ITK on the iOS

Release 1.00

Boris Shabash[1], Ghassan Hamarneh[1], Zhi Feng Huang[1], and Luis Ibanez[2]

August 31, 2010

[1]School of Computing Science, Simon Fraser University, BC, Canada
[2]The Insight Group

Abstract

ITK is one of the most powerful image segmentation and registration libraries available as an open source toolkit. Motivated by the recent popularization of the iPhone, iPod touch, and iPad, in this work, we describe the set of required steps for integrating the ITK framework into Apple's iOS mobile operating system. Our focus in this paper is on the process of importing the C++ based ITK to the Objective-C based iOS, and creating a simple application that demonstrates the ITK libraries are integrated. This paper brings to the reader a user-manual on how to integrate the ITK libraries into iOS applications and code. We present here the series of steps we performed in order to import the ITK libraries as well as report the results of importing them under different versions of the iOS and under different architectures (the Simulator and Device architecture).

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3209]

Contents

1 Introduction

Medical image analysis is an important field within health research. The processing and analysis of medical images to extract certain information from the images has been a motivation for the continuous development

of new algorithms and more powerful software and hardware. The leading software library designed for the analysis of medical images is the Insight Toolkit (ITK), first developed in 1999 and released in 2002 [1]. ITK is an open source library that is written in C++ and allows its users to write software for processing and analysis of images with emphasis on medical imaging applications. The library itself is written with the principle of "Generic Programming"[2] in mind, where users of the library should be able to integrate the code into any project, while adhering to a minimal set of requirements in order for the ITK code to be incorporated successfully.

With the release of the iPod touch and iPhone to the market in 2007 [3], Apple Inc. has also released the iPhone Software Development Kit (SDK), allowing programmers worldwide to create software for the iPhone and iPod touch and send it to Apple for approval, or even load it onto their own devices for private use. In 2010, Apple released the iPad with a larger multi-touch display and faster processor. For the iPhone, iPod touch, and iPad, apple provides a single iOS SDK. The focus of this work is on integrating ITK into Apple's iOS mobile operating system allowing programmers to integrate ITK's powerful image processing and analysis capabilities into Apple's iPod touch, iPhone and iPad products.

There are a handful of previous examples of the integration of ITK into other environments. Some of the most prominent examples of software environments integrating ITK include MATITK [4], SimITK [5], and WrapITK [6]. MATITK allows MATLAB (The MathWorks Inc., Natick, MA) users to access ITK functionality without having to use C++ code or do any sort of compilation. MATITK is basically a mediator between MATLAB, which is a scripting language gaining more and more popularity, and ITK, with its powerful filtering, segmentation and registration algorithms. SimITK also combines MATLAB with ITK, but adds on the principle of visual programming to the work environment. SimITK allows programmers to further abstract the code by constructing visual image processing workflow pipelines, as is common in Simulink [7]. WrapITK allows for the "wrapping" of ITK classes in Python, Tcl or Java. WrapITK basically contains code that mediates between these languages (much like MATITK does for MATLAB). Other examples software applications integrating ITK functionality include Slicer [8], MeVisLab [9], Seg3D [10], and VolView [11]. These software support code re-use by bridging the gaps that may exist between languages and computing environments rather than requiring the programmer to 're-invent the wheel'.

ITK has been designed from the outset to be a cross-platform software development toolkit. Software developed using the ITK software libraries on one OS platform (e.g. Microsoft Windows, Apple Mac OS, or Linux) can be easily complied on other platforms. However, to the best of our knowledge, there is very little previous work related directly to cross-compiling ITK on the iOS. The ITK support forum contains several posts regarding attempts of cross-compilation but without evidence of success [12, 13]. Previous attempts were done using Cross Platform Make (CMake) [14], in an attempt to create a CMake script that would create an iPhone application that is linked to ITK. Although this approach is loyal to the conventions of the ITK framework, it may be too great of a challenge to immediately attempt the use of CMake for this purpose when most iPhone, iPod touch, and iPad application development is done using Xcode [15], Apple's suite of tools for developing software on Mac OS.

