iPad Mobile Surgical Console

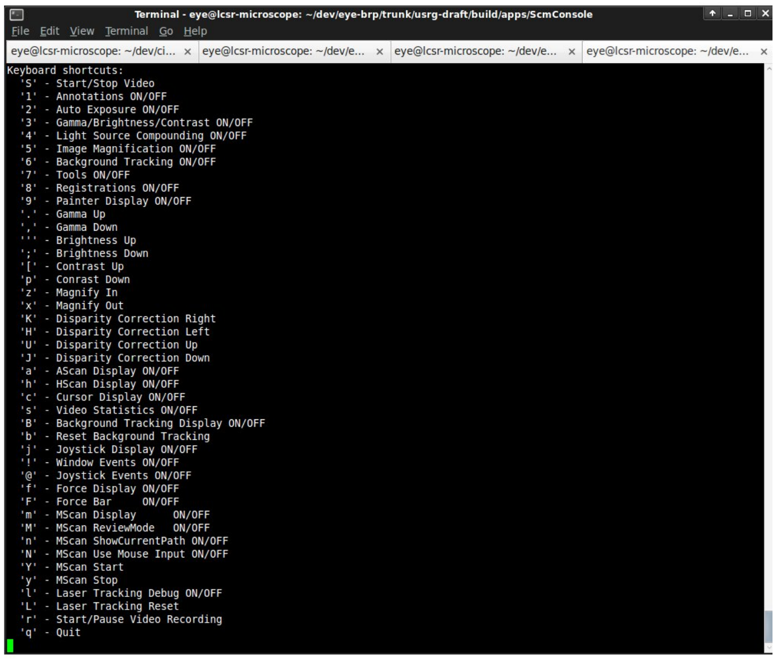
Hanlin Wan and Jonathan Satria

Mentors: Balazs Vagvolgyi and Russell Taylor

May 18, 2011

1 Problem and Significance

It has become increasingly more important to develop safe, efficient, and effective human-computer interface models as surgical robotics advance and begin to integrate more and more capabilities. Modern operating rooms that utilize a robot surgical assistant require several computer workstations, each controlling different devices or aspects of the robot. Such is the case with the EyeRobot, which has had the dilemma of requiring multiple computers and a difficult navigation scheme to control various components of the robot, such as brightness, magnification, and scan overlays. See figure 1 for an example of the keyboard shortcuts that are currently being used to control such settings.


Terminal - eye@lcsr-microscope: ~/dev/eye-brp/trunk/usrg-draft/build/apps/ScmConsole
File Edit View Terminal Go Help
eye@lcsr-microscope: ~/dev/ci...
eye@lcsr-microscope: ~/dev/e...
eye@lcsr-microscope: ~/dev/e...
eye@lcsr-microscope: ~/dev/e...

```
Keyboard shortcuts:
  'S' - Start/Stop Video
  '1' - Annotations ON/OFF
  '2' - Auto Exposure ON/OFF
  '3' - Gamma/Brightness/Contrast ON/OFF
  '4' - Light Source Compounding ON/OFF
  '5' - Image Magnification ON/OFF
  '6' - Background Tracking ON/OFF
  '7' - Tools ON/OFF
  '8' - Registrations ON/OFF
  '9' - Painter Display ON/OFF
  '.' - Gamma Up
  ',' - Gamma Down
  ''' - Brightness Up
  ';' - Brightness Down
  '[' - Contrast Up
  'p' - Conrast Down
  'z' - Magnify In
  'x' - Magnify Out
  'K' - Disparity Correction Right
  'H' - Disparity Correction Left
  'U' - Disparity Correction Up
  'J' - Disparity Correction Down
  'a' - AScan Display ON/OFF
  'h' - HScan Display ON/OFF
  'c' - Cursor Display ON/OFF
  's' - Video Statistics ON/OFF
  'B' - Background Tracking Display ON/OFF
  'b' - Reset Background Tracking
  'j' - Joystick Display ON/OFF
  '!' - Window Events ON/OFF
  '@' - Joystick Events ON/OFF
  'f' - Force Display ON/OFF
  'F' - Force Bar     ON/OFF
  'm' - MScan Display      ON/OFF
  'M' - MScan ReviewMode   ON/OFF
  'n' - MScan ShowCurrentPath ON/OFF
  'N' - MScan Use Mouse Input ON/OFF
  'Y' - MScan Start
  'y' - MScan Stop
  'l' - Laser Tracking Debug ON/OFF
  'L' - Laser Tracking Reset
  'r' - Start/Pause Video Recording
  'q' - Quit
```

Figure 1: A screenshot of the EyeRobot console where settings are controlled with inefficient keyboard shortcuts.

