LECTURE 1: Introduction

Dr. Ulas Bagci

HEC 221, Center for Research in Computer Vision (CRCV), University of Central Florida (UCF), Orlando, FL 32814.

bagci@ucf.edu or bagci@crcv.ucf.edu
• This is a special topics course, offered for the second time in UCF.
This is a special topics course, offered for the second time in UCF.

**Lectures:**
Mon/Wed, 10.30am-11.45am
• This is a special topics course, offered for the second time in UCF.

• **Lectures:**
  Mon/Wed, 10.30am-11.45am

• **Office hours:**
  Mon/Wed, 1pm-2.30pm

---

**CAP5937: Medical Image Computing**
• This is a special topics course, offered for the second time in UCF.
• **Lectures:** Mon/Wed, 10.30am-11.45am
• **Office hours:** Mon/Wed, 1pm-2.30pm
• No textbook is required, materials will be provided.
• Avg. grade was A- last spring.
Medical Image Computing

- Image Processing
- Computer Vision
- Imaging Sciences (Radiology, Biomedical)
- Machine Learning
Motivation

- Imaging sciences is experiencing a tremendous growth in the U.S. The NYT recently ranked biomedical jobs as the number one fastest growing career field in the nation and listed bio-medical imaging as the primary reason for the growth.
Motivation

• Imaging sciences is experiencing a tremendous growth in the U.S. The NYT recently ranked biomedical jobs as the number one fastest growing career field in the nation and listed bio-medical imaging as the primary reason for the growth.

• **Biomedical imaging** and **its analysis** are fundamental to (1) understanding, (2) visualizing, and (3) quantifying information.
Motivation

• Imaging sciences is experiencing a tremendous growth in the U.S. The NYT recently ranked biomedical jobs as the number one fastest growing career field in the nation and listed bio-medical imaging as the primary reason for the growth.

• Biomedical imaging and its analysis are fundamental to (1) understanding, (2) visualizing, and (3) quantifying information.

• This course will mostly focus on analysis of biomedical images, and imaging part will be briefly taught!
Syllabus

• Basics of Radiological Image Modalities and their clinical use (MRI, PET, CT, fMRI, DTI, …)
• Introduction to Medical Image Computing and Toolkits
• Image Filtering, Enhancement, Noise Reduction, and Signal Processing
• Medical Image Registration
• Medical Image Segmentation
• Medical Image Visualization
• Machine Learning in Medical Imaging
• Shape Modeling/Analysis of Medical Images
Syllabus

• Grading:
  – In-class Quiz (20%, approximately 20 quizzes, at the end of each lecture)
  – 3 Programming Assignments (each 10%, total 30%)
    • ITK/VTK packages should be used
    • ITK and VTK provide necessary codes/libraries for medical image processing and analysis.
    • C/C++ or Python can be used and call ITK/VTK functions
    • In-class collaboration is encouraged, but individual submission is required.
  – 1 Individual Project (50%)
    • Will be selected from a list of projects or you can come with your own project
    • A short presentation (15%), coding/method (25%), results (10%)
Optional Reading List

• Level-set Methods, by J. A. Sethian, Cambridge University Press.
• Insight into Images: Principles and Practice for Segmentation, Registration and Image Analysis, Terry S. Yoo (Editor) (FREE)
• Algorithms for Image Processing and Computer Vision, J. R. Parker
• Medical Imaging Signals and Systems, by Jerry Prince & Jonathan Links, Publisher: Prentice Hall
Conferences and Journals to Follow

- **The top-tier conferences** (double blind, acceptance rates are below 25%, high quality technical articles):
  - MICCAI (medical image computing & computer assisted intervention)
  - IPMI (Information Processing in Medical Imaging)
  - Other conferences: IEEE ISBI, EMBC and SPIE Med Imaging
  - Clinical Conferences: RSNA (>65,000 attendances), ISMRM, SNM

