Real-time Gaze-controlled Digital Page Turning System

Yao Chen
Electrical Engineering
Stanford Univeristy
yaochen1@stanford.edu

Zhenzhi Xia
Electrical Engineering
Stanford University
zxia3@stanford.edu

Aida Zhumabekova
SCPD
Stanford University
adazoom@stanford.edu

Abstract

In this paper we will discuss how we are using image processing to enable real time gaze tracking to use that as a signal to turn the page of a digitally uploaded note book. This project was inspired by the everyday problem many musicians face with not being able to simultaneously play an instrument and turn the pages of the music sheet. We utilize various image processing and computational imaging techniques we learned in class to showcase how they can be leveraged to solve real world problems with high accuracy and speed.

1. Introduction

For musicians, turning music sheets by hand while playing an instrument has always posed an unpleasant challenge. The musician often has to stop playing in the middle of the composition to turn the page, therefore a human page turner is commonly being used during a performance. However, during the personal practicing time musicians are left to deal with this problem on their own. A pianist for example, will usually either sacrifice several notes at the end of each page or spend considerable amount of time to practice turning the page as quickly as possible so that the music flow is not affected. Although, there is a fair amount of digital page turning applications available on the market that use bluetooth connected foot pedals/other physical attachments or apps that use music note detection and turn the pages accordingly, such devices still do not provide musicians with enough autonomy because they may require extra foot motion while a pianist needs to pedal the music or, which is the more common downfall, the note detection algorithm will turn the page way too early causing a lot of frustration for musicians. Therefore, we would like to provide a more robust page turning solution to free musicians hands and feet and let them devote wholeheartedly to playing music during their practice sessions and even performance.

Our proposed solution is a program that tracks user's pupil movement and turns pages automatically when the user signals the intent to turn the page by looking at one of the corners of the page to turn and intentionally blinking. This allows users to have a free range of hand/feet motions and to turn pages with an extraordinary ease and speed as the natural pupil movement is tremendously fast and outperforms the speed of moving a hand/foot/producing a sound cue to turn.

Since pupil tracking is a notoriously hard problem to solve as was shown in the Virtual Reality/Augmented Reality industry, our program will have to rely on a specialized hardware that is widely accepted as the go-to solution for all sorts of eye activity tracking. We will further discuss the specifics of our chosen hardware device later in this paper. We also considered using a webcam as our medium for this solution because many music players today heavily rely on the electronic music sheets and almost every electronic device has a webcam attached, however this forced us to rely on a third-party face tracking library which was not always producing robust output and therefore would tamper our performance in some cases. Again, we will discuss the specifics of the issues later in the paper.

2. Related Work

For Human-Computer-Interaction purpose, a page turning system usually consists of a real-time image acquisition system incorporated by some type of the electronic device, an eye tracking system, and a feedback system for page turning software. The core of the page turning system is to track the page turners eye position e.g. gaze, distance and its orientation in real time. Tremendous work has been done in this field [9, 3, 8, 11]. Traditional approach detects object position based on the intensity or color distribution. Eigen-face detection[11] is one of the methods to do so, and it could easily be extended into the eigen-eye detection. However, training such classifier actually requires a large amount of data due to eye shapes, light conditions, etc. To ease the computational burden of pre-processing the training data, Zhiwei Zhu et al. proposed a methodology for real-time eye tracking under various lighting conditions by combining the bright-pupil effect and the conventional

object detection techniques. Another interesting approach was used by Morimoto et al., who took advantage of pupil-corneal reflection technique for gaze tracking.

3. Method

We take a video stream from our specialized device and feed it into our pupil detection script. The image processing script is written in Python and uses a range of different image manipulation we have learned in class. First, we segment out the pupil from the rest of the eye, this allows us to obtain the gaze direction by calculating the shift between the geometric center and pupil center. We detect when the gaze is directed at either of the edges of the digital notebook that signals the page to turn. Since our notebook is hosted on a server, when the page turning signal is detected, we send the real time event to the server through web sockets. The server then translates the python response into JavaScript and sends the command to turn to the next or previous page in our HTML music sheet. We will discuss image processing steps in greater detail later in the paper.

