SimpleITK Tutorial

Image processing for mere mortals

Insight Software Consortium

Sept 23, 2011

What this Tutorial is about

- Provide working knowledge of the SimpleITK platform

Program

- Virtual Machines Preparation (10min)
- Introduction (15min)
- Basic Tutorials I (45min)
- Short Break (10min)
- Basic Tutorials II (45min)
- Coffee Break (30min)
- Intermediate Tutorials (45min)
- Short Break (10min)
- Advanced Topics (40min)
- Wrap-up (10min)

Virtual Machine Preparation

Virtual Machines Preparation

- Get DVD / USB Memory Stick
- Install VirtualBox from it
- Import the VirtualMachine file
- Boot the Virtual Machine
- Log in
- Get familiar with directories

Media Content

Directories and Files

- VirtualBoxInstallers
  - VirtualBox-4.0.8-71778-OSX.dmg (Mac)
  - VirtualBox-4.0.8-71778-Win.exe (Windows)
  - (Ubuntu Linux)
    - virtualbox-4.0_4.0.8-71778 Ubuntu lucid_amd64.deb
    - virtualbox-4.0_4.0.8-71778 Ubuntu lucid_i386.deb
    - virtualbox-4.0_4.0.8-71778 Ubuntu maverick_amd64.deb
    - virtualbox-4.0_4.0.8-71778 Ubuntu maverick_i386.deb
    - virtualbox-4.0_4.0.8-71778 Ubuntu natty_amd64.deb
    - virtualbox-4.0_4.0.8-71778 Ubuntu natty_i386.deb
    - ...
- VirtualMachine
  - SimpleITK-disk1.vmdk
  - SimpleITK.ovf

Install VirtualBox

- Select the installer for your platform
- Run it

Alternative Linux Installation

- You can also install VirtualBox by doing:
- sudo apt-get install virtualbox-ose-qt

Importing the Virtual Machine

- Run VirtualBox
- In `File` Menu select `Import Appliance`
- Provide the filename in the DVD / USB stick `VirtualMachine/SimpleITK.ovf`
- A progress bar will appear, and when it finishes you should see:

Now, on to the tutorial...

Introductions

- Daniel Blezek, Ph.D., Mayo Clinic
- Luis Ibáñez, Ph.D., Kitware
- Hans Johnson, Ph.D., University of Iowa
- Bradley Lowekamp, Lockheed Martin (National Library of Medicine)

Tutorial Goals

- Gentle introduction to ITK
- Introduce SimpleITK
- Provide hands-on experience
- Problem solving, not direction following
- ...but please follow directions!

ITK Overview

How many are familiar with ITK?

Ever seen code like this?

```
1  // Setup image types.
2  typedef    float InputPixelType;
3  typedef    float OutputPixelType;
4  typedef itk::Image<InputPixelType, 2> InputImageType;
5  typedef itk::Image<OutputPixelType,2> OutputImageType;
6  // Filter type
7  typedef itk::DiscreteGaussianImageFilter<
8                 InputImageType, OutputImageType >
9          FilterType;
10 // Create a filter
11 FilterType::Pointer filter = FilterType::New();
12 // Create the pipeline
13 filter->SetInput( reader->GetOutput() );
14 filter->SetVariance( 1.0 );
15 filter->SetMaximumKernelWidth( 5 );
16 filter->Update();
17 OutputImageType::Pointer blurred = filter->GetOutput();
```

What if you could write this?

```
1 import SimpleITK
2 input = SimpleITK.ReadImage ( filename )
3 output = SimpleITK.DiscreteGaussianFilter( input, 1.0, 5 )
```

What if you could write this?

```
1 import SimpleITK
2 input = SimpleITK.ReadImage ( filename )
3 output = SimpleITK.DiscreteGaussianFilter( input, 1.0, 5 )
```

We are here to tell you that you can...

Goals of SimpleITK

- Be an "on-ramp" for ITK
- Simplify the use of ITK by
  - Providing a templateless, typeless layer for ITK in C++
  - Providing wrappings in scripting languages
  - Providing access to most ITK algorithms

SimpleITK Architectural Overview

- Conceptually, SimpleITK is an application library built on ITK
- All functionality provided by ITK
- Components:
  - Template expansion system
  - C++ library
  - Small SWIG definition (more details later)
  - "Glue" code for several scripting languages
  - Some language utilities
- Open Source, Apache licensed project (`http://www.opensource.org/licenses/apache2.0.php`)
- Hosted by GitHub (`https://github.com/SimpleITK/SimpleITK`)

Templates in ITK

```
typedef unsigned char PixelType;
enum {ImageDimension = 2};
typedef itk::Image<PixelType,ImageDimension> ImageType;
typedef itk::Vector<float,ImageDimension> VectorType;
typedef itk::Image<VectorType,ImageDimension> FieldType;
typedef itk::Image<VectorType::ValueType,ImageDimension> FloatImageType;
typedef ImageType::IndexType  IndexType;
typedef ImageType::SizeType   SizeType;
typedef ImageType::RegionType RegionType;
typedef itk::MultiResolutionPDEDeformableRegistration
   <ImageType, ImageType, FieldType> RegistrationType;
```

Template Freedom

```
using itk::simple;
// Read the image file
ImageFileReader reader;
reader.SetFileName ( "/my/fancy/file.nrrd" );
Image image = reader.Execute();

// This filters perform a gaussian bluring with sigma in
// physical space. The output image will be of real type.
SmoothingRecursiveGaussianImageFilter gaussian;
gaussian.SetSigma ( 2.0 );
Image blurredImage = gaussian.Execute ( image );
```

Programming Models

- Object oriented
- Function oriented

More about this later

Wrapping

Transformed by SWIG

- Parses header and "interface" files
- Automatically creates scripting "glue"
- Wrappings available for:
  - Python
  - Java
  - C#
  - Tcl, Lua, Ruby
  - Others as requested...

Basic Tutorial

Virtual Machine Check

How are we doing with the Virtual Box images?

