CONTENT

Chapter 1- INTRODUCTION

There has been a rapid increase in the Quadriplegia prone persons with increasing population. Several systems have been made for disabled persons. Disable people who cannot move anything except their head and eyes. For these people head movement and blinks are the only way to communicate with outside world through computer. This research aims in developing a system that can aid the physically challenged by allowing them to interact with a computer system using only their head and eyes. Human- computer interaction has become an increasingly important part of our daily lives. Here, eyes as an input, the movement of user's head can provide a convenient, natural and high- bandwidth source of input.

A method for eye blink detection and head moment localization is proposed for controlling mouse cursor and drowsiness detection. To achieve higher level of accuracy and precision, an algorithm is furnished with various processing steps and develops an efficient system to reduce both the cost and the computational complexity. Drowsiness is the transition between awake state and sleep during which one's abilities to observe and analyze are strongly reduced. For disabled person it is difficult to perform task while he is drowsy moreover he/she can get hurt if continue to work in that position, so an alert system is being designed for taking protective measures while a person is sleepy. This is the reason why more and more researches are made to build automatic detectors of this harmful state. According to Renner and Mehring, drowsiness can be detected both in brain activity which refers to the ability to process the information and in eyes activity which refers to the perception ability. This study focuses on the visual signs of drowsiness: blinks.

1.1 PROBLEM STATEMENT

In the present situation, PCs has turned into a fundamental piece of everybody's everyday life as the comfort with the utilization of PCs is expanding, yet this is just limited to the general population who can without much of a stretch handle its peripherals like mouse, console and other information gadgets, yet shouldn't something be said about the general population who are appendage impaired. Through the upgrade of innovation, it is necessary to make a framework for them so they can collaborate with PC. The advancement of technology for recognizing or forestalling drowsiness is a noteworthy challenge in the field of accident evasion frameworks. It would in this manner be advantageous to figure out how to recognize drowsiness before it happens and to have the capacity to caution the person in time.

In our task we have attempted to make an application for them. Through their head movements and direction of their eye gaze they can interface with PC and using their eye blinks drowsiness can be detected. The system will precisely check in real time, the open or close condition of the eyes of person. By checking the person's eyes, the indications of drowsiness can be identified in the beginning to protect from accident, as well as person can operate a computer system using his head movements despite of his inabilities. Apart from helping disabled our system can also be helpful in many things interactive computer games, machine guided gymnastic, robot control and simulation.

1.2 MOTIVATION

Assistive technology (AT) promotes greater independence for people with disabilities by enabling them to perform tasks that they were formerly unable to accomplish. However, the communication with patients having neuro-Locomotor disabilities is a great challenge even today. Usually, the communication with these patients requires continuous presence of a caregiver who should guess patient's basic needs. There is a category of people with severe speech and motor impairment or with neuro-Locomotor disabilities who cannot speak and cannot use sign language. If these patients have a good level of understanding and perception, they should use their eye blinks and head movements for Human-Computer Interaction (HCI). HCI, driving assistance system, the sports cognition, drivers' fatigue detection systems, etc. We can also use an head tracking system to implement an "head mouse" and use it as control signals to enable users to interact with system interfaces directly without the need for input devices like mouse or keyboard, for ease of use of a computer for the disabled.

Current user-computer dialogues tend to be one sided, with the bandwidth from the computer to the user far greater than that from user to computer. A fast and effortless mode of communication from a user to a computer would help redress this imbalance. Therefore by investigating the possibility of introducing the movements of a user's head as an additional input medium. While the technology for measuring head movements and reporting them in real time has been improving, what is needed is appropriate interaction techniques that incorporate head movements into the user-computer dialogue in a convenient and natural way.

Since the introduction of the computers, advances in communication with the human, have mainly been made on the communication from the computer to the human (graphical representation of the data, window systems, use of sound), whereas communication from the human to the computer is still restricted to keyboards, joysticks and mice - all operated by hand. By tracking the direction of the visual gaze of the human for example, the bandwidth of communication from the human to the computer can be increased by using the information about what the human is looking at. This is only an example for possible direction towards increasing the human-computer bandwidth; by monitoring the entire state of the human, the computer can react to all kinds of gestures and/or voice commands. This leads to a new way of regarding the computer, not as a tool that must be operated explicitly by commands, but instead as an agent that perceives the human and acts upon that perception. In turn, the human is granted with the freedom to concentrate on interacting with the data presented by the computer, instead of using the computer applications as tools to operate on the data. Our eyes are crucial to us as they provide us an enormous amount of information about our physical world", i.e., vision is an essential tool in interaction with other human beings and the physical world that surrounds us. If computers were able to understand more about our attention as we move in the world, they could enable simpler, more realistic and intuitive interaction. To achieve this goal, simple and efficient methods of analyzing human cues are required, that would enable computers to interact with humans in more intelligent manner. We envision that a future computer user who would like to, say, use some information from the Internet for doing some work, could walk into the room where the computer agent is located and interact with it by looking, speaking and gesturing. When the user's visual gaze falls on the screen of the computer agent, it would start to operate from the user's favorite starting point or where he/she left of the last time.

Wherever the user looks, the computer agent will begin to emphasize the appropriate data carrying object (database information, ongoing movies, video-phone calls), and utterances like `more' combined with a glance or a pointing gesture will zoom in on the selected object. If the user's visual gaze utters over several things, the computer agent could assume that the user might like an overview, and an appropriate zooming out or verbalized data summery can take place.

For a disabled person drowsiness identification system has been proposed that generates alarms when patient falls asleep during operating a computer system. A number of different physical phenomena can be monitored and measured in order to detect drowsiness of person. Drowsiness or fatigue is an important element in case for a disabled person working on a computer system as there are many threats related to computer, person working if felt drowsy drops his head can result in endless scrolling of page due o his head downward moment, because system designed is based on cursor motion due to head movements, there are many such threats as data loss in sleepy state of user, termination of programs, unwanted information addition, even the person can get disbalanced in that state which can then be then hurtful for him. So to protect from these accidents a system is designed to generate a alert as a warning for computer operator that he is getting drowsy and he should rather stop the system or get awake. The advancement of technology for recognizing or forestalling drowsiness in the patient’s seat is a noteworthy challenge in the field of accident evasion frameworks. It would in this manner be advantageous to figure out how to recognize drowsiness before it happens and to have the capacity to caution the patient in time

Head tracking enables an application to recognize and identify a user's head movements. Head tracking is often found in conjunction with eye or face tracking, where it uses the facial features like the nose, mouth and eyes to track the user. Head tracking can be achieved using a typical basic camera or face-tracking software. It supports and enhances human-computer interaction.

Head tracking is used in various applications like games, home automation and security. The data is digitized and sent to the application to perform necessary tasks as per head movements Drowsiness Detection System is applicable mainly for driver drowsiness detection

In head tracking technology, the user’s face and head movements are tracked by capturing raw data via cameras, or it may require special equipment to be worn on the head to capture the movements. The facial features are recognized separately. It is possible to track the head movements from a particular distance with the use of webcams in laptops. Certain actions can be performed via applications with corresponding head movements. Direct movement and behaviors of characters within applications through face controls are also possible. Head tracking can be used in conjunction with augmented reality.

The concept of head tracking is commonly used in games where the player's head movements are tracked and changes are carried out in game controls as per the head movements. Head tracking is now most popularly seen in integration with smartphones to support various games and user authentication. It serves as another layer of security to traditional username and password authentication. A user can customize certain movement as to ensure their identity. Head tracking is also used in automated photo capture based on poses and face features.

OpenCv is an image processing library made by Intel which is accessible for C, C++ and Python. OpenCv supports loads of algorithm identified with computer vision and machine learning. OpenCv python is a library of python ties intended to tackle computer vision issues. OpenCv is utilized for all kind of image and video examination, similar to facial detection and recognition, tag perusing, photograph altering, advance robotic vision, optical character recognition and a ton more.

1.3 OBJECTIVE

In this project our main aim is to calculate the direction of gaze and user drowsiness. Our system will consist a laptop built-in web-cam which takes live image frames and Gaze Pointer application processes the frames to extract user's direction of Gaze while blink of patient will be used for person's drowsiness detection. Proposed system performance will be analyzed in different scenarios and some limitations are defined, which are as follows. User head should be at the same altitude as the web-cam. But it is highly robust in case of distance between user's head and computer screen. Currently system is only tested for frontal faces.

The first step we will be performing on each input frame is face detection and then eye detection is carried out. To detect the face and eye, the algorithm we will be using is Viola-Jones algorithm.
In Viola-Jones algorithm, Haar like features are used to detect the face. The three key points involved in this algorithm are:

- First is forming an integral image for fast feature evaluation.
- Second is an efficient classifier, built by an Adaboost Learning Algorithm, to select the most suitable Haar like features to detect faces.
- Third is cascading these classifiers, to discard the background region quickly and consider only the face like regions.

The input to the Viola-Jones Algorithm is a grayscale image. After detecting face and our aim is to detect eye features. Eye gaze direction tracking is a complex problem; it needs to acquire a number of facial and eye features to compute Direction of Gaze. After eye feature detection, eye blink is computed for drowsiness. In this regard, first problem is to identify necessary and sufficient eyes features which can result in an accurate Direction of Gaze calculation. We will use some calibration algorithm so that our head movements can easily be traced on computer screen. And for drowsiness we will only use number of blinks, and set a threshold value to 5-6 blinks which on crossing is determined as sleepy.

