A Project Report on

Drowsiness Detection System for Intelligent Vehicle Using Image Processing

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Information Technology

by

Akshay Kautkar(15204011)
Praneta Kasbe(15204051)
Shraddha Kedar(15204007)
Arvind Maurya(15204023)

Under the Guidance of

Prof. Ganesh Gourshete
Prof. Poonam Dhawale

Department of Information Technology
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI

Academic Year 2017-2018

Approval Sheet

This Project Report entitled ***"Drowsiness Detection System For Intelligent Vehicle Using Image Processing"*** Submitted by ***"Akshay Kautkar"(15204011), "Praneta Kasbe"(15204051), "Shraddha Kedar"(15204007), "Arvind Maurya"(15204023)***is approved for the partial fulfillment of the requirenment for the award of the degree of ***Bachelor of Engineering*** in ***Information Technology*** from ***University of Mumbai***.

(Prof. Poonam Dhawale)
Co-Guide

(Prof. Ganesh Gourshete)
Guide

Prof. Kiran Deshpande
Head Department of Information Technology

Place:A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled ***"Drowsiness Detection System For Intelligent Vehicle Using Image Processing"*** submitted by ***"Akshay Kautkar" (15204011),"Praneta Kasbe" (15204051),"Shraddha Kedar" (15204007),"Arvind Mourya" (15204023)*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Information Technology.***,to the University of Mumbai,is a bonafide work carried out during academic year 2018-2019.

(Prof. Poonam Dhawale)
Co-Guide

(Prof. Ganesh Gourshete)
Guide

Prof. Kiran Deshpande
Head Department of Information Technology

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

(Signature)

_____

(Akshay Kautkar 15204011)
(Praneta Kasbe 15204051)
(Shraddha Kedar 15204011)
(Arvind Mourya 15204023)

Date:

Acknowledgement

We have great pleasure in presenting the report on **Drowsiness Detection System for Intelligent vehicles Using Image processing** We take this opportunity to express our sincere thanks towards our guide **Prof.Ganesh Gourshete** & Co-Guide **Prof.Poonam Dhawale** Department of IT, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his/her constant encouragement, support and guidance through the development of project.

We thank **Prof. Kiran B. Deshpande** Head of Department,IT, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Prof. Vishal S. Badgujar** BE project co-ordinator, Department of IT, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Akshay Kautkar
15204011

Praneta Kasbe
15204051

Shraddha Kedar
15204007

ArvindKumar Mourya
15204023

Abstract

A Drowsy Driver Detection System is going to be developed, using a non-intrusive machine vision based concepts. The system will use a camera that points directly towards the driver's face and monitors the driver's eyes in order to detect fatigue. In such a case, when fatigue is detected, a warning signal is issued to alert the driver. This describes how to find the eyes, and also how to determine if the eyes are open or closed. The system deals with using information obtained from the binary version of the image to find the edges of the face, which narrows the area of where the eyes may exist. Taking into account the knowledge that eye regions in the face present great intensity changes, the eyes are located by finding the significant intensity changes in the face. Once the eyes are located, measuring the distances between the intensity changes in the eye area determine whether the eyes are open or closed. A large distance corresponds to eye closure. If the eyes are found closed for consecutive frames, the system draws the conclusion that the driver is falling a sleep and issues a warning signal. A warning signal like an alarm will ring in the car to make the driver alert and a notification of message will be send to the owner of the car if it is in commercial use. Keywords: non-intrusive, fatigue, warning signal, alarm, consecutive frames

Contents

List of Figures

listoftables

List of Abbreviations

IDS:	Intrusion Detection System
WSN:	Wireless Sensor Network
MANET:	Mobile Ad-Hoc Network
AODV:	Ad-Hoc On-demand Distance Vector Routing
DSR:	Dynamic Source Routing Protocol
NS2:	Network Simulator 2
ACK:	Acknowledgement
AGT:	Agent
RTR:	Router

Chapter 1

Introduction

Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes[2]. The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects. The aim of this project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the drivers eyes in real-time[1]. By monitoring the eyes, it is believed that the symptoms of driver fatigue can be detected early enough to avoid a car accident. Detection of fatigue involves the observation of eye movements and blink patterns in a sequence of images of a face. Initially, we decided to go about detecting eye blink patterns using Matlab. The procedure used was the geometric manipulation of intensity levels. The algorithm used was as follows. First we input the facial image using a webcam. Preprocessing was first performed by binarizing the image. The top and sides of the face were detected to narrow down the area where the eyes exist. Using the sides of the face, the center of the face was found which will be used as a reference when computing the left and right eyes. Moving down from the top of the face, horizontal averages of the face area were calculated. Large changes in the averages were used to define the eye area. There was little change in the horizontal average when the eyes were closed which was used to detect a blink. However Matlab had some serious limitations. The processing capacities required by Matlab were very high. Also there were some problems with speed in real time processing. Matlab was capable of processing only 4-5 frames per second. On a system with a low RAM this was even 8lower. As we all know an eye blink is a matter of milliseconds. Also a drivers head movements can be pretty fast. Though the Matlab program designed by us detected an eye blink, the performance was found severely wanting. This is where OpenCV came in. OpenCV is an open source computer vision library. We have used the Haartraining applications in OpenCV to detect the face and eyes. This creates a classifier given a set of positive and negative samples. The steps were as follows:-

Gather a data set of face and eye. These should be stored in one or more directories indexed by a text file. A lot of high quality data is required for the classifier to work well.

The utility application createsamples() is used to build a vector output file. Using this file we can repeat the training procedure. It extracts the positive samples from images before normalizing and resizing to specified width and height.

The Viola Jones cascade decides whether or not the object in an image is similar to

the training set. Any image that doesnt contain the object of interest can be turned into negative sample. So in order to learn any object it is required to take a sample of negative background image. All these negative images are put in one file and then its indexed.

Training of the image is done using boosting. In training we learn the group of classifiers one at a time. Each classifier in the group is a weak classifier. These weak classifiers are typically composed of a single variable decision tree called stumps. In training the decision stump learns its classification decisions from its data and also learns a weight for its vote from its accuracy on the data. Between training each classifier one by one, the data points are reweighted so that more attention is paid to the data points where errors were made. This process continues until the total error over the dataset arising from the combined weighted vote of the decision trees falls below a certain threshold.

