Text Editor And Text Reader For Disabled People Using An Eye Tracker

by
Görkem Özgül

Engineering Project Report

Yeditepe University
Faculty of Engineering
Department of Computer Engineering
2024

Text Editor And Text Reader For Disabled People Using An Eye Tracker

APPROVED BY:

Assist. Prof. Dr. Esin Onbaşıoğlu (Supervisor) .....

Assist. Prof. Dr. Mustafa B. Mutluoğlu .....

Prof. Dr. Gürhan Küçük .....

DATE OF APPROVAL: 01/2024

ACKNOWLEDGEMENTS

First of all I would like to thank my advisor Assist. Prof. Dr. Esin Onbaşıoğlu for her guidance and support throughout my project.

I am forever grateful to my family, especially my parents, for their support, belief in my abilities, and constant encouragement throughout my educational journey.

A special thanks goes to my university colleagues and those who attend my eye tracker project. Their collaborative spirit contributed to my research experience. Their perspectives and reviews have helped me to improve my work.

ABSTRACT

Text Editor And Text Reader For Disabled People Using An Eye Tracker

The combination of eye-tracking technology into virtual keyboards is an important advancement in assistive communication devices, particularly for people with disabilities that make traditional methods of interactions difficult. This project looks into the design and implementation of an eye-tracking virtual keyboard designed to provide efficient, simple to use communication solution for people with disabilities.

At the core of this system lies a Convolutional Neural Network (CNN), a form of deep learning, which is trained to interpret and translate eye images into precise cursor movements on a virtual keyboard. Training uses processed eye image datasets to teach model how to correlate specific eye positions and estimate blinks. Blink detection allows users to select keys on virtual keyboard which is a simulating action of a mouse click.

Evaluation part of this project is the development of three distinct keyboard layouts, each designed with goal of optimizing ease of use and typing efficiency. Experiment conducted with 9 persons from different age range. Participants interacted with each keyboard layout while eye-tracking translated their gaze patterns into cursor movements. Experiment measured time it took to type six different predetermined words, allowing for an accurate measurement of each layout's effectiveness.The findings of this experiment not only help to optimize eye-tracking virtual keyboards, but also represent a significant step forward in improving communication capabilities for people with disabilities.

ÖZET

Engelli Bireyler İçin Metin Düzenleyici ve Metin Okuyucu Göz Takip Cihazı

Göz izleme teknolojisinin sanal klavyelerle birleştirilmesi, özellikle engelli kişiler için geleneksel iletişim yöntemlerinin aksine iletişim cihazları arasında önemli bir yerdedir. Bu makale, engelli kişiler için verimli, kullanımı basit bir iletişim çözümü sağlamak üzere tasarlanmış bir göz izleme sanal klavyesinin tasarımını ve uygulanmasını içermektedir.

Bu sistemin merkezinde, göz görüntülerini sanal bir klavyede hassas imleç haraketlerine çevirmek ve yorumlamak için eğitilmiş bir derin öğrenme çeşidi olan CNN yer alıyor. Eğitim, modele belirli göz konumlarının nasıl ilişkilendirileceğini ve göz kırpmalarının nasıl tahmin edileceğini öğretmek için işlenmiş göz görüntüsü veri kümelerini kullanıyor. Göz kırpma, algılama algoritmaları kullancıların bir fare tıklamasını simüle ederek sanal klavyedeki tuşlara tıklamasını sağlıyor.

Bu projenin değerlendirilmesinde, her biri kullanım kolaylığını ve yazma verimliliğini optimize etme hedefiyle tasarlanan üç farklı klavye tasarımı yer alıyor. Deney, farklı yaş aralığında toplam 9 kişiyle gerçekleştirildi. Katılımcılar her klavye düzenini kullanırken, göz izleme teknolojisi de bakış verilerini imleç haraketlerine dönüştürdü. Deney, önceden belirlenmiş altı farklı kelimeyi yazmak için gereken süreyi ölçerek her tasarımın verimliliğin ölçülmesine olanak sağladı. Bu deneyin bulguları yalnızca göz izleme teknolojisi destekli sanal klavyelerin optimize edilmesine yardımcı olmakla kalmıyor, aynı zamanda engelli insanlar için iletişim yeteneklerinin iyileştirilmesinde ileriye yönelik atılmış önemli bir adımı da temsil ediyor.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

ρ	Landmark Point
CNN	Convolutional Neural Network
CRI	Cross Ratio Invariant
GTTS	Google Text-to-Speech
API	Application Programming Interface
WPM	Words Per Minute

1. INTRODUCTION

In recent years, there has been a major rise in research aimed at improving the lives of individuals with disabilities who are unable to communicate through voice, sign language, or writing. Among these tools, the use of eye movements as a human-computer interaction tool has gained importance.

The use of eye-gaze monitoring technologies represents an approach to accessibility and efficiency in the developing area of human-computer interaction. This work comprising an innovative eye-gaze controlled keyboard interface, stands at the forefront of this technological advancement.

1.1. Problem Definition

Patients with severe difficulties in speech and movement, who cannot speak or use sign language, need special ways to interact with computers to communicate with others. Depending on the patient's disability, communication tools must be adjusted to their needs. This can range from modifying existing devices (like keyboards) to creating advanced technology (such as devices that connect the brain to a machine, used for patients who are completely paralyzed). People with disabilities who can control their eye movements can use their eyes to communicate, such as using their eye movements to type on a virtual keyboard and then converting that text into speech.

An eye-tracker is a tool that tracks eye movements like where someone is looking, how long they look at something, and how fast and far their eyes move. There have been different types of eye-trackers for many years. Some require wearing special equipment on the head or attaching sensors to the person (like measuring eye movement with electrical signals), while others not making physical contact at all. When using an eye-tracker, it can be hard to tell if someone is looking at something on purpose or by accident, because we often move our eyes without meaning to. This can cause unintended selections, to fix this, you can set it so that looking at something for a certain amount of time means you choose it. You can also use the eye-tracker to point at something and then use a different tool, like a button or blinking to select it.

There are many methods discussed in research for estimating where someone is looking, tracking eye movements, and detecting head movements. Detecting head movements

is important, especially if the person can move their head during tests, instead of keeping it still on a chin rest like in some eye studies. A good eye-tracking system should be able to follow not just where the eyes are looking, but also the position and direction of the head to avoid mistakes. These systems use different ways to calibrate, like adjusting to the user or using a few specific points for calibration. The quality and price of these systems vary a lot, from basic ones using a regular webcam to high-quality ones with better optics. The more precise the system, especially for measuring quick eye movements, the more expensive it is. Because of the high cost, these advanced systems aren't widely developed, and it can be hard to get funding for them. Recently, more affordable eye-tracking options have come into the market. These use a camera placed near the computer screen and don't require the user to wear any special equipment. These less expensive devices are creating new interest in using eye-tracking for interacting with computers.

Even though there were early attempts to use eye-trackers for computer input, the use of these devices for virtual keyboards is still not very common. A good system for detecting and tracking eye movements is crucial for human-computer interaction. It helps us understand where people are looking and what they need. Virtual keyboards are a leading way to test new ways for people to interact with computers. There is a need to develop more virtual keyboards that can help many disabled people to communicate and use computers. When creating virtual keyboards for people with disabilities, it's important to consider their specific needs, like whether they can press a button or control their gaze effectively.

A major difficulty in creating a reliable, mobile, and budget-friendly virtual keyboard that uses gaze detection is considering the limitations of the eye-tracking technology and the design of how people interact with computers. For example, the design of a normal keyboard can be problematic for eye-tracking. This is because the keys are close to each other, which might confuse the system when it's trying to figure out where someone is looking. In our study, we suggest a virtual keyboard design that uses 41 keys includes 0 to 9 digits, all English letters and various functions like clear, backspace and speak. System also has a feature for completing words automatically.

1.2. Requirements

The main goal of eye-tracking application is to provide a means of communication and computer interaction for individuals with speech and mobility. It accomplishes this by allowing users to control a virtual keyboard with their eyes, making typing and navigation activities easier. System is designed to be compatible with major operating systems includ-

ing Windows, macOS, and Linux. Python is used for the core development, with libraries like OpenCV for advanced image processing, Keras for implementing machine learning methods, and Pygame for generating an interactive user interface. Text-to-speech capabilities are enabled by pyttsx3 and gTTS, ensuring clear and natural voice synthesis. Additional libraries like numpy for mathematical operations, textblob for language processing, and enchant for spell-checking play crucial roles in enhancing the system's efficiency and user experience.

2. BACKGROUND

Eye tracking technology plays a crucial role in the development of text editors and readers for people with disabilities, particularly those who have limited mobility or speech impairments. Eye tracking technology allows individuals with disabilities to interact with computers in a way that bypasses traditional input methods like keyboards and mouse. This is particularly beneficial for users who may have difficulty using these standard input devices due to physical limitations.

By tracking eye movements, text editor and reader can detect where the user is looking on the screen, enabling them to select, edit, or read text just by focusing their gaze on specific areas. This makes the technology intuitive and user-friendly, even for those who are not familiar with computers.

2.1. Previous Work

Hubert *et al* [1] developed an multi modal virtual keyboard using eye-tracking and hand gesture detection. This virtual keyboard is based on different menu sections and eight main commands that allows users to write 30 different characters with correction tool and delete function can be seen on Figure 2.1. System evaluated with 18 participant and main components of comparison are eye tracking system, mouse and eye tracking command section. The performance was evaluated by speed and consistency.

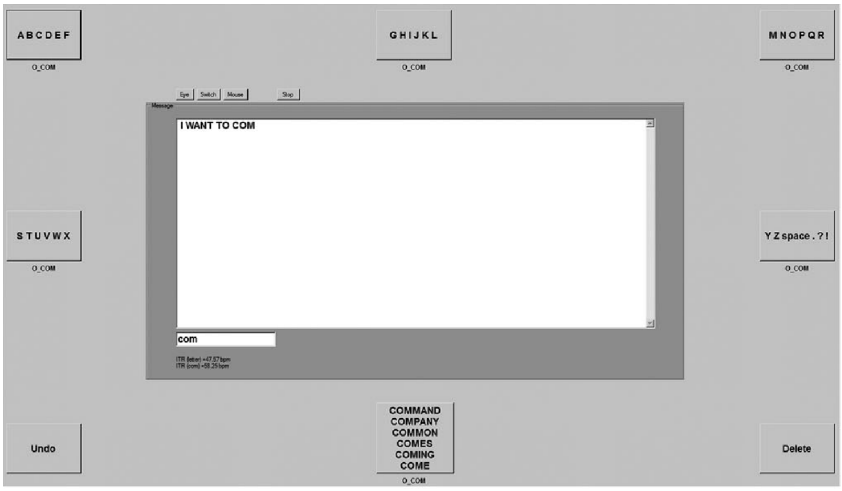

Figure 2.1. Eight command virtual keyboard

Metin *et al* [2] developed Gaze-Controlled Turkish Virtual Keyboard for ALS patients, communications need of disabled people to write their eyes. System covers the eye with a plastic glasses frame. Webcam fixed on computer directly in front of it. The virtual keyboard has 22 points placed around board especially 16 characters with a total usage frequency of Turkish 2.2. With single eye movements one of them is obtainable easily others are last for a while to capture. System uses machine learning to find iris circle which is the K-nearest neighbor's algorithm.

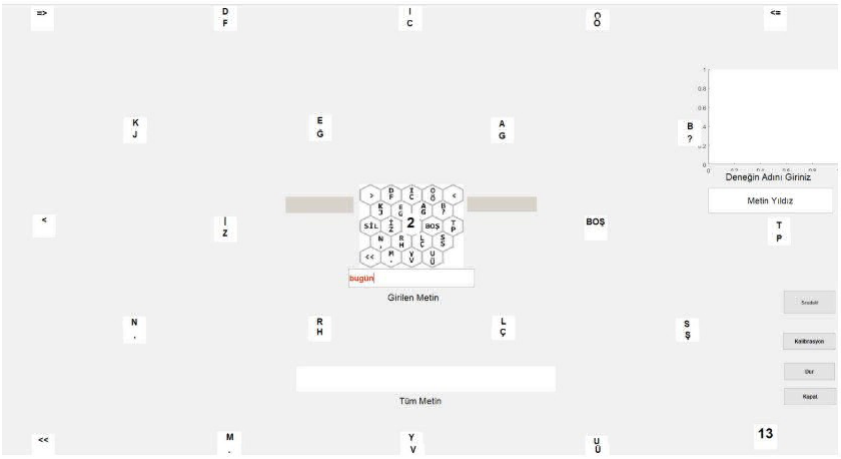

Figure 2.2. Eight command virtual keyboard

Robuil *et al* [3] developed Computer Vision Based Eye Gaze Controlled Virtual Keyboard. It is designed for people with disabilities, especially Tetraplegia patients which cannot control their limbs. This virtual keyboard design aids this people's communication needs. The keyboard has 40 different keys including delete function shown on Figure 2.3 . The patients eye gaze is navigating the activation key and blinking eyes corresponds to the selection function.

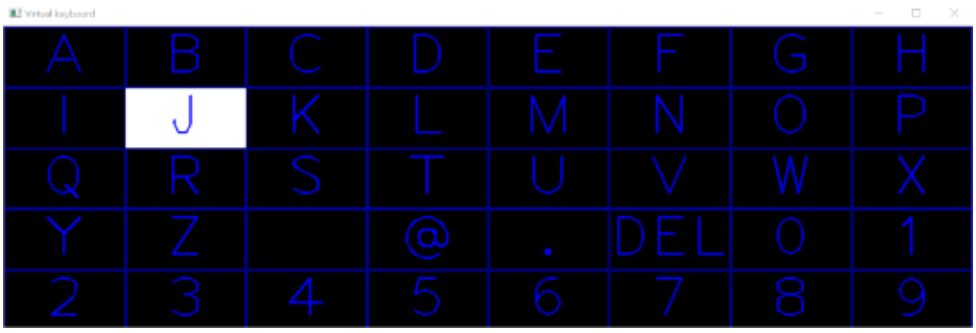

Figure 2.3. Visual of Virtual Keyboard

Yang *et al* [4] developed a method for eye detection that relies on the contrast in gray levels between the face, pupils, and points of corneal reflection shown in Figure 2.4. They tested this method using an eye tracking system that remains accurate regardless of the angle of gaze (cross-ratio-invariant). The testing included participants with glasses and various accessories, and the system proved effective in reducing the distortions caused by these objects' reflective properties.

