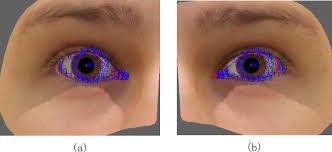
ACKNOWLEDGEMENT

I would like to thank Prof. Mr. **Suraj prakash yadav** for her valuable support throughout the development of this thesis. This thesis really would not have been reach to an end without her guidance and patience. I would like to thank to Mr. **Masood Ahmed Shaik** for her guidance at the beginning of this work. I also would like to thank to **Ms. Savitha Ramesh**. Especially to my department Test Engineering for letting me to involve in this thesis work. Finally I am grateful to my parents and my colleagues at work for their continuous encouragement.

ABSTRACT

The problem of eye detection in face images is very important for a large number of applications ranging from face recognition to gaze tracking. In this paper, we propose a new algorithm for eyes detection that uses iris geometrical information for determining in the whole image the region candidate to contain an eye, and then the symmetry for selecting the couple of eyes. The novelty of this work is that the algorithm works on complex images without constraints on the background, skin colour segmentation and so on. Different experiments, carried out on images of subjects with different eyescolours, some of them wearing glasses, demonstrate the effectiveness and robustness of the proposed algorithm***.***

******

**List of figures**

FIG 1.1. Histogram of oriented gradient extraction from face

Fig1.2. selection of eye using HOG

Fig 2.1. Existing system

Fig 2.1. Existing system

Fig 3.1 Proposed system

Fig 3.2 block diagram

Fig 5.1 face detectiom

Fig 5.2. Sequence diagram

Fig 5.3 ER diagram

**List of tables**

Tab 7.1 test cases 1

Tab 7.2 test cases 2

**TABLE OF CONTENTS**

**Contents Page No.**

Acknowledgement i

Abstract ii

List of figures iii

List of Tables iv

**CHAPTER-1 INTRODUCTION 7**

1.1 About the Project 7

1.2 Scope 10

**CHAPTER-2 EXISTING SYSTEM 12**

2.1 Introduction 12

2.2 Block diagram/Architecture 13

2.3 Disadvantages 14

**CHAPTER-3 PROPOSED SYSTEM 15**

3.1 Introduction 15

3.2 Block diagram/Architecture 16

3.3 Advantages 17

**CHAPTER-4 SYSTEM REQUIREMENT SPECIFICATION 18**

4.1 Functional and Non-functional Requirements 18

4.2 Software and Hardware Requirement **19**

**CHAPTER - 5 SYSTEM DESIGN 32**

5.1 UML diagrams

5.2 ER Diagram

**CHAPTER-6 SYSTEM IMPLEMENTATION 29**

6.1 sample code

**CHAPTER-7 TESTING 32** 7.1 Test Cases

**CHAPTER-8 SIMULATION RESULTS**

8.1 Screenshots

**CHAPTER - 9 CONCLUSION 34**

**BIBLIOGRAPHY 36**

**CHAPTER-1**

**INTRODUCTION**

**1.1 About the project**

Reliable object detection is very important in computer vision and robotics applications. The histogram of oriented gradients (HOG) is established as one of the most popular hand-crafted features, which along with support vector machine (SVM) classification provides excellent performance for object recognition. We investigate dimensionality deduction on HOG features in combination with SVM classifiers to obtain efficient feature representation and improved classification performance. In addition to lean HOG features, we explore descriptors resulting from dimensionality reduction on histograms of binary descriptors. We consider three-dimensionality reduction techniques: standard principal component analysis, random projections, a computationally efficient linear mapping that is data independent, and locality preserving projections (LPP), which learns the manifold structure of the data. Our methods focus on the application of eye detection and were tested on an eye