This algorithm is effective when a large number of training data are available. For our project face and eye classifiers are required. So we used the learning objects method to create our own haar classifier .xml files. Around 2000 positive and 3000 negative samples are taken. Training them is a time intensive process. Finally face.xml and haarcascade-eye.xml files are created. These xml files are directly used for object detection. It detects a sequence of objects here are face and eyes. Haar cascade-eye.xml is designed only for open eyes. So when eyes are closed the system doesnt detect anything. This is a blink. When a blink lasts for more than 5 frames, the driver is judged to be drowsy and an alarm is sounded[4].

1.1 Scope

1. The Project aims to detect drowsiness while driving a vehicle
2. Detection is the initial step to avoid unfortunate accident
3. By involvement of non-intrusive technology

You can also create section and subsection

1.2 Objective

Driver drowsiness detection is a car safety technology which helps to save the life of the driver by preventing accidents when the driver is getting drowsy.Our objective of the project is to ensure the safety system. For enhancing the safety,Once the eyes are detected as closed for consecutive frames, an alarm would ring. By implementation of this control system, number of accidents can be reduced.

1.3 Overview

During long journeys, its possible that the driver may lose attention because of drowsiness, which may be a potential reason for fatal accidents. With technologies like Driver Drowsiness Detection getting it is possible to detect drivers driving behavior that may prove fatal to the vehicle as well as the people boarding it.

Having such sleep detection system in cars embedded in vehicles could protect precious lives and property worth billion dollars. The outcome would be positive it would be suitable for fleet owners as well as individual vehicle users. In either case, the objective is identical by

sleep detection while driving. While driving, you may feel drowsy when youre under driving fatigue because of continuous driving for several hours. Its here that the driver drowsiness detection plays a significant role in preventing accidents that could otherwise cause massive loss of life and property.

Chapter 2

Literature Review

2.1 Paper 1: Driver Drowiness Detection Using Visual Information on Android Device

The number of casualties from road accidents keep arising each year. While there are many causes to road accidents, most are surprisingly caused by human errors, such as drowsiness. Therefore, this issue raises an idea for an application which will be able to track a persons eye movement while driving. The application, called Driver Drowsiness Detection, runs on an Android handheld and wearable. The purpose of this application is to alert drivers so that they can be cautioned to pull over and stop driving in a drowsy state. The application Driver Drowsiness Detection utilizes Haar-cascade Detection as well as template matching in OpenCV to detect and track the eyes using the front camera of an Android device. Testing has been conducted to ensure that the functionality, behavior, performance and user satisfaction are as expected. Even though, the input received by the application still has several restrictions, specifically in correlation to sufficient lighting and obscurity of the face and eyes area, the application has successfully detected the eye blinks at the angle of 30 to 60 degrees and distance of 20-50 cm, as well as measuring the heart rate.[1] .

2.2 Paper 2: A Fuzzy Based Method for Driver Drowsiness Detection

This paper describes a novel approach for an intel-ligent driver drowsiness detection system using visual behavior of the driver. The estimation of drivers vigilance is successfully made by combining facial and eye symptoms using fuzzy logic controller. Experimental result using fuzzy-logic simulation in Matlab show the performance of the developed approach in term of robustness and reliability.[2]

2.3 Paper 3:Yawning detection by the analysis of variational descrip- tor for monitoring driver drowsiness

The paper describes that road safety is a problem which was approached by several countries following a big raise of the number of accidents. The drowsiness represents one among the causes of the road accidents. The accidents related to the drowsiness often occur on the highways, but also on the main roads, even inside the localities. Today, it is possible to detect the state of tiredness of the driver with the development of the technology of the computer vision. The results of research in physiology show that the first level of lack of vigilance appears by an increase in the frequency of the yawn.[3]

Chapter 3

3.1 Problem Definition

Drowsy driving is a serious problem that leads to thousands of accidents each year.Motor vehicle collisions lead to significant death and disability as wells as significant financial cost to both security and individual due to the driver impairments. Drowsiness is one of the factors for collisions.Monitoring the drivers state of drowsiness and vigilance and providing feedback on their condition so that they can take appropriate action is one crucial step in a series of preventive measures necessary to address this problem.

3.2 Proposed System

A Drowsy Driver Detection System will be developed, using a non-intrusive machine vision based concepts. The aim of this project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the drivers eyes in real-time. The system uses a small monochrome security camera that points directly towards the drivers face and monitors the drivers eyes in order to detect fatigue. By monitoring the eyes, it is believed that the symptoms of driver fatigue can be detected early enough to avoid a car accident. Detection of fatigue involves a sequence of images of a face, and the observation of eye movements and blink patterns. This project is focused on the localization of the eyes, which involves looking at the entire image of the face, and determining the position of the eyes by a self developed image-processing algorithm. Once the position of the eyes is located, the system is designed to determine whether the eyes are opened or closed, and detect fatigue.

