DEVELOPMENT OF AN EYE TRACKING AND ON-SCREEN POINTING SYSTEM FOR PHYSICALLY IMPAIRED PEOPLE

KHANDAKER ANNATOMA ISLAM (SN. 0418140017)

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY
DHAKA, BANGLADESH

MARCH 2024

DEVELOPMENT OF AN EYE TRACKING AND ON-SCREEN POINTING SYSTEM FOR PHYSICALLY IMPAIRED PEOPLE

M. Engineering Thesis

By

Khandaker Annatoma Islam (0418140017)

Approved as to style and content by the Board of Examination on 25 March, 2024:

Dr. Md Akhtaruzzaman Assistant Professor of Computer Science and Engineering MIST, Dhaka	Chairman (Supervisor) Board of Examination
Dr. Muhammad Towfiqur Rahman Assistant Professor of Computer Science and Engineering Univresity of Asia Pacific, Dhaka	Member (External) Board of Examination
Dr. T. M. Shahriar Sazzad Assistant Professor of Computer Science and Engineering MIST, Dhaka	Member (Internal) Boar of Examination

Department of Computer Science and Engineering MIST, Dhaka

EVELOPMENT OF AN EYE TRACKING AND ON-SCREEN POINTING SYSTEM FOR PHYSICALLY IMPAIRED PEOPLE

DECLARATION

I hereby declare that the study reported in this project entitled above is my own original work and has not been submitted before anywhere for any degree or other purposes. Further I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this project and sources have been acknowledged and/or cited in the reference section.

Khandaker Annatoma Islam

Student No. 0418140017

Department of Computer Science and Engineering MIST, Dhaka

DEVELOPMENT OF AN EYE TRACKING AND ON-SCREEN POINTING SYSTEM FOR PHYSICALLY IMPAIRED PEOPLE

A Thesis

By

Khandaker Annatoma Islam

DEDICATION

Dedicated to my parents for always supporting and encouraging me

ABSTRACT

Development of an Eye Tracking and On-Screen Pointing System for Physically Impaired People

Eye tracking has become a popular field of research recently. It is not only getting more advanced but also giving rise to new research fields like eye tracking in neuro-architecture, e-sports etc. This project is a small step towards the wide-open research area of eye tracking. It has been developed with a vision to help the quadriplegic patients, who have limited movement capacity of their body. Due to their restricted movement, they have serious difficulties while working with computers and other electronic devices. Their needs might not be fully satisfied by current assistive technology, so a sophisticated eye tracking and on-screen pointing is required. With the help of this technology, physically disabled people will be able to communicate and use digital platforms more easily and with greater confidence thanks to a consistent, flexible, and intuitive interface. This technology aims to improve the quality of life for people with physical disabilities and encourage their active involvement in a variety of activities, such as social interactions, sports, education, and entertainment, by providing increased accessibility and inclusivity. The motivation behind developing an interactive system is to help people who have limited physical movement. Currently available solutions are in some cases costly. Whereas the proposed system in this research is more cost effective. This proposed system tracks the eye gaze using the webcam of a computer. Then it analyzes and responds with the output that in which region the person sitting in front of the camera is looking at. Taking input from the user's eye movement, the system moves the mouse on the screen. This study provides a landmark based machine learning model that can track eye gaze and suggest designated message for the gaze. The accuracy of the proposed system is around 90%. This system can be an easy to install and more cost-effective solution to the patients suffering from physical disability. Hopefully this system will provide a more user-friendly way to quadriplegic patients in terms of communicating with the outside world.

সারসংক্ষেপ

Development of an Eye Tracking and On-Screen Pointing System for Physically Impaired People

বর্তমানে Eye Tracking গবেষণার জনপ্রিয় একটি বিষয় হয়ে উঠেছে। এই ক্ষেত্রটি এখন বিকশিত হবার পাশাপাশি আরও নতুন গবেষণা ক্ষেত্রের জন্যও দ্বার উন্মোচন করেছে যেমন neuro-architecture, e-sports ইত্যাদি। এই গবেষণাটি eye tracking এর সম্ভাবনাময় গবেষণা জগতে একটি ছোট পদক্ষেপ। প্রজেক্টটি তাদের সাহায্য করার জন্য তৈরি হয়েছে যারা সীমিত চলাফেরার ক্ষমতা রাখেন। অঙ্গপ্রত্যঙ্গের সীমিত নড়াচড়ার ক্ষমতার কারণে, তারা কম্পিউটার এবং অন্যান্য ইলেকট্রনিক ডিভাইস ব্যবহারে সমস্যার সম্মুখীন হয়ে থাকে। বর্তমানে প্রচলিত প্রযুক্তি দ্বারা তাদের বিশেষ প্রয়োজনগুলি সম্পূর্ণভাবে পূরণ করা সম্ভব হয়না বলে, তাদের জন্য একটি eye tracking and on-screen pointing system প্রয়োজন। এই প্রযুক্তির সাহায্যে সীমিত চলাফেরার ক্ষমতার অধিকারী ব্যক্তিরা digital platform ব্যবহার করতে পারবে আরও সহজভাবে এবং আত্মবিশ্বাসের সাথে। এই প্রযুক্তির উদ্দেশ্য হল সীমিত চলাফেরার ক্ষমতা থাকা ব্যক্তিদের মানসিক স্বাস্থ্যের গুণগত মান উন্নত করা এবং দৈনন্দিন জীবনের অনেক ধরনের কার্যক্রমে তাদের সক্রিয় অংশগ্রহণের সংখ্যা বাড়ানো, যেমন সামাজিক আলোচনা, খেলাধুলা, শিক্ষা, এবং বিনোদন ইত্যাদি। একটি interactive system তৈরির পেছনে প্রধান উদ্দেশ্য হল সীমিত চলাফেরার ক্ষমতার অধিকারী ব্যক্তিদের সাহায্য করা। বর্তমানে প্রচলিত সমাধানগুলি বেশিরভাগ ক্ষেত্রে ব্যয়বহুল। প্রস্তাবিত সিস্টেমটি তুলনামূলক কম ব্যয়বহুলভাবে তৈরী করা হয়েছে। প্রস্তাবিত সিস্টেমটি Computer এর webcam ব্যবহার করে চোখের দৃষ্টি track করে। তারপর এটি বিশ্লেষণ করে জানিয়ে দেয় যে camera র সামনে বসে থাকা ব্যক্তি screen এর কোন অংশটাতে দেখছেন। ব্যবহারকারীর চোখের দৃষ্টির ডাটা নিয়ে এটি স্ক্রিনে মাউসের অবস্থানের পরিবর্তন করে। এই প্রজেক্টটি একটি Landmark ভিত্তিক machine leraning model প্রদান করে যা চোখের দৃষ্টি ট্র্যাক করতে পারে এবং দৃষ্টির জন্য নির্ধারিত বার্তা screen এ দেখাতে পারে। প্রস্তাবিত সিস্টেমের যথাযথতা প্রায় ৯০% এর কাছাকাছি। এই সিস্টেম দাম কম এবং ব্যবহারে সহজ যা সীমিত চলাফেরার ক্ষমতার অধিকারী ব্যক্তিদের জন্য একটি সমাধান হতে পারে। আশাকরি এই সিস্টেম দ্বারা quadriplegia এর মত রোগে ভুগতে থাকা ব্যাক্তিদের জন্য আশেপাশের প্রকৃতির সাথে যোগাযোগ করা সহজ হয়ে উঠবে।

ACKNOWLEDGEMENT

First and foremost, I would like to praise Allah the Almighty, the most gracious, and the most merciful for HIS blessing given to me during my study and in completing this thesis. May Allah's blessing goes to HIS final Prophet Muhammad Sallallahu.alaihi.wasallam , his family and his companions.

She is thankful to his research supervisor, Assistant Professor Dr. M. Akhtaruzzaman, Department of Computer Science and Engineering, Military Institute of Science and Technology (MIST), for providing patient guidance, enthusiastic encouragement and useful critiques throughout this project.

The author also acknowledges the support of Department of CSE, Military Institute of Science and Technology (MIST) for successful completion of this project. The author gratefully thanks Ministry of Defense for supporting this project.

The author also expressed his gratitude to the Quantum Robotics and Automation Research Group (QRARG) and the R&D Section of AZDREAM Technologies for providing insightful inputs in improving the overall project and the contents of the article.

The author would like to express her sincere gratitude to his family, friends, and others who helped her, directly or indirectly for their unwavering support and patience as she worked on this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Technology has always made human life easier. In terms of physically impaired individuals, its contribution is making history every now and then. With the passing days technology is improving the quality of life on earth. In terms of physically impaired peoples' lifestyle, technology has brought some remarkable changes. Eye tracking, being a developing field of study for many years now, has shown great promise in enabling impaired people to interact with computers and other electronic devices. Its capability increases manifold when on screen pointing is incorporated with it. Eye tracking technology means to control computer like interfaces with eye movements. With eye tracking and on screen pointing, impaired people are now able to perform tasks like selecting or typing in computer like interfaces. Also, on-screen pointing lets people control the cursor movement which can work as an alternative of mouse and joysticks.

By combining these two technologies, new opportunities have opened for physically challenged people. With the help of these technologies, they will get a chance to gain expertise in different computer-based systems using the human-computer interfaces. Their accessibility to information, communication process and participation in various fields will bring about significant changes in the society.

For designing a complete eye tracking and on-screen pointing system, it will require the knowledge of computer science, electrical engineering, human factor engineering and many more. The system must be designed in such a way that it is accurate, reliable, and cost effective. Also, the usability of physically impaired people needs to be considered above everything.

For example, Quadriplegia is a kind of disease which damages the spinal cord and makes the full body of the patient paralyzed. It limits the self-functioning capability of a human being by obstructing physical movement. A person suffering from quadriplegia completely depends on others for daily chores. Eye tracking can be a great help for quadriplegic patients for communicating with another person or machine.

The development of an eye tracking based on-screen pointing system will specially change the overall lifestyle of physically impaired people. It will provide another method of interacting with Human-Computer Interface replacing mouse or joystick. Thus, physically

impaired people will get chance to participate in a lot more field than before. As a result, their socio-economic status, mental strength will improve.

In this research, a system controlled by eye movement tracking has been introduced, which will be a significant help for physically impaired people in communicating with another people or machine.

1.1 Problem Statement

People with physical limitations, like quadriplegic patients, have a difficult time using computers and other electronic devices because of their limited range of motion. Available interfaces are not user friendly for them which lessens their involvement with the outside world. It also reduces their chances of acquiring knowledge and involvement in various activities. The variety of devices available in market does not fully meet their requirements. In some cases, they are costly to avail. It is therefore essential to develop an advanced eye tracking and on-screen pointing system that is appropriate for the special requirements of people with physical limitations. This system aims to empower people by offering a dependable, flexible, and easy-to-use interface that makes it easier to engage and communicate with digital platforms as well as outside world.

This problem statement describes the current obstacles that people with physical disabilities must overcome and draws attention to the shortcomings in the assistive technology available today, paving the way for the creation of a novel solution.

1.2 Objectives

The overall objective of this project is to design a prototype of eye pointing based HCI system for quadriplegic patient. The main objectives of this project can be segmented into following:

a. To detect and track eye pupil movement using Support Vector Machine (SVM)
b. To demonstrate on screen pointing, tracking, and action prediction through machine learning or any other approach.

1.3 Methodology

The methodology can be summed up in the following steps:

The research will start with an extensive literature review with a view to identifying problems and learn about the technologies that has been used for previously proposed solutions. Then system architecture will be designed, and a Machine Learning (ML) algorithm will be selected. After that, experimental setup will be conducted based on the system architecture. The selected algorithm will be implemented, and the full system will be tested. Lastly, an experiment with 5 participants will be conducted for overall system validation.

1. Selection of problem statement
2. Set specific objectives for the research
3. Perform literature review
4. Develop system architecture
5. Selection of machine learning algorithm
6. Evaluate the performance of machine learning algorithm result and elevate the efficiency.
7. Develop an experimental setup based on system testing and validation data.