Ubuntu Introduction

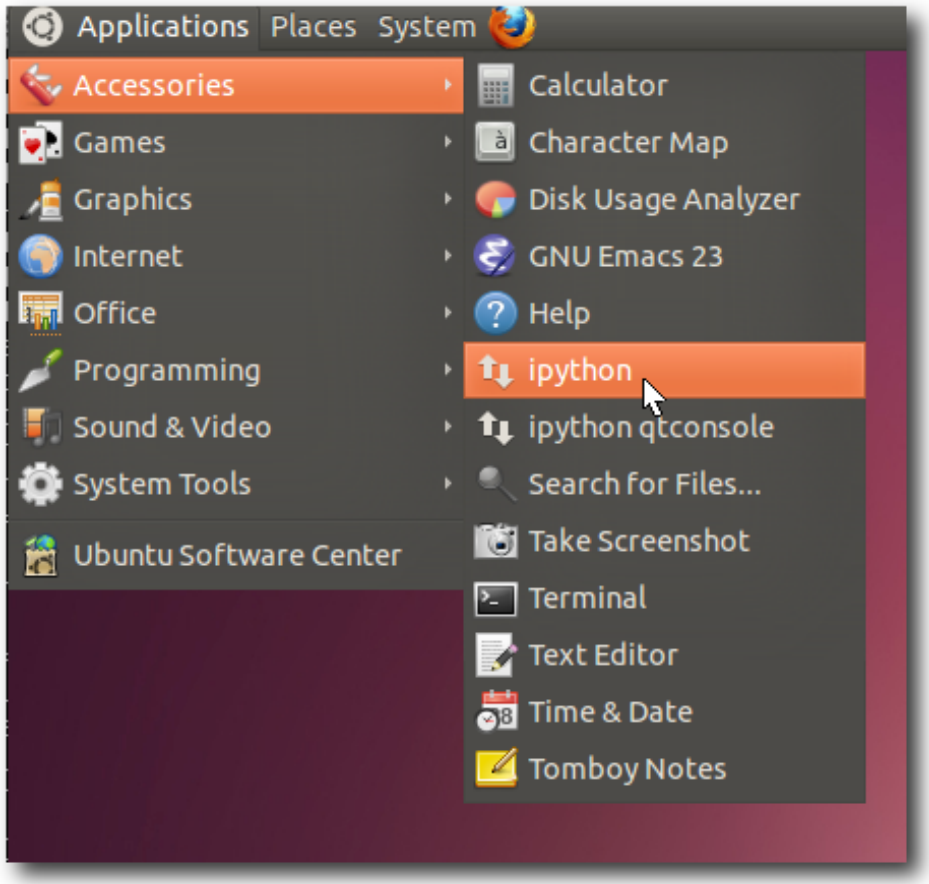
- User experience oriented version of Linux
- Familiar desktop paradigm
- TODO: finish after image is ready

Ubuntu - Hands On

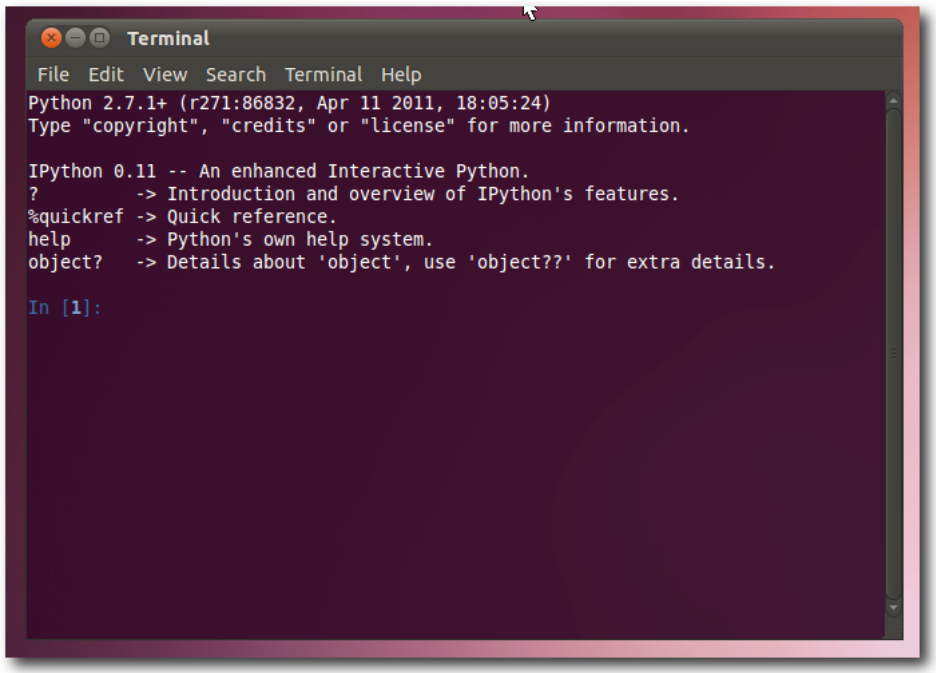
- TODO: what do we need to talk about here?

Just Enough Python to be Dangerous

- Start a terminal
- Run iPython

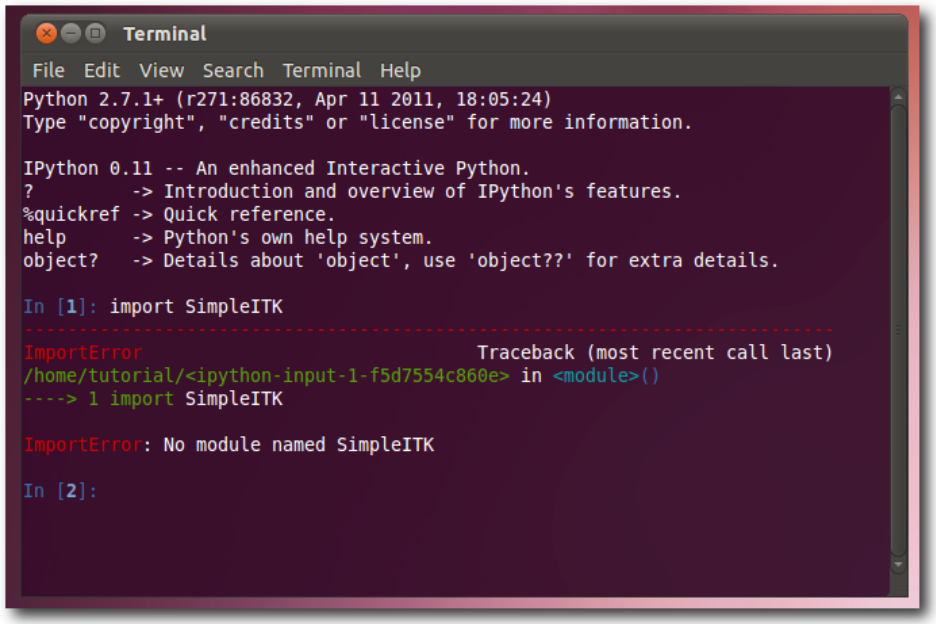


Just Enough Python to be Dangerous



Just Enough Python to be Dangerous

Import the SimpleITK package



What just happened?

Just Enough Python to be Dangerous

Need to tell iPython where to find SimpleITK
See /home/tutorial/.ipython/ipython.py

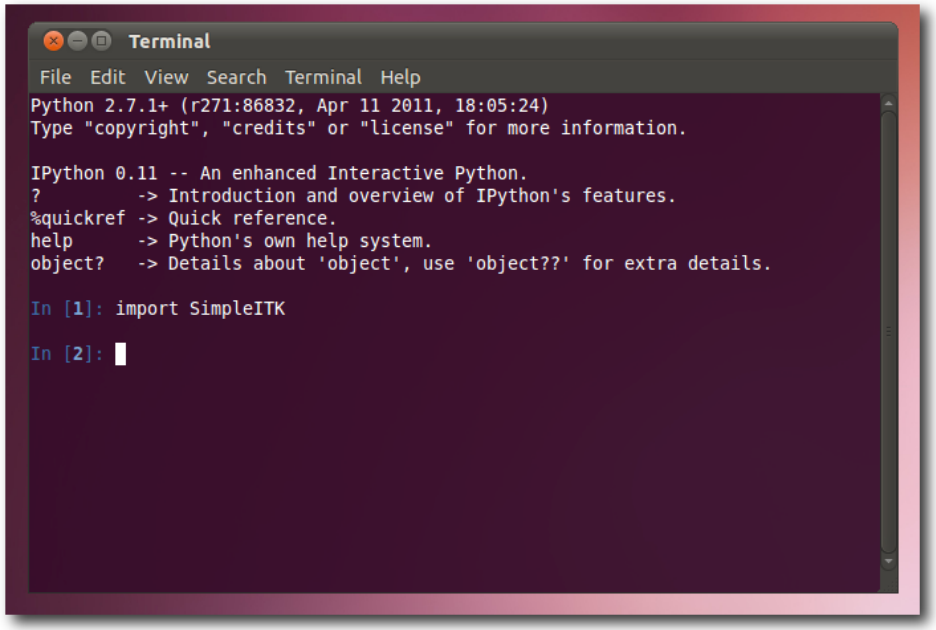


Image Class

- Creation
- Number of dimensions, size, origin, spacing
- Pixel access

Back to iPython

What just happened?

```
# Create an image
image = SimpleITK.Image ( 256, 256, 256,
                          SimpleITK.sitkInt16 );
# How about 2d?
twoD = SimpleITK.Image ( 64, 64,
                         SimpleITK.sitkFloat32 )
```