This decentralized approach places both ergonomic and efficiency constraints to the surgeon and the medical team. Practically speaking, several problems arise with the current implementation: lack of centralized control is inefficient. A user must approach the correct computer workstation to adjust settings. Sterilization concerns from using multiple keyboard and mouse pairs. Cluttering of limited operating room space.

With the recent commoditization of inexpensive tablet computers that feature touchscreen control, the opportunity to integrate all of the controls into one easy-to-use application is attractive. More specifically, the Apple iPad has many of the features that would solve the issues outlined above. Those are: a slick and clean touchscreen that does not require any peripherals; the ability to incorporate easy to use navigation schemas; and an inexpensive development suite in Objective C/C++ that allows practically anyone to develop applications.

For these reasons, the iPad was chosen as a "mobile surgical console" development device to control the configurable settings on the EyeRobot. In this project, we set out to create a framework iOS application that proves that an application can successfully communicate with other components (using cisst and ICE, described below). In addition, we aimed to create a tutorial that extends the lifetime of this project, allowing other users to create iOS applications that include the cisst libraries.

2 Technical Approaches

The iPad, along with other Apple mobile devices, runs the iOS operating system. This presents some unique challenges but also provides an avenue for developing an application for the EyeRobot. This is namely because all code is in Objective C/C++ and C/C++ can also be incorporated into the project. Including static libraries is also possible, and so for these reasons it became evident that incorporating the existing cisst libraries would not be very difficult.

In order to have the iPad communicate with other components, it would have to communicate with the scenario manager via the Internet Communications Engine (ICE). Therefore, the iPad application is required to include the cisst and ICE libraries.

To fully develop the application, several technologies were required to successfully compile and work together on the iPad architecture (armv7). This was done through the use of CMake, and those dependencies are listed below:

- **Cisst Library:** This library is the backbone of the project. All communications to and from the various components uses this library.
- **CMake:** CMake allows the compilation of the source code on any platform.
- **iOS:** Development of the iPad application relies on the iOS framework. Apple's iOS Development Kit will be used for the design and implementation of the GUI.
- **ICE:** The Internet Communications Environment manages the connections between multiple computers. This simplifies the programming required to get computers to communicate with each other.
- **Scenario Manager:** This tool controls the interactions between various components. On startup, it finds all connected components and forms links between them.

In our approach, we first focused on compiling the cisst library and including it as a static library in our project. We then spent quite some time integrating ICE into the application. The final scheme allows the iPad to act merely as another component, generating multitask calls that are relayed through ICE to the scenario manager. See figure 2 for the schematics of our application.

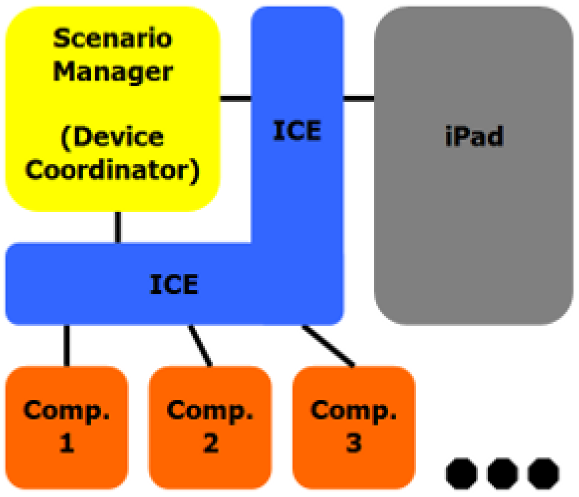


Figure 2: The schematics of our application. The iPad is connected through ICE to the Scenerio Manager, which is then connected through ICE to the various components that we wish to control.