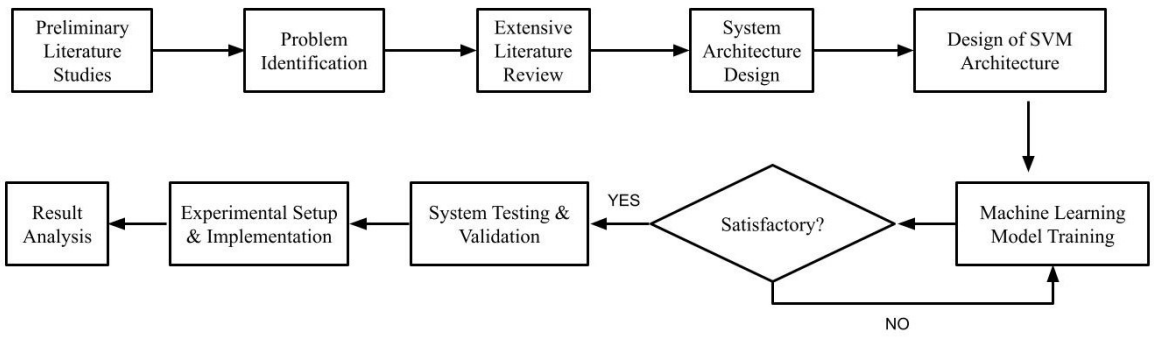


Fig. 1.1: Flow diagram of Methodology of the study.

1.4 Potential Application

The expected outcomes of the project are:

- An efficient eye pupil detection and tracking system.

- An efficient on-screen pointing and action prediction system.

1.5 Organization of the Remaining Chapters

The rest is organized in the following order:

In Chapter 2, literature review is discussed.

In Chapter 3, data collection, preprocessing, and design of model architecture are given.

In Chapter 4, experimental setup and implementation are discussed.

In Chapter 5, system testing, result analysis and benchmarking are presented.

In Chapter 6, the project is concluded.

In Chapter 7 and 8 reference and appendix are given.

CHAPTER 2
LITERATURE REVIEW

Eye tracking technology is discovering fascinating fields. Main part of this technology is tracking the eye movement and using the results for various purposes like user interest detection, attention span detection etc. Numerous solutions have come around in recent years. There are some popular available solutions or gadgets which can detect eye tracking like Tobii, EyeTech Digital Systems, SMI etc.

2.1 Eye Tracking

Eye tracking technology typically involves using specialized hardware, such as an eye tracker, that can detect and record the movement of the eyes as they look at a specific object or area. By analyzing the data gathered from an eye tracker, researchers can gain insights into how people process visual information, such as focusing part of a content, attention span, most viewed topics and so on. This information can be used to understand how people interact with technology, how they make decisions, how they perceive and remember visual information. Eye tracking has many practical applications, such as improving website and user interface design, evaluating the effectiveness of advertising and marketing campaigns, and diagnosing and treating vision problems. It has also been used to study a wide range of topics, such as reading comprehension, attention, and language processing. Overall, eye tracking is a valuable tool for researchers and professionals in many different fields, providing insights into human behavior and cognition that would be difficult or impossible to obtain through other methods.

Eye tracking technology offers a powerful means of understanding human behavior and cognition through the analysis of eye movements. By leveraging specialized hardware like eye trackers, researchers can delve into how individuals process visual information, make decisions, and interact with technology.

2.1.1 Ways of Eye Tracking

Eye Tracking can be done using various technologies. These technologies offer various options to gather eye tracking data. Some examples are:

- Electrooculogram (EOG): This method is mainly used in clinical purpose. In this process, electrodes are placed near the eye so that electrical signal generated by eye muscle movement can be detected.
- Optical tracking: In this method, eye movement or tracking is done using cameras or other optical devices. It is a convenient method for research and technological development.
- Magnetic tracking: In this process, magnets are used near eye to detect eye movements with the help of magnetic field. It is a less popular method than optical eye tracking or camera-based tracking.

In summary, eye tracking can be accomplished through various technologies, each with its own advantages and applications. These different approaches offer researchers flexibility in choosing the most suitable technique for their specific needs, contributing to advancements in understanding eye movements and their implications across various fields.

2.1.2 Use of Eye Tracking

Data gathered from eye tracking can be put into use in several ways. For example, to detect the behavioral patterns or to measure the time between two fixations, the changes of pupil while watching a content etc. can be valuable. Some other potential uses of eye tracking have been discussed below:

- Human-Computer Interaction: Eye tracking can revolutionize Human-Computer Interaction (HCI) field. It can be an alternative for mouse or joystick.
- Market research: Eye movement analysis can be used to determine how customer accepts a product or its advertisement. Depending on the feedback, companies will gain an insight into customer interest, need and expectation.
- Psychology and neuroscience: Eye tracking can be used in determining behavioral pattern of a person which can be helpful in understanding the person's personality. It can also be used in determining user’s ability to hold attention or concentration on specific matters.

- Medicine: Eye tracking can be useful in curing eye diseases. It can give us fast and accurate result about how a specific medicine is working on specific case.

Eye tracking and eye movement analysis are contributing to numerous fields. They are becoming an important part of upcoming research and technologies.

2.2 Eye Tracking and Physically Impaired People

Eye tracking technologies have shown great improvement in the lives of physically impaired people. With the help of this technology, physically impaired people have got another opportunity to communicate with the outside world without the need for using a mouse, joystick etc. For individuals who have limited body movement, eye tracking can be used as an alternative input method. It will allow them to participate in various activities, performing tasks, seeking assistance.

If eye tracking systems are combined with on-screen pointing, it can open huge opportunities for physically impaired people. With this, impaired people will be able to move cursors on-screen with the help of their eye movement, click on a particular area, seek assistance, or generate message on their own. It will give them a more convenient way of communicating with others, the potential to access information, and engage in various activities. It provides them with a sense of empowerment by providing them with the scope to participate in education, sports, entertainment and so on.

The adoption of eye tracking technology represents a significant step towards inclusivity and accessibility, enriching the lives of physically impaired individuals and promoting their active engagement with the world around them.

2.3 Quadriplegia

Quadriplegia, also known as tetraplegia, is a medical condition in which a person experiences loss of all four limbs and the torso. It is typically caused by damage to the spinal cord in the neck region, often because of a traumatic injury, such as a severe fall or car accident.

The severity of quadriplegia can vary, depending on the extent and location of the spinal cord injury. In some cases, a person may be able to move their arms or legs to some degree,

while in other cases they may have no control or sensation in any of their limbs. Paralysis may also affect the muscles that control breathing, swallowing, and other essential bodily functions.

People with quadriplegia often require significant assistance with activities of daily living, such as dressing, bathing, and eating. They may also require specialized medical care, such as respiratory support or pressure sore management.

Overall, quadriplegia presents significant challenges for affected individuals, highlighting the importance of comprehensive support and care to enhance their quality of life and functional independence.

2.4 Support Vector Machine (SVM) Algorithm

Support Vector Machine (SVM) was selected as the primary algorithm for detecting the screen region with the help of eye tracking. SVM is a powerful machine learning algorithm used for both classification and regression tasks (Fig. 1.2). It works by finding the optimal hyperplane that separates different classes or predicts continuous outcomes with maximum margin. SVM is known for its effectiveness in handling high-dimensional data and robustness to overfitting.

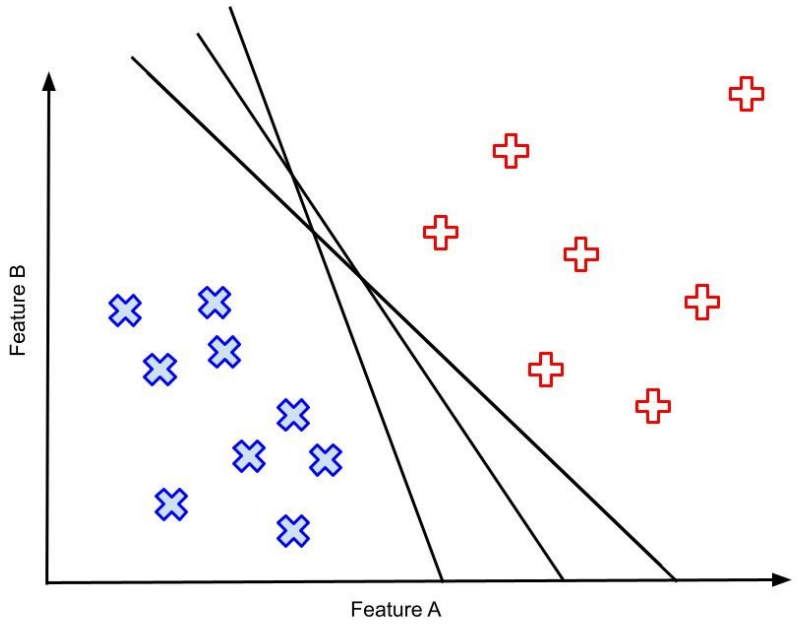


Fig. 1.2: Support Vector Machine (SVM)

2.5 Background Study

Several works have been done in this arena throughout the years. With the advancement of technology, eye tracking systems have evolved so much. More functional gadgets are available in market like Tobii, SMI etc. But not all of these technologies are in affordable range.

Lida et al. (2022) researched on a gaze-assisted keyboard typing system for individuals with motor disabilities. It uses eye tracking technology to allow the user to select keys on an on-screen keyboard and type out messages. The researchers have also presented an analysis that their proposed system performs better than other existing systems.

Natalie et al. (2020) has discussed about the use of eye tracking hardware for gaze-guided magnification for low-vision computer users. They have created a gaze-controlled magnifying tool to probe technical feasibility and assessed eye tracking quality for low-vision users as they performed reading and search tasks.

Radu et al. (2019) developed an eye-tracking-based interface for real-time applications, analyzing the performance of eight algorithms to identify the most appropriate pupil detection algorithm. They found circular Hough transform algorithm performs best for real time eye tracking based applications.

Alexandre et al. (2019) has presented a new assistive system based on eye tracking for controlling and monitoring a smart home, based on the Internet of Things. The system was tested by 29 participants without disability and 1 participant with severe disability and it was evaluated with good score from both group of participants.

Mina et al. (2020) has proposed a system to determine user performance in virtual training by eye gaze tracking and based on that elevate the level of the training and so on.

Teng et al. (2019) proposed an electrooculography-based human-computer interactive system that demonstrated better performance and accuracy than the conventional methods.

There have been several works on different kinds of predictive systems using machine learning algorithms as well. Khan et al. (2019) developed a predictive system with BSAS clustering and Naive Bayes algorithm to assist in the diagnosis of type 2 diabetes. In another study, an m-health application was developed using support vector machine (SVM) (Qureshi et al., 2020). Predictive systems have found their way into many other health-

informatics related subjects like prediction of cesarean child birth (Khan et al., 2020) and many others (Preetha, 2020).

A comparison table among the studied articles in presented in Table 2.1.

Table 2.1: Related studies

Author(s)	Eye tracking method	Motivation and objective	Findings	Limitations
Teng et al. (2019)	Electroocu logram (EOG)	To get feature from eye movements data by applying band power & HHT and classify them using PRNN. This will help is development of the assistive device for elderly people.	Band Power method given better result than HHT method	Only age and gender-based data have been considered to choose classification as well as recognition accuracy model
Mina et al. (2020)	Head mounted eye tracking glasses	Improve the quality of virtual training system using eye tracking data	Analysis shows that the proposed method can identify different patterns with hold-up time and distraction	Pattern determines whether the user can follow the virtual training or not. Other practical elements have not been taken to consideration
Alexandre et al. (2019)	The Eye Tribe 1.0, Tobii Pro	Design a smart home system for physically challenged people	The assistive system obtained around 80% score from the participants who evaluated it	In some cases, anti-reflective film has been used to prevent infrared eye tracker.
Radu et al. (2019)	Head-mounted eye tracking interface	Determine which of the six algorithm performs better for eye tracking application	CHT algorithm gave better results	Lighting conditions were nonuniform
Natalie et al. (2020)	GazePoint GP3	Improvement of magnification control through eye gaze	Video based eye tracking could be more effective if feedback from low-vision user is integrated.	Sample size was small

In order to evaluate the usability of the applications towards specific target groups, several researches have conducted with different usability measurement metrices. Suha et al.