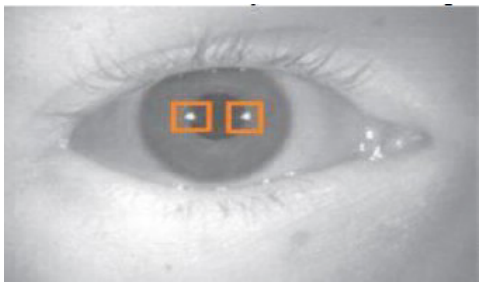

Figure 2.4. Corneal reflection points

Kuo *et al* [5] proposed an eye tracking system that uses the particle filter which estimates a sequence of hidden parameters depending on the data observed. After detecting possible eyes positions, the process of eye tracking starts. For effective and reliable eye tracking, the gray level histogram is selected as the characteristics of the particle filter. Using low-level features in the image makes it a fast algorithm.

The system demonstrates high precision, but its real-time effectiveness wasn't evaluated. The algorithm tested on static images rather than videos, and these images weren't sourced from a recognized database. Therefore, the algorithm's accuracy and functionality might decrease when applied in practical, real-world scenarios.

Li *et al* [6] developed an Gaze estimation from color image based on the eye model with known head pose. The approach involves creating an eye model through an innovative eyeball center calibration technique. This model is crucial in accurately projecting the pupil's position and shape. Additionally, the method incorporates a deformation model for pupils when observed through head-mounted cameras. This model, aided by the analysis of edge gradients in circular patterns, is key in determining the best fitting as shown in Figure 2.5 ellipse for the pupil boundary.

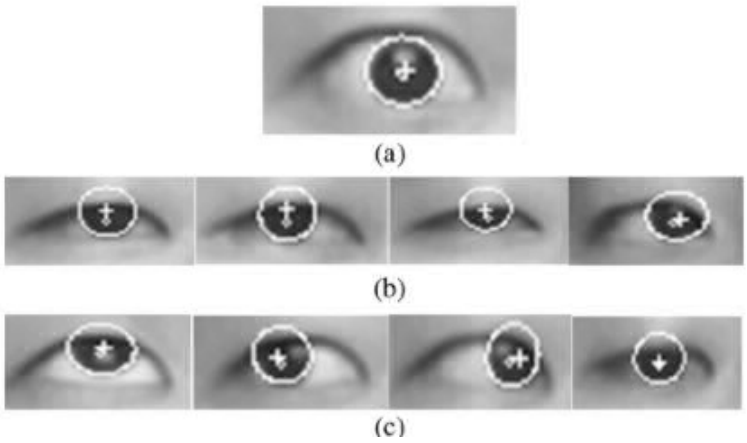

Figure 2.5. Iris Fitting in different conditions

Fu *et al* [7] developed a high-performance eye tracking algorithm that starts with manual extraction of two eye templates from the first video frame for system calibration. The algorithm detects the face region in captured frames and uses normalized 2-D cross-correlation for template matching. Eye gaze direction is determined through iris detection, employing edge detection and Hough circle techniques. This algorithm was applied in a display control application. However, it has a rigid calibration process and was not tested on a diverse group of subjects. Furthermore, the results were not reported in detail, suggesting the need for careful evaluation of the algorithm before implementation in practical scenarios.

Xiong *et al* [8] worked on Eye gaze tracking using an RGBD camera with one-time calibration method to determine the center of the eyeball. This involves the participant looking at nine specific points displayed on a monitor screen. The calibration process is refined by minimizing the differences between the predicted gaze directions and the actual directions. While using a nine-point pattern for eye calibration is quite common, different studies may apply various other principles for this process.

3. ANALYSIS AND DESIGN

The project aims to create an eye-tracking keyboard that allows users to type on a virtual keyboard displayed on the screen by looking at the keys and blinking to select them.

3.1. Eye Calibration

Calibration is the first phase of eye-tracking keyboard system. It is an essential procedure make sure the accurate interpretation of a user's gaze and eye movements to interact with the virtual keyboard displayed on the screen. The calibration process tunes the system to the individual characteristics of the user's eyes and their interaction with the device. This step is crucial for achieving a responsive user experience.

Calibration follows a group of methods.It starts with detecting face of a humans.There is 6 landmarks each on eyes, totally there is 12 landmarks to follow in order to detect the gaze of user. If gaze detection works properly, The board is created with 76 different dots which includes all important parts of the monitor then the key locations of keyboard. 36 Dots distributed across all of the monitor and the other 40 dots are key locations of the keyboard. Capturing eyes is the next step after dot placement, system follows user to look at each dot and capture their eye position while looking at the green dots. User can clearly understood how many calibration points are left by looking at the center of the green dot which shows remaining calibration points. Till all the calibration points data captured by webcam, calibration successfully ends the process. The flowchart of the methods is given in Figure 3.1 .The following are the steps of the virtual keyboard architecture:

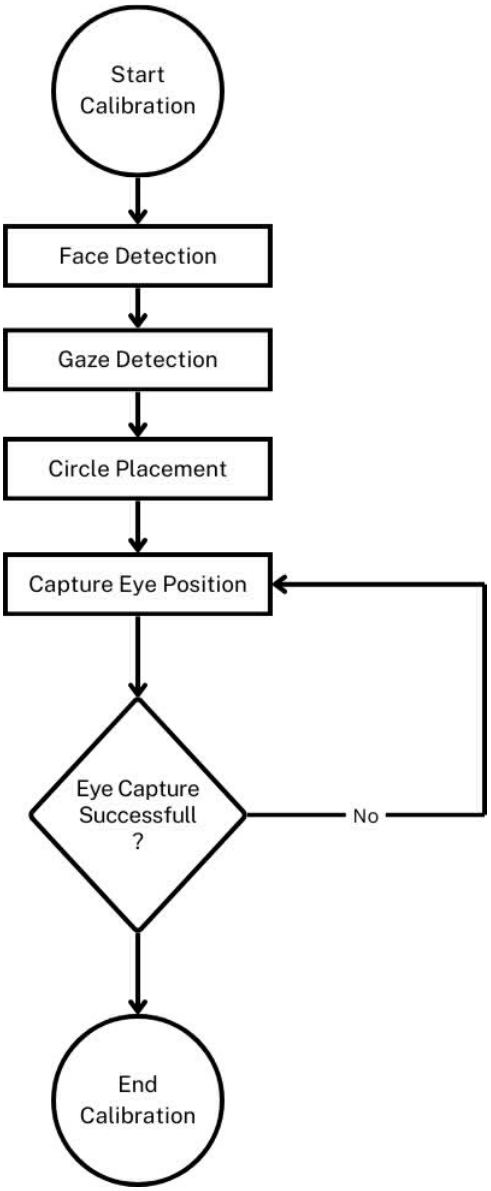

Figure 3.1. Flowchart of Calibration Process

Calibration phase adjusts the system to the unique eye characteristics of the user, such as pupil size, distance, and gaze angle. In that phase, calibration aligns the system's interpretation of gaze data with the actual points on the screen the user is looking at, reducing errors in eye-tracking and improving precision. It ensures that the system can accommodate a variety of environmental variables, such as lighting and background, as well as hardware configurations. In Figure 3.2 Calibration pseudo-code is given.

```
Begin Calibration
    Initialize camera and eye-tracking system
for each position in a predefined set of screen locations do
        Display a green dot at the current position
        Instruct user to look at the green dot
    repeat
            Attempt to capture the eye position using the camera
        if eye position capture is successful then
                Save the captured data corresponding to the screen location
        else
                Inform the user of the issue
                Prompt the user to look at the green dot again
        end if
    until a valid eye position is captured
        Move the green dot to the next position in the sequence
end for
End Calibration
```

Figure 3.2. Pseudo-code Of Calibration Process

After Calibration phase ended there will be 76 samples of eye positions located on the folder that user created. Now on prediction algorithm calculates eye position based on this samples.

The prediction algorithm is at the heart of the eye-tracking system. It uses various mathematical and statistical models to process raw data from the eye-tracking hardware and translates it into screen coordinates. This process involves several stages, from detecting the eyes and processing the image data to applying machine learning techniques for accurate predictions. In Figure 3.3 the sequence diagram of calibration process has shown.

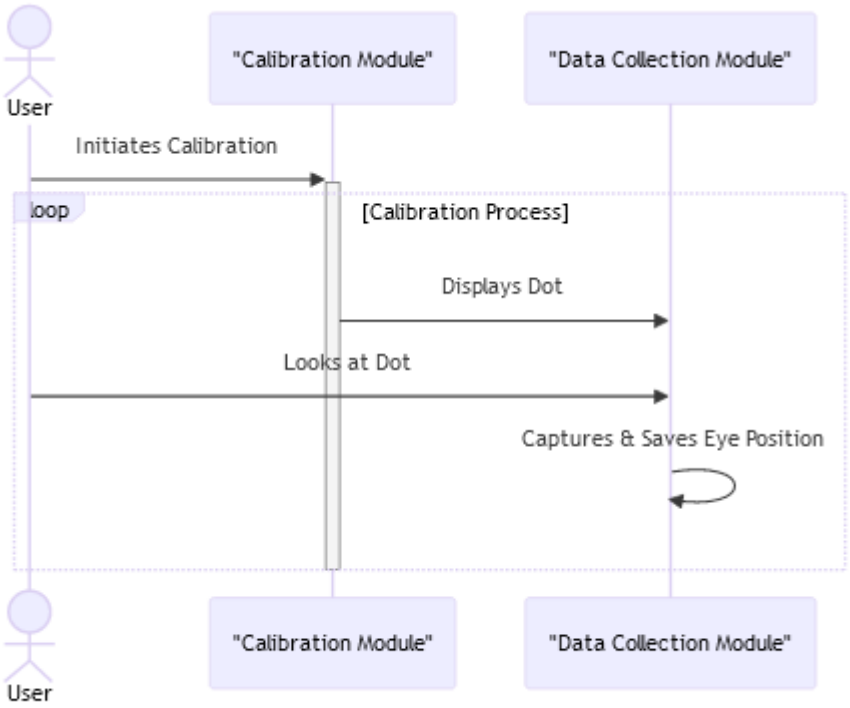

Figure 3.3. Sequence Diagram of Calibration

Total of 76 Dots is displayed until module successfully captures eye of the user. Calibration module ends and saves the image data to a file that user chooses at the beginning.

3.2. Gaze Prediction

After calibration module ends, Data collection module feeds the prediction with 61 different images based on locations. By looking at file names prediction module can process the images based on their names which is actually screen locations. In figure 3.4 prediction module of eye tracking system has shown.

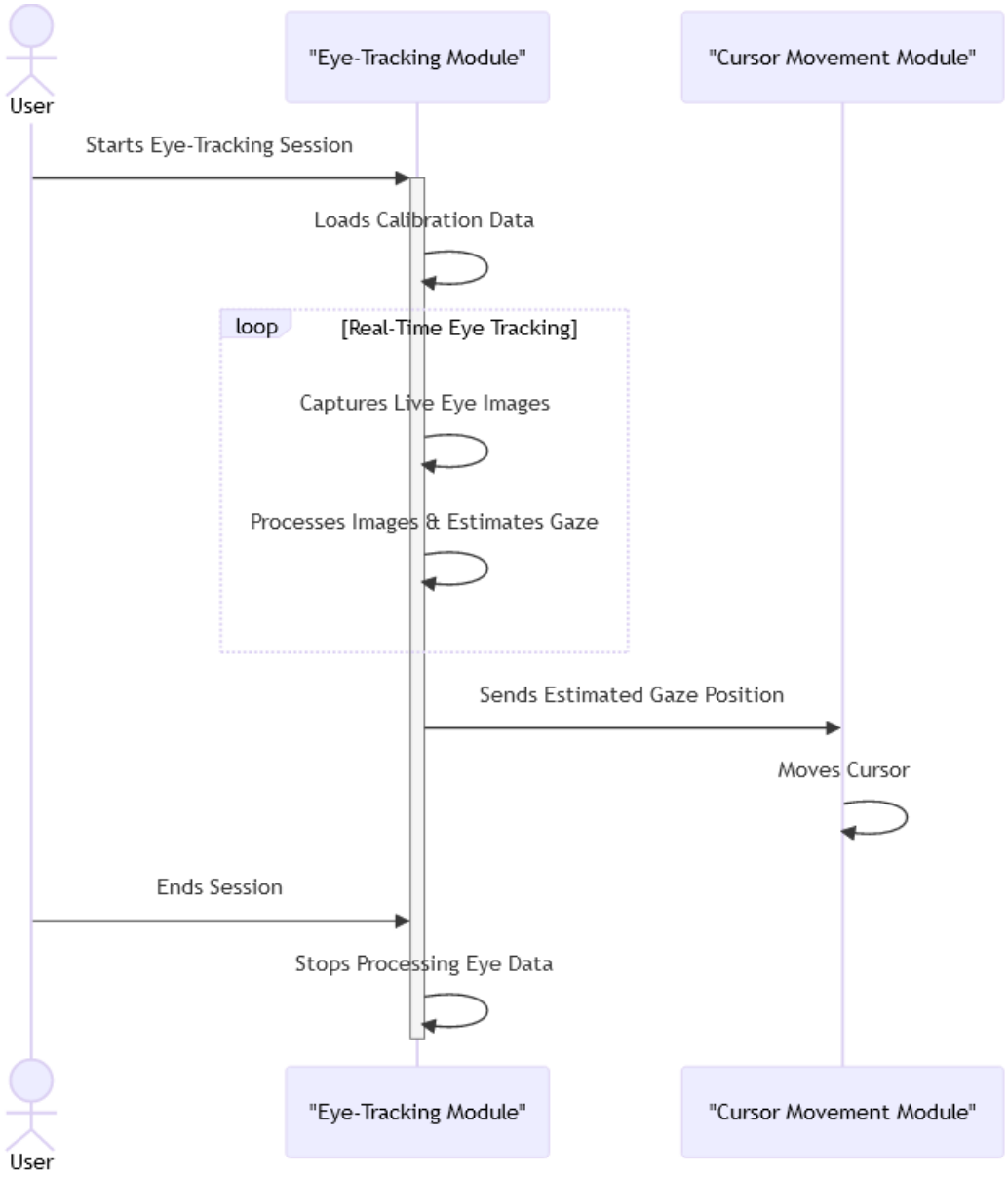

Figure 3.4. Sequence Diagram of Calibration

3.3. Blink Ratio

After gaze estimation module is set. Blinking module also runs and captures blinks from user. It is used for clicking function for cursor. Here is the flowchart of blinking module is shown below in Figure 3.5.

