DVS Winter 2014

# Table of Contents

[Table of Contents](#h.1ss21fo8zvmp)

[Team Contact Info](#h.ai1lzeiufzsr)

[DVS Mission /Introduction](#h.y6fmdzn0qbmc)

[Summary](#h.jmicpy63mr5v)

[Client Overview](#h.s6deyot9d265)

[Hardware](#h.phxyjak8wg0r)

[Purpose](#h.nmha129k5por)

[Quarter Progress](#h.mstth7vvyx)

[Next Steps](#h.a6g7cf1s0mbz)

[Long Term Goals](#h.umcio4a6whim)

[Software](#h.tmgrqckis3wx)

[Front End](#h.e75z4c5rxmzs)

[Purpose](#h.wtbl4en2b1mf)

[Quarter Progress](#h.2w4zob8ltt83)

[Next Steps](#h.1452vyrny1hi)

[Long Term Goals](#h.6sgezloijuc3)

[Back End](#h.i3btk77xgaud)

[Purpose](#h.u3fugoryykuu)

[Quarter Progress](#h.i8dzz2u8y2fa)

[Next Steps](#h.tt2fnfw3s37v)

[Long Term Goals](#h.sk2uzjplti8)

[Appendix](#h.5ktrzkpe21i3)

# Team Contact Info

|  |  |  |  |
| --- | --- | --- | --- |
| **Client/Shiley Eye Center Director** |  | **Illiana Molina** | [**imolina@ucsd.edu**](mailto:imolina@ucsd.edu) |
| **DVS Email Address:** | globaltiesdvs@gmail.com | **Password:** | **jacobs4702[insert door code here]** |

# DVS Mission /Introduction

*by Shumin*

The DVS team is creating a user-friendly system that identifies children’s eye diseases and helps our client, the Shiley Eye Center, to provide proper vision care for children from low income families. The whole system, which includes hardware and software, will be done by student engineers at a low cost.

The hardware team, in this quarter, designed a prototype bracket and solved the issue of camera and flash rotation. The software team’s goal is to complete three main algorithms, strabismus, cataracts, and astigmatism detection.

To be more specific, the process of eye diseases detection is divided into three teams. The Hardware Team designs a bracket that can hold a camera and flash to take red-eye photos in both vertical and horizontal. The Image Analysis/Backend Team is developing algorithms which detect any eye abnormalities present. The GUI/Frontend team is responsible for creating a user-friendly interface to integrate the work of Hardware Team and Backend Team.

# Summary

*by Xiaomeng*

Our project is aiming to design and update a digital vision screening system that identifies children with potential eye problems and provides comprehensive vision care for preschoolers and young or disabled children that are afraid of or cannot focus on an autorefractor held near their face. How our system works is that at first the hardware team will use the bracket we developed to take pictures of patients from a distance, which enables red eye photos both taken in vertical and horizontal. Those pictures will be used to be tested with our own algorithms, which determine if the patient’s eye have refractive errors. The diseases we detect include strabismus, cataracts, and astigmatism.

# Client Overview

*by David*

Our client is the Shiley Eye Center. We work with the Save Our Children’s sight program and interface with the center through Iliana Molina. The center performs visions screenings at local schools and at a location called the regional center. Most of the children that they screen are young and some have disabilities. This can cause problems with the traditional vision screening approach of putting an autorefractor right up to the child’s face and asking them to focus on it. As such, our client needs a system that can screen children from a fair distance away.

In future quarters we are planning on having the client give us feedback by emailing the dvs email address (see team contact info for specifics). Someone will have to monitor this address and list the issues on github issues (with appropriate tags for priority, feature request vs. bug fix, etc).

# Hardware

[Hardware Documentation on GitHub](https://github.com/UCSD-TIES/DVS-Python/blob/master/doc/HardwareDocumentation.docx)

### Purpose

*by Arvind Rao*

The purpose of the Hardware team is to provide the DVS team with a method to produce high-quality red-eye photos that can be used as inputs for the software. In order to analyze the eye for various defects, it is essential that a crescent formed by the Bruckner reflex is visible in the pupil within the patient photo. Our device must also differ from the conventional auto refractor, which gets very close to the children’s eyes. This can be troublesome for young, challenged or disabled children.

Our solution to this need is a rotatable bracket that holds the camera and the flash. The rotatable feature allows for the taking of “horizontal” and “vertical” photos, in which the orientations refer to the flash position relative to the camera. This allows us to obtain the crescents on both axis of the eye.