1.4 ORGANIZATION OF REPORT

The second chapter discusses about various methodologies that were used to achieve the desired result. Detailed step by step procedures are mentioned.

The third chapter discusses about the results (accuracy, prediction and f1 score) that were obtained on testing the model on various datasets.

The fourth chapter discusses in brief the limitations of the proposed model and the scope of future work to improve the model and deal the problem better.

Chapter 2 – METHODOLOGIES

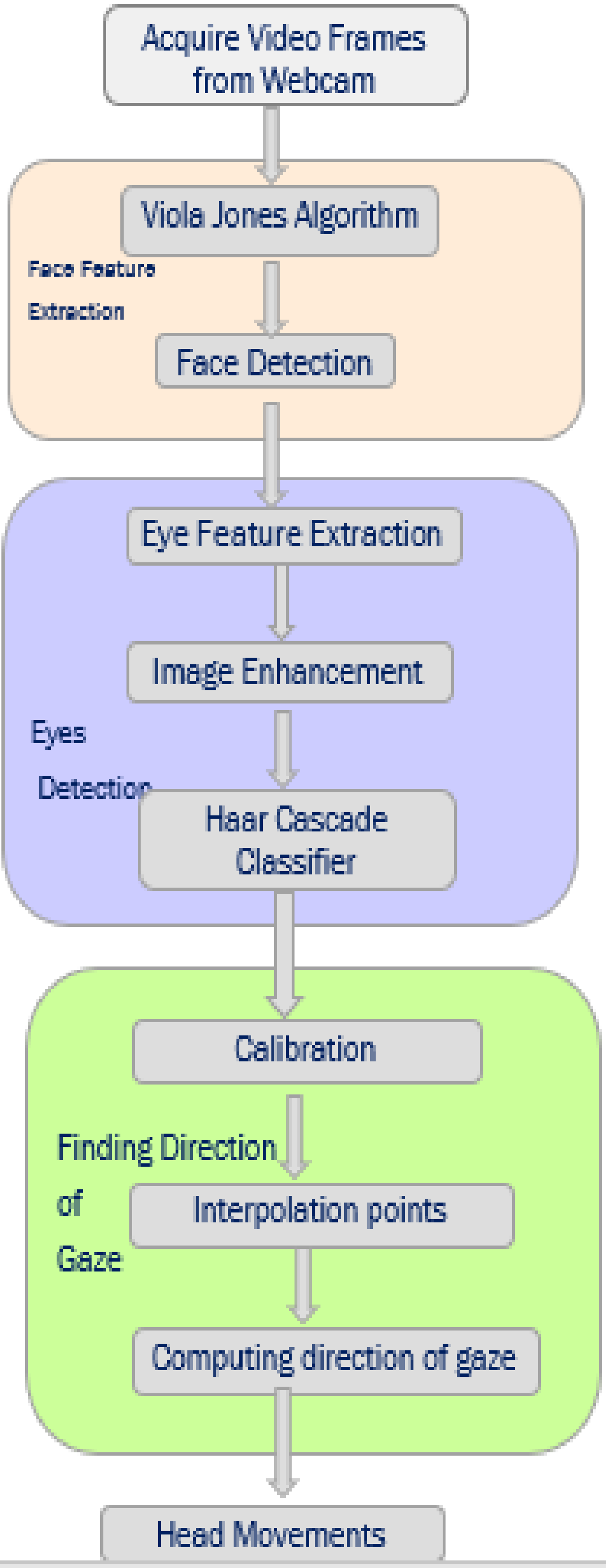


Figure 1. Flow Chart describing cursor movements

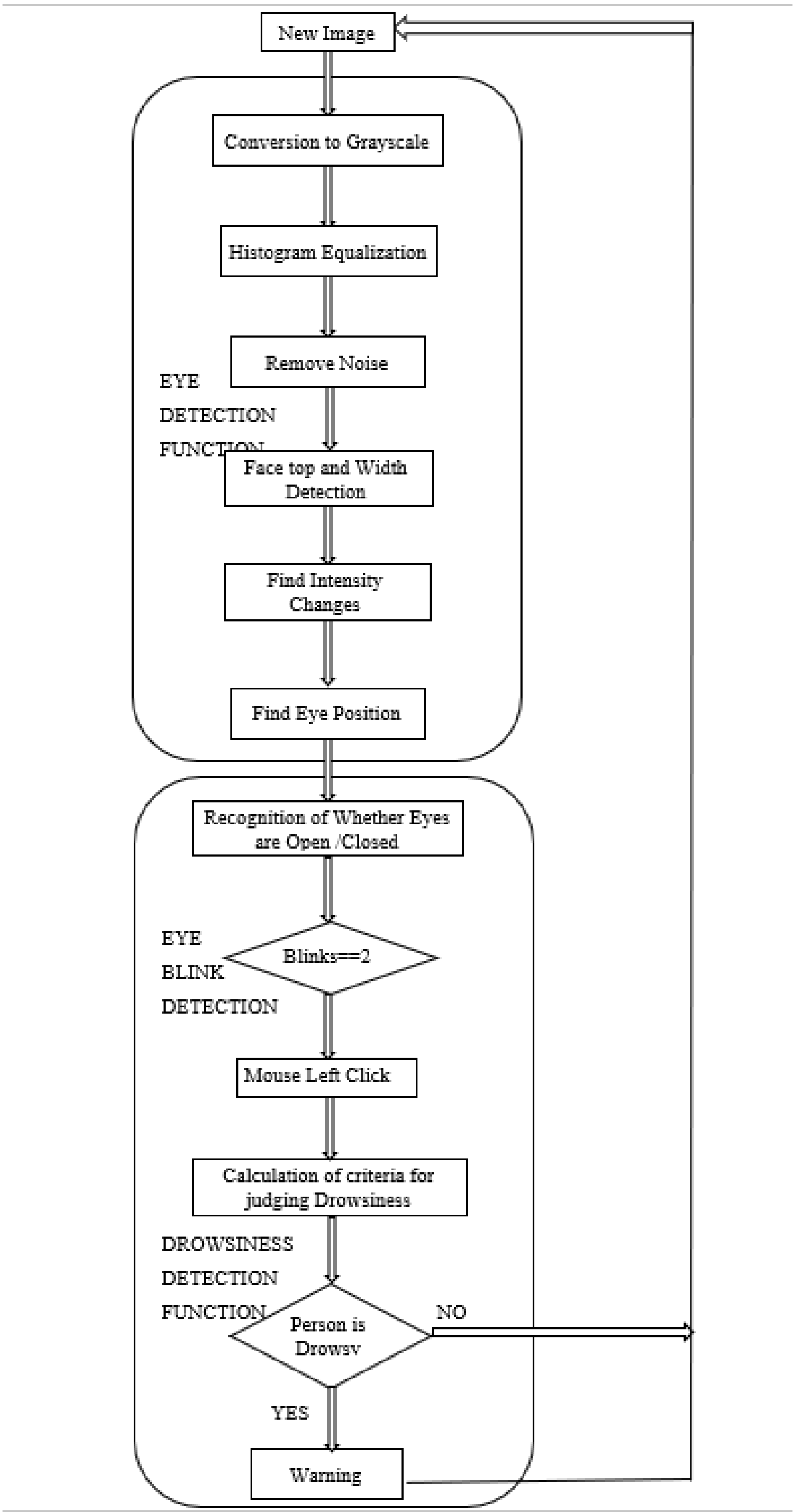


Figure 2. Flow Chart for methodology of Mouse Left Click and Drowsiness Detection

2.1 IMAGE ENHANCEMENT

While the shape of a histogram tells us quite a bit, frequency as a value on the y-axis is only useful in specialized cases. Changing the y axis values without changing the shape of the histogram is known as normalizing. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values. The method is useful in images with backgrounds and foregrounds that are both bright or both dark.

It is true that the background contrast gets improved after histogram equalization. But on comparing the images before and after. We lost most of the information there due to over-brightness. It is because its histogram is not confined to a particular region as we saw in previous cases. So to solve this problem, adaptive histogram equalization is used. In which, image is divided into small blocks called "tiles" (tileSize is 8x8 by default in OpenCV). Then each of these blocks are histogram equalized as usual. So in a small area, histogram would confine to a small region (unless there is noise). If noise is there, it will be amplified. To avoid this, contrast limiting is applied. If any histogram bin is above the specified contrast limit (by default 40 in OpenCV), those pixels are clipped and distributed uniformly to other bins before applying histogram equalization. After equalization, to remove artifacts in tile borders, bilinear interpolation is applied.

Figure 3: Image Enhancement

In this project we have used head movements for controlling the motion of mouse cursor which is implemented using few algorithms like first we need to detect face for head movement, which is done using Haar Feature-based Cascade Classifiers. Next is eye detection which is also detected using haar cascade. In this we select a region of interest (which is our eyes) as a bounding box, which is our region of movement for the motion of cursor. But before that we need to enhance the contrast, normalize the picture. After having eye coordinates we can finally collaborate these points for cursor movements

2.2 ALGORITHMS USED IN THE PROJECT:

Haar Feature-based Cascade Classifiers:

- We will see the basics of Face Detection using Haar Feature based cascade classifiers
- We will extend the same for Eye Detection

This is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. Many set of such features are found in human face, 80% to 90% of face contains haar features. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.

