ITK Meets OpenCV

A New Open Source Software Resource for CV

Insight Software Consortium

June 2011

Luis Ibáñez

Patrick Reynolds

Gabe Hart

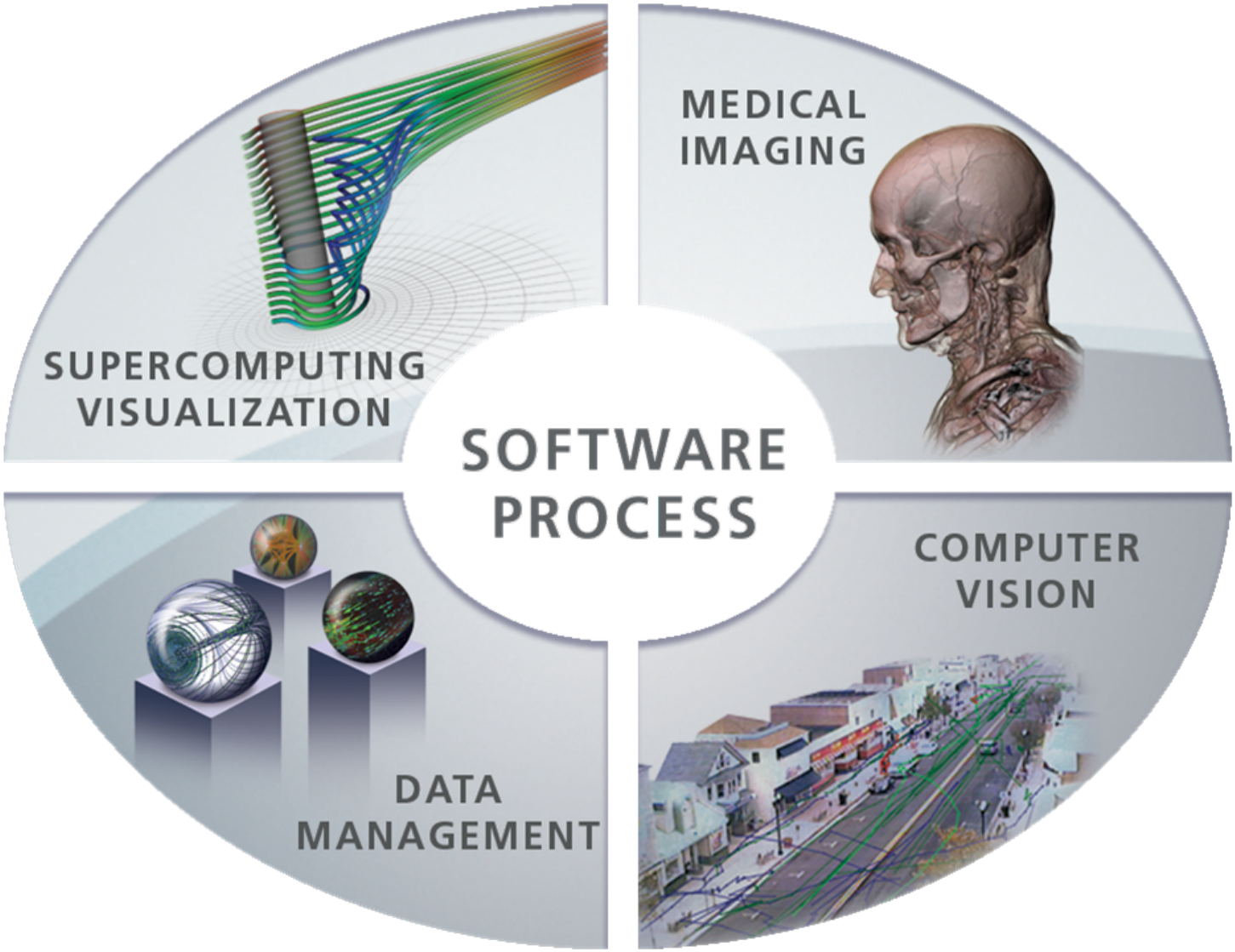
Matt Leotta

Amitha Perera

What this Tutorial is about

- Introduce the Insight Toolkit (ITK) to Computer Vision
- Hands-on code examples as a starting point for using ITK
- Preview of the upcoming video extensions to ITK
- Raffle of ASUS Eee Pad Transformer

Who is Kitware



Program

- Virtual Machines Preparation (10min)
- ITK Overview (30min)
- Short Break (10min)
- Software Environment (10min)
- ITK Introduction (30min)
- Break (30min)
- OpenCV Introduction (30min)
- OpenCV+ITK (30min)
- Short Break (10min)
- ITK Video Filters (40min)
- Wrap-up (10min)

Virtual Machines Preparation

Virtual Machines Preparation

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

USB Memory Stick 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
  - "859d59f9-ed19-4aa2-9cd6-a852ba47cdac.vmdk"
  - "OpenCV-ITK.ovf"

Install VirtualBox

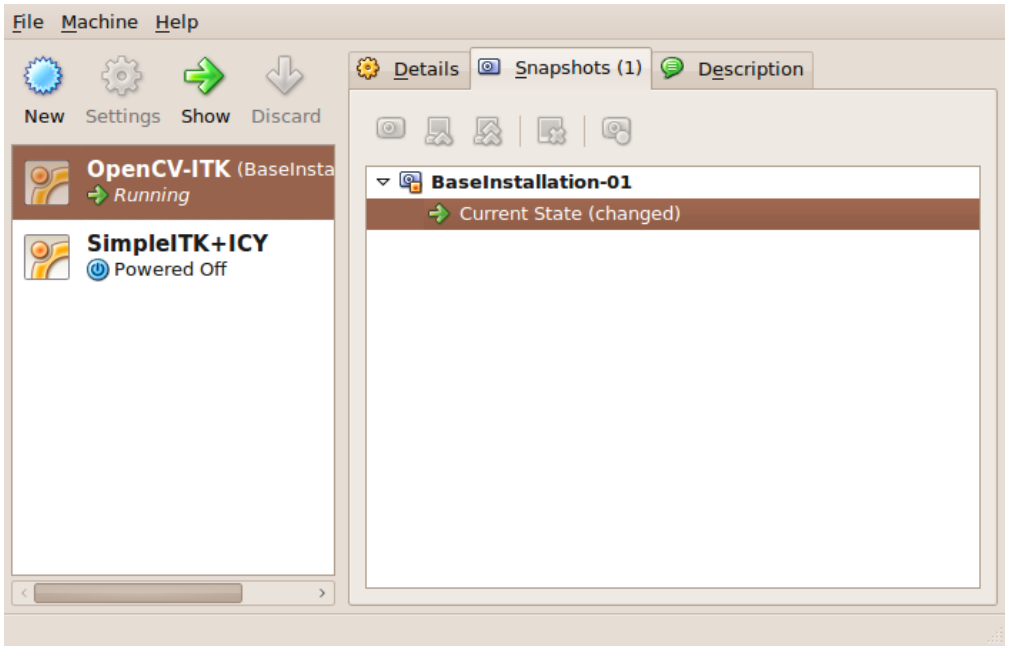
- Select the installer for your platform
- Run it

Alternative Linux Installation

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

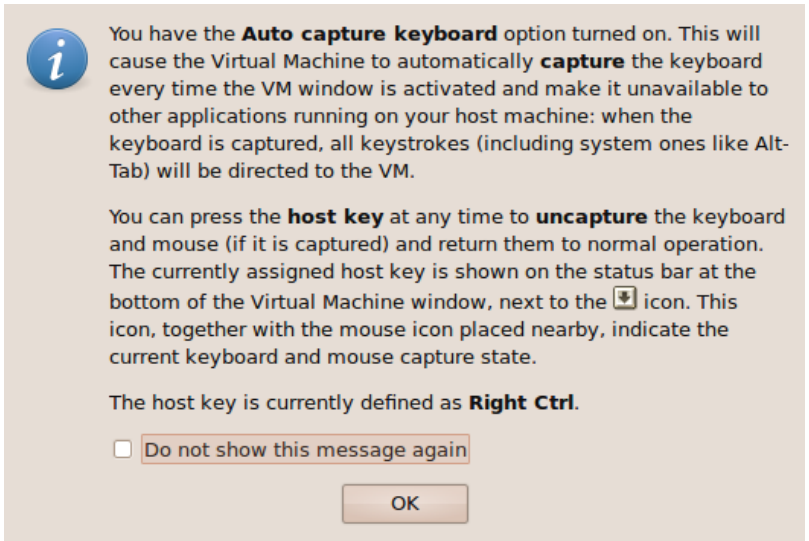
Importing the Virtual Machine

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



Booting the Virtual Machine

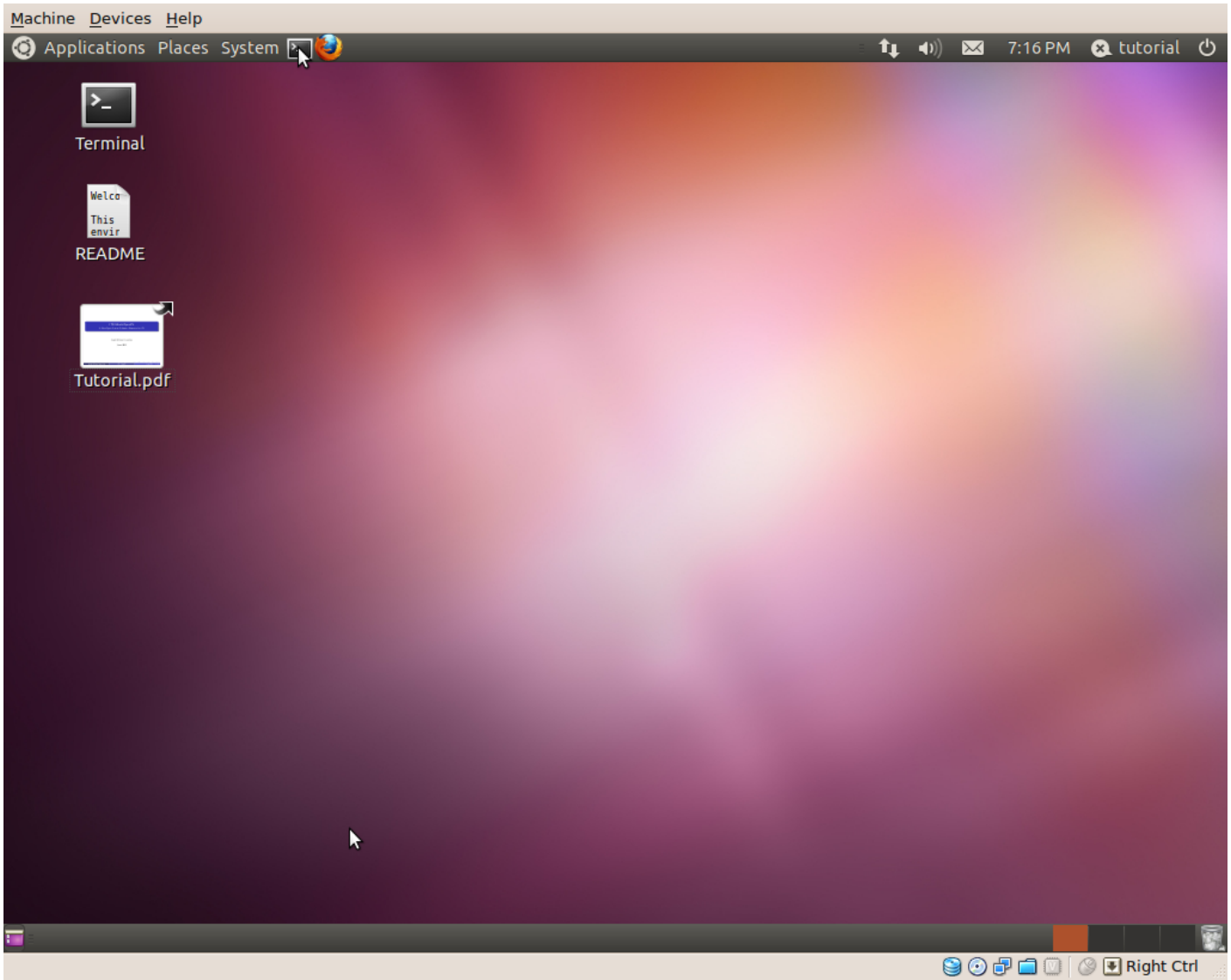
- Click on the "OpenCV-ITK" icon on the left, to select it.
- Click on the Green Arrow at the top "Show".
- The VM will start to boot and you will see the warning:



- Click "OK"

Booting the Virtual Machine

- The boot sequence should continue and you shold see:



Your Virtual Machine is Ready !

Additional Copies

- The same source trees are available in the USB key
- Outside of the VirtualBox image

ITK Overview

The Insight Toolkit (ITK)

What is ITK ?

- C++ Library
- Open Source (Apache 2.0 License)
- Generic Programming / C++ Templates
- Image Processing
- Image Segmentation
- Image Registration

What ITK is not ?

