

FACIAL GESTURE BASED HUMAN MACHINES INTERACTION FOR PHYSICALLY DISABLED PEOPLE

A PROJECT REPORT

Submitted by

DINESH KUMAR BJ

ANANTHKUMAR R

AKSHAY D

in partial fulfilment for the award of degree

Of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY



RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI-602 105

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2020

FACIAL GESTURES BASED HUMAN MACHINES INTERACTION FOR PHYSICALLY DISABLED PEOPLES

A PROJECT REPORT

Submitted by

DINESH KUMAR BJ

ANANTHKUMAR R

AKSHAY D

in partial fulfilment for the award of degree

Of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY



RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI-602 105

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2020

ANNA UNIVERSITY:CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “FACIAL GESTURES HUMAN MACHINES BASED INTERACTION FOR PHYSICALLY DISABLED PEOPLES” is the bonafide work of

DINESH KUMAR B.J(211616205031), ANANTHKUMAR R(211616205009)

AKSHAY D(211616205007) Who carried out the project work

My supervision.

SIGNATURE

Dr.L.Priya

Head of the Department

Department of Information

Technology

Rajalakshmi Engineering College

Thandalam,Chennai-602 105

SIGNATURE

Dr.Baghavathi Priya

SUPERVISOR

Professor

Department of Information

Technology

Rajalakshmi Engineering College

Thandalam,Chennai-602 105

Submitted for University Examination conducted on

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our sincere thanks to our respected chairman, **Dr.Thangam meghanathan.**,for providing us with with requisite infrastructure throughout the course.

We would like to express our sincere thanks to our respected principal **Dr.S.N.Murugesan.**,for his encouragement.We are highly indebted to the department of Information Technology for providing all the facilities for sucessful completion of the product.

With a deep sense of gratitude,we would like to sincerely thank our Head of the Department **Dr.L.Priya.**,for her constant support,encouragement and valuable guidance for my project work

We are extremely grateful to our project Coordinator **Dr.S.Baghavathi Priya.**,Professor,Department of Information Technology for her consent guidance,suggestion and kind help in bringing out this project within scheduled time frame

We would like to thank our project supervisor **Mrs.S.Sree Subha.**,Associate Professor,Department of Information Technology for the valuable suggestions and encouragement.

We would like to thank all our teaching and non teaching staffs for their kind cooperation throughout this project work

ABSTRACT

As there is great advancement in the technology in recent years, there has been much improvement in various fields of computing such as Human Computer Interaction (HCI), Computer Vision. Input to the computers has sensed information about physical properties of user, places, or things. For example, computer mouse and keyboard operates by motion imparted by user's hand. All these techniques may not be suitable to physically disabled people. A way to create an application which replaces the input devices such as mouse and keyboard by using face of the user is proposed. This project introduces how head motion of user can be used to control the mouse cursor and how gaze tracking can be used to control the keyboard. A face detecting system precisely records the motion parameters from video at real-time using a typical webcam. While the speed decreases when controlling the virtual keyboard and mouse, the system's performance remains functioning for severely disabled people who have their gaze and head movements as one of their only means of communication.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO.
	ABSTRACT	ii
	LIST OF FIGURES	vii
	LIST OF TABLES	ix
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	
	1.1 Introduction	
	1.2 Overview	
	1.3 Scope	
2	LITERATURE SURVEY	
	2.1 Introduction	
	2.2 Related Work	
	2.3 Conclusion	
3	PROPOSED WORK	
	3.1 Recommended System	
	3.2 Proposed Architecture Diagram	
	3.2.1 Face Detection	
	3.2.2 Feature Tracking	
	3.2.3 EAR,MAR and Dwell Time Calculation	
	3.2.4. Mouse and Key Actions	
4	SYSTEM REQUIREMENTS	
	4.1 Hardware Requirements	
	4.2 Software Requirements	

4.3 Description of tools used

4.3.1 Python

4.3.2 Numpy

4.3.3 Open CV

4.3.4 PyAutoGUI

4.3.5 Dlib

4.3.6 Imutils

4.3.7 Pandas

4.3.8 Scikit Learn

5 UML AND DFD DIAGRAM

5.1 UML DIAGRAM

5.1.1 Use Case Diagram

5.1.2 Sequence Diagram

5.1.3 Activity Diagram

5.2 DFD Diagram

6 IMPLEMENTATION MODULES

6.1 Face Detection

6.2 Feature Tracking

6.3 EAR, MAR and Dwell Time Calculation

6.4 Mouse and Keyboard Actions

7 SNAPSHOT

7.1 Detecting Facial Landmarks

7.2 Activating Mouse Cursor Controller

7.3 Scroll Mode Controller

7.4 Virtual keyboard Layout

7.5 Character Selection in Virtual Keyboard Layout

8 CONCLUSION AND FUTURE WORK

8.1 Conclusion

8.2 Future Work

9 APPENDIXES

9.1 Calibration.py

9.2 Eye.py

9.3 Gaze_tracking.py

9.4 Main_demo.py

9.5 Page.py

9.6 Pupil.py

10 REFERENCES

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.2	Proposed Architecture Diagram	
3.2.1	Hard Negative Mining	
3.2.1.2	Bounding Box and Non Max Supression	
4.1	Predicting 68.2D.Landmarks using Dlib	
6.1	Frame Filter	
6.2	Face Detection	
6.3.1	Eye Aspect Ratio	
6.3.2	Mouth Aspect Ratio	
6.4	Virtual Keyboard Layout	
7.1	Detecting Facial Landmarks	
7.2	Activating Mouse Cursor Controller RIGHT	
7.3	Reading Input:Right Left	
7.4	Reading Input:Left Up	
7.5	Reading Input:Up Down	
7.6	Reading Input:Down	
7.7	Right click	
7.8	Left Click	
7.9	Activating Scroll Mode	
7.10	Virtual Keyboard Layout	
7.11	Reading Input:Up Left CENTRE	
7.12	Up.Centre.RIGHT	

- 7.13 Up.Right
- 7.14 Down Left:CENTRE
- 7.15 Reading Input:Down Centre RIGHT
- 7.16 Reading Input:Down Right
- 7.17 Character Selection Virtual Keyboard

LIST OF TABLES

TABLE NO:	TITLE	PAGE NO
4.1	Hardware Requirements	
4.2	Software Requirements	
4.3	Mouse Movements	

LIST OF ABBREVIATIONS

HCI - Human Computer Interface

PUI - Perceptual User Interface

HOG - Histogram Oriented Gradient

SVM - Support Vector Machine

EEG - Electro Encephalography

EOG - Electro Oculogram

GUI - Graphical User Interface

GPU - Graphical Processing Unit

CV - Computer Vision

CL - Computing Language

EAR - Eye Aspect Ratio

MAR - Mouth Aspect Ratio

PDA - Personal Digital Assistant

GPS - Global Positioning System

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The primary application of this project is to provide an alternative input device for differently abled people by using the head and eye movements of the user. Recent development in the field of Computer Vision, Human Computer Interface (HCI), Gesture Recognition and Perceptual User Interface (PUI) has provided a way to make human life better. Nowadays various possibilities for interacting with computing device are tried, instead of our traditional input devices, like keyboard and mouse. Today computers are becoming faster and more reliable, so it is possible to introduce an alternative solution to control computers. This can be possible by making computers to capture the human's movements or perceptual behaviour. It can be done by tracking down the movement made by human body by live video feed, which computer understands and interprets it into action or response. This technology is referred as perceptual user interface technology.

Computer vision focuses on assisting the computers to identify objects by seeing. Computer vision's goal is to make use of observed image data to infer something useful and meaningful. It is a discipline which can be said as the sub-field of machine learning and AI that makes

use of general learning algorithms and exclusive techniques. Computer Vision tries to make sense of digital images. This includes developing techniques to replicate the potential of vision of humans. To understand digital images, we must be able to obtain information from an image, that can be an entity, a 3D model, a textual explanation etc. This is a very demanding work. We're very farther away from creating a versatile "seeing machine".

1.2 OVERVIEW

The HCI and PUI involves interaction with humans, so the computing devices needs to be efficient, reliable, secure and should provide fast performance to withstand various challenges. The most important part is the tracking of user's movement with a traditional normal webcam which records the user's movement and convert them to computer operations. Thus it provides a simple alternative way to control computing devices. It will also allow physically disabled people to access computer for various purposes. Thus it serves as assistive device to help disabled people. Also this system does not need to wear or use reflectors, or infrared devices, markers, etc. This may be problem to some users. Apart from helping disabled, it can also be used by general users for many things like interactive computer games; machine guided gymnastic, robot control, simulation. The progress in the field of face detection and tracking and improvement over the system hardware and software methodologies have made it possible to create a fast, reliable performing tracking system on computing devices. Hands-free control for human

computer interaction has become extremely crucial nowadays. For a solution, using human body movement in real time video input and utilize the tracking is an important kind of solution.

1.3 SCOPE

The project is primarily for used by physically challenged people. It can also be used in interactive computer games; machine guided gymnastic, robot control and simulation. It serves as a tool to provide effective communication between humans and the computers. This can be used to avoid the eye damage caused by hardware devices. However, complete automation of the system is not possible yet.

- Target: It is primarily for the differently abled people who are paralyzed or handicapped living with a caretaker and look forward to provide the necessary assistance.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

A literature review is a [scholarly paper](#) that presents the current knowledge including substantive findings as well as theoretical and methodological contributions to a particular topic. Literature reviews are [secondary sources](#) and do not report new or original experimental work. Most often associated with academic-oriented literature, such reviews are found in [academic journals](#) and are not to be confused with [book reviews](#), which may also appear in the same publication. Literature reviews are a basis for research in nearly every academic field. A narrow-scope literature review may be included as part of a [peer-reviewed](#) journal article presenting new research, serving to situate the current study within the body of the relevant literature and to provide context for the reader. In such a case, the review usually precedes the methodology and results sections of the work.

Literature reviews are also common in a [research proposal](#) or prospectus (the document that is approved before a student formally begins a dissertation or thesis).^[1]

2.2 RELATED WORK

Dutta, B. Bhushan, G. Prasad, H. Cecotti, Y. K. Meena (2019) state that the recent development of inexpensive and accurate eye-trackers allows the creation of gazed based virtual keyboards that can be used by a large population of disabled people in developing countries. Due to eye-tracking technology, gaze-based virtual keyboards can be designed in relation to constraints related to the gaze detection accuracy and the considered display device. A new multimodal multiscript gaze-based virtual keyboard where it is possible to change the layout of the graphical user interface in relation to the script is proposed. Traditionally, virtual keyboards are assessed for a single language (e.g. English). A multiscript gaze based virtual keyboard that can be accessed for people who communicate with the Latin, Bangla, and/or Devanagari scripts is made possible. The performance of the virtual keyboard with two main groups of participants: 28 people who can communicate with both Bangla and English, and 24 people who can communicate with both Devanagari and English is evaluated. The performance is assessed in relation to the information transfer rate when participants had to spell a sentence using their gaze for pointing to the command, and a dedicated mouth switch for commands selection. The results support the conclusion that the system is efficient, with no difference in terms of information transfer rate between Bangla and Devanagari.

Ashlesha Singh, Chandrakant Chandewar, Pranav Pattarkine (2018) state that in haar cascade classifier primarily the haar structures are slide over one by one on an image, throughout the pixel values masked in black portion are added similarly all the pixel values overlaid in the white part are added, finally the sum values are compared and accordingly a threshold value is determined. Image descriptor, Histogram of Oriented Gradient (HOG) along with Linear Support

Vector Machine (SVM) is used to set up highly accurate object classifiers. At first feature matrix is extracted using HOG descriptor and then these features are used to train SVM classifier. HOG uses merits of both multi-class and bi-class HOG based detectors to build three stage algorithms with low computational cost. In the first stage, the multi-class classifier with coarse features is used to estimate the orientation of a potential target object in the image; in the second stage, a biclass detector corresponding to the detected orientation with intermediate level features is used to filter out most of false positives; and in the third stage, a bi-class detector corresponding to the detected orientation using fine features is used to achieve accurate detection with low rate of false positives. In this way, features are extracted from an image. After the features are extracted, they are fed to linear SVM algorithm for classification.

Chairat Kraichan, Suree Pumrin (2014) state that in most desktop use scenarios the majority of head motion occurs in the horizontal plane. 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. The robustness of the system at different roll and pitch angles of head pose for detecting face and eye center is examined. Finally, the system maps eye-center coordinates in X and Y axis from (world) camera coordinates to computer screen and creates computer mouse control based on face and eye tracking.

Farida Mohamed, Haya Ansari, Maryam Mohamed, Mohamed Nasor and Mujeeb Rahman (2007) state that many researchers have tried to develop methods to help the disabled to interact with computers by using signals such as electroencephalography (EEG) from the brain, facial muscles signals (EMG) and electro-oculogram (EOG). Other methods include limbus, pupil and eye/eyelid tracking, contact lens method, corneal, pupil reflection relationship and head movement measurement. Other high end techniques that are based on infrared tracking of the eye movements to control computers were exceptionally expensive and were not affordable for those who need them. It does 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. 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 image of one eye was then extracted and normalized in order to remove the background noises and then it was converted to binary image to enhance contrast. The normalized pixel (0 – 1 scale) values.

H. Cecotti (2016) states that the layout of a regular keyboard can be an obstacle for its use with an eye-tracker because the proximity of the buttons increases the possible confusion of the gaze detection procedure. Virtual keyboards are difficult to evaluate as the performance depends on the subjects who use the system, their motivation and experience, and the amount of text that is spelled-out during the experiments. Extended menu selection with digits to write numbers and additional items increases the possibilities of the system.

Josephine Sullivan, Vahid Kazemi (2014) state that an ensemble of regression trees can be used to estimate the face's landmark positions directly from a sparse subset of pixel intensities, achieving super-realtime performance with high quality predictions. We present a general framework based on gradient boosting for learning an ensemble of regression trees that optimizes the sum of square error loss and naturally handles missing or partially labelled data. We show how using appropriate priors exploiting the structure of image data helps with efficient feature selection. Different regularization strategies and its importance to combat over fitting are also investigated. In addition, we analyse the effect of the quantity of training data on the accuracy of the predictions and explore the effect of data augmentation using synthesized data.

Mohamad Hassrol Bin Mat, Rasheed Nassr, Sara Bilal (2018) state that gaze is not as accurate as the computer mouse. Inaccuracy originates partly from technological reasons and partly from features of the eye

The size of the fovea and the inability of the remote camera to resolve the fovea position restrict the accuracy of the measured point of gaze to about 0.5 degrees, equivalent to a region spanning approximately 15 pixels on a typical display (17 inch display with a resolution of $1,024 \times 768$ pixels viewed from a distance of 70 cm). One problem is drifting; even if a newly calibrated eye tracking device is accurate at first, with continued use the measured point of gaze drifts away from the actual point of gaze. This is partly due to the technology and partly due to the interaction between head movement and eye movement. Therefore, the practical accuracy is about 0.5–1 degree of drifting corresponds to about 1–1.5 cm on the screen at a normal viewing distance. After the eye pupil is tracked, the system will take control of the mouse's cursor movement. Then the eye's gaze will become the pointing device and the selection of the focused key or button will be achieved by using dwell-time of 0.5seconds. The developed eye tacker is an affordable device and it has retained a good typing accuracy either by using the PS3 or the laptop camera. Hence, it can be used for several applications and especially for research purposes.

R. Sigit, T. Harsono, V. I. Saraswati (2016) state that the user's face is captured by a camera, then the camera does some feature detection of face to get position of eye gaze, and this position will become a reference of pointer position to choose some word in virtual keyboard. HaarCascade method was used to detect some feature of face, and Integral Projection method was used to get position of the eye movement. Based on the results of the experiment, the comparison ratio between duration of normal writing and typing using system for two words is 1:13 second.

Wang Jun, Wen Jing, Zhang Naizhong (2015) state that human mouth contains very strongly independent feature to the other region in face image, even in profile side of face image, and in face region, it always have a corresponsive and static position when human rotate or shake head. It selects center point of mouth as reference point to estimate face mouth rotation angle in X-axis and Y-axis. When face mouth rotated from one position to another (For Example: from frontal face to left 45 degree side in x direction), in the corresponding captured image, the mouth position also moved from horizontal center to a corresponding position linearly. So linear relationship between mouth position and face boundary to calculate the angle can be utilized.

Qurban Ali Memon, Zeenat Al-Kassim (2017) state that the keyboard works on detection and tracking of the user eyeball movements. The design of the system is done in phases; these are eye detection and tracking, followed by classification. The tracking phase incorporates skin color segmentation, black pupil detection and support vector machines. The role of eye red-channel, eye width and height for eye tracking is also explored. A scanning keyboard is developed and tested to work with three eyeball movements with 90% double detection accuracy. The object tracking (both moving and stationary) is not a new field, and applications exist in various disciplines like engineering and health care. Conventionally, technology has been exploited to build systems that try to simplify human life.