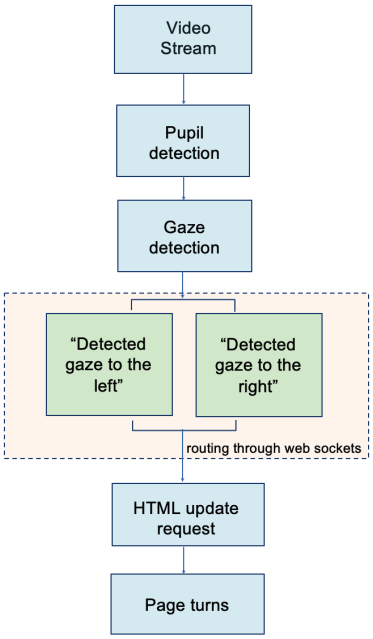

Figure 1. processing pipeline

3.1. Hardware and Data: Pupil Labs

For the real time data capture, we utilize the lightweight and unobtrusive eye tracker headset provided from Pupil Labs[2] to capture the raw video frames. The 3D printed frame holds research grade cameras that are able to record users' field of view and eye movements. The cameras are suspended right under the user's eye and can be adjusted to capture the entire eye region in high quality and resolution. For our purposes we mainly rely on monocular eye movement data due to the higher efficiency of streaming.

Figure 2. Pupil Labs eye tracker and example scene of capturing video data.

The raw image is given Figure 3. Here, the user can be seen to indicate right, center and left gazes in the order of mentioning. We continuously obtain raw images of user's eye, and then overlay the detected pupil and gaze information on the image during the display stage.

Figure 3. Example raw video frame captured by Pupil Labs Camera

3.2. Algorithm

This section is devoted to the detailed description of the pupil detection and gaze detection algorithm specially for our digital page turning system.

3.2.1 Pupil Detection

We proposed two algorithms pipelines for pupil detection. In the first algorithm, we detect the pupil based on fast radial symmetry transform (FRST). The pipeline of the first algorithm is shown as follows in Figure 4:

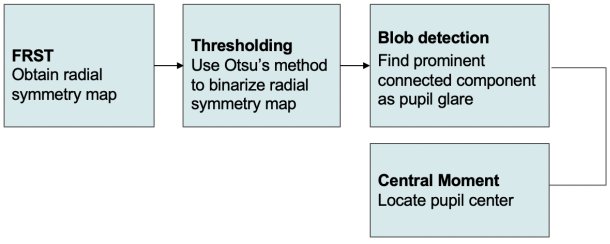

Figure 4. Pupil detection image processing pipeline 1 using FRST

For clarity, we describe the FRST algorithms as follows [7]. In FRST algorithm, pupil center estimation is obtained on the detection of the point presenting the highest radial symmetry in the image. First, we start with calculating the horizontal and vertical gradient of the image. Then we apply a empirical threshold to filter out the points with least significant gradient with small magnitude to reduce the number of pixels to be computed in the transform. The transform is calculated for a specific radius n. For each significant point in gradient map, we define the affected pixel points q as the points located at a distance n from point p and to which the gradient vector in p points at, as follows:

$$q = p + \lceil \frac{g(p)}{\|g(p)\|} * n \rfloor \tag{1}$$

Note that points with positive gradients are related to the direction from dark regions to bright regions. Therefore, the affected points mainly reside in bright regions. Since the pupil center contains a bright glare from reflection under lighting conditions, FRST could be applied to detect the bright zone in the pupil and thus estimate the pupil center. For each radius n, an orientation projection O_n and a magnitude projection image M_n are created using the affected pixels q in the following ways:

$$\begin{aligned} O_n(q) &= O_n(q) + 1 \\ M_n(q) &= M_n(q) + \|g(p)\| \end{aligned} \tag{2}$$

$O_n(q)$ is the number of affected pixels, and $M_n(q)$ gives the contribution of each affected pixels in the radius window n based on the magnitude of the gradient. We then combine both matrices and convolving them with a Gaussian smoothing mask $G \sim N(\mu, \sigma)$, to find the contribution of the radial symmetry of the radius n. α denotes the radial strictness parameter. High α values eliminate non-radially symmetric features, while choosing a low α value includes non-circular symmetry points of interest. Here we use the empirical value where $\alpha = 2$.

$$S_n = (M_n \cdot O_n^{\alpha}) * G \tag{3}$$

After obtaining the radial symmetry map from fast radial symmetry transform (FRST), we then apply Otsu's method to binarize the radial symmetry map and find the largest connected components on the radial symmetry map as the pupil region. The centroid of the largest connected component is calculated as the pupil center.

