SimpleITK Workshop

Maria A. Zuluaga

UCL Centre for Medical Image Computing, London

January 29, 2015

How this will go?

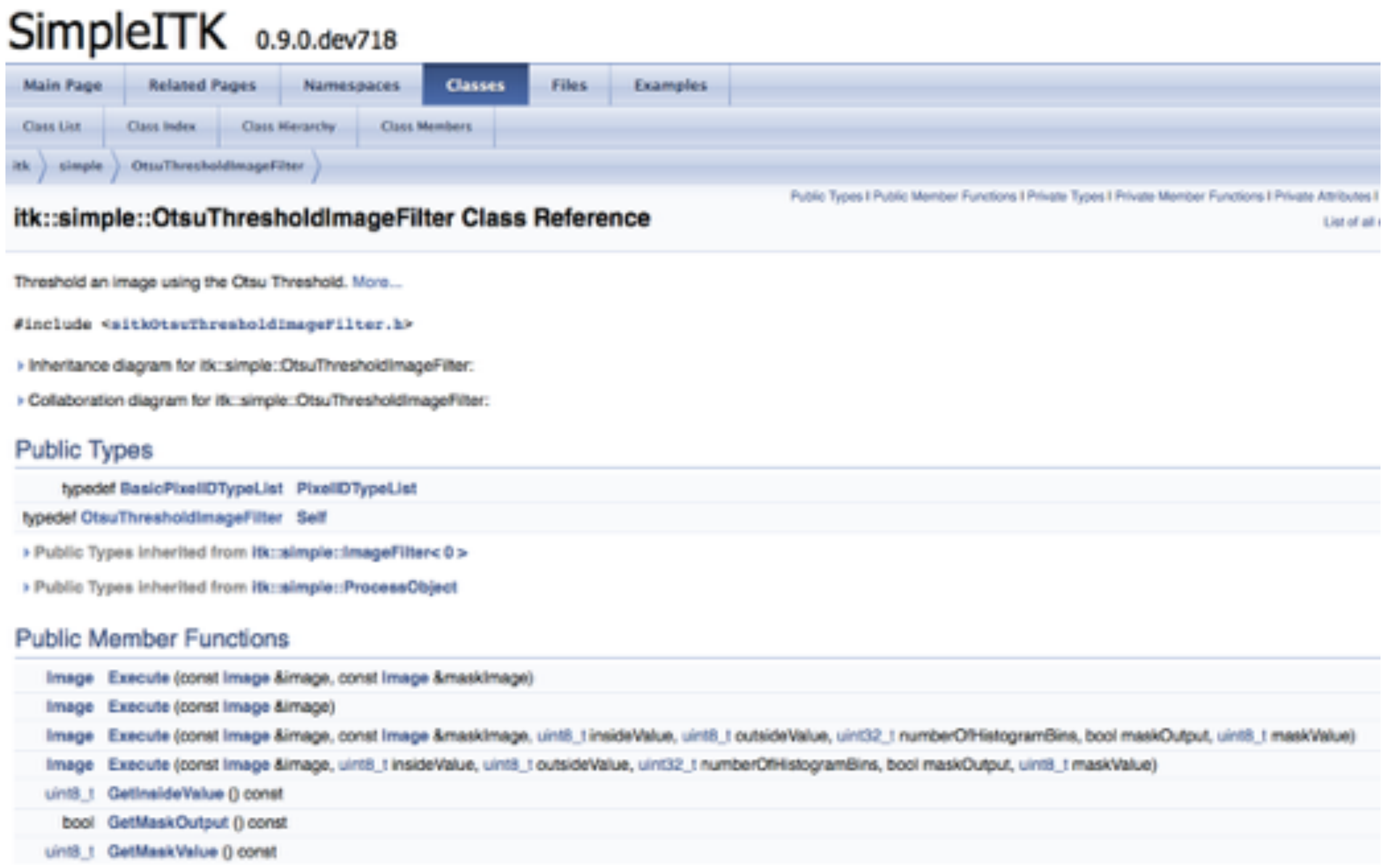
- Introduction to ITK and SimpleITK
- VirtualBox: Setup Instructions
- **Exercise 1**
- Verify solution
- **Exercise 2**
- Verify solution
- **Exercise 3**
- Verify solution

Important note to advanced users:
If you feel you can go ahead on your own, do so.

Goals

- Gentle introduction to ITK and SimpleITK
- Usage of ITK and SimpleITK documentation

Google: FilterName + SimpleITK



itk::simple::OtsuThresholdImageFilter Class Reference

Threshold an image using the Otsu Threshold. More...

```
#include <sitkOtsuThresholdImageFilter.h>
```

› Inheritance diagram for itk::simple::OtsuThresholdImageFilter:

› Collaboration diagram for itk::simple::OtsuThresholdImageFilter:

Public Types

typedef BasicPixelIDTypeList	PixelIDTypeList
typedef OtsuThresholdImageFilter	Self

› Public Types inherited from itk::simple::ImageFilter< 0 >

› Public Types inherited from itk::simple::ProcessObject

Public Member Functions

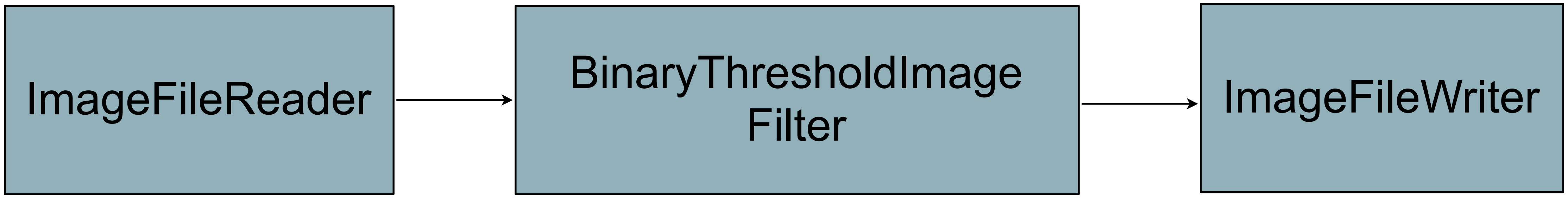
Image	Execute (const Image &image, const Image &maskImage)
Image	Execute (const Image &image)
Image	Execute (const Image &image, const Image &maskImage, uint8_t insideValue, uint8_t outsideValue, uint32_t numberOfHistogramBins, bool maskOutput, uint8_t maskValue)
Image	Execute (const Image &image, uint8_t insideValue, uint8_t outsideValue, uint32_t numberOfHistogramBins, bool maskOutput, uint8_t maskValue)
uint8_t	GetInsideValue () const
bool	GetMaskOutput () const
uint8_t	GetMaskValue () const