The hardware team is also responsible for finding the right measurements and specifications to maximize the efficiency of the camera and flash to produce high-quality red-eye photos on a consistent basis on people of all types. Since our device takes photos from a good distance (compared to auto refractor), it allows the screener to take a quick photo of the child, and screen for any problems just through the image itself.

### Quarter Progress

*by Arvind Rao*

The hardware team this quarter has made progress in several ways this quarter. In the prior quarter of Fall 2013, the hardware team only had prototype ideas designed on paper. We began this quarter by testing our camera and flash on some members of our team to see if we can consistently obtain red-eye photos. We conducted our tests in rooms with dark lighting for maximum effect. However, the client has informed us that the system will be implemented in a normally lit room. In darkness, we can obtain red-eyes very well on caucasian people. On Asians, we can somewhat obtain a discolored-pupil, enough to see the crescents being formed.

For the second half of the quarter, we got working on designing and creating a better functioning bracket. We have purchased a commercial, rotating camera bracket that we will add the camera-and-flash bracket onto, which solved the issue of the camera and flash rotation. Our camera-and-flash bracket places the flash as close to the camera as possible, to maximize the chance of inducing a red-eye effect. The camera-and-flash bracket was formed with acrylic.

### Next Steps

*by Andrew*

The next step for the hardware team is to implement an actual beta prototype. This beta prototype bracket will be actually used for the client testing and feedback. We are going to need to draft more designs that improve on what we have currently and we will need to pick out different materials of choice for our bracket that will make it sturdier and more aesthetically pleasing.

Furthermore, additional tests that the hardware team would like to make is taking more test photos with the bracket. Testing photos will enable tweaks to the bracket design to see if there can be ways to improve the design to take more red eye photos.

### Long Term Goals

*by Andrew*

Longs term goals for the hardware team is to have a finalized bracket that will work all the time. The bracket has to be extremely durable and easy to transport around. The bracket camera and bracket should also be easy to make if anything happens to it.

We would like for the camera to be able to take red eye photos consistently, and if there are issues, we would like to have instructions to troubleshoot through the problems to be able to take red eye photos.

# 

# Software

Our software and documentation resides on github, [here](https://github.com/UCSD-TIES/DVS-Python).

You can learn more about the software by looking at our [docs](https://github.com/UCSD-TIES/DVS-Python/tree/master/doc).

**Start with DevelopersGuide.docx** [**here**](https://github.com/UCSD-TIES/DVS-Python/blob/master/doc/DevelopersGuide.docx)**.**

## Front End

### Purpose

*by John*

The purpose of the front-end is to create a user interface that fits the client’s needs and facilitates the tasks that the client needs to perform. The client needs a product that’s simple and intuitive to use since they will be screening many kids per screening session; they want to be able to get through a screening as fast as possible. Our UI should ease this process. Simplicity and user friendliness should be prioritized over aesthetics.

Also, the front-end code should support the back-end code. This means that it’s the front-end’s responsibility to correct any of the mistakes that the back-end code might make. For example, the back-end code doesn’t have a 100% eye detection rate. This is why the UI provides the user with the capability of boxing the eyes on the picture if the back-end detected the eyes incorrectly. Then the UI can use this correction and pass the information back to the back-end so that the eye analysis can continue accurately.

### 

### Quarter Progress

*by Rolando*

Last quarter the front-end created the user interface and bridged the front-end with the back-end. But it was not interactive. We only had one functionality at the beginning of the quarter: uploading pictures. What we did this quarter was to present the analysis from the back-end to the user. The current iteration allows the user to visualize back-end detection algorithm and analysis through us. This is facilitated in two ways.

For eye detection, we grab the coordinates off back-end’s detection algorithm and draw rectangles into the uploaded pictures. In case the back-end method fails and the eyes are not detected, we have added mouse event handlers and listeners to allow the user to manually correct the eye detection algorithm by simply drawing the square themselves. This new set of coordinates is then send to back-end for reanalysis.

Once that is done, we have added a results page to present the relevant information acquired by the back-end analysis. Do note that this rough diagnosis will only get shown in the results page if the back-end code analyses the eyes correctly.

### Next Steps

*by Mark*

The next step we need to work on for the frontend UI is improving the elements of the results page. This includes dividing the entire panel into different sections that contain related information about the result(s). For example, currently we have all the results displayed on the result page. We need to format the results so that it is easier for the user to read and understand the information displayed. In addition, we need to display the patient’s status as “Refer” or “Do not refer”, which is not reflected in current prototype result page.

We are also adding new pages like the pupil correction page, crescent correction page, and white dot correction page. These pages all come after eye correction page, which is already implemented in the current UI and is functional.