3.3 Proposed Architecture

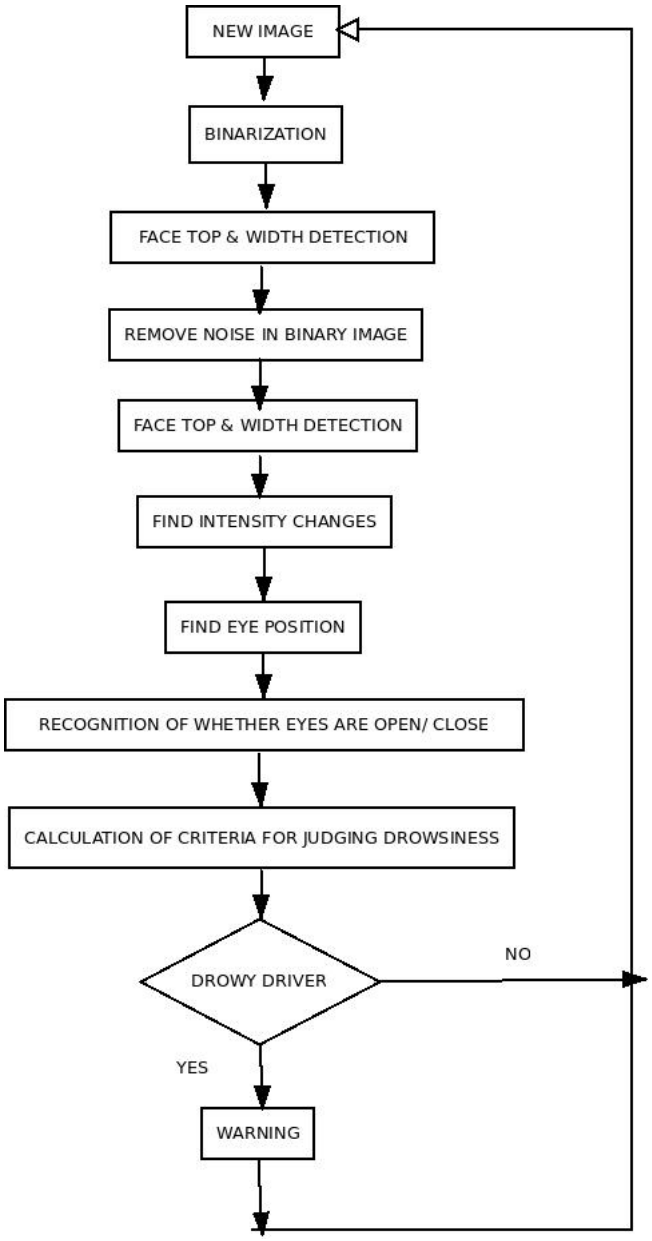


Figure 3.1: Flow Chart Of Drowsiness Detection System

3.4 Implementation Idea

There is a substantial amount of evidence that suggests that driver drowsiness plays a significant role in road accidents. Alarming recent statistics are raising the interest in equipping vehicles with driver drowsiness detection systems. This dissertation describes the design and implementation of a driver drowsiness detection system that is based on the analysis of visual input consisting of the driver's face and eyes. The resulting system combines off-the-shelf software components for face detection and eye state classification in a novel way. It follows a behavioral methodology by performing a non-invasive monitoring of external cues describing a driver's level of drowsiness. We look at this complex problem from a systems engineering point of view in order to go from a proof-of-concept prototype to a stable software framework. Our system utilizes two detection and analysis methods: (i) face detection with eye region extrapolation and (ii) eye state classification. Additionally, we use confirmation processes based on nod detection - to make the system more robust and resilient while not sacrificing speed significantly. The system was designed to be dynamic and adaptable to conform to the current conditions and hardware capabilities.

3.5 Haar Cascade Algorithm

OpenCV comes with a trainer as well as detector. If you want to train your own classifier for any object like car, planes etc. you can use OpenCV to create one. Its full details are given here: Cascade Classifier Training. Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/ folder. Let's create face and eye detector with OpenCV. First we need to load the required XML classifiers. Then load our input image (or video) in grayscale mode.

Now we find the faces in the image. If faces are found, it returns the positions of detected

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread('sachin.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

faces as Rect(x,y,w,h). Once we get these locations, we can create a ROI for the face and apply eye detection on this ROI (since eyes are always on the face !!!).

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

cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Result looks like below:

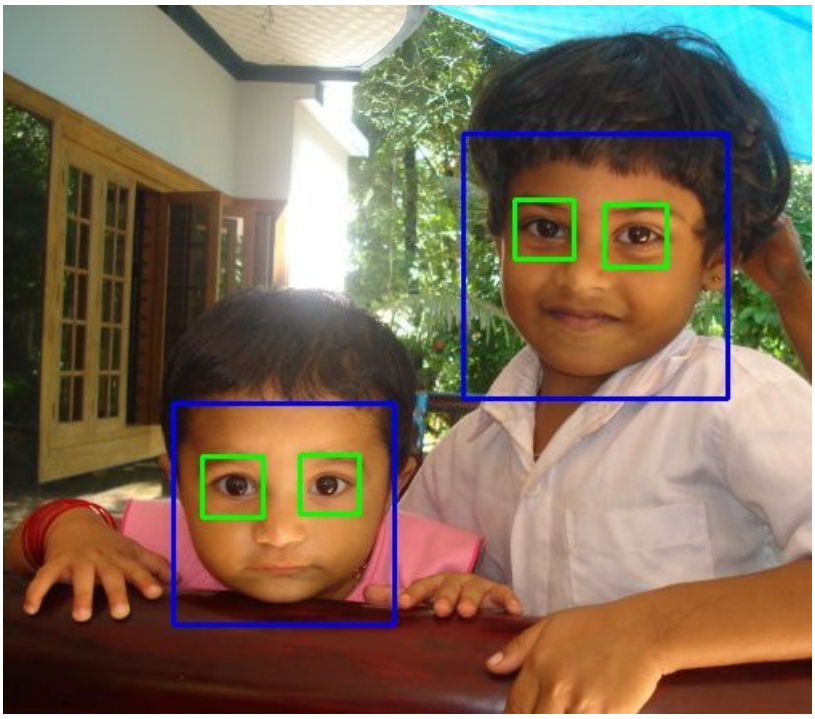
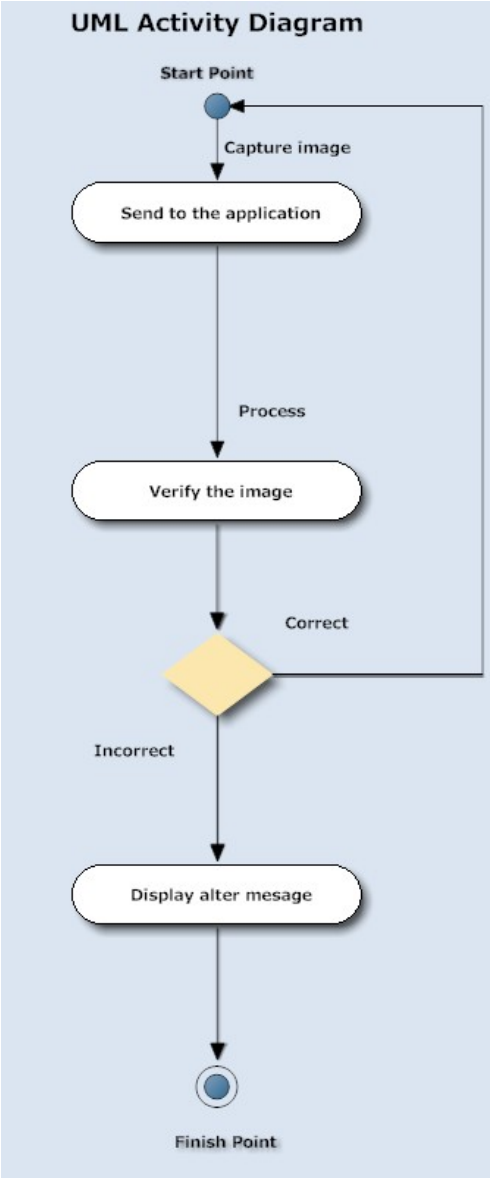