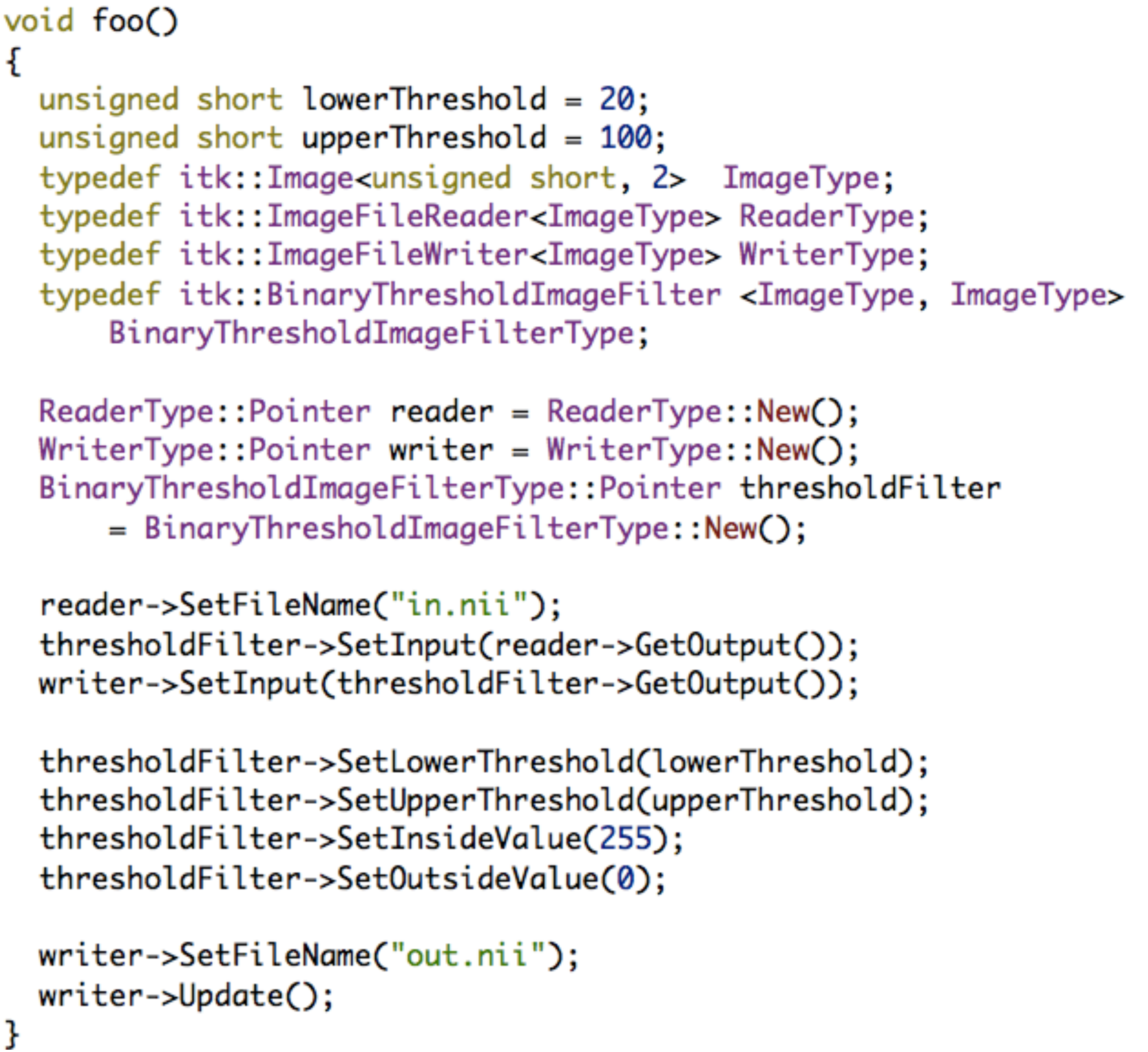
Getting started: What is ITK?

- Image Processing and Analysis Toolkit
- Does not do visualization
- Does not include a GUI framework
- Designed for Medical Imaging Applications
- In general algorithms work in N-dimensions

How is code written in ITK

- Typically in C++.
- Heavily templated/generic programming.
- Multi-threaded capable.
- Pipeline architecture.
- Uses CMAKE to build.


ImageFileReader
BinaryThresholdImage Filter
ImageFileWriter



```
void foo()
{
  unsigned short lowerThreshold = 20;
  unsigned short upperThreshold = 100;
  typedef itk::Image<unsigned short, 2>  ImageType;
  typedef itk::ImageFileReader<ImageType> ReaderType;
  typedef itk::ImageFileWriter<ImageType> WriterType;
  typedef itk::BinaryThresholdImageFilter <ImageType, ImageType>
      BinaryThresholdImageFilterType;

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

  reader->SetFileName("in.nii");
  thresholdFilter->SetInput(reader->GetOutput());
  writer->SetInput(thresholdFilter->GetOutput());

  thresholdFilter->SetLowerThreshold(lowerThreshold);
  thresholdFilter->SetUpperThreshold(upperThreshold);
  thresholdFilter->SetInsideValue(255);
  thresholdFilter->SetOutsideValue(0);

  writer->SetFileName("out.nii");
  writer->Update();
}
```

Type definitions

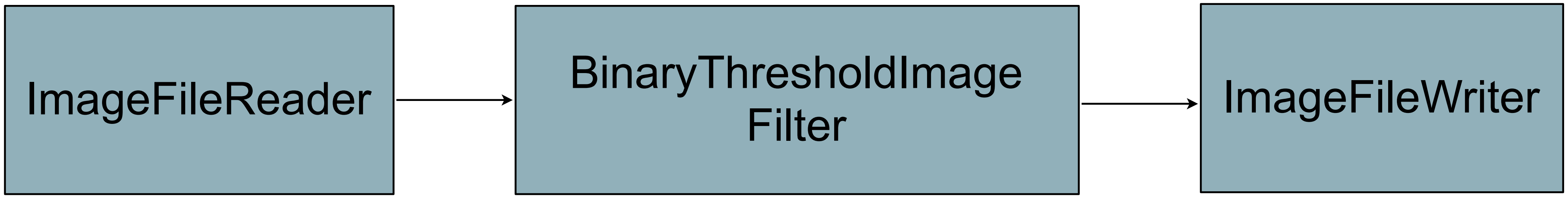
“Wiring”