Overall, we will need to refine some of the pages so that it looks better to the users and to build the necessary addition pages..

### Long Term Goals

*by Steven*

For the project moving forward, the front-end team plans to add in user correction for pupil detection. Possible additional pages would be user correction for white dot and crescent detection, depending on the accuracy of the backend’s white dot and crescent detection. The ending results/diagnosis page may as well be revamped to give more relevant data. After that, the core functionality will be tentatively complete. What will be left will be code cleanup, refactoring, optimizations, taking in client feedback, and taking account of user friendliness.

## 

## Back End

### Purpose

*by Brian*

The purpose of the back-end is to analyse the image and develop an algorithm behind the user interface. These algorithms determine if patient’s eyes have anisometropia (unequal refractive power), astigmatism (blurred vision), or strabismus (misaligned eyes). A back-end application or programs serves indirectly in support of front end services, by being closer to the required resources or having capabilities to communicate with the required resources.

### Quarter Progress

*by Brian*

This quarter we were able to get strabismus/anisometropia detection, got the reset functions working and ran through a lot of testing. The strabismus detection we ran while with everything normal, but we ran into problems because the pupil detection isn’t perfect yet, so that’ll cause some problems. Also, the picture we tested on was low quality, the pictures we’ll be using will have higher quality allow us to get more accurate results. Other than that, we able to run it with the resets and got expected results.

After testing the detections a tad, we moved over to pupil detection again to try to make it better. We were able to get an old version of a working source code that we’ll try to implement into our python code. We’ve been approaching the problem in many ways and hopefully we’ll reach one sooner or later.

### Next Steps

*by* Si

Next quarter, the back-end team hopes to accomplish three major goals. First, we want to write an algorithm to detect cataracts. Second, we want to strengthen the reliability of the eye diseases detection by improving detection algorithm and eliminate bugs. Also, we hope to use more standard medical algorithms to improve overall correctness. Pupil detection needs to be proved by using perhaps an arc object detection oriented algorithm instead of using a circle object detection oriented algorithm. Third, we want to analyze the accuracy and reliability of detection and diagnosis on people with different ages, different genders, different skin colors, different eye abnormalities, different fabric patterns on their outfits, with and without earring, different ethnicities and any other variation in appearance that might affect detection. These factors are all variables that present their own unique challenges to accurate detection and diagnosis.

The table below shows the detail of bug fixings

|  |
| --- |
| **Hardware:** |
| Red-eye doesn't work except on white people |
| Bracket, hard to take out flash |
| Looks unappealing |
| Not attached to rotating bracket yet |
| Eye fix not set up yet |
| Second prototype needs to be done |
| **Front-end:** |
| 2nd page - Photos get shifted to the left |
| Eye correction page - Drawing 2nd rectangle on a picture causes the first one to disappear until mouse is released. |
| Eye correction page - MACS ONLY - Zooms in when trying to draw rectangle on picture |
| Pupil correction page not implemented |
| White dot and crescent correction not implemented |
| Reset button on eye correction not fully functional |
| Reset button is not working |
| Not aesthetically appealing yet |
| Comments and documentation need to be updated |
| **Back-end:** |
| Pupil detection trips out |
| Eye detection is not perfect |
| Crescent detection is not functionally tested |
| White dot detection is not really tested |
| Cataracts detection is not drafted or implemented |
| Reset pupil is not tested or robust |
| We don’t delete the photos written to disk |
| The code is rather slow on large photos |
| Code needs to be commented better |
| Pupillary distance is not converted to mm |
| Pupillary distance is not averaged |
| Strabismus code is not based on healthcare standards |
| All disease detection algorithms have not been well tested |
| resetEyes (patient ((1,10,20,30), (2,11,21,32)), ((101,102,140,156), (123,141,202,200))) gives a local var pleftcenterx reffed before assignment error |
| Code documentation needs to be updated and more documentation needs to happen |
| Left and right might be interpreted differently in front end and back end |

### Long Term Goals

*by Shannon*

Long term the Backend team would like to improve eye, pupil, white dot, and crescent detection to the point that the user does not need to correct any of the detection with the front end the vast majority of the time. We would like to provide support for strabismus (esotropia,exotropia, etc), astigmatism, hyperopia, myopia, anisocoria, cataracts, and anything other disease we can use our data for. However, the client has expressed that the most important of these are strabismus, astigmatism,hyperopia, myopia and cataracts. We would like our diagnoses to be accurate to a medical vision screening standard, proven reliable, and fast.

# Appendix