DROWSINESS DETECTION SYSTEM

Project Report submitted in partial fulfillment for the award of the degree of

Bachelor Of Technology

Submitted by

ROHIT PARIHAR (18SCSE1010218)

ARHAM KHAN (18SCSE1010350)

IN

COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Under the Supervision of

Dr. Shobha Tyagi

Associate Professor

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

November - December 2021

GALGOTIAS UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report “**DROWSINESS DETECTION SYSTEM”** is the bonafide work of “**Rohit Parihar, Arham KHAN”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
DR MUNISH SABHARWAL	Dr. Shobha Tyagi
Dean of School	**SUPERVISOR**
	PROFESSOR
SCSE	SCSE

Approval Sheet

This thesis/dissertation/report entitled DROWSINESS DETECTION SYSTEM by Rohit Parihar, Arham KHAN, is approved for the degree of BACHELORS OF TECHNOLOGY IN COMPUTER SCIENCE ENGINEERING.

Examiners

____

____

Supervisor (s)

____

____

____

Chairman

____

Date:_________

Place:_________

Statement of Project Report Preparation

1. Thesis title: DROWSINESS DETECTION SYSTEM.
2. Degree for which the report is submitted: BACHELORS OF TECHNOLOGY.
3. Project Supervisor was referred to for preparing the report.
4. Specifications regarding thesis format have been closely followed.
5. The contents of the thesis have been organized based on the guidelines.
6. The report has been prepared without resorting to plagiarism.
7. All sources used have been cited appropriately.
8. The report has not been submitted elsewhere for a degree.

(Signature of the student)

Name: Rohit Parihar

Roll No.:

18SCSE1010218

Statement of Preparation:

Statement of Project Report Preparation

1. Thesis title: DROWSINESS DETECTION SYSTEM.

3. Degree for which the report is submitted: BACHELORS OF TECHNOLOGY.

3. Project Supervisor was referred to for preparing the report.

8. Specifications regarding thesis format have been closely followed.

9. The contents of the thesis have been organized based on the guidelines.

10. The report has been prepared without resorting to plagiarism.

11. All sources used have been cited appropriately.

8. The report has not been submitted elsewhere for a degree.

(Signature of the student)

Name: Arham Khan

Roll No.:18SCSE1010350

Statement of Preparation:

ABSTRACT

This document is a review report on the research conducted and the project made in the field of computer engineering to develop a system for driver drowsiness detection to prevent accidents from happening because of driver fatigue and sleepiness. The report proposed the results and solutions on the limited implementation of the various techniques that are introduced in the project. Whereas the implementation of the project gives the real-world idea of how the system works and what changes can be done in order to improve the utility of the overall system.

Furthermore, the paper states the overview of the observations made by the authors in order to help further optimization in the mentioned field to achieve the utility at a better efficiency for a safer road.

Keywords—Driver drowsiness; eye detection; yawn detection; blink pattern; fatigue

Contents

Introduction

1.1 PURPOSE

Humans have always invented machines and devised techniques to ease and protect their lives, for mundane activities like traveling to work, or for more interesting purposes like aircraft travel. With the advancement in technology, modes of transportation kept on advancing and our dependency on it started increasing exponentially. It has greatly affected our lives as we know it. Now, we can travel to places at a pace that even our grandparents wouldn't have thought possible. In modern times, almost everyone in this world uses some sort of transportation every day. Some people are rich enough to have their own vehicles while others use public transportation. However, there are some rules and codes of conduct for those who drive irrespective of their social status. One of them is staying alert and active while driving.

Neglecting our duties towards safer travel has enabled hundreds of thousands of tragedies to get associated with this wonderful invention every year. It may seem like a trivial thing to most folks but following rules and regulations on the road is of utmost importance. While on road, an automobile wields the most power and in irresponsible hands, it can be destructive and sometimes, that carelessness can harm lives even of the people on the road. One kind of carelessness is not admitting when we are too tired to drive. In order to monitor and prevent a destructive outcome from such negligence, many researchers have written research papers on driver drowsiness detection systems. But at times, some of the points and observations made by the system are not accurate enough. Hence, to provide data and another perspective on the problem at hand, in order to improve their implementations and to further optimize the solution, this project has been done.

