Spring 5-24-2021

A Hybrid Gaze Pointer with Voice Control

Indhuja Ravi

A Hybrid Gaze Pointer with Voice Control

A Project Presented to

The Faculty of Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements of the Degree

Master of Science

By

Indhuja Ravi

May 2021

The Designated Project Committee Approves the Project Titled

A Hybrid Gaze Pointer with Voice Control

by

Indhuja Ravi

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

San Jose State University

May 2021

Dr. Robert Chun	Department of Computer Science
Dr. Nada Attar	Department of Computer Science
Dr. Fabio Di Troia	Department of Computer Science

ABSTRACT

Accessibility in technology has been a challenge since the beginning of the 1800s. Starting with building typewriters for the blind by Pellegrino Turri to the on-screen keyboard built by Microsoft, there have been several advancements towards assistive technologies. The basic tools necessary for anyone to operate a computer are to be able to navigate the device, input information, and perceive the output. All these three categories have been undergoing tremendous advancements over the years. Especially, with the internet boom, it has now become a necessity to point onto a computer screen. This has somewhat attracted research into this particular area. However, these advancements still have a lot of room for improvement for better accuracy and reduced latency. This project focuses on building a low-cost application to track eye gaze which in turn can be used to solve the navigation problem. The application is targeted to be helpful to people with motor disabilities caused by medical conditions such as Carpel Tunnel Syndrome, Arthritis, Parkinson's disease, tremors, fatigue, and Cerebral Palsy. It may also serve as a solution for people with amputated limbs or fingers. For others, this could end up being a solution to situational impairments or a foundation for further research. This tool aims to help users feel independent and confident while using a computer system.

Index terms – eye gaze tracking, assistive technology, image processing, camera-based eye tracker

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

Pointing devices are essential to input spatial and multi-dimensional data to computing devices. People with dexterity limitations are unable to use hand-held devices to use a computer. They might find difficulty in clicking, scrolling, and navigating. This could end up being a discouraging experience and people may tend to refrain from willfully using computer devices. Figure 1 displays some hand positions of people with motor disabilities trying to navigate a computer system [21]. In this modern age, where the internet has become essential, a problem such as this is setting people back. Their limitations, however, depend on the cause of the problem and their needs must be carefully analyzed to build an efficient pointer solution. Jacob O. Wobbrock in [21] stresses approaching the problem of designing assistive solutions with an ability-based approach. However, if accessible pointers have a steep learning curve or prove to be laborious to use, then they may set the users back. In this scenario, a user might demonstrate increased productivity while using a tool that was not designed for accessibility [24]. Therefore, it is essential to keep in mind that the ultimate goal is usability. Although we see several adaptable solutions in the market, little has been done to understand the user's need. Hence, researchers conducted experiments on younger and older adults to determine the specifics relating to the problems faced by individuals while using navigational devices to communicate with a computer [24]. It has been noticed that children, young and older adults have different patterns of using these pointing devices as mentioned in [22, 23]. Conducting experiments on separate samples based on age has proven to provide a better understanding of their needs. In the current period, computing devices refer to tablets, phones, smartwatches, and more. And these devices are also being made more accessible. Situational impairments are heavily dependent on the context. For example, a person texting while driving is partially impaired since his concentration is divided between driving and texting. Accessible solutions are sometimes built also keeping in mind situational impairments [21]. Although computing devices could refer to devices that do not have graphical user interfaces such as embedded devices, we mainly refer to computers that do in this particular report.

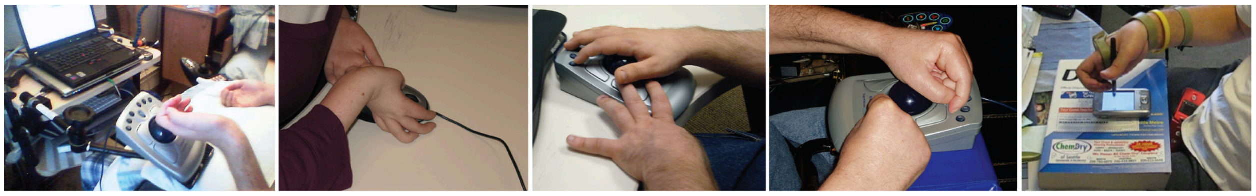


Fig 1. Navigation using various pointing devices for people with disabilities. Image adapted from [21]

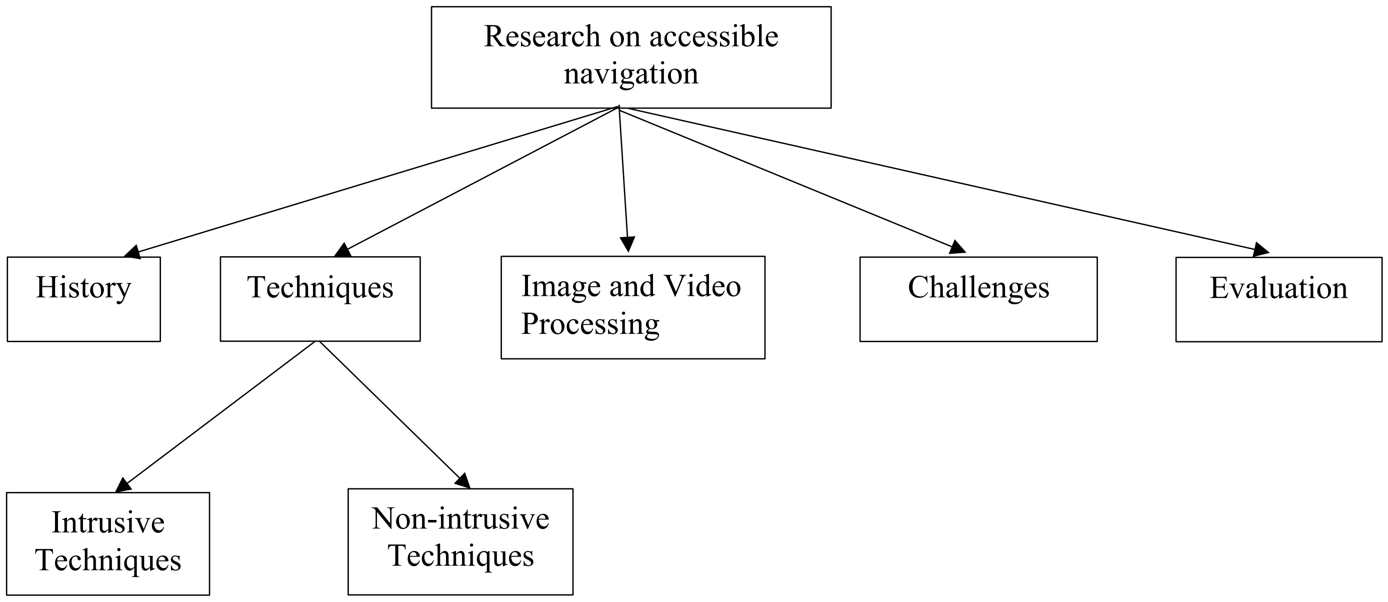


Fig 2. Organization of the Literature Review

CHAPTER 2

History

There are a wide variety of mice that are available as a solution to accessibility. Each of them was designed to solve a subset of accessibility problems. Like we saw in the introduction part of this report, each population forms a subset of people with unique requirements and several factors determine these requirements. Some of these factors could be age, gender, type of disability, and so on. After carefully testing a trackball, joystick, and touch screen as assistive options for a woman with cerebral palsy, [20] determined that the best option would be a joystick. The effectiveness and correctness of each of the options were tested using a tool called eTAO followed by questioning the subject about various aspects to learn about their preference. For several years, Apple has been a front runner in providing accessibility. Apple started to improve accessibility in Macs in April 2005. Macintosh carries a set of tools that enhance mouse functions for people with such requirements. For people who are completely unable to use a mouse, Macs have mouse-keys on the numeric pad of their keyboards that act as a navigator for the mouse pointer [15, 19]. The control panels for mice allow the user to alter tracking speed and double-click speed [19]. For people facing trouble with mouse clicks, a modified use of the trackball is recommended. A pair of switches that are hardwired to the device could also be used instead. This emulates the mouse clicks where the computer is informed that the person is using a modified version of the device for clicks, drag, and drops, and other actions [15, 20].

Previously, we learned about studies relating to devices that require the usage of hands to navigate the mouse pointer. The following discussion will be regarding previous studies on using various kinds of eye-tracking applications. Eye-tracking has gained popularity in many fields other than for accessibility reasons. The history of eye-tracking goes back to the 1800s. Although they were not tracked for accessibility reasons at that time, eye tracking was prominently used in many other fields. It has also been widely used for evaluating website design and user experience. This is one of the contributing reasons for some websites having a better flow when compared to others that are quite uncomfortable to navigate. Djamasbi in [13] explains the usage of commercially available eye trackers which capture eye movements using video-based corneal reflections. This follows the popular method of using infrared lights to create a speck on the surface of the eye. This bright speck has made it easy to track the pupil. The outputs are heat maps that describe the areas of the website that has received the most attention from the user or blind zone maps that show which areas received zero attention. This is a way of viewing web applications through the eyes of the users. The completely contrasting use of eye-tracking is to study reading patterns and speeds. Charles H Judd and Edmund Huey [9] both came up with their tools for eye tracking to analyze reading patterns and speeds. Their discoveries have been a huge contribution to the field of education and literacy. Eye-tracking has also gained popularity in the field of marketing [2]. Their main goal was to study parts of magazine advertisements that grabbed attention from users. It was used as a tool for continuously learning patterns that prove to be effective advertisements. A case study where the effectiveness of eye tracking in improving an underperforming advertisement was discussed in [12]. Apart from the above areas, gaze tracking is used in cognitive science, driver fatigue analysis [11], game theory, and medicine [2]. It is also used to analyze stress levels in patients and employees in the field of banking, IT, BPO, and accounting [4]. It is also used in assessing defect detection by factory workers and for video summarization. In contrast to [20], where a user with cerebral palsy preferred a trackball, the research in [17, 18] shows that using an eye tracker improves confidence in kids and adolescents. It encouraged them to use a computer device. This study also proves to be more accurate than [20] since the study was extensive and was conducted over 6 weeks. [17, 18] also show how subjects displayed better communication, involvement, and reduced depression among adults.

CHAPTER 3

Techniques

3.1 Gaze V. Eye tracking:

So far in this report, eye tracking and gaze tracking have been used synonymously. However, they both are not entirely the same thing. During this research, I found that this is also noticeable in other literature. The difference between the two lies mainly in the way the eye is perceived. Gaze-tracking mainly refers to tracking the eye from a remote device placed in front of a subject. In this way, the angle of the gaze directly translates into coordinates that can be made sense of in the real world. However, other devices measure the eye movement using methods that attach to the subject's head. In this case, only the eye could be tracked, and the gaze angle needs to be calculated from the eye movement by considering the angle at which the tracking device captures images or videos of the eye. This could be a useful consideration while the various methods of tracking are studied in the following sections.

3.2 Intrusive Techniques:

3.2.1 Electro-oculography:

This is a cheap but extremely invasive method to track eye movements. It involves attaching sensors around the eye area and tracking changes in the electric field while the eye moves. Although the practicality of the method for everyday use is low, it has been widely used in clinical environments since it can even record eye movements when the eyelids are shut as discussed by Chennamma in [3]. A variation of this is the infrared oculography where the reflection of infrared light from the sclera is captured and the difference between this and the actual light intensity is used to find the gaze point.

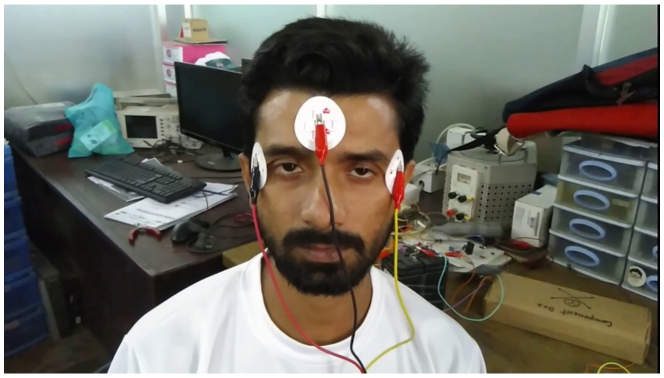


Fig 3. Illustration of Electro-oculography

3.2.2 Head-mounted eye tracker:

Recognizing the need for cheaper and robust eye trackers, a minimally invasive technique was proposed using the combination of open-hardware and open-software [8]. The literature details out the steps for a low-cost headgear construction that will assist users in capturing videos of the eye while using open-software solutions to calculate the gaze point. The accuracy of the system is up to 1 degree. As a result of the cheaper headgear, the resultant videos were of lower quality with a lot of noise. Another head mount system was proposed in [7] where two consumer-grade CCD cameras were mounted on a pair of safety glasses. While one camera captures the eye, the other captures the same scene viewed by the eye at that moment. The 640x480 image results from the two cameras are synchronized at 30hz to predict the target point. A similar structure could also be used with videos.