- No Visualization
- No GUI

Funding

- Funded (mostly) by the US National Library of Medicine
- With contributions from
  - National Institute of Dental and Craniofacial Research
  - National Science Foundation
  - National Eye Institute
  - National Institute of Neurological Disorders and Stroke
  - National Institute of Mental Health
  - National Institute on Deafness and Other Communication Disorders
  - National Cancer Institute

History

- Started in 2000
- Developed by Companies and Universities
  - GE Corporate Research
  - Insightful
  - Kitware
  - UNC Chapel Hill
  - University of Pennsylvania
  - University of Utah

History

- Recent contributions by a larger community
  - Harvard - Brigham and Women's Hospital
  - University of Iowa
  - Georgetown University
  - INRA - France
  - German Cancer Research Center
  - . . . and many others . . .

Project Profile

- Total lines of code: 1,886,674
- Active developers: 56 (past 12 months)
- Total developers: 146 (full history of the project)
- Top 2% largest open source teams in the world
- Estimated cost of development: $34.5 M[1]

[1] Ohloh: `http://www.ohloh.net/p/itk`

Funding

- First 10 years of development = $15M
- Refactoring of ITKv4 in 2011 = $5M
- Yearly maintenance supported by NLM

ITK by the Numbers

- Community Size
  - Users Mailing List
    - 2,071 Subscribers
    - About 400 emails/month
  - Developers Mailing List
    - 475 Subscribers
    - About 350 emails/month
  - 4,800 Downloads/month
    - from 116 countries

What ITK has to offer

- Image Processing
- Segmentation
- Registration

Image Processing

- Anisotropic Diffusion
- Mathematical Morphology
- Anti-Aliasing
- Resampling
- Distance Maps
- Bias Correction
- Diffussion Tensor Imaging
- Hessians, Laplacian, Gaussians
- Hough, Canny

Image Segmentation

- Level Sets
- Region Growing
- Watersheds
- Label Voting
- Connected Components
- Label Image Processing
- Deformable Models
- Cellular Models
- Statistical Classification
  - K-Means
  - Gaussian Mixture Models
  - Markov Random Fields

Image Registration

- Generic Framework
  - Image Metrics
  - Interpolators
  - Transforms
  - Optimizers
- Deformable Registration
  - BSplines (Free-form)
  - Demons
  - Diffeomorphic Demons
  - PDE Solving
- Multi-Resolution
- Multi-Modality

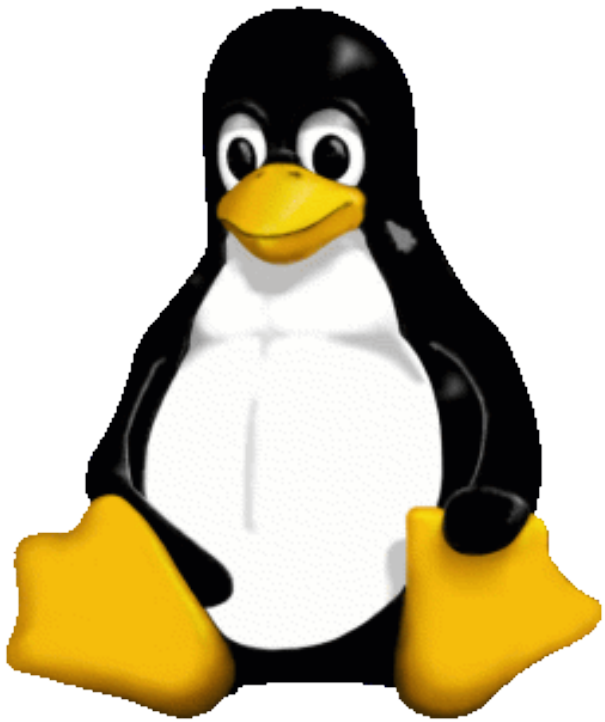
The Insight Journal

New Algorithms Every Week...

- 2,129 Registered Users
- 472 Publications
- 834 Open Reviews
- Open Access Journal
- Reproducible Research (RR)

Software Environment

Welcome to Ubuntu GNU/Linux



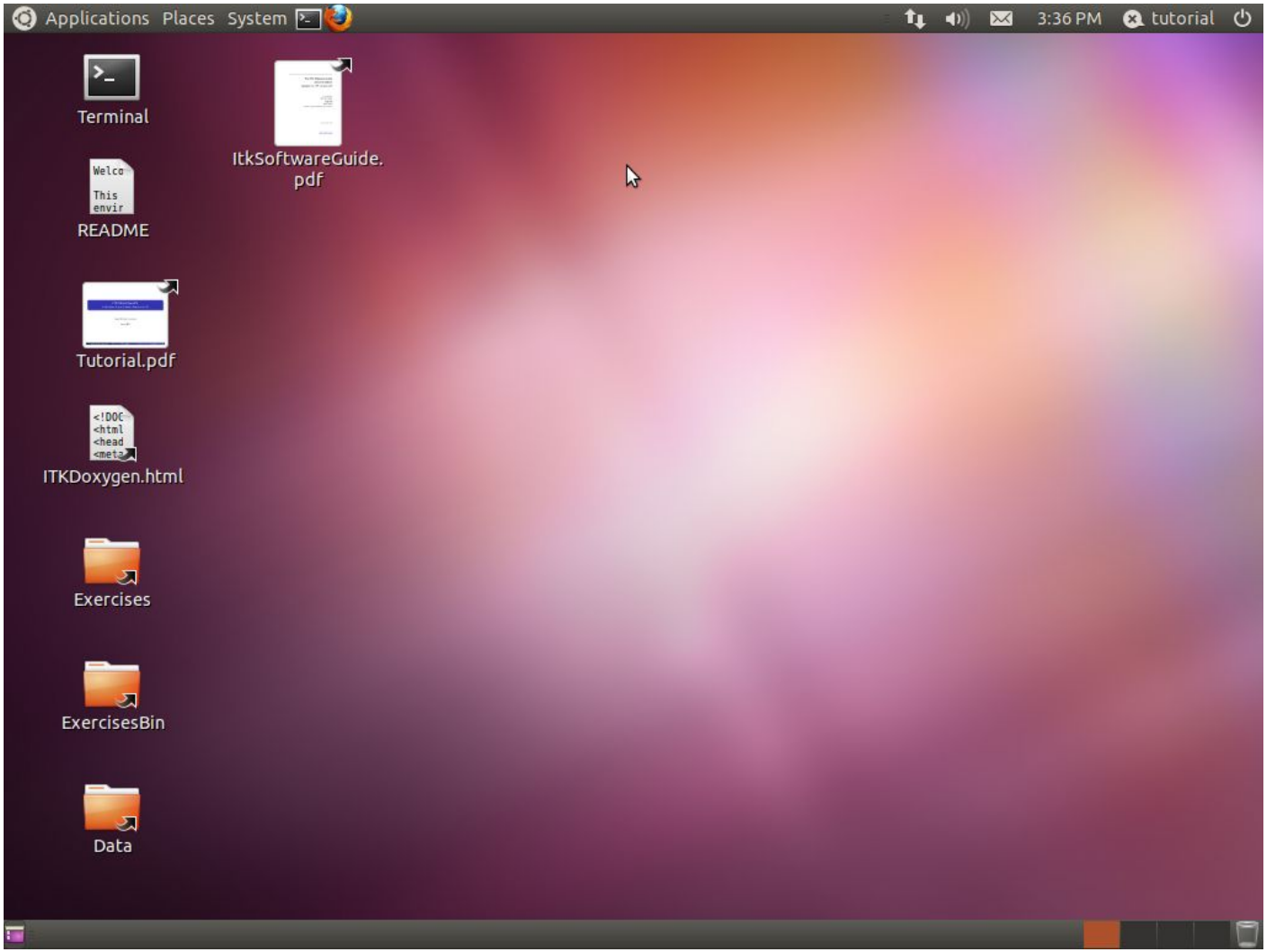
ubuntu

How to take the mouse out of the Virtual Machine

- Hit the **RIGHT CTRL** Key

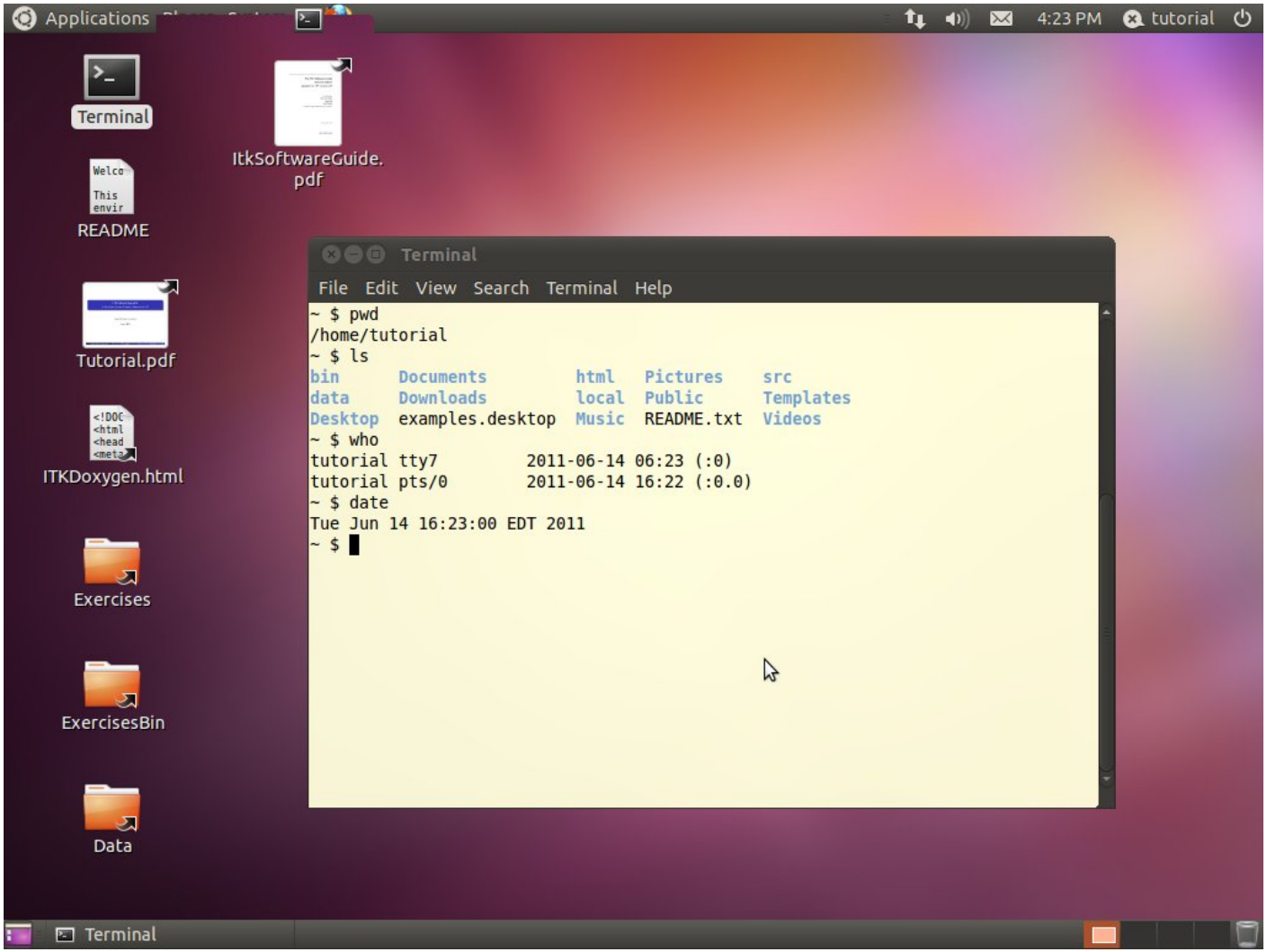
How to Open a Terminal - Icon in Upper Left Corner