- SimpleITK is the module
- Image is the constructor for the Image class
- Height, width, depth (omit depth for 2D images)
- Datatype (more on this later)

Back to iPython

What just happened?

```
# Addressing pixels
image.GetPixel ( 0, 0, 0 )
image.SetPixel ( 0, 0, 0, 1 )
image.GetPixel ( 0, 0, 0 )
```

- Get the voxel value at [0,0,0]?
- Hmm, I don't like it, so set to 1
- What is the value at [0,0,0] now?

Back to iPython

What just happened?

```
1 # Addressing pixels
2 image[0,0,0]
3 image[0,0,0] = 10
4 image[0,0,0]
```

Without warning, we sprinkled syntatic sugar on you!

- image[0,0,0] is shorthand for Image.GetPixel(0,0,0)
- image[0,0,0] = 10 is shorthand for Image.SetPixel(0,0,0,1)

Summary

- Images are created using SimpleITK.Image (w, h, d, Type)
- Images can be 2- or 3-dimensional
- Images can describe themselves
- Images have simple pixel accessors

Questions before we move on?

Memory Management

Images

Images...

- usually allocated on the stack
- are copy-on-write
- use internal smart-pointers

Image Memory Management

```
image = SimpleITK.Image ( 32, 32, 32, SimpleITK.sitkInt16 )
print image
...
Image (0x94f2d98)
  Reference Count: 1
...
# Clone image
b = SimpleITK.Image ( image )
print image
...
Image (0x94f2d98)
  Reference Count: 2
...
print b
...
Image (0x94f2d98)
```

Image Memory Management

```
print b
...
Image (0x94f2d98)
...
b[0,0,0] = 1
print b
...
Image (0x94f4cb0)
  Reference Count: 1
...
print image
...
Image (0x94f2d98)
  Reference Count: 1
...
```

Filters

Filters...

- usually allocated on the stack
- tend to clean up after themselves
- do not hold on to images

...more on this later...

Memory Management Strategies

C++...

- No need for explicit management
- Let images clean up after themselves
- Let filters clean up after themselves

Wrapped...

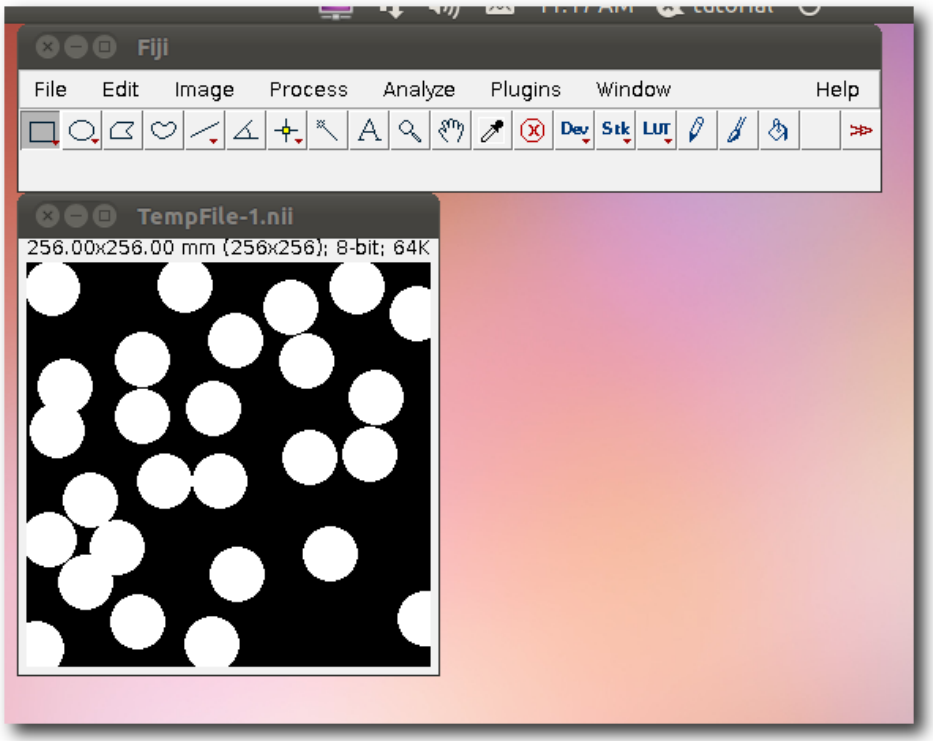
- Utilize language-specific memory management
- Automatic in Python, Java, Ruby, C#, Lua
- More manual in Tcl