CHAPTER 3

PROPOSED WORK

3.1 RECOMMENDED SYSTEM

The objective of this project is to implement the mouse cursor movements and the eye tracking keyboard using the head movements and gaze tracking to help differently abled people. The system includes mechanisms to detect face, track the features of the face, extract eye blinking and mouth movements and to capture the keystrokes. Initially, a typical web camera is set to monitor the recorded image containing the object which is a face. For accurate and rapid image processing, OpenCV library is used. Once a face has been detected, 68 facial landmarks are detected, following which the eye aspect ratio and mouth aspect ratio are determined in order to recognize whether the eyes are open and whether the mouth is open respectively. The mouse cursor controller gets activated and deactivated by the opening of the mouth. Upon blinking the left eye alone, the cursor performs left click operation and upon blinking the right eye alone, the cursor performs right click operation. Furthermore, upon turning the face to the left, the cursor moves to the left and upon moving the face to the right, the cursor moves to the right, upon moving the head up, the cursor moves up and upon moving the head down, the cursor moves down. Upon squinting the eyes for a particular amount of time, the scroll mode is activated. The face detection and image processing for the keyboard is done in the same way as the mouse

cursor controller. Since the traditional keyboard layout is inefficient, a tree selection method is used for the eye tracking keyboard. The keyboard consists of selection of letters, numbers, commonly used sentences and shortcuts to help disabled people in times of emergency.

3.2 PROPOSED ARCHITECTURE DIAGRAM

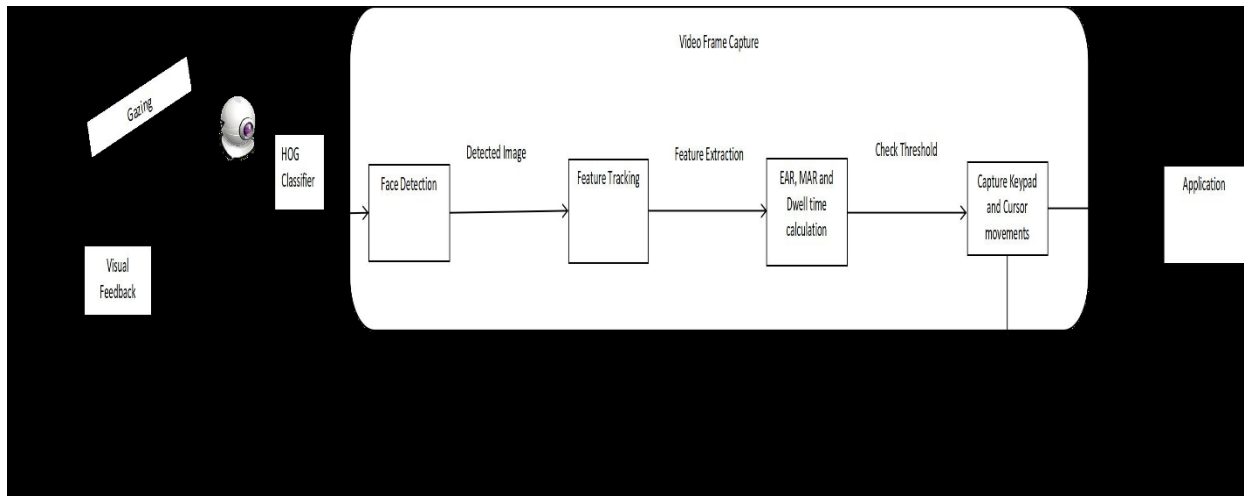


Figure 3.2 Proposed Architecture Diagram

The following are the modules involved in the model:

- Face Detection
- Feature Tracking
- EAR, MAR and Dwell Time Calculation
- Mouse and Keyboard Actions

3.2.1 Face Detection

Initially, a typical web camera is set to monitor the recorded image containing the object which is a face. For accurate and rapid image processing, OpenCV library is used. All the pixels of the image recognized as a face is categorized into one of the types, namely skin and non skin pixels. To examine if the detected area is face or not face, connectivity analysis is used to analyse the established skin area.

General algorithm for training an object detection using Histogram of Oriented Gradients.

Step 1: Sample P positive samples from your training data of the object(s) you want to detect and extract HOG descriptors from these samples.

Step 2: Sample N negative samples from a negative training set that does not contain any of the objects you want to detect and extract HOG description from these samples as well. In practice $N \gg P$.

Step 3: Train a Linear Support Vector Machine on your positive and negative samples.

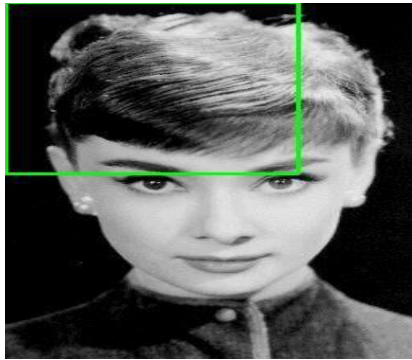


Figure 3.2.1.1 Hard Negative Mining

Apply hard-negative mining for each image and each possible scale of each image in your negative training set, apply the sliding window technique and slide your window across the image. At each window compute your HOG descriptors and apply your classifier. If your classifier (incorrectly) classifies a given window as an object (and it will, there will absolutely be false-positives), record the feature vector associated with the false-positive patch along with the probability of the classification. This approach is called hard-negative mining.

Step 5: Take the false-positive samples found during the hard-negative mining stage, sort them by their confidence (i.e. probability) and re-train your classifier using these hard-negative samples.

Step 6: Your classifier is now trained and can be applied to your test dataset. Again, just like in Step 4, for each image in your test set, and for each scale of the image, apply the sliding window technique. At each window extract HOG descriptors and apply your classifier. If your classifier detects an object with sufficiently large probability

record the bounding box of the window. After you have finished scanning the image, apply non-maximum suppression to remove redundant and overlapping bounding boxes.

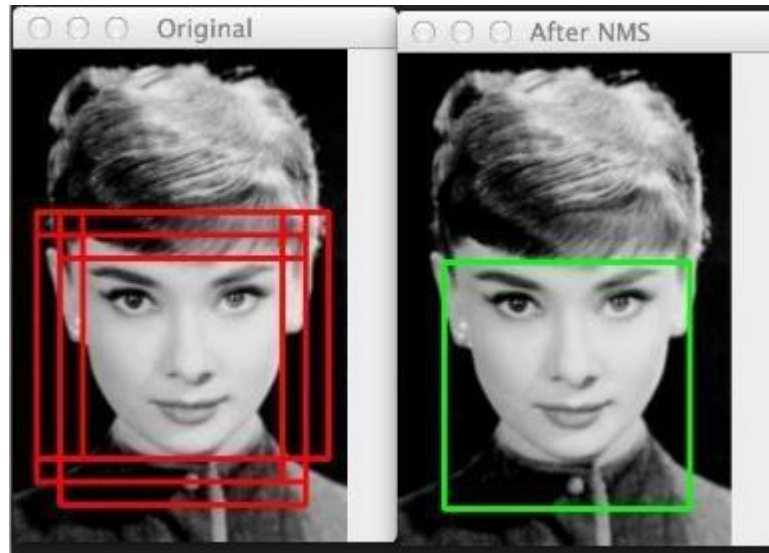


Figure 3.2.1.2 Bounding box and Non-Maximum Suppression

3.2.2 Feature Tracking

If a face is identified, dlib is used to map the facial features' landmarks. Facial landmark is an inbuilt HOG SVM classifier which is used to determine the position of 68(x,y) co-ordinates which maps the facial structure. The indices of the 68 co-ordinates are marked.

3.2.3 EAR, MAR and Dwell Time Calculation

Once the landmarks of eye and mouth are determined, EAR, MAR and Dwell Time are computed. The mouse events are performed if in case EAR and MAR values reaches the specified threshold value and the keyboard keys are selected if the gaze exceeds the dwell time.

3.2.4 Mouse and Keyboard Actions

The cursor movements are controlled by the head of the user. The scroll mode is activated by the squinting of the eyes. The right wink and left wink actions are used to perform right click and left click actions. A visual feedback is continuously provided regarding the position of the mouse. Basically the GUI of the virtual keyboard is made up of two components - First, the center of the screen where the selected text is displayed and second is the edges of the screen where the character sets are displayed. Since the traditional keyboard layout is inefficient, a tree selection method is used with 6 sections. Both mouse cursor controller and the eye tracking keyboard work well in sub optimal lighting conditions. People wearing spectacles could also use the system with ease. Since an in-built webcam is used rather than an external webcam, the proposed system is cost effective.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

PROCESSOR : Intel Core i7 8th Gen

HARD DISK DRIVE : 1TB

RAM : 16GB

GPU : NVIDIA GEFORCE GTX 1050 Ti

4.2 SOFTWARE REQUIREMENTS

OPERATING SYSTEM : PROGRAMMING	Microsoft Windows 10
LANGUAGE :	Python 3.6
IDE :	Python 3.6 (64-bit)

4.3 DESCRIPTION OF THE TOOLS USED

The project is mainly implemented in Python using several packages. The following are the tools used:

- Python
- Numpy
- OpenCV
- PyAutoGUI
- dlib
- Imutils
- Pandas
- Scikit-learn

4.3.1 Python

Python is an interpreted, high-level, general-purpose programming language. Python has a design philosophy that emphasizes code readability, notably using significant white space. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, it also has a comprehensive standard library.

Python is considered a scripting language, like [Ruby](#) or [Perl](#) and is often used for creating Web [applications](#) and dynamic Web content. It is also supported by a number of 2D and 3D imaging programs, enabling users to create custom [plug-ins](#) and extensions with Python.

4.3.2 Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

The core functionality of NumPy is its "ndarray", for n-dimensional array, data structure. These arrays are stride views on memory. In contrast to Python's built-in list data structure (which, despite the name, is a dynamic array), these arrays are homogeneously typed: all elements of a single array must be of the same type.

4.3.3.OPEN CV

OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations.

4.3.4 PyAutoGUI

PyAutoGUI is a cross-platform GUI automation Python module for human beings. Used to programmatically control the mouse & keyboard.

PyAutoGUI only works on the primary monitor. PyAutoGUI isn't reliable for the screen of a second monitor (the mouse functions may or may not work on multi-monitor setups depending on your operating system and version). All keyboard presses done by PyAutoGUI are sent to the window that currently has focus, as if you had pressed the physical keyboard key.

4.3.5 Dlib

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments. Since development began in 2002, Dlib has grown to include a wide variety of tools. As of

2016, it contains software components for dealing with networking, threads, graphical user interfaces, data structures, linear algebra, machine learning, image processing, data mining, XML and text parsing, numerical optimization, Bayesian networks, and many other tasks.

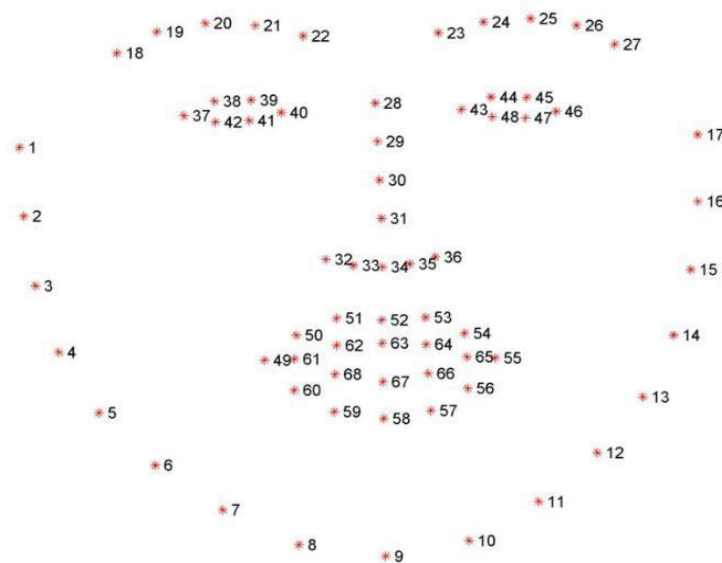


Figure 4.1 Predicting 68 2D landmarks using dlib

4.3.6 Imutils

A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

4.3.7 Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

The library features include DataFrame object for data manipulation with integrated indexing, tools for reading and writing data between in-memory data structures and different file formats, data alignment and integrated handling of missing data, reshaping and pivoting of data sets etc.

4.3.8 Scikit-learn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

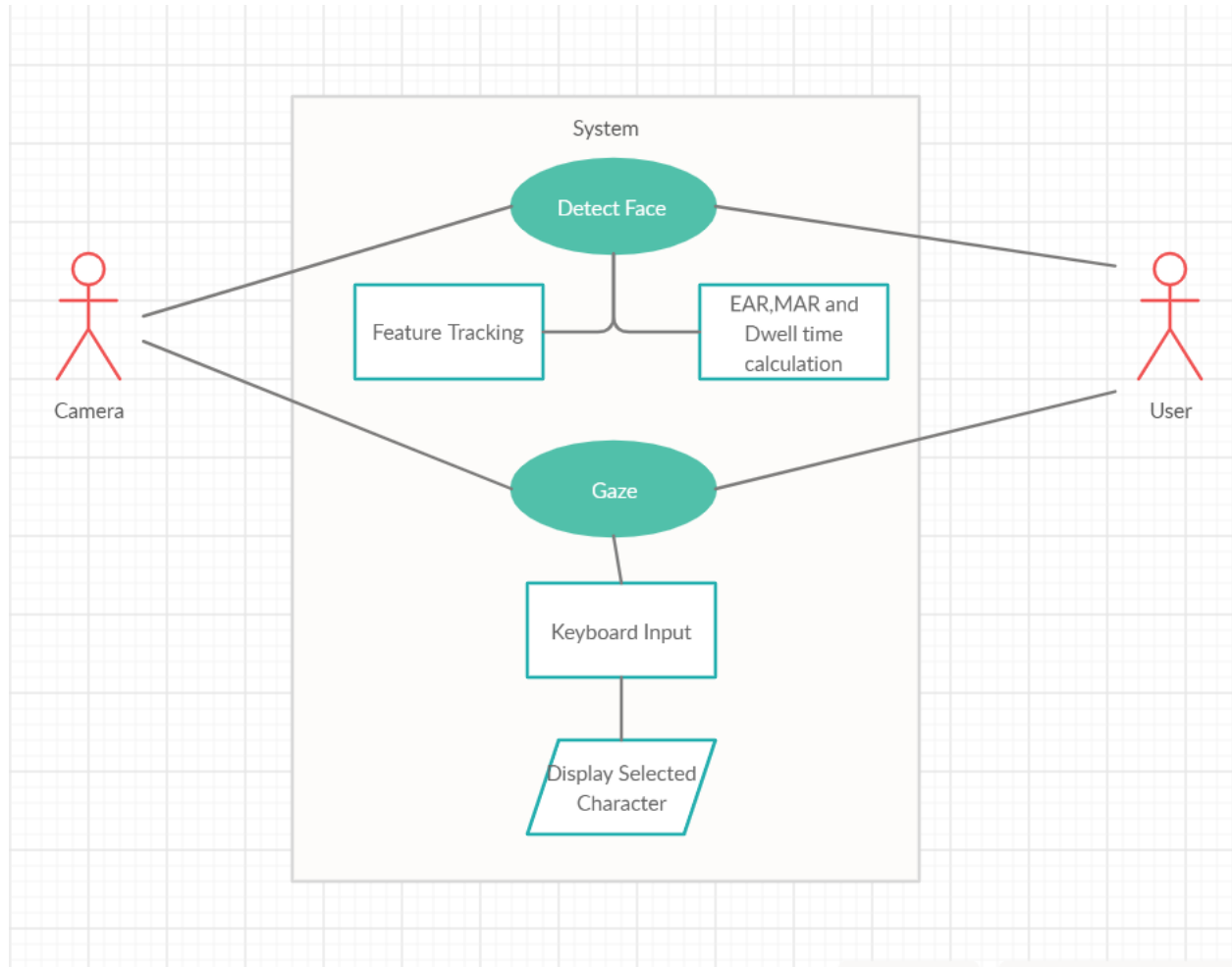
Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

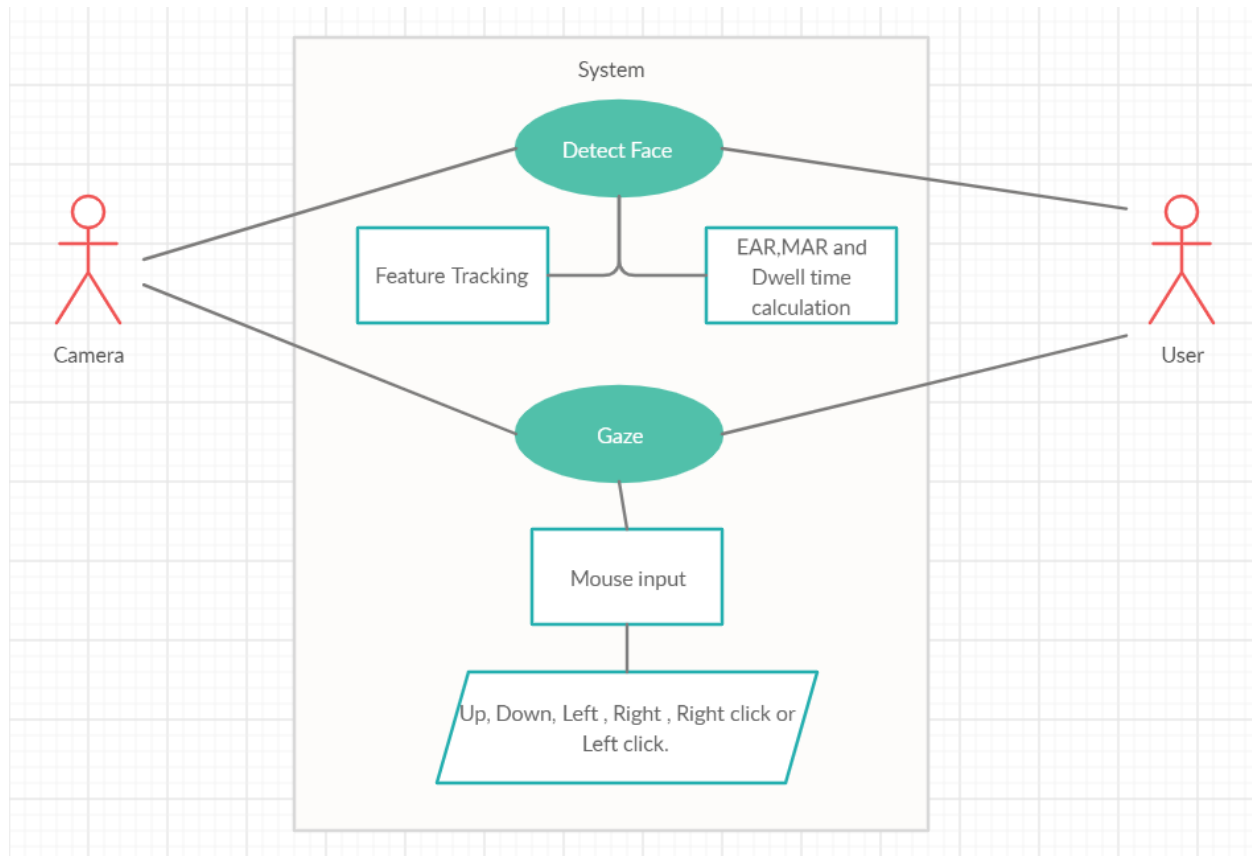
CHAPTER 5

UML AND DFD DIAGRAMS

5.1 UML DIAGRAMS

5.1.1 USE CASE DIAGRAM





CHAPTER 6

IMPLEMENTATION MODULES

Modular design defines the structure of the overall system. Modularity is a general system concept typically defined as the degree to which a system's components may be separated and recombined. The overall system consists of the following modules:

1. Face Detection
2. Feature Tracking
3. EAR, MAR and Dwell Time Calculation
4. Mouse and Keyboard Actions

6.1 FACE DETECTION

6.1.1 Accessing Computer Webcam Using OpenCV

6.1.1.1 Description

Live stream is captured with camera. OpenCV provides a very simple interface to this. Let's capture a video from the camera (in-built webcam of the laptop is used), convert it into grayscale video and display it. Just a simple task to get started.

