LEVERAGING MACHINE LEARNING TO BUILD A DRIVER DROWSINESS DETECTION SYSTEM

A MINI PROJECT REPORT

Submitted by

Siva Shanker P S (962219104105)

Vishal V S (962219104116)

Roshan Roby (962219104090)

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ST. XAVIER'S CATHOLIC COLLEGE OF ENGINEERING

ANNA UNIVERSITY::CHENNAI 600 025

JUNE 2022

ANNA UNIVERSITY::CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this mini-project report"**LEVERAGING MACHINE LEARNING TO BUILD A DRIVER DROWSINESS DETECTION SYSTEM**" is the bonafide work of **"Siva Shanker P S(962219104105), Roshan Roby(962219104090)** and **Vishal V S(962219104116)"**who carried out the mini project work under my supervision.

SIGNATURE **SIGNATURE**

Mrs. P.R. Sheebha Rani, M.E., M.B.A., Dr. A. Bamila VirginLouis, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT **PROJECT CO-ORDINATOR**

Computer Science and Engineering
St. Xavier's Catholic College of Engineering
Chunkankadai-629003

Assistant Professor
Computer Science and Engineering
St. Xavier's Catholic College of Engineering
Chunkankadai-629003

Submitted for the Mini Project Work (CS8611) Viva held on …….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Drowsiness and Fatigue of drivers are amongst the significant causes of road accidents. Every year, they increase the amounts of deaths and fatalities injuries globally. In this paper, a module for Advanced Driver Assistance System (ADAS) is presented to reduce the number of accidents due to drivers fatigue and hence increase the transportation safety; this system deals with automatic driver drowsiness detection based on visual information and Artificial Intelligence. We propose an algorithm to locate, track, and analyse both the drivers face and eyes to measure PERCLOS, a scientifically supported measure of drowsiness associated with slow eye closure.

Keywords-Drowsiness detection, ADAS, Face Detection and Tracking, Eyes Detection and Tracking, Eye state, PERCLOS.

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Currently, transport systems are an essential part of human activities. We all can be victim of drowsiness while driving, simply after too short night sleep, altered physical condition or during long journeys. The sensation of sleep reduces the driver's level of vigilance producing dangerous situations and increases the probability of an occurrence of accidents. Driver drowsiness and fatigue are among the important causes of road accidents. Every year, they increase the number of deaths and fatalities injuries globally. In this context, it is important to use new technologies to design and build systems that are able to monitor drivers and to measure their level of attention during the entire process of driving. In this paper, a module for ADAS (Advanced driver assistance System) is presented in order to reduce the number of accidents caused by driver fatigue and thus improve road safety. This system treats the automatic detection of driver drowsiness based on visual information and artificial intelligence. We propose an algorithm to locate, track and analyse both the driver face and eyes to measure PER- CLOS (percentage of eye closure).

1.2 FACTS & STATISTICS

Our current statistics reveal that just in 2015 in India alone, 148,707 people died due to car related accidents. Of these, at least 21 percent were caused due to fatigue causing drivers to make mistakes. This can be a relatively smaller number still, as among the multiple causes that can lead to an accident, the involvement of fatigue as a cause is generally grossly underestimated. Fatigue combined with bad infrastructure in developing countries like India is a recipe for disaster. Fatigue, in general, is very difficult to measure or observe unlike alcohol and drugs, which have clear key indicators and tests that are available easily. Probably, the best solutions to this problem are awareness about fatigue-related accidents and promoting drivers to admit fatigue when needed. The former is hard and much more expensive to achieve, and the latter is not possible without the former as driving for long hours is very lucrative. When there is an increased need for a job, the wages associated with it increases leading to more and more people adopting it. Such is the case for driving transport vehicles at night. Money motivates drivers to make unwise decisions like driving all night even with fatigue. This is mainly because the drivers are not themselves aware of the huge risk associated with driving when fatigued. Some countries have imposed restrictions on the number of hours a driver can drive at a stretch, but it is still not enough to solve this problem as its implementation is very difficult and costly.

1.3 PRODUCT SCOPE

There are many products out there that provide the measure of fatigue level in the drivers which are implemented in many vehicles. The driver drowsiness detection system provides the similar functionality but with better results and additional benefits. Also, it alerts the user on reaching a certain saturation point of the drowsiness measure.

1.4 PROBLEM DEFINITION

Fatigue is a safety problem that has not yet been deeply tackled by any country in the world mainly because of its nature. Fatigue, in general, is very difficult to measure or observe unlike alcohol and drugs, which have clear key indicators and tests that are available easily. Probably, the best solutions to this problem are awareness about fatigue-related accidents and promoting drivers to admit fatigue when needed. The former is hard and much more expensive to achieve, and the latter is not possible without the former as driving for long hours is very lucrative.

CHAPTER 2

RELATED WORKS

Some efforts have been reported in the literature on the development of the not-intrusive monitoring drowsiness systems based on the vision. Malla et al. develop a light-insensitive system. They used the Haar algorithm to detect objects and face classifier implemented by in OpenCV libraries. Eye regions are derived from the facial region with anthropometric factors. Then, they detect the eyelid to measure the level of eye closure. Vitabile et al. implement a system to detect symptoms of driver drowsiness based on an infrared camera. By exploiting the phenomenon of bright pupils, an algorithm for detecting and tracking the driver's eyes has been developed. When drowsiness is detected, the system warns the driver with an alarm message. Bhowmick et Kumar use the Otsu thresholding to extract face region. The localization of the eye is done by locating facial landmarks such as eyebrow and possible face centre. Morphological operation and K- means is used for accurate eye segmentation. Then a set of shape features are calculated and trained using non-linear SVM to get the status of the eye. Flowchart of the proposed system. Hong et al. define a system for detecting the eye states in real time to identify the driver drowsiness state. The face region is detected based on the optimized Jones and Viola method . The eye area is obtained by an horizontal projection. Finally, a new complexity function with a dynamic threshold to identify the eye state. Tianet Qin build a system that checks the driver eye states. Their system uses the Cb and Cr components of the YCbCr color space. This system locates the face with a vertical projection function, and the eyes with a horizontal projection function. Once the eyes are located the system calculates the eyes states using a function of complexity.

CHAPTER 3

THE PROPOSED SYSTEM

In this section, we discuss our presented system which detects driver drowsiness. The overall flowchart of our system is shown in Figure 3.1