In this paper, we describe in detail a procedure for building ITK on the iOS in Section 2.1, culminating in the successful execution of the rudimentary ITK "Hello World" program in Section 2.2. We conclude and summarize possible future work in Section 3.

2 Method

We first describe the steps adopted to compile ITK on the iOS in Section 2.1. We then describe building an application that executes the ITK HelloWorld program (Section 2.2.1 of The ITK Software Guide 2.4.0 [1])

in Section 2.2.

2.1 Compiling ITK on iOS

1. **Download ITK**: We downloaded ITK source code version 3.20.0 from http://www.itk.org/ITK/resources/software.html. We use ITK_SOURCE_DIR to refer to the path to the directory where all the ITK source files are uncompressed into. In order to successfully compile the ITK libraries for the iOS, we made three modifications:

   - in CMakeLists.txt, line 529, comment out the line SET(ITK_REQUIRED_C_FLAGS "$ITK_REQUIRED_C_FLAGS -Wno-long-double") by adding the # character at the beginning of the line. Do the same to line 530, SET(ITK_REQUIRED_CXX_FLAGS "$ITK_REQUIRED_CXX_FLAGS -Wno-long-double")
   - in CMakeLists.txt, line 540, comment out the line SET(ITK_REQUIRED_CXX_FLAGS "$ITK_REQUIRED_CXX_FLAGS -msse2") by adding the # character at the beginning of the line
   - in ITK_SOURCE_DIR/Utilities/vxl/core/vnl/vnl_math.cxx, after line 77, add the line #define finite(x) __inline_isfinited((double)x)

2. **Download CMake**: We downloaded CMake 2.8.2 from http://www.cmake.org/cmake/resources/software.html. ITK is usually compiled on the system by the use of CMake to produce a script file called Makefile, which contains all the commands to compile the code. However, in this case we used CMake to produce an Xcode project for the MacOSX/iOS work environment.

3. **Download iOS SDK**: We downloaded iOS SDK 3.2 from http://developer.apple.com/iphone/index.action

4. **Download Xcode**: We downloaded Xcode 3.2.3 from http://developer.apple.com/technologies/xcode.html.

5. **Create Xcode project**: We created a new directory where the binary executables and projects will be created. We use ITK_BINARY_DIR to refer to the path to this binary directory. From that directory, we issued the command:

   ```
   ccmake -GXcode ITK_SOURCE_DIR
   ```

   The CMake configurations used are shown Figure 1. Note that there is no configuration at the first time of running ccmake. We hit the key c to generate the initial configuration and then modify it to look like the one in Figure 1. We then hit the key c again. After the configuration was done, we hit the key g to generate the Xcode project. This creates the required Xcode project as the file with extension .xcodeproj in folder ITK_BINARY_DIR.

6. **Compile Xcode project under Mac OS**: We built the Xcode project using Xcode 3.2.3 with its original settings. We used the *Release* architecture in Mac OS X 10.6.4. We built and ran the project using the the command "Build and Run" under the "Build" menu in Xcode (for short, the noteation: Xcode → Build → Build and Run, will be used later). This step is important to create the source file g3states.h for the target itktiff by using the command line tool itkmkg3states (this is done automatically).

   Reaching this stage without any errors is an indication of a successful build of the ITK libraries for the Mac OS.