To capture a video, you need to create a Video Capture object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Normally one camera will be connected (as in my case). So simply pass 0 (or -1). Select the second camera by passing 1 and so on. After that, capture frame-by-frame. But at the end, don't forget to release the capture. `cap.read()` returns a bool (True/False). If frame is read correctly, it will be True. So you can check end of the video by checking this return value.

Sometimes, cap may not have initialized the capture. In that case, this code shows error. Check whether it is initialized or not by the method `cap.isOpened()`. If it is True, OK. Otherwise open it using `cap.open()`.

Some of the features of video can be accessed using `cap.get(propId)` method where `propId` is a number from 0 to 18. Each number denotes a property of the video (if it is applicable to that video) and full details can be seen here: Property identifier. Some of these values can be modified using `cap.set(propId, value)`. Value is the new value you want.

For example, Check the frame width and height by `cap.get(3)` and `cap.get(4)`. It gives me 640x480 by default. But to modify it to 320x240. Just use `ret = cap.set(3,320)` and `ret = cap.set(4,240)`.

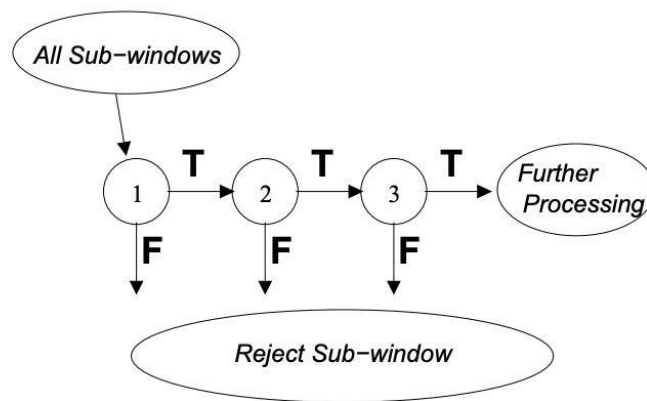


Figure 6.1 Frame Filter

6.1.1.2 Algorithm

```
import numpy as np import cv2 cap = cv2.VideoCapture(0) while(True):  
# Capture frame-by-frame ret, frame = cap.read()  
  
# Our operations on the frame come here gray = cv2.cvtColor(frame,  
cv2.COLOR_BGR2GRAY)  
  
# Display the resulting frame cv2.imshow('frame',gray)  
  
if cv2.waitKey(1) & 0xFF == ord('q'):  
  
Break
```

```
# When everything done, release the capture cap.release()  
cv2.destroyAllWindows()
```

6.1.2 Histogram of Oriented Gradients (HOG)

6.1.2.1 Description

The Histogram of Oriented Gradients (HOG) feature descriptor is popular for object detection.

Computing a Histogram of Oriented Gradients (HOG) is done by:

- (optional) global image normalisation.
- computing the gradient image in x and y.
- computing gradient histograms.
- normalising across blocks.
- flattening into a feature vector.

The first stage applies an optional global image normalisation equalisation that is designed to reduce the influence of illumination effects. In practice gamma (power law) compression is used, either computing the square root or the log of each color channel. Image texture strength is typically proportional to the local surface illumination so this compression helps to reduce the effects of local shadowing and illumination variations.

The second stage computes first order image gradients. These capture contour, silhouette and some texture information, while providing further resistance to illumination variations. The locally dominant color channel is used, which provides color invariance to a large extent. Variant methods may also include second order image derivatives, which act as primitive bar detectors - a useful feature for capturing.

The second stage computes first order image gradients. These capture contour, silhouette and some texture information, while providing further resistance to illumination variations. The locally dominant color channel is used, which provides color invariance to a large extent. Variant methods may also include second order image derivatives, which act as primitive bar detectors - a useful feature for capturing.

The fourth stage computes normalisation, which takes local groups of cells and contrast normalises their overall responses before passing to next stage. Normalisation introduces better invariance to illumination, shadowing, and edge contrast. It is performed by accumulating a measure of local histogram “energy” over local groups of cells that we call “blocks”. The result is used to normalise each cell in the block. Typically each individual cell is shared between several blocks, but its normalisations are block dependent and thus different. The cell thus appears several times in the final output vector with different normalisations. This may seem redundant but it improves the performance. We refer to the normalised block descriptors as Histogram of Oriented Gradient (HOG) descriptors.

The final step collects the HOG descriptors from all blocks of a dense overlapping grid of blocks covering the detection window into a combined feature vector for use in the window classifier.

6.1.2.2 Face Detection Algorithm

```
face_detect = dlib.get_frontal_face_detector()
rects = face_detect(gray, 1)
for (i, rect) in enumerate(rects):
    (x, y, w, h) = face_utils.rect_to_bb(rect)
    cv2.rectangle(gray, (x, y), (x + w, y + h), (255, 255, 255), 3)
plt.figure(figsize=(12,8))
plt.imshow(gray, cmap='gray')
plt.show()
```



Figure 6.2 Face Detection

6.2 FEATURE TRACKING

6.2.1 Detecting Key Facial Structures in the face region

6.2.1.1 Description

Extracting the following facial regions:

- Right Eye
- Left Eye
- Nose
- Mouth

6.2.1.2 Feature Tracking Algorithm

```
# Extract the nose, mouth, left and right eye coordinates
mouth = shape[mStart:mEnd]

leftEye = shape[lStart:lEnd] rightEye = shape[rStart:rEnd] nose =
shape[nStart:nEnd] 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)
```

6.3.1 Eye Aspect Ratio (EAR)

6.3.1.1 Description

The Eye Aspect Ratio is an estimate of the eye opening state. “The Eye Aspect Ratio is a constant value when the eye is open, but rapidly falls to 0 when the eye is closed.

$$EAR = \frac{||p_2 - p_6|| + ||p_3 - p_5||}{2 ||p_1 - p_4||} \dots (5.1)$$

A program can determine if a person’s eyes are closed if the Eye Aspect Ratio falls below a certain threshold.

CImtrackr is another facial landmark plotter. In previous logs, attempted extracting accurate and consistent eye position data from the plot. Because of this experience, to explore completely different blink detection algorithms and techniques.

Frame differencing is another blink detection technique. Essentially, a program compare subsequent video frames to determine if there was any movement in a select eye region.

6.3.1.2 EAR Algorithm

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

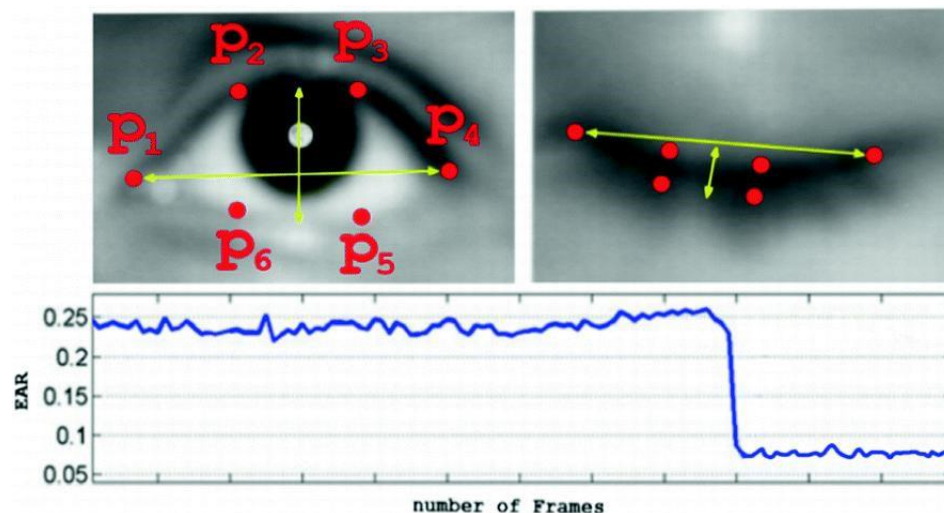


Figure 6.3.1 Eye Aspect Ratio (EAR)

6.3.2 Mouth Aspect Ratio (MAR)

6.3.2.1 Description

Highly inspired by the EAR feature, the formula a little bit tweaked to get a metric that can detect open/closed mouth.

$$\text{MAR} = \frac{||p_2 - p_1|| + ||p_3 - p_7|| + ||p_4 - p_6|| + \dots}{2 ||p_1 - p_5||} \quad (5.2)$$

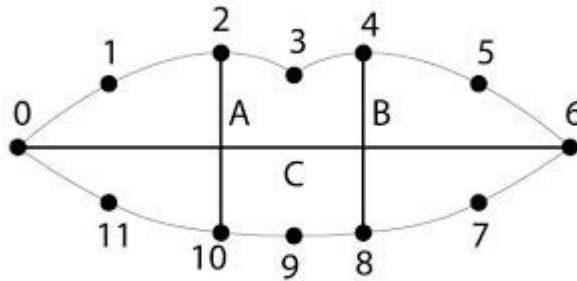


Figure 6.3.2 Mouth Aspect Ratio (MAR)

6.3.2.2 MAR Algorithm

Compute the euclidean distances between the three sets

of vertical mouth landmarks (x, y)-coordinates

D. = np.linalg.norm(mouth[13] - mouth[19])

E. = np.linalg.norm(mouth[14] - mouth[18])

F. = np.linalg.norm(mouth[15] - mouth[17])

Compute the euclidean distance between the horizontal

mouth landmarks (x, y)-coordinates

G. = 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

6.3.3 Dwell Time

The desired key is selected when the focus time has reached the threshold limit. The gaze bias effect, i.e., the tendency to look longer at items that are eventually chosen, has been shown to occur in the first dwell (initial cohesion of fixations for an item). In the two-alternative forced-choice (2AFC) paradigm, participants would look at one of the items first (defined as first look; FL), and they would then move and look at another item (second look; SL). The focus time will start new count when the previous count has reached.

6.4 MOUSE AND KEYBOARD ACTIONS

Opening of the mouth is used to control the activation and deactivation. Cursor movements are controlled by moving the head in the respective directions such as up, down, right and left. Wink actions are used to control the clicks of the mouse. Right wink is used to perform right click and left wink is used to perform left click. In addition, the scroll movements of the mouse is introduced in this model. Squinting of the eyes will activate and deactivate the scroll movements. A visual feedback is continuously provided regarding the position of the mouse.





ACTIONS	FUNCTIONS
	Activate/Deactivate Mouth Control
	Right Click
	Left Click
	Cursor Movement

Table 6.4 Mouse Movements

In virtual keyboard, the image processing is executed in the same way as the computer mouse cursor controller. Basically the GUI of the virtual keyboard is made up of two components - First, the center of the screen where the selected text is displayed and second is the edges of the screen where the character sets are displayed. Since the traditional keyboard layout is inefficient, a tree selection method is used with 6 sections. There are 3 levels. In the first level, the 6 sections are scanned by using dwell-time method. In the second level, one of the 6 section is elaborated for selection of alphabets “ABCDE, FGHIJ, KLMNO, PQRST, UVWXYZ”.

For example, if a person selects the box “ABCDE” the partitions s1, s2, s3, s4 and s5 become “A,” “B,” “C,” “D,” “E”. The third level is dedicated to numbers and commonly used predefined sentences. The predefined sentences helps the users to communicate with ease at times of emergency. A typed character can be deleted by closing the eyes for a short period of time. Furthermore, to display the key chosen by the system, a feedback is provided for the chosen key to the user in a visual form. In the beginning, the color of the button chosen is light blue. When the user gaze a particular section, color varies with respect to dwell time t . the button becomes dark blue .



A	B	C
D	F	E

Figure 6.4 Virtual Keyboard Layout

Both mouse cursor controller and the eye tracking keyboard work well in sub optimal lighting conditions. People wearing spectacles could also use the system with ease. Since an in-built webcam is used rather than an external webcam, the proposed system is cost effective.

