CHAPTER 1

INTRODUCTION

Paralysis is the loss of muscle function in the body. Paralysis is sometimes temporary and in a few cases, it is permanent. Paralysis is not limited to any particular part of the body, but most cases of paralysis are observed in limbs. Partial and complete paralysis can occur at any point in time. A patient suffering from paralysis is not inflicted by any pain upon occurrence. Depending on the nature of the underlying cause, the treatment plan is charted either to cure or to treat the condition to ensure that the daily lifestyle of the patient is not drastically affected. A stroke is the most common underlying condition that triggers partial or complete paralysis in a patient. In partial paralysis, the patient is still in partial control of the affected muscle, incomplete paralysis, the patient has no control over the affected muscle tissue.

TYPES OF PARALYSIS

1) Monoplegia

Monoplegia is a type of paralysis where the person loses control over one limb. Often the paralysis is limited to a single-arm, sometimes even limited to certain muscles in an arm. Monoplegia is often the side effect of cerebral palsy. In cerebral palsy, the brain loses the ability to control certain muscles of the body. Infants and early adolescents are mostly affected by cerebral palsy. Monoplegia is considered a good sign in the prognosis of this syndrome as the paralysis is limited to a single limb and the patient can still work around his daily activities with relative ease.

2) Hemiplegia

Hemiplegia is a type of paralysis where a person loses control of one side of the body. The effect of this is generally limited to one arm and one limb and sometimes the effect is also seen in the torso region. Hemiparesis is a condition when the person does not lose the complete functionality of their limbs but their functionalities are significantly reduced in terms of strength and endurance. Hemiparesis evolves into hemiplegia in a few cases.

Hemiplegia is caused due to injuries to the spinal cord or when the left and right side of the brain are not communicating properly through the corpus callosum. Hemiplegia is also caused by a stroke, which affects the functioning of one side of the brain.

Patients suffering from hemiplegia may find it difficult to speak and resume normal motor functions. They experience total or partial loss of sensation in one side of the body and this leads to Pusher’s syndrome where the patient shifts the weight of the paralysed side to the normal side, thereby adversely affecting motor functions.

3) Paraplegia

Paraplegia is a form of paralysis where the patient loses control over the muscles from the waist down. The effects of paraplegia vary from person to person. People going through paraplegia have perfectly healthy legs and the underlying cause of this condition mostly originates in the brain or spinal cord. Paraplegia is sometimes complete and sometimes only affects one limb in the lower torso. Partial paraplegia is often a result of consistent physiotherapy and medication. Paraplegia is often caused after the patient experiences an injury in the brain, or the spinal cord or both. In paraplegic patients, the signals that are sent to the lower part of the body from the brain are not sent back up to the brain through the spinal cord. This lack of communication with the brain, causes the patients to not only lose motor functions but also results in loss of sensation..

In partial or incomplete paraplegia, the patient can retain functionalities of a single leg while in complete paraplegia, the patient loses sensation and functionalities of both the legs. Car accidents, sports-related injuries, side effects to surgeries, spinal cord injuries, violence, and falls are the major causes of paraplegia..

4) Quadriplegia

A quadriplegia is a form of paralysis where all four limbs are affected. In this condition, the signals that are sent from the brain to the regions below the neck are not sent back, thereby causing impalement in the hands and legs of the patient. The spinal cord is charged with the job of sending signals to and receiving signals from the brain. In quadriplegia, this ceases to function, thereby causing the condition. Injuries to the spinal cord or the brain can trigger the onset of quadriplegia.

Patients suffering from quadriplegia experience fatigue, loss of sensation in the region below the neck, sudden spasms, difficulty in clearing urine from the body, respiratory distress, bedsores, and depression among other side effects.

5) Locked-in Syndrome

Locked-in Syndrome is a form of Paralysis where the patient is not able to control any part of their body from under the eyes. Locked-in syndrome is majorly a side-effect of a major injury to the brain or a stroke or cancer in the brain. A person affected by locked-in syndrome will not be able to move his lips, his jaws, no up & down or side to side moments in the neck or move any other limb of his body. However, people suffering from locked-in syndrome can blink and move their eyeballs up and down. Locked-in syndrome often mimics the symptoms of a coma and doctors rely on the moment of the eyes to make a perfect diagnosis.

OVERVIEW

In these days electronic devices are improving day by day and their demand is also improving. Smart phones, tablets are example of this. The system detects the eye blink and differentiates between an intentional long blink and a normal eye blink. Tetraplegia is a condition where people cannot move parts below neck. The proposed system can be used to control and Communicate with other people. In the recent years due to the rapid advancement in the technology there has been a great demand of human computer or mobile interaction (HCI or HMI). Eye blink is a quick action of closing and opening of the eyelids. Blink detection is an important enabling component in various domains such as human computer interaction, mobile interaction, health care, and driving safety. For example, blink has been used as an input modality for people with disabilities to interact with computers and mobile phones.

In Viola the chain of single-feature filters, Haar Cascade Classifier for identifying sub-region image is used. With the fast calculation of integral image technique, it can work in real time. Eye tracking provides an almost seamless form of interaction with the modern graphical user interface, representing the fastest non-invasive method of measuring user interest and attention. While the mouse, keyboard, and other touch-based interfaces have long reigned as the primary mediums associated with the field of human computer interaction, as advances continue to improve the cost and accuracy of eye tracking systems, they stand poised to contend for this role. An open and close eye template for blink pattern decisions based on correlation measurement is used in. The method was specifically useful for people with severely paralyzed. A real-time eye blinking detection was proposed based on SIFT feature tracking with GPU based implementation.

The method is based on image processing techniques for detecting human eye blinks and generating intereye blink intervals A Haar Cascade Classifier and Camshaft algorithms for face tracking and consequently are applied for getting facial axis information. Adaptive Haar Cascade Classifier from a cascade of boosted classifiers based on Haar-like features using the relationship between the eyes and the facial axis applied for positioning the eyes. The algorithm results show that the proposed method can work efficiently in real-time applications. An EyePhone application which is developed in is a system that capable of driving mobile applications/functions using only the user's eyes movement and actions (e.g. Wink). EyePhone tracks the user's eye movement across the phone's display using the camera mounted on the front of the phone. The results indicate that EyePhone is a promising approach to driving mobile applications in a hand-free manner.

An efficient eye tracking system is presented in having a feature of blink detection for controlling an interface that provides an alternative way of Communication for the people who are suffering from severe physical disabilities the proposed system uses pupil portion for tracking the movement of eyes.

CHAPTER 2

LITERATURE SURVEY

P. A. A. Mohammed, ‘‘Effificient eye blink detection method for disabled-helping domain,’’ Eye, vol. 10, no. (P1), p. P2, 2014- In this method, they have designed a real-time interactive system that can help the paralyzed to control the appliances such as fan, lights etc. through the pre- recorded sample of eyes blinking. Detection of eyes blinking by using video camera with region of interest.51

K. Mukherjee and D. Chatterjee, ‘‘Augmentative and alternative communication device based on eye-blink detection and conversion to morsecode to aid paralyzed individuals,’’ in Proc. Int. Conf. Commun., Inf. Comput. Technol. (ICCICT), Jan. 2015, pp. 1–5-In this study the motivation of this research is those people are physically disable or who cannot control the human mobile calls for interaction without using hands. For eyes and face detection they have used the haar cascade classifier. There are several medical disorders that can lead to an individual becoming paralyzed or having motor speech disorders that inhibits speech or voice production. Conditions such as Locked-In Syndrome (LIS) or motor neuron diseases such as Amyotrophic Lateral Sclerosis (ALS) and Cerebral Palsy are among the common diseases that affect speech. In all or most such cases, the patient loses the ability to communicate with the external world in an effective manner even though his intelligence is mostly unaffected. Not only does that cause extreme distress to that individual, but also to his family and friends. Some customized Augmentative and Alternative Communication (AAC) devices have been developed that uses signals from the patient and converts them into some form of data that can be communicated but such devices are very expensive and are practically out of reach for most people. In this document, we have designed an extremely low-priced device that reads and converts eye-blinks from the patient to a universally accepted communication code-The Morse code.

Atish udayshankar, Amit R Kaushik”Assistance for the Paralyzed using Eye Blink Detection” || IEEE paper Fourth International Conference on Digital Home 2012-In this system, the device that uses the signals from patient and convert it into some form of data that is for communication but this system is very expensive, so that they have developed extremely low priced device that read and convert eyes blinking of patient to universally accepted code that is morse code. 2.2 Using eyes blink detection assistance for the paralyzed person.

Królak and P. Strumiłło ‘‘Eye-blink detection system for human– computer interaction,’’ Universal Access Inf. Soc., vol. 11, no. 4, pp. 409–419, 2012.In this paper, a vision-based system for detection of voluntary eye-blinks is presented, together with its implementation as a Human–Computer Interface for people with disabilities. The system, capable of processing a sequence of face images of small resolution (320 × 240 pixels) with the speed of approximately 30 fps, is built from off-the-shelf components: a consumer-grade PC or a laptop and a medium quality webcam. The proposed algorithm allows for eye-blink detection, estimation of the eye-blink duration and interpretation of a sequence of blinks in real time to control a non-intrusive human–computer interface. The detected eye-blinks are classified as short blinks (shorter than 200 ms) or long blinks (longer than 200 ms). Separate short eye-blinks are assumed to be spontaneous and are not included in the designed eye-blink code.36

S. He and Y. Li, ‘‘A Single-channel EOG-based Speller’’ IEEE Trans. Neural Syst. Rehabil. Eng., vol. 25, no. 11, pp. 1978–1987, Nov. 2017. In this study, we propose a single-channel EOG-based asynchronous speller. Forty buttons corresponding to English letters, numerical digits and punctuation marks were displayed on a graphical user interface (GUI), and the buttons alternatively flashed in a random order (each flash corresponded to a button). The user could select a button by blinking in synchrony with the stimulus onset, or flash, of the desired button. A detection algorithm combining support vector machine (SVM) classification and waveform detection was used to detect the blink and to determine the target button.27

A novel biometric approach for human identifification and verifification using eye blinking Signal-In this paper, a novel technique is adopted for human recognition based on eye blinking waveform extracted from electro-oculogram signals. For this purpose, a database of 25 subjects is collected using Neurosky Mindwave headset. Then, the eye blinking signal is extracted and applied for identification and verification tasks. The pre-processing stage includes empirical mode decomposition to isolate electro-oculogram signal from brainwaves. Then, time delineation of the eye blinking waveform is utilized for feature extraction. Finally, linear discriminant analysis is adopted for classification. Based on the achieved results, the proposed system can identify subjects with best accuracy of 97.3% and verify them with an equal error rate of 3.7%. The obtained results in this paper confirm that eye blinking waveform carries discriminant information and is therefore appropriate as a basis for human recognition task.1

R. Ahmad and J. N. Borole, ‘‘Drowsy driver identifification using eye blink detection,’’ Int. J.Comput. Sci. Inf. Technol., vol. 6, no. 1, pp. 270–274, Jan. 2015. In this study, we present a low cost and fully automatic solution for handling the drowsy driver detection problem. Our system uses a standard webcam and detects the pattern of long duration eye-lid closures. The eye blink duration is the time spent while upper and lower eye-lids are connected. The pattern indicates a potential drowsiness prior to the driver falling asleep and then alerts the driver by voice messages.

K. Grauman, M. Betke, J. Gips, and G. R. Bradski, ‘‘Communication via eye blinks-detection and duration analysis in real time,’’ in Proc. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), vol. 1, Dec. 2001, pp. 1–2. propose a robust, accurate algorithm to detect eye blinks, measure their duration, and interpret them in real time to control a non-intrusive interface for computer users with severe disabilities. The method presented employs visual information about the motion of eyelids during a blink and the changing appearance of the eye throughout a blink in order to detect the blink’s location and duration. Moreover, the system is designed to initialize itself automatically, and it adjusts for changes in the user’s position in depth. Using the ”Blink Link,” as our prototype system is called, a user who is capable of blinkixig voluntarily can generate mouse clicks through his or her eye blinks in order to operate software applications requiring such input. The system uses various computer vision techniques in combination. Eye blink motion is used to automatically locate the user’s eyes in the video sequence. In particular, candidate motion patterns are compared against a stored model of the properties of actual eye blink motion in order to eliminate motion that is unlikely to have resulted from blinks. The location information gained from the blink motion then offers an opportunity to select an eye template online for further tracking. The correlation between the open eye template and the current eye in the scene reveals the extent of the eye’s openness which, together with the complementary motion information obtained from both eye areas, allows us to classify the eye as either open or closed at each frame. An array of techniques have been explored previously for locating eyes in images and eye blink detection. Methods for detecting the eyes include the use of gradient flow fields ill, color-based techniques for detection of the eye sclera, horizontal gradient maps of a skin-colored region, and pupil detection using infrared lighting . Temporal differencing is often used to segment moving regions of interest from a stable background . Methods for analyzing the eye and its closure motion are suggested . A blink detector has been developed to detect drowsy drivers. Preliminary work on facial feature tracking has been reported that provides video-based interfaces for people with disabilities. We are not aware of any papers that address the issues of communication interfaces which op erate on eye blinks. Such interfaces demand the robust and accurate classification of voluntary and involuntary blinks, must work with assistive technology software, and require exceptionally fast processing.