1.1.1 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.

1.2 DOCUMENT CONVENTIONS

Main Heading Font size: 24 (bold fonts)

Sub-headings Font size: 16 (bold fonts)

Sub-headings Content Font size: 14 (normal fonts)

1.3 INTENDED AUDIENCE

The intended audience for this document are the development team, the project evaluation jury, and other tech-savvy enthusiasts who wish to further work on the project.

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.

Literature Survey

2.1 SYSTEM REVIEW

This survey is done to comprehend the need and prerequisite of the general population, and to do as such, we went through different sites and applications and looked for the fundamental data. Based on these data, we made an audit that helped us get new thoughts and make different arrangements for our task. We reached the decision that there is a need of such application and felt that there is a decent extent of progress in this field too.

2.2 TECHNOLOGY USED

a. PYTHON - Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed AND supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

b. JUPYTER Lab - Project Jupyter is a nonprofit organization created to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

c. IMAGE PROCESSING - In computer science, digital image processing is the use of computer algorithms to perform image processing on digital images.

d. MACHINE LEARNING - Machine learning is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly told.

Chapter 3

Software Requirements Specification

3.1 Python:

- Python 3

3.2 Libraries

- Numpy
- Scipy
- Playsound
- Dlib
- Imutils
- opencv, etc.

3.3 Operating System

- Windows or Ubuntu

Hardware Requirements Specification

I. Laptop with basic hardware.

II. Webcam

4.Requirement Analysis

Python

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a cycle-detecting garbage collection system (in addition to reference counting). Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Design philosophy and features Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming[58] and metaobjects (magic methods)).[59] Many other paradigms are supported via extensions, including design by contract[60][61] and logic programming.[62] Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management.[63] It also features dynamic name resolution (late binding), which binds method and variable names during program execution. Python's design offers some support for functional programming in the Lisp tradition. It has filter,mapandreduce functions; list comprehensions, dictionaries, sets, and generator expressions.[64] The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.[65] The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:[66] Beautiful is better than ugly. Explicit is better than implicit. Simple is better than complex. Complex is better than complicated. Readability counts. Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach. It is often described as a "batteries included" language due to its comprehensive standard library.

Numpy

NumPy (pronounced /ˈnʌmpaɪ/ (NUM-py) or sometimes /ˈnʌmpi/[3][4] (NUM-pee)) is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project.
NumPy targets the CPython reference implementation of Python, which is a non-

optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy. Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted,[20] and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations. Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

Open CV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.
Let’s start the chapter by defining the term "Computer Vision".
Computer Vision
Computer Vision can be defined as a discipline that explains how to reconstruct,

interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modeling and replicating human vision using computer software and hardware.

Computer Vision overlaps significantly with the following fields:

Image Processing: It focuses on image manipulation.

Pattern Recognition: It explains various techniques to classify patterns.

Photogrammetry: It is concerned with obtaining accurate measurements from images.

Computer Vision Vs Image Processing

Image processing deals with image-to-image transformation. The input and output of
image processing are both images.

Computer vision is the construction of explicit, meaningful descriptions of physical
objects from their image. The output of computer vision is a description or an interpretation of structures in 3D scene.

Applicationsof Computer Vision

Here we have listed down some of major domains where Computer Vision is heavily used.

Robotics Application

- Localization ─ Determine robot location automatically
- Navigation
- Obstacles avoidance
- Assembly (peg-in-hole, welding, painting)
- Manipulation (e.g. PUMA robot manipulator)
- Human Robot Interaction (HRI): Intelligent robotics to interact with and serve people

Medicine Application

Classification and detection (e.g. lesion or cells classification and tumor detection)

1. OpenCV –Overview

OpenCV

2D/3D segmentation

3D human organ reconstruction (MRI or ultrasound)

Vision-guided robotics surgery

Industrial Automation Application

- Industrial inspection (defect detection)
- Assembly
- Barcode and package label reading
- Object sorting
- Document understanding (e.g. OCR)
- Security Application
- Biometrics (iris, finger print, face recognition)
- Surveillance ─ Detecting certain suspicious activities or behaviors
- Transportation Application
- Autonomous vehicle
- Safety, e.g., driver vigilance monitoring

Features of OpenCV Library

Using OpenCV library, you can –

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and
- track objects in it.