CHAPTER 7

SNAPSHOTS

7.1 DETECTING FACIAL LANDMARKS

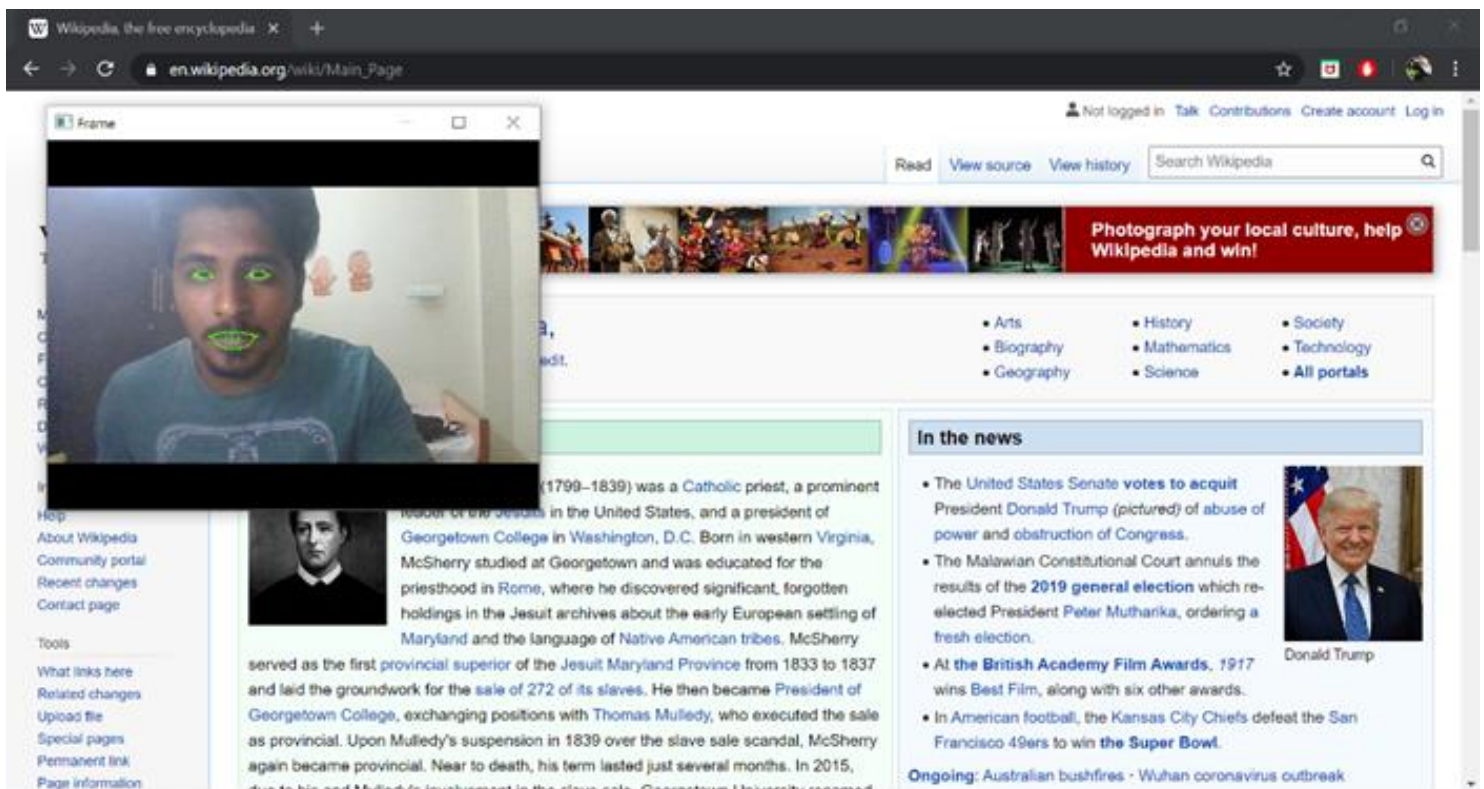


FIGURE 7.1 FACIAL LANDMARKS

7.2 ACTIVATING MOUSE CURSOR CONTROLLER

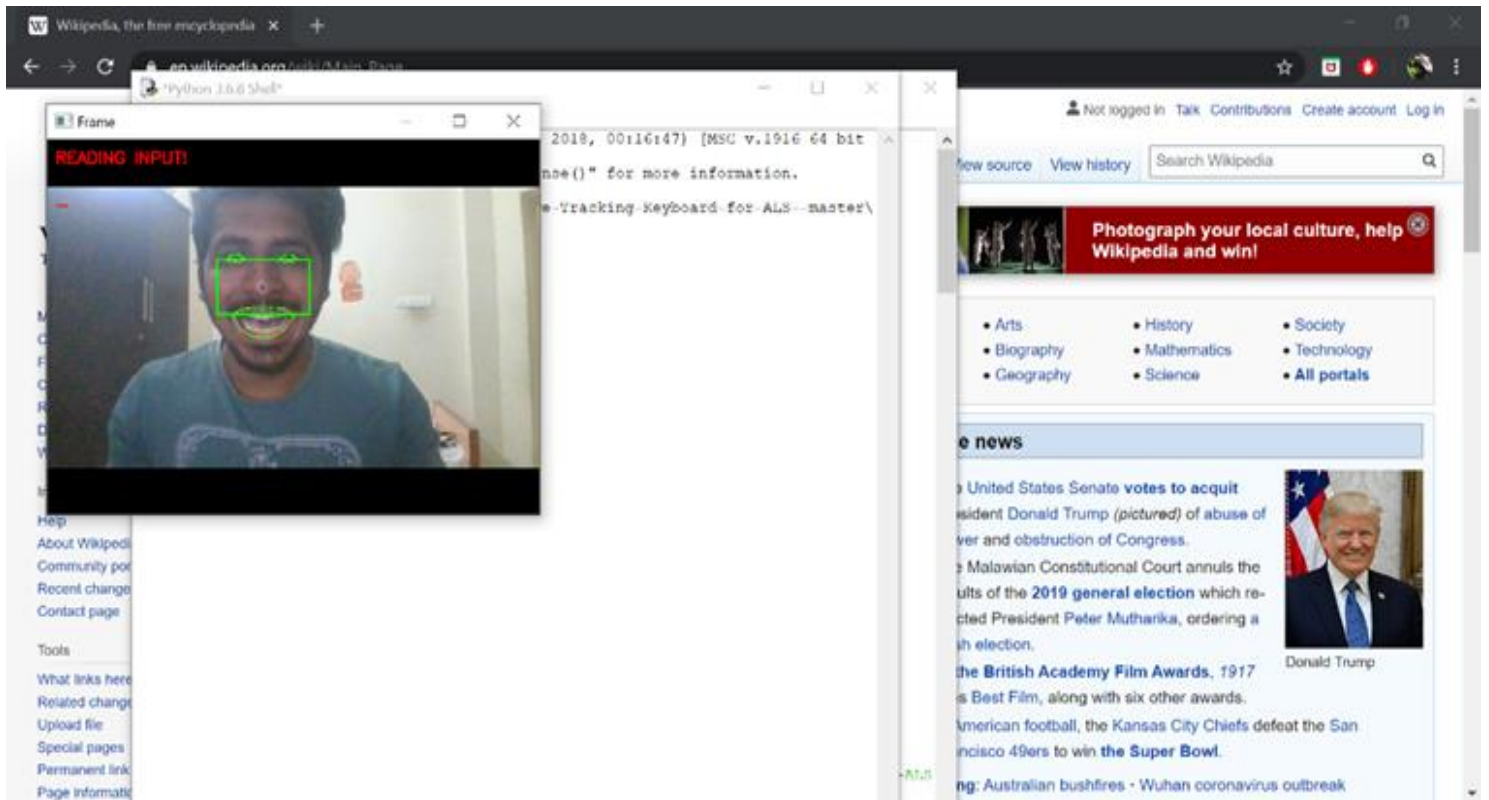


Figure 7.2 Activating Mouse Cursor Controller

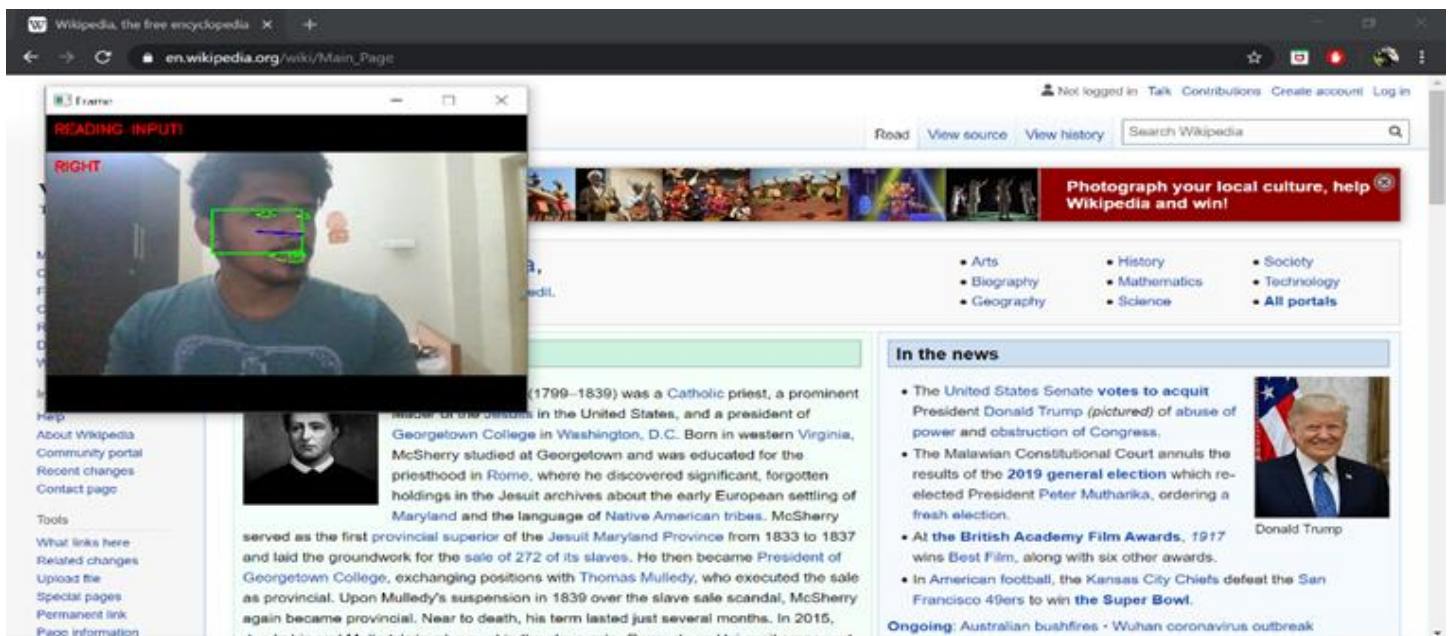


Figure 7.3 Reading Input : Right