CHAPTER 3

PROBLEM STATEMENT

To develop our everyday activities we have to interact with the environment using our communicative abilities (oral, written or gestural). Paralyzed people lack the ability to control muscle function in one or more muscle group. Fortunately, Augmentative and Alternative Communication (AAC) technologies have promoted the integration of this group of people by converting their capabilities into a communicative action. Some AAC devices rely on transforming users intentions into commands sent to a computer. For people with severe disabilities, the most typical computer interaction is by means of a device that generates a binary on/off signal, to select a highlighted ideogram during the scanning of the elements contained on a communication board. The physicist Stephen Hawking, for example, controlled a scanning-based communication program (ACAT) s via his cheek movements detected by an Infrared (IR)-based device placed on his spectacles.

Several diseases, like Amyotrophic Lateral Sclerosis (ALS), Locked-in Syndrome (LIS), etc., Locked in Syndrome (LIS) is a form of paralysis where a patient has lost in control of nearly all voluntary muscles. These people are unable to control any part of their body, besides eye moment and blinking. Due to their condition these people are unable to talk, text, and communicate in general. Even through people that have LIS are cognitively aware their thoughts and . These people depend on eye blinks to communicate. They rely on nurse and caretakers to interpret and decode their blinking. Whenever patients do not have a person to read their Eyeblink, they have no means of self expression.

3.1 EXISTING SYSTEM

Several AAC solutions exist to facilitate communication in these cases. Eye-Tracker Interfaces (ETI) can be used by people with good eye gaze control. Most of these are based on the reflection of an IR light on the surface of the eye. A camera tracks the pupil and the shiny IR spot in the eye. The relative position between them serves to determine the eye gaze after an initial calibration process.

Disadvantages of ETI

- The cost of the overall setup is very high

- The so-called Midas touch effect which consists of the random selection of an icon on the computer screen followed by the user's gaze. This is a problem when the user is not sure of the icon's meaning and requires a longer time to understand and mentally process the action related to it
- Contact lenses, glasses and pupil color can all impact the eye tracking cameras ability to record eye movements

Brain Computer Interface (BCI) is another plausible solution that does not need precise BCI depending on how brain activity is measured, the type of information extracted and the nature of the stimuli. The scientifific literature includes plenty of BCI systems measuring electrical cortical activity through electrodes placed on the scalp. There are several types of BCI. Amongst which we would like to highlight: Slow cortical potential (SCP), Motor Imagery (MI), Steady state visual potential (SSVEP) and P300. SSVEP and P300 are evoked potentials that differ from the other two in that an external stimulus is necessary. The stimulus in SSVEP is visual whereas in P300 it can be visual, auditory or tactile. The main drawback with this technology is the cost of the equipment, the training time for some BCI modalities and the technical knowledge needed by the caregivers to setup the system. Nevertheless, inexpensive BCI, such as NeuroSky Mindwave (NM), Muse and Emotiv, have recently been released. The first is the cheapest, with one electrode, followed by Muse, with 4 channels.and finally Emotiv with up to 14 channels.

Disadvantages of BCI

- The cost of the equipment required in this kind of system.
- The training time for some BCI modalities is very high.
- The technical knowledge needed by the caregivers to setup the system is also immense.

3.2 PROPOSED SYSTEM

As said earlier the patients lose the ability to speak and write they can only contact the outside world through human-computer interaction; e.g. controlling brain waves or tracking eye movements. Currently, brain wave controlling devices need to be worn by users, so they are not convenient for people to use. There exists eye-motion based software which enables the tetraplegic patients to write in the computer by using their eye functions only. When they are away from PC and lie on the bed, they cannot communicate with care providers. With the goal of helping tetraplegic patients on the bed to call for other people with a simple and easy approach, this research aims to develop a real time video processing system, which can successfully detect the eye blinks regardless the head directions, day or night.

BlinkToSpeak offers a form of independence to paralyzed people. The software platform converts eye blinks to Speak. Every feature of the software can be controlled by eye movement. Thus, the software can be independently operated by paralyzed people. Using the software, patients can record messages, recite those messages aloud, and send the messages to others. The software can be run on any low-end computer, from a Raspberry Pi to an IBM ThinkPad. The software uses computer vision and Haar cascades to detect eye blinking and convert the motion into text. The program uses language modelling to predict the next words that the user might blink. The software can be easily customized for each patient as well. Blink to Text is free open source software. It is distributed under the MIT Permissive Free Software License.

The BlinkToSpeak also offers the payients to access certain home appliancein case their caretaker ornurse is not present at that moment. We can modify the system and connect the system to appliances. So that once the system detects pattern of blinks it switches on the fan,light bulb etc, Any extra appliances can be connected to the arduino board.

Facial Landmarks

Eye blinks can be detected by referencing significant facial landmarks. Many software libraries can plot significant facial features within a given region of interest. Python’s dlib library uses Kazemi and Sullivan’s One Millisecond Face Alignment with an Ensemble of Regression Trees to implement this feature. The program uses a facial training set to understand where certain points exist on facial structures. if they exist, the program plots the same points on regions of interest in other images. The program uses priors to estimate the probable distance between key points

.

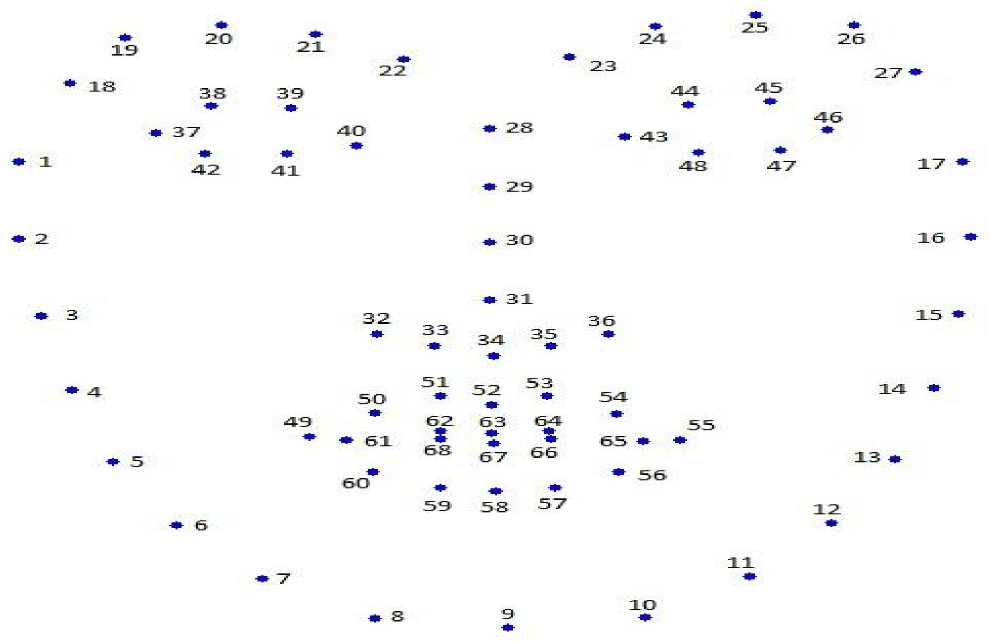


Fig 3.2.1 Facial Landmark

Advantages of the Proposed System

- Efficiency
- Accuracy.
- Electrode less system:
- Faster when compared to other algorithms

3.3 Objectives

- **Allow paralysis victims to communicate independently.**

Many paralysis victims already use eye blinks as a form of communication. It is common for nurses and caretakers to read a patient's eye blinks and decode the pattern. The ALS association even offers a communication guide that relies on eye blinks. BlinkToSpeak automates this task. The software reads a person's eye blinks and converts them into text. A key feature of the software is that it can be started, paused, and operated entirely with eye blinks. This allows patients to record their thoughts with complete independence. No nurses or caretakers are required to help patients express themselves. Not only does this reduce the financial burden on paralysis patients, but this form of independence can be morally uplifting as well.

- **Be accessible to people with financial constraints.**.

Many companies are developing technologies that are controlled by eye movement. These technologies rely on expensive hardware to track a user`s eyes. While these devices can absolutely help LIS victims, they are only available to people that can afford the technology. BlinkToSpeak focuses on a different demographic that are often ignored. Twenty-eight percent of U.S. households with a person who is paralyzed make less than $15,000 per year. BlinkToSpeak is free and open source. The software runs on wide variety of low-end computers. The only required peripheral is a basic webcam. Not only is this software accessible to paralyzed people, but paralyzed people of almost all financial classes as well.

CHAPTER 4

SYSTEM REQUIREMENTS

Hardware Requirements

System	:	Pentium 1V 2.4 GHZ or below
Hard Disk	:	250GB.
Ram	:	2GB
Components	:	Camera,light bulb,power supply,relay,HDMI,system,arduino board,model fan

Software Requirements

Operating System	:	Windows XP/7/Linux
Coding Language	:	Python 3.5.0
IDE	:	Open CV

CHAPTER 5

SYSTEM ARCHITECTURE

System design is the process of the defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements .Systems design could be seen as the application of systems theory to product development. Object- oriented analysis and methods are becoming the most widely used methods for computer systems design. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

