ES50 Final Project Writeup
Team Cyclops: Motion-Controlled Laser Shooting
Team # 19

Student A, Student B

December 2014

School of Engineering and Applied Sciences
Harvard University
33 Oxford Street Cambridge MA.

Description: A motion-controlled laser pointer using a native web camera.

Class: ES50
Instructor: Marko Loncar Christopher Lombardo

Abstract

We adapted object-tracking software to control the movement of a custom-constructed laser with the motions of a hand or any object. This project is especially of interest to us because of the wide range of applications of motion-tracking. Some of these include assistive technologies, computer usability, and augmented reality systems. The primary goal of our project is to be able to successfully track the movement of an object to control the direction of a laser beam.

1 Introduction

As two computer science concentrators, we wanted to explore the intersection of software and hardware. We were particularly interested in how visual cues like hand motions could be leveraged as a salient way to control the movement of objects. Thus, we built a motion-controlled laser beam that capitalized on our software engineering skills. The direction of the laser beam was determined by the relative location of an object that was being tracked by a web cam. For example, as one of us waved our hand up and down in front of the camera, the laser beam moved up and down as well. A rudimentary predecessor to technologies like the PlayStation EyeCamera and the XBox Kinect, our project developed from our interests in computer vision - a computer science research field dedicated to processing visual information - and the many applications that motion-tracking can have, such as a means for those with poor fine-motor control to interact with computers.

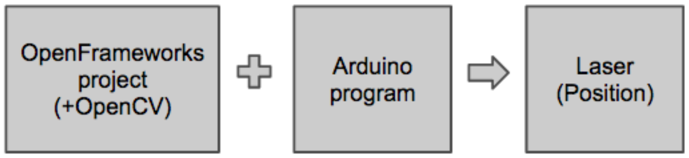
2 Design

2.1 Project Pivot

Originally, we had wanted to make a eye-controlled laser pointer and had acquired an eye-tracking camera called the EyeTribe from the David Cox vision lab that Student A is an undergraduate researcher in. However, we had some trou-ble getting timely accessing to the software and developer's kit, so we chose to switch over to making our laser pointer motion-controlled instead. We found a great tutorial that was roughly what we wanted to do (see References) and used it as our design framework.

2.2 Overarching Design

Our project had two software components and one hardware component. The below figure shows how the three components interact with one another.



2.3 Software Architecture

The first component was a piece of software that used the OpenFrameworks and OpenCV libraries to extract the location of the largest object in a camera's view that is in contrast to the background (i.e. Emily's black iPhone case in contrast to a white board background) and sent those locations to a specific serial port.

The second component was Arduino code that listened to for those position coordinates at the pre-determined serial port and used them to control the movement of two servos. Basically, a servo controlling the laser's horizontal movement was set to the x-coordinate of the location received on the serial port, while the servo controlling the laser's vertical movement was set to the y-coordinate of the location received on the serial port.

This had the effect of pointing the laser in the direction of the location that the web cam detected an object at.

2.4 Hardware Design

The third component was our hardware piece which comprised of two servos mounted together with the laser attached to one of them so that one servo controlled the horizontal movement of the laser while the other controlled the vertical movement of the laser.

In the next section on project implementation, we discuss our several prototypes.

3 Parts List

Here we include details and links to the parts we used in our project.

3.1 Parts Details

Part Name (Qty)	Part Number	Total Price
PlayStation EyeCam 3 (1)		$9.25
Logitech HD C615 Webcam (1)		$49.99
Laser Pointer (1)		$4.95
Standard (non-continuous) servo* (2)	900-00005-ND	$25.98
Arduino UNO board* (1)	DEV11021	$29.45
Solderless Breadboard* (1)	377-2094-ND	$4.50
Battery Pack* (1)	LR6XWA/BF2	$1.79
AA Batteries* (2)	LR6XWA/B	$0.70
	Total Price:	$126.61

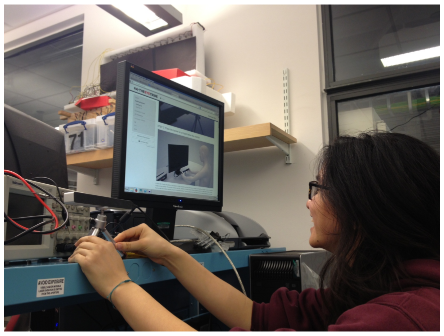
* Denotes parts that we used from the ES50 lab.