Figure 3.2: Face And Eyes Detection

Chapter 4

Design

4.1 Activity Diagram



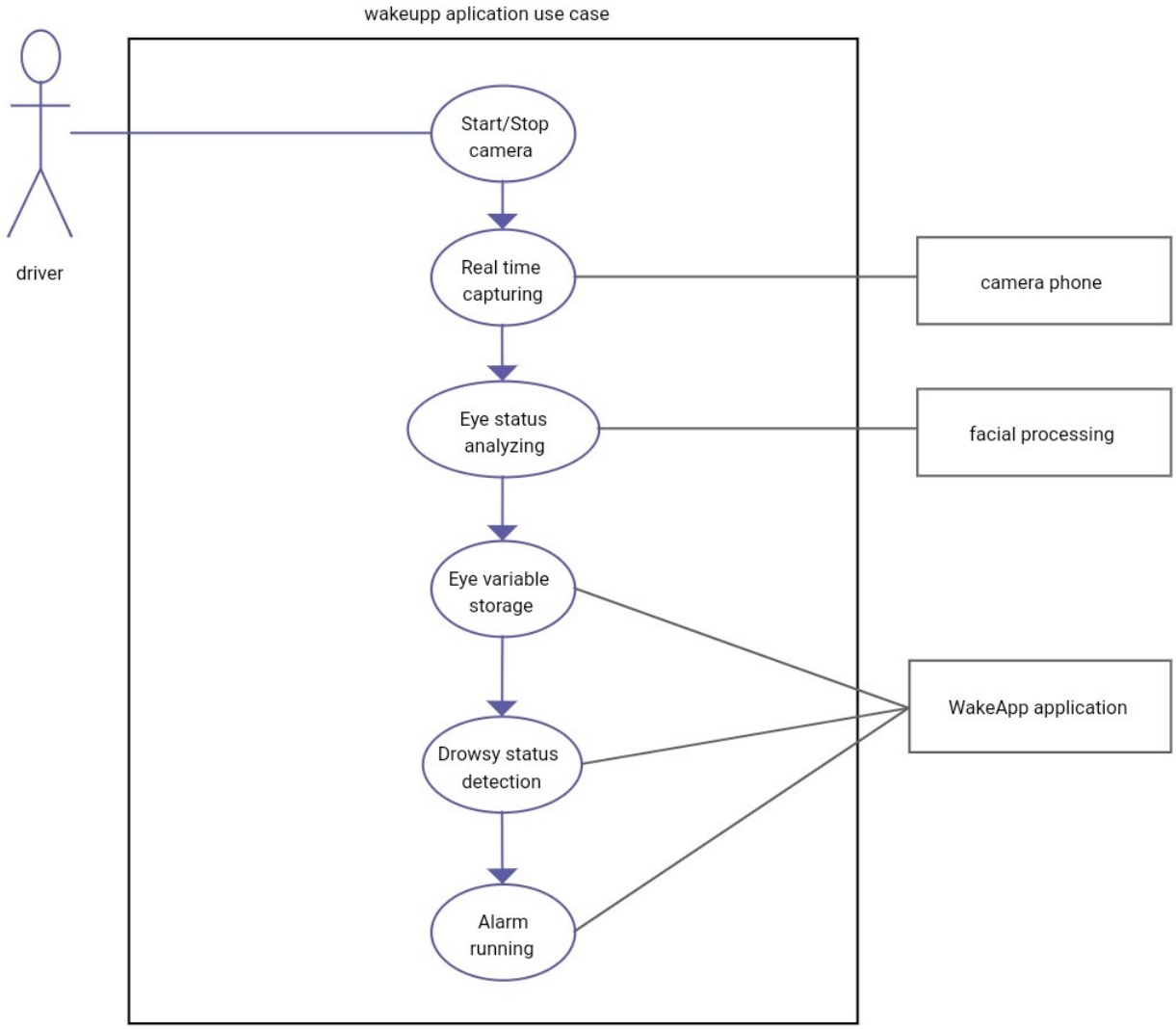


Figure 4.1: Use Case Of Wakeup Application

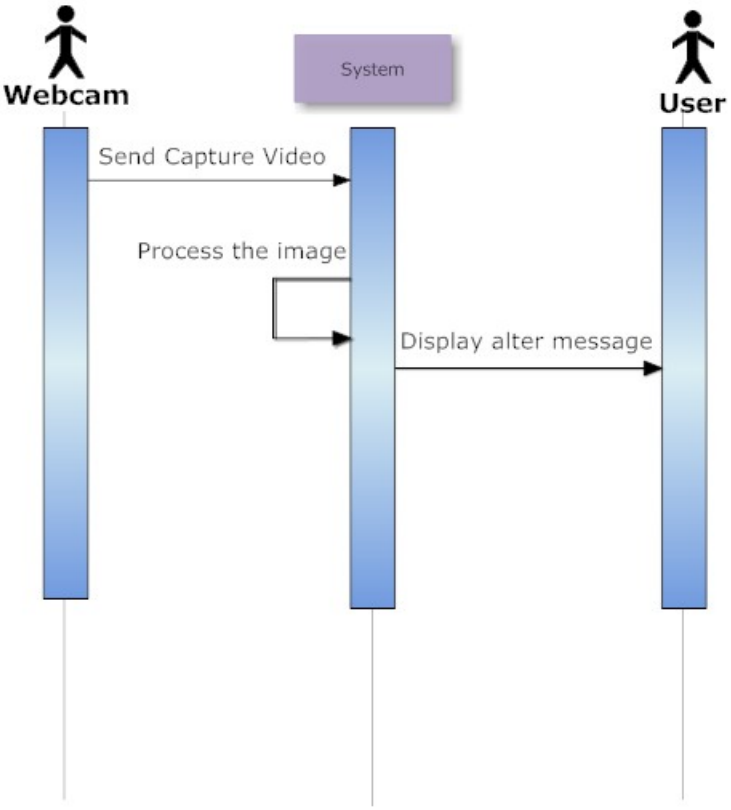


Figure 4.2: Sequence Diagram

Chapter 5

Adaptation Of Technology

5.1 What Is OpenCV?

OpenCV [OpenCV] is an open source (see http://opensource.org) computer vision library available from http://SourceForge.net/projects/opencvlibrary. OpenCV was designed for computational efficiency and having a high focus on real-time image detection. OpenCV is coded with optimized C and can take work with multicore processors. If we desire more automatic optimization using Intel architectures [Intel], you can buy Intels Integrated Performance Primitives (IPP) libraries [IPP]. These consist of low-level routines in various algorithmic areas which are optimized. OpenCV automatically uses the IPP library, at runtime if that library is installed.One of OpenCVs goals is to provide a simple-to-use computer vision infrastructure which helps people to build highly sophisticated vision applications fast. The OpenCV library, containing over 500 functions, spans many areas in vision. Because computer vision and machine learning oft en goes hand-in-hand, OpenCV also has a complete, general-purpose, Machine Learning Library (MLL). This sub library is focused on statistical pattern recognition and clustering. The MLL is very useful for the vision functions that are the basis of OpenCVs usefulness, but is general enough to be used for any machine learning problem.

5.2 What Is Computer Vision?

Computer vision is the transforming of data from a still, or video camera into either a representation or a new decision. All such transformations are performed to achieve a particular goal. A computer obtains a grid of numbers from a camera or from the disk, and thats that. Usually, there is no built in pattern recognition or automatic control of focus and aperture, no cross-associations with years of experience. For the most part, vision systems are still fairly nave.

5.3 The Origin of OpenCV

OpenCV came out of an Intel Research initiative meant to advance CPU-intensive applications. Toward this end, Intel launched various projects that included real-time ray tracing

and also 3D display walls. One of the programmers working for Intel at the time was visiting universities. He noticed that a few top university groups, like the MIT Media Lab, used to have well-developed as well as internally open computer vision infrastructurescode that was passed from one student to another and which gave each subsequent student a valuable foundation while developing his own vision application. Instead of having to reinvent the basic functions from beginning, a new student may start by adding to that which came before.

5.4 OpenCV Structure and Content

OpenCV can be broadly structured into five primary components, four of which are shown in the figure. The CV component contains mainly the basic image processing and higher-level computer vision algorithms; MLL the machine learning library includes many statistical classifiers as well as clustering tools. HighGUI component contains I/O routines with functions for storing, loading video images, while CXCore contains all the basic data structures and content.

5.5 Why OpenCV?

OpenCV was designed for image processing. Every function and data structure has been designed with an Image Processing application in mind. Meanwhile, Matlab, is quite generic. You can get almost everything in the world by means of toolboxes. It may be financial toolboxes or specialized DNA toolboxes.

5.6 Characteristics

1. Speedy

Matlab is just way too slow. Matlab itself was built upon Java. Also Java was built upon C. So when we run a Matlab program, our computer gets busy trying to interpret and compile all that complicated Matlab code. Then it is turned into Java, and finally executes the code. If we use C/C++, we dont waste all that time. We directly provide machine language code to the computer, and it gets executed. So ultimately we get more image processing, and not more interpreting. After doing some real time image processing with both Matlab and OpenCV, we usually got very low speeds, a maximum of about 4-5 frames being processed per second with Matlab. With OpenCV however, we get actual real time processing at around 30 frames being processed per second. Sure we pay the price for speed a more cryptic language to deal with, but its definitely worth it. We can do a lot more, like perform some really complex mathematics on images using C and still get away with good enough speeds for your application.