Figure 8 shows the output of different modules in the first image processing pipeline for pupil detection:

The second algorithm takes advantage of dynamic thresholding and contour recognition to locate the pupil center. It incorporates the following fives modules: bilateral filtering, dynamic thresholding, morphological transformation, pupil contour extraction and central moment calculation. Figure 6 shows the flowchart of the second image processing pipeline for pupil detection.

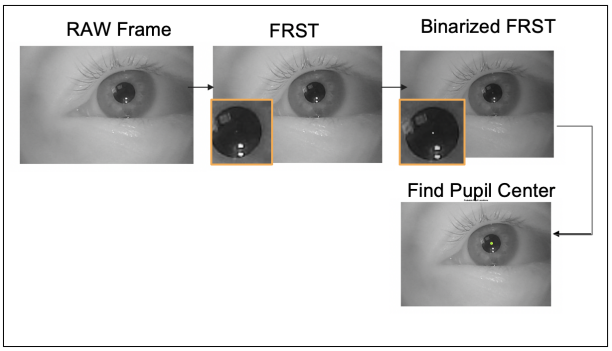

Figure 5. Results from image processing pipeline 1 using threshold and contour

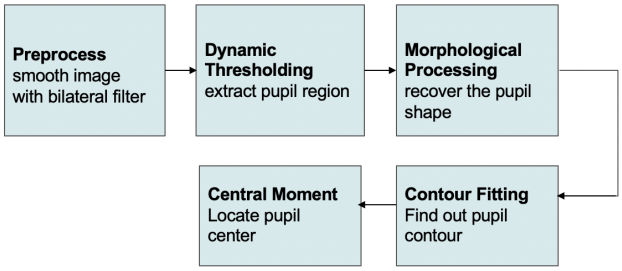

Figure 6. Pupil detection image processing pipeline 2 using threshold and contour

In order to make the proposed algorithm robust under different light condition, we introduce the dynamic threshold to extract the pupil region. Note that pupil region is the darkest region in the eye regardless of the color of iris. Even if the iris color is dark, it would not affect the pupil center localization since the iris is moving together with the pupil.

We start finding the dynamic threshold by converting the eye image from RGB to gray-scale. Next, a dynamic threshold is found by using adjusted peak and valley method [4]. Finally, a binary image is obtained with this dynamic threshold. The histogram of the gray image is calculated in Figure 7 , let g be gray levels, N(g) be number of pixels in g gray level. Adjusted peak and valley method could be described as follows.

First, we find the all the peaks on the histogram of the gray-scale image, g is a peak if

$$N(g) > N(g \pm \Delta g)), \Delta g = 1, ...255 \tag{4}$$

We store all the peak in P_g. Then find the peak with smallest level g_{min}, which is then used as the threshold to extract the pupil region. The resulting binary pupil image looks hollow and noisy, thus we apply the morphological

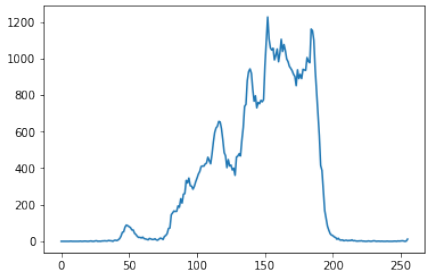

Figure 7. Example histogram of gray video frame

operations, dilation and erosion to fills the holes and remove the noise. After recovering the shape of the pupil, contours are detected from boundary of the binary eye image. Next, based on the number of elements in these contours, we choose contours that have this number higher than a certain amount. These contours contain the pupils boundary. We then calculate the central moment of the chosen contour regionμ_{pq} as follows:

$$
\begin{aligned}
M_{pq} &= \sum_{x,y \in Region} x^p \cdot y^q \\
\mu_{pq} &= \sum_{x,y \in Region} (x - \bar{x})^p \cdot (y - \bar{y})^q \\
\bar{x} &= \frac{M_{10}}{M_{00}}, \bar{y} = \frac{M_{01}}{M_{00}}
\end{aligned}
\tag{5}
$$

The central moment of pupil contour is used as the pupil center. Last but not least, we fitted an ellipse to the chosen to emphasize the detected pupil shape.