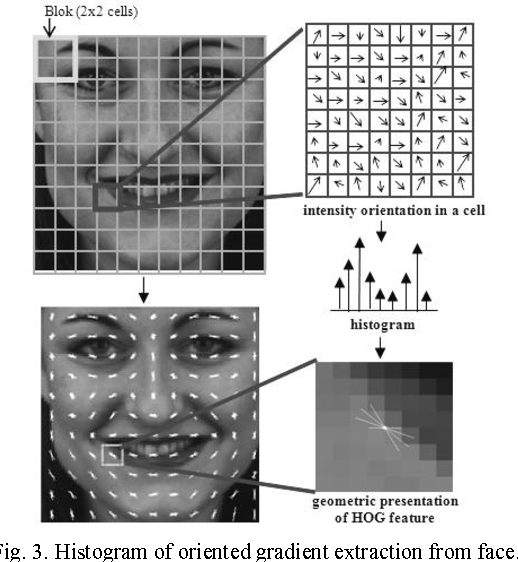
******

FIG 1.1. Histogram of oriented gradient extraction from face

Data base created using the Bio ID and FERET face databases. Our results indicate that manifold learning is beneficial to classification utilizing HOG features. To demonstrate the broader usefulness of lean HOG features for object class recognition, we evaluated our system's classification performance on the CalTech-101 dataset with favourable outcomes.

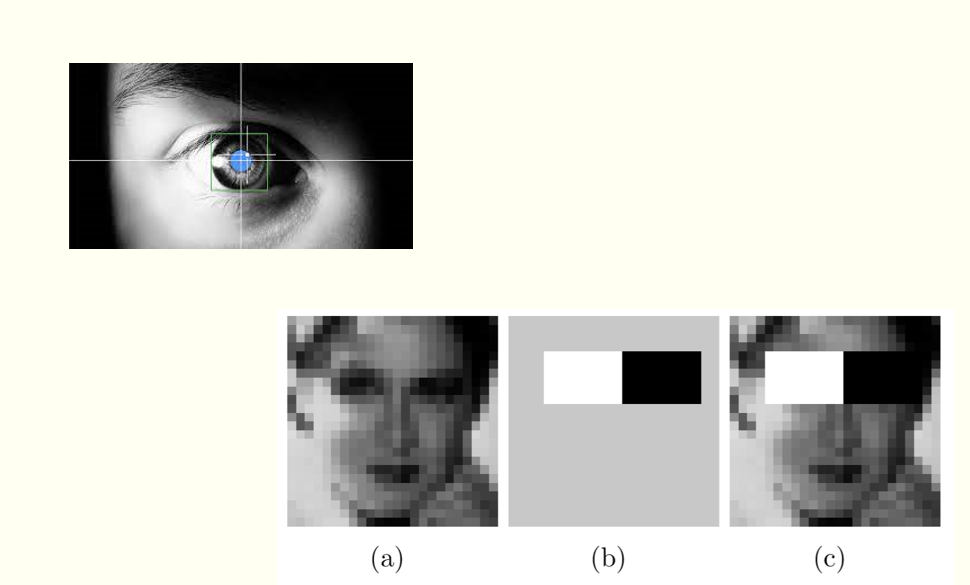


Fig1.2. selection of eye using HOG

**1.2 Scope**

* Based on the optical tracking of corneal reflections, eye tracking technology allows to record the eye position and its movement, thus analyzing the eye movements and gaze positions in both 2D and 3D environment. It helps to examine the human processing of visual information for diagnostic and interactive applications.
* In recent years, eye tracking technology has been experiencing rapid development with improvement in the stability, accuracy, and sampling rates.  The acquisition of eye tracking startups by large companies and the rollout of software and several devices that support eye tracking is also on the rise. Recently, VR headset firms invested a large amount in eye tracking technology to push it further and create an immersive experience. Facebook and Google have acquired eye tracking startups, Eye Tribe and Eyefluence respectively, and are expected to implement the technology in their future applications. A kick starter-funded project “Fove” is the first VR (virtual reality) headset to have embedded eye-tracking. Investing and funding in new startups and acquisition of companies boosts the demand for eye tracking technology.
* There’s a future scope that eye tracking will be a standard feature of a new generation of laptops, desktop monitors, and smartphones, setting the stage for a huge reassessment of the way how the devices communicate with us or we communicate with them. Mobile phones, such as iPhone X or the Xiaomi Mi8, and laptops with user-facing depth-sensing cameras can be upgraded to run 3D eye tracking without the need for an additional eye tracking device.

**CHAPTER-2**

**Existing system**

**2.1 Introduction**

Current most of the objection detection model just detects the region of the eye using Haar cascade detector which use filters to detect. This is basically a machine learning based approach where a cascade function is trained from a lot of images both positive and negative. Based on the training it is then used to detect the objects in the other images.

