OPERATING COMPUTER CURSOR USING EYE AND FACE MOVEMENTS

A Project report submitted in partial fulfilment of the

requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

P. Pravallika(20NM1A05D9)

S. Devi Vinuthna(20NM1A05H1)

R. Harshini(20NM1A05F3)

P. Charvi Dedeepya(20NM1A05D8)

Under the guidance of

Mrs. Sradhanjali Pattanaik

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

VIGNAN'S INSTITUTE OF ENGINEERING FOR WOMEN (A)

Approved by AICTE, New Delhi & Affiliated to JNTU-GV, Vizianagaram) Estd. 2008

Kapujaggarajupeta, VSEZ (Post), Visakhapatnam-530 049

2024

VIGNAN'S INSTITUTE OF ENGINEERING FOR WOMEN (A)
Approved by AICTE, New Delhi & Affiliated to JNTU-GV, Vizianagaram) Estd. 2008
Kapujaggarajupeta, VSEZ (Post), Visakhapatnam-530 049

DEPARTMENT OF
COMPUTER SCIENCE &ENGINEERING

CERTIFICATE

This is to certify that the project report titled **"OPERATING COMPUTER CURSOR USING EYE AND FACE MOVEMENTS"**is being submitted to Department of CSE by **P. PRAVALLIKA(20NM1A05D9), S. DEVI VINUTHNA(20NM1A05H1), R.HARSHINI (20NM1A05F3), P. CHARVI DEDEEPYA (20NM1A05D8)** during the study of IV B.Tech II Semester of **Bachelor of Technology** in **COMPUTER SCIENCE & ENGINEERING** during the period **DECEMBER 2023 – APRIL 2024** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in COMPUTER SCIENCE & ENGINEERING. This project report has not been previously submitted to any other University/Institution for the award of any other degree.

INTERNAL GUIDE **HEAD OF THE DEPARTMENT**

EXTERNAL EXAMINER

DECLARATION

We hereby declare that this project entitled "**OPERATING COMPUTER CURSOR USING EYE AND FACE MOVEMENTS**" is the original work done by us in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science & Engineering, Jawaharlal Nehru Technological University, Vizianagaram. This project work/project report has not been previously submitted to any other university/Institution for the award of any other degree.

P.Pravallika(20NM1A05D9)
S.Devi Vinuthna(20NM1A05H1)
R.Harshini (20NM1A05F3)
P.Charvi Dedeepya (20NM1A05D8)

ACKNOWLEDGEMENT

We avail this opportunity to thank the persons with whom we have been associated the completion of our project. It has been an experience of great pleasure and satisfaction to us while performing this project.

We gratefully acknowledge the patronage extended by our honorable Principal **Dr. B. Arundathi**. Our sincere thanks are also due to other faculty members who have extended their support to during our project.

Our special thanks to **Dr. P. Vijaya Bharati, Professor & HOD**, Department of Computer Science & Engineering for providing us with the requiredsoftware and hardware and guide us in doing this project work.

We express our deep sense of gratitude and indebtedness to our guide **Mrs.Sradhanjali Pattanaik, Assistant Professor**, Department of Computer Science & Engineering, Vignan's Institute of Engineering for Women, for her valuable guidance and co-ordination throughout the completion of this project.

P. Pravallika

S. Devi Vinuthna

R. Harshini

P. Charvi Dedeepya

TABLE OF CONTENTS

ABSTRACT

The advent of modern human computer interfaces has seen a considerable progress in Hands-free Human Computer Interaction (HCI) solutions. This project focuses on developing a methodology to facilitate computer cursor control for people with physical disabilities such as Quadriplegics and amputees. The proposed methodology takes real-time video input from the user using OpenCV and performs face recognition. The 68-point landmark algorithm is used to locate the various facial features which can be used for cursor control. Opening/closing the mouth based on Mouth Aspect Ratio (MAR) indicates activation/deactivation of the cursor control. The nose tip is used for controlling and moving the cursor in all 4 directions by moving the head left, right, up and down. Eye Aspect Ratio (EAR) is used to detect eyes and eye flickering. Left and right eye blinks indicate left and right clicks respectively. Squinted eyes indicate scrolling of pages, which is beneficial while working with PDFs and other such documents. The proposed system requires very basic requirements like webcam and a few Python libraries such as OpenCV, Numpy, imutils, dlib and PyAutoGUI. Thus, it would help the physically disabled users to efficiently use the computer, thus eliminating the need of a physical mouse interaction.

Keywords: Eye Tracking, Virtual Mouse, web-cam based interaction, Human-computer interaction, OpenCV, 68point landmark algorithm, MAR, EAR.

LIST OF FIGURES

LIST OF TABLES

LIST OF SCREENS

CHAPTER 1
INTRODUCTION

1. INTRODUCTION

Computers have become an indispensable part of daily life. The standard method for navigating the screen remains the traditional computer mouse. However, for individuals dealing with quadriplegia, limb loss, or Locked-in syndrome (LIS), these tools prove impractical. Existing systems, such as eye movement-based cursor controls, offer a means to interact without physical contact. Yet, their inefficiency arises from an inability to differentiate deliberate from unintentional eye movements. Recognizing this limitation, the necessity for a hands-free cursor control system became apparent. Hence, we present a novel approach utilizing eye movements to precisely manage the computer cursor. This innovation promises a practical, convenient, and empowering solution for users facing physical challenges, enhancing their digital experience with ease. Thus we have proposed a methodology for operating computer cursor using eye and face movements, which facilitates operations such as left click, right click, move the cursor up, down, left, right, scrolling up and down.

In today's digital age, computers have become indispensable tools in various aspects of daily life, from communication and work to entertainment and education. The standard method for navigating the computer screen remains the traditional computer mouse, providing users with precise control and interaction. However, for individuals facing physical challenges such as quadriplegia, limb loss, or Locked-in syndrome (LIS), using conventional input devices like the mouse proves impractical or impossible. While existing systems, such as eye movement-based cursor controls, offer a means for individuals with physical disabilities to interact with computers without physical contact, they often suffer from inefficiencies. One of the primary challenges arises from the inability to differentiate deliberate eye movements from unintentional ones, leading to inaccuracies and frustration for users. Recognizing this critical limitation, there arises a pressing need for a hands-free cursor control system that can accurately interpret and respond to the user's intentions. In response to this need, we present a novel approach to cursor control that utilizes eye movements as a precise means of managing the computer cursor. This innovative solution aims to address the shortcomings of existing systems by providing users with a practical, convenient, and empowering method for navigating digital interfaces without the need for physical input devices.

The proposed methodology for operating the computer cursor using eye and face movements offers a comprehensive suite of functionalities designed to meet the diverse needs of users with physical challenges. These functionalities include left-clicking, right-clicking, as well as moving the cursor in all directions—up, down, left, and right—enabling seamless navigation across the screen. Additionally, the system facilitates scrolling up and down, further enhancing the user's ability to interact with digital content effortlessly.

By harnessing the power of eye and facial movements, this innovative approach promises to

revolutionize the digital experience for individuals facing physical challenges. It empowers users to navigate computer interfaces with ease and precision, enabling greater independence and inclusivity in their interactions with technology. Ultimately, the proposed methodology for hands-free cursor control represents a significant step towards enhancing accessibility and usability in computing, reaffirming the transformative potential of technology to improve the lives of individuals with disabilities.

1.1 MOTIVATION

Empowerment through Technology:

By developing a hands-free HCI solution tailored to the needs of quadriplegics and amputees, you're empowering these individuals to regain control over their digital lives. This technology not only facilitates computer cursor control but also fosters independence and enhances their ability to communicate, work, and engage with the world.

Inclusivity and Accessibility:

Accessibility should be at the forefront of technological advancements, and your project embodies this principle. By utilizing OpenCV and facial recognition algorithms, you're creating a system that accommodates diverse physical abilities, ensuring that everyone, regardless of their limitations, has equal access to the digital realm.

Enhancing Quality of Life:

For individuals with physical disabilities, simple tasks like operating a computer cursor can be daunting challenges. Your project alleviates this burden, offering a solution that seamlessly integrates with their natural movements and gestures. This not only improves efficiency but also enhances their overall quality of life by opening up avenues for education, employment, and social interaction.

Cost-Effective and Scalable:

One of the remarkable aspects of your project is its simplicity and accessibility. By requiring only basic hardware and Python libraries, you're creating a solution that is not only cost-effective but also easily deployable on a wide scale. This means that individuals in diverse geographic locations and socioeconomic backgrounds can benefit from this technology without significant barriers to entry.

Pushing the Boundaries of Assistive Technology:

Your project represents a significant advancement in assistive technology, pushing the boundaries of what's possible in terms of hands-free HCI solutions. By integrating sophisticated algorithms with user-friendly interfaces, you're demonstrating the potential for technology to enhance the lives of individuals with disabilities in profound ways.

In essence, your project transcends mere technological innovation; it embodies a commitment to inclusivity, empowerment, and human dignity. By addressing the unique needs of

individuals with physical disabilities, you're not just developing a system for computer cursor control—you're building a bridge to a more equitable and accessible future for all.

1.2 PROBLEM DEFINITION

This project focuses on making it easier for people with physical disabilities like Quadriplegia, Locked-in Syndrome, or amputations to use computers. These users have trouble interacting with computers because they can only move their neck and head muscles properly. We want to create a cursor system that works mainly with eye and facial movements. This means they won't need to touch the computer to control it. Our goal is to help them navigate the digital world more easily and freely.

Individuals with severe physical disabilities, such as Quadriplegia, Locked-in Syndrome, or limb amputations, face significant challenges in interacting with computers and navigating the digital world. Traditional input devices like keyboards and mice are often inaccessible to these individuals due to their limited ability to move or control their limbs. Consequently, they experience barriers to accessing information, communication, and participating in daily activities that rely on digital technologies.

The primary issue lies in the lack of suitable input methods that cater to the specific needs and abilities of individuals with severe physical disabilities. While some assistive technologies exist, they often require manual interaction or dexterity beyond what many users can manage. Moreover, these solutions may not fully leverage the potential of emerging technologies to provide a seamless and intuitive user experience.

1.3 OBJECTIVE OF PROJECT

- The objective of this project is to develop an advanced cursor system that empowers individuals with physical disabilities, particularly those with limited mobility in their limbs, to interact with computers and digital devices effectively.
- By harnessing the capabilities of eye and facial movements, we aim to create a non-intrusive and intuitive interface that enables users to control the cursor and perform actions with precision and ease.
- To transform computer accessibility for disabled users and overcome operational challenges. Introducing a hands-free cursor control system, that can be accessed without any physical contact
- Introducing a hands-free cursor control system, that can be accessed without any physical contact.

1.4 LIMITATIONS OF THE PROJECT

Creating a system to operate a computer cursor using eye and face movements can be both innovative and challenging. Here are some considerations and potential limitations for such a project:

1. **Accuracy and Precision:** Eye and face movements can be quite nuanced, and the system would need to accurately interpret these movements to control the cursor effectively. Ensuring precise tracking and minimizing errors would be crucial.
2. **Processing Power:** Analyzing eye and face movements in real-time can be computationally intensive. The system would need sufficient processing power to perform these calculations quickly and accurately, especially if it's intended for real-time applications like gaming or video editing.
3. **Accessibility:** While eye and face tracking can offer new interaction possibilities, it may not be suitable for all users, particularly those with certain disabilities or medical conditions that affect eye or facial movements. Ensuring accessibility and providing alternative input methods would be essential.
4. **Training and Learning Curve:** Users may need time to adjust to controlling the cursor with their eye and face movements, especially if they're accustomed to traditional input methods like a mouse or touchpad. Providing adequate training and support resources could help mitigate the learning curve.

Despite these potential limitations, advancements in sensor technology, machine learning algorithms, and user interface design continue to push the boundaries of what's possible with eye and face tracking systems. With careful design and implementation, such a project could offer significant benefits in terms of accessibility, usability, and user experience.

CHAPTER 2
LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 INTRODUCTION

The utilization of eye and facial movements to operate computer interfaces has emerged as a fascinating area of research with profound implications for human-computer interaction (HCI). In recent years, advancements in sensor technology and machine learning algorithms have paved the way for novel approaches in controlling computer cursors through natural human gestures. This literature survey explores the landscape of research and development in this field, focusing on the techniques, methodologies, challenges, and applications associated with operating computer cursors using eye and face movements.

The ability to navigate digital interfaces without manual input holds tremendous promise for enhancing accessibility, particularly for individuals with physical disabilities or impairments. By harnessing the subtle movements of the eyes and face, individuals can interact with computers in a manner that is intuitive, non-intrusive, and potentially more efficient than traditional input methods. Moreover, the integration of eye and facial tracking technologies into mainstream computing devices opens up new avenues for immersive gaming experiences, hands-free control in virtual environments, and improved productivity in various professional domains.

This literature survey delves into the diverse array of approaches employed to interpret and utilize eye and facial movements for cursor control. From early research utilizing basic webcam-based tracking systems to sophisticated gaze-tracking algorithms powered by deep learning models, the evolution of this field showcases a continuous quest for accuracy, reliability, and user-friendliness. Additionally, the survey examines the challenges inherent in this technology, such as variability in individual facial expressions, lighting conditions, and the need for real-time processing capabilities.

By synthesizing existing literature and identifying gaps in knowledge, this survey aims to provide valuable insights for researchers, developers, and practitioners interested in advancing the capabilities of eye and face-based cursor control systems. Through a comprehensive understanding of the current state-of-the-art techniques and their practical implications, this survey lays the groundwork for future innovations that promise to redefine the way we interact with computers and digital interfaces.

1) **Cursor Control Using Eye Ball Movement by Vandana Khare, S.Gopala Krishna and Sai Kalyan Sanisetty**

An eyeball-based cursor system [1] developed on Raspberry Pi with the pupil as the main point. The cursor moves according to the eyeball movements. The Blinks are translated into clicks based on EAR (Eye Aspect Ratio) value. The system uses an Internet protocol camera

to take the image of an eye frame for cursor movement. The cursor will move on the screen depending on how the eye ball is moving, i.e. by mimicking the movement of eye. The user has to move his eyes and focus on where he wants the cursor to move, and it will move accordingly. Clicks are performed by eye winking.