Fig 4. A head-mounted camera eye tracker

3.3 Non-Intrusive Techniques:

3.3.1 Area Cursor:

Area cursors are one of the simpler ways to include accessibility in pointing. Unlike point cursors, these consider a small area on the screen as a target [21]. This somewhat assists in focusing on a target since it accounts for a larger area and makes this more visible. Over the years, there have been multiple advancements even in area cursors each accounting for failures from previous versions.

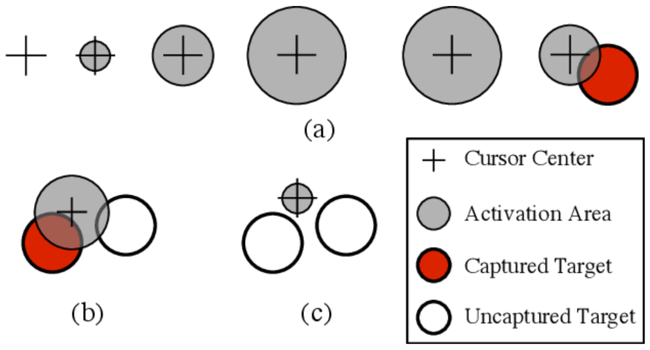


Fig 5. Bubble cursor

3.3.2 Infrared light:

Infrared glints which are corneal eye reflections of infrared light are used to track the gaze in [5]. Burton, Albert, and Flynn in [6] conducted an extensive study to compare webcam-based technology called Sticky and an infrared tracker called SMI. For the SMI, they used an SMI iViewX eye tracker. After comparing the webcam-based technology and infrared technology, it was found that the infrared technology displayed better accuracy. However, the camera option was fast and cost-effective. It also performed better when they used larger images on the screen to detect fixations. Another infrared imaging system was proposed in [7] using the dark-pupil technique. The system was built in a way such that it could be extended to the visible spectrum as well. Although it was built using the dark-pupil technique, it could be easily modified to use light-pupil techniques. The difference between the light and dark pupil techniques lies in the way the eyes are illuminated and the angles they are illuminated from. Infrared light is also used in invasive infrared oculography which is popular for Magnetic Resonance Imaging (MRI) since it can record eye movements even in the dark [3].

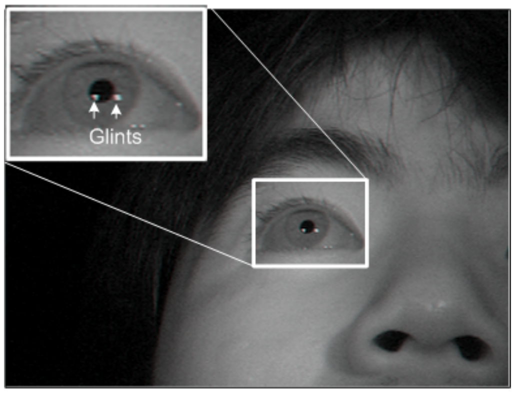


Fig 6. Corneal Glint

3.3.3 Camera-based models for eye tracking:

Eye-tracking using cameras could be both intrusive and non-intrusive and can use one or more cameras. Further, they may use visible light or infrared light to reproduce corneal glints on the eye [3]. Multiple camera models use multiple cameras to allow free head movement in a wide range and take high-resolution images within the limited range to detect the glint. A stereo head that allows head motion due to its ability to tilt, rotate and cover a wider angle has proven to be a high accuracy solution to tracking the head movement while allowing the subject to move his head [5]. A very detailed 3D model of the eye is used to track the eye. Similar to the tracking method above, another less complicated 3D model was used in [10] followed by a personal calibration step. The user is first required to look at only two points on the screen to calibrate. This happens as a step to configure the tool for a particular user. Once this step is completed, the FreeGaze system is ready to be used by that particular user. The 3D eye model comes into play since the angle of measurement is not always the same as the visual angle. These models have been useful in the calibration step to avoid this margin of error [5, 10]. Purkinje images are reflections of objects on multiple surfaces of the eye. The first image also known as glint and the fourth image on the cornea and lens respectively are popular choices for eye tracking. The pupil position along with the Purkinje images serves to detect the gaze angle [10]. Another camera-based technique was proposed using the Sticky webcam technology [6]. A Logitech Webcam Pro 9000 camera was used, and the processing of the videos was done asynchronously by the Sticky software. The system performed well with a deviation of at most 5 degrees in both axes. In most of these techniques, the eye is illuminated, and the image or video of the eye is captured and translated into dimensions in the real space. In some other methods, the center of the eye is first located and then the eye corners are found. Following this, the dimensions are found in the real space.

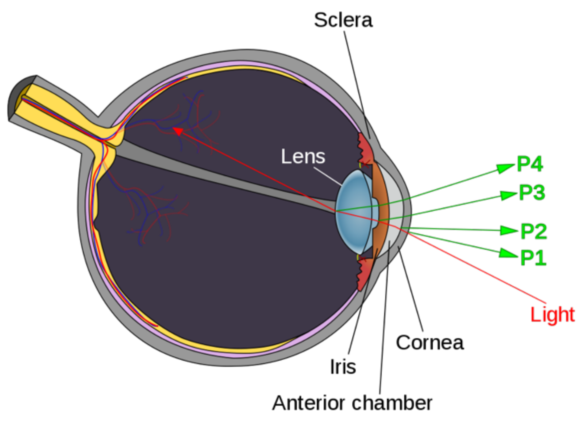


Fig 7. Purkinje images

Although these methods are built to work in real-time, they do not perform very well. Sahay and Biswas in [4] propose a pupil-center localization technique using real-time facial landmark detectors as shown in figure 8. These landmark detectors detect facial features using the face of a human as input. It has been used in a variety of machine learning problems with emotion and facial expression detection being one of the most popular applications. An artificial Neural Network-based eye-tracking system was proposed in [1]. Here, the dataset is created by taking images of the user while the user is trying to focus on a specific target visible on the computer screen. This was built in a way that could be customizable for each user. This is a non-invasive technique that does not allow the use of any headgear or chin mount. The 8x2 hidden layer network was trained for 260 epochs and performs at 15 Hz with a relatively higher degree of accuracy of about 1.5 degrees without any restrictions for free head movements.

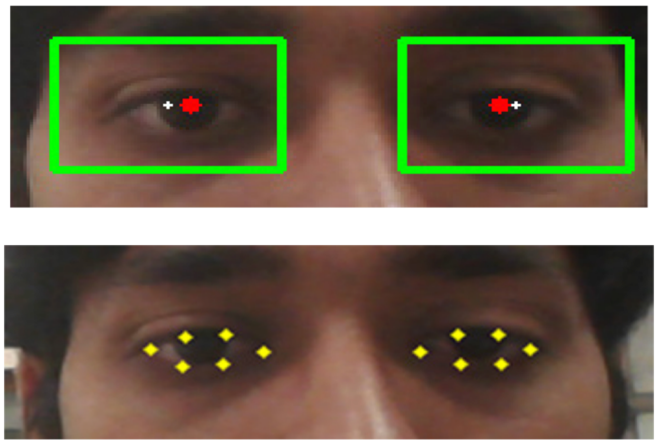


Fig 8. Eye Landmark Detectors

CHAPTER 4

Disadvantages and Challenges

One of the main challenges with gaze tracking is building non-invasive solutions that allow free head movement. In most cases, using a non-invasive technique will result in no allowance for head movement. However, using an intrusive technique such as wearing head gear might assist in tracking the eye while allowing head movements [5]. Some computers provide added support and modifiable settings to allow this. These settings are provided for users to improve accessibility without using additional tools or gadgets. However, these tend to be confusing, time-consuming, and complex for some people [24]. It might drive people away instead of attracting them to use such benefits. As an alternative, other systems were proposed where the pointing ability of a user is first assessed, and the interface is adjusted accordingly. This is so that the user does not have to go through manual setup steps. However, assessments are time-consuming and tailored specifically for a particular user. Using AUIs raises concerns with predictability, unobtrusiveness, comprehensibility, and controllability. Other popular tracking methods with head movements like the stereo method above require intense hardware setup to cope with the latency and accuracy issues. Even with simpler and straightforward solutions such as area cursors, there have been several challenges. When multiple possible targets overlap each other, it becomes challenging for the area cursor to pinpoint a target. Static area cursors lost their popularity quite quickly for this reason. Dynamic-area cursors such as the Bubble cursors [21] came with their own set of problems. Dynamic-area cursors need to take into consideration the boundaries of targets. These cursors need to be target-aware at all times. Suppose the cursor is placed at an area where multiple target boundaries are present, then it needs to decide on target selection with the information it has regarding the boundaries.

This application is aimed at building a mouse pointing solution that will initiate interaction with the computer. One of the most important challenges is finding out the gaze direction. We have seen so far that this proves to be cumbersome. Now building a tool for gaze interaction becomes even more difficult [10]. Another disadvantage with non-intrusive techniques is the additional calibration steps that may be needed at the beginning of use. This task may cause inconvenience to users as these are not required by other interaction tools such as hand-held mouse and joystick. Users may be thrown off and confused at these steps if they were not expecting them. Challenge lies in educating users and simplifying the process for them [10]. While some studies have shown that a short 2-marker system is sufficient [10], some others still prefer to use an extensive calibration step to avoid risking poor accuracy. For example, the open eyes system proposed in [8] uses a 9-marker calibration step and an additional lens to account for accuracy. While non-intrusive low-cost camera-based techniques are less invasive, they are low in accuracy and are not that adaptable when compared with head-mounted systems [7]. The same is the concern with remote video-based eye-tracking systems where recorded videos are used. While subjects try to focus their gaze on a particular target, this gaze point does not always fall within the boundaries of the specific target. There might be situations where it lies close to the actual target but not within the boundaries of it. Burton, Albert, and Flynn in [6] suggest conducting experiments on considering a border of specific width around each target. Gaze falling on these defined borders will then be considered as gaze falling on the object that this border belongs to. This is similar to a challenge faced by area cursors which is the overlapping boundary problem [21]. For a densely filled screen with lots of targets, borders might overlap making it harder to find the target the user is aiming at. And so, additional challenges lie in finding a desirable border width and dwell time. For the Sticky software solution, the researchers faced a lot of trouble during the testing phase.

Session uploads to the sticky server continuously failed because it could not handle the larger amount of test samples and the longer duration of testing for each sample [7]. For these reasons, testing was delayed, and the test sample went through a major reduction. Only two-thirds of the elected participants could take part in the test. With single camera detection techniques, the field of view is very limited. Hence, capturing high-resolution images becomes a challenge [3]. This could be improved by the addition of external sources of light but reproducing the setup for each use becomes difficult. In the case where an artificial Artificial Neural Network was built in [1], there is an additional step to calculate the offset. This is similar to the calibration steps for building eye trackers. But here the offset is calculated at the beginning of the process where the user is required to trace the movements of the cursor with his eye. This offset is the difference between the gaze point calculated by the neural network and the actual position of the cursor. This adds to the latency problem since the gaze point found through the trained model needs to go through the offset step.

Although we have seen several techniques and existing solutions for gaze tracking, the task is still complex. Integrating them into computer systems depends on several factors such as availability, latency, compatibility, and cost (especially in the case of hardware options) [8]. Software specifications may be a huge deciding factor for the integration step. Camera-based eye tracking has gained popularity and scientists have come up with multiple solutions. Yet, infrared still has proven to perform better especially when the eye is tracked for small points on the screen. Also, head movements affect infrared tracking very little when compared to camera-based tracking. However, for scientific and research purposes, significant funding must be dedicated to the equipment set up in a lab [6]. Hence, setting up a large-scale infrared eye-tracking system presents financial barriers. Using infrared lighting in an outdoor environment or a brightly lit area becomes challenging. This is due to ambient infrared illumination [7].

