

# SimpleITK Tutorial

Image processing for mere mortals

Insight Software Consortium

Sept 23, 2011

This presentation is copyrighted by  
The **Insight Software Consortium**

distributed under the  
**Creative Commons by Attribution License 3.0**  
<http://creativecommons.org/licenses/by/3.0>

# 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

- 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

- SimplelTK-disk1.vmdk
- SimplelTK.ovf

# Install VirtualBox

- Select the installer for your platform
- Run it



- 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!

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

```
1 typedef unsigned char PixelType;
2 enum {ImageDimension = 2};
3 typedef itk::Image<PixelType,ImageDimension> ImageType;
4 typedef itk::Vector<float,ImageDimension> VectorType;
5 typedef itk::Image<VectorType,ImageDimension> FieldType;
6 typedef itk::Image<VectorType::ValueType,ImageDimension> FloatImageType;
7 typedef ImageType::IndexType IndexType;
8 typedef ImageType::SizeType SizeType;
9 typedef ImageType::RegionType RegionType;
10 typedef itk::MultiResolutionPDEDeformableRegistration
11 <ImageType, ImageType, FieldType> RegistrationType;
```

# Template Freedom

```
1 using itk::simple;
2 // Read the image file
3 ImageFileReader reader;
4 reader.SetFileName ( "/my/fancy/file.nrrd" );
5 Image image = reader.Execute();
6
7 // This filters perform a gaussian blurring with sigma in
8 // physical space. The output image will be of real type.
9 SmoothingRecursiveGaussianImageFilter gaussian;
10 gaussian.SetSigma ( 2.0 );
11 Image blurredImage = gaussian.Execute ( image );
```

# Programming Models

- Object oriented
- Function oriented

More about this later

## 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

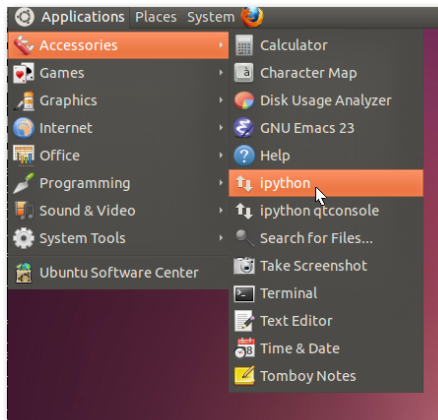
How are we doing with the Virtual Box images?

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

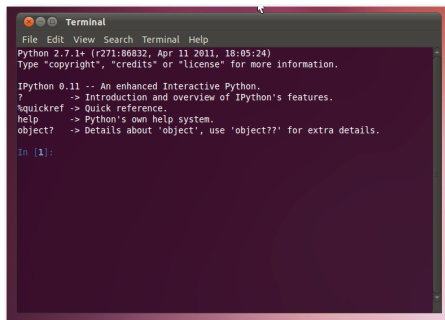
- TODO: what do we need to talk about here?

# Just Enough Python to be Dangerous

- Start a terminal
- Run iPython



# Just Enough Python to be Dangerous

A screenshot of a macOS Terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The text inside the terminal shows the output of running Python 2.7.1+, followed by the IPython 0.11 startup banner. The banner includes a list of shortcuts: '?' for introduction, %quickref for quick reference, 'help' for Python's help system, and 'object?' for details about the 'object' type. The prompt "In [1]:" is visible at the bottom.

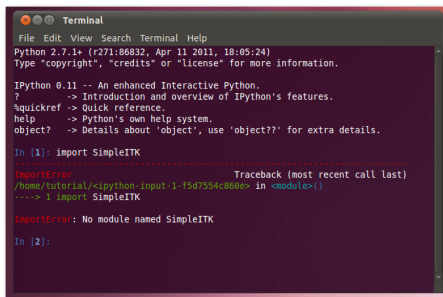
```
Python 2.7.1+ (r271:86832, Apr 11 2011, 18:05:24)
Type "copyright", "credits" or "license" for more information.

IPython 0.11 -- An enhanced Interactive Python.
?      -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help    -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

In [1]:
```

# Just Enough Python to be Dangerous

Import the SimpleITK package



```
Terminal
File Edit View Search Terminal Help
Python 2.7.1+ (r271:86832, Apr 11 2011, 18:05:24)
Type "copyright", "credits" or "license" for more information.

IPython 0.11 -- An enhanced Interactive Python.
?                -> Introduction and overview of IPython's features.
%quickref        -> Quick reference.
help             -> Python's own help system.
object?         -> Details about 'object', use 'object??' for extra details.

In [1]: import SimpleITK
-----
ImportError                               Traceback (most recent call last)
/home/tutorial/cipython-input-1-f5d7554c800e in <module>()
----> 1 import SimpleITK

ImportError: No module named SimpleITK

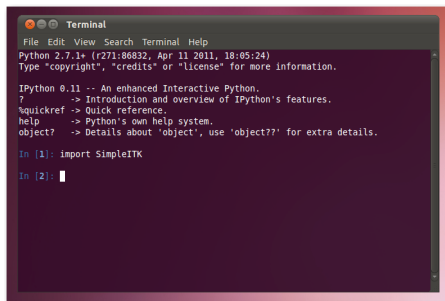
In [2]:
```

What just happened?