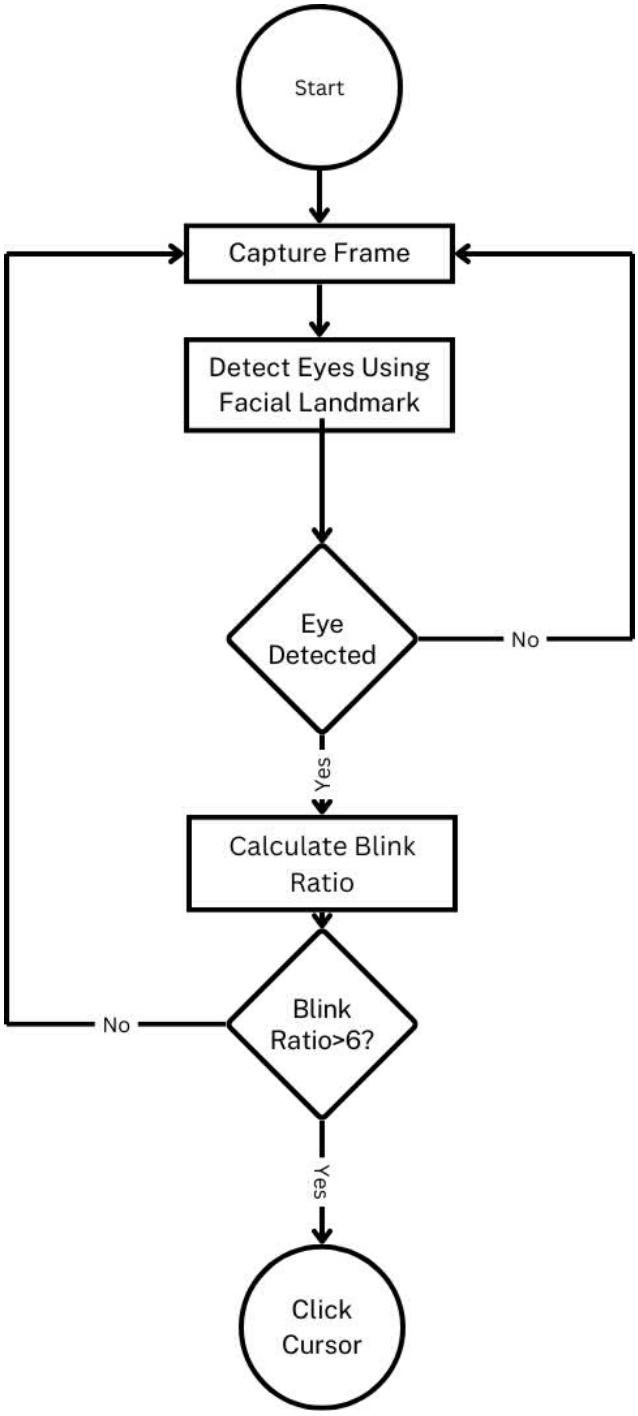

Figure 3.5. Flowchart of Blink detection process

3.3.1. Calculating Blink Ratio

The mid finding function calculates the midpoint between two points. It's used to find the center points of the top and bottom of the eye, which are essential for calculating the vertical eye aspect. Blink rate function calculates the eye aspect ratio (EAR) for blink detection. It determines the distance between the horizontal and vertical points around the eye. A significant change in this ratio indicates a blink. The EAR is typically lower when the eye is closed (during a blink). Image shown on Figure 3.6 displays a visual representation of lines that used in calculations.

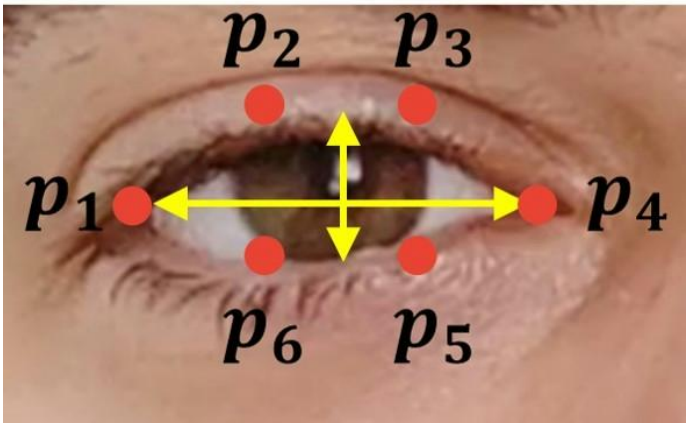

Figure 3.6. Eye Landmark Points

Horizontal line is a straight line that will be used in Eye Ratio calculation. Horizontal line is calculated using:

$$\text{Horizontal Distance} = |p_4 - p_1| \tag{3.1}$$

After calculating horizontal distance, the midpoints of vertical line have been calculated using:

$$\text{TopCenter} = \frac{p_2 + p_3}{2} \tag{3.2}$$

$$\text{BottomCenter} = \frac{p_5 + p_6}{2} \tag{3.3}$$

Since there are total of 6 landmarks on eye, p2 and p3 cannot be used individually therefore midpoints of that two points will be considered in the calculations.

After Calculating the midpoints of top and bottom of the eye, a straight line has been drawn and vertical distance has a variable rate of length that calculated each second. Vertical distance is calculated by using:

$$\text{Vertical Distance} = \frac{|\text{TopCenter} - \text{BottomCenter}|}{2} \tag{3.4}$$

Knowing that this point shows the location of eye in format of x and y. Calculation based on this is made by hypot function in Python. Hypot function returns the square root of the sum of square of x and y.

$$\text{hypot}(x_1, x_2, \ldots, x_n) = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2} \tag{3.5}$$

The straight line of horizontal distance is generally constant, but the vertical distance is changing when blinking or depends on the tiredness of person. Using the equation below, this proposition was applied to calculate the eye ratio.

$$\text{aspectEyeRatio} = \frac{|\text{Horizontal Distance}|}{|\text{Vertical Distance}|} \tag{3.6}$$

Knowing that this equation will be used twice for both left and right eye. The landmark points for two eyes have been given to function to calculate. As a result, we get two separate eye ratios for left and right eye. Each person has a different sample of blinking ratio, and this may change between the eyes. Blink ratio variable has been calculated by using:

$$\text{BlinkRatio} = \frac{\text{LeftEyeRatio} + \text{RightEyeRatio}}{2} \tag{3.7}$$

Since there is an alternative keyboard implementation, there is two other keyboard modules that can be used by users. Other than main implementation, dual type keyboard and full-size keyboard is using eye gaze ratio calculation to decide if person is looking left or right. In Figure 3.7 flowchart for alternative keyboard has been shown.

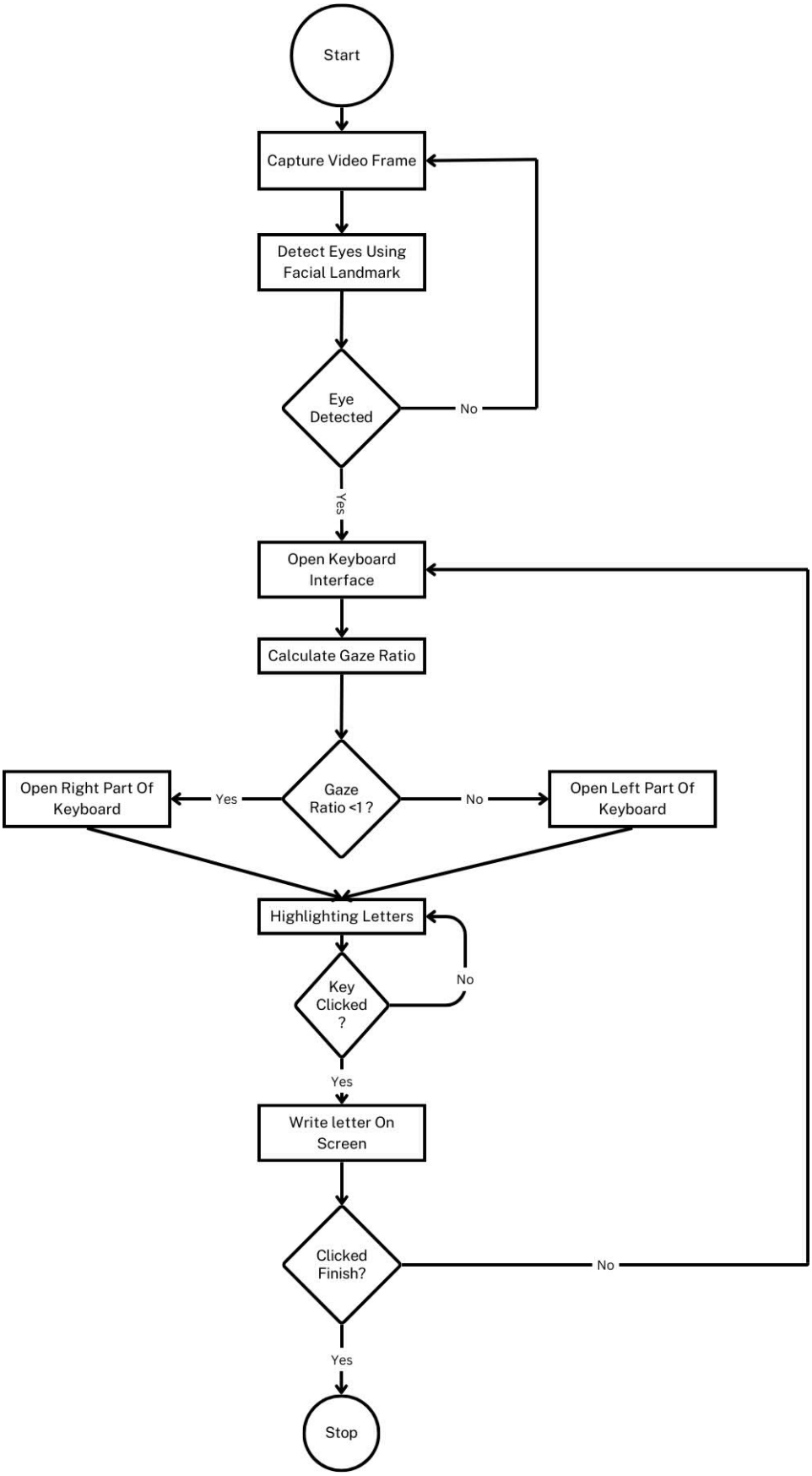

Figure 3.7. Eye Landmark Points

3.4. Gaze Estimation Model

The gaze estimation model is trained on a dataset of eye images to predict the on-screen gaze point. Using convolutional and dense layers, the model learns to extract features from the eyes and output coordinates that represent where a user is looking. Process begins with data collection, after all data is set, model definition and model training is made during gaze estimation. Finally, the model is integrated into an application where it controls the cursor in real-time, enhancing user interaction through eye movements.

3.4.1. Data Collection and Preprocessing

The first step is to gather a dataset of eye images paired with the corresponding screen coordinates that indicate where the person was looking when each image was captured. Data collection includes resizing images to a standard size (like 32x32 pixels), normalizing pixel values to the range [0, 1] by dividing by 255, and converts images to gray scale as shown in Figure 3.8.

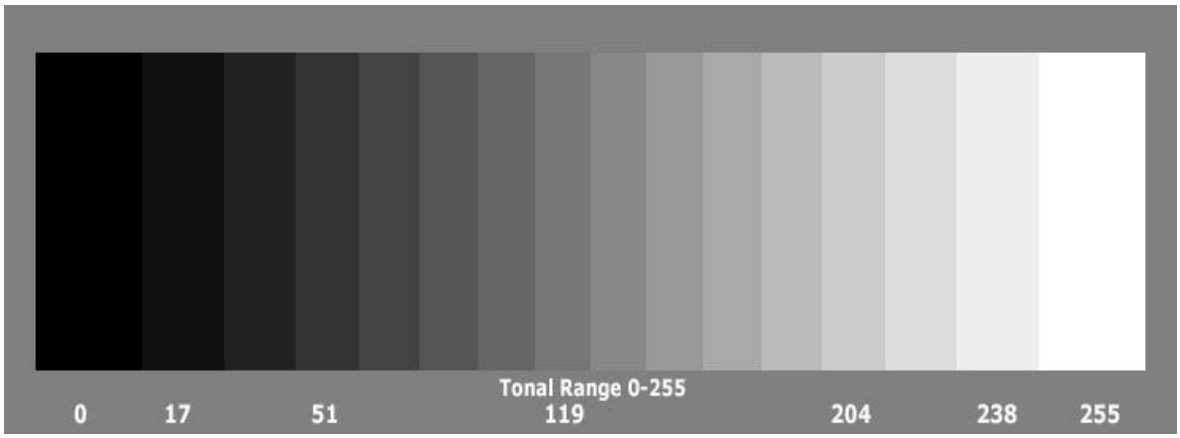

Figure 3.8. RGB image to Grayscale [9]

3.4.2. Prediction Process

The prediction model is a crucial component of the eye-tracking system, responsible for predicting where the user is looking on the screen. This model integrates computer vision techniques and machine learning algorithms to translate eye movement data into accurate screen coordinates.

Prediction process begins with loading cascade classifier which is located on the directory. Haar Cascade classifiers are a powerful tool in computer vision for rapidly detecting objects like eyes, an essential step in many eye-tracking systems. Their speed and efficiency make them well-suited for real-time applications, although they may need to be combined with other techniques for optimal performance under varied conditions.

Same with the Calibration process, calibration model normalize and scan functions take place in prediction phase too. Normalize function is used for standardizing the pixel values of images, crucial for consistent image processing. The scan function captures a video frame and converts it to grayscale. It then uses the cascade classifier to detect the eyes within the frame. The grayscale conversion enhances contrast, making it easier to detect features like the eyes.

Once the eyes are detected, the function processes each eye by cropping, resizing to a predefined size (32x32 pixels) as shown in Figure 4.3, normalizing, and further cropping to refine the eye image. The processed images of both eyes are concatenated horizontally and converted to an 8-bit format. This concatenated image likely serves as input for the gaze prediction model.

3.4.3. Model Definition and Compilation

The system uses a deep learning model using Keras, a high-level neural networks API. This model is a convolutional neural network (CNN) as shown in Figure 3.9, typically used for image processing tasks.

