Indexing and Slicing

The Image class's member functions GetPixel and SetPixel provide an ITK-like interface for pixel access.

```
In [7]: help(image_3D.GetPixel)
```

```
Help on method GetPixel in module SimpleITK.SimpleITK:

GetPixel(*idx) method of SimpleITK.SimpleITK.Image instance
    Returns the value of a pixel.

    This method takes 2 parameters in 2D: the x and y index,
    and 3 parameters in 3D: the x, y and z index.
```

```
In [8]: print(image_3D.GetPixel(0, 0, 0))
        image_3D.SetPixel(0, 0, 0, 1)
        print(image_3D.GetPixel(0, 0, 0))
```

```
        # This can also be done using pythonic notation.
        print(image_3D[0,0,1])
        image_3D[0,0,1] = 2
        print(image_3D[0,0,1])
```

```
0
1
0
2
```

Slicing of SimpleITK images returns a copy of the image data. This is similar to slicing Python lists and differs from the "view" returned by slicing numpy arrays.

The Python standard slice interface for 1-D object:

Operation
Result
d[i]
ith item of d, starting index 0
d[i:j]
slice of d from i to j
d[i:j:k]
slice of d from i to j with step k

With this convient syntax many basic tasks can be easily done.

In [9]:
```
logo_subsampled = logo[::2,::2]

        # Get the sub-image containing the word Simple
        simple = logo[0:115,:]

        # Get the sub-image containing the word Simple and flip it
        simple_flipped = logo[115:0:-1,:]

        n = 4

        plt.subplot(n,1,1)
        plt.imshow(sitk.GetArrayFromImage(logo))
        plt.axis('off');

        plt.subplot(n,1,2)
        plt.imshow(sitk.GetArrayFromImage(logo_subsampled))
        plt.axis('off');

        plt.subplot(n,1,3)
        plt.imshow(sitk.GetArrayFromImage(simple))
        plt.axis('off')

        plt.subplot(n,1,4)
        plt.imshow(sitk.GetArrayFromImage(simple_flipped))
        plt.axis('off');
```

1.2.6 Operations

SimpleITK supports many overloaded arithmetic, comparative, logical, and bitwise operators between images, taking into account their physical space.

Repeatedly run this cell. Uncomment out the SetDirection, SetOrigin, and SetSpacing lines one at a time. Why doesn't the SetOrigin line cause a problem? How close do two physical attributes need to be in order to be considered equivalent?

```
In [10]: img1 = sitk.Image(24,24, sitk.sitkUInt8)
         img1[0,0] = 0

         img2 = sitk.Image(img1.GetSize(), sitk.sitkUInt8)
         #img2.SetDirection([0,1,0.5,0.5])
         #img2.SetSpacing([0.5,0.8])
         #img2.SetOrigin([0.000001,0.000001])
         img2[0,0] = 255

         img3 = img1 + img2
         print(img3[0,0])
```

```
255
```

The overloaded image operators provide a "broadcast" way to do may operations on the whole image when initially they may have been done on a per-pixel basis.

Consider the following ITK filter which generates an image where the value at each pixel is it's physical location.

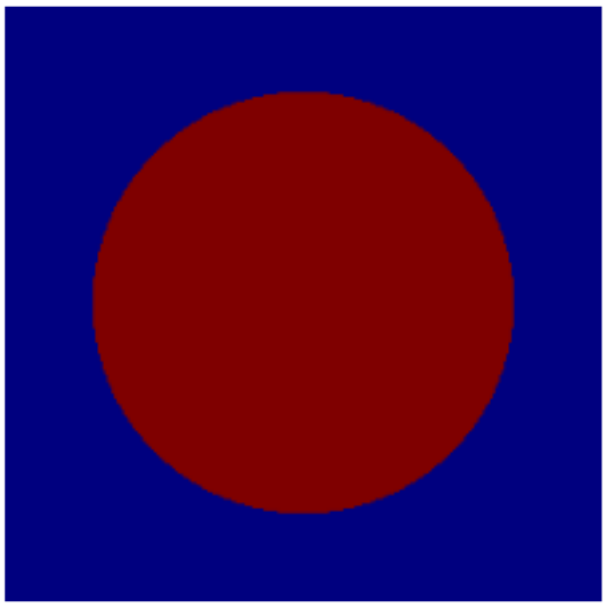
```
In [11]: size=256
         img = sitk.PhysicalPointSource(sitk.sitkVectorFloat32, [size]*2, [-1]*2, [2.0/(size-1)]*2)
         imgx = sitk.VectorIndexSelectionCast(img, 0)
```

```
        imgy = sitk.VectorIndexSelectionCast(img, 1)

        print(img[0,0], img[size//2,size//2], img[size-1,size-1])
```

```
(-1.0, -1.0) (0.003921568859368563, 0.003921568859368563) (1.0, 1.0)
```

From this a circle can be generated, by thresholding an equation of a circle.

```
In [12]: circle = (imgx**2+imgy**2)<0.5
         plt.imshow(sitk.GetArrayFromImage(circle))
         plt.axis('off');
```



1.2.7 Excercise 1:

Use the physical location images, to generate a square over the closed interface [-0.5,0.5] with the overloaded operators?

```
In [ ]:
```

1.2.8 Excercise 2:

Combine the circle and square into a single image, such that 1 value is the square, and 2 is the value of the circle pixels not in the circle.

```
In [ ]:
```

Flexible Options for Application Integration

Additional Languages: C++, CSharp, Java, Lua, Python, R, Ruby, Tcl…

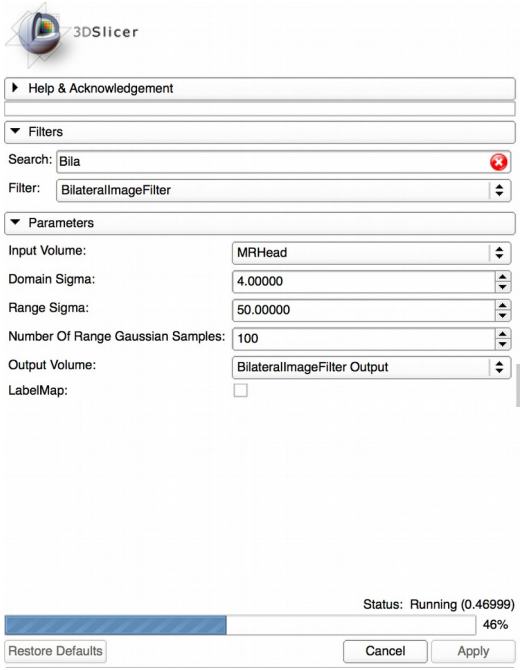
Bradley Lowekamp
Medical Science and Computing Inc.
National Library of Medicine

Advanced Multithreading for Application Integration

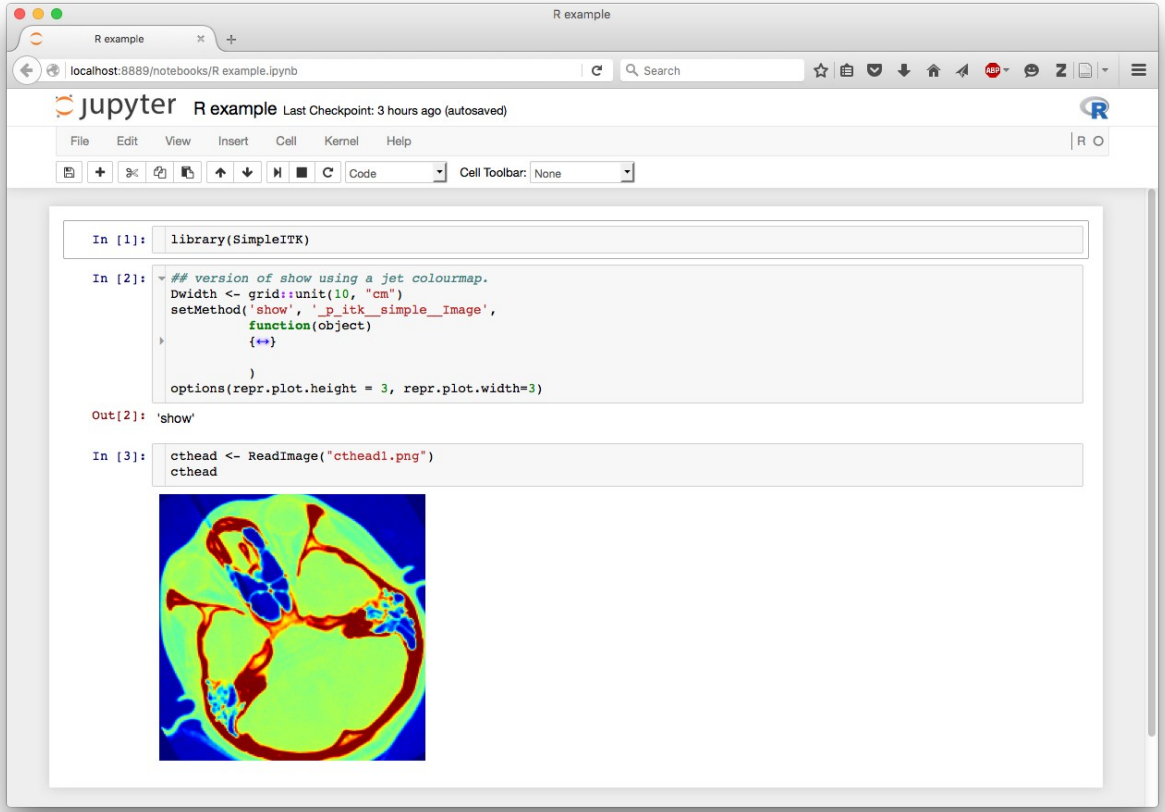
- ITK Algorithms are Multi-Threaded
- Events and Callbacks
  - progress reporting
  - process aborting
- Python Multithreading is enabled

SimpleFilters Module For 3DSlicer

- Provides a GUI for ~250 ITK image filters
- Dynamic GUI created from SimpleITK filter description
- A scripted 3D Slicer module
- Makes use of Python Multithreading to perform ITK processing in the background
- Provide GUI feedback with progress reporting and a cancel button



Jupyter Notebooks with R and SimpleITK



Simple Gaussian Example in 8 Languages

C++

```
#include <SimpleITK.h>
#include <cstdlib>
namespace sitk = itk::simple;
int main ( int argc, char* argv[] ) {
  // Read the image file
  sitk::ImageFileReader reader;
  reader.SetFileName ( std::string ( argv[1] ) );
  sitk::Image image = reader.Execute();

  // This filters perform a gaussian bluring with sigma in physical
  // space. The output image will be of real type.
  sitk::SmoothingRecursiveGaussianImageFilter gaussian;
  gaussian.SetSigma ( atof ( argv[2] ) );
  sitk::Image blurredImage = gaussian.Execute ( image );

  // Covert the real output image back to the original pixel type, to
  // make writing easier, as many file formats don't support real
  // pixels.
  sitk::CastImageFilter caster;
  caster.SetOutputPixelType( image.GetPixelIDValue() );
  sitk::Image outputImage = caster.Execute( blurredImage );

  // write the image
  sitk::ImageFileWriter writer;
  writer.SetFileName ( std::string ( argv[3] ) );
  writer.Execute ( outputImage );

  return 0;
}
```

C#

```
using System;
using itk.simple;

namespace itk.simple.examples {
    class SimpleGaussian {
        static void Main(string[] args) {
            try {
                if (args.Length < 3) {
                    Console.WriteLine("Usage: SimpleGaussian <input> <sigma> <output>");
                    return;
                }
                // Read input image
                ImageFileReader reader = new ImageFileReader();
                reader.SetFileName(args[0]);
                Image image = reader.Execute();

                // Execute Gaussian smoothing filter
                SmoothingRecursiveGaussianImageFilter gaussian =
                  new SmoothingRecursiveGaussianImageFilter();
                gaussian.SetSigma(Double.Parse(args[1]));
                image = gaussian.Execute(image);

                // Write output image
                ImageFileWriter writer = new ImageFileWriter();
                writer.SetFileName(args[2]);
                writer.Execute(image);

            } catch (Exception ex) {
                Console.WriteLine(ex);
            }
        }
    }
}
```

Java

```
import org.itk.simple.*;

class SimpleGaussian {

  public static void main(String argv[]) {

    if ( argv.length < 3 ) {
      System.out.println("Usage: java SimpleGaussian <input> <sigma> <output>");
      return;
    }

    org.itk.simple.ImageFileReader reader = new org.itk.simple.ImageFileReader();
    reader.setFileName(argv[0]);
    Image img = reader.execute();

    SmoothingRecursiveGaussianImageFilter filter =
      new SmoothingRecursiveGaussianImageFilter();
    filter.setSigma( Double.valueOf( argv[1] ).doubleValue() );
    Image blurredImg = filter.execute(img);

    CastImageFilter caster = new CastImageFilter();
    caster.setOutputPixelType( img.getPixelIDValue() );
    Image castImg = caster.execute( blurredImg );

    ImageFileWriter writer = new ImageFileWriter();
    writer.setFileName(argv[2]);
    writer.execute( castImg );

  }

}
```

Lua

```
if #arg < 3 then
  print ( "Usage: SimpleGaussian <input> <sigma> <output>" )
  os.exit ( 1 )
end

reader = SimpleITK.ImageFileReader()
-- Remember that Lua arrays are 1-based,
-- and that arg does not contain the application name!
reader:SetFileName ( arg[1] )
image = reader:Execute();

inputPixelType = image:GetPixelIDValue()

gaussian = SimpleITK.SmoothingRecursiveGaussianImageFilter()
gaussian:SetSigma ( arg[2] )
image = gaussian:Execute ( image );

caster = SimpleITK.CastImageFilter();
caster:SetOutputPixelType( inputPixelType );
image = caster:Execute( image )

writer = SimpleITK.ImageFileWriter()
writer:SetFileName ( arg[3] )
writer:Execute ( image );
```

Python

```
import SimpleITK as sitk
import sys

if len ( sys.argv ) < 4:
    print "Usage: SimpleGaussian <input> <sigma> <output>";
    sys.exit ( 1 )


reader = sitk.ImageFileReader()
reader.SetFileName ( sys.argv[1] )
image = reader.Execute()

pixelID = image.GetPixelIDValue()

gaussian = sitk.SmoothingRecursiveGaussianImageFilter()
gaussian.SetSigma ( float ( sys.argv[2] ) )
image = gaussian.Execute ( image )

caster = sitk.CastImageFilter()
caster.SetOutputPixelType( pixelID )
image = caster.Execute( image )

writer = sitk.ImageFileWriter()
writer.SetFileName ( sys.argv[3] )
writer.Execute ( image );
```

R

```
library(SimpleITK)

args <- commandArgs( TRUE )

myreader <- ImageFileReader()
myreader$SetFileName(args[[1]] )
myimage <- myreader$Execute()

pixeltype <- myimage$GetPixelID()

myfilter <- SmoothingRecursiveGaussianImageFilter()
myfilter$SetSigma(as.numeric(args[2]) )
smoothedimage <- myfilter$Execute(myimage )

mycaster <- CastImageFilter()
mycaster$SetOutputPixelType(pixeltype )
castedimage <- mycaster$Execute(smoothedimage )

mywriter <- ImageFileWriter()
mywriter$SetFileName(args[[3]] )
mywriter$Execute(castedimage )
```

Ruby

```ruby
require 'simpleitk'

if ARGV.length != 3 then
  puts "Usage: SimpleGaussian <input> <sigma> <output>";
  exit( 1 )
end

reader = Simpleitk::ImageFileReader.new
reader.set_file_name( ARGV[0] )
image = reader.execute

inputPixelType = image.get_pixel_idvalue

gaussian = Simpleitk::SmoothingRecursiveGaussianImageFilter.new
gaussian.set_sigma ARGV[1].to_f
image = gaussian.execute image;

caster = Simpleitk::CastImageFilter.new
caster.set_output_pixel_type inputPixelType
image = caster.execute image

writer = Simpleitk::ImageFileWriter.new
writer.set_file_name ARGV[2]
writer.execute image
```

Tcl

```
if { $argc < 3 } {
    puts "Usage: SimpleGaussian <input> <sigma> <output>"
    exit 1
}

ImageFileReader reader
reader SetFileName [ lindex $argv 0 ]
set image [ reader Execute ]

set pixelID [ $image GetPixelIDValue ]

SmoothingRecursiveGaussianImageFilter gaussian
gaussian SetSigma [ lindex $argv 1 ]
set image [ gaussian Execute $image ]

CastImageFilter caster
caster SetOutputPixelType $pixelID
set image [ caster Execute $image ]

ImageFileWriter writer
writer SetFileName [ lindex $argv 2]
writer Execute $image

# Tcl requires explicit cleanup Cleanup
reader -delete
gaussian -delete
caster -delete
$image -delete
writer -delete
```

ITK and JavaScript - the Future is Here!

ITK in Biomedical Research and Commercial Applications

SPIE Medical Imaging, 2016

Matthew McCormick, PhD., Kitware, Inc.

JavaScript Background

JavaScript: Language of the Web Browser

The only programming language universally available in web browsers is JavaScript. JavaScript first appeared in **1995** with the Netscape browser, and it was first standardized in the **ECMAScript** language specification in 1997.

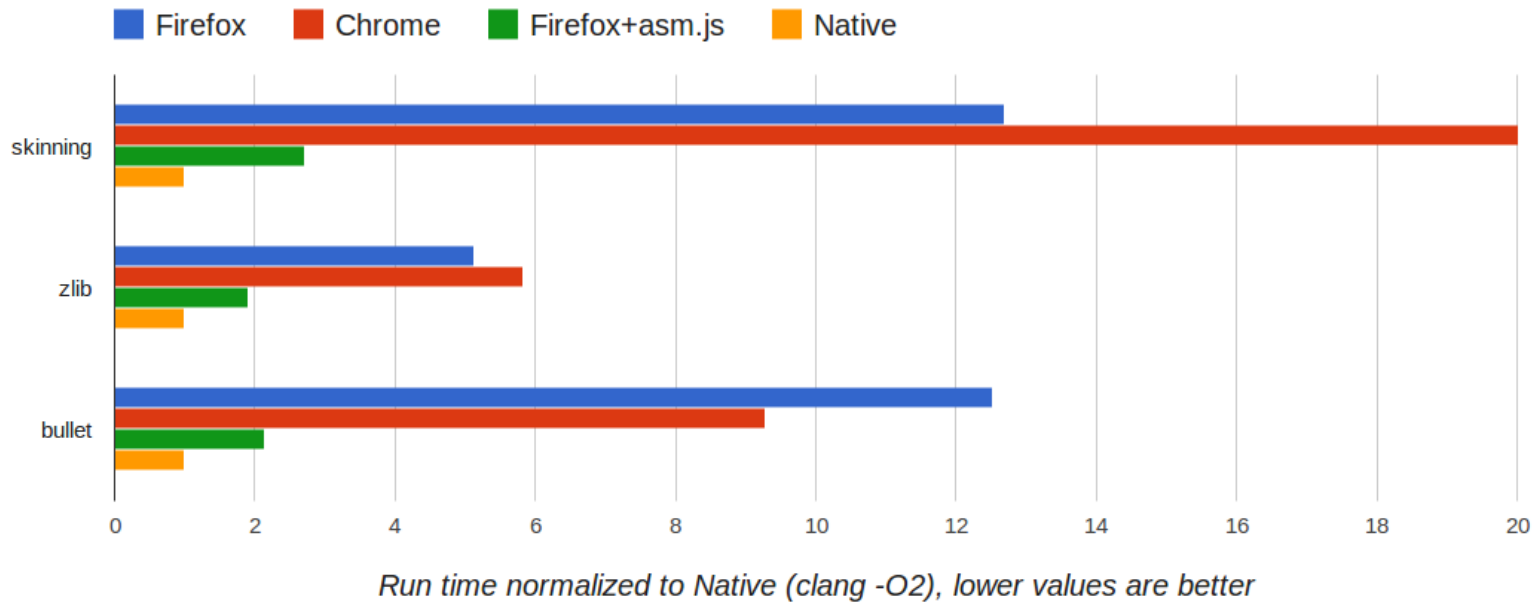
Brendan Eich

JavaScript: Language of the Web Browser (cont.)

While it was initially used to provide limited interactivity to web pages, the advent of asynchronous JavaScript and XML in **2005** began the development of featureful **web applications**.

JavaScript: Relevant to Scientific Computing

The introduction of highly optimized JavaScript interpreters and technologies like *asm.js* in **2013** made JavaScript's performance relevant to scientific applications.



Node.js

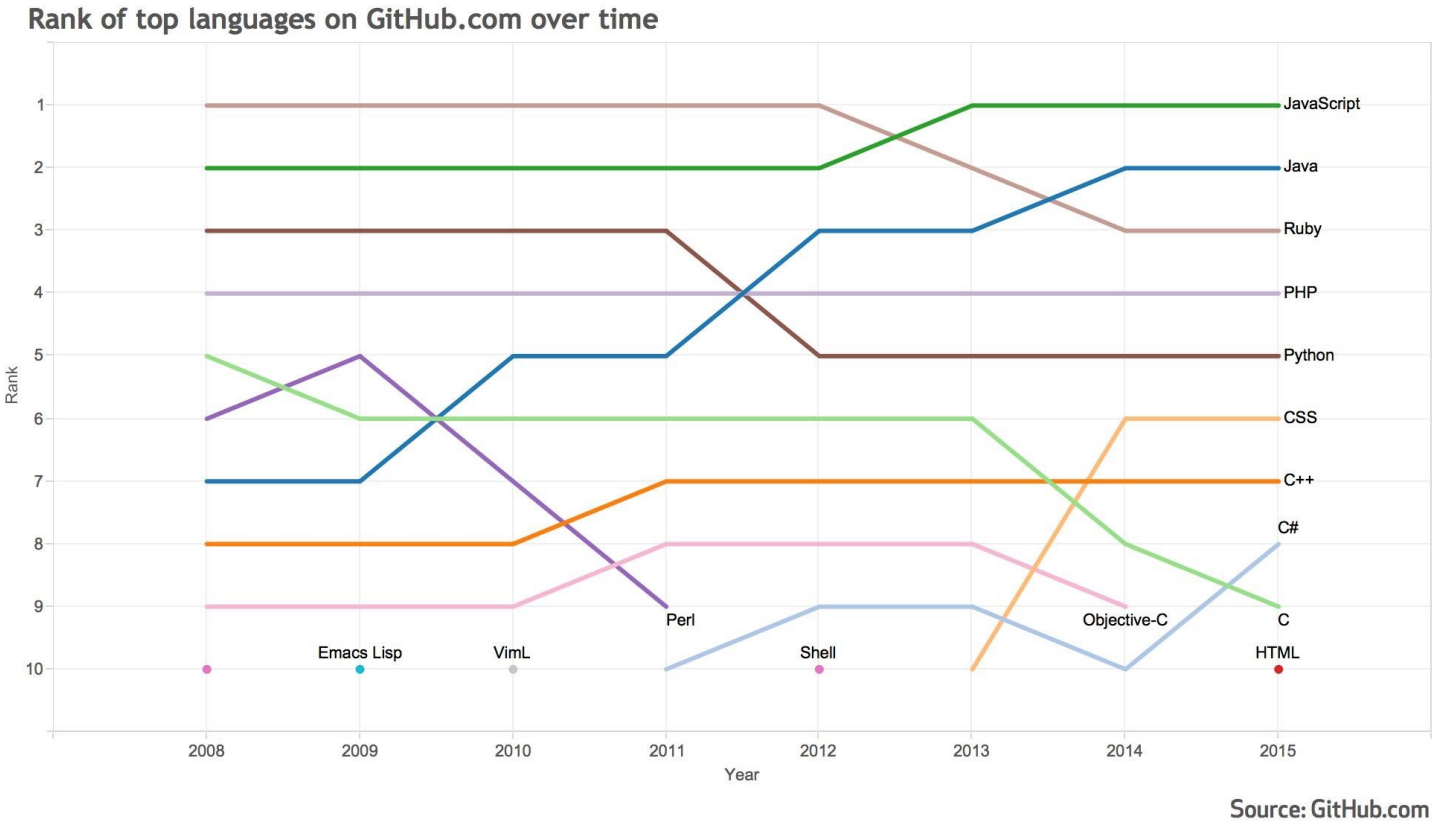
The Node.js allows JavaScript to be executed on a workstation or server

The *node* executable is a JavaScript interpreter: *$node myscript.js argv1*

Node.js is useful for testing

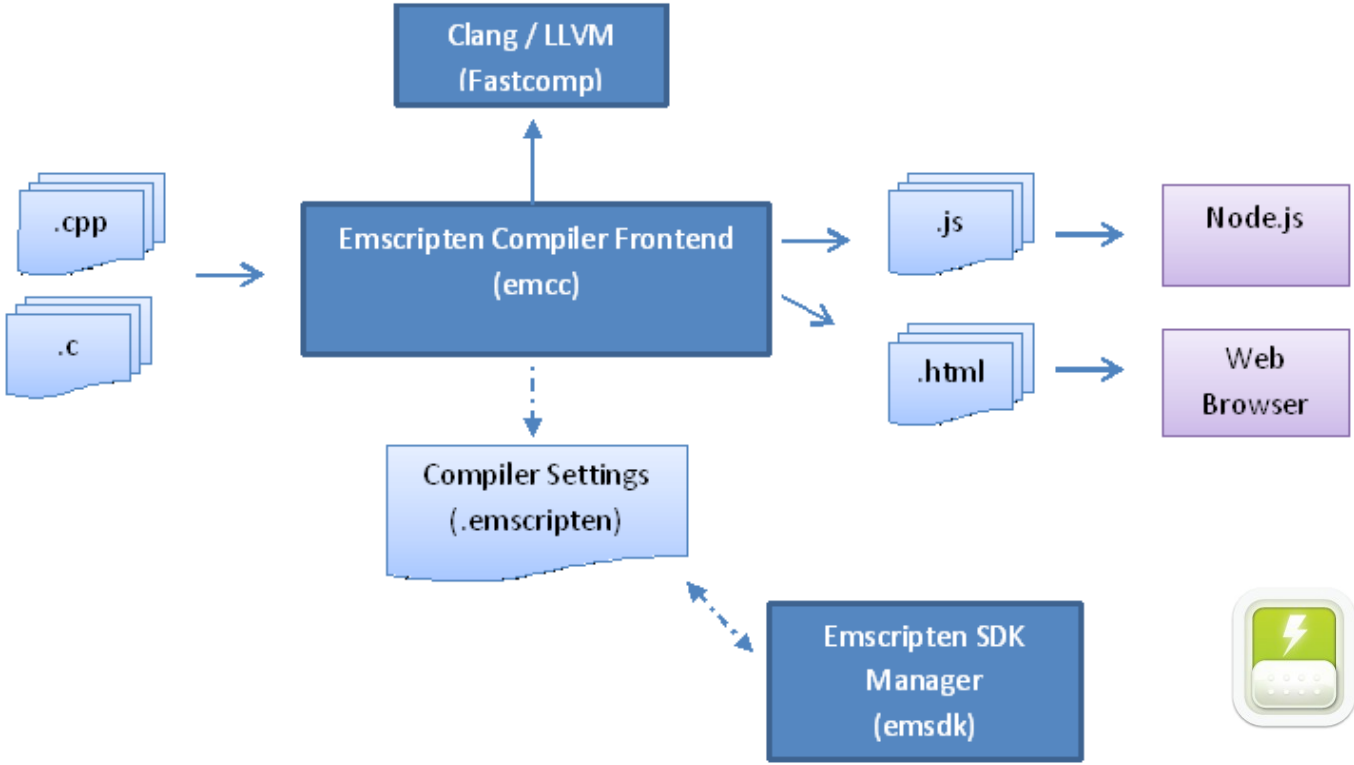
The JavaScript ecosystem is dominated by the Node Package Manager, *npm*

The most popular language on GitHub: JavaScript



Compiling ITK to JavaScript

Emscripten: Compile C++ to Highly Optimized JS



emscripten

CMake + ITK + Emscripten = JavaScript

CMake has excellent cross-compilation capabilities

CMake can use Emscripten as a cross-compilation toolchain

Docker can be used to provide a quick setup of CMake + Emscripten

https://blog.kitware.com/compile-cc-into-javascript-with-emscripten-and-docker/

All of ITK's default modules (including third-party modules) and their tests can

Expected Nightly Exotic

Site	Build Name	Update	Configure		Build		Test			
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass	Build Time
eldorado.kitware	Emscripten-Javascript	0	0	0	0	0	0	53 -1	2159 +1	14 hours ago

ITK Emscripten Tips

Re-use ITK C++ executables without modification!

Build against an ITK built with Emscripten, and use *CMAKE_TOOLCHAIN_FILE*

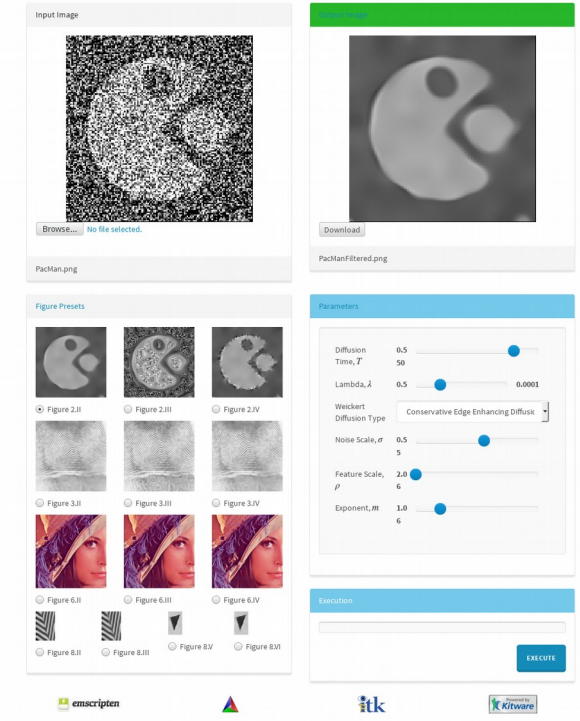
find_package(ITK COMPONENTS ModuleA … is critical to reduce the output JavaScript file size

Understand the limitations and approaches available to allow Node.js to access the local filesystem

See the ITK TestDriver code for reference

A Reproducible Publication Example

http://insightsoftwareconsortium.github.io/ITKAnisotropicDiffusionLBR/

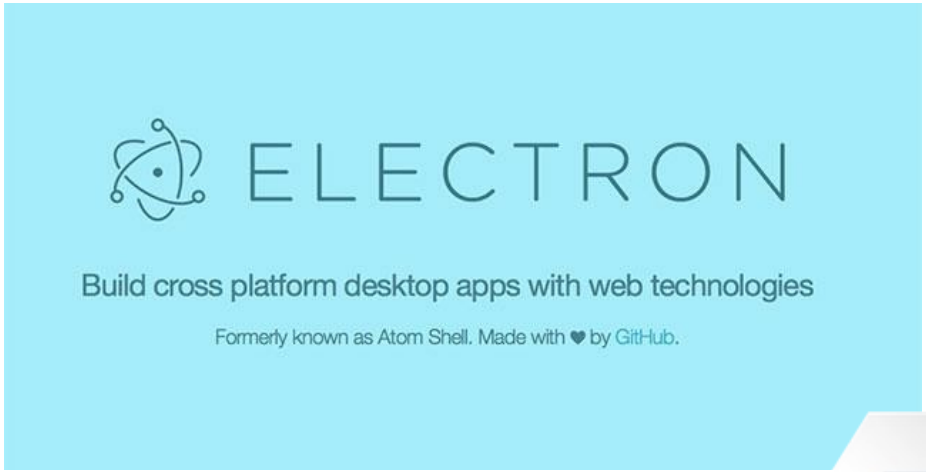


What’s Next

JavaScript Performance Improvements

- Multi-threaded parallism (?)
- WebAssembly
  - Even faster
  - Smaller generated code

JavaScript Applications on the Desktop and Mobile



APACHE
CORDOVA™

ITKBridgeJavaScript Module

JavaScript / Emscripten interface to itk::Image

....

Juan Carlos Prieto, Matt McCormick, and You!

https://github.com/InsightSoftwareConsortium/ITKBridgeJavaScript

4_a_SimpleITK_Segmentation

February 28, 2016

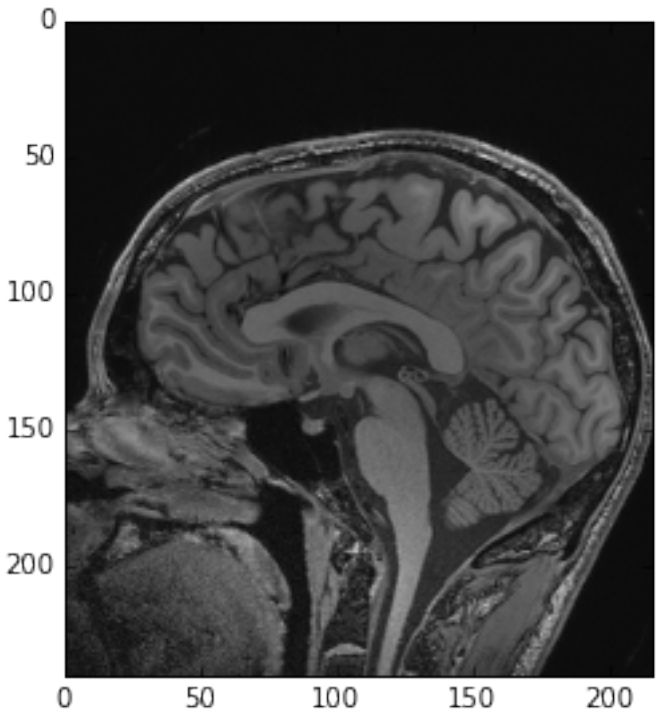
1 Introduction to ITK Segmentation in SimpleITK Notebooks

Goal: To become familiar with basic segmentation algorithms available in ITK, and interactively explore their parameter space.

Image segmentation filters process an image to partition it into (hopefully) meaningful regions. The output is commonly an image of integers where each integer can represent an object. The value 0 is commonly used for the background, and 1 (sometimes 255) for a foreground object.