Parameters

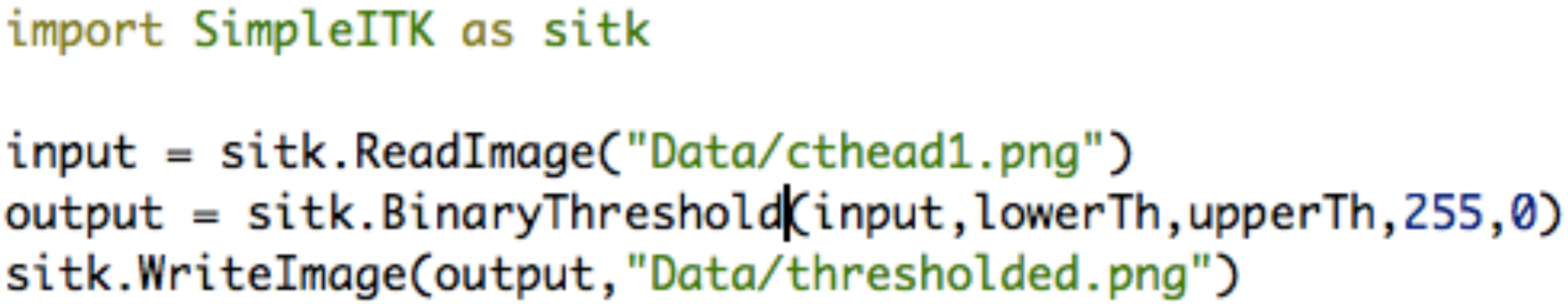
Trigger

Getting started: Why SimpleITK?

- To provide a template-less, layer to ITK in C++
- To provide wrappers to several scripting languages:
  - Python
  - R
- To provide access to many ITK algorithms


ImageFileReader
BinaryThresholdImage Filter
ImageFileWriter

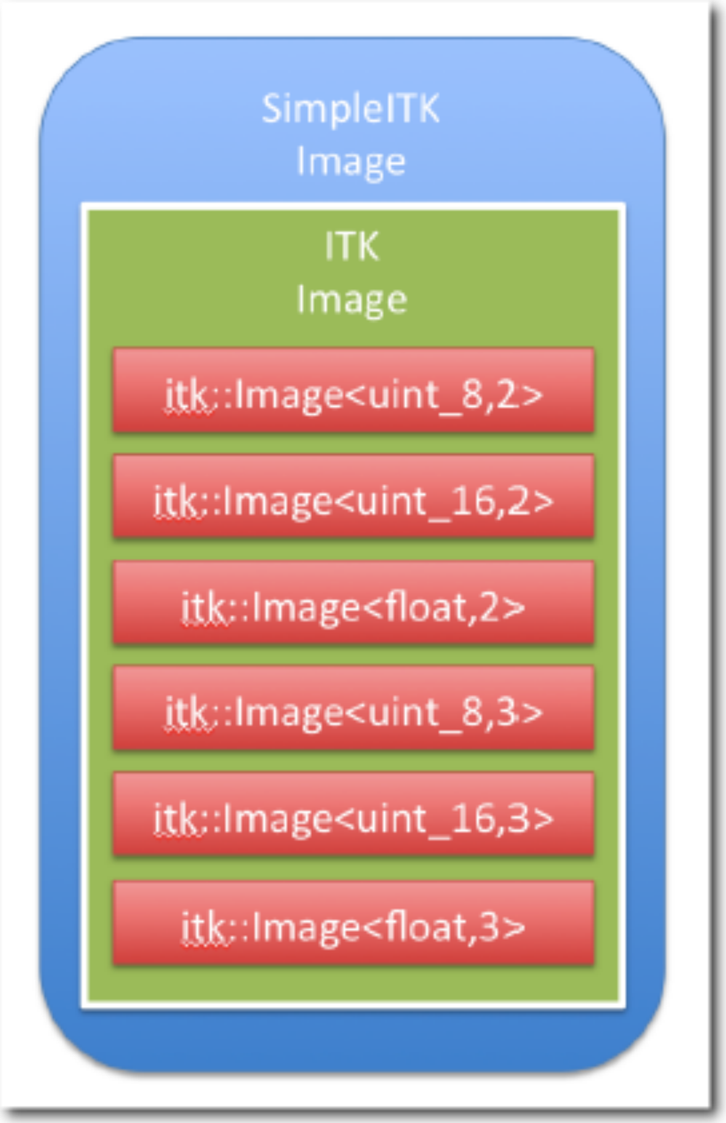
SimpleITK



```
import SimpleITK as sitk

input = sitk.ReadImage("Data/cthead1.png")
output = sitk.BinaryThreshold(input,lowerTh,upperTh,255,0)
sitk.WriteImage(output,"Data/thresholded.png")
```