Some of the methods that use infrared to detect the gaze rely on Purkinje images of the cornea and lens. However, usage of contact lens could even make it impossible to detect the two images and the line of centroid [10]. While using infrared oculography, a disadvantage apart from it being invasive is the effect of external light on the results. Also, it can measure eye movements in the horizontal and vertical axes only up to a certain degree [3]. Although using a commodity camera-based tracking system makes it affordable, challenges once again lie in the setup step. The accuracy of the system depends on the position of the camera setup concerning the user [6]. This may also require modifications for each user. A user unaware of the impact this has may end up in frustration even if the camera-based tracking system displayed better accuracy during the trials. Although we have several solutions that use external devices or additional support for eye-tracking, they are at most times a very expensive option [8]. Although hand-held, high computational power computers, and high-resolution digital cameras improve accuracy, they cost a fortune. Several external factors such as light, resolution, angle, and marker properties determine the results. Hence, reproducing a particular setup exactly for research purposes might be difficult.

CHAPTER 5

Image and Video Processing

5.1 Open Eyes: Computer Vision Hardware Abstraction Layer:

A software package called the cvHAL was built to allow developers to invest their time and work on building computer vision algorithms rather than worrying about compatibility issues that arise due to camera devices. Some of the functions provided by cvHAL include multiple camera syncing, server-side video pre-processing, and color format transformations (BW, Greyscale, RGB) [8]. Although it is possible to perform the above by using simple computer vision algorithms or with existing software layers, cvHAL boasts of its ability to provide an abstraction for camera devices that act as smart cameras by connecting to wireless networked computers [8].

5.2 Starburst:

As a result of poor performance in [8], they recommend following a combination of model-based and feature-based image processing techniques to deal with the noise. The steps involved in the process include i) Noise reduction, ii) Corneal reflection detection, iii) Feature detection, iv) Ellipse fitting, and v) Calibration. To eliminate noise, a 5 x 5 Gaussian filter and normalization factors are used [7, 8]. For step ii), the corneal reflections are detected using an adaptive brightness thresholding technique followed by linear interpolation for its removal. For feature detection in step (iii), a feature-based approach is used to detect the edge points. Hence, the combination of model-based and feature-based approaches are used in estimating the gaze point. The ellipse fitting for step iv) is performed using RANSAC which stands for Random Sample Consensus. It is a technique to fit feature points in the best possible way iteratively while considering outliers. Finally, to determine the gaze point, the popular calibration step is performed. A polynomial mapping function determines the relation between the user's view angles and finally determines the gaze point [8].

5.3 Click-alternatives:

Gaze pointing is somewhat straightforward. However, this does make the application completely accessible. Since inputs to a GUI heavily depend on typical mouse actions such as the various kinds of clicks, this also needs to be substituted to make the application accessible. An option to do this would be building a gaze-only clickable solution [16]. However, there are many challenges associated with this mainly because of the physiological limitations of the eye. Multiple confirm and Actiglaze are some of the popular gaze-only click solutions available [16].

5.4 Landmark Detection:

Landmark detection refers to extracting facial features by using pre-defined facial points and comparing them to the input image or video. For example, the detection of facial expression by analyzing the position of eyebrows. In this kind of detection, the input images need to be standardized if they were taken from different sources. Converting images to BW and using histogram normalization might help improve the accuracy. A Haar-cascade classifier was trained to detect the eye region. After the eye region was detected, the pupil-center was found with difficulty since performing this detection in real-time is a challenge using low-resolution cameras [4].

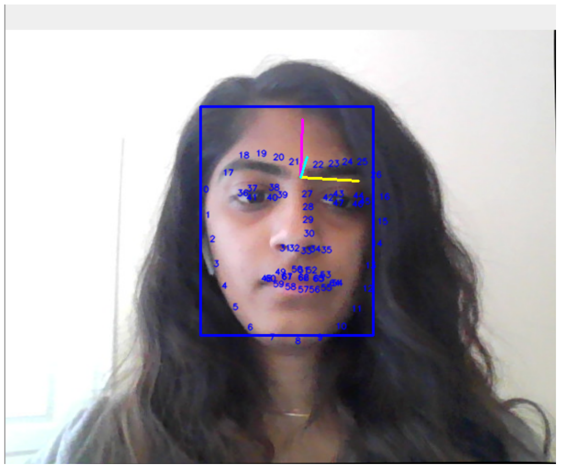


Fig 9. Facial Landmark Detectors

CHAPTER 6

Evaluation Metrics

6.1 Summary of popular evaluation methods

One of the main challenges with detecting the gaze point was the calibration step. However, a short study conducted in [10] showed no improvement in accuracy when additional calibration steps were done. Hence, the study concluded that having at most two calibration markers is sufficient and will result in the same accuracy as having multiple markers to calibrate. Considering the 3D space where this modeling was performed, there was a notable difference in accuracy between the x and y axes with the x-axis displaying higher accuracy. Although some studies show this, some still prefer to use multiple markers on the screen [6]. The markers are usually placed in cells of fixed grid sizes, most popularly in 3x3 grids. They may also use markers of different sizes and experiment with the time duration each marker is displayed on the screen. One of the important analyses done in this project is concerning the number of calibration pointers used to personalize the model. Using AUIs stresses the tradeoffs between usability and the learning curve [24]. An important step while building a gaze pointer solution is studying the eye movements. It has also proven to be useful to study relative eye patterns of multiple people than studying individual eye patterns. Gaze maps and heat maps are popular choices for this study. However, gaze maps are clumsy and confusing when it plots the eye patterns of multiple people [13]. Heat maps on the other hand are useful for studying eye patterns of multiple people as well as studying intensities. In addition to gaze pointing, this could be useful to build gaze-only clickable solutions. After studying and building an eye-gaze tracker, it must be integrated into a computer system because it is not just enough to track the eye, but this must be integrated with the cursor. With the vast number of operating systems available, each of them has different requirements. Building a system that proves to be compatible with the majority of operating systems is the next challenge [8]. Hence, there is much work to be done following the construction of an eye gaze tracker.

6.2 Loss Evaluation:

In several previous works discussed above; error analysis is done in terms of the gaze angle which is in degrees. However, the main goal in this project is to project the gaze point of regard on a 2-dimensional screen. Hence, the loss and accuracy analysis are done in terms of pixel values. The loss calculation methods are different for the gaze tracking and voice control modules. For voice control classification problem, the loss would be a relationship between incorrectly classified commands and a total number of input test commands like below:

$$\boldsymbol{loss} = \frac{Number\ of\ \boldsymbol{Incorrect}\ Classifications}{Total\ Number\ of\ \boldsymbol{Classifications}}$$

For the gaze prediction problem, the loss is calculated as the mean of the absolute difference in the pixel value of the input point and predicted point.

$$n = Total\ Number\ of\ Predictions$$

$$mean\ loss = \frac{\sum_0^n |\ Actual\ pixel\ location - Predicted\ pixel\ location\ |}{Total\ Number\ of\ Predictions}$$

The above loss calculation is done separately for the x and y axes which will reveal some more useful information on the gaze prediction loss.

6.3 Heat Maps:

Heat maps are another way to visualize data. For eye and gaze tracking, heatmaps have been a popular data visualization tool since they simplify visualizing complex statistical data collected during the tracking process. In this project, heatmaps are used to describe the magnitude of loss at different parts of the screen. The loss for each point is plotted using python’s seaborn package and then interpolated using SciPy’s interpolate module. Since the total number of pixel points will be extremely large when compared to the sample test points, interpolation is used to avoid confetti heat maps. The loss is plotted separately for the x and y axes for identifying if there is a dependency between the amount of loss and the axis. The colormap shown in the below figure will be used to plot the heatmap.

```
sns.color_palette("flare", as_cmap=True)
```

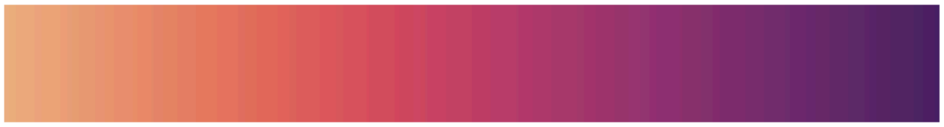


Fig 10. Seaborn color palette used for heatmaps

CHAPTER 7

Hypothesis

At the time when eye or gaze tracking was new, it was predominantly in the research phase and was undergoing clinical trials in controlled environments for several other applications apart from accessibility improvement. They were majorly built for diagnostic purposes. Although some of these solutions were also proposed for computer users to improve accessibility, they were still restricted in terms of usability and applicability. Some of the reasons for this were unmanageable size, intrusive nature, restriction of free head movement, latency, and many more. This field proves to be challenging because of the setup costs, longer periods of testing, need for continuous improvement, and the existence of several equivalent products from competitors. With this research project, I aim to build an eye-gaze tracker with the assistance of voice inputs to perform clickable actions. A simple camera and microphone would be used to collect inputs from the user. The choice of input devices could be a laptop's defaults or devices connected to the computer using USB receivers. Each video frame will be processed individually to predict the gaze point of regard on the screen at that instant. The application also continuously listens to audio inputs from the user which will be used to perform mouse clicks. There are two main modules in this project which are the gaze tracker and the voice control. Gaze tracking will be a prediction problem whereas voice control will be a classification problem. The final application will support the user to navigate a computer completely hands-free. The scope of this project is also extended to use voice control to perform corrections on the screen while the eye tracker is in progress. To incorporate this correction, we need to stop the pointer from moving with the gaze and take voice inputs to correct its position. Therefore, there will be dedicated voice commands to pause the eye tracker, correct the cursor position and proceed with the tracking or other clickable actions through further voice commands. Features that will make this application attractive and unique are:

1. Cost-effectiveness
2. Voice control
3. Free head movement
4. Real-time camera-based eye-tracking

Tobii, GazePoint, and SensoMotoric Instruments (SMI acquired by Apple) are some companies that have successfully launched their non-invasive eye-trackers in the industry. However, they rely heavily on infrared tracking and cost between $150 and $20,000. They also do not provide any facility to perform mouse clicks without the use of hand motions. Hence, they fail to make the navigation experience truly hands-free.

CHAPTER 8

Datasets

8.1 Datasets for Gaze Tracking:

A. GazeCapture:

GazeCapture is the very first large-scale dataset crowd-sourced for eye-tracking. It contains data collected from over 1450 participants and contains about 2.5M frames. Their data was collected both continuously and while participants were looking at specific targets. The data was also recorded under a wide range of illumination conditions to help build robust models that are practical for the real world as shown in the figure below. GazeCapture gained its fame from its use in iTracker which is a real-time eye tracker built for mobile devices and tablets using CNN. They have also open-sourced this pre-trained model. However, they are targeted for processing videos collected from mobile phones and tablets. Hence, their models are not re-engineered in this project. However, it is useful to know that they were able to achieve an accuracy of about 1.3cm and 2.1cm with the use of additional calibration steps in mobile phones and tablets respectively. Calibration steps similar to the ones used in the above model are implemented for the hybrid gaze tracker constructed in this project.

Fig 11. Some sample frames displaying various illumination conditions, head positions, and backgrounds [25]

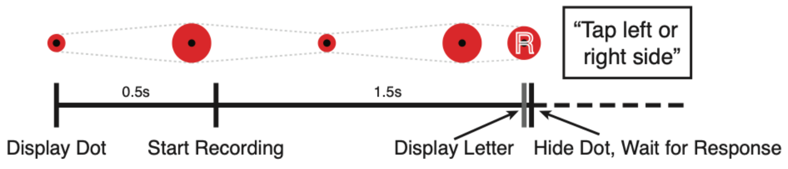


Fig 12. Timeline from start to end of a single data collection step

B. MPIIFaceGaze:

The MPIIFaceGaze dataset is a large-scale dataset that is based on the original MPIIGaze dataset. The FaceGaze comes with facial feature annotations, unlike the original dataset. It consists of frames collected from 15 participants over a few days [26]. This way, they were able to capture the same participants in different illumination conditions. For each person, the data is collected using the same machine even if it is collected over a few days. The calibration details such as the screen size are provided separately for each user. The frames and the corresponding pixel coordinates are stored. These pixel values can be normalized to be used consistently across all participants using laptops and computers with different dimensions. The facial feature annotations for the 4 eye corners and 2 mouth corners are stored for each frame. The actual eye gaze is recorded only for one of the two eyes and the dataset contains information on whichever eye was used for each of the frames. The 3D head poses in the camera coordinate system are recorded, unlike the target which is in the monitor's pixel coordinate system.