To type your command line instructions



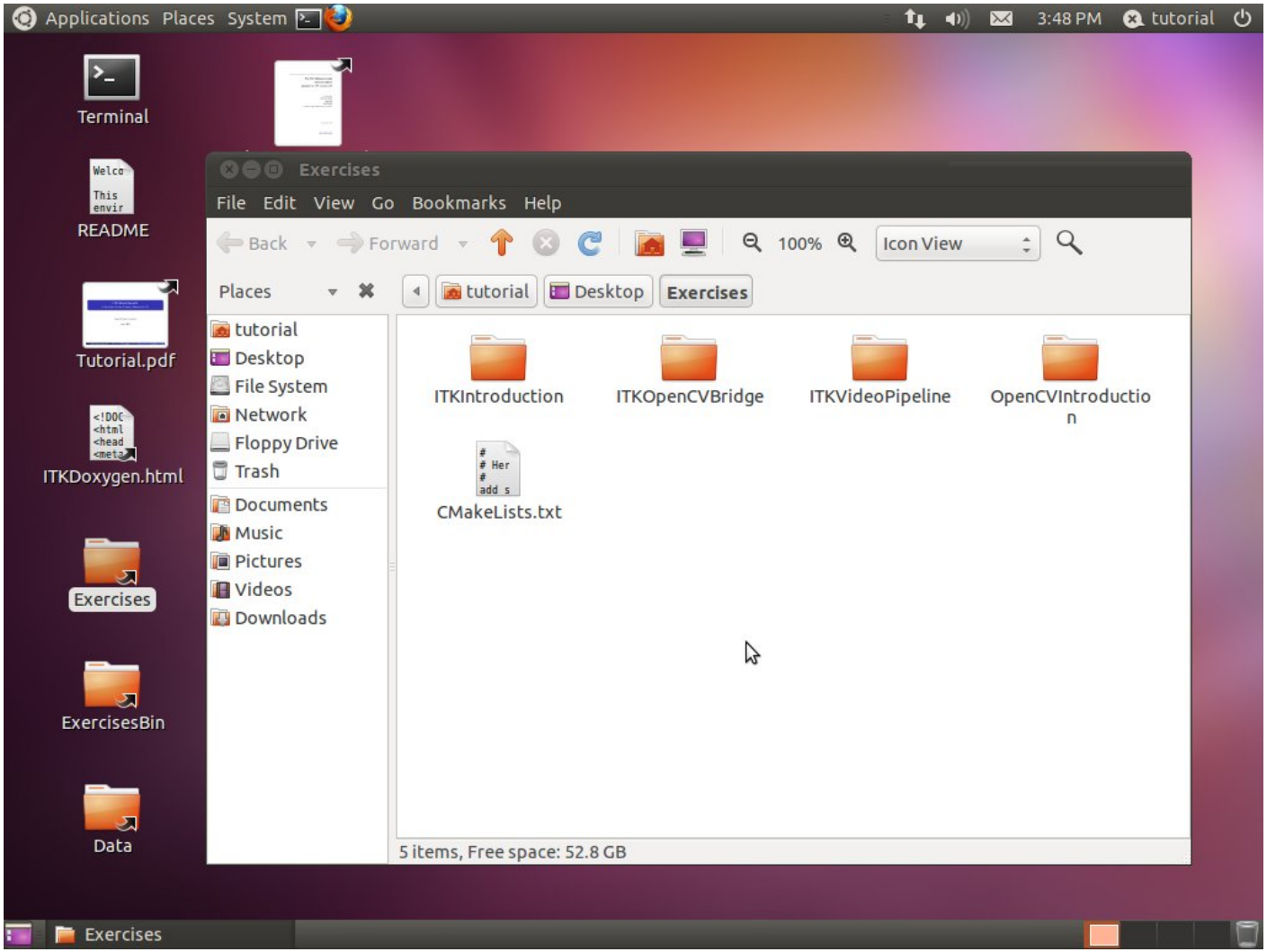
How to Open a Terminal - Icon in Upper Left Corner

To type your command line instructions



How to Navigate Directories

Double Click in Folder Icons in the Desktop



Walk through the directories

- Find source code of exercises

```
cd ~/src/ITK-OpenCV-Bridge-Tutorial/Exercises
pwd
ls
nautilus .
```

- Find binary build of exercises

```
cd ~/bin/ITK-OpenCV-Bridge-Tutorial/Exercises
pwd
ls
```

How to View Images

- Go to the directory
- Invoke "eye of gnome" eog application

```
cd ~/data
eog mandrill.png
```

- Hit ESC key to quit the application

How to View Videos

- Go to the directory
- Invoke "VideoLAN" vlc application

```
cd ~/data
vlc Walk1.mpg
```

- Hit CTRL-Q to quit the application
- or use the menus

ITK Introduction

ITK is a Templated Library

You will typically do:

- Include headers
- Pick pixel type
- Pick image dimension
- Instantiate image type
- Instantiate filter type
- Create filters
- Connect pipeline
- Run pipeline

Basic Filtering - Median Filter

ITKIntroduction/exercise1/BasicImageFilteringITK.cxx

- Include headers

```
19  #include <itkImageFileReader.h>
20  #include <itkImageFileWriter.h>
21  #include <itkMedianImageFilter.h>
```

- Read images from files
- Write images from files
- Apply a Median filter in an image

Basic Filtering - Median Filter

ITKIntroduction/exercise1/BasicImageFilteringITK.cxx

- Declare pixel types and image dimension

```
32  typedef  unsigned char  InputPixelType;
33  typedef  unsigned char  OutputPixelType;
34
35  const unsigned int Dimension = 2;
```

- Declare input and output image types

```
37  typedef itk::Image< InputPixelType,  Dimension >  InputImageType;
38  typedef itk::Image< OutputPixelType, Dimension >  OutputImageType;
```

Basic Filtering - Median Filter

ITKIntroduction/exercise1/BasicImageFilteringITK.cxx

- Declare the types for reader and writer

```
40  typedef itk::ImageFileReader< InputImageType  >  ReaderType;
41  typedef itk::ImageFileWriter< OutputImageType >  WriterType;
```

- Instantiate the reader and writer objects (source and sink)

```
43  ReaderType::Pointer reader = ReaderType::New();
44  WriterType::Pointer writer = WriterType::New();
```

- Set input and output filenames

```
46  reader->SetFileName( argv[1] );
47  writer->SetFileName( argv[2] );
```

Basic Filtering - Median Filter

ITKIntroduction/exercise1/BasicImageFilteringITK.cxx

- Declare the Median filter type

```
49  typedef itk::MedianImageFilter< InputImageType, OutputImageType > FilterType;
```

- Create the filter

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

Basic Filtering - Median Filter

ITKIntroduction/exercise1/BasicImageFilteringITK.cxx

- Define the Median kernel radius (Manhattan Radius)

```
55  indexRadius[0] = atoi( argv[3] ); // radius along x
56  indexRadius[1] = atoi( argv[4] ); // radius along y
57
58  filter->SetRadius( indexRadius );
```

- Connect the pipeline

```
60  filter->SetInput( reader->GetOutput() );
61  writer->SetInput( filter->GetOutput() );
```

Basic Filtering - Median Filter

ITKIntroduction/exercise1/BasicImageFilteringITK.cxx

- Trigger the pipeline execution by calling Update().

```
63  try
64    {
65    writer->Update();
66    }
67  catch( itk::ExceptionObject & excp )
68    {
69    std::cerr << excp << std::endl;
70    return EXIT_FAILURE;
71    }
```

- ITK uses C++ exceptions for error management
- Exceptions are typically thrown during Update() calls
- Applications must catch the exceptions and solve them

How to Configure and Build

cmake-gui

- Create a binary directory
- Configure the code with CMake
- Build (compile and link an executable)
- Run it in example image

How to Configure and Build

cmake-gui

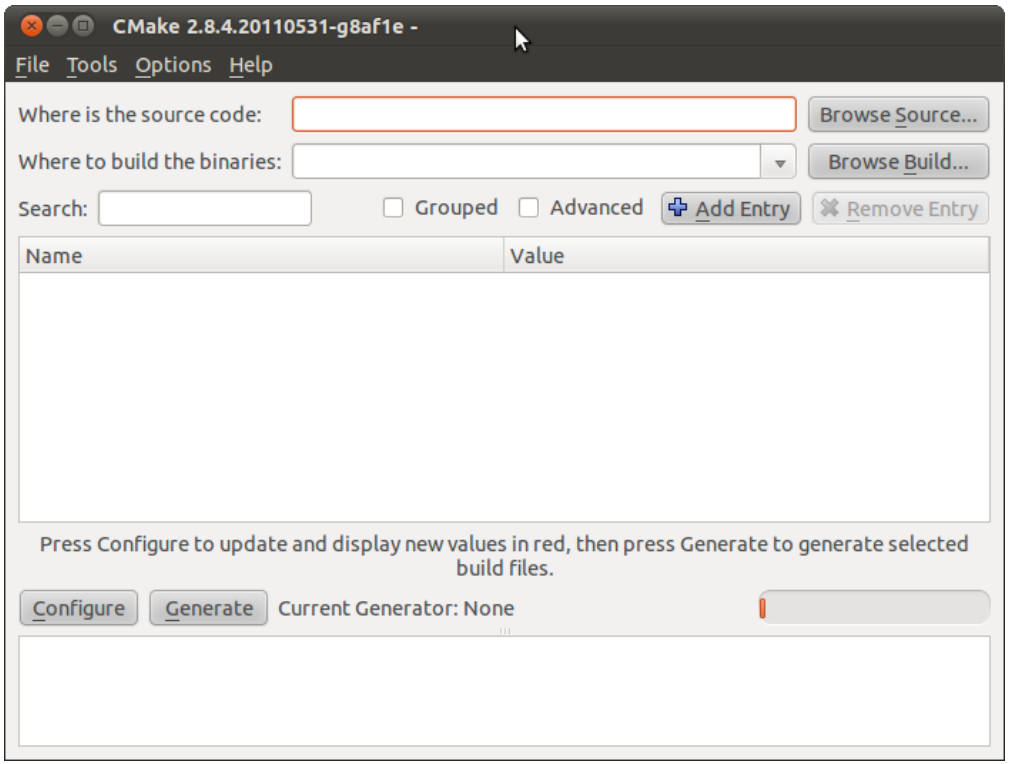
- Create a binary directory

```
cd ~/bin
mkdir itkexercise1
cd itkexercise1
```

How to Configure and Build

cmake-gui

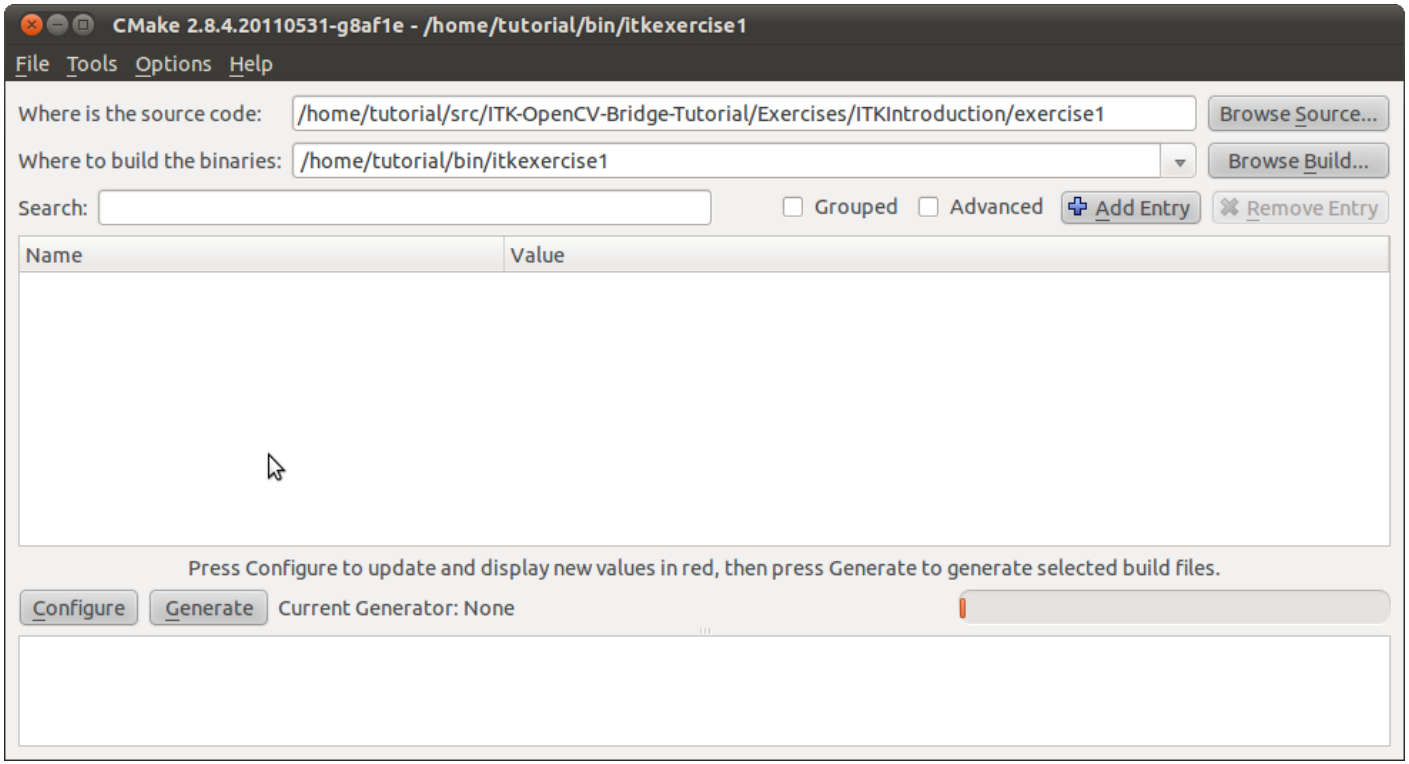
- Run “cmake-gui”