- **The top-tier technical journals:**
  - IEEE TMI, TBME, PAMI, and TIP
  - Medical Image Analysis, CMIG, and NeuroImage

- **The top-tier clinical journals relevant to MIC:**
  - Radiology, Journal of Nuclear Medicine, AJR, Nature Methods, Nature Medicine, PlosOne, …
Required skill set

- **Basic** programming experience (any language is fine)
- Linear Algebra/Matrix Algebra
- Differential Equations
- Basic Statistics
Biomedical Images

• (Bio)medical images are different from other pictures
Biomedical Images

• (Bio)medical images are different from other pictures
  – They depict distributions of various physical features measured from the human body (or animal body).
Biomedical Images

• (Bio)medical images are different from other pictures
  – They depict distributions of various physical features measured from the human body (or animal body).

• Analysis of biomedical images is guided by very specific expectations
Biomedical Images

• (Bio)medical images are different from other pictures
  – They depict distributions of various physical features measured from the human body (or animal).

• Analysis of biomedical images is guided by very specific expectations
  – Automatic detection of tumors, characterizing their types,
  – Measurement of normal/abnormal structures,
  – Visualization of anatomy, surgery guidance, therapy planning,
  – Exploring relationship between clinical, genomic, and imaging based markers
Free Software to use in this course

- ImageJ (and/or FIJI)
- ITK-Snap
- SimpleITK
- MITK
- FreeSurfer
- Slicer
- OsiriX
- An extensive list of software: www.idoimaging.com and

blue: will be frequently used in this course
Medical Image Formats

- Dicom
- Nifti
- Analyze (img/hdr)
- Raw data
- ...
DICOM (the mostly used)

- Digital Imaging and Communications in Medicine standard
- Since its first publication in 1993, DICOM has revolutionized the practice of radiology, allowing the replacement of X-ray film with a fully digital workflow.
- It is the international standard for medical images and related information (ISO 12052)
- defines the formats for medical images that can be exchanged with the data and quality necessary for clinical use.
- It is implemented in almost every radiology, cardiology imaging, and radiotherapy device (X-ray, CT, MRI, ultrasound, etc.), and increasingly in devices in other medical domains such as ophthalmology and dentistry.
3D Slicer Software

• A software platform for the analysis (including registration and interactive segmentation) and visualization (including volume rendering) of medical images and for research in image guided therapy.

• A free, open source software available on multiple operating systems: Linux, MacOSX and Windows

• Extensible, with powerful plug-in capabilities for adding algorithms and applications.
Brief History of 3D Slicer

- 1997: Slicer started as a research project between the Surgical Planning Lab (Harvard) and the CSAIL (MIT)
- 80 authorized developers contributing to the source code of Slicer
- Open Source + Open Data + Open Community
Slicer Volume Module

The Welcome module is the default start-up module
Each module of Slicer includes a series of tabs, which gives access to different functionalities.

Click on the arrow symbol to display the content of each tab.
Slicer Welcome Module

- **Toolbar**
- **User Interface (UI) panel of the Slicer Welcome Module**
- **3D Viewer**
- **Data Probe**
- **2D anatomical viewers**
Loading A DICOM Volume

Drag and drop the "dataset1_Thorax_Abdomen" file into slicer
Loading A DICOM Volume

A pop-up window appears: Select Load directory into DICOM database and click on OK
Loading A DICOM Volume

Click on OK once the directory import is completed
Click on **Load Selection to Slicer** to load the DICOM volume into Slicer (note: this may take a few minutes)
Interactive exploration

Select the module **Volume Rendering** in the modules menu.
Slicer displays the 3D rendered volume of the CT_Thorax_Abdomen dataset
Interactive exploration

Use the mouse in the 3D window to rotate the volume rendered image
3D Slicer Sources

- http://slicer.org/
- https://vimeo.com/37671358
Libraries to be Used

• **ITK and VTK**
  – National Library of Medicine **Insight Segmentation and Registration Toolkit (ITK).**
  – ITK is an open-source, cross-platform system that provides developers with an extensive suite of software tools for image analysis.
  – C/C++, Python, Matlab, ...