Input/Output

Read/Write/Display Images

Back to iPython

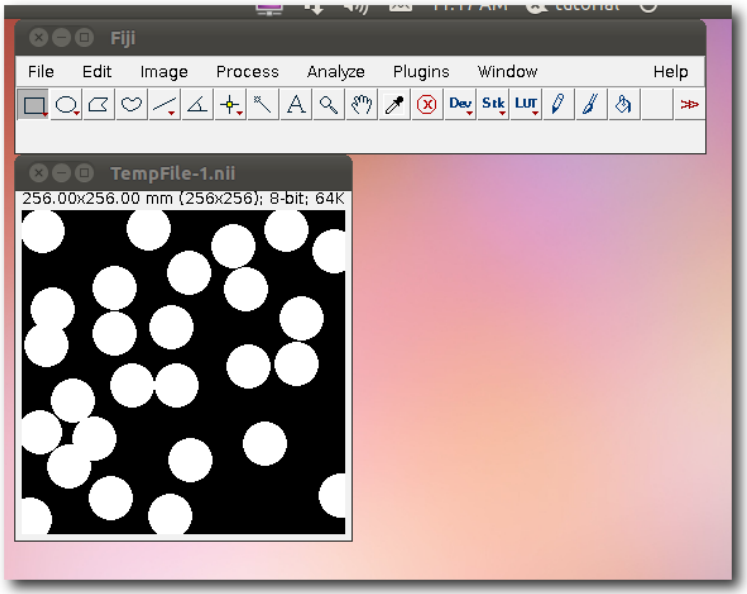
Display



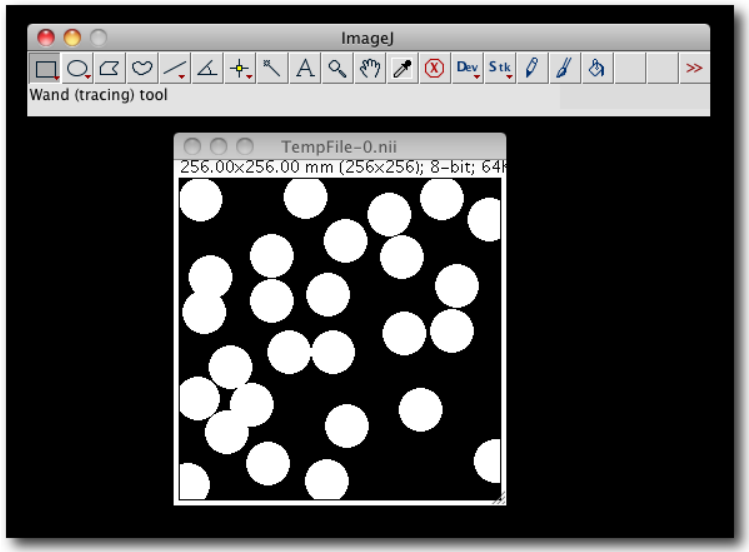
What just happened?

```
1 # What's the image look like?
2 SimpleITK.Show ( image )
```

Display



Ubuntu



Mac

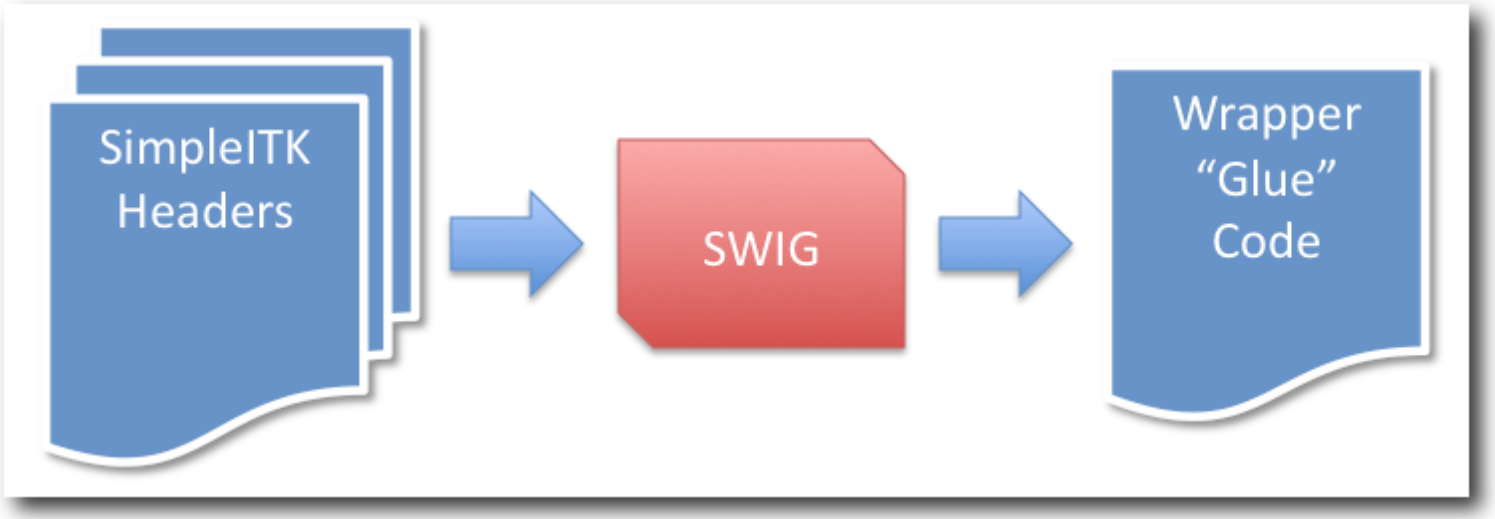
- ImageJ/Fiji used for display
- SimpleITK looks in most likely location for ImageJ
- Image written in Nifti format
- Need to install Nifti plugin for ImageJ
- `http://rsbweb.nih.gov/ij/plugins/nifti.html`

Break

Basic Tutorial 2

Wrapped Languages

Wrapping Process



- SimpleITK headers are constructed for wrapping
- SWIG is an open source package
  - Parses C/C++ code, produces "glue" code
  - Well supported, covers 10+ languages
- Main languages: Python, Java, C#
- Also supported: Tcl, Lua, R

Object Paradigm (Python)

The paradigms translate to the wrapped languages (C++ $\rightarrow$ Python)

```cpp
#include <SimpleITK.h>
using itk::simple;
...
// Create a smoothing filter
SmoothingRecursiveGaussianImageFilter gaussian;
// Set a parameter
gaussian.SetSigma ( 2.0 );
// "Execute" the Filter
Image blurredImage = gaussian.Execute ( image );
```

```python
from SimpleITK import *
# Create a smoothing filter
SmoothingRecursiveGaussianImageFilter gaussian
# Set a parameter
gaussian.SetSigma ( 2.0 );
# "Execute" the Filter
blurredImage = gaussian.Execute ( image );
```

Object Paradigm (Java)

```
1  import SimpleITK.*;
2  ...
3
4  // Create a smoothing filter
5  SmoothingRecursiveGaussianImageFilter gaussian =
6        new SmoothingRecursiveGaussianImageFilter();
7
8  // Set a parameter
9  gaussian.SetSigma ( 2.0 );
10
11 // "Execute" the Filter
12 Image blurredImage = gaussian.Execute ( image );
```

Object Paradigm (C#)

```
1  using System;
2  using itk.simple;
3  ...
4  // Create a smoothing filter
5  SmoothingRecursiveGaussianImageFilter gaussian =
6        new SmoothingRecursiveGaussianImageFilter();
7
8  // Set a parameter
9  gaussian.SetSigma ( 2.0 );
10
11 // "Execute" the Filter
12 Image blurredImage = gaussian.Execute ( image );
```

Note on the Tutorial

- Most examples will be Python
- Obvious translation to other languages
- C++ usage (generally) obvious

Hands On

What just happened?

```
# Simple smoothing
smooth = SimpleITK.SmoothingRecursiveGaussian ( image, 2.0 )
SimpleITK.Show ( SimpleITK.Subtract ( image, smooth ) )
...
RuntimeError: Exception thrown in SimpleITK Subtract: ...
sitk::ERROR: Both images for SubtractImageFilter don't match type or dimension!
...
```

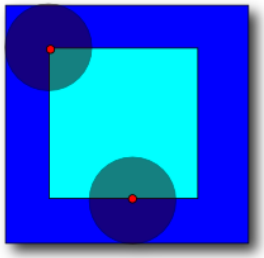
- The output of SmoothingRecursiveGaussian is of type float
- The input image is signed short
- Most SimpleITK filters with 2 inputs require the same type
- Let's fix the problem

Introducing Cast

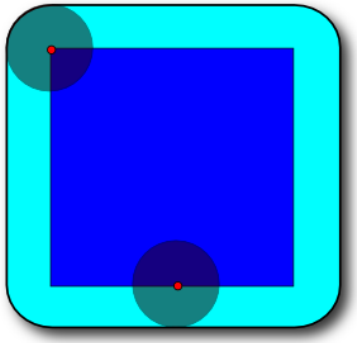
```
# Much better
smooth = SimpleITK.Cast ( smooth, image.GetPixelIDValue() )
print image.GetPixelIDTypeAsString()
print smooth.GetPixelIDTypeAsString()
SimpleITK.Show ( SimpleITK.Subtract ( image, smooth ) )
```