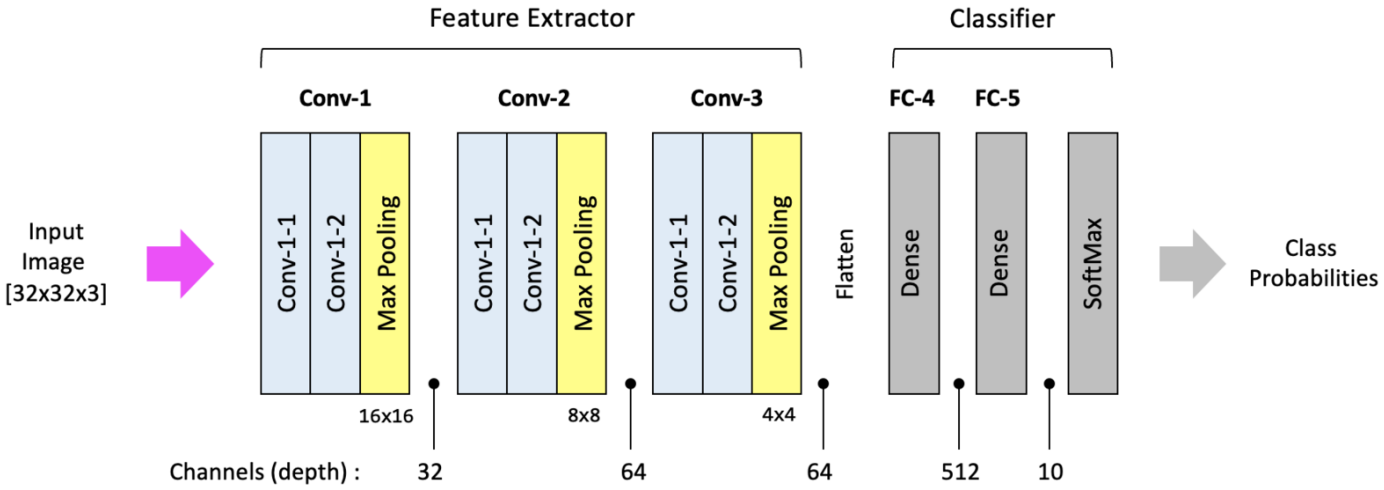

Figure 3.9. Keras CNN Implementation [10]

The model consists of several layers, including two convolutional layers (Conv2D) and dense (fully connected) layers. The convolutional layers are designed to extract features from the input images, while the dense layers are used for making predictions.

After the convolutional layers, a Flatten layer is used to convert the 2D feature maps into a 1D feature vector. Following the Flatten layer, one or more Dense layers are used for further processing and to make final predictions. The last Dense layer's size corresponds to the output's dimension, which, for this task, is likely the x and y coordinates on the screen.

Before training, the model must be compiled, which configures the model for training. This step involves specifying the optimizer, loss function, and metrics for evaluation.

Optimizer algorithm will adjust the weights of the network to minimize the loss, In this project Adam optimizer is used. Loss Function algorithm used for a regression task like predicting x,y coordinates, a mean squared error (MSE) is used on the optimization.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.8}$$

3.4.4. Model Training

The model is trained over a specified number of epochs (200 in this case) using the fit method. Training involves adjusting the model's weights based on the input data (X - processed eye images) and target data (Y - corresponding screen coordinates).

This process enables the model to learn how to predict the cursor's screen position based on eye image data. Summarization of Sequential model of gaze estimation is shown in Figure 3.10

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 5, 21, 32)	896
conv2d_1 (Conv2D)	(None, 2, 10, 64)	8,256
flatten (Flatten)	(None, 1280)	0
dense (Dense)	(None, 32)	40,992
dense_1 (Dense)	(None, 2)	66

Total params: 50,210 (196.13 KB)
Trainable params: 50,210 (196.13 KB)
Non-trainable params: 0 (0.00 B)

Figure 3.10. Summarization of Sequential model

4. IMPLEMENTATION

To properly detect and interpret eye movements in real-time, advanced machine learning algorithms and image processing techniques combined with OpenCV and Keras has been used. This ensures a smooth user experience, mimicking natural eye behavior. Additionally, the system's design incorporates a normalization process for eye images, crucial for the stable performance of our tracking algorithms.

The system includes an eye-tracking interface that detects and follows the user's eye movements. This interface will serve as the primary input method, allowing users to select keys on a virtual keyboard without the need for physical activity. The virtual keyboard, with a user-friendly layout optimized for eye-tracking interaction, will be presented on the screen. All necessary characters and functions, such as letters, numbers, and basic punctuation, will be included, as well as specific commands for erasing text, clear text and enabling the text-to-speech functionality.

4.1. Calibration Process

A critical component of the system is its calibration process. This involves displaying a series of points (green dots) across the screen shown in Figure 4.1, which the user is instructed to look at. When the user focuses on a dot, a corresponding action (like blinking) records the eye position 4.3 This calibration is vital for customizing the system to each user's specific eye movement patterns.

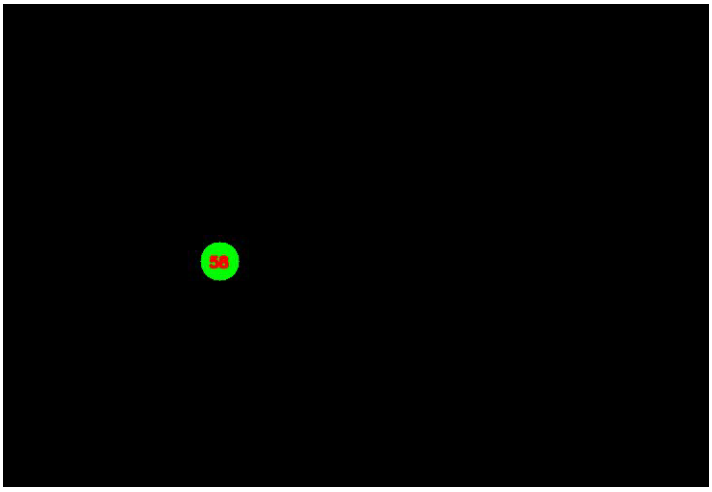

Figure 4.1. Calibration Process Dot Placement

As the first step, the system initializes the calibration process by generating 61 dots on the computer screen. These dots are strategically placed to cover the entire visual area of the screen, ensuring data collection across different gaze points. The placement and number of dots are critical. They are distributed to capture a wide range of eye movement patterns, accommodating users with diverse eye movement characteristics. Calibration starts with 6x6 dots extending all over the screen, after capturing eye positions, there is extra 25 dots for the keyboard positions. This extra calibration process ensures eye positions according to key positions is saved.

During the calibration phase, if the camera does not capture the eye position, it warns the user and asks him/her to repeat. The calibration phase is completed after the eye positions of all 61 All eye positions are named as x and y coordinates of the screen.

Normalizer function normalizes the pixel values of an image (or any array-like structure). It takes an array x as input, finds its minimum (minn) and maximum (maxx) values, and then normalizes the array elements to a range between 0 and 1. Normalization is a common preprocessing step in image processing and computer vision, helping to standardize data and often improving the performance of machine learning algorithms.

Scan function is responsible for capturing and processing eye images from a video feed, crucial for detecting and analyzing the user's eyes during calibration. Gray Scale function reads a frame from the video feed, then converts it to grayscale using OpenCV's cvtColor function. Grayscale images shown in Figure 4.2 are often used in computer vision tasks as they reduce complexity compared to full-color images.

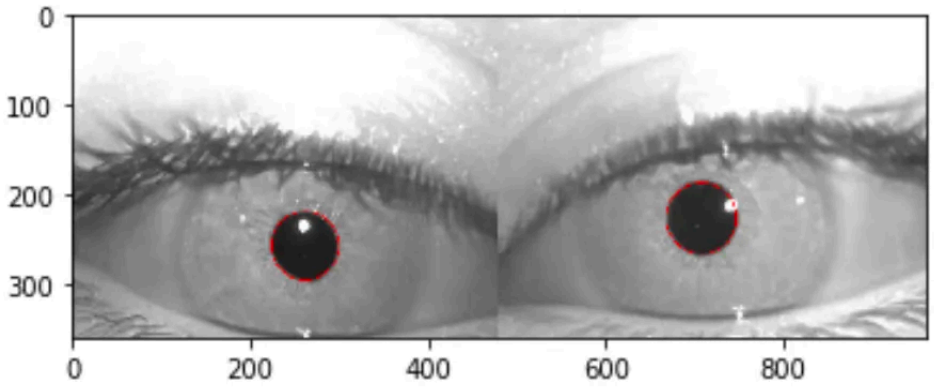

Figure 4.2. OpenCV Grayscale eye image

Grayscale images contain only intensity information, representing the brightness of pixels. In contrast, color images contain additional data for color channels (typically red, green, and blue). By converting images to grayscale, the amount of data processed is significantly reduced. This simplification leads to faster processing times, which is crucial in real-time applications like eye tracking.

System uses a cascade classifier to detect eyes within the grayscale frame. The parameters 1.3 and 10 are scale factors and minNeighbours, respectively, which control the detection process. If exactly two eyes are detected (as indicated by len function, the function enters a loop to process each detected eye. It crops the eye region from the original frame, resizes it to 32x32 pixels shown in Figure 4.3, and normalizes it using the previously defined normalize function.

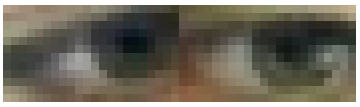

Figure 4.3. 32x32 pixels recorded eye position of left eye and right eye

If the eyes are not detected properly (i.e., not exactly two eyes detected), the function returns None. This could be a case where the system may prompt the user to adjust their position or the camera.

4.2. Smoothing Cursor Movement

To ensure that cursor movement is smooth, a history length of the last few predicted positions is maintained using a deque. Deque function creates a double ended queue, which is an ordered collection of items to a list. Based on history length value which is determined by user, deque will only hold up that much predictions. Once its full, new items are append to the list. It is primarily used to store the most recent positions for the purpose of smoothing cursor movement and prevent it from jumping too far between frames. With this method user experience is more enhanced.

4.3. Main Loop for Cursor Movement

In an infinite loop, the system continuously captures and processes eye images and then uses the trained model to predict the gaze position. The predicted positions are normalized and scaled to the screen's width and height, and then smoothed and speed-limited for natural cursor movement. The cursor's position on the screen is updated using Pyautogui function, which moves the cursor to the predicted screen coordinates.

4.4. Blink Detection and Feedback

Dlib [11] is a useful tool for its high-quality face recognition and detection model which this system uses. Dlib's face detection is used in this method to identify faces in each frame of the video stream. This is the first step in locating the eyes for blink detection. After detecting faces, Dlib's shape predictor with the face landmarks [12] is necessary for detection process. This model is trained to identify 68 specific points (landmarks) on a face, including key points around the eyes.

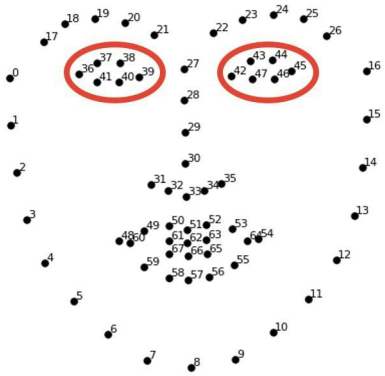

Figure 4.4. OpenCV Grayscale eye image

In section 3.3 Blink ratio has been calculated by mid point formula. After calculations, eye shape points function extracts the coordinates of landmarks around the eyes in Figure 4.6. It separates the landmarks for the left and right eyes, which are then used to draw polygons (representing the eye shape) on the frame. A ratio higher than a set threshold (6 in this case) indicates a blink. This threshold may change for different persons.

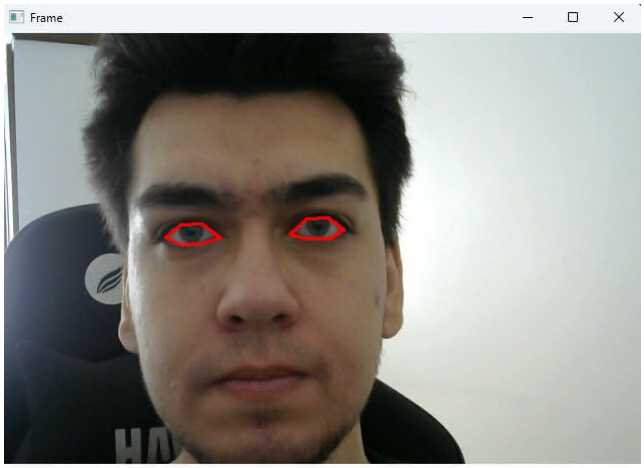

Figure 4.5. Different landmarks on both left and right eye

When a blink is detected, the system changes the color of the eye polygons from red to green as visual feedback is shown in Figure 4.6. This change makes it visually evident when a blink is registered. Blink functions includes a one-second pause after a blink is detected. This delay can be intended to avoid multiple detections for a single blink or to ensure that the blink is fully captured.

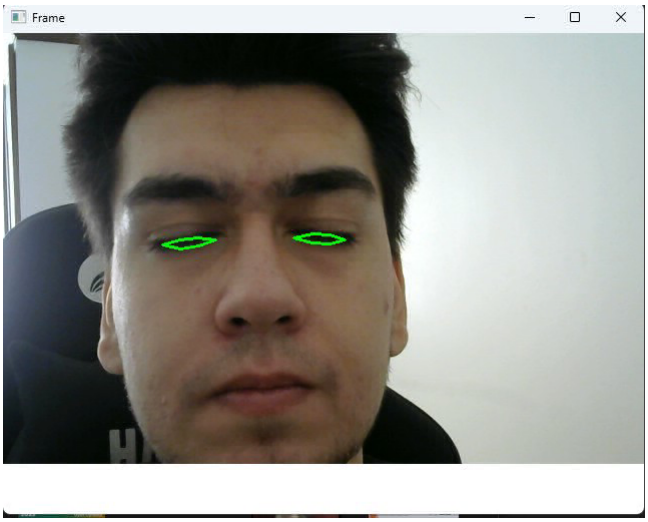

Figure 4.6. Different landmarks on both left and right eye

4.5. Virtual Keyboard Structure

The virtual keyboard that has been developed for text entry using eye blinking has been presented in Figure 4.7.

