DVS Fall 2013

# Table of Contents

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

[Team Contact Info](#h.qcm64uqi325)

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

[Summary\*](#h.jmicpy63mr5v)

[Client Overview](#h.s6deyot9d265)

[Hardware](#h.p8w4fcs5cwqp)

[Purpose](#h.nmha129k5por)

[Quarter Progress](#h.mstth7vvyx)

[Next Steps](#h.a6g7cf1s0mbz)

[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.vhjv06bg4v5d)

[Next Steps](#h.tt2fnfw3s37v)

[Long Term Goals](#h.sk2uzjplti8)

[Appendix](#h.5ktrzkpe21i3)

# Team Contact Info

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Role*** | ***Subteam*** | ***Name*** | ***Email*** | ***Phone*** | ***Github ID*** |
| *Team Lead* | *Backend* | *Shannon Harris* | *svjharris@gmail.com* | *510-690-5523* | *svjharris* |
|  | *Backend* | *Arvind Rao* | *arvind.rao.us@ieee.org* | *408 439 2488* | *arvievlv* |
|  | *Frontend* | *Steven Yuan* | *s6yuan@ucsd.edu* | *408-316-1017* | *keytrain* |
|  | *Backend* | *Shumin Gao* | *shumin0809@gmail.com* | *253-320-6334* | *shumin0809* |
| *Client Liaison* | *Backend* | *David Chan* | *snodrew.chan@gmail.com* | *805-708-7633* | *snodrew* |
| *Backend Lead/CGI/Budget Master* | *Backend* | *Brian Nguyen* | *bhn013@ucsd.edu* | *619-252-6341* | *bhn013* |
| *Frontend Lead/CGI/Budget Master* | *Frontend* | *John Chan* | *jmchan100@gmail.com* | *858-413-6823* | *jmchan100* |
|  | *backend* | *Jonevan Brown* | *rospacebrown@gmail.com* | *209-485-4388* | *plasmasandwich* |
|  |  | *Tianyu(Mark) Ma* | *tianyu.ma2009@gmail.com* | *415-519-3357* | *408796571* |
|  | *frontend* | *Andrew Fong* | *awallfong@gmail.com* | *916-396-5570* | *awfong* |
| *Webmaster/Hardware Team Lead* | *frontend* | *Andrew Nguyen* | *nguyen.andrew1995@gmail.com* | *619-252-8859* | *anguy95* |
| *TA of Awesome* |  | *Andreas H Djokic* | *adjokic@ucsd.edu* | *909-844-2607* | *andreasdjokic* |

# DVS Mission /Introduction

*by Mark*

The Digital Vision Screening(DVS) team strives to produce stable software that, together with the hardware we create, can diagnose children’s eye abnormalities such as myopia and hyperopia, thereby advising them on whether or not they should do further testing to further confirm the potential eye related diseases. The system will be implemented natively (without internet connection) and at a low cost.

To be more specific, currently our client goes out into the greater San Diego area to screen young children for vision problems. One or two screeners will go to a location and set up shop in a small room (perhaps about 8-10 feet wide). Children will come in and be screened using an autorefractor like [this](http://farm6.staticflickr.com/5108/5578507404_cd6ca2ebc0_z.jpg). The problem our client faces is that children with disabilities such as autism are scared by the closeness of such a screening. Thus, we’re tasked with creating a product similar to the [Plus Optix](http://www.plusoptix.com/lang-en/vision-screener.html).

# Summary\*

*by Shumin*

The Digital Vision Screening (DVS) project, which is a student project of the University of California, San Diego Teams in Engineering Services (TIES), works to serve the Shiley Eye Center. The program aims to identify vision disorders in children from low-income households early enough to prevent them from turning into more serious conditions in the future. It is very common that if a child’s vision disorder is left undetected, the affected eye will stop functioning and the child will only be able to see through one remaining eye. One of the biggest challenges our client faces is that their system is hard to use on anxious, scared, or autistic children.

Our objective is to update the hardware and software by employing an up to date high-resolution digital camera that will use a custom made bracket to hold a flash and an operating software interface that operates with a computer. DVS employs the use of a new-age digital camera, an external flash, a computer, and vision screening software.