How to Configure and Build

cmake-gui

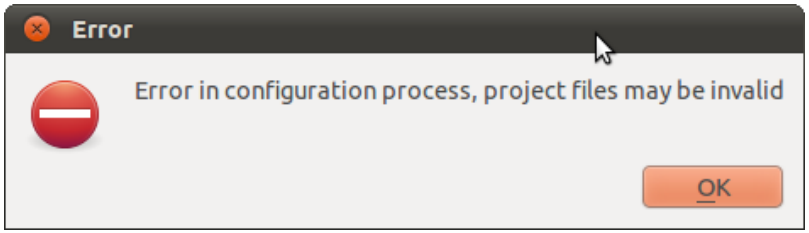
- Set "Source Directory" (where the source code is)
- Set "Binary Directory" (where to build the executable)
- Click on "Configure"



How to Configure and Build

cmake-gui

- You will get an error message

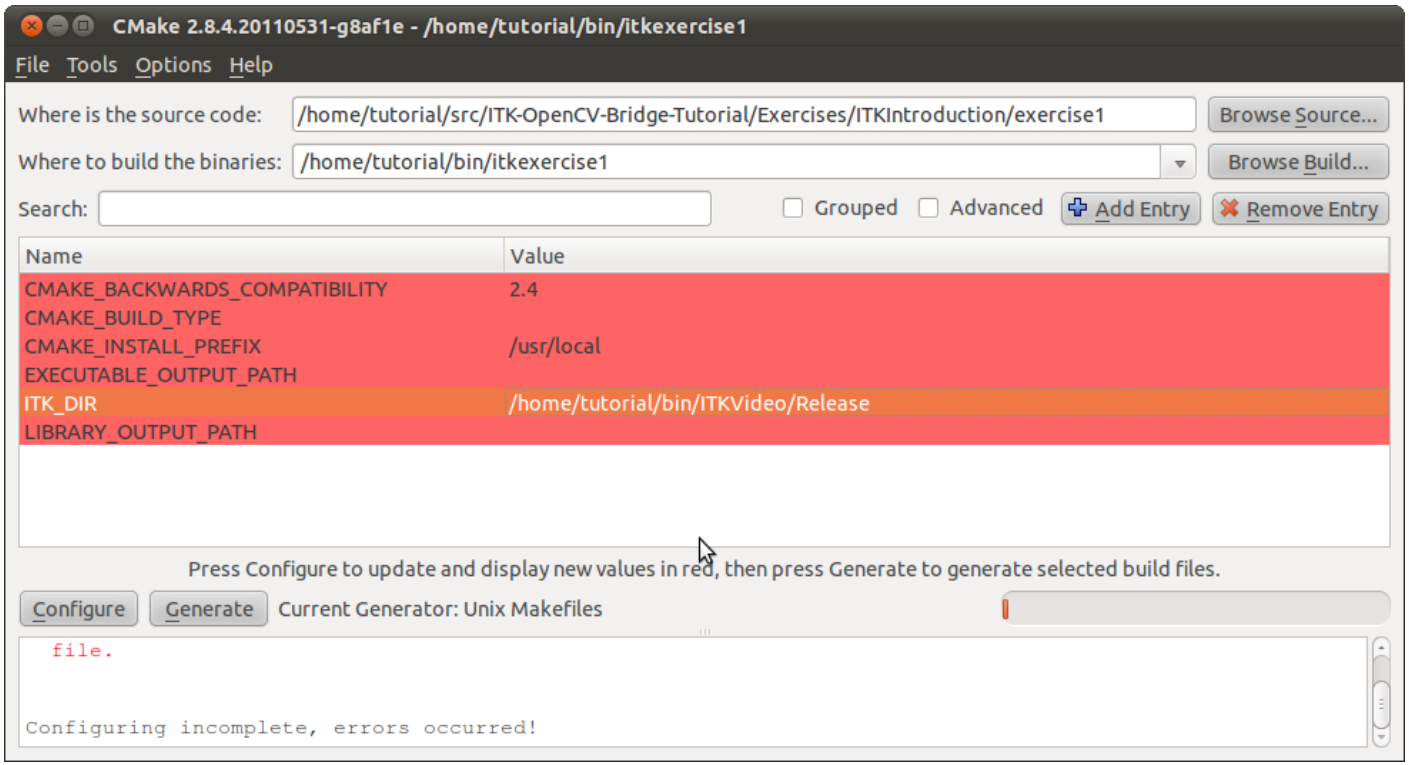


- Because the project needs ITK
- and we have not provided ITK_DIR yet

How to Configure and Build

cmake-gui

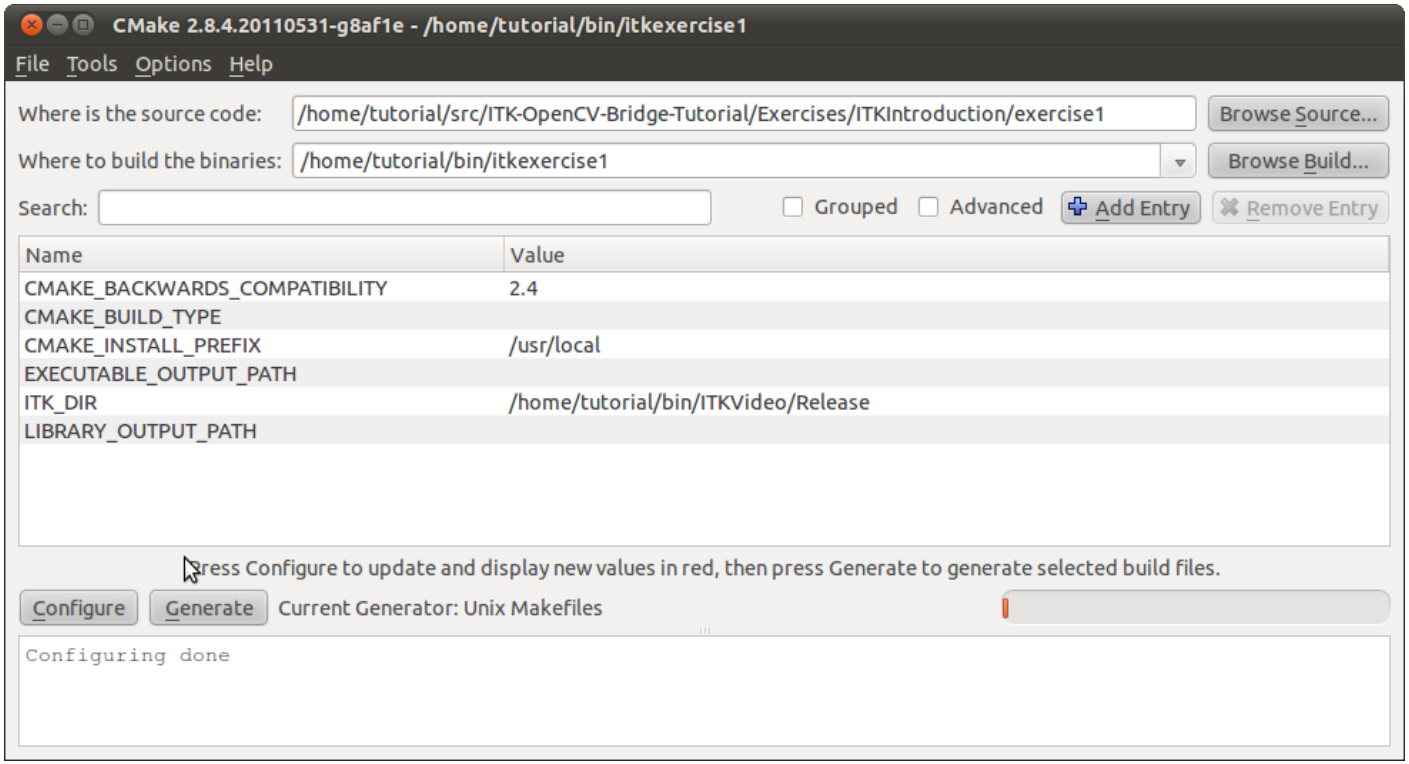
- Provide the path to ITK in the ITK_DIR variable
- /home/tutorial/bin/ITKVideo/Release



How to Configure and Build

cmake-gui

- Click on “Configure”
- Click on “Generate”



How to Build

make

- In the command line do:

```
cd  /home/tutorial/bin/itkexercise1
make
```

How to Run

/home/tutorial/bin/itkexercise1

- While in the binary directory:

  `/home/tutorial/bin/itkexercise1`

- In the command line type:

```
./BasicImageFilteringITK          \
      ~/data/mandrillgray.png       \
      ./mandrillgrayMedian.png      \
      3  3
```

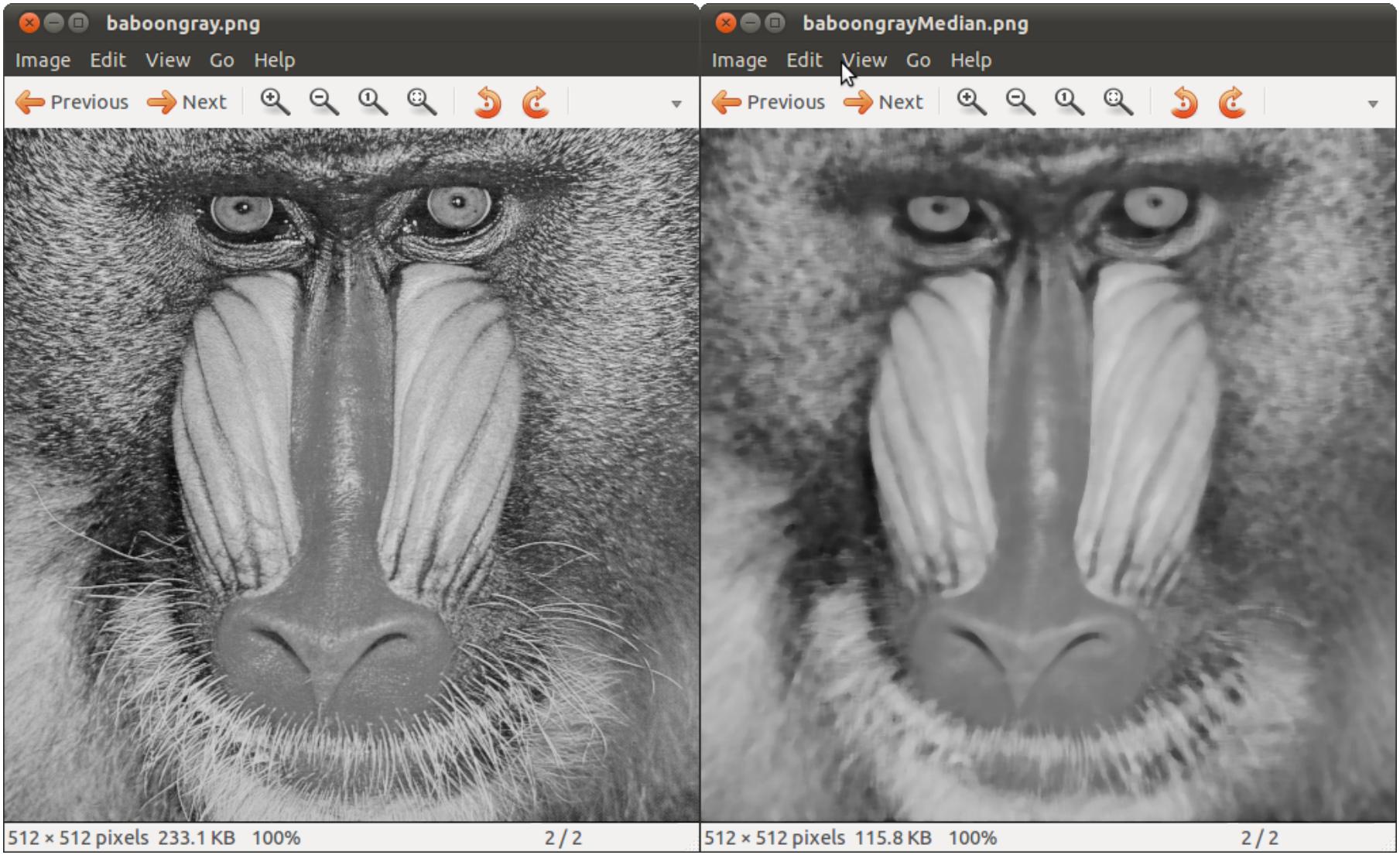
How to View the Result

Image viewing application "eye of gnome": eog

- In the command line type:

```
eog ~/data/mandrillgray.jpg &
eog ./mandrillgrayMedian.png &
```