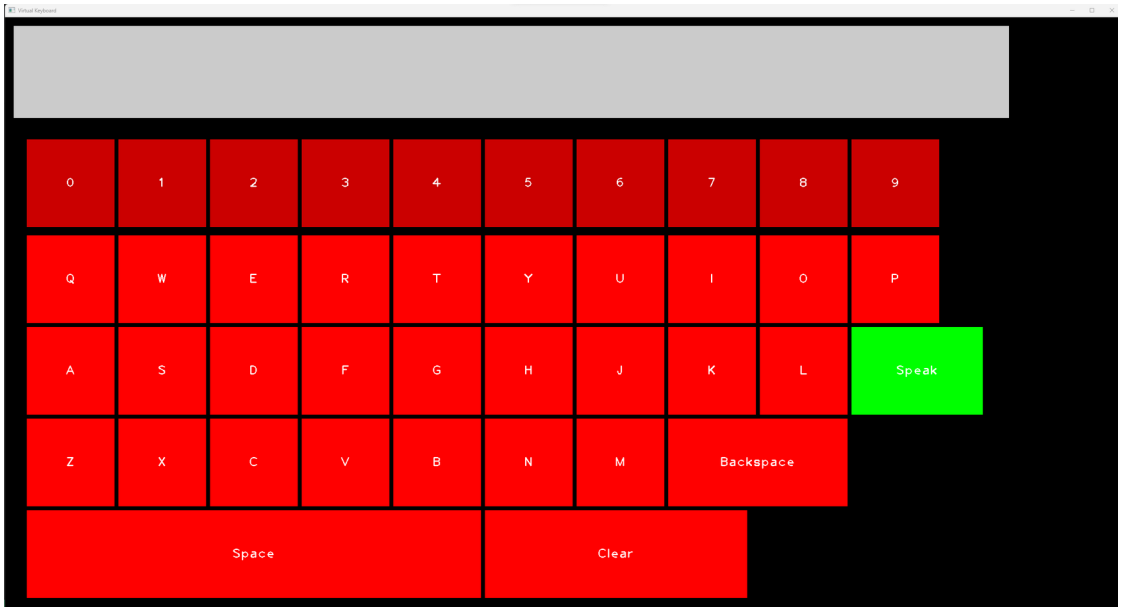

Figure 4.7. Visualization of main virtual keyboard

Virtual keyboard defines several global variables to track user inputs, such as mouse position, selected language, and typed text. These variables play a crucial role in the keyboard's functionality.

The Keyboard defines the layout of the virtual keyboard, including the positioning and labeling of each key. It illustrates the use of OpenCV to render the keyboard graphically, ensuring it is both functional and visually appealing. OpenCV is used to design and display the visual elements of the virtual keyboard. This includes drawing the keys as rectangles on the screen and labeling them with text. The rectangle and putting text functions are important in creating each key of the keyboard.

Language selection function creates a user interface in Figure 4.8 for language selection, allowing the user to choose between English and Turkish for the output. It leaves mouse event handling to capture the user's choice, demonstrating the keyboard's interactive capabilities.

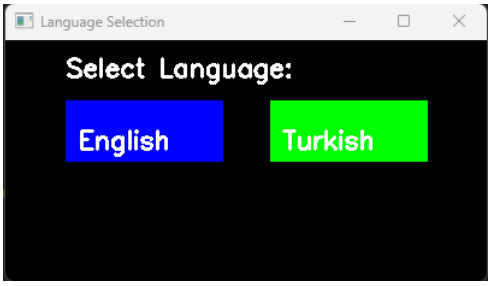

Figure 4.8. Language selection interface

Speaking function is a key component of the Keyboard, converting the typed text into audible speech in the selected language using gTTS. It demonstrates the keyboard's ability to bridge communication gaps for users with speech impairments. Because gTTS library does not support Turkish Language voice support, system has unable to sound Turkish in this implementation.

Mouse interaction handler is central to the Keyboard's usefulness. It handles mouse events to detect when the user clicks on a key and updates the typed text variable accordingly. This feature is fundamental to the keyboard's utility, allowing users to type and interact with the keyboard.

4.6. User interface and Text Processing

The virtual keyboard not only displays the text typed by the user but also dynamically generates spelling suggestions show in Figure 4.9 using the enchant library. This functionality enhances the keyboard's usability, particularly for users who may rely on spelling assistance. User may be able to choose which word they meant to.

Figure 4.9. Spell Corrector

4.7. Accessibility Features and Visual Feedback

The virtual keyboard has visual feedback mechanisms, such as highlighting keys when hovered over or selected. This feature not only enhances the keyboard's aesthetics but also improves its accessibility. The inclusion of a 'Speak' key, which activates the text-to-speech function, underscores the Keyboards commitment to accessibility. It provides a voice output for users who are unable to speak.

4.8. Alternative Keyboard Implementations

Main virtual keyboard could not be preferable by users, so there are 2 other virtual keyboard alternatives that users may want to use. Significant amount of people tends to choose comfort and ergonomic writing to speed.

In second alternative of keyboard, there is a consecutive letter highlighting system is used. Letters are highlighted specific amount of time (in this case 1 second) and if user wants to choose the highlighted letter, like same as the others blinks eyes. Then it continuously moves to the end of the keyboard. There is a space and backspace option on keyboard.Virtual keyboard that has been developed as an alternative for text entry using eye blinking has been presented in Figure 4.10.

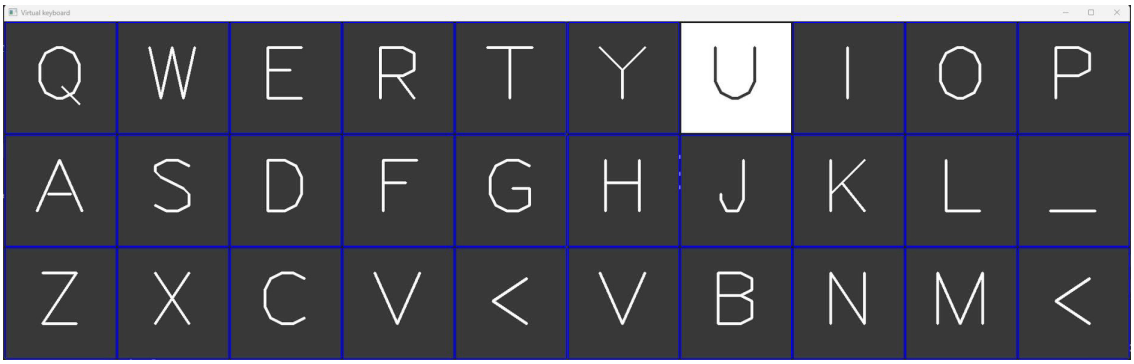

Figure 4.10. Alternative full-size keyboard implementation

Keyboard also has an external board that saves what has been written on.The output interface of alternative keyboard is shown in Figure 4.11 .

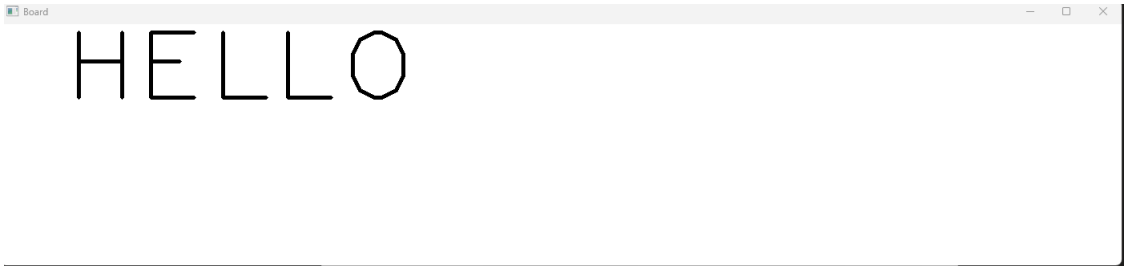

Figure 4.11. Output screen of alternative keyboard.

One other alternative is divided keyboard. It divides the keyboard into half. Left Side has approximately 15 keys and right side as well. With the divided design of the keyboard, it is much easier to choose which key user wants. This dual design is much faster than the full-size keyboard but cot comfortable as the other two. User will use much more blinking therefore it is much more exhausting.

Interface starts from choosing left or right side. The keyboard interface offers the user two options, left and right, the user chooses which side to type based on the letters of the word. Here is the visual of dual keyboard interface shown on 4.12

Figure 4.12. Alternative Dual keyboard design interface

If user choose to open left side of the keyboard then user should be looking left and interface show on Figure 4.13 will be opened. When interface appears, highlighting the letters begins from letter 'Q' until the end of the keyboard. If user want to get back to first interface then user should choose '<' to change the interface.

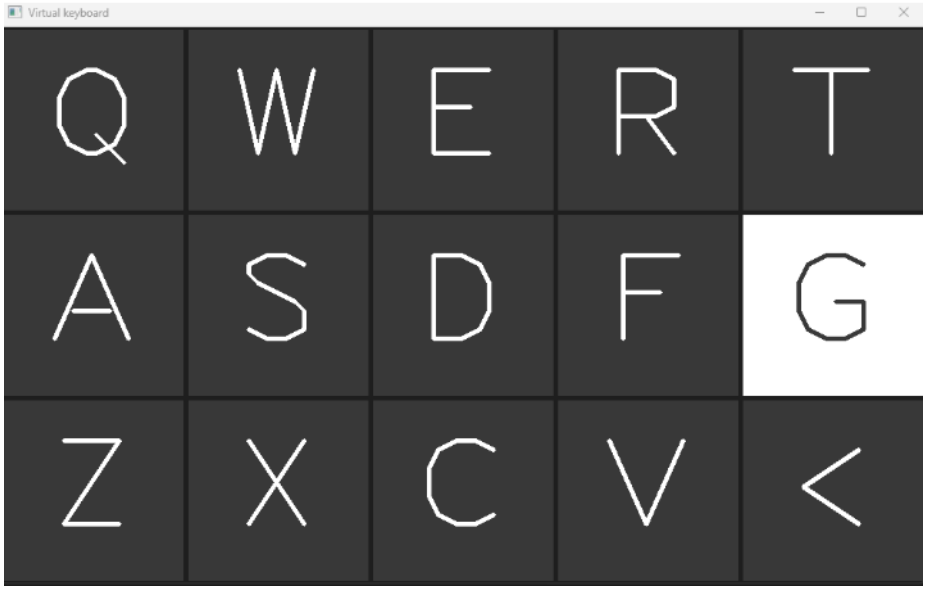

Figure 4.13. Left part of the dual keyboard design

4.9. Eye Gaze Estimation

As shown on the system uses 76 different landmarks focusing particularly on the eye regions to understand eye movements. The gaze direction is determined by analyzing each 6 landmarks on human eye.

After extracting eye region shown in Figure 4.14, there is a step to create a mask that isolates the eye region in 4.15. This is done by focusing specifically on the eye and eliminate distractions from surrounding areas. The gaze ratio is basically calculated by comparing the white (sclera region) and colored part(iris) of the eye in the horizontal direction.

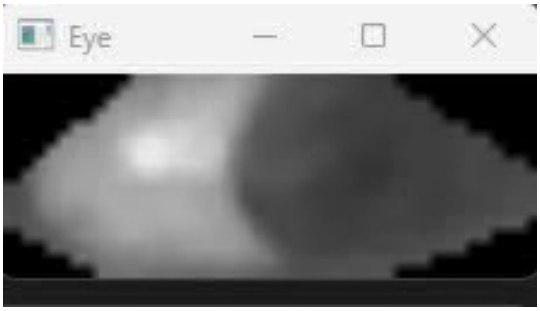

Figure 4.14. Extracted Eye Region

The process involves dividing the eye region into two parts and calculates the amount of white region in each half. The gaze ratio is the white region in the two parts of the eye. If more white is detected on the right, it indicates the person is looking left and vice versa. If white balance is equal in each side of the eye, that means user is looking straight which is center.

Figure 4.15. Eye masked with black and white

Before determining gaze direction, binary threshold is calculated while capturing eye. Pixel intensity above 70 are set to 255 which is white, and others are set to 0 which is black. This helps distinguishing the white and dark (iris and pupil) parts of the eye.

If there are no white pixels in the left half, the gaze ratio is set to 1, indicating the gaze is probably to the right, the right looking gaze estimation is shown in Figure 4.16 . If there are no white pixels in the right half, the gaze ratio is set to 5, indicating the gaze is probably to the left. Otherwise, the gaze ratio is the ratio of white pixels in the left half to the right half.

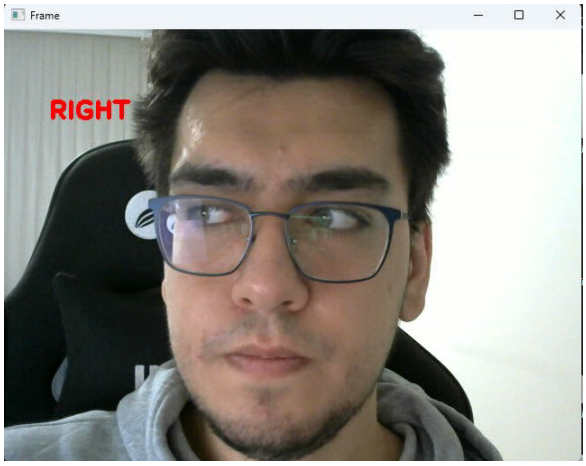

Figure 4.16. Right looking eye gaze estimation

5. TEST AND RESULTS

The tests were carried out by testing 3 different keyboard implementations with the participation of a total of 9 people. The age range of those who take this test is 23-27.

There is total of 6 different inputs tested which is:

- Hello
- Scared
- Money
- Name Surname
- Eye Tracker
- This is my first experiment

These words are chosen on purpose. Hello is a basic word that everyone should do first. After that “scared” is a important word that all of the letters are left side of the keyboard, this words is one of best words that indicates the difference between 3 keyboards. Since dual keyboard design offers left and right side of the keyboard each time and keyboard is working in consecutive highlighted mode it is easier to capture letters rather than main keyboard implementation.

On main keyboard, since the letters are so close between them, capturing the right letter is much more hard than the other two keyboards. Money is a word that has a letter on every side of the keyboard which is left, right bottom and upper side of the keyboard. A broad distribution like this has an advantage for main virtual keyboard design since it is easier to capture the right letters by looking. On the other hand, full-size and dual-size keyboards are so hard to write with this kind of words.