The external flash, which will be powered independently from the digital camera, allows for easy integration on newer camera models. The idea behind the flash is to provide a flash device that will maximize the occurrence of red-eye. Since capturing red-eye photos is a key precursor to analyzing the Bruckner reflex, maximizing the red-eye effect will better facilitate the detection of vision disorders. The refractive properties of the retina will result in a completely filled donut shape around the iris for a healthy eye, whereas this shape varies in those with vision disorders. The most common occurrences are crescent shapes either below or above the pupil which can be diagnosed as hyperopic or myopic.

The Digital Vision Screening team is split into three sub teams: Hardware, Graphical User Interface (GUI) (Also called Front End), and Image Analysis(also called Backend). The hardware team has made great strides in red-eye research; discovering that a stronger flash at a certain angle and distance is needed. The Image Analysis Team has continued coding, developing and implementing image analysis algorithms. The Graphical User Interface team made progress in implementing GUI functions for the Image Analysis program. The application will be combined with the Image Analysis implementation and will provide a more user-friendly interface to facilitate ease of use. The user will be able to select that patient’s picture, run the Image analysis program, correct any detections if they have gone wrong, and output a result that determines whether the patient should undergo a full eye exam. Some of the user interface functionalities are operational and most of the image analysis detections are completed, though the disease detection code for the backend has not been written or implemented.

\*This section has been adapted from previous continuity reports (which can be found on the DVS [dropbox](https://www.dropbox.com/sh/8nsmamaiklpvr9l/Fo5xgv4ogH))

# Client Overview

*by David*

The Shiley Eye Center of UC San Diego is a cutting edge facility in research of surgical methods and treatments for eye diseases, clinical treatment and care for patients with eye disorders, as well as education in the field of ophthalmology. In addition, the center reaches out to the community through innovative programs. The Digital Vision Screening team of UCSD Global TIES works with a program of the center, the [EyeMobile](http://shileyresearch.ucsd.edu/eyemobile/), helping reach out to the those in need within the community and provide less costly screening diagnosis.

# Hardware

### Purpose

*by Andrew N.*

The purpose of DVS’s hardware team is to supply a way to obtain red-eye photos that will be used as input to DVS’s software in order to analyse for any defects or abnormalities in the eye. To do this we are to create a bracket that houses both the camera and flash. Because we need to analyze the crescent formed by the [Bruckner reflex](https://www.dropbox.com/sh/8nsmamaiklpvr9l/CDAFMoZM7A/Dr.%20Bartsch%27s%20DVS%20slides/vision-screening.ppt) we need to create a bracket that can take one photo with the flash on the side of the camera and another with it below.

This bracket will it allow for quick and easy eye analyses for vision screeners dealing with mentally retarded or disabled children. Also, while creating the bracket the hardware team is responsible for finding the right measurements and specifications to maximize the efficiency of the flash and camera to produce consistent and quality red eye photos that can be analysed. All of the hardware is designed to make diagnoses of patients’ eyes easier. For example, if a child comes in for screening and has autism, conventional methods of putting an autorefractor in front of their face is not optimal. So, we are trying to allow screeners to just take a quick picture of the child and diagnose this autistic child solely through the image.

### Quarter Progress

*by Andrew N.*

Hardware team for DVS’s fall quarter headed off at a bad start. The previous team’s hardware was not found until the middle of Fall quarter. Also, there was little to no hardware bracket instructions. What previous quarters’ hardware teams have done can be found in the DVS [dropbox](https://www.dropbox.com/sh/8nsmamaiklpvr9l/Fo5xgv4ogH). However, that was not really an issue as we made a pretty large comeback. The hardware team has come up with a few new designs and is currently working on finalizing one of these designs. The bracket is designed in order to accommodate a Canon EOS Rebel T3 and YN560 flash that can be rotated around an axis to take both landscape and portrait pictures. The hardware team has also managed to take a few red eyed photos consistently testing it on JT and Shannon – these photos worked about four out of five times. All of our hardware information can be found in the [Camera Documentation file](https://github.com/UCSD-TIES/DVS-Python/blob/master/doc/HardwareDocumentation.docx) in github.

