VIRTUAL MOUSE CONTROL USING EYE MOVEMENT

A PROJECT REPORT

Submitted by

G KAVIYA	**812619104012**
USHA KUMARI	**812619104041**

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

M.A.M COLLEGE OF ENGINEERING, TIRUCHIRAPPALLI – 621105

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“VIRTUAL MOUSE CURSOR CONTROL USING EYE MOVEMENT”** is the bonafide work of **“G. KAVIYA (812619104012), USHA KUMARI (812619104041) ”** who carried out the project work under my supervision.

SIGNATURE
Mr.V.PUGAZHENTHI,M.E(CSE).,

HEAD OF THE DEPARTMENT
Computer Science and Engineering
M.A.M. College Of Engineering,
Siruganur,Trichy-621105

SIGNATURE
Mr.V.PUGAZHENTHI,M.E(CSE).,

SUPERVISOR
ASSOCIATE PROFESSOR
Computer Science and Engineering
M.A.M. College Of Engineering,
Siruganur,Trichy-621105

Certificated that the candidate is examined by us in the Project Viva Voce Examination held on ________________.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

“THANKS” is a simple word but its eloquence is magnified when it comes from the depth of the heart. We take this opportunity to thank encourage and supporters of this project.

First and foremost we praise and thank “The Lord Almighty” from the depthof our hearts which has been an unfailing source of strength, comfort and inspiration for this word.

We sincerely thank our beloved Secretary **Dr. M. A. MOHAMED NIZAM,** Director **Dr.V.SHANMUGANATHAN** and Principal **Dr.M.SHANMUGAPRIYA, M.E., PHD.,** for providing all the necessary facilities to do our work successfully.

We take this opportunity to express our deep sense of gratitude to our Headof the Department Computer Science Engineering and our guide **Mr.V.PUGAZHENTHI, M.E.** for supporting us throughout our venture and for their invaluable interest and support, constructive, suggestions, encouragement.
We are thankful to all staff members of CSE, M.A.M College of Engineering for their comments during our project . A hearty thanks to the non-teaching staff of CSE who provided us the necessary facilities and help in completing our project.

We are very thankful to our parents, friends who have been so encouraging and supporting us normally and helping us in time of despair.

ABSTRACT

Controlling the mouse by a physically challenged person is really a tough one. To find a solution for the people who cannot use the Mouse physically, we have proposed this mouse cursor control using Eye Movements. Eye gaze is an alternative way of accessing a computer using eye movements to control the mouse. For someone who fine touchscreens, mouse inaccessible, eye gaze is an alternative method to allow a user to operate their computer, using the movement of their eyes. Eye movement can be regarded as a pivotal real-time input medium for human-computer communication, which is especially important for people with physical disability. In order to improve the reliability, mobility, and usability of eye tracking technique in user-computer dialogue, a novel eye control system is proposed in this system using Webcam and without using any extra hardware. The proposed system focuses on providing a simple and convenient interactive mode by only using user's eye. The usage flow of the proposed system is designed to perfectly follow human natural habits. The proposed system describes the implementation of both iris and movement of cursor according to iris position which can be used to control the cursor on the screen using webcam and implemented using Python.

சுருக்கம்

உடல் ஊனமுற்ற நபரால் சுட்டியைக் கட்டுப்படுத்துவது உண்மையில் கடினமான ஒன்றாகும். உடல் ரீதியாக மவுஸைப் பயன்படுத்த முடியாதவர்களுக்கு தீர்வு காண, கண் அசைவுகளைப் பயன்படுத்தி இந்த மவுஸ் கர்சர் கட்டுப்பாட்டை முன்மொழிந்துள்ளோம். கண் பார்வை என்பது சுட்டியைக் கட்டுப்படுத்த கண் அசைவுகளைப் பயன்படுத்தி கணினியை அணுகுவதற்கான ஒரு மாற்று வழியாகும். சிறந்த தொடுதிரைகள், மவுஸ் அணுக முடியாத, கண் பார்வை என்பது ஒரு பயனர் தனது கண்களின் இயக்கத்தைப் பயன்படுத்தி தங்கள் கணினியை இயக்க அனுமதிக்கும் ஒரு மாற்று முறையாகும். மனித-கணினி தகவல்தொடர்புக்கான முக்கிய நிகழ்நேர உள்ளீட்டு ஊடகமாக கண் இயக்கம் கருதப்படுகிறது, இது உடல் ஊனமுற்றவர்களுக்கு மிகவும் முக்கியமானது. பயனர்-கணினி உரையாடலில் கண் கண்காணிப்பு நுட்பத்தின் நம்பகத்தன்மை, இயக்கம் மற்றும் பயன்பாட்டினை மேம்படுத்துவதற்காக, இந்த அமைப்பில் வெப்கேமைப் பயன்படுத்தி மற்றும் கூடுதல் வன்பொருளைப் பயன்படுத்தாமல் ஒரு புதிய கண் கட்டுப்பாட்டு அமைப்பு முன்மொழியப்பட்டது. முன்மொழியப்பட்ட அமைப்பு பயனரின் கண்ணை மட்டுமே பயன்படுத்துவதன் மூலம் எளிய மற்றும் வசதியான ஊடாடும் பயன்முறையை வழங்குவதில் கவனம் செலுத்துகிறது. முன்மொழியப்பட்ட அமைப்பின் பயன்பாட்டு ஓட்டம் மனித இயற்கை பழக்கவழக்கங்களை முழுமையாக பின்பற்ற வடிவமைக்கப்பட்டுள்ளது. முன்மொழியப்பட்ட அமைப்பு கருவிழியின் நிலையின்படி கருவிழி மற்றும் கர்சரின் இயக்கம் இரண்டையும் செயல்படுத்துவதை விவரிக்கிறது, இது வெப்கேமைப் பயன்படுத்தி திரையில் கர்சரைக் கட்டுப்படுத்தவும் பைத்தானைப் பயன்படுத்தி செயல்படுத்தவும் பயன்படுகிறது.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ACKNOWLEDGEMENT	**iii**
	ABSTRACT	**iv**
	LIST OF FIGURES	**viii**
	LIST OF ABBREVIATIONS	**ix**
1	**INTRODUCTION**	1
	1.1 Scope of the project	1
	1.2 Project Description	2
2	**LITERATURE SURVEY**	3
	2.1 Virtual mouse with RGB Colored Tapes	3
	2.2 AI Virtual mouse using hand recognition gestures	3
	2.3 Real-Time virtual mouse system using RGB-D images using fingertip detection	4
	2.4 Hand Gesture Recognition System as Virtual Mouse for HCI (GLOVE)	5
3	**SYSTEM ANALYSIS**	7
	3.1 Existing System	7
	3.1.1 Disadvantages	7
	3.2 Proposed System	8
	3.2.1 Advantages	8
	3.3 Modules Specifications	8
	3.4 Modules Description	9
	3.5 System Study	11

4 DEVELOPMENT ENVIORNMENT 13

4.1 System Specification 13

4.1.1 Hardware Requirements 13

4.1.2 Software Requirement 13

4.2 Software Description 13

5 SYSTEM DESIGN 26

5.1 System Architecture 26

5.2 Block Diagram 27

5.3 Data Flow Diagram 27

5.4 Use Case Diagram 28

5.5 Class Diagram 29

5.6 Activity Diagram 30

6 SYSTEM IMPLEMENTATION 31

6.1 Implementation Details 31

6.2 Methodology 32

7 SYSTEM TESTING 33

7.1 White Box Testing 34

7.2 Black Box Testing 34

7.3 Unit Testing 35

7.4 Test Strategy and Approach 35

7.5 Integration Testing 35

7.6 Acceptance Testing 35

8 PERFORMANCES AND LIMITATIONS 36

8.1 Conclusion 36

8.2 Future Enhancement 36

APPENDICES 37

A. Source code 37

B. Screenshots 45

REFERENCES 48

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO
3.4	EYE-ASPECT-RATIO	10
3.4	MOUTH-ASPECT-RATIO	11
5.1	SYSTEM ARCHITECTURE	26
5.2	BLOCK DIAGRAM	27
5.3	DATA FLOW DIAGRAM	28
5.4	USE CASE DIAGRAM	29
5.5	CLASS DIAGRAM	30
5.6	ACTIVITY DIAGRAM	30
6.1	TRACKING EYE MOVEMENT	31

LIST OF ABBREVIATIONS

ABBREVIATIONS	EXPANSIONS
MAR	MOUTH-ASPECT-RATIO
EAR	EYE-ASPECT-RATIO
GPL	GENERAL PUBLIC LICENSE
GUI	GRAPHICAL USER INTERFACE
CPU	CENTRAL PROCESSING UNIT
DBMS	DATABASE MANAGEMENT SYSTEM
DFD	DATA FLOW DIAGRAM
UML	UNIFIED MODELING LANGUAGE
WBT	WHITE BOX TESTING

CHAPTER 1

1.INTRODUCTION

1.1 SCOPE OF THE PROJECT

There are hundreds of cameras capturing our faces and using it for various purposes. Despite of such advancement in technology, there is no such proper system that can help physically challenged people for using a personal computer. The current systems which exist, help the users in controlling a desktop without the usage of their hands. But the major disadvantage of these systems is that, they are expensive. Our main objective is to make the system cost effective, so that the system is affordable. The main beneficiaries of our system will be the people who have physical impairment of their hands. Our proposed system will use the concepts of eye gaze, eye movement, head movement, blink detection to track the user's eye.

The scope of our project is to make the work of 'amputees' (people who don't have their arms to be operational) easy. Amputees or quadriplegics can benefit from our project (people affected by paralysis of all four limbs) can use and operate the mouse using their facial features and actions of their eyes.

To deliver a user-friendly human-computer interaction project, this project will design a system that will just require a camera to employ human eyes and facial characteristics as a pointing device for the computer system. The following are the objectives:

- Eyes and face Detection
- Eye end points extraction
- Develop an algorithm to calculate the point gaze based on eye features extracted
- Develop a GUI that shows a result
- Develop a Calibration technique

By accurately detecting the position of the eye and mapping that to a specific position on the computer screen, the algorithm enables physically disabled individuals to control the computer cursor movement to the left, right, up and down.

By our project the person with disabilities can also access the computer for their purposes like a normal human. This increases the human computer interaction for everyone.

1.2 PROJECT DESCRIPTION

The system is a mouse-like eye-based interface that converts eye movements like blinking, staring, and squinting into mouse cursor actions. The software requirements for this technique include Python, OpenCV, NumPy, and a few more facial recognition tools, as well as a basic camera.