In [1]:
```
import matplotlib.pyplot as plt
%matplotlib inline
from ipywidgets import interact, FloatSlider

from __future__ import print_function

import SimpleITK as sitk

# Download data to work on
from downloaddata import fetch_data as fdata
from myshow import myshow, myshow3d
```

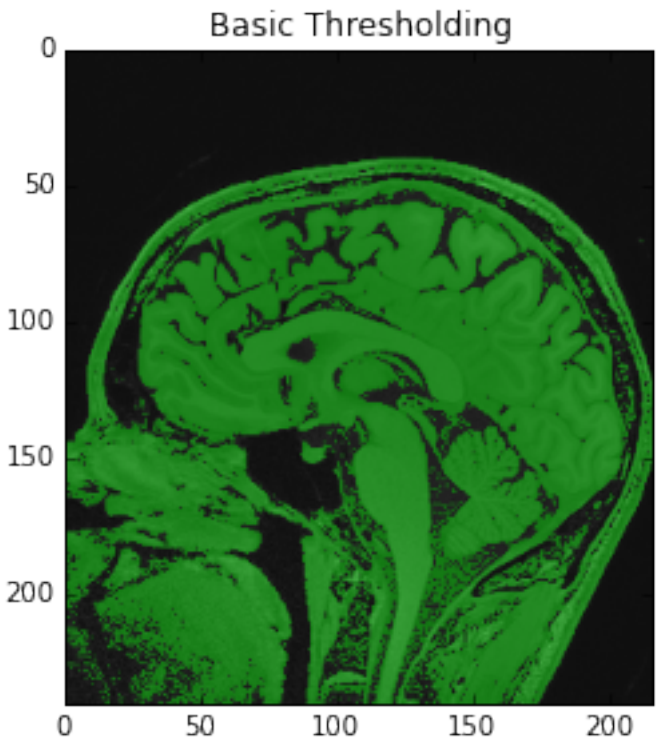
In [2]:
```
img_T1 = sitk.ReadImage(fdata("nac-hncma-atlas2013-Slicer4Version/Data/A1_grayT1.nrrd"))
img_T2 = sitk.ReadImage(fdata("nac-hncma-atlas2013-Slicer4Version/Data/A1_grayT2.nrrd"))

# To visualize the labels image in RGB with needs a image with 0-255 range
img_T1_255 = sitk.Cast(sitk.RescaleIntensity(img_T1), sitk.sitkUInt8)
img_T2_255 = sitk.Cast(sitk.RescaleIntensity(img_T2), sitk.sitkUInt8)

myshow3d(img_T1)
```



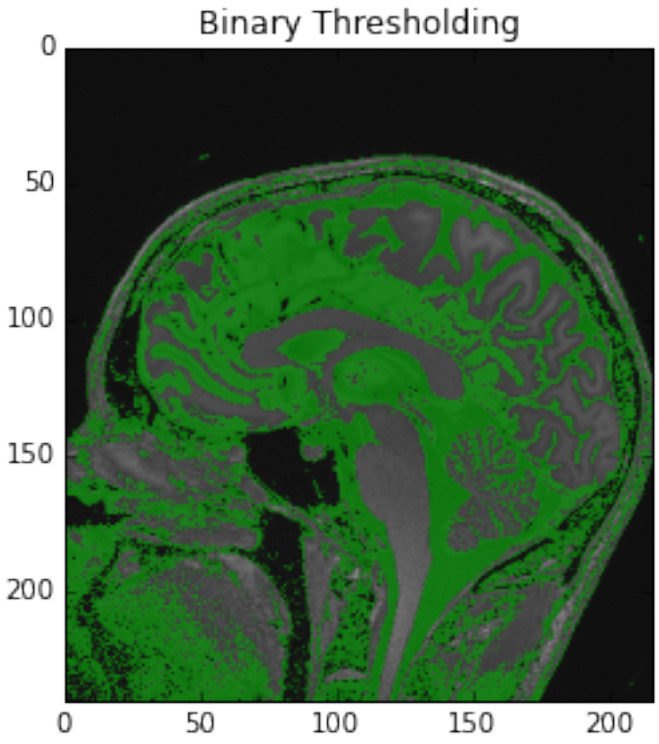
1.1 Thresholding

Thresholding is the most basic form of segmentation. It simply labels the pixels of an image based on the intensity range without respect to geometry or connectivity.

```
In [3]: seg = img_T1>200
        myshow(sitk.LabelOverlay(img_T1_255, seg), "Basic Thresholding")
```

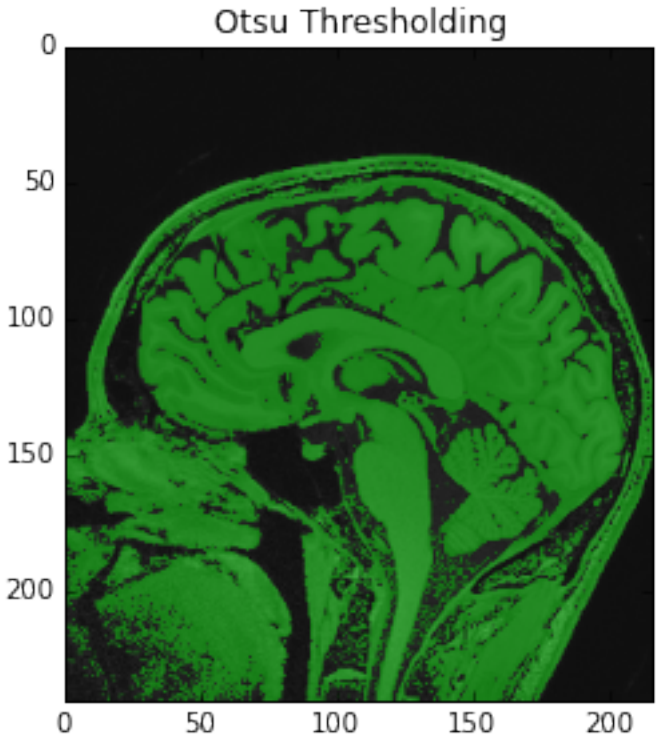



```
In [4]: seg = sitk.BinaryThreshold(img_T1, lowerThreshold=100, upperThreshold=400, insideValue=1, outsi
        myshow(sitk.LabelOverlay(img_T1_255, seg), "Binary Thresholding")
```



ITK has a number of histogram based automatic thresholding filters including Haung, MaximumEntropy, Triangle, and the popular Otsu's method. These methods create a histogram then use a heuristic to determine a threshold value.

```
In [5]: otsu_filter = sitk.OtsuThresholdImageFilter()
        otsu_filter.SetInsideValue(0)
        otsu_filter.SetOutsideValue(1)
        seg = otsu_filter.Execute(img_T1)
        myshow(sitk.LabelOverlay(img_T1_255, seg), "Otsu Thresholding")

        print(otsu_filter.GetThreshold() )
```




```
227.550582886
```

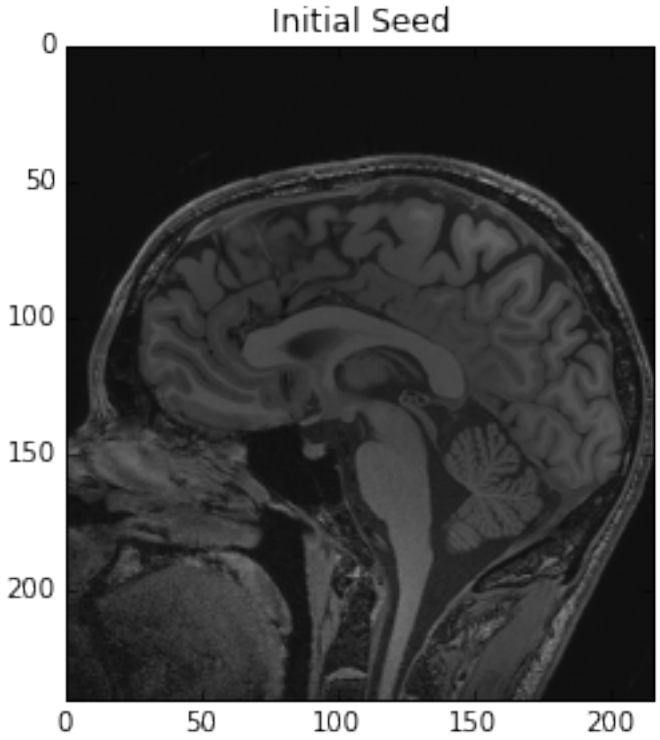
1.2 Region Growing Segmentation

The first step of improvement upon the naive thresholding is a class of algorithms called region growing. This includes:

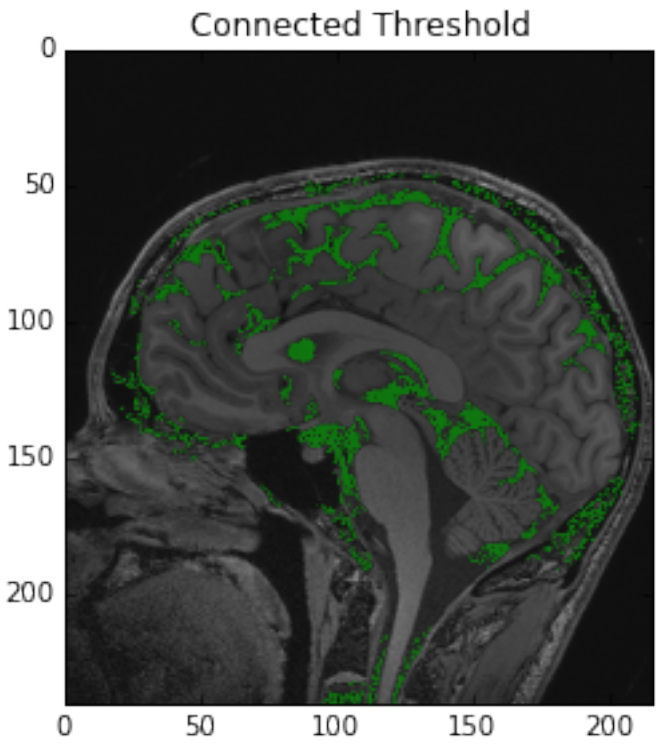
ConnectedThreshold
ConfidenceConnected
VectorConfidenceConnected
NeighborhoodConnected

Earlier we used 3D Slicer to determine that index: (136,139,97) was a good seed for the left lateral ventricle.

In [6]:
```
seed = (132,142,96)
seg = sitk.Image(img_T1.GetSize(), sitk.sitkUInt8)
seg.CopyInformation(img_T1)
seg[seed] = 1
seg = sitk.BinaryDilate(seg, 3)
myshow(sitk.LabelOverlay(img_T1_255, seg), "Initial Seed")
```

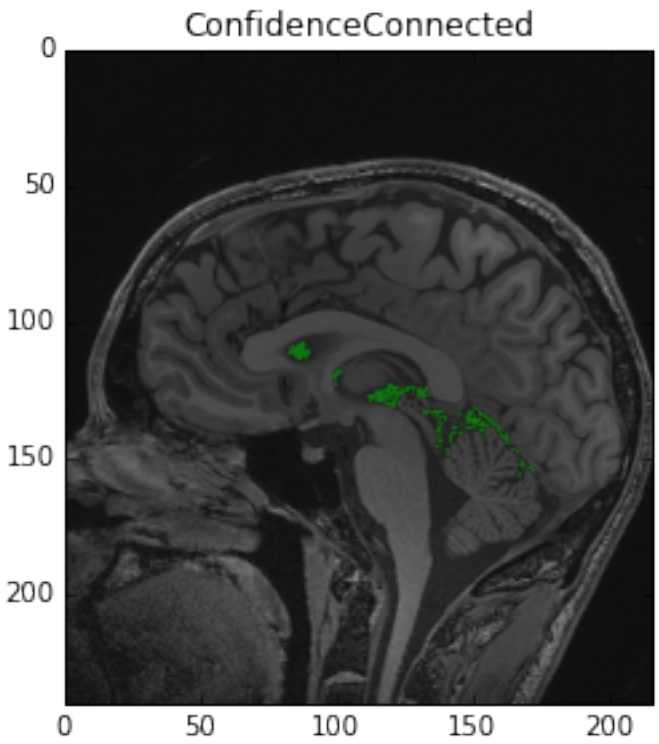


In [7]:
```
seg = sitk.ConnectedThreshold(img_T1, seedList=[seed], lower=100, upper=190)

myshow(sitk.LabelOverlay(img_T1_255, seg), "Connected Threshold")
```

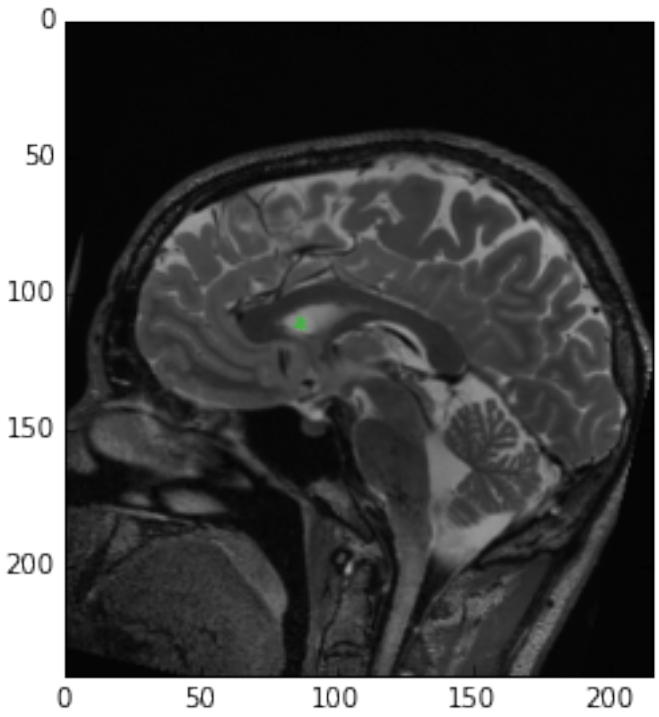


Improving upon this is the ConfidenceConnected filter, which uses the initial seed or current segmentation to estimate the threshold range.

```
In [8]: seg = sitk.ConfidenceConnected(img_T1, seedList=[seed],
                                       numberOfIterations=1,
                                       multiplier=2.5,
                                       initialNeighborhoodRadius=1,
                                       replaceValue=1)

        myshow(sitk.LabelOverlay(img_T1_255, seg), "ConfidenceConnected")
```



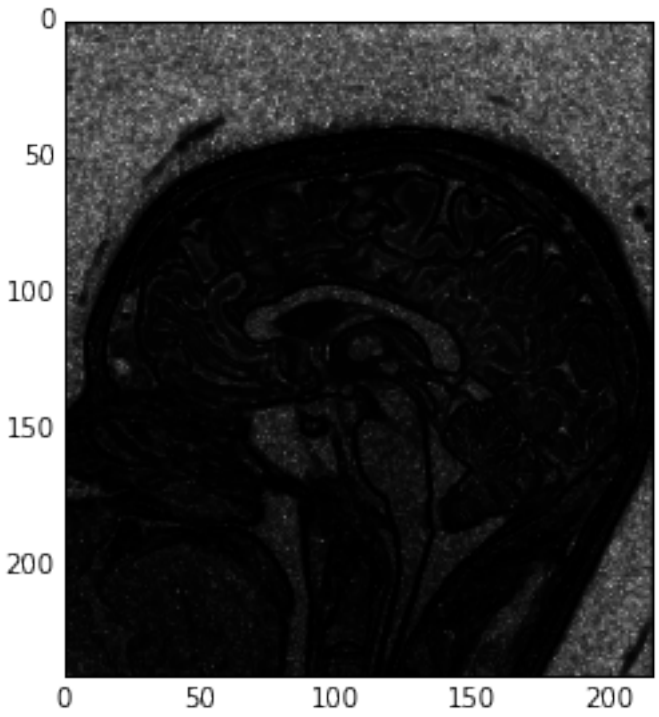

```
In [9]: img_multi = sitk.Compose(img_T1, img_T2)
        seg = sitk.VectorConfidenceConnected(img_multi, seedList=[seed],
                                                        numberOfIterations=1,
                                                        multiplier=2.5,
                                                        initialNeighborhoodRadius=1)
        myshow(sitk.LabelOverlay(img_T2_255, seg))
```



1.3 Fast Marching Segmentation

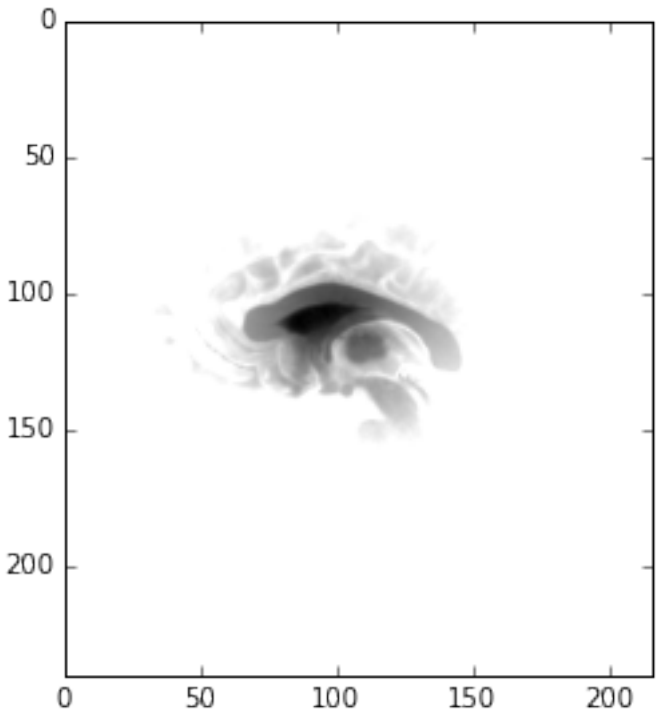
The FastMarchingImageFilter implements a fast marching solution to a simple level set evolution problem (eikonal equation). In this example, the speed term used in the differential equation is provided in the form of an image. The speed image is based on the gradient magnitude and mapped with the bounded reciprocal $1/(1+x)$.

```
In [10]: seed = (132,142,96)
         feature_img = sitk.GradientMagnitudeRecursiveGaussian(img_T1, sigma=.5)
         speed_img = sitk.BoundedReciprocal(feature_img) # This is parameter free unline the Sigmoid
         myshow(speed_img)
```

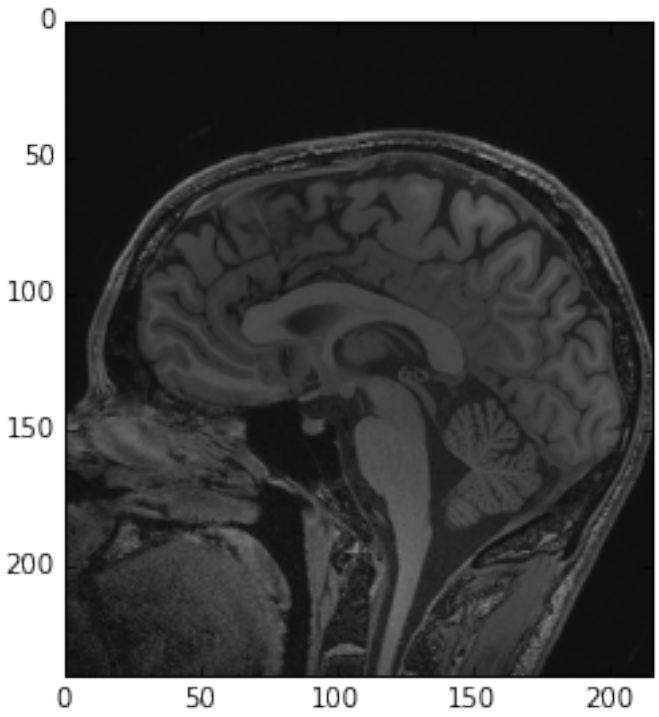


The output of the FastMarchingImageFilter is a time-crossing map that indicates, for each pixel, how much time it would take for the front to arrive at the pixel location.

```
In [11]: fm_filter = sitk.FastMarchingBaseImageFilter()
         fm_filter.SetTrialPoints([seed])
         fm_filter.SetStoppingValue(1000)
         fm_img = fm_filter.Execute(speed_img)
         myshow(sitk.Threshold(fm_img,
                               lower=0.0,
                               upper=fm_filter.GetStoppingValue(),
                               outsideValue=fm_filter.GetStoppingValue()+1))
```




```
In [12]: def fm_callback(img, time, z):
             seg = img<time;
             myshow(sitk.LabelOverlay(img_T1_255[:,:,z], seg[:,:,z]))

         interact( lambda **kwargs: fm_callback(fm_img, **kwargs),
                     time=FloatSlider(min=0.05, max=1000.0, step=0.05, value=100.0),
                     z=(0,fm_img.GetSize()[2]-1))
```



Out[12]: <function __main__.<lambda>>

1.4 Level-Set Segmentation

There are a variety of level-set based segmentation filter available in ITK:
GeodesicActiveContour
ShapeDetection
ThresholdSegmentation
LaplacianSegmentation
ScalarChanAndVese
There is also a modular Level-set framework which allows composition of terms and easy extension in C++.
First we create a label image from our seed.

```
In [13]: seed = (132,142,96)

         seg = sitk.Image(img_T1.GetSize(), sitk.sitkUInt8)
         seg.CopyInformation(img_T1)
         seg[seed] = 1
         seg = sitk.BinaryDilate(seg, 3)
```

Use the seed to estimate a reasonable threshold range.

```
In [14]: stats = sitk.LabelStatisticsImageFilter()
         stats.Execute(img_T1, seg)

         factor = 3.5
         lower_threshold = stats.GetMean(1)-factor*stats.GetSigma(1)
```

```
        upper_threshold = stats.GetMean(1)+factor*stats.GetSigma(1)
        print(lower_threshold,upper_threshold)
```

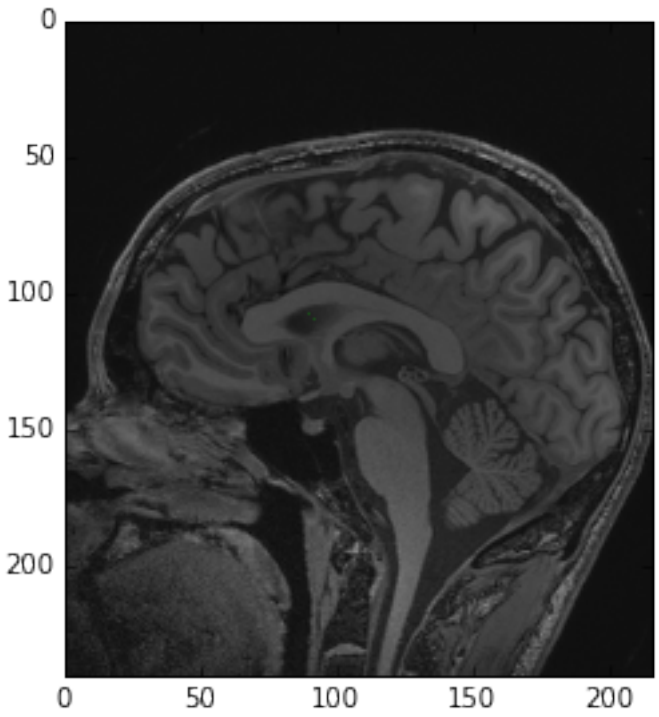
```
81.2518454131 175.008446683
```

In [15]:
```
init_ls = sitk.SignedMaurerDistanceMap(seg, insideIsPositive=True, useImageSpacing=True)
```

In [16]:
```
lsFilter = sitk.ThresholdSegmentationLevelSetImageFilter()
lsFilter.SetLowerThreshold(lower_threshold)
lsFilter.SetUpperThreshold(upper_threshold)
lsFilter.SetMaximumRMSError(0.02)
lsFilter.SetNumberOfIterations(1000)
lsFilter.SetCurvatureScaling(.5)
lsFilter.SetPropagationScaling(1)
lsFilter.ReverseExpansionDirectionOn()
ls = lsFilter.Execute(init_ls, sitk.Cast(img_T1, sitk.sitkFloat32))
print(lsFilter)
```

```
itk::simple::ThresholdSegmentationLevelSetImageFilter
  LowerThreshold: 81.2518
  UpperThreshold: 175.008
  MaximumRMSError: 0.02
  PropagationScaling: 1
  CurvatureScaling: 0.5
  NumberOfIterations: 1000
  ReverseExpansionDirection: 1
  ElapsedIterations: 119
  RMSChange: 0.0180966
  Debug: 0
  NumberOfThreads: 8
  Commands: (none)
  ProgressMeasurement: 0.119
  ActiveProcess: (none)
```

In [17]:
```
myshow(sitk.LabelOverlay(img_T1_255, ls>0))
```


0
50
100
150
200
0
50
100
150
200

4_b_SimpleITK_Registration

February 28, 2016

Introduction to ITK Registration in SimpleITK Notebooks
SimpleITK features:
Registraiton is either 2D/2D or 3D/3D.
Pixel types are either sitkFloat32 or sitkFloat64 (use the SimpleITK Cast() function if your image's pixel type is something else).

There are many options for creating an instance of the registration framework, all of which are configured in SimpleITK via methods of the ImageRegistrationMethod class. This class encapsulates many of the components available in ITK for constructing a registration instance.

0.1 Registration Components

Currently, the available choices from the following groups of ITK components are:

0.1.1 Optimizers

The SimpleITK registration framework supports several optimizer types via the SetMetricAsX() methods, these include:
Exhaustive
Nelder-Mead downhill simplex, a.k.a. Amoeba.
Variations on gradient descent:
GradientDescent
GradientDescentLineSearch
RegularStepGradientDescent
ConjugateGradientLineSearch
L-BFGS-B (Limited memory Broyden, Fletcher,Goldfarb,Shannon-Bound Constrained) - supports the use of simple constraints ($l \leq x \leq u$)

0.1.2 Similarity metrics

The SimpleITK registration framework supports several metric types via the SetMetricAsX() methods, these include:
MeanSquares
Demons
Correlation
ANTSNeighborhoodCorrelation
JointHistogramMutualInformation
MattesMutualInformation

0.1.3 Interpolators

The SimpleITK registration framework supports several interpolators via the SetInterpolator() method, which receives one of the following enumerations:
sitkNearestNeighbor

sitkLinear
sitkBSpline
sitkGaussian
sitkHammingWindowedSinc
sitkCosineWindowedSinc
sitkWelchWindowedSinc
sitkLanczosWindowedSinc
sitkBlackmanWindowedSinc

In [1]:
```
import SimpleITK as sitk

from __future__ import print_function

# Utility method that either downloads data from the MIDAS repository or
# if already downloaded returns the file name for reading from disk (cached data).
from downloaddata import fetch_data as fdata

# Always write output to a separate directory, we don't want to pollute the source directory.
import os
OUTPUT_DIR = 'Output'
```

0.2 Utility functions

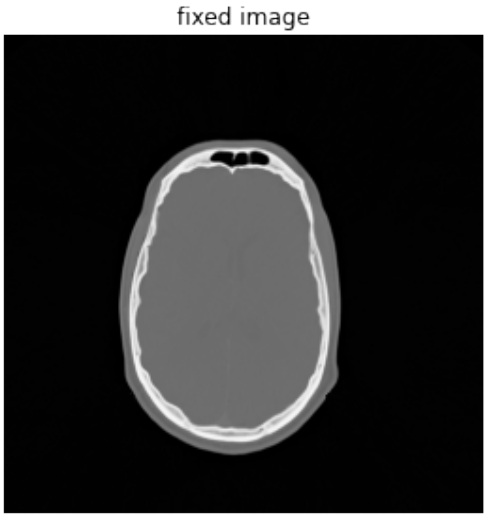
Callback functions for image display and for ploting the similarity metric during registration.

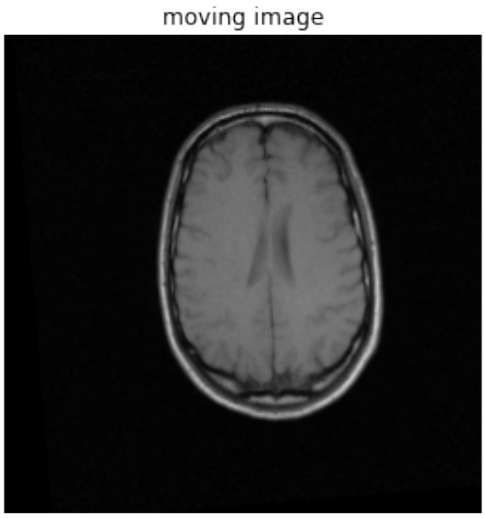
In [2]:
```
%matplotlib inline
%run registration_utilities.py
```

0.3 Read images

We first read the images, casting the pixel type to that required for registration (Float32 or Float64) and look at them.

In [3]:
```
fixed_image =  sitk.ReadImage(fdata("RIRE/training_001_ct.mha"), sitk.sitkFloat32)
moving_image = sitk.ReadImage(fdata("RIRE/training_001_mr_T1.mha"), sitk.sitkFloat32)

interact(lambda image1_z, image2_z, image1, image2,:display_scalar_images(image1_z, image2_z, i
                                                                           title1='fixed image',
                                                                           title2 = 'moving imag
         image1_z=(0,fixed_image.GetSize()[2]-1),
         image2_z=(0,moving_image.GetSize()[2]-1),
         image1 = fixed(fixed_image),
         image2=fixed(moving_image));
```

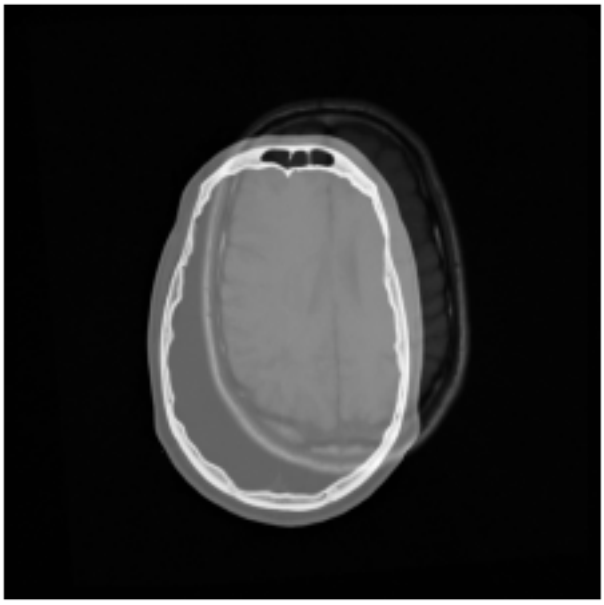




0.4 Initial Alignment

Use the CenteredTransformInitializer to align the centers of the two volumes and set the center of rotation to the center of the fixed image.

```
In [4]: initial_transform = sitk.CenteredTransformInitializer(fixed_image,
                                                              moving_image,
                                                              sitk.Euler3DTransform(),
                                                              sitk.CenteredTransformInitializerFilter.GE

        display_registration_results(fixed_image, moving_image, initial_transform)
        print(initial_transform)
```



```
itk::simple::Transform
 Euler3DTransform (0x252c450)
   RTTI typeinfo:   itk::Euler3DTransform<double>
   Reference Count: 1
   Modified Time: 1829
   Debug: Off
   Object Name:
   Observers:
     none
   Matrix:
     1 0 0
     0 1 0
     0 0 1
   Offset: [-7.61852, -7.61852, -6]
   Center: [166.994, 166.994, 56]
   Translation: [-7.61852, -7.61852, -6]
   Inverse:
     1 0 0
     0 1 0
     0 0 1
   Singular: 0
   Euler's angles: AngleX=0 AngleY=0 AngleZ=0
   m_ComputeZYX = 0
```

0.5 Registration - Understanding Your Input and Output

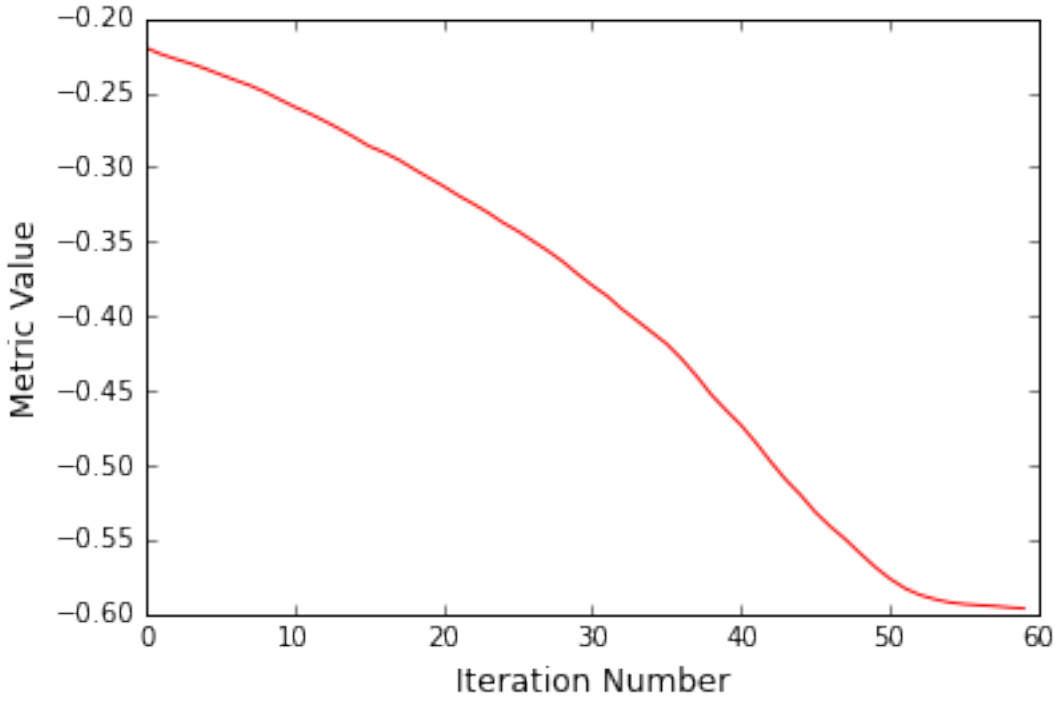
The specific registration task at hand estimates a 3D rigid transformation between images of different modalities. There are multiple components from each group (optimizers, similarity metrics, interpolators) that are appropriate for the task. Note that each component selection requires setting some parameter values. We have made the following choices:

Similarity metric, mutual information (Mattes MI):
Number of histogram bins, 50.
Sampling strategy, random.
Sampling percentage, 1%.
Interpolator, sitkLinear.
Optimizer, gradient descent:
Learning rate, step size along traversal direction in parameter space, 1.0 .
Number of iterations, maximal number of iterations, 60.