OpenCV was originally developed in C++. In addition to it, Python and Java bindings were provided. OpenCV runs on various Operating Systems such as windows, Linux, OSx, FreeBSD, Net BSD, Open BSD, etc. This tutorial explains the concepts of OpenCV with examples using Java bindings.

OpenCV OpenCV Library Modules Following are the main library modules of the

OpenCV library.

Core Functionality

This module covers the basic data structures such as Scalar, Point, Range, etc., that are used to build OpenCV applications. In addition to these, it also includes the multidimensional array Mat, which is used to store the images. In the Java library of OpenCV, this module is included as a package with the name org.opencv.core.

Image Processing

This module covers various image processing operations such as image filtering, geometrical image transformations, color space conversion, histograms, etc. In the Java library of OpenCV, this module is included as a package with the name org.opencv.imgproc. Video This module covers the video analysis concepts such as motion estimation, background subtraction, and object tracking. In the Java library of OpenCV, this module is included as a package with the name org.opencv.video. Video I/O This module explains the video capturing and video codecs using OpenCV library. In the Java library of OpenCV, this module is included as a package with the name org.opencv.videoio. calib3d This module includes algorithms regarding basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence and elements of 3D reconstruction. In the Java library of OpenCV, this module is included as a package with the name org.opencv.calib3d. features2d This module includes the concepts of feature detection and description. In the Java library of OpenCV, this module is included as a package with the name org.opencv.features2d. Objdetect This module includes the detection of objects and instances of the predefined classes such as faces, eyes, mugs, people, cars, etc. In the Java library of OpenCV, this module is included as a package with the name org.opencv.objdetect. Highgui This is an easy-to-use interface with simple UI capabilities. In the Java library of OpenCV, the features of this module is included in two different packages namely, org.opencv.imgcodecs and org.opencv.videoio.OpenCV

A Brief History of OpenCV

OpenCV was initially an Intel research initiative to advise CPU-intensive applications. It was officially launched in 1999.In the year 2006, its first major version, OpenCV 1.0 was released. In October 2009, the second major version, OpenCV 2 was released. In August 2012, OpenCV was taken by a nonprofit organization OpenCV.org

DLib

DLib-ml implements numerous machine learning algorithms:SVMs, K-Means clustering, Bayesian Networks,and many others. DLib also features utility functionality including

- Threading,
- Networking,
- Numerical Algorithms,
- Image Processing,
- and Data Compression and Integrity algorithms.