### Next Steps

*by Andrew N.*

Our goal for the hardware team as of now is to fabricate a cost effective and efficient bracket system. The design that we are leaning towards to fabricate are designs 1 and 3 (these designs can be found on the [Hardware Documentation file](https://github.com/UCSD-TIES/DVS-Python/blob/master/doc/HardwareDocumentation.docx) in github). The hardware team would like to fabricate a bracket by the Winter or Spring quarter of 2014. Also, we would like for the bracket to be modular, allowing for different cameras, lenses, and flashes to be attached. This bracket will be made of acrylic and or aluminum because these materials are relatively cheap and can be easily assembled together using solvent welding and or nuts and bolts.

Long Term Goals

*by Andrew N.*

Our main goal for the hardware is to have it be ergonomic, long lasting, cost effective, and efficient when the final product is created and given to the client. The reason being for the above is because the doctors who are scanning the eyes of the children must have a way to take the image quickly, so the device cannot be heavy and maneuverable. Not only does the hardware have to fit the description above, but it also has to be made so that there will be a consistent and high level of confidence in providing quality red eye photos (information on reasons for red eye can be found on [Hardware Documentation](https://github.com/UCSD-TIES/DVS-Python/blob/master/doc/HardwareDocumentation.docx) in github). Another major goal is to have the bracket be finalized and a prototype made by Spring Quarter of 2014.

# 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.

## Front End

### Purpose

*by John and Steven*

The purpose of the front-end team is to design an interface that abstracts enough of the back-end to let the user do only what is necessary: find the possible diagnosis of a patient.

An outline of the steps that need to be taken to get a user through the whole process of the front end can be found in the Whiteboard Pics section of some of the agendas from Spring 2013. Specifically [this agenda](https://github.com/UCSD-TIES/DVS-Python/blob/master/doc/Agendas/Spring2013/DVS%20Agenda%2005-04-13.docx), though one might also find adjacent agendas useful.

### Quarter Progress

*by John and Steven*

At the start of the quarter, pagination had been completed, but we were finding trouble modifying the existing code to do a task as simple as centering the text. Since there is far more to implement than text centering, code refactoring had to be done. We started new, trying to implement OOP principles, and wxPython had to be learnt and dealt with from scratch. Circular dependencies delayed progress for a while, but by week 8 we had all the progress of the previous UI and then some. With the core of the front-end now solid, the rest of the steps should go much more smoothly.

***We’ve split up the front-end into 4 files***:

**Base** - This file is the foundation for all the other files. It’s the class for what is commonly called a “window”, but in GUI design, it’s the frame. The frame is the object that holds everything in the UI inside of it. It contains the panels, the sizers, etc.

**Page** - This file contains the panel part of the UI. The panels hold the content inside of the frame, which can be anything like text boxes, buttons, the pictures, etc. Each panel represents a page of the UI. To move to the next page, you hide the current panel and you display the next panel.

**Interactions** - This file contains all of the buttons that are on the panels. A class called Interactions controls all of these button; an interactions object is made in the Page file to call all of these functions.

**Main** - Just runs the program.

### Next Steps

*by John and Steven*

With the code cleaned up, the next steps will be to implement the rest of the pages of the program. These pages include asking the user if certain parts of eye detection were done correctly, and if not, to let the user modify it as needed. The final page(s) will tell the user the resulting data and possible diagnosis. The number of pages will vary depending on the amount that the back-end will need; it depends on how many parts of the eye that they need to analyze. Say they need to analyze 3 parts of the eye. There will be a total of 8 pages. The first page will ask the user to upload the picture; this we have done. The following 6 pages will follow a pattern. On one page, The front-end will call the back-end code which will detect a component of the eye. The front-end will then display to the user the results of the back-end code and ask the user whether or not the back-end detected the component correctly. If the user says no, the front-end will display the second page, which will ask the user to correct the detection. For example, if the code didn’t properly detect the eyes, then the user will be able to draw boxes around the eyes. We can then take the user input and pass it back to the back end. The back-end can then take the corrected input to properly continue the eye analysis. It’ll follow this pattern for the next 6 pages. The final page will have the results of the analysis displayed.