**ITK**

ITK provides leading-edge segmentation and registration algorithms in two, three, and more dimensions; it is distributed as an open-source software package.
Goals of ITK

– Supporting the Visible Human Project.
– Establishing a foundation for future research.
– Creating a repository of fundamental algorithms.
– Developing a platform for advanced product development.
– Support commercial application of the technology.
– Create conventions for future work.
– Grow a self-sustaining community of software users and developers.
History of ITK

• ITK was initially conceived by the NLM (National Library of Medicine).
• An initiative for open source software tools to analyze human dataset.
• Developed by group of both commercial and academic organizations (kitware, GE research, Mathsoft, Upenn, UT, UNC).
• Goal: provide a foundation to enable research in image processing and biomedical image computing, Providing catalog of algorithms.
What is ITK?

• Image Processing
• Segmentation
• Registration
• No Graphical User Interface (GUI)
• No Visualization
Coordinate System for Reading Files

• Multiple coordinate frames
  – Physical
  – Patient
  – Index

• ITK uses LPS (Left Posterior Superior) for DICOM
How to Integrate ITK in application

C++ Glue Code

ITK
Image Processing

GUI
{MFC, Qt, wxWin, FLTK}

Visualization
{OpenGL, VTK}

Credit: itk.org
Installation/Requirements

C++ Compiler
- gcc 2.95 – 3.3
- Visual C++ 6.0
- Visual C++ 7.0
- Visual C++ 7.1
  - Intel 7.1
  - Intel 8.0
  - IRIX CC
- Borland 5.5
- Mac – gcc

CMake
www.cmake.org

Credit: itk.org
Installation process

• Google ITK, go to the download page, download the zip file (or directly install using github or console functions in mac/linux)

• Google cmake, go to the download page, get the binaries and install the binaries.
Configuring ITK – MS-Windows

- Run CMake
- Select the SOURCE directory
- Select the BINARY directory
- Select your Compiler
Configuring ITK

Right click on a cache value for additional options (delete, ignore, and help).
Press Configure to update and display new values in red.
Press OK to generate selected build files and exit.

For backwards compatibility, what version of CMake commands and syntax should this version of CMake allow.
Configuring ITK

- Disable BUILD_EXAMPLES
- Disable BUILD_SHARED_LIBS
- Disable BUILD_TESTING

- Click “Configure” to configure
- Click “OK” to generate project files
Building ITK

- Open ITK.sln in the Binary Directory
- Select ALL_BUILD project
- Build it

...It will take about 15 minutes...
Verify the Built

Libraries will be found in

\texttt{ITK\_BINARY / bin / \{ Debug, Release \}}

- The following libraries should be there:
  - ITKCommon
  - ITKBasicFilters
  - ITKAlgorithms
  - ITKNumerics
  - ITKFEM
  - ITKIO
  - ITKStatistics
  - ITKMetaIO
  - itkpng
  - itkzlib
Use ITK from an external Project

“HelloWorld.cxx”
“CMakeLists.txt”
from the Examples/Installation Directory into another directory
Run CMake
• Select Source Dir
• Select Binary Dir

“HelloWorld”
“CMake 2.0 - patch 3”

Cache Values
- CMAKE_BACKWARDS_COMPATIBILITY: 2.0
- EXECUTABLE_OUTPUT_PATH
- ITK_DIR: E:/cygwin/home/ibanez/bin/Insight
- LIBRARY_OUTPUT_PATH

Right click on a cache value for additional options (delete, ignore, and help).
Press Configure to update and display new values in red.
Press OK to generate selected build files and exit.