The method described in this paper is distinctive because unlike existing methods we did not use electrodes, infrared, or any other light source to track the eyes. The only hardware that is required is a PC or laptop along with a webcam, which makes it practical and feasible.

By taking consecutive snaps of the user from the camera, the program is designed to process these frames individually at very high speed of processing and compare the iris shift in each frame with respect to the initial frame.

The frame undergoes several stages of processing before the eyes can be tracked. After obtaining the processed image, the iris shift is calculated and the program prompts the cursor on the screen to move to the respective location.

The first step was to use a face detection algorithm locate the face on an image frame captured by an ordinary webcam. The next step was to detect only the eyes from this frame. We consider tracking only one eye movement for faster processing time. Then the iris movement was tracked. Since the color of the iris is black, its image has a significantly lower intensity compared to the rest of the eye.

This helps us in easy detection of the iris region. Taking the left and right corners of the eye as reference points, the shift of the iris as the person changed his eyes focus was determined. The shift was then used to map cursor location on the test graphical user interface (GUI).

CHAPTER 2

2. LITERATURE SURVEY

2.1 TITLE: VIRTUAL MOUSE WITH RGB COLORED TAPES

AUTHOR: UPASANA V, MONISHA JOSEPH, KANCHANA V

DESCRIPTION: The aim of this research is to investigate a new approach for controlling the mouse movements using a real-time camera. Instead of changing or adding mouse parts, a method where access to the mouse/ cursor by hand gestures with colored tapes can be enabled. It tries to use a camera and computer vision technologies such as gesture recognition and image segmentation to control mouse tasks with colored tapes and shows how it can perform all the mouse functions a current mouse device can.

In the first phase, hand gestures are acquired using a camera based on colored detection technique using segmentation and image subtraction algorithm.

In the second phase, RGB colored tapes are used to control different functions of the mouse and also the combination of these three colors by considering the area of each object/colored tape using the blob analysis and a bounding box algorithm. The user must wear the red , green and blue tapes to the finger such that it is easy to make the movements for each tape and also the combination of colored tapes to acquire the desired output of the cursor movement in the system.

ADVANTAGES

- System will avoid COVID-19 spread by eliminating the human intervention and dependency of devices to control the computer.
- RGB colored tapes are used to control different functions of the mouse

DISADVANTAGE

- The web camera captures the video at a fixed frame rate .
- The camera captures an image, it is an inverted one.

2.2 TITLE: AI VIRTUAL MOUSE USING HAND GESTURE RECOGNITION

AUTHOR: JOY G , SHREYA KUMARI , SHIV KUMAR VERMA

DESCRIPTION: The PC mouse is one of the wondrous developments of people in the field of Human-Computer Interaction (HCI) innovation. In new age of innovation, remote mouse or a contact less mouse actually utilizes gadgets and isn't liberated from gadgets completely, since it utilizes power from the gadget or might be from outside power sources like battery and gain space and electric power, likewise during COVID pandemic it is encouraged to make social separating and keep away from to contact things which gave by various people groups. Inside the projected AI virtual mouse utilizing hand signal framework, this constraint might be resolve by involving advanced camera or sacred camera for perceive the hand motions and fingers recognition abuse PC machine vision. The algorithmic rule used in the framework utilizes the man-made consciousness and AI algorithmic rule. Upheld the hand signals, the gadget might be controlled pretty much and might do left click, right snap, looking over capacities, and PC gadget pointer perform while not the utilization of the genuine mouse. Index Terms: deep learning base computer vision, real time mouse system, Media-pipe , HCI.

ADVANTAGES

- Provides 99% accurate results compared to other techniques.
- Image processing operations such as capturing image, removing noise, background subtraction, and edge detection**.**

DISADVANTAGE

- Difficulties in clicking and dragging to select the text.
- Small decrease in accuracy in right click mouse function
- Depend upon lighting condition in the environment

2.3 TITLE: REAL-TIME VIRTUAL MOUSE SYSTEM USING RGB-D IMAGES FINGERTIP DETECTION

AUTHOR: DINH-SON TRAN1 & NGOC-HUYNH HO1 & HYUNG-JEONG YANG1 & SOO-HYUNG KIM1 & GUEE SANG LEE

DESCRIPTION: A real-time fingertip-gesture-based interface is still challenging for human–computer interactions, due to sensor noise, changing light levels, and the complexity of tracking a fingertip across a variety of subjects. Using fingertip tracking as a virtual mouse is a popular method of interacting with computers without a mouse device. In this work, we propose a novel virtual mouse method using RGB-D images and fingertip detection. The hand region of interest and the center of the palm are first extracted using in-depth skeleton-joint information images from a Microsoft Kinect Sensor version 2, and then converted into a binary image. Then, the contours of the hands are extracted and described by a border-tracing algorithm. The K cosine algorithm is used to detect the fingertip location, based on the hand-contour coordinates. Finally, the fingertip location is mapped to RGB images to control the mouse cursor based on a virtual screen. The system tracks fingertips in real-time at 30 FPS on a desktop computer using a single CPU and Kinect V2. The experimental results showed a high accuracy level; the system can work well in real-world environments with a single CPU. This fingertip-gesture-based interface allows humans to easily interact with computers by hand.

ADVANTAGES:

- Provides information about person’s view when moving over a wide area.
- The rate of the considered devices ranges from 30 Hz to 400 Hz. Devices with high sampling frequency can record up to respectively 400 Hz and 250 Hz, thus allowing the measurement of faster eye movement

DISADVANTAGES:

- The devices can track the eye using several cameras and multiple corneal reflections per eye.
- The eye gaze tracking devices usually have a smaller field of view within which the gaze point can be resolved.

2.4 TITLE: HANDS GESTURES RECOGNITION SYSTEM AS VIRTUAL MOISR FOR HCI (GLOVE)

AUTHOR: Mr. VENKATESHWAR A , MAHESHWARI PRAKASH BHAIRAWADAG , PAVAN KUMAR P, GOUTHAM U, KALYAN KUMAR P

DESCRIPTION: The mouse is one of the invention in HCI (human computer interaction) technology. Though wireless are Bluetooth mouse technology is invented still, that technology is not completely device free. A Bluetooth mouse has the requirement of battery power it requires extra power supply. Presence of extra devices in a mouse increases the difficulty level of more hardware components. The proposed mouse system is outside this limitation. This paper proposes a virtual mouse system using colored hand glove based on HCI using computer vision and hand gestures. Gestures captured with a webcam on processed with color segmentation, detection technique and feature extraction. The user will be allowed to control some of the computer cursor functions with a colored glove on the hand. Primarily, a user can perform with their fingers, scrolling up or down using their hands in different gestures. This system captures frames using a webcam or built-in cam it is based on the camera quality. So the usage of colored glove mouse system eliminates device dependency in order to use a mouse.

ADVANTAGES:

- Able to recognize RGB movements, combinations, and translate actual mouse functions to perform operations.
- Cverrules the touch technology.
- Cheapest system which works fine in a standardized operating system and easy to use.

DISADVANTAGES:

- Captures images in regular time interval.
- Algorithm used may or may not work in all environment.
- Does not support on low camera quality**.**

CHAPTER 3

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Chin et al. proposed a cursor control system for computer users, which integrated the electromyogram signals from muscles in the face and point-of-gaze coordinates produced by an eye-gaze tracking system as inputs. Although it could facilitate a reliable click operation, it was slower than the control system that only used eye tracking and the accuracy was low.

Missimer and Betke constructed a system that uses the head position to control the mouse cursor and simulates left-click and right-click of the mouse by blinking left or right monocular. This system relied on the position of user’s head to control the mouse cursor position. The irregular movement of user’s head would affect the accuracy of click function.

Lupu et al. proposed a communication system for people with disabilities, which was based on a special designed device composed of a webcam mounted on glasses frame for image acquisition and processing. The eye movement is detected by the device and the voluntary eye blinking is correlated with a pictogram or keyword selection reflecting patient’s needs. The drawback of this system is that the image processing algorithm could not accurately detect the acquired image with low quality and is not robust to light intensity.

3.1.1 DISADVANTAGES

- The existing system designed eye trackers are quite complicated and expensive.
- Users should wear inconvenient devices and make specific actions to control the system.
- In the existing system user should choose the desired function first and then do the real interaction with computer, which goes against user intuition and it is not natural to use.

3.2 PROPOSED SYSTEM

In this paper, we have described a system that has facial tracking, detection of face, detection of eye and continuous eye blinks for controlling a nonintrusive human computer interface.

The mouse cursor is controlled by head movement as moving face up, down, sidewise movements and mouse events such as right and left click are controlled through eye blinks and squinting using OpenCV. This system is mainly for disabled people to have effective communication with the computer. Human eye movement has been used in the place of the conventional mouse. The main aim of this system is to help physically challenged and amputee people especially person who do not have their arms.

Camera receives the input from the eye. After receiving these streaming videos from the camera, it will break into frames. After receiving frames, it will check for lighting conditions because cameras require sufficient lights from external sources otherwise error message will be displayed on the screen. Images (frames) from the input source focusing the eye are analyzed for Iris detection (center of eye). The second step is to find the exact position of the iris within the eye window. Mapping the iris to a point in the video from the scene camera. Using pre-determined calibration points, the position of the iris is mapped to a position on the screen. After this, a mid-point is calculated by taking the mean of left and right eye center point. Finally the mouse will move from one position to another on the screen based on the movements tracked. For example, if you just close and open your right eyes, then the right click operation of mouse is performed. In such way, the mouse cursor movements and other operations of mouse is done.

3.2.1 ADVANTAGES

- Increased productivity due to reduction of mouse interaction time
- Increases HCI for physically challenged people.