0.5.1 Version 1 - Classic

Use the v4 registration framework in the ITK v3 manner.

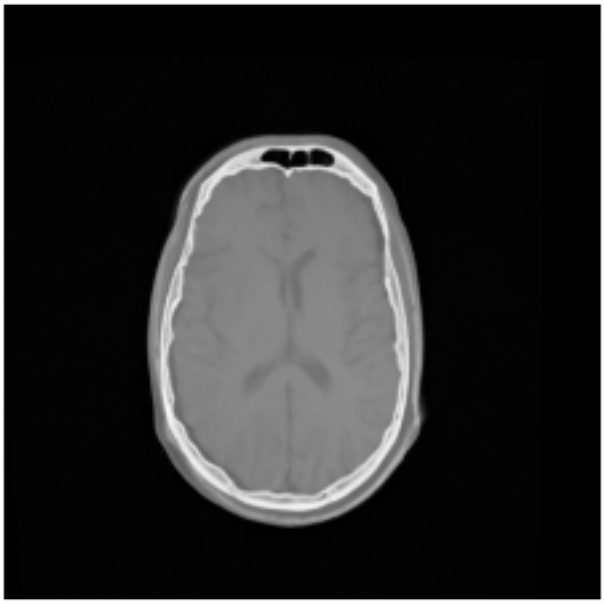
In [5]:
```
registration_method = sitk.ImageRegistrationMethod()

registration_method.SetMetricAsMattesMutualInformation(numberOfHistogramBins=50)
registration_method.SetMetricSamplingStrategy(registration_method.RANDOM)
registration_method.SetMetricSamplingPercentage(0.01)

registration_method.SetInterpolator(sitk.sitkLinear)

registration_method.SetOptimizerAsGradientDescent(learningRate=1.0, numberOfIterations=60)
registration_method.SetOptimizerScalesFromPhysicalShift()

registration_method.SetInitialTransform(initial_transform, inPlace=False)

registration_method.AddCommand(sitk.sitkStartEvent, metric_start_plot)
registration_method.AddCommand(sitk.sitkEndEvent, metric_end_plot)
registration_method.AddCommand(sitk.sitkIterationEvent,
                               lambda: metric_plot_values(registration_method))

final_transform_v1 = registration_method.Execute(fixed_image, moving_image)

print(final_transform_v1)
```




```
itk::simple::Transform
 CompositeTransform (0x25a2860)
   RTTI typeinfo:   itk::CompositeTransform<double, 3u>
   Reference Count: 1
   Modified Time: 325846
   Debug: Off
   Object Name:
   Observers:
     none
   Transforms in queue, from begin to end:
   >>>>>>>>>
   Euler3DTransform (0x3605cb0)
     RTTI typeinfo:   itk::Euler3DTransform<double>
     Reference Count: 1
     Modified Time: 325837
     Debug: Off
     Object Name:
     Observers:
       none
     Matrix:
       0.999835 0.010531 -0.0147723
       -0.0101895 0.999683 0.023006
       0.0150099 -0.0228516 0.999626
     Offset: [17.7146, -31.0683, -19.3129]
     Center: [166.994, 166.994, 56]
     Translation: [18.6185, -31.5344, -20.6434]
     Inverse:
       0.999835 -0.0101895 0.0150099
```

```
      0.010531 0.999683 -0.0228516
      -0.0147723 0.023006 0.999626
    Singular: 0
    Euler's angles: AngleX=-0.0228536 AngleY=-0.0150144 AngleZ=-0.010534
    m_ComputeZYX = 0
  End of MultiTransform.
<<<<<<<<<<
  TransformsToOptimizeFlags, begin() to end():
     1
  TransformsToOptimize in queue, from begin to end:
  End of TransformsToOptimizeQueue.
<<<<<<<<<<
  End of CompositeTransform.
<<<<<<<<<<
```

```
In [6]: display_registration_results(fixed_image, moving_image, final_transform_v1)
        print('Final metric value: {0}'.format(registration_method.GetMetricValue()))
        print('Optimizer\'s stopping condition, {0}'.format(registration_method.GetOptimizerStopConditi
```



```
Final metric value: -0.5960219962620066
Optimizer's stopping condition, GradientDescentOptimizerv4Template: Maximum number of iterations (60) e
```

0.5.2 Version 1.1 the v4 approach

The previous example illustrated the use of the ITK v4 registration framework in an ITK v3 manner. We only referred to a single transformation which was what we optimized.

In ITK v4 the registration method accepts three transformations (if you look at the diagram above you will only see two transformations, Moving transform represents $T_{opt} \circ T_m$):

SetInitialTransform, T_{opt} - composed with the moving initial transform, maps points from the virtual image domain to the moving image domain, modified during optimization.

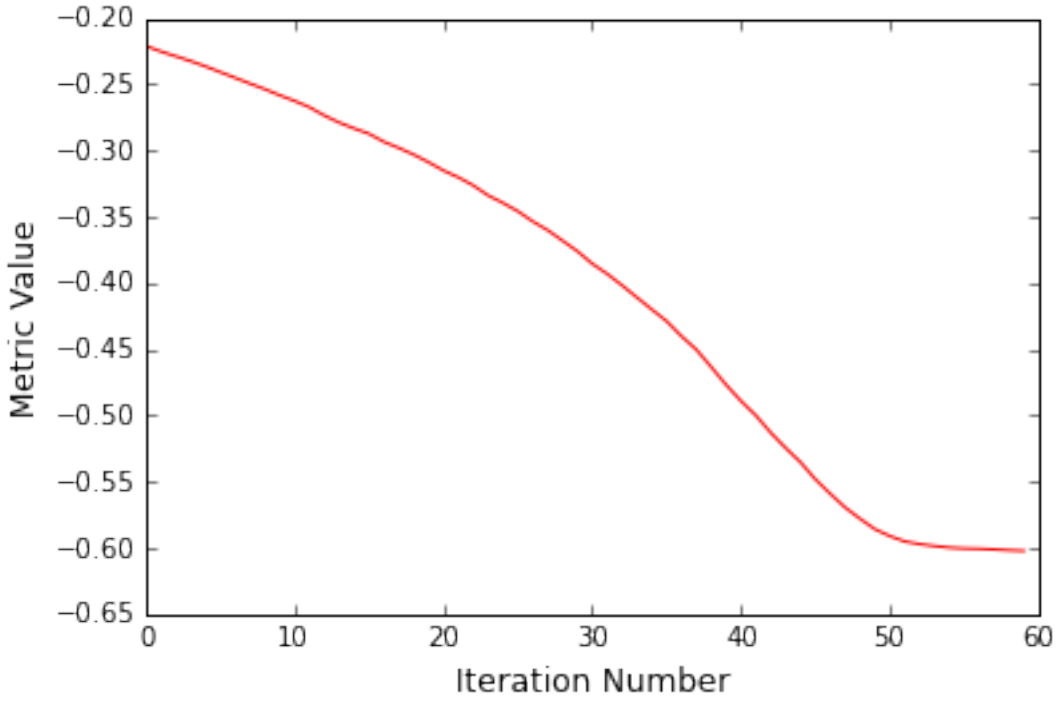
SetFixedInitialTransform T_f- maps points from the virtual image domain to the fixed image domain, never modified.

SetMovingInitialTransform T_m- maps points from the virtual image domain to the moving image domain, never modified.

The transformation that maps points from the fixed to moving image domains is thus: ${}^{M}\mathbf{p} = T_{opt}(T_m(T_f^{-1}({}^{F}\mathbf{p})))$

We now modify the previous example to use T_{opt} and T_m.

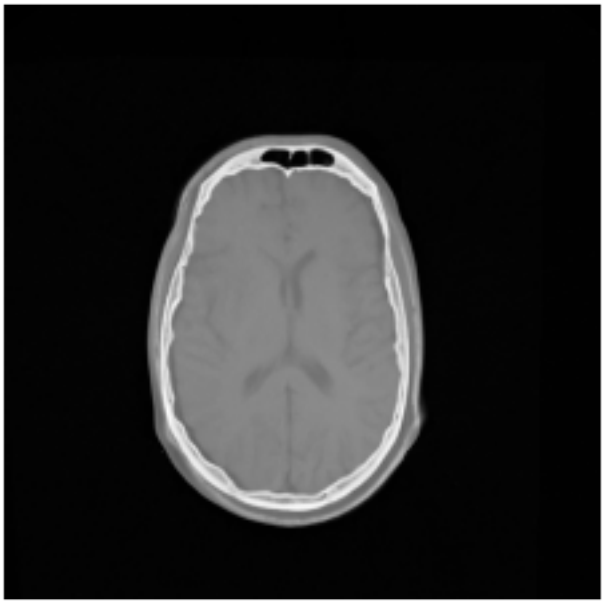
```
In [7]: registration_method = sitk.ImageRegistrationMethod()
        registration_method.SetMetricAsMattesMutualInformation(numberOfHistogramBins=50)
        registration_method.SetMetricSamplingStrategy(registration_method.RANDOM)
        registration_method.SetMetricSamplingPercentage(0.01)
        registration_method.SetInterpolator(sitk.sitkLinear)
        registration_method.SetOptimizerAsGradientDescent(learningRate=1.0, numberOfIterations=60)
        registration_method.SetOptimizerScalesFromPhysicalShift()

        # Set the initial moving and optimized transforms.
        optimized_transform = sitk.Euler3DTransform()
        registration_method.SetMovingInitialTransform(initial_transform)
        registration_method.SetInitialTransform(optimized_transform)

        registration_method.AddCommand(sitk.sitkStartEvent, metric_start_plot)
        registration_method.AddCommand(sitk.sitkEndEvent, metric_end_plot)
        registration_method.AddCommand(sitk.sitkIterationEvent,
                                       lambda: metric_plot_values(registration_method))

        registration_method.Execute(sitk.Cast(fixed_image, sitk.sitkFloat32),
                                    sitk.Cast(moving_image, sitk.sitkFloat32))

        # Need to compose the transformations after registration.
        final_transform_v11 = sitk.Transform(optimized_transform)
        final_transform_v11.AddTransform(initial_transform)

        print(final_transform_v11)
```




```
itk::simple::Transform
 CompositeTransform (0x302dff0)
   RTTI typeinfo:   itk::CompositeTransform<double, 3u>
   Reference Count: 1
   Modified Time: 650010
   Debug: Off
   Object Name:
   Observers:
     none
   Transforms in queue, from begin to end:
   >>>>>>>>>
   Euler3DTransform (0x302e300)
     RTTI typeinfo:   itk::Euler3DTransform<double>
     Reference Count: 1
     Modified Time: 650004
     Debug: Off
     Object Name:
     Observers:
       none
     Matrix:
       0.999821 0.0133511 -0.0134239
       -0.0130334 0.999639 0.0234811
       0.0137326 -0.0233019 0.999634
     Offset: [24.6323, -23.4926, -13.5713]
     Center: [166.994, 166.994, 56]
     Translation: [26.0802, -24.4144, -15.1898]
     Inverse:
       0.999821 -0.0130334 0.0137326
```

```
      0.0133511 0.999639 -0.0233019
      -0.0134239 0.0234811 0.999634
    Singular: 0
    Euler's angles: AngleX=-0.023304 AngleY=-0.0137367 AngleZ=-0.0133551
    m_ComputeZYX = 0
  >>>>>>>>>
  Euler3DTransform (0x252c450)
    RTTI typeinfo:   itk::Euler3DTransform<double>
    Reference Count: 3
    Modified Time: 1829
    Debug: Off
    Object Name:
    Observers:
      none
    Matrix:
      1 0 0
      0 1 0
      0 0 1
    Offset: [-7.61852, -7.61852, -6]
    Center: [166.994, 166.994, 56]
    Translation: [-7.61852, -7.61852, -6]
    Inverse:
      1 0 0
      0 1 0
      0 0 1
    Singular: 0
    Euler's angles: AngleX=0 AngleY=0 AngleZ=0
    m_ComputeZYX = 0
  End of MultiTransform.
<<<<<<<<<<
  TransformsToOptimizeFlags, begin() to end():
     0 1
  TransformsToOptimize in queue, from begin to end:
  End of TransformsToOptimizeQueue.
<<<<<<<<<<
  End of CompositeTransform.
<<<<<<<<<<
```

```
In [8]: display_registration_results(fixed_image, moving_image, final_transform_v11)
        print('Final metric value: {0}'.format(registration_method.GetMetricValue()))
        print('Optimizer\'s stopping condition, {0}'.format(registration_method.GetOptimizerStopConditi
```



```
Final metric value: -0.6021162938438883
Optimizer's stopping condition, GradientDescentOptimizerv4Template: Maximum number of iterations (60) e
```

0.6 Saving your results

In the end, if we are satisfied with the results we would like to save them. This is all but trivial:

```
In [9]: # Resample the moving image onto the fixed image's grid.
        moving_resampled = sitk.Resample(moving_image, fixed_image, final_transform_v11, sitk.sitkLinea
                                         0.0, moving_image.GetPixelIDValue())
        sitk.WriteImage(moving_resampled, os.path.join(OUTPUT_DIR, 'moving_transformed.mha'))
        sitk.WriteTransform(final_transform_v11, os.path.join(OUTPUT_DIR, 'final_transform_v1.1.tfm'))
```

0.7 Going multi-resolution and using reference data

ITKv4 introduced an easy to use multi-resolution framework which we will explore in the following example. We
also incorporate the usage of reference data to evaluate the quality of registration.

What does the usage of reference data have to do with SimpleITK registration? Short answer - Nothing. Long answer - registration in general is dependent on many parameter settings. You will want to optimize these settings for your particular task. This requires that you evaluate results using reference data. Often this reference data comes in the form of corresponding points in the two images, and our quantitative evaluation of registration corresponds to the Target Registration Error (TRE).

In this notebook we are using the data from the Retrospective Image Registration Evaluation (RIRE) project.

0.7.1 Load our reference data

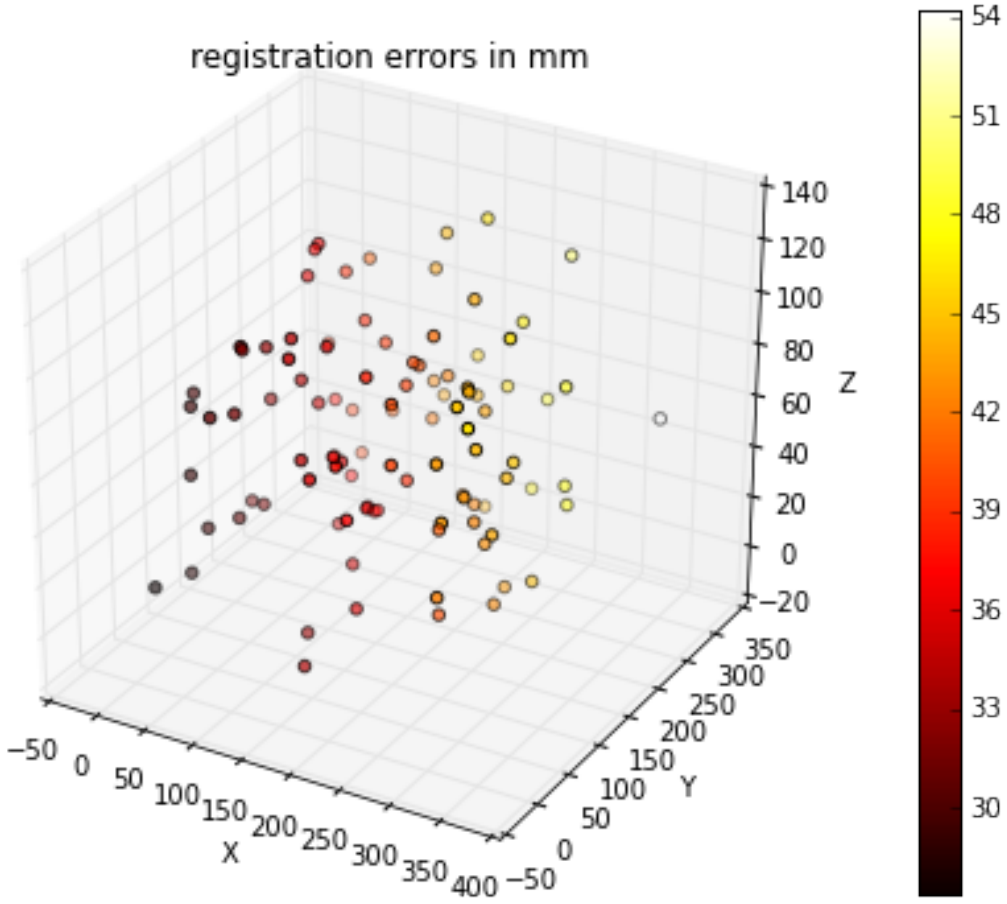
```
In [10]: fixed_fiducial_points, moving_fiducial_points = load_RIRE_ground_truth(fdata("RIRE/ct_T1.standa
```

```
        # Estimate the reference_transform defined by the RIRE fiducials
        R, t = absolute_orientation_m(fixed_fiducial_points, moving_fiducial_points)
        reference_transform = sitk.Euler3DTransform()
        reference_transform.SetMatrix(R.flatten())
        reference_transform.SetTranslation(t)

        # Generate a reference dataset from the reference transformation
        # (corresponding points in the fixed and moving images).
        fixed_points = generate_random_pointset(image=fixed_image, num_points=100)
        moving_points = [reference_transform.TransformPoint(p) for p in fixed_points]

        # Compute the TRE prior to registration.
        pre_errors_mean, pre_errors_std, _, pre_errors_max, pre_errors = registration_errors(sitk.Eule
        print('Before registration, errors (TRE) in millimeters, mean(std): {:.2f}({:.2f}), max: {:.2f
```

```
Fetching RIRE/ct_T1.standard
Before registration, errors (TRE) in millimeters, mean(std): 43.80(5.50), max: 56.94
```

0.7.2 Evaluate the registration after initial alignment

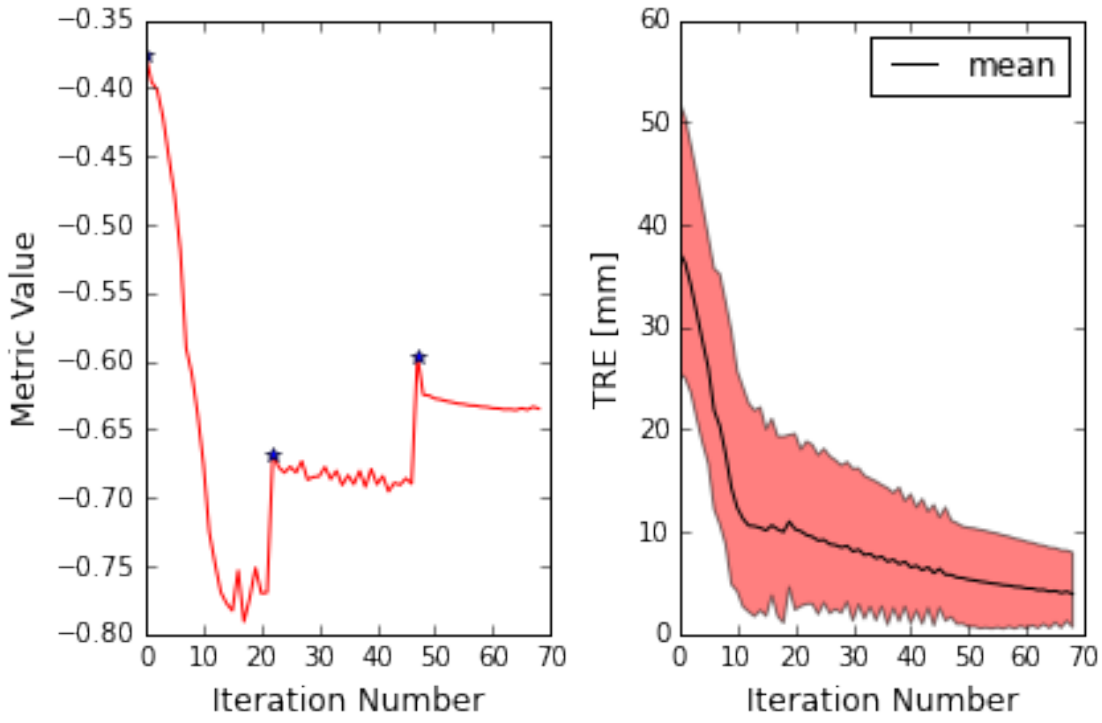
```
In [11]: # The initial transform was estimated at the begining of this notebook.
         initial_errors_mean, initial_errors_std, _, initial_errors_max, initial_errors = registration_
         print('After initial alignment, errors (TRE) in millimeters, mean(std): {:.2f}({:.2f}), max: {
```



```
After initial alignment, errors (TRE) in millimeters, mean(std): 39.49(5.50), max: 54.20
```

0.7.3 Version 2 - v4 multi-resolution approach

In [12]:
```
registration_method = sitk.ImageRegistrationMethod()

registration_method.SetMetricAsMattesMutualInformation(numberOfHistogramBins=50)
registration_method.SetMetricSamplingStrategy(registration_method.RANDOM)
registration_method.SetMetricSamplingPercentage(0.01)
registration_method.SetInterpolator(sitk.sitkLinear)
registration_method.SetOptimizerAsGradientDescent(learningRate=1.0, numberOfIterations=60)
registration_method.SetOptimizerScalesFromPhysicalShift()

# Make a copy of the initial transformation so that we optimize in place but can still reuse i
# specific to our notebook's workflow. In a regular regisration setting we would not create th
# the variable as global so that it is accessible outside this cell even when using timeit.
global final_transform_multi_res
final_transform_multi_res = sitk.Euler3DTransform(initial_transform)
registration_method.SetInitialTransform(final_transform_multi_res)
registration_method.SmoothingSigmasAreSpecifiedInPhysicalUnitsOn()

# Multi-resolution framework - it's this easy:
registration_method.SetShrinkFactorsPerLevel(shrinkFactors = [4,2,1])
registration_method.SetSmoothingSigmasPerLevel(smoothingSigmas = [2,1,0])

registration_method.AddCommand(sitk.sitkStartEvent, metric_and_reference_start_plot)
registration_method.AddCommand(sitk.sitkEndEvent, metric_and_reference_end_plot)
registration_method.AddCommand(sitk.sitkIterationEvent,
                               lambda: metric_and_reference_plot_values(registration_method, f
# Additional callback specific to the multi-resolution framework
registration_method.AddCommand(sitk.sitkMultiResolutionIterationEvent,
                               metric_update_multires_iterations)

registration_method.Execute(sitk.Cast(fixed_image, sitk.sitkFloat32),
                            sitk.Cast(moving_image, sitk.sitkFloat32))
print('Final metric value: {0}'.format(registration_method.GetMetricValue()))
print('Optimizer\'s stopping condition, {0}\n'.format(registration_method.GetOptimizerStopCondi
print(final_transform_multi_res)
```

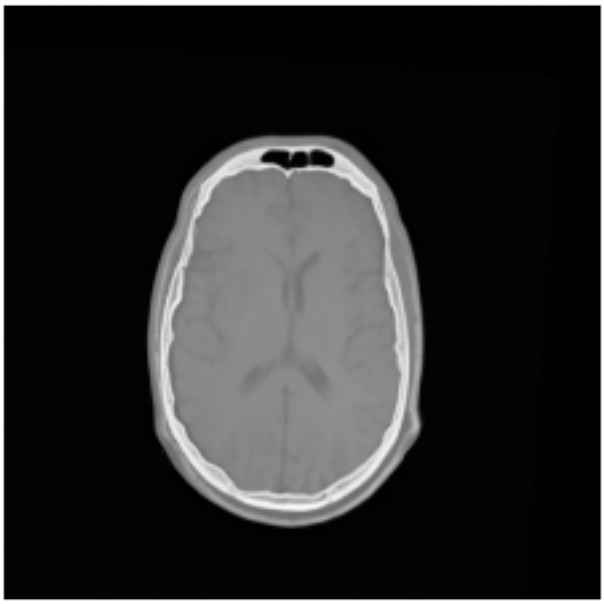



```
Final metric value: -0.6320637464770482
Optimizer's stopping condition, GradientDescentOptimizerv4Template: Convergence checker passed at itera

itk::simple::Euler3DTransform
 Euler3DTransform (0x2636590)
   RTTI typeinfo:   itk::Euler3DTransform<double>
   Reference Count: 2
   Modified Time: 1015856
   Debug: Off
   Object Name:
   Observers:
     none
   Matrix:
     0.997667 0.0679171 0.00685529
     -0.0679551 0.997673 0.00546584
     -0.00646812 -0.00591894 0.999962
   Offset: [5.87315, -21.5367, -19.5621]
   Center: [166.994, 166.994, 56]
   Translation: [17.2092, -32.9672, -21.6328]
   Inverse:
     0.997667 -0.0679551 -0.00646812
     0.0679171 0.997673 -0.00591894
     0.00685529 0.00546584 0.999962
   Singular: 0
   Euler's angles: AngleX=-0.00591898 AngleY=0.00646828 AngleZ=-0.0679706
   m_ComputeZYX = 0
```

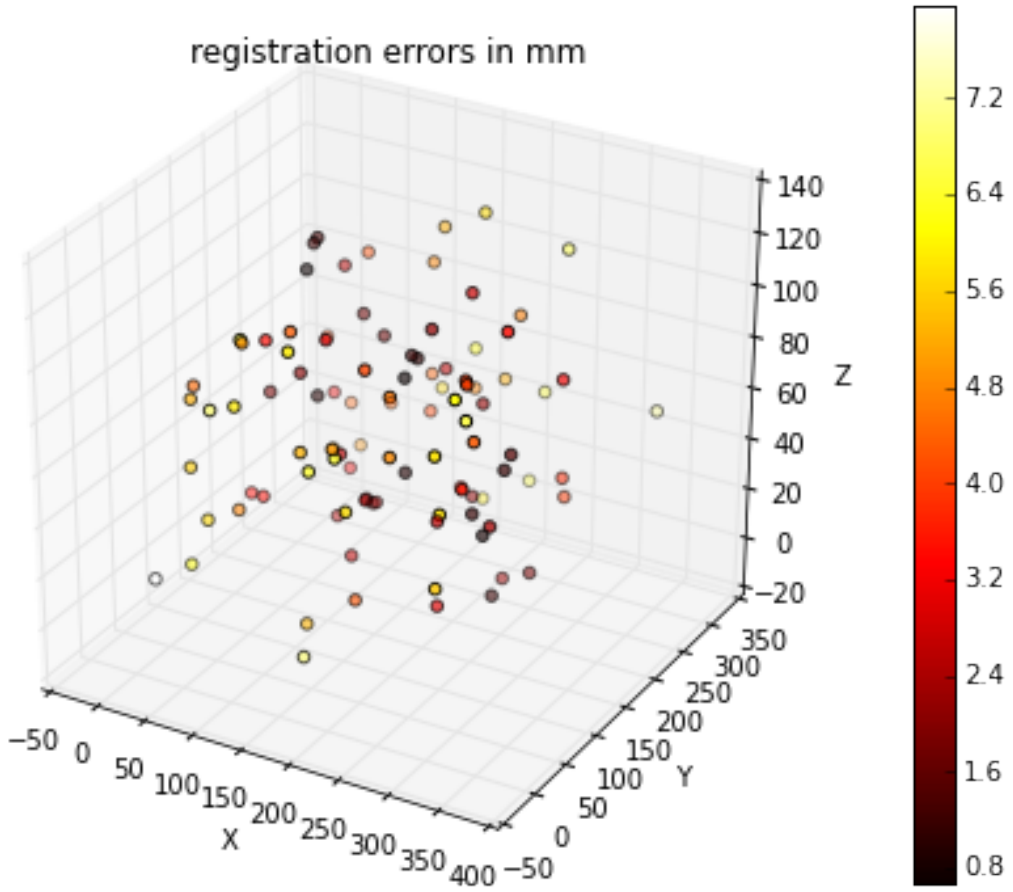
0.7.4 Evaluate the registration after final alignment

Qualitative evaluation via visual inspection and quantitative evaluation via TRE.

```
In [13]: display_registration_results(fixed_image, moving_image, final_transform_multi_res)
```



```
In [14]: final_errors_mean, final_errors_std, _, final_errors_max, final_errors = registration_errors(f
         print('After final alignment, errors (TRE) in millimeters, mean(std): {:.2f}({:.2f}), max: {:.
```




```
After final alignment, errors (TRE) in millimeters, mean(std): 3.92(1.85), max: 7.97
```

0.8 Exercises

In this section you will explore the effects of various settings on the registration framework:

Modify the initial alignment estimation to use sitk.CenteredTransformInitializerFilter.MOMENTS - does this have an effect on our registration?

Modify version 1 above so that the transformation is modified in place (SetInitialTransform) - does this have any effect on the result (hint: transformation type)

Modify version 1 or 1.1, exploring the effects of various component settings on the results:

Scale estimation: Comment out the SetOptimizerScalesFromPhysicalShift method, try the SetOptimizerScalesFromIndexShift,SetOptimizerScalesFromJacobian and SetOptimizerScales methods.

Modify the number of iterations - we already know the optimizer is likely terminating earlier than it should.

Modify the sampling strategy (SetMetricSamplingStrategy) - try no/regular sampling (registration_method.NONE, REGULAR).

Modify version 2, exploring the effects of changing the smoothingSigmas and shrinkFactors on accuracy. You can also modify the units of the smoothingSigmas from the default physical ones to voxel units (SmoothingSigmasAreSpecifiedInPhysicalUnitsOff).

Compare the runtime of the single resolution registration, version 1, to that of the multi-resolution registration, version 2. Add the following line at the begining of the registration cells:%%timeit -r1 -n1 This will wrap the cell as a function and run it once, reporting the runtime.

In registration version 2 the number of samples per resolution was set at 1%, SetMetricSamplingPercentage(0.01), this ignores the fact that there are far fewer voxels at the higher resolutions. Use the method SetMetricSamplingPercentagePerLevel with a list of percentages corresponding to the number of resolutions.

In [ ]:

Create Your Own ITK Module

ITK in Biomedical Research and Commercial Applications

SPIE Medical Imaging, 2016

Matthew McCormick, PhD., Kitware, Inc.

Module Definitions

ITK Modules

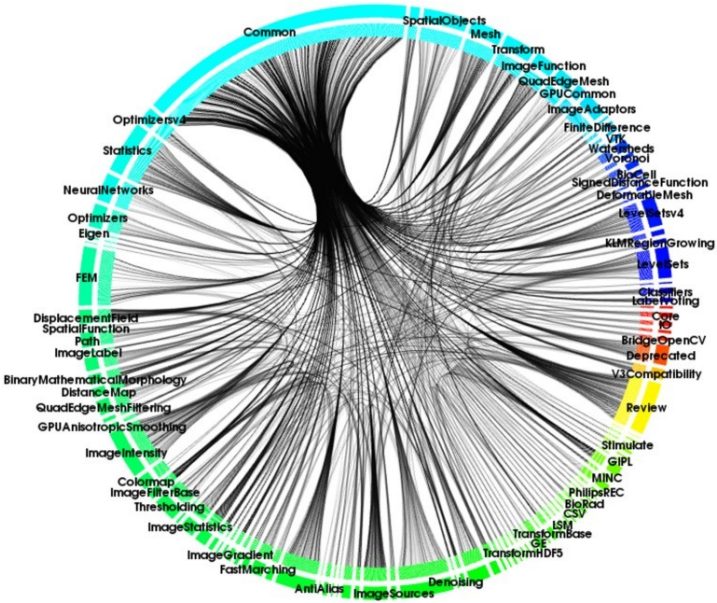
Since its inception in 2000, ITK was designed as a collection of about seven core libraries and about ten third party libraries. This monolithic organization of the code led over time to very large sub-libraries, as more classes were added to the toolkit. Once the core code of ITK surpassed **half-a-million lines of code**, it became evident that a more **modular approach was needed** in order to support the future continued growth of the toolkit.

Modularization

As a result of modularization in ITKv4, the initial monolithic code base of about 12,000 files was partitioned into **more than 100 modules**.

Module dependencies are identified and explicitly declared in the CMake configuration system. ITK adopters can select, at configuration time, the pieces of ITK that they wanted to use in their own projects.

A selected module automatically enables its required dependencies.



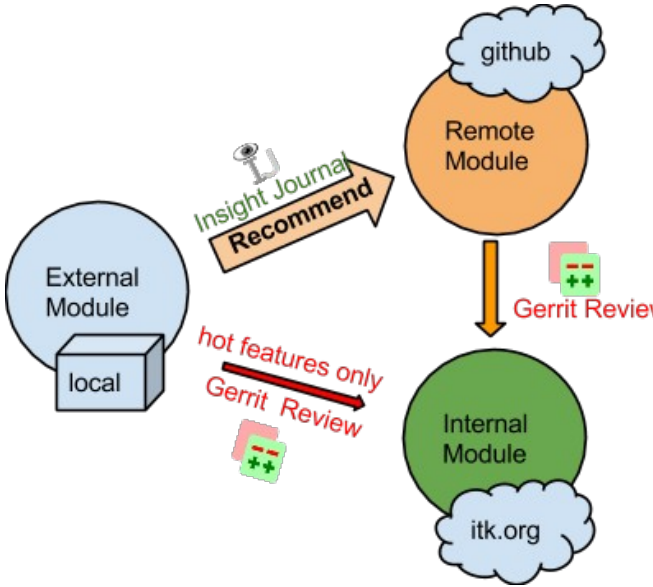
Types of Modules

Default Internal - modules in the ITK source tree that are enabled in the default build configuration.

Non-default Internal - are in the ITK source tree, but they are not enabled by default because of missing third-party dependencies, etc.