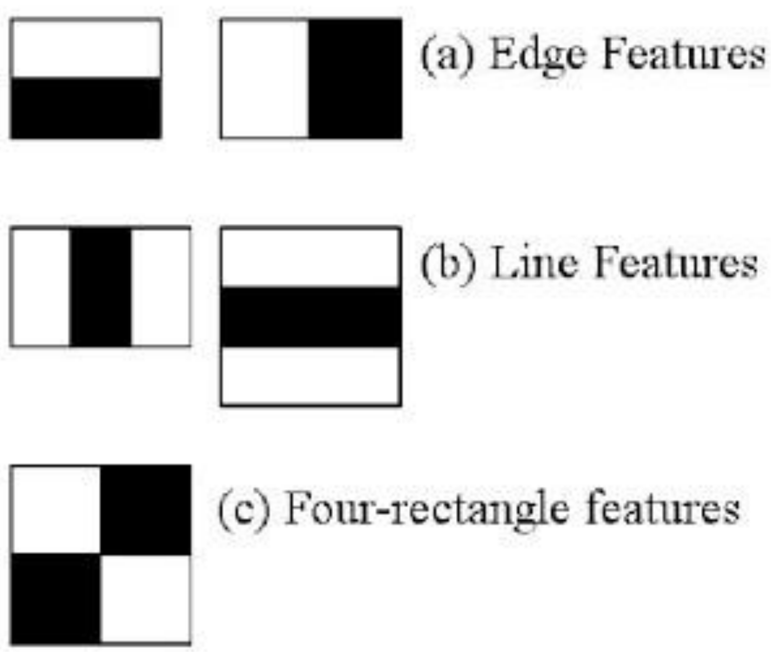


Figure 4: Haar Cascade Features

Now, all possible sizes and locations of each kernel are used to calculate lots of features. For each feature calculation, we need to find the sum of the pixels under white and black rectangles. More specifically, let's use three kinds of features. First, the two-rectangle feature which is the difference between the sums of the pixels within rectangular region. The region is having same size and shape and is adjacent vertically and horizontally. Second, the three- rectangle feature which computes the sum within the two outside rectangles subtracted from the sum of centre rectangle. Third, a four-feature which computes the sum between diagonal pairs of rectangles.Then through Haar feature, the difference between face parts and non-face parts can be described.

Figure 5: Haar Features application on face

We know that as long as the rectangular position or sizes different will correspond to different features. So even for a 19×19 image, it will be thousands of features. In the calculation of feature value, the same pixel values will be repeated used for calculation of the different features value, and this will spend a lot of time. Therefore, in order to improve the calculation feature value speed, we introduce integral figure method.

The idea of integral figure is that the sum of pixels of rectangle region formed by the points from the up-left to the low-point within image will be as an array in memory [8]. when calculate the pixel within a given area, we can directly to using the elements of array and do not need to recalculate the pixel value within the region again. This greatly improves the computational speed. For the point within image f(x, y), the value of integral figure is:

$$\boxed{I(u,v)=\sum x \leq u, y \leq v f(x,y}$$

We select the best features out of many features by Adaboost. AdaBoost algorithm mainly through selecting the most effective simple classifier as weak classifier, then according to the classification results of simple classifier, increase weights of misclassification sample, and reduce the weights of correct classification sample. Thus in the next round of training, the main training focus on the first sample of classification in error, and sort out weak classifier of the smallest wrong sample classification after updated weights, then put every training weak classifier linear combination into strong classifier.

Fig below illustrates the cascade. An image window (region) is passed to the first classifier. It is either classified as non-face or a decision is deferred and the image is passed to the second, etc. classifier. The goal of each classifier is to prune the training set for the next stage classifier of the cascade. Since easily Recognizable non-face images are classified in the early stages, classifiers of the later stages of the cascade can be trained rapidly only on the harder, but smaller, part of the non-face training set. Stages in cascade are constructed by training classifiers using AdaBoost.

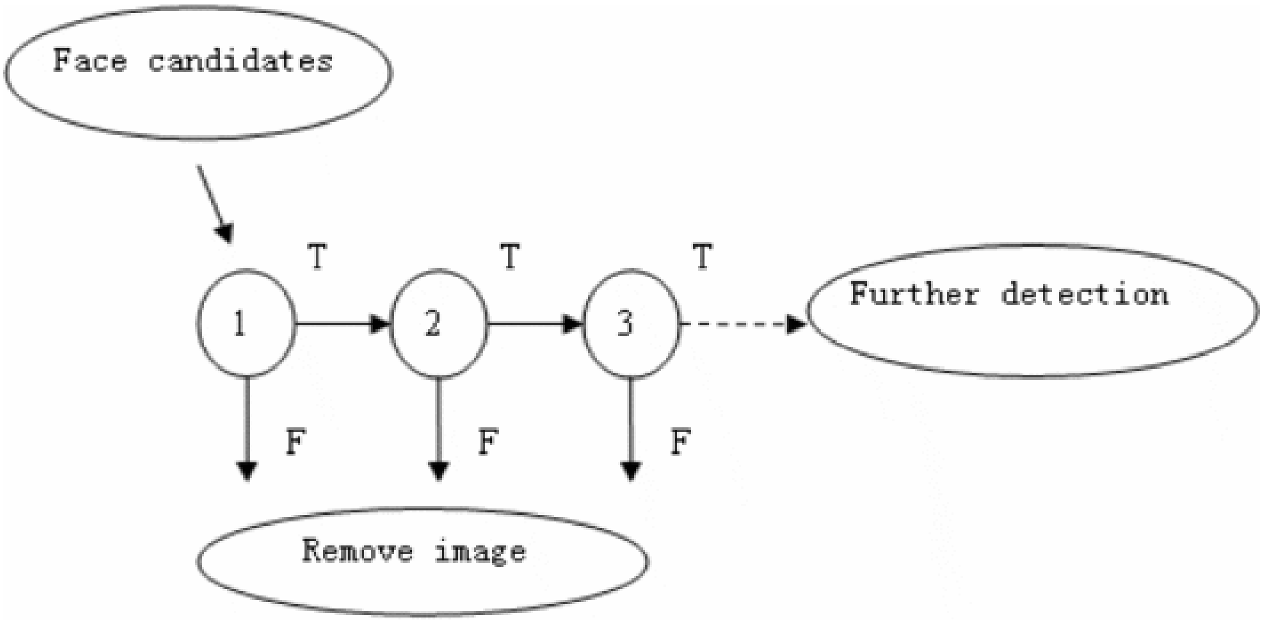


Figure 6: Cascade Classifier

Many common image operations like image enhancement, changing color-spaces etc., are performed using region of interest(ROI) in OpenCV. The ROI of a image not needed to be always a rectangle, but it is always a matrix, or a rectangle image, so you cannot have a round image, but you may have a circle on a black background that contains the information that you want.

A ROI allows us to operate on a rectangular subset of the image. The typical series of steps to use ROI is:

- Create a ROI on image
- Perform the operation you want on this sub-region of the image
- Reset back the ROI

In OpenCv, the origin is top-left corner of the image. And a ROI is specified using a CvRect Structure.

Given below are two illustrations which presents Face Detection and Eye Detection.

The rectangle areas surrounding them are ROIs which will be used for next Process that is Image Enhancement.

```
# Specify a vector of rectangles(ROI)
rects = []
fromCenter = false
# Select multiple rectangles
```

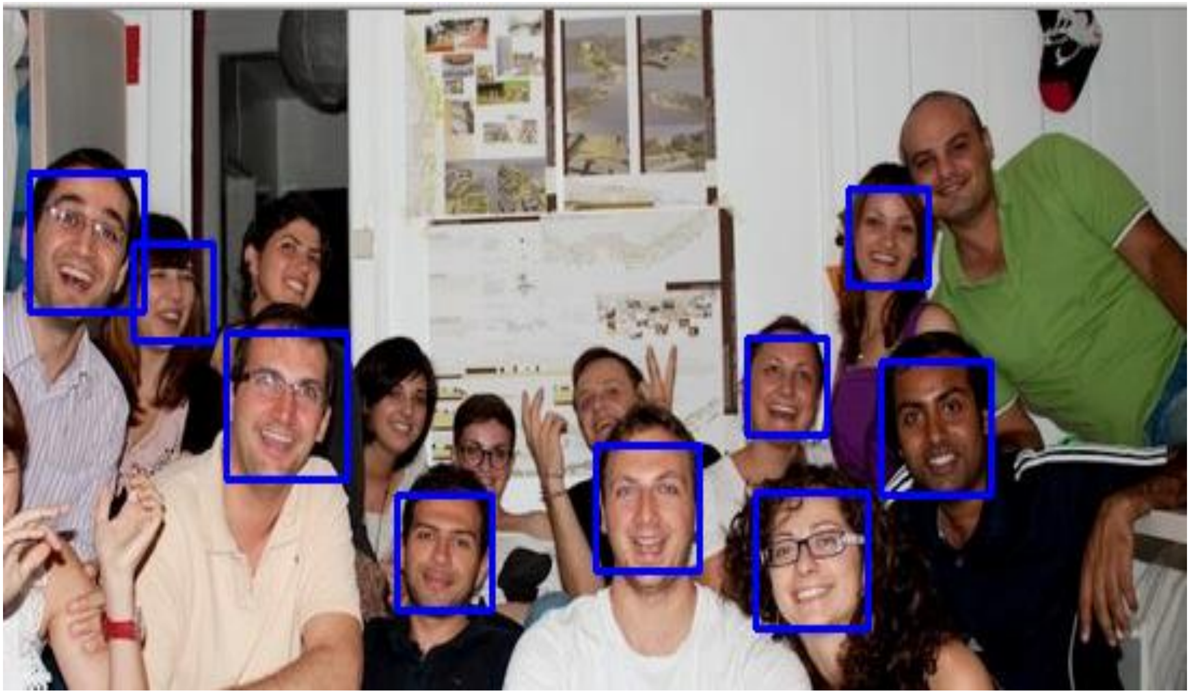


Figure 7: Face Detection Using Viola Jones

Once the face is identified, the Region of Interest (ROI) is set to the face rectangle, detected by the Viola Jones algorithm. On this region again, the Viola Jones Cascade classifier is applied to detect eyes. Viola and Jones introduced Haar like features for eye detection. A haar cascade classifier is again applied that uses these features (see Figure 4) to detect the eye region. Both eye regions are then extracted for further processing. Applying Viola Jones Classifier for eyes detection gives the results as illustrated in Fig 8.