3.2 MODULES SPECIFICATIONS

- Importing the necessary libraries
- Open cv
- Eye detection Using Dlib
- How It Works

- Eye-Aspect-Ratio (EAR)
- Mouth-Aspect-Ratio (MAR)
- Prebuilt Model Details

3.4 MODULES DESCRIPTION

Importing the necessary libraries:

We will be using Python language for this. First we will import the necessary libraries such as open cv and dlib for building the main model

Open cv:

Open CV (Open Source Computer Vision Library) is an open library of python that is mainly aimed at real-time computer-vision. It is also available in C++ and Java. It is an open source machine learning software library. It makes use of Numpy, which is a python library that is used for implementing multi-dimensional arrays and matrices along with high-level mathematical operations on these arrays. OpenCV is mainly used to capture data from a live video hence it is mainly focuses on image processing and video capture. In this paper OpenCV is focused mainly on video capture. OpenCV is also used for applications such as face detection, OCR, Vision-guided robotics surgery, 3D human organ reconstruction, QR code Scanner etc., Using OpenCV we can perform detection of specific objects such as eyes, faces etc., we can also analyze videos such as estimating the motion in the video or subtracting the background from the video, and tracking objects [8] in it. OpenCV covers the basic data structures such as Scalar, point etc. that are used for building OpenCV applications. OpenCV library is imported into the python using the code ‘import cv2’.

Eye detection Using Dlib:

The first thing to do is to find eyes before we can move on to image processing and to find the eyes we need to find a face. The facial keypoint detector takes a rectangular object *of the dlib module* as input which is simply the coordinates of a face. To find faces we can use the inbuilt frontal face detector of dlib. You can use any classifier for this task. If you want high accuracy and speed is not an issue for you then I would suggest you use a CNN as it will give much better accuracy especially for non-frontal facing faces

How It Works:

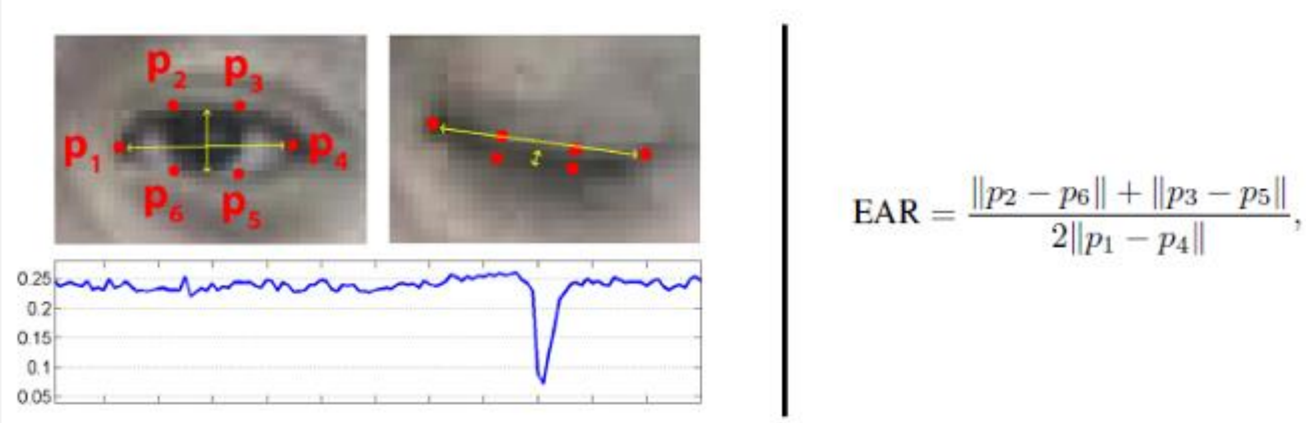
This project is deeply centered around predicting the facial landmarks of a given face. We can accomplish a lot of things using these landmarks. From detecting eye-blinks in a video to predicting emotions of the subject. The applications, outcomes, and possibilities of facial landmarks are immense and intriguing.

Dlib’s prebuilt model, which is essentially an implementation of , not only does a fast face-detection but also allows us to accurately predict 68 2D facial landmarks. Very handy

Using these predicted landmarks of the face, we can build appropriate features that will further allow us to detect certain actions, like using the eye-aspect-ratio (more on this below) to detect a blink or a wink, using the mouth-aspect-ratio to detect a yawn etc or maybe even a pout. In this project, these actions are programmed as triggers to control the mouse cursor. PyAutoGUI library was used to move the cursor around.

Eye-Aspect-Ratio (EAR):

You will see that Eye-Aspect-Ratio is the simplest and the most elegant feature that takes good advantage of the facial landmarks. EAR helps us in detecting blinks and winks etc.



$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|},$$

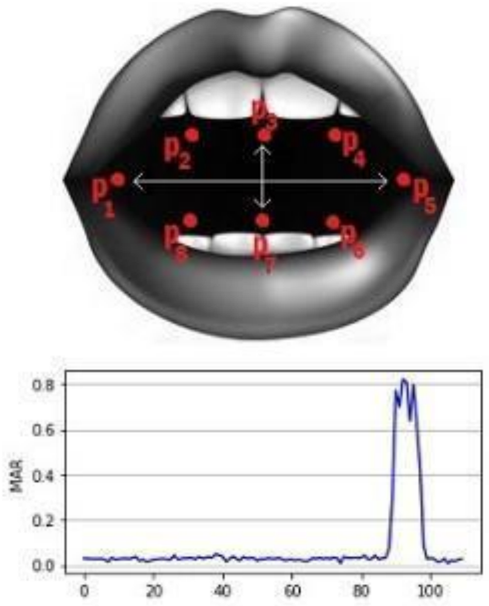
Source: Tereza Soukupova´ and Jan C ˇ ech. *Real-Time Eye Blink Detection using Facial Landmarks* at https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf

You can see that the EAR value drops whenever the eye closes. We can train a simple classifier to detect the drop. However, a normal *if* condition works just fine. Something like this:

```
if EAR <= SOME_THRESHOLD:
    EYE_STATUS = 'CLOSE'
```

Mouth-Aspect-Ratio (MAR):

Highly inspired by the EAR feature, I tweaked the formula a little bit to get a metric that can detect opened/closed mouth. Unoriginal but it works.



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

Similar to EAR, MAR value goes up when the mouth opens. Similar intuitions hold true for this metric as well.

Prebuilt Model Details:

The model offers two important functions. A detector to detect the face and a predictor to predict the landmarks. The face detector used is made using the classic Histogram of Oriented Gradients (HOG) feature combined with a linear classifier, an image pyramid, and sliding window detection scheme

You can get the trained model file from http://dlib.net/files, click on **shape_predictor_68_face_landmarks.dat.bz2**. The model, .dat file has to be in the project folder.

3.5 SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 4

4. DEVELOPMENT ENVIRONMENT

4.1 SYSTEM SPECIFICATION

4.1.1 HARDWARE REQUIREMENTS

● System	:	Pentium i3 Processor.
● Hard Disk	:	500 GB.
● Monitor	:	15’’ LED
● Input Devices	:	Keyboard, Mouse
● Ram	:	4 GB

4.1.2 SOFTWARE REQUIREMENTS

● Operating system	:	Windows 10.
● Coding Language	:	Python
● Web Framework	:	Flask

4.2 SOFTWARE DESCRIPTION

Python:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python https://www.python.org.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link for the Windows installer python-XYZ.msifile where XYZ is the version you need to install.
- To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

First Python Program

Let us execute programs in different modes of programming.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

```
$ python

Python2.4.3(#1,Nov112010,13:34:43)

[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2

Type"help","copyright","credits"or"license"for more information.

>>>
```

Type the following text at the Python prompt and press the Enter –

```
>>>print"Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**. However in Python version 2.4.3, this produces the following result −

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file −

```
print"Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

```
$ python test.py
```

This produces the following result −

```
Hello, Python!
```

Flask Framework:

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods −

Sr.No	Methods & Description
1	**GET** Sends data in unencrypted form to the server. Most common method.
2	**HEAD** Same as GET, but without response body
3	**POST** Used to send HTML form data to server. Data received by POST method is not cached by server.

4	**PUT** Replaces all current representations of the target resource with the uploaded content.
5	**DELETE** Removes all current representations of the target resource given by a URL

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

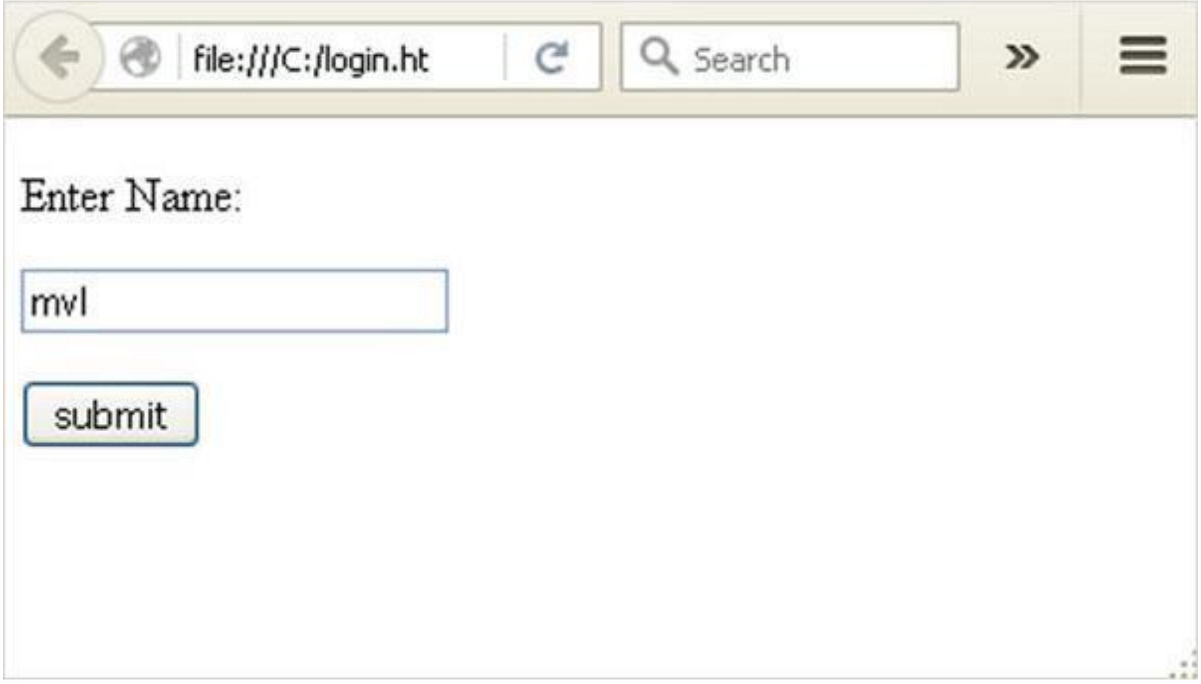
Save the following script as login.html

```
<html>
<body>
<formaction="http://localhost:5000/login"method="post">
<p>Enter Name:</p>
<p><inputtype="text"name="nm"/></p>
<p><inputtype="submit"value="submit"/></p>
</form>
</body>
</html>
```

Now enter the following script in Python shell.

```
from flask importFlask, redirect,url_for, request

app=Flask(__name__)

@app.route('/success/<name>')

def success(name):

return'welcome %s'% name

@app.route('/login',methods=['POST','GET'])

def login():

ifrequest.method=='POST':

user=request.form['nm']

return redirect(url_for('success',name= user))

else:

user=request.args.get('nm')

return redirect(url_for('success',name= user))

if __name__ =='__main__':

app.run(debug =True)
```

After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**.