# Just Enough Python to be Dangerous

Need to tell iPython where to find SimpleITK

See `/home/tutorial/.ipython/ipython.py`

A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The text inside the terminal shows the Python version (2.7.1+), the date and time (Apr 11 2011, 18:05:24), and the IPython version (0.11). It lists several shortcuts: ? for introduction, %quickref for quick reference, help for Python's own help system, and object? for details about 'object'. The prompt "In [1]:" is followed by the command "import SimpleITK". The prompt "In [2]:" is followed by a cursor.

```
Python 2.7.1+ (r271:86832, Apr 11 2011, 18:05:24)
Type "copyright", "credits" or "license" for more information.

IPython 0.11 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]: import SimpleITK

In [2]:
```

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

Back to iPython ([Examples/BasicTutorial1/Image.py](#))



# What just happened?

```
1 # Create an image
2 image = sitk.Image ( 256, 256, 256,
3                       sitk.sitkInt16 );
4 # How about 2d?
5 twoD = sitk.Image ( 64, 64,
6                     sitk.sitkFloat32 )
```

- sitk 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 (Examples/BasicTutorial1/Image.py)

# What just happened?

```
1 # Addressing pixels
2 image.GetPixel ( 0, 0, 0 )
3 image.SetPixel ( 0, 0, 0, 1 )
4 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 (Examples/BasicTutorial1/Image.py)

# 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 syntactic 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(0,0,0,10)`

# 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...

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

Back to `iPython (Examples/BasicTutorial1/MemoryManagement.py)`

# Image Memory Management

```
1 image = SimpleITK.Image ( 32, 32, 32, SimpleITK.sitkInt16 )
2 print image
3 ...
4 Image (0x94f2d98)
5   Reference Count: 1
6 ...
7 # Clone image
8 b = SimpleITK.Image ( image )
9 print image
10 ...
11 Image (0x94f2d98)
12   Reference Count: 2
13 ...
14 print b
15 ...
16 Image (0x94f2d98)
```

# Image Memory Management

```
1 print b
2 ...
3 Image (0x94f2d98)
4 ...
5 b[0,0,0] = 1
6 print b
7 ...
8 Image (0x94f4cb0)
9   Reference Count: 1
10 ...
11 print image
12 ...
13 Image (0x94f2d98)
14   Reference Count: 1
15 ...
```



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...

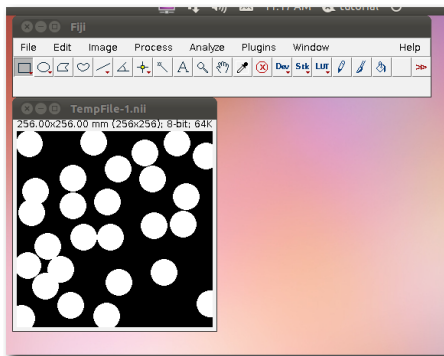
- Utilize language-specific memory management
- Automatic in Python, Java, Ruby, C#, Lua
- More manual in Tcl

# Input/Output

# Read/Write/Display Images

Back to iPython ([Examples/BasicTutorial1/InputOutput.py](#))

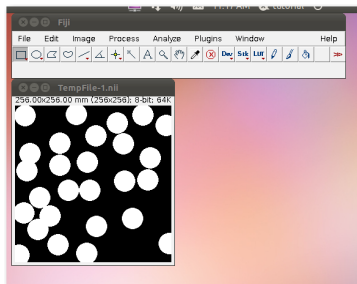
# Display



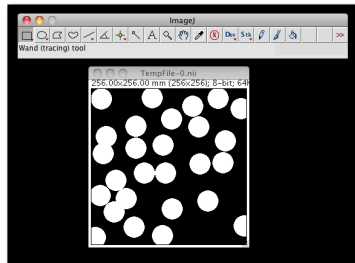
What just happened?