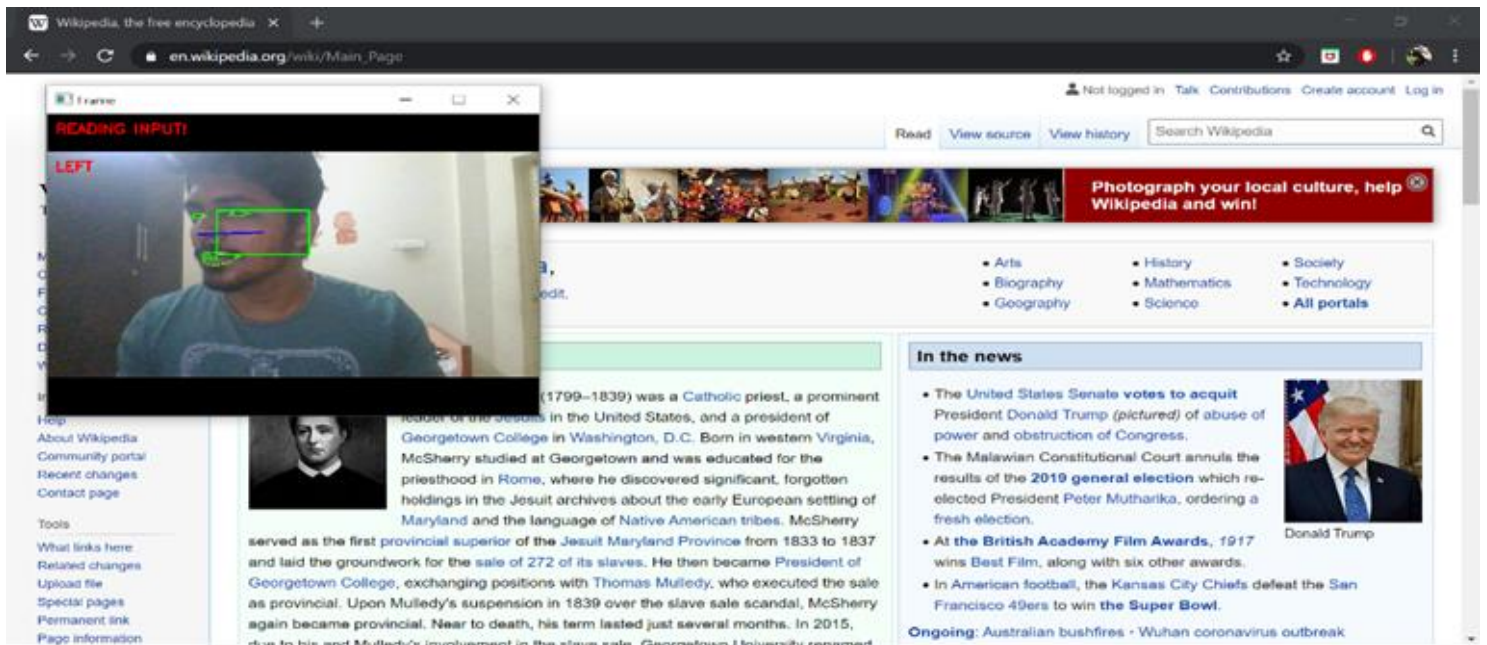


Figure 7.4 Reading Input : Left

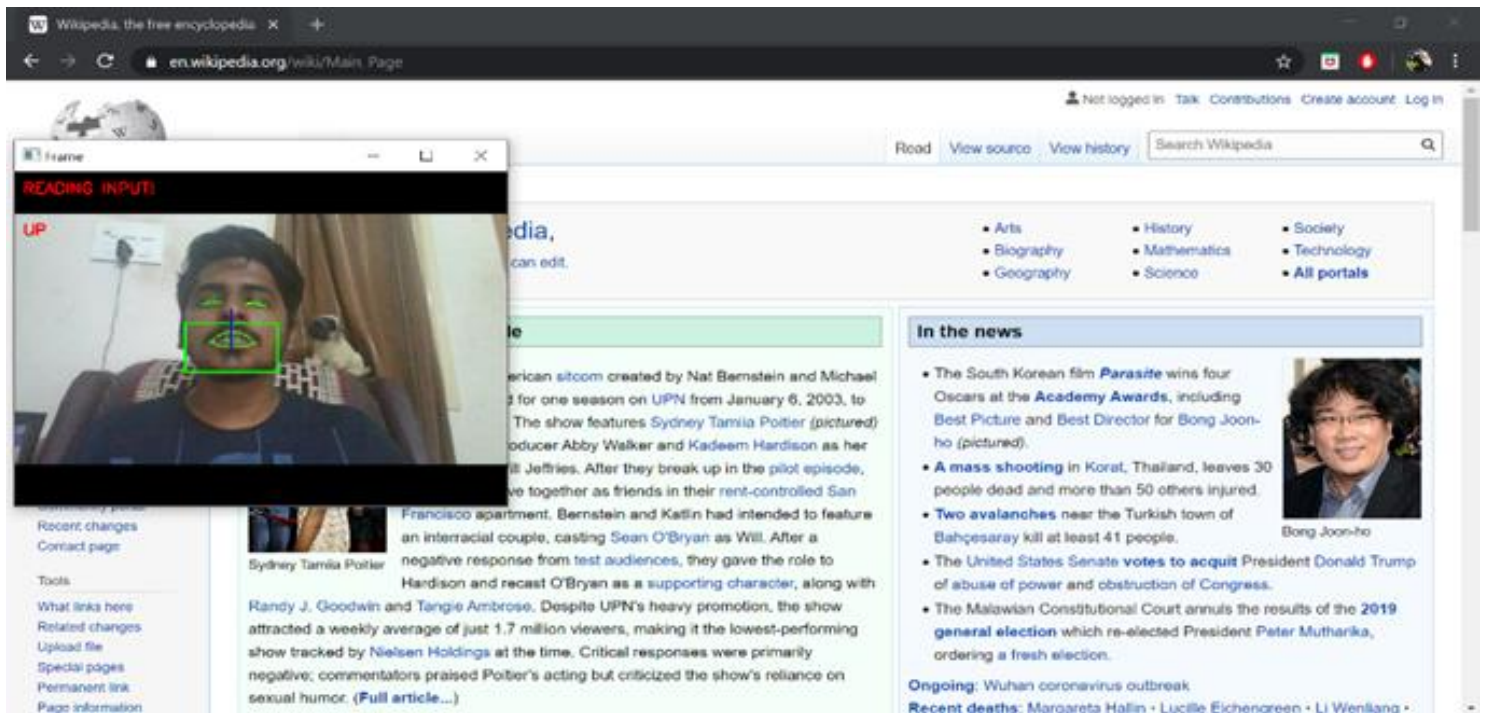


Figure 7.5 Reading Input : Up

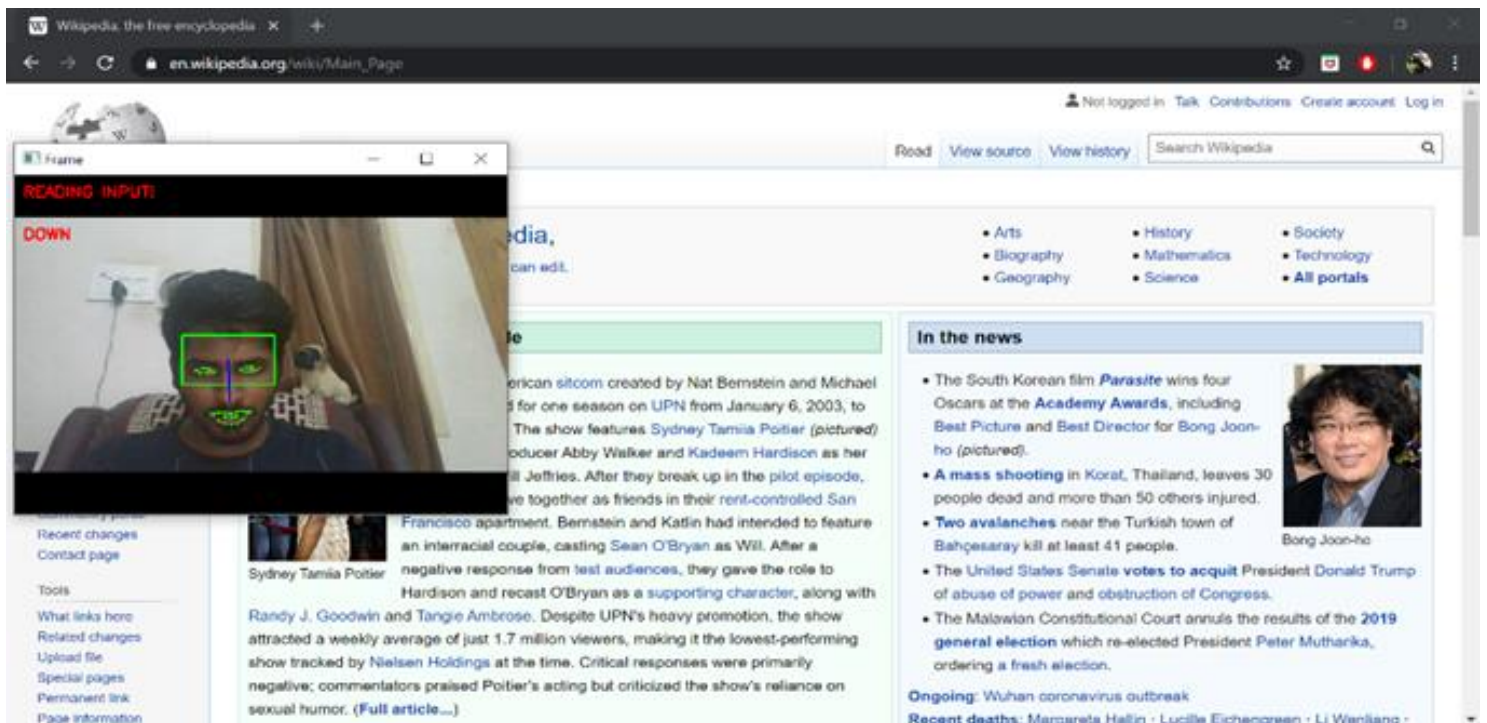


Figure 7.6 Reading Input : Down

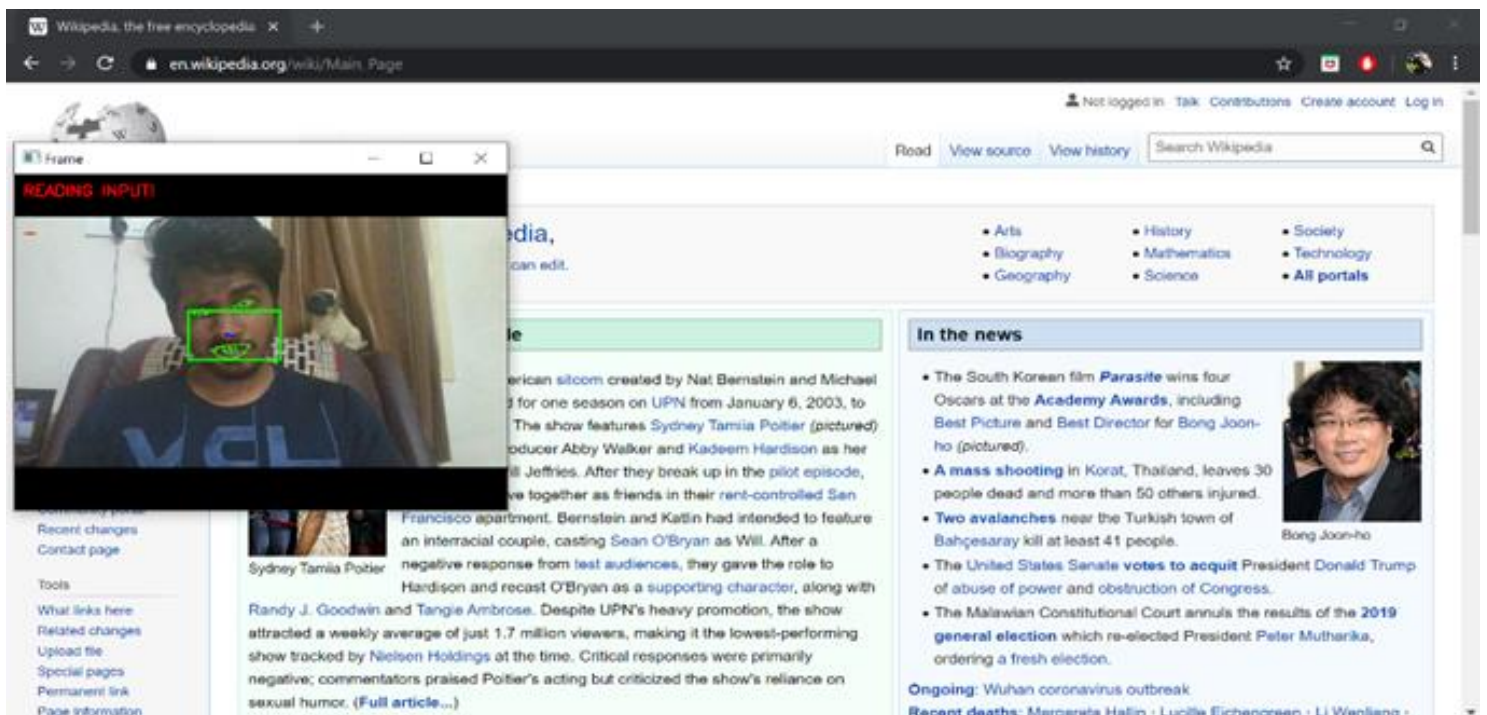
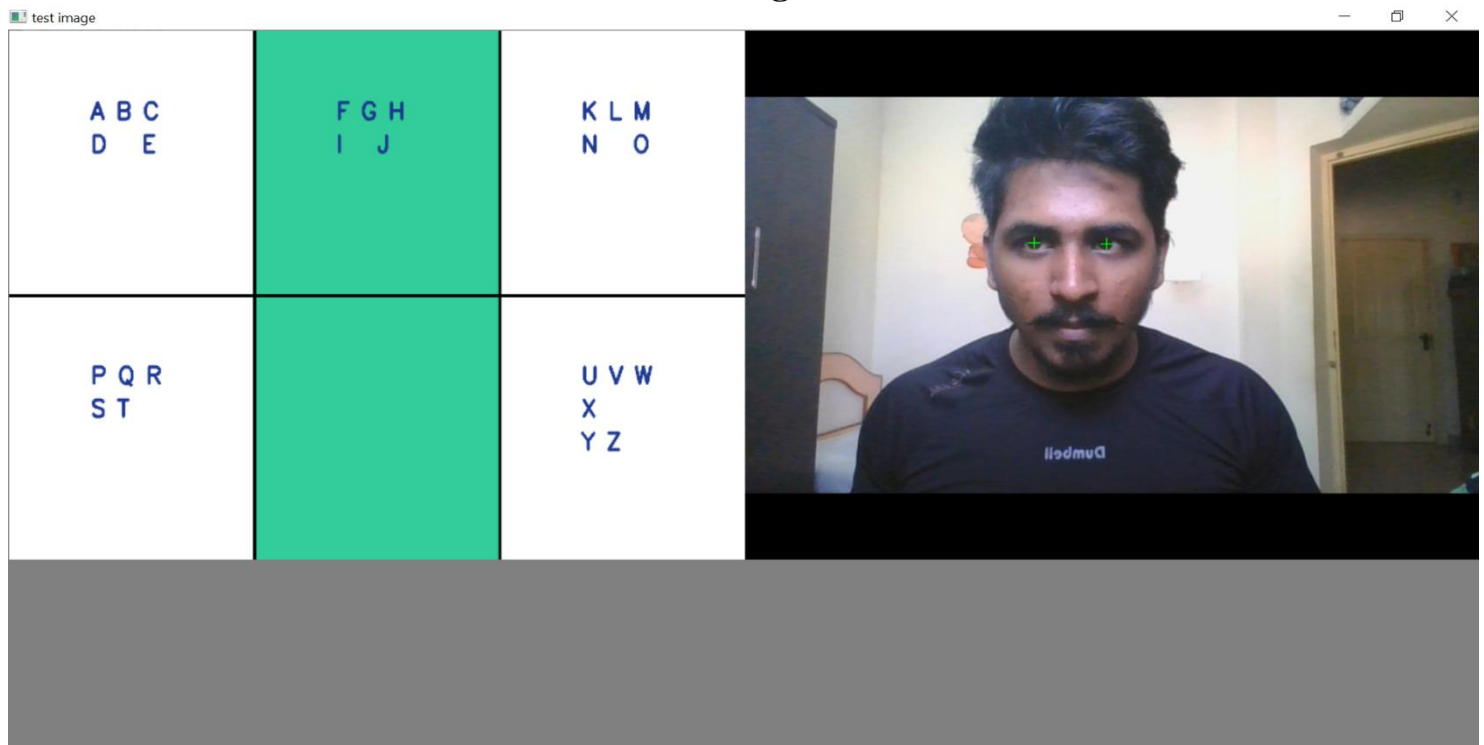