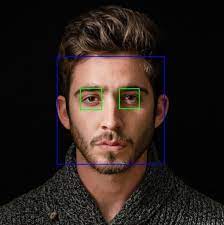
So how this works are they are huge individual .xml files with a lot of feature sets and each xml corresponds to a very specific type of use case.

Fig 2.1. Existing system

**2.2 Block diagram/Architecture**

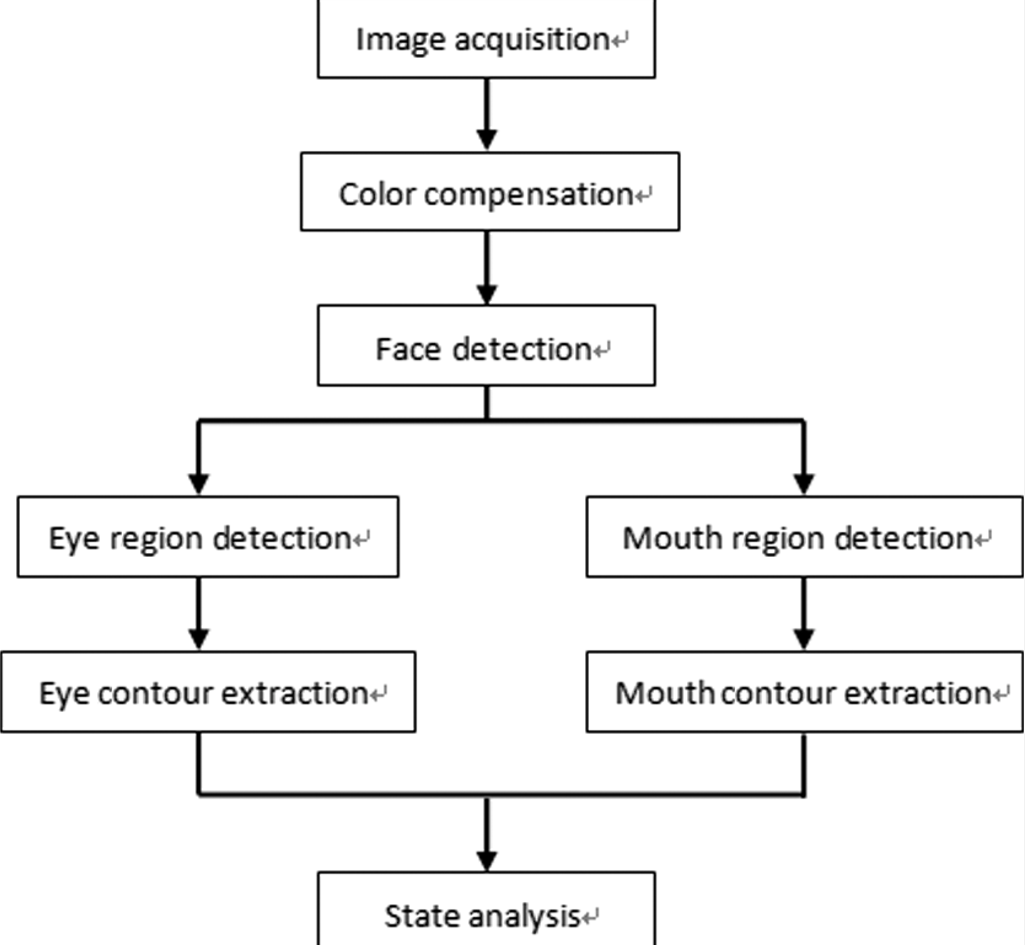
****

Fig 2.1. Existing system

**2.3 Disadvantages**

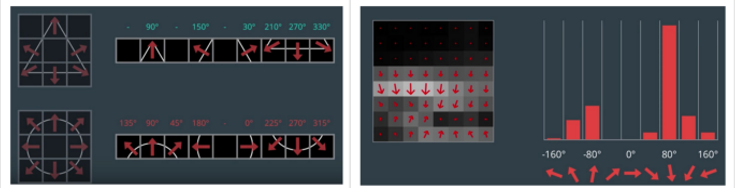
It detects the region with bounding box which contains the eye but not the eye curve, just by using HAAR cascade method