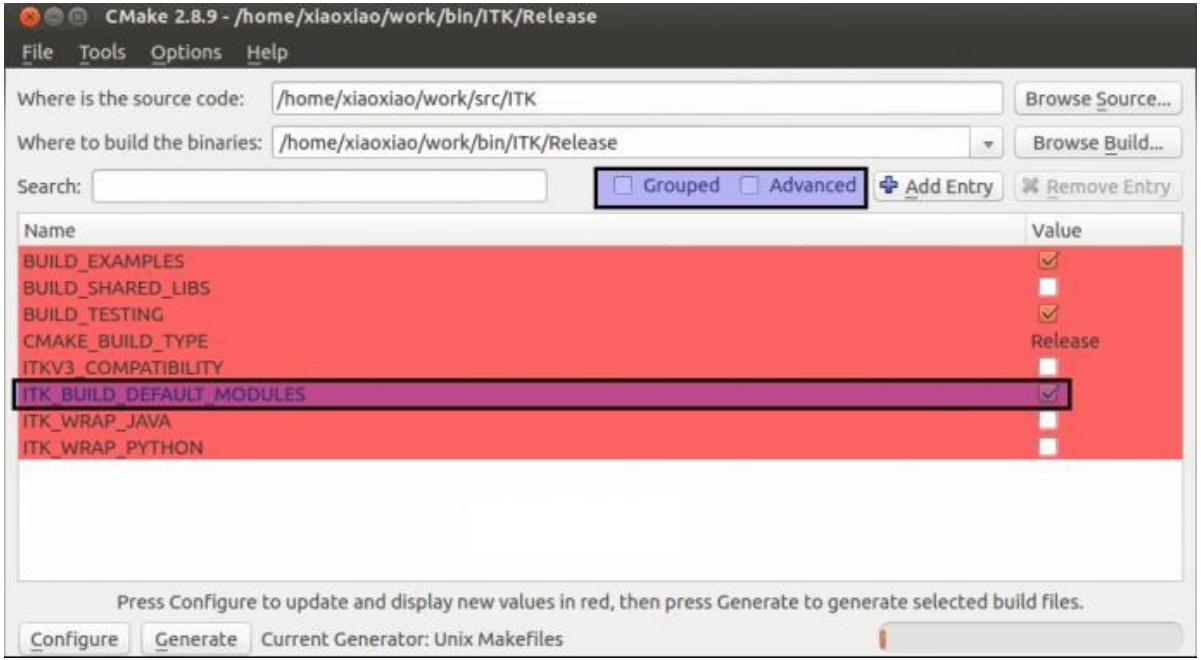
External - modules developed outside of the ITK source tree.

Remote - externally developed modules made available in ITK's CMake configuration. When enabled, the module's Git repository is dynamically fetched for the build.



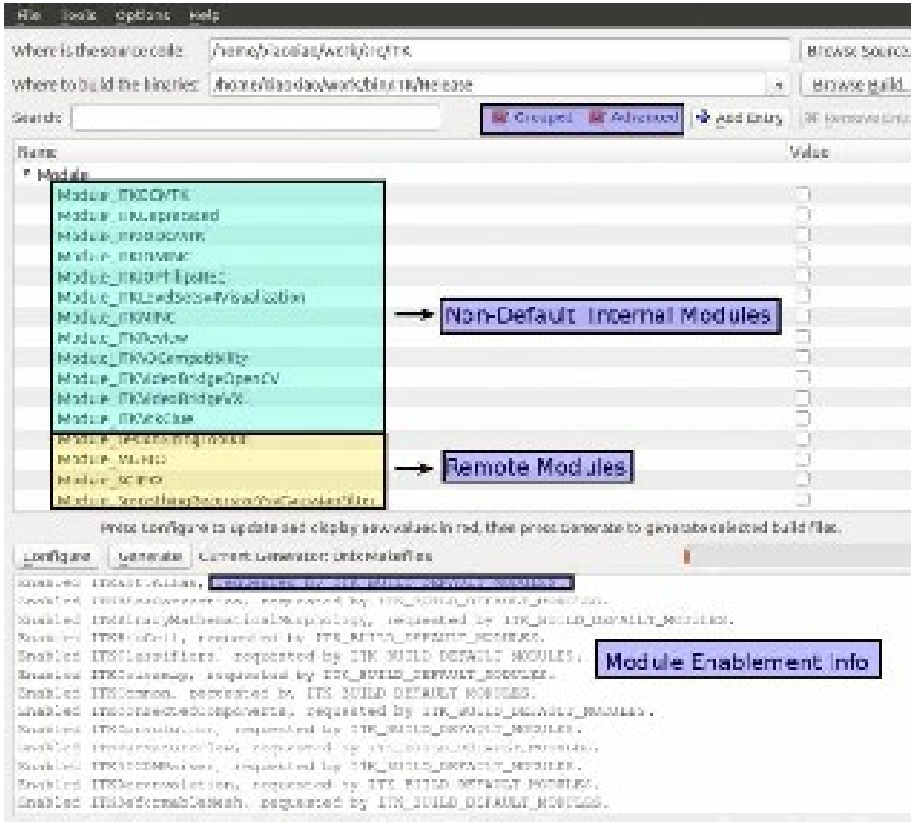
Using Modules

Default Modules in ITK's CMake Configuration



ITK_BUILD_DEFAULT_MODULES is the CMake option to request that all default modules be built, and by default this option is **ON.**

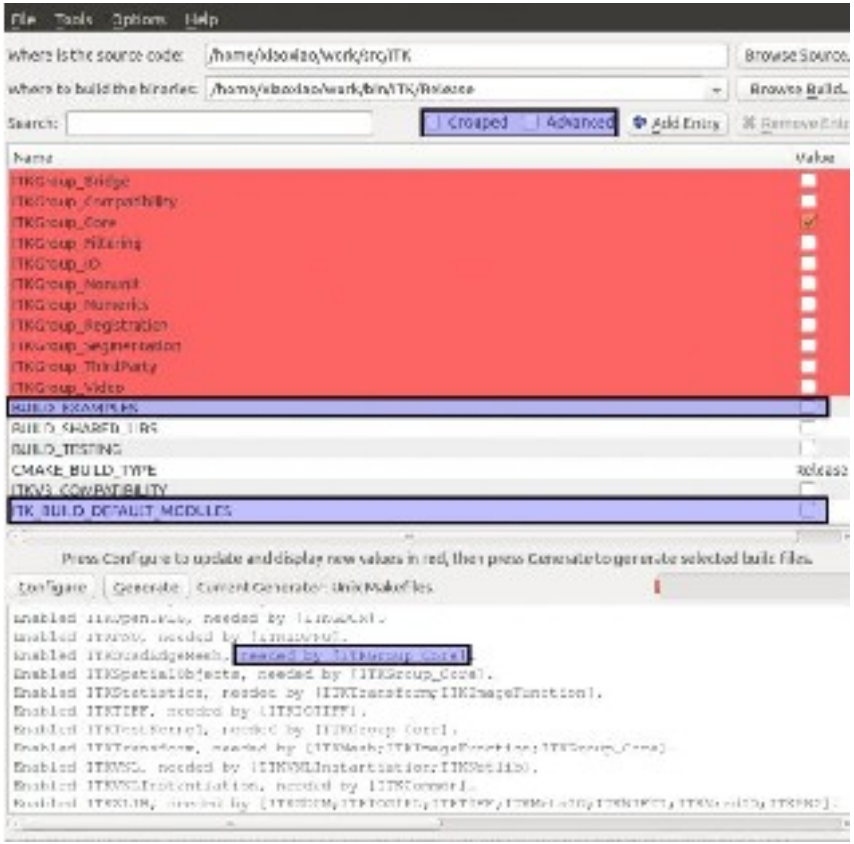
Enabling Non-Default Modules



In the advanced mode of the CMake GUI, you can manually toggle the non-default modules via the **Module_{module name}** options.

Internal modules have the **Module_ITK** prefix, while External and Remote modules do not.

Group Module Enablement



ITK is organized so that groups of modules that have close relationships or similar functionalities reside in the same directory. Presently, there are 11 groups (not including the Remote group). The CMake **ITKGroup_{group name}** option is created for the convenience of enabling or disabling multiple modules at once.

Module Selection for Projects that Use ITK

```
find_package(ITK COMPONENTS ITKFoo ITKBar <More ITK Modules>  REQUIRED)
include (${USE_ITK_FILE})
```

```
ExternalProject_Add( ITK
      GIT_REPOSITORY "git://itk.org/ITK.git"
      GIT_TAG      "<tag id>"   # specify the commit id or the tag id
      SOURCE_DIR    <ITK source tree path>
      BINARY_DIR    <ITK build tree path>
      CMAKE_GENERATOR ${gen}
      CMAKE_ARGS
            ${ep_common_args}
            -DBUILD_SHARED_LIBS:BOOL=OFF
                -DBUILD_EXAMPLES:BOOL=OFF
                -DBUILD_TESTING:BOOL=OFF
            -DITK_BUILD_DEFAULT_MODULES:BOOL=ON
            -DModule_ITKReview:BOOL=ON
            [-DModule_LevelSetv4Visualization:BOOL=ON]
       INSTALL_COMMAND ""
       DEPENDS [VTK]  [DCMTK]  # if some of the modules requested require extra
 )
```

For a project that uses ITK as an external library, it is recommended to specify the individual desired ITK modules in the **COMPONENTS** argument of the **find_package** CMake command.

The script **ITK/Utilities/Maintenance/WhatModulesITK.py** can inspect a project and determine which modules it requires.

Modules in Doxygen

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

Modules in Doxygen (cont.)

- ▶ Group IO
- ▼ Group Filtering
  - Module ITKDiffusionTensorImage
  - Module ITKFastMarching
  - Module ITKConvolution
  - Module ITKImageGradient
  - Module ITKImageFusion
  - Module ITKFFT
  - Module ITKDenoising
  - Module ITKGPUThresholding
  - Module ITKBinaryMathematicalMorphology
  - Module ITKImageLabel

Modules in Doxygen (cont.)

ITK 4.10.0
Insight Segmentation and Registration Toolkit

Main Page | Related Pages | Modules | Namespaces | Classes | Files | Examples | Search

Classes

Module ITKConvolution

Group Filtering

Collaboration diagram for Module ITKConvolution:

Classes

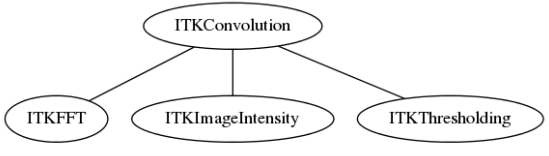
class	itk::ConvolutionImageFilter< TInputImage, TKernelImage, TOutputImage >
class	itk::ConvolutionImageFilterBase< TInputImage, TKernelImage, TOutputImage >
class	itk::FFTConvolutionImageFilter< TInputImage, TKernelImage, TOutputImage, TInternalPrecision >
class	itk::FFTNormalizedCorrelationImageFilter< TInputImage, TOutputImage >
class	itk::MaskedFFTNormalizedCorrelationImageFilter< TInputImage, TOutputImage, TMaskImage >
class	itk::NormalizedCorrelationImageFilter< TInputImage, TMaskImage, TOutputImage, TOperatorValueType >

Detailed Description

This module contains filters that convolve an image with a kernel. Convolution is a fundamental operation in many image analysis algorithms.

Dependencies:

- Module ITKFFT
- Module ITKImageIntensity
- Module ITKThresholding



Tarballs of the nightly generated Doxygen documentation are available for the html, xml, and tag file.

Generated on Fri Feb 26 2016 04:39:25 for ITK by doxygen 1.8.5

External Module Development Process

Building External Modules Against an ITK Build Tree

Since ITK 4.9.0, it is possible to build External modules against an existing ITK build tree.

- Convenient for revision control
- Faster CMake configuration times
- Facilitates CI testing
- Easily add Python wrapping to high performance C++ code
- Caveats:
- Only works with a build tree; cannot build against an install tree
- All module dependencies must already be built

External Modules Differences

1. Pass the ITK build directory in the **ITK_DIR** CMake configuration variable when building an External module

2. The module's top level CMakeLists.txt file should be

```
cmake_minimum_required(VERSION 2.8.9)
project(ModuleName)

if(NOT ITK_SOURCE_DIR)
 find_package(ITK REQUIRED)
 list(APPEND CMAKE_MODULE_PATH ${ITK_CMAKE_DIR})
 include(ITKModuleExternal)
else()
 itk_module_impl()
endif()
```

Instead of:

```
project(ModuleName)
itk_module_impl()
```

External Modules Differences (cont.)

3. When a project is using a version of ITK that has been extended with an External module, it should explicitly specify the module dependency in its CMake **find_package COMPONENT’**s calls. For example:

```
cmake_minimum_required(VERSION 3.0)

project(MyProject)

find_package(ITK REQUIRED
        COMPONENTS
                ITKImageFeature
                ITKFastMarching
                MyExternalModule
        )
include(${ITK_USE_FILE})
```

Adding CI Testing to GitHub Repositories

It is possible to add continuous integration testing to an External module GitHub repository.

Services that utilize **Docker** are recommended to reduce software dependency acquisition and build times, like CircleCI or TravisCI.

Adding CI Testing to GitHub Repositories (cont.)

The **ITKMinimalPathExtraction** GitHub repository can be used as an example


README.rst

ITKMinimalPathExtraction

PASSED

This is an ITK module that implements a minimal path extraction framework based on Fast Marching arrival functions.

A more detailed description can be found in the Insight Journal article:

https://blog.kitware.comzautomated-tests-on-github-for-your-itk-dependent-project-with-circleci-and-docker/

https://github.com/InsightSoftwareConsortium/ITKMinimalPathExtraction

Adding CI Testing to GitHub Repositories (cont.)

Configure the tools involved:

Docker: install CMake, GCC, and optionally CMake into the Docker Image. Configure and build ITK with the required module dependencies enabled.

CircleCI: configure CircleCI in *circle.yml* to download the Docker build dependency image, and build and test the source repository in that image. Add a status badge to your README.

CDash: Add a *CTestConfig.cmake* file so a visuaization of your pull request build is available on the ITK dashboard.

Writing a Module

Module Organization

The **top level** directory

The **include/** directory

The **src/** directory

The **test/** directory

The **wrapping/** directory

The itk-module.cmake File (cont.)

1. The module name.
2. Dependencies on other modules
3. Module properties
4. A description of the module

The itk-module.cmake File

```
set(DOCUMENTATION "This module contains the central classes of the ITK
toolkit.  They include, basic data structures \(such as Points, Vectors,
Images, Regions\) the core of the process objects \(such as base
classes for image filters\) the pipeline infrastructure classes, the support
for multi-threading, and a collection of classes that isolate ITK from
platform specific features. It is anticipated that most other ITK modules will
depend on this one.")

itk_module(ITKCommon
  ENABLE_SHARED
  PRIVATE_DEPENDS
    ITKDoubleConversion
  COMPILE_DEPENDS
    ITKKWSys
    ITKVNLInstantiation
  TEST_DEPENDS
    ITKTestKernel
    ITKMesh
    ITKImageIntensity
    ITKIOImageBase
  DESCRIPTION
    "${DOCUMENTATION}"
)
```

The itk-module.cmake File (cont.)

DEPENDS: Modules that will be publically linked to this module. The header's used are added to *include/*.{h,hxx}* files.

PRIVATE_DEPENDS: Modules that will be privately linked to this module. The header's used are only added to *src/*.cxx* files.

COMPILE_DEPENDS: Modules that are needed at compile time by this module. The header's used are added to *include/*{h,hxx}* files sbut there is not a library to link against.

TEST_DEPENDS: Modules that are needed by this modules testing executables

Headers: the include/ Directory

Simply add all **.h* declaration headers and **.hxx* template definition headers to the *include/* directory.

Libraries: the src/ Directory

```
set(AModuleName_SRCS
  itkFooClass.cxx
  itkBarClass.cxx
  )

add_library(AModuleName ${AModuleName_SRCS})
itk_module_link_dependencies()
itk_module_target(AModuleName)
```

Tests: the test/ Directory

```
itk_module_test()

set(ModuleTemplateTests
  itkMinimalStandardRandomVariateGeneratorTest.cxx
  itkLogNormalDistributionImageSourceTest.cxx
  )

CreateTestDriver(ModuleTemplate "${ModuleTemplate-
Test_LIBRARIES}" "${ModuleTemplateTests}")

itk_add_test(NAME itkMinimalStandardRandomVariateGeneratorTest
  COMMAND ModuleTemplateTestDriver itkMinimalStandardRandomVariateGeneratorTest
  )

itk_add_test(NAME itkLogNormalDistributionImageSourceTest
  COMMAND ModuleTemplateTestDriver --without-threads
  --compare
    ${ITK_TEST_OUTPUT_DIR}/itkLogNormalDistributionImageSourceTestOutput.mha
    DATA{Baseline/itkLogNormalDistributionImageSourceTestOutput.mha}
  itkLogNormalDistributionImageSourceTest
    ${ITK_TEST_OUTPUT_DIR}/itkLogNormalDistributionImageSourceTestOutput.mha
```

Wrapping: the wrapping/ Directory

CMakeLists.txt:

```
itk_wrap_module(ITKImageFilterBase)

set(WRAPPER_SUBMODULE_ORDER
  itkRecursiveSeparableImageFilter
  itkFlatStructuringElement
  itkKernelImageFilter
  itkMovingHistogramImageFilterBase
)
itk_auto_load_submodules()
itk_end_wrap_module()
```

itkImportImageFilter.wrap:

```
itk_wrap_class("itk::ImportImageFilter" POINTER)

  foreach(d ${ITK_WRAP_IMAGE_DIMS})
    foreach(t ${WRAP_ITK_SCALAR})
      itk_wrap_template("${ITKM_${t}}${d}" "${ITKT_${t}},${d}")
    endforeach()
  endforeach()

itk_end_wrap_class()
```

Remote Modules

http://www.itk.org/Wiki/ITK/Policy_and_Procedures_for_Adding_Remote_Modules

1. Publish an open access article in the Insight Journal describing the module
2. Submit a Gerrit patch that adds a file named *Modules/Remote/<module name>.remote.cmake*. This file must have:
    a. Maintainer name and email in the comments.
    b. A call to the *itk_fetch_module* CMake function whose arguments are:
        i. The name of the remote module: **Note that in each <remote module name>.remote.cmake, the first argument of the function itk_fetch_module() is the name of the remote module, and it has to be consistent with the module name defined in the corresponding itk-module.cmake. To better distinguish the remote modules from the internal ITK modules, the names of the remote modules should NOT contain the "ITK" string prefix.**

Remote Modules (cont.)

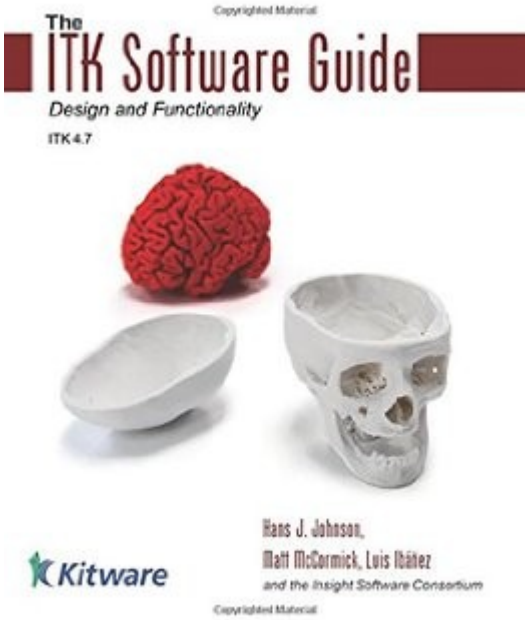
Benefits:

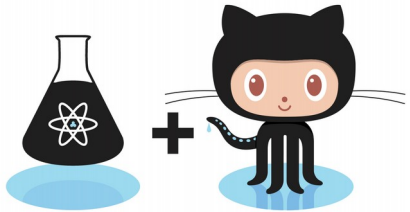
1. Your work is exposed to a community of thousands of research software engineers around the world.
    a. The module is listed in ITK's CMake configuration variables.
    b. The classes are indexed in ITK's Doxygen documentation.
2. It becomes easier to develop a second module that depends on the first module.

For more information...

See the **ITK Software Guide:**

- Chapter 2, Configuring and Building ITK
  - Section 2.1.3 Advanced Module Configuration
- Chapter 3, How to Create a Module
- http://itk.org/ITK/help/documentation.html
- Examples:
- In the ITK source tree
- On the InsightSoftwareConsortium GitHub organization





Integrating your developments with other projects

Hans J. Johnson
University of Iowa

Sonia Pujol, Ph.D.
Surgical Planning Laboratory
Harvard Medical School

CXX Executable

- Objectives:
  - Implement you medical imaging algorithm using the ITK C++ libraries (You need the speed and extended capabilities of coding your own solution)
  - Incoroporate your tool into a larger processing and/or visualization environment

Sorry... Not a hands-on

- The challenge of configuring each of your laptops compilers, IDE, CMake, Editor etc is too onerous for a 45 minutes timeslot
- I will provide live demonstrations
- Feel free to do the demo's yourself if you have Cmake and compilers setup.

What do I need?

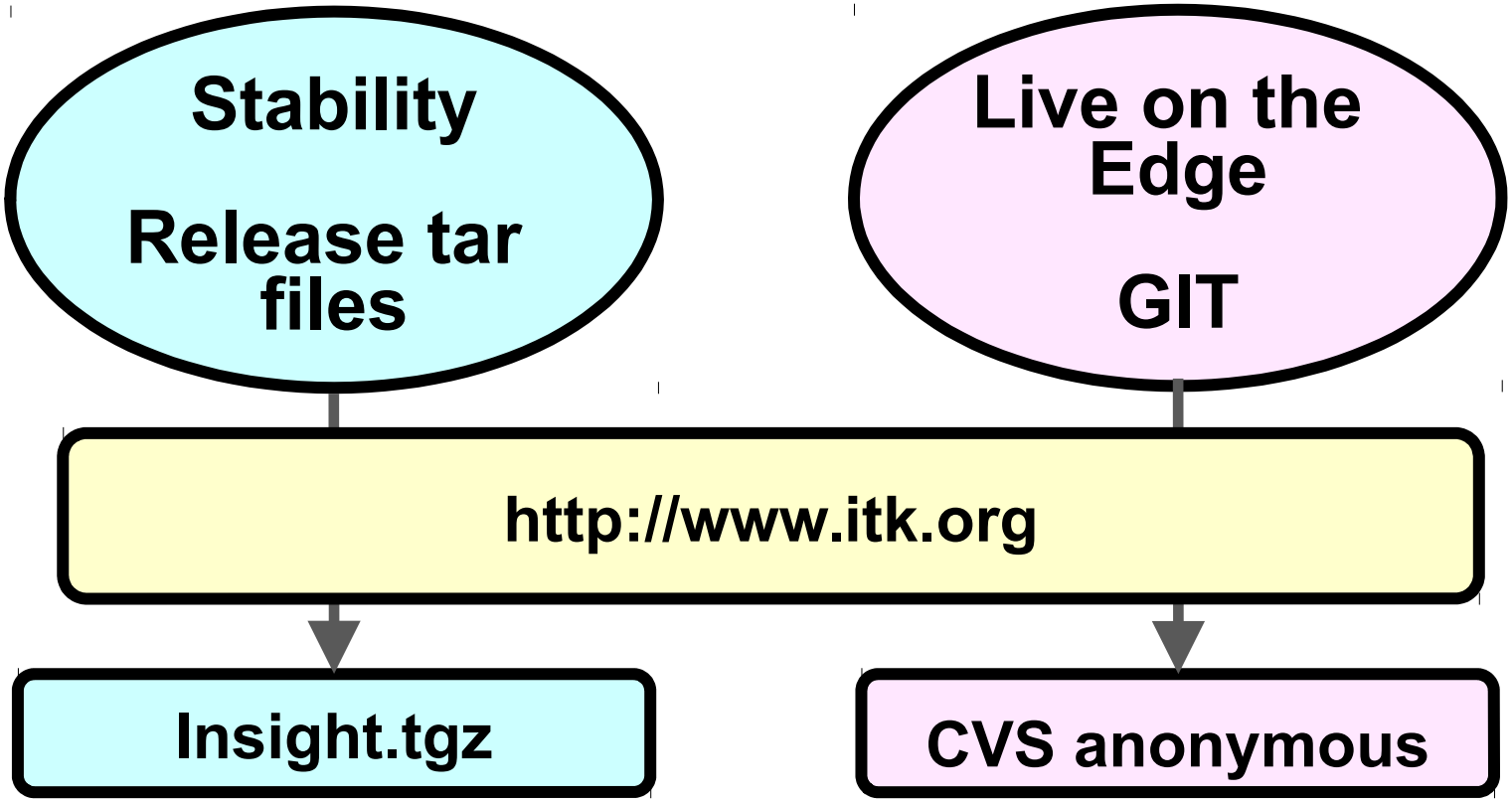
C++ Compiler

GCC 4.2+
Visual C++ 9.0+
Intel v 11
Clang
Etc..

CMake

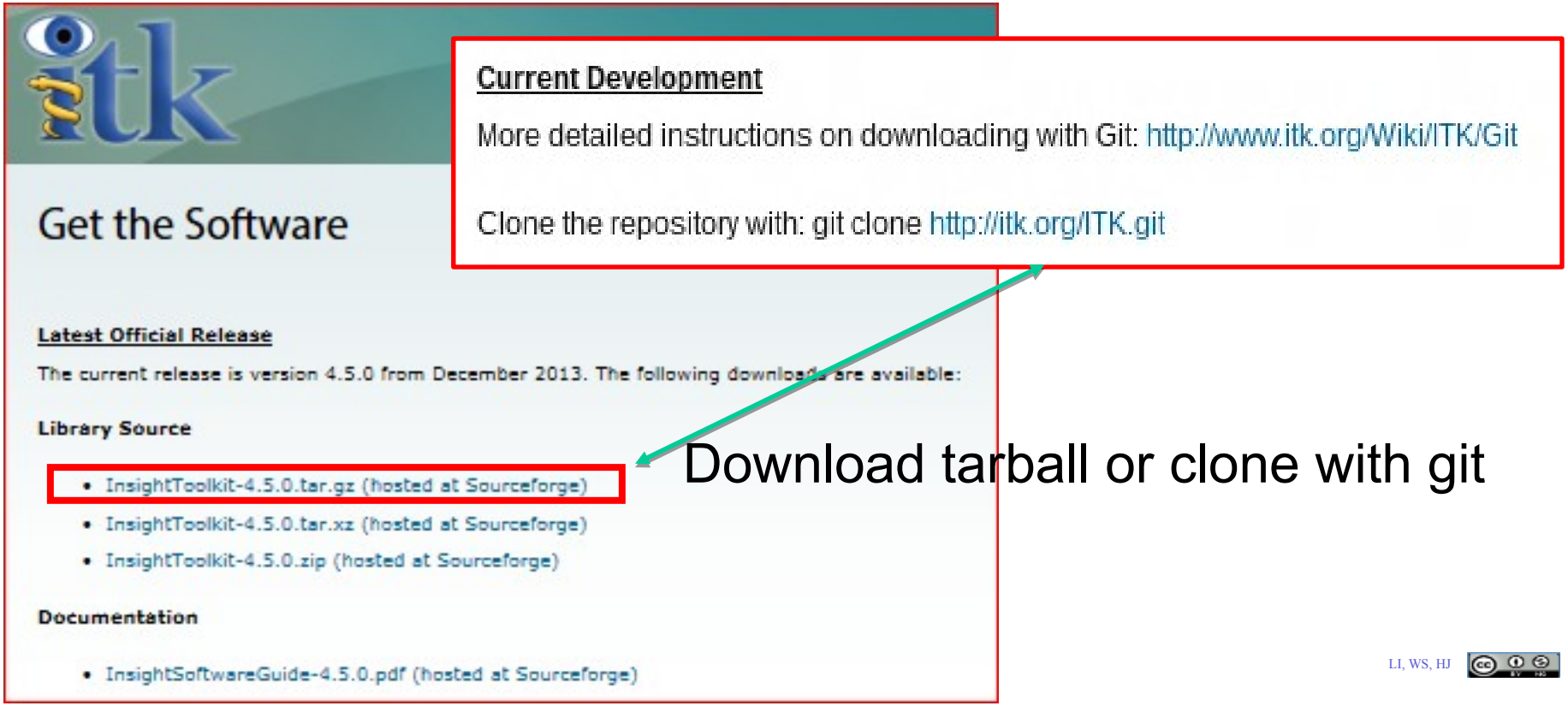
www.cmake.org

Step 1. Download ITK



Download ITK from the Web

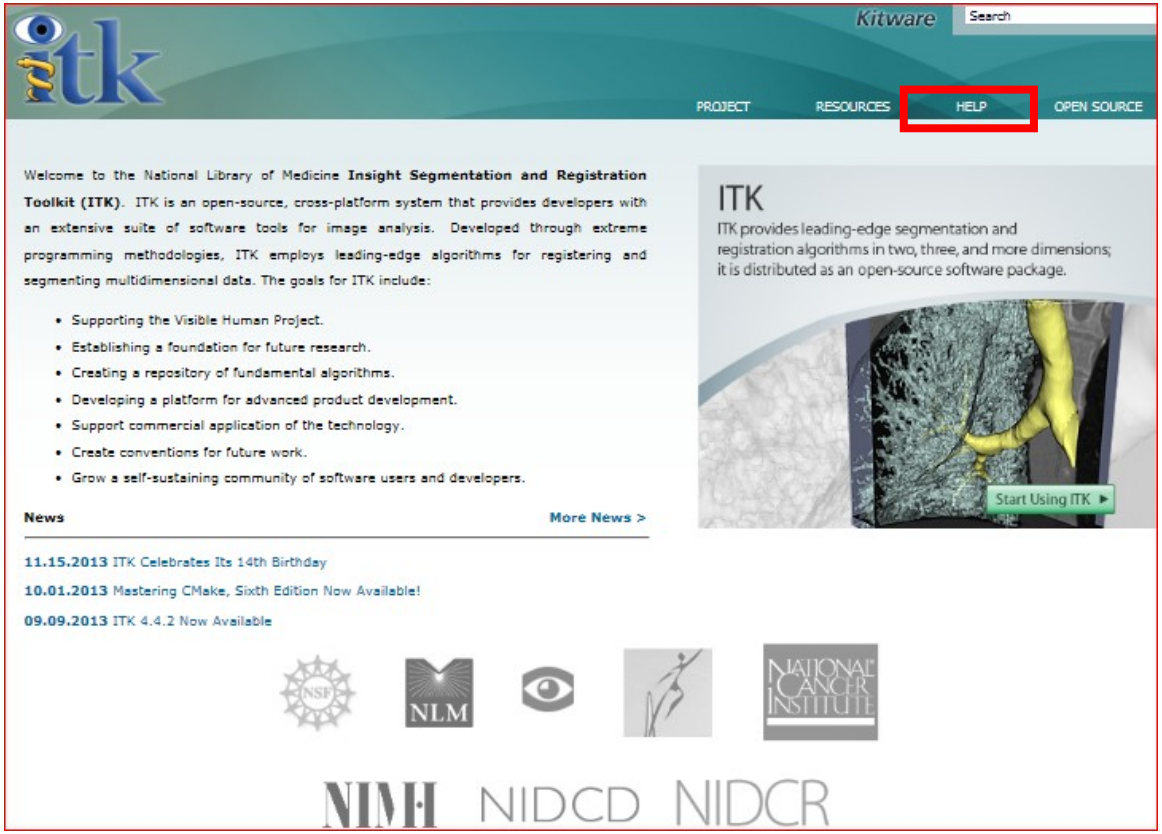
http://www.itk.org



Download tarball or clone with git

Download ITK from the Web

http://www.itk.org



Cross platform (C++) development

- ITK builds on a large combination of operating systems and platforms
- Each C++ programming environment has it's own input format: Makefiles, workspaces, etc.

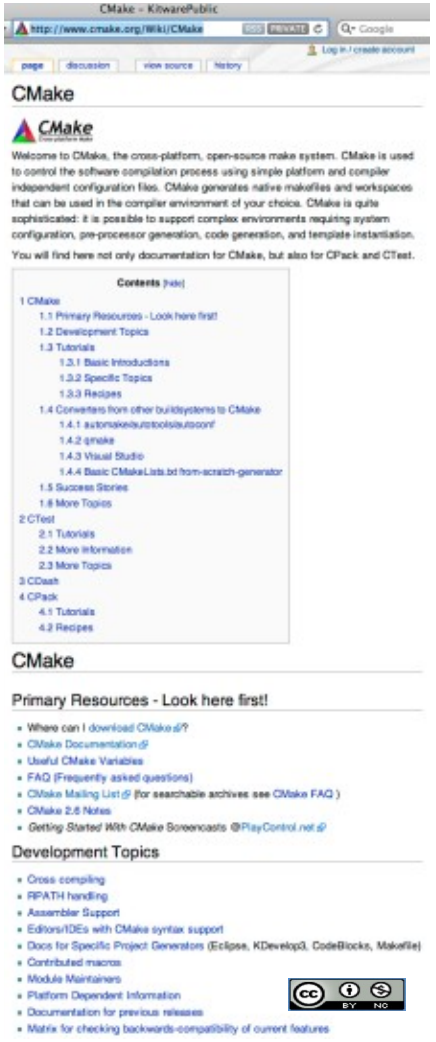
- Q: How can you possibly support all of these diverse build environments/platforms?

