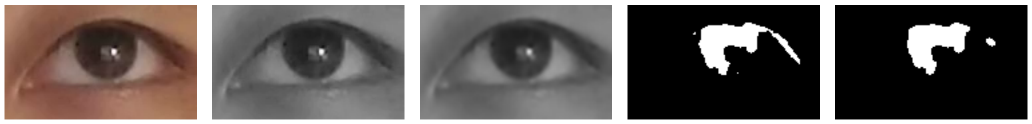
Eye Tracking with Phone Cameras

John Xue
Clemson University
Clemson, South Carolina, USA
zheyuax@g.clemson.edu



(a) Source (b) Grayscale (c) Gaussian blur (d) Thresholding (e) Opening morphology filter

Figure 1: Image preprocessing pipeline to isolate the pupil region.

ABSTRACT

We implement an inexpensive, post-processing eye-tracking system that calibrates, models, and predicts gaze location on a screen based on fixed position camera input. We also show the variety of techniques employed to achieve these results.

KEYWORDS

computer vision, eye tracking, gaze tracking

1 INTRODUCTION

Eye tracking and gaze detection has been used in a plethora of applications and systems, from detecting driver drowsiness in real-time [3] to evaluating learning and performance efficacy in education [6]. With these applications comes a wide variety of hardware configurations. As the need for reasonably-cost, accurate gaze tracking increases, limitations of expensive modern eye-tracking systems become more and more self-evident. Here we take advantage of high-resolution cameras in smartphones, hardware that is widely available, to do gaze tracking.

2 BACKGROUND

As mentioned earlier, gaze tracking takes place on many types of hardware and eye-tracking systems. However, a lot of the research uses expensive technologies, such as high-resolution images for medical purposes and infrared LED lights to illuminate and increase contrast of the pupil. Although these systems do accurately track eye movement, they do so at the cost of availability.

Of the literature that utilizes mobile devices for gaze tracking, most limit their scope to applying gaze tracking on the onboard device screen with the front-facing camera. Krafka et al. took advantage of the large audience of smartphone users by crowdsourcing and generating a large dataset of head poses and training on that dataset with multiple convolutional neural networks (CNN) for robust gaze prediction [2]. Our system takes footage from phone cameras and applies gaze prediction to an adjacent computer monitor.

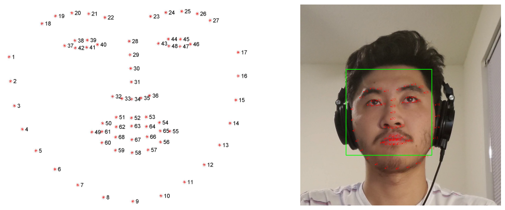


Figure 2: Obtaining facial landmarks from image with Dlib.

3 METHODOLOGY AND DESIGN

3.1 Getting Eye Region of Interest

There are many approaches to getting ROIs of facial features. However, some methods were better than other in terms of accuracy and stability. Although Haar cascades are generally performant, both the face and eye ROIs can move around drastically between frames. We eventually settled upon the Dlib implementation of Histogram of Oriented Gradients (HOG) linear classifiers and regression tree ensembles [1] for face and facial landmark detection, respectively. We find that their pretrained models gave us more stable results from frame to frame compared to Haar cascades.

This model gives us 68 facial landmarks, corresponding to the 68 landmarks used in the iBUG 300-W dataset [4] that the model trained on (see Fig. 2). We use points 37–48 to determine the left and right pupil ROIs, making sure they are the same dimensions and are anchored in the same position to help us in future image transformations.

3.2 Detecting Pupil Location

Before determining the pupil location, we must reject all images in where the subject has their eyes closed. We use the Eye Aspect Ratio (EAR) metric [5] to determine the state of the eye, rejecting the image if the EAR is below 0.25.

Otherwise, with our eye ROIs calculated, we apply preprocessing on the image (see Fig. 1). We convert the image to grayscale, blur the image with a 5×5 Gaussian kernel, isolate the darkest parts of the eye with thresholding, and finally apply morphological filters (erosion followed by dilation) with a 5×5 circular kernel. We find

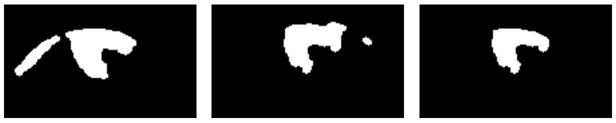

(a) Left pupil (b) Right pupil (c) Composite region

Figure 3: Combining the left and right pupil regions reduces noise and results in an overall cleaner image.

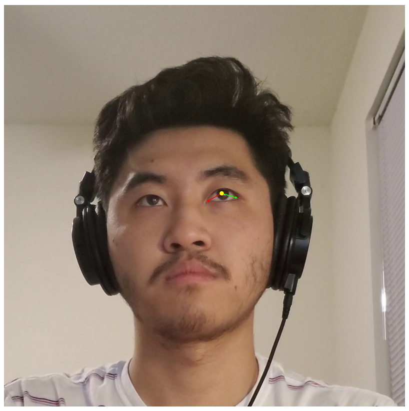


Figure 4: Rendered pupil location using our pupil detection process with respect to the corners of the eye.

this isolates most of the pupil region, while minimizing some extra noise from other dark regions of the eye, such as the eyelashes.