(2022) and Muaz et al. (2021) assessed the usability different mobile applications for target users like autistic users. Other works like learning system for deaf and mute users were also developed (Hasnin et al., 2023).

2.6 Research Gap

A research gap in this area is the affordability and accessibility of eye tracking technologies for individuals with motor disabilities. While existing studies have explored various applications and functionalities of eye tracking systems, there is limited research on the development of affordable and user-friendly eye tracking solutions specifically tailored to the needs of individuals with disabilities. Additionally, there is a need for comparative studies evaluating the performance and effectiveness of different eye tracking technologies in real-world settings, particularly for users with diverse levels of motor impairment. Addressing these gaps could contribute to the development of more inclusive and accessible assistive technologies for individuals with disabilities.

2.7 Summary

This chapter looks at recent studies on eye tracking and explores different technologies already out there. It shows that while some progress has been made, there's still a big need for a system that's easier to use and doesn't cost as much, especially for people who can't move their limbs much. Many of the current options are often expensive or too complicated for people with mobility issues. Therefore, there is need for a solution that's affordable and simple to use. By pointing out these gaps in existing technology, this chapter emphasizes the importance of creating new solutions that user-friendly for people with disabilities.

CHAPTER 3

DATA COLLECTION, PREPROCESSING, AND DESIGN OF MODEL ARCHITECTURE

In this project, a detailed system has been designed to achieve the goals. This chapter explains how the system works, including eye tracking, data labelling, data preprocessing, creating machine learning models, and setting up the environment needed.

Before getting into the technical details, a lot of research has been done to understand what has already been done. Based on this, a solution is aimed for building that's easy to use, doesn't cost much, and helps people with limited movement. This solution should be easier for the patients to use technology and improve their lives.

This chapter has been segmented into how the system works, explaining the methods and tools used. By doing this, a clear picture has been drawn of how the system functions and how it could make technology more accessible for people with mobility challenges.

3.1 Data Collection

Data collection process is described below with the help of a pseudocode.

Algorithm 1 Data Collection Process

```
Initialize camera object.
Initialize file to save dataset.
while True do
    Read frame from camera.
    Preprocess frame (flip, convert to RGB, etc.).
    Detect face landmarks using face Landmarks.dat.
    Extract left and right eye landmarks from detected landmarks.
    Calculate normalized landmark values for each eye.
    Calculate the white part proportion for each eye.
    Take the ratio of the previous value for left eye and right eye.
    Combine landmark and ratio value into a single row of data.
    Display frame with detected landmarks.
    Write region value based on eye tracking data to file.
end while
```

3.1.1 Eye Tracking Process

The process starts by tracking the face through video camera (webcam) of a PC. Then landmark points in the face have been detected. Six landmark points (Left Eye: 36, 37, 38, 39, 40, 41 and Right Eye: 42, 42, 44, 45, 46, 47) around an eye is used to locate the position of the eye.

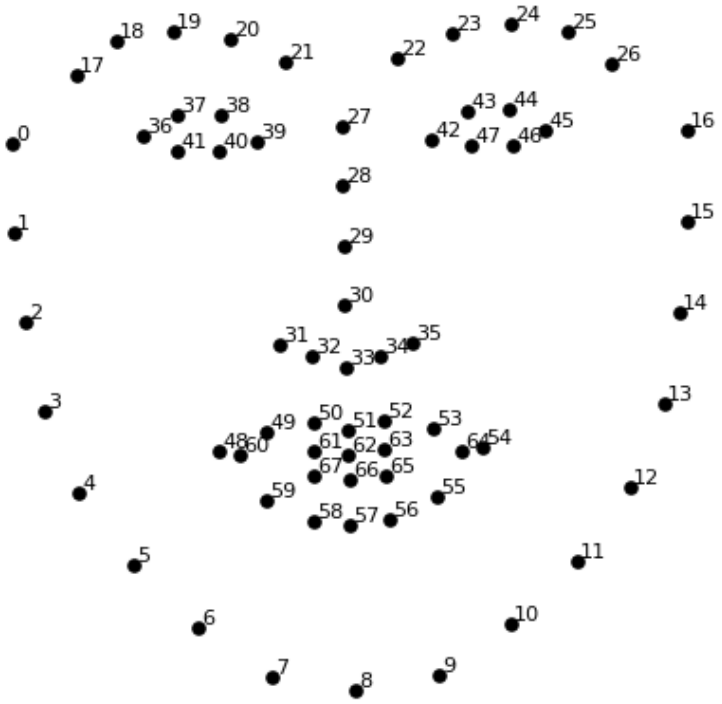


Fig. 3.1: Facial Landmark Image.

Then the position of the pupil is detected. If we take left eye for example, we have to determine the midpoints of (37,38) and (40,41). Then a vertical line connecting the midpoints and a horizontal line connecting (36,39) is considered and their intersecting point is considered to be the pupil location. Same process has been followed for the right eye just the landmark points will be different for it.

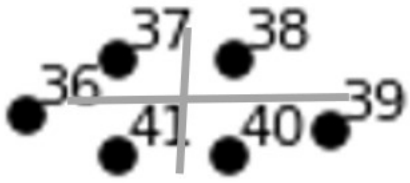


Fig. 3.2: Pupil Locating Image

After detecting pupil for both eyes, white region around the pupil has been analyzed. With the help of it, eye tracking has been done. The process has been described by a flow chart below

As shown in Fig 3.1, in this project we have worked with only eye landmark points that is Left Eye: 36, 37, 38, 39, 40, 41 and Right Eye: 42, 42, 44, 45, 46, 47. For each of these landmark points a pixel (x, y) data has been generated. This data is stored in a row for one time eye detection. With it the region a user is gazing at is stored also. So, we are getting 13 points for each traversal of eye detection. These data are stored in an excel file. At first these data are used to train the SVM. Then based on the training, real time eye gaze detection is tested to see whether it is detecting eye gaze correctly or not.

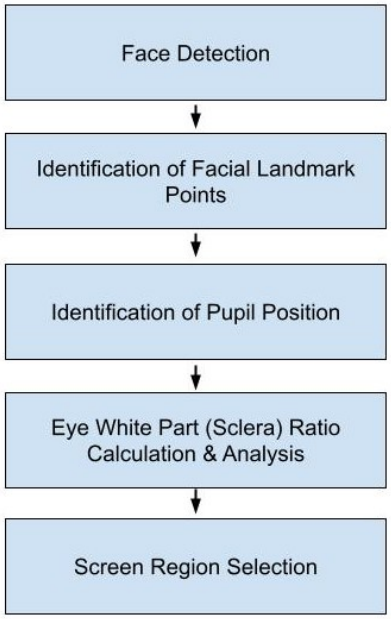


Fig. 3.3 Eye Tracking Process

Left_0_x	Left_0_y	Left_1_x	Left_1_y	Left_2_x	Left_2_y	Left_3_x	Left_3_y	Left_4_x	Left_4_y	Left_5_x	Left_5_y
245	207	255	200	268	199	280	209	268	213	254	213
244	205	255	197	270	197	281	208	268	212	254	212
244	205	255	196	269	196	281	207	268	211	254	211
244	204	255	196	269	196	281	207	268	211	254	210
245	205	256	197	270	196	282	208	269	211	254	211

Right_0_x	Right_0_y	Right_1_x	Right_1_y	Right_2_x	Right_2_y	Right_3_x	Right_3_y	Right_4_x	Right_4_y	Right_5_x	Right_5_y	Region	Gaze Ratio
333	207	344	197	358	196	368	203	359	209	345	210	4	0.9150323
332	206	344	195	358	193	368	201	359	208	345	209	1	0.5441233
333	205	345	194	359	192	369	200	360	207	346	208	1	0.5752348
332	205	344	195	358	193	369	201	359	207	345	208	1	0.5094111
333	206	345	195	359	193	369	201	360	208	346	208	1	0.5756853

Fig. 3.4 Sample of the Collected Data

This is the dataset which was used in this project. Here (x, y) pixel values have been gathered for the 6 landmark points of each eye. Then the region was denoted with 1-4 numerical values were 1 is for left-up, 2 is for right-up and 3 is for right-down and, 4 is for left-down, 4Region was segmented based on Gaze Ratio. The box plot of the left eye (Fig. a) and the right eye (Fig. b) shows a summary of the collected data. It can be seen that the average values of 'x' coordinates for left eye are within 250-300 and for right eye are within 325-375, while the average values of 'y' coordinates are near to 150.

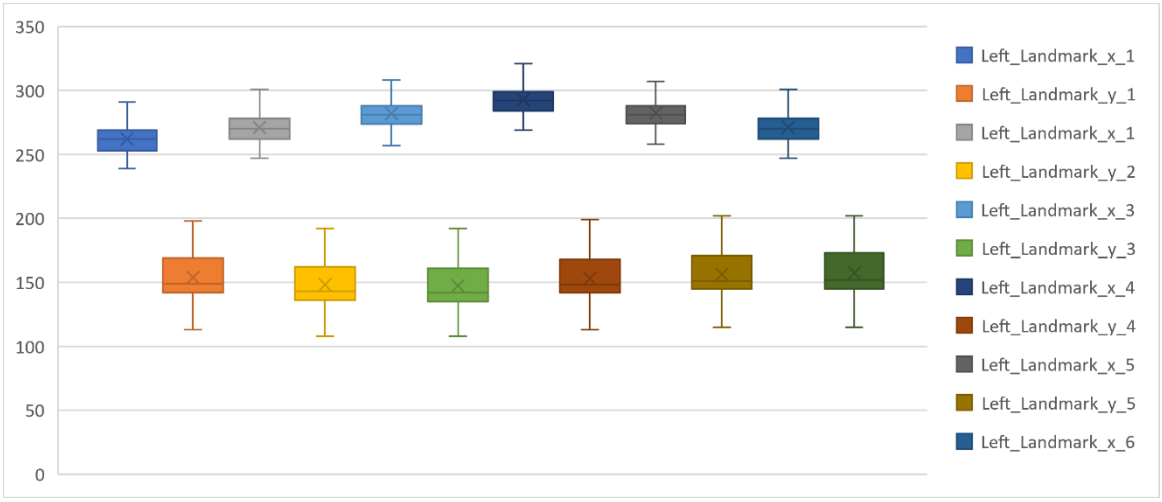


Fig. 3.5: Box plot of left eye data

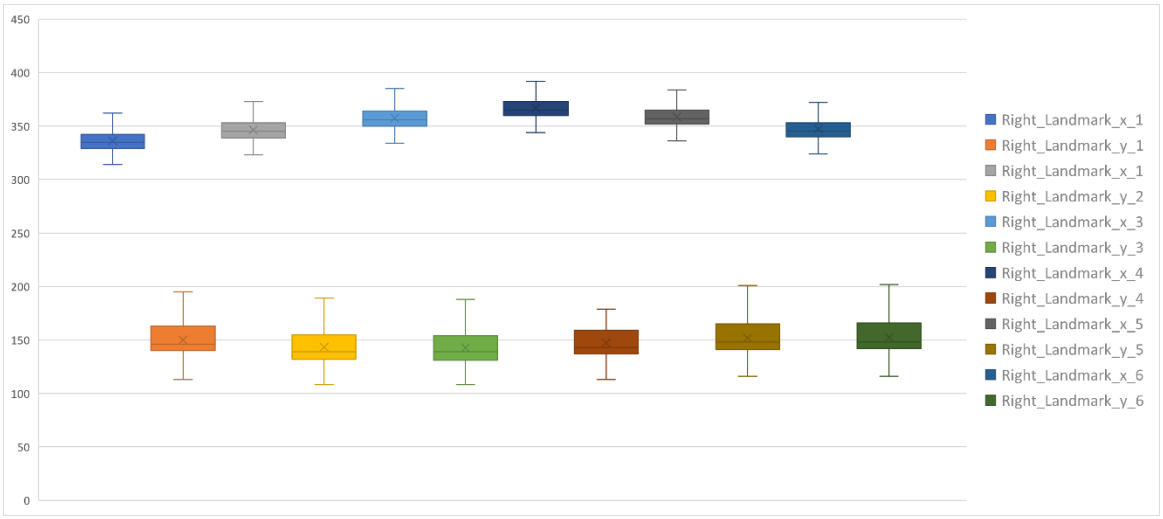


Fig. 3.6: Box plot of right eye data

3.1.2 Data Labeling

While collecting the eye tracking data, labeling was commenced simultaneously. Here are the steps that were taken to data labeling.