JG

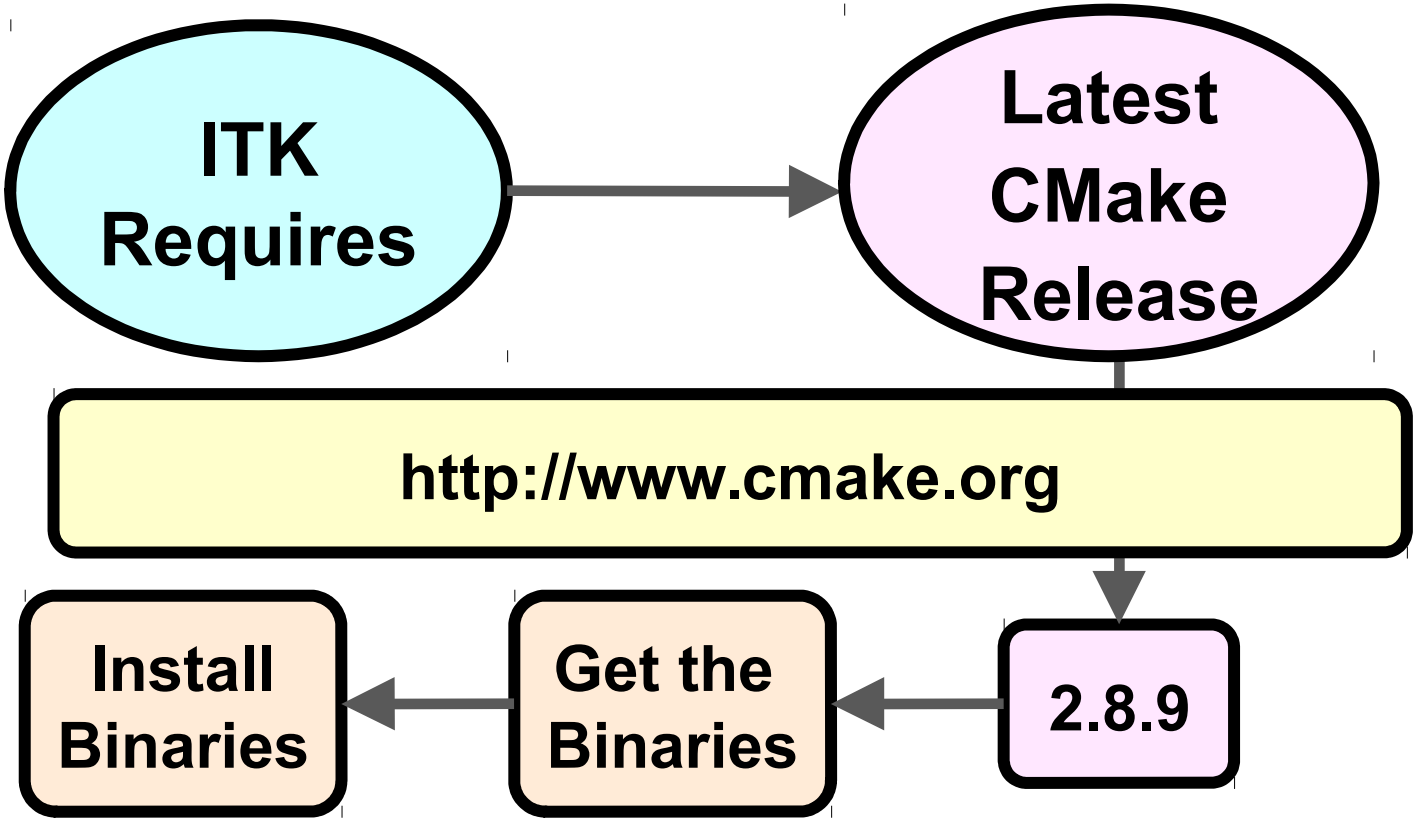
The answer: CMake

- Cross platform tool to manage the build process
- Simplifies the build process
- Auto-configuration
- Easy access to external libraries
- Used by several other open source projects

www.cmake.org



Step 2. Download CMake



Step 1 - Install CMake

- Check if a recent version of CMake is already installed on your computer.
- If not, …
- Download and install a binary distribution of CMake 2.8.9 from:
  - http://www.cmake.org/

JG

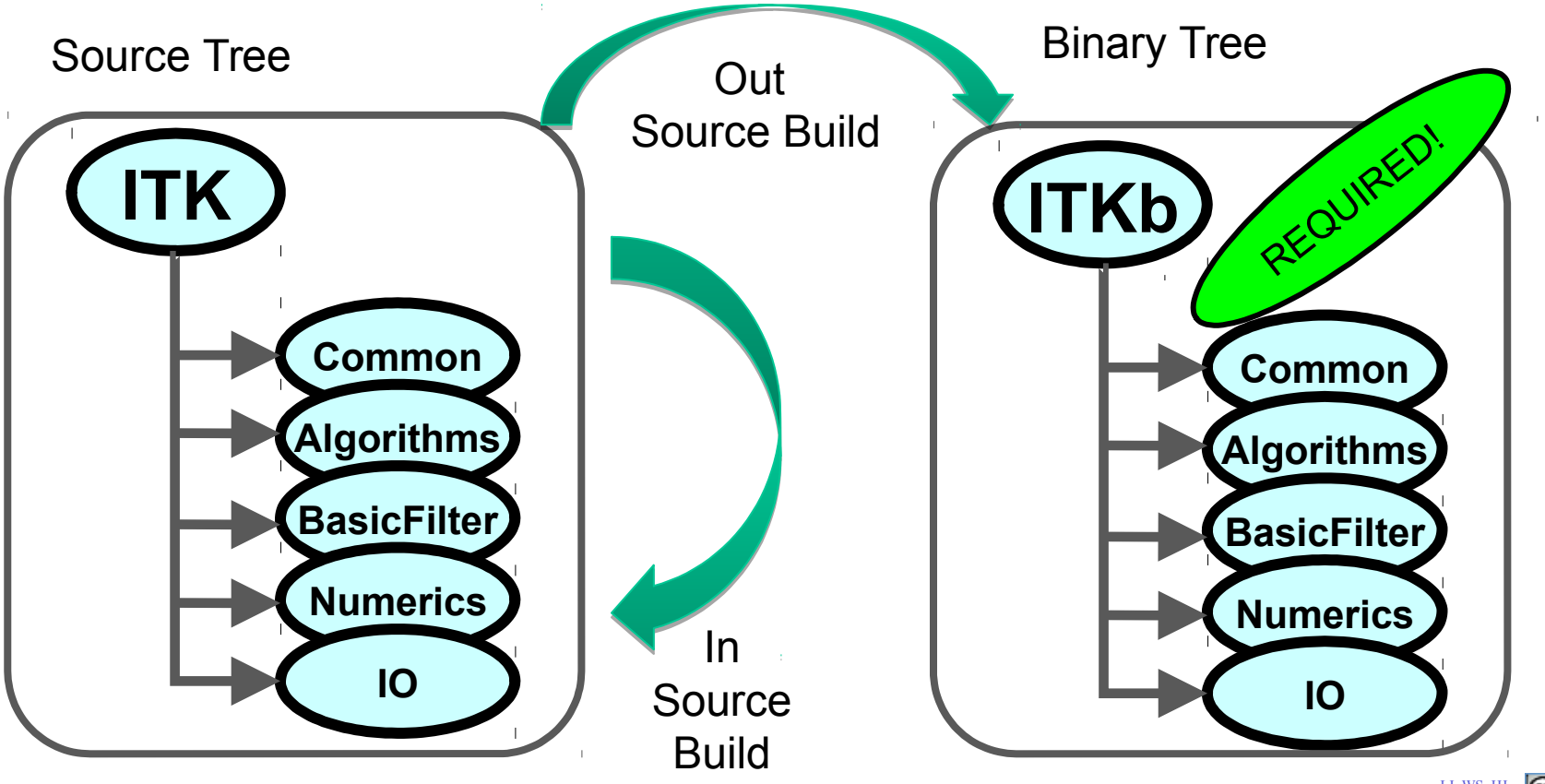
CMake is:

- Required to build native (C++) ITK
- Cross-platform project generator
- Often simpler than particular environments
- Text as input
- Project file as output:

Windows	Visual Studio Solution
UNIX	Makefile
Mac OS X	Xcode project or Makefile

JG

Step 3. Configure ITK



LI, WS, HJ

Why use two trees?

- Keeps your C++ source and binary code separate
- Minimizes the amount of damage you can do to your GIT repository tree
- ITK is the SOURCE_DIR folder
- I suggest that you create your BINARY_DIR folder as ITK-bld adjacent to the SOURCE_DIR

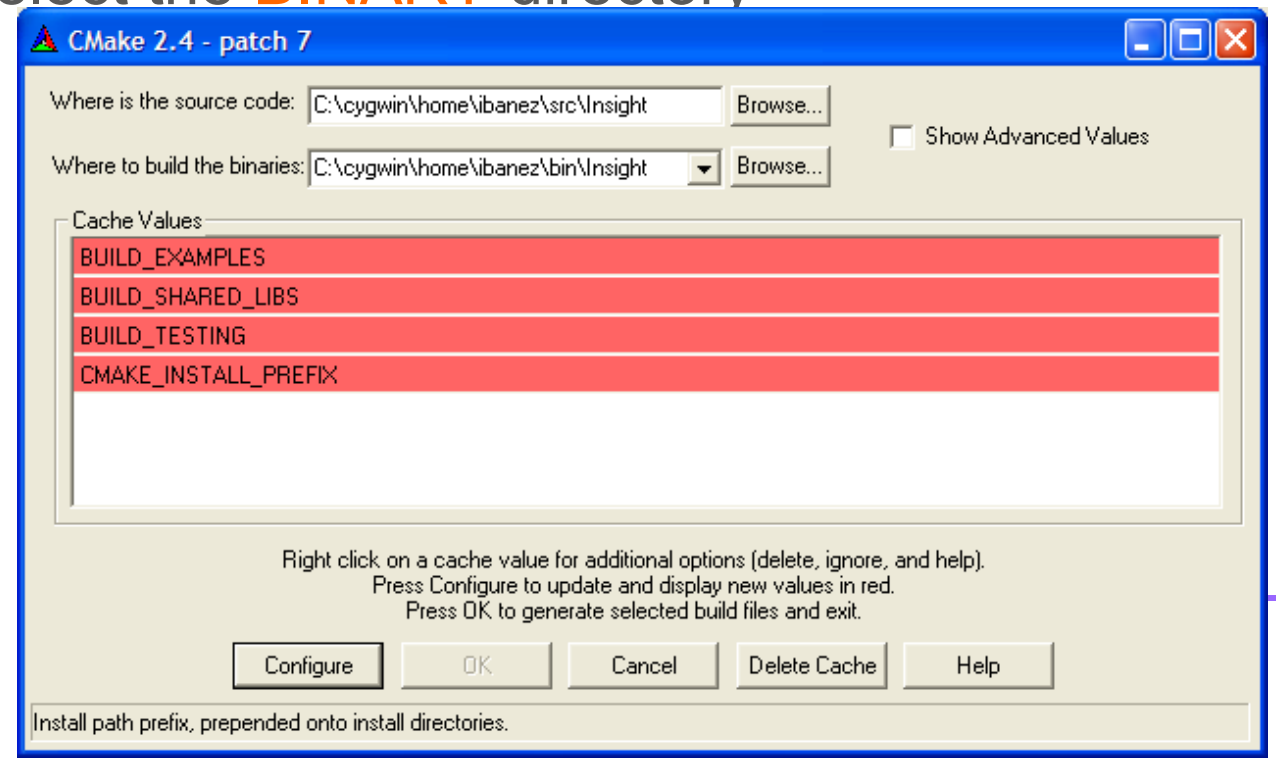
JG

Configuring ITK – MS-Windows

- Run CMake

www.cmake.org

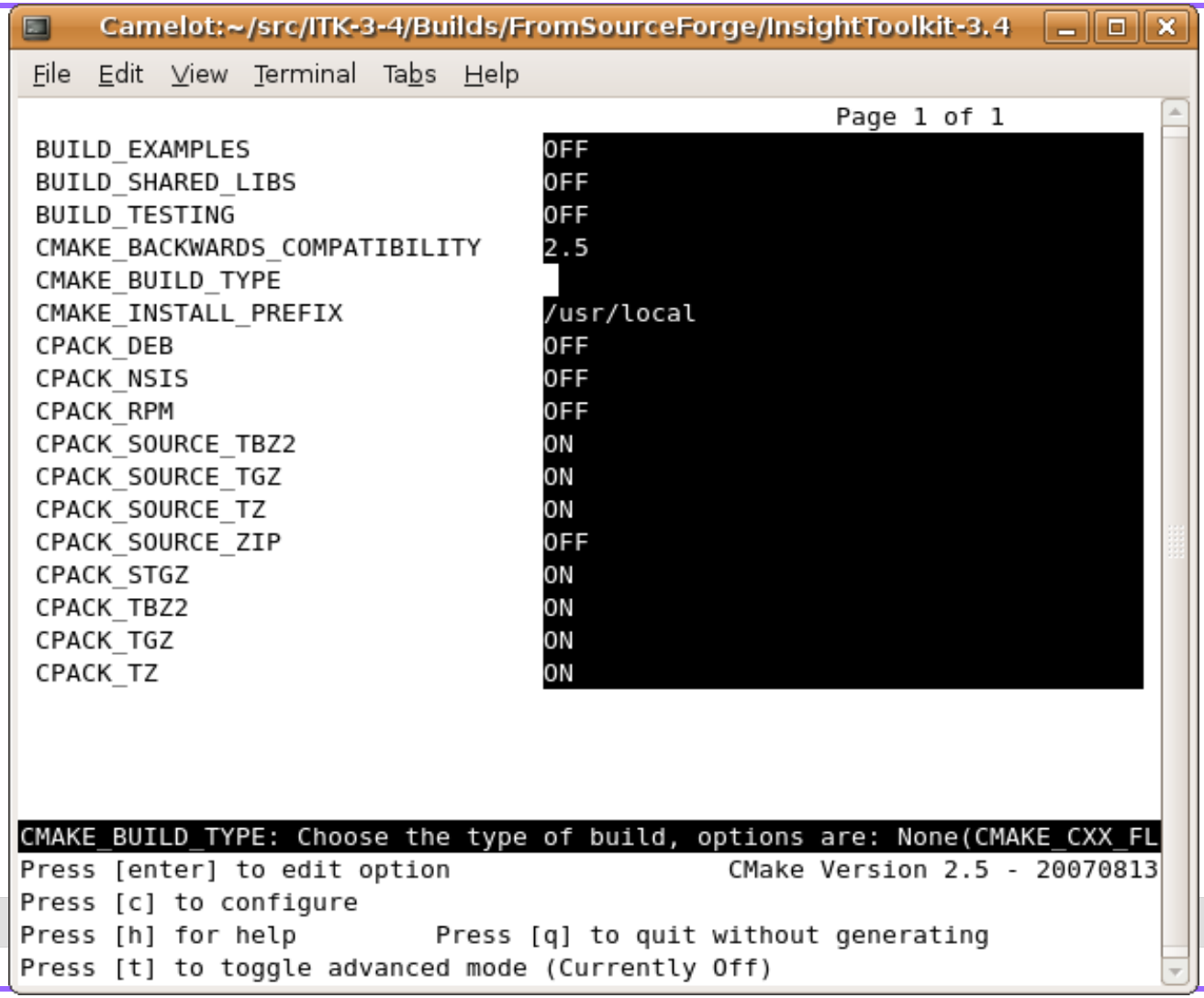
- Select the SOURCE directory
- Select the BINARY directory



Configuring ITK – Unix

- Create the BINARY directory (mkdir)
- Change directory to the BINARY directory (cd)
- Type ccmake with argument the SOURCE directory

Configuring ITK – Unix



LI, WS, HJ

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 Build

Libraries will be found in

ITK_BINARY / bin / { Debug, Release }

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

CMake for your project

- Write a CMakeLists.txt file describing your project in CMake's language
- Run CMake to generate an appropriate makefile/project/workspace for your compiler
- Compile as you normally would

JG

How CMake runs, cont.

- This is not unlike the configure-make process you may be familiar with from various Unix systems
- But… it works with many compilers
- CMakeLists.txt files are easy to perform revision control on

JG

CMakeLists.txt syntax

```
#Make sure the user's CMake is recent enough
cmake_minimum_required(VERSION 2.8.9)

#Give the project a name:
project(cool_demo)

# Find ITK version 4.8 for use with this program
find_package(ITK 4.8 REQUIRED)
include(${ITK_USE_FILE})

#The command line executable "demo" is requested
# to be built from the source file "demo_code.cxx"
# and must be linked to the ITK libraries
add_executable(demo demo_code.cxx)
target_link_libraries(demo ${ITK_LIBRARIES})
```

Building an application

- ITK comes with a simple application you can build in order to test the ITK libraries “out of source” (i.e. not built inside ITK)

- It can be found in:

  cp -R ITK/Examples/Installation ./test_src

JG

How to build HelloWorld

- Copy & rename the *Installation* directory somewhere outside of the Insight directory
  - cp -R ITK/Examples/Installation ./test_src
  - mkdir test-bld
- Run CMake in test-bld pointing to test_src
  - Remember the source/binary distinction
  - use test-bld as your build location
- Cmake will not automatically find ITK
  - edit the ITK_DIR option

JG

How to build HelloWorld, cont.

- Once CMake is happy, generate the makefile/project for your compiler
- Build HelloWorld
- Give it a try

JG

More examples

- You can turn on ITK's *Examples* option in CMake, which will build all of the examples for you
- Or… you can copy the examples out-of-source and build them like you did HelloWorld
- These examples link into ITK Software Guide; read the chapter, poke the code and see what happens…

JG

C++ Workflow thoughts

You should get used to the idea of:

1. Writing some code
2. Writing a CMakeLists.txt file
3. Running CMake
4. Building your code
5. Rinse, repeat

JG

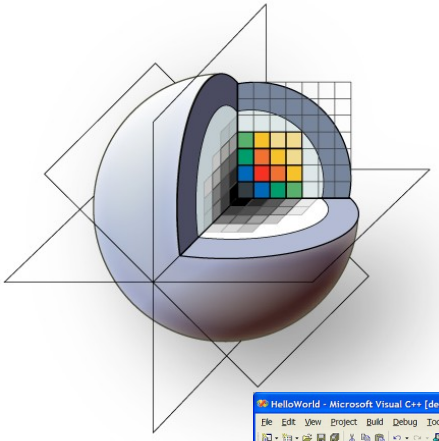
Learning objective

Following this tutorial, you'll be able

1) to plug-in an external program into Slicer3
2) to implement an image filter and to run the analysis from Slicer3
3) to write and run a test using the CTest tool

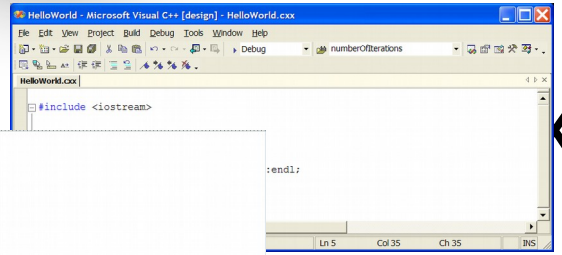
Overview

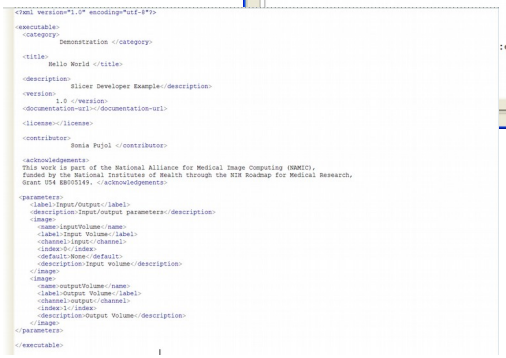
- Part A: integration of the HelloWorld program into Slicer3



3DSlicer

t A:
egrating an
xecutable to Slicer3

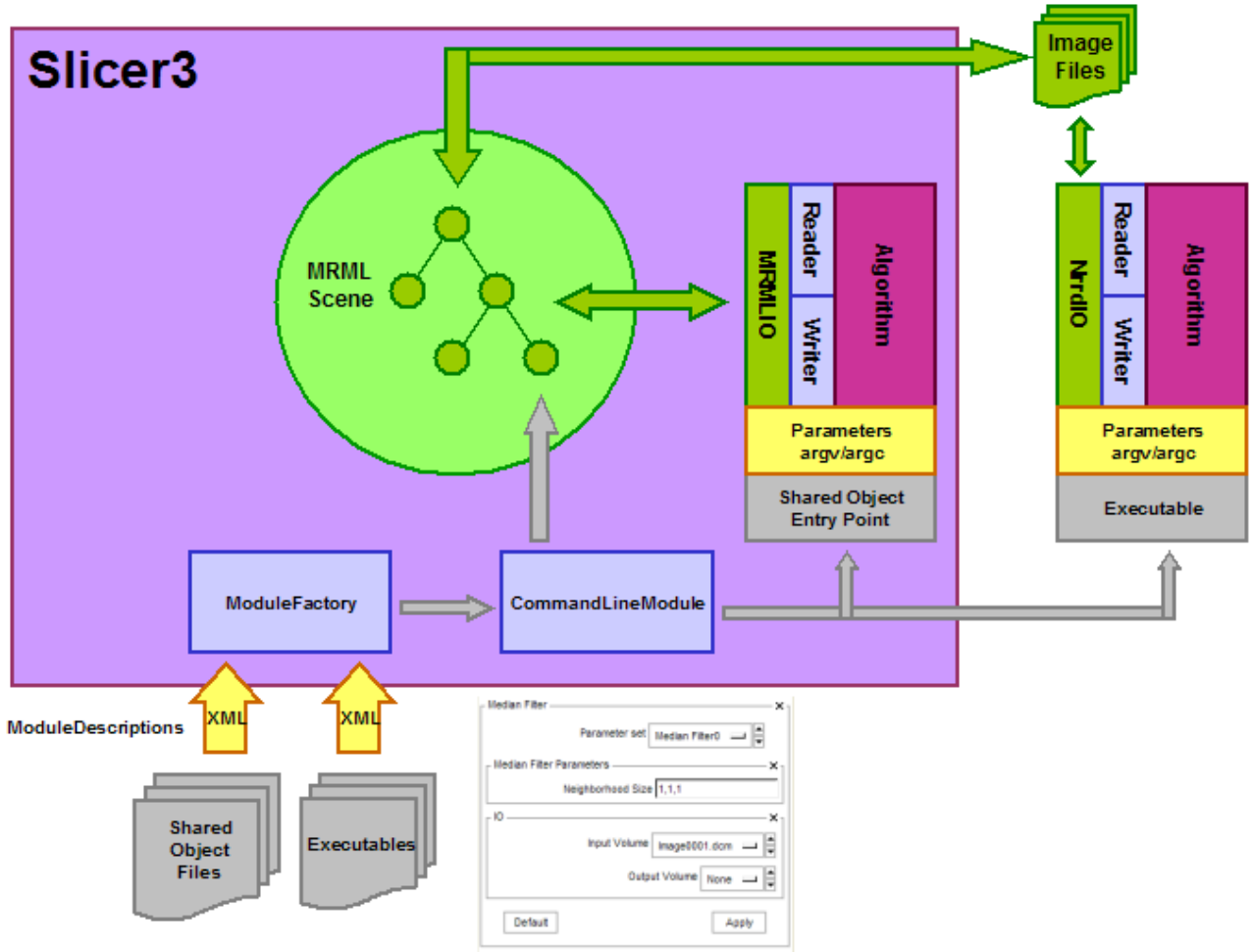




Slicer3 Execution Model

- This course is based on the Execution Model which provides a mechanism for incorporating command line programs as Slicer modules.
- The Slicer modules are described using XML files which are used to generate the C++ command line code and the Graphical User Interface (GUI).
- Execution Model URL

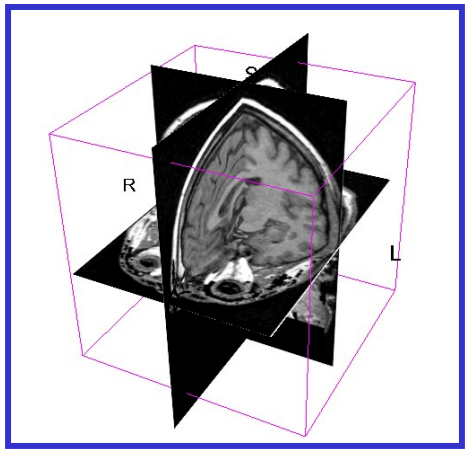
Slicer3 Execution Model



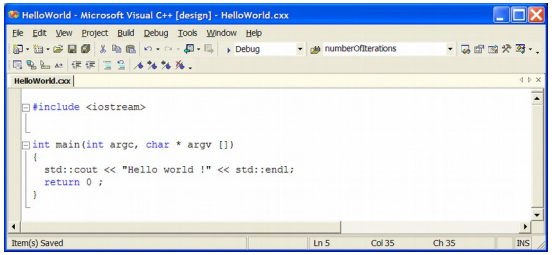
http://wiki.na-mic.org/Wiki/index.php/Slicer3:Execution_Model_Documentation

HelloWorld_CourseMaterial.tgz archive

```
unzip HelloWorld.zip
```

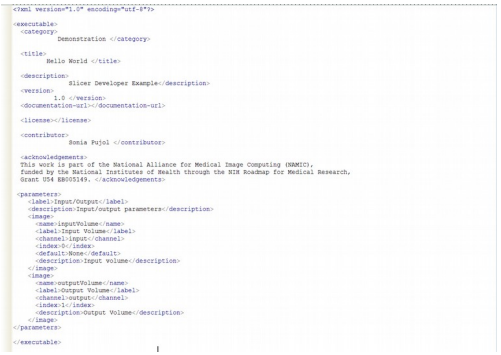


(data image)



HelloWorld.cxx

(application)



HelloWorld.xml

(Execution Model)

HelloWorld.xml

Module Description

```
<?xml version="1.0" encoding="utf-8"?>
<executable>
  <category>
          Demonstration</category>
  <title>
        Hello World</title>
  <description>
             Slicer Developer Example</description>
  <version>
         1.0</version>
  <documentation-url></documentation-url>
  <license></license>
  <contributor>
             Sonia Pujol, Ph.D. </contributor>
  <acknowledgements>
                  This work is part of the National Alliance for Medical Image Computing (NAMIC), funded by the National Institutes
        of Health through the NIH Roadmap for Medical Research, Grant U54 EB005149. </acknowledgements>
```

Module Parameters

```
  <parameters>
    <label>Input/Output</label>
    <description>Input/output parameters</description>
    <image>
      <name>helloWorldInputVolume</name>
      <label>Input Volume</label>
      <channel>input</channel>
      <index>0</index>
      <default>None</default>
      <description>Input volume</description>
    </image>
    <image>
      <name>helloWorldOutputVolume</name>
      <label>Output Volume</label>
      <channel>output</channel>
      <index>1</index>
      <description>Output filtered</description>
    </image>
</parameters>
</executable>
```

Module Description

```
<?xml version="1.0" encoding="utf-8"?>
<executable>
  <category>
          Demonstration</category>
  <title>Hello World</title>
  <description>
             Slicer Developer Example</description>
  <version>
         1.0</version>
  <documentation-url></documentation-url>
  <license></license>
  <contributor> Sonia Pujol, Ph.D. </contributor>
  <acknowledgements>
                  This work is part of the National Alliance for Medical Image
     Computing (NAMIC), funded by the National Institutes of Health through the
     NIH Roadmap for Medical Research, Grant U54 EB005149.
     </acknowledgements>
```

Module Parameters

Input Volume

Output Volume

```
<parameters>
  <label>Input/Output</label>
  <description>Input/output parameters</description>
  <image>
    <name>helloWorldInputVolume</name>
    <label>Input Volume</label>
    <channel>input</channel>
    <index>0</index>
    <default>None</default>
    <description>Input volume</description>
  </image>
  <image>
    <name>helloWorldOutputVolume</name>
    <label>Output Volume</label>
    <channel>output</channel>
    <index>1</index>
    <default>None</default>
    <description>Output filtered</description>
  </image>
</parameters>
</executable>
```


http://www.na-mic.org/Wiki/index.php/Slicer3:Execution_Model_Documentation

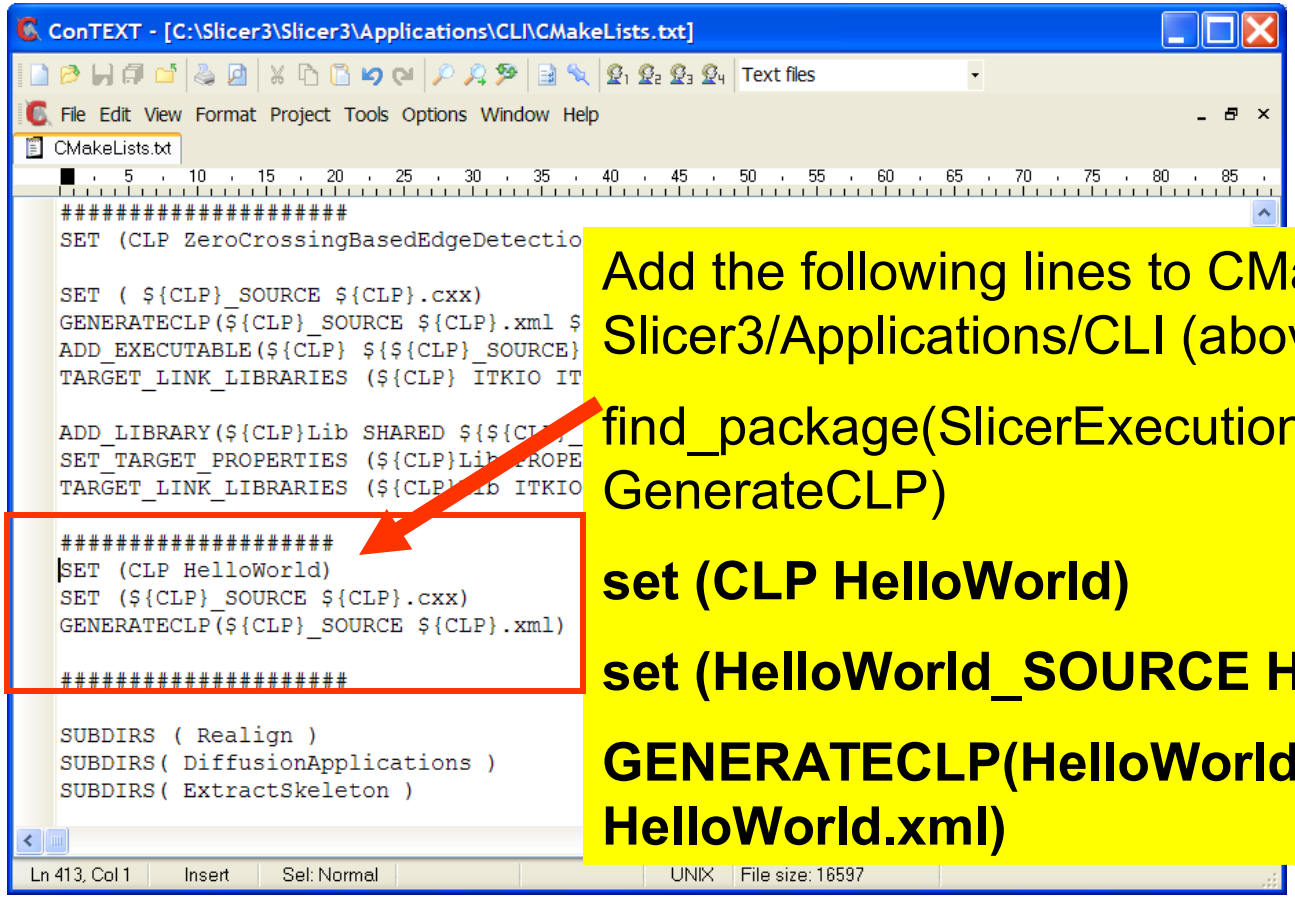
Modifying the source code

```
# include <iostream>

int main(int argc, char * argv [])
{

   std::cout<< “Hello World !”<<std::endl;

   return 0 ;
}
```

Modifying the source code

```
# include <iostream>
#include “HelloWorldCLP.h”
int main(int argc, char * argv [])
{
  PARSE_ARGS;
  std::cout<< “Hello World !”<<std::endl;


  return EXIT_SUCCESS ;
}
```

Editing CMakeLists.txt



Add the following lines to CMakeLists.txt located in Slicer3/Applications/CLI (above the SUBDIRS lines):

find_package(SlicerExecutionModel REQUIRED GenerateCLP)

set (CLP HelloWorld)

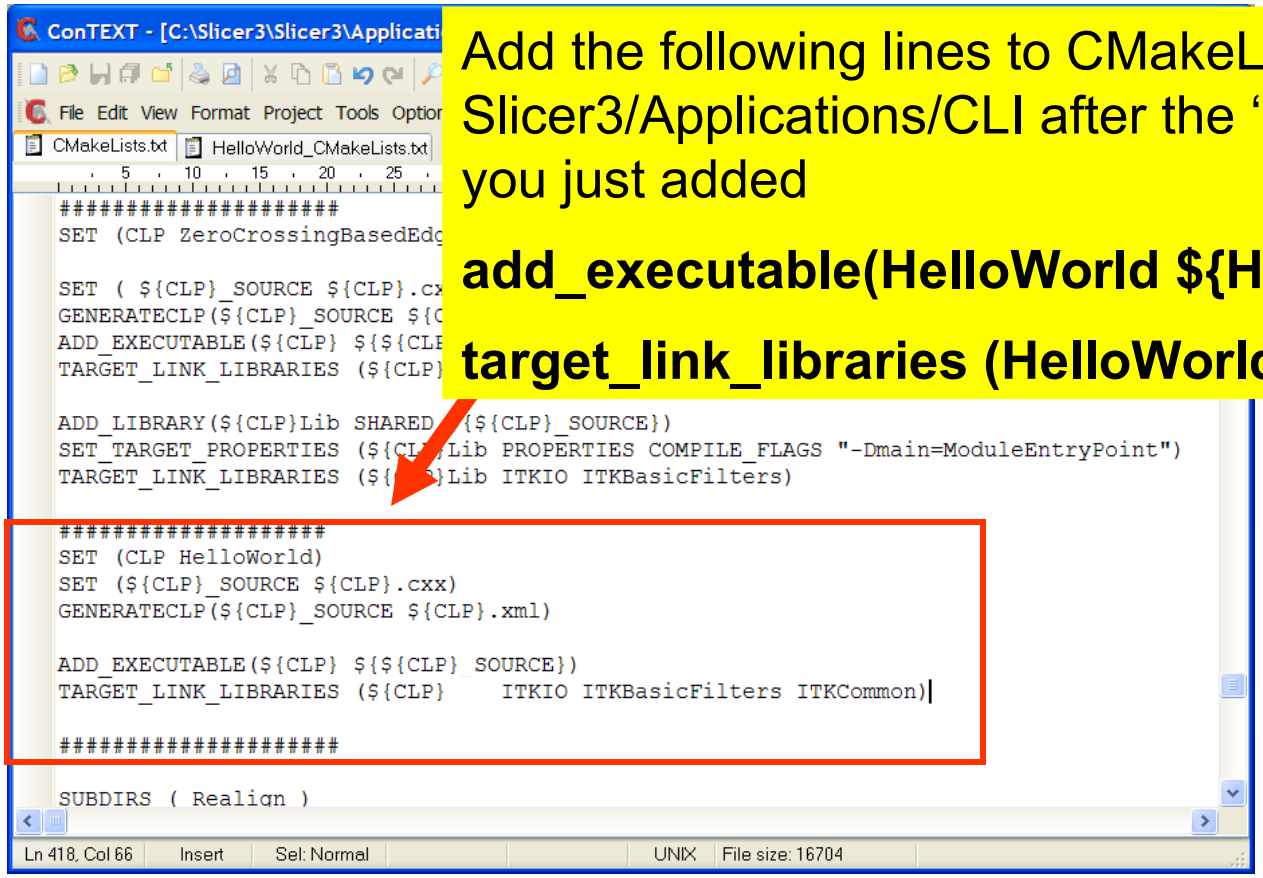
set (HelloWorld_SOURCE HelloWorld.cxx)