2) **Real-Time Driver Drowsiness Detection Using Eye Aspect Ratio and Eye Closure Ratio by Sukrit Mehta, Sharad Dadhich, Sahil Gumber, Arpita Jadhav Bhatt**

The Re-Time driver drowsiness system [2] is used to detect the drowsiness level of a driver and send alarm of the same. It makes use of dlib's pre-trained face detector and then captures facial landmarks. These landmarks are used to calculate the EAR value, i.e., Eye Aspect Ratio. If the EAR is lesser than the threshold value, it would indicate a state of fatigue/drowsiness of the driver. ECR value (Eye Closure Ratio) is calculate using the sleep counter (number of times EAR is lesser than the threshold value). If the ECR value exceeds the threshold value, then an alarm is generated to indicate the drowsiness state of the driver.

3) **Facial Expression Based Computer Cursor Control System for Assisting Physically Disabled Person by M. Vasanthan, M. Murugappan, R. Nagarajan, Bukhari Ilias, J. Letchumikanth**

In this proposed system, a set of five facial expressions are used for controlling cursor movement in left, right, up and down direction and click, respectively [3] . Four luminous stickers are placed on the user's cheeks, forehead, and mouth. Movement of these markers is detected the coordinate changes on the input video and each of the five expressions is represented by five ASCII characters, which are in turn represented by binary numbers.

4) **Design of Real-time Drowsiness Detection System using Dlib by Shruti Mohanty, Shruti Hegde, Supriya Prasad and J. Manikandan**

This paper deals with driver drowsiness detection using Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) [4]. The system makes use of Dlib library for face and landmark detection. The calculated aspect ratio values are compared with the eye and mouth threshold values, i.e., if the EAR value drops down , it is considered as a state of drowsiness. IF the MAR value exceeds the threshold value, it is taken as a yawn and the corresponding alarms are fired.

2.2 EXISTING SYSTEM

The Eyeball Movement based Cursor (EMC) is an innovative approach aimed at bridging the gap between individuals with disabilities and computer systems. This system leverages advances in computer vision and hardware technologies to enable cursor control through the

movement of the user's eyes. Below is a detailed explanation of the components and functioning of the EMC system:

1. **Image Capture with Internet Protocol Camera:** The process begins with an Internet protocol (IP) camera capturing an image of the user's eye frame. IP cameras are commonly used for surveillance but are also effective for capturing high-quality images in real-time, making them suitable for eye-tracking applications.
2. **Pupil Identification using Raspberry Pi:** The captured image is then processed using a Raspberry Pi, a small and affordable single-board computer. The Raspberry Pi serves as the control point for the cursor and is responsible for identifying the user's pupil within the eye image. This identification process is crucial for accurately tracking the movement of the user's eye.
3. **Eye Photo Capture with Eye Aspect Ratio (EAR):** To facilitate accurate pupil identification, the system utilizes the Eye Aspect Ratio (EAR) threshold comparison technique. EAR is a measure derived from facial landmarks and is used to determine the openness of an eye. By comparing the EAR threshold, the system can detect when the user blinks or moves their eyes, enabling precise cursor movement control.
4. **Cursor Movement Imitating Eye's Movement:** Once the user's eye movement is detected, the cursor on the computer screen moves accordingly, imitating the direction and speed of the eye movement. This ensures that the cursor aligns with the user's gaze and follows their intended actions. Additionally, the cursor responds to the user's concentration, adjusting its movement based on the direction of the user's gaze.
5. **Click Actions with Eye Winks:** In addition to cursor movement, the system also enables click actions through eye winks. By detecting a deliberate wink of the eye, the system interprets this as a command to perform a click action, such as selecting an item or activating a function on the computer interface.

Overall, the Eyeball Movement based Cursor system offers a promising solution for individuals with disabilities by providing them with a means to interact with computer systems using natural eye movements. Through the integration of hardware components, computer vision algorithms, and user-friendly interfaces, this approach facilitates intuitive and accessible computer interaction, enhancing the overall user experience and fostering greater inclusivity in digital environments.

2.3 DISADVANTAGES OF EXISTING SYSTEM

While the Eyeball Movement based Cursor (EMC) system represents a significant advancement in enhancing accessibility for individuals with disabilities, it also faces certain

limitations and challenges that impact its efficiency and usability. Here's an elaboration on the noted drawbacks:

1. **Assuming Every Eye Movement is Intentional:** One of the primary limitations of the EMC system is its reliance on detecting every eye movement as intentional. This assumption can lead to decreased efficiency, as not all eye movements are indicative of the user's desire to interact with the computer interface. For example, involuntary eye movements, such as saccades or quick glances, may inadvertently trigger cursor movements, causing unintended actions or disruptions to the user's workflow. To mitigate this issue, the system may require more sophisticated algorithms capable of distinguishing between intentional and unintentional eye movements, thus improving overall accuracy and user experience.
2. **Difficulty in Reaching Screen Edges:** Another challenge faced by the EMC system is the difficulty in moving the cursor to the edges of the screen, particularly when navigating to elements located in the corners or edges of the interface. This limitation arises from the finite range of motion associated with eye movements, making it challenging to reach distant areas of the screen efficiently. As a result, tasks such as accessing the full-screen button or interacting with interface elements positioned at the periphery may require multiple eye movements or additional navigation techniques, potentially slowing down the user's interaction with the system. Addressing this challenge may involve implementing alternative navigation methods, such as combining eye movements with keyboard shortcuts or employing predictive algorithms to anticipate user intentions and assist in cursor positioning.
3. **Cursor Movement Interruptions during Reading:** A common issue encountered with the EMC system is the unintentional cursor movement that occurs while the user is reading content on the screen. Since the system interprets eye movements as commands to move the cursor, even minor shifts in gaze direction can trigger cursor movement, leading to distractions or difficulties in maintaining focus on the text. This phenomenon can be particularly pronounced when reading lengthy passages or scrolling through web pages, where frequent cursor movements may disrupt the user's reading flow. To address this challenge, the system may incorporate features such as gaze stabilization algorithms or user-defined preferences to reduce the sensitivity of cursor movements during reading tasks, thus providing a more seamless and uninterrupted reading experience for the user.

Overall, while the Eyeball Movement based Cursor system offers a novel approach to enhancing computer accessibility, addressing these limitations is crucial to optimizing its functionality and usability in real-world applications. By exploring alternative navigation strategies, refining gesture recognition algorithms, and incorporating user feedback,

developers can overcome these challenges and further improve the effectiveness and user-friendliness of eye-based cursor control systems.

2.4 PROPOSED SYSTEM

We've developed an innovative way to control the computer cursor using only eye and facial movements. Here's how our system works: it begins by capturing real-time video input using OpenCV. Then, it employs dlib's HOG (Histogram of Oriented Gradients) feature combined with a linear classifier, an image pyramid, and a sliding window detection scheme to detect the user's face in the video. Next, using dlib's 68-point facial landmark detector, it pinpoints specific facial features crucial for action recognition. For instance, a blink or wink is detected through the Eye Aspect Ratio (EAR), while a yawn can activate or deactivate controls using the Mouth Aspect Ratio (MAR). When the MAR value rises, it signals activation, and a click or scroll is triggered when the EAR value falls below a predefined threshold. This groundbreaking approach offers a seamless, hands-free method for users to effortlessly control the computer cursor, revolutionizing the way we interact with technology. The innovative system for controlling the computer cursor using eye and facial movements represents a significant advancement in human-computer interaction. Here's a detailed elaboration on how the system works and the actions it enables:

1. **Real-time Video Input with OpenCV:** The system begins by capturing real-time video input using OpenCV (Open-Source Computer Vision Library), a widely-used open-source library for computer vision and image processing tasks. This enables the system to continuously analyze the user's facial expressions and movements.
2. **Face Detection with dlib's HOG Feature and Linear Classifier:** Utilizing dlib's (a C++ library) HOG feature combined with a linear classifier, an image pyramid, and a sliding window detection scheme, the system detects the user's face in the video stream. The Histogram of Oriented Gradients (HOG) feature descriptor is effective for object detection tasks, such as detecting faces, while the linear classifier helps classify the detected regions as faces.
3. **Facial Landmark Detection with dlib's 68 Point Facial Landmark Detector:** Once the face is detected, dlib's 68-point facial landmark detector is employed to pinpoint specific facial features crucial for action recognition. This detector identifies key landmarks on the face, such as the eyes, nose, mouth, and eyebrows, which are essential for interpreting the user's facial expressions and movements accurately.
4. **Action Recognition using Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR):** The system leverages the detected facial landmarks to recognize various actions performed by the user. For instance, a blink or wink is detected through the Eye Aspect

Ratio (EAR), which measures the ratio of the distances between certain landmarks on the eye region. Similarly, a yawn can activate or deactivate controls using the Mouth Aspect Ratio (MAR), which indicates the openness of the mouth.

5.Cursor Control Actions Enabled by the Proposed Algorithm: The system offers a range of hands-free actions for controlling the computer cursor, including:

Squinting Eyes for Scroll Mode Activation: By squinting their eyes, the user can activate scroll mode, allowing them to scroll through documents or web pages seamlessly.

Winking for Left and Right Clicks: Winking one eye triggers a left click, while winking the other eye triggers a right click, enabling the user to perform common mouse actions without physical input.

Head Movement (Pitch and Yaw) for Cursor Movement: Moving the head around, both in pitch (up and down) and yaw (left and right), allows the user to control the cursor's movement across the screen, providing precise navigation control.

Mouth Opening for Cursor Control Activation/Deactivation: Opening the mouth activates or deactivates cursor control, allowing the user to switch between hands-free cursor control and traditional input methods as needed.

Overall, this groundbreaking approach offers users a seamless and intuitive method for interacting with computers, revolutionizing the way we navigate digital interfaces and empowering individuals with disabilities to access technology more effectively. Through the integration of advanced computer vision techniques and intelligent action recognition algorithms, the proposed system opens up new possibilities for hands-free computing and enhances the accessibility and usability of modern technology.

2.5 FEASIBILITY STUDY

Preliminary investigation examines the project feasibility, the likelihood that the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economic feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

1. Technical Feasibility
2. Operational Feasibility
3. Economic Feasibility

TECHNICAL FEASIBILITY:

Technical feasibility assesses whether the proposed project is technically possible within the existing technological infrastructure. It involves evaluating:

The availability of necessary technology: This includes assessing whether the required technology exists to implement the proposed functionalities.

Capacity of proposed equipment: This involves determining if the hardware components have the technical capacity to handle the data and processes required by the new system.

Upgradability: Evaluating whether the system can be upgraded or scaled up if needed in the future.

Technical guarantees: Ensuring aspects such as accuracy, reliability, ease of access, and data security meet the required standards. In the context of the project described, the technical feasibility is demonstrated by the use of OpenCV and Dlib libraries in Python for face detection, which indicates that the necessary technology exists. The system utilizes existing hardware and software resources available at NIC (National Informatics Centre) or are freely available as open source, suggesting minimal additional technical requirements.

The technical issues usually raised during the feasibility stage of the investigation include the following:

- Does the necessary technology exist to do what is suggested?
- Does the proposed equipment have the technical capacity to hold the data required to use the new system?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?
- Is the necessary eye-tracking and facial recognition technology available?
- Can the proposed system accurately track eye and face movements in real-time?
- Does the system have the technical capacity to translate eye and face movements into cursor actions effectively?
- Are there technical guarantees regarding the accuracy, reliability, and security of the eye-tracking and facial recognition components?

OPERATIONAL FEASIBILITY:

Operational feasibility evaluates whether the proposed system will be operationally effective and acceptable within the organization. Key considerations include:

Support from management and users: Assessing whether there is sufficient support from management and whether users are likely to embrace and effectively use the system.

Successful development and implementation: Ensuring that the system can be developed and implemented without significant issues or resistance from users.

Mitigation of user resistance: Addressing potential concerns or resistance from users that could undermine the benefits of the system.

The project aims to address these issues by considering management support and user requirements beforehand. Additionally, a well-planned design is expected to optimize

resource utilization and enhance performance.

Proposed projects are beneficial only if they can be turned out into information systems that will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised to test the operational feasibility of a project include the following:

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?
- Will users be able to adapt to controlling the computer cursor using eye and face movements?
- Is there sufficient support from management and users for implementing such a system?
- Are there any operational challenges or resistance from users that need to be addressed?
- Can the system be seamlessly integrated into existing workflows without disrupting productivity?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

ECONOMIC FEASIBILITY:

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any additional hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, there is nominal expenditure and economic feasibility for certain. Economic feasibility evaluates whether the proposed system is financially viable and beneficial for the organization. Key considerations include:

Development cost vs. benefits: Comparing the costs associated with developing and implementing the system against the anticipated benefits.

Return on investment: Ensuring that the financial benefits derived from the system outweigh the costs. In the described project, economic feasibility is demonstrated by the minimal expenditure required since the system utilizes existing resources and technologies available at NIC. The investment in the system is deemed worthwhile as it does not necessitate additional hardware or software, and the benefits are expected to exceed the costs.

- What are the development and implementation costs associated with the project?
- Are the anticipated benefits of using eye and face movements to control the cursor worth the investment?

- Will the financial benefits, such as increased productivity or accessibility, outweigh the costs over time?
- Are there any ongoing maintenance or support costs that need to be considered?

2.6 CONCLUSION

In conclusion, the development of an innovative system for controlling the computer cursor using eye and facial movements represents a significant breakthrough in human-computer interaction. By leveraging state-of-the-art computer vision techniques and intelligent action recognition algorithms, the proposed system offers users a seamless and intuitive method for interacting with digital interfaces, revolutionizing the way we navigate and interact with technology.

Through the integration of real-time video input with OpenCV, face detection using dlib's HOG feature and linear classifier, and facial landmark detection with dlib's 68-point facial landmark detector, the system accurately identifies and tracks the user's facial expressions and movements. This enables precise recognition of actions such as blinking, winking, head movement, and mouth opening, facilitating hands-free control of the computer cursor.

The system's ability to interpret specific facial features, such as the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR), enables a wide range of cursor control actions, including scroll mode activation through eye squinting, left and right clicks via winking, cursor movement with head pitch and yaw, and cursor control activation/deactivation by mouth opening.