For backwards compatibility, what version of CMake commands and syntax should this version of CMake allow.
Use ITK from an external Project

- accept the default in `CMAKE_BACKWARD_COMPATIBILITY`
- leave empty `EXECUTABLE_OUTPUT_PATH`
- leave empty `LIBRARY_OUTPUT_PATH`
- Set `ITK_DIR` to the binary directory where ITK was built
Build Sample Project

- Open `HelloWorld.sln` generated by CMake.
- Select `ALL_BUILD` project.
- Build it.
Run the example

• Locate the file HelloWorld.exe

• Run it…

• It should produce the message:

  ITK Hello World!
Starting your own project

- Create a clean new directory
- Write a \texttt{CMakeLists.txt} file
- Write a simple \texttt{.cxx} file
- Configure with \texttt{CMake}
- Build
- Run
Writing CMakeLists.txt

PROJECT( myProject )

FIND_PACKAGE ( ITK )
IF ( ITK_FOUND )
    INCLUDE( ${ITK_USE_FILE} )
ENDIF( ITK_FOUND )

ADD_EXECUTABLE( myProject myProject.cxx )

TARGET_LINK_LIBRARIES ( myProject ITKCommon ITKIO )
#include "itkImage.h"
#include "itkImageFileReader.h"
#include "itkGradientMagnitudeImageFilter.h"

int main( int argc, char **argv ) {
    typedef itk::Image<unsigned short,2> ImageType;
    typedef itk::ImageFileReader<ImageType> ReaderType;
    typedef itk::GradientMagnitudeImageFilter<ImageType,ImageType> FilterType;

    ReaderType::Pointer reader = ReaderType::New();
    FilterType::Pointer filter = FilterType::New();

    reader->SetFileName( argv[1] );
    filter->SetInput( reader->GetOutput() );
    filter->Update();
    return 0;
}
Run CMake
How to find what you need?

http://www.itk.org/ItkSoftwareGuide.pdf


• Follow the link Alphabetical List
• Follow the link Groups
• Post to the insight-users mailing list
Introduction

Welcome to the National Library of Medicine Insight Segmentation and Registration Toolkit (ITK). ITK is an open-source software system to support the Visible Human Project. Currently under active development, ITK...
## ITK Compound Index

<table>
<thead>
<tr>
<th>A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessorFunctor (itk::Functor)</td>
<td>Histogram::Iterator (itk::Statistics)</td>
</tr>
<tr>
<td>Acos (itk::Functor)</td>
<td>HistogramImageToImageMetric (itk)</td>
</tr>
<tr>
<td>AcosImageAdaptor (itk)</td>
<td>HistogramMatchingImageFilter (itk)</td>
</tr>
<tr>
<td>AcosImageFilter (itk)</td>
<td>HMaximalImageFilter (itk)</td>
</tr>
<tr>
<td>AcosPixelAccessor (itk::Accessor)</td>
<td>HMinimalImageFilter (itk)</td>
</tr>
<tr>
<td>AdaptImageFilter (itk)</td>
<td>HoughTransform2DCirclesImageFilter (itk)</td>
</tr>
<tr>
<td>AdaptiveHistogramEqualizationImageFilter (itk)</td>
<td>HoughTransform2DLinesImageFilter (itk)</td>
</tr>
<tr>
<td>Add1 (itk::Functor)</td>
<td>HoughTransformCircles2DImageFilter (itk)</td>
</tr>
<tr>
<td>Add2 (itk::Functor)</td>
<td>HypersphereKernelMeanShiftModeSeeker (itk::Statistics)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Add3 (itk::Function)</td>
<td>IdentifierArrayEqualsFunction (itk)</td>
</tr>
<tr>
<td>AddImageAdaptor (itk)</td>
<td>IdentifierArrayHashFunction (itk)</td>
</tr>
<tr>
<td>AddImageFilter (itk)</td>
<td>IdentityTransform (itk)</td>
</tr>
<tr>
<td>AdditiveOperators (itk::Concept)</td>
<td>Image (itk)</td>
</tr>
<tr>
<td>AdditiveOperators::Constraints (itk::Concept)</td>
<td>ImageAdaptor (itk)</td>
</tr>
<tr>
<td>AddPixelAccessor (itk::Accessor)</td>
<td>ImageAndPathToImageFilter (itk)</td>
</tr>
<tr>
<td>AffineTransform (itk)</td>
<td>ImageBase (itk)</td>
</tr>
<tr>
<td>AmoebaOptimizer (itk)</td>
<td>ImageBoundaryCondition (itk)</td>
</tr>
</tbody>
</table>
VTK