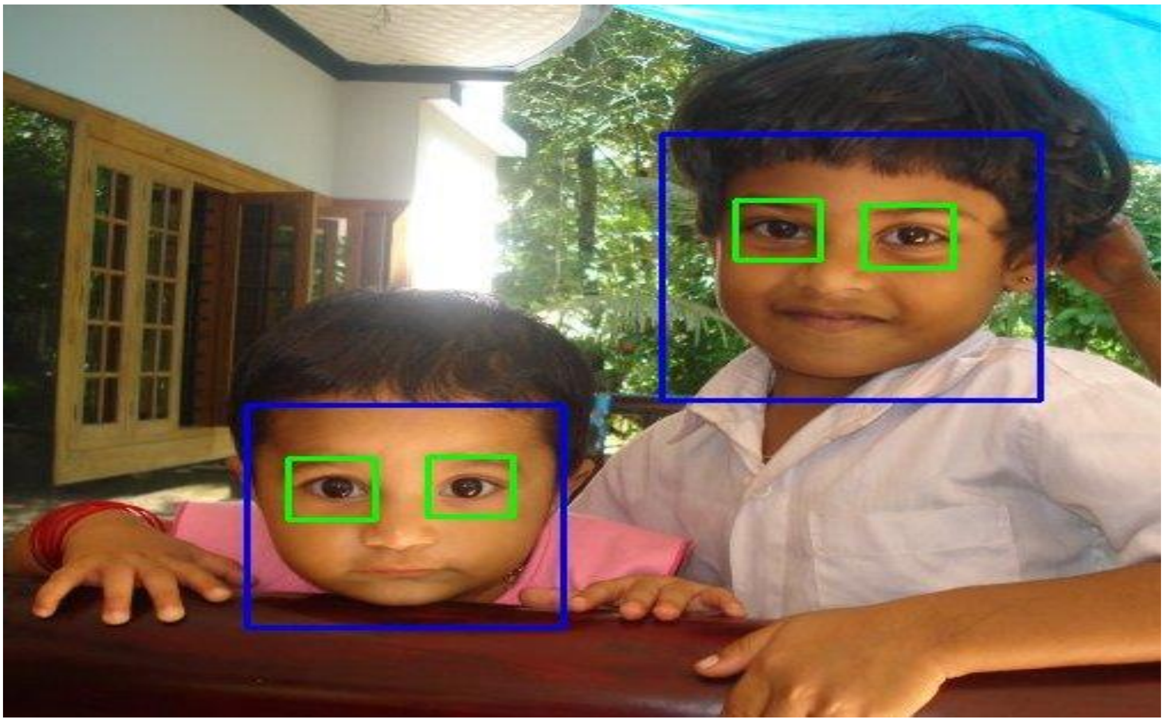


Figure 8: Eye Detection

2.3 CALIBRATION

When a new image is captured through the web camera the first component of the visual gaze estimation pipeline constructs facial feature-pupil center vectors between the detected location of the user's facial features and the detected location of the user's eyes. Since the assumption in this scenario is that the user's head is static, the location of the facial features should not change too much in time, so they can play the role of anchor points.

Then, the vectors collected during the calibration procedure will capture the offsets of the location of the user's eyes responsible for looking at different points on the computer screen; the precise mapping from facial feature-pupil center vectors to points on the computer screen is necessary for accurate visual gaze tracking system. The calibration procedure is done using a set of 25 points distributed evenly on the computer screen. The user is requested to draw points with the mouse in locations that are as near as possible to the locations in the template image.

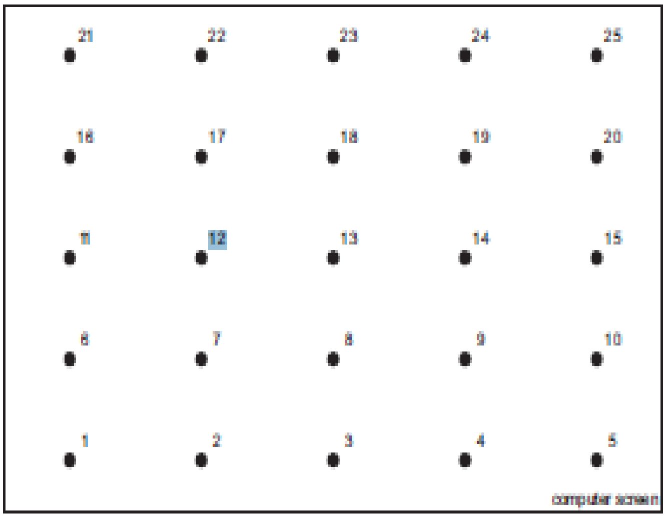


Figure 9: Calibration template image

2.4 ONE-DIMENSIONAL INTERPOLATION

Interpolation involves predicting the co-ordinates of a point given the co-ordinates of points around it. Interpolation can be done in one or more dimensions. One dimensional interpolation involves considering consecutive points along the X-axis with known Y co-ordinates and predicting the Y co-ordinate for a given X co-ordinate.

There are several types of interpolation depending on the number of known points used for predicting the unknown point, and several methods to compute them, each with their own varying accuracy. Methods for interpolation include the classic Polynomial interpolation with Lagrange's formula or spline interpolation using the concept of spline equations between points.

Linear Interpolation: This is the simplest kind of interpolation. It involves simply considering two points such that *x* j

< *num* < *x*

j+1j+1

, where *num* is the unknown point, and considering the slope of the straight line between x[j],y[j]x[j],y[j] and x[j+1],y[j+1]x[j+1],y[j+1], predicts the Y co-ordinate using a simple linear polynomial.

Linear interpolation uses this equation:

y=(y[j]+(interpolant−x[j])(x[j+1]−x[j])×(y[j+1]−y[j])y=(y[j]+(interpolant−x[j])(x[j+1]−x[j])×(y[j+1]−y[j])

Here *interpolant* is the value of the X co-orinate whose corresponding Y-value needs to found.

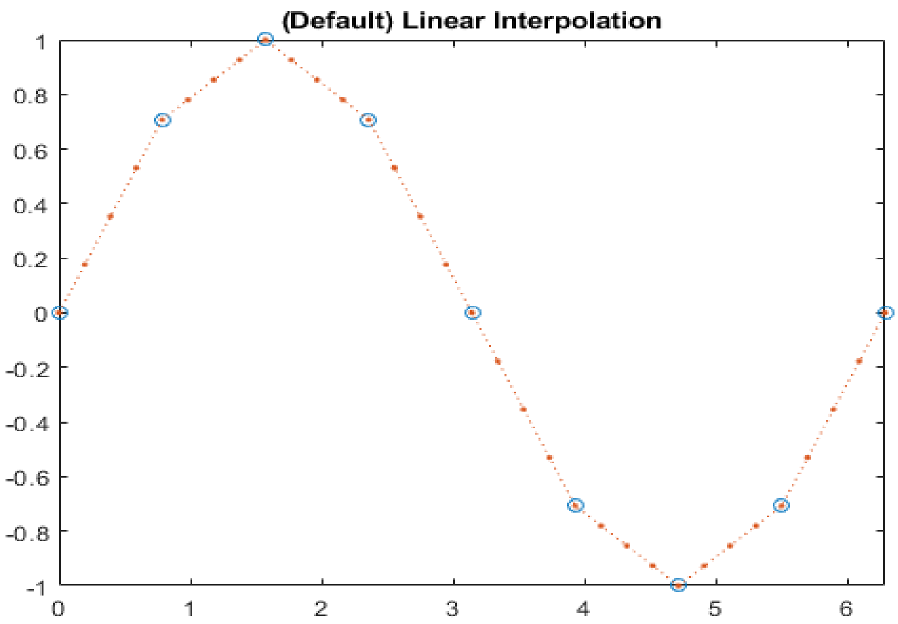


Figure 10: Linear Interpolation

For estimating the direction and point of visual gaze we will be discussing various models:

The problem we are facing is to compute the point and direction of visual gaze in screen coordinates given the current facial feature-pupil center vector in camera coordinates. The facial feature-pupil center vector is constructed upon the current location of the user's eye and the location of the user's facial feature. We can see this problem, as like there is an unknown function that maps our input (facial feature-pupil center vector) to specific output (point on the computer screen).