These capabilities offer significant benefits for users, particularly those with disabilities or impairments, by providing a more accessible and inclusive means of interacting with technology. Moreover, the proposed system enhances user convenience and efficiency by eliminating the need for traditional input devices and enabling hands-free cursor control.

In summary, the development of this innovative system represents a transformative advancement in human-computer interaction, offering a seamless and intuitive method for controlling the computer cursor using only eye and facial movements. As technology continues to evolve, such advancements hold promise for reshaping the landscape of accessibility and usability in computing.

CHAPTER 3
SYSTEM ANALYSIS AND DESIGN

3. SYSTEM ANALYSIS AND DESIGN

3.1 INTRODUCTION

System analysis and design play a crucial role in the development of innovative solutions for complex problems in various domains, including human-computer interaction. In the context of the proposed system for controlling the computer cursor using eye and facial movements, thorough system analysis and design are essential to ensure the effectiveness, reliability, and usability of the solution. The development of such a system involves a structured approach that encompasses understanding user requirements, identifying system functionalities, designing the system architecture, and implementing robust algorithms and interfaces. This introduction provides an overview of the system analysis and design process for the proposed eye and facial movement-based cursor control system, highlighting key aspects and considerations involved in its development.

Understanding User Needs and Requirements:

The first step in system analysis and design is to understand the needs and requirements of the system's intended users. In the case of the proposed system, the primary users may include individuals with disabilities or impairments who seek a hands-free method of interacting with computers. Understanding their specific challenges, preferences, and expectations is essential for designing a solution that meets their needs effectively.

Functionalities and Features:

Once user requirements are identified, the next step is to define the system's functionalities and features. This involves determining the actions and interactions that the system should support, such as cursor movement, click actions, scroll mode activation, and cursor control activation/deactivation. Additionally, the system should incorporate intelligent action recognition algorithms capable of accurately interpreting eye and facial movements to perform these functionalities seamlessly.

System Architecture and Components:

Designing the system architecture involves defining the structure and components of the system, including hardware and software components. For the proposed system, hardware components may include cameras for capturing video input, while software components may include computer vision libraries such as OpenCV and dlib for facial detection and landmark detection. The system architecture should be designed to facilitate real-time processing of video input and efficient execution of algorithms for action recognition.

Algorithm Design and Implementation:

A critical aspect of system design is the design and implementation of algorithms for action recognition and cursor control. This involves developing algorithms that can accurately detect and interpret eye and facial movements, such as blinking, winking, head movement, and

mouth opening. These algorithms should be robust, efficient, and capable of operating in real-time to provide users with a seamless and responsive interaction.

User Interface Design:

The design of the user interface is another important aspect of system design, as it directly impacts the usability and user experience of the system. The user interface should be intuitive, easy to navigate, and visually appealing, allowing users to interact with the system effortlessly. It should provide feedback to users regarding their actions and the system's response, enhancing the overall interaction experience.

An effective system analysis and design are essential for the development of the proposed eye and facial movement-based cursor control system. By understanding user needs, defining system functionalities, designing the system architecture, implementing robust algorithms, and designing an intuitive user interface, the system can offer a seamless and accessible method for individuals to interact with computers using only their eye and facial movements.

3.2 SOFTWARE REQUIREMENT SPECIFICATION

3.2.1 User Requirements

User Requirements for Eye and Facial Movement-Based Cursor Control System:

Accessibility: The system should be accessible to individuals with disabilities or impairments, providing them with a hands-free method of interacting with computers. It should accommodate users with a range of abilities and disabilities, including those with limited mobility, motor control, or vision impairment.

Ease of Use: The system interface should be intuitive and easy to use, requiring minimal training for users to become proficient. Users should be able to perform cursor control actions effortlessly using only their eye and facial movements, without the need for additional physical input devices.

Accuracy and Precision: The system should accurately detect and interpret eye and facial movements to perform cursor control actions reliably. Cursor movements should be precise and responsive, allowing users to navigate digital interfaces with ease and accuracy.

Customization and Personalization: The system should offer customization options to accommodate individual user preferences and needs. Users should be able to adjust settings such as sensitivity, speed, and gesture recognition thresholds to tailor the system to their preferences.

Real-Time Performance: The system should operate in real-time, providing immediate feedback and response to user actions. It should minimize latency and processing delays to ensure a smooth and seamless interaction experience.

Versatility and Flexibility:

The system should support a variety of cursor control actions, including cursor movement, left and right clicks, scroll mode activation, and cursor control activation/deactivation. It should accommodate different types of eye and facial movements, allowing users to perform actions such as blinking, winking, head movement, and mouth opening to control the cursor.

Compatibility and Integration:

The system should be compatible with standard computer hardware and software platforms, ensuring seamless integration with existing computing environments. It should support popular operating systems and applications, allowing users to interact with a wide range of software and digital interfaces.

Reliability and Robustness:

The system should be reliable and robust, capable of operating consistently under varying conditions and environments. It should minimize false positives and false negatives in gesture recognition to prevent unintended actions and ensure user confidence in the system's performance.

By fulfilling these user requirements, the eye and facial movement-based cursor control system can offer an accessible, intuitive, and efficient means for individuals to interact with computers, empowering users with disabilities and enhancing the overall usability and inclusivity of computing technology.

3.2.2 Software Requirements

Software Requirements for Eye and Facial Movement-Based Cursor Control System:

- Operating System : Windows
- Language : Python 3.12
- Libraries : OpenCV, numpy, imutils, dlib, PyAutoGUI
- IDE : Visual Studio Code

3.2.3 Hardware Requirements

Hardware Requirements for Eye and Facial Movement-Based Cursor Control System:

- RAM : 4 GB or 8 GB
- Processor : Intel i3 and above
- Hard Disk : 500GB Minimum

About Python

Its clean syntax accelerates development, while libraries like TensorFlow, PyTorch, and Django empower tasks in AI, web development, and data science. Python's community support fosters collaboration and problem-solving, and its cross-platform compatibility

ensures seamless deployment. Python comprehensive suite of libraries, such as OpenCV and dlib, which streamline image processing and facial recognition tasks Python's cross-platform compatibility ensures seamless deployment across diverse operating systems, while its ability to interface with hardware components like webcams facilitates real-time interaction.

OpenCV

OpenCV (Open-Source Computer Vision Library) is a powerful tool for image and video processing tasks. Its extensive functionality includes features like image manipulation, object detection, and facial recognition. Specifically, it offers robust algorithms for detecting and tracking facial landmarks, such as eyes and facial contours, with high accuracy. Its efficient algorithms enable real-time processing of webcam feeds. OpenCV's cross-platform compatibility ensures seamless integration with various operating systems and hardware configurations

NumPy

NumPy, a fundamental library for numerical computing in Python. Its efficient handling of multi-dimensional arrays facilitates the manipulation and processing of image data captured by cameras. NumPy's array operations and mathematical functions enable quick and effective computations needed for tasks like image filtering, feature extraction, and data normalization. By leveraging NumPy, developers can perform complex calculations on pixel values, aiding in the precise detection and tracking of facial features such as eyes and facial landmarks. Its optimized algorithms ensure fast processing speeds, essential for real-time interaction with computer cursors based on eye and face movements. Furthermore, NumPy's seamless integration with other Python libraries, including OpenCV and scikit-image, enhances its utility in building comprehensive eye and face-tracking systems.

Imutils

Imutils provides a set of utility functions that streamline operations such as resizing, rotating, and displaying images. In eye and face-tracking applications, Imutils offers functionalities like easy resizing and manipulation of webcam frames, ensuring compatibility with different processing pipelines and algorithms. Its straightforward API simplifies the implementation of image preprocessing steps, facilitating efficient feature extraction and analysis. Imutils' compatibility with OpenCV and other image processing libraries enhances its utility in integrating with existing workflows seamlessly. Its lightweight nature and ease of use make it a valuable tool for optimizing and refining image processing pipelines, ultimately contributing to more accurate and responsive

Dlib

Dlib is a robust C++ toolkit renowned for its machine learning algorithms and tools for computer vision tasks. Its Python bindings provide access to powerful functionalities such as facial landmark detection, face recognition, and shape prediction. Dlib's facial landmark

detection capabilities are particularly valuable, allowing for precise identification and tracking of key facial features like eyes and nose. By leveraging Dlib, developers can implement sophisticated algorithms for real-time analysis of facial expressions and movements, enabling accurate control of computer cursors based on eye and face motions. Its efficient and reliable algorithms ensure high performance, essential for interactive applications requiring quick response times.

PyAutoGUI

PyAutoGUI is a Python library designed for automating tasks involving mouse movement, keyboard input, and GUI interactions. Its cross-platform compatibility ensures consistent behavior across different operating systems, making it suitable for a wide range of applications. Additionally, PyAutoGUI's simplicity and ease of use make it accessible to developers without extensive experience in low-level mouse control or GUI programming. PyAutoGUI simulate mouse clicks programmatically.

3.3 ALGORITHM

To create the suggested algorithm, the subsequent actions must be taken:

- Assign "ap" to the anchor point and "np" to the nose point.
- "w" stands for width and "h" for height.
- The ear is the EYE ASPECT RATIO, and the mouth is the mar.
- D is the distance between the horizontal landmarks of the mouth, and A, B, and C are the distances between its vertical landmarks. The Euclidean distances M, N between the vertical landmarks of the eye and P between the horizontal landmarks of the eye are respectively.

/*Real time Video Input*/

video=cv2.VideoCapture(0)

/*Image Preprocessing*/

frame=cv2.flip(frame)

frame=resize()

gray=cvtColor(frame)

/*Face Detection*/

Detector= dlib. Face_detector()

/* Landmark Estimation */

predictor = dlib.shape_predictor()

/*Aspect Ratio Calculation*/

mar= A+B+C/2D

ear=M+N/2P

/* Activating Cursor Control*/

if mar> MOUTH_AR_THRESHOLD:

then cursor_control_activated

if MOUTH_COUNTER>=MAR_CONSECUTIVE_FRAMES:

then deactivate_cursor_control

/*Comparison with threshold values and mouse action*/

if leftEAR < rightEAR:

then click(button='left')

else if leftEAR> rightEAR:

then click(button='right')

dir= direction(np,ap,w,h)

if dir== 'right':

then move right

else if dir=='left':

then move left

else if dir=='up':

    if SCROLL_MODE:

        then scroll up

    else move up

else if dir=='down':

    if SCROLL_MODE:

        then scroll down

    else move down

Breakdown of the algorithm step by step:

1. **Real-time Video Input**: A video stream is captured from the default camera (index 0) using OpenCV's VideoCapture function.
2. **Image Preprocessing:** The captured frame is flipped horizontally using cv2.flip() to correct the mirror effect. The frame is resized to a suitable size. The frame is converted to grayscale using cv2.cvtColor() for efficient processing.
3. **Face Detection:** The dlib library is used to create a face detector object Detector. This detector is applied to the grayscale frame to detect faces.
4. **Landmark Estimation:** Another dlib function predictor is used with a pre-trained shape predictor model to estimate facial landmarks. These landmarks include points around the eyes, mouth, nose, etc.
5. **Aspect Ratio Calculation:** The Mouth Aspect Ratio (MAR) and Eye Aspect Ratio (EAR) are calculated based on the detected facial landmarks. MAR is calculated as the ratio of the distance between two points on the lips to the width of the mouth. EAR is calculated

as the ratio of the distance between two points on the eye to the distance between two points on the eyebrow.

6. **Activating Cursor Control:** If the calculated MAR exceeds a predefined threshold MOUTH_AR_THRESHOLD, cursor control is activated. If the MAR remains above the threshold for a certain number of consecutive frames (MAR_CONSECUTIVE_FRAMES), cursor control is deactivated.
7. **Comparison with Threshold Values and Mouse Action**: Based on the calculated EAR values for the left and right eyes, mouse clicks are simulated. If the left eye's EAR is less than the right eye's EAR, a left-click action is triggered; otherwise, a right-click action is triggered. The direction of head movement (right, left, up, down) is determined using a custom function direction() based on the position of the nose tip relative to the previous position. Depending on the detected direction, the cursor is moved accordingly. If in scroll mode, scrolling up and down actions are also considered.

This algorithm effectively combines facial landmark detection, aspect ratio calculations, and head movement analysis to enable hands-free cursor control for users with physical disabilities. It utilizes a simple webcam and basic image processing techniques, making it accessible and easy implementation.

3.4 FLOWCHART

A flowchart is a type of diagram that, rather than emphasizing implementation, shows how a system behaves and flows. Another name for it is an object-oriented flowchart. Activities that are composed of actions that are applicable to behavioral modeling technologies make up activity diagrams. These are the graphical depictions of the sequential tasks and actions with choice, iteration, and concurrency support in the workflows.

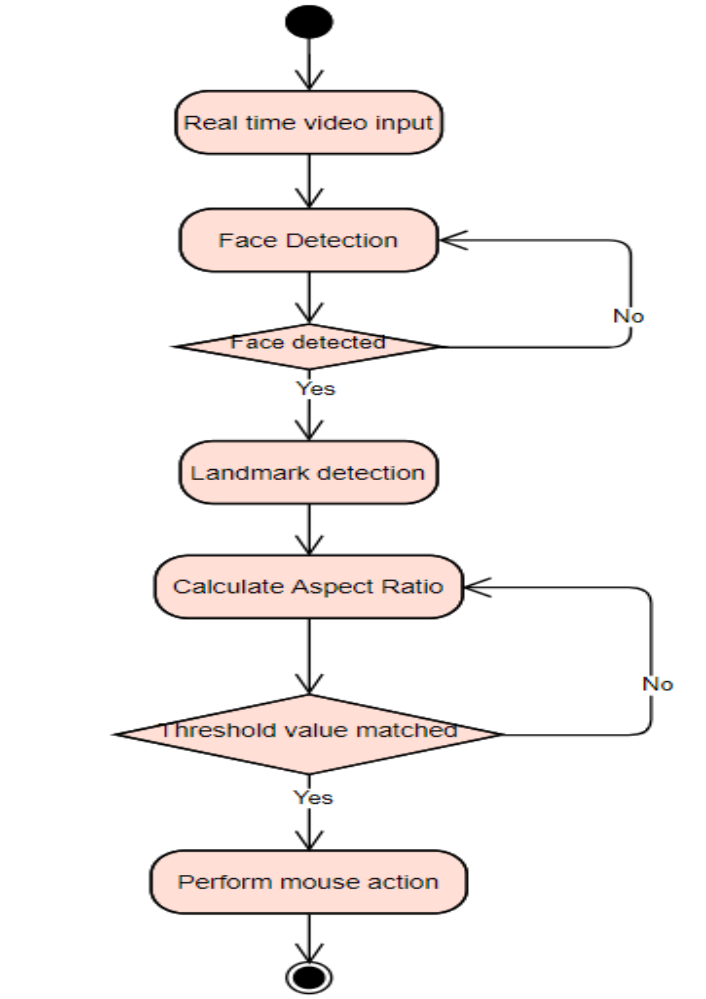

Figure 3.4.1 Flowchart of proposed system

Figure 3.4.1 shows the flow of execution of the project

Elaboration of each step of the flow chart for operating a computer cursor using eye and face movements:

1. Start: This step simply signifies the beginning of the process. It's the point where the system is initialized and begins its operation.