- Defining Labeling Criteria: Determining the specific eye movement characteristics or features those needs to be capture through labeling. These could include types of eye movements (e.g., saccades, fixations), regions of interest on the screen, or specific behaviors associated with eye gaze.
- Preparing Data: Eye movement data is collected or recorded using an eye tracking device or software. It is ensured that the data is properly formatted and organized for labeling.
- Selecting Labeling Tools: Labeling tools or softwares are chosen that allow the visualization of the eye movement data and the efficient application of labels. This could be specialized eye tracking software or general-purpose annotation tools. In this project, labeling was done manually in Microsoft Excel.
- Commencing Labeling: The eye movement data is labeled according to the defined criteria. The selected labeling tools are used to mark relevant features or behaviors

directly on the data visualization. Appropriate labels or tags are assigned to each data point or segment.

- Reviewing and Quality Control: The labeled data is regularly reviewed to ensure consistency and accuracy. Quality control measures, such as double-checking labels by multiple annotators or comparing labeled data with ground truth information, are implemented.
- Resolving Ambiguities: Any ambiguities or uncertainties encountered during labeling are addressed by consulting with domain experts or revising labeling guidelines as needed. It is ensured that all annotators are aligned on interpretation and application of labels.

3.2 Data Preprocessing

The steps which have been followed for data pre-processing are shown below:

- Feature Scaling of Numeric Data: Numeric values were normalized to eliminate bias and to obtain better accuracy.
- Missing Value Management: The columns with missing values were dropped in order to improve performance of the prediction systems.
- Non-Numerical Feature Handling: There were no non-numerical features in the dataset.
- Division of Dataset: After completion of pre-processing, the dataset was ready for being used in the SVM prediction model. The whole dataset was randomly divided into Training set and test set with a ratio of 80:20.

3.3 Development of SVM Architecture

In this project, the Support Vector Machine (SVM) algorithm was utilized for machine learning. SVM is recognized for its effectiveness in classification and regression tasks. Landmark point data from eye tracking was collected and stored in an Excel spreadsheet, where pixel positions (x, y) were recorded. Initially, the SVM was trained using a

substantial amount of data. Subsequently, the performance of the SVM model was evaluated using test data.

3.3.1 Reasons for Selecting SVM

A wide range of algorithms were considered for the eye tracking application. But, Support Vector Machine (SVM) was preferred for the eye tracking applications due to several reasons:

- Effectiveness in Classification: SVM is known for its robust performance in classification tasks. In eye tracking, SVM can effectively classify eye movements into different categories, such as gaze direction or fixation duration.
- Handling Non-linear Relationships: SVM can effectively handle non-linear relationships between input features and output labels. This is crucial in eye tracking, where the relationship between eye movement patterns and corresponding actions or stimuli may not be linear.
- High Dimensionality: Eye tracking data often involve a high-dimensional feature space, especially when considering multiple eye movement parameters. SVM can handle high-dimensional data efficiently and is less prone to overfitting compared to other algorithms.
- Generalization: SVM tends to generalize well to unseen data, making it suitable for eye tracking applications where the model needs to perform accurately on new individuals or in different environmental conditions.
- Kernel Trick: SVM allows for the use of different kernel functions to map input data into higher-dimensional feature spaces. This flexibility can help capture complex relationships in eye tracking data that may not be linearly separable in the original feature space.

Overall, SVM's ability to handle classification tasks in high-dimensional spaces, its effectiveness in handling non-linear relationships, and its generalization capabilities make it a suitable choice for eye tracking applications.

3.3.2 Hyperparameter Tuning

Hyperparameter tuning is a crucial step in optimizing the performance of machine learning models, including Support Vector Machines (SVMs). Hyperparameters are parameters that are set prior to the training process and cannot be directly learned from the data. Examples of hyperparameters for SVMs include the choice of kernel type, the regularization parameter (C), and the kernel-specific parameters (such as gamma for the RBF kernel).

Random search is a hyperparameter optimization technique that involves randomly selecting combinations of hyperparameters from a predefined search space and evaluating their performance using cross-validation. Random search was chosen for tuning hyperparameters because it is a simple yet effective approach that can often outperform exhaustive grid search in finding good hyperparameter configurations, especially in high-dimensional search spaces (Fig. 3.7).

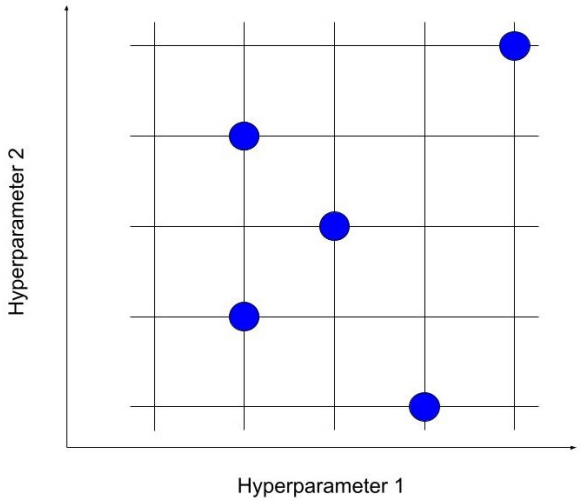


Fig. 3.7 Hyperparameter tuning with random search

3.3.3 Model Training

After preprocessing training data and selection of hyperparameters, the SVM model was trained with the training data. To initialize the model, the following code snippets were run.

$from\ sklearn.svm\ import\ SVC$
$model\ =\ SVC()$

Sci-kit learn is an open-source library with an extensive collection of implementations of data science and ML algorithms. The SVM classifier was imported from sci-kit learn and in instance of the classifier was created. Then the model was trained with the training data.

$$model.fit(x\_tr, y\_tr)$$

3.4 Summary

This project aimed to design a detailed system focusing on achieving specific goals, which are explained in this chapter. The system's functioning, including eye tracking, machine learning model creation, and environment setup, is elaborated upon. Prior to delving into technical details, extensive research was conducted to understand existing solutions, aiming to develop a user-friendly, cost-effective solution to aid individuals with limited movement in using technology and improving their quality of life.

CHAPTER 4
EXPERIMENTAL SETUP AND IMPLEMENTATION

This chapter describes how the whole system works in a more technical level. Through this chapter how data is collected, how the process is designed and implemented, how SVM has been trained and tested is discussed.

4.1 Overall Plan for the Eye Tracking Application

The full process of the eye movement tracker can be represented with a flow diagram in Fig. 3.8. The whole process is repeated as long as the application is running.

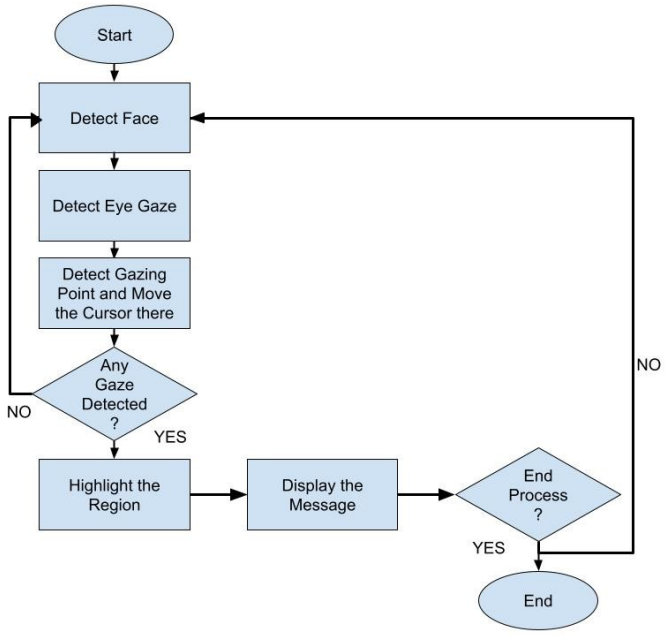


Fig. 4.1 Overall Process Flow Chart

4.2 Software Interface Design

For eye tracking, PC screen has been divided into four regions as Left-Up, Left-Down, Right-Up, Right-Down.

Left-Up	**Right-Up**
Left-Down	**Right-Down**

Fig. 4.2 PC Screen Segmentation

Suppose when a person will gaze at “Left-Up” segment, it will be highlighted as shown below.

Left-Up	**Right-Up**
Left-Down	**Right-Down**

Fig. 4.3 PC Screen Segmentation (example 1).

Similarly, when he will stare at “Right-Down” it will be highlighted as shown below.

Left-Up	**Right-Up**
Left-Down	**Right-Down**

Fig. 4.4 PC Screen Segmentation (example 2).

4.3 Environment Setup

The following hardware and software components were setup as a part of preparing the environment (Table 4.1).

Table 4.1: System test environment

Component Type	Component Name	Specification
Hardware	Processor	Intel Core i5-8265U (8 CPUs) @ 1.60GHz 1.80 GHz
	Memory	12 GB
	Graphics Processor	NVIDIA GeForce MX150
	Graphics Memory	8 GB
	Screen Resolution	1920 x 1080
Software	Operating System	Windows 11 Home 64 bit
	Python	3.8.2
	OpenCV	4.4.0.46
	Pandas	0.20.3
	Dlib	19.8.1
	Matplotlib	3.3.4
	Numpy	1.19.5

4.4 Implementation of Eye Movement Tracker in Python

To implement the eye movement tracker using the trained model, the following python modules were imported: cv2, numpy, dlib, hypot, pandas, sklearn and svc. After importing the libraries, the following steps are followed:

- Step 1: Reading the image from webcam
- Step 2: Converting the image to greyscale
- Step 3: Locating landmarks from the image
- Step 4: Calculation gaze ratio from the landmarks
- Step 5: Feed the gaze ratio of both eyes to the predictor model and find the output region from the model
- Step 6: Mark the section of the screen corresponding to the predicted region from eye movement tracking

- Step 7: Repeat from Step 1

4.4.1 System Development

System Language : Python

System Version : Python 3.6

Libraries : OpenCV, Pandas, Dlib, Matplotlib, Numpy etc.

The program has been written in python using the libraries mentioned before. ML classifier SVM was trained with the collected test data and then tested with test data from the faces of different subjects. The results have been analyzed and presented afterwards. The application can detect the regions properly based on eye gaze tracking and by blinking in a specific region it displays some necessary messages.

4.4.2 Software Tool

For coding and system implementation, PyCharm software has been used. It is an open-source IDE built by JetBrains. Python file execution, Excel data storing, SVM training and testing were conducted using this IDE.

4.5 Conduction of Primary Test

While testing the application, when a user gazes at a region it is highlighted. Here four type eye movement have been considered: Left_up, Left_Down, Right_Up, Right_Down. Images have been shown below.