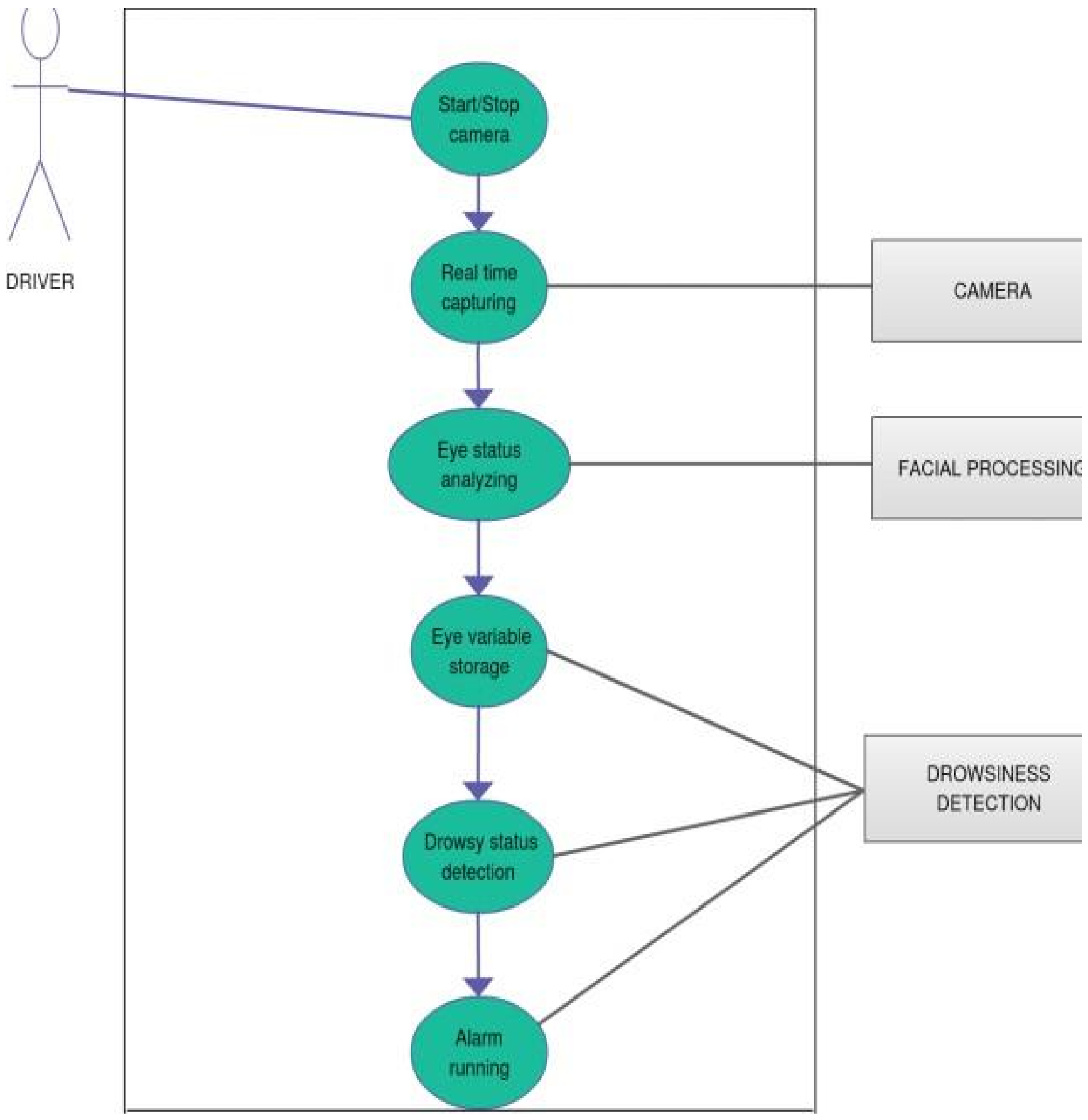
DLib includes extensive unit testing coverage and examples using the library. Every class and function in the library is documented. This documentation can be found on the library's home page. DLib provides a good framework for developing machine learning applications in C++.

DLib is much like DMTL in that it provides a generic high-performance machine learning toolkit with many different algorithms, but DLib is more recently updated and has more examples. DLib also contains much more supporting functionality.

What makes DLib unique is that it is designed for both research use and creating machine learning applications in C++