Furthermore, this problem can be seen as a simple regression problem, namely, suppose we observe a real-valued input variable x (facial feature-pupil center vector) and we wish to use this observation to predict the value of a real-valued target variable t (point on the computer screen). We would like to learn the model (function) that is able to generate the desired output given the input.

Suppose we are given a training set (constructed during the calibration procedure) comprising N observations of x, together with the corresponding observations of the values of t. Our goal is to exploit the training set in order to make predictions of the value ^t of the target variable for some new value ^x of the input variable. If we consider a simple approach based on curve fitting then we want to fit the data using a polynomial function of the form:

$$y(x,w) = w_0 x + \ w_2 x^2 \ \ldots\ldots + wMx^M = \sum_{j=0}^{M} w_j \, x^j$$

where M is the order of the polynomial.

The values of the coefficients will be determined by fitting the polynomial to the training data. This can be done by minimizing an error function that measures the misfit between the function y(x;w), for any given value of w, and the training set data points. One simple choice of error function, which is widely used, is given by the sum of the squares of the errors between the predictions y(xn;w) for each data point xn and the corresponding target values tn, so that we minimize

$$(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2$$

The error function is a nonnegative quantity that would be zero if, and only if, the function y(x;w) were to pass exactly through each training data point. We can solve the curve _tting problem by choosing the value of w for which E(w) is as small as possible. There remains the problem of choosing the order M of the polynomial or so-called model selection. The models chosen for fitting our data are discussed next,

- 2D mapping with interpolation model takes into account 2 calibration points (figure 3, points: 21 and 5 or 1 and 25). The mapping is compute using linear interpolation as follows:

$$S_y = S_{y1} + \frac{y - y1}{y2 - y1}(s_{y2} - s_{y1})$$

$$S_x = S_{x1} + \frac{x - x1}{x2 - x1}(s_{x2} - s_{x1})$$

For example, suppose the screen coordinates and the facial feature-pupil center vectors used for calibration in points P1 and P2 are respectively (sx1; sy1), (x1; y1) and (sx2; sy2), (x2; y2). Then after the measurement of the current facial feature-pupil center vector (x; y) is taken, the screen co- ordinates is computed using the above equations.

- Linear model takes into account 5 calibration points (figure 3, points: 1, 5, 13, 21, 25). The mapping between facial feature-pupil center vectors and screen coordinates is done using the following equations:

$$S_x = a_0 + a_1 x$$
$$s_y = b_0 + b_1 y$$

where (sx; sy) are screen coordinates and (x; y) is the facial feature-pupil center vector. The coefficients a0; a1 and b0; b1 are the unknowns and can be found using least squares approximation.

- Second order model takes into account a set of 9 (figure 3, points: 1, 3, 5, 11, 13, 15, 21, 25, 23) or a set of 25 (figure 3, all points) calibration points. The mapping between facial feature-pupil center vectors and screen coordinates is done using the following equations:

$$S_x = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2$$

$$S_y = b_0 + b_1 x + b_2 y + b_3 xy + b_4 x^2 + b_5 y^2$$

Where (sx; sy) are screen coordinates and (x; y) is the facial feature-pupil center vector. The coefficients a0; a1; a2; a3; a4; a5 and b0; b1; b2; b3; b4; b5 are the unknowns and can be found using least squares approximation.

2.5 EYE BLINK DETECTION

The eye blink is a fast closing and reopening of a human eye. Each individual has a little bit different pattern of blinks. The pattern differs in the speed of closing and opening, a degree of squeezing the eye and in a blink duration. The eye blink lasts approximately 100-400 ms. The face and eye localization algorithm is based on the face detection proposed by Viola and Jone. This detector returns the center and size of a region that contains the face.

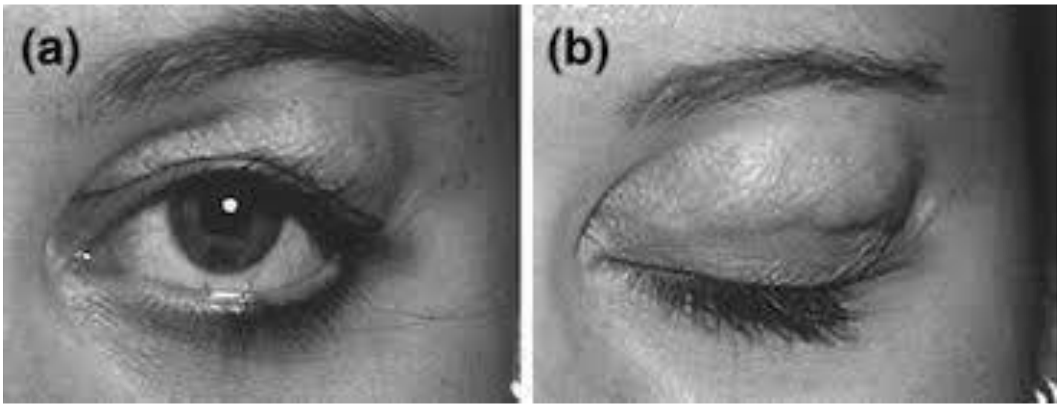


Figure 11. Eye Blink Detection

Once the eyes center points are found, two squares of side equal to half distance between the eyes centers delimit the ROIs that contains the eyes image, as shown in Figure 11(b). The orientation is determined by a frontal face rotation invariant axis formed by eyes centers, which is the reference axis for further processing.

In this eye blinking rate and eye closure duration is measured to detect patient's drowsiness. Because when person felt sleepy at that time his/her eye blinking and gaze between eyelids are different from normal situations so they easily detect drowsiness. In this system the position of irises and eye states are monitored through time to estimate eye blinking frequency and eye close duration.

Percentage value of eye lid closure, maximum closure duration, blink frequency, average value of opening of the eyes and closing velocity of the eyes were calculated. Adaboost based face detector is used for face detection. Eyes were identified by active sharp model. And in this type of system uses a system camera to acquire video and computer vision methods are then applied to sequentially localize face, eyes and eyelids positions to measure ratio of closure. Using these eyes closure and blinking ratio one can detect drowsiness of person.

2.5.1 Calculation of Eye Blinking

The main feature for drowsiness detection is eye blinking. The normal eye blinking rate is varies from 12-19 per minute. The frequency less than this normal range indicates the drowsy condition of a person/driver.
In this project we have considered all the possibilities of an eye. Eye may be fully open, fully closed and partially open/closed. Instead of calculating blinking rate, we have calculated average drowsiness. For eye blinking, detected eye is equated with non-zero, which indicates closed eye. Whereas zero value is considered as fully open /partially open eye. The average drowsiness is calculated as follows:

$$\boxed{D = \text{Consecutive closed eyes} > \text{threshold value}}$$

Where d: drowsiness After calculating % drowsiness, if this value is found to be more than a set threshold value then alert signal is generated for patient. The calculation is based on either eye's status if only non-zero is detected.

2.5.2 Eye State Determination

Finally, the decision for the eye state is made on the basis of distance 'd' calculated in the previous step. If the distance is zero or is close to zero, the eye state is classified as "closed" otherwise the eye state is identified as "open". we detect an eye blink if several consecutive frames capture the quick motion of closing and opening the eyelids.

As eyes are open for most of the time, closing-eye in a video frame is a strong indication of a possible eye blink. We therefore develop an eye blink detection approach that first detects closing-eye frames and then detects eye blink An eye blink action consists of a small number of consecutive frames that captures an open-close-open blink cycle. The results are illustrated in Fig. 12 (a) and (b):