Enter Name:

mvl

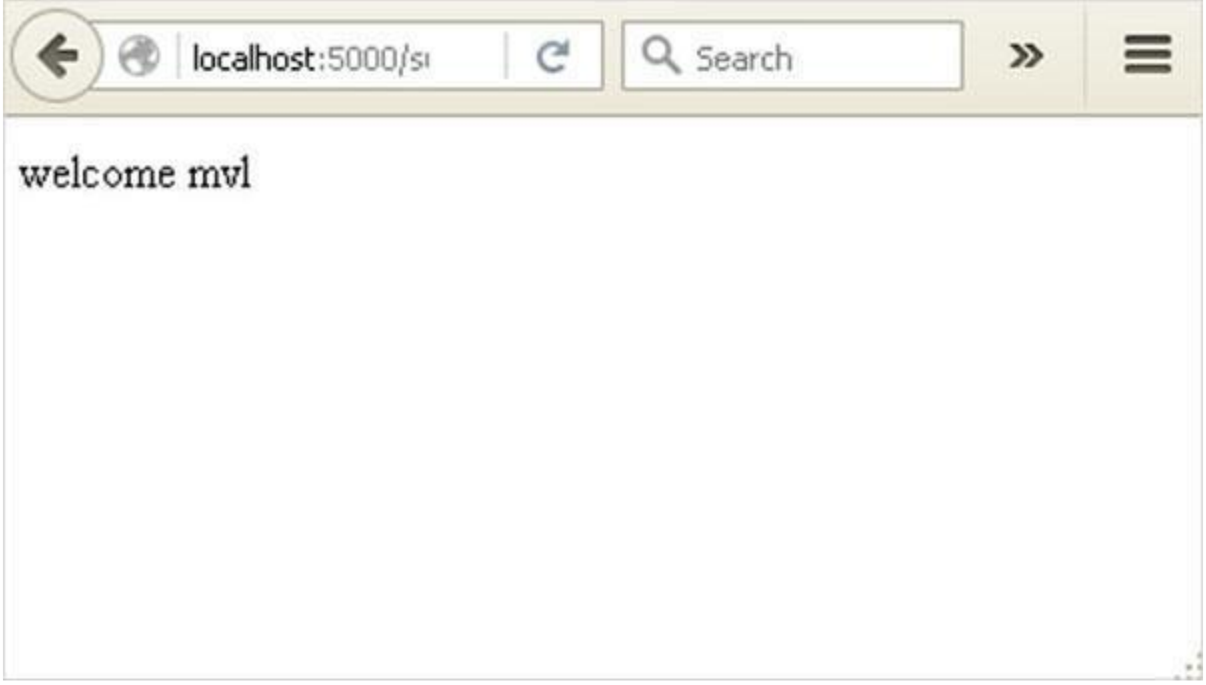
submit

Form data is POSTed to the URL in action clause of form tag.

http://localhost/login is mapped to the **login()** function. Since the server has received data by **POST** method, value of ‘nm’ parameter obtained from the form data is obtained by −

```
user = request.form['nm']
```

It is passed to **‘/success’** URL as variable part. The browser displays a **welcome** message in the window.



welcome mvl

Change the method parameter to **‘GET’** in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of ‘nm’ parameter is now obtained by −

```
User = request.args.get(‘nm’)
```

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to ‘nm’ parameter is passed on to ‘/success’ URL as before.

What is Python?

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: https://www.python.org/

Python Quick start

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

```
helloworld.py
```

```
print("Hello, World!")
```

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

```
C:\Users\Your Name>python
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
```

```
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
```

Which will write "Hello, World!" in the command line:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
```

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

```
exit()
```

Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

```
>>> print("Hello, World!")
Hello, World!
```

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

```
C:\Users\Your Name>python myfile.py
```

Python Indentations

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important.

Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
  print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

```
if 5 > 2:
print("Five is greater than two!")
```

Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

Example

Comments in Python:

```
#This is a comment.
print("Hello, World!")
```

Docstrings

Python also has extended documentation capability, called docstrings.

Docstrings can be one line, or multiline.

Python uses triple quotes at the beginning and end of the docstring:

Example

Docstrings are also comments:

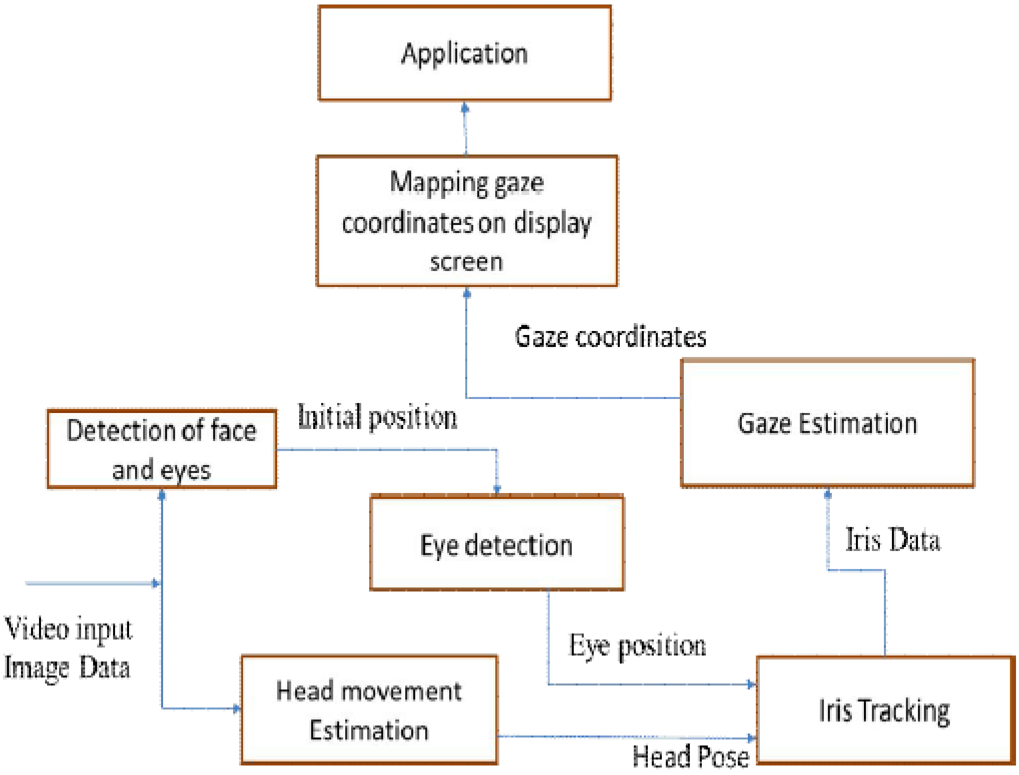
```
"""This is a
multiline docstring."""
print("Hello, World!")
```

CHAPTER-5

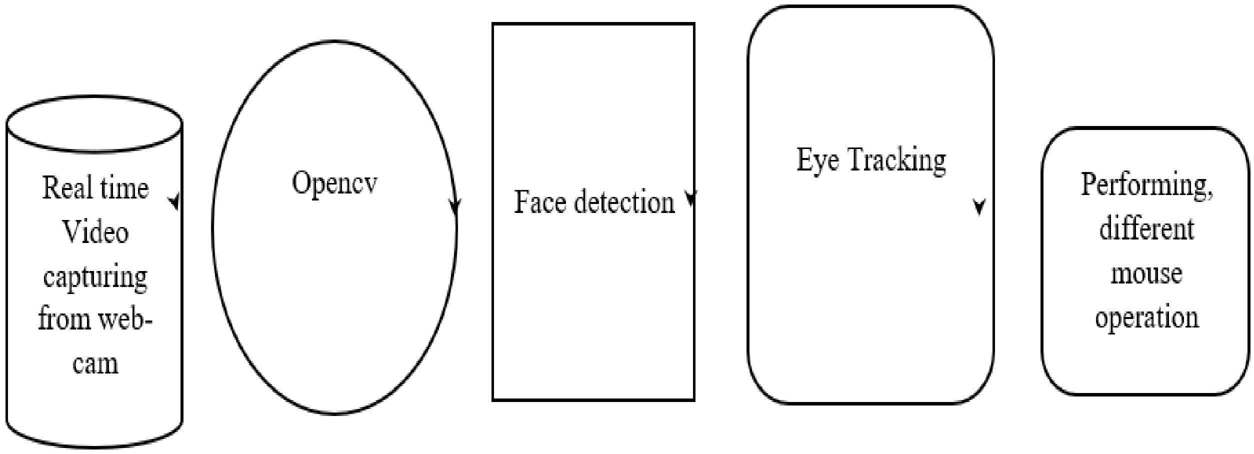
5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

System architecture refers to the overall design and structure of a computer system or software application. It involves the organization and interconnection of hardware components, software modules, and communication protocols to ensure that the system functions efficiently, reliably, and securely. 8.System architecture typically includes several key components, such as the central processing unit (CPU), memory, input/output devices, storage devices, network interfaces, and operating system software. The architecture also includes the application software and the database management system (DBMS), which manage the data and processes of the system.

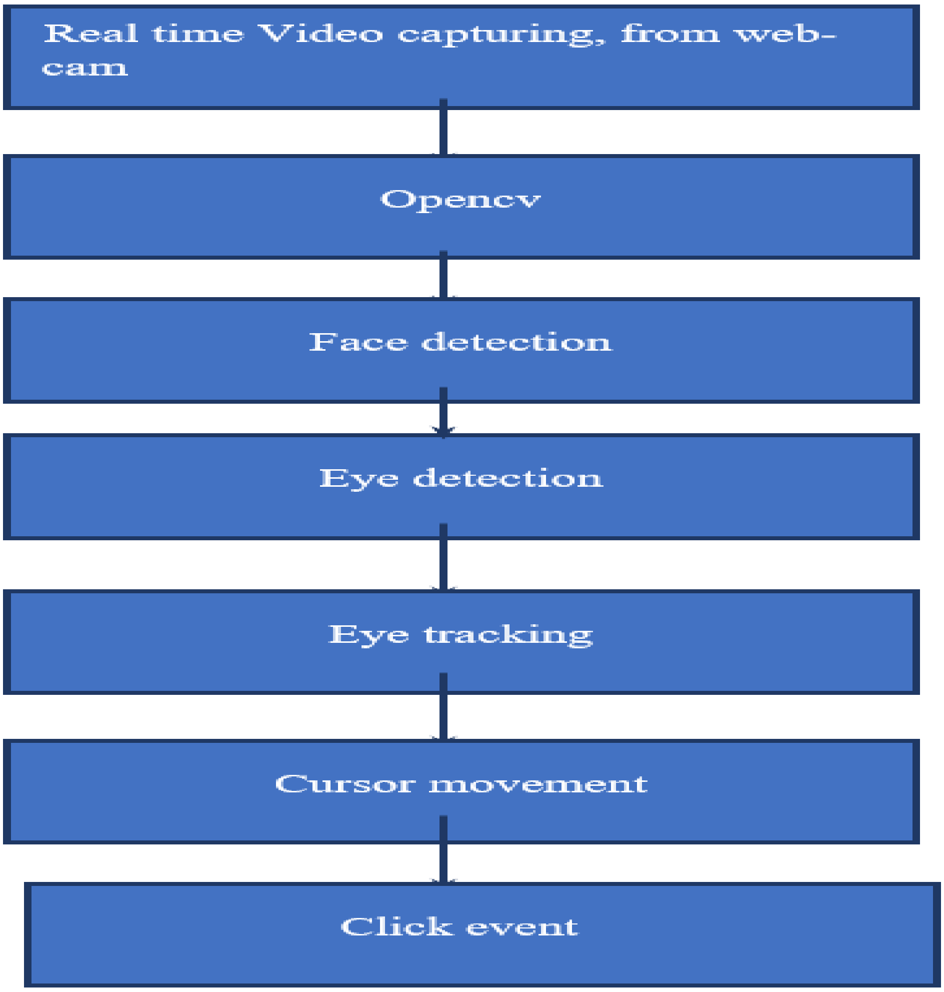


5.2 BLOCK DIAGRAM


Real time Video capturing from web-cam
Opencv
Face detection
Eye Tracking
Performing, different mouse operation

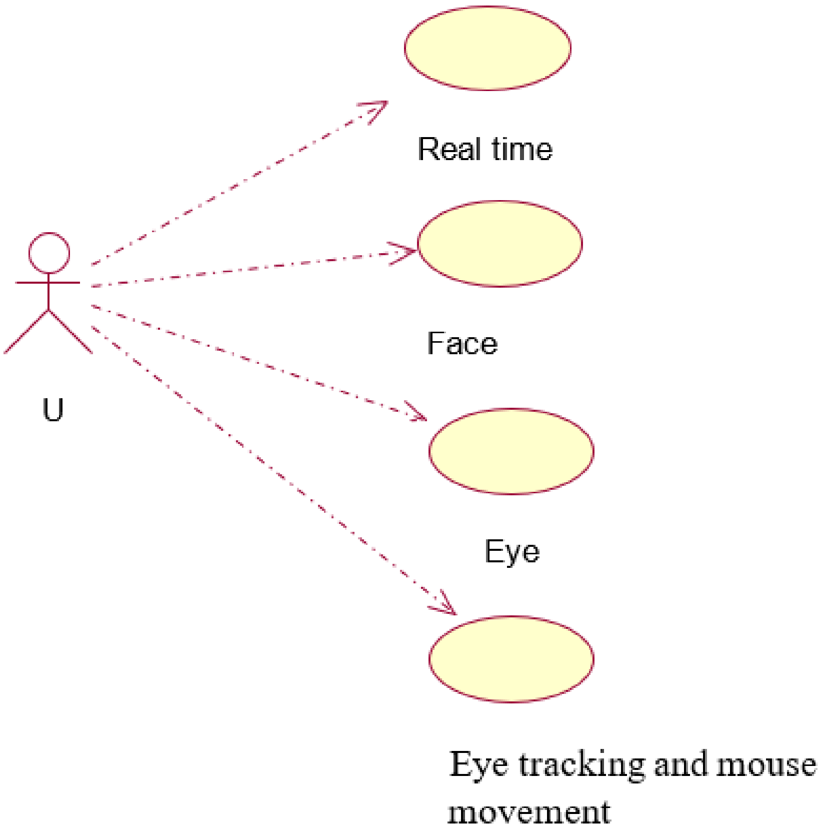
5.3 DATA FLOW DIAGRAM

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



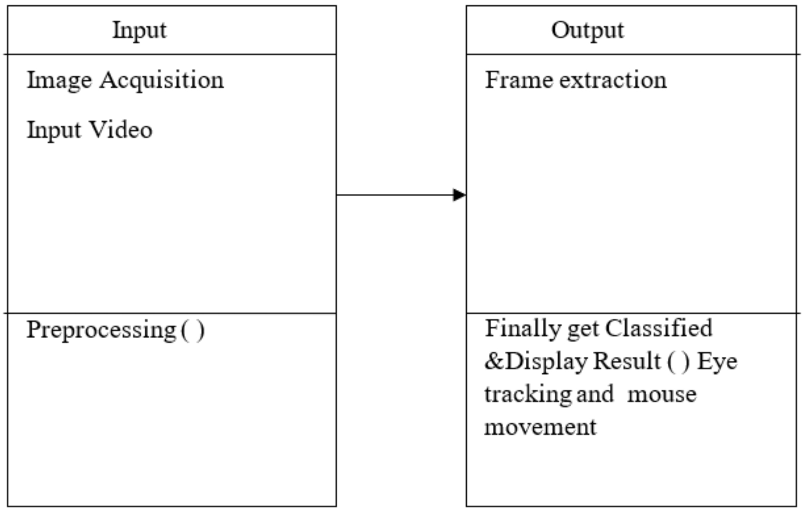
5.4 USECASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



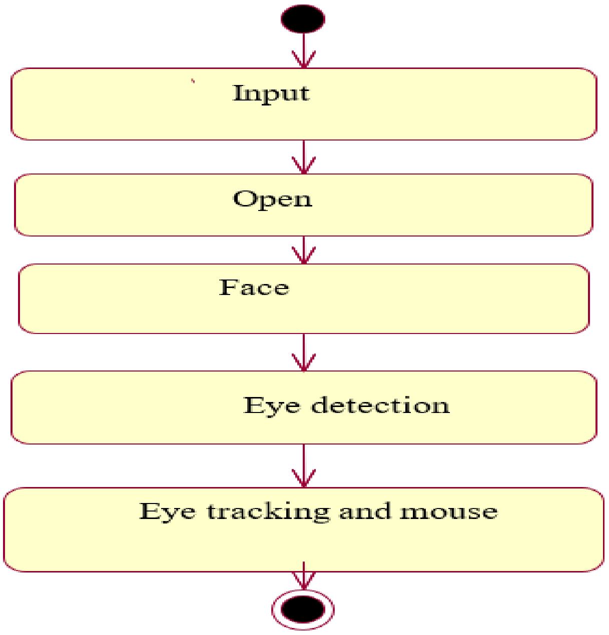
5.5 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.


Input
Image Acquisition
Input Video
Preprocessing ( )
Output
Frame extraction
Finally get Classified &Display Result ( ) Eye tracking and mouse movement

5.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



CHAPTER 6

6. SYSTEM IMPLEMENTATION

6.1 IMPLEMENTATION DETAILS

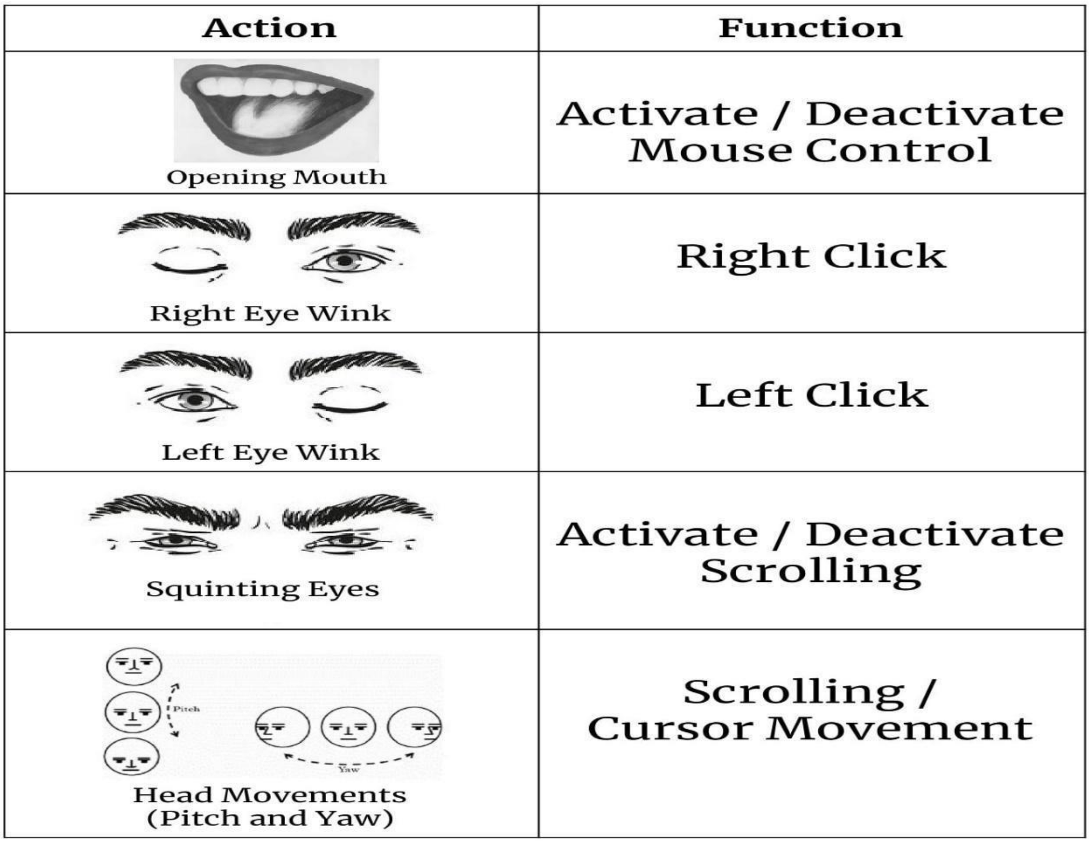
A. Face detection

In order to capture the face image accurately, the user sat upright with the eye level parallel to the webcam.

B. Extracting eye area

In order to extract the eyes region, the image of the face is divided into three equal horizontal areas. The upper third where the eyes are located is extracted

C. Tracking eye movement



Action	Function
Opening Mouth	Activate / Deactivate Mouse Control
Right Eye Wink	Right Click
Left Eye Wink	Left Click
Squinting Eyes	Activate / Deactivate Scrolling
Head Movements (Pitch and Yaw)	Scrolling / Cursor Movement

6.2 METHODOLOGY

The first step was to use a face detection algorithm locate the face on an image frame captured by an ordinary webcam. The next step was to detect only the eyes from this frame. Consider tracking only one eye movement for faster processing time. Then the iris movement was tracked. Since the color of the iris is black, its image has a significantly lower intensity compared to the rest of the eye. This helps us in easy detection of the iris region. Taking the left and right corners of the eye as reference points, the shift of the iris as the person changed his eyes focus was determined. The shift was then used to map cursor location on the test graphical user interface (GUI)

CHAPTER 7

7. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

7.1White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

7.2 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7.3 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

7.4 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.5 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.6 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered

CHAPTER 8

8. PERFORMANCE AND LIMITATIONS

8.1 CONCLUSION

In order to make user interact with computer naturally and conveniently by only using their eye, we provide an eye tracking based control system. The system combines both the mouse functions and keyboard functions, so that users can use our system to achieve almost all of the inputs to the computer without traditional input equipment. The system not only enables the disabled users to operate the computer the same as the normal users do but also provides normal users with a novel choice to operate computer. According to our experiments, it is considered our eye movement system to be easy to learn. Meanwhile, participants show their interest in using the proposed eye control system to search and browse information. They are looking forward to see more of our research results on the use of eye tracking technique to interact with the computer.

8.2 FUTURE ENHANCEMENTS

Currently, this system is applied for the general operating behavior to interact with computer by simulating mouse. In future, we will try to add new operation functions for more usage situations for users to communicate with media and adjust our system on new platform, such as tablet or phone. We will also develop series operation modules in order to achieve a complete operating experience for users from turning on to turning off the computer.

APPENDICES

SOURCE CODE