2. Initialize System: During initialization, the system starts up all necessary hardware components, such as cameras or sensors, that are required for eye and face tracking. This step ensures that the system is ready to begin tracking user movements.

3. Calibrate Sensors: Before tracking can commence, the sensors need to be calibrated to the specific user. Calibration involves adjusting sensor settings to account for factors like lighting conditions, user-specific characteristics (e.g., eye shape), and any environmental variables that could affect tracking accuracy. This step ensures that the system accurately interprets the user's movements.

4. Start Tracking: Once calibration is complete, the system begins tracking the user's eye and face movements in real-time. This involves continuously capturing data from the sensors

and processing it to determine the user's gaze direction, head orientation, facial expressions, and other relevant movements.

5. Detect Eye Movements: In this step, the system analyzes the input from the eye-tracking sensors to detect various types of eye movements, such as gaze direction, blinks, and saccades (rapid eye movements). Eye tracking algorithms interpret the sensor data to determine where the user is looking on the screen and how their gaze is shifting.

6. Detect Face Movements: Similarly, the system analyzes input from the face-tracking sensors to detect movements such as head orientation, facial expressions, and gestures. Facial tracking algorithms interpret the sensor data to determine the user's facial movements and expressions, which can provide additional input for cursor control or interaction.

7. Translate Movements to Cursor Control: Based on the detected eye and face movements, the system translates this input into cursor movements on the computer screen. For example, if the user shifts their gaze to the left, the system moves the cursor to the left on the screen. Similarly, tilting or rotating the head could result in corresponding cursor movements in different directions.

8. Update Cursor Position: The cursor position on the screen is updated according to the translated movements detected from the user's eyes and face. This ensures that the cursor accurately reflects the user's intended actions and remains synchronized with their movements.

9. Check for User Input: At this stage, the system checks if the user has provided any additional input, such as clicking, selecting, or performing specific gestures. This could involve analyzing additional sensor data or monitoring for predefined actions that indicate user intent.

10. Perform Actions: If the user provides input, the system performs the corresponding actions, such as clicking on an icon, dragging an object, or executing a command. These actions are based on the user's input signals and are carried out within the context of the current application or interface.

11. Loop Back to Tracking: After performing actions, the system loops back to continue tracking the user's eye and face movements for ongoing cursor control. This ensures that the system remains responsive to the user's input and can continuously update the cursor position based on their movements.

12. Stop: This step signifies the endpoint of the process, where the system stops tracking and may shut down the hardware components. This could occur when the user exits the application, turns off the system, or when tracking is no longer needed.

Each step in this elaboration contributes to the overall functionality of the system for operating a computer cursor using eye and face movements, from initialization and tracking to cursor control and user interaction.

3.5 CONCLUSION

The system proposed for hands-free cursor control utilizing eye and face movements presents a promising solution for individuals with physical disabilities, such as quadriplegics and amputees. By leveraging real-time video input and advanced computer vision techniques, the system enables users to interact with computers without the need for traditional input devices like mice or keyboards. Through a combination of facial landmark detection, aspect ratio calculations, and head movement analysis, the system accurately interprets user gestures to control the cursor in four directions, perform left and right clicks, and activate scrolling functions. Furthermore, the system incorporates a user-friendly interface with customizable threshold values and activation/deactivation mechanisms to ensure ease of use and adaptability to individual user preferences.

Analysis of Design:

Efficiency and Accuracy: The use of dlib's face detector and shape predictor, along with aspect ratio calculations, ensures robust and accurate detection of facial landmarks essential for cursor control. Real-time processing of video input allows for seamless interaction with the computer, providing users with efficient control over the cursor movements.

Accessibility and Inclusivity: The system's hands-free nature makes it highly accessible for individuals with physical disabilities, eliminating the need for manual input devices and enabling them to navigate and interact with digital content independently. By incorporating customizable threshold values and activation mechanisms, the system accommodates varying user capabilities and preferences, promoting inclusivity and user empowerment.

Scalability and Affordability: The system's reliance on basic hardware components, such as a webcam, and open-source libraries like OpenCV and dlib, makes it cost-effective and easily deployable on a wide scale. Its compatibility with standard computing platforms ensures scalability and broad applicability across diverse user environments and technological settings.

User Experience and Adaptability: The system's intuitive interface and responsive feedback mechanisms contribute to a positive user experience, facilitating seamless interaction with computer applications and minimizing user fatigue. Additionally, the system's adaptability to different user environments and lighting conditions enhances its usability and reliability in real-world settings.

Overall, the proposed system represents a significant advancement in hands-free human-computer interaction solutions, offering a practical and accessible means for individuals with physical disabilities to engage with digital technology effectively. By leveraging the

capabilities of modern computer vision techniques, the system empowers users to overcome barriers to accessibility.

CHAPTER 4
METHODOLOGY

4. METHODOLOGY

4.1 ARCHITECTURE DESIGN

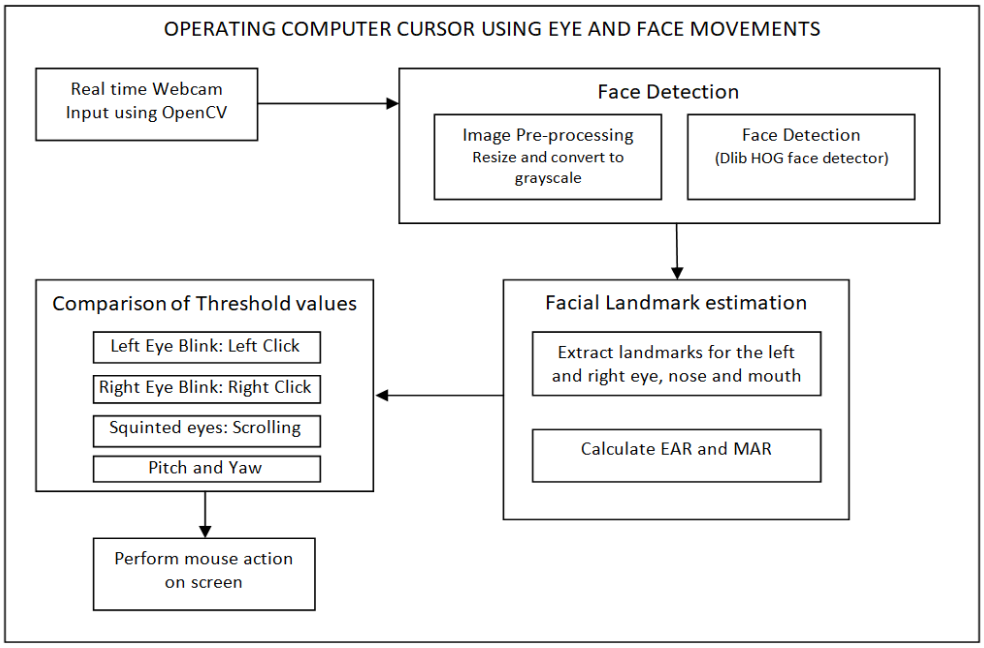

Figure 4.1.1 System Architecture

Figure 4.1.1 shows the architecture of Operating Computer Cursor using eye and Face Movements

Software architectural design serves as the bedrock of the software engineering process, irrespective of the development paradigm or application domain. It represents the crucial first step in transforming abstract system requirements into tangible, implementable solutions. At its core, software design is about creating a blueprint or model that encapsulates the structure, behavior, and interactions of the software system to be developed. This model serves as a guide for the subsequent phases of development, including coding and testing, ensuring that the final product meets the specified requirements and performs effectively.

In the software engineering lifecycle, design follows the thorough examination and definition of system requirements. Once the functional and non-functional requirements have been clearly articulated and validated, the focus shifts to designing the architecture and components of the software system. System design involves breaking down the high-level requirements into manageable units, defining interfaces, and establishing relationships between different system elements. It encompasses various aspects such as architectural design, detailed design of modules, data design, and user interface design, among others.

Architectural design lays the foundation for the entire software system, determining its overall structure and organization. It involves selecting appropriate architectural styles, patterns, and frameworks to address key concerns such as scalability, performance, and maintainability. The architectural design phase also involves identifying and defining the major components

of the system, their responsibilities, and the interactions between them.

Detailed design focuses on specifying the internal workings of individual modules or components identified during architectural design. It involves defining data structures, algorithms, and interfaces required for the implementation of each module. Through detailed design, developers create a clear roadmap for coding, ensuring that each component is well-defined, cohesive, and loosely coupled with other components.

User interface design focuses on creating intuitive and user-friendly interfaces that facilitate effective interaction between users and the software system. It involves understanding user needs, preferences, and workflows to design interfaces that are visually appealing, easy to navigate, and responsive to user input.

Overall, software design is a critical phase that sets the stage for the successful development and deployment of software systems. It bridges the gap between requirements and implementation, providing developers with a comprehensive plan to guide their coding efforts and ensuring that the resulting software meets user expectations and quality standards.

4.2 MODULES AND DESCRIPTION

In the proposed system for hands-free cursor control using eye and face movements, various modules work cohesively to achieve efficient and intuitive interaction between the user and the computer. Each module is designed to perform specific tasks, contributing to the overall functionality and effectiveness of the system. Let's explore the key modules:

1) Real Time Video Input:

The OpenCV package in Python is used to capture user-provided video input in real time. The inbuilt camera or the web camera can be utilized as the input source for the VideoCapture() function.

2) Image Pre-processing and Face Detection:

Prior to proceeding to the subsequent stage of the algorithm pipeline, the real-time video input needs to be preprocessed. The preprocessed input can then be used for face detection.

Preparing images: Pre-processing of the input is required, this includes flipping, resizing, and grayscale conversion. We must use the OpenCV flip() function to flip the frame again because it is automatically flipped when it is received from the web camera. The resize() function in imutils is used to resize. Since handling greyscale pixels is less complicated than handling colored ones, we use OpenCV cvtColor() to transform the input to greyscale.

Face Detection: The pre-processed video input is scanned for faces using Dlib's Histogram of Oriented Gradients (HOG) algorithm-based face detector. This approach builds histograms by using the gradient direction of an image in specific locations. Because HOG has a lower false

positive ratio and is more accurate than Haar cascades, it is recommended. It is also more practical because it permits the user's movements to be ignored. To turn on the face detector, simply call the get_frontal_face_detector() function.

3) Facial Landmark Estimation:

The 68-point facial landmark algorithm-based predictor from the Dlib package is employed. Around the face, its system finds 68 different landmarks. We make use of the shape_predictor() function in order to initialize it.

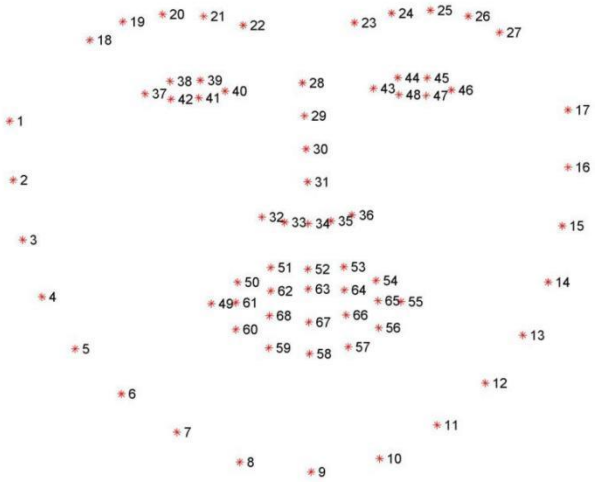

Figure 4.2.1 68 facial landmarks mapped on the face

The above figure displays the 68 unique landmarks that were determined using the 68-point landmark estimation algorithm developed by dlib. We are particularly interested in the landmarks surrounding the nose, mouth, and eyes among them.

We use OpenCV's drawContours() method to draw contours around the eyes and mouth after extracting the left and right eye coordinates, mouth coordinates, and nose pointer.

The values of the Mouth Aspect Ratio (MAR) and Eye Aspect Ratio (EAR) are computed using the retrieved landmarks.

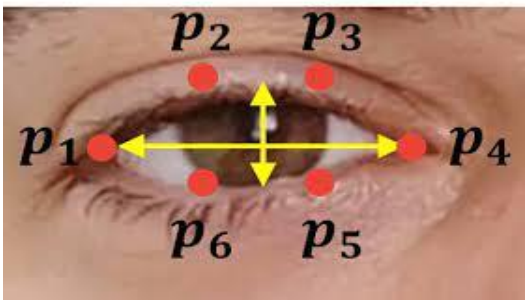

Figure 4.2.2 Landmarks around the eye

Figure 4.1.2 depicts the 6 distinct landmarks around the eyes which are used to calculate the EAR value. It has two pairs of vertical landmarks and one pair of horizontal landmarks.

EAR can be calculated using the formula: $EAR = \frac{||p_2-p_6||+||p_3-p_5|}{2*||p_1-p_4||}$

The above formula calculates the Eye Aspect Ratio (EAR), which is a measure used in facial landmark detection to assess the openness of an eye. The formula is derived from the distances between specific landmarks around the eye region. Break down of the formula and explaining each component:

- ||p2-p6||: This represents the Euclidean distance between two key points on the eye's vertical axis. Specifically, it's the distance between points 2 (the vertical midpoint between the top and bottom of the eye) and 6 (the bottom of the eye).
- ||p3-p5||: Similarly, this represents the Euclidean distance between two other key points on the eye's vertical axis. It's the distance between points 3 (the top of the eye) and 5 (the vertical midpoint between the top and bottom of the eye).
- 2*||p1-p4||: This component involves the Euclidean distance between two points on the eye's horizontal axis, multiplied by 2. It's the distance between points 1 (the leftmost point on the eye) and 4 (the rightmost point on the eye), scaled by a factor of 2.