3.2 Parts Links

- PlayStation EyeCam 3: `http://www.amazon.com/PlayStation-Eye-3/dp/B000VTQ3LU/ref=sr_1_1?ie=UTF8&qid=1414855780&sr=8-1&keywords=ps3+eye+cam`
- Logitech HDC615 Webcam: `http://www.amazon.com/Logitech-Portable-1080p-Webcam-Autofocus/dp/B004YW7WCY/ref=sr_1_1?ie=UTF8&qid=1415570105&sr=8-1&keywords=Logitech+-+HD+Webcam+C615`
- Laser Pointer: Dickson Bros.
- Standard (non-continuous) servo: `http://www.digikey.com/product-detail/en/900-00005/900-00005-ND/361277`
- Arduino UNO Board: `https://www.sparkfun.com/products/11021`
- Solderless Board: `http://www.digikey.com/product-detail/en/BB-32621/377-2094-ND/4156445`
- Battery Pack: `http://www.digikey.com/product-detail/en/LR6XWA%2FBF2/P646-F021-ND/2043805`
- AA Batteries: `http://www.digikey.com/product-detail/en/LR6XWA%2FB/P646-ND/2043737`

4 Project Implementation

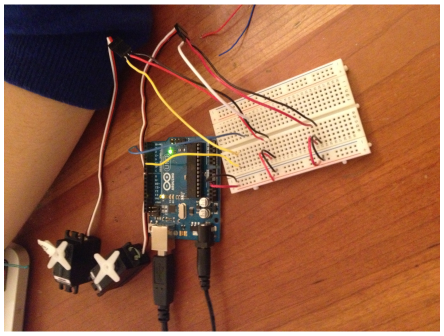
Week 1: After doing substantial research to see if it was possible to get Arduino code to talk with the EyeTribe eye tracking code, our first step was to attempt to obtain access to the EyeTribe Software Development Kit and set up our coding environment. We ran into some difficulties accessing the SDK with the proper credentials and thus quickly shifted focus from eye tracking to motion detection instead.



Week 2: After shifting the focus of our project to motion detection, we spent much of the second week of project implementation on reading up on various object-tracking software. We soon came across OpenCV (Open Source Computer Vision Library), an easy to use and open source computer vision and machine learning software library. We also discovered openFrameworks, an open source C++ toolkit that is designed specifically for creative and experimental projects, bringing together a number of packages including OpenGL (graphics), OpenCV (computer vision), and OpenAL (audio) among others. After doing a bit more research, we discovered an Arduino + Servo + openCV tutorial that seemed to have goals that were closely aligned with the goals of our own project. This tutorial was exactly what we were looking for. It provided an intuitive walk-through of how to get openFrameworks and Arduino programs talking to each other.

As we learned in the tutorial, communication between the Arduino board and the openFrameworks code can be handled through a Serial port. Thus, we connected the Arduino to our computers using a USB cable. The Arduino application then listens for data and the openFrameworks application listens on a particular port.

Week 3: During the third week of our project implementation, we began to build the hardware portion of the project. Using the two continuous servo motors that we received from the lab, we wrote Arduino code to control the two servo motors (one for horizontal movement and one for vertical movement). To do this, we provided the servo with power from the 5V pin on the Arduino.

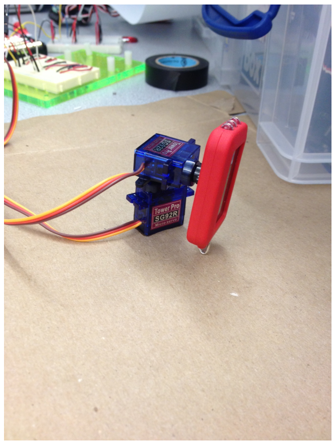


We played around with the parameters sent to the servo motors until we were satisfied that the servo moves were accurate representations of the motion detected by the openFrameworks code. Because these servos were continuous, however, we had some difficulties getting the parameters right.

Week 4: In order to escape the difficulties we were having with the continuous servo motors, we talked to our lab TF to obtain standard non-continuous servo motors instead. The important distinction between these two types of servo motors is that on a standard servo, the `write()` function sets the angle of the shaft (in degrees), moving the shaft to that specific orientation. On a continuous rotation servo, however, the `write()` function sets the speed of the servo (with 0

being full-speed in one direction, 180 being full speed in the opposite direction, and a value near 90 being no movement. Clearly, working with the standard servos made it much easier to accurately control the direction of our laser beam.