```
1 # What's the image look like?
2 sitk.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>

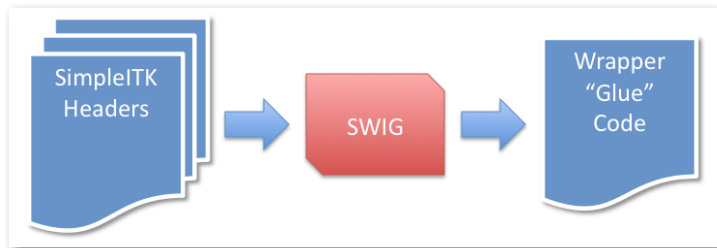
# Questions/Break

# 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)

```
1 #include <SimpleITK.h>
2 using itk::simple;
3 ...
4 // Create a smoothing filter
5 SmoothingRecursiveGaussianImageFilter gaussian;
6 // Set a parameter
7 gaussian.SetSigma ( 2.0 );
8 // "Execute" the Filter
9 Image blurredImage = gaussian.Execute ( image );
```

```
1 from SimpleITK import *
2 # Create a smoothing filter
3 SmoothingRecursiveGaussianImageFilter gaussian
4 # Set a parameter
5 gaussian.SetSigma ( 2.0 );
6 # "Execute" the Filter
7 blurredImage = gaussian.Execute ( image );
```

# Object Paradigm (Java)

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

`Examples/BasicTutorial2/Filters.py`

# What just happened?

```
1 # Simple smoothing
2 smooth = sitk.SmoothingRecursiveGaussian ( image, 2.0 )
3 sitk.Show ( sitk.Subtract ( image, smooth ) )
4 ...
5 RuntimeError: Exception thrown in SimpleITK Subtract: ...
6 sitk::ERROR: Both images for SubtractImageFilter don't match type or dimension!
7 ...
```

- 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

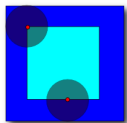
```
1 # Much better
2 print "Before: ", smooth.GetPixelIDTypeAsString()
3 smooth = sitk.Cast ( smooth, image.GetPixelIDValue() )
4 print "After: ", smooth.GetPixelIDTypeAsString()
5 sitk.Show ( sitk.Subtract ( image, smooth ), "DiffWithGaussian" )
```

Back to iPython ([Examples/BasicTutorial2/Filters.py](#))

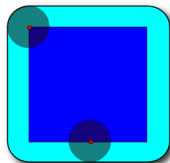
# Morphology

`Examples/BasicTutorial2/Morphology.py`

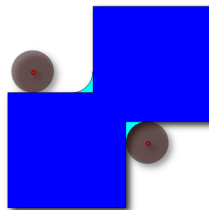
# Operators



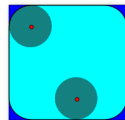
Erosion



Dilation



Closing



Opening

Images from

[http://en.wikipedia.org/wiki/Mathematical\\_morphology](http://en.wikipedia.org/wiki/Mathematical_morphology)

Back to iPython (`Examples/BasicTutorial2/Morphology.py`)

# Label Maps

# 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 <sup>†</sup>
+	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”	$\neg A$

<sup>†</sup>  $A$  and  $B$  are images (2D or 3D),  $s$  is a scalar

# Masking

# Edge Detection

# Threshold-based Segmentation

# Advanced Topics

# 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

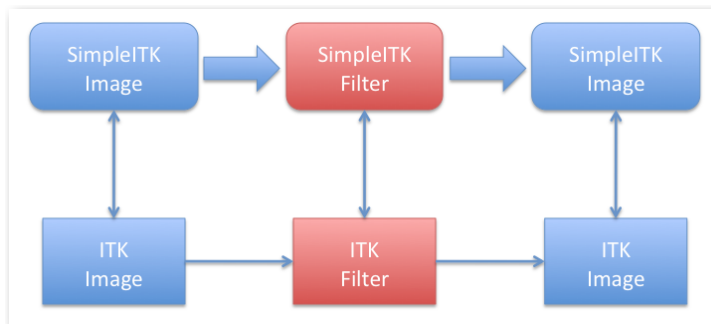


# Supported Platforms

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

# SimpleTK 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

- *int8\_t*
- *uint8\_t*
- *int16\_t*
- *uint16\_t*
- *int32\_t*
- *uint32\_t*
- *float*
- *double*
- *std :: complex < float >*
- *std :: complex < double >*