2. Efficient

Matlab uses just way too much system resources. With OpenCV, we can get away with as little as 10mb RAM for a real-time application. Although with todays computers, the RAM factor isnt a big thing to be worried about. However, our drowsiness detection system is to be used inside a car in a way that is non-intrusive and small; so a low processing requirement is vital.

Thus we can see how OpenCV is a better choice than Matlab for a real-time drowsiness detection system.

5.7 System Requirements

HARDWARE	SPECIFICATION
RAM	2 GB and ABOVE
HARD DISK	160 GB
PROCESSOR	INTEL PROCESSOR IV
WEB CAMERA	-

SOFTWARE	SPECIFICATION
OPERATING SYSTEM	WINDOWS 10
TECHNOLOGY USED	.NET
SOFTWARE	VISUAL STUDIO 2013

Chapter 6

Methodology

```
public partial class Webcam : Form
{

    Variables
    int eye_counter = 0;
    int speed_counter = 0;

    1 reference
    public Webcam()
    {
        InitializeComponent();
        _faces = new HaarCascade("haarcascade_frontalface_alt_tree.xml");
        _eyes = new HaarCascade("haarcascade_eye.xml");
        stpWatch = new Stopwatch();
        stpWatch2 = new Stopwatch();
    }



    private void Webcam_Load(object sender, EventArgs e)
    {
        capWebCam = new Capture(0);
        //capWebCam = new Capture("1.3gp");
        haar = new HaarCascade("haarcascade_frontalface_alt_tree.xml");
        haar_eye = new HaarCascade("parojosG.xml");
       // haar_eye = new HaarCascade("haarcascade_eye.xml");
        timer1.Start();

    }
```

```
private void timer1_Tick(object sender, EventArgs e)
{
    using (Image<Bgr, Byte> nextFrame = capWebCam.QueryFrame())
    {
       // label2.Visible = false;
        if (nextFrame != null)
        {
            Image<Gray, Byte> grayframe = nextFrame.Convert<Gray, Byte>();

            var faces = grayframe.DetectHaarCascade(haar, 1.4, 4,
 HAAR_DETECTION_TYPE.FIND_BIGGEST_OBJECT,new Size(nextFrame.Width / 8, nextFrame.Height / 8))[0];
            if (faces == null || faces.Length <= 0)
            {
                if (stpWatch.IsRunning)
                {
                    //check if the face has not been detected for 10 secs
                    if (stpWatch.Elapsed.Seconds >= 5)
                    {
                        Console.Beep(1000, 1000);
                        System.Media.SystemSounds.Question.Play();
                        Extra Sounds
                        //MessageBox.Show("driver is drowsy!");
                       // label2.Visible = true;
```

```
    eye_counter = 0;
if ( eye_counter >=5) // NO. of Consecutive Frames
{
    if (stpWatch2.IsRunning)
    {
        //check if the face has not been detected for 10 secs
        if (stpWatch2.Elapsed.Seconds >= 5)
        {
            Console.Beep(5000, 1000);
            System.Media.SystemSounds.Question.Play();
            //SoundPlayer simpleSound = new SoundPlayer(@"C:\voice.wav");
            //simpleSound.Play();
            Extra Sounds
            //MessageBox.Show("driver is drowsy!");

            stpWatch2.Reset();
            stpWatch2.Stop();

            SmtpClient smtpserver = new SmtpClient();
            MailMessage mail = new MailMessage();
            smtpserver.Credentials = new System.Net.NetworkCredential("togale4@gmail.com","tej@12345
            smtpserver.Port = 587;
            smtpserver.EnableSsl = true;
            smtpserver.Host = "smtp.gmail.com";
            mail = new MailMessage();
            mail.From = new MailAddress("togale4@gmail.com");
            mail.To.Add("pranetakasbe@gmail.com");
            mail.Subject = "Worning";
            mail.Body = " Details: driver drowsy detected " ;
            smtpserver.Send(mail);
```