• The Visualization Toolkit (VTK) is an open-source, freely available software system for 3D computer graphics, image processing, and visualization.

• Consists of a C++ class library and several interpreted interface layers including Tcl/Tk, Java, and Python
**VTK**

- Download VTK from [vtk.org](http://vtk.org)
- Configure VTK
  - Run CMake
  - Select the SOURCE directory
  - Select the BINARY directory
  - Select your Compiler (same used for ITK)

---

<table>
<thead>
<tr>
<th>Cache Values</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILD_EXAMPLES</td>
<td>OFF</td>
</tr>
<tr>
<td>BUILD_SHARED_LIBS</td>
<td>OFF</td>
</tr>
<tr>
<td>CMAKE_BACKWARDS_COMPATIBILITY</td>
<td>1.6</td>
</tr>
<tr>
<td>VTK_DATA_ROOT</td>
<td>E:/cygwin/home/ibanez/data/VTKData</td>
</tr>
<tr>
<td>VTK_USE_HYBRID</td>
<td>ON</td>
</tr>
<tr>
<td>VTK_USE_PARALLEL</td>
<td>ON</td>
</tr>
<tr>
<td>VTK_USE_PATENTED</td>
<td>ON</td>
</tr>
<tr>
<td>VTK_USE_RENDERING</td>
<td>ON</td>
</tr>
<tr>
<td>VTK_WRAP_JAVA</td>
<td>OFF</td>
</tr>
<tr>
<td>VTK_WRAP_PYTHON</td>
<td>OFF</td>
</tr>
<tr>
<td>VTK_WRAP_TCL</td>
<td>OFF</td>
</tr>
</tbody>
</table>

Right click on a cache value for additional options (delete, ignore, and help).
Press Configure to update and display new values in red.
Press OK to generate selected build files and exit.

Build VTK examples.
Configuring VTK

Disable
• BUILD_EXAMPLES
• BUILD_SHARED

Leave unchanged
• CMAKE_BACKWARD_COMPATIBILITY
• VTK_DATA_ROOT

Enable
• VTK_USE_HYBRID
• VTK_USE_RENDERING
• VTK_USE_PARALLEL
• VTK_USE_PATENTED
Disable

- VTK.WRAP_JAVA
- VTK.WRAP_PYTHON
- VTK.WRAP_TCL

Enable (Advanced)

- VTK_USE_ANSI_STDLIB
Build VTK

• Open VTK.dswin the Binary Directory
• Select ALL_BUILD project
• Build it

…it may take about 90 minutes …

Verify the Build

Libraries will be found in

VTK_BINARY / bin/ { Debug, Release}
Verify the Build

The following libraries should be there:

• vtkCommon
• vtkFiltering
• vtkImaging
• vtkGraphics
• vtkHybrid
• vtkParallel
• vtkPatented

• Vtkexpat
• Vtkfreetype
• Vtkftgl
• Vtkjpeg
• Vtkpng
• Vtktiff
• vtkzlib
Starting your own project with ITK + VTK

• Create a clean new directory
• Write a CmakeLists.txt file
• Write a simple .cxx file
• Configure with CMake
• Build
• Run
PROJECT(myProject)

FIND_PACKAGE (ITK)
IF (ITK_FOUND)
INCLUDE(${USE_ITK_FILE})
ENDIF(ITK_FOUND)