5.1. Results of Main Keyboard Design

Results of 9 participants on 6 different inputs shown in Figure 5.1 and 5.2

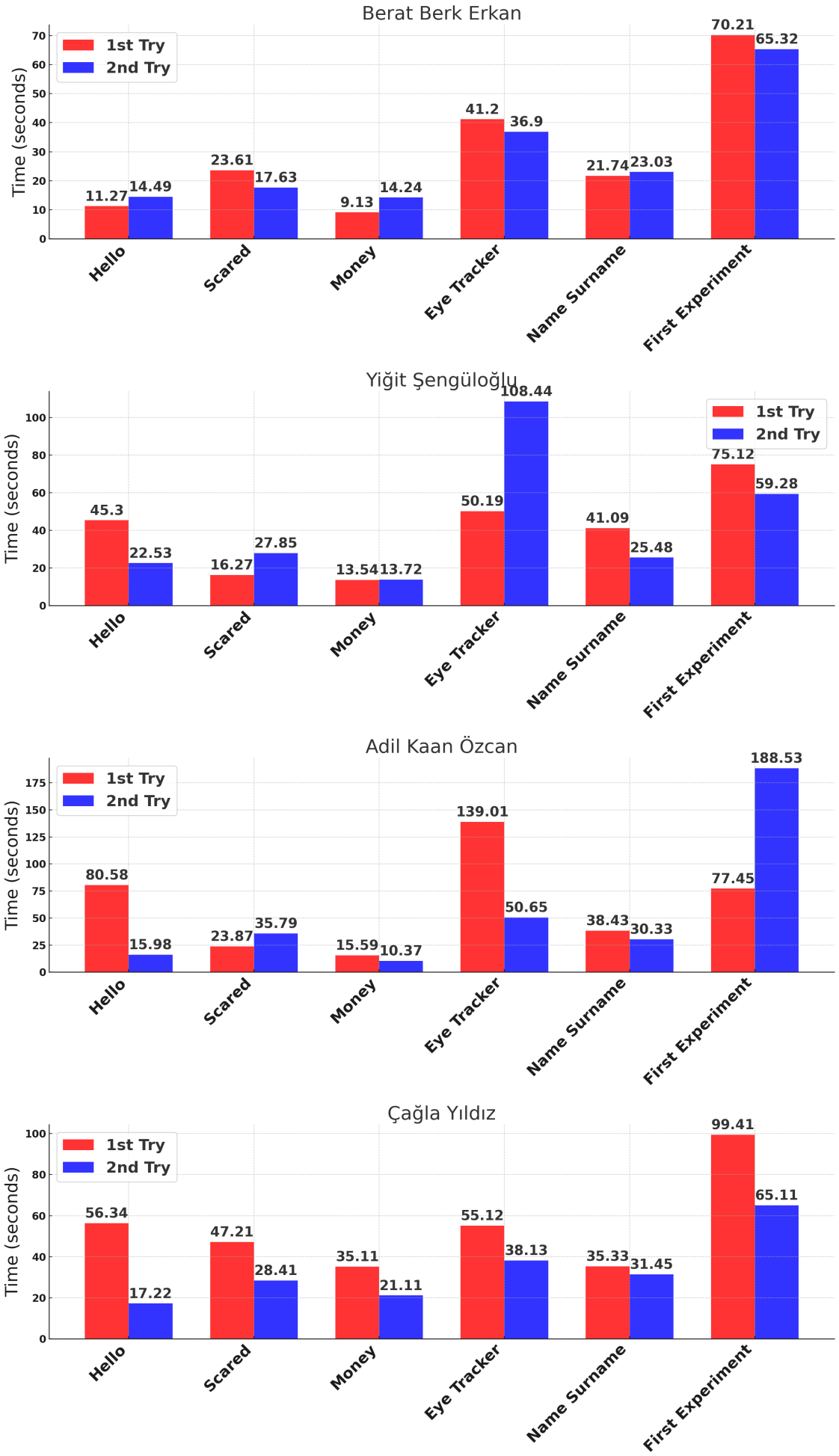

Figure 5.1. Test results of first 4 participants on 6 different words

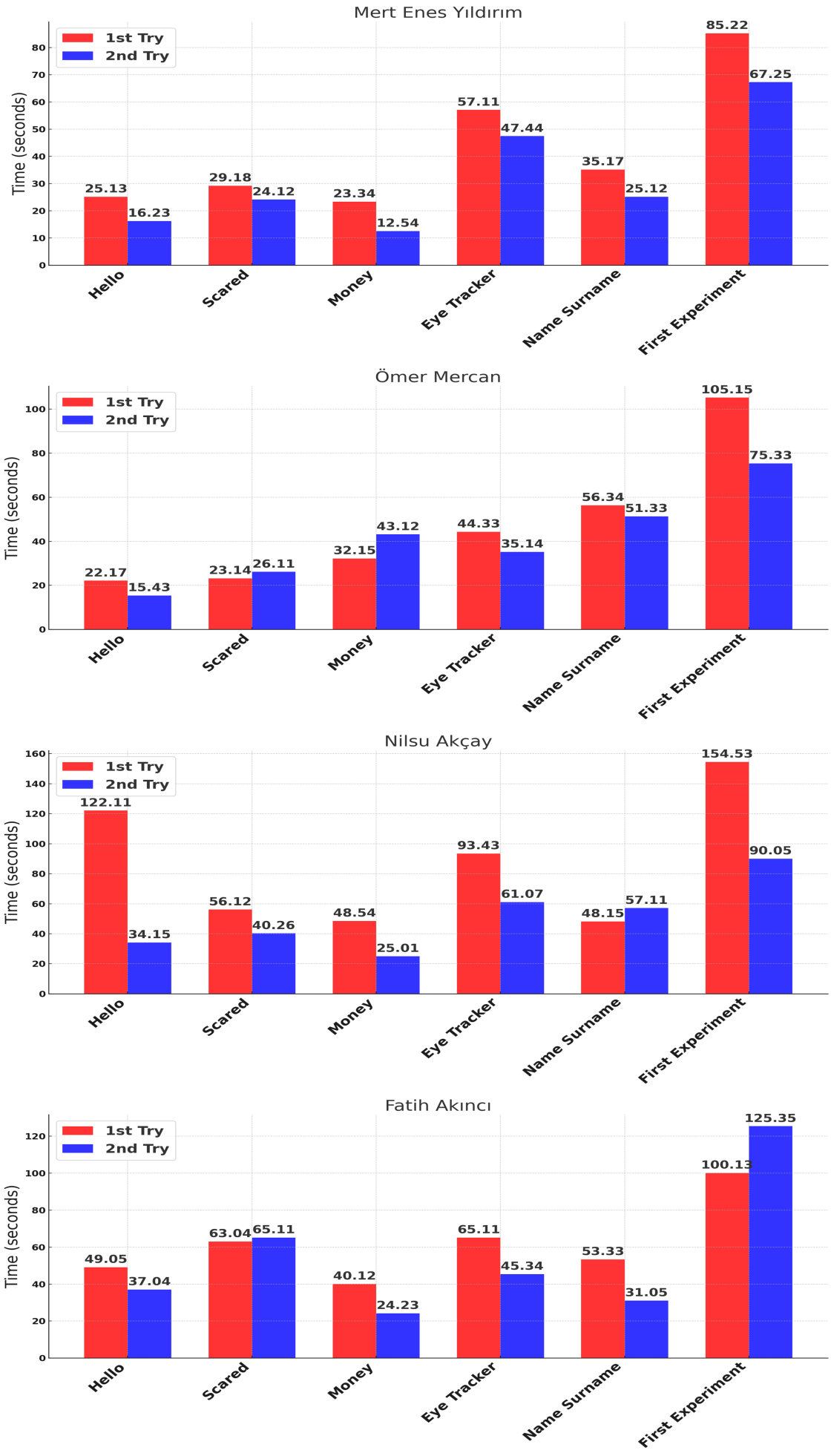

Figure 5.2. Test results of last 4 participants on 6 different words

There is a clear indication that most of the time participants is more successful on their second try which is expected. This is indicative of learning effects increased familiarity with tasks. For instance, almost every participant recorded a shorter duration in the “Hello” word on their second try, experiment suggests that they became more efficient or comfortable with the task.

Each participant displayed unique strengths and weaknesses. For instance, “Eye Tracker” and “Hello” words are done quite good by some participants while others showed more proficiency in writing their names and surnames. Such results could be due to individual skill sets, prior experience or personal interest in the task.These charts could be used in various settings. These calculations are made in the Laboratory of Yeditepe University shown in Figure 5.3 and 5.4, making experiment inside laboratory makes participant more focused.

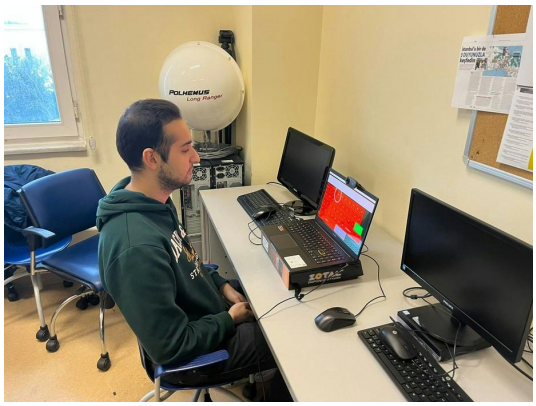

Figure 5.3. Participant 1 Experimenting

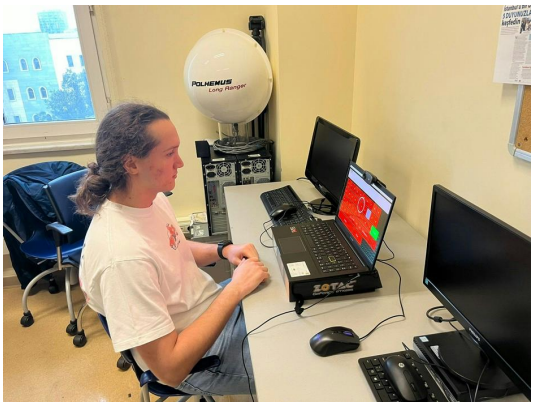

Figure 5.4. Participant 2 Experimenting

5.2. WPM Calculation of Participants

Words Per Minute (WPM) is a calculation of words processed in a minute. It is often used measuring typing speed and reading. In these experiments words are big scale for calculating, instead of words, calculations are made by looking letters per minute. With this calculation there will be more accurate results.

Based on WPM equation, WPM performances of 9 Participants shown in Figure 5.5

$$WPM = \left(\frac{\text{TotalLettersWritten}}{\text{Seconds}} \right) \times 60 \tag{5.1}$$