Result of Median Filter

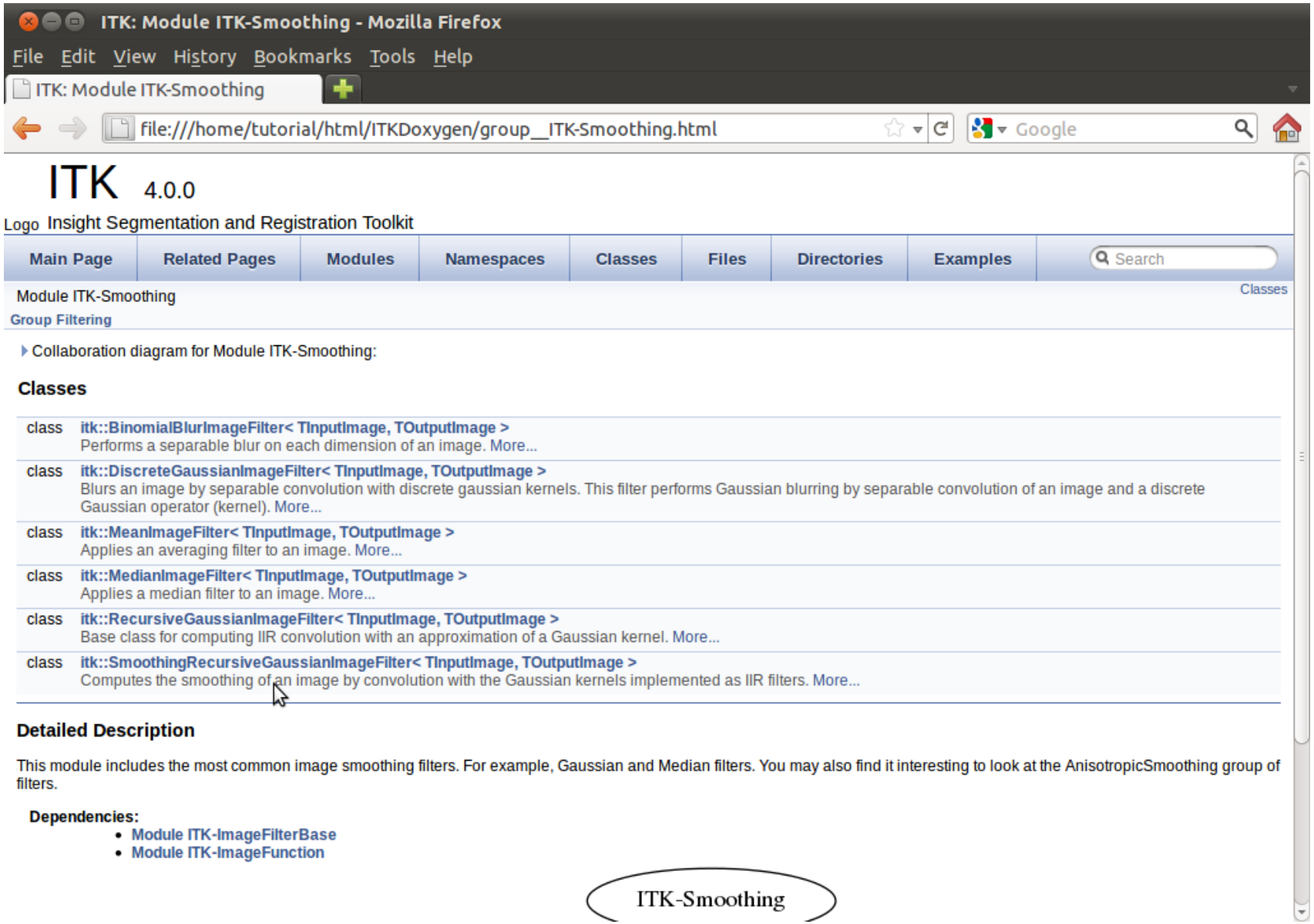


Excercise 1

Replace the filter with another one

- Select a Filter from the Doxygen documentation (e.g. MeanImageFilter)
- Replace the MedianImageFilter with the selected filter
- Recompile
- Rerun

ITK Doxygen Documentation



Excercise 1

ITKIntroduction/exercise1/BasicImageFilteringITKAnswer1.cxx

- First we replace the Header file:

```
21  #include <itkMeanImageFilter.h>
```

- Then we replace the Filter instantiation:

```
49    typedef itk::MeanImageFilter< InputImageType, OutputImageType > FilterType;
```

How to Build

make

- In the command line do:

```
cd  /home/tutorial/bin/itkexercise1
make
```

How to Run

/home/tutorial/bin/itkexercise1

- While in the binary directory:

  `/home/tutorial/bin/itkexercise1`

- In the command line type:

```
./BasicImageFilteringITK          \
      ~/data/mandrillgray.png      \
      ./mandrillgrayMean.png       \
      3  3
```

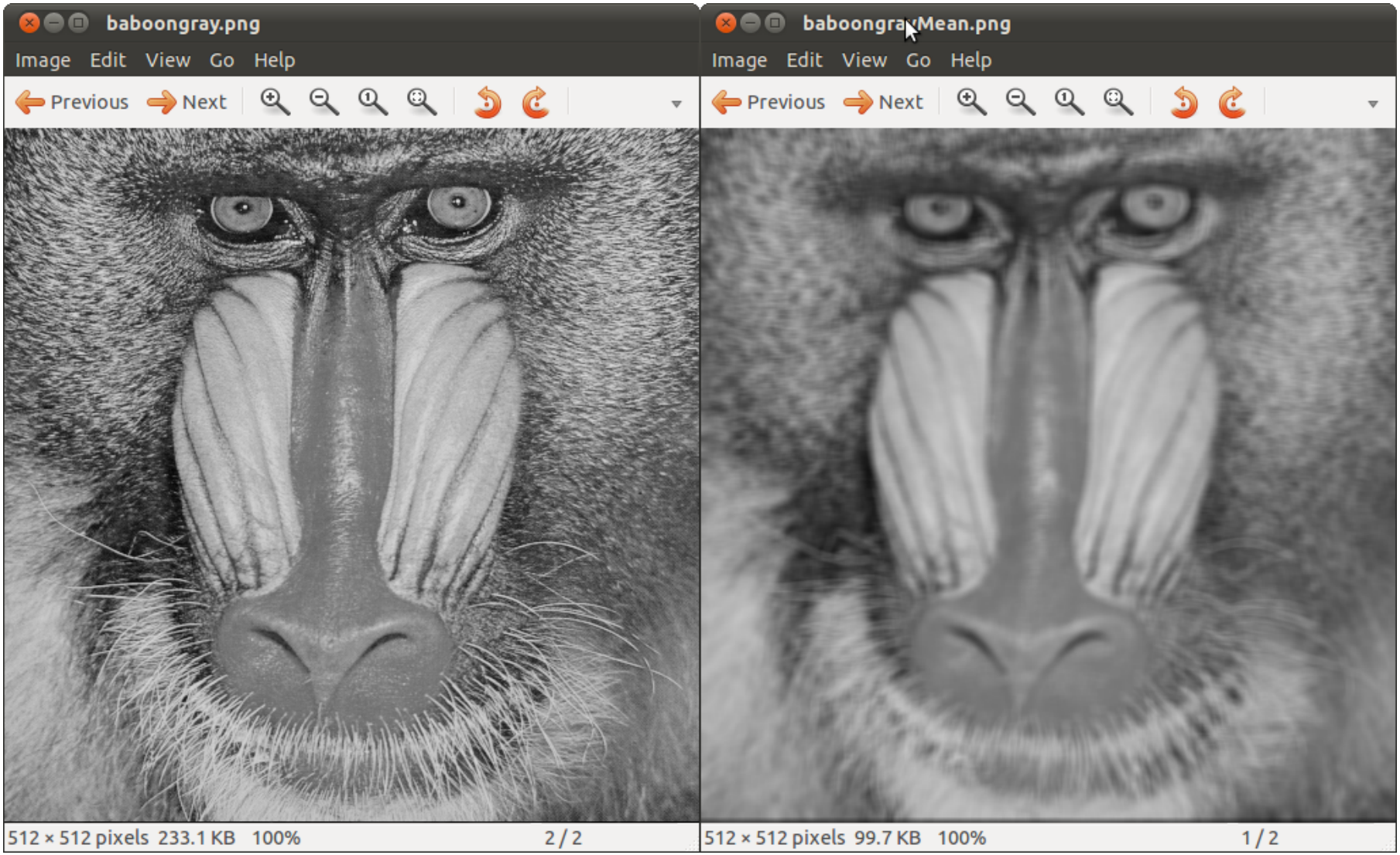
How to View the Result

Image viewing application "eye of gnome": eog

- In the command line type:

```
eog ~/data/mandrillgray.jpg &
eog ./mandrillgrayMean.png &
```

Result of Mean Filter



Find All Other Exercises

- Go to the binary directory

  ```
  cd ~/bin/ITK-OpenCV-Bridge-Tutorial/Exercises
  ```

Basic Filtering - Canny Filter

ITKIntroduction/exercise1/BasicImageFilteringITKAnswer2.cxx

- Some filters expect specific pixel types
- Canny Edge detection is an example
- Here we Cast the image before Canny
- Then we Cast/Rescale it after Canny

Basic Filtering - Canny Filter

ITKIntroduction/exercise1/BasicImageFilteringITKAnswer2.cxx

- Let's start with the relevant headers:

```
21  #include <itkCastImageFilter.h>
22  #include <itkCannyEdgeDetectionImageFilter.h>
23  #include <itkRescaleIntensityImageFilter.h>
```

- We then declare the relevant pixel types

```
34    typedef    unsigned char   InputPixelType;
35    typedef    float           RealPixelType;
36    typedef    unsigned char   OutputPixelType;
```

- Then we declare the relevant image types

```
38    typedef itk::Image< InputPixelType,  2 >   InputImageType;
39    typedef itk::Image< RealPixelType,   2 >   RealImageType;
40    typedef itk::Image< OutputPixelType, 2 >   OutputImageType;
```

Basic Filtering - Canny Filter

ITKIntroduction/exercise1/BasicImageFilteringITKAnswer2.cxx

- We declare the Casting filter and instantiate it:

```
52  typedef itk::CastImageFilter<
53    InputImageType, RealImageType >  CastFilterType;
54
55  CastFilterType::Pointer caster = CastFilterType::New();
```

- We declare and instantiate the Canny filter:

```
58  typedef itk::CannyEdgeDetectionImageFilter<
59    RealImageType, RealImageType >  FilterType;
60
61  FilterType::Pointer canny = FilterType::New();
```

- and do the same for the RescaleIntensity filter:

```
64  typedef itk::RescaleIntensityImageFilter<
65    RealImageType, OutputImageType >  RescaleFilterType;
66
67  RescaleFilterType::Pointer rescaler = RescaleFilterType::New();
```

Basic Filtering - Canny Filter

ITKIntroduction/exercise1/BasicImageFilteringITKAnswer2.cxx

- We connect the pipeline:

```
70  caster->SetInput( reader->GetOutput() );
71  canny->SetInput( caster->GetOutput() );
72  rescaler->SetInput( canny->GetOutput() );
73  writer->SetInput( rescaler->GetOutput() );
```

- Set the parameters of the Canny Edge detection filter:

```
76  canny->SetVariance( atof( argv[3] ) );
77  canny->SetLowerThreshold( atof( argv[4] ) );
78  canny->SetUpperThreshold( atof( argv[5] ) );
```

Basic Filtering - Canny Filter

ITKIntroduction/exercise1/BasicImageFilteringITKAnswer2.cxx

- Trigger the execution of the pipeline:

```
81  try
82    {
83    writer->Update();
84    }
85  catch( itk::ExceptionObject & excp )
86    {
87    std::cerr << excp << std::endl;
88    return EXIT_FAILURE;
89    }
```

- Note that the pipeline only runs when we call Update()
- That's the point where we should catch exceptions

How to Run

/home/tutorial/bin/itkexercise1

- While in the binary directory:

  `/home/tutorial/bin/itkexercise1`

- In the command line type:

```
./BasicImageFilteringITKAnswer2     \
      ~/data/mandrillgray.png         \
      ./mandrillgrayCanny.png         \
      6   1   8
```