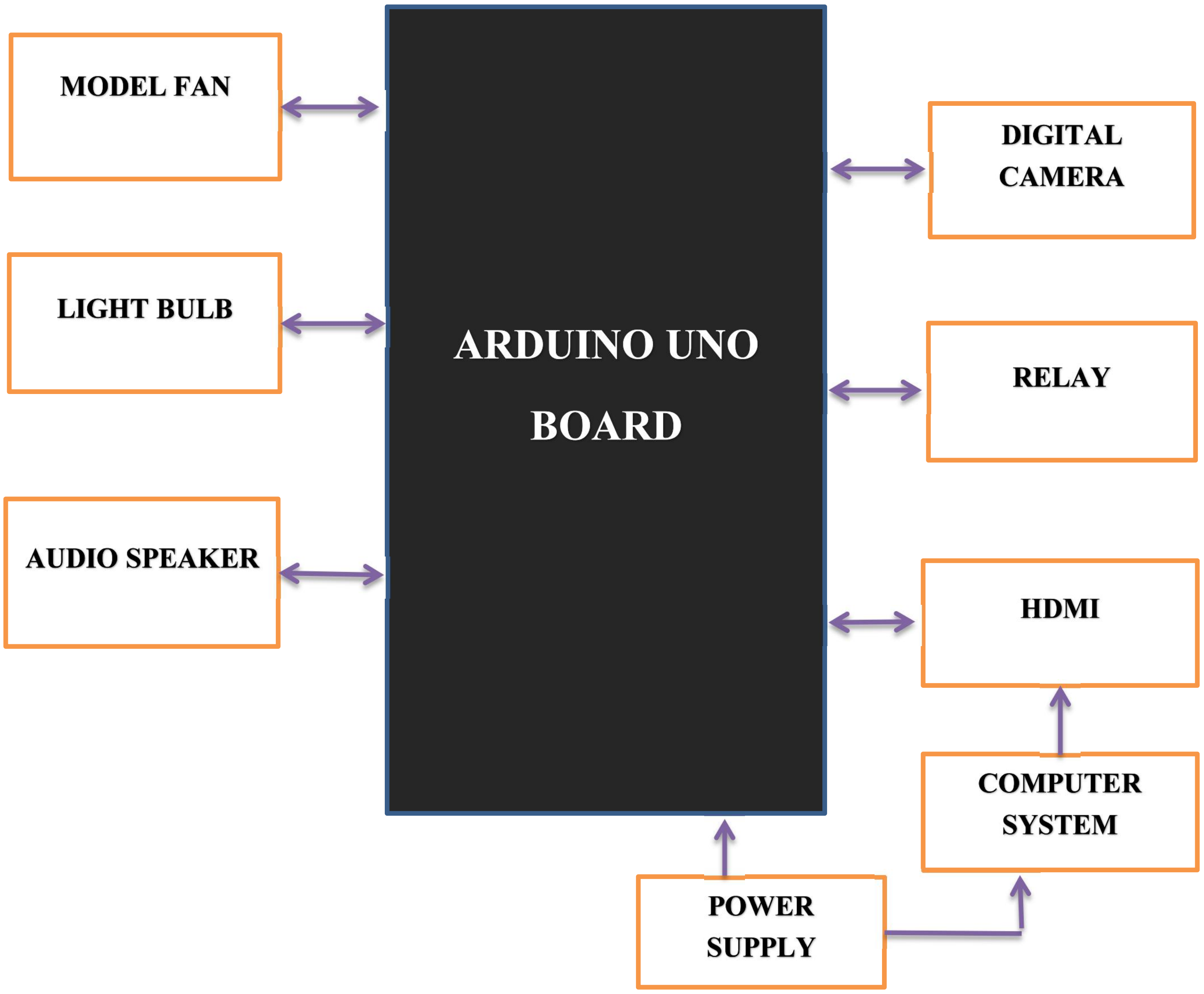


Fig 5.1 System Architecture

1. Arduino Uno Board

Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

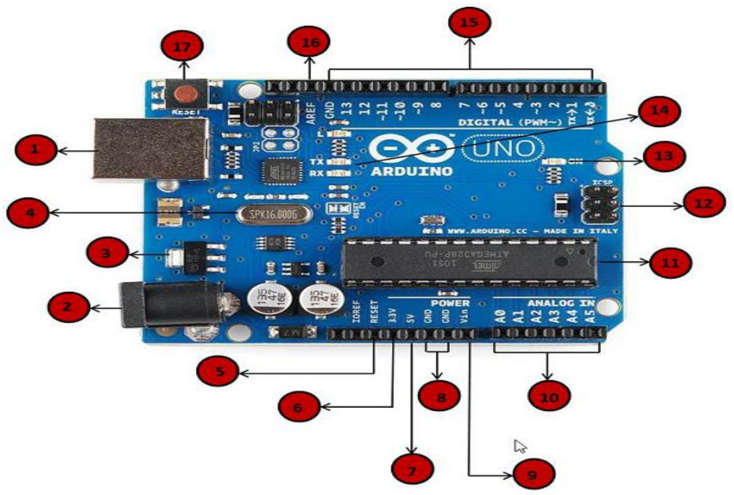


Fig 5.2 Arduino Uno Board

2. Digital Camera

A digital camera is a hardware device that takes photographs and stores the image as data on a memory card. Unlike an analog camera, which exposes the chemicals on film to light, a digital camera uses digital optical components to register the intensity and color of light, and converts it into pixel data.

Fig 5.3 iBall super-view C8.0

3. HDMI

HDMI (High-Definition Multimedia Interface) is a proprietary audio/video interface for transmitting uncompressed video data and compressed or uncompressed digital audio data from an HDMI-compliant source device, such as a display controller, to a compatible computer monitor, video projector, digital television

4. Relay

Relays are switches that open and close circuits electromechanically or electronically. Relays control one electrical circuit by opening and closing contacts in another circuit. As relay diagrams show, when a relay contact is normally open (NO), there is an open contact when the relay is not energized.

5. Audio Speaker

A computer speaker is an output hardware device that connects to a computer to generate sound. The signal used to produce the sound that comes from a speaker is created by the sound card.

Fig 5.4 Audio Speaker

6. Computer System

A computer system is a set of integrated devices that input, output, process, and store data and information. Computer systems are currently built around at least one digital processing device. There are five main hardware components in a computer system: Input, Processing, Storage, Output and Communication devices.

7. Power Supply

A power supply is an electrical device that supplies electric power to an electrical load. The primary function of a power supply is to convert electric current from a source to the correct voltage, current, and frequency to power the load.

8. Light Bulb

The incandescent light bulb turns electricity into light by sending the electric current through a thin wire called a filament. Electrical filaments are made up mostly of tungsten metal. The resistance of the filament heats the bulb. Eventually the filament gets so hot that it glows, producing light.

9. Model Fan

A model fan is designed to circulate air in a home or building. It is sometimes confused with a powered attic ventilator, which exhausts hot air from the attic to the outside through an opening in the roof or gable at a low velocity.

Use Case Diagram

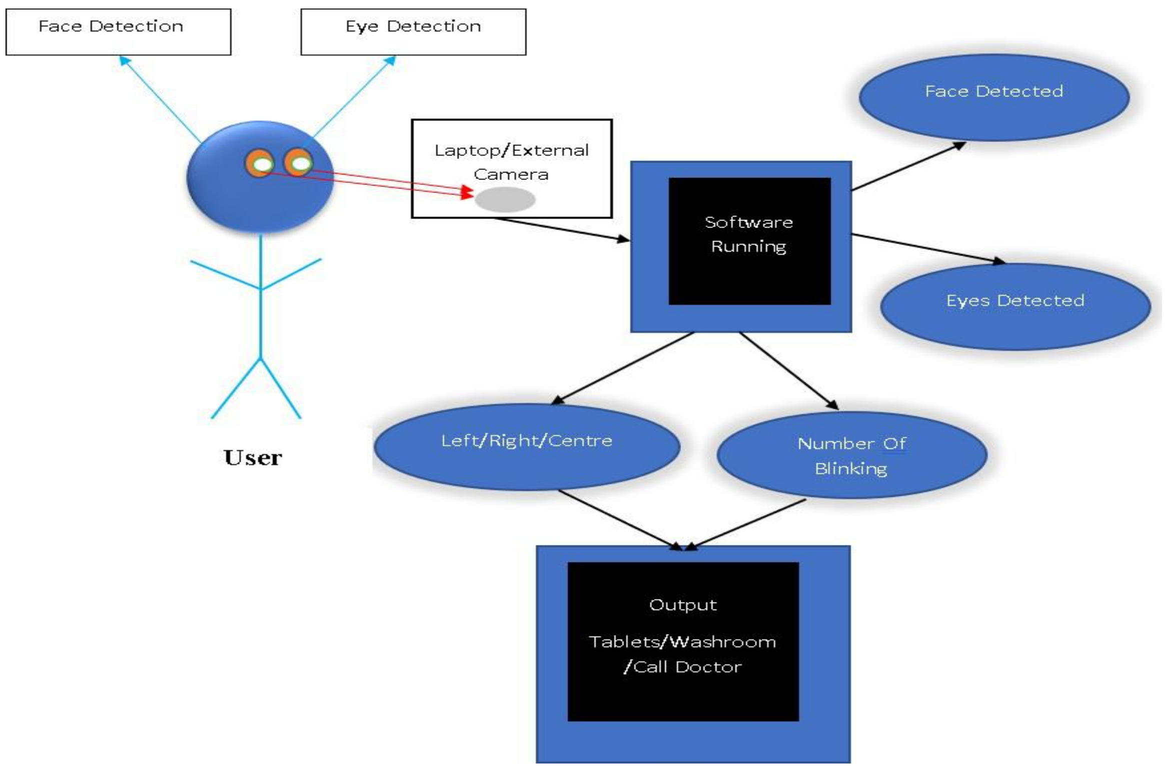


Fig 5.5 Use Case Diagram

Description of use case

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. While a use case itself might drill into a lot of detail about every possibility, a use case diagram can help provide a higher-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system". They provide the simplified and graphical representation of what the system must actually do.

In Fig 5.5 shows use case diagram; the system accomplishes to have the following steps.

1. Open the webcam on the laptop and show the image of a person.
2. Face detection action is performed.
3. The system detects the eyes of a person.
4. After the above action system move on to the next operation.
5. In the next step the system detects eyes and face through webcam of a laptop.
6. The blinking is detected with Right eye and left eye respectively, then display the respected output.

Work Flow Diagram

The design part includes the data flow diagram. The data flow diagram explains the workflow of the system proposed. The work flow mainly highlights the data flow direction that means how the data is being modified? How it is being used? and how the results vary with it?

Steps in Work Flow Diagram

1) Capturing the frame from the video using the system’s camera initialises the execution of the proposed system.
2) The Face Detection Algorithm then processes on the captured video frames to give out the rectangular boxed face. This output from Face Detection Algorithm then gets processed using AdaBoost Classifier to detect the eye region in the face.

3) Eye detected will be sent to check if there is any movement of eyeball.

4) If it's there, then this movement will be tracked to give out the combination the patient is using to express the dialogue.

5) If not, then the blink pattern will be processed to give out the voice as well as the text input with respective dialogue.

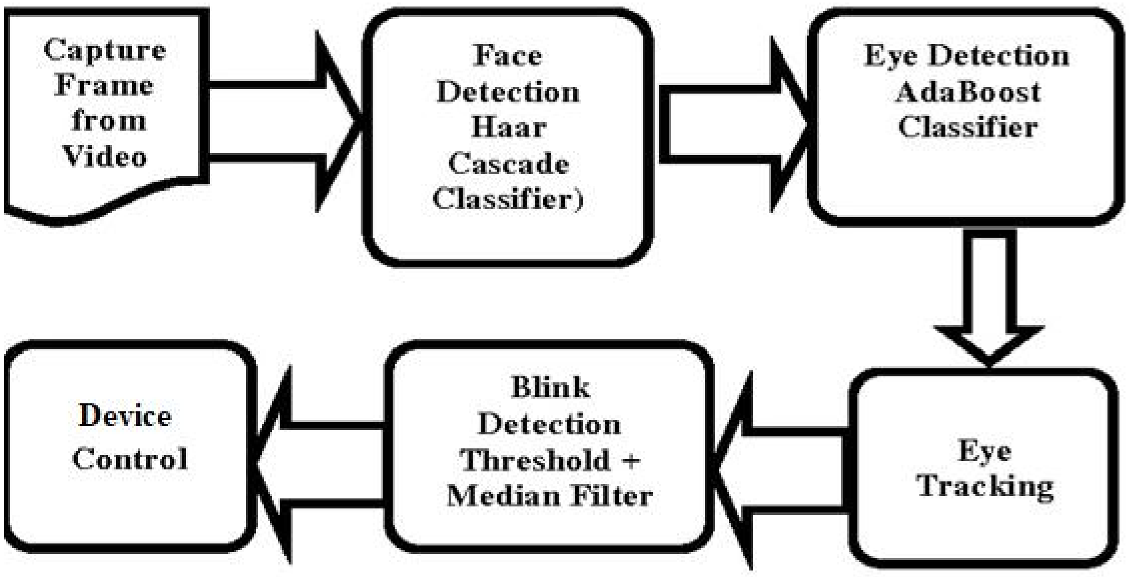

Capture Frame from Video
Face Detection Haar Cascade Classifier)
Eye Detection AdaBoost Classifier
Eye Tracking
Blink Detection Threshold + Median Filter
Device Control

Fig 5.6 Workflow diagram

Sequence Diagram

In Fig. 5.7 the system sequence diagram, elaborates the five basic modules of our system. In the first module, the system detects the face of a person through the webcam using detection algorithms (Haar Cascade). Then the system detects the Eyes (AdaBoost). After that the system detects and captures the eyes moments (Centre, Left, Right). Then the system detects the Blinking (Count Blink). In the last module, the system starts displaying a message depend upon the moments of eye and number of blinking.