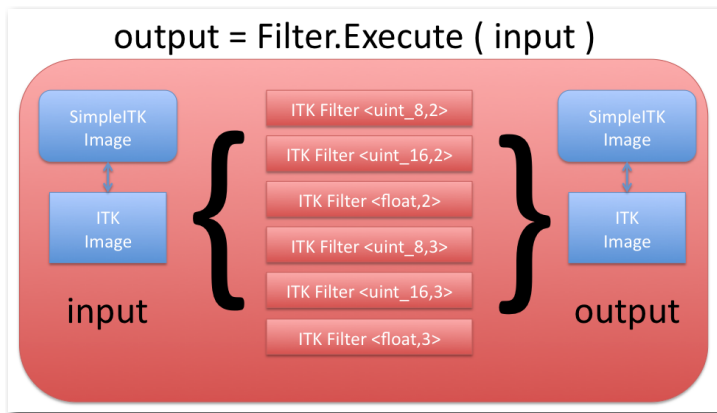
- Vector Types

- *int8\_t*
- *uint8\_t*
- *int16\_t*
- *uint16\_t*
- *float*
- *double*

- Label Types

- *uint8\_t*
- *uint16\_t*
- *uint32\_t*

# Filter Anatomy



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

# Using Filters

# Object Paradigm (C++)

```
1 #include <SimpleITK.h>
2 namespace sitk = itk::simple;
3 ...
4 // Create a smoothing filter
5 sitk::SmoothingRecursiveGaussianImageFilter gaussian;
6
7 // Set a parameter
8 gaussian.SetSigma ( 2.0 );
9
10 // "Execute" the Filter
11 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++)

```
1 #include <SimpleITK.h>
2 namespace sitk = itk::simple;
3 ...
4 blurredImage = sitk::SmoothingRecursiveGaussianImageFilter()
5                 .SetSigma ( 2.0 )
6                 .SetRadius ( 5 )
7                 .Execute ( image );
```

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

# “Function” Paradigm (C++)

```
1 #include <SimpleITK.h>
2 namespace sitk = itk::simple;
3 ...
4 // Call the function version
5 // NB: Drop the "ImageFilter"!
6 // Signature:
7 /*
8     sitk::Image SmoothingRecursiveGaussian (
9         const Image&,
10         double inSigma = 1.0,
11         bool inNormalizeAcrossScale = false );
12 */
13 sitk::Image blurredImage = sitk::SmoothingRecursiveGaussian (
14     image,
15     2.0,
16     false );
```

# Mix & Match (C++)

```
1 #include <SimpleITK.h>
2 namespace sitk = itk::simple;
3 ...
4 // Get our gaussian ready
5 sitk::SmoothingRecursiveGaussianImageFilter gaussian;
6 gaussian.SetSigma ( 2.0 );
7
8 // What is the effect on the image
9 sitk::Image difference = sitk::Subtract (
10     image,
11     gaussian.Execute ( image )
12 );
13 sitk::Image difference2 = sitk::Subtract (
14     image,
15     sitk::SmoothingRecursiveGaussian (
16     image, 2.0
17     )
18 );
```

# 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:
  - *int8\_t, uint8\_t*
  - *int16\_t, uint16\_t*
  - *int32\_t, uint32\_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

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() );
```

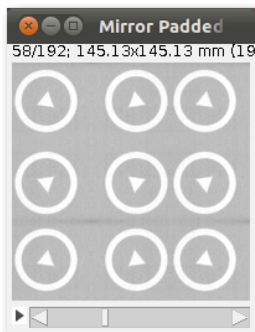


# ToITK – (Optional) Step 5: Show

```
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

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     ...
```

## 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  }
```

## Get SimpleITK pixel data

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

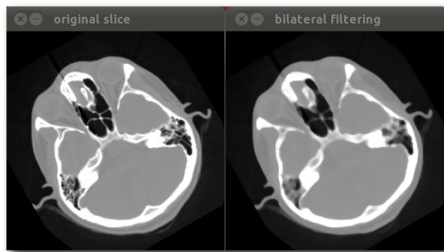
# ToOpenCV – Step 3 (Optional)

## Display the before and after

```
1 // NB: the imshow function requires 8-bit data, so convert
2 cv::Mat temp;
3 cvImage.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 );
```

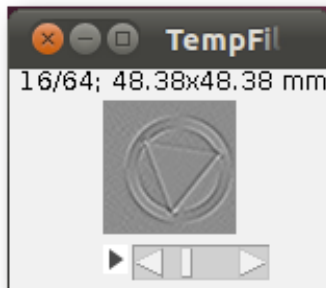
# To OpenCV and Back – (Optional) Step 6: Show

```
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
```



# Using SimpleTK from Java/Groovy

`Examples/AdvancedTutorial/Java/Example.groovy`

- Language build on Java
- Superset of Java
- Can be used interactively

```
1 import org.itk.simple.*;
2
3 Image i;
4 i = new Image ( 64, 64, 64, PixelIDValueEnum.sitkInt16 );
5 SimpleITK.show ( i, "Blank" );
```

groovysh -classpath

./SimpleITK-Build/SimpleITK-Build/Wrapping/org.itk.simple.jar

# Conclusion

- 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