- 6 = Variance for Gaussian
- 1 = Lower Threshold
- 8 = Upper Threshold

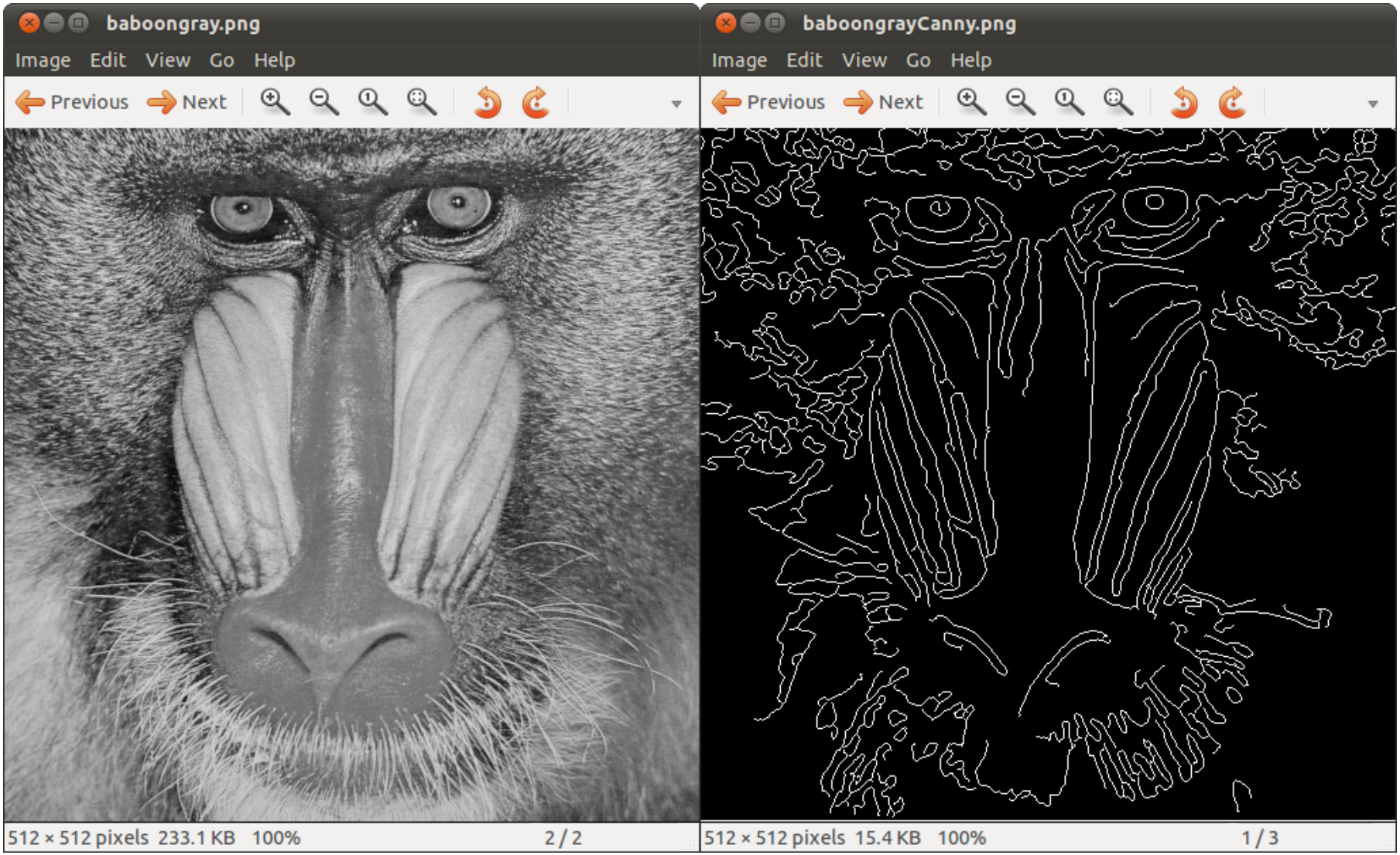
How to View the Result

Image viewing application "eye of gnome": eog

- In the command line type:

```
eog ~/data/mandrillgray.jpg &
eog ./mandrillgrayCanny.png &
```

Result of Canny Filter



OpenCV Introduction

OpenCV Overview

- What is OpenCV? OpenCV is . . .
  - an open source computer vision library.
  - written in C, but has C++ and Python APIs.
  - released under a BSD license.
  - supported and guided by Willow Garage.
  - found at `http://opencv.willowgarage.com/wiki/`.
- We will be using OpenCV 2.2 (from subversion).
- The new C++ interface will be used when possible.
- We assume you have some prior experience with OpenCV
- . . . but we will cover the basics in case you don't.

Include Header Files

OpenCVIntroduction/exercise1/BasicFilteringOpenCV.cxx

We need to include the OpenCV headers

```
19  #include <opencv2/imgproc/imgproc.hpp>
20  #include <opencv2/highgui/highgui.hpp>
```

and standard library headers for `cout` and `string`

```
22  #include <iostream>
23  #include <string>
```

The Main function

OpenCVIntroduction/exercise1/BasicFilteringOpenCV.cxx

```
26  int main ( int argc, char **argv )
27  {
28    if( argc < 2 )
29    {
30      std::cout << "Usage: "<< argv[0] <<" input_image output_image"<<std::endl;
31      return -1;
32    }
33
34    cv::Mat inputImage = cv::imread( argv[1] );
35    cv::Mat resultImage;
36    cv::medianBlur( inputImage, resultImage, 9 );
37
38    if(argc < 3)
39    {
40      std::string windowName = "Exercise 1: Basic Filtering in OpenCV";
41      cv::namedWindow( windowName, CV_WINDOW_FREERATIO);
42      cvResizeWindow( windowName.c_str(), resultImage.cols, resultImage.rows+50 );
43      cv::imshow( windowName, resultImage );
44      cv::waitKey();
45    }
46    else
47    {
48      cv::imwrite( argv[2], resultImage );
49    }
50
51    return 0;
52  }
```

Loading an Image / Applying a Filter

OpenCVIntroduction/exercise1/BasicFilteringOpenCV.cxx

- If no arguments, print usage and exit
  Note: `argv[0]` contains the executable name

```
28  if( argc < 2 )
29  {
30    std::cout << "Usage: "<< argv[0] <<" input_image output_image"<<std::endl;
31    return -1;
32  }
```

- Load an image from the specified file into a matrix object

```
34  cv::Mat inputImage = cv::imread( argv[1] );
```

- Create a matrix for the output and apply a median filter

```
35  cv::Mat resultImage;
36  cv::medianBlur( inputImage, resultImage, 9 );
```

- The last argument represent the size of the filter, 9×9 in this case

Displaying an Image

OpenCVIntroduction/exercise1/BasicFilteringOpenCV.cxx

- If no output file is specified, use HighGUI to display the resulting image.

```
38    if(argc < 3)
39    {
40      std::string windowName = "Exercise 1: Basic Filtering in OpenCV";
41      cv::namedWindow( windowName, CV_WINDOW_FREERATIO);
42      cvResizeWindow( windowName.c_str(), resultImage.cols, resultImage.rows+50 );
43      cv::imshow( windowName, resultImage );
44      cv::waitKey();
45    }
```

Create the Display Window

OpenCVIntroduction/exercise1/BasicFilteringOpenCV.cxx

- A title string is used to identify the GUI window.

```
40    std::string windowName = "Exercise 1: Basic Filtering in OpenCV";
```

- Create a named window.
  - `CV_WINDOW_FREERATIO` is needed for display using OpenGL.

```
41    cv::namedWindow( windowName, CV_WINDOW_FREERATIO );
```

- Resize the window to match the image size.
  - `cvResizeWindow()` is a C function with no C++ API equivalent.
  - It is not in the `cv` namespace.
  - It requires a `const char*` name, provided by `std::string::c_str()`.
  - The +50 in height to account for the height of tool/status bars.

```
42    cvResizeWindow( windowName.c_str(), resultImage.cols, resultImage.rows+50 );
```

Display and Wait

OpenCVIntroduction/exercise1/BasicFilteringOpenCV.cxx

- Show the image in the previously created window.

```
43      cv::imshow( windowName, resultImage );
```

- Wait for the user to press a key, then continue.

```
44      cv::waitKey();
```

Saving an Image

OpenCVIntroduction/exercise1/BasicFilteringOpenCV.cxx

- If an output file is specified, write the image to the specified file

```
46    else
47    {
48      cv::imwrite( argv[2], resultImage );
49    }
```

Exercise 1

OpenCVIntroduction/exercise1/BasicFilteringOpenCV.cxx

- Replace the median filter with a Canny edge detector

```
35    cv::Mat resultImage;
36    cv::medianBlur( inputImage, resultImage, 9 );
```

- Hint: The OpenCV Canny function has this signature

```
void Canny(const Mat& image, Mat& edges, double threshold1, double threshold2);
```

- Canny requires grayscale image input, but our images are color
- Hint: Use this function to convert a color space

```
void cvtColor(const Mat& src, Mat& dst, int code);
```

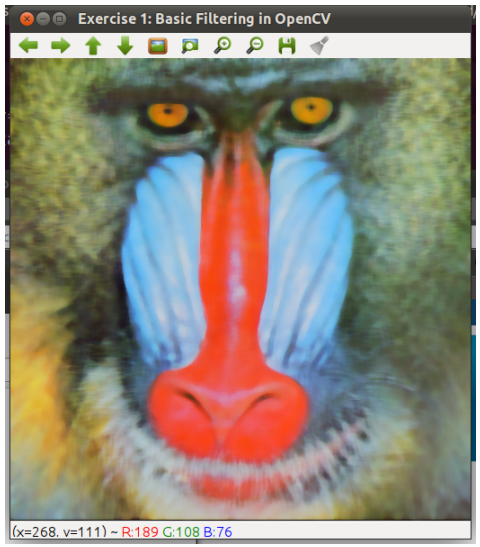
- Use `CV_BGR2GRAY` for code to convert BGR color to gray.

Exercise 1: Answer

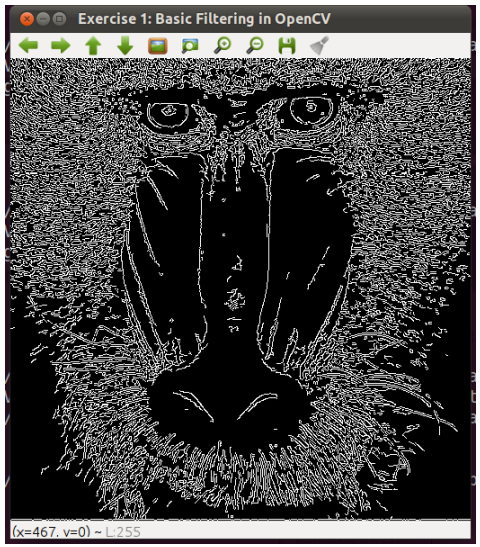
OpenCVIntroduction/exercise1/BasicFilteringOpenCVAnswer.cxx

```
34    cv::Mat inputImage = cv::imread( argv[1] );
35    cv::Mat grayImage;
36    cv::Mat resultImage;
37    cv::cvtColor(inputImage, grayImage, CV_BGR2GRAY);
38    cv::Canny( grayImage, resultImage, 128, 255 );
```

- Run `./BasicFilteringOpenCV ~/data/mandrill.png`



Median Filter



Canny Edges

Exercise 2: Basic Video Filtering

OpenCVIntroduction/exercise2/BasicVideoFilteringOpenCV.cxx

- Video filtering is similar to image filtering, except
  - use a `VideoCapture` object to read a video file.
  - loop over each frame in the video and process each one.
  - display or encode each output frame within the loop.

Loading a Video File

OpenCVIntroduction/exercise2/BasicVideoFilteringOpenCV.cxx

- The first step is to open the video file and parse the headers.

```
88  cv::VideoCapture vidCap( argv[1] );
89  if( !vidCap.isOpened() )
90  {
91    std::cerr << "Unable to open video file: "<< argv[1] << std::endl;
92    return -1;
93  }
```

- If unable to parse the headers then terminate the program.

Display or Save

OpenCVIntroduction/exercise2/BasicVideoFilteringOpenCV.cxx

- Same command line usage as before for selecting between displaying or saving results.
- The processing code has been encapsulated into functions.
- If only input video is specified, then display the resulting video.

```
95   if(argc < 3)
96   {
97     processAndDisplayVideo( vidCap );
98   }
```

- If a second file is specified, then save the resulting video to the file.

```
99   else
100  {
101    processAndSaveVideo( vidCap, argv[2] );
102  }
```

Prepare the Display Window

OpenCVIntroduction/exercise2/BasicVideoFilteringOpenCV.cxx

- This function processes the video and displays the results.

```
35  void processAndDisplayVideo(cv::VideoCapture& vidCap)
36  {
```

- Use the `get()` member function to access video properties: frame rate, width, and height.

```
37    double frameRate = vidCap.get( CV_CAP_PROP_FPS );
38    int width = vidCap.get( CV_CAP_PROP_FRAME_WIDTH );
39    int height = vidCap.get( CV_CAP_PROP_FRAME_HEIGHT );
```

- Setup the HighGUI window as in the previous example.

```
41    std::string windowName = "Exercise 2: Basic Video Filtering in OpenCV";
42    cv::namedWindow( windowName, CV_WINDOW_FREERATIO);
43    cvResizeWindow( windowName.c_str(), width, height+50 );
```

Process and Display

OpenCVIntroduction/exercise2/BasicVideoFilteringOpenCV.cxx

- Number of milliseconds to wait before drawing the next frame.

```
45    unsigned delay = 1000 / frameRate;
```

- Loop until no more frames can be read from the video capture object.

```
47    cv::Mat frame;
48    while( vidCap.read(frame) )
49    {
50      cv::Mat outputFrame = processFrame( frame );
51      cv::imshow( windowName, outputFrame );
52
53      if( cv::waitKey(delay) >= 0 )
54      {
55        break;
56      }
57    }
```

- Call `processFrame()` on each frame and and display the result.
- `waitkey(`*`delay`*`)` returns positive value indicating the key pressed or returns -1 after *`delay`* ms have passed.