FIND_PACKAGE (VTK)
IF (VTK_FOUND)
INCLUDE(${USE_VTK_FILE})
ENDIF(VTK_FOUND)

(continue...)
Writing CMakeLists.txt

INCLUDE_DIRECTORIES(
  
  
  )

ADD_EXECUTABLE(

)

TARGET_LINK_LIBRARIES(

)
Writing myProject.cxx
Writing myProject.cxx

#include "itkImage.h"
#include "itkImageFileReader.h"
#include "itkImageToVTKImageFilter.h"
#include "vtkImageViewer.h"
#include "vtkRenderWindowInteractor.h"
int main(int argc, char **argv) {
  typedef itk::Image<unsigned short, 2> ImageType;
  typedef itk::ImageFileReader<ImageType> ReaderType;
  typedef itk::ImageToVTKImageFilter<ImageType> connectorType;
  ReaderType::Pointer reader = ReaderType::New();
  ConnectorType::Pointer connector = ConnectorType::New();
```
reader->SetFileName( argv[1]);
connector->SetInput( reader->GetOutput() );
vtkImageViewer* viewer= vtkImageViewer::New();
vtkRenderWindowInteractor* renderWindowInteractor=
vtkRenderWindowInteractor::New();
viewer->SetupInteractor( renderWindowInteractor);
viewer->SetInput( connector->GetOutput() );
viewer->Render();
viewer->SetColorWindow( 255 );
viewer->SetColorLevel( 128 );
renderWindowInteractor->Start();
return 0;
```
Index to Physical Coordinates

Spacing (Sx)

Pixel Index

Spacing (Sy)

Origin (Ox,Oy)

P[0] = Index[0] x Spacing[0] + Origin[0]

Index[0] = floor( ( P[0] - Origin[0] ) / Spacing[0] + 0.5 )
Resample Image Filter Example

```cpp
#include "itkImage.h"
#include "itkResampleImageFilter.h"
#include "itkIdentityTransform.h"
#include "itkLinearInterpolateImageFunction.h"

typedef itk::Image< char, 2 >   ImageType;

ImageType::Pointer inputImage = GetImageSomeHow();

typedef itk::ResampleImageFilter< ImageType >   FilterType;

FilterType::Pointer resampler = FilterType::New();

ImageType::SizeType size;
size[0] = 200;
size[1] = 300;

ImageType::IndexType start;
start[0] = 0;
start[1] = 0;
```

ITK/Examples/Filtering/ResampleImageFilter.cxx
Resample Image Filter

ImageType::PointType origin;
origin[0] = 10.0;    // millimeters
origin[1] = 25.5;   // millimeters

ImageType::SpacingType spacing;
spacing[0] = 2.0;    // millimeters
spacing[1] = 1.5;    // millimeters

resampler->SetOutputSpacing( spacing );
resampler->SetOutputOrigin( origin );

resampler->SetSize( size );
resampler->SetOutputStartIndex( start );

resampler->SetDefaultPixelValue( 100 );
resampler->SetInput( inputImage );
typedef itk::LinearInterpolateImageFunction<
    ImageType,
    double > InterpolatorType;

InterpolatorType::Pointer interpolator = InterpolatorType::New();

typedef itk::TranslationTransform< double, 2 > TransformType;

TransformType::Pointer transform = TransformType::New();

transform->SetIdentity();

resampler->SetInterpolator( interpolator );
resampler->SetTransform( transform );

resampler->Update();

const ImageType * outputImage = resampler->GetOutput();
SimpleITK

- New Wrapper for the insight segmentation & registration toolkit
- Goal: to help rapid prototyping and expand the user based of ITK by exposing the algorithms to new users

- Simplify the algorithms so they don’t depend on types of images.
- Binary built in distributions
- Supports 2D & 3D image, multi component images
- Easy importing & exporting
- data through Numpy
• QUESTIONS?
Quiz!

(1 pt)