**CHAPTER-3**

**Proposed system**

**3.1 Introduction**

We are using features extraction and feature recognition to recognize the region which contains the eye then advance image processing techniques to get the boundary of the eye which are used in all modern face recognition model. To allow for some variability in shape, we’ll use features known as Histogram of Oriented Gradients

******

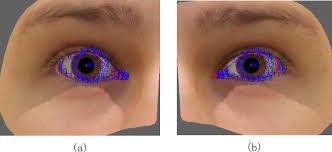
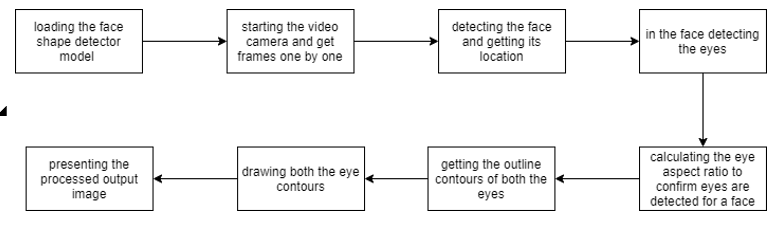
******

Fig 3.1 Proposed system

The idea of HOG is instead of using each individual gradient direction of each individual pixel of an image, we group the pixels into small cells. For each cell, we compute all the gradient directions and group them into a number of orientation bins. We sum up the gradient magnitude in each sample. So stronger gradients contribute more weight to their bins, and effects of small random orientations due to noise is reduced. This histogram gives us a picture of the dominant orientation of that cell. Doing this for all cells gives us a representation of the structure of the image.

**3.2 Block diagram/Architecture**

Fug 3.2 block diagram

**3.3 Advantages**

It uses advanced features extraction techniques used by all modern face recognition projects by companies like Google, apple. It gives us the exact eye instead of a box containing the eye.

**CHAPTER-4**

**SYSTEM REQUIREMENT SPECIFICATIONS**

**4.1 Functional requirements**

## 1.Lighting

Almost all eye trackers use the reflection of infrared (or near-infrared) light from the pupil (with the help of a few algorithms) to follow the movement of the eyes. This is often used to create a [bright pupil](https://en.wikipedia.org/wiki/File:Bright_pupil_by_infrared_or_near_infrared_illumination.jpg) that can reveal the location of the eyes.

The bright pupil method works well in controlled environments – as long as the lighting condition is stable, the experiment can be carried out, without too much concern for lighting levels (within reason of course).

Problems with this method arise when the eye tracking is carried out outside, in the wild. Differing levels of infrared light can interfere with the signal that the eye tracking camera receives, ultimately reducing the accuracy of the tracking.

## 2.Movement

Some of the earliest eye tracking devices required that the respondent’s head was attached to it (temporarily, I should add). This method ensured that there wasn’t any movement that could disrupt the signal collected by the camera, but also wasn’t exactly the most realistic set of conditions for an experiment.

## 3.Obstruction

Eye tracking requires a clean line of sight between the eyes and the camera. This seems obvious for an entire experiment, but even brief interruptions can take away crucial data.

While the first step in preventing interruptions is to make sure that respondent knows not to obstruct their eyes, the specifications of the eye tracker can also have an impact.

**4.2 Non functional requirements**

* **Security**

It should be secure.

* **Reliability**

Highly reliable in a properly secured environment.

* **Availability**

The user will be able to easily download and install the system software.

* **Testing**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner.

* 1. **Software Requirement Specification**

Front End : HTML

Back End : JAVA

Operating System : Windows-10

Frame Work : JAVA

* 1. **Hardware Requirements**

Processor : Intel Pentium

Hard disk : Minimum 500MB

RAM : Minimum 2GB

**CHAPTER-5**

**SYSTEM DESIGN**

The design document that we develop during this phase is the blueprint of the software. It describes how the solution to the customer problem is to be built. Since solution to complex problems isn’t usually found in the first try, iterations are most likely required. This is true for software design as well. For this reason, any design strategy, design method, or design language must be flexible and must easily accommodate changes due to iterations in the design. Any technique or design needs to support and guide the partitioning process in such a way that the resulting sub-problems are as independent as possible from each other and can be combined easily for the solution to the overall problem.