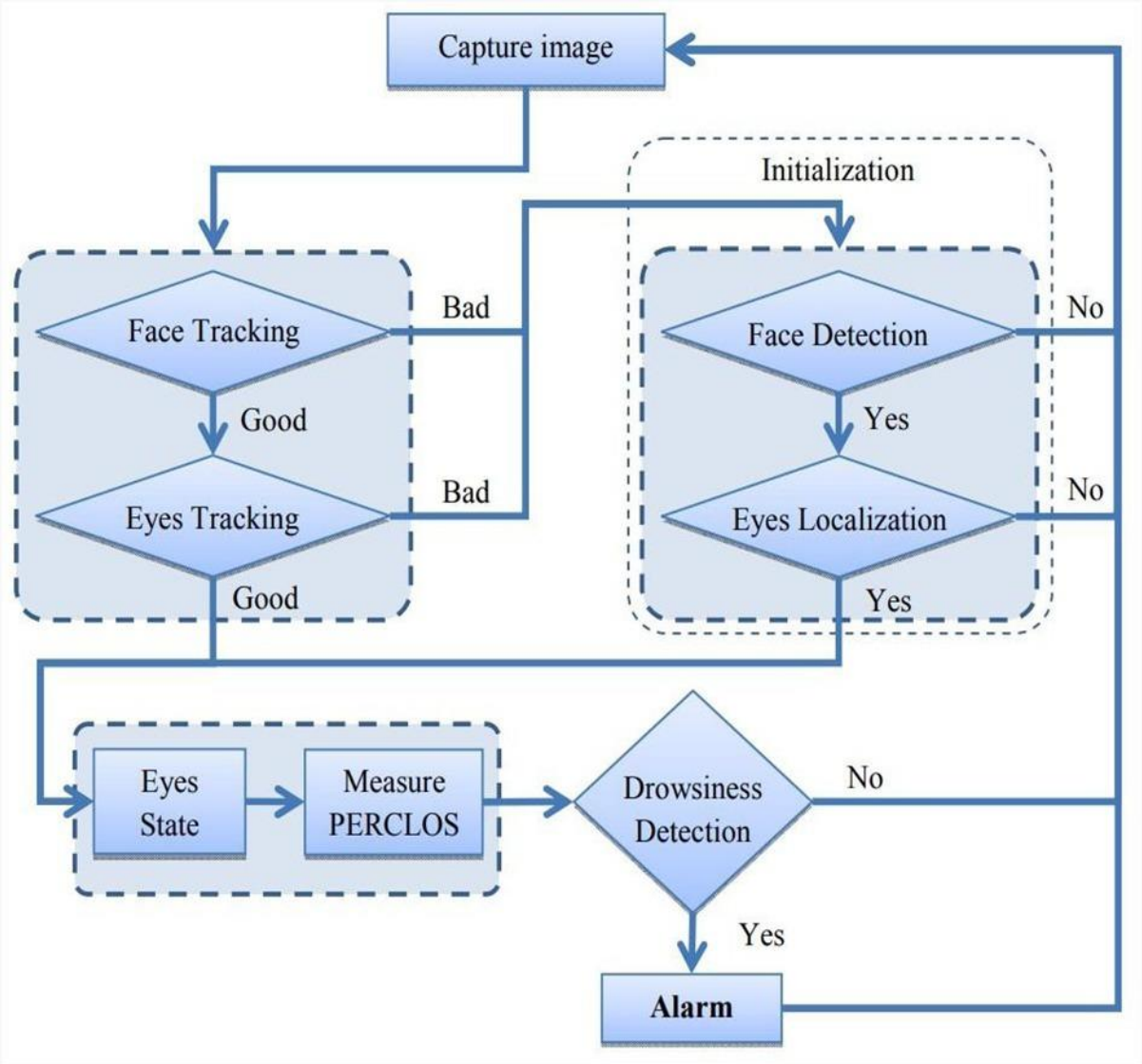

Figure 3.1

3.1 FACE DETECTION

The symmetry is one of the most important facial features. We modelled the symmetry in a digital image by a one-dimensional signal (accumulator vector) with a size equal the width of the image, which gives us the value corresponding to the position of the vertical axis of symmetry of objects in the image. The traditional principle to calculate the signal of symmetry is for each two white pixels which are on the same line we increment the value in the medium between these two pixels in the accumulator vector. (The algorithm is applied on an edge image, we called a white pixel: the pixel with value 1). We introduce improvements on the calculation algorithm of symmetry into an image to adapt it to the detection of face, by applying a set of rules to provide a better calculation of symmetry of the face. Instead of computing the symmetry between two white pixels in the image, it is calculated between two windows (Z1 and Z2) (Figure 3.2).

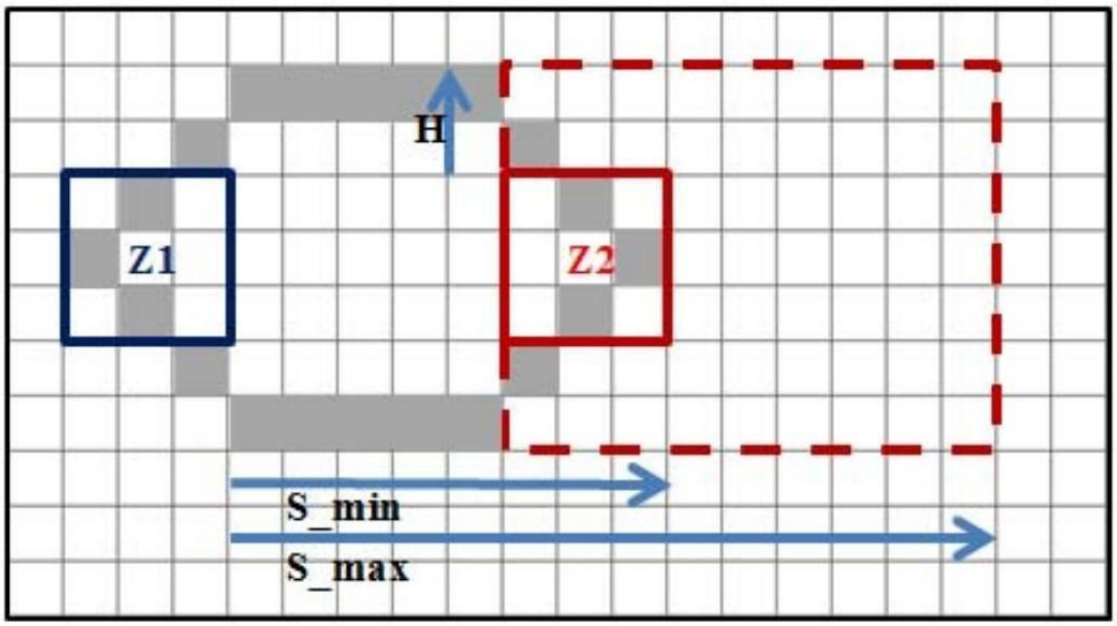

Figure 3.2

Figure 3.2- The new method to improve the calculation of the symmetry in the image. For each window Z1, we sweep the window Z2 in the area determined by the parameters S_min, S_max, and H. We increment the signal of symmetry between these two windows if the sum of white pixels is located between two thresholds S1 (maximum) and S2 (mini- mum). Then we extract the vertical region of the image contours (Region of Interest ROI) corresponding to the maximum index of the obtained signal of symmetry. Next, we take a rectangle with an estimated size of face (Because the camera is fixed and the driver moves in a limited zone so we can estimate the size of the face using the camera focal length after the step of camera calibration) and we scan the ROI by searching the region that contains the maximum energy corresponding to the face. We propose a checking on two axes: the position variance of the face detected according to time; i.e, in several successive images, it is necessary that the variance of the positions of the detected face is limited; because the speed of movement of the face is limited of some pixels from a frame to another frame which follows.

3.2 EYES LOCALIZATION

Since the eyes are always in a defined area in the face (facial anthropometric properties), we limit our research in the area between the forehead and the mouth (Eye Region of Interest ‘eROI’). We benefit from the symmetrical characteristic of the eyes to detect them in the face. First, we sweep vertically the eROI by a rectangular mask with an estimated height of height of the eye and a width equal to the width of the face, and we calculate the symmetry. The eye area corresponds to the position which has a high measurement of symmetry. Then, in this obtained region, we calculate the symmetry again in both left and right sides. The highest value corresponds to the center of the eye.

3.3 TRACKING

The tracking is done by Template Matching using the SAD Algorithm (Sum of Absolute Differences).