Morphology

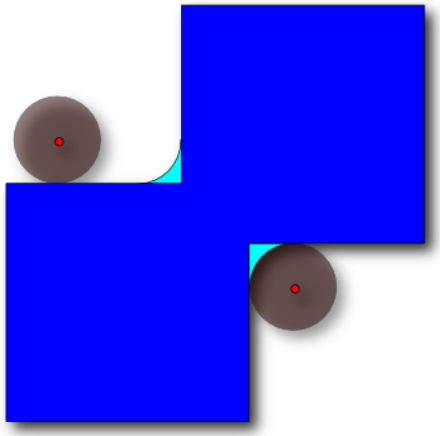
Operators



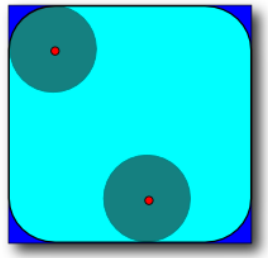
Erosion



Dilation



Closing



Opening

Images from
`http://en.wikipedia.org/wiki/Mathematical_morphology`

Morphology in Action

Back to iPython

Label Maps

Break

Let's take a break

True utility of SimpleITK

- Interacting with data

Image Statistics

Label Statistics

Pixel-wise Operators

Table: SimpleITK Pixel-wise Operators

Operator	Description	Usage[†]
$+$	Addition	$A + B$, $s + A$, $s + B$
$-$	Subtraction	$A - B$, $s - A$, $s - B$
$*$	Multiplication	$A * B$, $s * A$, $s * B$
$/$	Division	A/B, s/A, B/s
&	Logical "and"	$A\&B$
|	Logical "or"	$A\|B$
	Logical "not"	A

[†] A and B are images (2D or 3D), s is a scalar

Masking

Edge Detection

Threshold-based Segmentation

Building SimpleITK in 5 Easy Commands

```
git clone --recursive https://github.com/SimpleITK/SimpleITK
( cd SimpleITK && git checkout va01 )
mkdir SimpleITK-build && cd SimpleITK-build
cmake ../SimpleITK/SuperBuild
make -j 5
```

More complete version

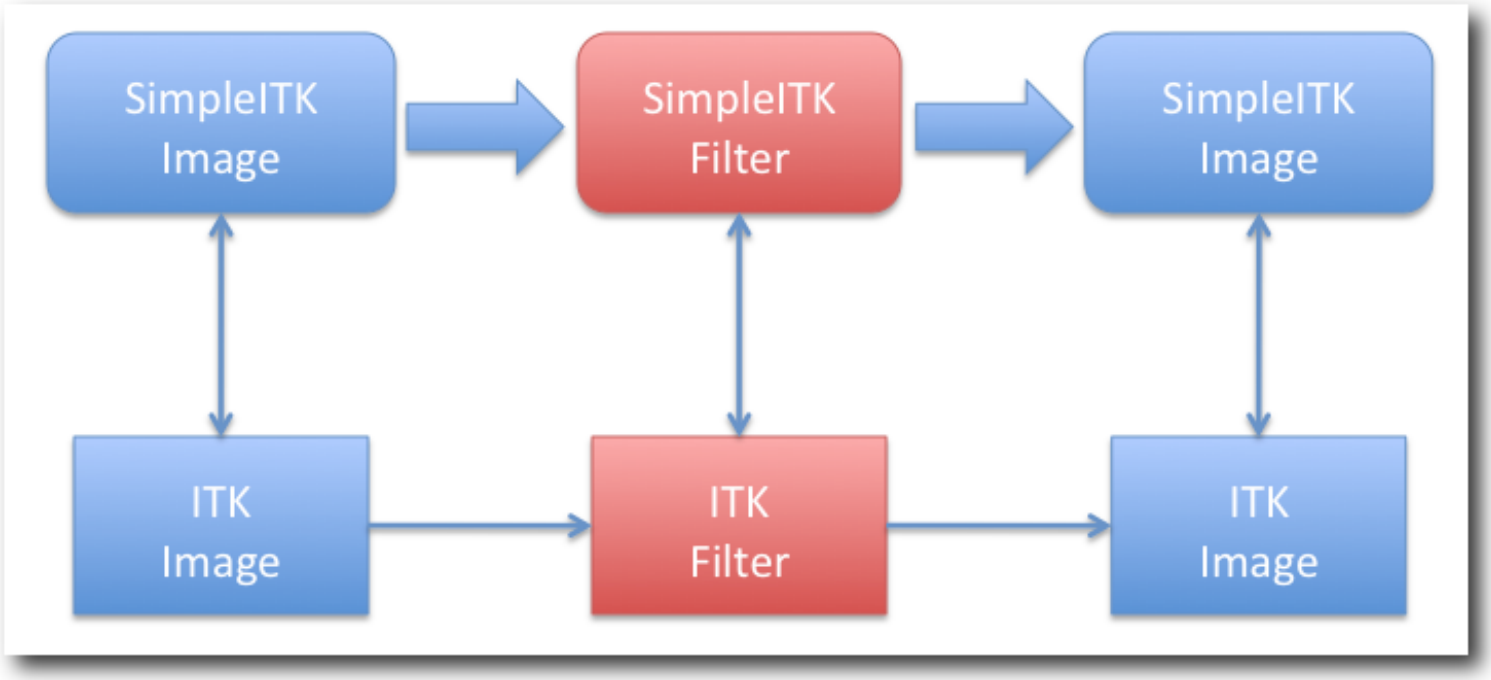
- Check out the code from GitHub (`https://github.com/SimpleITK/SimpleITK`)
- Run CMake (`http://www.cmake.org/`) using SimpleITK/SuperBuild as the source directory
- Build using your favorite compiler

Supported Platforms

- Windows: Visual Studio 10
- Windows: Visual Studio 9 (Requires TR1 service pack)
- Mac OSX: gcc 4.x
- Linux: gcc 4.x