Sub-problem independence and easy combination of their solutions reduces the complexity of the problem. This is the objective of the partitioning process. Partitioning or decomposition during design involves three types of decisions: - Define the boundaries along which to break; Determine into how money pieces to break; and identify the proper level of detail when design should stop and implementation should start. Basic design principles that enable the software engineer to navigate the design process suggest a set of principles for software design, which have been adapted and extended in the following list: A good designer should consider alternative approaches, judging each based on the requirements of the problem, the resources available to do the job.

The design should be traceable to the analysis model. Because a single element of the design model often traces to multiple requirements, it is necessary to have a means for tracking how requirements have been satisfied by the design model.

The design should not repeat the same thing. Systems are constructed using a set of design patterns, many of which have likely been encountered before. These patterns should always be chosen as an 11 alternative to reinvention. Time is short and resources are limited! Design time should be invested in representing truly new ideas and integrating those patterns that already exist. The design should “minimize the intellectual distance” between the software and the

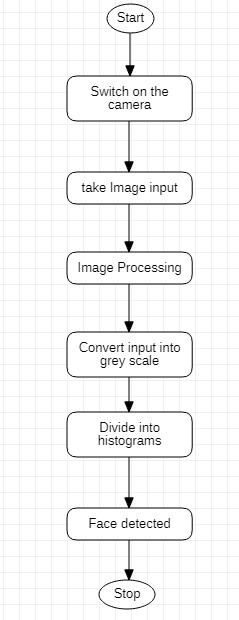
******

Fig 5.1 face detectiom

* Open the camera and take the input
* Analyze the image.
* Divide it into histograms.
* Face is detected

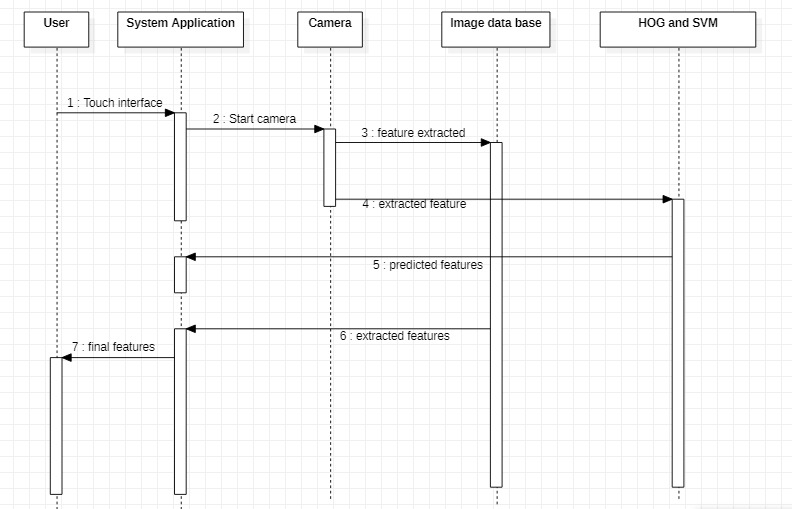
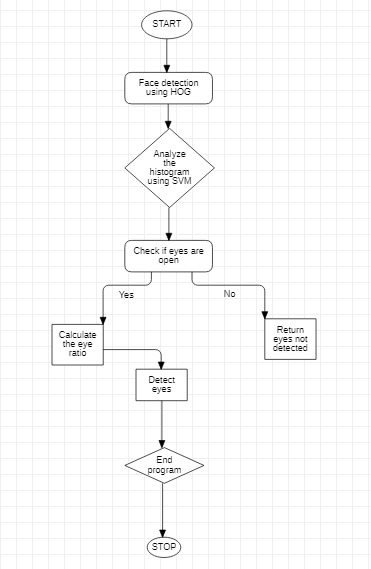
******