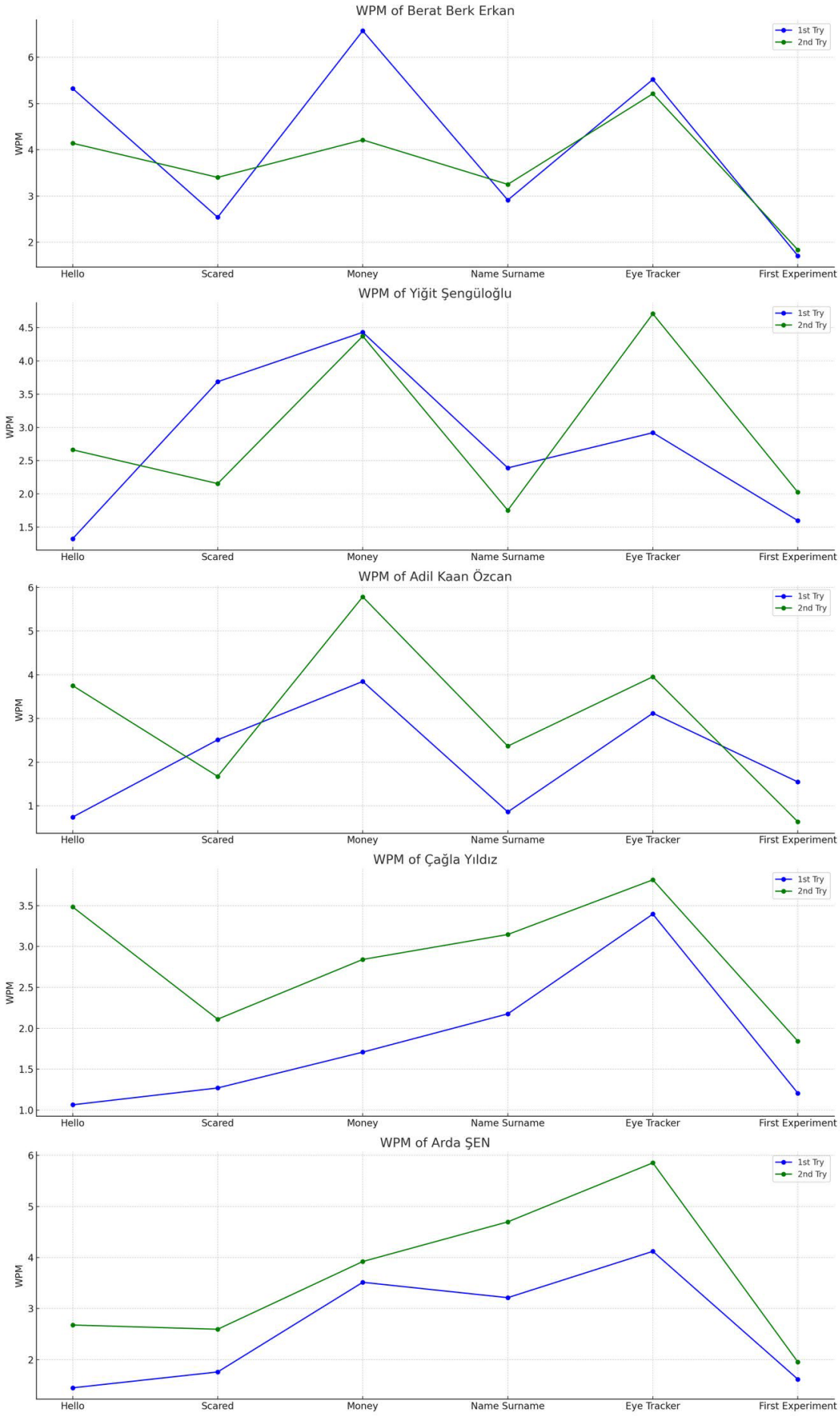

Figure 5.5. WPM Performance of first 5 participants shown in line graph

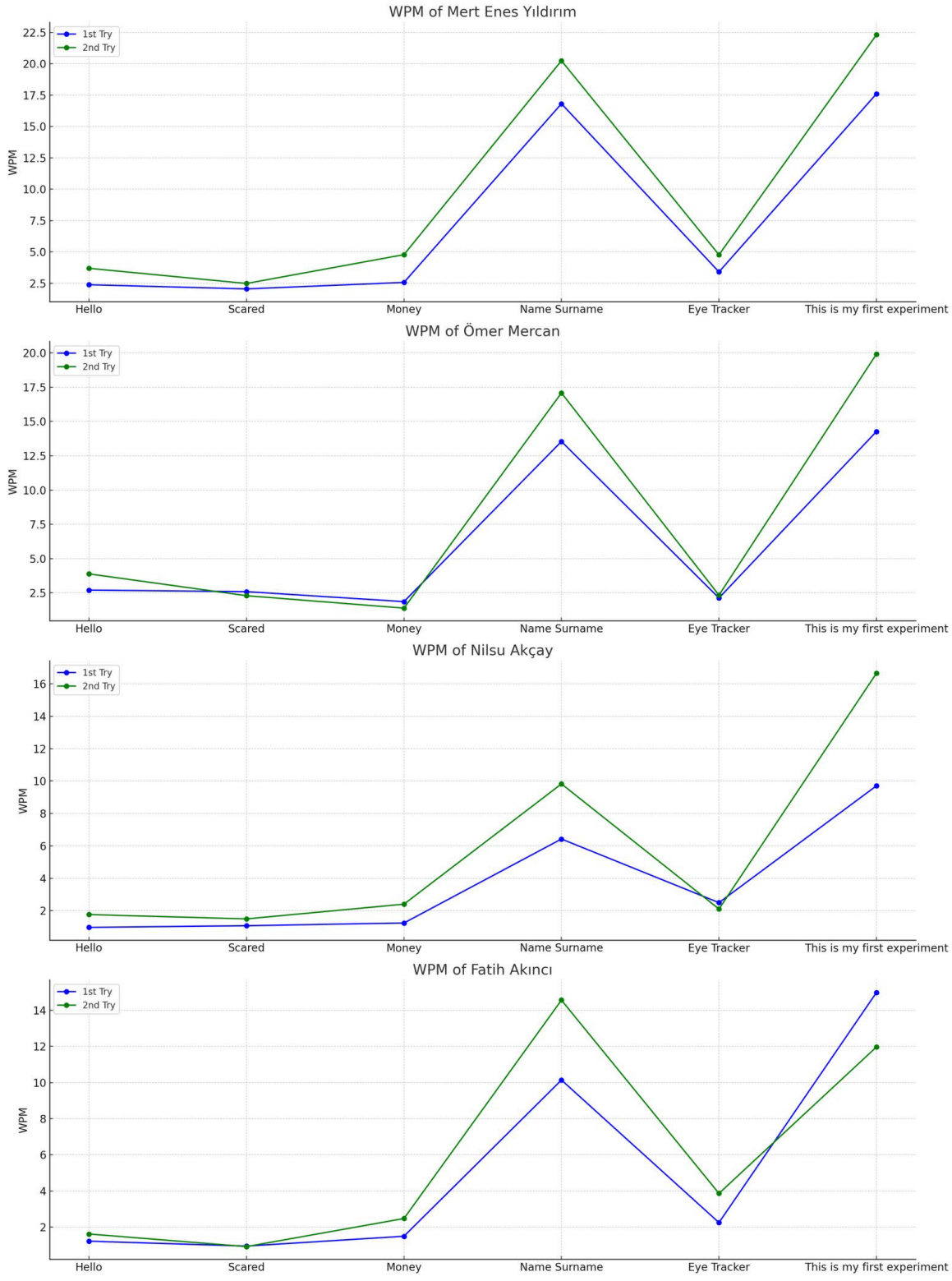

Figure 5.6. WPM Performance of last 4 participants shown in line graph

A notable situation is the improvement in WPM scores in the second attempt for many participants across various tasks. This improvement can be credited to learning effect, where participants become more familiar to the task, leading to increased speed and efficiency. This is particularly evident in tasks like ”This is my first experiment,” where the significant increase in WPM reflects a higher comfort level and proficiency gained after the first attempt.

The complexity of a task significantly affects WPM. For example, ”Name Surname,” which involves typing out full names, shows varying WPM rates based on the length and complexity of each name. Participants with shorter names tend to have higher WPM due to fewer characters needing to be typed. This task-specific variation highlights the importance of considering the nature of the task when evaluating typing performance.

5.3. Results of Dual Keyboard Design

Since there are 2 other keyboard implementations, participants also tried dual-size keyboard and full-size highlighted keyboard.

In dual-size keyboard, it is much harder to complete same task with less time. Because participants need to specify which side of keyboard they want, this makes writing much harder rather than main virtual keyboard implementation. The results are shown in Figure 5.7 and 5.8

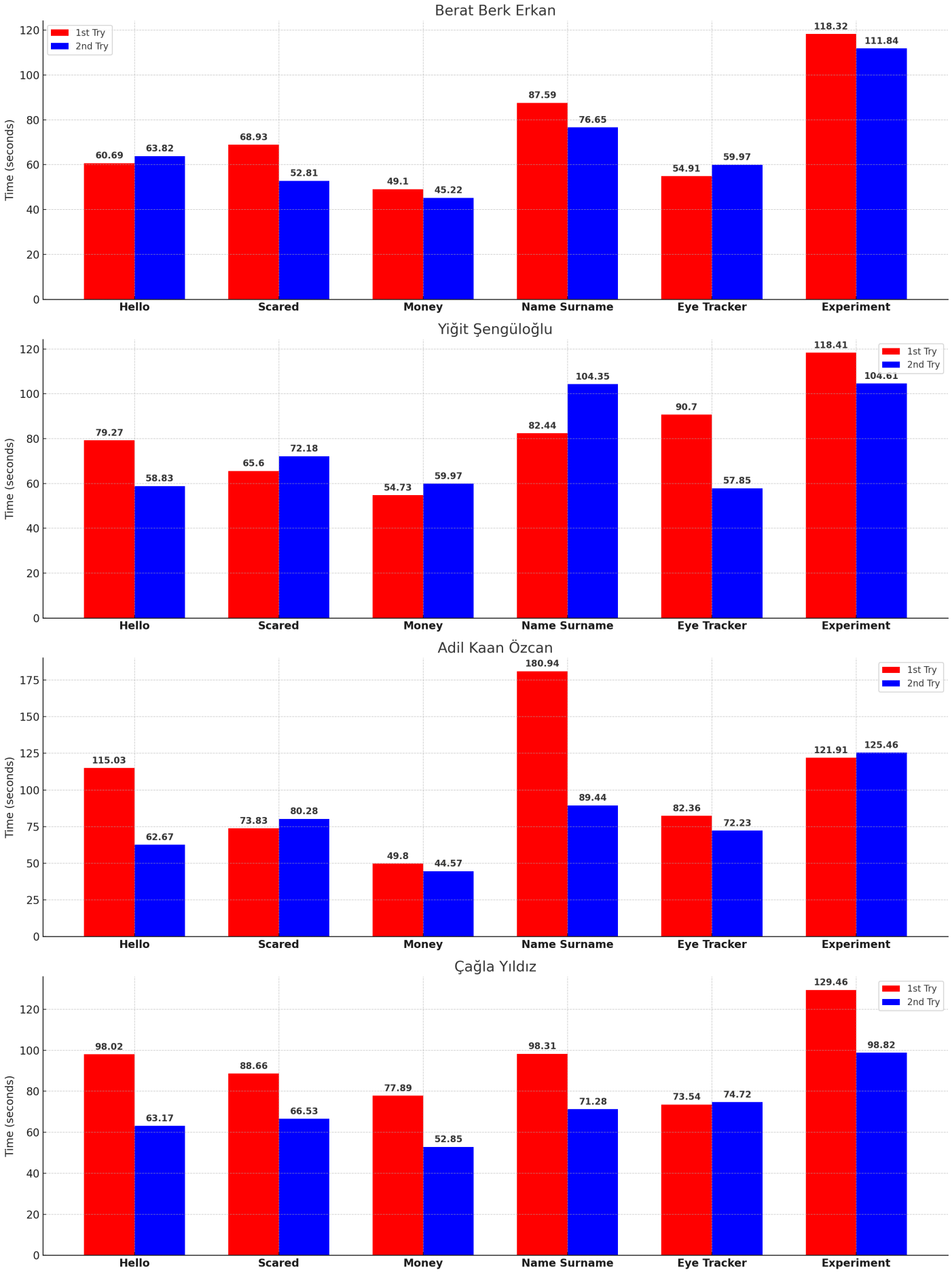

Figure 5.7. Test results of first 4 participants on 6 different words in dual keyboard.

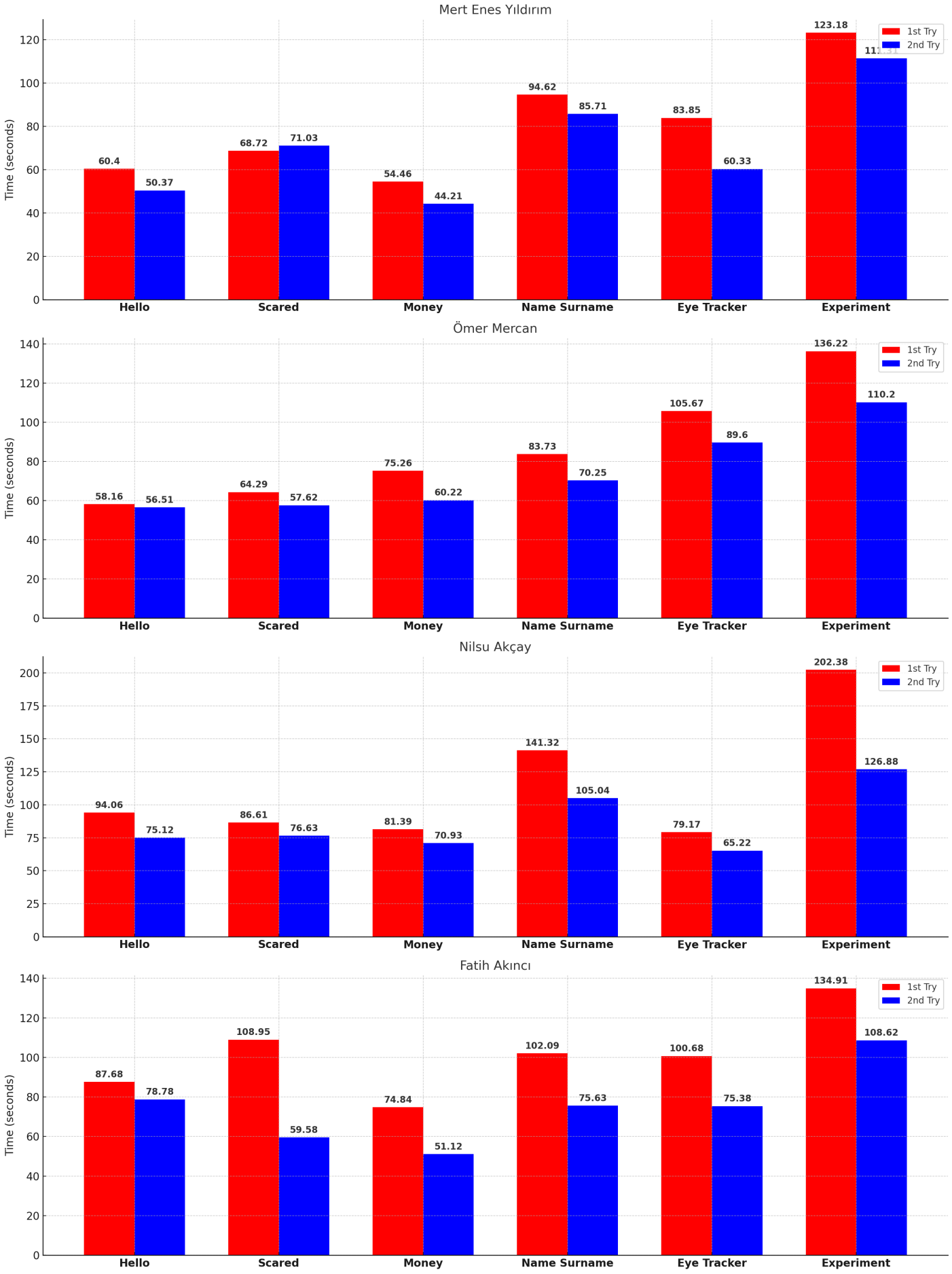

Figure 5.8. Test results of last 4 participants on 6 different words in dual keyboard.

Comparisons of three keyboard shown in Table 5.1.

Participant	Main Keyboard	Dual Keyboard	Fullsize Keyboard
Adil Kaan Özcan	55.28s	68.28s	75.28s
Arda Şen	67.92s	84.92s	89.92s
Berat Berk Erkan	28.60s	43.60s	38.60s
Fatih Akıncı	50.44s	85.44s	91.44s
Mert Enes Yıldırım	32.12s	47.12s	52.12s
Nilsu Akçay	69.21s	84.21s	78.21s
Yiğit Şengüloğlu	36.21s	51.21s	56.21s
Çağla Yıldız	33.57s	47.57s	43.57s
Ömer Mercan	41.08s	57.08s	51.08s

Table 5.1. Average Typing Speeds of Participants

5.4. Comparison of Three Keyboard Designs

It can be easily seen that, Main keyboard which has the eye tracking with gaze estimation performs much better performance than the other two keyboard. Even though some words can be complicated through main keyboard, still outperforms dual and full size keyboard.

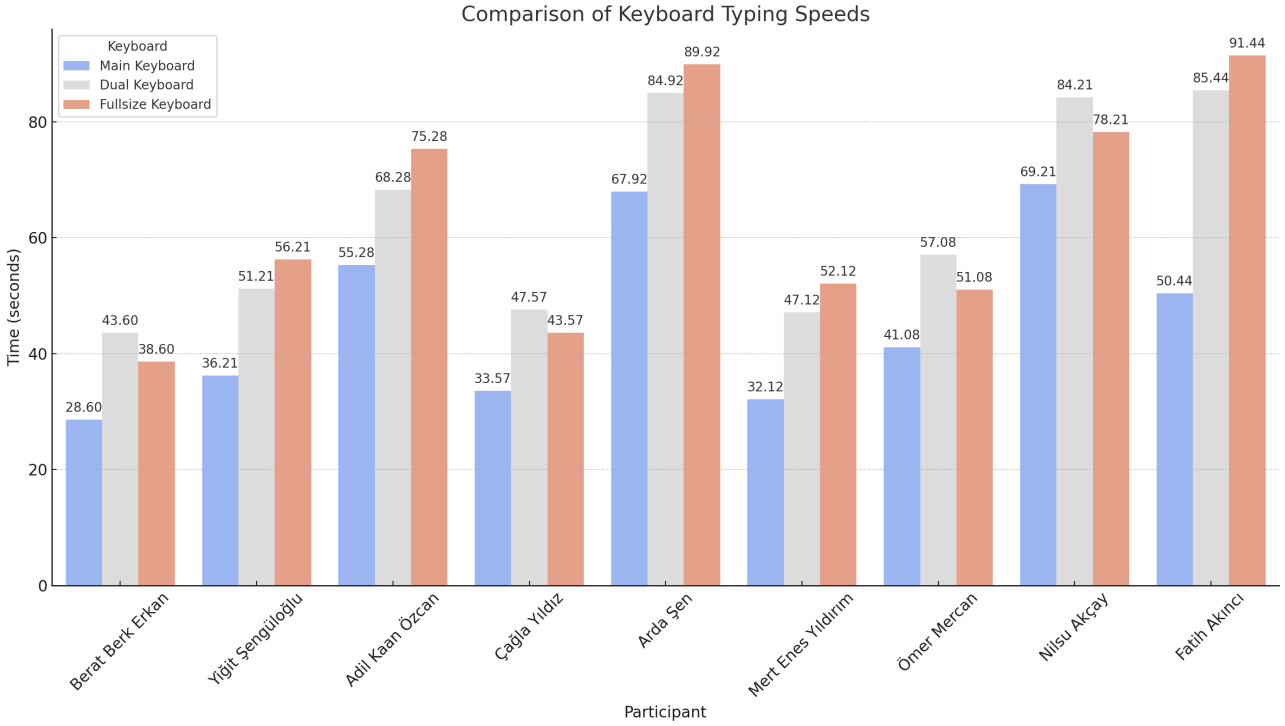

Figure 5.9. Average time to Write on 3 different keyboard

5.5. Comparison of Different Monitor Size

Experiment has been conducted by 4 different monitor sizes: 14 inch , 15.6 inch, 24 inch, 27 inch. Performance calculated by errors that has been made while writing the predetermined word. The performance of eye tracker is shown in Figure 5.10.