SimpleITK Architecture

Filter Anatomy

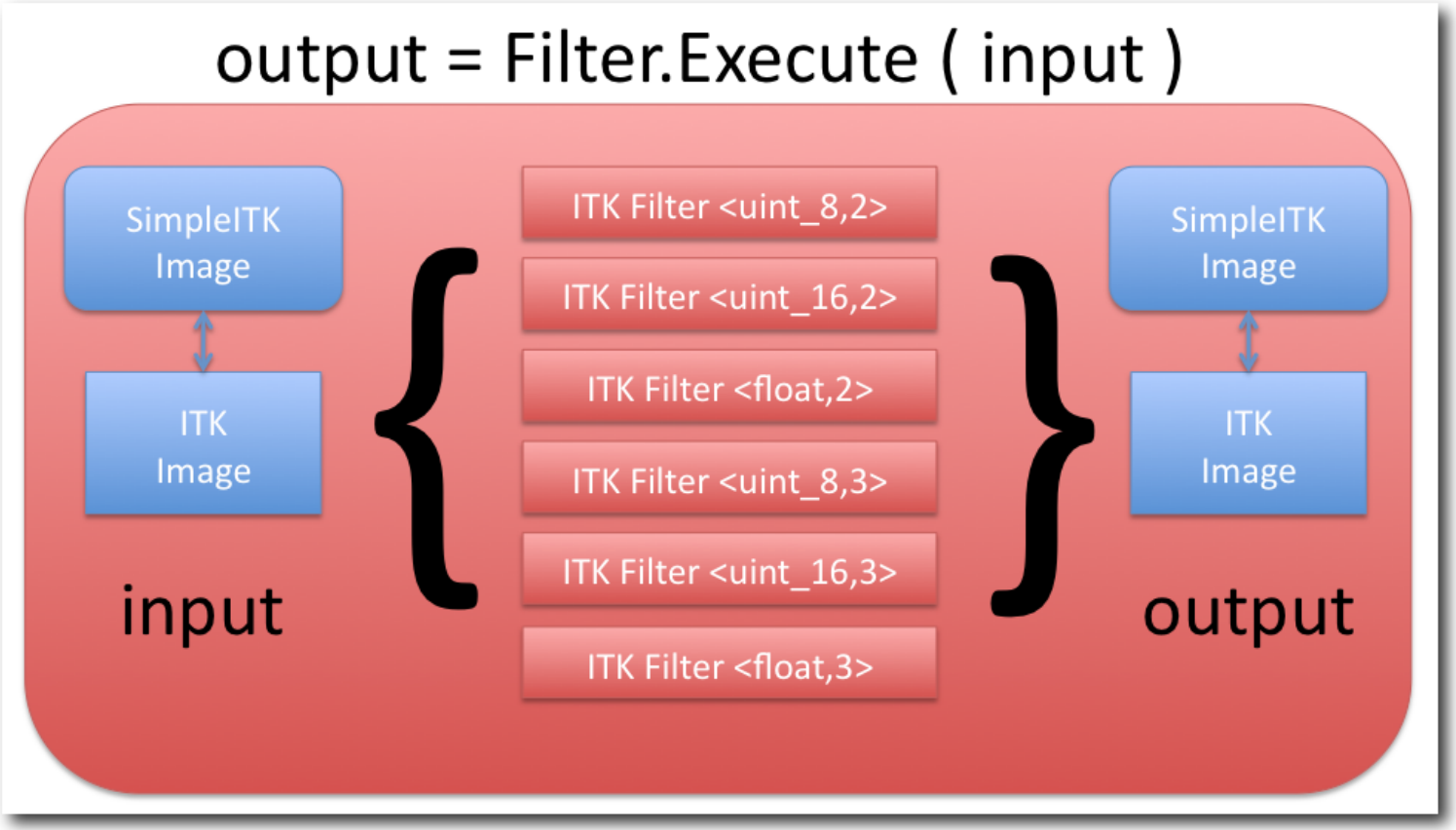


- SimpleITK filters create ITK filters
- Templated based on input type
- Output type is usually the same as input type
- Instantiated for many possible image types

Image and Filter Types

- Dimensions
  - 2 dimensional
  - 3 dimensional
- Scalar types
  - *int*8_*t*
  - *uint*8_*t*
  - *int*16_*t*
  - *uint*16_*t*
  - *int*32_*t*
  - *uint*32_*t*
  - *float*
  - *double*
  - *std* :: *complex* < *float* >
  - *std* :: *complex* < *double* >
- Vector Types
  - *int*8_*t*
  - *uint*8_*t*
  - *int*16_*t*
  - *uint*16_*t*
  - *float*
  - *double*
- Label Types
  - *uint*8_*t*
  - *uint*16_*t*
  - *uint*32_*t*

Filter Anatomy



- Filter interrogates *input*
- Instantiates proper ITK filter
- Executes ITK filter
- Constructs *output* from ITK image

Using Filters

Object Paradigm (C++)

```
#include <SimpleITK.h>
namespace sitk = itk::simple;
...
// Create a smoothing filter
sitk::SmoothingRecursiveGaussianImageFilter gaussian;

// Set a parameter
gaussian.SetSigma ( 2.0 );

// "Execute" the Filter
sitk::Image blurredImage = gaussian.Execute ( image );
```

Object Paradigm (C++)

Flexibility

```
1 #include <SimpleITK.h>
2 namespace sitk = itk::simple;
3 ...
4 // Create a smoothing filter
5 sitk::SmoothingRecursiveGaussianImageFilter gaussian;
6
7 // Set parameter(s), then execute
8 sitk::Image blurredImage = gaussian
9                              .SetSigma ( 2.0 )
10                             .Execute ( image );
```

Object Paradigm (C++)

```
#include <SimpleITK.h>
namespace sitk = itk::simple;
...
blurredImage = sitk::SmoothingRecursiveGaussianImageFilter()
                          .SetSigma ( 2.0 )
                          .SetRadius ( 5 )
                          .Execute ( image );
```

One line: create anonymous filter, set parameters, and execute

"Function" Paradigm (C++)

```
#include <SimpleITK.h>
namespace sitk = itk::simple;
...
// Call the function version
// NB: Drop the "ImageFilter"!
// Signature:
/*
    sitk::Image SmoothingRecursiveGaussian (
            const Image&,
            double inSigma = 1.0,
            bool inNormalizeAcrossScale = false );
*/
sitk::Image blurredImage = sitk::SmoothingRecursiveGaussian (
                                image,
                                2.0,
                                false );
```

Mix & Match (C++)

```
#include <SimpleITK.h>
namespace sitk = itk::simple;
...
// Get our gaussian ready
sitk::SmoothingRecursiveGaussianImageFilter gaussian;
gaussian.SetSigma ( 2.0 );

// What is the effect on the image
sitk::Image difference = sitk::Subtract (
                           image,
                           gaussian.Execute ( image )
                           );
sitk::Image difference2 = sitk::Subtract (
                        image,
                        sitk::SmoothingRecursiveGaussian (
                          image, 2.0
                          )
                        );
```

Code Philosophy

Filter Class Overview (C++)

```
1 class SmoothingRecursiveGaussianImageFilter :
2     public ImageFilter {
3   typedef SmoothingRecursiveGaussianImageFilter Self;
4
5   /** Default Constructor that takes no arguments
6       and initializes default parameters */
7   SmoothingRecursiveGaussianImageFilter();
```

- In line 1, we declare a subclass of ImageFilter
- Line 3 creates a special typedef for use later
- The default constructor is line 7 (never any parameters)

Filter Class Overview (C++) Continued

```
1  /** Define the pixels types supported by this filter */
2  typedef BasicPixelIDTypeList  PixelIDTypeList;
```

- Notice *PixelIDTypeList* in line 2
- Used to instantiate ITK filters
- Determines valid input image types
- *BasicPixelIDTypeList* expands to:
  - *int*8_*t*, *uint*8_*t*
  - *int*16_*t*, *uint*16_*t*
  - *int*32_*t*, *uint*32_*t*
  - *float*, *double*

Filter Class Overview (C++) Continued

```
1  Self& SetSigma ( double t ) { ... return *this; }
2  double GetSigma() { return this->m_Sigma; }
3
4  Self& SetNormalizeAcrossScale ( bool t ) { ... }
5  Self& NormalizeAcrossScaleOn() { ... }
6  Self& NormalizeAcrossScaleOff() { ... }
7
8  bool GetNormalizeAcrossScale() { ... }
```

