CS7321 Fall 2019
Project 2
Team: Gentry Atkinson and Ajmal Hussain

	Gentry Time Committed	Ajmal Time Committed
Planning	2.5 hours	2.5 hours
Hardware Preparation	3 hours	1 hour
Video Recording	1 hours	1 hour
Coding and Debugging	2 hours	5 hours
Troubleshooting	4 hours	4 hours
Report Writing	3 hours	3 hours
Slide Preparation	2 hours	2 hours

1. Introduction:

This team has implemented an eye tracker based on Video Corneal Reflection. This method observes a distance between the pupil center and a glint reflected in the cornea from a direct light source. The error between these two displacements (in the X and Y directions) is then solved using a Least Square regression, using ground truth generated during the calibration process. After the displacement is solved during calibration the eye tracker can function smoothly as an eye tracker.

2. Background:

2.1 VCR Video Corneal Reflection is a refinement of Video Oculography which uses the displacement between a subjects pupil and a glint cast by a fixed light source to calculate the point of the subject's gaze. IR light is generally used as a fixed light source rather than a visible light source because shining a visible light directly in a participant's eye could be unnecessarily annoying.

2.2 LS Regression The Least Square Regression attempts to find a polynomial of best fit for a given sampling of data points by minimizing the sum of squared errors between the data points and the polynomial. The data points being fit by this regression are the displacements from the glint to the pupil at various X and Y displacements of the calibration points.

2.3 Spatial Accuracy The accuracy of a prediction can be broadly as the inverse average displacement between the prediction and some ground truth. In this case we are measuring the accuracy of a predicted point of gaze on a user's screen and the ground truth of the calibration points.

2.4 Spatial Precision Noiser signals are defined as being less precise. Here the standard deviation of the points constituting a single fixation is used as a measure of precision.

3. Hardware:

3.1 Webcam Our team chose to work with a Logitech C270 camera. This camera is well supported in several operating systems through its use of the UVC video over USB connection rather than proprietary drivers. This made it a good choice for use in the Linux operating system which was the team's initial choice of development environment. The camera proved easy to modify as the case was secured with phillip's head screws which were easily removed. The lense enclosure also screwed into place which allowed for simple removal and modification of the filter.

3.2 IR Filter As an IR filter the team employed developed color film. This film was cheap and easily procured, making it an ideal choice for this project. The film was trimmed to size with household scissors and placed on the lense enclosure between the lense and the Charge Coupled Device in the Logitech camera.
3.3 IR Emitter An IR emitter was assembled using a simple IR emitting LED. This LED was rated by the distributor Gikfun to emit light at the 940nm wavelength which is safely with the IR band. A power source was assembled from 4 AA batteries and wired in series with a switch and a 330ohm resistor.

3.4 Chin-Rest Since this implementation of the VCR algorithm relies on the stability of the subjects head in order to achieve reliable accuracy, it is important to employ a chin-rest while the system is calibrating and running. This team chose the expedient measure of incorporating the white, plastic box which was used to transport the webcam and IR emitter as a chin-rest. The box is approximately 5" high and was placed so that the subjects eye was 12" from the screen.

3.5 Testing Platform The code which was provided for and modified by this team was compiled and run in the Visual Studio 2019 IDE on a consumer Dell laptop running Windows 10 with i3 processor. The code was run on the .NET 4.7 framework.

4. Implementation:

This project was provided with a working framework for eye tracking which was originally authored[1] by Javier San Augustin with the IT University of Copenhaggen (javier@itu.dk) and later modified by Martin Tall (tall@stanford.edu) and Adrian Voßkühle (email unknown). A small number of changes were necessary for this code to perform sufficiently.

The CalculateDegreesLeft method defined in Interpolation.cs initially returned a random value. This method was modified so that it would take an input of pixels of error and return a value of degrees of error defined as:

$$\text{Degrees} = (180/\Pi) * 2 * \arctan(\text{error\_in\_mm} / (2 * \text{eye\_distance}))$$

This equation first converts the mm of error to radian (which is the default return of the C# arctan function) and then converts the radian to degrees.

A further change was introduced by writing the calculated error to the AverageError member of the CalibrationDataLeft object in Interpolation.cs. These changes were not made to the corresponding CalculateDegreesRight method as only monocular tracking was implemented in this project.

The CalibrationSettings.cs file was altered by setting the distanceFromScreen property to 300 rather than its initial default value of 600. We found that positioning the subject head closer to the camera allowed us to distinguish the subjects pupil and corneal glint with greater clarity.