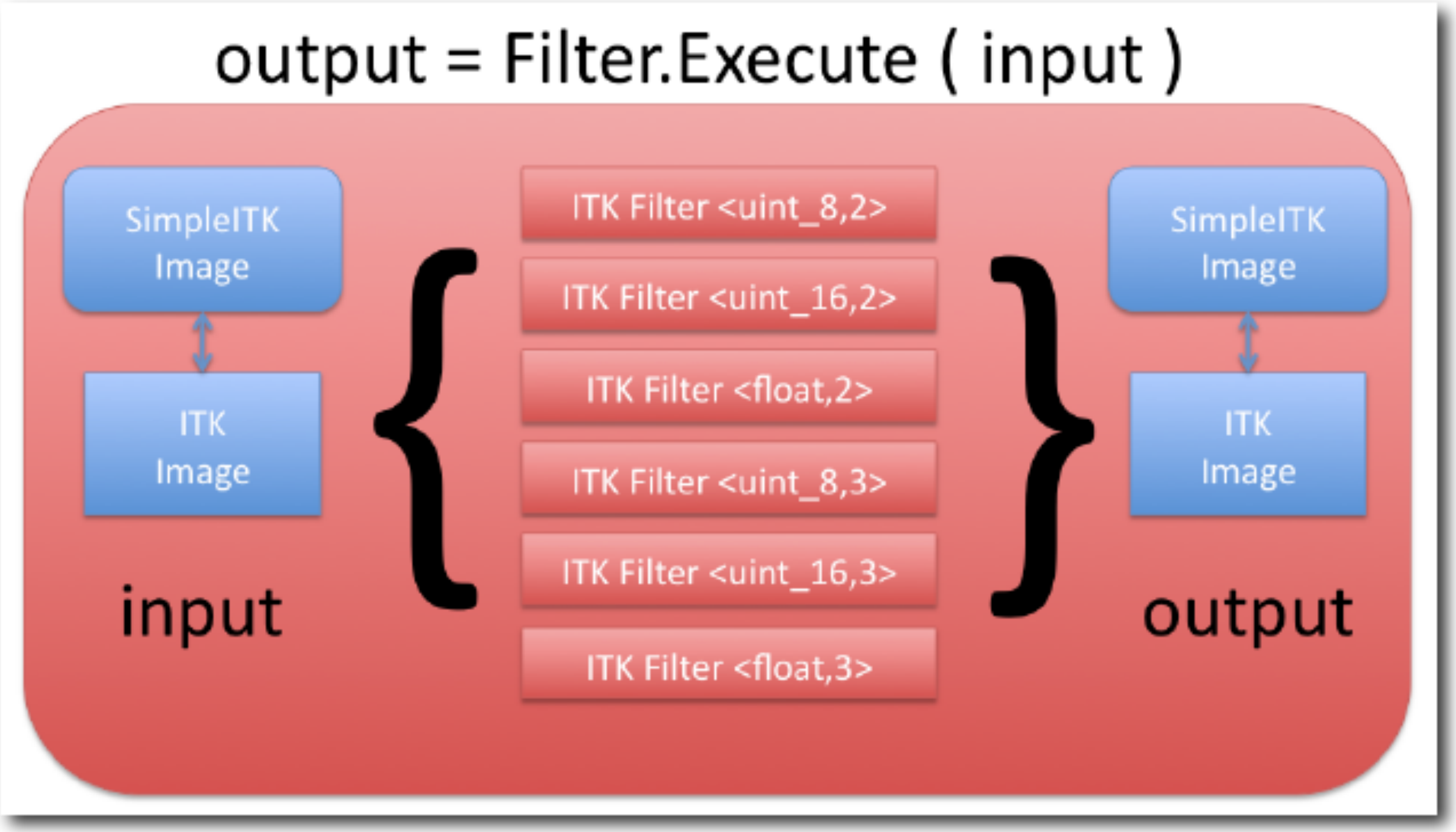
SimpleITK: Anatomy of an image

- Wraps up an ITK image
- Adds some functionalities
- No need to take care of the image
  - Dimension
  - Pixel/voxel type



Source: SimpleITK tutorial, MICCAI 2011

SimpleITK: Anatomy of a filter



Source: SimpleITK tutorial, MICCAI 2011

SimpleITK: Disadvantages

- Not as widely used as ITK
- Less support
- Limited number of algorithms

VirtualBox: Setup instructions

- Contains all the required tools to work with SimpleITK

  - SimpleITK
  - QtCreator - Coding
  - ImageJ - Image visualisation

- username: simpleitk
- password: simpleitk

Go to:

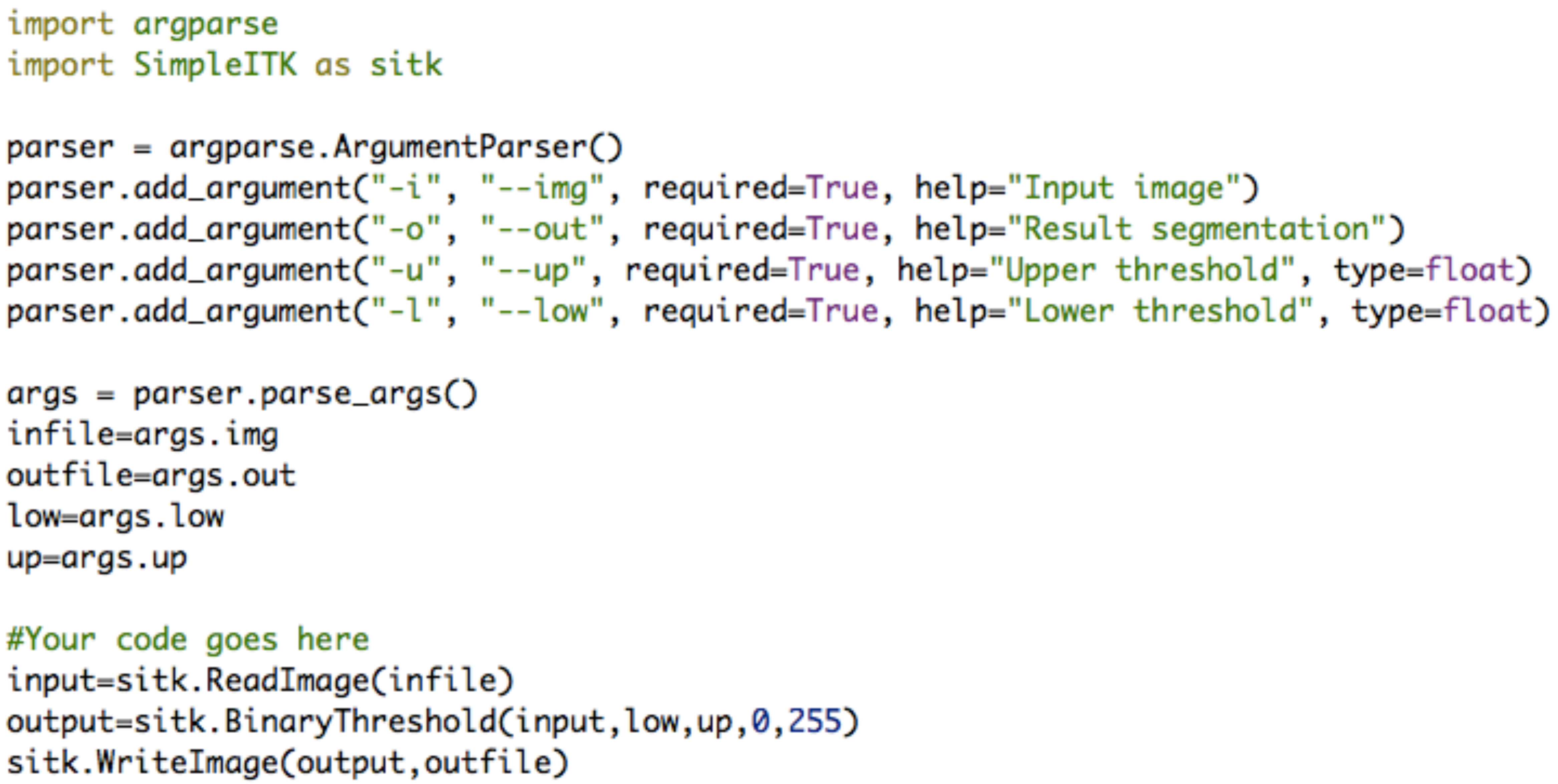
/cs/research/medic/mzuluaga0/mzuluaga/tools/SimpleITK

Virtual machine: simpleitk.ova
(available along the term)

Case 1: A very simple example

- exercise_one.py
- We want to get rid of the skull of a 2D image of the brain through simple thresholding.
- Complete the code for this purpose.
- Evaluate different thresholds until you find the best option.
- Can you mask the full brain?
- Ignore problems with the background.

