

Methods in Image Analysis - Assignment #3

February 16, 2006

Due Thursday, 3/2/06

This assignment is the first one to integrate a visualization component, so you will be given more time in anticipation of potential complications integrating all of the components. Please don't take this extra time as more of a procrastination buffer...it will be much better to tackle potential issues with toolkit integration early. It is assumed that you have downloaded and installed ITK, VTK, and FLTK.

The basecode - named `myITKgui` - for this assignment can be downloaded from the website. As with other ITK code, you can build this with CMake. Note that you will need to specify ITK, VTK, and FLTK directories in CMake. Note that you should specify the *binary* directories, not the *source* directories.

`myITKgui` is designed as three main classes, `migApp`, `migAppBase`, and `migGUI`. The `migAppBase` class contains the ITK pipeline and is the primary functional unit (in regards to instantiation). The GUI class is used to visualize the input (pre-pipeline) and output (post-pipeline) images. The pipeline itself is set up to read Meta format files and visualize 3D 8-bit unsigned char images. There are several synthetic MRI images in meta format available for download off the website; note that the file extension is `.mha`. The `migApp` class is a dummy class with no real functionality. It lies at the bottom of the class hierarchy and is mostly there to invoke the gui.

There are two parts to this assignment. First, build `myITKgui` as distributed to verify that it works on your computer. Second, replace the default filter (binary threshold) with an inversion filter that you will write. Feel free to develop the filter independent from the GUI system if you like (via a command-line interface that outputs to an image file, as in your last assignment), and just connect it to the GUI once it's finished. Let's talk about the filter.

Your job is to write a filter - let's call it `itkInvertPixelValueImageFilter` - that will invert the intensity values in a grayscale image. For instance, if the data type is `unsigned char`:

`inputIntensity = 0 => outputIntensity = 255`

This results in input values of 0 mapping to 255 in the output and vice versa. A subtlety of this

assignment is that you do not know beforehand what the input and output pixel types will be. In other words, I might want to have the input image be short int and the output image be unsigned char. Because of the differences in max/min between the input and output types, you will need to use numeric traits to identify the characteristics of the requested types and scale the values appropriately.

You will probably want to start with an existing filter and perform find-and-replace operations to change the name of the class. A good starting point is `itkReflectImageFilter.txx` and `.h` found in `ITK/Code/BasicFilters`. Be sure to change the names of the files and bring the copies into your own source directory as well! Obviously, the implementation of your filter will be different than that of the reflect filter, so you will need to remove the code specific to the reflection. All of the code within the filter `.h` and `.txx` will be inside the ITK namespace so you don't need to use the `itk::` prefix when referring to other ITK classes.

Presuming you wrote the filter correctly, it should work in `myITKgui` without modification. You should modify `migAppBase::CreateITKPipeline` and `migAppBase::UpdatePipelineCallback` in order to replace the sample filter included with the `myITKgui` code. If you're feeling particularly adventurous, you can use Fluid (the FLtk UI Design tool) to design additional controls for your pipeline.

To submit the assignment, please ZIP up any source code you've written, along with your `CMakeLists.txt` file, and e-mail the ZIP file to `kjr21@pitt.edu`.