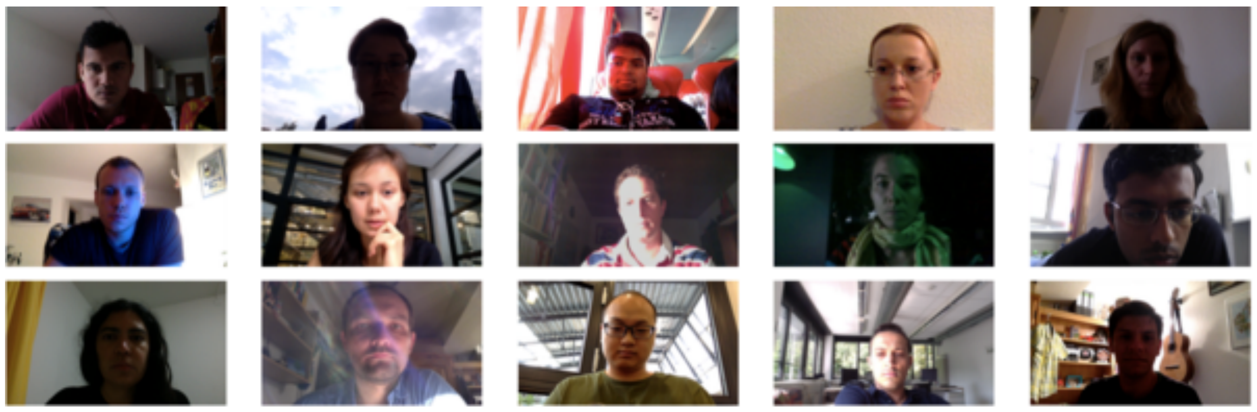


Fig 13. Samples from MPIIGaze with various illumination conditions

8.2 Voice-control dataset:

The commands used in this project include left-click, right-click, double-click, pause, play, scroll-up, and scroll-down. These specific voice recordings were not available in already collected datasets. Therefore, voice recordings were collected specifically for this project from 3 subjects: 2 female subject and 1 male subject. For each subject, 10 recordings of each command were collected. Hence, for each command, we have 30 recordings. The collected dataset consists of 210 Apple MPEG-4 audio files. Each recording lasts from 1 to 5 seconds. The voice control module built in this project is therefore a person-dependent classification model. During the progression of this project, the scope was extended to include voice commands to enhance the gaze tracker. Commands like up, down, left and right will be included in this dataset to adjust the cursor when the gaze tracker is not accurate enough. The “pause” and “play” commands were added for this purpose as well.

CHAPTER 9

Gaze Estimation

9.1 Corneal reflection-based estimation:

A popular choice of gaze estimation is using a light placed in front of the face to detect corneal reflections. The quality of the estimation will depend heavily on these external lighting conditions. One of the main goals of this project is to make gaze tracking affordable and to avoid the use of additional hardware. Since corneal reflections vary based on the lighting conditions, we do not perform this sort of estimation in this project.

9.2 Shape and appearance-based estimation

Shape-based methods estimate gaze direction using the shape and geometry of the eye. For example, in the MPIIFaceGaze dataset [26], we have the eye corners and mouth corners annotated. This can be used to determine the gaze direction. The disadvantage of using this method is the impact of low-quality frames on the accuracy of gaze direction. Appearance-based methods read video frames of the eye to predict gaze direction. They work well with low-quality images but also require personal calibration steps for each user. Appearance-based estimation is used in this project to find the point of regard. In addition to using the eye patch, head position is also used in the estimation process.

CHAPTER 10

Data Preprocessing

10.1 Pygaze:

PyGaze is an open-source python package that provides ways to perform python experiments for eye tracking [27]. It is also compatible with other eye trackers such as Tobii, EyeLink, and SMI. The scripts that are written using PyGaze are extremely portable and can run on any OS for any type of eye tracker. PyGaze was mainly targeted for effective scientific experimentation. While Graphical Experiment Builders (GEBs) are popular with productivity and efficiency, they often lack in terms of customization. GEBs does not provide a lot of flexibility as compared to building experiments by scripting. This is where PyGaze takes the upper hand since it is relatively easy to perform experiments and it allows for flexibility at the same time. Another advantage of PyGaze is that it can also be used along with GEBs to stretch the extent of experiments that can be performed.

Edwin Dalmaijer introduced a new module to the existing PyGaze library for webcam-based eye tracking. This software works on the basic idea that the pupil is one of the darkest regions of the face. However, we have the control of defining what dark is. The software inputs a threshold value and considers any pixel with a value below this threshold as dark. The darkness of the pupil depends on ambient lighting. Hence, to find the most common pixel value of the pupil, we shall run relevant experiments in determining this value by using a larger sample and finding the values with the highest frequency. Another way to perform this experiment would be to use a calibration step at each time a user chooses to use the gaze pointer. During this experiment, we read the user's face at that instant with the ambient lighting and fix the threshold for the pupil. This is achieved by first detecting the face and then finding the location of the pupil. The software also requires the user to select the location of his/her pupil for the very first time. Fortunately, this is required by the software only once. This is included as part of the custom calibration step. A face detection algorithm could be used for this as well. However, it would be an unnecessary use of resources since this is a one-time step. Also, the accuracy would not be close to the users selecting the location of the pupil themselves.

PyGaze is a popular package in python for forced retinal locations (FRL), gaze-contingent cursors, and areas of interest (AOI). It also provides a communication pipeline to provide inputs to the cursor. Since this project also involves taking inputs from the user through voice commands, PyGaze proves useful to communicate these inputs to the device.

PyGaze is a popular choice because it is based on python and allows developers to enhance its capabilities by adding user-defined modules. One of the goals of this project is to make eye trackers more affordable and simpler to use. Other strong contenders for camera-based tracking were Ogama (OpenGazeAndMouseAnalyzer) and OpenEyes. Both of them do not have any support available right now. OpenEyes works on MATLAB and doesn't have provisions for data analysis. Whereas, PyGaze currently has support and is based on python. It is modular and allows for a lot of customization if one has working knowledge of python.

Although the PyGaze web tracker provides customization, its frame processing speed is very low. It affects user experience when used for real-time gaze prediction. However, for applications that do not need synchronous outputs, the thread-based frame processing of PyGaze can be utilized. Since this only detects the location of the pupil within the cropped eye patch, further transformations are required to convert this to screen coordinates. The steps involved include,

1. Identify face and crop face
2. Identify eyes and crop left and right eye using Pygaze

3. Locate the pupil and save coordinates
4. Translate to screen coordinates
5. Move the pointer to the location

All these steps need to be executed for each frame in real-time and the time delay makes the process tedious. The cropped eye patches are also of different sizes depending on the position of the user's face concerning the camera. The closer a person is to the camera, the larger the eye patch. Due to the existence of several variables that affect the result of PyGaze tracking, purely computer vision-based predictions were eliminated. The eye patches and head detections are now used with the machine learning models discussed in the next chapter to improve the accuracy and speed of predictions.

10.2 Image data Normalization:

The frames from the MPIIFaceGaze and GazeCapture datasets contain data collected and stored separately for each person. The MPIIFaceGaze contains data collected from mobile phones and tablets. Different cameras are used to collect this data within and across datasets. To use both these datasets to train the same models, the image data needs to be normalized. The frames are loaded using OpenCV's imread function. Since early developers at OpenCV selected the BGR format for image processing, we convert all the loaded images to RGB. The rotation matrix is found using OpenCV's Rodrigues function. The 3D position of the face is also used to find more information about the gaze angle. Structure from Motion (Sfm) is a popular method to transform 2D models to 3D [28]. The 3D landmarks of the face are obtained by performing matrix multiplication of the rotation matrix and the pre-recorded face model's 3D coordinates. OpenCV's warpPerspective function is used to perform a normalization perspective transformation to the image. This transformation is applied to all the pixels in the input image to map to the transformation matrix. Further visualization steps such as color to grayscale conversion and OpenCV's arrowedLine for displaying gaze angle were added for testing purposes and they do not enhance the input image in any way.

10.3 Voice data preprocessing:

The individual voice commands are sampled at a frequency rate of 22 kHz. These raw audio files are converted to spectrograms using Short-term Fourier Transform. The loading and transformations are performed using the Librosa python library. Figures 14 and 15 are the wave plots of the amplitude in the time domain for some sample commands from the dataset.

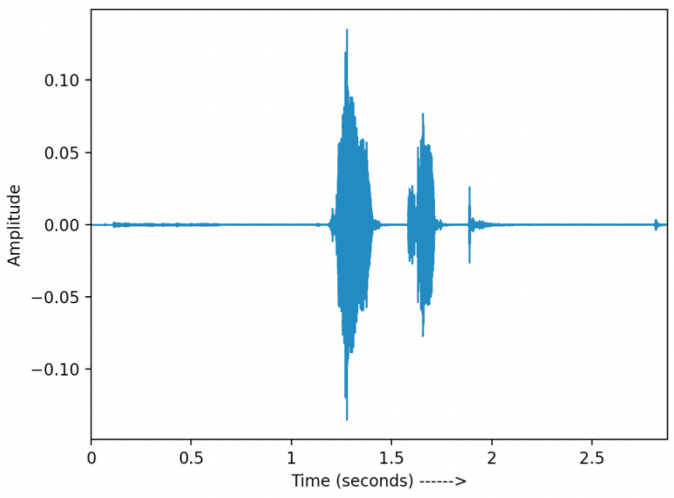


Fig 14. Amplitude plot for Right-click sample

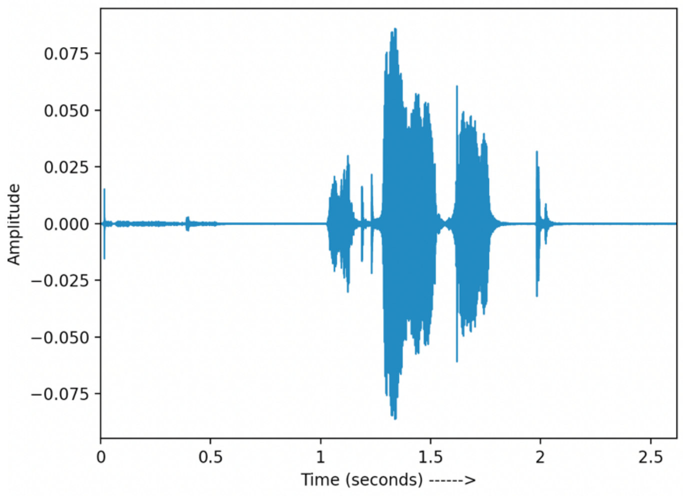


Fig 15. Amplitude plot for Scroll-up sample

The amplitude of the samples plotted above only gives us information about the “loudness” of each sample. It would be useful to decompose these signal samples into their corresponding frequencies. Fourier transforms can be used to achieve this. It converts a signal to its frequencies and also provides the magnitude of each frequency. In speech recognition, we also need the time at which each word is spoken along with the frequency information. A spectrogram is a visual representation of these frequencies as it varies with time. We use Short-time Fourier Transform (STFT) to generate spectrograms. The x and y axes in the spectrogram denote the time and frequency while the colors denote the magnitude of that frequency.

To perform spectro-analysis, STFT uses a wavelet to perform convolution on segments of the signal in the amplitude domain instead of the whole wave. For example, our input signal is in the time domain. We take a portion of this and perform Fast Fourier Transform (FFT). The result from FFT is in the frequency domain. This is rotated and colored using wavelet convolution to obtain the result in the frequency-time domain. The amplitude or magnitude is represented with color. Hence, we have converted our input from the time domain to the time-frequency domain. This procedure is repeated for all the signals and different time windows are used to analyze the various time segments. With most of the voice commands being recorded under 5 seconds, the time windows are very narrow. This narrow time window helps with getting high temporal precision. However, this also means that we have very few data points within that time window Fewer data points result in lowered spectral resolution. The below figures denote the spectrograms generated for the right-click and scroll-up sample commands.