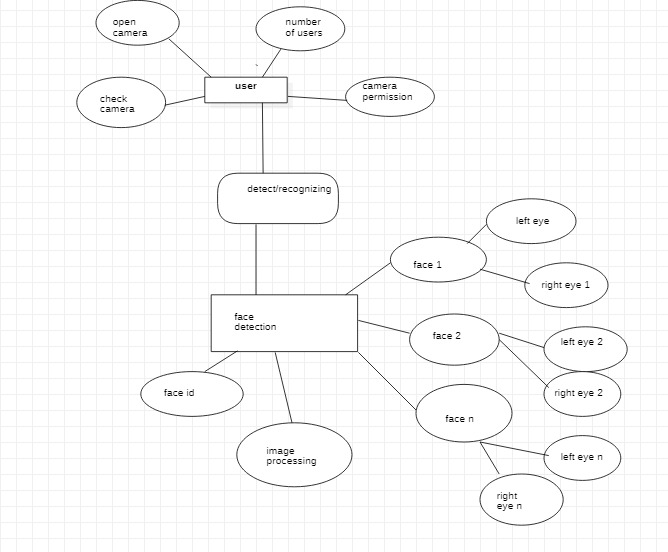
Fig 5.2. Sequence diagram

******

* + Once the face is detected analyze the histograms of different grids to extract the features using svm .
  + Check if eyes are closed or not.
  + Do the computation and calculate the eye ratio
  + Eyes are detected!
  + Return the image to user

**5.2 ER DIAGRAM**

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

******fig 5.3 ER diagram

**CHAPTER-6**

**SYSTEM IMPLEMENTATION**

**6.1 SAMPLE CODE**

##Importing libraries

from imutils import face\_utils as Futility

import cv2 as ComputerVision

import systemcheck

import imutils as ImageUtilities

import dlib as digitalLibrary

from scipy.spatial import distance as length

import cv2

ColortoBWconverter = ComputerVision.COLOR\_BGR2GRAY

THICKNESS = 2

COLOR = (0,0,255)

sensitivity = 0.19  # Threshold to consider eye as closed

ww = 450

det = 0

countFrm = 15  # increase eye close time

check = digitalLibrary.get\_frontal\_face\_detector()

forecast = digitalLibrary.shape\_predictor("faceShape.dat")

(LeftEyeStart, LeftEyeEnd) = Futility.FACIAL\_LANDMARKS\_68\_IDXS["left\_eye"]

(RightEyeStart, RightEyeEnd) = Futility.FACIAL\_LANDMARKS\_68\_IDXS["right\_eye"]

Internalcamera = ComputerVision.VideoCapture(0)

# Initialize the Internal Camera

def getRatio(gridEye):

    global length

    ratio1 = length.euclidean(gridEye[1], gridEye[5])

    ratio2 = length.euclidean(gridEye[2], gridEye[4])

    ratio3 = length.euclidean(gridEye[0], gridEye[3])

    ratio = (ratio1 + ratio2) / (2.0 \* ratio3)

    return ratio

def drawEyeCurve(eyeCurve):

    global ComputerVision

    global RawImage

    ComputerVision.drawContours(RawImage, [eyeCurve], -1, COLOR, thickness=THICKNESS)

def getLeftEyeCurveDetails(Lrat):

    global ComputerVision

    x = ComputerVision.convexHull(Lrat)

    return x

def getRightEyeCurveDetails(Rrat):

    global ComputerVision

    x = ComputerVision.convexHull(Rrat)

    return x

while True:

    done,RawImage =Internalcamera.read()

    # Read the Image from Internal Camera

    if (done == 0):

        print("Unable to get Image data")

        continue

    RawImage = ImageUtilities.resize(RawImage, width= ww)

    #Resizing the Image in suitable size

    BWimg = ComputerVision.cvtColor(RawImage, ComputerVision.COLOR\_BGR2GRAY)

    # Making the Color Black&White for size Reduction and faster Processing

    BWimg = ImageUtilities.resize(BWimg, width= ww)

    #Resizing the B&W Image in suitable size

    multiFaceData = check(BWimg, 0)

    # Get data of all the multiple faces found

    det = 0

    #print(det)

    for singleFaceData in multiFaceData:

        # getting the detected face locations

        x1,y1,x2,y2 = singleFaceData.left(),singleFaceData.top(),singleFaceData.right(),singleFaceData.bottom()

        # Take all the faces one by one from image

        det = 0

        geometry = forecast(BWimg, singleFaceData)

        # Try Predicting Location of the face

        geometry = Futility.shape\_to\_np(geometry)