```
from imutils import face_utils
from utils import *
import numpy as np
import pyautogui as pag
import imutils
import dlib
import cv2

# Thresholds and consecutive frame length for triggering the mouse action.
MOUTH_AR_THRESH = 0.3
MOUTH_AR_CONSECUTIVE_FRAMES = 5
EYE_AR_THRESH = 0.20
EYE_AR_CONSECUTIVE_FRAMES = 5
WINK_AR_DIFF_THRESH = 0.001
WINK_AR_CLOSE_THRESH = 0.2
WINK_CONSECUTIVE_FRAMES = 4

# 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 = 1300
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)
```

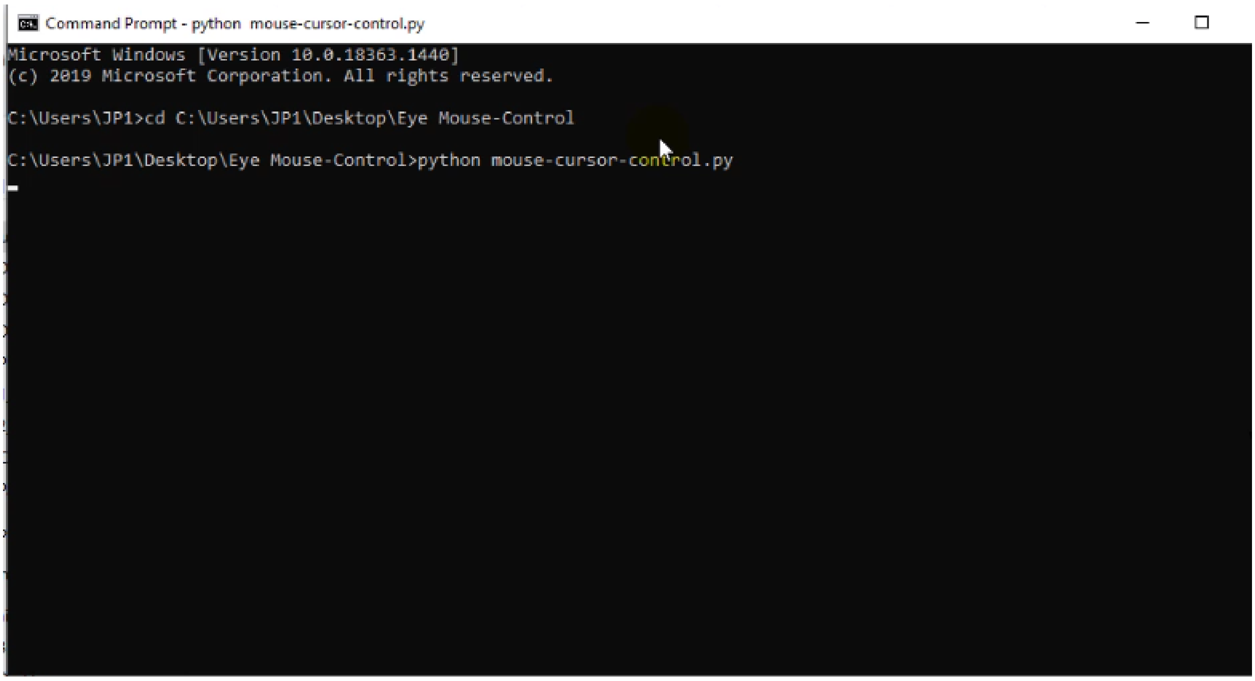
```
# Detect faces in the grayscale frame
rects = detector(gray, 0)

# 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:
                pag.click(button='left')

                WINK_COUNTER = 0

    elif leftEAR > rightEAR:
        if rightEAR < EYE_AR_THRESH:
            WINK_COUNTER += 1
            if WINK_COUNTER > WINK_CONSECUTIVE_FRAMES:
                pag.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':
            pag.moveRel(drag, 0)
        elif dir == 'left':
            pag.moveRel(-drag, 0)
        elif dir == 'up':
            if SCROLL_MODE:
                pag.scroll(40)
            else:
                pag.moveRel(0, -drag)
        elif dir == 'down':
            if SCROLL_MODE:
                pag.scroll(-40)
            else:
                pag.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_SIMPLqEX, 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

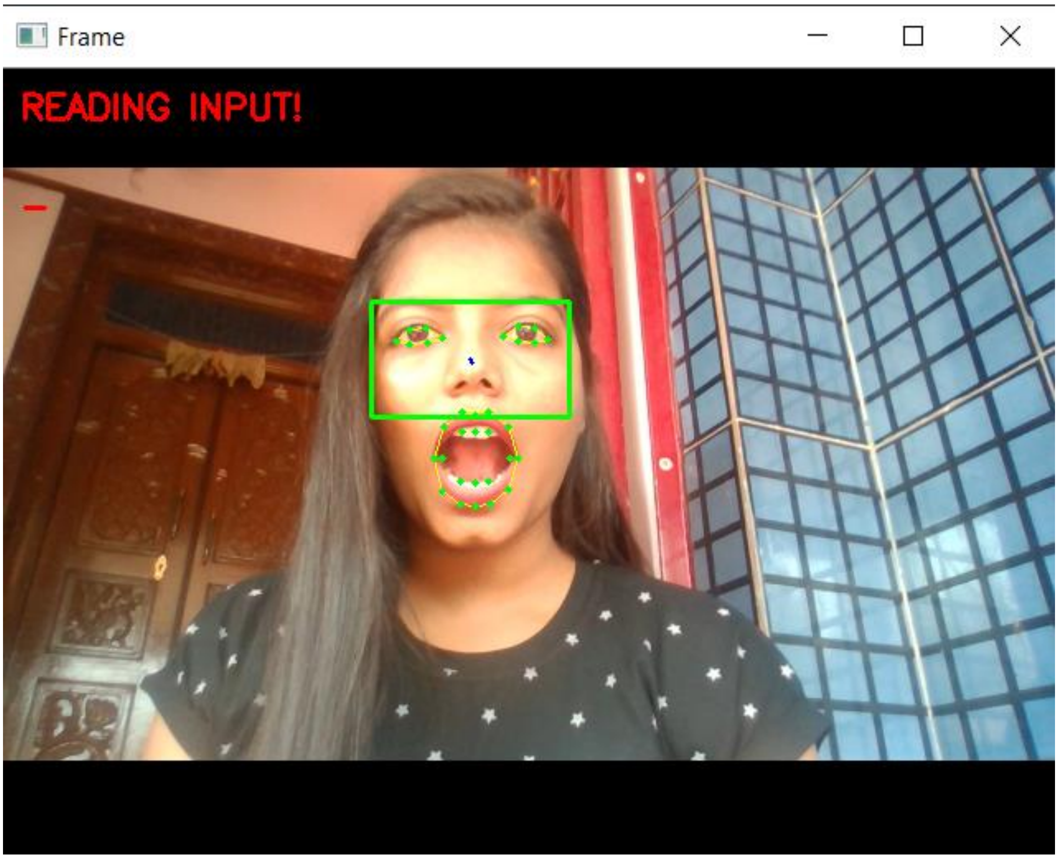
```
import numpy as np

# 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
    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:
        return 'up'

    return '-'
```