- Get/Set parameters
- Set methods always return *Self*& (more later)
- Generally, a direct mapping to ITK
- Boolean parameters generate *On* and *Off* methods

Filter Class Overview (C++) Continued

```
1  /** Name of this class */
2  std::string GetName() const { ... }
3
4  /** Print ourselves out */
5  std::string ToString() const;
```

- Return the name and description of the filter

Filter Class Overview (C++) Continued

```
1   /** Execute the filter on the input image */
2   Image Execute ( const Image & );
3
4   /** Execute the filter with parameters */
5   Image Execute ( const Image &,
6     double inSigma,
7     bool inNormalizeAcrossScale );
8 };  /* End of class SmoothingRecursiveGaussian */
9
10 Image SmoothingRecursiveGaussian ( const Image& ,
11   double inSigma = 1.0,
12   bool inNormalizeAcrossScale = false );
```

- Run the filter on an image and return the result
- Notice extra function (line 10), adds flexibility
- Drop *ImageFilter* from class name to get function name

Questions?

Using ITK with SimpleITK

Problem: Use ITK from SimpleITK (or vice versa)

```
./ToITK input.nii output.nii
```

Steps:

- Load image using SimpleITK
- Filter using ITK
- Save using OpenCV

To ITK

Starting code: `ToITK/ToITK.cxx`

Directory:

`SimpleITK-MICCAI-2011-Tutorial/Examples/AdvancedTutorial`

```
1  namespace sitk = itk::simple;
2  ...
3    // Load the image via SimpleITK
4    sitk::Image sitkImage = sitk::ReadImage ( inputFilename );
5
6    // Construct the ITK Pipeline
7    // Link pipeline to SimpleITK
8    // Update pipeline
9    // Create output SimpleITK image
10   // Save image via SimpleITK
11   sitk::WriteImage ( sOutput, outputFilename );
12   return EXIT_SUCCESS;
```

ToITK – Step 1: Construct the ITK Pipeline

```
1   // Construct the ITK Pipeline
2   typedef itk::Image<float,3> ImageType;
3   typedef itk::MirrorPadImageFilter<ImageType,ImageType> PadFilterType;
4   PadFilterType::SizeType upperBound, lowerBound;
5
6   PadFilterType::Pointer pad = PadFilterType::New();
7   for ( unsigned int i = 0; i < 3; i++ )
8     {
9       upperBound[i] = sitkImage.GetSize()[i];
10      lowerBound[i] = sitkImage.GetSize()[i];
11    }
12  pad->SetPadUpperBound ( upperBound );
13  pad->SetPadLowerBound ( lowerBound );
```

ToITK – Step 2: Link pipeline to SimpleITK

```
1  // Link pipeline to SimpleITK
2  ImageType::Pointer inputImage = (ImageType*) sitkImage.GetImageBase();
3  pad->SetInput ( inputImage );
```

ToITK – Step 3: Update ITK Pipeline

```
1  // Update pipeline
2  pad->Update();
```

ToITK – Step 4: Create the SimpleITK output image

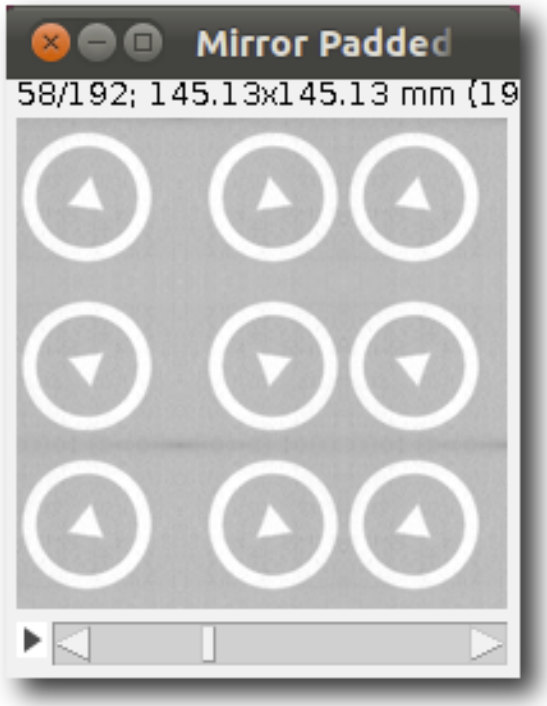
```
1  // Create output SimpleITK image
2  sitk::Image sOutput ( pad->GetOutput() );
```

ToITK – (Optional) Step 5: Show

```
1  // (Optional) Show the results
2  sitk::Show( sOutput );
```

To ITK Solution

```
~/Source/AdvancedTutorial-build/ToITK/ToITKSolution \
~/Source/SimpleITK/Testing/Data/Input/RA-Float.nrrd \
/tmp/foo.nii
```



To OpenCV

Problem: Use SimpleITK from another image processing library (OpenCV)

`./ToOpenCV input.png output.png`

Steps:

- Load image using SimpleITK
- Convert to OpenCV
- Filter using OpenCV
- Save using OpenCV

ToOpenCV

Starting code: `ToOpenCV/ToOpenCV.cxx`
Directory:
`SimpleITK-MICCAI-2011-Tutorial/Examples/AdvancedTutorial`

```
1  #include <SimpleITK.h>
2  #include <opencv2/opencv.hpp>
3  namespace sitk = itk::simple;
4  ...
5    sitk::Image sitkImage = sitk::ReadImage ( inputFilename );
6
7    // Convert SimpleITK to OpenCV image
8    cv::Mat ocvImage;
9
10   // Filter and write using OpenCV
11   cv::Mat output;
12   cv::medianBlur ( ocvImage, output, 5 );
13
14   cv::imwrite ( outputFilename, output );
15 ...
```

ToOpenCV – Step 1

Convert the SimpleITK image to a float

```
1  if ( sitkImage.GetPixelIDValue() != sitk::sitkFloat32 )
2    {
3    std::cout << "Input image is " << sitkImage.GetPixelIDTypeAsString()
4              << " converting to float" << std::endl;
5    sitkImage = sitk::Cast ( sitkImage, sitk::sitkFloat32 );
6    }
```

ToOpenCV – Step 2

Get SimpleITK pixel data

```
1  // Convert SimpleITK to OpenCV image
2  cv::Mat ocvImage ( sitkImage.GetHeight(), sitkImage.GetWidth(), CV_32F,
3                     (void*)sitkImage.GetBufferAsFloat() );
```