Create a Video Writer

OpenCVIntroduction/exercise2/BasicVideoFilteringOpenCV.cxx

- This function processes the video and saves the results.

```
62 void processAndSaveVideo(cv::VideoCapture& vidCap, const std::string& filename)
63 {
```

- Get the video properties as before.

```
64   double frameRate = vidCap.get( CV_CAP_PROP_FPS );
65   int width = vidCap.get( CV_CAP_PROP_FRAME_WIDTH );
66   int height = vidCap.get( CV_CAP_PROP_FRAME_HEIGHT );
```

- Specify that the output video will be encoded in DIVX format.
- Create a video writer object with the specified filename.

```
68   int fourcc = CV_FOURCC('D','I','V','X');
69   cv::VideoWriter vidWrite( filename, fourcc, frameRate,
70                             cvSize(width, height) );
```

Process and Save

OpenCVIntroduction/exercise2/BasicVideoFilteringOpenCV.cxx

- Loop until no more frames can be read from the video capture object.

```
71  cv::Mat frame;
72  while( vidCap.read(frame) )
73  {
74    cv::Mat outputFrame = processFrame( frame );
75    vidWrite << outputFrame;
76  }
```

- Call `processFrame()` on each frame.
- Write the resulting image to the output video file with the insertion operator (<<).

Exercise 2

OpenCVIntroduction/exercise2/BasicVideoFilteringOpenCV.cxx

- Implement `processFrame()` by adding the Canny edge detector.

```
27  cv::Mat processFrame( const cv::Mat& inputImage )
28  {
29    // ADD FRAME PROCESSING CODE HERE
30    return inputImage;
31  }
```

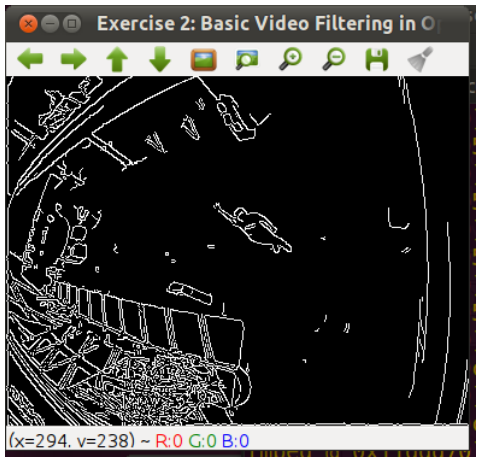
- Saving to a video requires color images, but Canny produces grayscale images.
- Hint: Use `cvtColor()` again to convert back to color.
- Use code `CV_GRAY2BGR` for `code` to convert gray to BGR.

Exercise 2: Answer

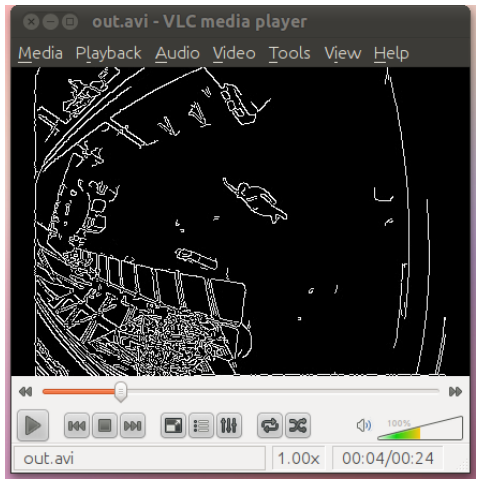
OpenCVIntroduction/exercise2/BasicVideoFilteringOpenCVAnswer.cxx

```
27  cv::Mat processFrame( const cv::Mat& inputImage )
28  {
29    cv::Mat grayImage, edgeImage, resultImage;
30    cv::cvtColor(inputImage, grayImage, CV_BGR2GRAY);
31    cv::Canny( grayImage, edgeImage, 128, 255 );
32    cv::cvtColor(edgeImage, resultImage, CV_GRAY2BGR);
33
34    return resultImage;
35  }
```

- Run `./BasicVideoFilteringOpenCV ~/data/Walk1.mpg`



HighGUI Display



Saved Video (Opened in vlc)

ITK OpenCV Bridge

Introduction

- ITK Module for working with other libraries
- Moving frame and/or video data between OpenCV and ITK
- Bring biomedical and computer vision folks together
- `https://github.com/itkvideo/ITK`

Funding Source

- Contract from the National Library of Medicine (HHSN276201000579P)
- Algorithms, Adapters & Data Distribution Outreach 2010: Increasing the Impact of the Insight Toolkit (ITK)

Design Choices

- OpenCV users and ITK users should both be comfortable
- Image to image utility functions
- cv::Source to itk::VideoStream
- (Later) Focus on performance

Basic Image Filtering (Revisited)

Include Header Files

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridge.cxx

- We include ITK and OpenCV headers (like before):

```
20 #include <opencv2/imgproc/imgproc.hpp>
21 #include <opencv2/highgui/highgui.hpp>
```

```
23 #include <itkImage.h>
24 #include <itkMedianImageFilter.h>
```

- We also need to include the bridge header:

```
25 #include <itkOpenCVImageBridge.h>
```

Basic Layout

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridge.cxx

- The basic layout of this file is the same as the OpenCV Examples:

```
27  int main ( int argc, char **argv )
28  {
29    if( argc < 2 )
30    {
31      std::cout << "Usage: "<< argv[0] <<" input_image output_image"<<std::endl;
32      return -1;
33    }
34
35    cv::Mat inputImage = cv::imread( argv[1] );
```

```
69    if(argc < 3)
70    {
71      std::string windowName = "Exercise 1: Basic Filtering in OpenCV & ITK";
72      cv::namedWindow( windowName, CV_WINDOW_FREERATIO);
73      cvResizeWindow( windowName.c_str(), resultImage.cols, resultImage.rows+50 );
74      cv::Mat scaled;
75      resultImage.convertTo( scaled, CV_8UC1 );
76      cv::imshow( windowName, scaled );
77      cv::waitKey();
78    }
79    else
80    {
81      cv::imwrite( argv[2], resultImage );
82    }
83
84    return 0;
```

Adding ITK

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridge.cxx

- The type definitions should also be familiar from the ITK Material:

```
38  typedef unsigned char                                   OutputPixelType;
39  const unsigned int Dimension =                          2;
40  typedef itk::Image< InputPixelType, Dimension >         InputImageType;
41  typedef itk::Image< OutputPixelType, Dimension >        OutputImageType;
42  typedef itk::OpenCVImageBridge                          BridgeType;
43  typedef itk::MedianImageFilter< InputImageType, OutputImageType >
44                                                          FilterType;
```

- However, notice the bridge class. It contains the conversion function between OpenCV and ITK.

```
42  typedef itk::OpenCVImageBridge                          BridgeType;
```

From OpenCV to ITK

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridge.cxx

- We call our conversion function to go from a cv::Mat to an itk::Image

```
47    BridgeType::CVMatToITKImage< InputImageType >( inputImage );
```

Filtering with ITK

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridge.cxx

- The median filtering is normal ITK code, but we do not connect our output to a writer

```
49  FilterType::Pointer filter = FilterType::New();
50  InputImageType::SizeType neighborhoodRadius;
51  neighborhoodRadius[0] = 9;
52  neighborhoodRadius[1] = 9;
53  filter->SetRadius( neighborhoodRadius );
54
55  filter->SetInput( itkImage );
56  try
57    {
58    filter->Update();
59    }
60  catch( itk::ExceptionObject & excp )
61    {
62    std::cerr << excp << std::endl;
63    return EXIT_FAILURE;
64    }
```

- Instead, we set it to our conversion function

```
66  cv::Mat resultImage =
67    BridgeType::ITKImageToCVMat< OutputImageType >( filter->GetOutput() );
```

Running the Example

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridge.cxx

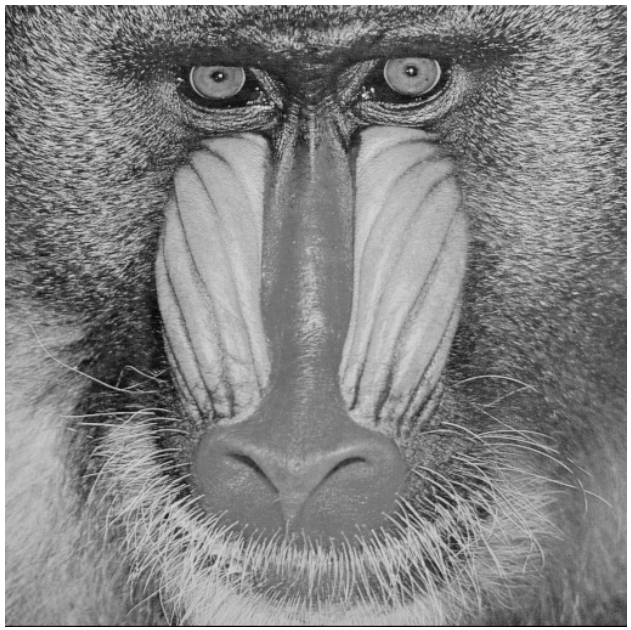
- Run the example with the following command

```
./BasicFilteringITKOpenCVBridge      \
~/data/mandrillgray.png              \
./mandrillgrayMedian.png
```

Viewing the Results

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridge.cxx

- Running the example the same way as before, we see a nicely median-filtered image.



Original

Median Filter

- Now, the fun part. Let's modify our example to use Curvature Flow, an anisotropic diffusion filter built into ITK.

Exercise 1

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridge.cxx

- Hint 1: Curvature Flow requires `float` as the output pixel type.
- Hint 2: Curvature Flow does not take a radius parameter. It's salient functions are:

```
50    filter->SetTimeStep( 0.5 );
51    filter->SetNumberOfIterations( 20 );
```

Exercise 1: Answer

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridgeAnswer.cxx

- First, change the included filter

```
24  #include <itkCurvatureFlowImageFilter.h>
```

- The type definitions also need to change to reflect the new filter and output image types.

```
37  typedef unsigned char                                   InputPixelType;
38  typedef float                                           OutputPixelType;
39  const unsigned int Dimension =                          2;
40  typedef itk::Image< InputPixelType, Dimension >         InputImageType;
41  typedef itk::Image< OutputPixelType, Dimension >        OutputImageType;
42  typedef itk::OpenCVImageBridge                          BridgeType;
43  typedef itk::CurvatureFlowImageFilter< InputImageType, OutputImageType >
44                                                          FilterType;
```

Exercise 1: Answer

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridgeAnswer.cxx

- The semantics of calling the filter also has to change:

```
50    filter->SetTimeStep( 0.5 );
51    filter->SetNumberOfIterations( 20 );
```

- That's it! Now you're using a "better" blurring scheme.

Exercise 1: Answer

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridgeAnswer.cxx

- The final pipeline should look like this:

```
35  cv::Mat inputImage = cv::imread( argv[1] );
```

```
46  InputImageType::Pointer itkImage =
47    BridgeType::CVMatToITKImage< InputImageType >( inputImage );
48
49  FilterType::Pointer filter = FilterType::New();
50  filter->SetTimeStep( 0.5 );
51  filter->SetNumberOfIterations( 20 );
52
53  filter->SetInput( itkImage );
54  try
55    {
56    filter->Update();
57    }
58  catch( itk::ExceptionObject & excp )
59    {
60    std::cerr << excp << std::endl;
61    return EXIT_FAILURE;
62    }
```

Running the Answer

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridgeAnswer.cxx

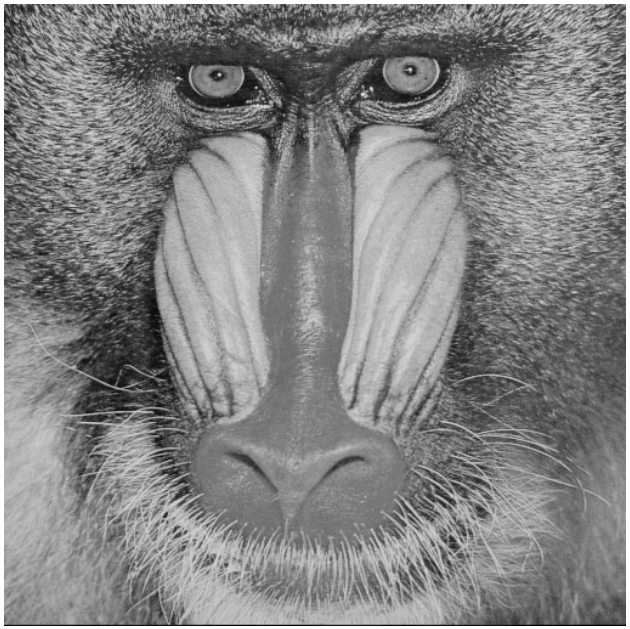
- Run the answer with the following command

```
./BasicFilteringITKOpenCVBridgeAnswer      \
~/data/mandrillgray.png                    \
./mandrillgrayCurvatureFlow.png
```

Exercise 1: Answer

ITKOpenCVBridge/exercise1/BasicFilteringITKOpenCVBridgeAnswer.cxx

- The results should look like this:



Original

Median Filter

Curvature Flow

Basic Video Filtering (Revisited)

Basic Layout

ITKOpenCVBridge/exercise2/BasicVideoFilteringITKOpenCVBridge.cxx

- The basic layout of this file is the same as the OpenCV Video Examples:
- There are three functions as before.

```
28  cv::Mat processFrame( const cv::Mat& inputImage )
```

```
63  void processAndDisplayVideo(cv::VideoCapture& vidCap)
```

```
89  void processAndSaveVideo(cv::VideoCapture& vidCap, const std::string& filename)
```

- We only need to modify `processFrame` to incorporate ITK.