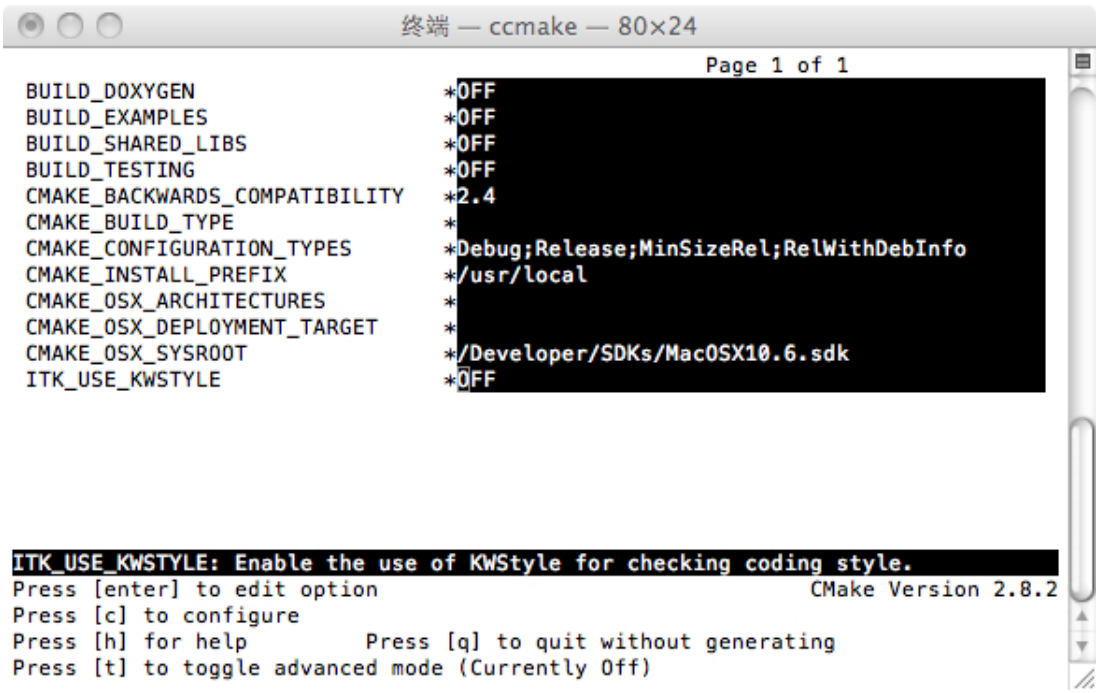

Figure 1: The CMake configuration used to create the Xcode project.

7. **Compile Xcode project under iOS**: We then built the Xcode project under the iPhone architecture by editing the build setting and target dependencies. To edit the build setting of the project, use Xcode → Project → Edit Project Settings → Build Tab (Figure 2):

   - Change the `Base SDK` to `iPhone Simulator 4.0`. By changing this, ITK is complied under iOS SDK 4.0.
   - Enable `armv6` and `armv7` architectures to the 'Valid Architectures' label.
   - Change the 'Architectures' label `Standard`.

   To modify the dependency of the target `ALL_BUILD`, follow Xcode → Project → Edit Active Target "`ALL_BUILD`" → General Tab:

   - Remove the dependency on `itkmkg3states` and `itkTestDriver` under 'Direct Dependencies'.

   After removing the dependencies, Xcode may pop up an internal error message. To resolve the issue, click on the 'Continues' button on the error message, exit Xcode and open the project again. To modify the dependency of the target `itktiff`, navigate to and expand the 'Targets' tree on the left side of the Xcode window and find the target `itktiff`:

   - Right click on the target `itktiff` → Get Info → General Tab.
   - Remove the dependency on `itkmkg3states` under 'Direct Dependencies'.

   The reason for these modifications is that `itkmkg3states` and `itkTestDriver` are both command line utilities that are not supported on the iOS. Finally, build the Xcode project under the new settings using Xcode → Build → Build , creating iOS compatible libraries.

   At this stage, success can again be asserted if no errors arose in the compilation process. The compiled libraries are placed in `ITK_BINARY_DIR/bin/Release`. We rename the folder to be `4.0simRelease`

for indicating that the libraries inside the folder are compiled for the iPhone Simulator 4.0. The reason for this renaming is that the `Release` folder is overwritten for a new 'Build' even if the new 'Build' is for iPhone Simulator 3.2. This way, the folder will not be overwritten.