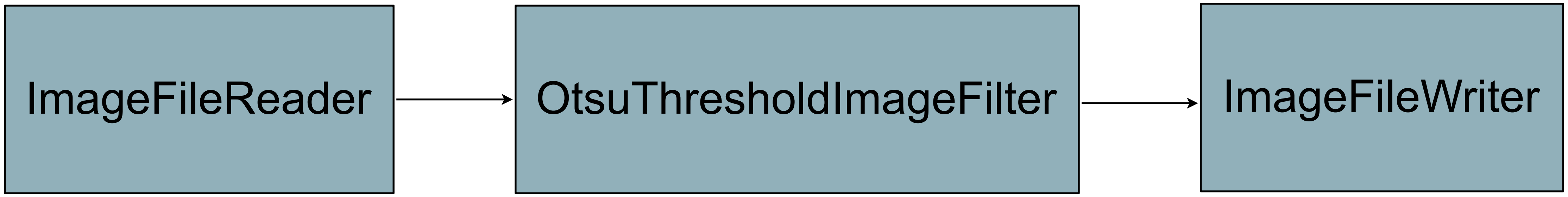
Solution

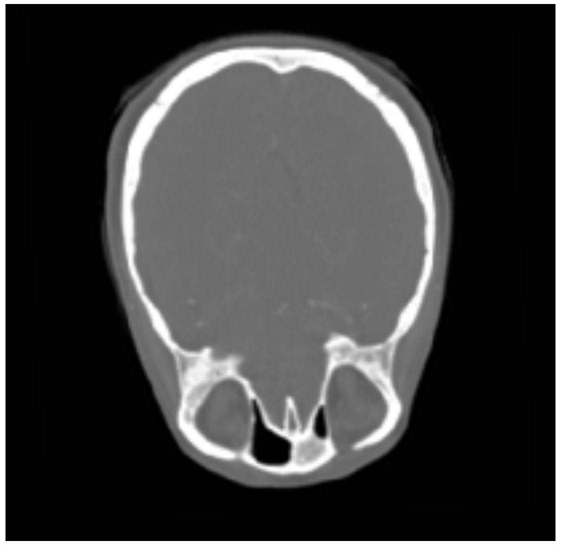


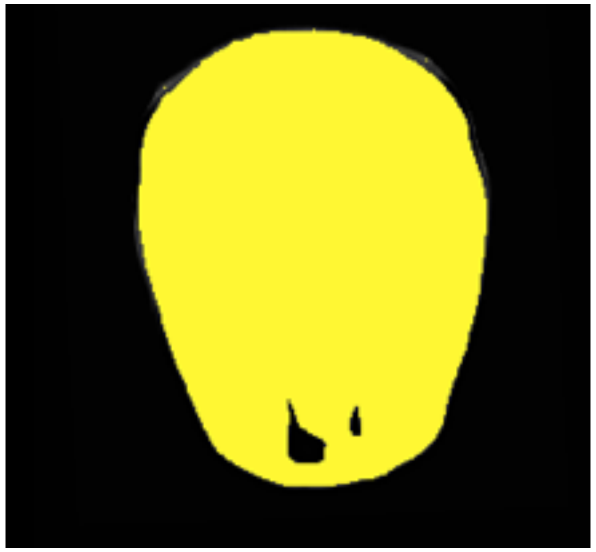
```
import argparse
import SimpleITK as sitk

parser = argparse.ArgumentParser()
parser.add_argument("-i", "--img", required=True, help="Input image")
parser.add_argument("-o", "--out", required=True, help="Result segmentation")
parser.add_argument("-u", "--up", required=True, help="Upper threshold", type=float)
parser.add_argument("-l", "--low", required=True, help="Lower threshold", type=float)

args = parser.parse_args()
infile=args.img
outfile=args.out
low=args.low
up=args.up

#Your code goes here
input=sitk.ReadImage(infile)
output=sitk.BinaryThreshold(input,low,up,0,255)
sitk.WriteImage(output,outfile)
```

Case 2: A simple filter

- Given an image, obtain a mask of the full brain using Otsu thresholding.