Figure 8 shows the output of different modules in the second image processing pipeline for pupil detection:

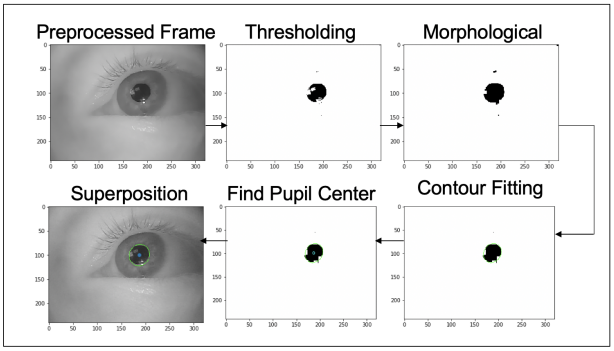

Figure 8. Results from image processing pipeline 2 using threshold and contour

3.3. Gaze Detection

As for gaze detection, we start a new algorithm pipeline by applying the morphological edge detector on a raw video frame to obtain the binary eye contour. Then, corners from binary image are extracted by using the Harris corner detector [5]. Next, corners that belong to the eye corners areas are selected based on geometric structures eye. After that, the positions of two eye corners $(x_l, y_l), (x_r, y_r)$ are detected. Note that for this algorithm to work, we need to make sure both ends of the eye appears on the image. Then we calculate the arithmetic mean based on the eye corners coordinates (O_x, O_y) to obtain position of the geometric center of the eye. Based on the geometric structure of the eye in Figure 9 , the gaze direction is calculated as follows

$$
G_r = \frac{P_x}{2 \cdot O_x} \tag{6}
$$

$$
G_s = \begin{cases} left, (G_r < 0.45) \\ right, (G_r > 0.56) \\ center, (0.45 < G_r < 0.56) \\ blink, (G_r = None) \end{cases} \tag{7}
$$

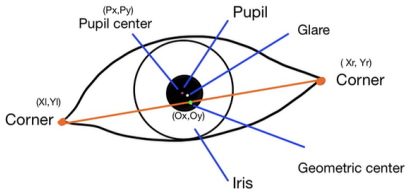

Figure 9. Geometric structure of Eye

Figure 10 shows the output of different modules in the second image processing pipeline for pupil detection:

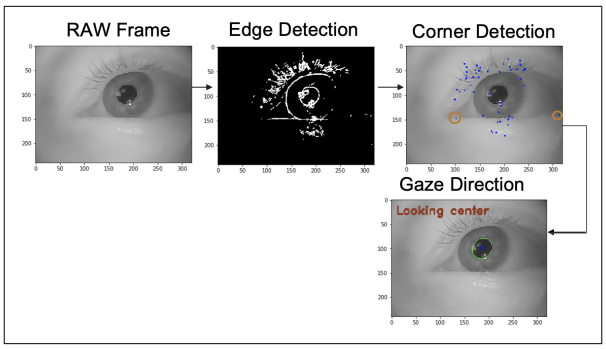

Figure 10. Results from image processing pipeline 3 for gaze detection

G_r denotes the ratio of between horizontal coordinates of the geometric center O_x and pupil center P_x. G_s is the resulting gaze state. Note that blink state is obtained here if we could not detect any pupil contours. Adding blink support would make the algorithm more robust, we will illustrate the reason in page turning system section.

3.4. Software: Page Turning System Design

First of all, we initialize the center gaze by having the user focus on the center dot of the screen. In order to easily trigger the page flipping event, our initial method was to specify four hot corners as shown in Figure 11. When the gaze is directed to one of these hot corners, the server will receive a real time event through web socket and then translates this python response into JavaScript and thus command the digital music sheet to turn to the previous or next page depending on which hot corner is detected. In addition, the page will only flip when the current gaze is different from the last preserved gaze in order to prevent the duplicated triggering.