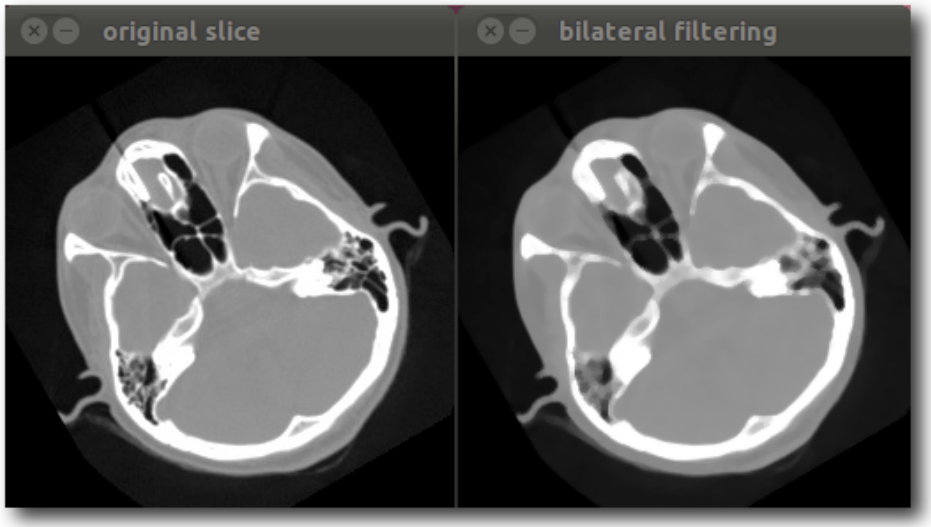
ToOpenCV – Step 3 (Optional)

Display the before and after

```
1 // NB: the imshow function requires 8-bit data, so convert
2 cv::Mat temp;
3 ocvImage.convertTo ( temp, CV_8U );
4 cv::imshow ( "original slice", temp );
5 output.convertTo ( temp, CV_8U );
6 cv::imshow ( "bilateral filtering", temp );
7
8 std::cout << "Press any key to continue" << std::endl;
9 cv::waitKey();
```

To OpenCV Solution

```
~/Source/AdvancedTutorial-build/ToOpenCV/ToOpenCV \
~/Source/SimpleITK/Testing/Data/Input/cthead1.png \
/tmp/head.png
```



To OpenCV and Back

Problem: Use OpenCV to process a SimpleITK volume slice-by-slice

`./ToOpenCVAndBack input.nii output.nii`

Steps:

- Load image using SimpleITK
- Extract slice
- Convert to OpenCV
- Filter using OpenCV
- Past slice back to SimpleITK
- Save result

To OpenCV and Back

Starting code: `ToOpenCVAndBack/ToOpenCVAndBack.cxx`
Directory:
`SimpleITK-MICCAI-2011-Tutorial/Examples/AdvancedTutorial`

```
1  namespace sitk = itk::simple;
2  ...
3  sitk::Image sitkImage = sitk::ReadImage ( inputFilename );
4
5  for ( unsigned int s = 0; s < sitkImage.GetDepth(); s++ )
6    {
7    // Extract a slice
8    // Go through ITK to grab the data
9    // Convert ITK to OpenCV image
10   // Filter using OpenCV
11   // Convert back to SimpleITK
12   // Paste the image back into SimpleITK
13   }
14 sitk::WriteImage ( sOutput, outputFilename );
```

To OpenCV and Back - Step 1: Extract a slice

```
1  // Extract a slice
2  std::vector<unsigned int> size = sitkImage.GetSize();
3  size[2] = 1;
4  std::vector<int> index ( 3, 0 );
5  index[2] = s;
6  std::cout << "Extracting: " << s << std::endl;
7  sitk::Image slice = sitk::RegionOfInterest ( sitkImage, size, index );
8
9  if ( slice.GetPixelIDValue() != sitk::sitkFloat32 )
10   {
11   slice = sitk::Cast ( slice, sitk::sitkFloat32 );
12   }
```

To OpenCV and Back - Step 2: Convert to OpenCV

```
1  // Convert ITK to OpenCV image
2    cv::Mat ocvImage ( slice.GetHeight(), slice.GetWidth(),
3                       CV_32F, (void*)sitkImage.GetBufferAsFloat() );
```

To OpenCV and Back - Step 3: Filter using OpenCV

```
1  // Filter using OpenCV
2  cv::Mat output;
3  cv::Sobel ( ocvImage, output, -1, 1, 1 );
```

To OpenCV and Back - Step 4: Back to SimpleITK

```
1  // Convert back to SimpleITK
2  sitk::ImportImageFilter importer;
3  importer.SetSize ( size );
4  importer.SetSpacing ( sitkImage.GetSpacing() );
5  importer.SetOrigin ( sitkImage.GetOrigin() );
6  importer.SetBufferAsFloat ( output.ptr<float>() );
7
8  sitk::Image toSimpleITKImage = importer.Execute();
```

To OpenCV and Back - Step 5: Paste back into SimpleITK volume

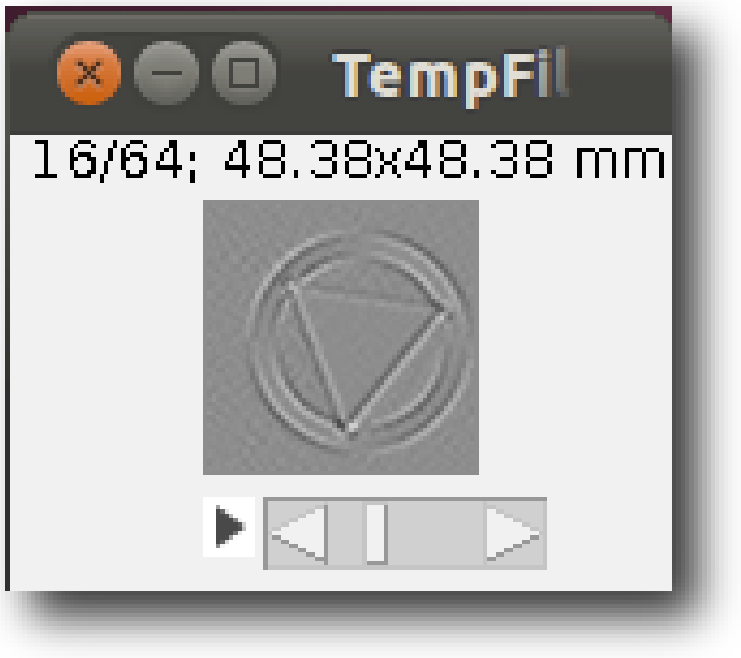
```
1  // Paste the image back into SimpleITK
2  // Paste ( Destination, Source, SourceSize, SourceIndex, DestIndex )
3  sOutput = sitk::Paste ( sOutput, toSimpleITKImage,
4            toSimpleITKImage.GetSize(), std::vector<int> ( 3,0 ), index );
```

To OpenCV and Back – (Optional) Step 6: Show

```
1  // (Optional) Show the results
2  sitk::Show( sOutput );
```

To OpenCV and Back Solution

```
~/Source/AdvancedTutorial-build/\
ToOpenCVAndBack/ToOpenCVAndBack \
~/Source/SimpleITK/Testing/Data/Input/RA-Float.nrrd \
/tmp/foo.nii
```



Wrap-up

- Gentle introduction to ITK
- Introduce SimpleITK
- Provide hands-on experience
- Problem solving, not direction following

Where to go next

Some resources for using and extending SimpleITK

- Documentation
  `http://erie.nlm.nih.gov/~blowek1/SimpleITK/pages.html`
- Conventions
  `http://erie.nlm.nih.gov/~blowek1/SimpleITK/Conventions.html`
- Contributions
  `http://erie.nlm.nih.gov/~blowek1/SimpleITK/Developer.html`

SimpleITK Tutorial

Image processing for mere mortals

Insight Software Consortium

Sept 23, 2011