Figure 7.7 Right Click



Figure 7.8 Left Click

7.4 VIRTUAL KEYBOARD LAYOUT



7.10 Virtual Keyboard Layout

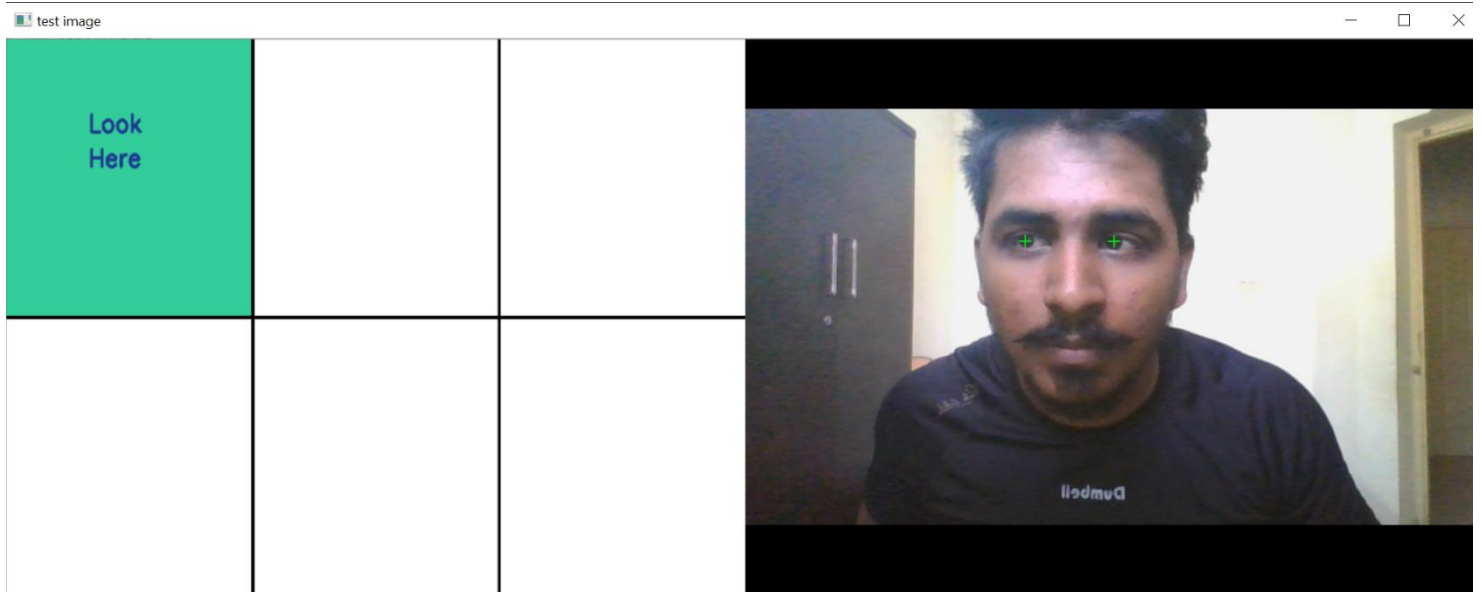


Figure 7.11 Reading Input : Up

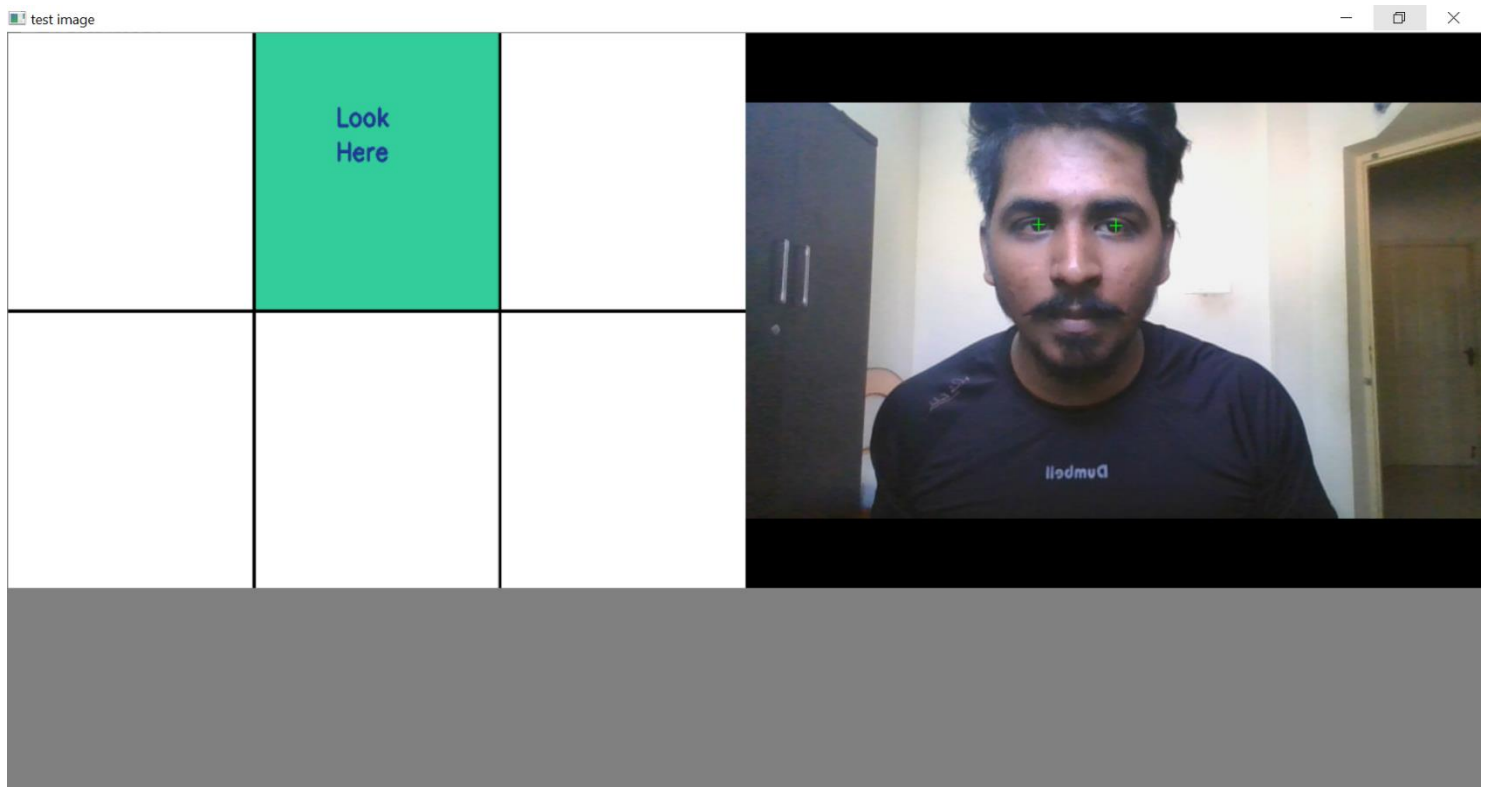


Figure 7.12 Up Center

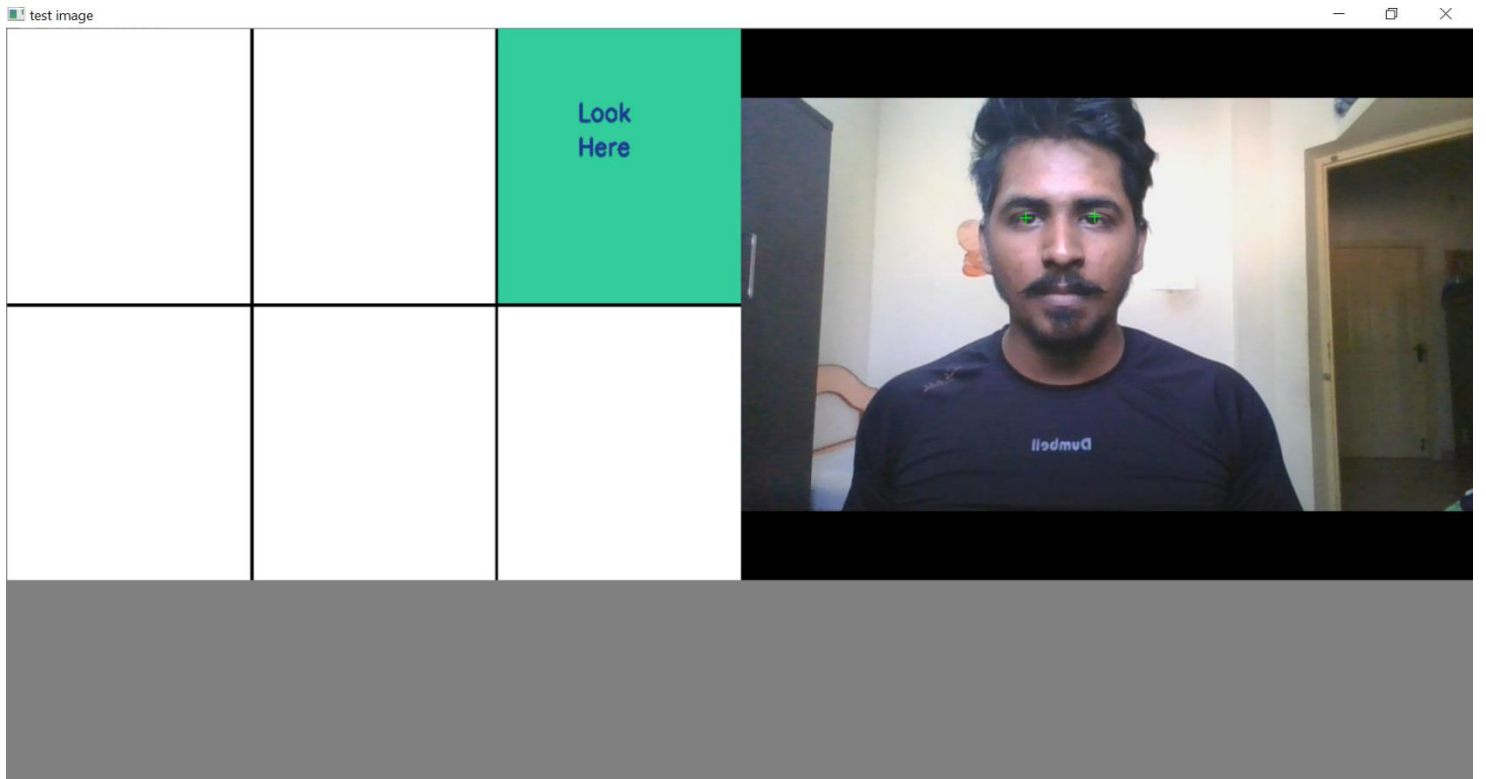


Figure 7.13 Up Right

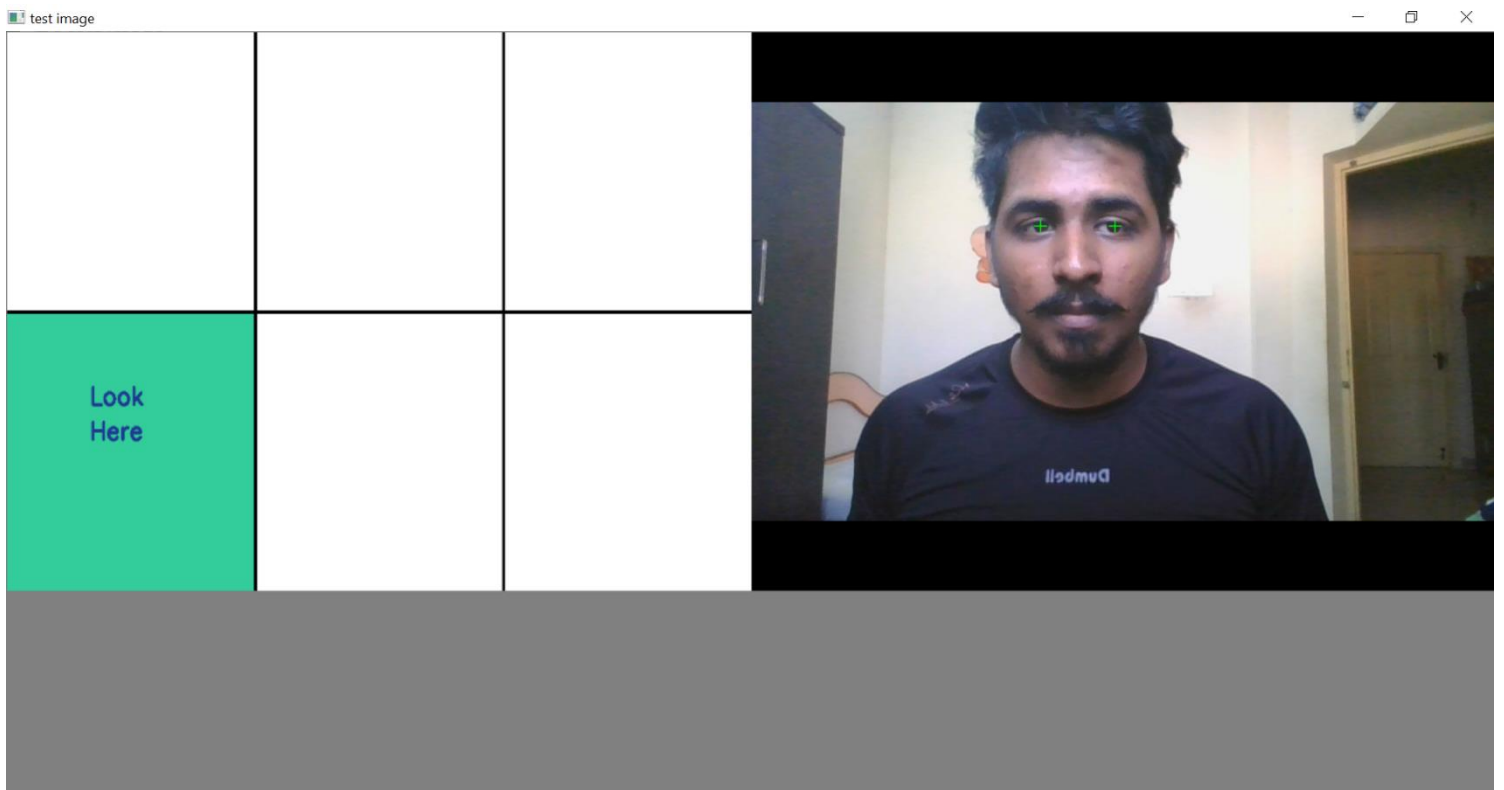


Figure 7.14 Down Left

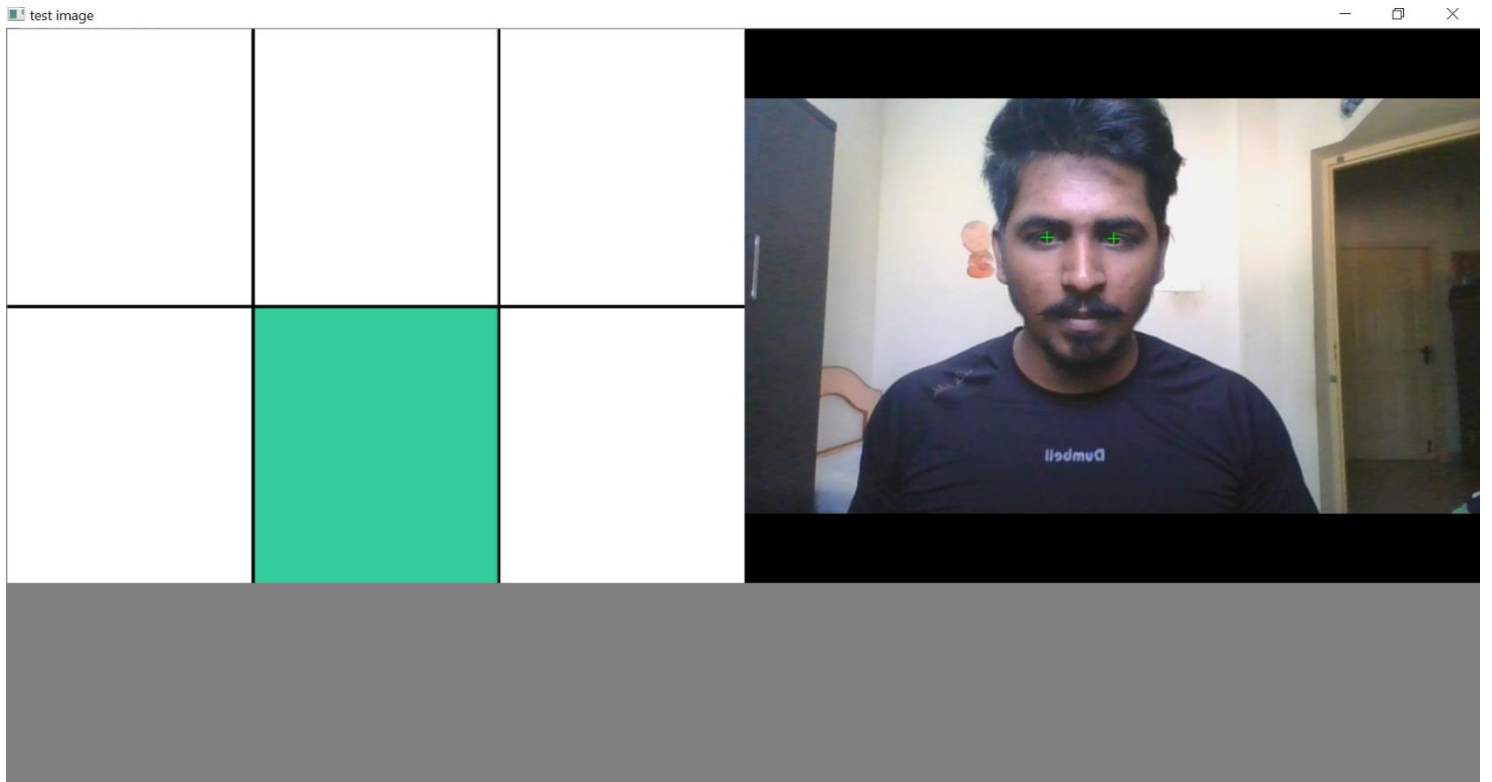


Figure 7.15 Reading Input : Down Center

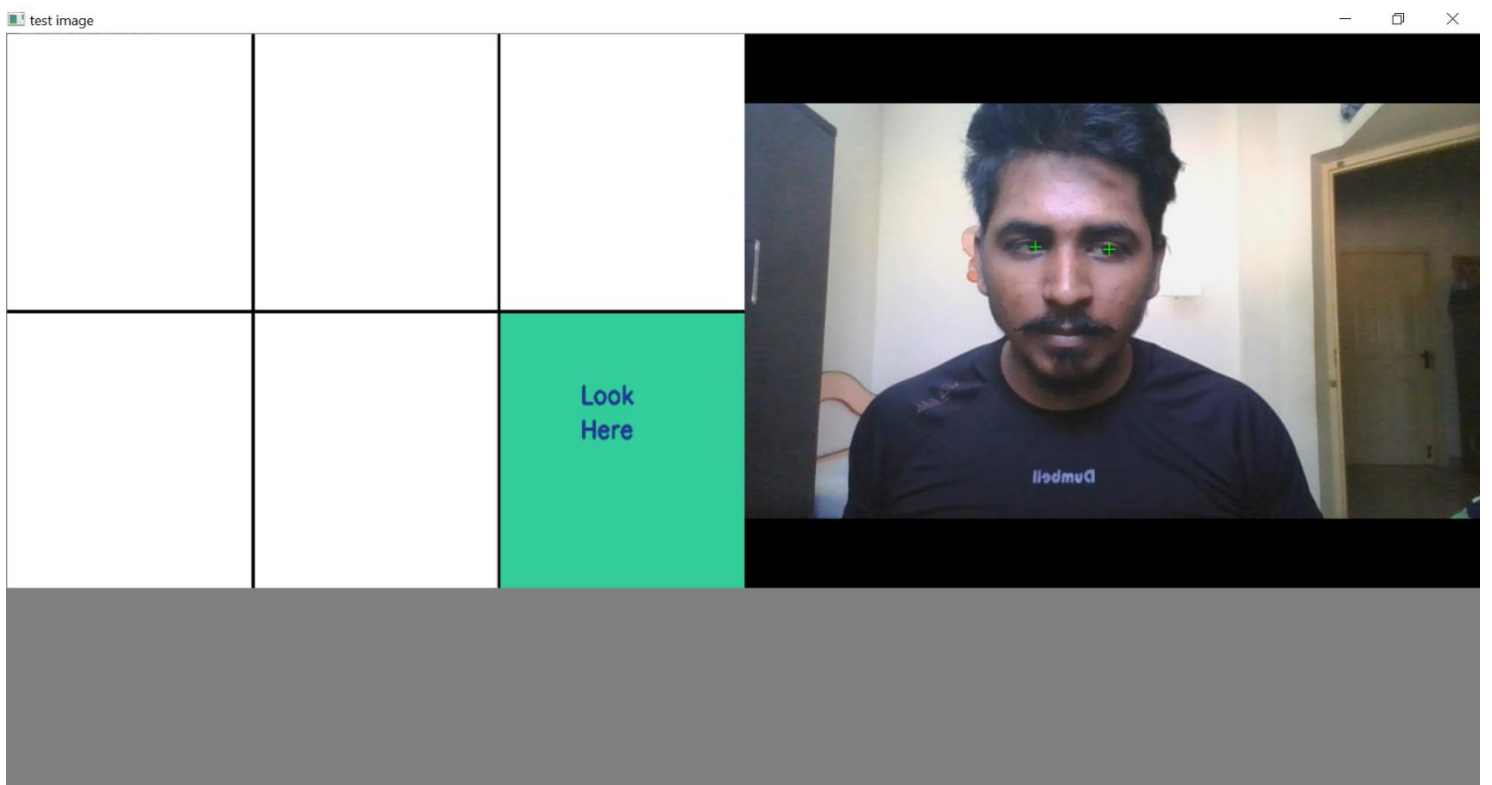


Figure 7.16 Reading Input : Down Right

7.4 CHARACTER SELECTION IN VIRTUAL KEYBOARD



Figure 7.17 Character Selection In Virtual Keyboard

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

The suggested model provides an alternative solution to external input devices. The system provides an alternative solution for future computing devices, such as PDA, Tablet, Laptops, and GPS. It inspires various PC based application like gaming, entertainment, robotics, etc. The suggested system improves the living conditions of disabled people since the system works efficiently even under sub-optimal lighting and with the use of spectacles.

8.2 FUTURE WORK

Further research will endeavour to produce efficient movements to perform the click events. The frame work could be improved to reduce the error and handle partial or unwanted labels. Tween function can be integrated in the model which dictates the progress of the mouse as it moves to its destination. In addition to visual feedback, audio feedback can also be provided to intimate the user when an action is performed. Several applications can be developed based on this man-machine interface.

CHAPTER 9

APPENDIXES

Coding:

9.1 Calibration.py :

```
from __future__ import division
```

```
import cv2
```

```
from pupil import Pupil
```

```
class Calibration(object):
```

```
    """
```

```
    This class calibrates the pupil detection algorithm by finding the  
    best binarization threshold value for the person and the webcam.
```

```
    """
```

```
    def __init__(self):
```

```
        self.nb_frames = 20
```

```
        self.thresholds_left = []
```

```
        self.thresholds_right = []
```

```
    def is_complete(self):
```

```
        """Returns true if the calibration is completed"""
```

```
        return len(self.thresholds_left) >= self.nb_frames and  
len(self.thresholds_right) >= self.nb_frames
```

```
def threshold(self, side):
```

```
    """Returns the threshold value for the given eye.
```

```
    Argument:
```

```
        side: Indicates whether it's the left eye (0) or the right eye (1)
```

```
    """
```

```
    if side == 0:
```

```
        return int(sum(self.thresholds_left) / len(self.thresholds_left))
```

```
    elif side == 1:
```

```
        return int(sum(self.thresholds_right) / len(self.thresholds_right))
```

```
@staticmethod
```

```
def iris_size(frame):
```

```
    """Returns the percentage of space that the iris takes up on
    the surface of the eye.
```

```
    Argument:
```

```
        frame (numpy.ndarray): Binarized iris frame
```

```
    """
```

```
    frame = frame[5:-5, 5:-5]
```

```
    height, width = frame.shape[:2]
```

```
    nb_pixels = height * width
```

```
    nb_blacks = nb_pixels - cv2.countNonZero(frame)
```

```
return nb_blacks / nb_pixels
```

```
@staticmethod
```

```
def find_best_threshold(eye_frame):
```

```
    """Calculates the optimal threshold to binarize the  
    frame for the given eye.
```

```
Argument:
```

```
    eye_frame (numpy.ndarray): Frame of the eye to be analyzed
```

```
    """
```

```
    average_iris_size = 0.48
```

```
    trials = {}
```

```
    for threshold in range(5, 100, 5):
```

```
        iris_frame = Pupil.image_processing(eye_frame, threshold)
```

```
        trials[threshold] = Calibration.iris_size(iris_frame)
```

```
    best_threshold, iris_size = min(trials.items(), key=(lambda p: abs(p[1] -  
    average_iris_size)))
```

```
    return best_threshold
```

```
def evaluate(self, eye_frame, side):
```

```
    """Improves calibration by taking into consideration the  
    given image.
```


Arguments:

eye_frame (numpy.ndarray): Frame of the eye

side: Indicates whether it's the left eye (0) or the right eye (1)

"""

threshold = self.find_best_threshold(eye_frame)

if side == 0:

self.thresholds_left.append(threshold)

elif side == 1:

self.thresholds_right.append(threshold)

9.2 Eye.py :

import math

import numpy as np

import cv2

from pupil import Pupil

class Eye(object):

"""

This class creates a new frame to isolate the eye and

initiates the pupil detection.