Example:

```
if right ==1 and left ==1 and blink==5:
right =0
left =0
blink=0
print('I am Not Ok ')
```

```
service=1
elif centre ==1 and right ==0 and left ==1 and blink==1
right =0
left =0
blink=0
print('Call Guardian')
```

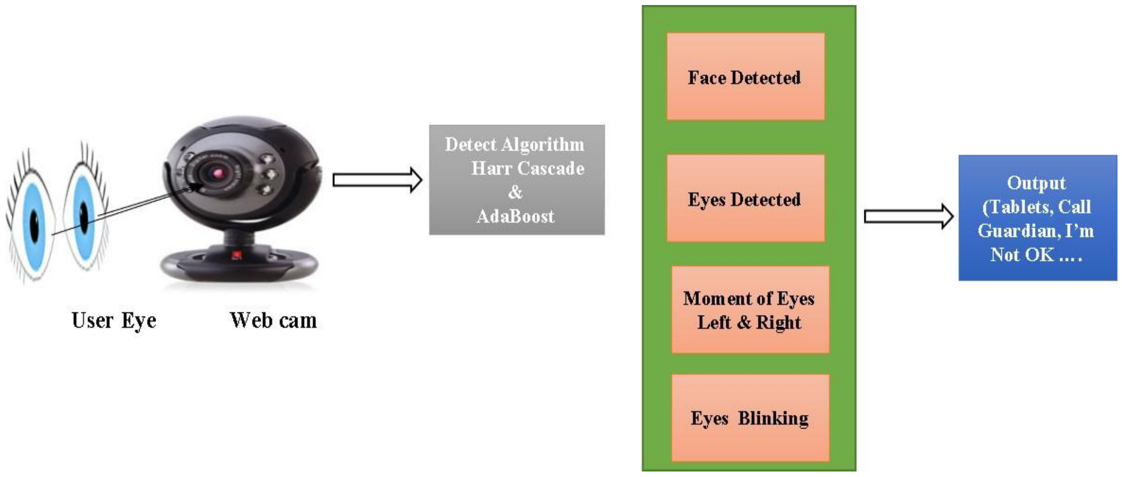

User Eye
Web cam
Detect Algorithm Harr Cascade & AdaBoost
Face Detected
Eyes Detected
Moment of Eyes Left & Right
Eyes Blinking
Output (Tablets, Call Guardian, I'm Not OK ….

Fig 5.7 Sequence Diagram

5.2 System Implementation Design

Modules Description

1. Video Capturing Module

After taking a short video of the participant's face using the front camera of the device used which is more likely a laptop. A process Frame method will be used to create the frames from the captured video. Afterwards the coloured frames will be converted to gray scale frames by extracting only the luminance component. The luminosity method is a more sophisticated version of the average method. It also averages the values, but it forms a weighted average to account for human perception. We're more sensitive to green than other colours, so green is weighted most heavily. The formula for luminosity is 0.21 R + 0.72 G + 0.07 B. The luminosity method works best overall.

2. Face Detection Module

The Haar classifier is used in algorithm for face detection. Haar classifier rapidly detects any object, based on detected feature not pixels, like facial feature. However, the area of the image being analysed for a facial feature needs to be regionalized to the location with the highest probability of containing the feature. By regionalizing the detection area, false positives are eliminated. As the result face is detected and marked with rectangle box and will be used later to approximate an axis of the eyes for eye detection step.

3. Eye Detection Module

To detect the eye, first, the Haar cascade classifier should be trained, in order to train the classifiers, the AdaBoost algorithm and Haar classifier algorithms must be implemented, two set of images are needed. One set contains an image or scene that does not contain the object.

The EBCM used all detected elements of Haar Cascade Classifier, and the result show the detected eye in rectangle box.

AdaBoost algorithm is used to train node classifiers on a Haar-like feature set to improve the generalization ability of the node classifier. Consequently, the face detection performance of the face detector is improved. Experimental results have proved that the proposed algorithm can significantly reduce the number of weak classifiers, increase the detection speed, and slightly raise the detection accuracy as well.

4. Eye Pupil Tracking Module

The corneal-reflection and pupil-centre are the two eye’s parts that are the most important parts to extract the features that will be used in EBCM method. These features help us in tracking the eyes movement. By identifying the center of the pupil and the location of the corneal reflection, the vector between them is measured. Besides, with further trigonometric calculations, point-of-regard can be found. The EBCM method succeeded in making the face and the eye's pupil moved together in the same direction synchronously and with the same direction. Let suppose that X is the human face which has been detected , P1 and P2 are two points related to the left eye, and they are moving synchronously with the movement of X.

5. Eye Blinking Module

Eye blinking and movement can be detected with relatively high reliability by unobtrusive techniques. Though, there are few techniques discovered for the active scene where the face and the camera device move independently, and the eye moves freely in every direction independently of the face. Although care must be taken, that eye-gaze tracking data is used in a sensible way, since the nature of human eye movements is a combination of several voluntary and involuntary cognitive processes.

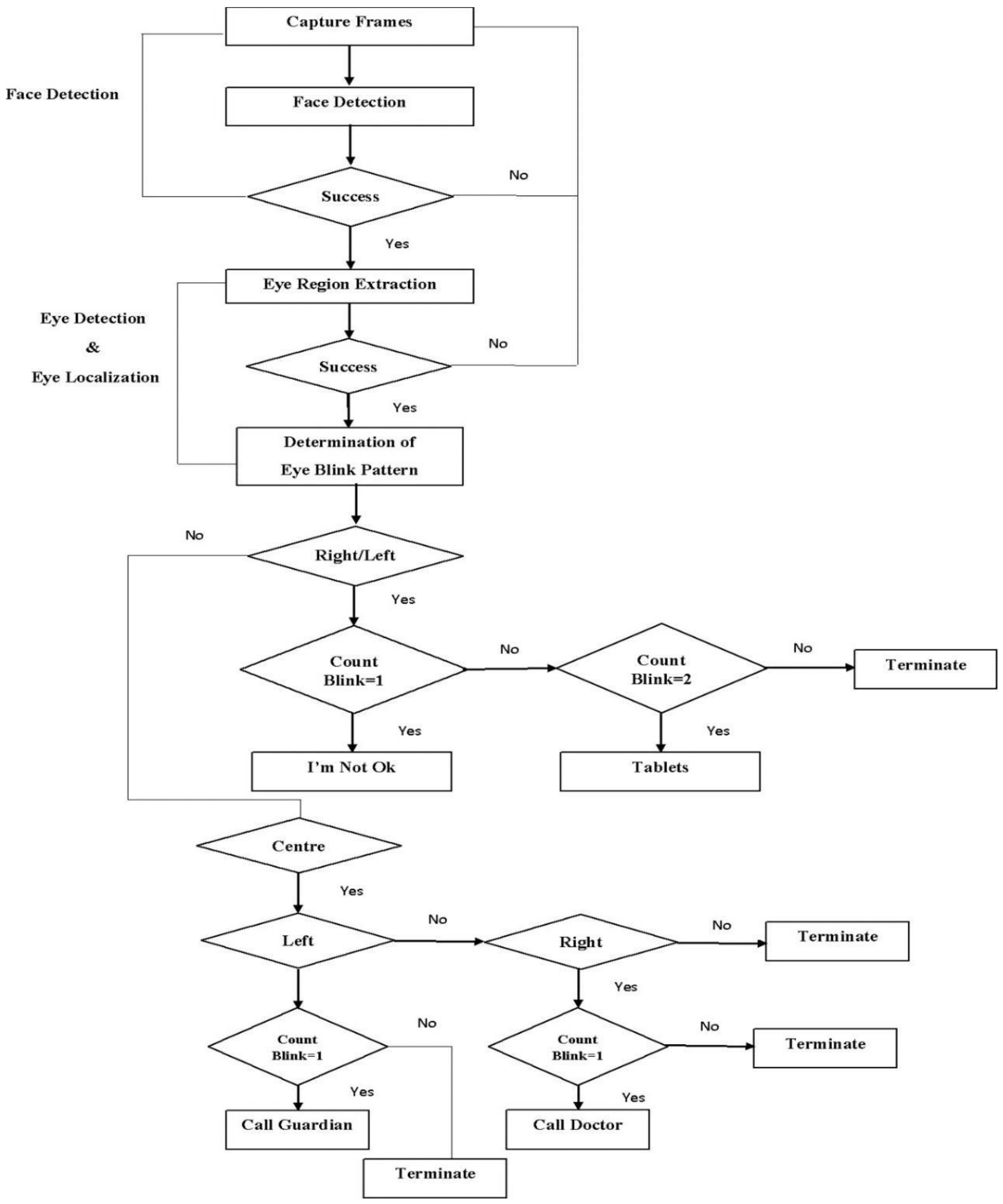


Fig 5.8 Various Modules in the Application

CHAPTER 6

ALGORITHMS

6.1 Face Detection Module

Haar Cascade Algorithm:

- Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/ folder. Let’s create face and eye detector with OpenCV.
- First we need to load the required XML classifiers. Then load our input image (or video) in grayscale mode.

```
import numpy as npimport cv2
face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')eye_cascade =
cv2.CascadeClassifier('haarcascade_eye.xml')
img = cv2.imread('sachin.jpg')gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

- Now we find the faces in the image. If faces are found, it returns the positions of detected faces as Rect(x,y,w,h). Once we get these locations, we can create a ROI for the face and apply eye detection on this ROI (since eyes are always on the face !!!).

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)for (x,y,w,h) in faces:
img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
roi_gray = gray[y:y+h, x:x+w]
roi_color = img[y:y+h, x:x+w]
eyes = eye_cascade.detectMultiScale(roi_gray)
for (ex,ey,ew,eh) in eyes:
cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
cv2.imshow('img',img)cv2.waitKey(0)cv2.destroyAllWindows()
```

6.2 Eye Detection Module:

ADA Boost Algorithm:

```
# Load libraries

from sklearn.ensemble import AdaBoostClassifier
from sklearn import datasets

# Import train_test_split function

from sklearn.model_selection import train_test_split

#Import scikit-learn metrics module for accuracy calculation

from sklearn import metrics

# Load data

iris = datasets.load_iris()

X = iris.data

y = iris.target

# Split dataset into training set and test set

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30%
test

# Create adaboost classifer object

abc = AdaBoostClassifier(n_estimators=50, learning_rate=1)

# Train Adaboost Classifer

model = abc.fit(X_train, y_train)

#Predict the response for test dataset y_pred = model.predict(X_test)

# Model Accuracy, how often is the classifier correct?

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 0.8888888888888888

import cv2
import dlibimg = cv2.imread('image.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # convert to grayscale detector =
dlib.get_frontal_face_detector()
rects = detector(gray, 1) # rects contains all the faces detected

def shape_to_np(shape, dtype="int"):
    coords = np.zeros((68, 2), dtype=dtype)
    for i in range(0, 68):
        coords[i] = (shape.part(i).x, shape.part(i).y)
    return coordspredictor = dlib.shape_predictor('shape_68.dat')
for (i, rect) in enumerate(rects):
    shape = predictor(gray, rect)
    shape = shape_to_np(shape)
    for (x, y) in shape:
        cv2.circle(img, (x, y), 2, (0, 0, 255), -1)
```

6.3 Eye Pupil Detection Module

```
import cv2

import dlib

import numpy as np

def shape_to_np(shape, dtype="int"):

        # initialize the list of (x, y)-coordinates

        coords = np.zeros((68, 2), dtype=dtype)

        # loop over the 68 facial landmarks and convert them

        # to a 2-tuple of (x, y)-coordinates

        for i in range(0, 68):

                coords[i] = (shape.part(i).x, shape.part(i).y)

        # return the list of (x, y)-coordinates

        return coords
```

```
def eye_on_mask(mask, side):
    points = [shape[i] for i in side]
    points = np.array(points, dtype=np.int32)
    mask = cv2.fillConvexPoly(mask, points, 255)
    return mask


def contouring(thresh, mid, img, right=False):
    try:
        cnt = max(cnts, key = cv2.contourArea)
        M = cv2.moments(cnt)
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
        if right:
            cx += mid
        cv2.circle(img, (cx, cy), 4, (0, 0, 255), 2)
    except:
        pass
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_68.dat')
left = [36, 37, 38, 39, 40, 41]
right = [42, 43, 44, 45, 46, 47]
cap = cv2.VideoCapture(0)
ret, img = cap.read()
```

```
thresh = img.copy()


cv2.namedWindow('image')

kernel = np.ones((9, 9), np.uint8)


def nothing(x):

    pass

cv2.createTrackbar('threshold', 'image', 0, 255, nothing)


while(True):

    ret, img = cap.read()

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    rects = detector(gray, 1)

    for rect in rects:


        shape = predictor(gray, rect)

        shape = shape_to_np(shape)

        mask = np.zeros(img.shape[:2], dtype=np.uint8)

        mask = eye_on_mask(mask, left)

        mask = eye_on_mask(mask, right)

        mask = cv2.dilate(mask, kernel, 5)

        eyes = cv2.bitwise_and(img, img, mask=mask)

        mask = (eyes == [0, 0, 0]).all(axis=2)
        eyes[mask] = [255, 255, 255]

        mid = (shape[42][0] + shape[39][0]) // 2

        eyes_gray = cv2.cvtColor(eyes, cv2.COLOR_BGR2GRAY)

        threshold = cv2.getTrackbarPos('threshold', 'image')

        _, thresh = cv2.threshold(eyes_gray, threshold, 255, cv2.THRESH_BINARY)

        thresh = cv2.erode(thresh, None, iterations=2) #1

        thresh = cv2.dilate(thresh, None, iterations=4) #2

        thresh = cv2.medianBlur(thresh, 3) #3

        thresh = cv2.bitwise_not(thresh)

        contouring(thresh[:, 0:mid], mid, img)

        contouring(thresh[:, mid:], mid, img, True)

        # for (x, y) in shape[36:48]:

        #     cv2.circle(img, (x, y), 2, (255, 0, 0), -1)

    # show the image with the face detections + facial landmarks

    cv2.imshow('eyes', img)

    cv2.imshow("image", thresh)

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

        break

cap.release()

cv2.destroyAllWindows()
```