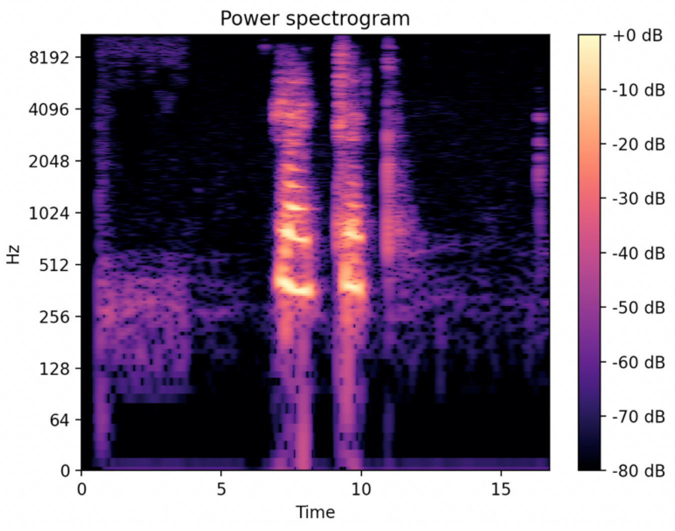


Fig 16. Spectrogram for Right-click sample

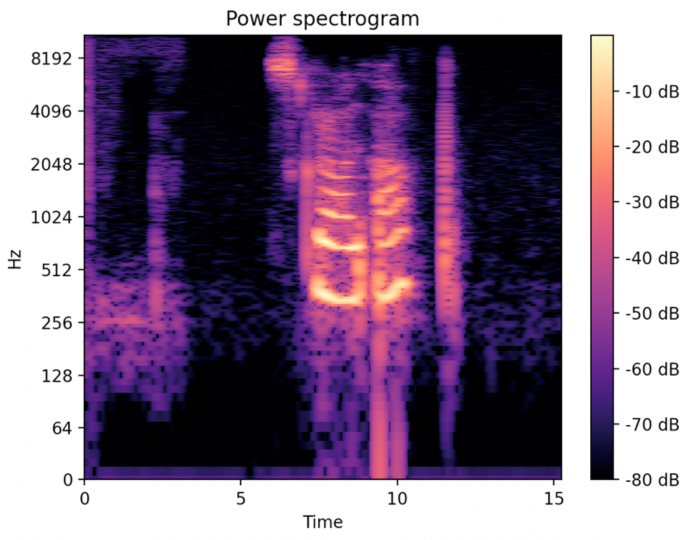


Fig 17. Spectrogram for Scroll-up sample

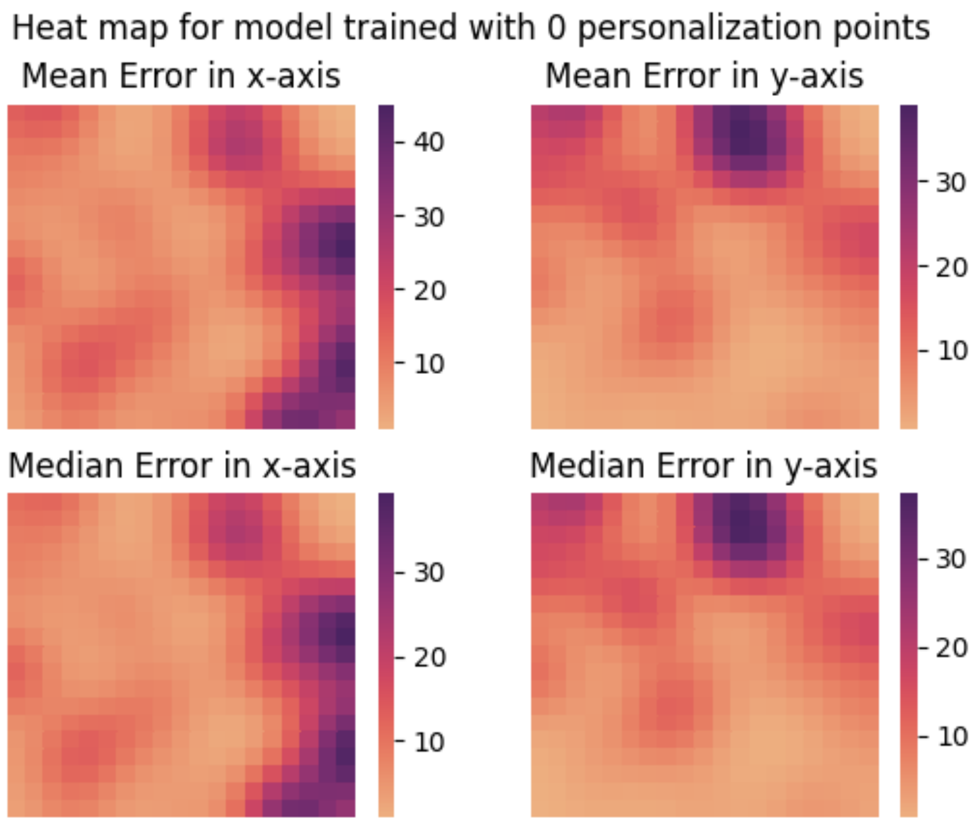
CHAPTER 11

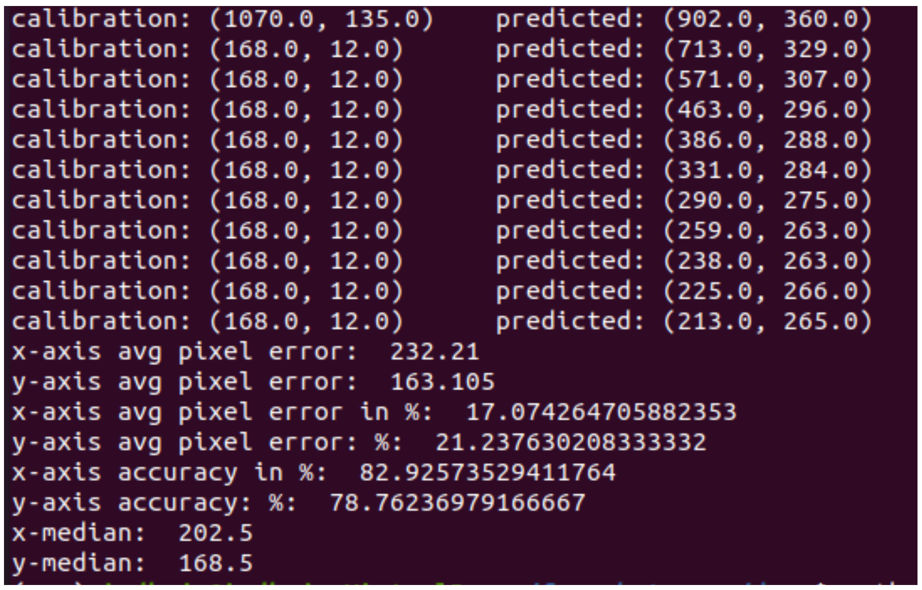
Model Training and Results

11.1 DT-ED:

Facial features for each person are very unique. The datasets used in this project contain a large collection of subjects captured under different illumination conditions and head poses. However, training a regression model to predict gaze points for a new face is challenging. Transforming encoder-decoder architectures have gained popularity for improving the training to map inputs and latent features to labels [29]. In this project, our inputs include frames from videos and outputs are the gaze points. These outputs are in the form of x and y coordinates. We also use the face and head angle position along with eye coordinates to train a disentangling encoder-decoder architecture.

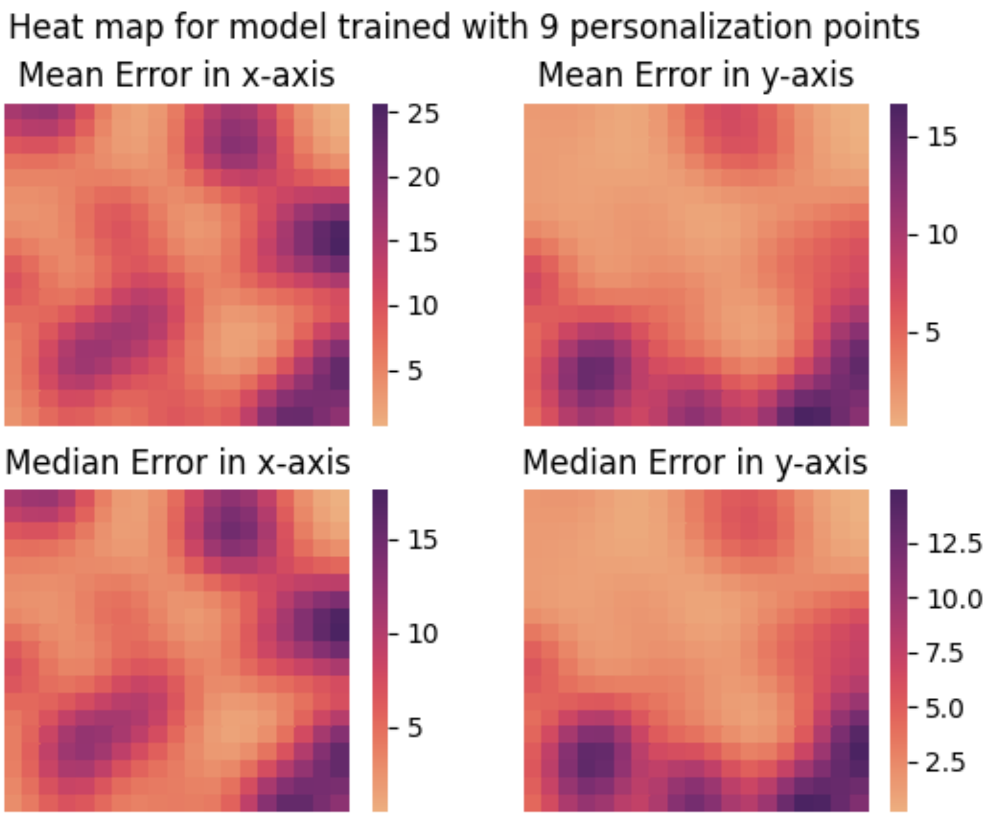
A similar architecture in [29] has been shown to work well with few-shot learning steps of 9 calibration points. In this project, we experiment with 0, 9, 16, 32, and 64 personal calibration points. To generalize the model better for new people without overfitting while using fewer calibration points, we use the MAML algorithm. After training all the models for 1000 steps with the different set of calibration points, a test dataset is created with 40 random points on the screen. The user is required to look at that point and click the key mentioned on the screen. If an incorrect key is pressed, another random point is shown. There will be no frames captured for the incorrect key. This is to ensure that the user fixates on the screening point while clicking the key. Ten frames from the point of clicking are captured yielding 400 frames. We have the ground truth data, and these frames are now passed to the models trained with different personal calibration points for prediction. Although a popular method to calculate accuracy is to take a portion of the ground truth data from the dataset, ground truth data is collected from the user in this analysis. This helps with analyzing the performance of the model for the person it was especially re-trained for and on the device, they intend to use this application. The absolute difference between the actual and predicted coordinates is averaged for each set of frames collected for that point. This is to analyze the relationship between the number of calibration points and the percentage of error. For each set of calibration points, heatmaps below are created to understand which parts of the screen experience a higher rate of error. From figures 18a, 19a, 20a, 21a, and 22a, we can notice that majority of the errors lie along the borders of the screen. This is in contrast with the errors observed in the x-axis. Here, the errors are distributed in different parts of the screen without any pattern. However, while using 0 calibration points in figure 18a, it can be seen that the errors are majorly along the borders of the screen. It is interesting to note the effect of the calibration points on the geography of these errors. In cases where 9, 16, and 32 calibration points are used, these points were arranged uniformly on the screen in a grid format. This tends to cover the borders of the screen. Hence, errors tend to be more evenly distributed as observed in figures 19a, 20a and 21a. In the case of 64 calibration points, there was no uniform display. The points were spread randomly across the screen. Hence, the errors are heavy in the borders as observed in 22a. The advantage of covering the bordering with a grid of calibration points is also reflected in the accuracies found in figures 19b, 20b, and 21b. A grid of 16 calibration points has the best accuracy of 89.69% on the x-axis and 93.77% on the y-axis. The random set of 64 points performs poorer than the case where no calibration points were used. Refer to figures 18 and 22. Figure 23 shows a consolidation of the accuracies observed while using the different number of calibration points. The accuracy increases until 16 points and then starts to decrease.