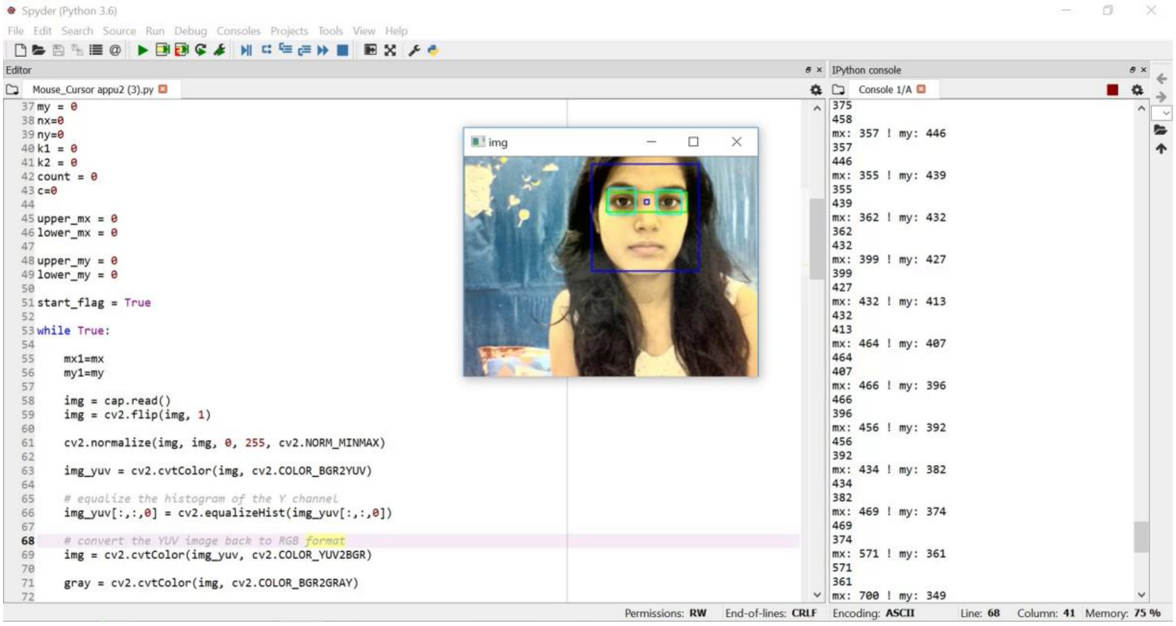


Figure 12 (a): Determines Open Eyes

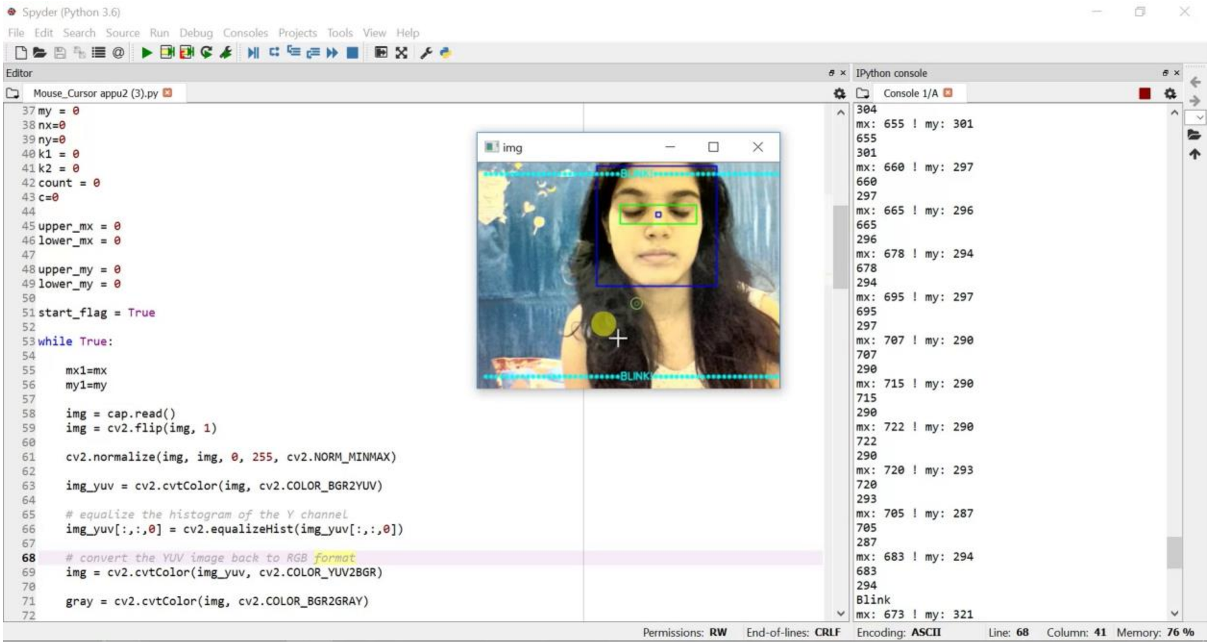


Figure 12 (b): Determines Close Eyes with “Blink” shown on window

2.6 DROWSINESS DETECTION SYSTEM

Long periods of mental effort influence alertness, working memory, judgment and executive control. In industry many incidents and accidents are related to mental fatigue as a result of sustained performance. Moreover, mental fatigue has been identified as one of the main reasons of road accidents worldwide and is believed to account for up to 40% of road crashes. The management of fatigue is therefore very important not only for enhancing productivity, but also for protecting occupational health.

The developed vision-based system can be used as a basis for the human computer interface for disabled users who are capable of blinking voluntarily. The detected eye-blinks are classified as short blinks (<200 ms) or long blinks (>=200 ms). Separate short eye-blinks are assumed to be spontaneous and are not included in the designed eye-blink code. The combination of the long eye-blinks and long breaks between the consecutive eye-blinks (2 s) was used to create a pattern for communication with the computer.

The last step of the algorithm is to determine the person's condition on the basis of a pre-set condition for drowsiness. The average blink duration of a person is 100-400 milliseconds [14]. This is 0.1-0.4 of a second. Hence if a person is drowsy his eye closure must be beyond this interval. We set a time frame of five seconds. If the eyes remain closed for five or more seconds, an alarm is triggered alerting the person.

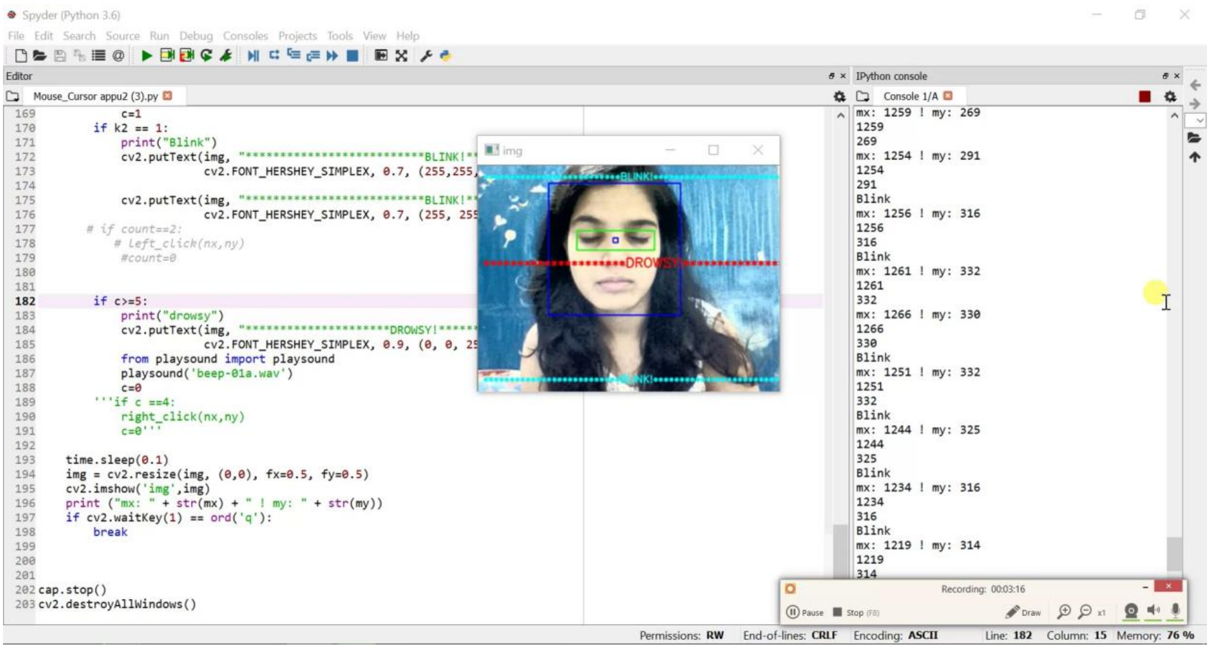


Fig. 13 shows the drowsiness warning.

CHAPTER 3. RESULTS AND MODEL ANALYSIS

In this paper, we researched various ways of camera mouse and human drowsiness techniques. We identified the problem domain and an alternative solution to it. We proposed to use head motion to control the movement of mouse cursor and his eye blinks as a medium to determine the drowsiness. Our system will be helpful for quadriplegic patients, the disabled and elder people with less motion ability to access computer comfortably.

We believe that our system can be also be alternative solution for future computing devices, such as PDA, Tablet, Laptops, and GPS. It may also inspire various PC based application like gaming, entertainment, robotics, etc.

Experiment Accuracy Results

Table I: ACCURACY RATE OF FACE AND EYE DETECTION

Extracted Features	Accuracy
Face features	95%
Eye Features	90%

The tested accuracy rate for face detection and eye detection from the group of ten volunteers was effective, as shown in Table I. There are some definitions of true positive, false positive, and false negative that need to be mentioned. Firstly, true positive means the number of times that the algorithm correctly detects eye blinking of the person. Secondly, false positive can be described as the number of times that the algorithm counts the value of eye blink when the person does not blink. Lastly, the meaning of false negative deals with the number of times the algorithm does not count the value of eye blink when the person actually does blink. An analysis table for the proposed algorithm is shown in Table II.

Two kinds of errors were identified: false detection (the system detected an eye-blink when it was not present) and missed blinks (a present eye-blink that was not detected by the system). The possible distribution of the eye-blink detector output is presented in graphical form in Fig. 13. The correctly detected eye-blinks are denoted as True Positives (TP), false detections are denoted as False Positives (FP), and missed eye-blinks are denoted as False Negatives (FN). Based on these parameters three measures of the system performance were introduced: recall, precision and accuracy, defined by equation given below.

$$\textit{Recall} = \frac{TP}{TP + FN} \qquad \textit{Precision} = \frac{TP}{TP + FP}$$

$$\mathbf{Accuracy} = \frac{\mathbf{TP}}{\mathbf{TP + FP + FN}}$$

These three measures were used to assess the system robustness in detecting eye-blinks .