Filtering with ITK

ITKOpenCVBridge/exercise2/BasicVideoFilteringITKOpenCVBridge.cxx

- The median filtering is again normal ITK code.

```
28  cv::Mat processFrame( const cv::Mat& inputImage )
29  {
```

```
39    FilterType::Pointer filter = FilterType::New();
40    InputImageType::SizeType neighborhoodRadius;
41    neighborhoodRadius[0] = 9;
42    neighborhoodRadius[1] = 9;
43    filter->SetRadius( neighborhoodRadius );
44
45    InputImageType::Pointer itkFrame =
46      BridgeType::CVMatToITKImage< InputImageType >( inputImage );
47
48    filter->SetInput(itkFrame);
49    try
50      {
51      filter->Update();
52      }
53    catch( itk::ExceptionObject & excp )
54      {
55      std::cerr << excp << std::endl;
56      }
57
58    return BridgeType::ITKImageToCVMat<OutputImageType>( filter->GetOutput(),
59                                                         true );
60  }
```

Running the Example

ITKOpenCVBridge/exercise2/BasicVideoFilteringITKOpenCVBridge.cxx

- Run the example with the following command

```
./BasicVideoFilteringITKOpenCVBridge      \
~/data/Walk1.mpg                          \
./Walk1Median.mpg
```

Exercise 2

ITKOpenCVBridge/exercise2/BasicVideoFilteringITKOpenCVBridge.cxx

- Let's modify this example to use the same canny edge detection from the ITK examples
- Hint 1: You should use the following function instead of ITK's cast image filter for producing output

```
72    frameOut.convertTo( frameOut, CV_8U );
```

- Hint 2: You should only have to modify `processFrame`

Exercise 2: Answer

ITKOpenCVBridge/exercise2/BasicVideoFilteringITKOpenCVBridgeAnswer.cxx

- You should now have some extra typedefs:

```
39  typedef itk::CastImageFilter< InputImageType, RealImageType >
40                                                  CastFilterType;
41  typedef itk::CannyEdgeDetectionImageFilter< RealImageType, RealImageType >
42                                                  FilterType;
43  typedef itk::RescaleIntensityImageFilter< RealImageType, OutputImageType >
44                                                  RescaleFilterType;
```

Exercise 2: Answer

ITKOpenCVBridge/exercise2/BasicVideoFilteringITKOpenCVBridgeAnswer.cxx

- Your new ITK pipeline will look like this:

```
46  CastFilterType::Pointer caster = CastFilterType::New();
47  FilterType::Pointer canny = FilterType::New();
48  RescaleFilterType::Pointer rescaler = RescaleFilterType::New();
49
50  InputImageType::Pointer itkFrame =
51    itk::OpenCVImageBridge::CVMatToITKImage< InputImageType >( inputImage );
52  caster->SetInput( itkFrame );
53  canny->SetInput( caster->GetOutput() );
54  rescaler->SetInput( canny->GetOutput() );
55
56  canny->SetVariance( 6 );
57  canny->SetLowerThreshold( 1 );
58  canny->SetUpperThreshold( 8 );
59
60  try
61    {
62    rescaler->Update();
63    }
64  catch( itk::ExceptionObject & excp )
65    {
66    std::cerr << excp << std::endl;
67    }
```

Exercise 2: Answer

ITKOpenCVBridge/exercise2/BasicVideoFilteringITKOpenCVBridgeAnswer.cxx

- With the following code for output purposes:

```
69    cv::Mat frameOut =
70      BridgeType::ITKImageToCVMat< OutputImageType >(rescaler->GetOutput(),true);
71
72    frameOut.convertTo( frameOut, CV_8U );
73
74    return frameOut;
```

Running the Answer

ITKOpenCVBridge/exercise2/BasicVideoFilteringITKOpenCVBridgeAnswer.cxx

- Run the answer with the following command

```
./BasicVideoFilteringITKOpenCVBridgeAnswer    \
~/data/Walk1.mpg                              \
./Walk1Edges.mpg
```

ITK Video Filters

Introduction

- Support video processing natively in ITK
- Standard framework for multi-frame filters
- Use ITK's library of image filters in video context

Video Filtering - Median Filter

ITKVideoPipeline/exercise1/ITKVideoSingleFrameFilters.cxx

- Video data structure

```
21  #include <itkVideoStream.h>
```

- Video file reader and writer
- Use OpenCV for video IO

```
24  #include <itkVideoFileReader.h>
25  #include <itkVideoFileWriter.h>
26  #include <itkOpenCVVideoIOFactory.h>
```

- Median image filter
- Image filter $\longrightarrow$ video filter wrapper

```
22  #include <itkMedianImageFilter.h>
23  #include <itkImageFilterToVideoFilterWrapper.h>
```

Video Filtering - Median Filter

ITKVideoPipeline/exercise1/ITKVideoSingleFrameFilters.cxx

- Types for data structures

```
36  const unsigned int Dimension =                          2;
37  typedef unsigned char                                   PixelType;
38  typedef itk::Image< PixelType, Dimension >              FrameType;
39  typedef itk::VideoStream< FrameType >                   VideoType;
```

- Types reader and writer

```
41  typedef itk::VideoFileReader< VideoType >               ReaderType;
42  typedef itk::VideoFileWriter< VideoType >               WriterType;
```

- Types for video median filter
- Use image filter type to define video filter type

```
43  typedef itk::MedianImageFilter< FrameType, FrameType > ImageFilterType;
44  typedef itk::ImageFilterToVideoFilterWrapper< ImageFilterType >
45                                                          VideoFilterType;
```

Video Filtering - Median Filter

ITKVideoPipeline/exercise1/ITKVideoSingleFrameFilters.cxx

- Create reader and writer

```
47  ReaderType::Pointer reader = ReaderType::New();
48  WriterType::Pointer writer = WriterType::New();
```

- Create image filter and video median filters

```
49  ImageFilterType::Pointer imageFilter = ImageFilterType::New();
50  VideoFilterType::Pointer videoFilter = VideoFilterType::New();
```

Video Filtering - Median Filter

ITKVideoPipeline/exercise1/ITKVideoSingleFrameFilters.cxx

- Set up reader and writer
- Tell ITK that we're using OpenCV for IO

```
52  itk::ObjectFactoryBase::RegisterFactory( itk::OpenCVVideoIOFactory::New() );
53  reader->SetFileName( argv[1] );
54  writer->SetFileName( argv[2] );
```

- Set up radius for (image) median filter
- Set video filter wrapper to use image median filter internally

```
56  FrameType::SizeType neighborhoodRadius;
57  neighborhoodRadius[0] = 10;
58  neighborhoodRadius[1] = 10;
59  imageFilter->SetRadius( neighborhoodRadius );
60  videoFilter->SetImageFilter( imageFilter );
```

Video Filtering - Median Filter

ITKVideoPipeline/exercise1/ITKVideoSingleFrameFilters.cxx

- Connect the pipeline
- reader → videoFilter → writer

```
62  videoFilter->SetInput( reader->GetOutput() );
63  writer->SetInput( videoFilter->GetOutput() );
```

- Try calling Update() to process the entire video

```
65  try
66    {
67    writer->Update();
68    }
69  catch( itk::ExceptionObject & excp )
70    {
71    std::cerr << excp << std::endl;
72    return EXIT_FAILURE;
73    }
```

Excercise 1

Replace median filter with curvature flow filter

- Hint 1: Curvature Flow requires `float` as the pixel type for output.
- Hint 2: OpenCV uses ffmpeg which requires `unsigned char` as the pixel type for writing.
- Hint 3: Curvature Flow does not take a radius parameter. It's salient functions are:

```
70  imageFilter->SetTimeStep( 0.5 );
71  imageFilter->SetNumberOfIterations( 20 );
```

- Hint 4: To use an image filter in a video pipeline use `ImageFilterToVideoFilterWrapper`

Excercise 1: Answer

Replace median filter with curvature flow filter

- Include curvature and cast image filters

```
22 #include <itkCurvatureFlowImageFilter.h>
23 #include <itkCastImageFilter.h>
```

- Re-define types using seperate IO and Real pixel types

```
37   const unsigned int Dimension =                    2;
38   typedef unsigned char                             IOPixelType;
39   typedef float                                     RealPixelType;
40   typedef itk::Image< IOPixelType, Dimension >      IOFrameType;
41   typedef itk::Image< RealPixelType, Dimension >    RealFrameType;
42   typedef itk::VideoStream< IOFrameType >           IOVideoType;
43   typedef itk::VideoStream< RealFrameType >         RealVideoType;
```

Excercise 1: Answer

Replace median filter with curvature flow filter

- Define cast filter to convert from Real to IO pixel type

```
47  typedef itk::CastImageFilter< RealFrameType, IOFrameType >
48                                                  CastImageFilterType;
49  typedef itk::ImageFilterToVideoFilterWrapper< CastImageFilterType >
50                                                  CastVideoFilterType;
```

- Replace median filter with curvature flow filter
- Use Real pixel type for output

```
51  typedef itk::CurvatureFlowImageFilter< IOFrameType, RealFrameType >
52                                                  ImageFilterType;
53  typedef itk::ImageFilterToVideoFilterWrapper< ImageFilterType >
54                                                  VideoFilterType;
```

Excercise 1: Answer

Replace median filter with curvature flow filter

- Set up video cast filter

```
68  videoCaster->SetImageFilter( imageCaster );
```

- Set up curvature flow filter

```
70  imageFilter->SetTimeStep( 0.5 );
71  imageFilter->SetNumberOfIterations( 20 );
72  videoFilter->SetImageFilter( imageFilter );
```

- Connect the pipeline
- reader $\longrightarrow$ curvature flow $\longrightarrow$ output caster $\longrightarrow$ writer

```
74  videoFilter->SetInput( reader->GetOutput() );
75  videoCaster->SetInput( videoFilter->GetOutput() );
76  writer->SetInput( videoCaster->GetOutput() );
```

Running the Example

/home/tutorial/bin/ITK-OpenCV-Bridge-Tutorial/Exercises/ITKVideoPipeline

- Run the example with the following command

```
./exercise1/ITKVideoSingleFrameFilters \
~/data/Walk1_short.avi                 \
./Walk1_short_median.avi
```

Excercise 2

Compute frame differences after diffusion

- Start with solution to Excersize 1 (ITKVideoMultiFrameFilters.cxx)
- Hint 1: Use FrameDifferenceVideoFilter
- Hint 2: Difference filter's important parameter is:

```
73    frameDifferenceFilter->SetFrameOffset(1);
```

- This sets the spacing of the frames to be differenced. Setting it to 1 means adjacent frames will be used.

Excercise 2: Answer

Compute frame differences after diffusion

- Include frame difference filter

```
25  #include <itkFrameDifferenceVideoFilter.h>
```

- Define type for frame difference filter
- Since this is a native video filter, no wrapper is necessary

```
57  typedef itk::FrameDifferenceVideoFilter< IOVideoType, IOVideoType >
58                                                  FrameDifferenceFilterType;
```

- Create frame difference filter

```
66  FrameDifferenceFilterType::Pointer frameDifferenceFilter =
67    FrameDifferenceFilterType::New();
```

Excercise 2: Answer

Compute frame differences after diffusion

- Set the frame offset to 1

```
73    frameDifferenceFilter->SetFrameOffset(1);
```

- Connect the pipeline
- reader → curvature flow → output caster → frame differ → writer

```
81    videoFilter->SetInput( reader->GetOutput() );
82    videoCaster->SetInput( videoFilter->GetOutput() );
83    frameDifferenceFilter->SetInput( videoCaster->GetOutput() );
84    writer->SetInput( frameDifferenceFilter->GetOutput() );
```

Running the Answer

/home/tutorial/bin/ITK-OpenCV-Bridge-Tutorial/Exercises/ITKVideoPipeline

- Run the example with the following command

```
./exercise2/ITKVideoMultiFrameFiltersAnswer \
~/data/Walk1_short.avi                      \
./Walk1_short_diff.avi
```

- Extra Credit: Add a threshold to the difference output to supress background noise
- ITKVideoPipeline/exercise1/ITKVideoMultiFrameFiltersAnswer2.cxx