The EAR formula calculates the ratio between the average vertical distance of the eye's upper and lower eyelids and twice the average horizontal distance between the eye's inner and outer corners. This ratio provides a numerical measure of how open or closed the eye is. In applications such as facial landmark detection for gaze tracking or blink detection, the EAR is used as a threshold to classify eye states (e.g., open, closed, or partially closed).

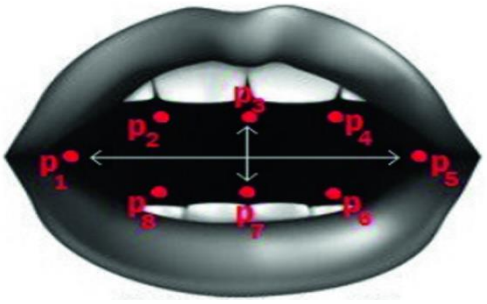

Figure 4.2.3 Landmarks around the mouth

Figure 4.1.3 depicts the 8 distinct landmarks around the mouth which are used to calculate the MAR value. It has 3 pairs of vertical landmarks and 1 pair of horizontal landmarks

Similarly, **MAR can be calculated using the following formula:**

$$MAR = \frac{||p_2 - p_8|| + ||p_3 - p_7|| + ||p_4 - p_6||}{2 * ||p_1 - p_5||}$$

The formula you provided calculates the Mouth Aspect Ratio (MAR), which is a measure used in facial landmark detection to assess the openness of the mouth. Similar to the Eye Aspect Ratio (EAR), the MAR is derived from the distances between specific landmarks around the mouth region. Let's break down the formula and explain each component:

- **||p2-p8||**: This represents the Euclidean distance between two key points on the mouth's vertical axis. Specifically, it's the distance between points 2 (the midpoint between the upper lip and the base of the nose) and 8 (the midpoint between the lower lip and the chin).
- **||p3-p7||**: Similarly, this represents the Euclidean distance between two other key points on the mouth's vertical axis. It's the distance between points 3 (the top of the upper lip) and 7 (the bottom of the lower lip).
- **||p4-p6||**: This component involves the Euclidean distance between two points on the mouth's horizontal axis. It's the distance between points 4 (the left corner of the mouth) and 6 (the right corner of the mouth).
- **2*||p1-p5||**: This component involves the Euclidean distance between two points on the mouth's horizontal axis, multiplied by 2. It's the distance between points 1 (the left corner of the mouth) and 5 (the midpoint of the upper lip), scaled by a factor of 2.

The MAR formula calculates the ratio between the average vertical distance of the mouth's upper and lower edges and twice the average horizontal distance between the mouth's corners. This ratio provides a numerical measure of how open or closed the mouth is. In applications such as facial landmark detection for mouth movements analysis or gesture recognition, the MAR is used as a threshold to classify mouth states (e.g., open or closed).

4) Comparison of Threshold values:

Now that the computed MAR and EAR levels have been compared to the threshold values, the appropriate mouth action is carried out :

- The cursor control system is engaged if the input is recognized by the system and MAR is larger than the mouth threshold value. Now, the user's facial movements will translate into mouse clicks on the screen
- A left click is made if left_EAR < right_EAR. Similarly, right click is made if left_EAR > right_EAR
- The scroll mode is activated whenever the user squints their eyes, or if the aspect ratio of their combined left and right eyes is less than the threshold number. To scroll the document down or up, the user can now incline their head downward or upward
- Head movements in pitch and yaw will cause the nose pointer to move, which will ultimately move the cursor in all four directions

5) Perform Mouse Action

By utilizing the PyAutoGUI package, the user's eye and facial gestures are converted into the appropriate mouse movements on the screen.

4.3 SAMPLE CODE

mouse.py code:

```
from imutils import face_utils
from utils import *
import numpy as np
import pyautogui as pyag
import imutils
import dlib
import cv2
# Thresholds and consecutive frame length for triggering the mouse action.
MOUTH_AR_THRESH = 0.5
MOUTH_AR_CONSECUTIVE_FRAMES = 10
EYE_AR_THRESH = 0.25
EYE_AR_CONSECUTIVE_FRAMES = 10
WINK_AR_DIFF_THRESH = 0.03
WINK_AR_CLOSE_THRESH = 0.1
WINK_CONSECUTIVE_FRAMES = 5
# Initialize the frame counters for each action as well as
# booleans used to indicate if action is performed or not
MOUTH_COUNTER = 0
EYE_COUNTER = 0
WINK_COUNTER = 0
INPUT_MODE = False
EYE_CLICK = False
LEFT_WINK = False
RIGHT_WINK = False
SCROLL_MODE = False
ANCHOR_POINT = (0, 0)
WHITE_COLOR = (255, 255, 255)
YELLOW_COLOR = (0, 255, 255)
RED_COLOR = (0, 0, 255)
GREEN_COLOR = (0, 255, 0)
BLUE_COLOR = (255, 0, 0)
```

```
BLACK_COLOR = (0, 0, 0)
# Initialize Dlib's face detector (HOG-based) and then create
# the facial landmark predictor
shape_predictor = "model/shape_predictor_68_face_landmarks.dat"
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(shape_predictor)
# Grab the indexes of the facial landmarks for the left and
# right eye, nose and mouth respectively
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
(nStart, nEnd) = face_utils.FACIAL_LANDMARKS_IDXS["nose"]
(mStart, mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]
# Video capture
vid = cv2.VideoCapture(0)
resolution_w = 1366
resolution_h = 768
cam_w = 640
cam_h = 480
unit_w = resolution_w / cam_w
unit_h = resolution_h / cam_h
while True:
    # Grab the frame from the threaded video file stream, resize
    # it, and convert it to grayscale
    # channels)
    _, frame = vid.read()
    frame = cv2.flip(frame, 1)
    frame = imutils.resize(frame, width=cam_w, height=cam_h)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Detect faces in the grayscale frame
    rects = detector(gray, 0)
    # Loop over the face detections
    if len(rects) > 0:
        rect = rects[0]
    else:
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        continue
```

```
# Determine the facial landmarks for the face region, then
# convert the facial landmark (x, y)-coordinates to a NumPy
# array
shape = predictor(gray, rect)
shape = face_utils.shape_to_np(shape)
# Extract the left and right eye coordinates, then use the
# coordinates to compute the eye aspect ratio for both eyes
mouth = shape[mStart:mEnd]
leftEye = shape[lStart:lEnd]
rightEye = shape[rStart:rEnd]
nose = shape[nStart:nEnd]
# Because I flipped the frame, left is right, right is left.
temp = leftEye
leftEye = rightEye
rightEye = temp
# Average the mouth aspect ratio together for both eyes
mar = mouth_aspect_ratio(mouth)
leftEAR = eye_aspect_ratio(leftEye)
rightEAR = eye_aspect_ratio(rightEye)
ear = (leftEAR + rightEAR) / 2.0
diff_ear = np.abs(leftEAR - rightEAR)
nose_point = (nose[3, 0], nose[3, 1])
# Compute the convex hull for the left and right eye, then
# visualize each of the eyes
mouthHull = cv2.convexHull(mouth)
leftEyeHull = cv2.convexHull(leftEye)
rightEyeHull = cv2.convexHull(rightEye)
cv2.drawContours(frame, [mouthHull], -1, YELLOW_COLOR, 1)
cv2.drawContours(frame, [leftEyeHull], -1, YELLOW_COLOR, 1)
cv2.drawContours(frame, [rightEyeHull], -1, YELLOW_COLOR, 1)
for (x, y) in np.concatenate((mouth, leftEye, rightEye), axis=0):
    cv2.circle(frame, (x, y), 2, GREEN_COLOR, -1)
# Check to see if the eye aspect ratio is below the blink
# threshold, and if so, increment the blink frame counter
if diff_ear > WINK_AR_DIFF_THRESH:
    if leftEAR < rightEAR:
        if leftEAR < EYE_AR_THRESH:
```

```
                WINK_COUNTER += 1
                if WINK_COUNTER >= WINK_CONSECUTIVE_FRAMES:
                    pyag.click(button='left')
                    WINK_COUNTER = 0
            elif leftEAR > rightEAR:
                if rightEAR < EYE_AR_THRESH:
                    WINK_COUNTER += 1
                    if WINK_COUNTER >= WINK_CONSECUTIVE_FRAMES:
                        pyag.click(button='right')
                        WINK_COUNTER = 0
            else:
                WINK_COUNTER = 0
        else:
            if ear <= EYE_AR_THRESH:
                EYE_COUNTER += 1
                if EYE_COUNTER >= EYE_AR_CONSECUTIVE_FRAMES:
                    SCROLL_MODE = not SCROLL_MODE
                    # INPUT_MODE = not INPUT_MODE
                    EYE_COUNTER = 0
                    # nose point to draw a bounding box around it
            else:
                EYE_COUNTER = 0
                WINK_COUNTER = 0
        if mar > MOUTH_AR_THRESH:
            MOUTH_COUNTER += 1
            if MOUTH_COUNTER >= MOUTH_AR_CONSECUTIVE_FRAMES:
                # if the alarm is not on, turn it on
                INPUT_MODE = not INPUT_MODE
                # SCROLL_MODE = not SCROLL_MODE
                MOUTH_COUNTER = 0
                ANCHOR_POINT = nose_point
        else:
            MOUTH_COUNTER = 0
        if INPUT_MODE:
            cv2.putText(frame, "READING INPUT!", (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
0.7, RED_COLOR, 2)
            x, y = ANCHOR_POINT
```

```
        nx, ny = nose_point
        w, h = 60, 35
        multiple = 1
        cv2.rectangle(frame, (x - w, y - h), (x + w, y + h), GREEN_COLOR, 2)
        cv2.line(frame, ANCHOR_POINT, nose_point, BLUE_COLOR, 2)
        dir = direction(nose_point, ANCHOR_POINT, w, h)
        cv2.putText(frame, dir.upper(), (10, 90), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
RED_COLOR,2)
        drag = 18
        if dir == 'right':
            pyag.moveRel(drag, 0)
        elif dir == 'left':
            pyag.moveRel(-drag, 0)
        elif dir == 'up':
            if SCROLL_MODE:
                pyag.scroll(40)
            else:
                pyag.moveRel(0, -drag)
        elif dir == 'down':
            if SCROLL_MODE:
                pyag.scroll(-40)
            else:
                pyag.moveRel(0, drag)
    if SCROLL_MODE:
        cv2.putText(frame, 'SCROLL MODE IS ON!', (10, 60),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, RED_COLOR, 2)
    # cv2.putText(frame, "MAR: {:.2f}".format(mar), (500, 30),
    #             cv2.FONT_HERSHEY_SIMPLEX, 0.7, YELLOW_COLOR, 2)
    # cv2.putText(frame, "Right EAR: {:.2f}".format(rightEAR), (460, 80),
    #             cv2.FONT_HERSHEY_SIMPLEX, 0.7, YELLOW_COLOR, 2)
    # cv2.putText(frame, "Left EAR: {:.2f}".format(leftEAR), (460, 130),
    #             cv2.FONT_HERSHEY_SIMPLEX, 0.7, YELLOW_COLOR, 2)
    # cv2.putText(frame, "Diff EAR: {:.2f}".format(np.abs(leftEAR - rightEAR)), (460, 80),
    #             cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    # Show the frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
```

```
    # If the Esc key was pressed, break from the loop
    if key == 27:
        break
# Do a bit of cleanup
cv2.destroyAllWindows()
vid.release()
```

utils.py code:

```
import numpy as np
# Returns EAR given eye landmarks
def eye_aspect_ratio(eye):
    # Compute the euclidean distances between the two sets of
    # vertical eye landmarks (x, y)-coordinates
    A = np.linalg.norm(eye[1] - eye[5])
    B = np.linalg.norm(eye[2] - eye[4])
    # Compute the euclidean distance between the horizontal
    # eye landmark (x, y)-coordinates
    C = np.linalg.norm(eye[0] - eye[3])
    # Compute the eye aspect ratio
    ear = (A + B) / (2.0 * C)
    # Return the eye aspect ratio
    return ear

# Returns MAR given eye landmarks
def mouth_aspect_ratio(mouth):
    # Compute the euclidean distances between the three sets
    # of vertical mouth landmarks (x, y)-coordinates
    A = np.linalg.norm(mouth[13] - mouth[19])
    B = np.linalg.norm(mouth[14] - mouth[18])
    C = np.linalg.norm(mouth[15] - mouth[17])
   # Compute the euclidean distance between the horizontal
    # mouth landmarks (x, y)-coordinates
    D = np.linalg.norm(mouth[12] - mouth[16])
    # Compute the mouth aspect ratio
    mar = (A + B + C) / (2 * D)
   # Return the mouth aspect ratio
    return mar
```

```
# Return direction given the nose and anchor points.
def direction(nose_point, anchor_point, w, h, multiple=1):
    nx, ny = nose_point
    x, y = anchor_point
    if nx > x + multiple * w:
        return 'right'
    elif nx < x - multiple * w:
        return 'left'
    if ny > y + multiple * h:
        return 'down'
    elif ny < y - multiple * h:
```

CHAPTER 5

SYSTEM TESTING AND RESULTS

5.SYSTEM TESTING AND RESULTS

5.1 INTRODUCTION

Software testing is a vital component of software quality assurance, serving as the final assessment of specification, design, and code. Testing is the process of executing a program with the intent of finding an error, running a software in order to detect errors. The creation of tests for software and other engineering goods can be as difficult as the product's original design.

5.1.1 System Testing

System testing is a crucial phase in the software development lifecycle aimed at verifying that the implemented system meets its specified requirements and functions as intended. It involves testing the integrated system as a whole to assess its performance, reliability, and compliance with user expectations. System testing encompasses various testing activities, including functional testing, non-functional testing, performance testing, and usability testing. By conducting comprehensive system testing, software engineers can identify and rectify defects, validate system behavior, and ensure the delivery of a high-quality software product to end-users.