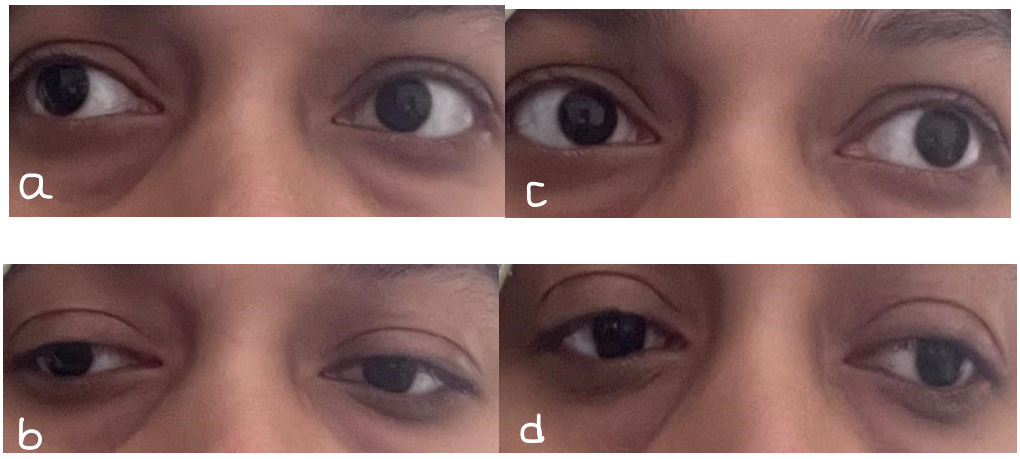


Fig. 4.5 Eye Pupil Position for (a) Left_Up, (b) Left_Down, (c) Right_Up, (d) Right_Down Region

While 4.6 (a) case occurs, PC screen highlights the Left_Up region. Similarly, for 4.6 (b) case it highlights the Left_Down region, 4.6 (c) case it highlights the Right_Up region and lastly for 4.6 (d) case it highlights the Right_Down region. Images have been shown below.

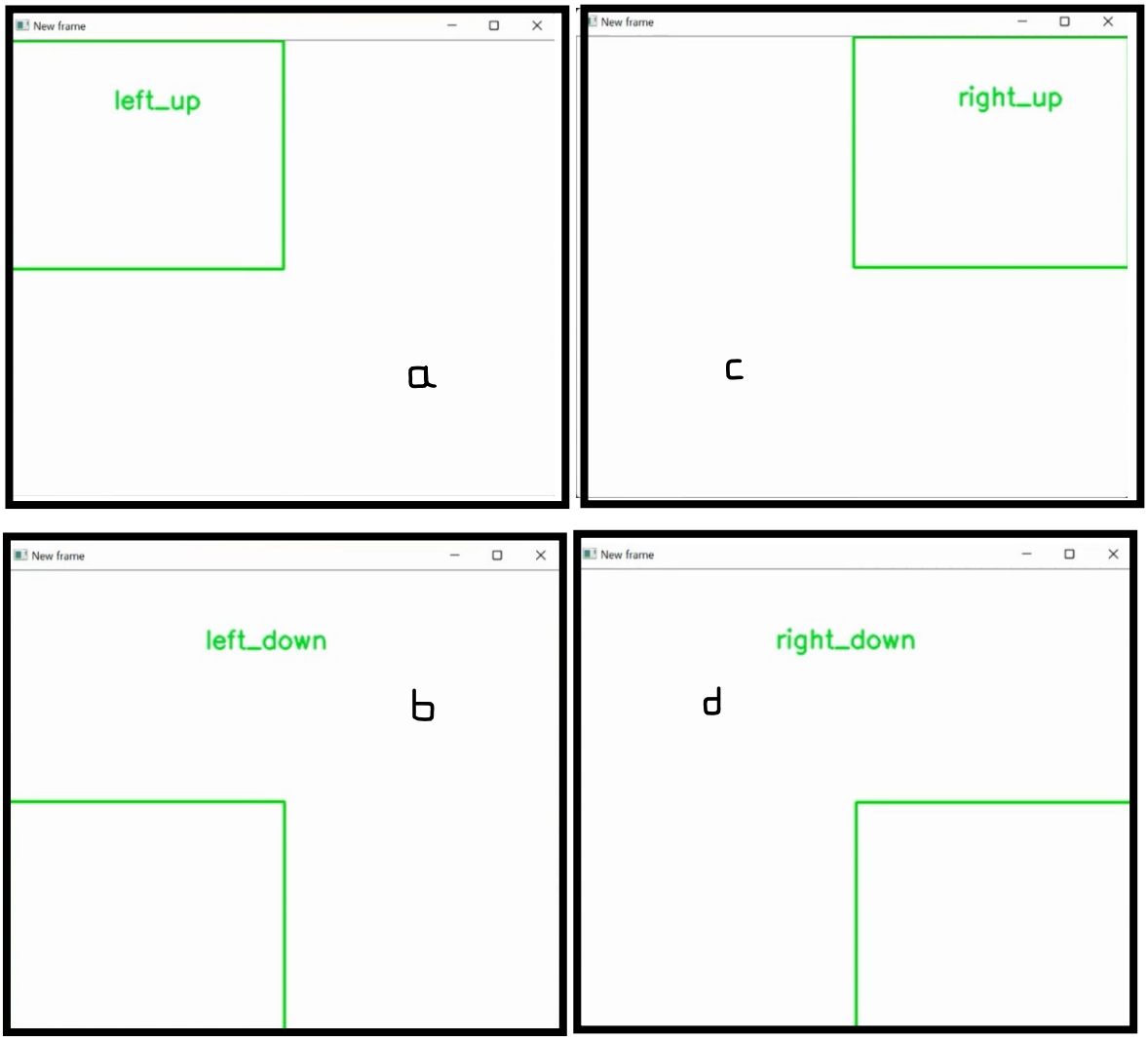


Fig. 4.7 Output Screen for (a) Left_Up, (b) Left_Down, (c) Right_Up, (d) Right_Down Eye Gaze

4.6 Summary

This chapter explains all the technical details about how the system works. It talks about how data is collected, how the software is designed, and how testing is done. The data collected is used to teach the SVM, which is then tested to see how well the system can

detect where someone is looking. This helps us understand how well the system works and sets the stage for further testing and evaluation.

CHAPTER 5
SYSTEM TESTING, RESULTS ANALYSIS AND BENCHMARKING

The thorough testing and analysis phase is crucial in assessing the efficacy and potential of any system. In this section, we delve into the rigorous testing conducted on our system, aimed at evaluating its performance and reliability in real-world scenarios. Furthermore, we present a detailed analysis of the results obtained from these tests, highlighting key findings and insights gleaned from comparing different methodologies. Additionally, we discuss the implications of these results and outline potential avenues for future enhancements and integration into broader systems. This section serves as a comprehensive overview of the testing, analysis, and future prospects of our eye tracking and on-screen pointing system for physically impaired individuals.

5.1 System testing

The system has been designed keeping the objectives in mind. At first two screen will pop-up in-front of the user. On the left one, there will be pointing as per the user gaze. And on the right one, user's image will be shown. If the user is gazing at the left-up region, left-up region on the left screen will be marked. Similarly, if he is gazing at the right-down region, it will be marked on the left screen.