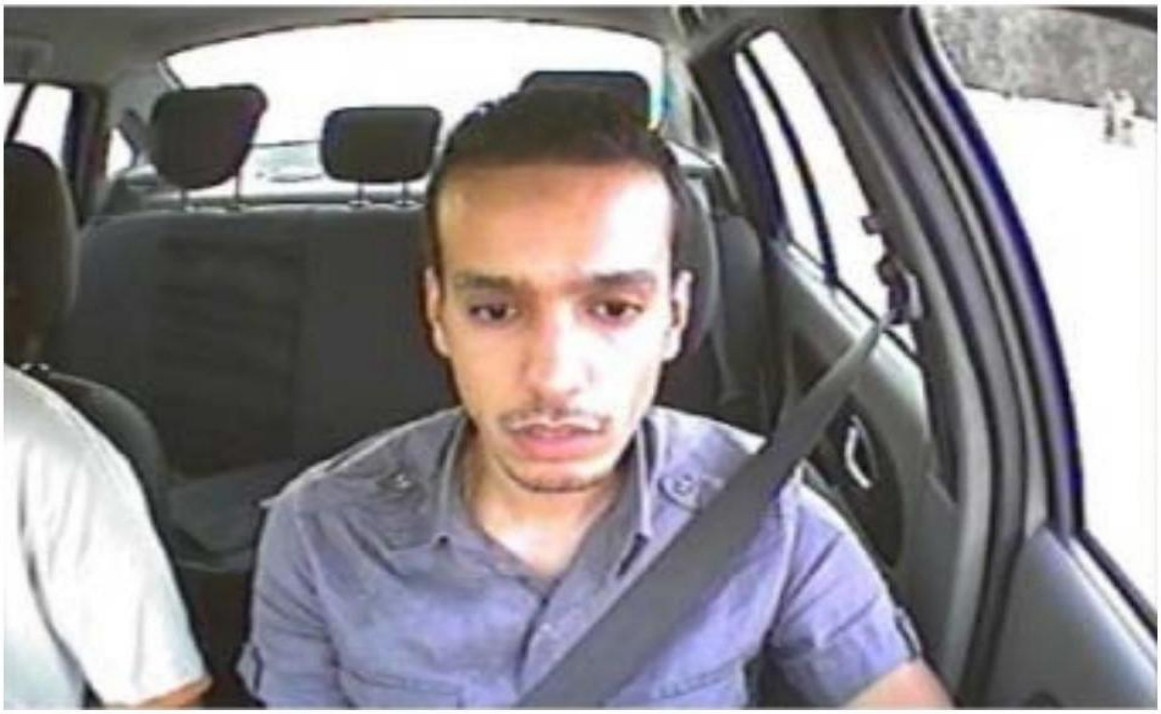

Figure 3.3

Figure 3.4

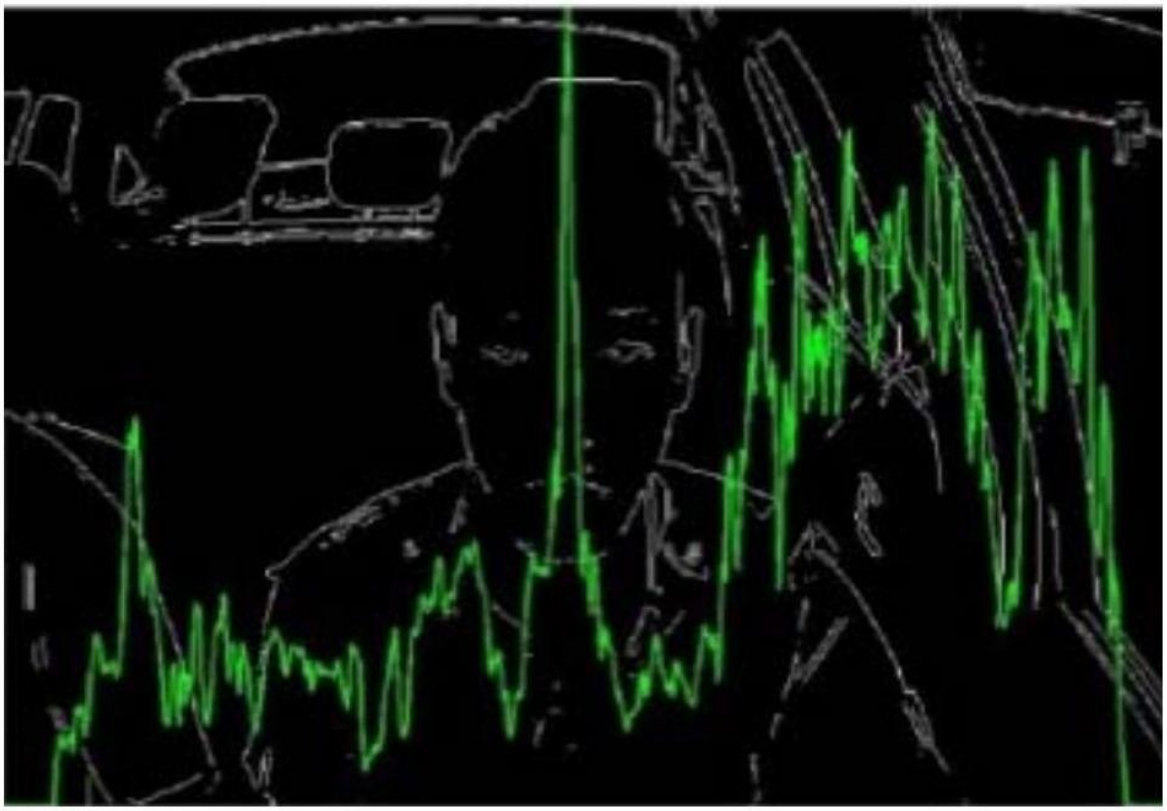

Figure 3.5

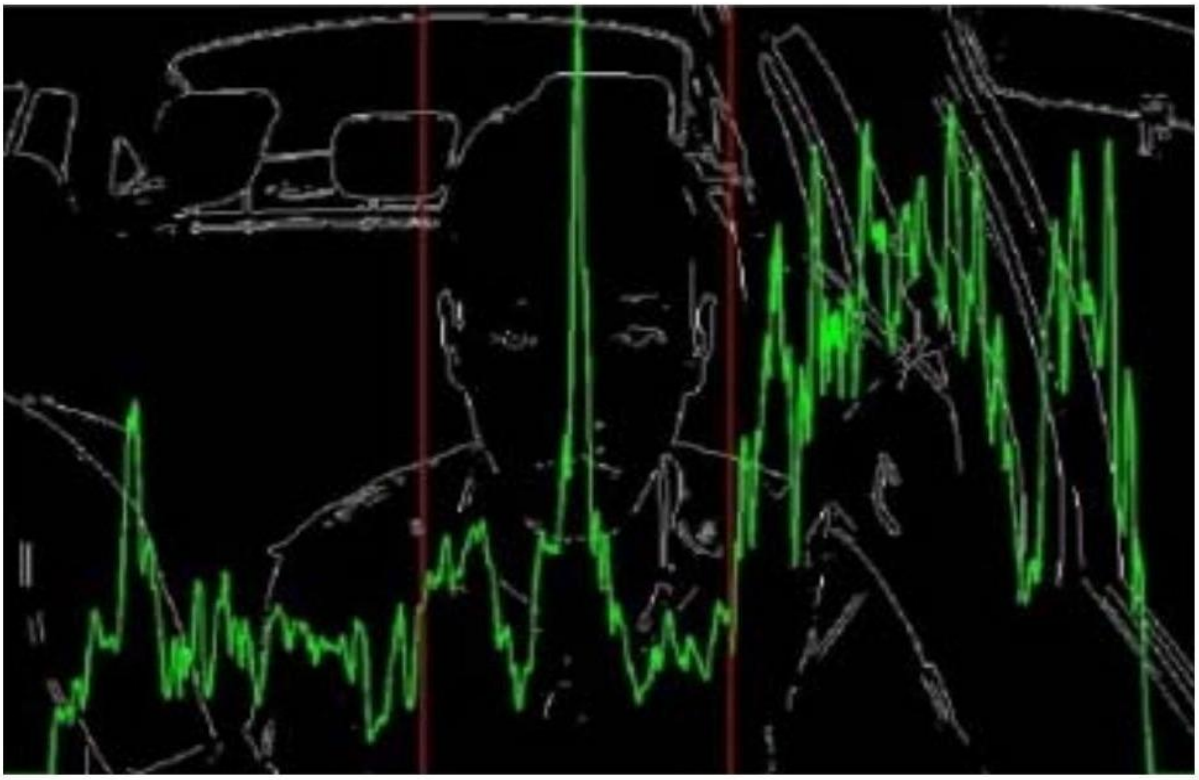

Figure 3.6

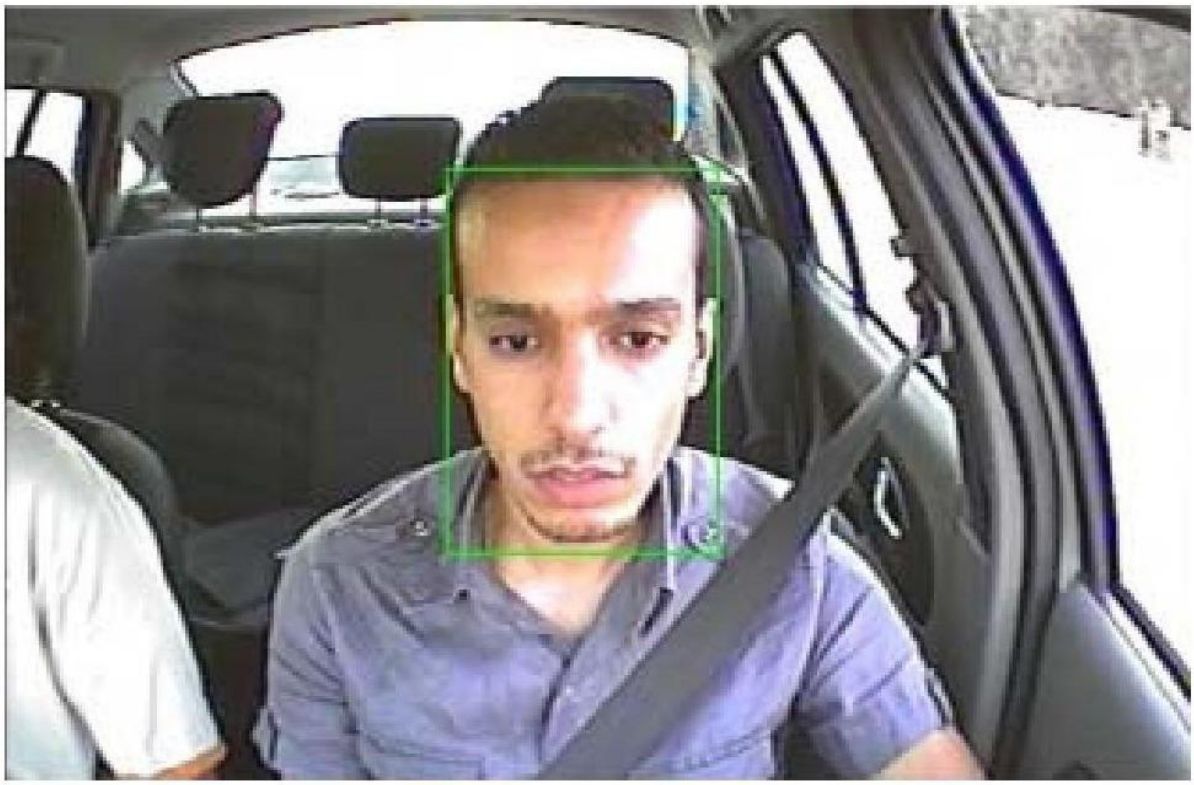

Figure 3.7

Face detection using symmetry. (a) Original image, (b) Edge detection, (c) Symmetry signal, (d) Localization of the maximum of symmetry, (e) Region of interest ROI (f) Result.

$$SAD(x,y) = \sum_{j=1}^{N}\sum_{i=1}^{M} | I(x+i,y+j) - M(i,j)$$

We proposed to make a regular update of the reference model M to adjust it every time when light conditions changes while driving, by making a tracking test:

$$Tracking \begin{cases} good & if\ SAD \leq Th \\ bad & if\ SAD > Th \end{cases}$$

3.4 EYES STATE

The determination of the eye state is to classify the eye into two states: open or closed. We use the Hough transform for circles [10] (HTC) on the image of the eye to detect the iris. For that, we apply the HTC to the edge image of the eye to detect the circles with defined rays, and we take at the end the circle which has the highest value in the