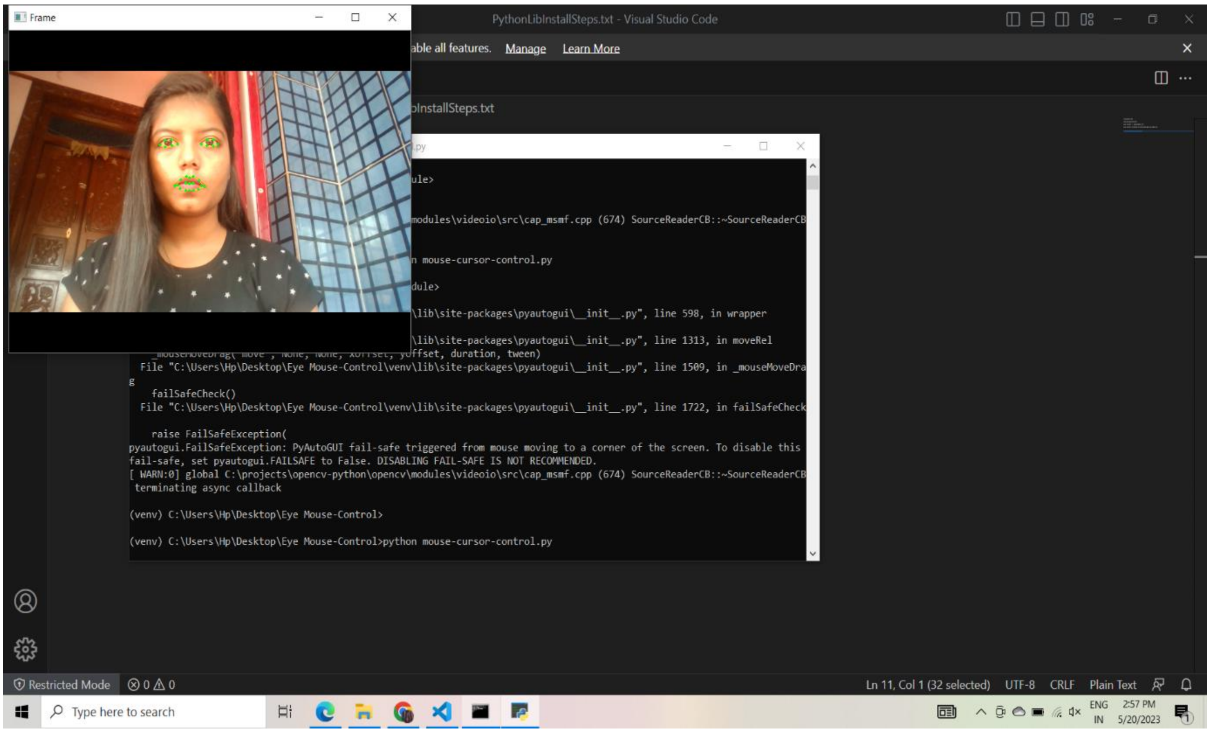
B. SCREESHOTS



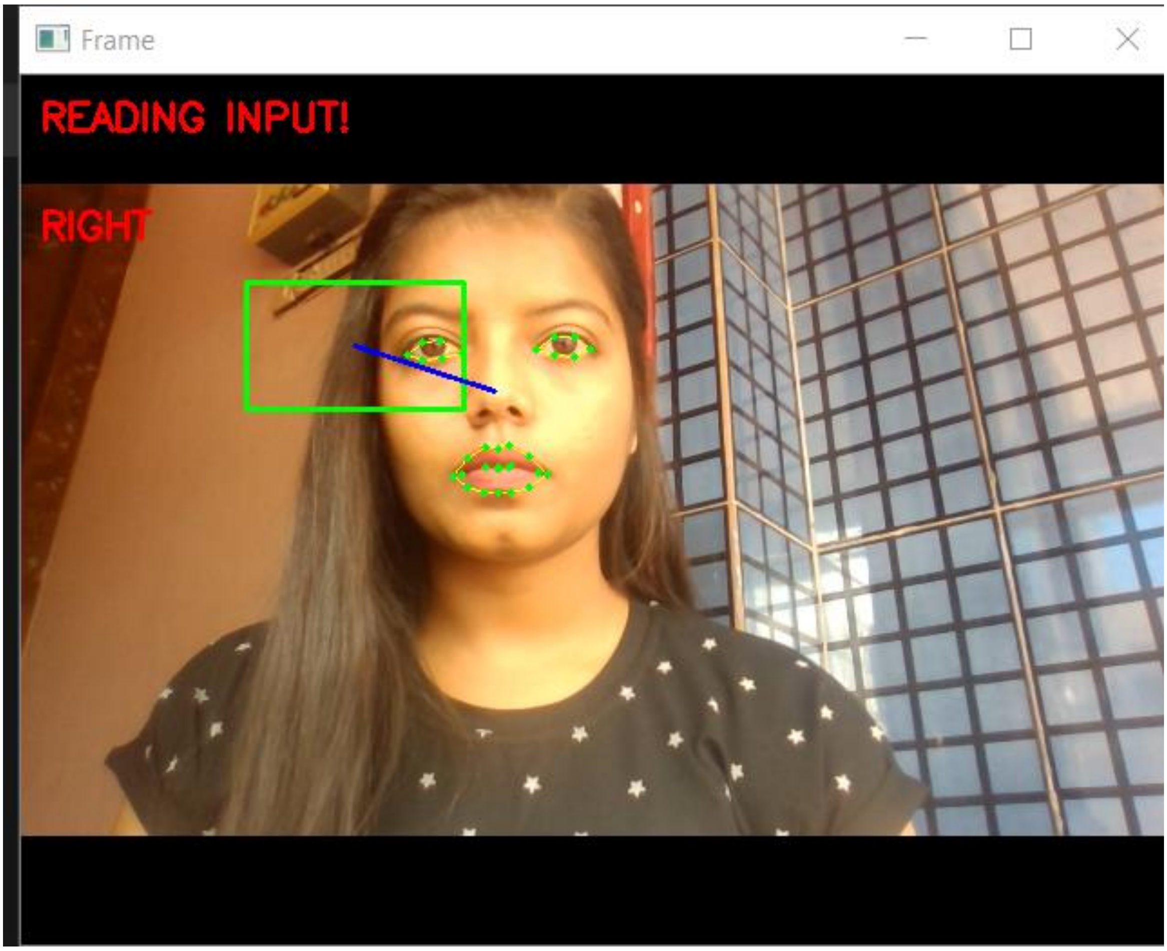
Command Prompt - python mouse-cursor-control.py

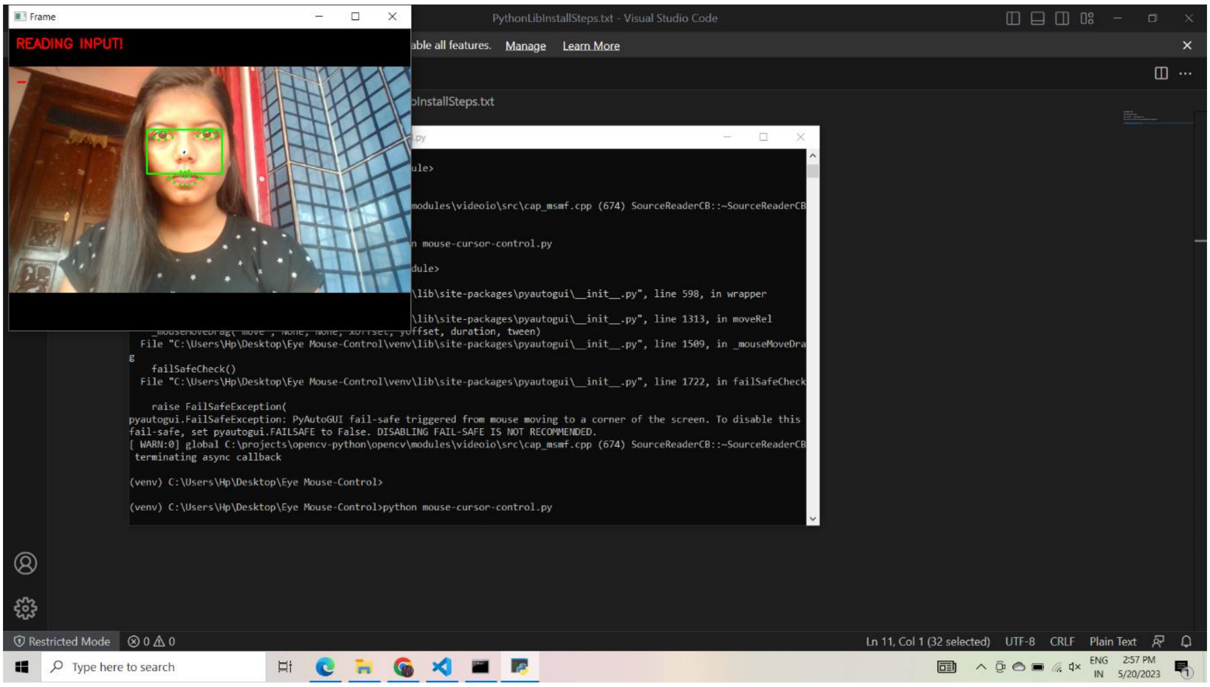
```
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\JP1>cd C:\Users\JP1\Desktop\Eye Mouse-Control

C:\Users\JP1\Desktop\Eye Mouse-Control>python mouse-cursor-control.py
```