3 Results and Discussion

3.1 Cisst Compilation

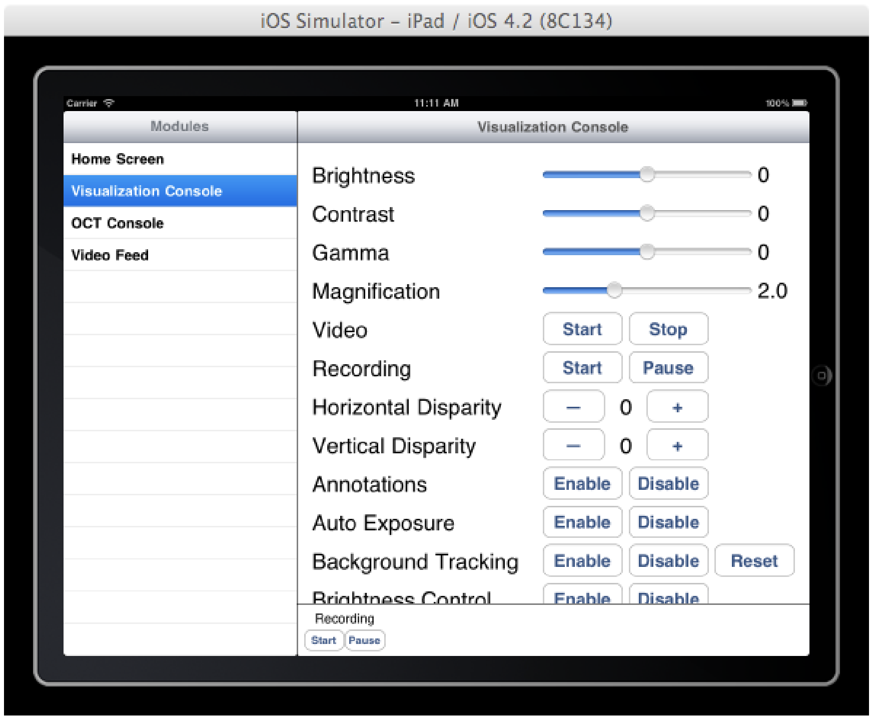
Our biggest hurdle at first seemed to be compiling cisst for the armv7 architecture, something that had never been done before using CMake. Several CMake settings had to be tweaked in order for the code to compile into a static library, and the reader is directed to our online documentation tutorial at `https://trac.lcsr.jhu.edu/cisst/wiki/Private/cisst_iOS`. There, the necessary CMake files are detailed.

3.2 ICE Compilation

After successfully compiling and testing cisst for non-multitask duties, it became necessary to compile ICE so that cisstMultiTask could also be compiled and the iPad could be made to communicate with other components. Compiling ICE became an unforeseen challenge, as it required much manipulation of its build settings as well as the use of a third-party tool, which also itself needed patching.

3.3 iPad Application Development

The iPad application was developed concurrently with each of the compiling milestone just mentioned. It was relatively easy to incorporate the static libraries into the code and with minor setbacks the application easily served as a platform to test the success of compiling the different technologies. Those setbacks were related to unfamiliar use Xcode, the Integrated Developing Environment for iOS projects. Coincidentally, Xcode was upgraded to Xcode 4 in mid-development of our project. It integrated interface builder, the program used to easily create the GUI view files of the application, which simplified some of the major aspects of development.


iOS Simulator – iPad / iOS 4.2 (8C134)
Carrier
11:11 AM
100%
Modules
Home Screen
Visualization Console
OCT Console
Video Feed
Visualization Console
Video Start Stop
Recording Start Pause
Annotations Enable Disable
Auto Exposure Enable Disable
Background Tracking Enable Disable Reset
Brightness Control Enable Disable
Recording
Start Pause

Brightness 0
Contrast 0
Gamma 0
Magnification 2.0
Horizontal Disparity 0
Vertical Disparity 0

Figure 3: A screenshot of our application. On the left are menus for various modules whose settings can be changed. On the right are those settings, set through a combination of buttons and sliders.