accumulator of Hough for all the rays. Then, we apply the logical 'AND' logic between edges image and complete circle obtained by the HTC by measuring the intersection level between them "S".

Finally, the eye state “” is defined by test- ing the value “S” by a threshold:

$$\text{State}_{\text{eye}} = \begin{cases} \text{Open} & \text{if} \quad S \geq \text{Th} \\ \text{Closed} & \text{if} \quad S < Th \end{cases}$$

3.5 DRIVER STATES

We determine the driver state by measuring PERCLOS. If the driver closed his eyes in at least 5 successive frames several times over a period of up to 5 seconds.

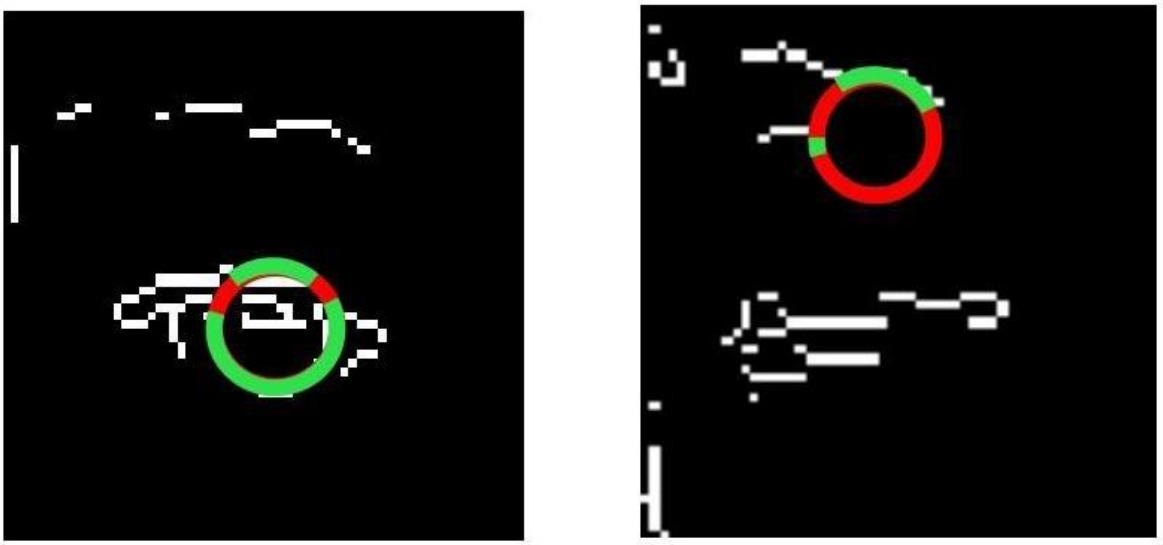

Figure 3.8

3.6 HOW THE ALGORITHM WORKS

Our system starts with the initialization phase, which is face and eyes detection to extract both face and eyes regions and take them as templates to track them in the following frames. For each tracking we test if that tracking is good or bad? If the tracking is bad we return to the initialization step, else we pass to the following steps which are: eyes states identification and driver state.