6.4 Eye Blinking Module

```
from scipy.spatial import distance as dist

from imutils.video import FileVideoStream
from imutils.video import VideoStream

from imutils import face_utils

import numpy as np

import argparse

import imutils

import time

import dlib

import cv2

def eye_aspect_ratio(eye):

# compute the euclidean distances between the two sets of

# vertical eye landmarks (x, y)-coordinates

A = dist.euclidean(eye[1], eye[5])

B = dist.euclidean(eye[2], eye[4])

# compute the euclidean distance between the horizontal

# eye landmark (x, y)-coordinates

C = dist.euclidean(eye[0], eye[3])

# compute the eye aspect ratio

ear = (A + B) / (2.0 * C)

# return the eye aspect ratio

return ear

# construct the argument parse and parse the arguments

ap = argparse.ArgumentParser()

ap.add_argument("-p", "--shape-predictor", required=True,
help="path to facial landmark predictor")

ap.add_argument("-v", "--video", type=str, default="",

help="path to input video file")

args = vars(ap.parse_args())

# define two constants, one for the eye aspect ratio to indicate

# blink and then a second constant for the number of consecutive

# frames the eye must be below the threshold

EYE_AR_THRESH = 0.3

EYE_AR_CONSEC_FRAMES = 3

# initialize the frame counters and the total number of blinks

COUNTER = 0

TOTAL = 0

(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]

(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
```

CHAPTER 7

IMPLEMENTATION

REFLECTION OF INFRARED LIGHT

For eyelid movement measurement, an infrared (IR) emitting diode (model VSMB2020X01) and a phototransistor (VEMT2520X01) were mounted on some 3D glasses. These IR elements were placed more than 2 cm away from the eye and were approximately aligned with the right area of the right-eye pupil. To do so, the IR elements were mounted on a structure giving them two degree of freedom (DOF) and allowing their placement in an area in front of the right eye. With open eyes, the phototransistor mainly captures the reflflectedIR radiation coming from the cornea. With closed eyes, most of IR radiation is absorbed by the eyelid skin, reducing the reflflected component. Therefore, opening/ closing the eyes inflfluences the amount of reflflected IR signal and makes the phototransitor current change between high and low values.

The algorithm searches for the positive and negative waves, using an adaptive K-means clustering method that classififies samples as belonging to negative or positive waves or baseline. The position of the peaks is then obtained to measure the distance between the N-P waves. A short or long blink is detected if the distance between N-P peaks is more/less than a threshold. The exact algorithm is described in Appendices A and E.

BLINK DETECTION IN IR-OG

The software has been split into 5 different layers distributed like in a pipeline: the lowest, or hardware layer, receives data from the analog to digital converter (ADC) while the top layer identififies the short and long blinks. Each layer acts as a unit processing receiving input data and sending results out toward the next layer in the pipeline. We used the open-source library described in to build the application.

1) LAYER 1: HARDWARE LAYER

This layer guarantees the acquisition at a fifixed rate of 250 Hz and is completely implemented in the LBSP library.

2) LAYER 2: FILTERING

Data coming from the hardware layer might contain noise due to room lighting. In Europe the frequency of power lines is 50Hz, so an IR interference can be found at 100Hz. A low pass digital fifilter, with Q15 coeffificients h = [691, 3358, 7509, 9647, 7509, 3358, 691]/32764, which make a gain of -98dB at 100Hz, was implemented.

3) LAYER 3: CLASSIFIER

Here, data is classifified into three main categories: negative, baseline and positive. It contains an adaptive K-means algorithm that classififies data according to the distance to three centroids, which, in turn, are being continuously adjusted as well. More details about how this layer 3 algorithm works can be found in the system.

4) LAYER 4: WAVES IDENTIFICATION

This layer receives the signal, in, the classifification results of lower layer, clas, and outputs an integer, out = [1, 0, 1], identifying groups of samples belonging to the N,
P waves or the baseline. To do that, we have implemented a two-state fifinite state
machine (FSM), including an area estimation procedure and the search of the minimum/maximum samples position in respective N/P waves (Algorithm 2). In its initial state, the FSM waits for a non-baseline sample to come. When it does, the integration process, to estimate the area of the wave, and the search for the wave extrema is initiated. On the one hand, the calculation of the area helps to fifilter small amounts of samples that can be found inside the wave that belong to the opposite class. On the other hand, by searching for the maximum/minimum extrema in the wave, we get a steadier feature for distinguishing short blinks from long ones than if we used other features such as the position of the fifirst sample associated with the N and P waves, which could be more inflfluenced by signal noise. In the next state, the FSM keeps on calculating the area and the extrema as long as the input does not belong to the baseline. Otherwise, the FSM assesses the area, outputs its sign and returns to the initial state. The delaymax variable contains the position of the extrema from the sample in which this layer signaled an N or P wave. Fig. 20 illustrates the operation of this layer in cyan and the position of the extrema in red.

CHAPTER 8

TESTING

Unit Testing

Unit testing is a development procedure where programmers create tests as they develop software. The tests are simple short tests that test functionally of a particular unit or module of their code, such as a class or function.

Using open source libraries like c unit, oppunit and nun it (for C, C++ and C#) these tests can be automatically run and any problems found quickly. As the tests are developed in parallel with the source unit test demonstrates its correctness.

Validation and System Testing

Validation testing is a concern which overlaps with integration testing. Ensuring that the application fulfils its specification is a major criterion for the construction of an integration test. Validation testing also overlaps to a large extent with System Testing, where the application is tested with respect to its typical working environment. Consequently for many processes no clear division between validation and system testing can be made. Specific tests which can be performed in either or both stages include the following.

- **Regression Testing:** Where this version of the software is tested with the automated test harness used with previous versions to ensure that the required features of the previous version are skill working in the new version.
- **Recovery Testing:** Where the software is deliberately interrupted in a number of ways off, to ensure that the appropriate techniques for restoring any lost data will function.
- **Security Testing:** Where unauthorized attempts to operate the software, or parts of it, attempted it might also include attempts to obtain access the data, or harm the software installation or even the system software. As with all types of security determined will be able to obtain unauthorized access and the best that can be achieved is to make this process as difficult as possible.
- **Stress Testing:** Where abnormal demands are made upon the software by increasing the rate at which it is asked to accept, or the rate t which it is asked to produce information. More complex tests may attempt to crate very large data sets or cause the software to make excessive demands on the operating system.

- **Performance testing:** Where the performance requirements, if any, are checked. These may include the size of the software when installed, type amount of main memory and/or secondary storage it requires and the demands made of the operating when running with normal limits or the response time.
- **Usability Testing:** The process of usability measurement was introduced in the previous chapter. Even if usability prototypes have been tested whilst the application was constructed, a validation test of the finished product will always be required.
- **Alpha and beta testing:** This is where the software is released to the actual end users. An initial release, the alpha release, might be made to selected users who be expected to report bugs and other detailed observations back to the production team. Once the application changes necessitated by the alpha phase can be made to larger more representative set users, before the final release is made to all users.

The final process should be a Software audit where the complete software project is checked to ensure that it meets production management requirements. This ensures that all required documentation has been produced, is in the correct format and is of acceptable quality. The purpose of this review is: firstly to assure the quality of the production process and by implication construction phase commences. A formal hand over from the development team at the end of the audit will mark the transition between the two phases.

Integration Testing:

Integration Testing can proceed in a number of different ways, which can be broadly characterized as top down or bottom up. In top down integration testing the high level control routines are tested first, possibly with the middle level control structures present only as stubs. Subprogram stubs were presented in section2 as incomplete subprograms which are only present to allow the higher. Level control routines to be tested.

Top down testing can proceed in a depth-first or a breadth-first manner. For depth-first integration each module is tested in increasing detail, replacing more and more levels of detail with actual code rather than stubs. Alternatively breadth-first would processed by refining all the modules at the same level of control throughout the application .in practice a combination of the two techniques would be used. At the initial stages all the modules might be only partly functional, possibly being implemented only to deal with non-erroneous data. These would be tested in breadth-first manner, but over a period of time each would be replaced with successive refinements which were closer to the full functionality. This allows depth-first testing of a module to be performed simultaneously with breadth-first testing of all the modules.

The other major category of integration testing is Bottom Up Integration Testing where an individual module is tested form a test harness. Once a set of individual module have been tested they are then combined into a collection of modules ,known as builds, which are then tested by a second test harness. This process can continue until the build consists of the entire application. In practice a combination of top down and bottom-up testing would be used. In a large software project being developed by a number of sub-teams, or a smaller project where different modules were built by individuals. The sub teams or individuals would conduct bottom-up testing of the modules which they were constructing before releasing them to an integration team which would assemble them together for top-down testing.

Unit Testing:

Unit testing deals with testing a unit as a whole. This would test the interaction of many functions but confine the test within one unit. The exact scope of a unit is left to interpretation. Supporting test code, sometimes called *Scaffolding*, may be necessary to support an individual test. This type of testing is driven by the architecture and implementation teams. This focus is also called black-box testing because only the details of the interface are visible to the test. Limits that are global to a unit are tested here.
In the construction industry, scaffolding is a temporary, easy to assemble and disassemble, frame placed around a building to facilitate the construction of the building. The construction workers first build the scaffolding and then the building. Later the scaffolding is removed, exposing the completed building. Similarly, in software testing, one particular test may need some supporting software. This software establishes can a correct evaluation of the test take place.
The scaffolding software may establish state and values for data structures as well as providing dummy external functions for the test. Different scaffolding software may be needed form one test to another test. Scaffolding software rarely is considered part of the system. Sometimes the scaffolding software becomes larger than the system software being tested. Usually the scaffolding software is not of the same quality as the system software and frequently is quite fragile. A small change in test may lead to much larger changes in the scaffolding.

Internal and unit testing can be automated with the help of coverage tools. Analyzes the source code and generated a test that will execute every alternative thread of execution. Typically, the coverage tool is used in a slightly different way. First the coverage tool is used to augment the source by placing information prints after each line of code. Then the testing suite is executed generating an audit trail. This audit trail is analyzed

8.2 TEST CASE

Test Case Number	**Title**	**Description**	**Excepted Output**	**Assumption**	**Result**
1	Help	On one eye blink give the speech output Help	Help	One eye blinked	Expected Output
2	Good Morning	On two eye blink give the speech output Good Morning	Good Morning	Two eye blinked	Expected Output
3	Switch On Light	On three eye blink give the speech output Switch On Light	Switch On Light	Three eye blinked	Expected Output
4	Switch Off Light	On four eye blink give the speech output Switch Off Light	Switch Off Light	Four eye blinked	Expected Output

CHAPTER 9

RESULT ANALYSIS

Table 9.1: COMPARING FEATURES

Parameters	**Existing System**	**Proposed System**
Face Detection module	Pseudo-random algorithm	Haar Cascade Algorithm
Eye Detection Module	BuildIndex Algorithm	ADA Boost classifier Algorithm
Eye Pupil Detection Module	Search Algorithm	Modified Iris Algorithm

- **Face Detection Module**

We have changed the existing algorithm with the Haar Cascade Algorithm which decreases the time required by the camera to detect the human face hence decreasing the overall system processing time.

- **Eye Detection Module**

We have changed the existing algorithm with the ADA Boost classifier Algorithm which increases the accuracy to detect the human eye hence improving the overall system performance time.