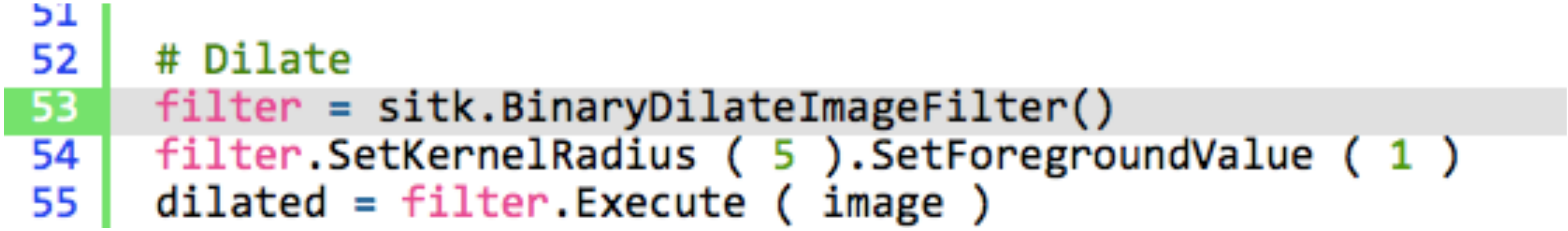

ImageFileReader → OtsuThresholdImageFilter → ImageFileWriter





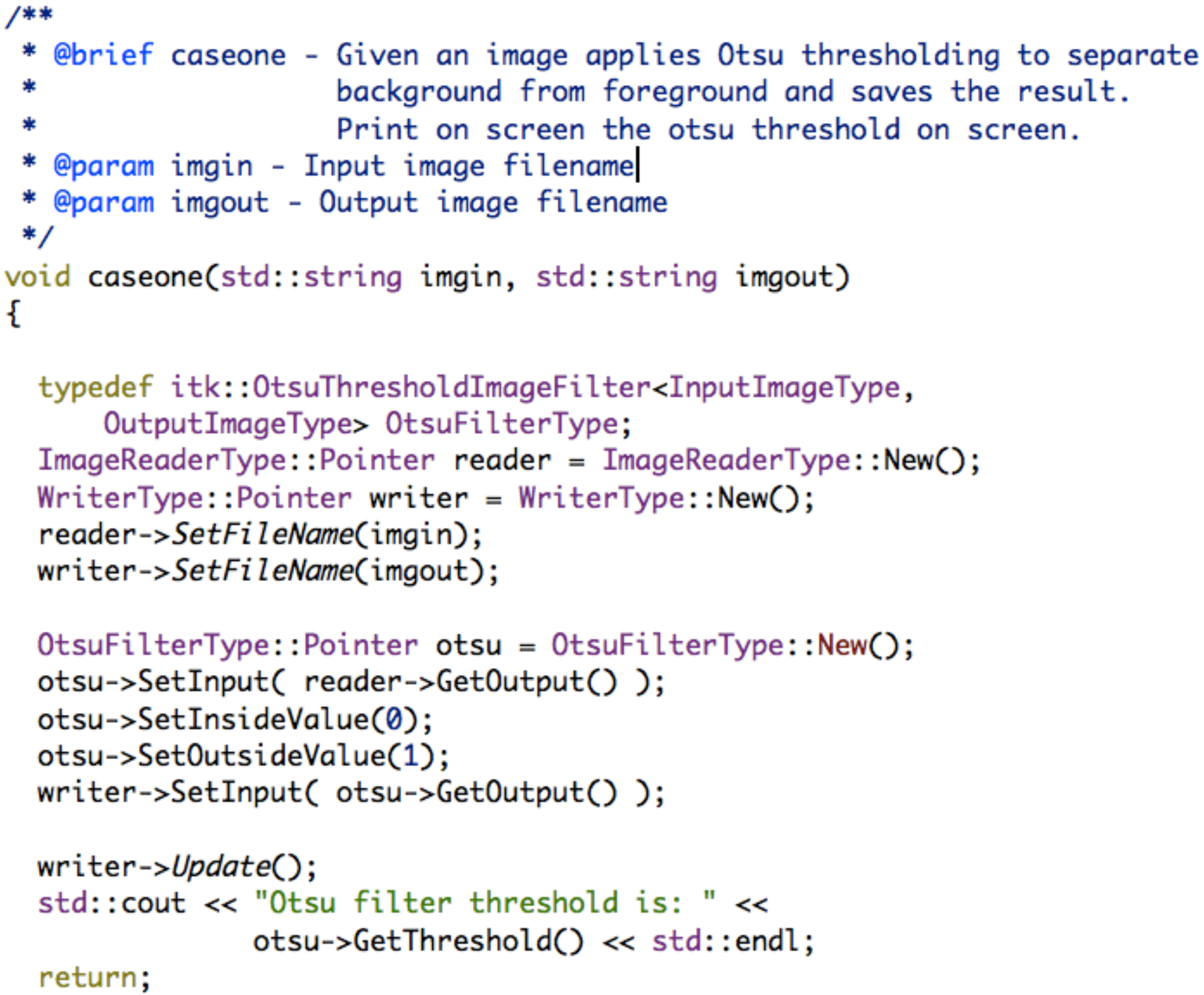
Some helpful information

- Some filters require configuration before they can be used



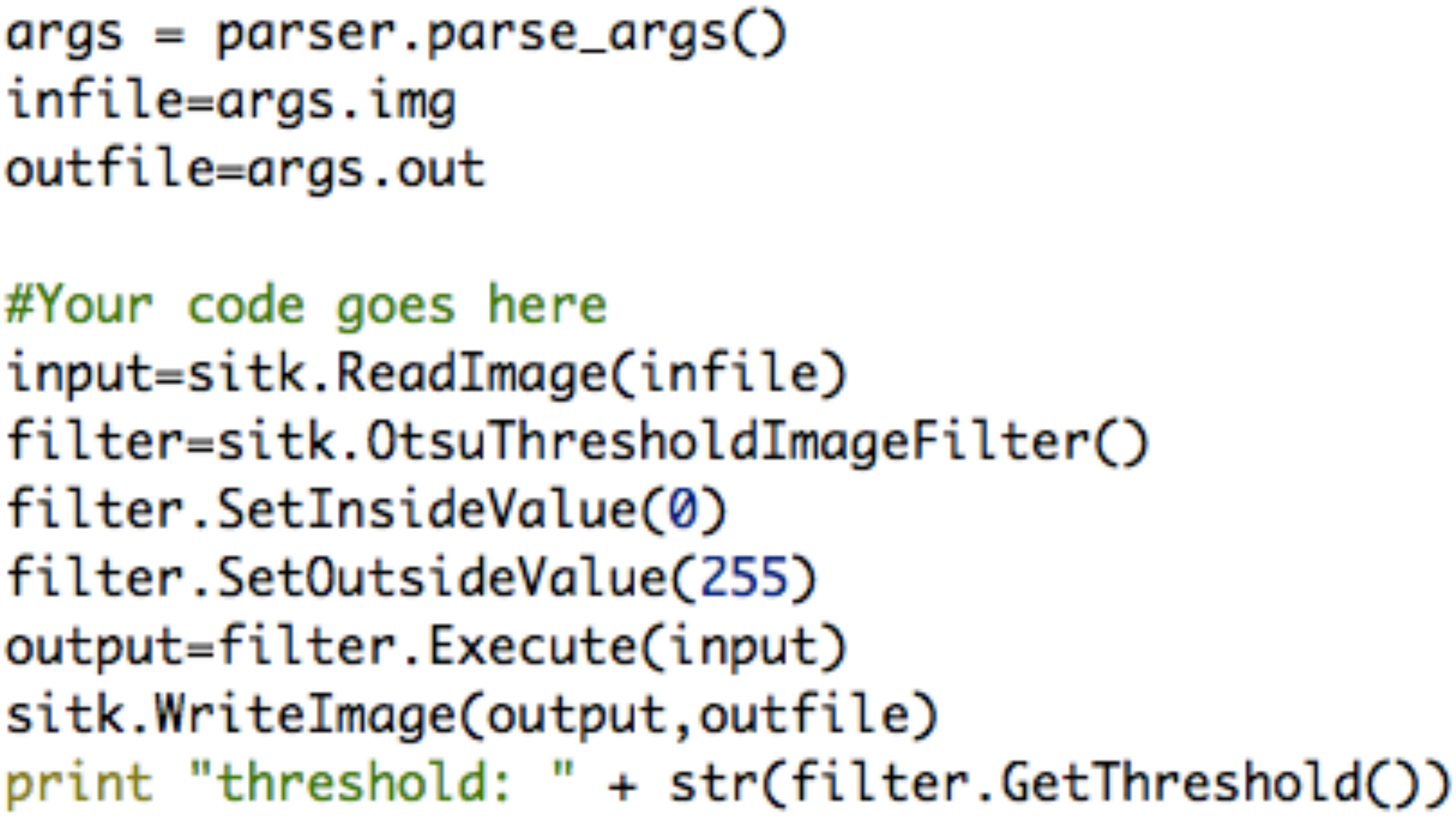
```
# Dilate
filter = sitk.BinaryDilateImageFilter()
filter.SetKernelRadius ( 5 ).SetForegroundValue ( 1 )
dilated = filter.Execute ( image )
```