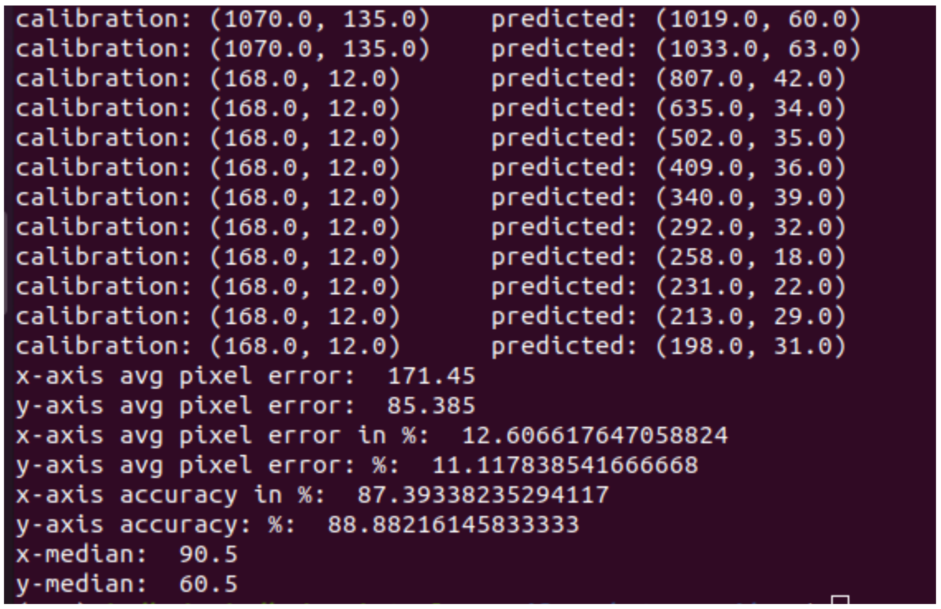




```
calibration: (1070.0, 135.0)    predicted: (902.0, 360.0)
calibration: (168.0, 12.0)      predicted: (713.0, 329.0)
calibration: (168.0, 12.0)      predicted: (571.0, 307.0)
calibration: (168.0, 12.0)      predicted: (463.0, 296.0)
calibration: (168.0, 12.0)      predicted: (386.0, 288.0)
calibration: (168.0, 12.0)      predicted: (331.0, 284.0)
calibration: (168.0, 12.0)      predicted: (290.0, 275.0)
calibration: (168.0, 12.0)      predicted: (259.0, 263.0)
calibration: (168.0, 12.0)      predicted: (238.0, 263.0)
calibration: (168.0, 12.0)      predicted: (225.0, 266.0)
calibration: (168.0, 12.0)      predicted: (213.0, 265.0)
x-axis avg pixel error:  232.21
y-axis avg pixel error:  163.105
x-axis avg pixel error in %:  17.074264705882353
y-axis avg pixel error: %:  21.237630208333332
x-axis accuracy in %:  82.92573529411764
y-axis accuracy: %:  78.76236979166667
x-median:  202.5
y-median:  168.5
```

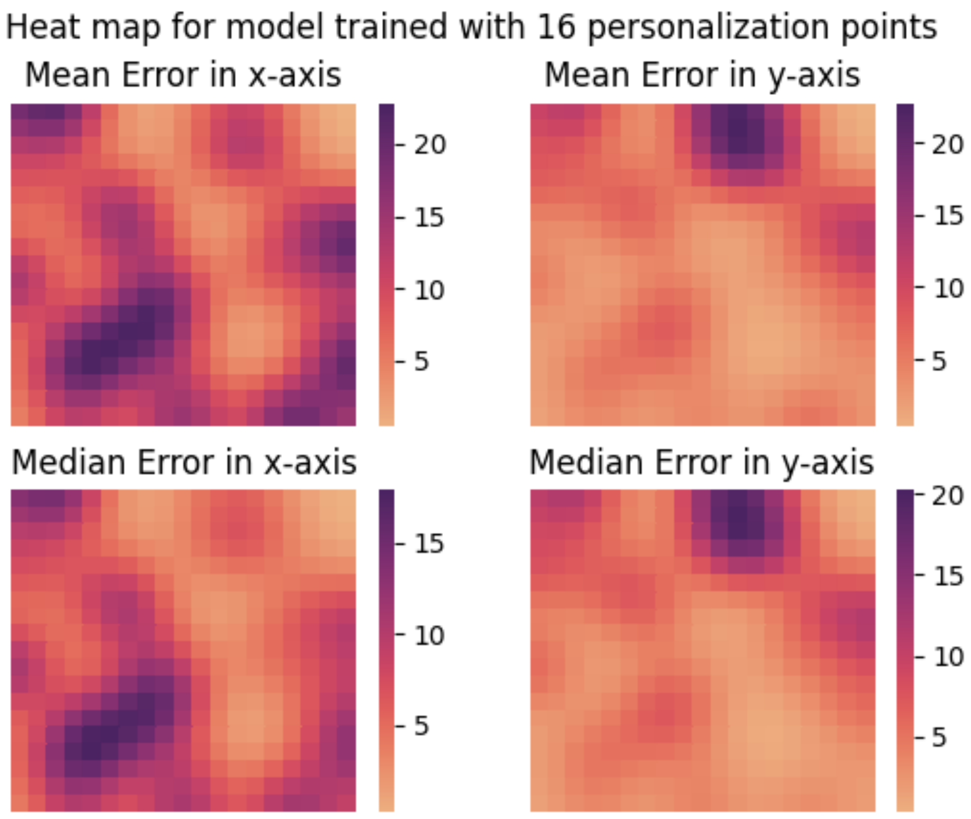
Fig 18. (a) Heat-map for a model trained with 0 calibration points (b) Model accuracy analysis with 400 test frames

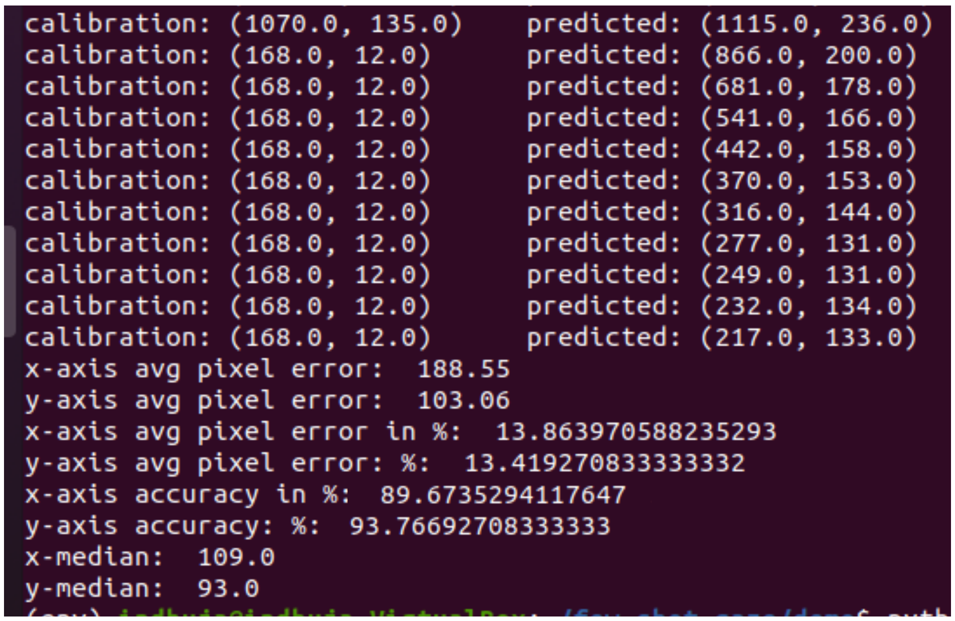




```
calibration: (1070.0, 135.0)    predicted: (1019.0, 60.0)
calibration: (1070.0, 135.0)    predicted: (1033.0, 63.0)
calibration: (168.0, 12.0)      predicted: (807.0, 42.0)
calibration: (168.0, 12.0)      predicted: (635.0, 34.0)
calibration: (168.0, 12.0)      predicted: (502.0, 35.0)
calibration: (168.0, 12.0)      predicted: (409.0, 36.0)
calibration: (168.0, 12.0)      predicted: (340.0, 39.0)
calibration: (168.0, 12.0)      predicted: (292.0, 32.0)
calibration: (168.0, 12.0)      predicted: (258.0, 18.0)
calibration: (168.0, 12.0)      predicted: (231.0, 22.0)
calibration: (168.0, 12.0)      predicted: (213.0, 29.0)
calibration: (168.0, 12.0)      predicted: (198.0, 31.0)
x-axis avg pixel error:  171.45
y-axis avg pixel error:  85.385
x-axis avg pixel error in %:  12.606617647058824
y-axis avg pixel error: %:  11.117838541666668
x-axis accuracy in %:  87.39338235294117
y-axis accuracy: %:  88.88216145833333
x-median:  90.5
y-median:  60.5
```

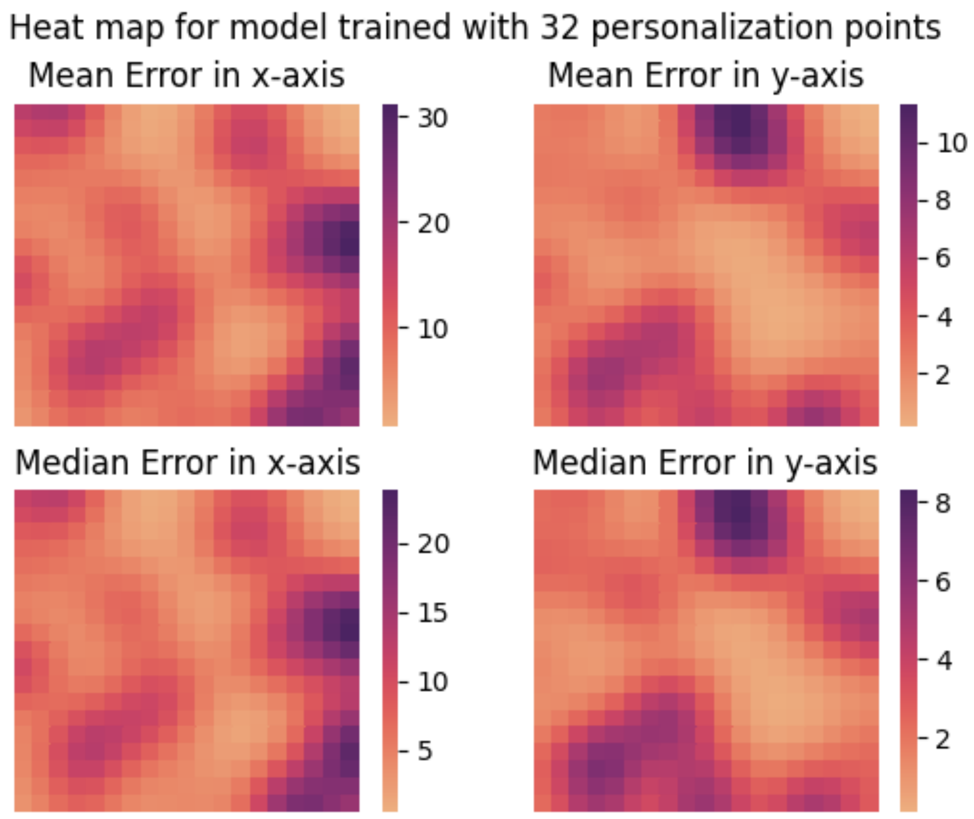
Fig 19. (a) Heat-map for a model trained with 9 calibration points (b) Model accuracy analysis with 400 test frames

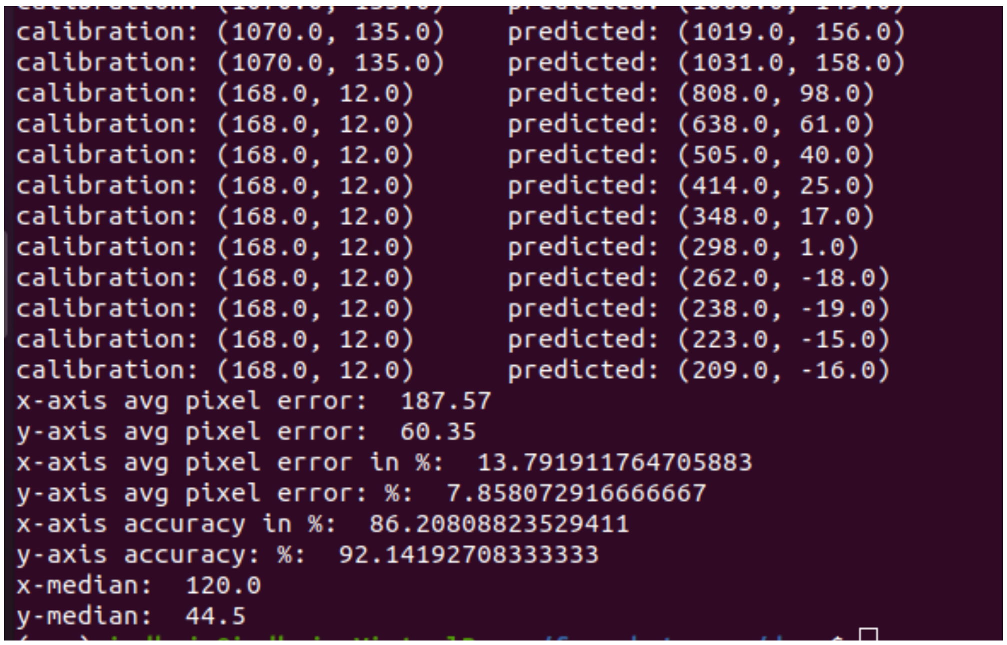




```
calibration: (1070.0, 135.0)    predicted: (1115.0, 236.0)
calibration: (168.0, 12.0)      predicted: (866.0, 200.0)
calibration: (168.0, 12.0)      predicted: (681.0, 178.0)
calibration: (168.0, 12.0)      predicted: (541.0, 166.0)
calibration: (168.0, 12.0)      predicted: (442.0, 158.0)
calibration: (168.0, 12.0)      predicted: (370.0, 153.0)
calibration: (168.0, 12.0)      predicted: (316.0, 144.0)
calibration: (168.0, 12.0)      predicted: (277.0, 131.0)
calibration: (168.0, 12.0)      predicted: (249.0, 131.0)
calibration: (168.0, 12.0)      predicted: (232.0, 134.0)
calibration: (168.0, 12.0)      predicted: (217.0, 133.0)
x-axis avg pixel error:  188.55
y-axis avg pixel error:  103.06
x-axis avg pixel error in %:  13.863970588235293
y-axis avg pixel error: %:  13.419270833333332
x-axis accuracy in %:  89.6735294117647
y-axis accuracy: %:  93.76692708333333
x-median:  109.0
y-median:  93.0
```