6.1 Result Representations

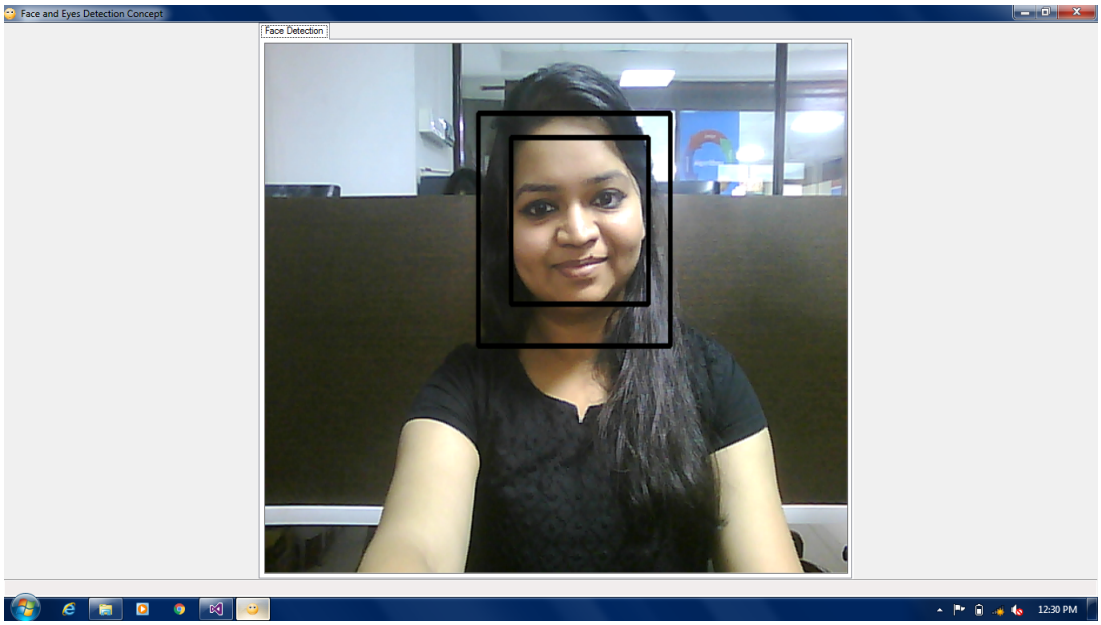


Figure 6.1: Detection Of Face And Open Eyes

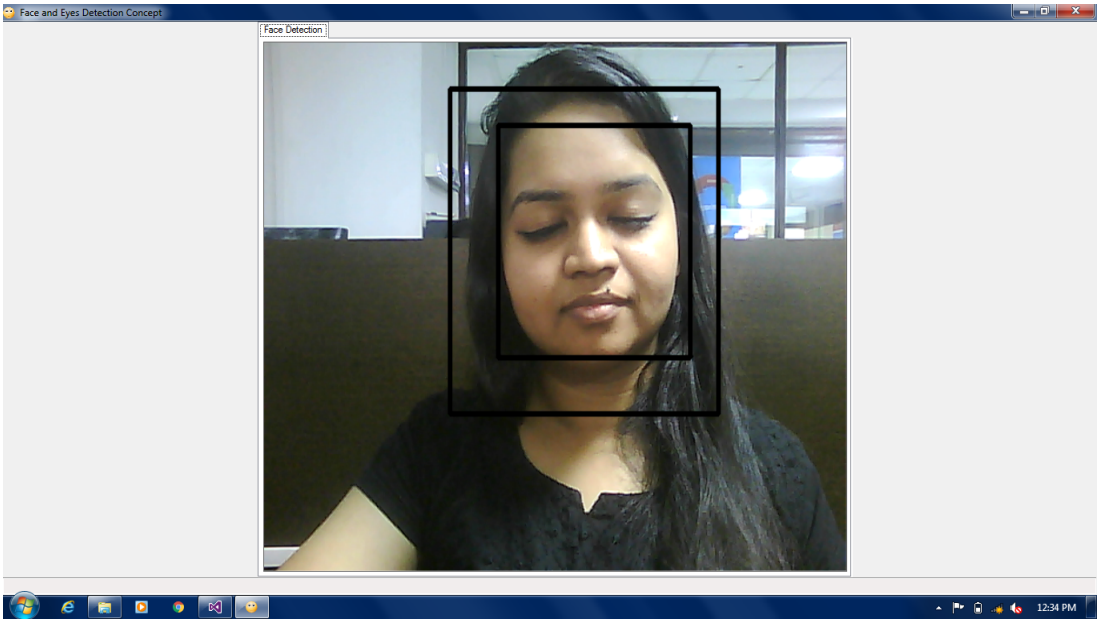


Figure 6.2: Detection Of Face And Closed Eyes

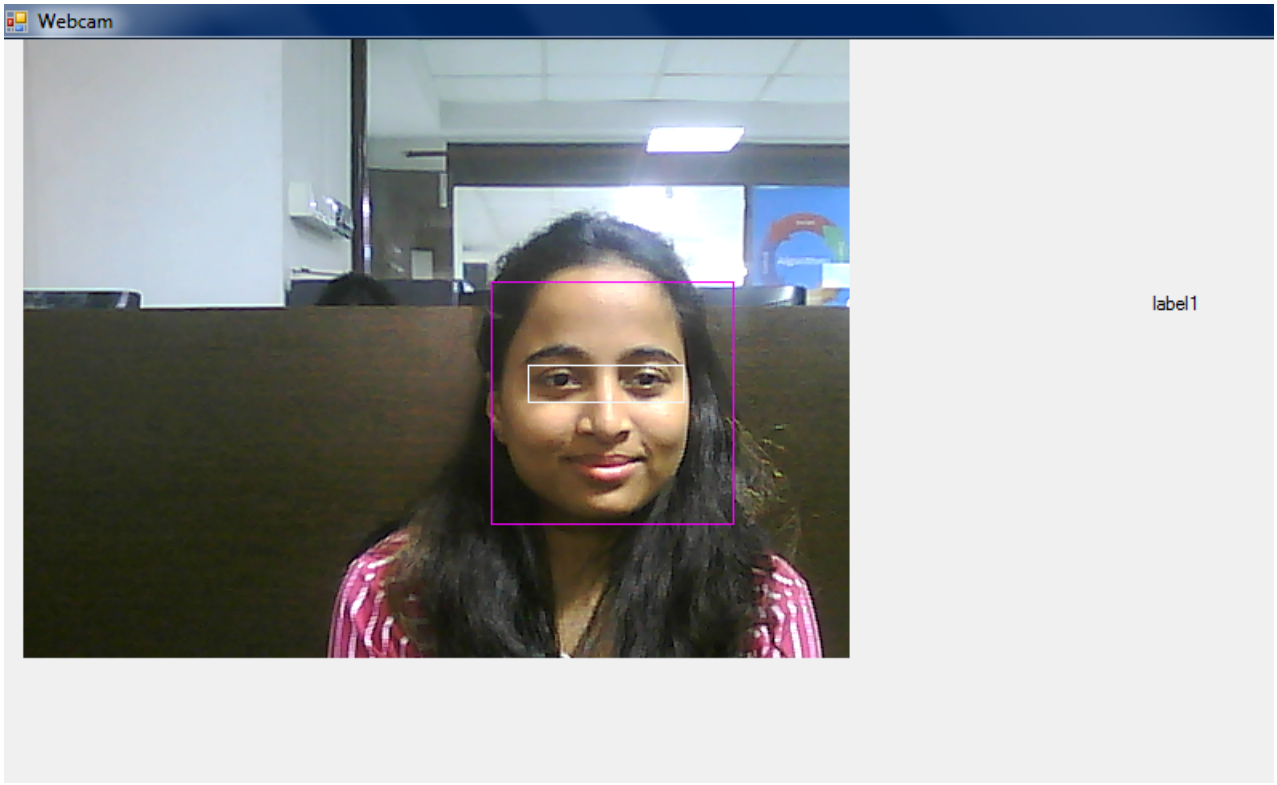


Figure 6.3: Detection Of Face And Open Eyes

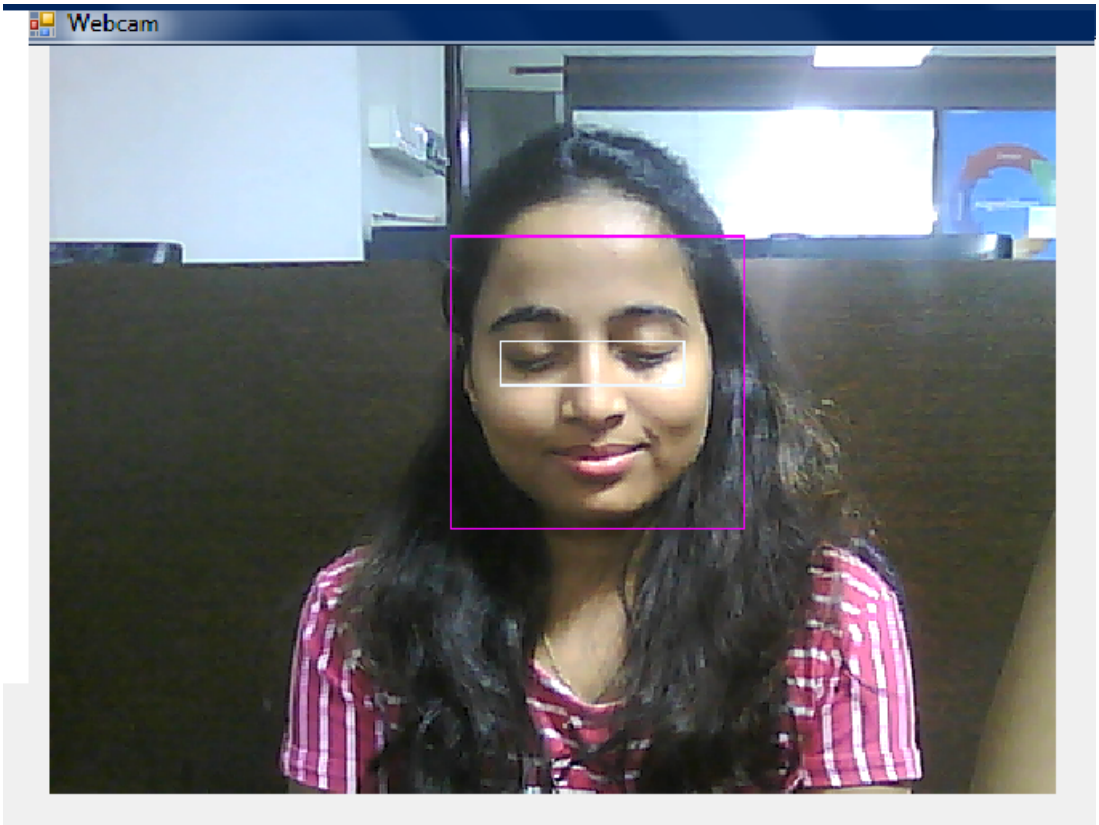


Figure 6.4: Detection Of Face And Closed Eyes

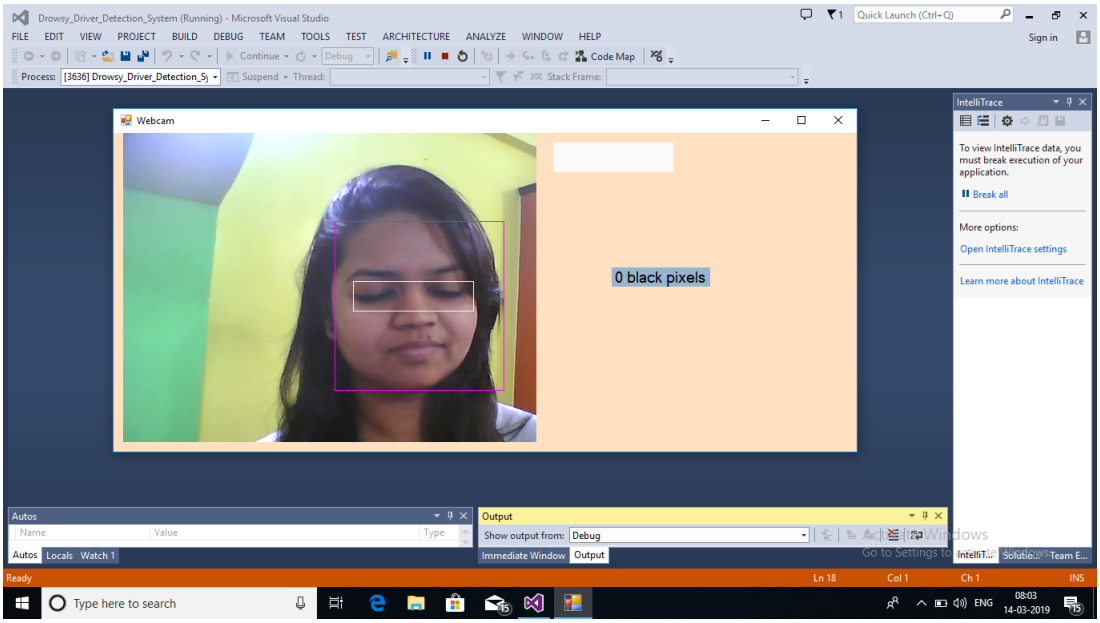


Figure 6.5: Detection Of Closed Eyes With Alarming

Chapter 7

Conclusions

A non-invasive system to localize the eyes and monitor fatigue was developed. Information about the head and eyes position is obtained though various self-developed image processing algorithms. During the monitoring, the system is able to decide if the eyes are opened or closed. When the eyes have been closed for too long, a warning signal is issued. In addition, during monitoring, the system is able to automatically detect any eye localizing error that might have occurred. In case of this type of error, the system is able to recover and properly localize the eyes.
The following conclusions were made:
Image processing achieves highly accurate and reliable detection of drowsiness. Image processing offers a non-invasive approach to detecting drowsiness without the annoyance and interference. A drowsiness detection system developed around the principle of image processing judges the drivers alertness level on the basis of continuous eye closures.

7.1 Future Scope

In the real time driver fatigue detection system it is required to slow down a vehicle automatically when fatigue level crosses a certain limit. Instead of threshold drowsiness level it is suggested to design a continuous scale driver fatigue detection system. It monitors the level of drowsiness continuously and when this level exceeds a certain value a signal is generated which controls the hydraulic braking system of the vehicle.