Possible decision distribution		Actual state	
		Eye-blink present	No eye-blink
Result of detection	Eye-blink detected	TP	FP
	No eye-blink detected	FN	

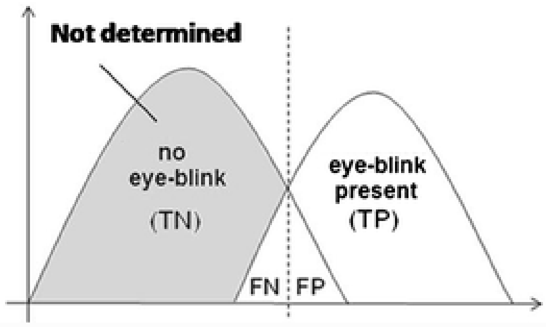


Figure 13: Possible distribution of the eye-blink detector output

Table II: Eye Blink Detection Accuracy

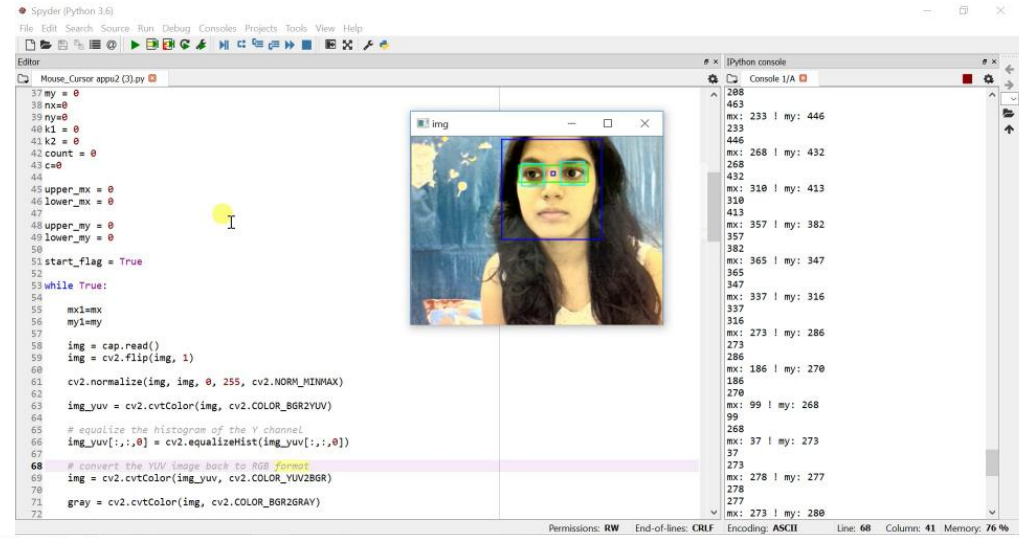
Serial No. of Person	Total Blinks (200)	True Positive %	False Positive %	False Negative %
1	20	85	15	0
2	10	80	20	0
3	30	83.3	13.3	3.33
4	25	96	4	0
5	25	92	4	4
6	15	86.67	13.33	0
7	20	85	10	5
8	20	95	5	0
9	25	100	0	0
10	10	90	10	0
Total	**100%**	**89.297 %**	**9.463 %**	**1.233 %**

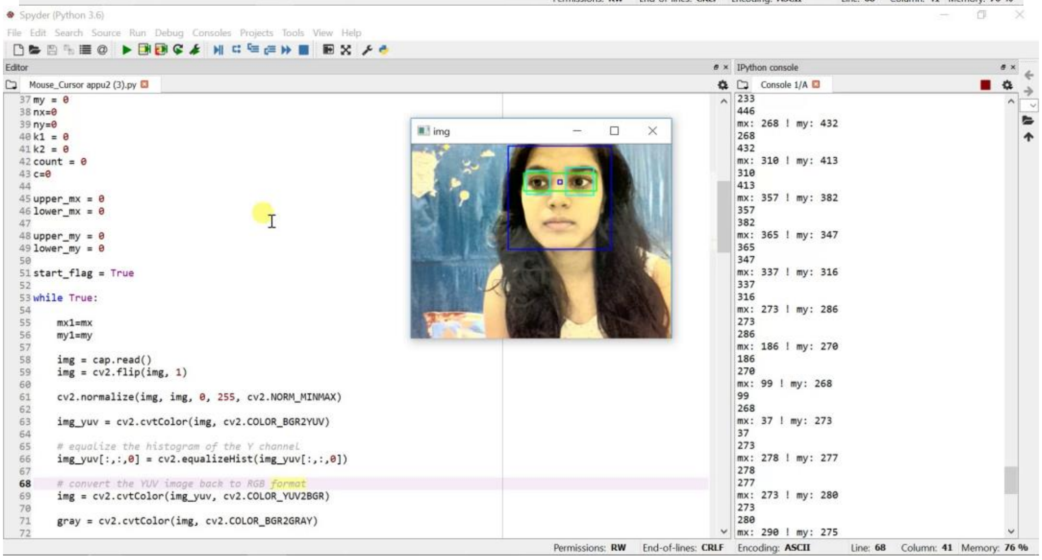
According to Table II, the number of false positives and false negatives was very low when the we applied the methods to check eye blink, the number of true positives is very to the total blink. In addition, we tested with ten volunteers, so the formula for recall and precision is respective:

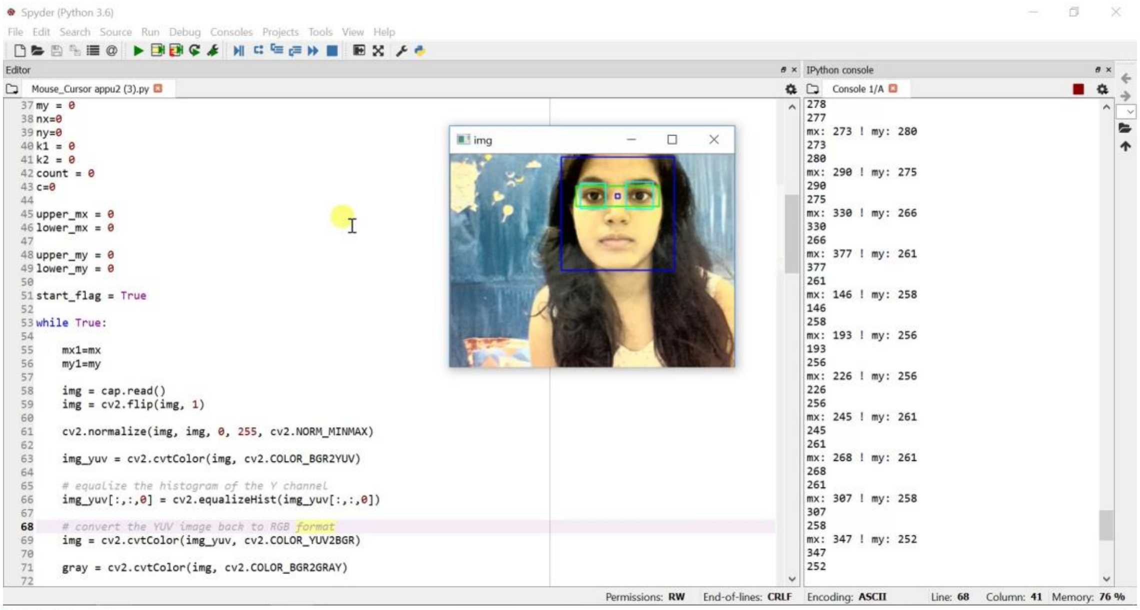
The value for recall is 98.64 percent, whereas the value for precision is 90.326 percent.

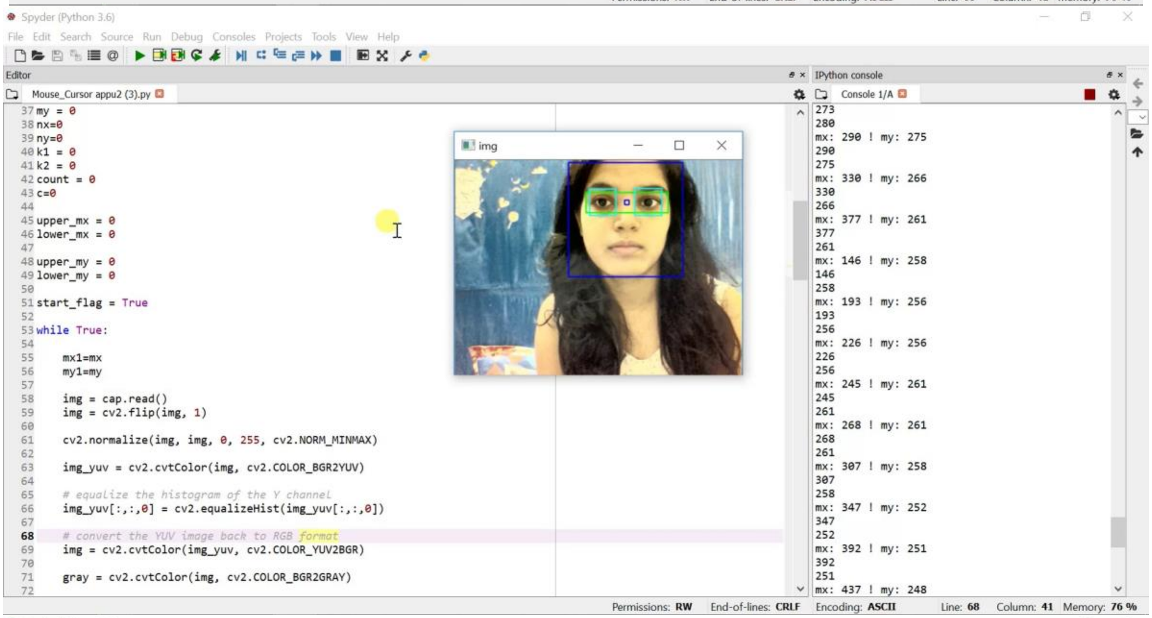
MEASURE	PRECISION (%)	RECALL (%)	ACCURACY (%)
EYE BLINK DETECTION	90.326	98.64	89.29