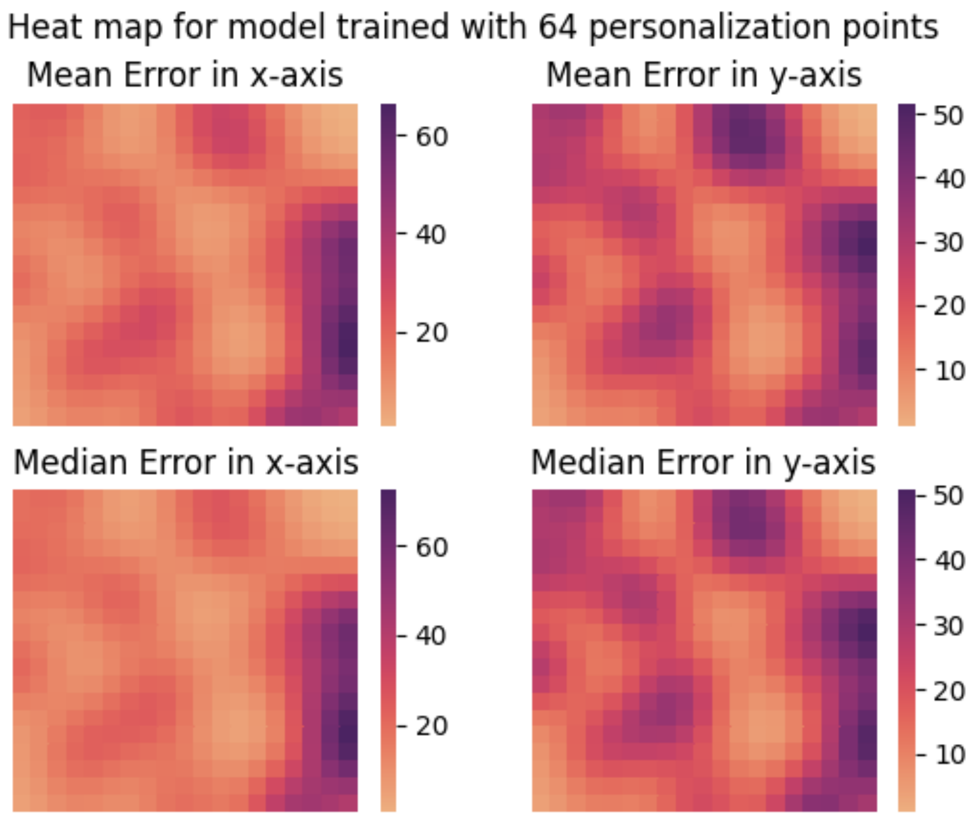
Fig 20. (a) Heat-map for a model trained with 16 calibration points (b) Model accuracy analysis with 400 test frames

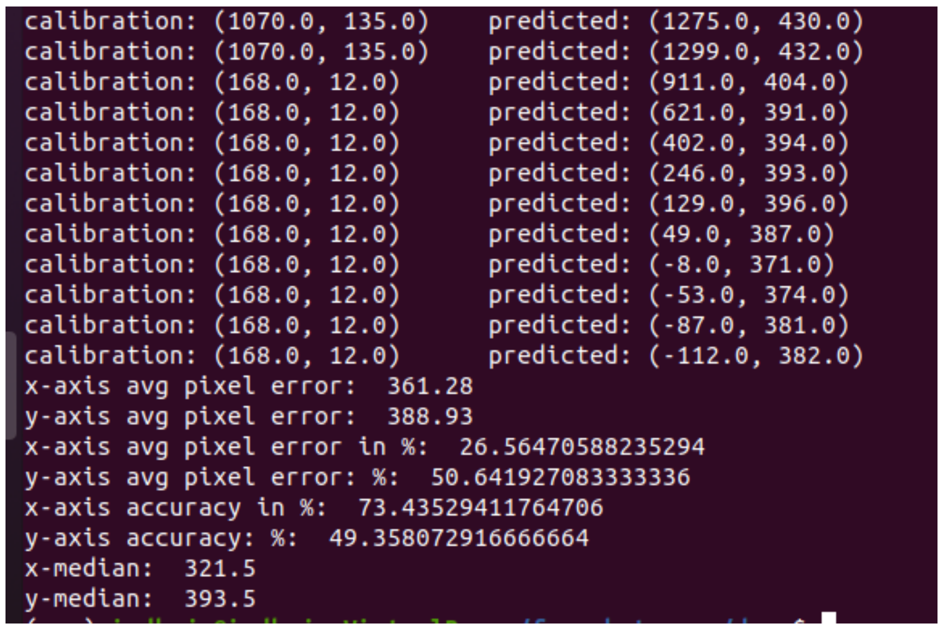




```
calibration: (1070.0, 135.0)    predicted: (1019.0, 156.0)
calibration: (1070.0, 135.0)    predicted: (1031.0, 158.0)
calibration: (168.0, 12.0)      predicted: (808.0, 98.0)
calibration: (168.0, 12.0)      predicted: (638.0, 61.0)
calibration: (168.0, 12.0)      predicted: (505.0, 40.0)
calibration: (168.0, 12.0)      predicted: (414.0, 25.0)
calibration: (168.0, 12.0)      predicted: (348.0, 17.0)
calibration: (168.0, 12.0)      predicted: (298.0, 1.0)
calibration: (168.0, 12.0)      predicted: (262.0, -18.0)
calibration: (168.0, 12.0)      predicted: (238.0, -19.0)
calibration: (168.0, 12.0)      predicted: (223.0, -15.0)
calibration: (168.0, 12.0)      predicted: (209.0, -16.0)
x-axis avg pixel error:  187.57
y-axis avg pixel error:  60.35
x-axis avg pixel error in %:  13.791911764705883
y-axis avg pixel error: %:  7.858072916666667
x-axis accuracy in %:  86.20808823529411
y-axis accuracy: %:  92.14192708333333
x-median:  120.0
y-median:  44.5
```

Fig 21. (a) Heat-map for a model trained with 32 calibration points (b) Model accuracy analysis with 400 test frames





```
calibration: (1070.0, 135.0)    predicted: (1275.0, 430.0)
calibration: (1070.0, 135.0)    predicted: (1299.0, 432.0)
calibration: (168.0, 12.0)      predicted: (911.0, 404.0)
calibration: (168.0, 12.0)      predicted: (621.0, 391.0)
calibration: (168.0, 12.0)      predicted: (402.0, 394.0)
calibration: (168.0, 12.0)      predicted: (246.0, 393.0)
calibration: (168.0, 12.0)      predicted: (129.0, 396.0)
calibration: (168.0, 12.0)      predicted: (49.0, 387.0)
calibration: (168.0, 12.0)      predicted: (-8.0, 371.0)
calibration: (168.0, 12.0)      predicted: (-53.0, 374.0)
calibration: (168.0, 12.0)      predicted: (-87.0, 381.0)
calibration: (168.0, 12.0)      predicted: (-112.0, 382.0)
x-axis avg pixel error:  361.28
y-axis avg pixel error:  388.93
x-axis avg pixel error in %:  26.56470588235294
y-axis avg pixel error: %:  50.641927083333336
x-axis accuracy in %:  73.43529411764706
y-axis accuracy: %:  49.358072916666664
x-median:  321.5
y-median:  393.5
```

Fig 22. (a) Heat-map for a model trained with 64 calibration points (b) Model accuracy analysis with 400 test frames

Number of Personalization Points	Accuracy in the x-axis (%)	Accuracy in the y-axis (%)
0	82.93	78.76
9	87.39	88.88
16	89.69	93.77
32	86.21	92.14
64	73.44	49.36

Fig 23. Consolidation of results observed with different no of personalization points

11.2 CNN

A hybrid CNN model was constructed to solve the gaze prediction problem. The model is made of three parts:

1. Global CNN: This captures the global information by taking the whole frame with the face as input.
2. Right eye-specific CNN: The cropped eye patch of the right eye is taken as input.
3. Left eye-specific CNN: The cropped eye patch of the left eye is taken as input.

(Note that the right eye appears on the left of the image and the left eye appears on the right.)

These three CNNs act as encoders and generate a feature representation for each frame in the video. These features are combined using a linear layer to perform regression where the x, y, and z coordinates are predicted. The global CNN is a model pre-trained on ImageNet data. The model is pre-trained using Resnet34, Vision Transformer, and EfficientNetB2. The left and right eye specific CNNs are smaller and trained from scratch on the MPIIFaceGaze dataset. The loss function used to optimize and update the weights at each step is Mean Squared Loss (MSE). L1 regularization method was used on the weights of the model. Experiments were also conducted by varying the learning rate for all 3 CNNs since the global model is pre-trained. To avoid overtraining the model, early stopping was used. A challenge with neural nets is the duration of training. Very little training will result in underfitting and too much will result in overfitting. A middle ground can be reached by pausing training when the model performance on the validation set begins to decrease. With the goal of this project being to develop a person-independent gaze predictor, early stopping will help generalize the model. The observed loss is plotted in the graphs in figures 24, 25, and 26. From figure 24, we can see the ResNet34 reaches convergence at the 7th epoch with a loss of 40.21%. Using Vision Transformer, we can observe in figure 25 that the convergence is at the 9th epoch with a loss of 40.19%. While comparing Vision Transformer with ResNet34, it can be seen that the loss of the former reduces almost consistently. From figure 26, we can see that EfficientNetB performs the worst out of the three with frequent fluctuations in loss. In all three global CNN's the loss does not improve beyond 40.1%. Hence, these models are not suitable for the real-time prediction we aim for in this project.