In order to develop an application with an easy-to-use navigation scheme (see figure 3), a feature emphasized earlier, we implemented a split-view navigation window. This allows the different components of the EyeRobot to be displayed in a left pane and the relevant selected component's settings in the right hand pane. Again, the reader is directed to our tutorial page at the website mentioned earlier.

3.4 Clinical trials

After major functionality was completed, the application was tested in a bunny experiment with great success. Users reported that there was improved ease and efficiency. It allowed users to easily change settings from anywhere in the operating room and with a few simple "clicks" of the touchscreen.

4 Conclusion

Using the iPad we successfully developed a framework application to control a select number of the components on the EyeRobot. We have completed exactly what we originally proposed – creating a framework application and an easy-to-follow tutorial that explains how to develop future applications. Without any compromises, we have been able to compile the cisst libraries and ICE and have shown that the armv7 architecture and iOS are viable development platforms that can incorporate these libraries.

5 Management Summary

5.1 Who Did What?

Hanlin Wan:

- Completed much of the development for getting cisst, ICE, and iOS to compile and linker properly.
- Created an automated script for easy compilation and wrote a tutorial for manual compilation.
- Wrote primary GUI design.

Jonathan Satria:

- Worked on the GUI design and programming.
- Wrote the tutorials for developing different levels of iOS applications that incorporate cisst and ICE.

5.2 Management Accomplishments

We have completed everything outlined in our original project proposal and have been able to meet part of our maximum deliverables. We have completed the following expected deliverables:

- Compiling and linking cisst and ICE libraries on the armv7 architecture.
- Detailed tutorial for future applications.
- Testing of the device with EyeRobot.

- Perform revision to the GUI based on user feedback.

We were able to meet part of the maximum deliverable of incorporating live video display feed on the iPad. All of the code is in place to do so, it only requires the video packets to be injected to the application.

5.3 Future Plans

As both of us are graduating this year we will have a minimal role in the future of this application. We hope to see the iPad application to be used in other surgical robot fields, perhaps in the Da Vinci. We will maintain a supporting role for future developments and will ensure the integrity of the tutorials page, as it will be the go-to resource for users wishing to developed cisst/iOS applications.

5.4 What We Learned

Through the trial and error we faced in compiling cisst and ICE for the armv7 architecture, we learned a lot about CMake, which was the major tool necessary for this project. Without it, none of the compilation would have been possible. We also learned a lot about cross compiling software for various different architectures. This was a huge stumbling block for multiple reasons. We faced issues compiling the necessary components for the iOS architecture. We had issues linking libraries due to architecture inconsistencies. We also had little issues over naming conflicts between C++ and Objective C.

On the GUI development side, we learned how to make iPad applications. This can easily be extended to creating applications for other Apple devices such as the iPhone and the iPod Touch. We can now create applications that interact with the user using buttons, sliders, and text boxes. We have also mastered some more difficult but more interesting components of the iOS GUI design such as multi-finger swipe gestures, alert boxes, and multiple windows.

6 References

1. Apple, iOS Reference Library, <http://developer.apple.com/library/ios/navigation/>.

2. A. Deguet, R. Kumar, R. Taylor, and P. Kazanzides, "The cisst libraries for computer assisted intervention systems," in MICCAI Workshop on Systems and Arch. for Computer Assisted Interventions, Midas Journal, Sep 2008.

3. M. Henning, M. Spruiell, Distributed Programming with Ice, <http://www.zeroc.com/doc/Ice-3.4.1-IceTouch/manual/>.

4. M.Y. Jung, G. Sevinc, A. Deguet, R. Kumar, R. Taylor, P. Kazanzides, "Surgical Assistant Workstation (SAW) Communication Interfaces for Teleoperation."

5. P. Kazanzides, A. Deguet, A. Kapoor, O. Sadowsky, A. LaMora, R. Taylor, "Development of open source software for computer-assisted intervention systems," In MICCAI Workshop on Open-Source Software, Oct 2005.