GENERATECLP(HelloWorld_SOURCE HelloWorld.xml)

*GENERATECLP generates the file **HelloWorldCLP.h** for parsing the command line arguments.*

'CLP' means Command Line Processing

Editing CMakeLists.txt



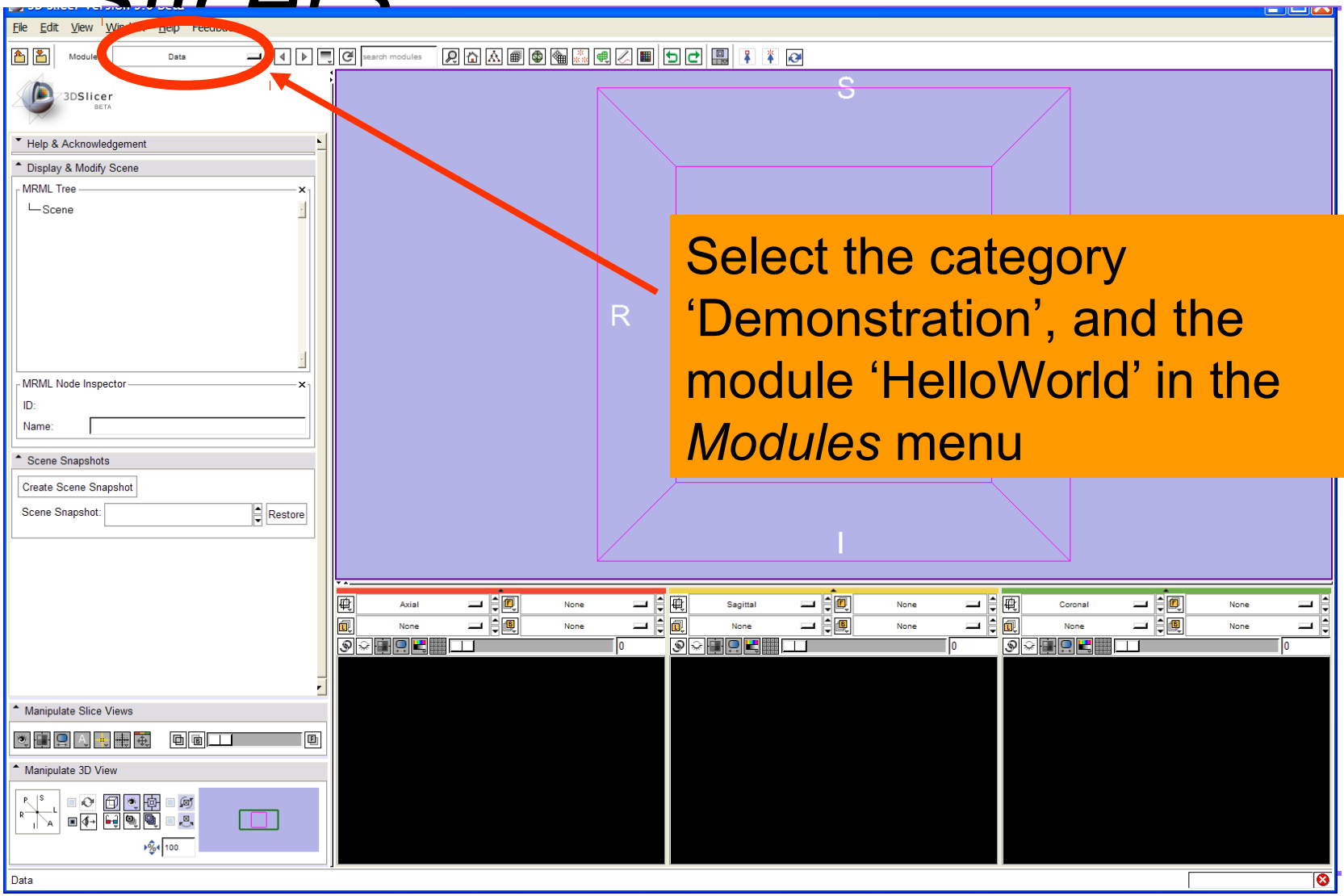
Add the following lines to CMakeLists.txt located in Slicer3/Applications/CLI after the 'GENERATECLP' line you just added

add_executable(HelloWorld ${HelloWorld_SOURCE})

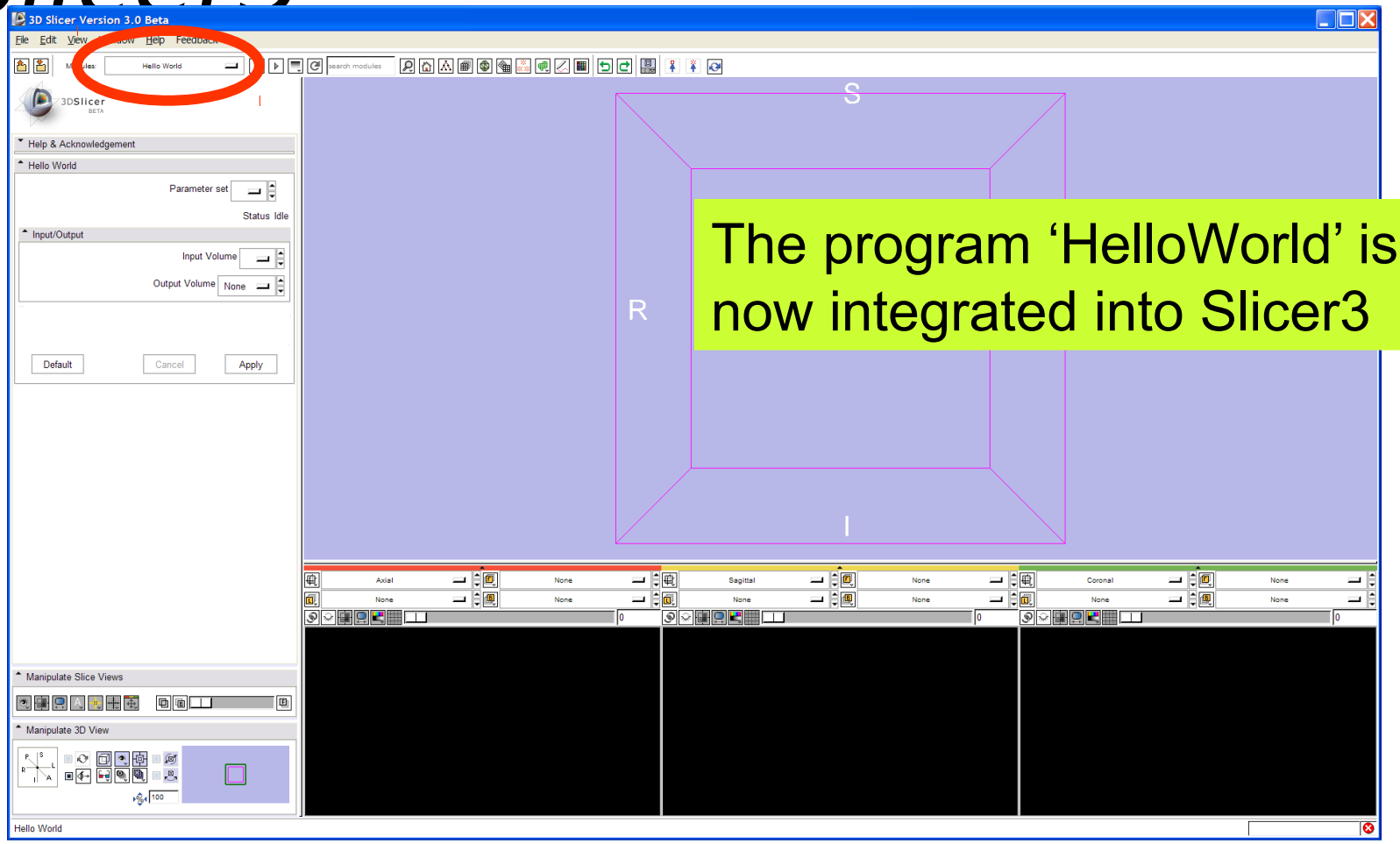
target_link_libraries (HelloWorld ${ITK_LIBRARIES})

ADD_EXECUTABLE creates the stand-alone executable ***HelloWorld.exe*** *that can be run from a command line.*

HelloWorld module in Slicer3



HelloWorld Module in Slicer3



The program 'HelloWorld' is now integrated into Slicer3

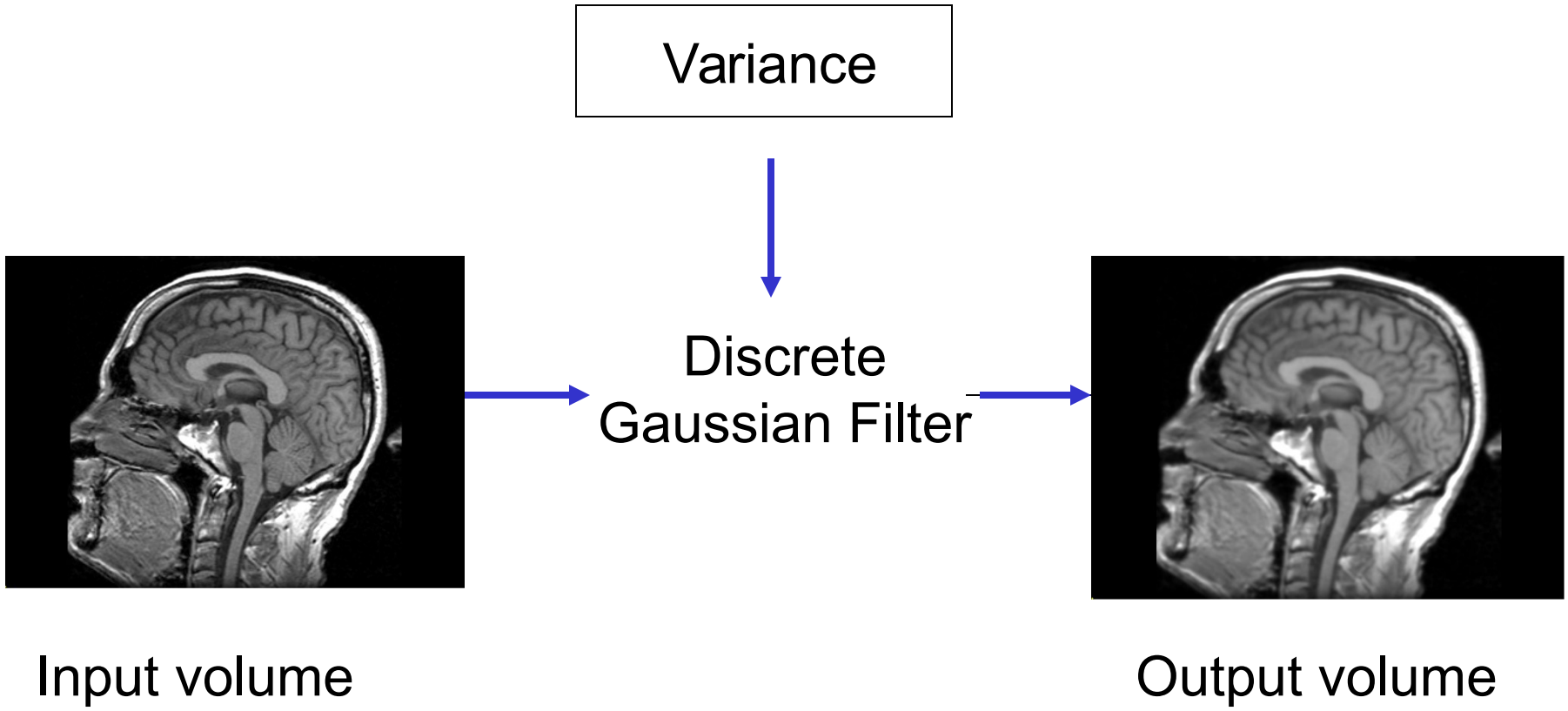
Part B:
ementing an
e filter

Goal

- In this section, we'll implement a Gaussian smoothing operator to 'blur' the images and remove detail and noise.

- This implementation will allow us to run the filter on volumes loaded in Slicer, and to integrate the resulting filtered volumes as MRML nodes.

Discrete Gaussian Filter



Implementing Input/Output functionalities

Add the following lines to HelloWorld.cxx

```
#include <iostream>
#include "HelloWorldCLP.h"
#include "itkImage.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"
int main(int argc, char * argv [])
{
  PARSE_ARGS;
  std::cout << "Hello World!" << std::endl;
  return EXIT_SUCCESS ;
}
```

Implementing Input/Output functionalities

Add the following command lines to set-up the reading and writing functionalities in the 'main' procedure in HelloWorld.cxx

```
int main ( int argc, char * argv[]){
PARSE_ARGS;
std::cout << "Hello World!" << std::endl;
typedef itk::Image< short,  3 >   ImageType;
typedef itk::ImageFileReader< ImageType  >  ReaderType;
typedef itk::ImageFileWriter< ImageType >  WriterType;
ReaderType::Pointer reader = ReaderType::New();
WriterType::Pointer writer = WriterType::New();

return EXIT_SUCCESS;
}
```

Implementing Input/Output functionalities

Set the input and output volumes parameters defined in HelloWorld.xml

```
int main ( int argc, char * argv[]){
PARSE_ARGS;
std::cout << "Hello World!" << std::endl;
typedef itk::Image< short,  3 >   ImageType;
typedef itk::ImageFileReader< ImageType  >  ReaderType;
typedef itk::ImageFileWriter< ImageType >  WriterType;
ReaderType::Pointer reader = ReaderType::New();
WriterType::Pointer writer = WriterType::New();

reader->SetFileName( helloWorldInputVolume.c_str() );
writer->SetFileName (helloWorldOutputVolume.c_str());

return EXIT_SUCCESS;
}
```

HelloWorld.xml

```
<?xml version="1.0" encoding="utf-8"?>
<executable>
  <category>
          Demonstration</category>
  <title>
        Hello World</title>
  <description>
             Slicer Developer Example</description>
  <version>
          1.0</version>
  <documentation-url></documentation-url>
  <license></license>
  <contributor>
             Sonia Pujol</contributor>
  <acknowledgements>
                This work is part of the National Alliance for Medical Image Computing (NAMIC), funded by the National Institutes of Health
        through the NIH Roadmap for Medical Research, Grant U54 EB005149. </acknowledgements>

<parameters>
    <label>Input/Output</label>
    <description>Input/output parameters</description>
     ….
</parameters>

<parameters>
           <label>Discrete Gaussian Parameters</label>
    <description>Parameters of the Discrete Gaussian Filter </description>

</parameters>

</executable>
```

Add a new parameter group to HelloWorld.xml

Adding a new parameter to HelloWorld.xml

```
<parameters>

    <label>Discrete Gaussian Parameters</label>
    <description>Parameters of the Discrete
  Gaussian Filter </description>
<double>
      <name>variance</name>
      <longflag>--variance</longflag>
      <description>Variance ( width of the filter
  kernel) </description>
      <label>Variance</label>
      <default>0.5</default>
</double>
</parameters>
```

Add the parameter ‘variance’ which corresponds to the variance of the Discrete Gaussian Filter to HelloWorld.xml

Implementing the filter in HelloWorld.cxx

Implement the filter itk::DiscreteGaussianImageFilter

```
#include "itkDiscreteGaussianImageFilter.h"
int main ( int argc, char * argv[]){
PARSE_ARGS;
std::cout << "Hello World!" << std::endl;
typedef itk::Image< short,  3 >   ImageType;
typedef itk::ImageFileReader< ImageType  >  ReaderType;
typedef itk::ImageFileWriter< ImageType >  WriterType;
ReaderType::Pointer reader = ReaderType::New();
WriterType::Pointer writer = WriterType::New();
reader->SetFileName( helloWorldInputVolume.c_str() );
writer->SetFileName(helloWorldOutputVolume.c_str());
typedef itk::DiscreteGaussianImageFilter <ImageType, ImageType>
   FilterType;
  FilterType::Pointer filter = FilterType::New();

return EXIT_SUCCESS;}
```

Implementing the filter in HelloWorld.cxx

Add the following lines for the filter execution:

```
try {
    filter->SetInput(reader->GetOutput());
    filter->SetVariance(variance);
    writer->SetInput(filter->GetOutput());
    writer->Update();
  }
catch (itk::ExceptionObject &excep)
{
     std::cerr << argv[0] << ": exception caught !" <<
std::endl;
     return EXIT_FAILURE;
}
```

HelloWorld.cxx

```
int main ( int argc, char * argv[]){
PARSE_ARGS;
std::cout << "Hello World!" << std::endl;
typedef itk::Image< short,  3 >   ImageType;
typedef itk::ImageFileReader< ImageType  >  ReaderType;
typedef itk::ImageFileWriter< ImageType >  WriterType;
ReaderType::Pointer reader = ReaderType::New();
WriterType::Pointer writer = WriterType::New();
reader->SetFileName( helloWorldInputVolume.c_str() );
writer->SetFileName (helloWorldOutputVolume.c_str());
typedef itk::DiscreteGaussianImageFilter <ImageType, ImageType> FilterType;
FilterType::Pointer filter = FilterType::New();
try {
   filter->SetInput(reader->GetOutput());
   filter->SetVariance(variance);
   writer->SetInput(filter->GetOutput());
   writer->Update();}
catch (itk::ExceptionObject &excep){
    std::cerr << argv[0] << ": exception caught !"
    return EXIT_FAILURE;}
return EXIT_SUCCESS;}
```

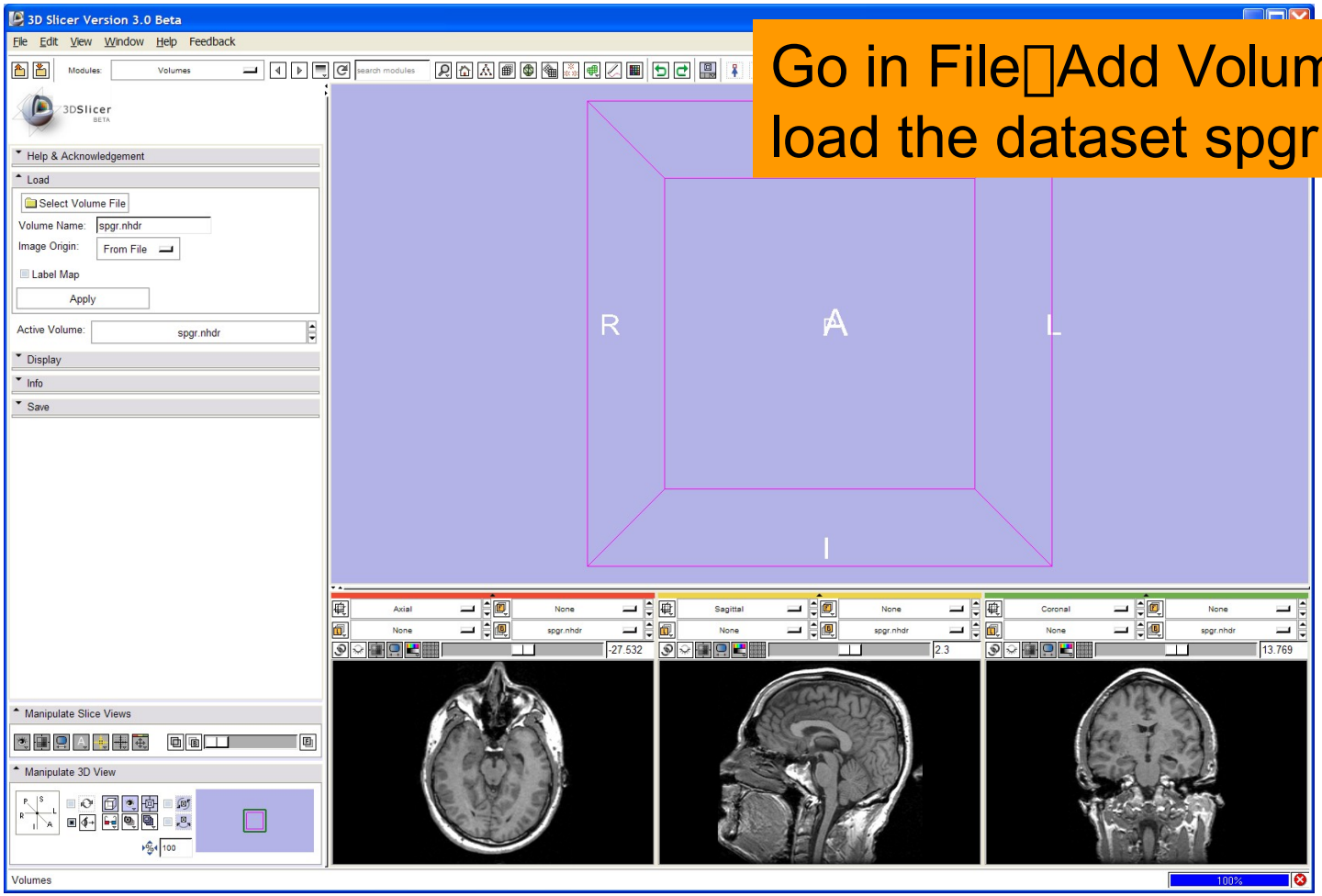
Re-Compile **HelloWorld**

- Linux and Mac:

Type **make** in **Slicer3-build/**

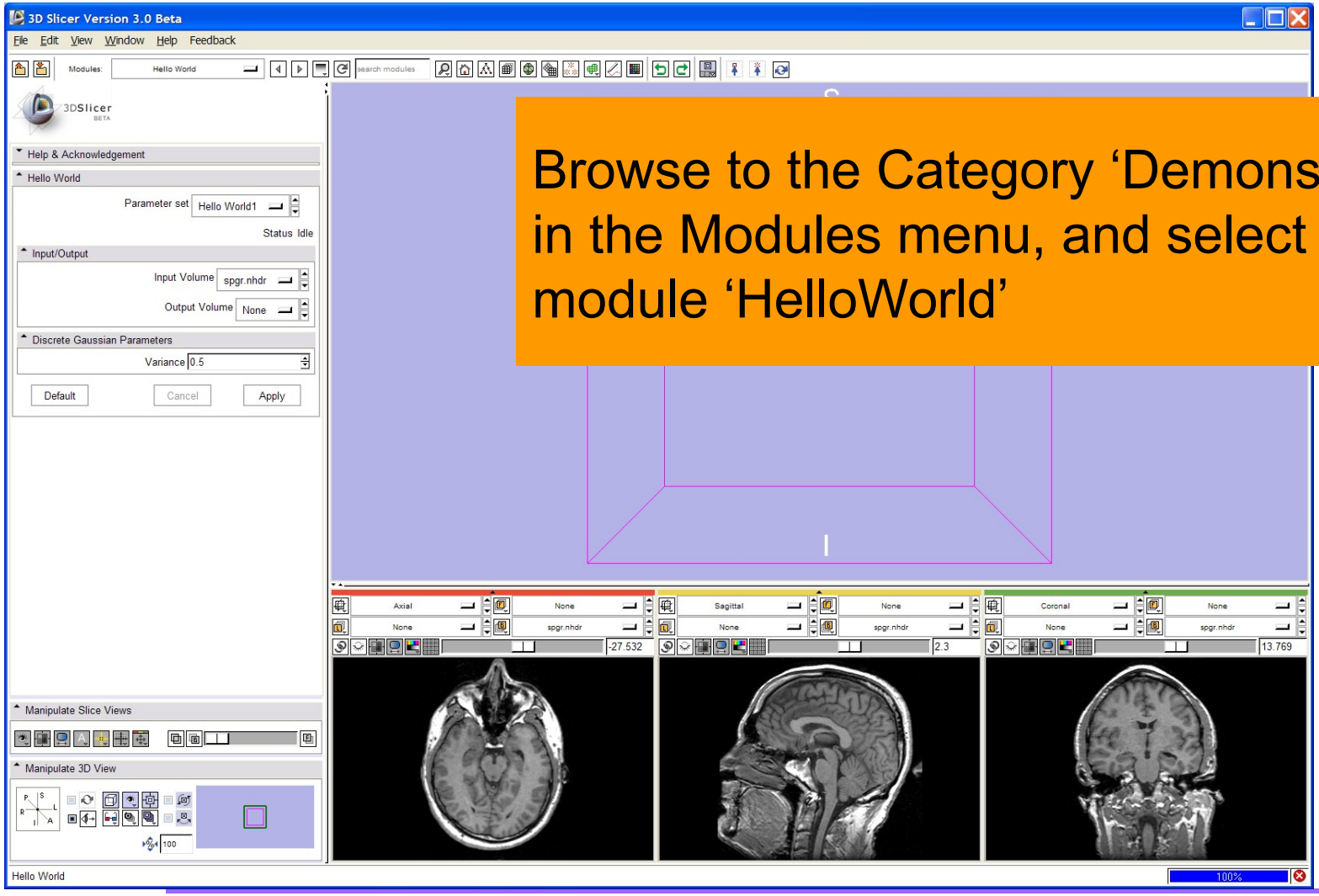
- **Windows:** build **Slicer3.sln**

Running the Filter



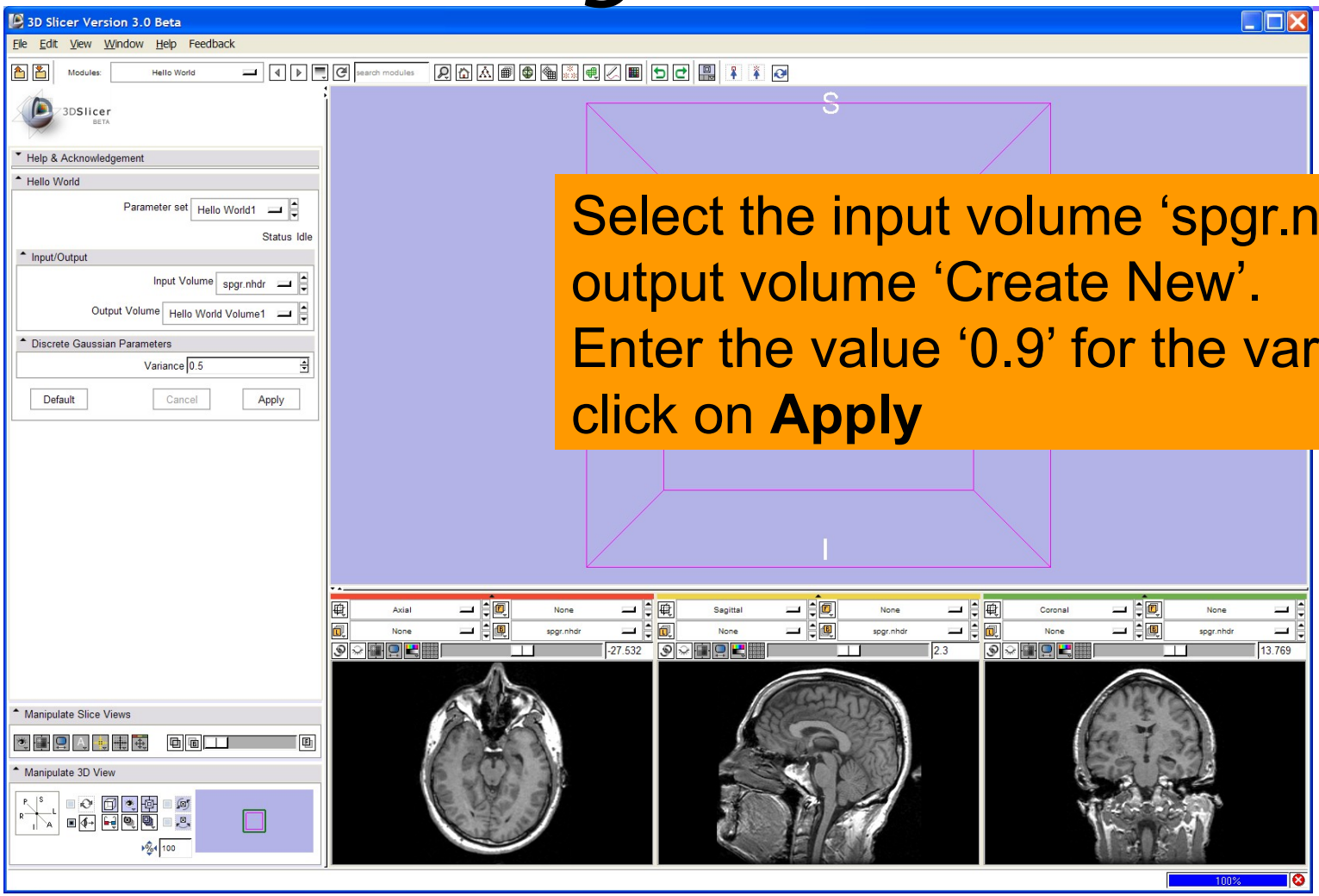
Go in File→Add Volume and load the dataset spgr.nrrd

Running the Filter



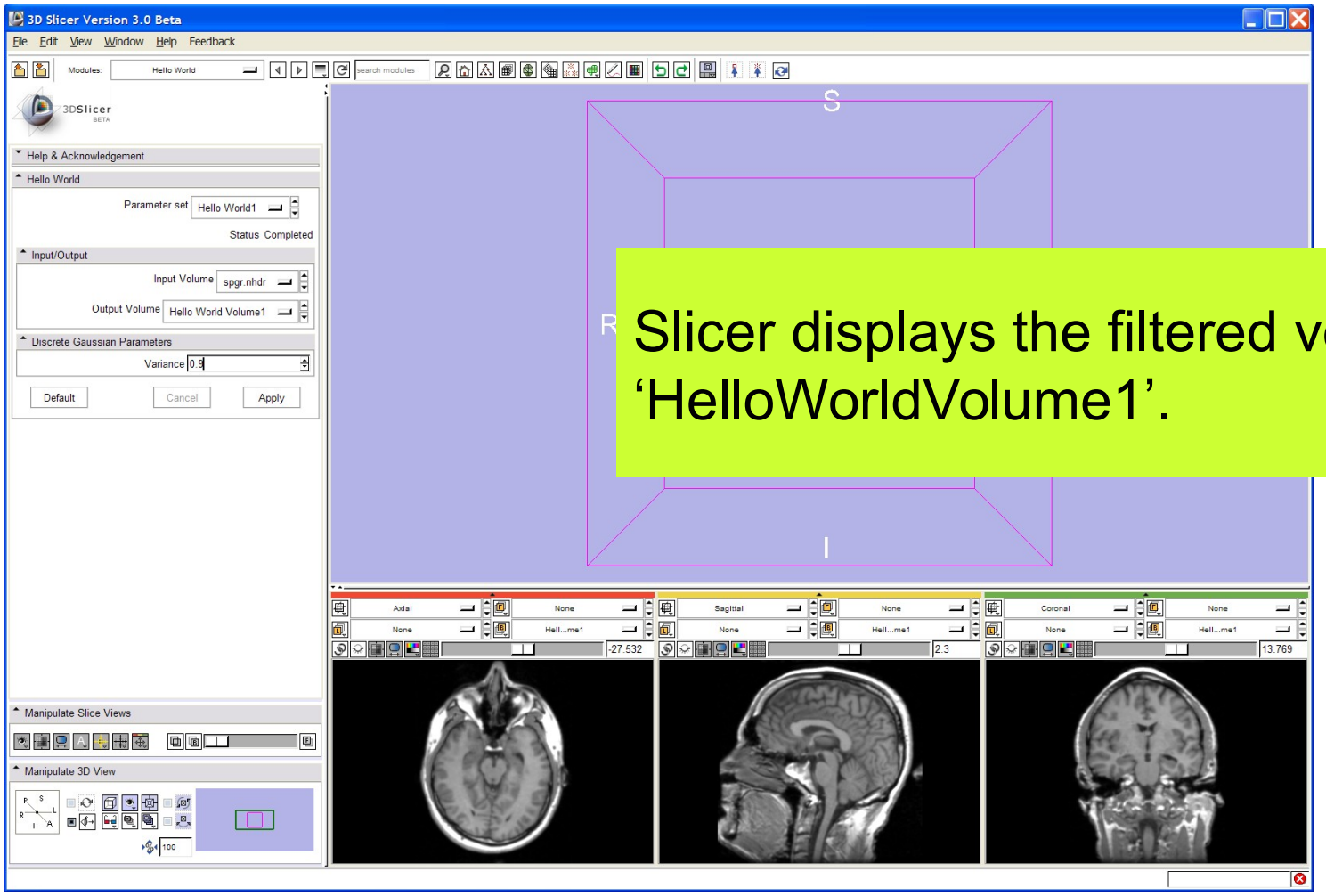
Browse to the Category ‘Demonstration’ in the Modules menu, and select the module ‘HelloWorld’

Running the Filter



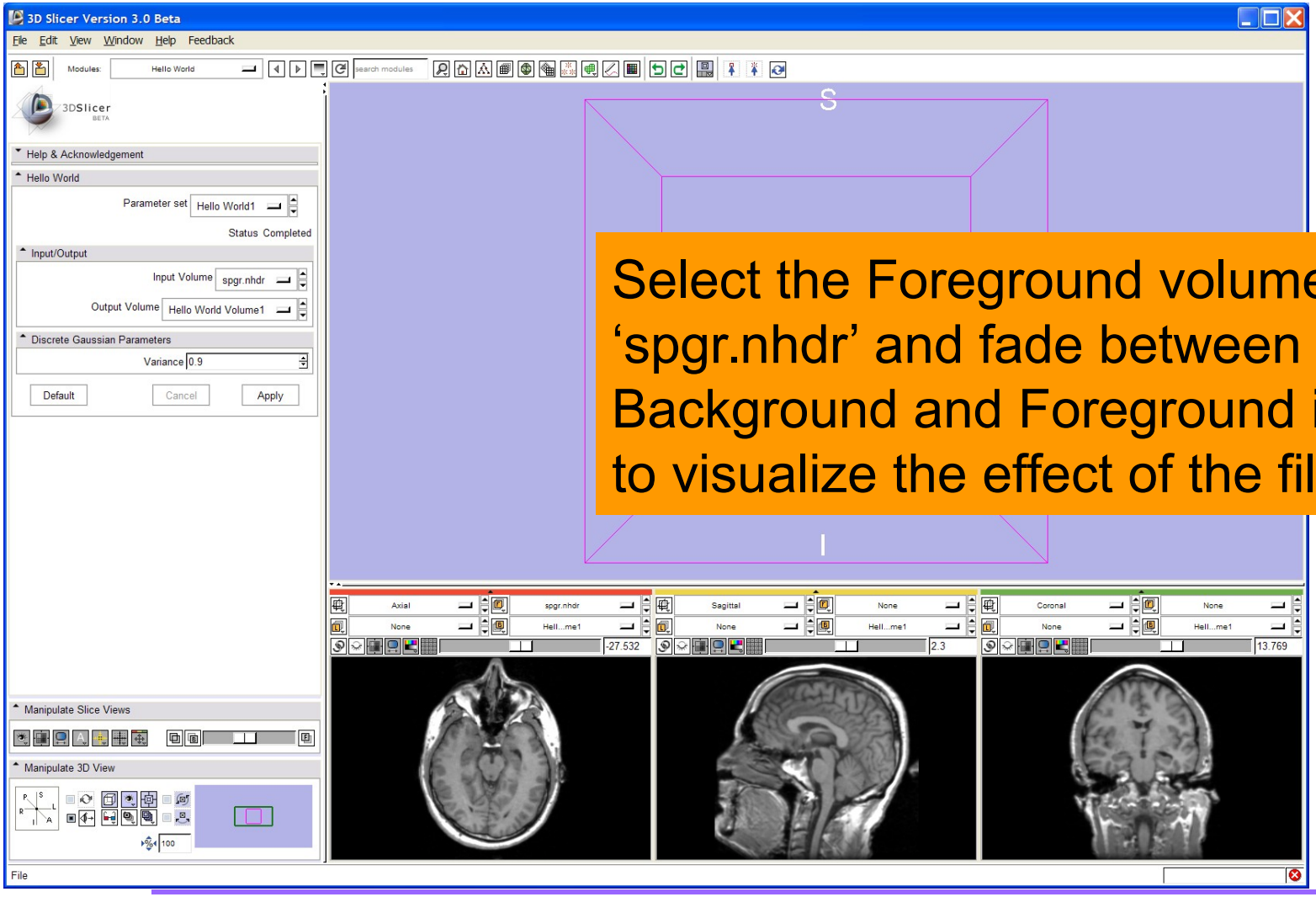
Select the input volume ‘spgr.nhdr’, the output volume ‘Create New’.
Enter the value ‘0.9’ for the variance and click on **Apply**

Running the Filter



Slicer displays the filtered volume ‘HelloWorldVolume1’.