System Design

5.1 USE CASE DIAGRAM



5.2 ACTIVITY DIAGRAM

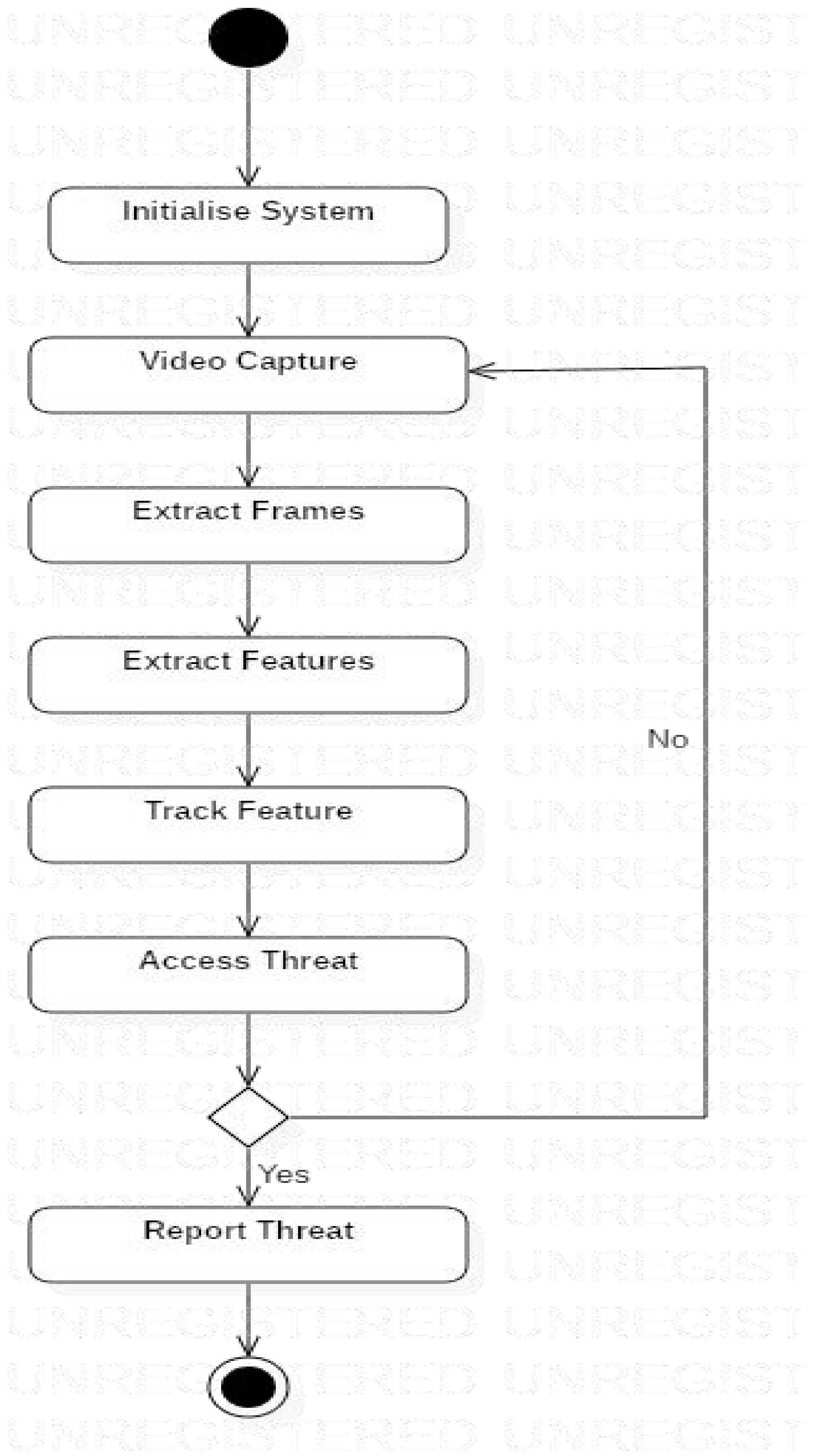


FIGURE 2

5.3 CLASS DIAGRAM

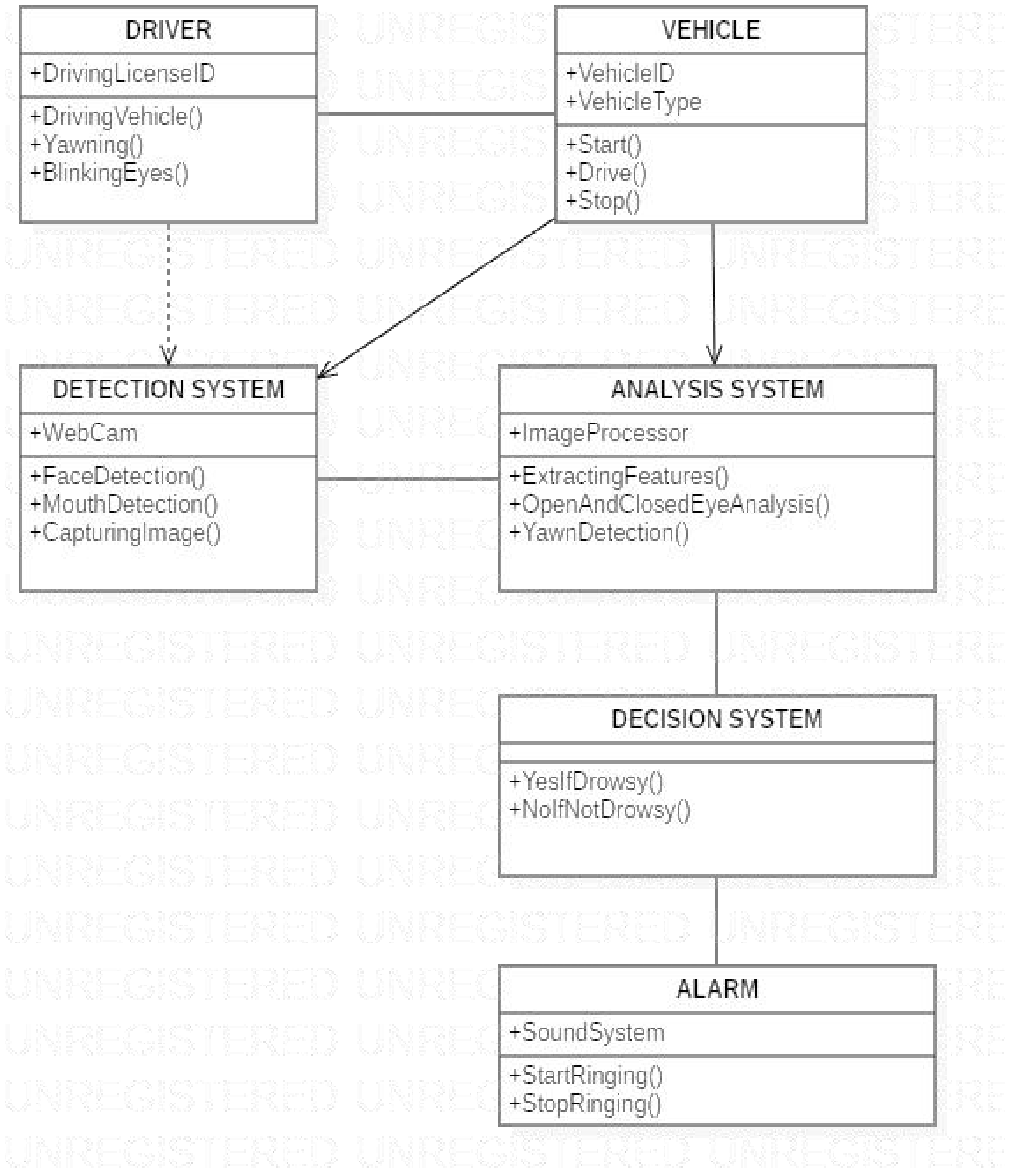


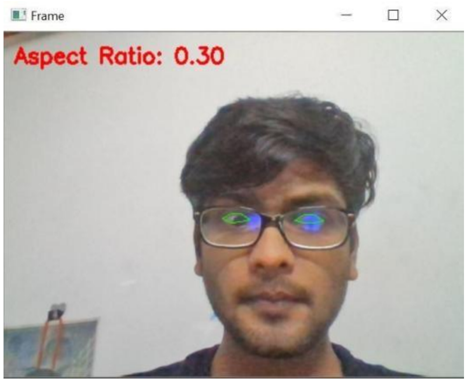
FIGURE 3

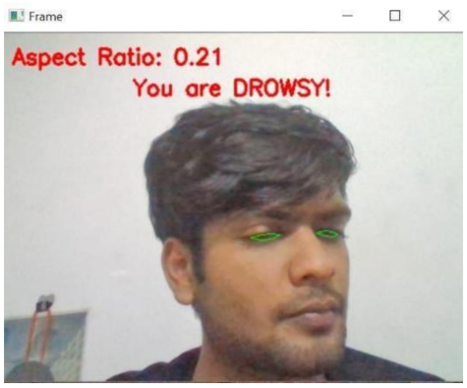
System Testing

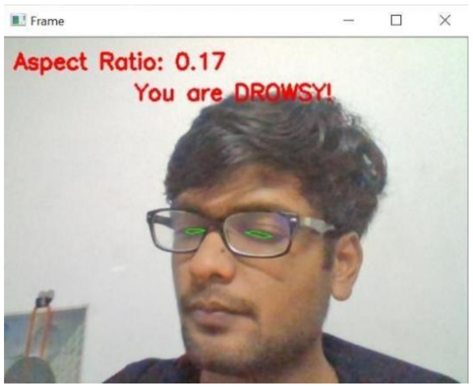
6.1 Test Cases and Test Results