CHAPTER 4

SOFTWARE REQUIREMENTS SPECIFICATION

4.1 Python

- Python 3

4.2 Libraries

- **OpenCV** – pip install opencv-python (face and eye detection).
- **TensorFlow** – pip install tensorflow (keras uses TensorFlow as backend).
- **Keras** – pip install keras (to build our classification model).
- **Pygame** – pip install pygame (to play alarm sound).

4.3 Operating System

- Windows or Ubuntu

HARDWARE REQUIREMENTS SPECIFICATION

I. Laptop with basic hardware.

II. Webcam

CHAPTER 5

Modelling diagram (project Work Flow)

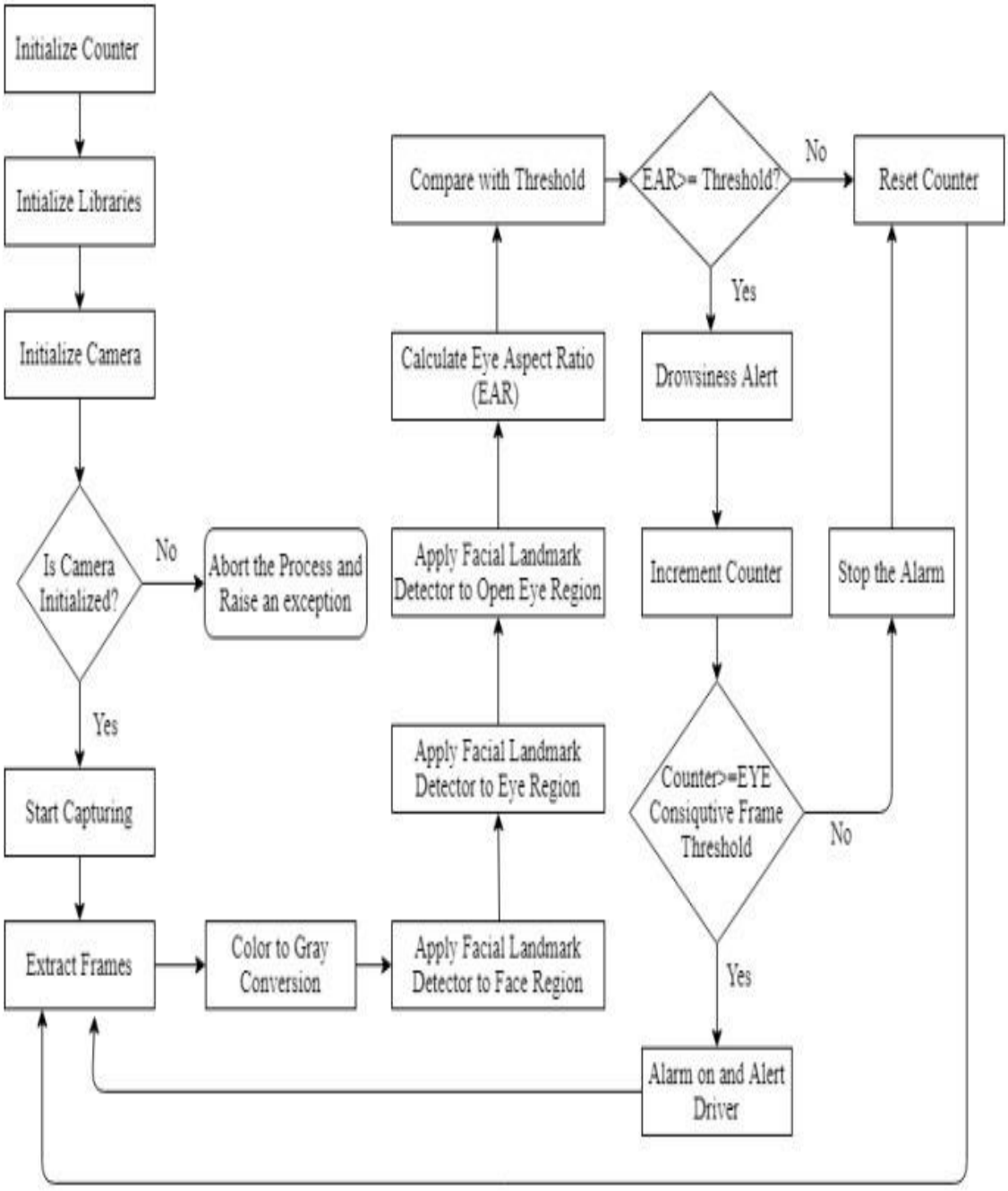

Figure 5.1

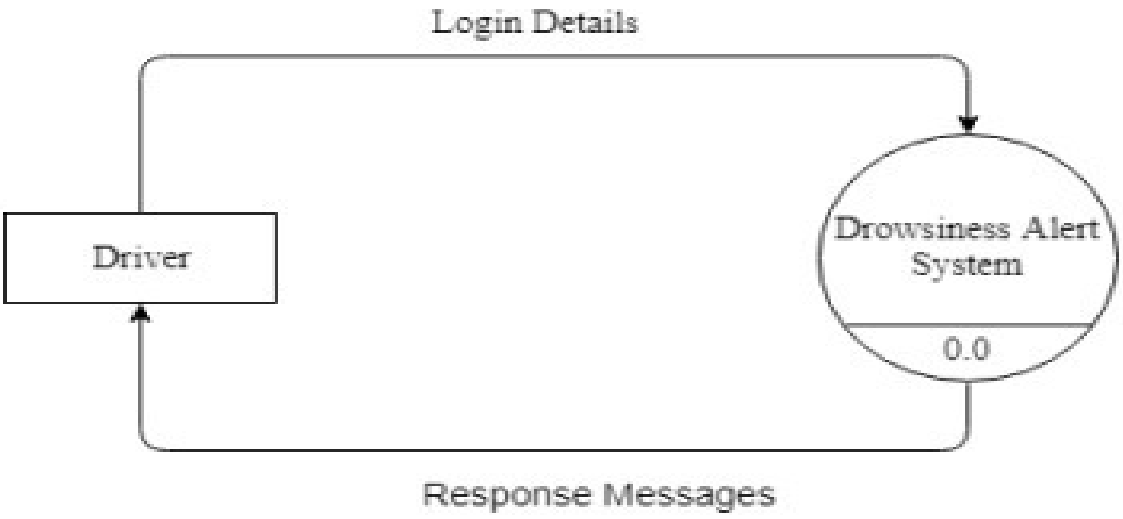

Figure 5.2 DFD level 0

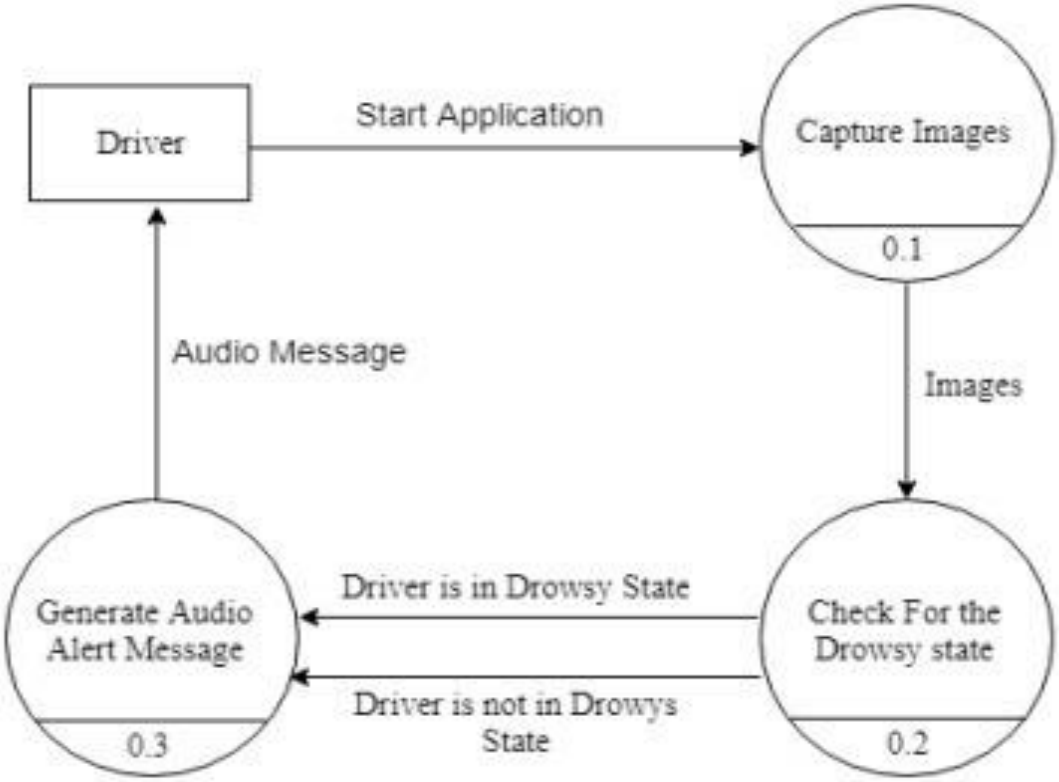

Figure 5.3 DFD level 1

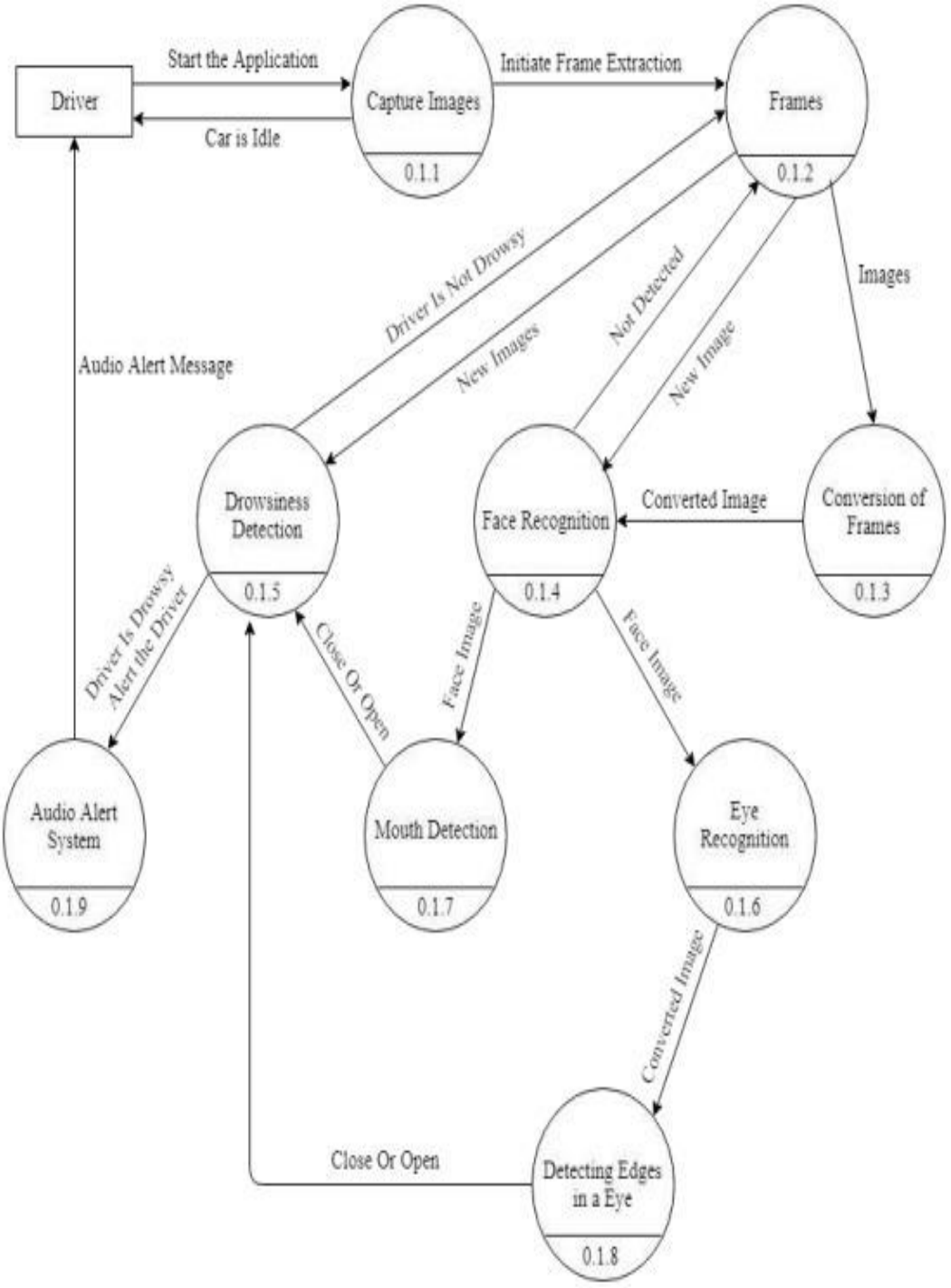

Figure 5.4 DFD level 2

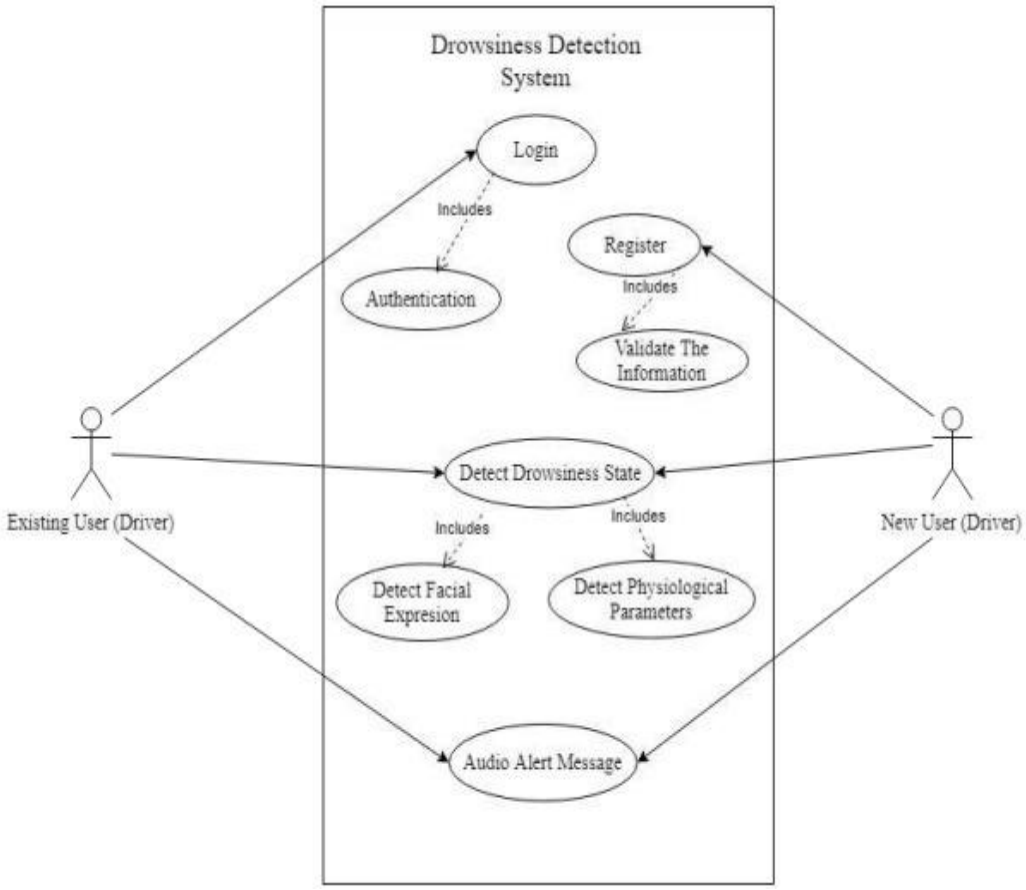

Figure 5.5 use-case diagram