        # Change the face data into Numpy array for numerical processing

        LEye = geometry[LeftEyeStart:LeftEyeEnd]

        #Get the Geometrical details of LeftEye

        if(LEye.any()):

            det = 1

        Lratio = getRatio(LEye)

        #Get the Aspect ratio of the left eye

        if(Lratio < 0):

            continue

        REye = geometry[RightEyeStart:RightEyeEnd]

        #Get the Geometrical details of RightEye

        if(REye.any()):

            det = 1

        Rratio = getRatio(REye)

        #Get the Aspect ratio of the right eye

        if(Rratio < 0):

            continue

        finalRatio = (Lratio + Rratio )/ 2.0

        # Get the arithmetic mean of both ratios

        LEyeCurve = getLeftEyeCurveDetails(LEye)

        drawEyeCurve(LEyeCurve)

        #Get the curve definition of Left Eye and Draw it

        REyeCurve = getRightEyeCurveDetails(REye)

        drawEyeCurve(REyeCurve)

        # drawing rectangle around the detected face

        ComputerVision.rectangle(RawImage, (x1,y1), (x2,y2), (255,0,255), 3)

        #Get the curve definition of Right Eye and draw it

    ComputerVision.imshow("Eye Detection", RawImage)

    #Show thw Update RawImage on Screen

    got = ComputerVision.waitKey(125) & 0xFF

    if got == 27:

        break

cv2.destroyAllWindows()

Internalcamera.release()

**CHAPTER-7**

**TESTING**

**7.1 Test cases**

|  |  |  |
| --- | --- | --- |
| Test case 1: Face detection | | Priority (H, L): High |
| Test Objective: To check whether the face is detected. | | |
| Test Description: Once the camera is switched on take the input image and divide it into histogram. | | |
| Requirements Verified: Yes | | |
| Test Environment: The image is captured by read() function in python. | | |
| Test Setup/Pre-Conditions: Make sure the face is detected. | | |
| **Actions** | **Expected Results** | |
| Face detection. | “ Connection successful” | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

Tab 7.1 test cases 1

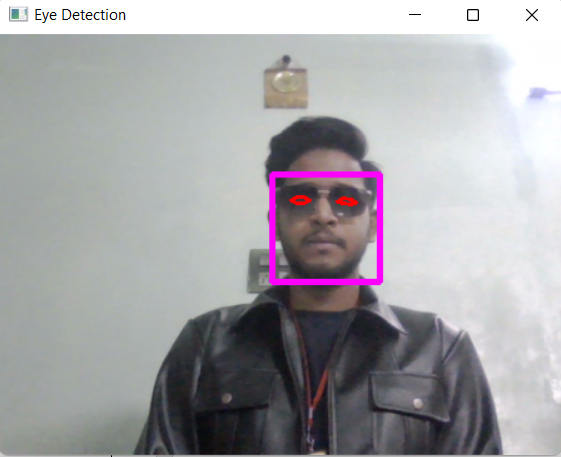
|  |  |  |
| --- | --- | --- |
| Test case 1: Face detection | | Priority (H, L): High |
| Test Objective: To check whether the face is detected. | | |
| Test Description: Once the camera is switched on take the input image and divide it into histogram. | | |
| Requirements Verified: Yes | | |
| Test Environment: The image is captured by read() function in python. | | |
| Test Setup/Pre-Conditions: Make sure the face is detected. | | |
| **Actions** | **Expected Results** | |
| Face detection. | “ Connection successful” | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