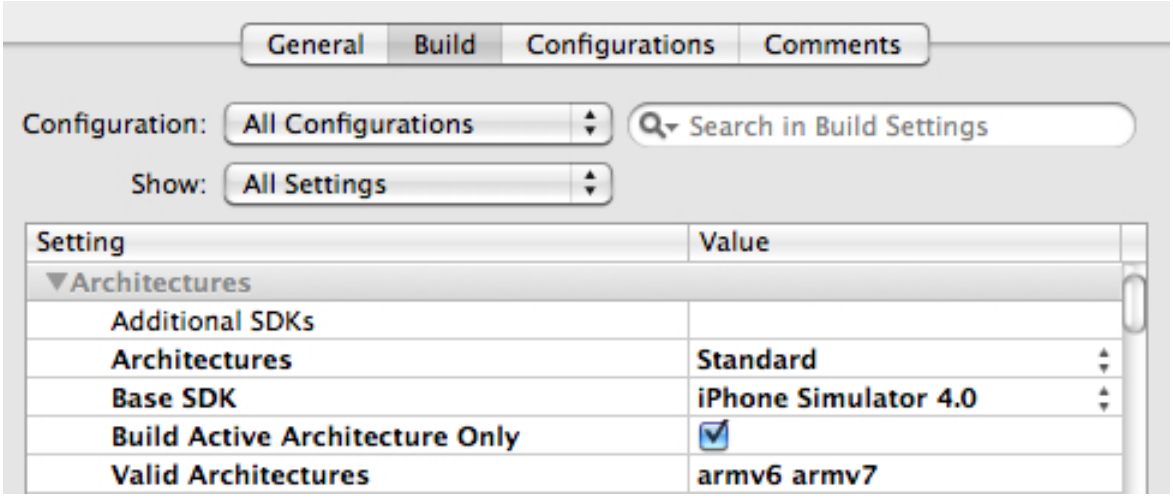

Figure 2: Project settings for Xcode 3.2.3 to compile the ITK under the iOS.

2.2 HelloWorld ITK application on iPhone

In this section, we detail the steps needed to build ITK's HelloWorld application on the iPhone. Similar steps apply for the iPod touch and iPad.

1. **Create iPhone application project**: We created a simple view-based iPhone application as follows: Xcode → File → New Project → iPhone OS → Application → View-based Application. Then, changed the extension of all `.m` files into `.mm` files allowing C++ code to be interpreted in the project. The following code was then added to the file `main.mm`:

   ```
   #include "ItkImage.h"
   ```

   and the following code was added to the `main` method:

   ```
   typedef itk::Image< unsigned short, 3 > ImageType;
   ImageType::Pointer image = ImageType::New();
   std::cout << "ITK Hello World !" << std::endl;
   ```

2. **Add search paths**: In order for `itkImage.h` and all of its required header files to be found, we added 6 search paths to the Xcode project by following Xcode → Project → Edit Project Setting → Build (tab) → Header Search Paths, and adding the following paths to the Value field (Figure 3):

   - `ITK_SOURCE_DIR/Code/Common`
   - `ITK_SOURCE_DIR/Utilities/vxl/vcl`
   - `ITK_SOURCE_DIR/Utilities/vxl/core`
   - `ITK_BINARY_DIR/Utilities/vxl/vcl`

- `ITK_BINARY_DIR/Utilities/vxl/core`
- `ITK_BINARY_DIR`

3. **Add ITK library into the iPhone application project**: In order to use the ITK library, we added the ITK library as follows (Figure 4):

   - Xcode → Project → New Group (alternatively, navigate to the 'Groups & Files' panel on the left side of Xcode window and right click on the name of the iPhone application project and choose Add → New Group).
   - Name the new group as 'ITK'.
   - With this new ITK group highlighted, choose Xcode → Project → Add to Project (alternatively, right click on the new group ITK and choose Add → Existing Files...).
   - Navigate to `ITK_BINARY_DIR/bin/4.0simRelease` (In general, the folder is called `Release` instead of `4.0simRelease`. We rename the folder at the end of Step 2.1.7).
   - Select all the libraries in the directory above.