"""

LEFT_EYE_POINTS = [36, 37, 38, 39, 40, 41]

```
RIGHT_EYE_POINTS = [42, 43, 44, 45, 46, 47]
```

```
def __init__(self, original_frame, landmarks, side, calibration):
```

```
    self.frame = None
```

```
    self.origin = None
```

```
    self.center = None
```

```
    self.pupil = None
```

```
    self._analyze(original_frame, landmarks, side, calibration)
```

```
@staticmethod
```

```
def _middle_point(p1, p2):
```

```
    """Returns the middle point (x,y) between two points
```

```
    Arguments:
```

```
        p1 (dlib.point): First point
```

```
        p2 (dlib.point): Second point
```

```
    """
```

```
    x = int((p1.x + p2.x) / 2)
```

```
    y = int((p1.y + p2.y) / 2)
```

```
    return (x, y)
```

```
def _isolate(self, frame, landmarks, points):
```

```
    """Isolate an eye, to have a frame without other part of the face.
```

Arguments:

frame (numpy.ndarray): Frame containing the face

landmarks (dlib.full_object_detection): Facial landmarks for the face region

points (list): Points of an eye (from the 68 Multi-PIE landmarks)

"""

```
region = np.array([(landmarks.part(point).x, landmarks.part(point).y) for point
in points])
```

```
region = region.astype(np.int32)
```

```
# Applying a mask to get only the eye
```

```
height, width = frame.shape[:2]
```

```
black_frame = np.zeros((height, width), np.uint8)
```

```
mask = np.full((height, width), 255, np.uint8)
```

```
cv2.fillPoly(mask, [region], (0, 0, 0))
```

```
eye = cv2.bitwise_not(black_frame, frame.copy(), mask=mask)
```

```
# Cropping on the eye
```

```
margin = 5
```

```
min_x = np.min(region[:, 0]) - margin
```

```
max_x = np.max(region[:, 0]) + margin
```

```
min_y = np.min(region[:, 1]) - margin
```

```
max_y = np.max(region[:, 1]) + margin
```

```
self.frame = eye[min_y:max_y, min_x:max_x]
```

```
self.origin = (min_x, min_y)
```

```
height, width = self.frame.shape[:2]
self.center = (width / 2, height / 2)
```

```
def _blinking_ratio(self, landmarks, points):
```

```
    """Calculates a ratio that can indicate whether an eye is closed or not.
    It's the division of the width of the eye, by its height.
```

Arguments:

landmarks (dlib.full_object_detection): Facial landmarks for the face region

points (list): Points of an eye (from the 68 Multi-PIE landmarks)

Returns:

The computed ratio

```
    """
```

```
    left = (landmarks.part(points[0]).x, landmarks.part(points[0]).y)
```

```
    right = (landmarks.part(points[3]).x, landmarks.part(points[3]).y)
```

```
    top = self._middle_point(landmarks.part(points[1]),
landmarks.part(points[2]))
```

```
    bottom = self._middle_point(landmarks.part(points[5]),
landmarks.part(points[4]))
```

```
    eye_width = math.hypot((left[0] - right[0]), (left[1] - right[1]))
```

```
    eye_height = math.hypot((top[0] - bottom[0]), (top[1] - bottom[1]))
```

```
    try:
```

```
        ratio = eye_width / eye_height
```

```
except ZeroDivisionError:
```

```
    ratio = None
```

```
return ratio
```

```
def _width_and_height(self, landmarks, points):
```

```
    left = (landmarks.part(points[0]).x, landmarks.part(points[0]).y)
```

```
    right = (landmarks.part(points[3]).x, landmarks.part(points[3]).y)
```

```
    top = self._middle_point(landmarks.part(points[1]),  
landmarks.part(points[2]))
```

```
    bottom = self._middle_point(landmarks.part(points[5]),  
landmarks.part(points[4]))
```

```
    eye_width = math.hypot((left[0] - right[0]), (left[1] - right[1]))
```

```
    eye_height = math.hypot((top[0] - bottom[0]), (top[1] - bottom[1]))
```

```
return eye_width, eye_height
```

```
def _analyze(self, original_frame, landmarks, side, calibration):
```

```
    """Detects and isolates the eye in a new frame, sends data to the calibration  
and initializes Pupil object.
```

Arguments:

original_frame (numpy.ndarray): Frame passed by the user

landmarks (dlib.full_object_detection): Facial landmarks for the face region

side: Indicates whether it's the left eye (0) or the right eye (1)

calibration (calibration.Calibration): Manages the binarization threshold value

```
"""
```

```
if side == 0:
```

```
    points = self.LEFT_EYE_POINTS
```

```
elif side == 1:
```

```
    points = self.RIGHT_EYE_POINTS
```

```
else:
```

```
    return
```

```
self.blinking = self._blinking_ratio(landmarks, points)
```

```
self.width, self.height = self._width_and_height(landmarks, points)
```

```
self._isolate(original_frame, landmarks, points)
```

```
if not calibration.is_complete():
```

```
    calibration.evaluate(self.frame, side)
```

```
threshold = calibration.threshold(side)
```

```
self.pupil = Pupil(self.frame, threshold)
```

9.3 Gacze_tracking.py:

```
from __future__ import division
```

```
import os
```

```
import cv2
```

```
import dlib
```

```
from eye import Eye
```

```
from calibration import Calibration
```

```
class GazeTracking(object):
```

```
    """
```

```
    This class tracks the user's gaze.
```

```
    It provides useful information like the position of the eyes  
    and pupils and allows to know if the eyes are open or closed
```

```
    """
```

```
    def __init__(self):
```

```
        self.frame = None
```

```
        self.eye_left = None
```

```
        self.eye_right = None
```

```
        self.calibration = Calibration()
```

```
        # _face_detector is used to detect faces
```

```
        self._face_detector = dlib.get_frontal_face_detector()
```

```
        # _predictor is used to get facial landmarks of a given face
```

```
        model_path = r"C:\Users\Ghajaanan Jeyakumara\Desktop\miya\Unfreeze-the-  
Frozen-World-An-Eye-Tracking-Keybaord-for-ALS--master\Unfreeze-the-Frozen-  
World-An-Eye-Tracking-Keybaord-for-ALS--  
master\trained_models\shape_predictor_68_face_landmarks.dat"
```

```
        self._predictor = dlib.shape_predictor(model_path)
```

```

@property
def pupils_located(self):
    """Check that the pupils have been located"""
    try:
        int(self.eye_left.pupil.x)
        int(self.eye_left.pupil.y)
        int(self.eye_right.pupil.x)
        int(self.eye_right.pupil.y)
        return True
    except Exception:
        return False

def _analyze(self):
    """Detects the face and initialize Eye objects"""
    frame = cv2.cvtColor(self.frame, cv2.COLOR_BGR2GRAY)
    faces = self._face_detector(frame)

    try:
        landmarks = self._predictor(frame, faces[0])
        self.eye_left = Eye(frame, landmarks, 0, self.calibration)
        self.eye_right = Eye(frame, landmarks, 1, self.calibration)

    except IndexError:
        self.eye_left = None
        self.eye_right = None

```



```
self.faces = faces
```

```
def refresh(self, frame):
```

```
    """Refreshes the frame and analyzes it.
```

```
    Arguments:
```

```
        frame (numpy.ndarray): The frame to analyze
```

```
    """
```

```
    self.frame = frame
```

```
    self._analyze()
```

```
def pupil_left_coords(self):
```

```
    """Returns the coordinates of the left pupil"""
```

```
    if self.pupils_located:
```

```
        x = self.eye_left.origin[0] + self.eye_left.pupil.x
```

```
        y = self.eye_left.origin[1] + self.eye_left.pupil.y
```

```
        return (x, y)
```

```
def pupil_right_coords(self):
```

```
    """Returns the coordinates of the right pupil"""
```

```
    if self.pupils_located:
```

```
        x = self.eye_right.origin[0] + self.eye_right.pupil.x
```

```
        y = self.eye_right.origin[1] + self.eye_right.pupil.y
```

```
        return (x, y)
```

```

def set_clf(self, clf):
    self.clf = clf

def current_gaze(self):
    if self.horizontal_ratio is None or self.blinking_ratio is None:
        return 6
    else:
        if self.is_top():
            return self.clf.predict([[self.horizontal_ratio, self.blinking_ratio]])%3
        else:
            return 3+(self.clf.predict([[self.horizontal_ratio, self.blinking_ratio]])%3)
        #if self.is_top() and
        #return self.clf.predict([[self.horizontal_ratio, self.blinking_ratio]])

@property
def horizontal_ratio(self):
    """Returns a number between 0.0 and 1.0 that indicates the
    horizontal direction of the gaze. The extreme right is 0.0,
    the center is 0.5 and the extreme left is 1.0
    """
    if self.pupils_located:
        pupil_left = self.eye_left.pupil.x / (self.eye_left.center[0] * 2 - 10)
        pupil_right = self.eye_right.pupil.x / (self.eye_right.center[0] * 2 - 10)
        return (pupil_left + pupil_right) / 2

```

```

def vertical_ratio(self):
    """Returns a number between 0.0 and 1.0 that indicates the
    vertical direction of the gaze. The extreme top is 0.0,
    the center is 0.5 and the extreme bottom is 1.0
    """
    if self.pupils_located:
        pupil_left = self.eye_left.pupil.y / (self.eye_left.center[1] * 2 - 10)
        pupil_right = self.eye_right.pupil.y / (self.eye_right.center[1] * 2 - 10)
        return (pupil_left + pupil_right) / 2

def is_right(self):
    """Returns true if the user is looking to the right"""
    if self.pupils_located:
        return self.horizontal_ratio() <= 0.4

def is_left(self):
    """Returns true if the user is looking to the left"""
    if self.pupils_located:
        return self.horizontal_ratio() >= 0.7

def is_top_right(self):
    """Returns true if the user is looking to the right"""
    if self.pupils_located:
        return self.horizontal_ratio() <= 0.4

```

```
def is_top_left(self):  
    """Returns true if the user is looking to the left"""  
    if self.pupils_located:  
        return self.horizontal_ratio() >= 0.7
```

```
@property
```

```
def blinking_ratio(self):  
    if self.pupils_located:  
        return (self.eye_left.blinking + self.eye_right.blinking) / 2
```

```
def is_top(self):  
    """Returns true if the user is looking to the up"""  
    if self.pupils_located:  
        return self.blinking_ratio < 2.95
```

```
def is_bottom(self):  
    """Returns true if the user is looking to the up"""  
    if self.pupils_located:  
        return self.blinking_ratio >= 2.95
```

```
def is_center(self):  
    """Returns true if the user is looking to the center"""  
    if self.pupils_located:  
        return self.is_right() is not True and self.is_left() is not True and  
self.is_top() is not True \
```

```
and self.is_bottom() is not True
```

```
def is_blinking(self):
```

```
    """Returns true if the user closes his eyes"""
```

```
    if self.pupils_located:
```

```
        return self.blinking_ratio > 3.8
```

```
    try:
```

```
        self.faces[0]
```

```
        return False
```

```
    except IndexError:
```

```
        return True
```

```
def width_and_height(self):
```

```
    if self.pupils_located:
```

```
        width = (self.eye_left.width + self.eye_right.width) / 2
```

```
        height = (self.eye_left.height + self.eye_right.height) / 2
```

```
        blinking_ratio = (self.eye_left.blinking + self.eye_right.blinking) / 2
```

```
        return blinking_ratio
```

```
def annotated_frame(self):
```

```
    """Returns the main frame with pupils highlighted"""
```

```
    frame = self.frame.copy()
```

```
    if self.pupils_located:
```

```
        color = (0, 255, 0)
```

```
x_left, y_left = self.pupil_left_coords()
x_right, y_right = self.pupil_right_coords()
cv2.line(frame, (x_left - 5, y_left), (x_left + 5, y_left), color)
cv2.line(frame, (x_left, y_left - 5), (x_left, y_left + 5), color)
cv2.line(frame, (x_right - 5, y_right), (x_right + 5, y_right), color)
cv2.line(frame, (x_right, y_right - 5), (x_right, y_right + 5), color)
```

```
return frame
```

9.4 Main_demo.py :

```
import cv2
from gaze_tracking import GazeTracking
from page import Page
from sklearn.svm import SVC
import pandas as pd
import numpy as np

gaze = GazeTracking()
webcam = cv2.VideoCapture(0)
outn="init.avi"

fps = webcam.get(cv2.CAP_PROP_FPS)
print(fps)
_, frame = webcam.read()
(hgt, wid, dep) = frame.shape
frc = cv2.VideoWriter_fourcc('M', 'J', 'P', 'G')
```

```
out = cv2.VideoWriter(outn, frc, 25.0, (wid*2, hgt), 1)
```

```
page_initialization = Page(-1, gaze, webcam, out)
```

```
gaze.set_clf(page_initialization.initialization())
```

```
# x = []
```

```
# y = []
```

```
#
```

```
# data_size = 150
```

```
#
```

```
# horizontal = pd.read_csv('trained_models/dict_horizontal.csv', sep=',',  
header=None).values
```

```
# vertical = pd.read_csv('trained_models/dict_vertical.csv', sep=',',  
header=None).values
```

```
# for i in range(6):
```

```
#     x = horizontal[i][1:]
```

```
#     y = vertical[i][1:]
```

```
#     xy = np.concatenate((np.array(x).reshape(data_size, 1),  
np.array(y).reshape(data_size, 1)), axis = 1)
```

```
#     if i>0:
```

```
#         data = np.concatenate((data, xy), axis = 0)
```

```
#     else: data = xy
```

```
#
```

```
# horizontal = pd.read_csv('trained_models/dict_horizontal.csv', sep=',',  
header=None).values
```

```
# vertical = pd.read_csv('trained_models/dict_vertical.csv', sep=',',  
header=None).values
```

```

# for i in range(6):
#     x = horizontal[i][1:]
#     y = vertical[i][1:]
#     xy = np.concatenate((np.array(x).reshape(data_size, 1),
# np.array(y).reshape(data_size, 1)), axis = 1)
#     if i>0:
#         data_test = np.concatenate((data_test, xy), axis=0)
#     else: data_test = xy
# data = np.concatenate((data, data_test), axis = 0)
# y = np.zeros((6*data_size,1))
# for i in range(6):
#     y[i*data_size:(i+1)*data_size] = i
# y2 = np.zeros((6*data_size,1))
# for i in range(6):
#     y2[i*data_size:(i+1)*data_size] = i
# y = np.concatenate((y, y2))
# y = np.ravel(y)
#
# clf = SVC(decision_function_shape='ovo')
# clf.fit(data, y)
#
# gaze.set_clf(clf)

pages = [Page(0, gaze, webcam,out), Page(1, gaze, webcam,out), Page(2, gaze,
webcam,out), Page(3, gaze, webcam,out),

```



```
Page(4, gaze, webcam,out), Page(5, gaze, webcam,out), Page(6, gaze,
webcam,out), Page(7, gaze, webcam,out),
```

```
Page(8, gaze, webcam,out), Page(9, gaze, webcam,out), Page(10, gaze,
webcam,out)]
```

```
index = 0
```

```
str_input = ""
```

```
while True:
```

```
    index, str_input, whe_break = pages[index].run(str_input, out)
```

```
    if(whe_break):
```

```
        break
```

```
    if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
        break
```

```
print(str_input)
```

```
webcam.release()
```

```
out.release()
```

```
print(hgt, wid)
```

9.5 Page.py:

```
import cv2
```

```
import numpy as np
```

```
from sklearn.svm import SVC
```

```
import time
```

```
colors = {0: [255, 255, 255], 1: [200, 200, 200], 2: [154, 205, 50], 3: (0, 255, 255)}
```

```
class Page(object):
```

```
    size = 400
```

```
    def __init__(self, index, gaze, webcam, out):
```

```
        self.out = out
```

```
        self.page_index = index
```

```
        self.color_array = [0, 0, 0, 0, 1, 0]
```

```
        self.dominator_num = -1
```

```
        self.dominator_time = -1
```

```
        self.whe_initialized = 0
```

```
        self.gaze = gaze
```

```
        self.webcam = webcam
```

```
        if self.page_index == 0:
```

```
            self.page_transition = [1, 2, 3, 4, -3, 5, -1]
```

```
        elif self.page_index == 1 or self.page_index == 2 or self.page_index == 3 or  
self.page_index == 4:
```

```
            self.page_transition = [0, 0, 0, 0, -3, 0, 0]
```

```
        elif self.page_index == 5:
```

```
            self.page_transition = [6, 7, 8, 9, -3, 10, 0]
```

```
        else: # 6, 7, 8, 9, 10
```

```
            self.page_transition = [0, 0, 0, 0, -3, 0, 5]
```

```
        self.add_key = []
```

```

if self.page_index == 1:
    self.add_key = ["A", "C", "D", "E", "", "F"]
elif self.page_index == 2:
    self.add_key = ["H", "I", "L", "M", "", "N"]
elif self.page_index == 3:
    self.add_key = ["O", "R", "S", "T", "", "U"]
elif self.page_index == 4:
    self.add_key = ["W", "Y", " ", "YES", "", "NO"]
elif self.page_index == 6:
    self.add_key = ["B", "G", "J", "K", "", "P"]
elif self.page_index == 7:
    self.add_key = ["Q", "V", "X", "Z", "", "0"]
elif self.page_index == 8:
    self.add_key = ["1", "2", "3", "4", "", "5"]
elif self.page_index == 9:
    self.add_key = ["6", "7", "8", "9", "", "I NEED HELP"]
elif self.page_index == 10:
    self.add_key = ["I NEED FOOD", "I NEED WATER", "I WANT TO USE
THE RESTROOM", "THANK YOU", "", "I LOVE YOU"]

self.strings = []
if self.page_index == 0:
    self.strings = \

        ["A C D\nE  F", "H I L\nM  N", "O R S\nT   U", "W Y Sp\nYes No",
        "", "Other\nNumbers \nShortcut"]

```

```

elif self.page_index == 1 or self.page_index == 2 or self.page_index == 3 or
self.page_index == 6 or \
    self.page_index == 7 or self.page_index == 8 or self.page_index == 9:
    self.strings = self.add_key
elif self.page_index == 4:
    self.strings = ["W", "Y", "Sp", "YES", "", "NO"]
elif self.page_index == 5:
    self.strings = \
        ["B G J\nK P", "Q V X\nZ 0", "1 2 3\n4 5", "6 7 8\n9 HELP", "",
"Shortcut"]
elif self.page_index == 10:
    self.strings = \
        ["I NEED\nFOOD", "I NEED\nWATER", "I WANT TO\nUSE
THE\nRESTROOM", "THANK\nYOU", "", "I LOVE\nYOU"]
    self.whe_break = False

```

```

def run(self, str_display, out):
    self.out = out
    length_of_past = 30
    thresh_dominator_counts = 10
    thresh_of_center = 0.3
    threshold_of_dominator = 0.5

    stack_of_past = []
    state_count = np.zeros((7, 1))
    while True:

```

```
next_index = self.page_index
next_color = np.zeros((6, 1))

_, frame = self.webcam.read()
self.gaze.refresh(frame)
frame = self.gaze.annotated_frame()

if self.gaze.is_blinking:
    pass

# 0, 1, 2, 3, 4, 5, 6 (eyes closed)
current_look_at = int(self.gaze.current_gaze())

if len(stack_of_past) < length_of_past:
    stack_of_past.append(current_look_at)
else:
    temp = stack_of_past.pop(0)
    state_count[int(temp)] -= 1
    stack_of_past.append(current_look_at)
state_count[current_look_at] += 1

# check whether eyes closed is the dominator, no matter initialized or not
print(np.max(state_count))
print("whe_init", self.whe_initialized)
```

```

print(state_count[4])
if self.whe_initialized == 0:
    if state_count[4] >= length_of_past * thresh_of_center:
        self.whe_initialized = 1

if np.max(state_count) > length_of_past * threshold_of_dominator:
    this_dominator = np.where(state_count == np.max(state_count))[0]
    this_dominator = this_dominator[0]
    #print("#", str(this_dominator))
    #print("last", str(self.dominator_num))

if this_dominator == 6: # eyes closed
    if this_dominator == self.dominator_num:
        self.dominator_time += 1
    else:
        self.dominator_time = 1
        self.dominator_num = 6
elif self.whe_initialized == 0:
    if this_dominator == 4:
        self.whe_initialized = 1
        self.dominator_time = 0
    else:
        if this_dominator == 4 or this_dominator == -1:
            self.dominator_num = -1
            self.dominator_time = 0

```

```

else:
    #print("panduan:", str(this_dominator == self.dominator_num))
    if this_dominator == self.dominator_num:

        self.dominator_time = self.dominator_time + 1
        #print("YESSS" + str(self.dominator_time))
    else:
        #print("AHHHHH")
        self.dominator_num = this_dominator
        self.dominator_time = 1
    #print("time", str(self.dominator_time))

if self.dominator_time > thresh_dominator_counts:
    next_index = self.page_transition[self.dominator_num]

# string
if next_index < 0:
    if len(str_display) > 0:
        str_display = str_display[:-1]
    elif next_index == 0 and self.page_index is not 0 and self.dominator_num
is not 6:
        str_display = str_display + self.add_key[self.dominator_num]

# color array
if self.whe_initialized == 0:
    if current_look_at == 4:

```

```

        self.color_array = [0, 0, 0, 0, 3, 0]
    else:
        self.color_array = [0, 0, 0, 0, 1, 0]
    else:
        self.color_array = [0, 0, 0, 0, 2, 0]
        if current_look_at is not 4 and current_look_at is not 6:
            self.color_array[current_look_at] = 3
        if self.dominator_num is not -1 and self.dominator_num is not 6:
            self.color_array[self.dominator_num] = 2

    if next_index != self.page_index:
        #print("Is this?")
        self.color_array = [0, 0, 0, 0, 1, 0]
        self.dominator_num = -1
        self.dominator_time = 0
        self.whe_initialized = 0
        break

    if next_index < 0:
        self.display(self.color_array, str_display, self.strings, frame)
    else:
        self.display(self.color_array, str_display, self.strings, frame)

    #cv2.putText(frame, str(current_look_at), (90, 130),
cv2.FONT_HERSHEY_DUPLEX, 0.9, (147, 58, 31), 1)

```



```
        #cv2.putText(frame, str(self.dominator_num), (90, 165),
cv2.FONT_HERSHEY_DUPLEX, 0.9, (147, 58, 31), 1)
```

```
    # cv2.waitKey(2)
```

```
    # cv2.imshow("demo", frame)
```

```
    if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
        self.whe_break = True
```

```
        break
```

```
    if next_index < 0:
```

```
        return 0, str_display, self.whe_break
```

```
    else:
```

```
        return next_index, str_display, self.whe_break
```

```
def initialization(self):
```

```
    list_name = ['up_left', 'up_center', 'up_right', 'left', 'center', 'right', ]
```

```
    iter_name = list_name.__iter__()
```

```
    name = iter_name.__next__()
```

```
    data_vertical = []
```

```
    data_horizontal = []
```

```
    dict_vertical = { }
```

```
    dict_horizontal = { }
```

```
    tot_L = 50 # 150
```

```
    std_horizontal = 0.3 # 0.07
```

```
    std_vertical = 0.6 # 0.2
```

```
tempCounter = 0
```

```
whe_sleep = 1
```

```
while True:
```

```
    temp_color_arr = [0, 0, 0, 0, 0, 0]
```

```
    temp_color_arr[tempCounter] = 2
```

```
    temp_string_arr = ["", "", "", "", "", ""]
```

```
    temp_string_arr[tempCounter] = "Look\nHere"
```

```
    # self.display(temp_color_arr, "", temp_string_arr, frame)
```

```
    if whe_sleep == 1:
```

```
        time.sleep(2)
```

```
        whe_sleep = 0
```

```
    _, frame = self.webcam.read()
```

```
    # We send this frame to GazeTracking to analyze it
```

```
    self.gaze.refresh(frame)
```

```
    frame = self.gaze.annotated_frame()
```

```
    # cv2.imshow("initialization", frame)
```

```
    self.display(temp_color_arr, "", temp_string_arr, frame)
```

```
    if not (self.gaze.is_blinking is not None and self.gaze.is_blinking()) \
```

```
        and self.gaze.horizontal_ratio is not None and \
```

```

        self.gaze.blinking_ratio is not None:
    if len(data_horizontal) >= tot_L:
        data_horizontal.pop(0)
        data_vertical.pop(0)
    data_horizontal.append(self.gaze.horizontal_ratio)
    data_vertical.append(self.gaze.blinking_ratio)

    # start next name
    if len(data_vertical) >= tot_L and np.std(data_vertical) < std_vertical and
len(
        data_horizontal) >= tot_L and \
        np.std(data_horizontal) < std_horizontal:
    dict_horizontal[name] = tuple(data_horizontal)
    dict_vertical[name] = tuple(data_vertical)
    # print(name, np.std(data_horizontal), np.std(data_vertical))
    data_horizontal = []
    data_vertical = []
    try:
        name = iter_name.__next__()
        tempCounter += 1
        whe_sleep = 1
    except:
        break

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

```

```

x = []
y = []
tempI = 0
for i in list_name:
    x = dict_horizontal[i]
    y = dict_vertical[i]
    xy = np.concatenate((np.array(x).reshape(tot_L, 1),
np.array(y).reshape(tot_L, 1)), axis=1)
    if tempI == 0:
        data = xy
        tempI = 1
    else:
        data = np.concatenate((data, xy), axis=0)

y = np.zeros((6 * tot_L, 1))
for i in range(6):
    y[i * tot_L:(i + 1) * tot_L] = i
y = np.ravel(y)
clf = SVC(decision_function_shape='ovo')
clf.fit(data, y)

return clf

def _create_new_image(self, color):
    im1 = np.ones((self.size, self.size, 3), np.uint8)

```

```
return im1 * colors[color]
```

```
def put_string(self, img, string):
```

```
    interval = 29
```

```
    iterr = iter(string)
```

```
    idx = 0
```

```
    tempory = "
```

```
    dy = 50
```

```
    d = 0
```

```
    while(True):
```

```
        try:
```

```
            a = iterr.__next__()
```

```
            tempory = tempory + a
```

```
            idx += 1
```

```
            if len(tempory) % interval == 0 and idx != 0:
```

```
                d += dy
```

```
                cv2.putText(img, tempory, (0, d), cv2.FONT_HERSHEY_DUPLEX,
```

0.9,

```
                    (147, 58, 31), 1)
```

```
                tempory = "
```

```
            except:
```

```
                d += dy
```

```
                cv2.putText(img, tempory, (0, d), cv2.FONT_HERSHEY_DUPLEX,
```

1.2,

```
                    (147, 58, 31), thickness=4)
```

```
break
```

```
return img
```

```
def display(self, color, string, text, frame):
```

```
    im = [0, 1, 2, 3, 4, 5]
```

```
    im[0] = self._create_new_image(color[0])
```

```
    im[1] = self._create_new_image(color[1])
```

```
    im[2] = self._create_new_image(color[2])
```

```
    im[3] = self._create_new_image(color[3])
```

```
    im[4] = self._create_new_image(color[4])
```

```
    im[5] = self._create_new_image(color[5])
```

```
    dy = 50
```

```
    for idx in range(6):
```

```
        if idx == 4:
```

```
            im[idx] = self.put_string(im[idx], string)
```

```
        else:
```

```
            for i, line in enumerate(text[idx].split('\n')):
```

```
                # print(idx)
```

```
                cv2.putText(im[idx], line, (int(self.size/3), int(self.size/3) + i * dy),  
cv2.FONT_HERSHEY_SIMPLEX, 1.2, (147, 58, 31), thickness=4)
```

```
partial1 = np.concatenate((im[0], im[3]))
```

```
partial2 = np.concatenate((im[1], im[4]))
```

```
partial3 = np.concatenate((im[2], im[5]))
```

```
image = np.concatenate((partial1, partial2), axis=1)
```

```
image = np.concatenate((image, partial3), axis=1)
```

```
image = np.array(image, np.uint8)
```

```
image = cv2.line(image, (self.size, 0), (self.size, self.size * 2), color=[0, 0, 0],  
thickness=3)
```

```
image = cv2.line(image, (2 * self.size, 0), (2 * self.size, self.size * 2),  
color=[0, 0, 0], thickness=3)
```

```
image = cv2.line(image, (0, self.size), (self.size * 3, self.size), color=[0, 0, 0],  
thickness=3)
```

```
h, w, _ = frame.shape
```

```
# print(frame.shape)
```

```
image = cv2.resize(image, (w, h))
```

```
frame = frame[:, ::-1, :]
```

```
image = np.concatenate((image, frame), axis=1)
```

```
# print(image.dtype)
```

```
self.out.write(np.array(image))
```

```
cv2.waitKey(2)
```

```
cv2.imshow('test image', image)
```

```
# class Page:
```

```
#
```

```
# def __init__(self, index, gaze, webcam, size=400):
```

```
#     self.page_index = index
#     self.dominator_num = -1
#     self.dominator_time = -1
#     self.whe_initialized = 0
#     self.gaze = gaze
#     self.webcam = webcam
#     self.size = size
#
#     def run(self):
#         pass
#
#     def _create_new_image(self, color):
#         im1 = np.ones((self.size, self.size, 3), np.uint8)
#         return im1 * colors[color]
#
#     def put_string(self, img, string):
#         interval = 29
#         iterr = iter(string)
#         idx = 0
#         tempory = "
#         dy = 50
#         d = 0
#         while(True):
#             try:
#                 a = iterr.__next__()
```



```

#
#         tempory = tempory + a
#         idx += 1
#         if len(tempory) % interval == 0 and idx != 0:
#             d += dy
#             cv2.putText(img, tempory, (0, d),
cv2.FONT_HERSHEY_DUPLEX, 0.9,
#                 (147, 58, 31), 1)
#             tempory = "
#         except:
#             d += dy
#             cv2.putText(img, tempory, (0, d), cv2.FONT_HERSHEY_DUPLEX,
0.9,
#                 (147, 58, 31), 1)
#         break
#
#     return img
#
# def display(self, color, text, string):
#     im = [0, 1, 2, 3, 4, 5]
#     im[0] = self._create_new_image(color[0])
#     im[1] = self._create_new_image(color[1])
#     im[2] = self._create_new_image(color[2])
#     im[3] = self._create_new_image(color[3])
#     im[4] = self._create_new_image(color[4])
#     im[5] = self._create_new_image(color[5])

```

```

#     dy = 50
#     for idx in range(6):
#         if idx == 4:
#             im[idx] = self.put_string(im[idx], string)
#         else:
#             for i, line in enumerate(text[idx].split('\n')):
#                 print(idx)
#                 cv2.putText(im[idx], line, (int(self.size/3), int(self.size/3) + i * dy),
cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
#
#     partial1 = np.concatenate((im[0], im[3]))
#     partial2 = np.concatenate((im[1], im[4]))
#     partial3 = np.concatenate((im[2], im[5]))
#
#     image = np.concatenate((partial1, partial2), axis=1)
#     image = np.concatenate((image, partial3), axis=1)
#     image = np.array(image, np.uint8)
#
#     image = cv2.line(image, (self.size, 0), (self.size, self.size * 2), color=[0, 0,
0])
#     image = cv2.line(image, (2 * self.size, 0), (2 * self.size, self.size * 2),
color=[0, 0, 0])
#     image = cv2.line(image, (0, self.size), (self.size * 3, self.size), color=[0, 0,
0])
#
#     cv2.imshow('test image', image)
#     cv2.waitKey(0)

```

```

# cv2.destroyAllWindows()
#
#
#
# page = Page(1, 2, 3)
# text = ['asdf\nsdfas', 'asdfas', "asdfas\nsd\nasdf", "asdf", "asd\nk", "asdf\n"]
# color = ["yellow", "green", "gray", "white", "yellow", "yellow"]
# Page.display(page, color, text, 'stringstringstringstringstringstringstring')

```

9.6 Pupil.py:

```

import numpy as np
import cv2

```

```

class Pupil(object):
    """
    This class detects the iris of an eye and estimates
    the position of the pupil
    """

    def __init__(self, eye_frame, threshold):
        self.iris_frame = None
        self.threshold = threshold
        self.x = None
        self.y = None

```

```
self.detect_iris(eye_frame)
```

```
@staticmethod
```

```
def image_processing(eye_frame, threshold):
```

```
    """Performs operations on the eye frame to isolate the iris
```

Arguments:

eye_frame (numpy.ndarray): Frame containing an eye and nothing else

threshold (int): Threshold value used to binarize the eye frame

Returns:

A frame with a single element representing the iris

```
    """
```

```
    kernel = np.ones((3, 3), np.uint8)
```

```
    new_frame = cv2.bilateralFilter(eye_frame, 10, 15, 15)
```

```
    try:
```

```
        new_frame = cv2.erode(new_frame, kernel, iterations=3)
```

```
    except:
```

```
        pass
```

```
    new_frame = cv2.threshold(new_frame, threshold, 255,
cv2.THRESH_BINARY)[1]
```

```
    return new_frame
```

```
def detect_iris(self, eye_frame):
```

```
    """Detects the iris and estimates the position of the iris by
```

calculating the centroid.

Arguments:

eye_frame (numpy.ndarray): Frame containing an eye and nothing else

"""

self.iris_frame = self.image_processing(eye_frame, self.threshold)

contours, _ = cv2.findContours(self.iris_frame, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)[-2:]

contours = sorted(contours, key=cv2.contourArea)

try:

moments = cv2.moments(contours[-2])

self.x = int(moments['m10'] / moments['m00'])

self.y = int(moments['m01'] / moments['m00'])

except (IndexError, ZeroDivisionError):

pass

References

[1] Dutta, B. Bhushan, G. Prasad, H. Cecotti, Y. K. Meena (2019) 'A multiscrypt gaze-based assistive virtual keyboard' in the 41st Annual International Conference of the IEEE, Vol. 8, No. 10, pp. 1306-1309.

[2] Ashlesha Singh, Chandrakant Chandewar, Pranav Pattarkine (2018) 'Driver Drowsiness Alert system with Effective Feature Extraction' in International Journal for Research in Emerging Science and Technology, Vol 5, No.1, pp. 26-31.

[3] Chairat Kraichan, Suree Pumrin (2014) 'Face and Eye Tracking for

Controlling Computer Functions’ in 11th International Conference on Electrical Engineering, Computer, Telecommunications and Information Technology, Vol.37, No.19, pp. 1-6.

[4] Farida Mohamed, Haya Ansari, Maryam Mohamed, Mohamed Nasor and Mujeeb Rahman (2007) ‘Eye-Controlled Mouse Cursor for Physically Disabled Individual’ in IEEE Intelligent Systems, Vol.11, No. 6, pp. 18-24.

[5] H. Cecotti (2016) ‘A Multimodal Gaze-Controlled Virtual Keyboard’ in IEEE Transactions on Human-Machine Systems, Vol. 46, No. 4, pp.

601-606.

[6] Josephine Sullivan, Vahid Kazemi (2014) ‘One millisecond face alignment with an ensemble of regression trees’ in Proceedings of the IEEE conference on computer vision and pattern recognition, Vol. 69, No. 2, pp. 1867-1874.

[7] Mohamad Hassrol Bin Mat, Rasheed Nassr, Sara Bilal (2018) ‘Design a Real-Time Eye Tracker’ in Proceedings of the 2nd International Conference on Video and Image Processing, Vol. 42, No. 7, pp. 187–191.

[8] Qurban Ali Memon, Zeenat Al-Kassim (2017) ‘Designing a low-cost eyeball tracking keyboard for paralyzed people’ in Computers and Electrical Engineering, Vol. 58, No. 11, pp. 20–29.

[9] R. Sigit, T. Harsono, V. I. Saraswati (2016) ‘Eye gaze system to operate virtual keyboard’ in International Electronics Symposium (IES), Vol. 16, No. 5, pp. 175-179.

[10] Wang Jun, Wen Jing,, Zhang Naizhong (2015) ‘Hand-Free Head Mouse Control Based on Mouth Tracking’ in the 10th International

Conference on Computer Science and Education, Vol. 45, No. 7, pp.

707-713.