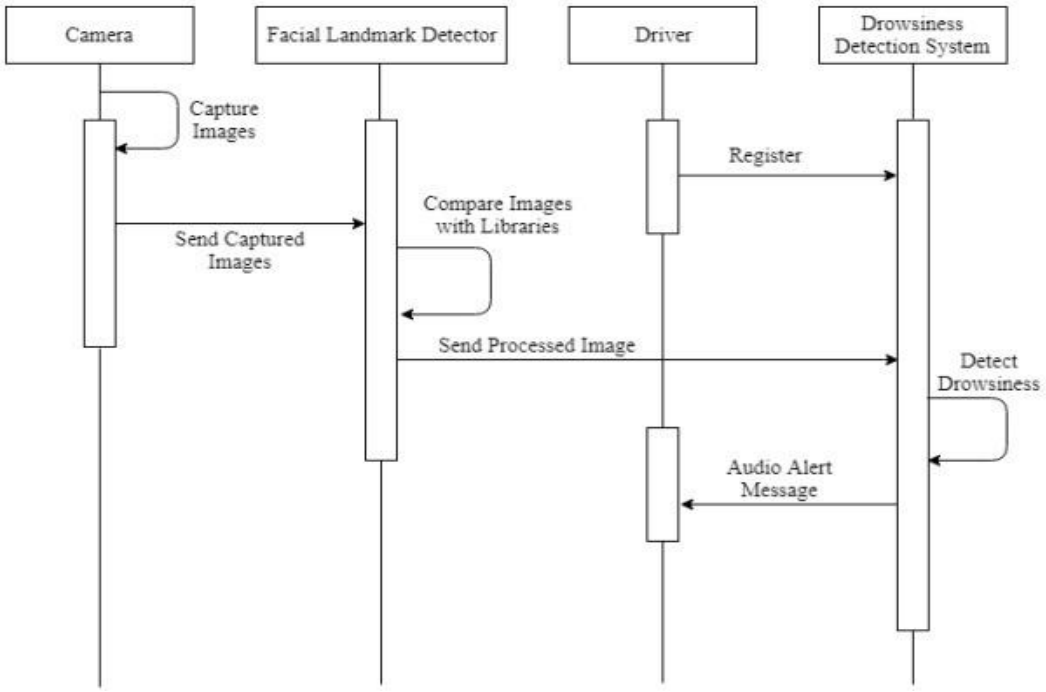

Figure 5.6 Sequence Diagram

CHAPTER 6

RESULTS AND DISCUSSIONS

To validate our system, we test on several drivers in the car with real driving conditions. We use an IR camera with infrared lighting system operates automatically under the conditions of reduced luminosity and night even in total darkness. The results of the eye states are illustrated in Table1, where the percentage error is the number of frames that have a false state of eye divided by the total number of frames multiplied by 100.

TABLE 1 – RESULTS OBTAINED FROM THE SYSTEM

Driver	frames Number	False ——— open	Eyes sates ——— Closed	false rate
D1/day	420	17	0	4%
D2/day	430	15	0	3.5%
D3/day	245	7	1	3.2%
D1/night	200	3	1	2%
D1/night	200	1	0	0.5%
D1/night	200	6	3	4.5%

According to the obtained results, our system can determine the eye states with a high rate of correct decision.