Hardware components required- Dedicated hardware for image acquisition processing and display interface support with the hydraulic braking system which includes relay, timer, stepper motor and a linear actuator. Function When drowsiness level exceeds a certain limit then a signal is generated which is communicated to the relay through the parallel port parallel data transfer required for faster results.The relay drives the on delay timer and this timer in turn runs the stepper motor for a definite time period .The stepper motor is connected to a linear actuator. The linear actuator converts rotational movement of stepper motor to linear motion. This linear motion is used to drive a shaft which is directly connected to the hydraulic braking system of the vehicle. When the shaft moves it applies the brake and the vehicle speed decreases. Since it brings the vehicle speed down to a controllable limit, the chances of accident occurrence is greatly reduced which is quite helpful for avoiding crashes caused by drowsiness related cases.

Bibliography

[1] Y. Dong, Z. Hu, K. Uchimura and N. Murayama, "Driver Inattention Monitoring System for Intelligent Vehicles: A Review," in IEEE Transactions on Intelligent Transportation Systems, vol. 12, no. 2, pp. 596-614, June 2015.

[2] S. Al-Sultan, A. H. Al-Bayatti and H. Zedan, "Context-Aware Driver Behavior Detection System in Intelligent Transportation Systems," in IEEE Transactions on Vehicular Technology, vol. 62, no. 9, pp. 4264-4275, Nov. 2013.

[3] Tianyi Hong and Huabiao Qin, "Drivers drowsiness detection in embedded system," 2017 IEEE International Conference on Vehicular Electronics and Safety, Beijing, 2007, pp. 1-5. doi: 10.1109/ICVES.2017.4456381

[4] W. Tipprasert, T. Charoenpong, C. Chianrabutra and C. Sukjamsri, "A Method of Drivers Eyes Closure and Yawning Detection for Drowsiness Analysis by Infrared Camera," 2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP), Bangkok, Thailand, 2018