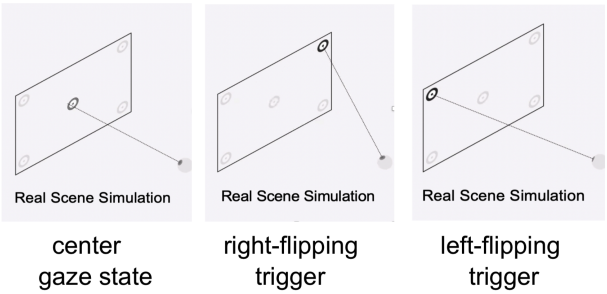

Figure 11. illustration of center gaze state and hot corners triggering

In the later development and testing stage, however, in order to make the page turning algorithm more robust, as shown in Figure 12, we extended these four hot corners method into two threshold regions along with the addition of blinking detection. Now every time before the users start using the system, they will follow the red dot first, and then two red edge lines so that their center gaze, left and right threshold gaze states will be recorded down. The computation of gaze shift is constantly running. The page turning mechanism will be only fulfilled when the gaze is directed to the space out of the red lines and a blink is detected. The blink detection is based on whether our algorithm cannot obtain the pupil contour within three frames. The two red calibration lines are able to be re-positioned by user by adjusting offset parameters. This is important in our design in the way that it avoids accidental flipping when users follows the notes on the digital sheet and blinks without intention to turn the page. We also add a four-frame wait to distinguish the page turning signaling blink from the natural blink. This waiting time proved to be the best solution that is long enough to be reliably different from the natural blink, but still fast enough for the user to turn the page at the last second, granting a seamless flow in between reading music notes and signaling the page to turn. In the end users are able to quickly turn the pages without posing a tedious long task.

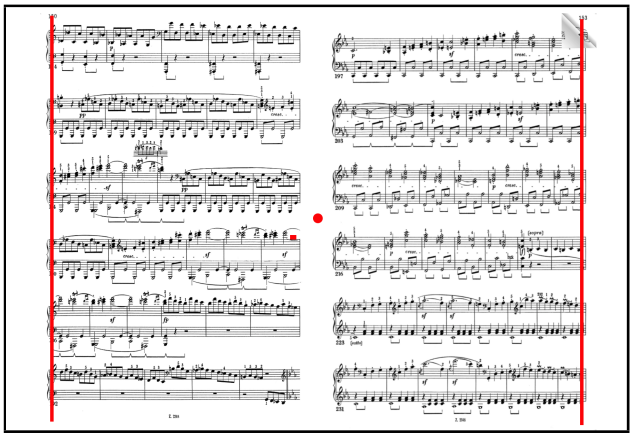

Figure 12. screenshot of the digital music sheet with calibration mode on before use

3.5. Web Camera Support

In addition to the eye tracker version, we also worked with the webcam only version to increase the accessibility of our page turning system. Instead of recording the video via Pupil Labs device, we obtain the real-time eye image from the built in webcam. This allows us to cater to a broader audience of potential users and significantly simplifies the set up process. We used Dlib Library [1] to extract the eye region. First, we find the face region with the Dlib face detection model based on Hessian of Gaussian (HoG) features and SVM. Then we locate the facial landmarks which including eyes, eyebrows, nose, ears, and mouth critical feature points and extract the eye features points.

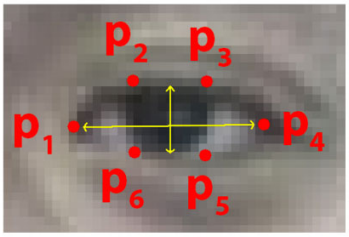

Figure 13. Eye lankmark points

From the eye feature points in Figure 13 , we could derive the eye aspect ratio (EAR) which is close related to blink state.

$$EAR = \frac{\|p2 - p6\| + \|p3 - p5\|}{2 \cdot \|p1 - p4\|} \quad (8)$$

Note that, the eye aspect ratio is approximately constant while the eye is open, but will rapidly fall to zero when a blink is taking place. Detecting blink would be significant to our page turning system due to fast movement of the eyeballs. Recall that we design the page turning system to be triggered only when it first detected the corner gaze and then followed by blinking. Since people would blink unconsciously, we ask the users to blink for a certain amount of time, approximately last for 1s for the algorithm to detect. In this way, the page turning would not be accidentally be triggered by sweeping the eyeballs to the corner nor unconsciously blinking.