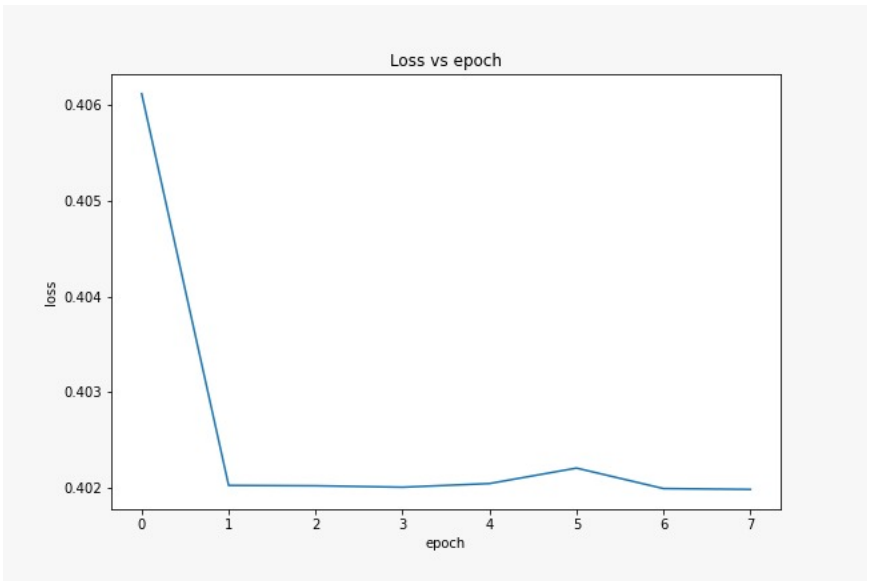


Fig 24. The plot of loss using ResNet34 to train Global CNN

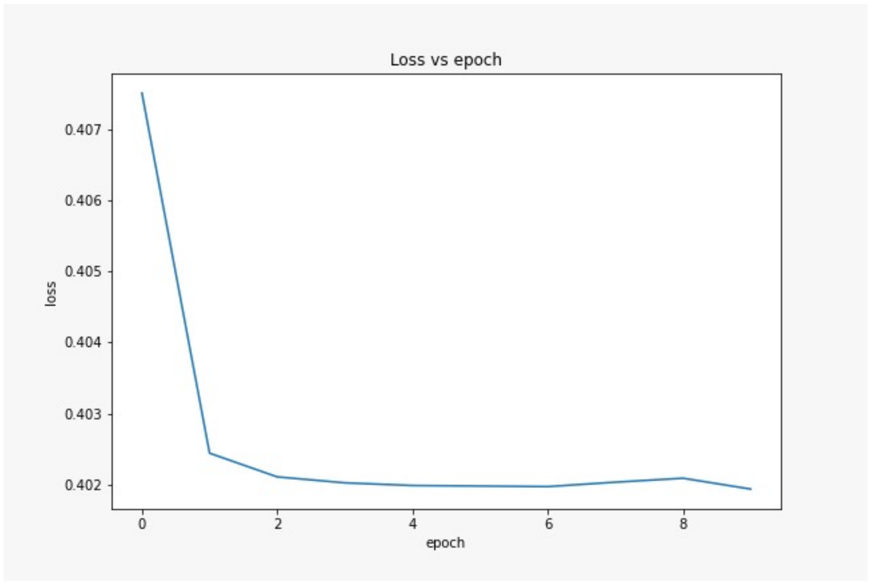


Fig 25. The plot of loss using Vision Transformer to train Global CNN

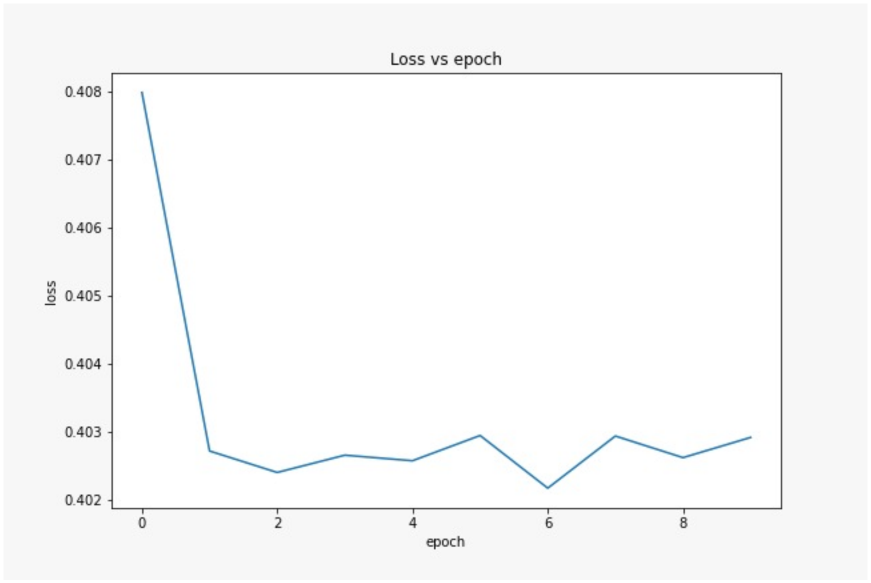


Fig 26. The plot of loss using EfficientNetB2 to train Global CNN

11.3 Voice command classification model using CNN:

The voice control dataset is trained on a Resnet18 architecture. A custom linear layer is included at the end of this model to perform transfer learning. Resnet18 is used to recognize the spatial features in the STFT spectrograms to classify the audio into one of the 7 classes. Therefore, the CNN can be thought of like a fingerprint recognizer where the spectrogram contains a unique fingerprint for specific audio along with some noise. The model is trained using a cross-entropy loss and Adam optimizer. Similar to the eye patch models, L1 weight regularization is used. The plot below in figure 27 is the training loss analysis plot. For each epoch, the total training loss is plotted. We can observe that there is a steep decrease in loss with several irregular peaks. However, the model converges with an error rate of 8%. Therefore, we have an accuracy of 92% for the voice controller.

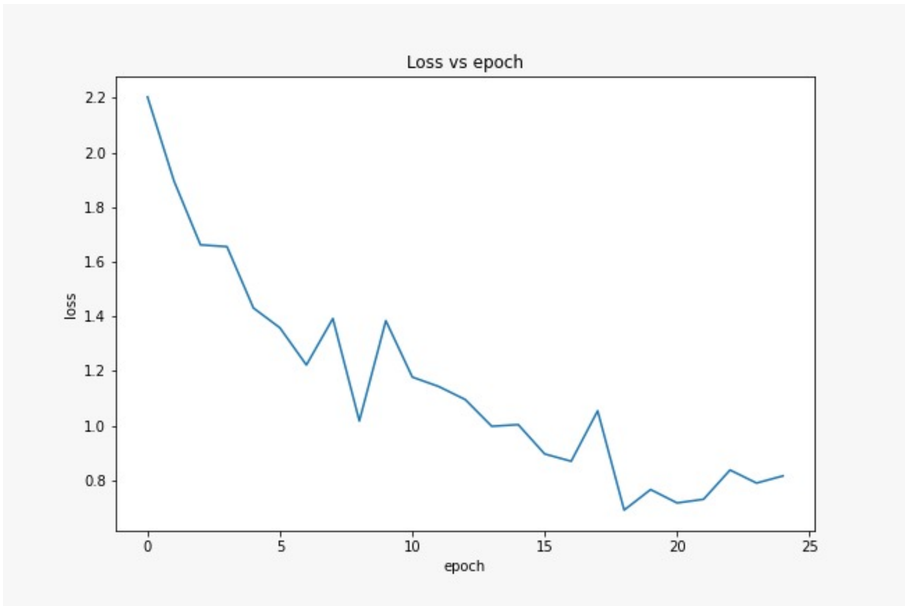


Fig 27. The plot of loss for Resnet18 audio model

CHAPTER 12

Conclusion

The field of accessibility improvement is constantly seeing the face of new and creative inventions. The problem of eye tracking is a popular one because of its challenges with using off-the-shelf commercial cameras. In this project, we saw that using an encoder-decoder model helps improve the accuracy of gaze tracking up to 93.77%. We also saw the importance of customizing the eye tracker for each person using personalization points and using grid formats to display these points. With the addition of person-dependent voice control, we are taking this model one step further for implementing a truly hands-free solution that is also affordable and easy to set up. Hence, this hybrid eye gaze tracker is an affordable and non-invasive solution for the problem of eye-tracking. It makes communication with a computing device possible with only voice and gazes tracking.

One of the goals with future work in this project is to allow customizations in the eye tracker. For people who already own state-of-the-art eye trackers, this model could be re-used for providing voice control and improving the accuracy of the gaze tracker with camera images. Another goal is also to arrange continuous learning modules to the existing models that will help with personalizing the eye tracker to a particular person. Having a generic model that is completely person-independent and continuously training this for a single person’s gaze would be helpful since personal computers are mostly always used by the same person. Further, we can also explore the impact of the multiple camera approach on the accuracy of the gaze tracker. As a continuation of this project, starting an effort to collect voice control data of commands and short-cuts frequently used can help related research in this field.

REFERENCES

[1] S. Baluja and D. Pomerleau, "Non-intrusive gaze tracking using artificial neural networks," in Advances in Neural Information Processing Systems, pp. 753-760, 1994.

[2] C.H. Morimoto and M.R. Mimica, "Eye gaze tracking techniques for interactive applications," Computer Vision Image Understanding, vol. 98, pp. 4-24, 2005.

[3] H.R. Chennamma and X. Yuan, "A survey on eye-gaze tracking techniques," arXiv Preprint arXiv:1312.6410, 2013.

[4] Sahay and P. Biswas, "Webcam Based Eye Gaze Tracking Using a Landmark Detector," in Proceedings of the 10th Annual ACM India Compute Conference, pp. 31-37, 2017.

[5] D. Beymer and M. Flickner, "Eye gaze tracking using an active stereo head," in 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. pp. II-451, 2003.

[6] L. Burton, W. Albert, and M. Flynn, "A comparison of the performance of webcam vs. infrared eye-tracking technology," in Proceedings of the Human Factors and Ergonomics Society Annual Meeting, pp. 1437-1441, 2014.

[7] D. Li, D. Winfield, and D.J. Parkhurst, "Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches," in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops, pp. 79, 2005.

[8] D. Li, J. Babcock and D.J. Parkhurst, "openEyes: a low-cost head-mounted eye-tracking solution," in Proceedings of the 2006 symposium on Eye-tracking research & applications, pp. 95-100, 2006.

[9] E.B. Huey, The psychology and pedagogy of reading, The Macmillan Company, 1908.

[10] T. Ohno, N. Mukawa and A. Yoshikawa, "FreeGaze: a gaze tracking system for everyday gaze interaction," in Proceedings of the 2002 symposium on Eye-tracking research & applications, pp. 125-132, 2002.

[11] W. Horng, C. Chen, Y. Chang, and C. Fan, "Driver fatigue detection based on eye-tracking and dynamic template matching," in IEEE International Conference on Networking, Sensing and Control, 2004, pp. 7-12, 2004.

[12] N.K. Malhotra, Review of marketing research, ME Sharpe, 2004.

[13] S. Djamasbi, "Eye tracking and web experience," AIS Transactions on Human-Computer Interaction, vol. 6, pp. 37-54, 2014.

[14] A.O. Mohamed, M.P. Da Silva, and V. Courboulay, "A history of eye gaze tracking," 2007.

[15] R.J. Cooper and J.C. Senge, "An attempt to define fully-accessible workstation levels of accessibility," in International Conference on Computers for Handicapped Persons, pp. 164-169, 1994.

[16] C. Lutteroth, M. Penkar, and G. Weber, "Gaze vs. mouse: A fast and accurate gaze-only click alternative," in Proceedings of the 28th annual ACM symposium on user interface software & technology, pp. 385-394, 2015.

[17] P. Karlsson, A. Bech, H. Stone, C. Vale, S. Griffin, E. Monbaliu and M. Wallen, "Eyes on communication: trialing eye-gaze control technology in young children with dyskinetic cerebral palsy," Developmental Neurorehabilitation, vol. 22, pp. 134-140, 2019.

[18] P. Karlsson, A. Allsop, B. Dee-Price and M. Wallen, "Eye-gaze control technology for children, adolescents and adults with cerebral palsy with significant physical disability: Findings from a systematic review," Developmental Neurorehabilitation, vol. 21, pp. 497-505, 2018.

[19] M. Paciello, Web accessibility for people with disabilities, CRC Press, 2000.

[20] T. Pousada, J. Pareira, B. Groba, L. Nieto and A. Pazos, "Assessing Mouse Alternatives to Access to Computer: A Case Study of a User with Cerebral Palsy," Null, vol. 26, pp. 33-44, 2014.

[21] G. Kouroupetroglou, Assistive Technologies and Computer Access for Motor Disabilities, Hershey: IGI Global, 2013, pp. 206-221.

[22] T. Jones, "An Empirical Study of Children's Use of Computer Pointing Devices," Journal of Educational Computing Research, vol. 7, pp. 61-76, 1991.

[23] M. Hertzum and K. Hornbæk, "How Age Affects Pointing with Mouse and Touchpad: A Comparison of Young, Adult, and Elderly Users," Null, vol. 26, pp. 703-734, 2010.

[24] Martin-Hammond, A. Ali, C. Hornback and A. Hurst, "Understanding design considerations for adaptive user interfaces for accessible pointing with older and younger adults," in Proceedings of the 12th Web for All Conference, pp. 1-10, May 18, 2015.

[25] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik and A. Torralba, "Eye Tracking for Everyone," pp. 2176-2184, Jun 2016.

[26] Xucong Zhang, Y. Sugano, M. Fritz and A. Bulling, "It's Written All Over Your Face: Full-Face Appearance-Based Gaze Estimation," pp. 2299-2308, Jul 2017.

[27] E.S. Dalmaijer, S. Mathôt and S. Van Der Stigchel, "PyGaze: An open-source, cross-platform toolbox for minimal-effort programming of eye-tracking experiments," Behav Res, vol. 46, pp. 913, -11-21. 2013.

[28] U. Park and A.K. Jain, "3D Model-Based Face Recognition in Video," in Advances in Biometrics, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1085-1094.

[29] S. Park, S. De Mello, P. Molchanov, U. Iqbal, O. Hilliges and J. Kautz, "Few-Shot Adaptive Gaze Estimation," May 06, 2019.