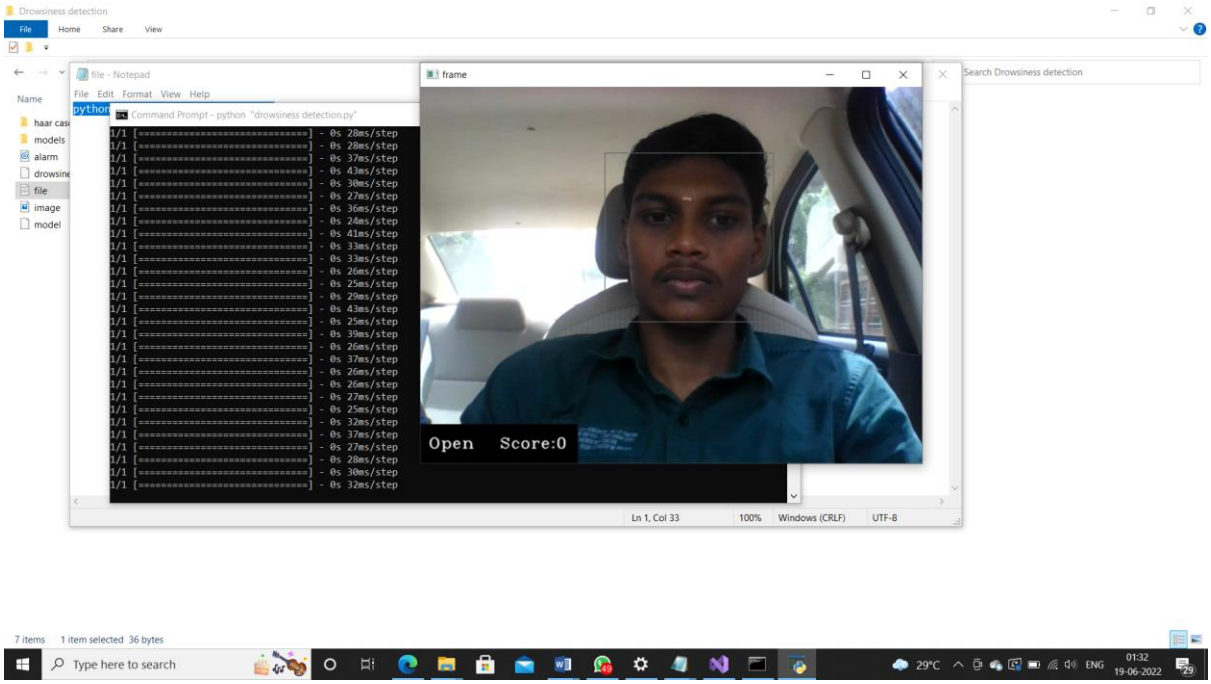

Figure 6.1 Eyes Open

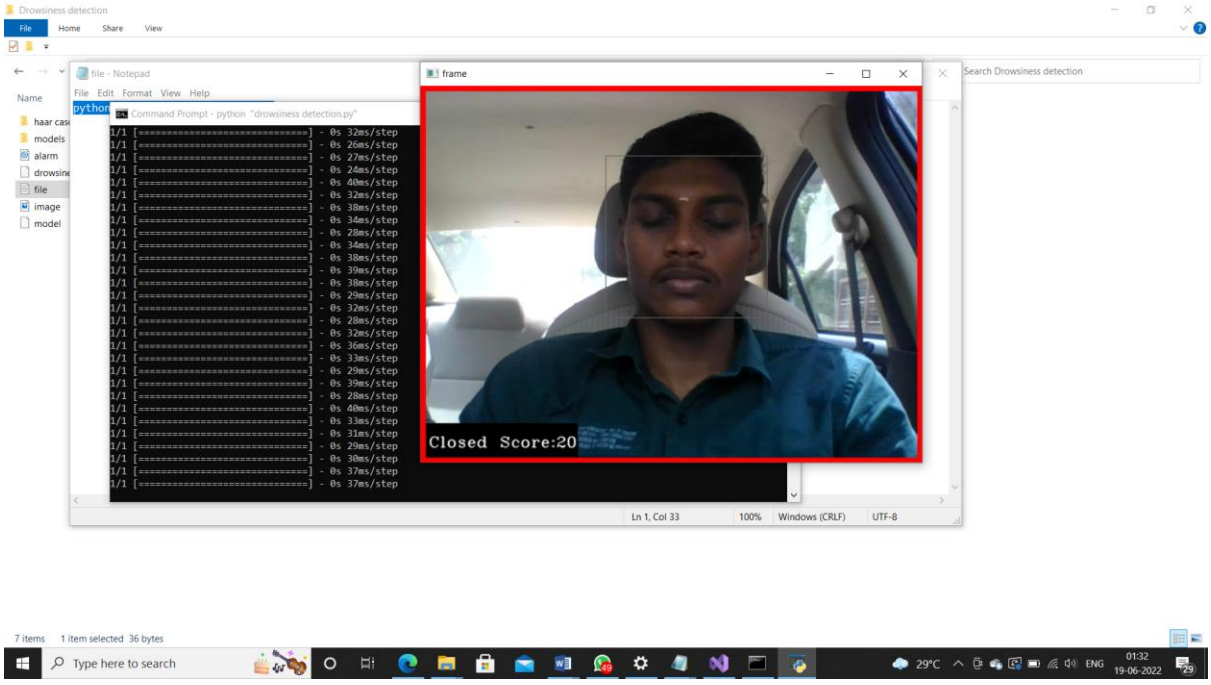

Figure 6.2 Eyes Close

CHAPTER 7

CONCLUSION AND PERSPECTIVES

In this paper, we presented the conception and implementation of a system for detecting driver drowsiness based on vision that aims to warn the driver if he is in drowsy state. This system is able to determine the driver state under real day and night conditions using IR camera. Face and eyes detection are implemented based on symmetry. Hough Transform for Circles is used for the decision of the eyes states. The results are satisfactory with an opportunity for improvement in face detection using other techniques concerning the calculation of symmetry. Moreover, we will implement our algorithm on a DSP (Digital Signal Processor) to create an autonomous system working in real time.

Limitations:

Use of spectacles: In case the user uses spectacle then it is difficult to detect the state of the eye. As it hugely depends on light hence reflection of spectacles may give the output for a closed eye as opened eye. Hence for this purpose the closeness of eye to the camera is required to avoid light.

Multiple face problem: If multiple face arises in the window then the camera may detect more number of faces undesired output may appear. Because of different condition of different faces. So, we need to make sure that only the driver face come within the range of the camera. Also, the speed of detection reduces because of operation on multiple faces.

CHAPTER 8

REFERENCES

[1] A. Malla, P. Davidson, P. Bones, R. Green and R. Jones, "Automated Video-based Measurement of Eye Closure for Detecting Behavioural Microsleep", in 32nd Annual International Conference of the IEEE, Buenos Aires, Argentina, 2010.

[2] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001.

[3] R. Lienhart and J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection", in Proceedings of the IEEE International Conference on Image Processing, 2002.

[4]OpenCV. Open Source Computer Vision Library Reference Manual, 2001.

[5]S.Vitabile,A.PaolaandF.Sorbello,"BrightPupil Detection in an Embedded, Real-time Drowsiness Monitoring System", in 24th IEEE International Conference on Advanced Information Networking and Applications, 2010.

[6] B. Bhowmick and C. Kumar, "Detection and Classification of Eye State in IR Camera for Driver Drowsiness Identification", in Proceeding of the IEEE International Conference on Signal and Image Processing Applications, 2009.

[7] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms", IEEE Transactions on Systems ,Man and Cybernatics , pp. 62-66, 1979.

[8] T. Hong, H. Qin and Q. Sun, "An Improved Real Time Eye State Identification System in Driver Drowsiness Detection", in proceeding of the IEEE International Conference on Control and Automation, Guangzhou, CHINA, 2007.

[9] Z. Tian et H. Qin, "Real-time Driver's Eye State Detection", in Proceedings of the IEEE International Conference on Vehicular Electronics and Safety, October 2005.