The Eye landmarks also play an important role in gaze detection. On the one hand, The eye corner coordinate would be obtained from the eye features points set based on the geometric structure of the eye. We could then derive the geometric center from the corner coordinates. On the other hand, we also make use of the eye features points to calculate the boundary of eye, and crop out the eye region. The cropped eye image would then be feed into the above algorithm to detect the pupil center. With pupil center and geometric center, we could calculate the gaze direction based on Formula 7.

However, the webcam application is not robust since the third-party library sometimes could not detect face region and thus might tamper the application in some cases. Meanwhile, webcam captures eye image at a distance, thus we have less number of eye pixels and add more difficulty in reconizing the corner gaze. It requires further parameter tuning to accurately obtain the gaze direction.

4. Results and Evaluation

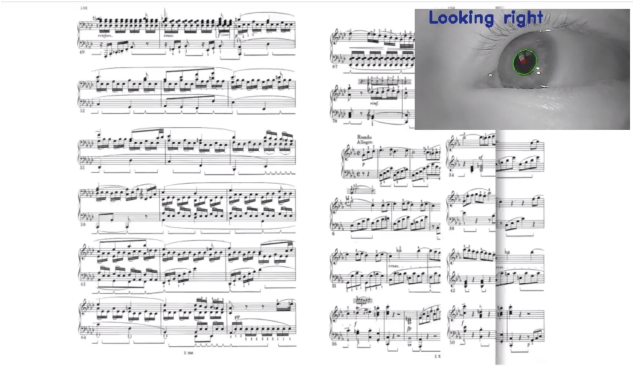

Figure 14. example frame capture of live demo.

Figure 14 shows a live demo of our final page turning system. The music score book is turning to the next page when the user looks at the right corner. In this section, we will evaluate our working page turning systems with the following aspects: run time and accuracy.

We used a subset of the labelled pupils in the wild (LPW) dataset [6, 10] to evaluate the pupil detection algorithm. LPW dataset contains 66 high-quality, high-speed eye region videos. The ground truth coordinates of the pupil is given for each frame in the video. We apply both pupil detection pipelines on 1000 frames and compare the resulting pupil center coordinates with the ground truth pupil center coordinates. The performance of the algorithm is evaluated by calculating the percentage of frames in which pupil center are precisely located with less then 5px error. Note that averaging over all frames reflects little about the algorithm quality, since a false detection would greatly increase the MSE and affect the results sharply. Thus, we only calculate the MSE of frames when pupil is precisely located. The result is given in Table 2. Since real time implementation of the page turning system is critical due to the practice consider, run time of the proposed algorithm for both pupil detection and gaze detection are given in table 1,

Method	Run Time
Pupil detection 1(FRST)	6.107s
Pupil detection 2(thresholding)	0.0235s
Gaze detection	0.0328s

Table 1. Run time evaluation of proposed algorithm pipeline

Method	Accuracy	trueMSE
Pupil detection 1(FRST)	75.34%	3.675
Pupil detection 2(thresholding)	70.09%	3.799
Gaze detection	-	-

Table 2. Accuracy evaluation of proposed algorithm pipeline

From the above tables, We could see that pupil detection algorithm 2 with thresholding method is computationally more efficient than pupil detection algorithm 1 with fast radial symmetry transform. However, with regard to accuracy and mean square error, the FRST performed slightly better than thresholding method. This might result in the fact that FRST related on the geometrical symmetrical structure, and thus are more robust under different light conditions. Both algorithm are strongly affected when there are thick eye lashes on the frames, which would result in false pupil center coordinates and greatly increase the MSE value as mentioned above.