It can be easily seen that the number of errors tends to decrease as the monitor size increases. This suggests that a larger monitor size may contribute to a reduction in errors, potentially due to better visibility. So it is recommended that users should prefer large size monitors in order to get the full performance of eye tracker.

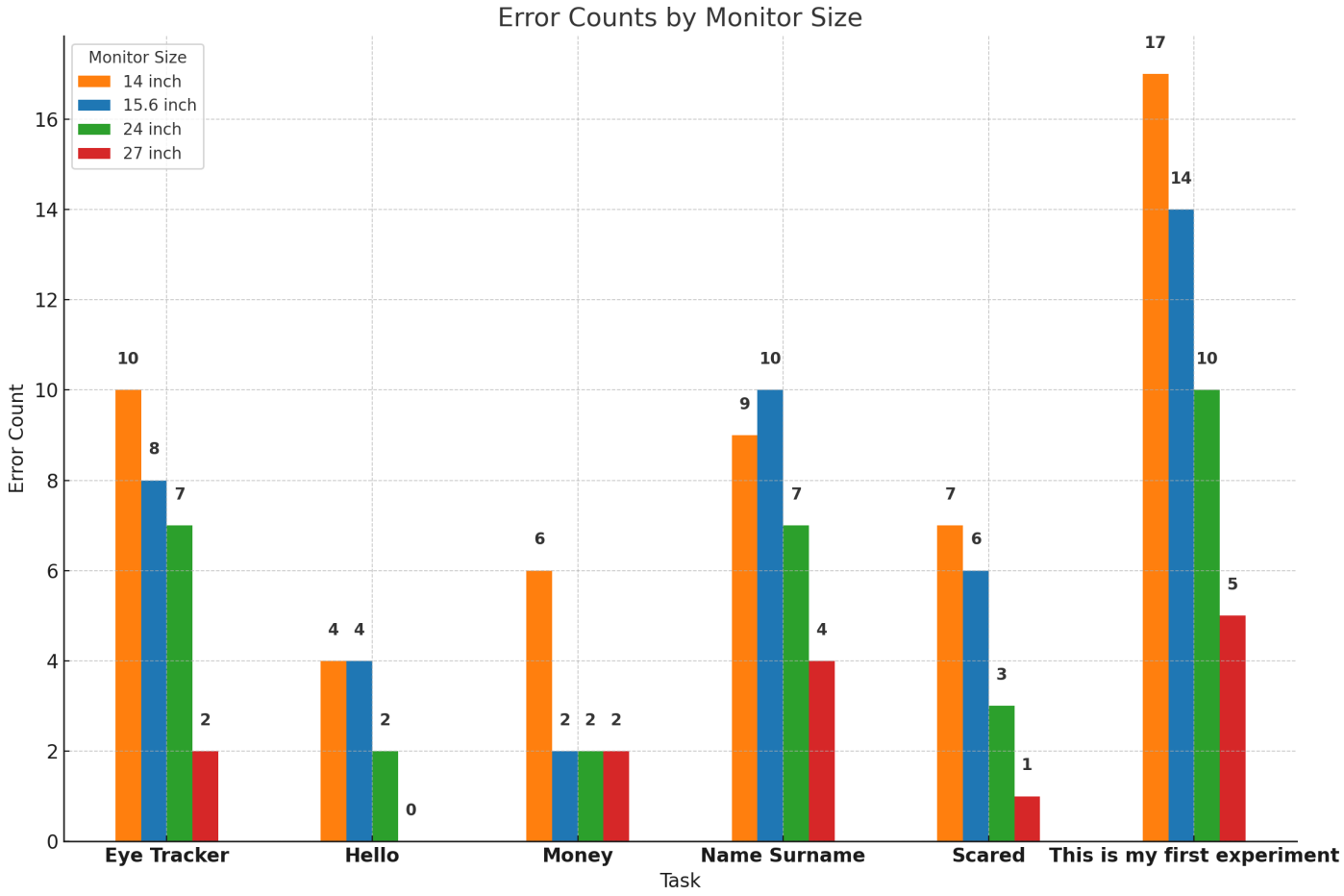

Figure 5.10. Error Counts by Monitor Size

Total of 21 Questions asked to the participants due to their experiences of three keyboards. The corresponding feedback is shown on Figure 5.13

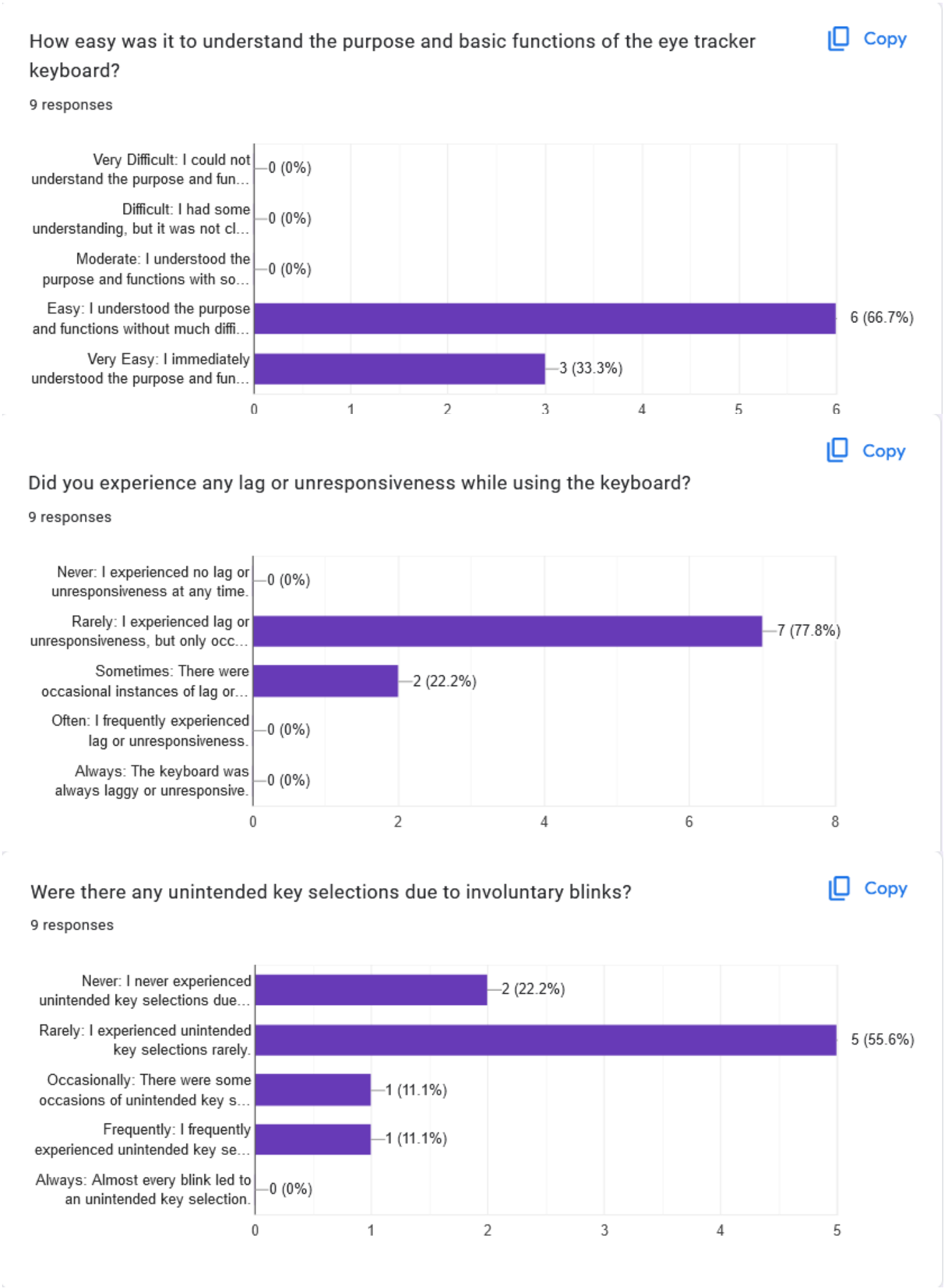

Figure 5.13. Results of conducted survey

6. CONCLUSION

The data collected from from surveys at the end of the experiment provides important insights into usability, efficiency, comfort, responsiveness, adaption of the eye tracker keyboard. These participants all agreed on the purpose and functionality of the virtual keyboard. The easy interface of virtual keyboard is essential for aiming at a larger user base. However, the replies revealed a bad experiences and discomfort for some participants. Two participant sometimes felt discomfort after using the keyboard.

Responses to the overall user experience of the keyboard ranged from average to highly efficient indicating that while keyboard serves the basic needs of user, there is an always a space for improvement. The keyboard layout's effectiveness was generally well-received,most participants find it highly effective, this also indicates that the layout corresponds to the user's eye motions and comfort concerns.

The participants adaptation to the eye tracker keyboard versus standard keyboard slightly under performed although the overall thought was system helps participant to adapt faster to the process. The feedback of helpfulness of the calibration were entirely good experience, indicating that these features of the eye tracker keyboard are well-designed and gave positive to the user experience.

Finally, the eye tracker keyboard shows significant potential in terms of usability, comfortable design and ease to use. These participants gives positive insights for future development. With easy design and customized interface, eye tracker keyboard gives disabled people a way of communication.

6.1. Future Work

Future eye tracker keyboard implementations should prioritize configurable features to broader range of users including those with special physical or cognitive limitations. Customized layouts, adaptable sensitivity and personal interaction modes that respond to users could be the subjects of future research of virtual keyboards.

While the keyboard showed promising ergonomics for users, individuals with restricted mobility or require extended duration's of use may not be satisfied with the current development. Future designs could include better eye tracking algorithms that reduce the need for unnecessary eye movement and blink rate modifications.

The integration of the eye tracker keyboard with assistive technologies could provide a big change for persons with disabilities. This integration might include compatibility with screen readers, voice recognition and other adaptive technologies which could improve the entire experience.

Building a feedback mechanism with users, especially those with impairments, may encourage people to share more experiences and support continual improvement and innovations. Things like Forums, user groups, workshop tools can be effective for feedback.

Artificial Intelligence and machine learning could improve the capabilities of keyboard. Improved predictive text input, adaptable user interfaces for individuals are just few examples that AI may make more easier and efficient tool for disabled users.

Bibliography

[1] H. Cecotti, Y. K. Meena, and G. Prasad, "A multimodal virtual keyboard using eye-tracking and hand gesture detection," in *2018 40th Annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, IEEE, 2018, pp. 3330–3333.

[2] M. Yildiz and M. Yorulmaz, "Gaze-controlled turkish virtual keyboard application with webcam," in *2019 Medical Technologies Congress (TIPTEKNO)*, IEEE, 2019, pp. 1–4.

[3] R. Islam, S. Rahman, and A. Sarkar, "Computer vision based eye gaze controlled virtual keyboard for people with quadriplegia," in *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*, IEEE, 2021, pp. 1–6.

[4] C. Yang, J. Sun, J. Liu, X. Yang, D. Wang, and W. Liu, "A gray difference-based pre-processing for gaze tracking," in *IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS*, IEEE, 2010, pp. 1293–1296.

[5] Y.-L. Kuo, J.-S. Lee, and S.-T. Kao, "Eye tracking in visible environment," in *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, IEEE, 2009, pp. 114–117.

[6] J. Li and S. Li, "Gaze estimation from color image based on the eye model with known head pose," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 3, pp. 414–423, 2015.

[7] B. Fu and R. Yang, "Display control based on eye gaze estimation," in *2011 4th International Congress on Image and Signal Processing*, IEEE, vol. 1, 2011, pp. 399–403.

[8] X. Xiong, Z. Liu, Q. Cai, and Z. Zhang, "Eye gaze tracking using an rgbd camera: A comparison with a rgb solution," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, 2014, pp. 1113–1121.

[9] (), [Online]. Available: `https://medium.com/javarevisited/converting-rgb-image-to-the-grayscale-image-in-java-9e1edc5bd6e7/`.

[10] (), [Online]. Available: `https://learnopencv.com/implementing-cnn-tensorflow-keras/`.

[11] (), [Online]. Available: `dlib%20library,%20http://dlib.net/`.

[12] (), [Online]. Available: `https://github.com/davisking/dlib-models`.

APPENDIX A: USER MANUAL

A.1. Overview of Virtual Keyboard

Three keyboard implementations are written on Python. It is recommended for users to use Python 2.8 or higher version. All three implementations are in separate files and runs independently from each other. In the files all of the implementations runs with the certain resolution for user needs. It is suggested that checking the resolution before running the virtual keyboard is essential for better experience.

A.2. Installation of Virtual Keyboard

The following python packages should be installed in order to run three implementations of keyboards.

- OpenCV (version 4.8.1 or higher)
- Numpy (version 1.26 or higher)
- pyenchant
- TextBlob
- dlib (version 19.22 required)
- math
- time
- pyglet

A.3. Overview of Eye Tracker

Calibration and Eye Tracking system both written in Python. It is recommended for users to use Python 2.8 or higher version to run tracking without any problems. The prediction system runs with keras which is a machine learning algorithm so it is required that correct versions of packages is so crucial to work without any error. It is important that, calibration must be completed before prediction algorithm. Blink detection file runs separately from eye gaze prediction algorithm so the run order should be calibration, prediction and blink capture.

A.4. Installation of Eye Tracker

The following python packages should be installed in order to run calibration, eye tracking and blink capture algorithm.

- Numpy (version 1.26 or higher)
- OpenCV (version 4.8.1 or higher)
- shutil
- Listener
- tensorflow (version 2.15.0 required)
- keras (version 3.0 required)
- pyautogui
- collections
- mediapipe (version 0.10 required)