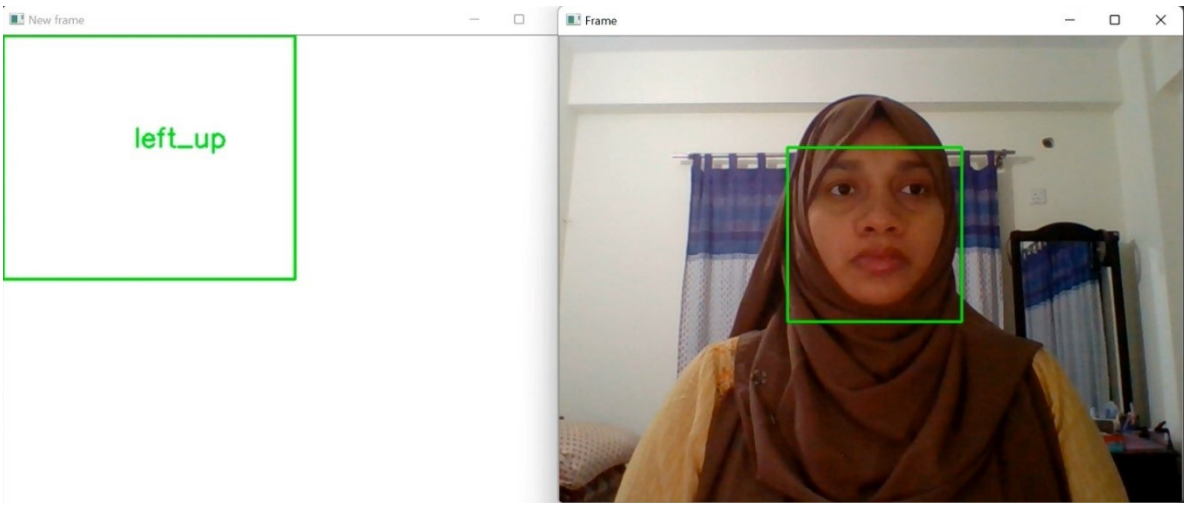


Fig. 5.1 User Gazing at Left-Up

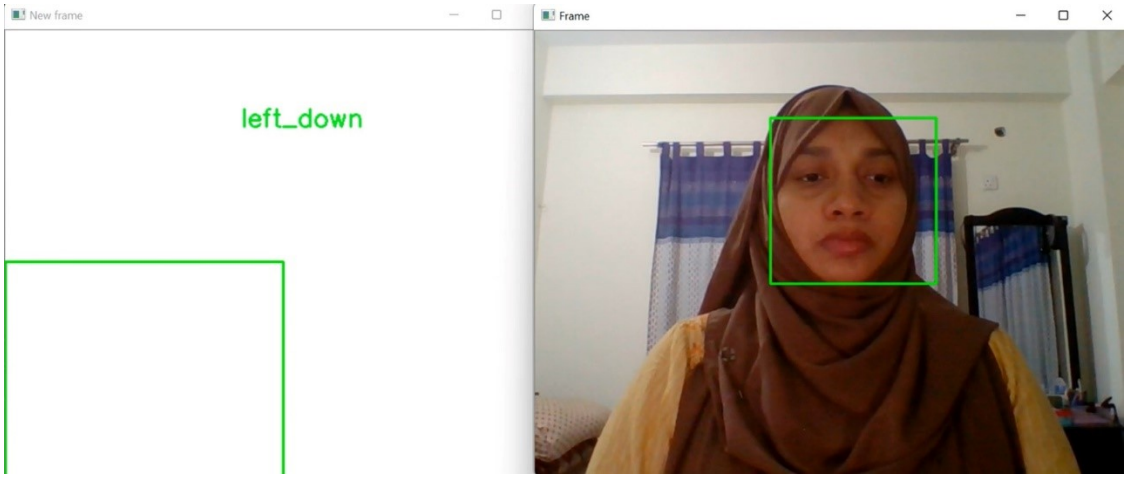


Fig. 5.2 User Gazing at Left-Down

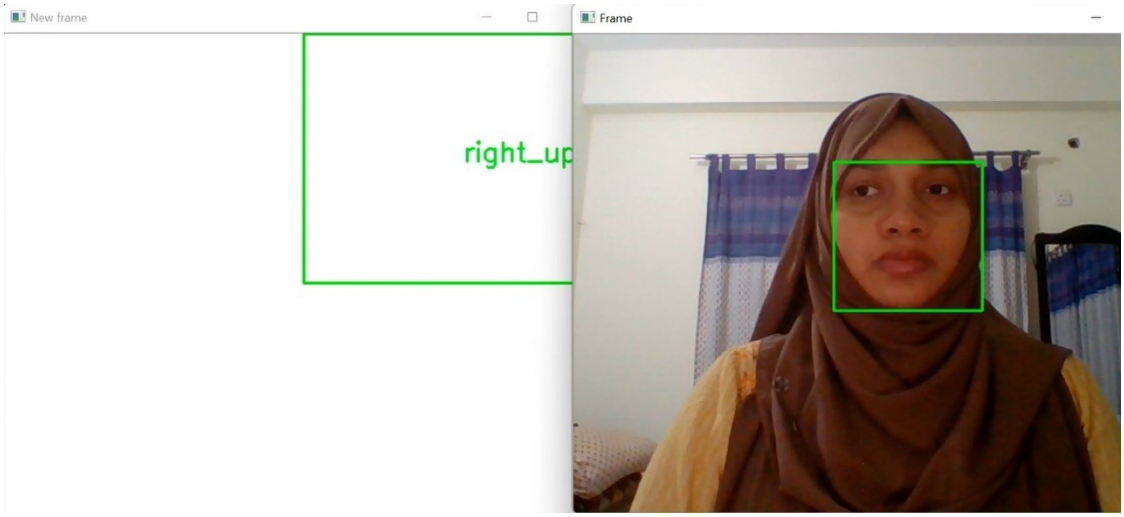


Fig. 5.3 User Gazing at Right-Up

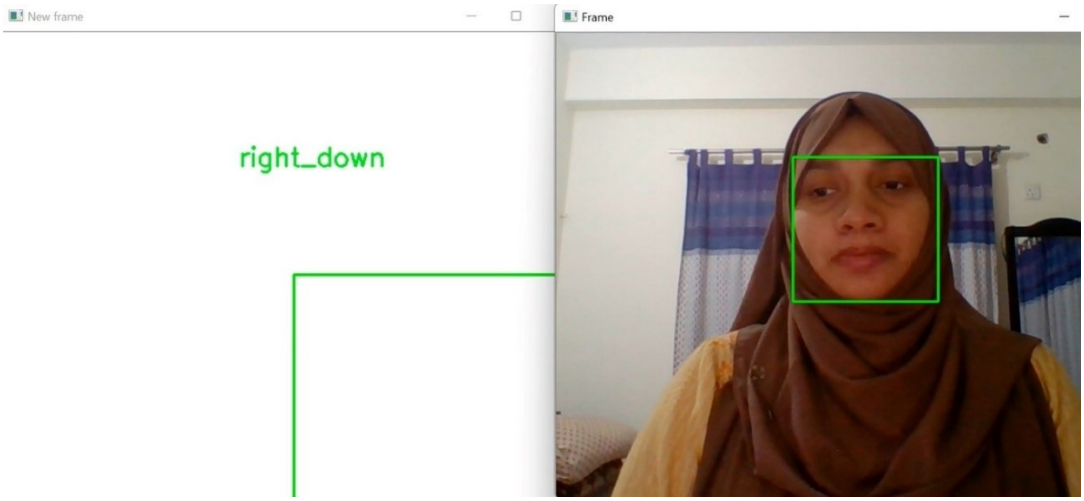


Fig. 5.4 User Gazing at Right-Down

The system has been tested on multiple people. So, there were variation in eye landmark point data. Also, lighting conditions were adjusted during multiple phases of testing. It had an effect on eye tracking.

Some patterns of gazing were followed during testing. Like a clockwise gazing has been performed to check whether the implemented system can detect it correctly or not. Similarly, anti-clock, z pattern, x pattern etc were tested to check system's performance.

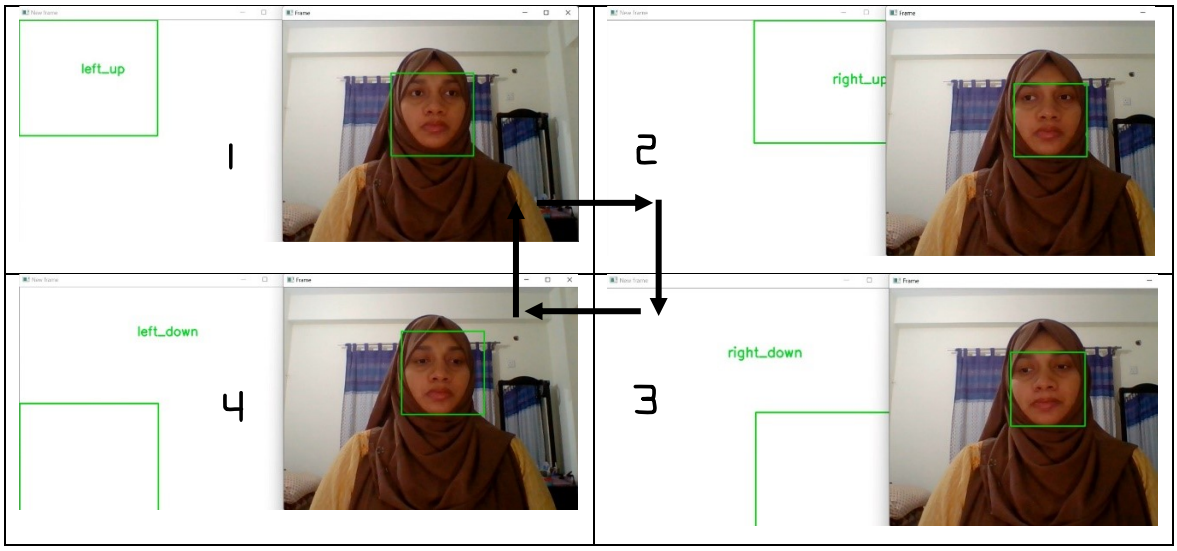


Fig. 5.5 Clockwise Pattern

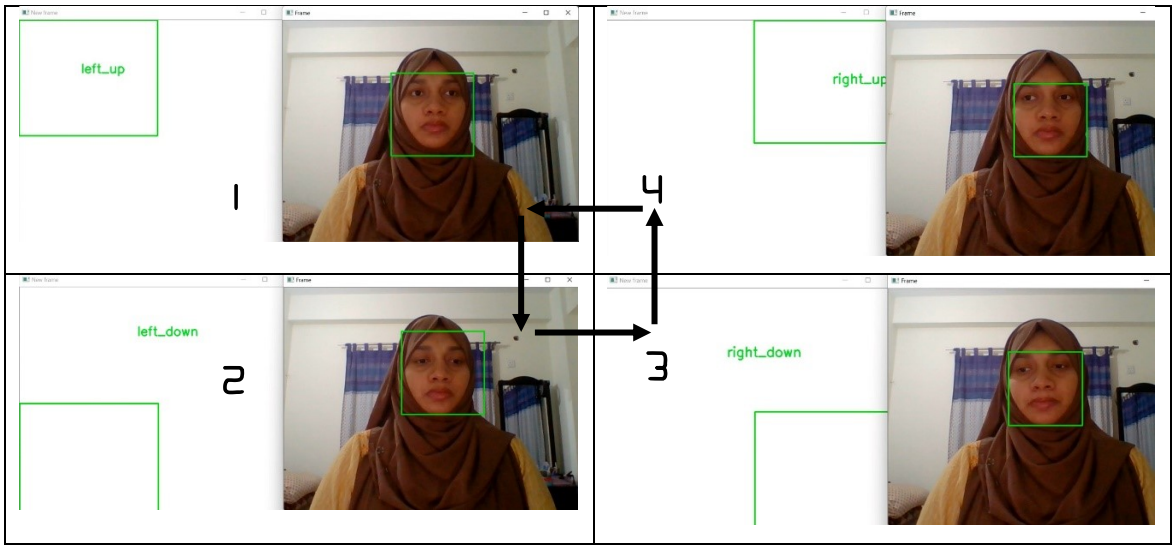


Fig. 5.6 Anti-Clockwise Pattern

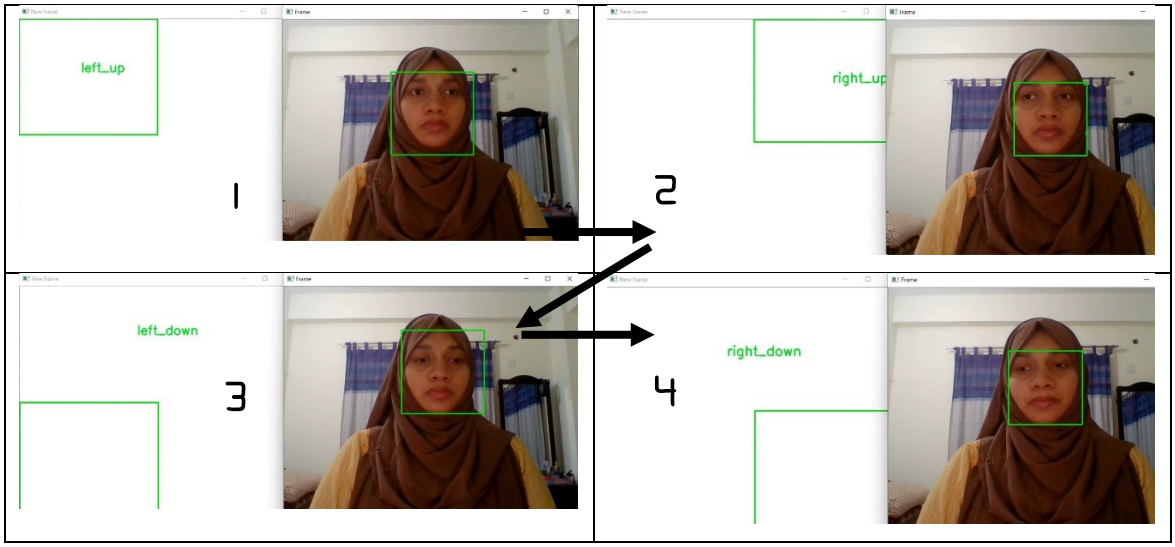


Fig. 5.7 Z Pattern

5.2 Result Analysis

To determine the performance of the SVM model for determining screen regions with eye tracking, a test dataset of 100 samples were chosen. The expected output of the test dataset can be seen in Table 5.1:

Table 5.1: Summary of test dataset

Class	No of Sample
Left Up	26
Right Up	20
Right Down	17
Left Down	37

After predicting the screen region from the test dataset with SVM multiclass classifier, the performance of the model was calculated.

5.2.1 Confusion Matrix

To measure the performance of the machine learning model that utilizes support vector machines (SVM) as its underlying algorithm, a multiclass confusion matrix was created using the model's predicted outputs. (Table 5.2).

Table 5.2: Multiclass confusion matrix for all four regions

		Expected			
Predicted		Left_Up	Right_Up	Right_Down	Left_Down
	Left_Up	23	0	0	0
	Right_Up	3	18	1	2
	Right_Down	0	2	16	5
	Left_Down	0	0	0	30

Some observations can be made from the confusion matrix. They are:

- The diagonal elements represent the correctly predicted outcomes. The overall accuracy of the system was found to be 87%.
- The model did not confuse any sample originally belonging to Left_Up with Right_Down or Left_Down. That means the boundary between Left_Up and Right_Down was well learned by the model. Similarly, the boundary between Left_Up and Left_Down was also learned well by the classifier.
- To improve the performance of the classifier, the predictive results of Left_Down can be improved further as 7 out of 37 (18.92%) samples belonging to this class were misclassified.

To better understand the performance of the model for individual regions, the combined confusion matrix was summarized into smaller one-vs-all confusion matrixes for individual regions.

5.2.2 Performance on Individual Regions

For Left_Up region, no false-positive prediction was observed in the one-vs-all confusion matrix (Table 5.3). The performance metrics observed are provided below.

Table 5.3: One-vs-all confusion matrix for detection of Left Up region

	Expected		
Predicted		Left_Up	Not Left_Up
	Left_Up	23 (TP)	0 (FP)
	Not Left_Up	3 (FN)	74 (TN)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{23 + 74}{23 + 3 + 0 + 74} = 0.97$$

$$Precision = \frac{TP}{TP + FP} = \frac{23}{23 + 0} = 1$$

$$Recall = \frac{TP}{TP + FN} = \frac{23}{23 + 3} = 0.88$$

For Right Up region, the one-vs-all confusion matrix is shown in Table 5.4. The performance metrics observed are provided below.