As for gaze detection, we are unable to find an appropriate data set labelled with ground-truth gaze to evaluate our gaze detection at this time. Instead, we perform a small subjective case study to evaluate the user experience with our page turning system by measuring the percentage of ef-

fective page turning operations. The results are given in Table 4. We could see that better accuracy are obtained when detecting the right corner gaze. This might result in the relatively different movement of eye ball required for looking left and look right. The pupil center shift might be more obvious when looking right than looking left.

	look left	look right
correct detection	125	138
total try	151	150
accuracy	82.7%	91.4%

Table 3. Accuracy of the page turning system

5. Discussion and Future Work

- During our evaluation trials we noticed that users who wear glasses are not consistently receiving accurate results. This proved to be an outcome of our algorithm wrongfully detecting glares on the glasses as pupil. This problem can be easily solved by adding a color detection fragment to our pipeline that matches the original color of the detected region to the possible pupil color, simply put if the color of the detected pupil is not black it should not be classified as pupil.

- Another user specific issue that affected our algorithm quality is posed by the false detection of the pupil region, which is caused by misinterpreting dark colored lashes as a pupil. If this happens during runtime, the page can be accidentally turned when the user did not intend to do so. However, if this happens during the set up process, which happened only once in our trial runs, then the center of the eye will not be detected correctly which will produce unexpected results for turning. The problem seems to only affect users with black, well defined long lashes. As it appears to be this is a known problem in other pupil tracking techniques as well, and the only way to overcome this is by making our pupil detection algorithm more robust.

- As an extension for our project we explored using the built in web camera for video recording instead of the specialized PupilLabs hardware. For that we utilized Dlib facial recognition library that allowed us to segment out user's eyes from the rest of the background in every frame. However, using Dlib library eye tracking considerably slowed down the entire pipeline to the point where the real time page turning was not effective. Instead, we can try using OpenCV facial recognition library to specify the region of the user's face and do the eye segmentation ourselves by using template matching or possibly a training dataset which would recognize the eye region. This will allow us to identify potential bottlenecks and address them accordingly to speed up the entire process.

- Also, as we collect more reference data and obtain more usage time, we will need to address the issue of quality testing. Right now, we are using the output video from the user's session to manually evaluate when the page turning is triggered and compare it against the turn history of our algorithm. This is not a trivial task and is not scale-able. As a potential solution we can automize this process by simultaneously running the gaze detection provided by PupilLabs and using that output as the source of ground truth against our custom output. Since PupilLabs is a library relied on by a large number of other users and research it is safe to assume their gaze tracking is fairly reliable. We can overlay the graphical output we produce with their video output and compare the two pupil outlines frame by frame in an automated manner which will produce a percentage of identical frames. Ideally, we should strive for ¿99

6. External Link

Initial result with prerecorded eye movement video data: video link here

Final result with live video stream data: video link here

References

[1] *Dlib Library, http://dlib.net/.*

[2] *Pupil Labs, https://pupil-labs.com/.*

[3] R. K. Bellamy, B. E. John, J. T. Richards, C. B. Swart, J. C. Thomas, S. M. Trewin, et al. Correcting systematic calibration errors in eye tracking data, Oct. 10 2017. US Patent 9,782,069.

[4] N. H. Cuong and H. T. Hoang. Eye-gaze detection with a single webcam based on geometry features extraction. In *2010 11th International Conference on Control Automation Robotics & Vision*, pages 2507–2512. IEEE, 2010.

[5] K. G. Derpanis. The harris corner detector. *York University*, 2004.

[6] Y. S. A. B. Marc Tonsen, Xucong Zhang. *Labelled pupils in the wild (LPW) dataset*.

[7] I. Martinikorena, R. Cabeza, A. Villanueva, I. Urtasun, and A. Larumbe. Fast and robust ellipse detection algorithm for head-mounted eye tracking systems. *Machine Vision and Applications*, 29(5):845–860, 2018.

[8] C. H. Morimoto and M. R. Mimica. Eye gaze tracking techniques for interactive applications. *Computer vision and image understanding*, 98(1):4–24, 2005.

[9] P. B. Nguyen, J. Fleureau, C. Chamaret, and P. Guillotel. Method for calibration free gaze tracking using low cost camera, Oct. 23 2018. US Patent App. 14/389,783.

[10] M. Tonsen, X. Zhang, Y. Sugano, and A. Bulling. Labelled pupils in the wild: a dataset for studying pupil detection in unconstrained environments. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, pages 139–142. ACM, 2016.

[11] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.