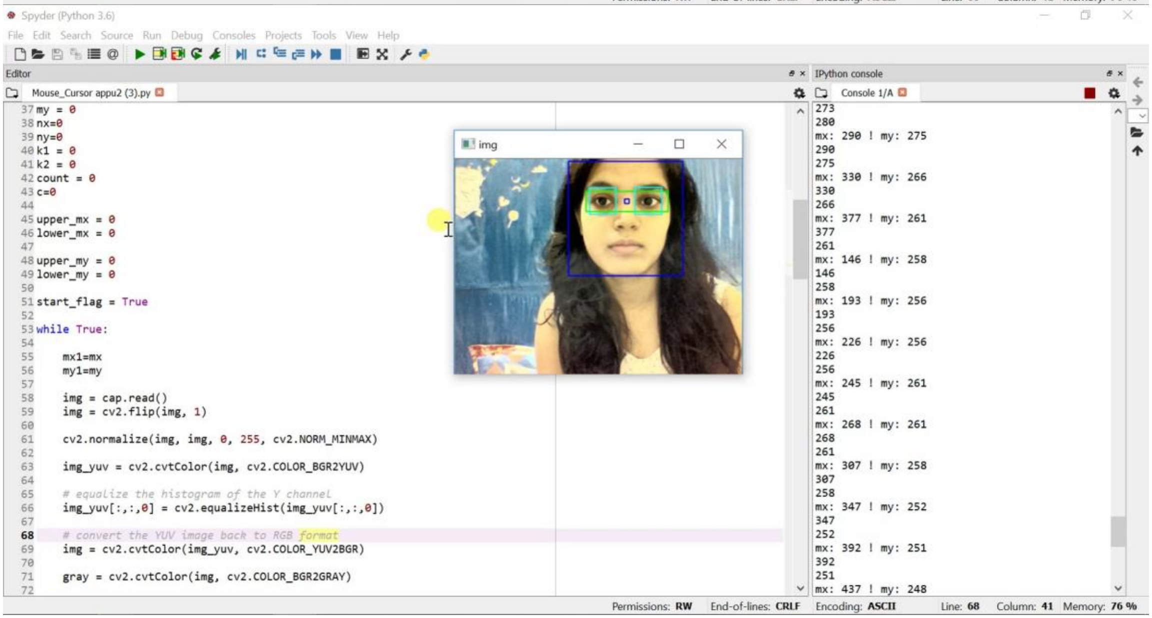
For Cursor Movements: 5 consecutive frames shown here depicts how the mouse cursor is moving through head movements.






Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor
Mouse_Cursor appu2 (3).py
my = 0
nx=0
ny=0
k1 = 0
k2 = 0
count = 0
c=0
upper_mx = 0
lower_mx = 0
upper_my = 0
lower_my = 0
start_flag = True
while True:
mx1=mx
my1=my
img = cap.read()
img = cv2.flip(img, 1)
cv2.normalize(img, img, 0, 255, cv2.NORM_MINMAX)
img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
# equalize the histogram of the Y channel
img_yuv[:,:,0] = cv2.equalizeHist(img_yuv[:,:,0])
# convert the YUV image back to RGB format
img = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img
IPython console
Console 1/A
278
277
mx: 273 ! my: 280
273
280
mx: 290 ! my: 275
290
275
mx: 330 ! my: 266
330
266
mx: 377 ! my: 261
377
261
mx: 146 ! my: 258
146
258
mx: 193 ! my: 256
193
256
mx: 226 ! my: 256
226
256
mx: 245 ! my: 261
245
261
mx: 268 ! my: 261
268
261
mx: 307 ! my: 258
307
258
mx: 347 ! my: 252
347
252
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 68 Column: 41 Memory: 76 %


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor
Mouse_Cursor appu2 (3).py
my = 0
nx=0
ny=0
k1 = 0
k2 = 0
count = 0
c=0
upper_mx = 0
lower_mx = 0
upper_my = 0
lower_my = 0
start_flag = True
while True:
mx1=mx
my1=my
img = cap.read()
img = cv2.flip(img, 1)
cv2.normalize(img, img, 0, 255, cv2.NORM_MINMAX)
img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
# equalize the histogram of the Y channel
img_yuv[:,:,0] = cv2.equalizeHist(img_yuv[:,:,0])
# convert the YUV image back to RGB format
img = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img
IPython console
Console 1/A
273
280
mx: 290 ! my: 275
290
275
mx: 330 ! my: 266
330
266
mx: 377 ! my: 261
377
261
mx: 146 ! my: 258
146
258
mx: 193 ! my: 256
193
256
mx: 226 ! my: 256
226
256
mx: 245 ! my: 261
245
261
mx: 268 ! my: 261
268
261
mx: 307 ! my: 258
307
258
mx: 347 ! my: 252
347
252
mx: 392 ! my: 251
392
251
mx: 437 ! my: 248
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 68 Column: 41 Memory: 76 %


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor
Mouse_Cursor appu2 (3).py
my = 0
nx=0
ny=0
k1 = 0
k2 = 0
count = 0
c=0
upper_mx = 0
lower_mx = 0
upper_my = 0
lower_my = 0
start_flag = True
while True:
mx1=mx
my1=my
img = cap.read()
img = cv2.flip(img, 1)
cv2.normalize(img, img, 0, 255, cv2.NORM_MINMAX)
img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
# equalize the histogram of the Y channel
img_yuv[:,:,0] = cv2.equalizeHist(img_yuv[:,:,0])
# convert the YUV image back to RGB format
img = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img
IPython console
Console 1/A
273
280
mx: 290 ! my: 275
290
275
mx: 330 ! my: 266
330
266
mx: 377 ! my: 261
377
261
mx: 146 ! my: 258
146
258
mx: 193 ! my: 256
193
256
mx: 226 ! my: 256
226
256
mx: 245 ! my: 261
245
261
mx: 268 ! my: 261
268
261
mx: 307 ! my: 258
307
258
mx: 347 ! my: 252
347
252
mx: 392 ! my: 251
392
251
mx: 437 ! my: 248
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 68 Column: 41 Memory: 76 %

CHAPTER 4. CONCLUSION

We have presented a new eye blink monitoring algorithm to detect drowsiness in real time. This technique gives highly accurate results when used under good illumination conditions and executed using a high-resolution camera. This indicates that it has worked well under ideal conditions.

4.1 LIMITATIONS

The possible reason for not so satisfactory results is that our system works for very little head movement and it shows large cursor movements for very less head movements. Secondly very few times, detecting blinks when person is not even blinking which causes lower accuracy results. One more limitation is that for any menu selection by mouse we had to still our head for few minutes.

4.2 FUTURE WORK

Our future work will thus be focused on making of any menu or app selection with help of eyes blinking and also to make our system more robust. We further aim to test the technique on persons with different physical features like wearing eyesight glasses, having facial hair, having some eye disease or having the mouth covered. This would give us more evidence about the versatility of the algorithm. To improve the blink monitoring, the eye point detection must be consistent with each frame. For future work maintaining consistency in the corner detection algorithm will be the priority. Some more effective techniques may be employed to detect the eye feature points more reliably with each video frame. Some Artificial Intelligence (AI) capabilities can be added to enhance reliability of the system. The system shall be able to learn the person’s particular blinking patterns, face expressions, head movement patterns, etc.

REFERENCES

[1] Dushyant Kumar Singh, Dharmender Singh Kushwaha "Analysis of Face Feature based Human Detection Techniques" **IJCTA,** 9(22), 2016, pp. 173-180 International Science Press

[2] Dushyant Kumar Singh "Object Detection: A Key Component in Image & Video Processing" CSI Communications September 2015

[3] Aniket Kudale "Mouse Navigation Control Using Head Motion" *et al*, International Journal of Advanced Research in Computer Science, 3 (3), May –June, 2012, 702-705

[4] A. Benoit and A. Caplier. Motion estimator inspired from biological model for head motion interpretation. In WIAMIS, 2005.

[5] M. Betke, J. Gips, and P. Fleming. The Camera Mouse: Visual tracking of body features to provide computer access for people with severe disabilities. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 10(1):1–10, Mar. 2002.

[6] Paul A. Viola, and Michael J. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," IEEE

[7] Drowsy Driving, Facts and Stats: Drowsy Driving – Stay Alert, Arrive Alive. http: / / drowsydriving.org/about/ facts- and- stats/ , 2016.

[8] Osamu TAKAMI, Kazuaki MORIMOTO, Tsumoru OCHIAI and Takakazu ISHIMATSU " Computer Interface to Use Head and Eyeball Movement for Handicapped People" 0-7803-2559-1195 $4.00 *0* 1995 IEEE

[9] Yuan-Pin Lin, Yi-Ping Chao, Chung-Chih Lin, and Jyh-Hong Chen, "Webcam Mouse Using Face and Eye Tracking in Various Illumination Environments", Proceedings of 2005 IEEE, Engineering in Medicine and Biology 27th Annual Conference, Shanghai, China, September 2005.

[10] M. Singh, G. Kaur, "Drowsiness detection on eye blink Duration using algorithm", International Journal of Emerging Technology and Advanced Engineering , Volume 2, Issue 4, April 2012.