Also in the fourth and final week, it was imperative that we obtain a laser beam for our laser beam rig. Though we did have the opportunity to use a small laser from lab, we instead wanted to use a light that did not need to be wired up to a breadboard and thus purchased a small solar-powered LED light as an alternative. We used superglue to glue together our newly acquired horizontal and vertical direction standard servo motors and also used superglue to attach the LED light to the servo that reflected vertical motion.



Fair preparation: The last step in our project was to prepare for the ES50 fair! In order to make our motion detection program more accurate, we obtained a whiteboard form lab and constructed a sheet of white paper with a black square on it that we could move around on the whiteboard. This black square would serve as our object to be detected and made the direction of our servo motors far more accurate than if we had not used the white background.

5 Team Management

Because we have worked together on projects in the past and have worked best by pair-programming, we continued following this working strategy and did the same for this ES50 project. Additionally, we were both interested in working on both the software and hardware components of the project. In turn, we worked together on building the laser beam rig and the software portion of the project. Lastly, we both attended the ES50 fair to showcase our project and split the work on the writeup evenly as well. We recognize that most other teams divvied up tasks, but given our small team size and our pre-existing working relationship with one another, we felt that we were most effective and learned the most from this project by tackling our different software and hardware problems together.

6 Outlook and Possible Improvements

6.1 Eye-Tracking Extension

We originally wanted to do an eye-tracking project but switched to motion-tracking when we struggled to gain access to the eye-tracking software for our special gaze-tracking camera in a timely manner. However, a day or two before the ES50 fair, we were finally authorized to access the software. Unfortunately, we did not have enough time to switch back to our original idea, but if granted more time, we would love to extend our project to eye-tracking.

6.2 Improvements to Current Design

There are several aspects of our project that we would have liked to improve of extend if given more time.

6.2.1 Improve Object Tracking

First, we would want to improve our object tracking software. We did not spend too much time perfecting it once we had a working prototype. A side effect of this was that it could not deal with diverse background color and the presence of multiple objects. The way it worked was that it tracked the largest moving object. This became problematic when the arms of individuals with darker skins entered the view of the camera and were tracked instead of the object in those individuals' hands. Thus, we would have wanted to improve our software so that it would track a specific object, such as a stuff animal, with an array of background noise.

6.2.2 Improve Software Architecture Interactions

Another aspect we would have wanted to improve was the connection between the two pieces of software. We suspect that our Arduino code did not pick up on all the local signals sent over the specified serial port. While time prevented us from debugging this issue thoroughly (as it did not severely negatively affect our project), we would have wanted to investigate this effect more.

6.2.3 Improve Laser Movement

Finally, our resulting motion-controlled laser pointer moved in the general direction of the location of a tracked object. However, its movements were often jerky and were not as specificly tuned to the location of the tracked object as we would have liked; thus, we would have wantd to fine-tune the Arduino code more to better control the speed and movement of the servos.

7 Acknowledgements

We are especially appreciative of all the help we got from the course staff. In particular, we would like to thank Professor Marko Loncar, Professor Chris Lombardo, Xuan Liang, and Joy Hui. Xuan and Joy were especially responsive in helping us find the necessary parts for our project.

8 Disclaimer

Yes, it is okay to share our report, codes that we wrote, photos and videos of our project.

9 References

9.1 Works Cited

Here's a list of miscellaneous references we used in our project.

- Arduino + Servo + openCV Tutorial (`http://www.creativeapplications.net/tutorials/arduino-servo-opencv-tutorial-openframeworks/`)
- OpenFramework API (for object tracking) (`http://openframeworks.cc`)
- OpenCV API (for object tracking) (`http://opencv.org/`)
- Arduino API (`http://arduino.cc`)

9.2 Works Consulted

Here's a list of references that we consulted for our original eye-tracking project idea:

- Eye Tribe API (`http://dev.theeyetribe.com/general/`)
- Example Project for Programming an Arduino in C++ without the Arduino IDE (`https://github.com/WeAreLeka/Bare-Arduino-Project`)

- Guide for Eye-tracking project that uses electrodes and an Arduino (`http://makezine.com/projects/Eyeboard-Electrooculography-EOG-System/`)
- Guide for eye-tracking project that uses a PlayStation 3 EyeCam and an Arduino (`http://www.instructables.com/id/The-EyeWriter-20/?ALLSTEPS`)

10 Appendix

Note that the **Arduino** and **openFrameworks** source code is included in the zipped archive.