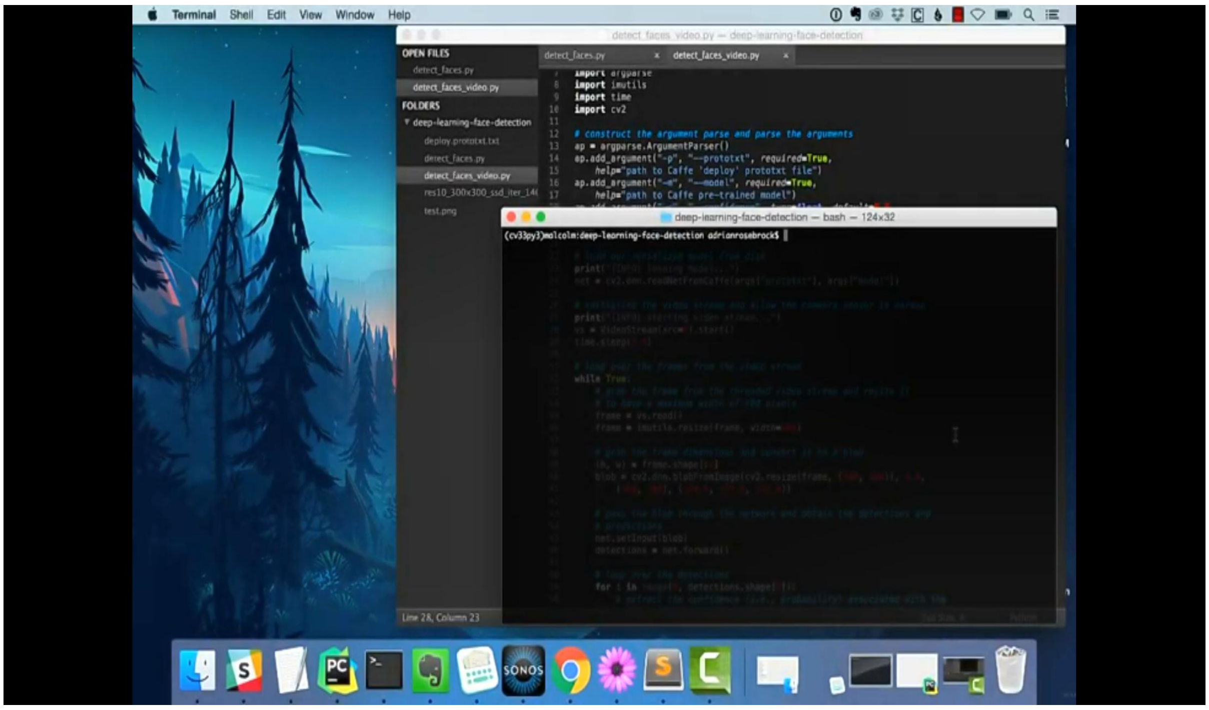
- **Eye Pupil Detection Module**

We have changed the existing algorithm with the Modified Iris Algorithm which increases the overall result of the system hence making it more desirable for medical use.

CHAPTER 10

SNAPSHOTS

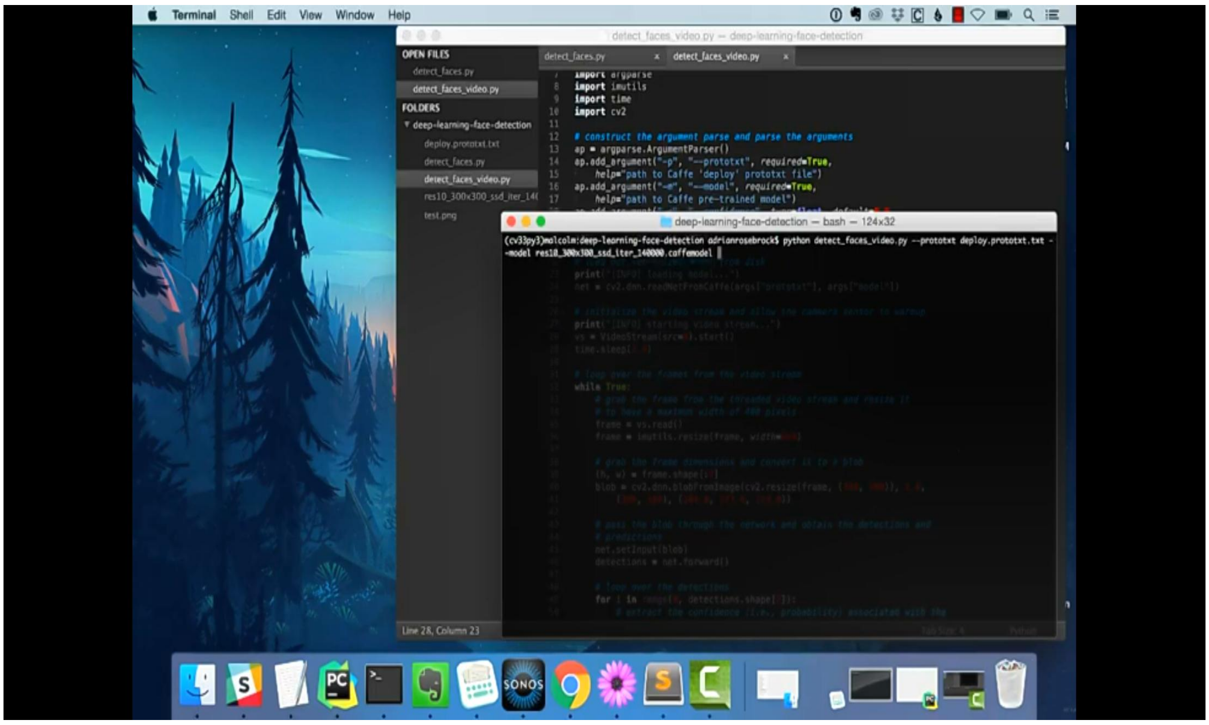
10.1 Running Face Detection Algorithm (Step 1)



```
import argparse
import imutils
import time
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
    help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
    help="path to Caffe pre-trained model")
```

```
(cv33py3)malcolm:deep-learning-face-detection adrianrosebrock$
```

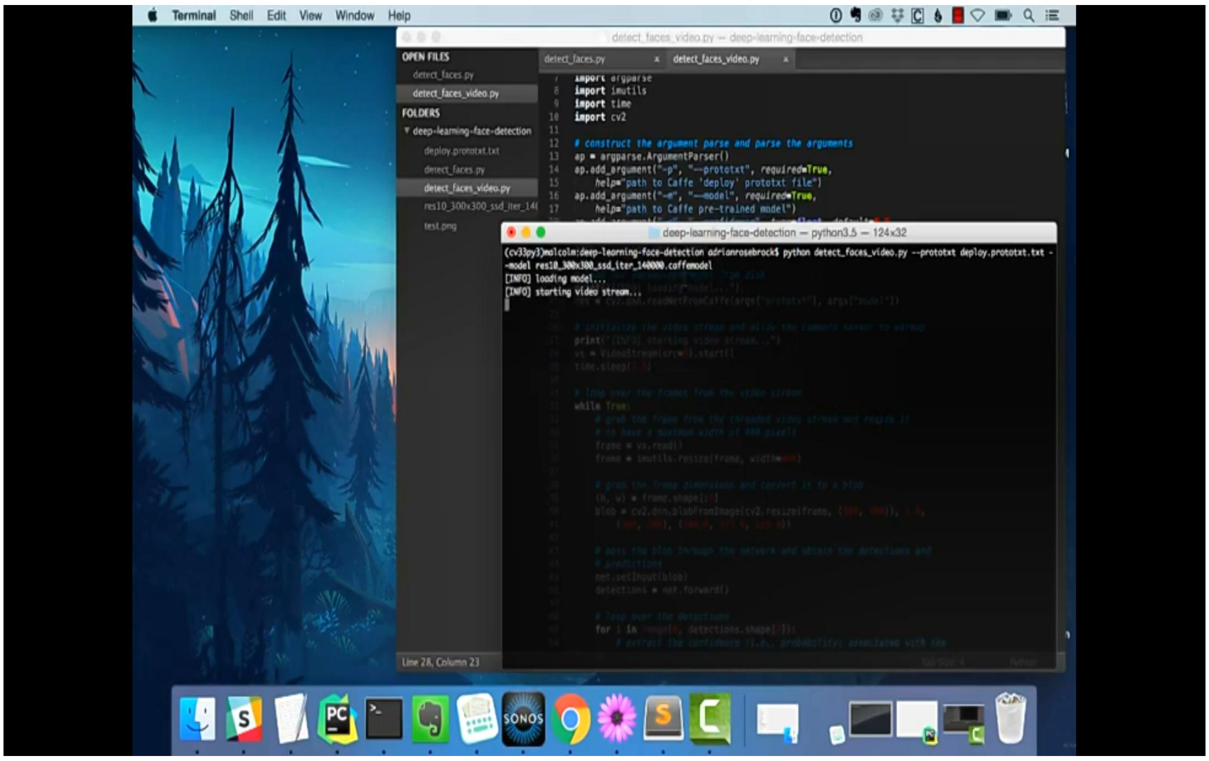
10.2 Running Face Detection Algorithm (Step 2)



```
import argparse
import imutils
import time
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
    help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
    help="path to Caffe pre-trained model")
```

```
(cv33py3)malcolm:deep-learning-face-detection adrianrosebrock$ python detect_faces_video.py --prototxt deploy.prototxt.txt -
-model res10_300x300_ssd_iter_140000.caffemodel
```

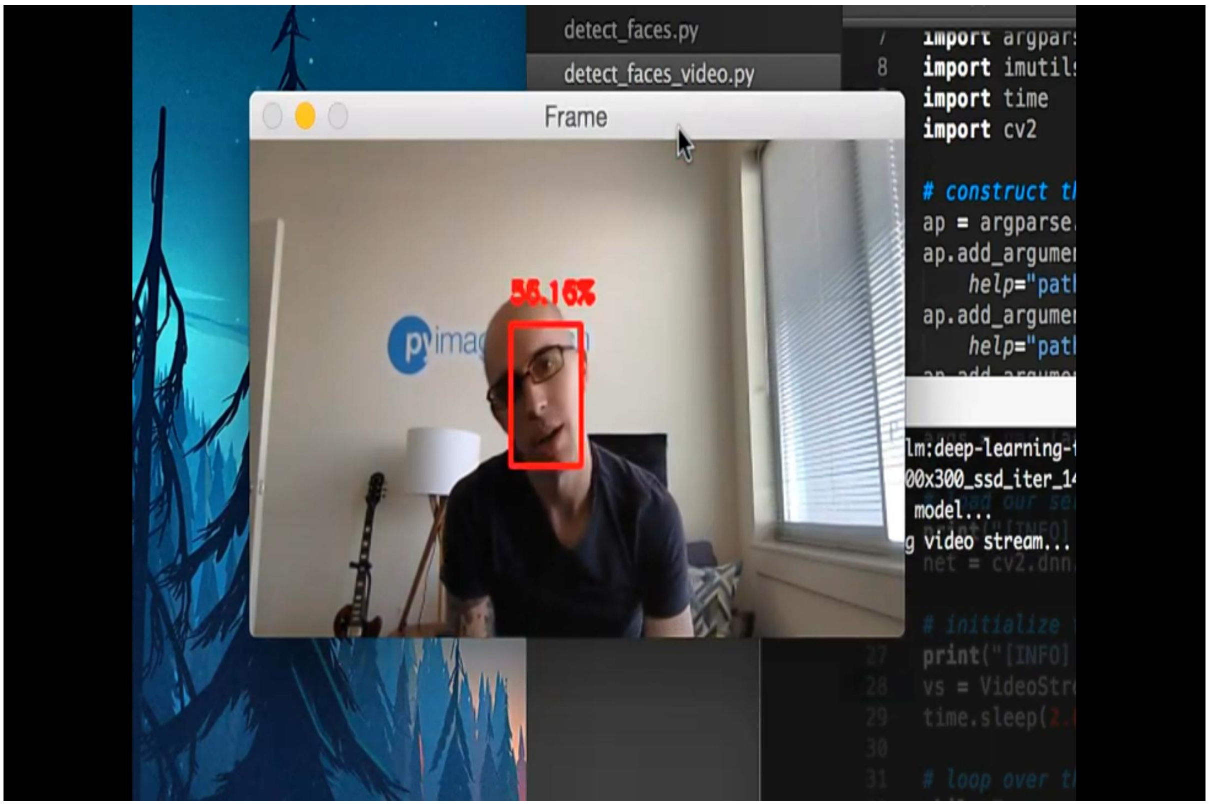
10.3 Face Detection Algorithm Accessing PC camera