The team was concerned that the provided code may not be calculating the Spatial Accuracy of the calibrated gaze detection correctly. We were able to confirm that the Distance method provided by Operations.cs file was correctly returning a Euclidean distance between the pupil center and the glint, defined as the square root of the differences of the positions squared. These distances were summed in the Interpolation.cs file for each point of fixation in the ground truth of the calibration data. Finally an average error was calculated as the mean of the Euclidean distances.

This gives us an error calculated as:

$\text{Error} = (\Sigma \text{sqrt}(\text{point}^2 - \text{prediction}^2))/n$ where n is the total observations.

This derivable from the accepted calculation of Spatial Accuracy:

$\text{Error} = \text{sqrt}(\Sigma((\text{point}^2-\text{prediction}^2)/n))$

Therefore no alterations were made to the platform's method of calculating Spatial Accuracy.

A further change was made to the CalibPolynomial method in Interpolation.cs. We summed the stdDeviationGazeCoordinatesLeft member over all of the Calibration Targets and then divided by the number of targets as the mean standard deviation. We print this value to the console as the Spatial Precision.

We also checked the ProcessSettings.cs file to ensure that the RemoteMonocular setting of the provided framework would call the CalibPolynomial method defined in Interpolation.cs which uses the more accurate polynomial regression to calculate the fitting of eye position to point of gaze. The other provided method CalibPupil applies a linear regression which will not produce as tight a fitting. We found that CalibPolynomial was being called so we made no changes to this file.

The final list of files that we changed is:

- Interpolation.cs
- CalibrationSettings.cs

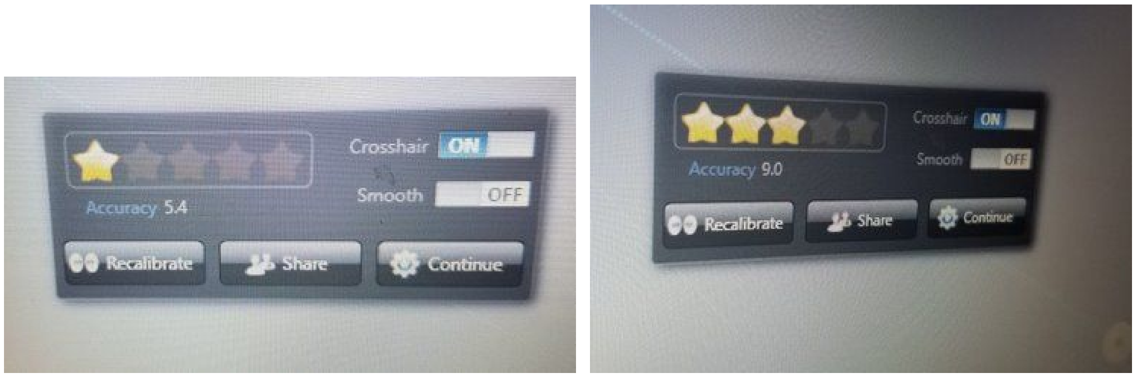
5. Challenges:

This team encountered several challenges during the execution of this project. The first that we encountered was that the Visual Studio solution provided for this project would not compile in the Linux Operating environment using the MonoDevelop IDE. While the .NET framework is available to Linux users, the UI elements employed by the original authors of this framework are not compatible with MonoDevelop at this time. The workstations in the team's lab were abandoned in favor of the Dell laptop described in Section 3.

Another issue was encountered with the team's assembled hardware. A single LED powered by approximately 13mA of current was selected as the IR emitter out of safety concerns. Although IR light is not visible, exposure to high intensity IR can potentially harm the subject's retina. This issue was compounded by the use of two layers of developed film as the IR-pass in the webcam. The extra layer appears to be unnecessary and causes the camera to produce an image which is extremely dark even when the subjects face is lit by the IR emitter. We have addressed this issue by removing a layer of filtering from the webcam and by adding a supplemental light source to illuminate the subjects face.

6. Results:

Our team was able to achieve a spatial accuracy of 5.4° with an unknown spatial precision on one run and an accuracy of 9.0° with an unknown spatial precision on another. The code was constructed to write the spatial precision to the console but we could never find the value after execution. While the results themselves are not competitive against what others have achieved using the same set-up it is encouraging to see the platform functioning at all. Further refinements to the hardware side of the platform could help with improving the usefulness of this implementation.


Crosshair ON
Smooth OFF
Accuracy 5.4

Crosshair ON
Smooth OFF
Accuracy 9.0

Figure 1: Calibration results of two runs of the platform.

References:

[1] Evaluation of a Low-Cost Open-Source Gaze Tracker; Javier San Agustin, Emilie Mollenbach, Maria Barret, Martin Tall, Dan Witzner Hansen, John Paulin Hansen; Eye Tracking Research and Applications (2010)