Running the Filter



Select the Foreground volume ‘spgr.nhdr’ and fade between the Background and Foreground images to visualize the effect of the filter.

Conclusion

- This course described functionalities for integrating, developing and testing external program within Slicer3.
- The Execution Model of Slicer3 provides a simple mechanism for incorporating command line programs as Slicer modules.
- www.slicer.org

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

Hello CLI:
contributing an algorithm into Slicer 4

Nadya Shusharina, Greg Sharp
Massachusetts General Hospital
nshusharina@partners.org

NA-MIC Tutorial Contest: Summer 2013

Learning Objective

This step by step tutorial leads you through developing command line interface (CLI) for Slicer 4 (*http://www.slicer.org*)

- Getting ready
- Building a template module
- Building module for image thresholding

Pre-requisite

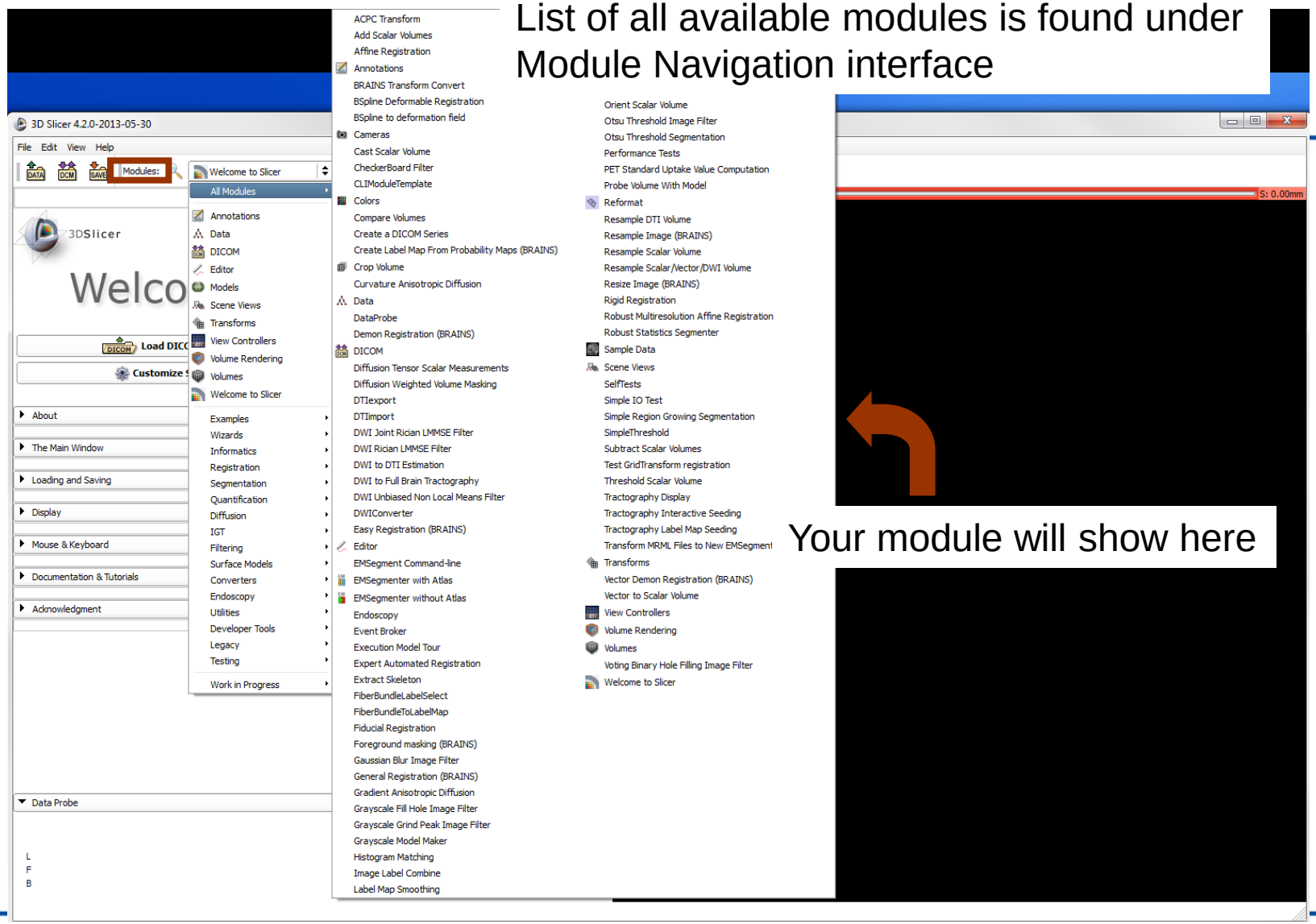
- Slicer is an open-source software for segmentation, registration and visualization of medical imaging data
- The platform is developed through a multi-institution effort of several NIH funded large-scale consortia
- Slicer is for medical research only, and is not FDA approved
- For the general information and “How to” tutorials please visit *http://www.slicer.org/slicerWiki/index.php/Documentation/Nightly/Training*

Material

- We recommend to build Slicer 4 from source
- Refer to the following page:
  http://www.slicer.org/slicerWiki/index.php/Documentation/Nightly/Developers/Build_Instructions
- Read prerequisites and platform specific instructions and install all required tools
- Checkout Slicer source
- Configure and build
- Become Slicer community member:
  http://www.slicer.org/slicerWiki/index.php/Documentation/Nightly/Developers/StartHere

Run Slicer: *path-to-SlicerSuperbuild/Slicer-build/Slicer.exe*

List of all available modules is found under Module Navigation interface



CLI module

- Standalone executables, shared libraries or scripts
- Introduced via plugin mechanism
- XML description produces UI
- Command line parsing code
- Link: *http://www.na-mic.org/Wiki/index.php/File:Slicer4_CLI.ppt*

Creating module: Step 1

- Make sure that any version of Python is installed on your computer
- From Slicer source directory run the command: ./Utilities/Scripts/ModuleWizard.py --template ./Extensions/Testing/CLIExtensionTemplate --target ../My_Module My_Module
- This command created a new directory “My_Module” parallel to Slicer source directory

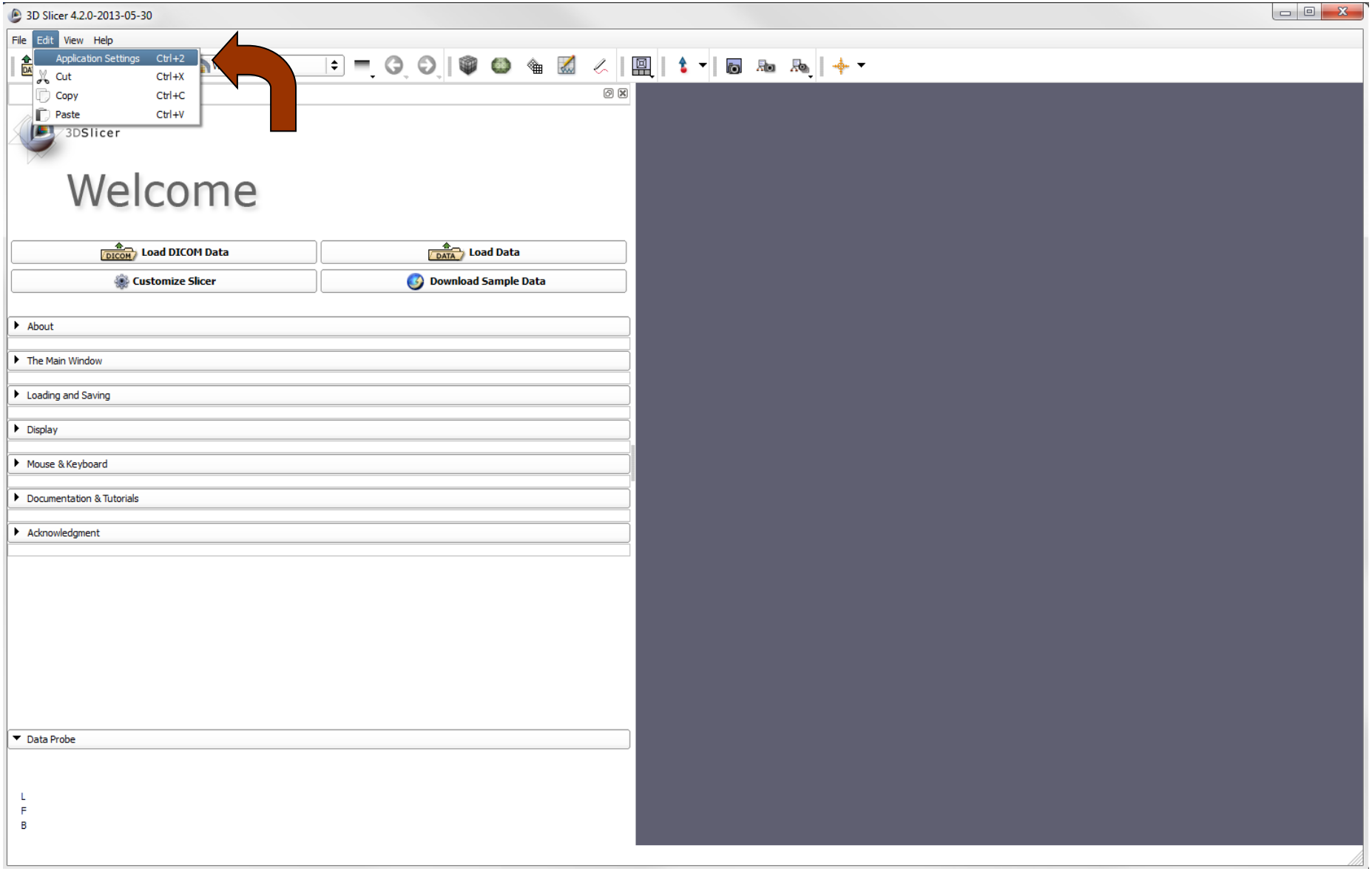
Build extension: Windows

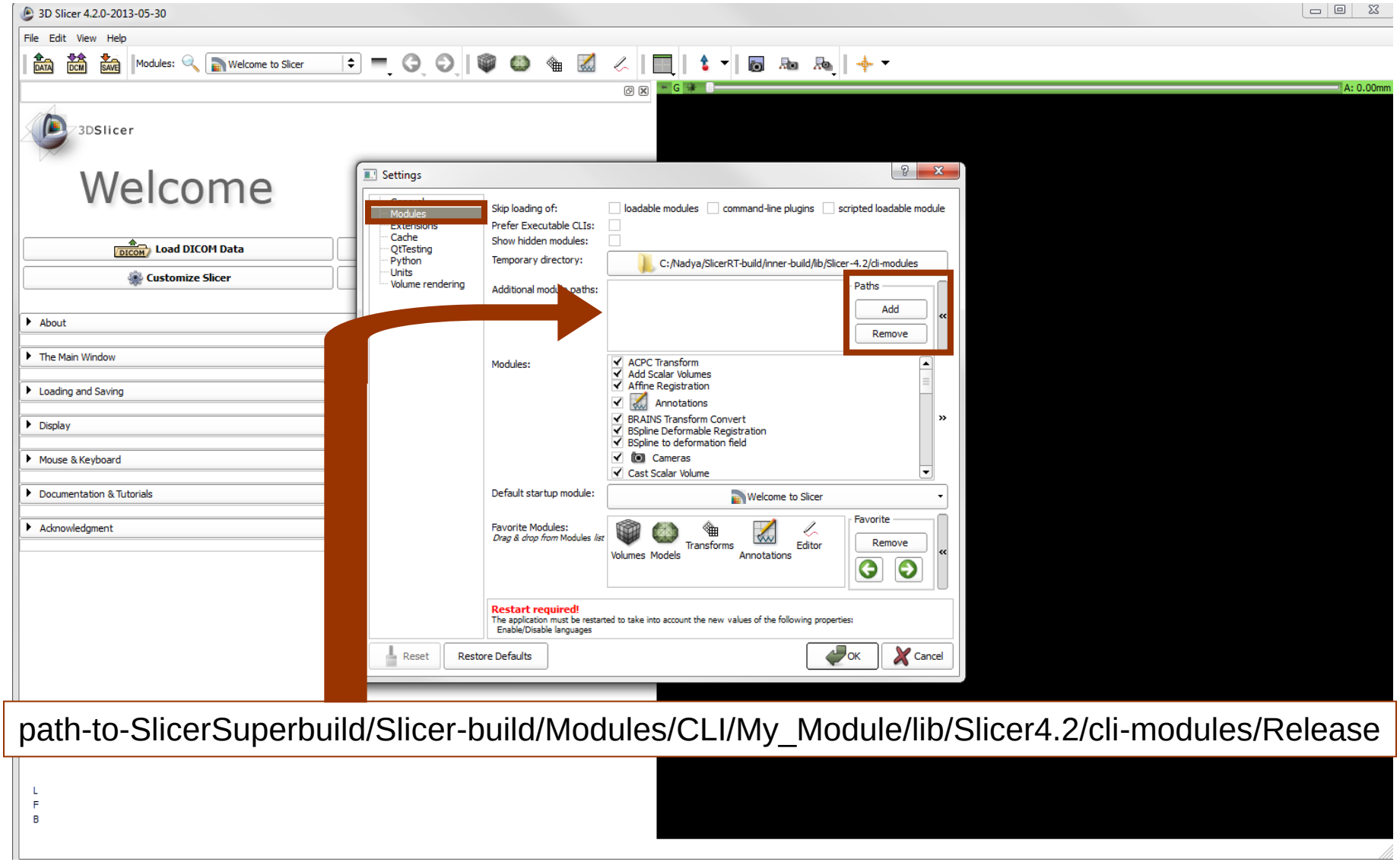
- Run cmake
- Set Slicer_DIR to path-to-Slicer-Superbuid/Slicer-build
- Set *../My_Module* as a source directory
- Choose a build directory
- Use default settings
- Compile using VC

Build extension: Linux

- $ mkdir My_Module-build
- $ cd My_Module-build
- $ cmake -DSlicer_DIR:PATH=/path-to-Slicer-Superbuild/Slicer-build ../My_Module
- $ make

Set the path to My_Module in the Application Settings

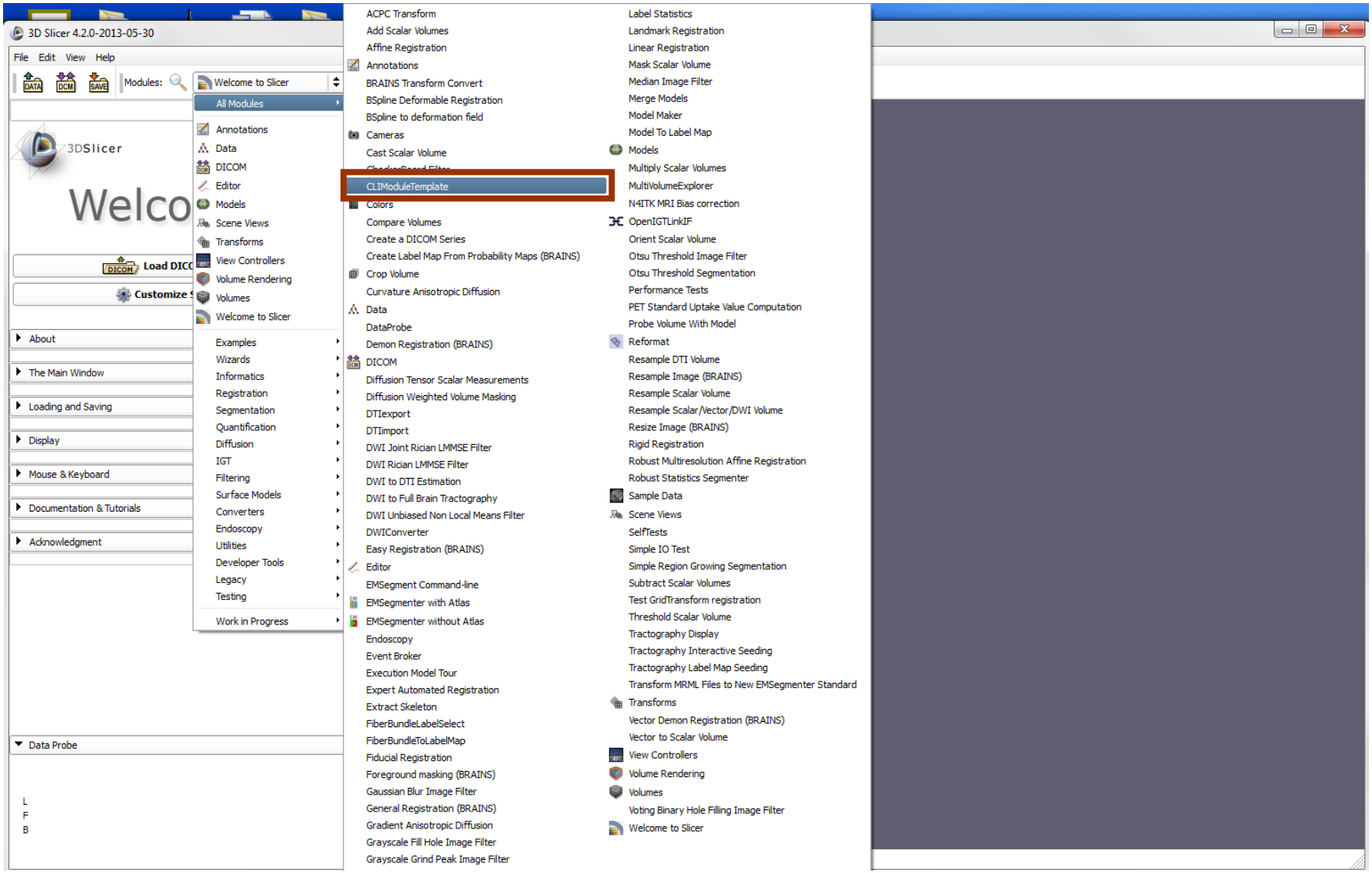




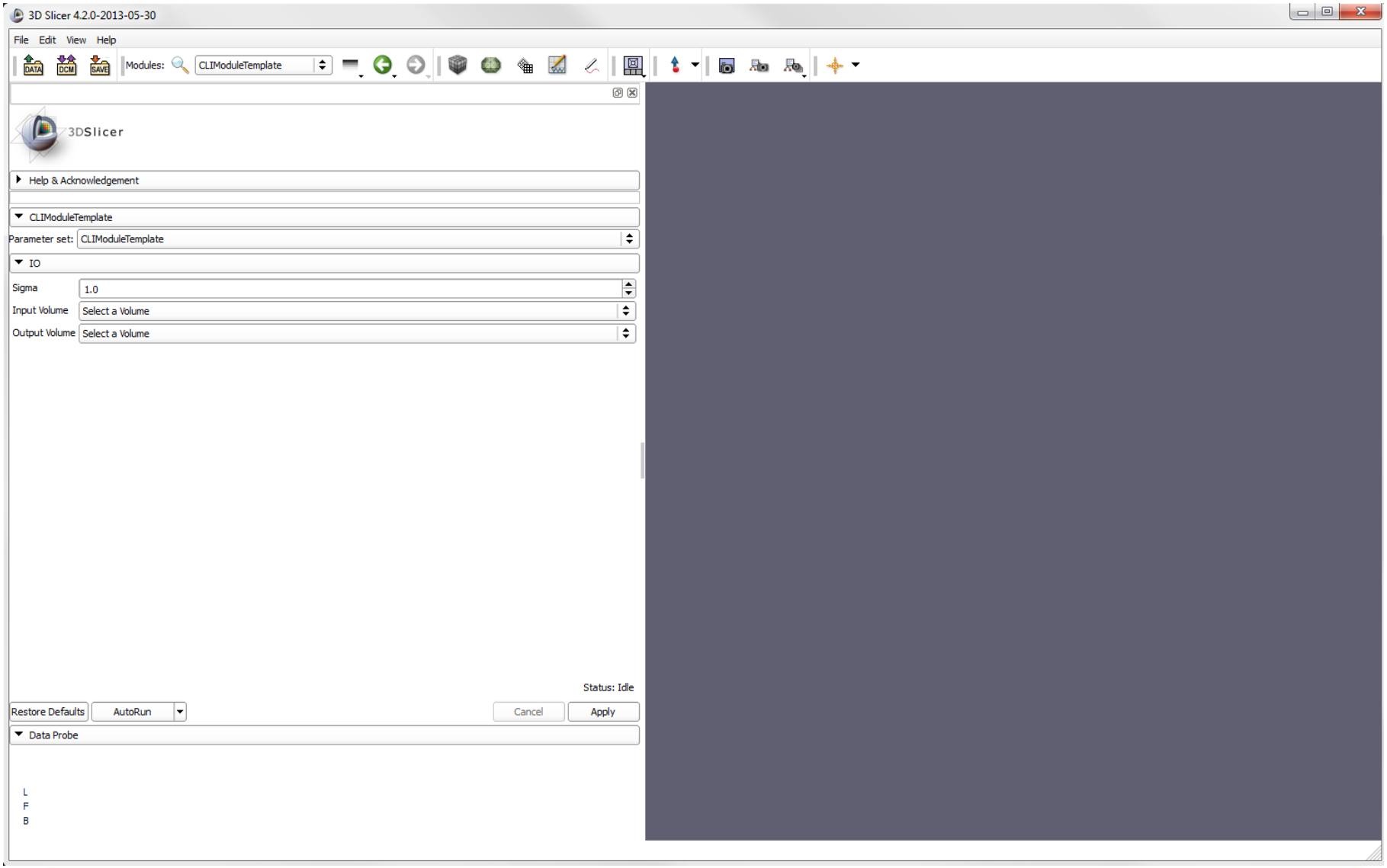
path-to-SlicerSuperbuild/Slicer-build/Modules/CLI/My_Module/lib/Slicer4.2/cli-modules/Release

Restart Slicer!

Find the module CLIModuleTemplate in the Module Navigation interface

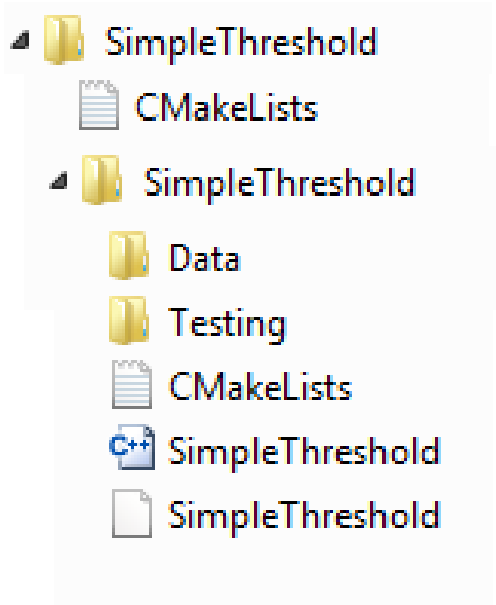


Open the module. Congratulations!



Creating module: Step 2

- Download sample data:
  https://forge.abcd.harvard.edu/gf/download/frsrelease/85/2851/hello_cli.zip
- The name of directory is the name of the Module as it appears in the list of modules



Module function

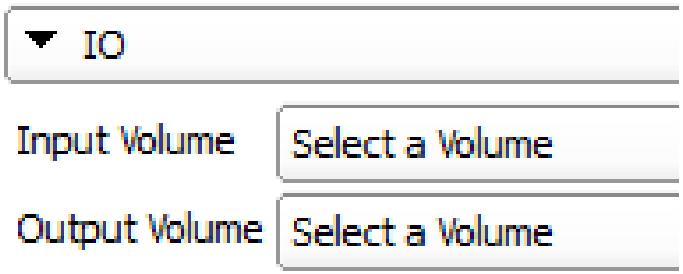
processing

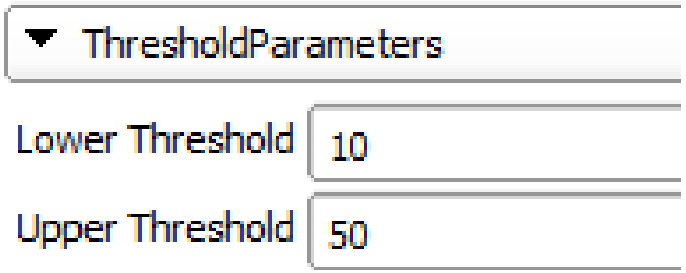
Input image

Output image

Module description

GUI





XML

```
<image>
    <name>inputVolume</name>
    <label>Input Volume</label>
    <channel>input</channel>
    <index>0</index>
    <description><![CDATA[Inputvolume]]></decription>
  </image>
```

```
<integer>
   <name>lowerThreshold</name>
   <longflag>--lowerThreshold</longflag>
   <description><![CDATA[The lower threshold]]></description>
   <label>Lower Threshold</label>
   <default>10</default>
 </integer>
 <integer>
```

Module coding

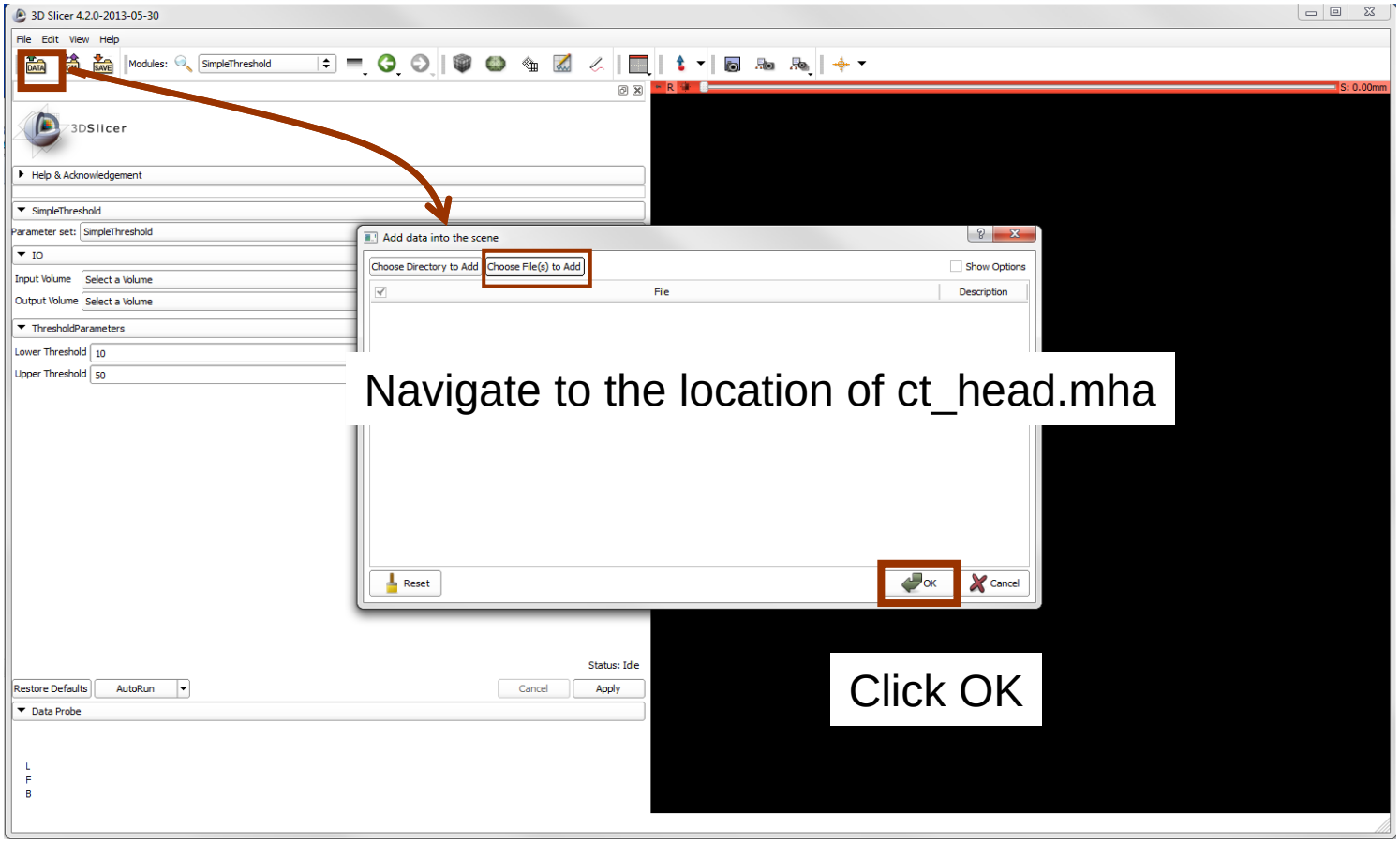
```
typedef itk::ImageFileReader<InputImageType>  ReaderType;
typedef itk::ImageFileWriter<OutputImageType> WriterType;
typename ReaderType::Pointer reader = ReaderType::New();
reader->SetFileName( inputVolume.c_str() );
itk::GetImageType(inputVolume, pixelType, componentType);
```

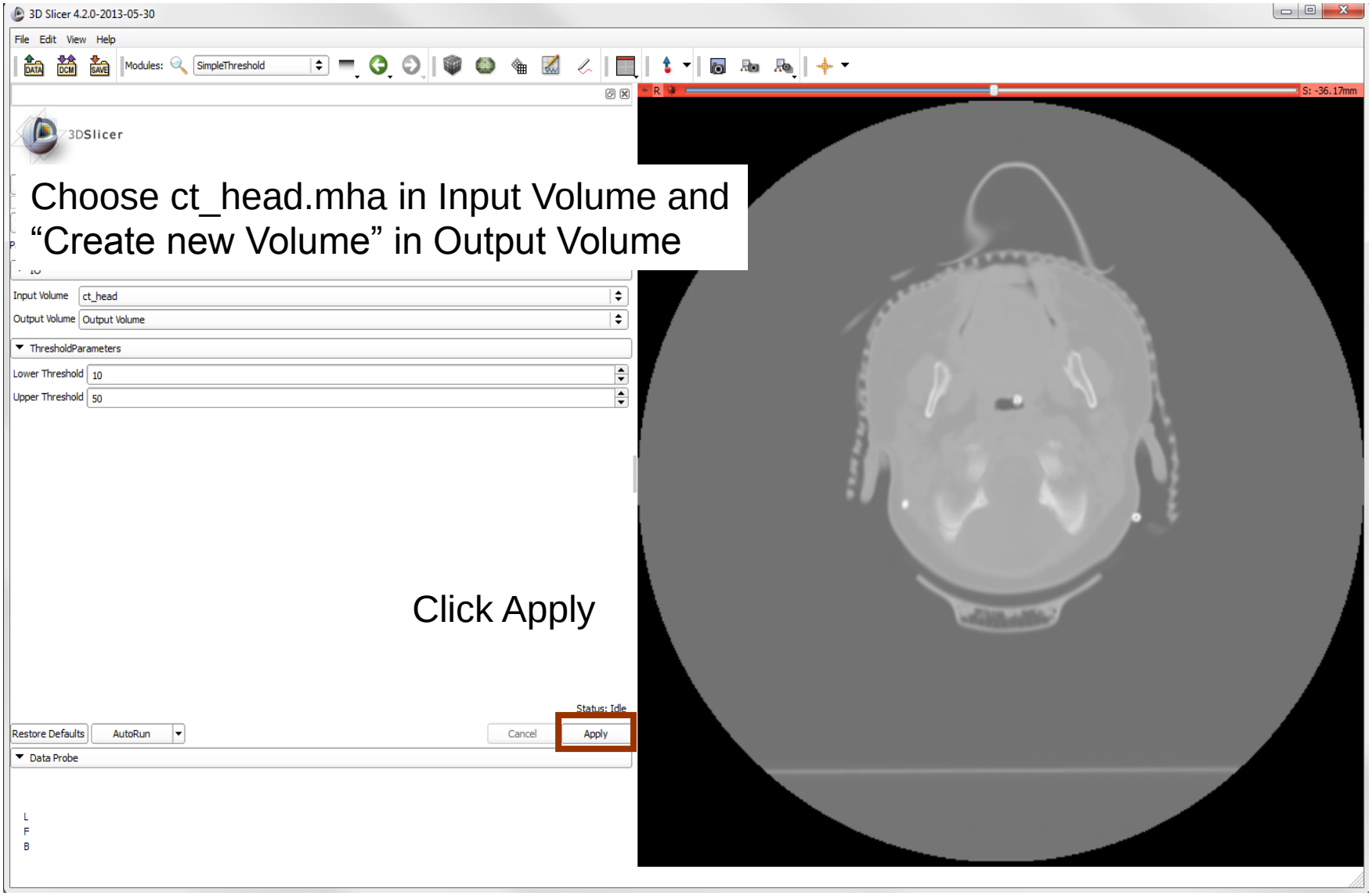
```
typedef itk::BinaryThresholdImageFilter<
    InputImageType, OutputImageType>  FilterType;
typename FilterType::Pointer filter = FilterType::New();
filter->SetLowerThreshold(lowerThreshold);
```