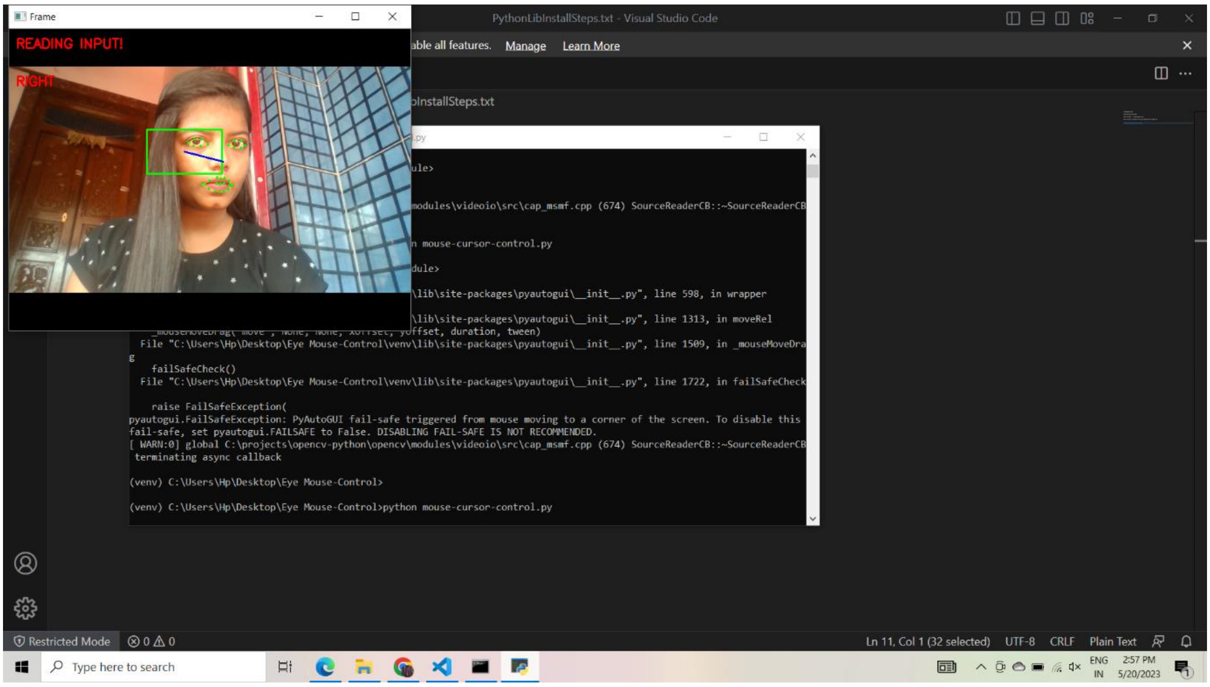

Frame
PythonLibInstallSteps.txt - Visual Studio Code
able all features. Manage Learn More
bInstallSteps.txt
.py
Restricted Mode
Ln 11, Col 1 (32 selected) UTF-8 CRLF Plain Text
Type here to search
ENG IN 2:57 PM 5/20/2023

```
ule>

modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB

n mouse-cursor-control.py

dule>

\lib\site-packages\pyautogui\__init__.py", line 598, in wrapper

\lib\site-packages\pyautogui\__init__.py", line 1313, in moveRel
    _mouseMoveDrag("move", None, None, xOffset, yoffset, duration, tween)
  File "C:\Users\Hp\Desktop\Eye Mouse-Control\venv\lib\site-packages\pyautogui\__init__.py", line 1509, in _mouseMoveDra
g
    failSafeCheck()
  File "C:\Users\Hp\Desktop\Eye Mouse-Control\venv\lib\site-packages\pyautogui\__init__.py", line 1722, in failSafeCheck

    raise FailSafeException(
pyautogui.FailSafeException: PyAutoGUI fail-safe triggered from mouse moving to a corner of the screen. To disable this
fail-safe, set pyautogui.FAILSAFE to False. DISABLING FAIL-SAFE IS NOT RECOMMENDED.
[ WARN:0] global C:\projects\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB
 terminating async callback

(venv) C:\Users\Hp\Desktop\Eye Mouse-Control>

(venv) C:\Users\Hp\Desktop\Eye Mouse-Control>python mouse-cursor-control.py
```





```
ule>

modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB

n mouse-cursor-control.py

dule>

\lib\site-packages\pyautogui\__init__.py", line 598, in wrapper

\lib\site-packages\pyautogui\__init__.py", line 1313, in moveRel
    _mouseMoveDrag("move", None, None, xOffset, yoffset, duration, tween)
  File "C:\Users\Hp\Desktop\Eye Mouse-Control\venv\lib\site-packages\pyautogui\__init__.py", line 1509, in _mouseMoveDra
g
    failSafeCheck()
  File "C:\Users\Hp\Desktop\Eye Mouse-Control\venv\lib\site-packages\pyautogui\__init__.py", line 1722, in failSafeCheck

    raise FailSafeException(
pyautogui.FailSafeException: PyAutoGUI fail-safe triggered from mouse moving to a corner of the screen. To disable this
fail-safe, set pyautogui.FAILSAFE to False. DISABLING FAIL-SAFE IS NOT RECOMMENDED.
[ WARN:0] global C:\projects\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB
 terminating async callback

(venv) C:\Users\Hp\Desktop\Eye Mouse-Control>

(venv) C:\Users\Hp\Desktop\Eye Mouse-Control>python mouse-cursor-control.py
```



```
ule>

modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB

n mouse-cursor-control.py

dule>

\lib\site-packages\pyautogui\__init__.py", line 598, in wrapper

\lib\site-packages\pyautogui\__init__.py", line 1313, in moveRel
    _mouseMoveDrag("move", None, None, xOffset, yoffset, duration, tween)
  File "C:\Users\Hp\Desktop\Eye Mouse-Control\venv\lib\site-packages\pyautogui\__init__.py", line 1509, in _mouseMoveDra
g
    failSafeCheck()
  File "C:\Users\Hp\Desktop\Eye Mouse-Control\venv\lib\site-packages\pyautogui\__init__.py", line 1722, in failSafeCheck

    raise FailSafeException(
pyautogui.FailSafeException: PyAutoGUI fail-safe triggered from mouse moving to a corner of the screen. To disable this
fail-safe, set pyautogui.FAILSAFE to False. DISABLING FAIL-SAFE IS NOT RECOMMENDED.
[ WARN:0] global C:\projects\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (674) SourceReaderCB::~SourceReaderCB
 terminating async callback

(venv) C:\Users\Hp\Desktop\Eye Mouse-Control>

(venv) C:\Users\Hp\Desktop\Eye Mouse-Control>python mouse-cursor-control.py
```

REFERENCES

[1] Q. Sun, J. Xia, N. Nadarajah, T. Falkmer, J. Foster, and H.Lee, “Assessing drivers’ visual-motor coordination using eye tracking, GNSS and GIS: a spatial turn in driving psychology,” Journal of Spatial Science, vol. 61, no. 2, pp. 299–316, 2016.

[2] N. Scott, C. Green, and S. Fairley, “Investigation of the use of eye tracking to examine tourism advertising effectiveness,” Current Issues in Tourism, vol. 19, no. 7, pp. 634–642, 2016.

[3] K. Takemura, K. Takahashi, J. Takamatsu, and T. Ogasawara, “Estimating 3-D point-of-regard in a real environment using a head-mounted eye-tracking system,” IEEE Transactions on Human-Machine Systems, vol. 44, no. 4, pp. 531–536, 2014.

[4] R. J. K. Jacob and K. S. Karn, “Eye Tracking in humancomputer interaction and usability research: ready to deliver the promises,” Minds Eye, vol. 2, no. 3, pp. 573–605, 2003.

[5] O. Ferhat and F. Vilarino, “Low cost eye tracking: the current panorama,” Computational Intelligence and Neuroscience, vol. 2016, Article ID 8680541, pp. 1–14, 2016.

[6] Tobii EyeX, “EyeX,” 2014, http://www.tobii.com/eyex.

[7] GazePoint, “Gazept,” 2013, http://www.gazept.com/category/gp3-eye-tracker.

[8] The eyeTribe, “EyeTribe,” 2014, http://www.theeyetribe.com.

[9] M. A. Eid, N. Giakoumidis, and A. El Saddik, “A novel eyegaze-controlled wheelchair system for navigating unknown environments: case study with a person with ALS,” IEEE Access, vol. 4, pp. 558–573, 2016.

[10] L. Sun, Z. Liu, and M.-T. Sun, “Real time gaze estimation with a consumer depth camera,” Information Sciences, vol. 320, pp. 346–360, 2015.

[11] Y.-M. Cheung and Q. Peng, “Eye gaze tracking with a web camera in a desktop environment,” IEEE Transactions on HumanMachine Systems, vol. 45, no. 4, pp. 419–430, 2015.

[12] P. S. Holzman, L. R. Proctor, and D. W. Hughes, “Eye-tracking patterns in schizophrenia,” Science, vol. 181, no. 4095, pp. 179–181, 1973.

[13] C. Donaghy, M. J. Thurtell, E. P. Pioro, J. M. Gibson, and R. J. Leigh, “Eye movements in amyotrophic lateral sclerosis and its mimics: a review with illustrative cases,” Journal of Neurology, Neurosurgery & Psychiatry, vol. 82, no. 1, pp. 110–116, 2011.

[14] M. Nehete, M. Lokhande, and K. Ahire, “Design an eye tracking mouse,” International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, no. 2, 2013.

[15] E. Missimer and M. Betke, “Blink and wink detection for mouse pointer control,” in Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments (PETRA ’10), Samos, Greece, June 2010.

[16] GitHub, “MasterLomaster/bkb,” 2015, https://github.com/MastaLomaster/bkb.

[17] H. Singh and J. Singh, “Human eye tracking and related issues: a review,” International Journal of Scientific and Research Publication, vol. 2, no. 9, pp. 146–154, 2012.

[18] C. A. Chin, A. Barreto, J. G. Cremades, and M. Adjouadi, “Integrated electromyogram and eye-gaze tracking cursor control system for computer users with motor disabilities,” Journal of Rehabilitation Research and Development , vol. 45, no. 1, pp. 161–174, 2008.

[19] R. G. Lupu, F. Ungureanu, and R. G. Bozomitu, “Mobile embedded system for human computer communication in assistive technology,” in Proceedings of the 2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing (ICCP ’12), pp. 209–212, Cluj-Napoca, Romania, September 2012.

[20] R. G. Lupu, F. Ungureanu, and V. Siriteanu, "Eye tracking mouse for human computer interaction," in Proceedings of the 4th IEEE International Conference on E-Health and Bioengineering (EHB '13), Iasi, Romania, November 2013.

[21] S. Wankhede and S. Chhabria, "Controlling Mouse Cursor Using Eye Movement," International Journal of Application or Innovation in Engineering & Management, no. 36, pp. 1–7, 2013.

[22] S. M. A. Meghna, K. L. Kachan, and A. Baviskar, "Head tracking virtual mouse system based on ad boost face detection algorithm," International Journal on Recent and Innovation Trends in Computing and Communication, vol. 4, no. 4, pp. 921–923, 2016.

[23] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," MIS Quarterly, vol. 13, no. 3, pp. 319–339, 1989.

[24] F. D. Davis, R. P. Bagozzi, and P. R.Warshaw, "User acceptance of computer technology: a comparison of two theoretical models," Management Science, vol. 35, no. 8, pp. 982–1003, 1989.

[25] V. Venkatesh and F. D. Davis, "A theoretical extension of the technology acceptance model: four longitudinal field studies," Management Science, vol. 46, no. 2, pp. 186–204, 2000.

[26] T. S. H. Teo, V. K. G. Lim, and R. Y. C. Lai, "Intrinsic and extrinsic motivation in Internet usage," Omega , vol. 27, no. 1, pp. 25–37, 1999.

[27] V. Venkatesh, C. Speier, and M. G. Morris, "User acceptance enablers in individual decision making about technology: towards an integrated model," Decision Sciences, vol. 33, no. 2, pp. 297–316, 2002.

[28] A. Bangor, K. Philip, and M. James, "Determining what individual SUS scores mean: adding an adjective rating scale," Journal of Usability Studies, vol. 4, no. 3, pp. 114–123, 2009