CALIBRATION INTERFACE FOR THE ELECTRONIC SWIM COACH

A.R.M. Amos-Binks and J. R. Green
Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada

INTRODUCTION

The overhead required to learn to swim with a severe visual impairment is much lower than learning other sports. The most popular system currently in place to assist visually impaired swimmers requires a coach to stand at either end of a pool lane and tap the swimmer on the head when they reach the location where they should begin their turn. This proves problematic since two volunteers are needed and constant attention to the swimmer is required by both volunteers to prevent collisions with the pool deck. Furthermore, there is no effective means to prevent the swimmer from veering laterally into other swimming lanes.

We here report significant improvements to a prototype electronic swim coach [1] for visually impaired swimmers being developed in our lab. This system tracks the swimmer in a lane through a webcam connected to a computer using image processing techniques and then notifies the swimmer if they are veering or approaching the end of the lane through a wireless communication component connected to the computer.

The previous electronic swim coach had a rudimentary Graphical User Interface (GUI) that was designed for a developer to test the functionality of the image processing and wireless communication components [1]. Furthermore, although the functional components work well, they were very tightly coupled to the testing GUI, which made improving the interface and reusing the image processing components for future improvements very complicated and time consuming.

BACKGROUND

Previous development of the electronic swim coach system has focused on three major components: the software which performs image processing, the wireless communication system, and the swimmer notification system.

The software communicates via a wireless transmitter (TLP434A) attached to a micro-controller chip (Arduino ATmega328) which receives commands over a USB connection. After the swimmer location and state is determined, the appropriate wireless command is then encoded and sent to a wireless receiver (RLP434) and microcontroller (Arduino ATmega328) in the swimmer's swim cap. The wireless command is then decoded and the necessary vibrational motors in the swimmer's swim cap are activated.

A fourth required component is the coach’s calibration interface that will run on a tablet PC connected to the webcam at one end of the pool lane. This interface is required for several purposes: 1) before the swimmer begins swimming independently, the system must be calibrated to identify the pool and lane boundaries, 2) the system must be provided with a description of the swimmer’s cap colour, 3) the coach requires a means to verify the operation of the wireless channel by sending test messages, 4) the coach should observe the system’s tracking performance during a test lap to ensure that it is able to identify the swimmer at all points in the pool lane as lighting varies with position.

We here report on the development of a coach’s calibration graphical user interface. Figure 1 shows the overall system in detail with the addition of the newly designed GUI.

As a secondary contribution, the software has been completely restructured since the software model used in the previous prototype tightly coupled the user interface with the program’s functionality, which made future usability of software components complicated.

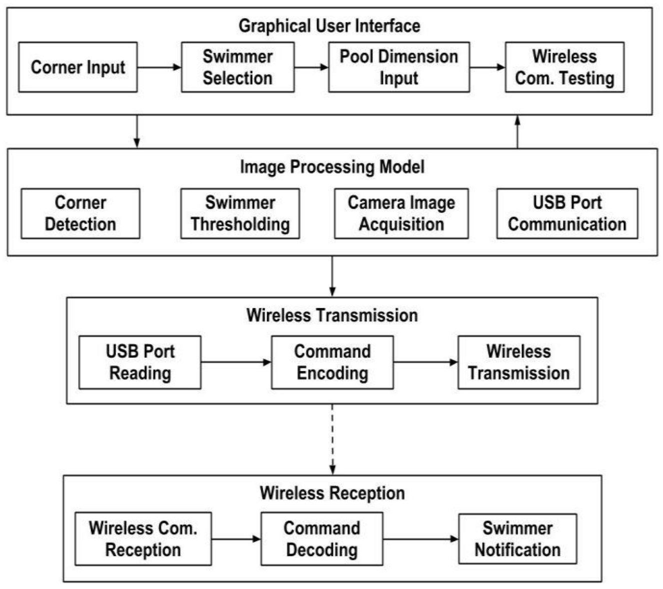


Figure 1 - System Diagram

As an object oriented language, Java lends itself well to GUI design. Object oriented GUIs are easier to develop and organize. The ability to create objects in Java helps in decoupling the function library from the user interface which will assist in code reuse in the further development of this project.

Object oriented applications can be designed in a number of complex patterns. In order to introduce some structure and facilitate future code reuse, a commonly used design pattern called Model-View-Controller (MVC) was used. The MVC design pattern separates the GUI (i.e. the view) from the model. The view portion is programmed to visualize the data, while the model does all the data processing. This creates a much simpler program which is easier to implement and reuse for future programmers [2].