However, we can exploit the symmetry of our eyes to effectively keep only the pupils. The typical pair of eyes move synchronously in the same direction when gazing at a particular point, making the pupils roughly in the same location when overlayed one on top of another. Since our left and right pupil ROIs are binary images and are of the same dimensions, anchored to the same position relative to the ROI, isolating the pupils is as simple as creating a composite image of the two pupils through a pixel-wise logical-and operation (see Fig. 3).

$$\begin{aligned} C(x, y) &= \sum_x \sum_y L(x, y) \wedge R(x, y), \\ L(x, y) &= R(x, y) = \{0, 1\}, \end{aligned} \tag{1}$$

Where L and R are the left and right pupil image intensities, respectively. We then calculate the centroid of the composite image using raw moments and consider that the approximate location of the pupil.

$$\begin{aligned} \{\bar{x}, \bar{y}\} &= \left\{ \frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right\}, \\ M_{ij} &= \sum_x \sum_y x^i y^j C(x, y), \end{aligned} \tag{2}$$

However, we also need a stable point that the pupil location can be based with respect to. Fortunately we can use the existing facial landmarks we have already found (using the corner of the right eye in this case) to create a vector that is the metric for our dataset (see Fig. 4).

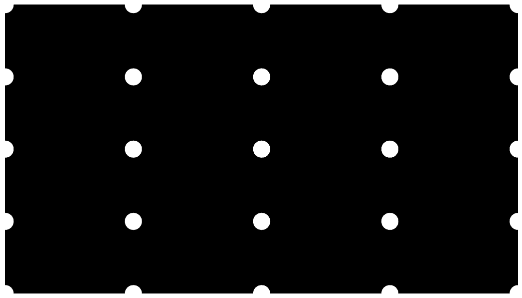


Figure 5: Image used to calibrate gaze tracking system. Points were placed at $\frac{1}{4}$, $\frac{1}{2}$, and $\frac{3}{4}$ of the screen, as well as the edges and corners.

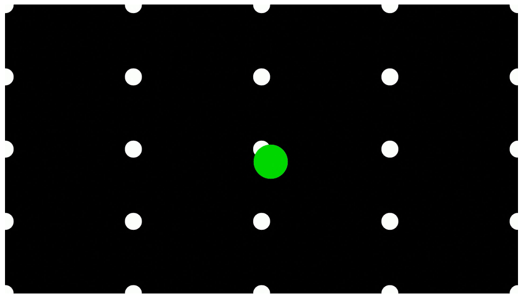


Figure 6: Gaze prediction in action. The green circle represents the predicted gaze location, with the subject accordingly looking at the center of the screen.

3.3 Creating the Dataset and Model

We create 25 video clips of a subject looking a particular calibration point (see Fig. 5). Each of these clips are tagged and associated with a pixel location on the screen ($(1, 1)$, $(1080, 1920)$, etc.). We then calculate the corner-pupil vectors for all of the clips, with the resulting pupil vector, screen vector relationship being our training data.

With this data, we use least-squares regression to train a model that contains quadratic features of the dataset ($[x^2, y^2, xy, x, y, 1]$), giving us a 2×6 matrix that is our model for predicting screen pixel location based on the pupil offset. We decided use nonlinear regression since gaze distance increases as the eyes look more towards the sides.

3.4 Extracting Gaze Location

After obtaining the model, predicting gaze location of very straightforward; simply detect the pupil location relative to the corner of the eye and run the model (see Fig. 6). However, despite the robustness of the Dlib landmark detector, the corner-pupil vector will still jitter and slip at a pixel level from frame to frame, which will affect the stability of our gaze prediction. Fortunately, we apply a Savitzky-Golay smoothing filter in both the x and y-dimensions of the input, giving us a stabler prediction path.

4 RESULTS

The gaze system generally produces acceptable gaze predictions in plausible pixel locations after calibration. Much of the variability in the input data is mitigated by smoothing filters at the expense of some accuracy in the outer extremes of the screen, where the model tended to undershoot predictions.

5 CONCLUSIONS AND FUTURE WORK

Overall, our system provides a decent gaze tracking result given the restrictions of resolution and hardware and runs in a reasonable amount of time.

One limitation of this system, however, is that the user must keep their head still in order to maintain accuracy of the model. A direction to extend this system would be to implement head pose estimation and integrate the direction and orientation the head is pointing in into fitting the model, possibly using something more sophisticated than a least-square regression, like a convolutional neural network (CNN).

Another hurdle with the usability of the system is the process of collecting calibration points. Currently, the user needs to manually record and edit videos into video clips for each calibration point. An improvement would be a way to automate the calibration process with the computer and mobile device communicating with each other.

REFERENCES

[1] Vahid Kazemi and Josephine Sullivan. 2014. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1867–1874.

[2] Kyle Krafka, Aditya Khosla, Petr Kellnhofer, Harini Kannan, Suchendra Bhandarkar, Wojciech Matusik, and Antonio Torralba. 2016. Eye tracking for everyone. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2176–2184.

[3] Sukrit Mehta, Sharad Dadhich, Sahil Gumber, and Arpita Jadhav Bhatt. 2019. Real-time driver drowsiness detection system using eye aspect ratio and eye closure ratio. In *Proceedings of international conference on sustainable computing in science, technology and management (SUSCOM), Amity University Rajasthan, Jaipur-India*.

[4] Christos Sagonas, Epameinondas Antonakos, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 2016. 300 faces in-the-wild challenge: Database and results. *Image and vision computing* 47 (2016), 3–18.

[5] Tereza Soukupova and Jan Cech. 2016. Eye blink detection using facial landmarks. In *21st computer vision winter workshop, Rimske Toplice, Slovenia*.

[6] Jerry Chih-Yuan Sun and Kelly Yi-Chuan Hsu. 2019. A smart eye-tracking feedback scaffolding approach to improving students' learning self-efficacy and performance in a C programming course. *Computers in Human Behavior* 95 (2019), 66–72.