[5] S. D. Lin, J. Lin and C. Chung, "Sleepy Eye's Recognition for Drowsiness Detection," 2013 International Symposium on Biometrics and Security Technologies, Chengdu, 2013, pp. 176-179.

[6] W. Baek, B. Han, K. Kim, Y. Chung and S. Lee, "Real-Time Drowsiness Detection Algorithm for Driver State Monitoring Systems," 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), Prague, 2018, pp. 73-75.

[7] D. Liu, P. Sun, Y. Xiao and Y. Yin, "Drowsiness Detection Based on Eyelid Movement," 2010 Second International Workshop on Education Technology and Computer Science, Wuhan, 2015, pp. 49-52.

Appendices

Appendix-A

Installation steps of Visual studio 2017

Step 1:- Visual Studio can be downloaded from the following link https://www.visualstudio.com/downloads/ You can select Visual Studio 2017 Community Edition Visual Studio 2017 Professional Edition (30 Day Free Trial) In this tutorial, we will install the Professional Edition

Step 2:- Click on the downloaded .exe file

Step 3:- In the next screen, click continue

Step 4:- Visual Studio will start downloading the initial files. Download speed will vary as per your internet connection.

Step 5:- In next screen, click install

Step 6:- In next screen, 1. Select ".Net desktop development" 2. Click install

Step 7:- Visual Studio will download the relevant files based on the selection in step 6

Step 8:- Once the download is done, you will be asked to reboot the PC

Step 9:- Post reboot, open the Visual Studio IDE 1. Select a theme of your choice 2. Click Start Visual Studio

Step 10:- In Visual Studio IDE, you can navigate to File menu to create new C applications

Publication

Paper entitled **"Paper Title"** is presented at **"International Conference/Journal Name"** by **"Author Name".**