Types of testing:

There are two primary sorts of testing methodologies.

Black-box testing:

Black-box testing is a software testing method that evaluates an application's functioning without examining its internal structures. This test approach can be implemented almost anywhere in the software testing process. Test cases are designed based on the specified requirements and external behavior of the software. Testers interact with the software through its user interface or APIs, providing inputs and observing outputs to verify that the expected results are produced.

White-Box testing:

White-Box examining involves examining software's internal structure, architecture, and coding to ensure input-output flow and improve design, usability, and security. Test cases are derived from an understanding of the software's internal logic, control flow, data structures, and algorithms. Testers analyze the code structure and identify critical paths, decision points, and boundary conditions to design test cases that exercise specific code segments.

EQUIPMENT TESTING:

Equipment testing for the project "Operating Computer Cursor Using Eye and Face Movements" involves evaluating the hardware components and devices necessary for the system's operation.

Step 1: Checking installed libraries

Figure 5.1.1.1 Checking installed libraries

Figure 5.1.1.1 Depicts checking the required libraries (Numpy,imutils,cv2,cmake,dlib, pyautogui, etc) are installed or not

Step 2: Camera Testing

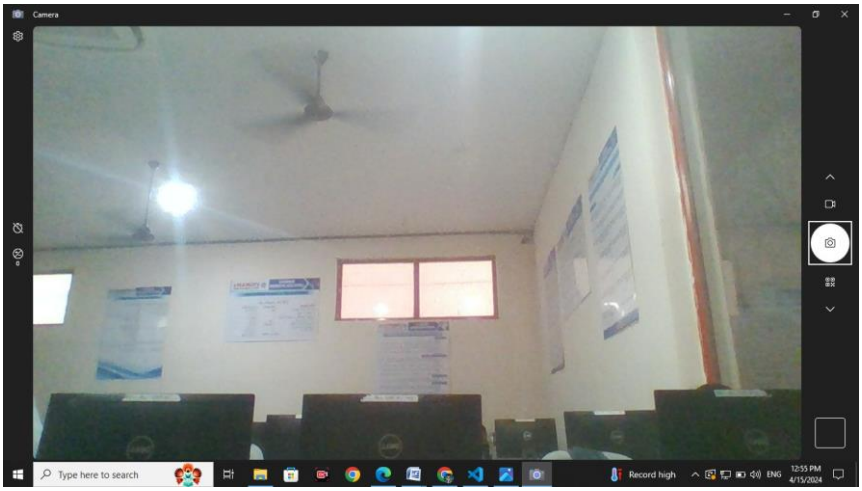

Figure 5.1.1.2 camera testing

Figure 5.1.1.2 depicts testing the camera. Make sure the camera device is in proper working conditions

5.1.2 Test Cases

S. No	Test Case Input	Expected Output	Actual Output	Test Result
1.	Cursor control system operated by user wearing glasses.	Reading input and performing mouse action on screen	Reading input and performing mouse action on screen	Pass
2.	Cursor control system used in a simple online game	Reading input and performing mouse action on screen	Reading input and performing mouse action on screen	Pass
3.	Cursor control system operated in dim light	Reading input and performing mouse action on screen	Reading input and performing mouse action on screen	Pass
4.	Cursor control system implemented to scroll pages.	Reading input and performing scroll action	Reading input and performing scroll action	Pass
5.	Cursor control system to perform left and right clicks	Reading input and performing click action	Reading input and performing click Action	Pass
6.	Moving the cursor in all directions	Reading input and moving the cursor	Reading input and moving the cursor	Pass
7.	Activating or deactivating cursor control system through opening the mouth	Reading input and activating or deactivating cursor	Reading input and activating or deactivating cursor	Pass

Table 5.1.2Testcases

Table 5.1.2 illustrates the various testcases performed on the proposed project. The system has performed all the mouse actions, i.e., clicking scrolling and moving in all directions. Activation and deactivation of the system through opening the mouth is performed successfully. The system also proved to be efficient when tested against dim light condition, in an online game, and for a user wearing glasses.

5.2 TESTING STRATEGIES

White-box and black-box testing are two fundamental strategies used in software testing to ensure the quality and reliability of a software application. Let's provide an introduction to each strategy:

1. **White-box Testing:**

Introduction: White-box testing, also known as clear-box or glass-box testing, involves examining the internal structure, architecture, and code implementation of a software application. Testers have access to the source code and use their knowledge of the internal workings of the software to design test cases and assess its behavior.

Approach: Test cases are derived from an understanding of the software's internal logic, control flow, data structures, and algorithms. Testers analyze the code structure and identify critical paths, decision points, and boundary conditions to design test cases that exercise specific code segments.

Purpose: The primary goal of white-box testing is to ensure code correctness, identify errors or bugs in the implementation, and assess the software's efficiency, security, and maintainability. It aims to validate the internal behavior of the software and improve its design, usability, and performance.

2. **Black-box Testing:**

Introduction: Black-box testing is a software testing method that evaluates the functionality of an application without considering its internal code structure, design, or implementation details. Testers treat the software as a "black box," focusing solely on the inputs and outputs of the system to verify its behavior.

Approach: Test cases are designed based on the specified requirements and external behavior of the software. Testers interact with the software through its user interface or APIs, providing inputs and observing outputs to validate that the expected results are produced.

Purpose: The main objective of black-box testing is to assess the functional correctness and compliance of the software with the specified requirements. It focuses on validating the software's external behavior from an end-user perspective, without requiring knowledge of internal code structures or implementation details.

Overall, white-box testing involves examining the internal structure and code implementation

of a software application to ensure correctness, efficiency, and security, while black-box testing evaluates the functionality of the software based on its external behavior without considering internal details. Both testing strategies are essential components of a comprehensive software testing approach and are used to validate different aspects of a software system's quality and reliability.

5.2.1 White Box Testing

1.Definition: White-box testing, also known as clear-box or glass-box testing, examines the internal structure, architecture, and code implementation of the software system. Testers have visibility into the internal workings of the software and design test cases based on this knowledge.

2.Approach: Test cases are derived from an understanding of the software's internal logic, control flow, data structures, and algorithms. Testers analyze the code structure and identify critical paths, decision points, and boundary conditions to design test cases that exercise specific code segments.

3.Benefits:

1. White-box testing provides insights into the software's internal behavior, allowing testers to identify and address issues related to code quality, efficiency, and security vulnerabilities
2. It enables thorough coverage of code paths, helping to identify errors, bugs, and potential failure points within the software.
3. White-box testing facilitates optimization of code structure and performance, as testers can pinpoint areas for improvement and refinement based on their understanding of the internal architecture.

4.Limitations:

1. White-box testing requires access to the source code and a deep understanding of programming languages and software development principles, which may limit its applicability in certain contexts.
2. Testers may inadvertently focus on testing known code paths or implementations, overlooking edge cases or scenarios that are not explicitly considered in the test plan.
3. White-box testing can be time-consuming and resource-intensive, especially for large and complex software systems, as it involves detailed analysis of code structures and extensive test case development.

5.Process:

1. Review code structure: Analyze the source code of the system, focusing on the implementation of algorithms for facial landmark detection, aspect ratio calculations, and cursor control logic.
2. Identify critical paths: Identify key code paths and decision points within the system that are critical for accurate cursor control and gesture interpretation.
3. Design test cases: Develop test cases targeting specific code segments, including boundary conditions, error handling, and edge cases to ensure comprehensive coverage.
4. Execute test cases: Execute the designed test cases, examining the behavior of the system at the code level to validate correctness, efficiency, and adherence to coding standards.
5. Evaluate code quality: Assess the quality of the code structure, identifying any potential issues such as code duplication, inefficient algorithms, or security vulnerabilities.

6.Results:

1. White-box testing confirms that the code implementation accurately reflects the intended functionality of the system, with correct algorithms for facial landmark detection and aspect ratio calculations.
2. Test cases validate that the cursor control logic effectively translates detected gestures into appropriate cursor movements, clicks, and scrolling actions.
3. The code review process identifies areas for optimization and refinement, such as improving algorithm efficiency or enhancing error handling mechanisms.
4. Security vulnerabilities or potential risks related to input validation or data sanitization are identified and addressed to ensure the system's robustness and integrity.

5.2.2 Black Box testing

1.Definition: Black-box testing is a software testing method that evaluates the functionality of an application without considering its internal code structure, design, or implementation details. It focuses solely on the inputs and outputs of the software system, treating it as a "black box" where the internal workings are not visible.

2.Approach: Test cases are designed based on the specified requirements and external behavior of the software. Testers interact with the software through its user interface or APIs, providing inputs and observing outputs to verify that the expected results are produced.

3.Benefits:

1.Black-box testing is independent of the internal implementation, making it suitable for testing at various stages of development without requiring knowledge of programming languages or implementation details.

2.It facilitates a broader coverage of test scenarios, allowing testers to explore different input combinations, boundary conditions, and error handling mechanisms.

3.Black-box testing promotes a user-centric approach, focusing on validating the software's functionality from an end-user perspective, which helps ensure that the software meets user requirements and expectations.

4.Limitations:

1.Test cases may overlook certain code paths or scenarios that are not explicitly defined in the requirements, leading to incomplete test coverage.

2.Since black-box testing relies solely on inputs and outputs, it may not uncover errors related to internal logic, algorithms, or performance issues.

3.Test cases may be redundant or repetitive if not carefully designed, potentially leading to inefficiencies in the testing process.

5.Process:

1. Identify functional requirements: Define the expected behavior of the system, including cursor movement based on eye and face gestures, left and right clicks, scrolling actions, etc.
2. Design test cases: Develop test cases based on the specified requirements, covering various scenarios such as different types of eye movements (blinking, squinting), facial gestures (smiling, frowning), and head movements (left, right, up, down).
3. Execute test cases: Interact with the system using the designed test cases, providing inputs (simulated eye and face movements) and observing outputs (cursor movements, click actions) to verify that the system behaves as expected.
4. Record and analyze results: Document the outcomes of each test case, noting any discrepancies between expected and actual behavior. Identify defects or inconsistencies in the system's functionality and report them for resolution.

6. Results:

1. The black-box testing process verifies that the system correctly interprets and responds to user inputs (eye and face movements) to control the computer cursor.
2. Test cases confirm that the cursor moves in the intended direction based on detected head movements, blinks trigger left or right clicks, and squinting actions activate scrolling functionality.
3. The system demonstrates robustness and reliability in handling various user gestures, ensuring that cursor control remains accurate and responsive across different usage scenarios.
4. Any deviations from expected behavior, such as incorrect cursor movements or failure to detect certain gestures, are identified and documented as defects for further investigation and resolution.

5.2.3 UNIT TESTING

1. Definition: Unit testing is a software testing technique where individual units or components of a software application are tested in isolation to ensure that each unit functions correctly as per its design specifications. These units are typically small, self-contained modules or functions within the codebase.

2. Approach:

1. Isolation: Each unit is tested independently of other units and external dependencies. This is achieved by using mock objects or stubs to simulate interactions with external components.
2. Automated Testing: Unit tests are automated and integrated into the development process. Developers write test cases to verify the behavior of individual units, and these tests are executed automatically whenever code changes are made.
3. Test Coverage: Test cases are designed to cover various code paths, boundary conditions, and error scenarios within the unit. This ensures comprehensive coverage of the unit's functionality.
4. Feedback Loop: Unit tests provide immediate feedback to developers about the correctness of their code. Failures in unit tests indicate potential defects in the code, which can be identified and addressed early in the development cycle.

3. Benefits:

1. Early Detection of Defects: Unit testing helps identify defects in the code early in the development process, reducing the cost and effort of fixing them later.
2. Improved Code Quality: By testing individual units in isolation, developers can ensure that each unit behaves as expected and meets its design specifications.
3. Regression Testing: Unit tests serve as a safety net to detect regressions introduced by code changes. They ensure that existing functionality remains intact as new features are added or bugs are fixed.
4. Facilitates Refactoring: Unit tests provide confidence to refactor code without introducing unintended side effects. Developers can make changes to the codebase with the assurance that unit tests will catch any regressions.

4. Limitations:

1. Incomplete Coverage: Unit tests may not cover all possible code paths or scenarios within a unit. Developers must ensure that test cases are comprehensive and effectively capture the unit's behavior.
2. Dependency Management: Testing units in isolation requires mocking or stubbing external dependencies, which can be complex and time-consuming, especially for units with many dependencies.
3. False Sense of Security: Passing unit tests do not guarantee the absence of defects in the

software. Integration issues or interactions between units may only surface during higher levels of testing.

4. Maintenance Overhead: Unit tests must be maintained alongside the codebase, which can add overhead in terms of time and effort. As the codebase evolves, unit tests may need to be updated or refactored to reflect changes in the code.

5. Process:

1. Identify Units: Break down the software into smaller, testable units such as functions or modules. In this project, units may include functions responsible for facial landmark detection, aspect ratio calculations, cursor control logic, etc.
2. Write Test Cases: Design test cases to verify the behavior of each unit. For example: Test facial landmark detection function with various input images containing faces. Test aspect ratio calculation function with different sets of facial landmarks. Test cursor control logic function with simulated eye and face movements to validate cursor movements and clicks.
3. Setup and Mock Dependencies: Create mock objects or stubs to simulate interactions with external dependencies such as the webcam or computer vision libraries (e.g., OpenCV, dlib). Ensure that the units under test are isolated from external dependencies to focus on testing their internal behavior.
4. Execute Tests: Run the unit tests automatically as part of the continuous integration (CI) process or manually during development. Monitor test execution and analyze results to identify any failures or deviations from expected behavior.
5. Debug and Refactor: Debug failures in unit tests to identify and fix defects in the code. Refactor unit tests and code as necessary to improve test coverage, readability, and maintainability.
6. Repeat and Iterate: Continuously add new test cases and update existing ones to cover additional code paths, edge cases, and error scenarios. Iterate on the unit testing process to ensure thorough validation of all units within the software.