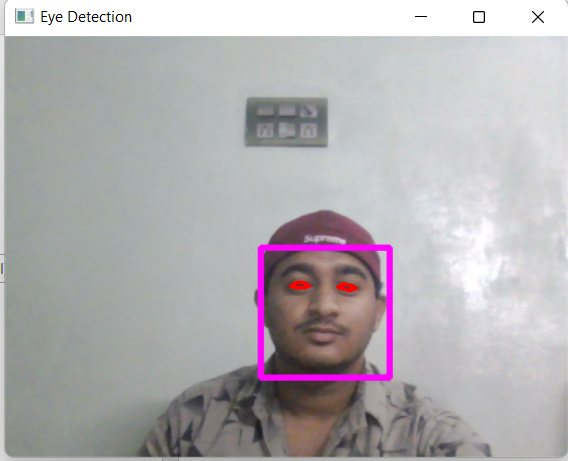
Tab 7.2 test cases 2

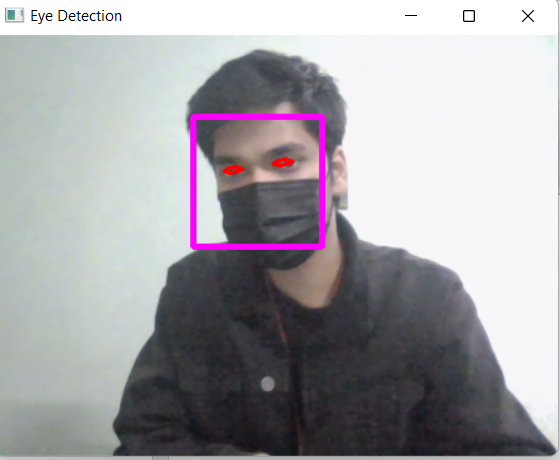
**CHAPTER-8**

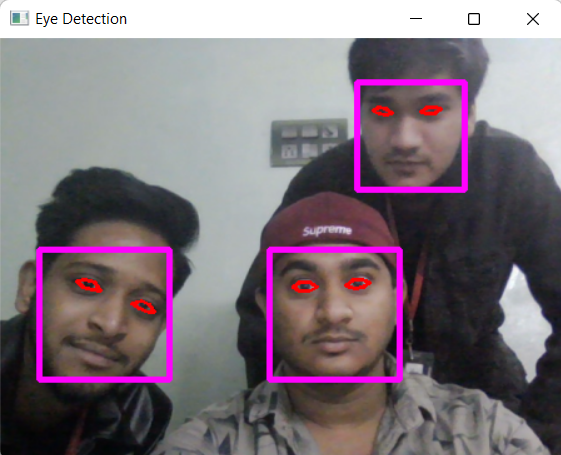
**SIMULATION RESULTS**

**8.1 Screenshots**

****

****

****

****

**CHAPTER-9**

**CONCLUSION**

In this study, a real-time system capable of face tracking, eye area tracking and extracting eye features was developed. The system’s face tracking algorithm is based on CAMSHIFT algorithm developed by [4] which a feature-based face is tracking algorithm searching for the colour of the human skin in images. Detection of eye areas after face localization is based on Principal Components Analysis technique which was previously used for face recognition. In order to detect eyes with different sizes without so much computational work and without using databases of different sizes for each image, we have developed an eigenface based method called adaptive Eugenie method. After detection of the eye areas, features of the eyes, region#1 which is the area containing eye pupil and iris and region#2 which is the visible eyeball area, are determined. The extraction of detailed location and shape information of eye features is performed by applying different image pre-processing

And region segmentation based on boundary estimation using edge features. Boundary of the final regions are found by circle detection algorithms or by an active contour (Snake) The system works well with cheap cameras and does not require camera lens calibration. Although the system performance is highly dependent on the system settings done by the user before the tracking process, it is very easy to make this adjustment by the help of friendly user interface.

From the obtained results it is clear that histogram of gradient provides better result in cataract detection as it directly identify the white layer responsible for cataract in an eye. It is less complexity and less time consuming when compared to exiting methods like support machine vector. This work can also further applied for traffic sign detection.

**CHAPTER-10**

**BIBLOGRAPHY**

[1] J.Ivins and J. Porrill, (2000). “Everything you always wanted to know about snakes (but were afraid to ask)". AIVRU Technical Memo 86, Artifcial Intelligence Vision Research Unit, University of Shefield, England.

[2] D. J. Williams and M. Shah, (1992). “A fast algorithm for active contours and curvature estimation", CVGIP: Image Understanding, vol. 55, pp. 14-26.

[3] E. Hjelmas and B. K. Low, (2001). “Face detection: A survey”. Computer Vision and Image Understanding, vol. 83, pp. 236-274.

[4] J. Canny, (1986). “A Computational Approach to Edge Detection”. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 8, no. 6, pp. 679-698.

[5] S.A. Sirohey, (1993). “Human Face Segmentation and Identification”. Technical Report CS-TR-3176, Univ. of Maryland.