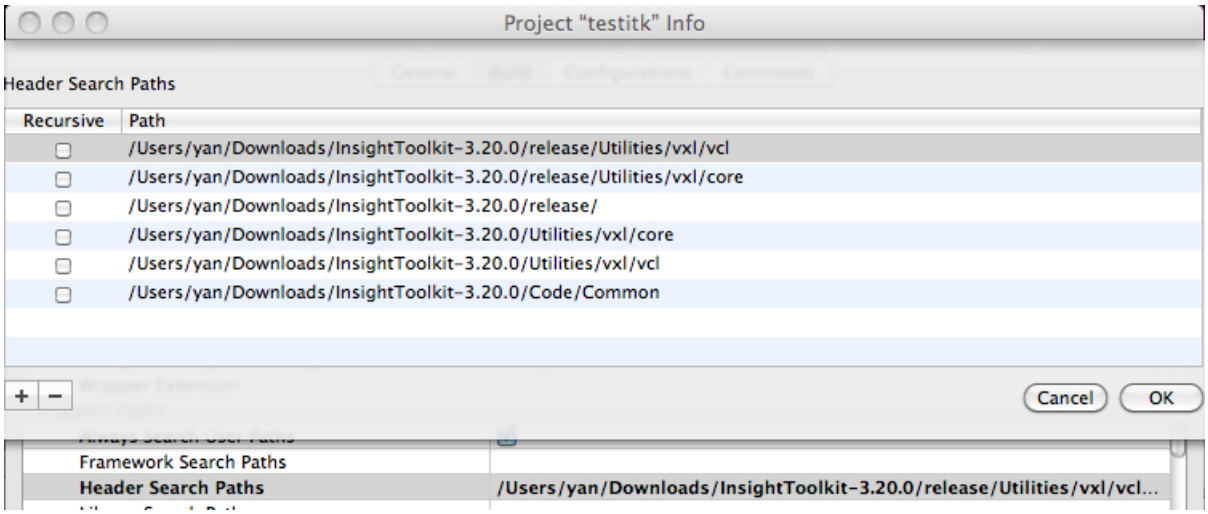

Figure 3: The addition of the 6 header search paths to the Xcode application project.

4. **Build and Run**: The project was then compiled and run using Xcode → Build → Build and Run.

The output is an app that does nothing in the simulator; it only displays an empty view (Figure 6(left)). However, on the debugger console, the text "ITK Hello World!" is output (Figure 6(right)):

In order to compile and run the app on a real device, we compile the ITK libraries by following the same steps in Section 2.1 but setting the `Base SDK` to `iPhone Device 4.0`. In Section Section 2.2, these ITK libraries are added into the iPhone application project but not the ones for the iPhone simulator. After the iPhone application project is set properly, we choose the top left controller in the main editor window, and select 'Device' (Figure 5). Finally, the project is then compiled and run using Xcode → Build → Build and Run.

For completeness, the method in Section 2 have been tested on the following combinations:

Notice that iOS SDK 3.2 is the one for developing iPad applications.