Table 5.4: One-vs-all confusion matrix for detection of Right Up region

		Expected	
Predicted		Right_Up	Not Right_Up
	Right_Up	18	6
	Not Right_Up	2	74

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{18 + 74}{18 + 74 + 6 + 2} = 0.92$$

$$Precision = \frac{TP}{TP + FP} = \frac{18}{18 + 6} = 0.75$$

$$Recall = \frac{TP}{TP + FN} = \frac{18}{18 + 2} = 0.90$$

For Right Up region, the one-vs-all confusion matrix is shown in Table 5.5. The performance metrics observed are provided below.

Table 5.5: One-vs-all confusion matrix for detection of Right_Down region

		Expected	
Predicted		Right_Down	Not Right_Down
	Right_Down	16	7
	Not Right_Down	1	76

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{16 + 76}{16 + 76 + 7 + 1} = 0.92$$

$$Precision = \frac{TP}{TP + FP} = \frac{16}{16 + 7} = 0.70$$

$$Recall = \frac{TP}{TP + FN} = \frac{16}{16 + 1} = 0.94$$

For Left Up region, no false-positive prediction was observed in the one-vs-all confusion matrix (Table 5.6). The performance metrics observed are provided below.

Table 5.6: One-vs-all confusion matrix for detection of Left_Down region

		Expected	
Predicted		Left_Down	Not Left_Down
	Left_Down	30	0
	Not Left_Down	7	63

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{30 + 63}{30 + 63 + 0 + 7} = 0.93$$

$$Precision = \frac{TP}{TP + FP} = \frac{30}{30 + 0} = 1$$

$$Recall = \frac{TP}{TP + FN} = \frac{30}{30 + 1} = 0.97$$

The compiled values of metrics from all four classes show that the model works better for Left Up region of the screen, while the classifier needs to be trained better for Right Up and Right Down region (Table 5.7).

Table 5.7: Combined list of accuracy, precision and recall for all four classes

Class/Region	Accuracy	Precision	Recall
Left_Up	0.97	1	0.88
Right_Up	0.92	0.75	0.90
Right_Down	0.92	0.70	0.94
Left_Down	0.93	1	0.97

5.3 Benchmark Testing of the Eye Tracking Application

In order to run benchmarking tests on the application, a test environment was set up with similar specifications as pointed in Table 5.1 for system test. After initial setup of the test system, the following benchmarking tests were conducted.

5.3.1 Eye Tracking Speed Test

The time needed for capturing the facial image from webcam and predicting the region of the screen can be seen in Fig. 5.8. The average time to detect the screen region from eye tracking is near to 1.30 seconds, which is satisfactory.

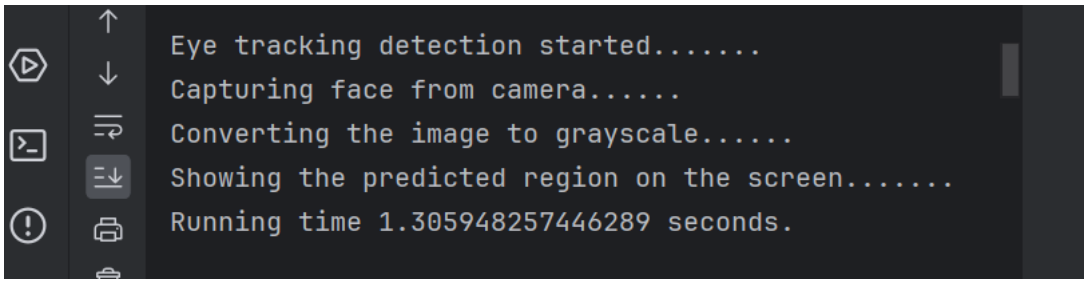


Fig. 5.8: Eye tracking detection time

5.3.2 Long Term Stability Test

The long-term stability and reliability of the eye tracking software was evaluated by conducting prolonged testing sessions over an extended period. This helps in identifying any potential drift or degradation in tracking performance over time. Fig. 5.9 shows that the performance of the eye tracking application retains its running speed even after almost 2 hours of operation.

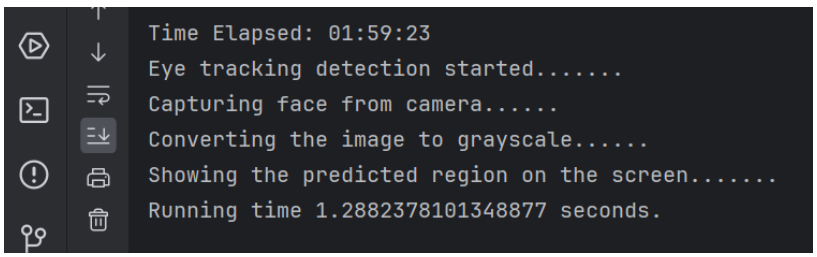


Fig. 5.9: Long term stability test

5.4 Summary

The testing and analysis phase of the eye tracking and on-screen pointing system for physically impaired individuals was conducted meticulously to assess its efficacy and potential in real-world scenarios. Through rigorous testing, we evaluated the system's performance and reliability, considering variations in eye landmark point data and lighting conditions.

Overall, this section provides a comprehensive overview of our testing, analysis, and outlines potential directions for future development and integration into broader systems.

CHAPTER 6
CONCLUSION

This chapter describes the overall thesis by using four subsections to summarize the thesis's outcome, contribution of the research, limitations, and scope for future work.

6.1 Project Outcome

The expected outcome of this thesis was to prepare a system which can provide quadriplegic patients an easier way for communicating with the outside world with the help of eye gaze tracking. Before building the project, literature review was done to understand the available solutions and research done in this area. Limitations were listed and system architecture was sketched. After that a python-based application has been developed to track eye gaze and display necessary message. Finally, the system was tested and result was analyzed.

6.2 Project Contribution

The main contribution of this project was to make a system which can track the eye gaze of physically impaired people and express their need easily. After doing literature review and knowing about the solutions, a cost-effective model has been designed. Then with the help of machine learning algorithm SVM its performance has been elevated. The proposed system can be a solution to the patients who have limited physical movement for communicating with the outer world and also provide a way to avail modern technology.

6.3 Limitations

The limitation of this project work are as follows:

- Dataset Limitations: Data from quadriplegic patients or impaired people was excluded from this project due to the challenges associated with collecting data from these sensitive individuals.
- Environmental Limitation: Data were collected in normal environment not in hospital environment. Lighting conditions varied in some data.

6.4 Future Works

This project was designed in a small scope. But it can be incorporated with bigger solutions making it special user friendly. It can be implemented in a large scale to help the physical impaired people and get them involved in the technological field. The future works of this project includes:

- To incorporate wearable device option for eye tracking.
- To add more functionalities using eye tracking like on screen typing, on screen scrolling etc.
- To use this as a tool which can tell us about user engagement to a digital content.
- To use this as an assistive tool for impaired patients. Using this tool patients can ask whatever they want or they can seek help from concerned persons.

REFERENCES

Bissoli, A., Lavino-Junior, D., Sime, M., Encarnação, L., & Bastos-Filho, T. (2019). A human–machine interface based on eye tracking for controlling and monitoring a smart home using the internet of things. Sensors, 19(4), 859.

Bozomitu, R. G., Păsărică, A., Tărniceriu, D., & Rotariu, C. (2019). Development of an eye tracking-based human-computer interface for real-time applications. Sensors, 19(16), 3630.

Diego-Mas, J. A., Garzon-Leal, D., Poveda-Bautista, R., & Alcaide-Marzal, J. (2019). User-interfaces layout optimization using eye-tracking, mouse movements and genetic algorithms. Applied ergonomics, 78, 197-209.

Eye Tracking Through History (2014) [Online]. Available: https://medium.com/@eyesee/eye-tracking-through-history-b2e5c7029443. [9 May,2023].

Fahimipirehgalin, M., Loch, F., & Vogel-Heuser, B. (2020). Using eye tracking to assess user behavior in virtual training. In Intelligent Human Systems Integration 2020, pp. 341-347.

Hasnin, M., Afrose, I. A., Himel, S. R., Suha, S. A., & Islam, M. N. (2023). A CNN Based Sign Language Learning System For Deaf & Mute User, 2023 IEEE 11th Region 10 Humanitarian Technology Conference (R10-HTC), pp. 237-242.

Huang, L., Xu, C., Westin, T., Dupire, J., Le Lièvre, F., & Shi, X. (2022, October). A Study of the Challenges of Eye Tracking Systems and Gaze Interaction for Individuals with Motor Disabilities, pp. 396-411.

Khan, N.I., Mahmud, T., Islam, M.N., and Mustafina, S.N. (2020). Prediction of Cesarean Childbirth using Ensemble Machine Learning Methods, Proceedings of the 22nd international conference on information integration and web-based applications & services, pp. 331-339.

Khan, N.S., Muaz, M.H., Kabir, A., and Islam, M.N. (2019). A Machine Learning-Based Intelligent System for Predicting Diabetes, International Journal of Big Data and Analytics in Healthcare (IJBDAH), Vol. 4, No. 2, pp. 1- 20.

Maus, N., Rutledge, D., Al-Khazraji, S., Bailey, R., Alm, C. O., & Shinohara, K. (2020, April). Gaze-guided magnification for individuals with vision impairments, pp. 1-8

Muaz, M. H., Islam, K. A., & Islam, M. N. (2021). Assessing the usability of truck hiring mobile applications in bangladesh using heuristic and semiotic evaluation, Advances in Design and Digital Communication: Proceedings of the 4th International Conference on Design and Digital Communication, pp. 90-101.

Suha, S. A., Islam, M. N., Akter, S., Bhowmick, M. C., & Halder, H. (2022). Assessing usability of mobile applications developed for autistic users through heuristic and semiotic evaluation, Proceedings of International Joint Conference on Advances in Computational Intelligence: IJCACI 2021, pp. 25-39.

Teng, G., He, Y., Zhao, H., Liu, D., Xiao, J., & Ramkumar, S. (2020). Design and development of human computer interface using electrooculogram with deep learning. Artificial intelligence in medicine, 102, 101765.

Preetha, S., Manohar, A., HM, A.A., and Tarikere, N.Y. (2020). A REVIEW OF MACHINE LEARNING ALGORITHMS IN HEALTHCARE, International Journal of Recent Trends in Engineering and Research, Vol. 06, No. 05, pp. 44-51.

Qureshi, K.N., Din, S., Jeon, G., and Piccialli, F. (2020). An accurate and dynamic predictive model for a smart M-Health system using machine learning, Information Sciences, Vol. 538, pp. 486-502.