For Otsu thresholding, we need to define inside and outside values.



```
/**
 * @brief caseone - Given an image applies Otsu thresholding to separate
 *                  background from foreground and saves the result.
 *                  Print on screen the otsu threshold on screen.
 * @param imgin - Input image filename
 * @param imgout - Output image filename
 */
void caseone(std::string imgin, std::string imgout)
{

  typedef itk::OtsuThresholdImageFilter<InputImageType,
      OutputImageType> OtsuFilterType;
  ImageReaderType::Pointer reader = ImageReaderType::New();
  WriterType::Pointer writer = WriterType::New();
  reader->SetFileName(imgin);
  writer->SetFileName(imgout);

  OtsuFilterType::Pointer otsu = OtsuFilterType::New();
  otsu->SetInput( reader->GetOutput() );
  otsu->SetInsideValue(0);
  otsu->SetOutsideValue(1);
  writer->SetInput( otsu->GetOutput() );

  writer->Update();
  std::cout << "Otsu filter threshold is: " <<
               otsu->GetThreshold() << std::endl;
  return;
```

Solution: SimpleITK



```
args = parser.parse_args()
infile=args.img
outfile=args.out

#Your code goes here
input=sitk.ReadImage(infile)
filter=sitk.OtsuThresholdImageFilter()
filter.SetInsideValue(0)
filter.SetOutsideValue(255)
output=filter.Execute(input)
sitk.WriteImage(output,outfile)
print "threshold: " + str(filter.GetThreshold())
```

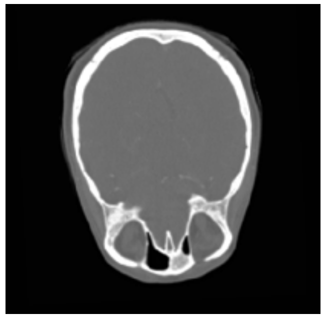
Case 3: Connecting

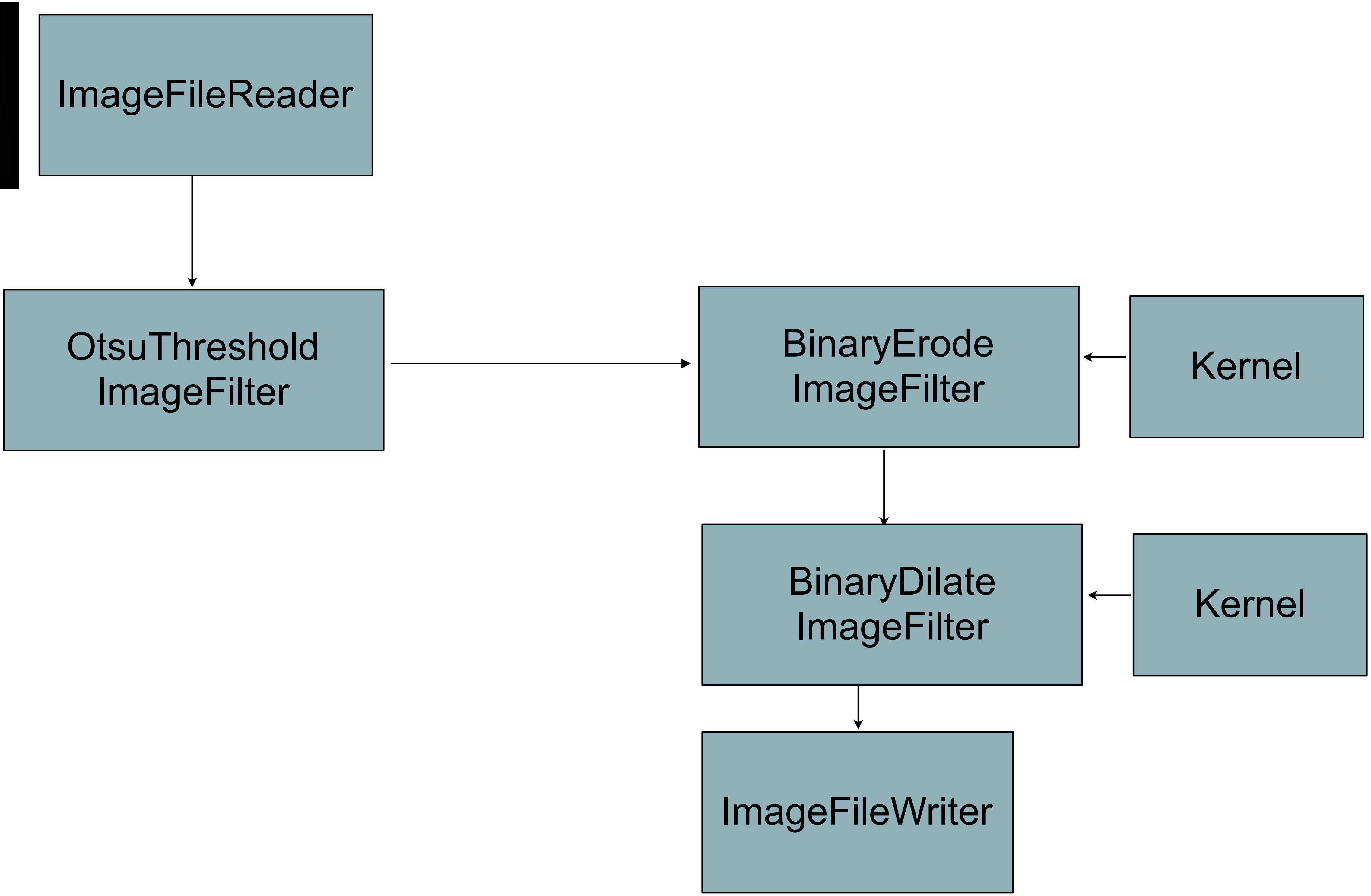
- Given the previous image, we want to obtain a mask that only contains the brain after applying the threshold.

- Results do not need to be 100% accurate.