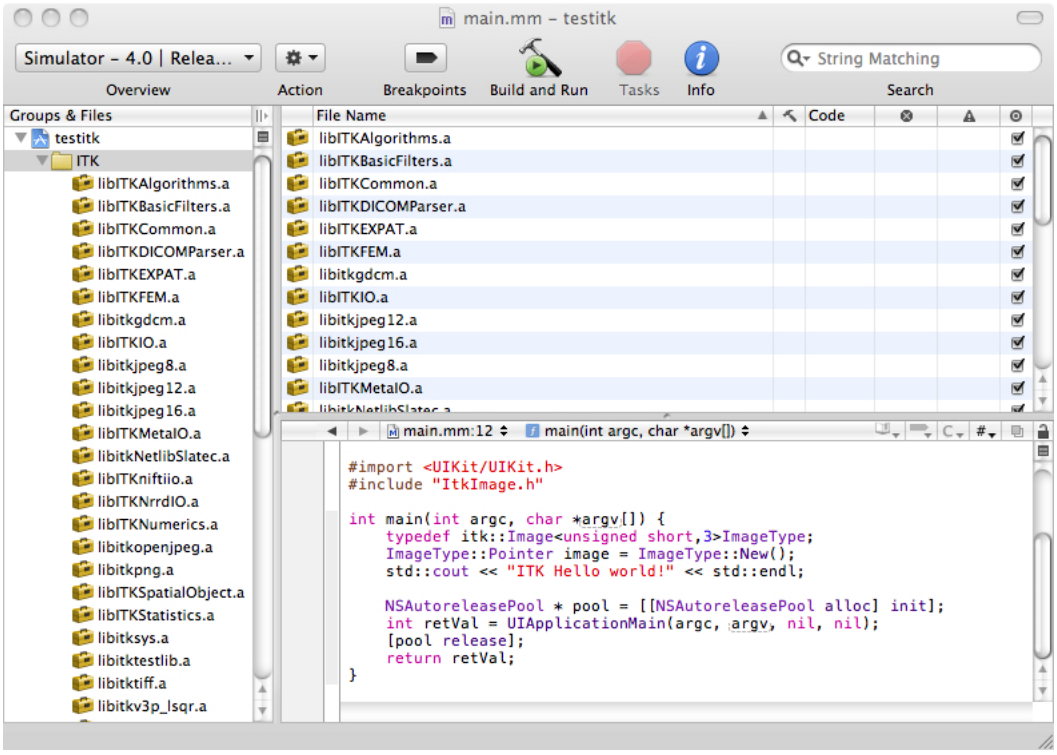

Figure 4: The ITK library is added into the iPhone application library.

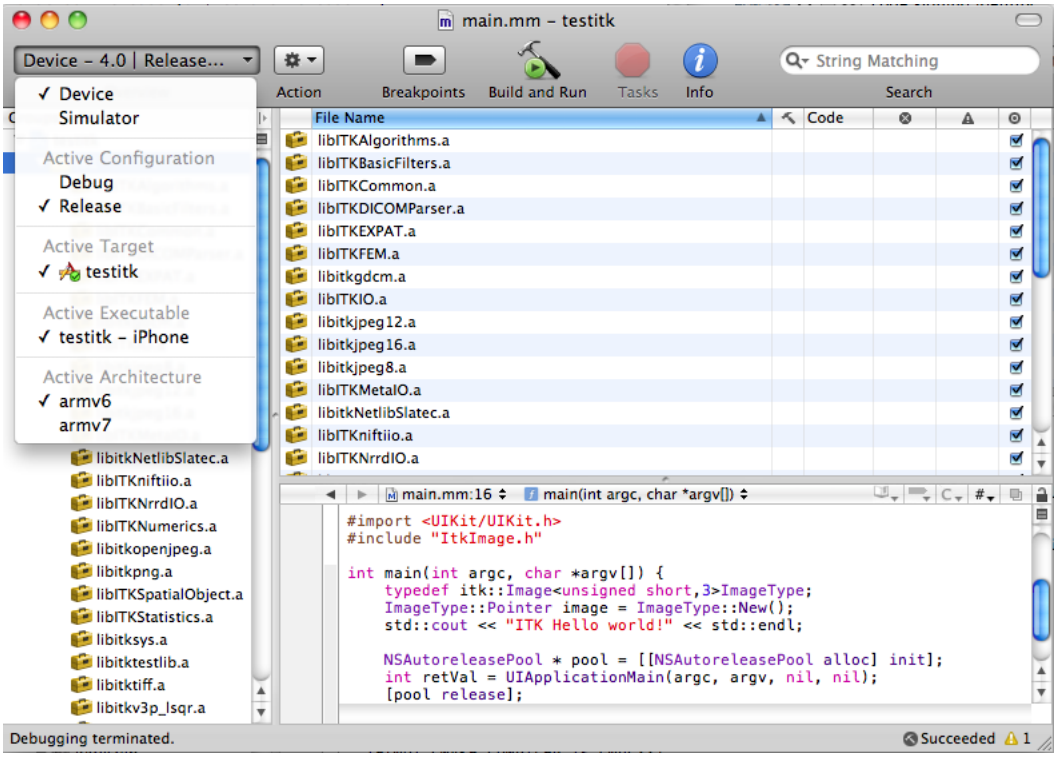

Figure 5: Select this controller to choose ‘Device’ as the ‘Active SDK’

Figure 6: The output of the ITK example HelloWorld program. (left) The iPhone simulator output (an empty view). (right) The debugger console output.