### Long Term Goals

*by John and Steven*

Eventually, all pages should be complete and fully functional, leading to a proper diagnosis at the end of the program. In addition to that, we want the UI to fit exactly what the client wants since they’re going to be the people interacting with it. As mentioned in the “Next Steps” section, we want to take input from the user, who will be the client. Our first prototype might not do this the way they want. That’s why we will refine our product until it reaches a state that the client is satisfied with.

## Back End

### Purpose

*by Brian*

The purpose of this team is to create a working program that can take in an image of a patient with red eyes and analyze the eye and return a report on possible eye diseases present including cataracts, strabismus, and amblyopia/hyperopia/myopia. The basic logic of the program flows as such: Take in photo, isolate the eyes, isolate the pupils, detect the white dot created by the flash and the crescent( if there is one present), run through some geometric analysis and machine learning of the white dot/crescents, then finally returning a report of possible eye anomalies.

### Quarter Progress

*by Arvind*

This quarter we have made significant progress on several contour extraction algorithms that is the groundwork for the various eye disease detections. We have both Crescent detection and White Dot detection to be working as intended on our sample red eye photo. Both tests begin by receiving a image of the patient’s eye. It then runs various image processing from grayscales, thresholds, blur / erode (for smoothening), so it can obtain an array of Contours (Figure 1). From there, Crescent detection searches for the largest Contour, while the White Dot Detection (Figure 2) looks for the closest-to-center contour.

It is a good idea to make sure that these detections work consistently well on a larger sample of red eye photos. In addition, we have drawn out the algorithm that we will be using for Strabismus detection. This will be done by comparing the vectors from Pupil Center to White Dot for both eyes. The pupil class should have the necessary methods for calculating the vector magnitude and angles.

### Next Steps

*by Shannon*

Most of the groundwork for the various detections has been finished. Since we don’t yet have a solid set of red eye photos generated from the equipment that we’ll be using the code from the detection hasn’t yet been tested. Therefore, one of the next steps is to go through the various detections in the code and test them in a thorough, methodical, and documented manner. Since that will be the last step in detecting the various structures that we need for analysis the next step would be to start working on the detection algorithms including strabismus, cataracts, and myopia/hyperopia/amblyopia.

To summarize:

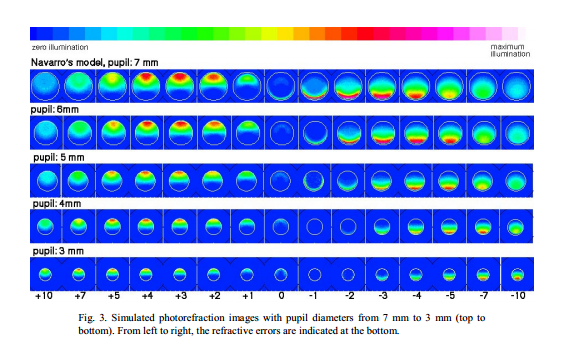
* Fix any errors currently in the codebase
* Double check to make sure all detection-related methods are written
* Methodically test detections for all parts of the code including:
  + eye detection
  + pupil detection
  + white dot detection
  + crescent detection
  + resets for all detections
* Develop, code, and test disease detection algorithms
* Integrate backend code with frontend code

### Long Term Goals

*by JT and Brian*

The long term goal is to get a rough working program which can perform at a relatively high and reasonable rate. After that is accomplished. We hope to be able to implement machine learning to the code so that it’ll be more effective and proficient.

We hope to improve our pupil detection in order to accurately get the data we need. With a better pupil detection algorithm we will be able to more accurately find the correct contour in our crescent detection, match that with the overall area of the eye and give the user a significant index output that can be read as either myopic or hyperopic, as shown in these figures.



With this done, we will implement a machine learning algorithm, in order to accurately take into account eye dilation and movement, which will make our robust software/hardware on par with current autorefractor hardware.

# Appendix

Figure 1. Backend Contour Detection

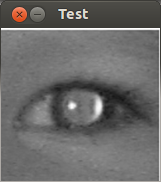
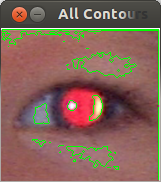


Figure 2. White Dot Detection