6. Results:

1. Facial Landmark Detection Unit:

Test cases verify that the facial landmark detection function correctly identifies key points on the face (eyes, nose, mouth) across a variety of facial expressions, poses, and lighting conditions. Results show high accuracy and reliability in detecting facial landmarks, with minimal false positives or false negatives.

2. Aspect Ratio Calculation Unit:

Test cases validate the accuracy of aspect ratio calculations based on detected facial landmarks. Results confirm that the aspect ratio calculation function produces correct values for eye aspect ratio (EAR) and mouth aspect ratio (MAR), enabling accurate interpretation of user gestures.

3. Cursor Control Logic Unit:

Test cases ensure that the cursor control logic accurately translates detected eye and face movements into cursor actions (movement, clicks, scrolling). Results demonstrate smooth and responsive cursor control, with precise alignment between user gestures and cursor behavior.

5.2.4 INTEGRATION TESTING

Integration testing is a software testing technique that focuses on testing the interactions between different components or modules of a software system to ensure that they work together as expected when integrated. Here's a breakdown of integration testing, including its definition, approach, benefits, and limitations:

1. Definition:

Integration testing is the process of combining individual software modules or components and testing them as a group to verify that they interact correctly and produce the expected outcomes. It aims to uncover defects or inconsistencies in the interactions between components and ensure the seamless integration of various parts of the system.

2. Approach:

Top-Down Integration Testing: In this approach, testing begins with the highest-level modules or components and gradually moves down the hierarchy, integrating and testing lower-level modules until the entire system is tested.

Bottom-Up Integration Testing: This approach starts with testing the lowest-level modules first and then progressively integrates and tests higher-level modules until the entire system is tested.

Big Bang Integration Testing: In this approach, all modules are integrated simultaneously, and the system is tested as a whole. It's suitable for smaller systems or when there are dependencies between multiple modules.

Incremental Integration Testing: This approach involves integrating and testing individual modules or subsystems incrementally, starting with the most critical or essential components and gradually adding more components until the entire system is tested.

3. Benefits:

1. Early Detection of Integration Issues: Integration testing helps identify integration-related defects early in the development process, making it easier and less costly to fix them.
2. Validation of Interfaces: It verifies that the interfaces between different components are functioning correctly and that data is passed between modules as expected.
3. Improved System Reliability: By testing the interactions between components, integration testing improves the overall reliability and stability of the system.
4. Enhanced Confidence in System Behavior: Successful integration testing provides confidence that the system behaves as intended when all components are integrated,

reducing the risk of unexpected behavior in production.

5. Faster Time to Market: Identifying and resolving integration issues early in the development cycle helps streamline the development process and accelerate time to market for the software product.

4. Limitations:

1. Complexity: Integration testing can be complex, especially for systems with many interconnected components or dependencies, making it challenging to design comprehensive test scenarios.
2. Dependency Management: Testing may be hindered by dependencies on external systems, databases, or third-party components, which can complicate the setup and execution of integration tests.
3. Resource Intensive: Integration testing requires substantial resources, including time, hardware, and testing environments, particularly for large-scale systems with numerous components.
4. Incomplete Coverage: It may not be feasible to test all possible interactions between components, leading to gaps in test coverage and the possibility of undiscovered defects.
5. Debugging Complexity: Identifying the root cause of integration failures can be challenging due to the complex interactions between components, requiring thorough investigation and debugging efforts.

5. Process:

1.Identify Components: Identify the key components involved in the project, including:

- Eye-tracking module
- Face-tracking module
- Cursor control module
- User interface module

1. Define Test Scenarios: Design test scenarios to validate the interactions between these components. Test scenarios could include:

- Moving the cursor based on eye movements
- Moving the cursor based on head movements
- Clicking or selecting objects using eye blinks or facial gestures
- Coordinating simultaneous eye and face movements for precise control

2. Set Up Testing Environment: Set up a testing environment with the necessary hardware (e.g., cameras, sensors) and software (e.g., tracking algorithms, user interface) components.
3. Perform Integration Tests:

- Eye-tracking Integration: Verify that the eye-tracking module accurately detects eye movements and communicates this data to the cursor control module.

- Face-tracking Integration: Ensure that the face-tracking module correctly interprets facial expressions and gestures, and communicates relevant information to the cursor control module.
- Cursor Control Integration: Test the integration between the cursor control module and the input from both the eye-tracking and face-tracking modules, ensuring that cursor movements correspond accurately to user movements.

4. Execute Test Scenarios: Run the predefined test scenarios in the testing environment and observe the behavior of the integrated system. Document any deviations from expected outcomes or unexpected behavior.
5. Record Test Results: Record the results of each test scenario, including any issues encountered, such as inaccuracies in cursor movement, delays in responsiveness, or failures to detect user input.
6. Debug and Fix Issues: Investigate any issues identified during testing, determine their root causes, and implement appropriate fixes. Retest the affected components to ensure that the issues have been resolved.
7. Regression Testing: After fixing issues, perform regression testing to ensure that the integration changes have not introduced new defects or affected previously working functionality.

6. Results:

1. Successful Integration: The components responsible for eye tracking, face tracking, and cursor control seamlessly integrate and function as expected, allowing the user to control the computer cursor effectively using eye and face movements.
2. Accuracy and Responsiveness: The integrated system accurately tracks and responds to the user's eye and face movements, with minimal latency or delay in cursor movement.
3. User Interface Validation: The user interface effectively communicates feedback to the user, such as highlighting selected objects or providing visual cues for cursor movement, enhancing the overall user experience.
4. Compatibility and Stability: The integrated system demonstrates compatibility with different hardware configurations and operating environments, and it remains stable under varying conditions, such as changes in lighting or user position.
5. Identified Issues: Any issues identified during integration testing, such as inaccuracies in tracking, delays in responsiveness, or unexpected behavior, are documented along with their severity and potential impact on system functionality.
6. Resolved Defects: Defects identified during integration testing are investigated, debugged, and resolved. Verification tests are performed to ensure that the fixes have effectively addressed the issues without introducing new regressions.

7. Test Coverage: The integration testing process achieves comprehensive test coverage, validating various aspects of the system's functionality and interactions between components.
8. Validation of Requirements: Integration testing validates that the integrated system meets the specified requirements and user expectations for cursor control using eye and face movements.

By following a structured integration testing process and carefully documenting the results, the project team can ensure the reliability, functionality, and usability of the system for operating a computer cursor using eye and face movements.

5.3 OUTPUT SCREENS

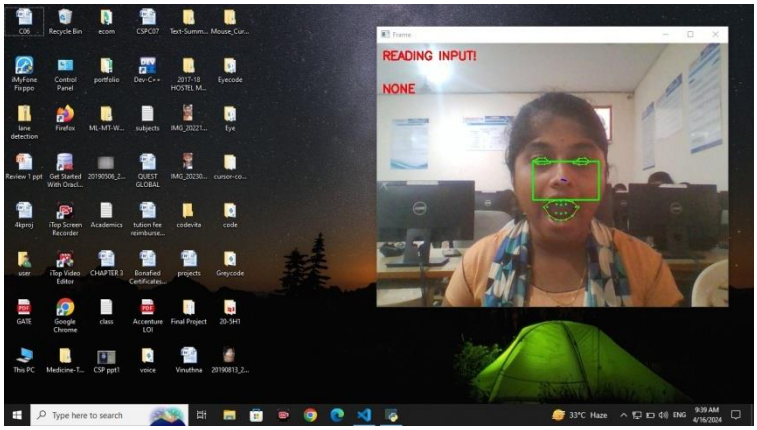

Figure 5.3.1: Reading Input

Figure 5.3.1 depicts activating the cursor. Open the mouth widely, then reading input is displayed in theframe. If mouth is opened for the second time, the cursor will be deactivated.

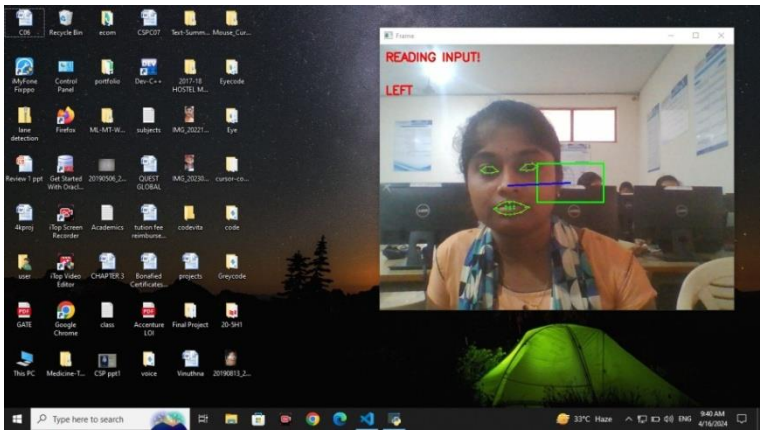

Figure 5.3.2: Moving cursor to the left

Figure 5.3.2 depicts moving the cursor towards left. If head is slightly moved left, the distance between nose pointerand anchor pointer increases and the cursor starts moving left.

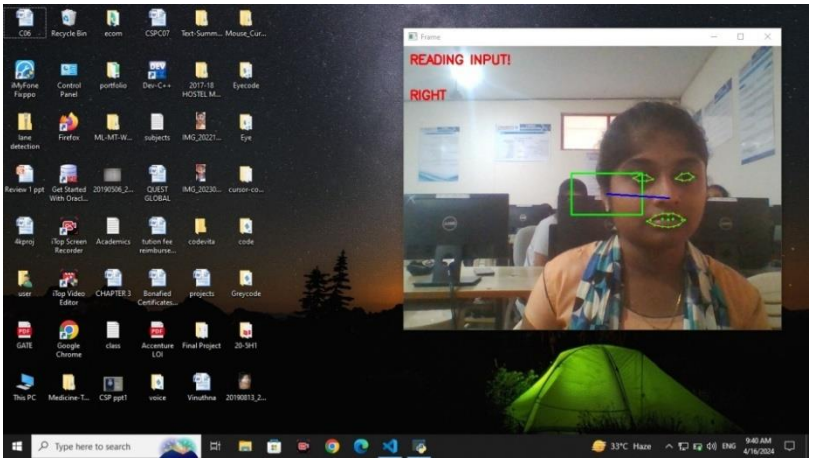

Figure 5.3.3: Moving cursor to the right

Figure 5.3.3 depicts moving the cursor towards right. If head is slightly moved right, the distance between nose pointerand anchor pointer increases and the cursor starts moving right.

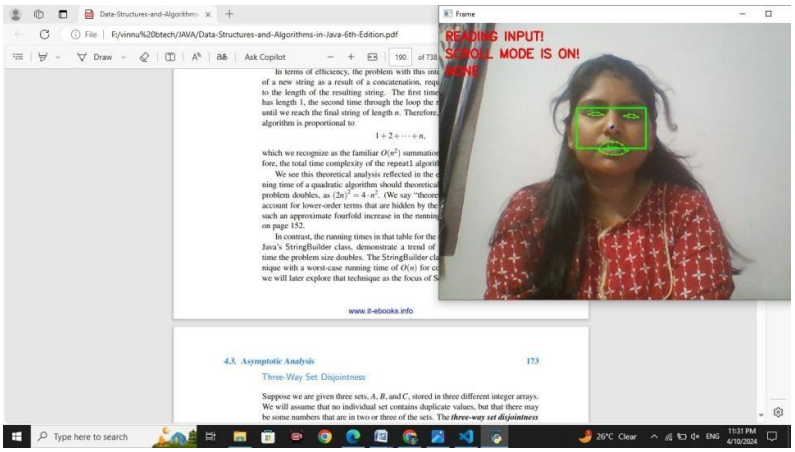

Figure 5.3.4: Enabling scrolling effect

Figure 5.3.4 depicts the enabling of scrolling effect. If both the eyes are squinted, it enables scrolling mode allowing us to scroll through documents or web pages seamlessly.

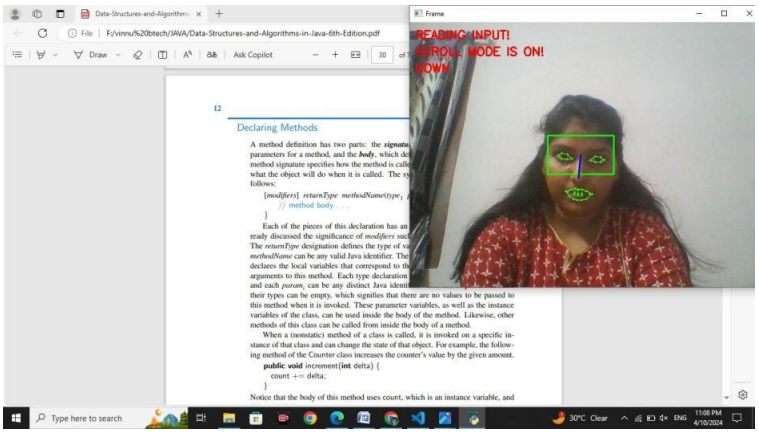

Figure 5.3.5: Scrolling Down

Figure 5.3.5 depicts scroll action. To start scrolling, start with squinting your eyes (to look with the eyes partly closedeyes) and scroll mode is turned on. Now scroll down by moving

your head downwards.

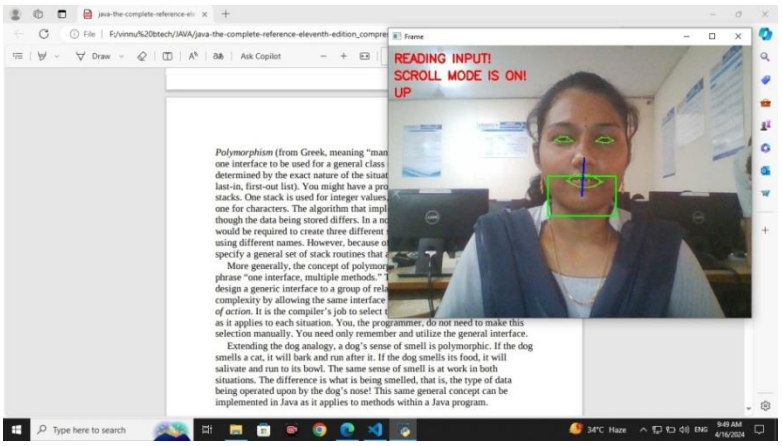

Figure 5.3.6: Scrolling up

Figure 5.3.6 depicts scroll up action. While scroll mode is on, move your head upwards to scroll up and to deactivatescrolling squint your eyes again.