Platform\SDK	3.0	3.1	3.1.2	3.1.3	3.2	4.0
iPhone Simulator	✓	✓	✓	✓	✓	✓
iPhone Device			✓	✓		✓

Table 1: The combinations of iPhone platform and iPhone SDK that have been tested

3 Conclusions and Future Work

This project symbolizes the first milestone in the integration of ITK to the iOS platform; the mobile computing OS used on the popular iPod touch, iPhone, and iPad. We detailed the steps needed for successfully building ITK on the iOS, leading to the execution of the basic ITK HelloWorld example.

The long term goal of this line of work is to have fully operating ITK based medical image analysis application running on popular, state-of-the-art portable devices, which we believe will become invaluable for health application in the coming years.

With the completion of this project, the required architecture settings for building ITK on the iOS are now known and tested. The developed view-based application may now serve as a template for iPhone, iPod touch, or iPad applications executing ITK code. Furthermore, the reported developments would allows ITK programmers to create CMake or Makefile scripts to automatically generate Xcode projects compatible with the iOS. For example, automatic CMake scripts would render all the tedious directory and path setting hidden from the programmers, allowing them to focus on the functionality of the final application. We are particularly excited about the potential of creating advanced medical image analysis applications that make use of ITK's powerful capabilities and the iPad's large multi-touch display and intuitive user-interaction gestures.

Another area of future work is the integration of other C++ based medical imaging software into the iOS, such as the Visualization Toolkit (VTK); ITK's graphics based cousin. This will allow not only processing

but also visualization of medical images using the powerful algorithms that have been developed in the last decade in C++.

References

[1] L. Ibanez, W. Schroeder, L. Ng, and J. Cates, *The ITK Software Guide*, 2nd ed., Kitware, Inc. ISBN 1-930934-15-7, 2005. [Online]. Available: http://www.itk.org/ItkSoftwareGuide.pdf 1, 2

[2] G. D. Reis and J. Jarvi. What is genetic programming? [Online]. Available: http://lcsd05.cs.tamu.edu/papers/dos_reis_et_al.pdf 1

[3] L. Grossman, "The Best Inventions of 2007," *TIME*, November 2007. [Online]. Available: http://www.time.com/time/specials/2007/article/0,28804,1677329_1678542,00.html 1

[4] H. G. Chu. V, *MATITK: EXTENDING MATLAB WITH ITK*, MATITK: EXTENDING MATLAB WITH ITK,
http://www.cs.sfu.ca/ hamarneh/ecopy/ij2005.pdf, 2005. [Online]. Available: http://www.cs.sfu.ca/~hamarneh/ecopy/ij2005.pdf 1

[5] P. Abolmaesumi, P. Mousavi, D. Gobbi, and A. Dickinson. simITK/simVTK. [Online]. Available: http://media.cs.queensu.ca/media/SimITKVTK/ 1

[6] G. Lehmann, Z. Pincus, and B. Regrain. (2008) WrapITK: Enhanced languages support for the Insight Toolkit. [Online]. Available: http://www.insight-journal.org/browse/publication/85 1

[7] The Mathworks, Inc. (2010) Simulink - simulation and model-based design. [Online]. Available: http://www.mathworks.com/products/simulink/ 1

[8] S. Pieper and R. Kikinis. (2010) 3D Slicer. [Online]. Available: http://www.slicer.org/ 1

[9] MeVisLab. [Online]. Available: http://www.mevislab.de/developer/documentation/itk-addon/ 1

[10] "Seg3d," 2007, seg3D: Volumetric Image Segmentation and Visualization. Scientific Computing and Imaging Institute (SCI). [Online]. Available: https://www.seg3d.org 1

[11] Kitware Inc;. (2009) Volview. [Online]. Available: http://www.kitware.com/products/volview.html 1

[12] S. McBride. (2008) [Insight-users] iPhone. [Online]. Available: http://www.itk.org/pipermail/insight-users/2008-June/026364.html 1

[13] K. Stump. (2010) [Insight-users] trying to cross compile ITK for iPhone. 1

[14] Kitware Inc;. Cmake. [Online]. Available: http://www.cmake.org/ 1

[15] Apple Inc. Xcode. [Online]. Available: http://developer.apple.com/technologies/tools/xcode.html 1