Terminal Shell Edit View Window Help
detect_faces_video.py — deep-learning-face-detection
OPEN FILES
detect_faces.py
detect_faces_video.py
FOLDERS
deep-learning-face-detection
deploy.prototxt.txt
detect_faces.py
detect_faces_video.py
res10_300x300_ssd_iter_14(
test.png
Line 28, Column 23
deep-learning-face-detection — python3.5 — 124x32

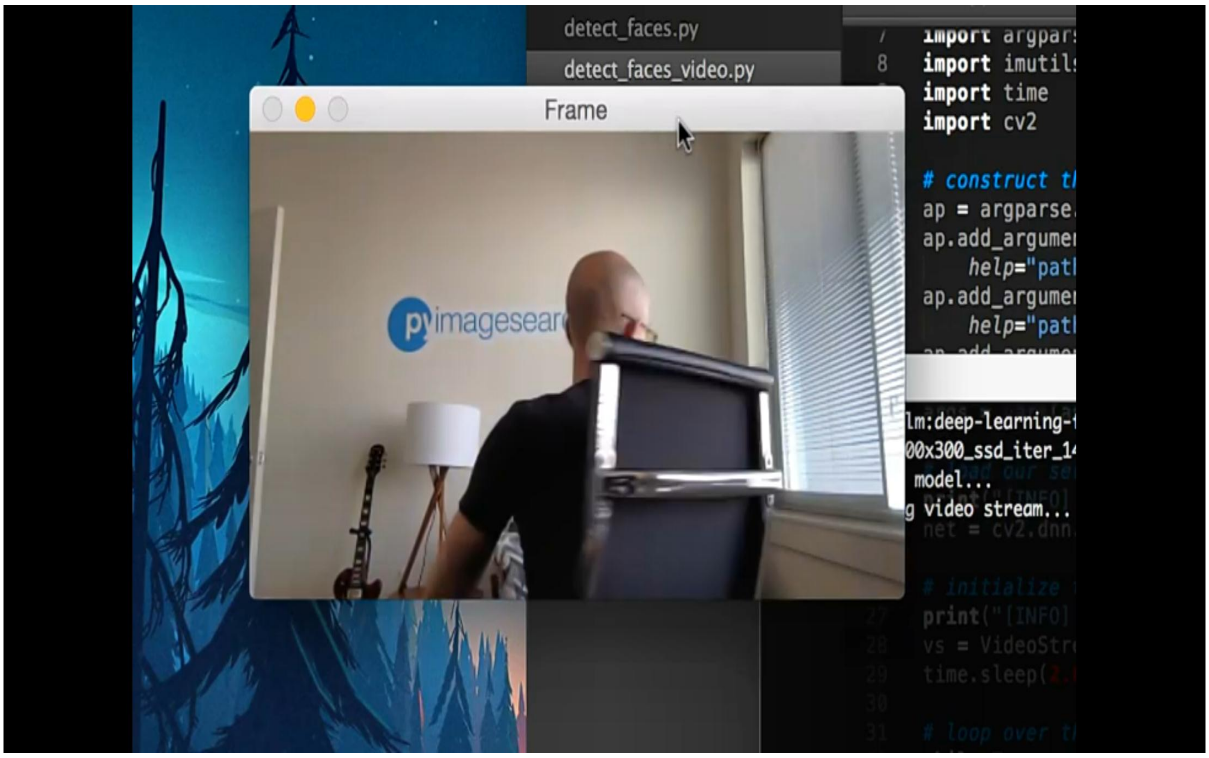
```
import argparse
import imutils
import time
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
	help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
	help="path to Caffe pre-trained model")
```

```
(cv33py3)malcolm:deep-learning-face-detection adrianrosebrock$ python detect_faces_video.py --prototxt deploy.prototxt.txt -
-model res10_300x300_ssd_iter_140000.caffemodel
[INFO] loading model...
[INFO] starting video stream...
```

10.4 Detecting Human Face Successfully



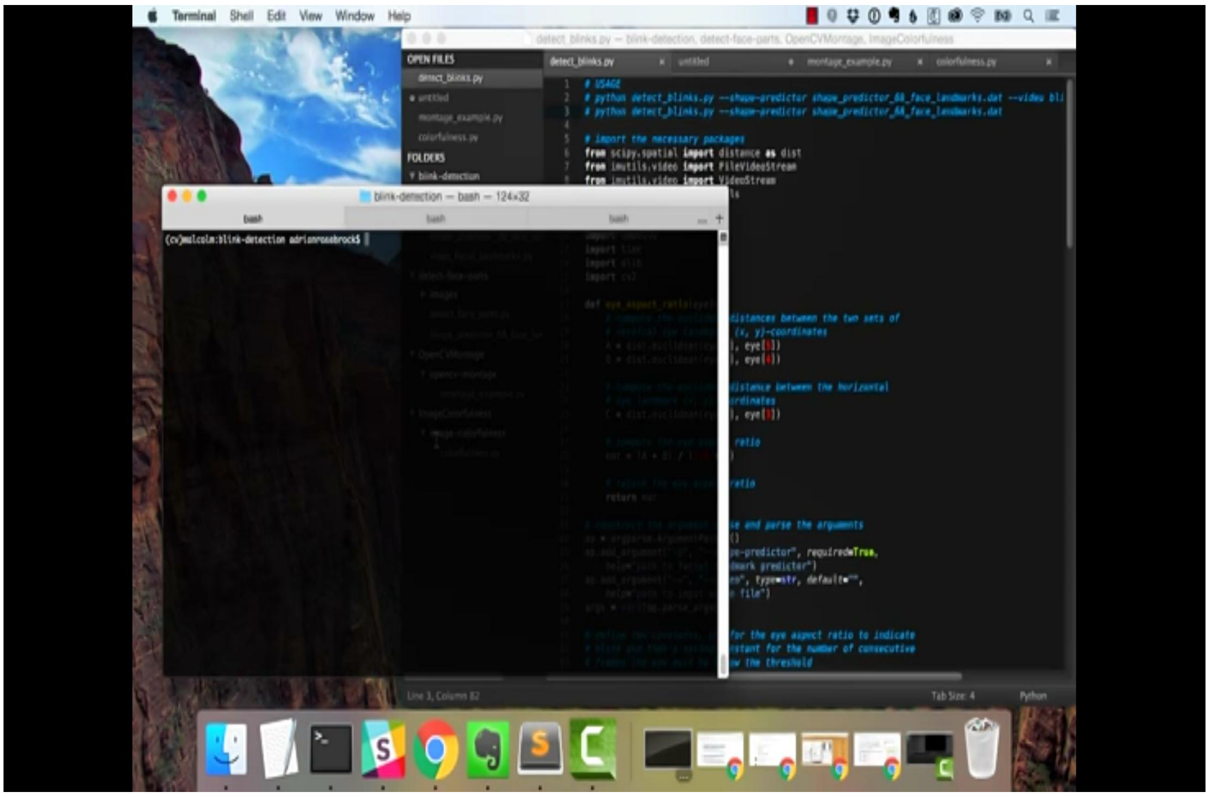
10.5 Checking The Accuracy Of The Face Detection Algorithm



```
import argpar
import imutil
import time
import cv2

# construct t
ap = argparse
ap.add_argume
    help="pat
ap.add_argume
    help="pat
```

```
lm:deep-learning-
00x300_ssd_iter_1
 model...
g video stream...
```

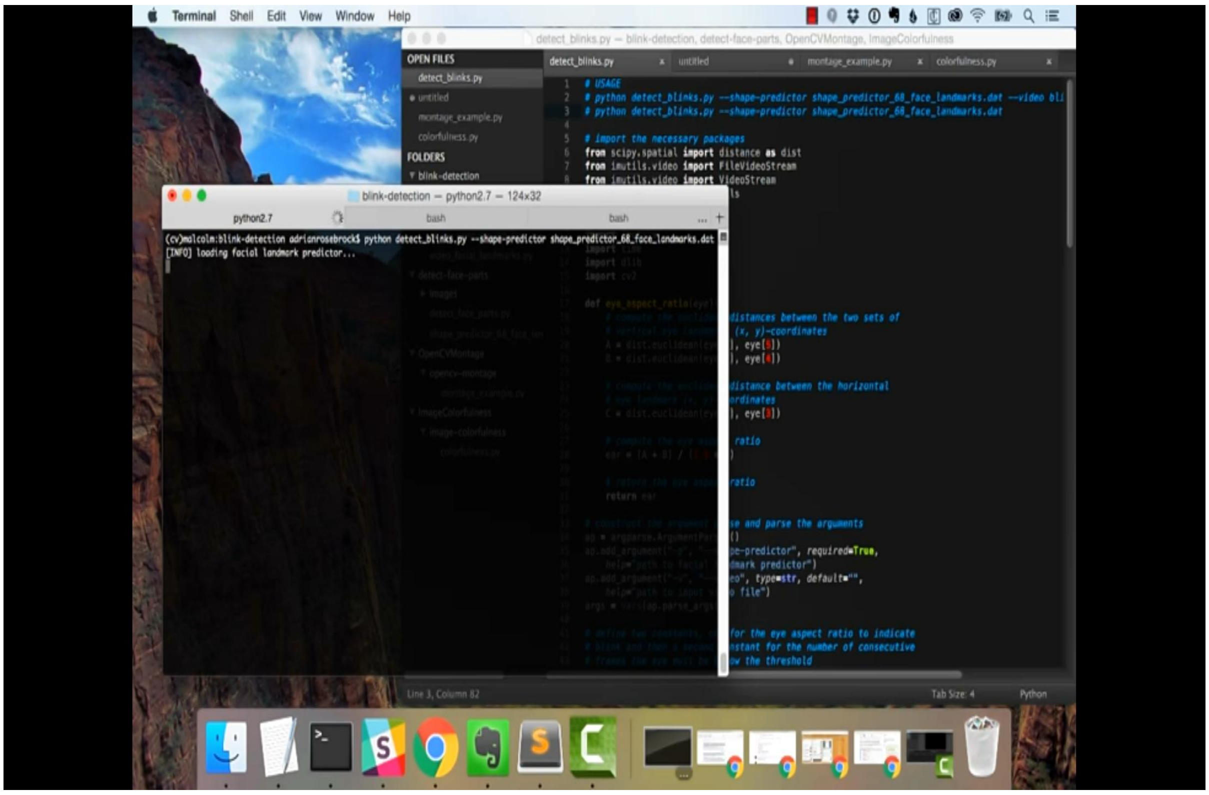
10.6 Running Eye Blink Detection Algorithm (Step 1)


Terminal Shell Edit View Window Help
detect_blinks.py — blink-detection, detect-face-parts, OpenCVMontage, ImageColorfulness
OPEN FILES
detect_blinks.py
untitled
montage_example.py
colorfulness.py
FOLDERS
blink-detection
blink-detection — bash — 124×32
bash
Line 3, Column 82
Tab Size: 4
Python

```
# USAGE
# python detect_blinks.py --shape-predictor shape_predictor_68_face_landmarks.dat --video bli
# python detect_blinks.py --shape-predictor shape_predictor_68_face_landmarks.dat

# import the necessary packages
from scipy.spatial import distance as dist
from imutils.video import FileVideoStream
from imutils.video import VideoStream
```

```
(cv)malcolm:blink-detection adrianrosebrock$
```