APPENDIX A

PROJECT IMPLEMENTATION CODE

```
import cv2
import numpy as np
import dlib
import csv
import matplotlib.pyplot as plt
from math import hypot
from pandas import read_csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
```

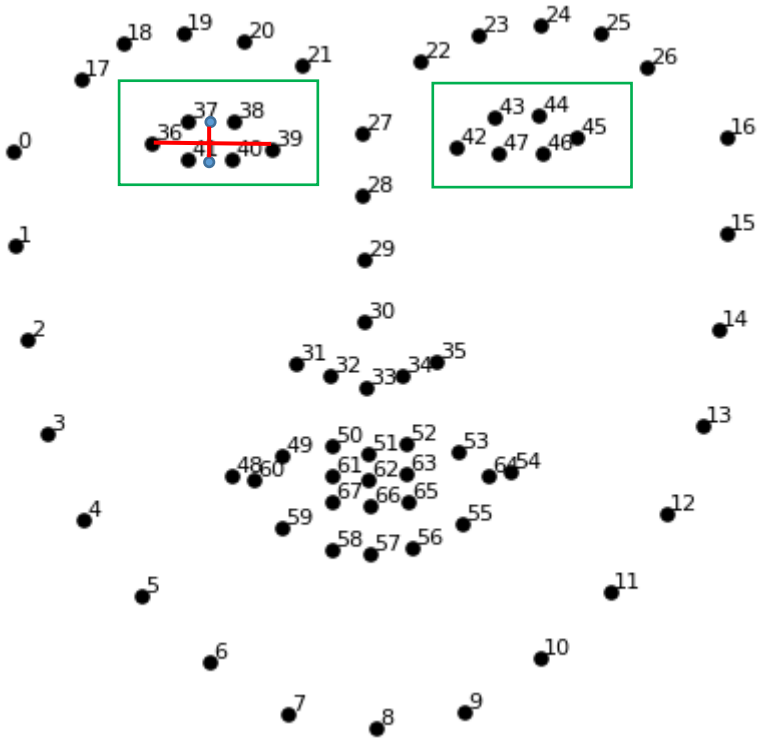
In this project, cv2, numpy, dlib, math, pandas, sklearn has been used.

```
from pynput.mouse import Button, Controller
mouse = Controller()

dataset = []
temp = []

cap = cv2.VideoCapture(0)

detector = dlib.get_frontal_face_detector()

font = cv2.FONT_ITALIC
```

The pynput.mouse is a whole package containing function for mouse control. Two empty arrays have been declared as dataset and temp. Then using a cv2 function named VideoCapture we are taking computer's default webcam video as an input. Using Dlib's face detector package frontal face is being detected from the input video. Then message font is being set to ITALIC.

```
def get_blinking_ratio(eye_points, facial_landmarks):
    left_point = (facial_landmarks.part(eye_points[0]).x, facial_landmarks.part(eye_points[0]).y)
    right_point = (facial_landmarks.part(eye_points[3]).x, facial_landmarks.part(eye_points[3]).y)

    center_top = midpoint(facial_landmarks.part(eye_points[1]), facial_landmarks.part(eye_points[2]))
    center_bottom = midpoint(facial_landmarks.part(eye_points[5]), facial_landmarks.part(eye_points[4]))

    #hor_line = cv2.line(frame, left_point, right_point, (0, 255, 0), 2)
    #ver_line = cv2.line(frame, center_top, center_bottom, (0, 255, 0), 2)

    hor_line_length = hypot((left_point[0] - right_point[0]), (left_point[1] - right_point[1]))
    ver_line_length = hypot((center_top[0] - center_bottom[0]), (center_top[1] - center_bottom[1]))

    ratio = hor_line_length / ver_line_length
    return ratio
```

In the above function, it is detected that if the person sitting in front of the webcam is blinking his eyes or not. Using facial landmarks, the computer recognizes a face and then processes the instructed steps. In the below picture, there are 67 points. In this project only the green marked points will be considered.



At first, two-line length has been taken for each eye. If we consider left eye, there are six points to consider - 36, 37, 38, 39, 40, 41. The line connecting point (36,39) and line connecting mid points of (37,38) and (41,40) are concerned two lines. In the above image lines have been shown in red color. The ratio of these two lines was considered. If the ratio falls under a certain percentage, eyes are closed. That's how blinking has been detected.

```
def get_gaze_ratio(eye_points, facial_landmarks):
    global temp
    left_eye_region = np.array([(facial_landmarks.part(eye_points[0]).x, facial_landmarks.part(eye_points[0]).y),
                                (facial_landmarks.part(eye_points[1]).x, facial_landmarks.part(eye_points[1]).y),
                                (facial_landmarks.part(eye_points[2]).x, facial_landmarks.part(eye_points[2]).y),
                                (facial_landmarks.part(eye_points[3]).x, facial_landmarks.part(eye_points[3]).y),
                                (facial_landmarks.part(eye_points[4]).x, facial_landmarks.part(eye_points[4]).y),
                                (facial_landmarks.part(eye_points[5]).x, facial_landmarks.part(eye_points[5]).y)
                                ], np.int32)
```

In the gaze detection part, eye points have been passed from the main function in two segments – left segment and right segment. Here received eye point values have been taken into a numpy array.

```
if eye_points[0] == 36:
    #print("its left eye")
    #cv2.polylines(frame, [left_eye_region], True, (0, 0, 255), 3)

    temp.append(left_eye_region[0][0])
    temp.append(left_eye_region[0][1])
    temp.append(left_eye_region[1][0])
    temp.append(left_eye_region[1][1])
    temp.append(left_eye_region[2][0])
    temp.append(left_eye_region[2][1])
    temp.append(left_eye_region[3][0])
    temp.append(left_eye_region[3][1])
    temp.append(left_eye_region[4][0])
    temp.append(left_eye_region[4][1])
    temp.append(left_eye_region[5][0])
    temp.append(left_eye_region[5][1])
```

Then the pixel value of that corresponding point (x,y) has been stored in an excel.

```
height, width, _ = frame.shape                                                    ⚠8 ⚠39
mask = np.zeros((height, width), np.uint8)

cv2.polylines(mask, [left_eye_region], True, 255, 2)
cv2.fillPoly(mask, [left_eye_region], 255)
eye = cv2.bitwise_and(gray, gray, mask=mask)

min_x = np.min(left_eye_region[:, 0])
max_x = np.max(left_eye_region[:, 0])
min_y = np.min(left_eye_region[:, 1])
max_y = np.max(left_eye_region[:, 1])

gray_eye = eye[min_y: max_y, min_x: max_x]
#gray_eye = cv2.cvtColor(eye, cv2.COLOR_BGR2GRAY)
_, threshold_eye = cv2.threshold(gray_eye, 70, 255, cv2.THRESH_BINARY_INV)
height, width = threshold_eye.shape
left_side_threshold = threshold_eye[0: height, 0: int(width / 2)]
left_side_white = cv2.countNonZero(left_side_threshold)

right_side_threshold = threshold_eye[0: height, int(width / 2): width]
right_side_white = cv2.countNonZero(right_side_threshold)

gaze_ratio = left_side_white / right_side_white
return gaze_ratio
```

First, a mask (a black image same size of our frame) is designed so that only eye region can be separated from the whole face. Here minimum and maximum pixel value of eye region has been determined with the help of numpy min and max function. Only the eye region part has been taken into gray_eye. The thresholding has been done to gray_eye image so that the image becomes binary. Then on that binary output cv2.nonzero function was used to count the non-zero values on the left and right. If the person is looking to the left, there will be more non-zero values on right and visa versa. A ratio of left side non-zero values and right side non-zero values has been taken and based on that it is determined whether the person is looking into left or right.

```
iris_data = read_csv('new_file4.csv')
dataset = iris_data.dropna()
D = dataset.values
x = D[:, 0:24]
y = D[:, 24]
x_tr, x_ts, y_tr, y_ts = train_test_split(x, y, test_size=0.20)

model = SVC()
model.fit(x_tr, y_tr)
```

Then values stored are used to train a SVM which will tell on which side a person in front of the webcam is looking into.

```
while True:
    temp = []

    _, frame = cap.read()
    new_frame = np.zeros((500, 600, 3), np.uint8)
    new_frame[:] = (255, 255, 255)

    # showing white background
    # path
    path = r'D:\Personal (Annatoma)\Python\New 24.2.2021\eye_detection_p1\white_back_500900.jpg'

    # Reading an image in default mode
    image = cv2.imread(path)

    # Window name in which image is displayed
    window_name = 'Image'

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

    def midpoint(p1, p2):
        return int((p1.x + p2.x)/2), int((p1.y + p2.y)/2) #cant be float
```

Here two frames have been declared. On one the captured video will be shown and on the other a white background with four regions will be shown so that it can be detected in which region the person is looking at. Then a midpoint determining function was defined.

```
faces = detector(gray)
for face in faces:
    x, y = face.left(), face.top()
    x1, y1 = face.right(), face.bottom()
    cv2.rectangle(frame, (x, y), (x1, y1), (0, 255, 0), 2)
```

Here the faces have been stored in faces array.

```
   #detect Blinking

   left_eye_ratio = get_blinking_ratio([36, 37, 38, 39, 40, 41], landmarks)

   right_eye_ratio = get_blinking_ratio([42, 43, 44, 45, 46, 47], landmarks)

   blinking_ratio = (left_eye_ratio+right_eye_ratio) / 2
```

For blinking detection. Left eye and right eye landmark points have been passed into the blinking function and the return value is stored in a variable. Then the average of those two values is the blinking ratio.

```
#gaze detection
gaze_ratio_left_eye = get_gaze_ratio([36, 37, 38, 39, 40, 41], landmarks)
#cv2.putText(frame, str(gaze_ratio_left_eye), (200, 400), font, 2, (0, 0, 255), 3)
gaze_ratio_right_eye = get_gaze_ratio([42, 43, 44, 45, 46, 47], landmarks)
#cv2.putText(frame, str(gaze_ratio_right_eye), (200, 450), font, 2, (0, 0, 255), 3)
gaze_ratio1 = (gaze_ratio_right_eye + gaze_ratio_left_eye) / 2

if gaze_ratio1 < 0.70:
    #new_frame[:] = (255, 0, 0)
    temp.append(1)
    temp.append(gaze_ratio1)
    cv2.rectangle(new_frame, (0, 0), (300, 250), (0, 255, 0), 2)
    cv2.putText(frame, "Left up", (50, 100), font, 2, (0, 0, 255), 3)
    mouse.position = (30, 30)

elif 0.75 < gaze_ratio1 < 1.10:
    #new_frame[:] = (255, 0, 0)
    temp.append(4)
    temp.append(gaze_ratio1)
    cv2.rectangle(new_frame, (0, 250), (300, 500), (0, 255, 0), 2)
    cv2.putText(frame, "Left down", (50, 100), font, 2, (0, 0, 255), 3)
    mouse.position = (30, 700)
```

Same as blinking_ratio, eye landmark points have been passed to gaze detection function one by one. Then the return values were stored and average of them has been taken. Then using if-else the gazing region has been determined while building the training data set. Here the regions have been marked as below:

Left-Up →1	Right-Up→2
Left-Down→4	Right-Down→3

```
######## SVM test start ##########
test_pd = pd.DataFrame(temp)
predicted_value = int(model.predict(test_pd.T).tolist()[0])
print(predicted_value)


############ SVM test ends ##############

if predicted_value == 1:
    cv2.rectangle(new_frame, (0, 0), (300, 250), (0, 255, 0), 2)
    cv2.putText(frame, "Left up", (50, 100), font, 2, (0, 0, 255), 3)
    mouse.position = (30, 30)
elif predicted_value == 4:
    cv2.rectangle(new_frame, (0, 250), (300, 500), (0, 255, 0), 2)
    cv2.putText(frame, "Left down", (50, 100), font, 2, (0, 0, 255), 3)
    mouse.position = (30, 700)
```

First the testing data has been loaded. Then Predicted value as per the training data has been shown. If the predicted value is 1, the region is Left-Up. If the predicted value is 4, the region is Right-Down.

```
        elif predicted_value == 2:
            cv2.rectangle(new_frame, (301, 0), (600, 250), (0, 255, 0), 2)
            cv2.putText(frame, "Right Up", (50, 100), font, 2, (0, 0, 255), 3)
            mouse.position = (1250, 30)
        elif predicted_value == 3:
            cv2.rectangle(new_frame, (301, 250), (600, 500), (0, 255, 0), 2)
            cv2.putText(frame, "Right Down", (50, 100), font, 2, (0, 0, 255), 3)
            mouse.position = (1250, 700)

        if blinking_ratio > 4.0:
            #new_frame[:] = (255, 0, 255)
            cv2.putText(frame, "BLINKING", (50, 50), font, 2, (0, 0, 255), 3)

    cv2.imshow("Frame", frame)
    cv2.imshow("New frame", new_frame)

    key = cv2.waitKey(1)
    if key == ord("s"):
        break

cap.realease()
cv2.destroyAllWindows()
```

Lastly, the based on the blinking_ratio value, blinking is detected. The video frame and the output result has been shown. At the end all frames have been destroyed to clear the memory.