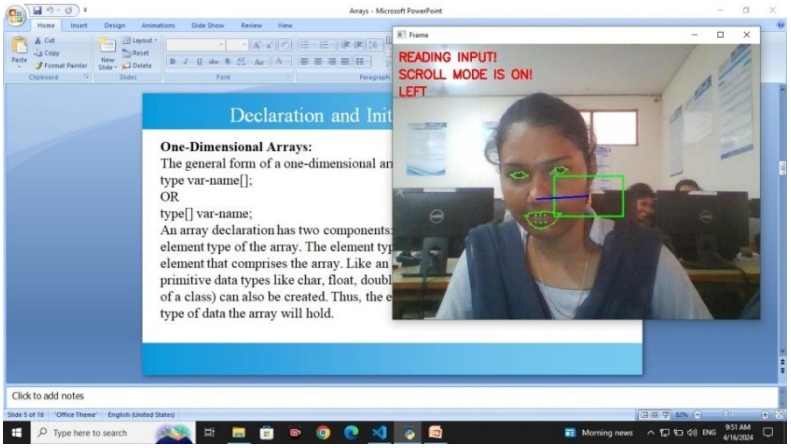

Figure 5.3.7: Scrolling left

Figure 5.3.7 depicts scrolling left. When scroll mode is on, move your head towards left to scroll the document left.We can similarly move head towards right for scrolling right.

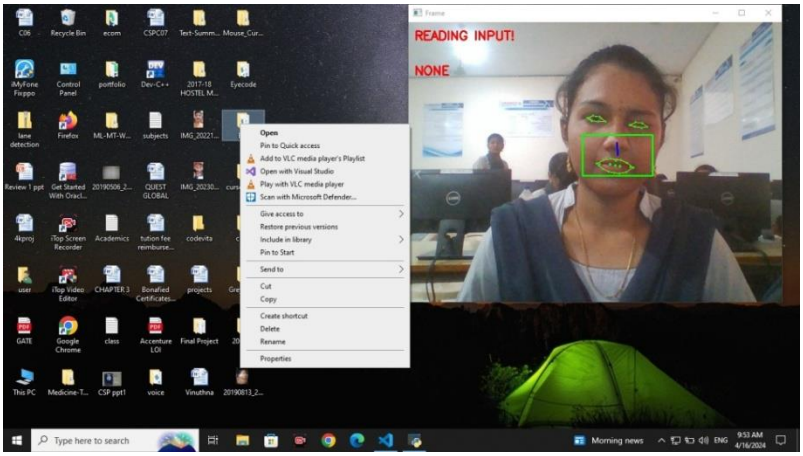

Figure 5.3.8: Right Click

To perform click action slightly tilt your head left and wink right eye for right click and left eye for left click as shown in Figure 5.3.8

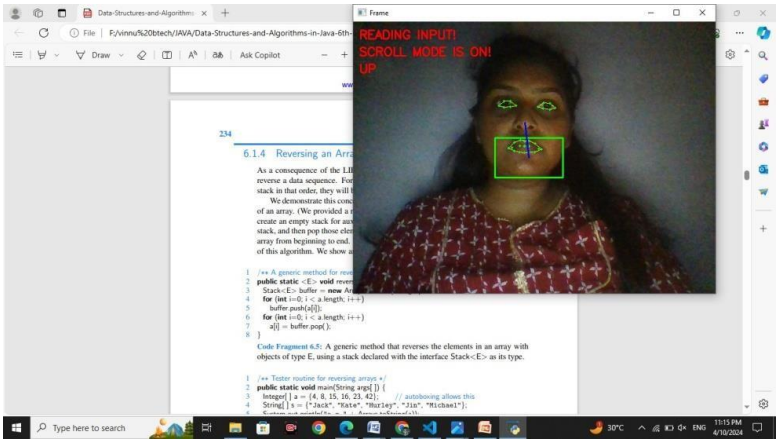

Figure 5.3.9: Operating cursor control system in dim light

Figure 5.3.9 depicts operating the proposed system in a dim light condition. The system is able to successfullyoperate and perform the required actions.

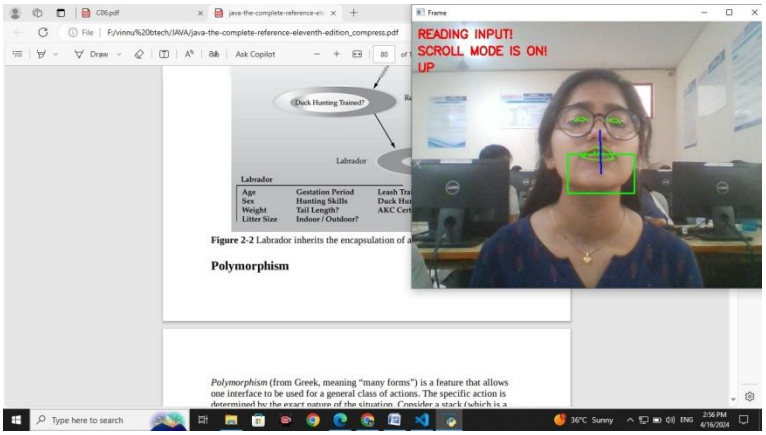

Figure 5.3.10: Operating cursor control system by a user wearing glasses

Figure 5.3.10 demonstrates operating the proposed system when the user is wearing glasses. The system is able to precisely locate the landmarks and perform the required action.

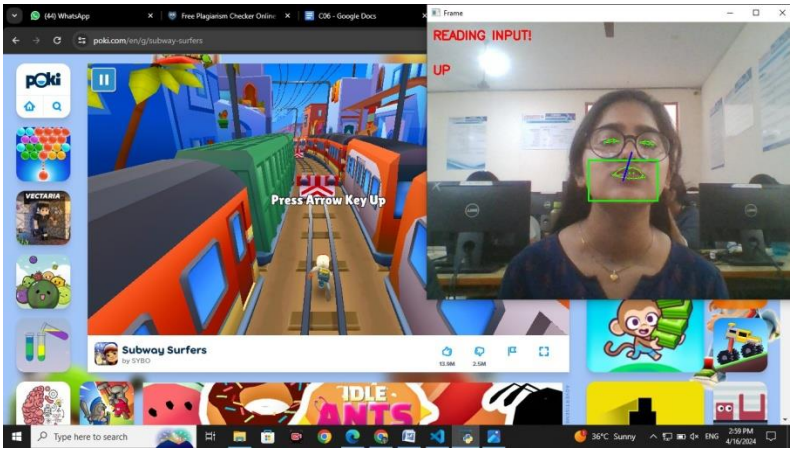

Figure 5.3.11: Playing a simple online game with the help of eye cursor system.

Figure 5.3.11 depicts operating the proposed system in a simple online video game. The system was able to perform all the required actions success

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.CONCLUSION AND FUTURE SCOPE

CONCLUSION:

In conclusion, the development of a system for operating a computer cursor using eye and face movements represents a significant advancement in human-computer interaction technology. By harnessing natural eye and facial gestures, this system offers a novel and intuitive method for users to interact with their computers, potentially enhancing accessibility, productivity, and user experience. Throughout the project, careful consideration must be given to factors such as accuracy, responsiveness, user comfort, and privacy to ensure the system's effectiveness and acceptance. Integration testing plays a crucial role in validating the interactions between various components, verifying the reliability and functionality of the system, and identifying any issues that need to be addressed. The project outcome is a sophisticated software application or system that seamlessly integrates eye and face tracking algorithms with cursor control mechanisms, providing users with a seamless and intuitive means of navigating their computer interfaces. Through thorough testing, documentation, and support, the system aims to deliver a user-friendly and ethically sound solution that empowers users to interact with their computers in a natural and efficient manner. In summary, the development of a system for operating a computer cursor using eye and face movements represents a promising frontier in human-computer interaction, with the potential to revolutionize the way we interact with technology and enhance accessibility for users of all abilities.

FUTURE SCOPE:

The project "Operating Computer Cursor Using Eye and Face Movements" holds significant potential for future advancements and expansions. Here's a detailed future scope for the project:

1. Gesture Recognition and Interaction:

Expand the range of supported gestures and interactions beyond basic cursor control. Introduce advanced gesture recognition capabilities to enable users to perform actions such as clicking, dragging, zooming, and rotating using eye and face movements.

Implement customizable gesture mapping functionalities, allowing users to define and configure their preferred gestures for specific actions or commands.

2. Adaptive User Interfaces:

Develop adaptive user interfaces that dynamically adjust based on user preferences, behavior patterns, and accessibility needs. Personalize the interface layout, interaction modes, and

cursor control settings to accommodate individual user preferences and requirements.

Utilize machine learning algorithms to continuously learn and adapt to users' interaction patterns and preferences, optimizing the user experience over time.

3. Multi-Modal Interaction:

Explore multi-modal interaction techniques that combine eye and face movements with other input modalities, such as voice commands, gestures, and touch inputs. Enable users to interact with the computer system using a combination of input modalities for enhanced flexibility and efficiency.

Implement fusion algorithms to intelligently combine inputs from different modalities, leveraging the strengths of each modality to provide a more natural and intuitive interaction experience.

4. Real-Time Feedback and Visualization:

Incorporate real-time feedback mechanisms and visualization tools to provide users with insights into their eye and face movements and their corresponding effects on the computer interface.

Display visual indicators or overlays to highlight detected facial landmarks, gaze direction, and cursor movements, helping users understand and control their interactions more effectively.

By pursuing these avenues for future development, the project can evolve into a sophisticated and versatile solution for hands-free computer interaction, empowering users with physical disabilities and providing a novel and intuitive computing experience for all users.

CHAPTER 7

PROJECT OUTCOMES-PO/PSO MAPPING

7.PROJECT OUTCOMES - PO/PSO MAPPING

Batch No: C06

Domain: Computer Vision

Project Type: Application

Project Tittle: Operating computer cursor using eye and face movements

CO No.	Course Outcomes	Relevance to POS/PSOS
CO1	Analyze principles and technologies behind eye tracking and facial recognition of existing models.(K4)	PO1 - PO12 PSO1,PSO2
CO2	Modify the existing algorithm , Architecture and formulate a new architecture in real-time eye tracking and facial recognition.(K3)	
CO3	Develop a prototype eye-controlled cursor system using open-source libraries or SDKs(K3)	
CO4	Design and conduct experiments to evaluate the accuracy and efficiency of the developed eye and face tracking system.(K5)	
CO5	Perceive applications of eye and face tracking in human -computer interaction, accessibility, and gaming.(K5)	

Project Outcomes - POs / PSOs Mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
C409.1	3	3	3	3	-	3	-	3	3	3	3	3	3	2
C409.2	3	3	3	3	3	3	-	3	3	3	2	3	3	2
C409.3	3	3	3	3	3	3	3	3	3	3	-	3	3	2
C409.4	3	3	-	-	3	3	2	3	3	3	3	3	3	2
C409.5	3	3	-	-	-	-	-	-	-	3	2	3	3	2
Avg	**3**	**3**	**3**	**3**	**3**	**3**	**2.5**	**3**	**3**	**3**	**2.5**	**3**	**3**	**2**

REFERENCES

REFERENCES

[1] V. Khare, S. G. Krishna and S. K. Sanisetty, "Cursor Control Using Eye Ball Movement," 2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), 2019, pp. 232- 235, doi: 10.1109/ICONSTEM.2019.8918780.

[2]Maruthapillai, Vasanthan & M, Murugappan & Nagarajan, R. & Ilias, Bukhari & Letchumikanth, J.. (2012). Facial expression based computer cursor control system for assisting physically disabled person. Proceeding - COMNETSAT 2012: 2012 IEEE International Conference on Communication, Networks and Satellite. 172-176. 10.1109/ComNetSat.2012.6380800.

[3] Mehta, Sukrit and Dadhich, Sharad and Gumber, Sahil and Jadhav Bhatt, Arpita, Real-Time Driver Drowsiness Detection System Using Eye Aspect Ratio and Eye Closure Ratio (March 20, 2019). Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur - India, February 26-28, 2019

[4] S. Mohanty, S. V. Hegde, S. Prasad and J. Manikandan, "Design of Real-time Drowsiness Detection System using Dlib," 2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), 2019, pp. 1-4, doi: 10.1109/WIECON-ECE48653.2019.9019910.

[5] A. Rosebrock, Facial Landmarks with dlib, OpenCV, and Python, April 2017, Available: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

[6] How to detect eye blinking in videos using dlib and OpenCV in Python, June 2020, Available: http://datahacker.rs/011-how-to-detect-eye-blinking-in-videos-using-dlib-and-opencv-in-python/

[7] Relangi S.P.K., Nilesh M., Kumar K.P., Naveen A. (2020) Full Length Driver Drowsiness Detection Model—Utilizing Driver Specific Judging Parameters. In: Reddy A., Marla D., Simic M., Favorskaya M., Satapathy S. (eds) Intelligent Manufacturing and Energy Sustainability. Smart Innovation, Systems and Technologies, vol 169. Springer, Singapore. https://doi.org/10.1007/978-981-15-1616-0_

[8] V. Khare, S. G. Krishna and S. K. Sanisetty, "Cursor Control Using Eye Ball Movement," 2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), 2019, pp.232-235, doi: 10.1109/ICONSTEM.2019.8918780.

[9] Chandra, B., M. Rohit, and R. Sriram Vignesh, “Eyeball Movement Cursor Control Using OpenCV”. ECS Transactions 107.1 (2022):10005.

[10] Robert J. K. Jacob, “The use of eye movements in human-computer interaction techniques: what you look at is what you get”. ACM Trans. Inf. Syst. 9, 2, pp. 152–169, 1991. doi:10.1145/123078.128728

[11] R. Fatima, A. Usmani and Z. Zaheer, “Eye movement based human computer interaction. 3rd International Conference on Recent Advances in Information Technology (RAIT), pp.489-494, 2016, doi: 10.1109/RAIT.2016.7507950.

[12] Fahim, Shahriar Rahman, et al., “A visual analytic in deep learning approach to eye movement for human-machine interaction based on inertia measurement.” IEEE Access 8: 45924-45937, 2020.