The image processing component of the electronic swim coach requires an extensive library of standard computer vision functions. The OpenCV project [3] was used as it provides open source implementations, in C++, of all functions needed for this project. Since the electronic swim coach was ported to a Java environment, the OpenCV C++ functions need to be called through an open source wrapper library, JavaCV. The JavaCV library provides the optimized functionality of the OpenCV library through standard Java object method calls [4].

METHODS

Two main goals of this project were adding a GUI designed for the target audience and creating a more efficient software model.

GUI Design

In the previous version of the electronic swim coach, each step in the system calibration process produced a separate popup window. Considering that the system is intended to be used by a swimming coach in a busy environment, it was important to simplify the user interface as much as possible. Therefore, the project was converted to have one window containing the lane image with menus along the bottom and right-hand side.

The newly designed GUI has incorporated automatic corner detection using the method described in [5]. The previous version had the user select the corners manually by clicking on an image of the lane. The user could not correct his/her selection without restarting the application. In the new GUI, corners are initially estimated automatically. The user is presented with the four estimates which can be dragged into place to allow the coach to refine the results of the automatic corner detector and provide the desired lane boundaries. This feature provides a more natural feel and will be useful when moving to a touch screen interface.

The second step in the system calibration requires the swimmer’s cap to be manually selected in an initial image since the system tracks the swimmer’s location based on the colour of his/her swim cap. The image is then converted from RGB colour space to hue-saturation-value (HSV) colour space, which facilitates isolation of a specific colour independent of the amount of white light and reflection in the image. For the hue channel in the HSV image, the mean of the full data is not used, only the middle 50% of the data around the mean. This eliminates outlier pixels and results in a more accurate initial threshold. The user can also adjust the threshold using track bars that appear along the right side of the application window.

Since it is desirable to know lap splits and swim time, a real time lap split/swim time table has been added to the bottom of the GUI screen, shown in Figure 2. The time table automatically detects changes in direction from the swimmer then records the lap time and updates the overall swim time.

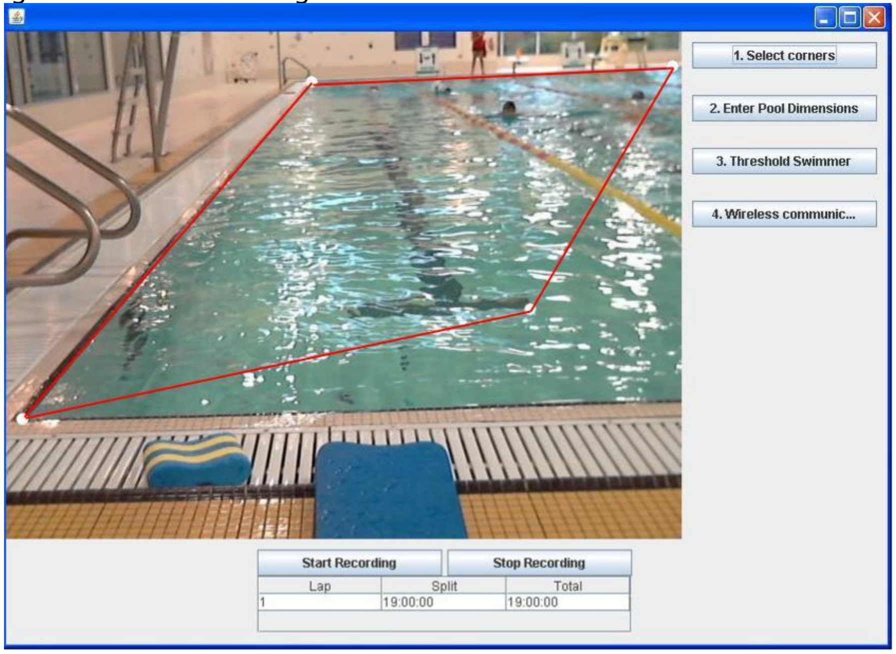
Software Model

The move from C++ to Java offered the opportunity to create a calibration interface built on a library of customized image processing functions that can be easily extended to other platforms. Extensibility and ease of development were the main motivations behind the move from Microsoft MFC C++ classes to a Java system that would ease the development of a future system on other platforms. To increase the organization of the software, the MVC design pattern was used. The project has been extended to have one centralized data processing model that contains all image data and performs all data processing. This centralized model can then be called by any view designed for the user. The two views used in this case were a calibration view, used to calibrate the system, and also the real-time swimmer tracking view. Since both these views are relatively simple, the controller functions were integrated into their design. This centralized library of functions can be reused to design new applications or extended to add functionality.