[10] M. S. Nixon and A. S. Aguado, Feature Extraction and Image Processing, 2nd ed., Jordan Hill, Oxford OX2 8DP, UK, 2008

APPENDIX

1. CODING

Importing our required Python packages.

drowsiness detection.py

```
import cv2
import os
from keras.models import load_model
import numpy as np
from pygame import mixer
import time


mixer.init()
sound = mixer.Sound('alarm.wav')

face = cv2.CascadeClassifier('haar cascade
files\haarcascade_frontalface_alt.xml')
leye = cv2.CascadeClassifier('haar cascade
files\haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('haar cascade
files\haarcascade_righteye_2splits.xml')



lbl=['Close','Open']

model = load_model('models/cnncat2.h5')
path = os.getcwd()
cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_COMPLEX_SMALL
count=0
score=0
thicc=2
rpred=[99]
lpred=[99]
```

```
while(True):
    ret, frame = cap.read()
    height,width = frame.shape[:2]

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

    faces =
face.detectMultiScale(gray,minNeighbors=5,scaleFactor=1.1,minSize=(25,25))
    left_eye = leye.detectMultiScale(gray)
    right_eye =  reye.detectMultiScale(gray)

    cv2.rectangle(frame, (0,height-50) , (200,height) , (0,0,0) ,
thickness=cv2.FILLED )

    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y) , (x+w,y+h) , (100,100,100) , 1 )

    for (x,y,w,h) in right_eye:
        r_eye=frame[y:y+h,x:x+w]
        count=count+1
        r_eye = cv2.cvtColor(r_eye,cv2.COLOR_BGR2GRAY)
        r_eye = cv2.resize(r_eye,(24,24))
        r_eye= r_eye/255
        r_eye=  r_eye.reshape(24,24,-1)
        r_eye = np.expand_dims(r_eye,axis=0)
        rpred = np.argmax(model.predict(r_eye), axis=-1)
        if(rpred[0]==1):
            lbl='Open'
        if(rpred[0]==0):
            lbl='Closed'
        break

    for (x,y,w,h) in left_eye:
        l_eye=frame[y:y+h,x:x+w]
        count=count+1
        l_eye = cv2.cvtColor(l_eye,cv2.COLOR_BGR2GRAY)
        l_eye = cv2.resize(l_eye,(24,24))
        l_eye= l_eye/255
        l_eye=l_eye.reshape(24,24,-1)
        l_eye = np.expand_dims(l_eye,axis=0)
        lpred = np.argmax(model.predict(l_eye ), axis=-1)
        if(lpred[0]==1):
            lbl='Open'
```

```
        if(lpred[0]==0):
            lbl='Closed'
        break

    if(rpred[0]==0 and lpred[0]==0):
        score=score+1
        cv2.putText(frame,"Closed",(10,height-20), font,
1,(255,255,255),1,cv2.LINE_AA)
    # if(rpred[0]==1 or lpred[0]==1):
    else:
        score=score-1
        cv2.putText(frame,"Open",(10,height-20), font,
1,(255,255,255),1,cv2.LINE_AA)


    if(score<0):
        score=0
    cv2.putText(frame,'Score:'+str(score),(100,height-20), font,
1,(255,255,255),1,cv2.LINE_AA)
    if(score>15):
        #person is feeling sleepy so we beep the alarm
        cv2.imwrite(os.path.join(path,'image.jpg'),frame)
        try:
            sound.play()

        except:  # isplaying = False
            pass
        if(thicc<16):
            thicc= thicc+2
        else:
            thicc=thicc-2
            if(thicc<2):
                thicc=2
        cv2.rectangle(frame,(0,0),(width,height),(0,0,255),thicc)
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```