- Use an erosion and dilation operations to obtain the desired result

Case 3: Pipeline





ImageFileReader

OtsuThreshold ImageFilter

BinaryErode ImageFilter

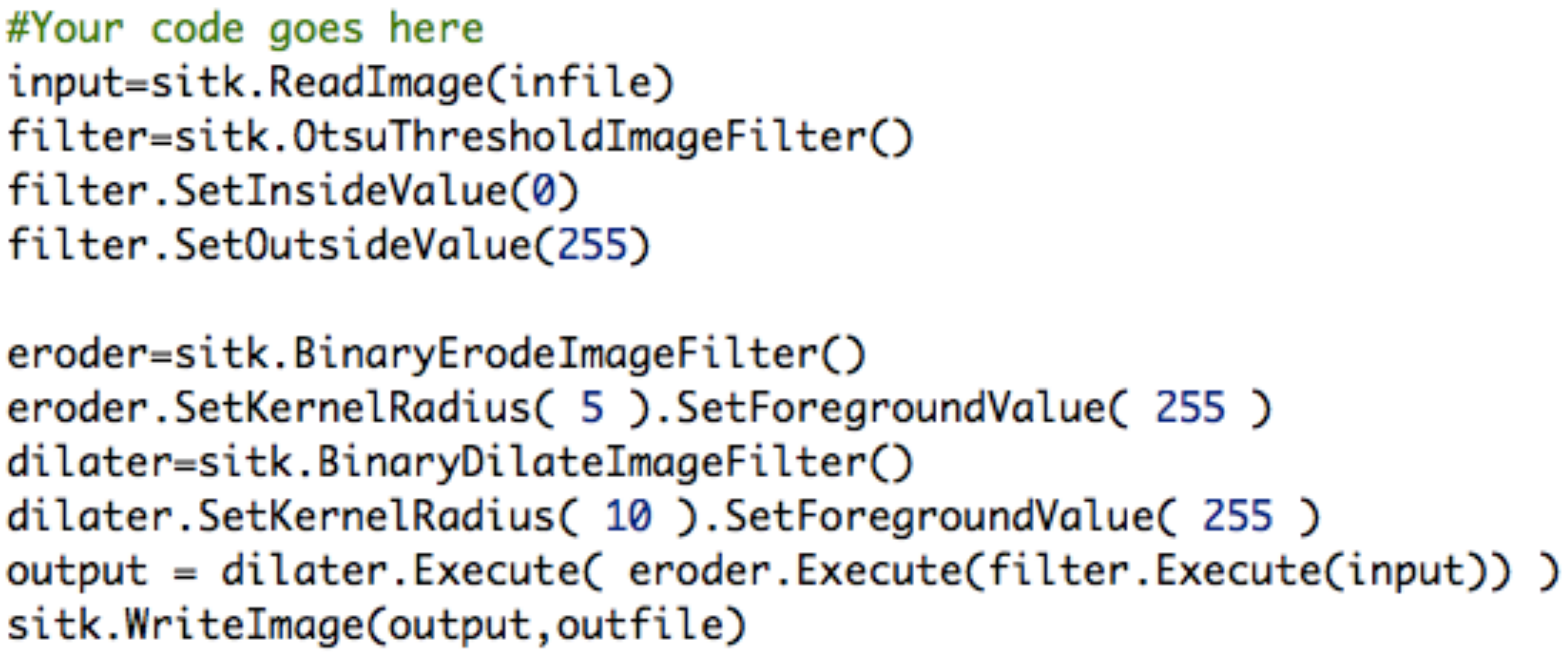
Kernel

BinaryDilate ImageFilter

Kernel

ImageFileWriter

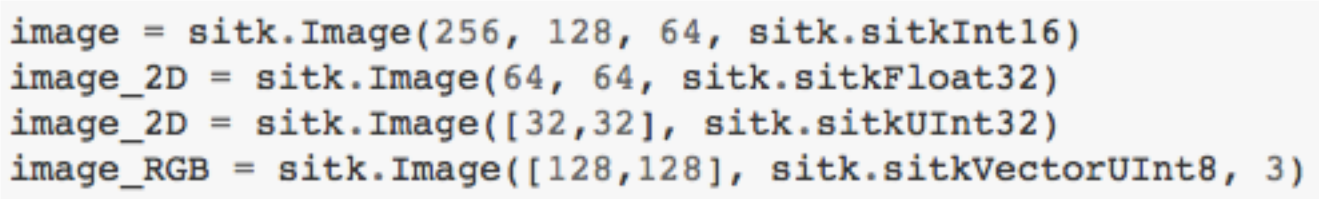
Solution



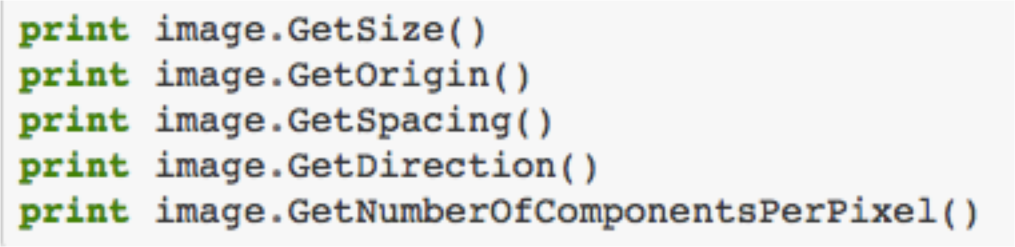
```
#Your code goes here
input=sitk.ReadImage(infile)
filter=sitk.OtsuThresholdImageFilter()
filter.SetInsideValue(0)
filter.SetOutsideValue(255)

eroder=sitk.BinaryErodeImageFilter()
eroder.SetKernelRadius( 5 ).SetForegroundValue( 255 )
dilater=sitk.BinaryDilateImageFilter()
dilater.SetKernelRadius( 10 ).SetForegroundValue( 255 )
output = dilater.Execute( eroder.Execute(filter.Execute(input)) )
sitk.WriteImage(output,outfile)
```

For Python users

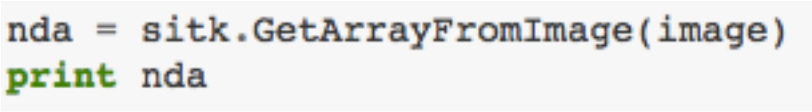
- How to take SimpleITK to numpy



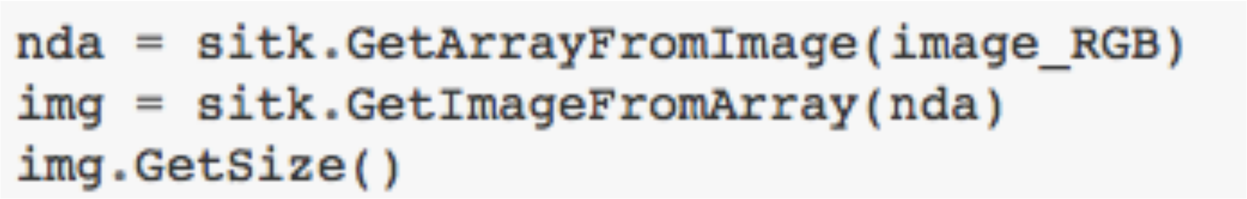
```
image = sitk.Image(256, 128, 64, sitk.sitkInt16)
image_2D = sitk.Image(64, 64, sitk.sitkFloat32)
image_2D = sitk.Image([32,32], sitk.sitkUInt32)
image_RGB = sitk.Image([128,128], sitk.sitkVectorUInt8, 3)
```



```
print image.GetSize()
print image.GetOrigin()
print image.GetSpacing()
print image.GetDirection()
print image.GetNumberOfComponentsPerPixel()
```



```
nda = sitk.GetArrayFromImage(image)
print nda
```



```
nda = sitk.GetArrayFromImage(image_RGB)
img = sitk.GetImageFromArray(nda)
img.GetSize()
```

Further resources

- How to install SimpleITK from scratch:

  http://www.itk.org/Wiki/SimpleITK/GettingStarted

- SimpleITK notebooks:

  http://simpleitk.github.io/SimpleITK-Notebooks/

Any further questions?