Compile the module

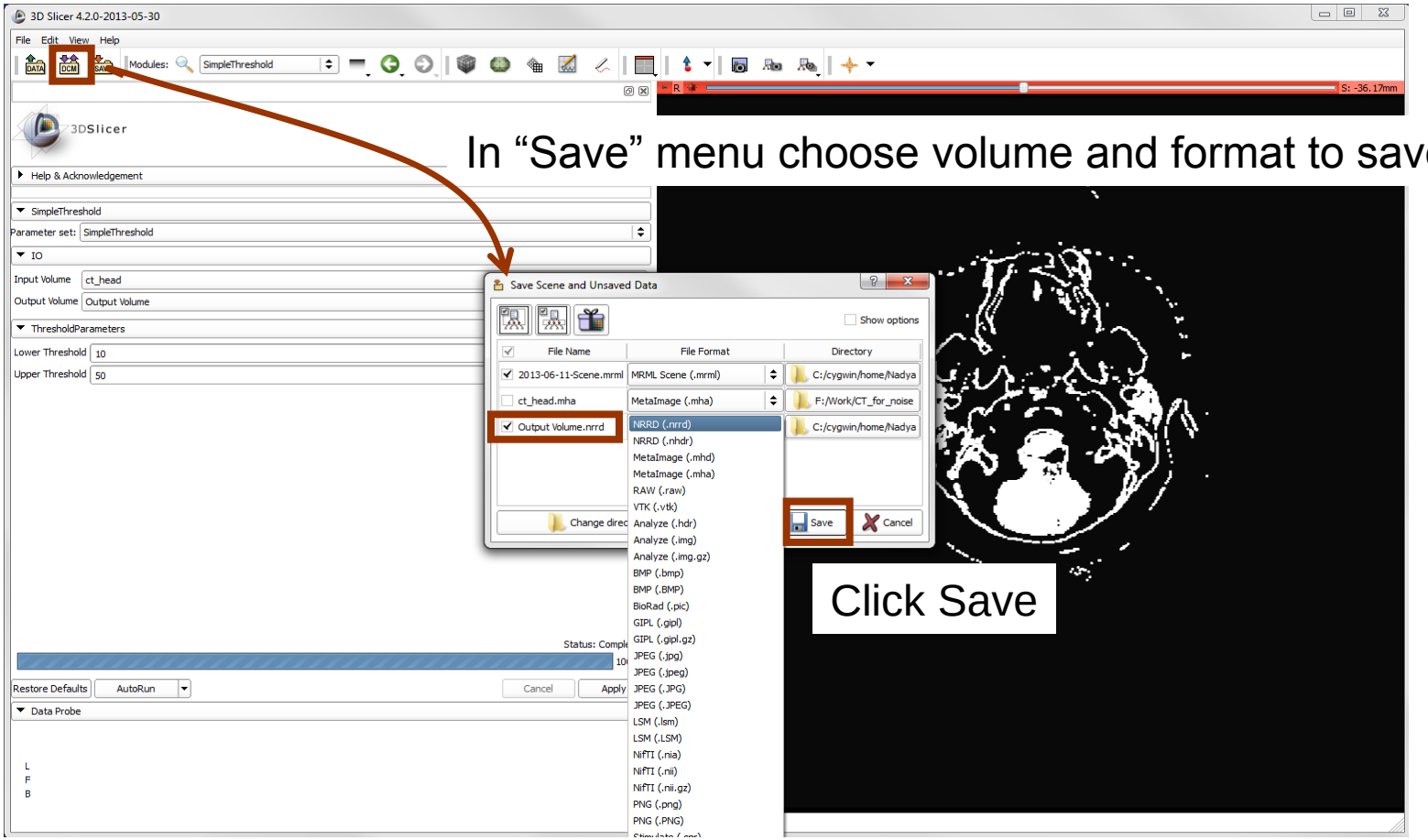
- From Slicer source directory run the command: ./Utilities/Scripts/ModuleWizard.py --template ../SimpleThreshold --target ../SimpleThreshold SimpleThreshold
- Build extension (pp 9,10)
- Start Slicer
- Set the path to SimpleThreshold in the Application Settings (pp 11,12)
- Restart Slicer

Loading data




3D Slicer 4.2.0-2013-05-30
File Edit View Help
Modules: SimpleThreshold
3DSlicer
Choose ct_head.mha in Input Volume and
“Create new Volume” in Output Volume
Input Volume ct_head
Output Volume Output Volume
ThresholdParameters
Lower Threshold 10
Upper Threshold 50
Click Apply
Status: Idle
Restore Defaults AutoRun Cancel Apply
Data Probe
L
F
B
S: -36.17mm

Saving data


3D Slicer 4.2.0-2013-05-30
File Edit View Help
Modules: SimpleThreshold
3DSlicer
Help & Acknowledgement
SimpleThreshold
Parameter set: SimpleThreshold
IO
Input Volume ct_head
Output Volume Output Volume
ThresholdParameters
Lower Threshold 10
Upper Threshold 50
Restore Defaults AutoRun Cancel Apply
Data Probe
In “Save” menu choose volume and format to save
Save Scene and Unsaved Data
Show options
File Name
File Format
Directory
2013-06-11-Scene.mrml
MRML Scene (.mrml)
C:/cygwin/home/Nadya
ct_head.mha
MetaImage (.mha)
F:/Work/CT_for_noise
Output Volume.nrrd
NRRD (.nrrd)
C:/cygwin/home/Nadya
NRRD (.nhdr)
MetaImage (.mhd)
MetaImage (.mha)
RAW (.raw)
VTK (.vtk)
Analyze (.hdr)
Analyze (.img)
Analyze (.img.gz)
BMP (.bmp)
BMP (.BMP)
BioRad (.pic)
GIPL (.gipl)
GIPL (.gipl.gz)
JPEG (.jpg)
JPEG (.jpeg)
JPEG (.JPG)
JPEG (.JPEG)
LSM (.lsm)
LSM (.LSM)
NifTI (.nia)
NifTI (.nii)
NifTI (.nii.gz)
PNG (.png)
PNG (.PNG)
Change direc
Save
Cancel
Click Save

Contact information

If you have questions please contact Greg Sharp at

E-mail: gcsharp@partners.org
Tel: (617) 724 3866

Acknowledgments

National Alliance for Medical Image Computing

NIH U54EB005149

National Institutes of Health

NIH / NCI 6-PO1 CA 21239

Federal share of program income earned by MGH on C06CA059267

Progetto Rocca Foundation

A collaboration between MIT and Politecnico di Milano

Bridging Academia and Industry Using Open-Source Software

Stephen R. Aylward, Ph.D.
Senior Director of Operations and Medical Research

Kitware

The professor...

...and his students

Bob (the “perfect” student)

Bob graduates

I cannot find Bob’s code.

Bob’s code won’t compile

I cannot replicate Bob’s results

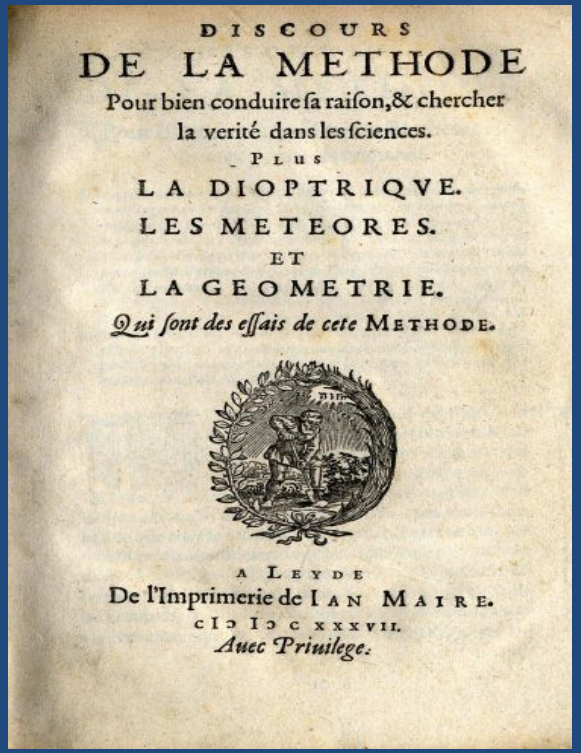
I emailed Bob a question, but he hasn’t responded

The open-source “attitude” is the answer

Focus on writing code
that other people can use
and will want to use.

Open Science

Science must be reproducible



DISCOURS
DE LA METHODE
Pour bien conduire ſa raiſon,& chercher
la verité dans les ſciences.
PLUS
LA DIOPTRIQVE.
LES METEORES.
ET
LA GEOMETRIE.
Qui ſont des eſſais de cete METHODE.

A LEYDE
De l'Imprimerie de IAN MAIRE.
cIↃ IↃ c xxxvii.
Auec Priuilege.

Discourse
on the (Scientific) Method
--
Descartes 1637

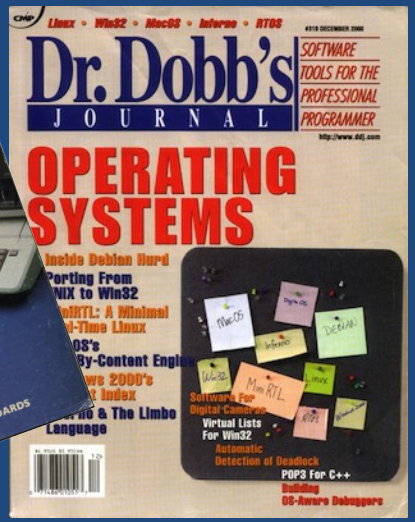
“DOUBT EVERYTHING and only believe in those things that are evidently true (reproducible)”

Outline

- History
- Methods
- Bridges and research

History of open source

- 1970’s Open source was the norm
- 1980’s Almost all software was proprietary



History of open source

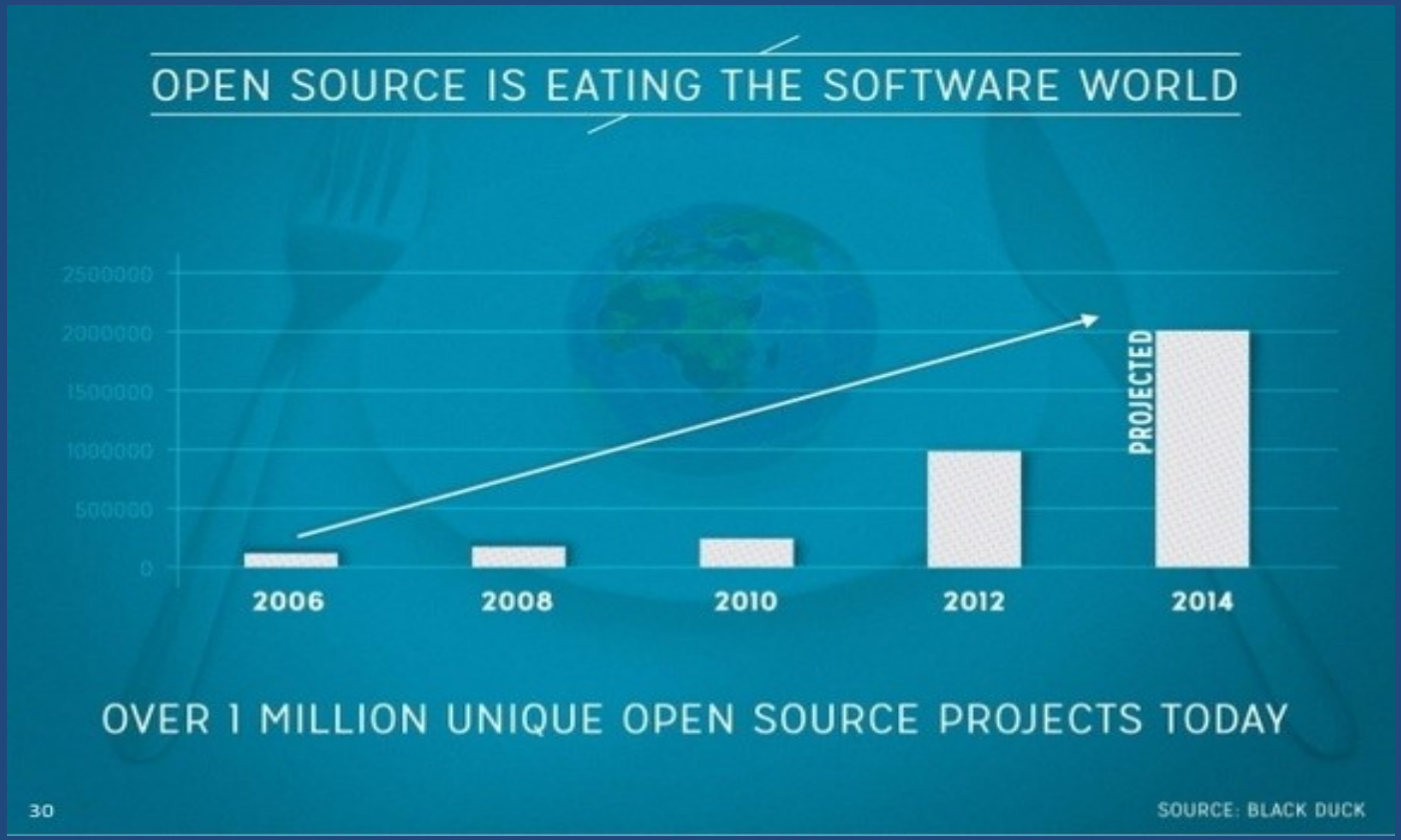
- 1983 GNU forced cooperative software development

“Copyleft”

History of open source

- Now Open source permeates academia and industry
  - BSD-style license
  - Platforms: Android, Linux
  - Tools: MySQL, CMake
  - Technologies: ITK, VTK, Slicer

History of open source



Over 80% of commercial software contains open source.

History of open source

VTK, ITK, CMake, Slicer, ParaView

= 130,000++ downloads per month

World of Warcraft

Excuses

“I’d rather spend my time on things that will help me…

publish

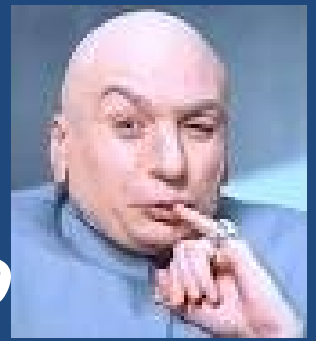
finish my dissertation

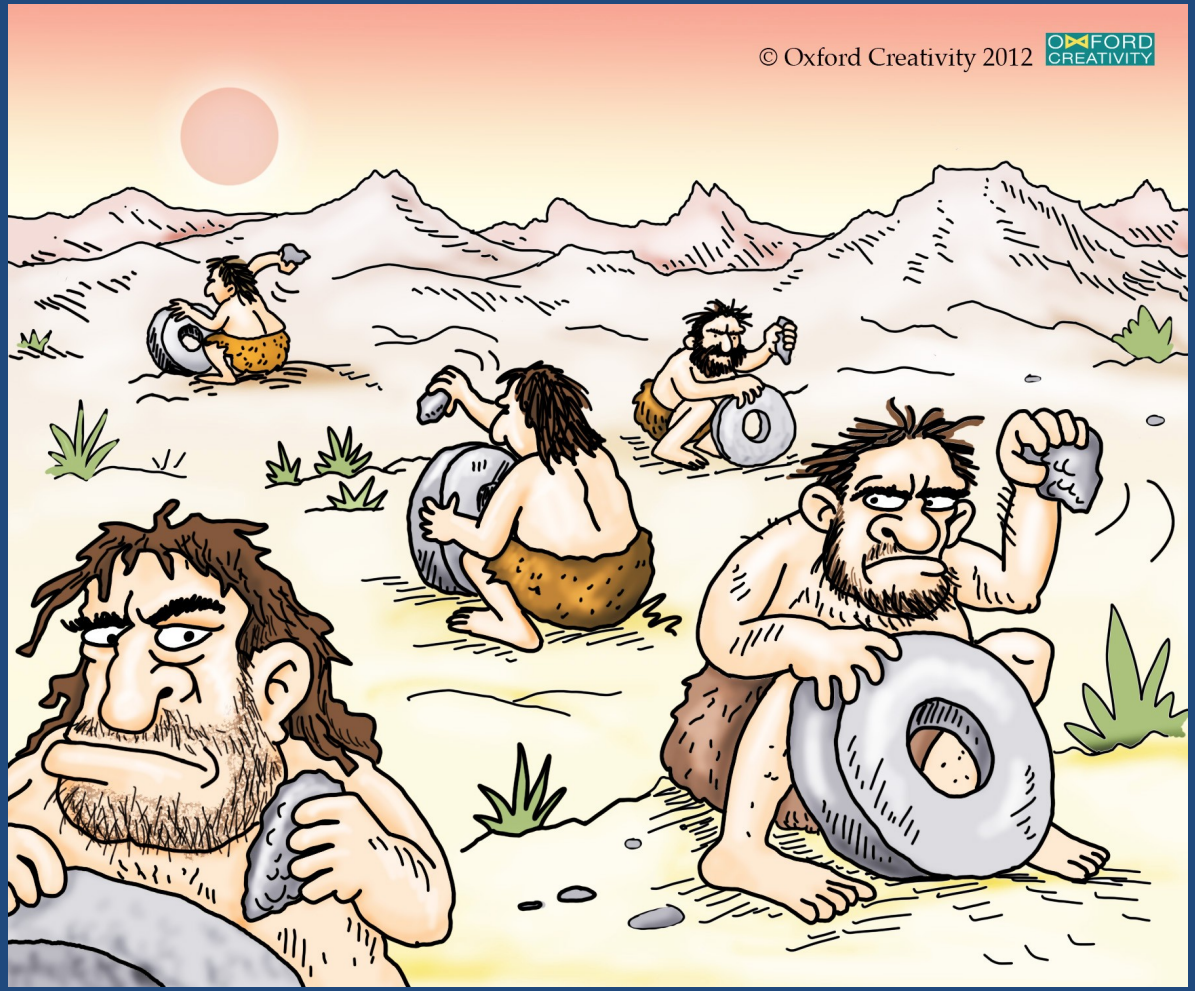
write a proposal

start my company

sell my company

Make $1,000,000




© Oxford Creativity 2012
OXFORD CREATIVITY

The open-source “attitude” is the answer

“Who ever has the most ideas stolen, wins.”

-- Dr. Fred Brooks

Impact

- An Object-Oriented Approach To 3D Graphics (Schroeder)
  - 3,065 Citations
- The ITK Software Guide (Yoo)
  - 1,039 Citations
- Object-oriented modeling and design (Lorensen)
  - 10,089

3D Slicer: 179,000 downloads

Failure of Reproducibility

Nature *(March 2012)*

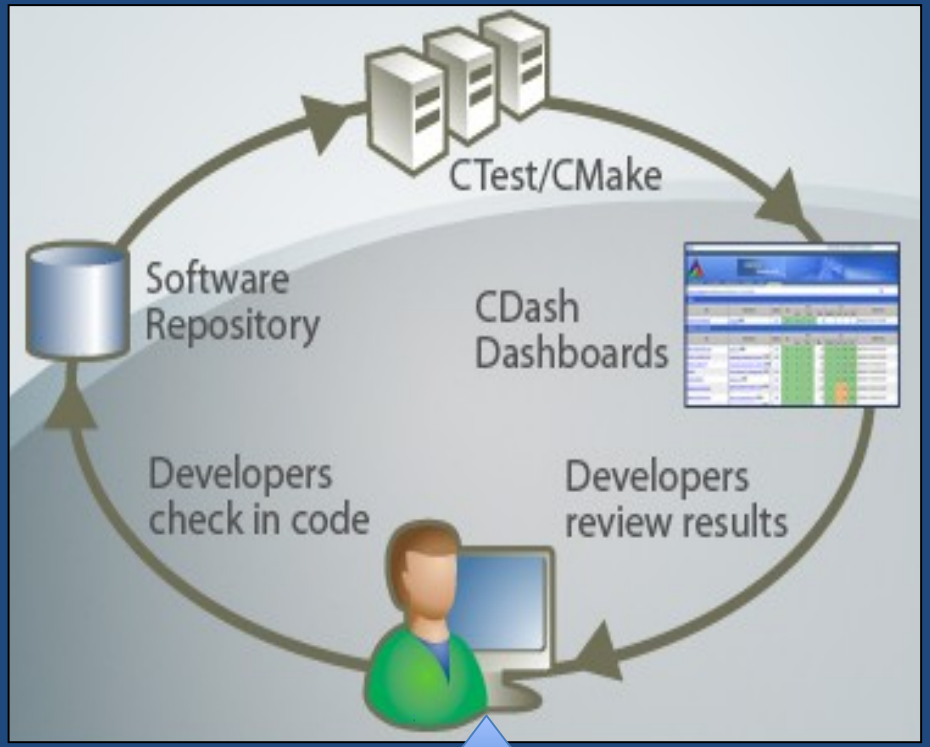
Glenn Begley, head of cancer research at pharma giant Amgen

Lee M. Ellis, cancer researcher at the University of Texas

“During a decade as head of global cancer research at Amgen, Glenn Begley identified 53 'landmark' publications — papers in top journals, from reputable labs — for his team to reproduce. Begley sought to double-check the findings before trying to build on them for drug development. Result: 47 of the 53 could not be replicated.”

Methods of Open Source

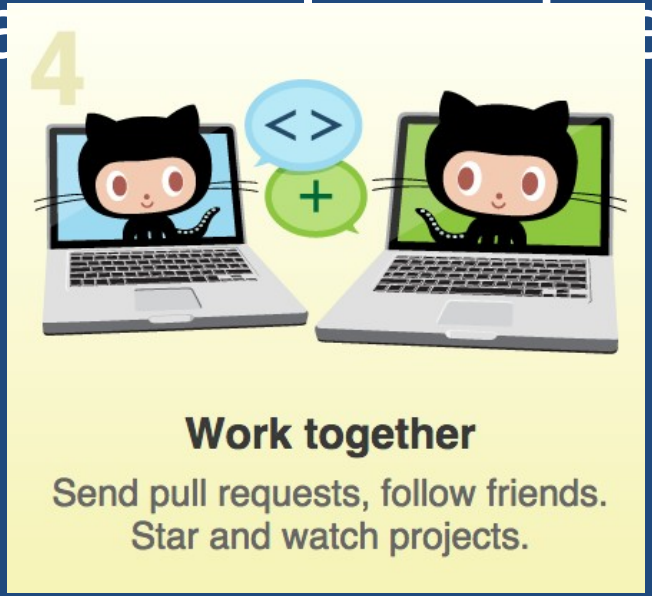
High-quality software processes...



Tools for the Open-Source Attitude

GITHUB as a software repository...

"Relea elease often."

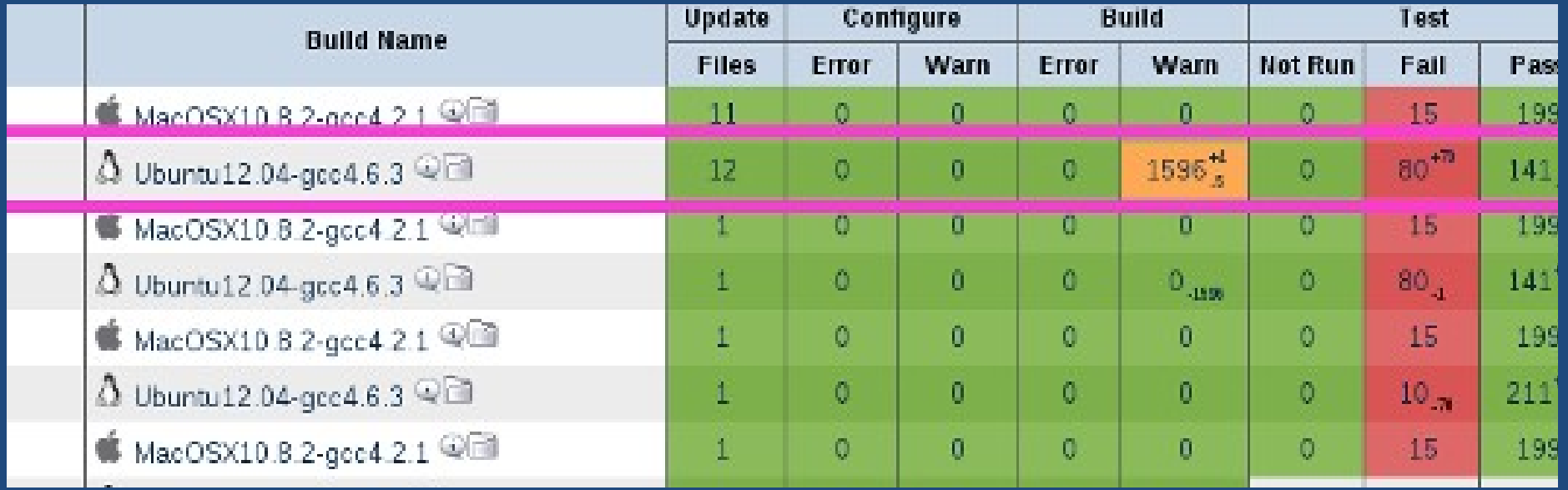


Tools for the Open-Source Attitude

"Test driven development."

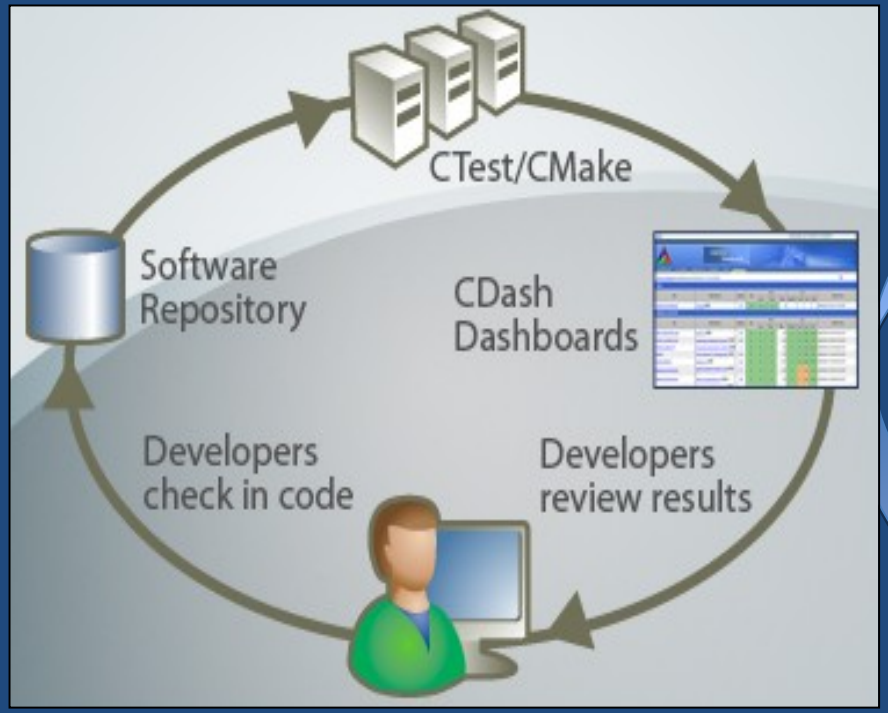
"Continuous reporting of results."

CMake & CDash



Build Name	Update	Configure		Build		Test		
	Files	Error	Warn	Error	Warn	Not Run	Fail	Pass
MacOSX10.8.2-gcc4.2.1	11	0	0	0	0	0	15	199
Ubuntu12.04-gcc4.6.3	12	0	0	0	1596 +1 -5	0	80 +73	141
MacOSX10.8.2-gcc4.2.1	1	0	0	0	0	0	15	199
Ubuntu12.04-gcc4.6.3	1	0	0	0	0 -1529	0	80 -1	141
MacOSX10.8.2-gcc4.2.1	1	0	0	0	0	0	15	199
Ubuntu12.04-gcc4.6.3	1	0	0	0	0	0	10 -70	211
MacOSX10.8.2-gcc4.2.1	1	0	0	0	0	0	15	199


CTest/CMake
Software Repository
CDash Dashboards
Developers check in code
Developers review results



3DSlicer

179,000 downloads

Industry and academia are converging…

- Open source is a conduit from academia to industry: ideas and people
- Reduces development time, expertise, risks, and costs
- No recurring licenses
- Lifecycle from prototype to production

Proprietary applications and methods

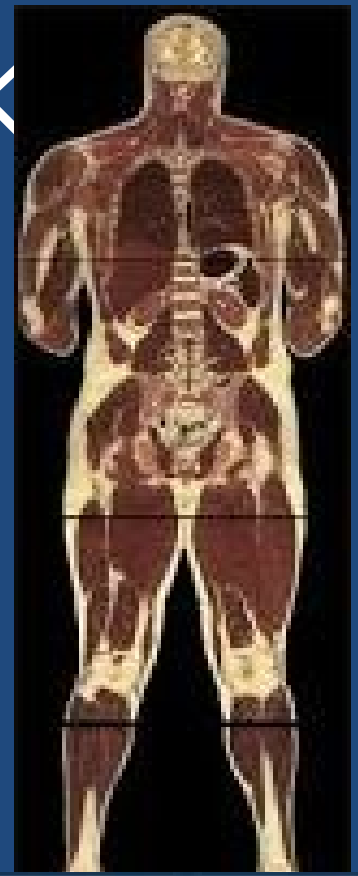
Open source

Proprietary hardware: MRI, Phone

The Insight Toolkit (ITK

1999: NLM organized $13.5M for ITK

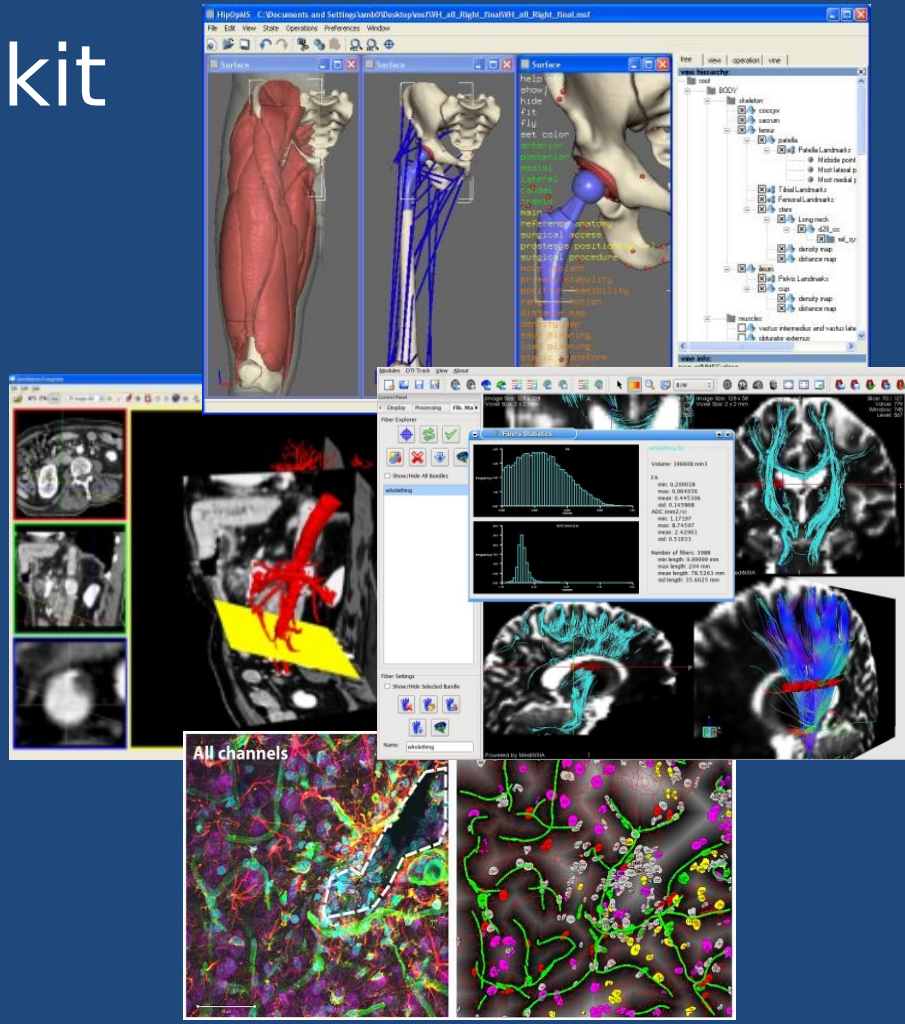
- GE Research / Harvard
- Kitware, Inc.
- Insightful / UPenn
- UNC / UPitt
- UPenn / Columbia
- University of Utah
- Mayo Clinic
- Harvard / Brigham and Women’s Hospital
- Cognita, Inc.
- Imperial and King’s College London
- University of Iowa
- Georgetown University
- Carnegie Mellon University



The Insight Toolkit

2014: ITK is the dominant toolkit for medical image segmentation and registration

- 1.5M lines of code
- Estimated at 452 years of effort



Thank you...

Focus on writing code
that other people can use
and will want to use.