Wireless Communication

The original electronic swim coach had a well-designed wireless notification system that communicated over a 434MHz frequency band. To communicate with the swimmer, different commands are written to a USB port. From the USB port these commands are then encoded using Manchester encoding with a long preamble. Manchester encoding uses bit transitions: a logical one is a transition from a digital zero to a one and the reverse is true for a zero. The preamble is the stream of bits that unambiguously establishes the start of a command. To transmit a command, Java writes an ASCII character to the USB port. This command is then converted to the appropriate Manchester encoding and transmitted via the Arduino chip[1].

Unfortunately, writing to communication ports is not supported in standard versions of Java. To implement the wireless system as is, an open source java library from [6] was used. This library allows commands to be written to the com port through Java object method calls.



Lap	Split	Total
1	19:00:00	19:00:00

Figure 2 - Electronic Swim Coach GUI

RESULTS

The recording button shown in Figure 2 cannot be pressed until all the setup tasks have been completed. Once each lap is finished the split and swim time will be updated.

The pool dimensions are then entered in order to perform homography of the pool image needed to accurately track the swimmer. This eliminates the 'keystone' effect due to perspective. The swim cap is then selected and thresholded. At this point, the coach conducts a wireless communication test where a known pattern is sent to the swimmer. Once these setup tasks are complete the swimmer can then be tracked. During swimmer tracking, the same GUI appears however a different view is used.

In order to verify the increased ease of use of the GUI, eighteen individuals tested and compared the old system GUI to the newly designed GUI. Individuals tested were not necessarily swim coaches since the goal of this interface is to be usable by the general public. Two times were recorded for each participant, the time spent before s/he clicked a button and the time it took to complete the setup procedure successfully. Nine participants started with the calibration of the old GUI and nine started with the calibration of the new GUI.

Of the eighteen people surveyed four were unable to complete the calibration process on the old system and one did not complete it on the new system.

The mean calibration time for users on the old GUI was 138 seconds with a standard deviation of 13.75 seconds. The mean calibration time on the new GUI was 119 seconds with a standard deviation of 12.8 seconds. This indicates that the new GUI is more intuitive and facilitates a faster calibration process particularly for first time users.

The ability to correct the corner values was used by all users in the survey. These results indicate that the new GUI was more user-friendly as well as extending the functionality by adding the ability for coaches to automatically record swimmer's lap and swim time.

CONCLUSION

The new GUI was successful making the electronic swim coach more intuitive and easier for non-technical users to calibrate.

Moving the electronic swim coach project from C++ to Java and employing the MVC design pattern has resulted in a more robust and versatile application that facilitates code reuse and further application.

In future, implementations on a touch-screen tablet, implementing a security system for swimmer data and improving the functionality to allow real time feedback from the coach to the swimmer are possible further improvements.

ACKNOWLEDGEMENTS

This work has been partially supported by the Kiwanis Medical Foundation. We would like to thank the previous students who worked on this project: Ramzi Marjaba, Catalin Patulea, Trevor Gelowski, Jaclyn Baldwin, David Galarneau, Adam Jones, Alexa Loiskandl, Abhilash Narra, Myna Moharib, Hugo Vihvelin, and Davide Agnello. Dr. Andrew Marble has consulted and assisted in many stages of this project. Finally we would like thank all volunteer participants and Miss Lucy Barnes for her contributions.

REFERENCES

[1] A. Narra, M. Moharib, and H. Vihvelin, "Development of a Low-Profile Electronic Notification System for Visually Impaired Swimmers," Ottawa, 2011.

[2] S. J. Metsker and W. C. Wake, *Design Patterns in Java*. Addison-Wesley Professional, 2006.

[3] I. Culjak, D. Abram, and T. Pribanic, "A brief introduction to OpenCV," *MIPRO, 2012 Proceedings of the 35th International Convention*, pp. 1725–1730, 2012.

[4] S. Audet, "Java CV." [Online]. Available: http://code.google.com/p/javacv/. [Accessed: 07-Feb-2013].

[5] C. Harris and M. Stephens, "A combined corner and edge detector," *Alvey vision conference*, pp. 147–152, 1988.

[6] "RXTX," 2008. [Online]. Available: http://rxtx.qbang.org/wiki/index.php/Main_Page. [Accessed: 07-Feb-2013].