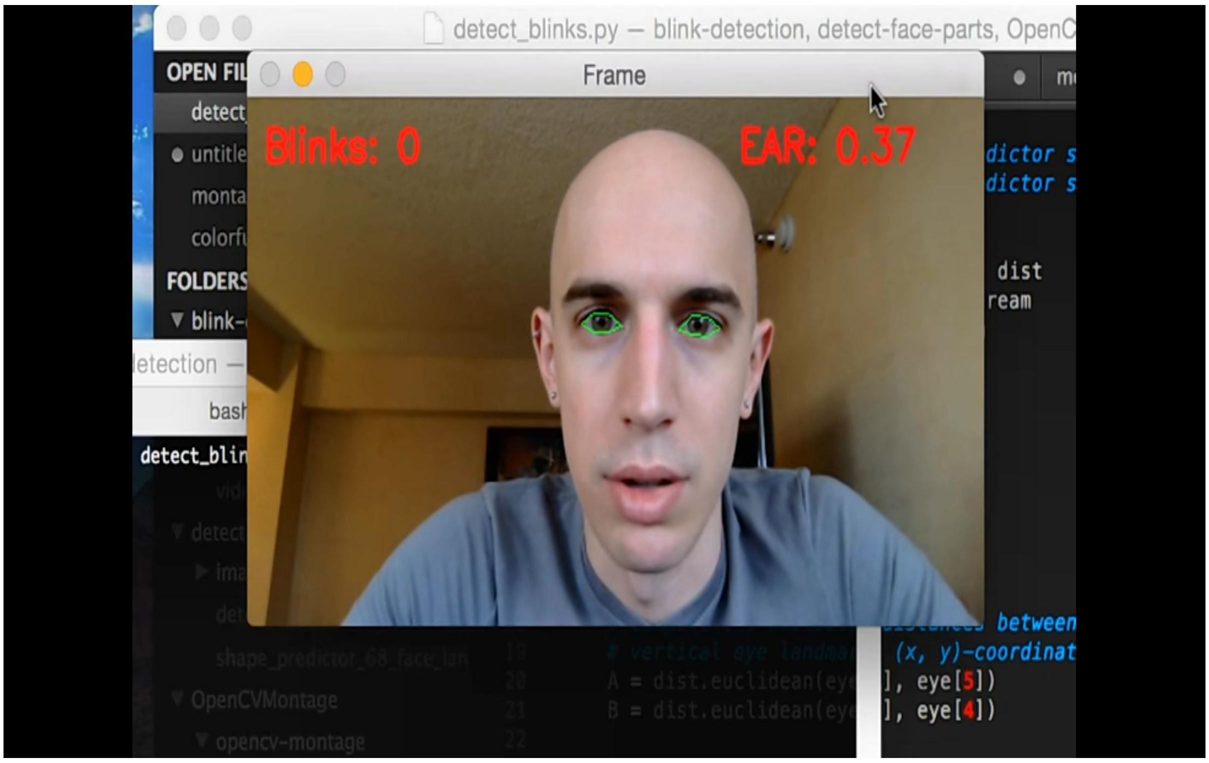
10.7 Running Eye Blink Detection Algorithm (Step 2)



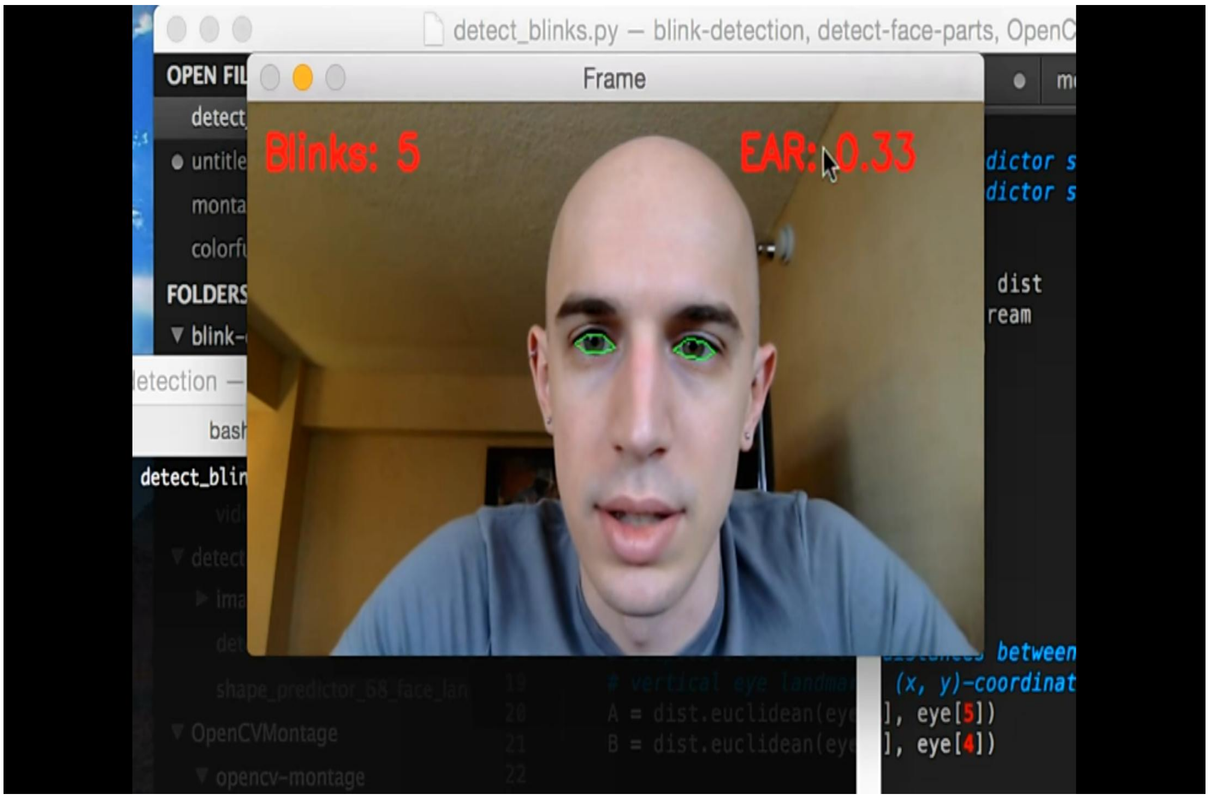
```
(cv)malcolm:blink-detection adrianrosebrock$ python detect_blinks.py --shape-predictor shape_predictor_68_face_landmarks.dat
[INFO] loading facial landmark predictor...
```

```
# USAGE
# python detect_blinks.py --shape-predictor shape_predictor_68_face_landmarks.dat --video bli
# python detect_blinks.py --shape-predictor shape_predictor_68_face_landmarks.dat

# import the necessary packages
from scipy.spatial import distance as dist
from imutils.video import FileVideoStream
from imutils.video import VideoStream
```

10.8 Successfully Detected Human Eyes

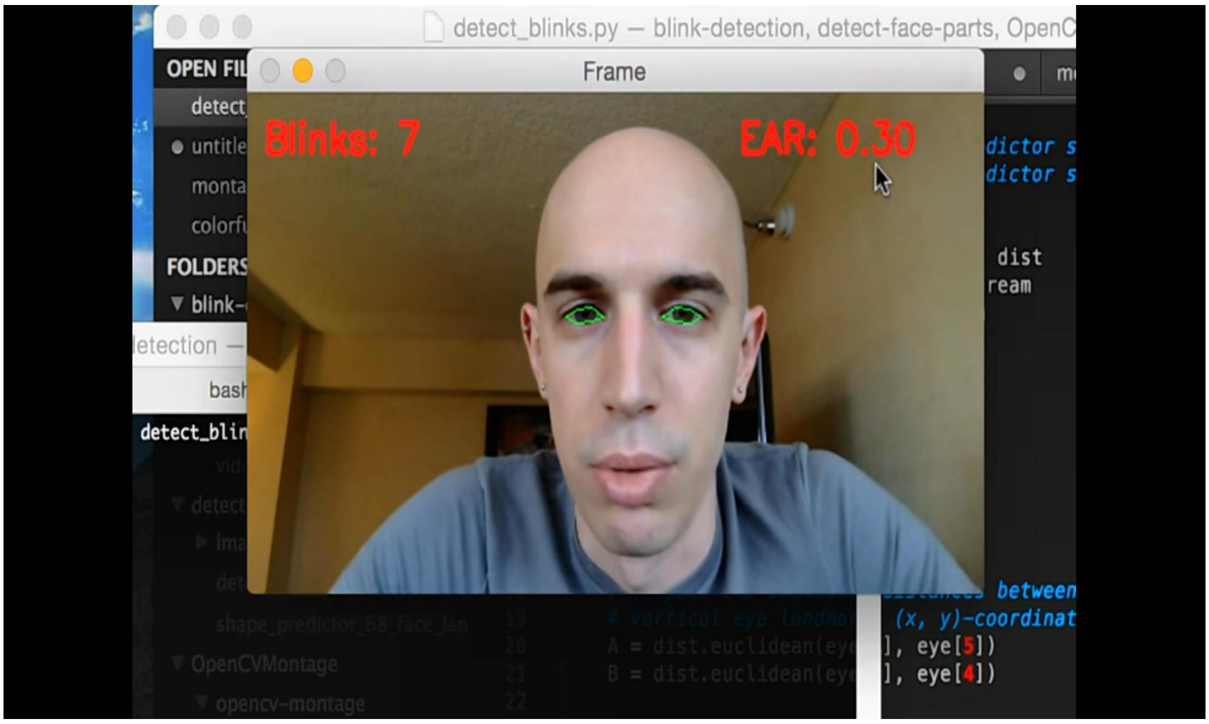


10.9 Detecting Fast Eye Blinks Successfully



```
A = dist.euclidean(eye    ], eye[5])
B = dist.euclidean(eye    ], eye[4])
```

10.10 Detecting Slow Eye Blinks Successfully



```
A = dist.euclidean(eye    ], eye[5])
B = dist.euclidean(eye    ], eye[4])
```

CHAPTER 11

CONCLUSIONS

The OSHW IR-OG equipment outperforms in accuracy (99.3 %) all similar devices in the reviewed literature and the three other tested technologies when they are used for people without disabilities. The averaged sensitivity in the experiment was 98.2% for that group, which is very close to the 100% obtained in [20] with similar experimental conditions.There were non-signifificant differences in accuracy between people without disabilities and ALS. For people with CP, with associated uncontrolled head movements, the accuracy dropped drastically due to small displacements, actions that mimicked long blinks (such as laughing), etc. The effificiency of IR-OG has been demonstrated although it needs another more appealing, comfortable and harmless support to host the IR diodes and that are compatible for people wearing corrective glasses. Despite all theses issues, it was the preferred wearable technology together with EOG for people without disabilities.

11.1 FUTURE ENHANCEMENT

- Making the overall time to process the blinks into actions much more faster.
- Increasing the number of applications connected to the board.

BIBLIOGRAPHY

[1] A. P. A. A. Mohammed, ''Effificient eye blink detection method for disabled helping domain,'' Eye, vol. 10, no. (P1), p. P2, 2014.

[2] K. Grauman, M. Betke, J. Gips, and G. R. Bradski, ''Communication via eye blinks-detection and duration analysis in real time,'' in Proc. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), vol. 1, Dec. 2001, pp. 1–2.

[3] K. Mukherjee and D. Chatterjee, ''Augmentative and alternative communication device based on eye-blink detection and conversion to morsecode to aid paralyzed individuals,'' in Proc. Int. Conf. Commun., Inf. Comput. Technol. (ICCICT), Jan. 2015, pp. 1–5.

[4] K. Mukherjee, R. Karmakar, and S. Das, ''Effective estimation of driver drowsiness based on eye status detection and analysis,'' in Proc. Int. Conf. Devices, Circuits Commun. (ICDCCom), Sep. 2014, pp. 1–4.

[5] A. Królak and P. Strumiłło, ''Eye-blink detection system for human– computer interaction,'' Universal Access Inf. Soc., vol. 11, no. 4, pp. 409–419, 2012.

[6] S. He and Y. Li, ''A Single-channel EOG-based Speller,'' IEEE Trans. Neural Syst. Rehabil. Eng., vol. 25, no. 11, pp. 1978–1987, Nov. 2017.

[7] M. Abo-Zahhad, S. M. Ahmed, and S. N. Abbas, ''A novel biometric approach for human identifification and verifification using eye blinking Signal,'' IEEE Signal Process. Lett., vol. 22, no. 7, pp. 876–880, Jul. 2015.

[8] L. Burton, W. Albert, and M. Flynn, ''A comparison of the performance of webcam vs. infrared eye tracking technology,'' in Proc. Hum. Factors Ergonom. Soc. Annu. Meeting, vol. 58. Los Angeles, CA, USA: SAGE, 2014, pp. 1437–1441.

[9] P. Biswas and P. Langdon, ''A new interaction technique involving eye gaze tracker and scanning system,'' in Proc. Conf. Eye Tracking South Africa, 2013, pp. 67–70.

[10] Y. Li, S. He, Q. Huang, Z. Gu, and Z. L. Yu, ''A EOG-based switch and its application for 'start/stop' control of a wheelchair,'' Neurocomputing, vol. 275, pp. 1350–1357, Jan. 2018.

[11] Z. Lv, C. Zhang, B. Zhou, X. Gao, and X. Wu, ‘‘Design and implementation of an eye gesture perception system based on electrooculography,’’ Expert Syst. Appl., vol. 91, pp. 310–321, Jan. 2018.