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	NSGY	Straight Face, Good Light, With Glasses	Non Drowsy	Non Drowsy
T02	YTGN	Tilted Face, Good Light, No Glasses	Drowsy	Drowsy
T03	YTGY	Tilted Face, Good Light, With Glasses	Drowsy	Drowsy

Note: Testing is performed manually







Project Planning

7.1 SYSTEM MODEL

The framework is created utilizing the incremental model. The center model of the framework is first created and afterwards augmented in this way in the wake of testing at each turn. The underlying undertaking skeleton was refined into expanding levels of ability.

At the following incremental level, it might incorporate new execution backing and improvement.

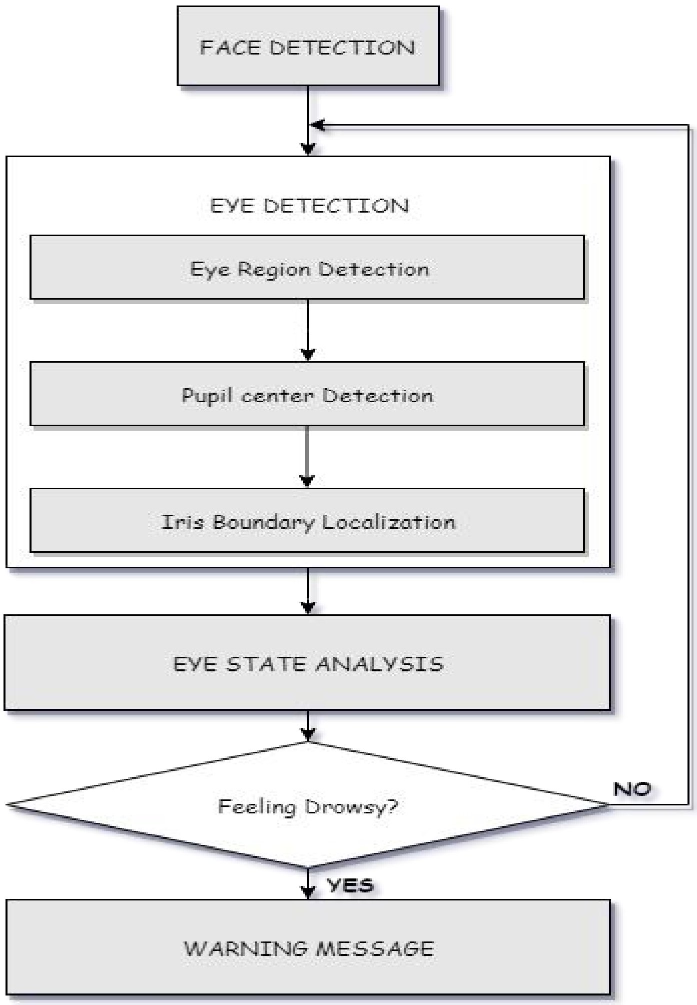
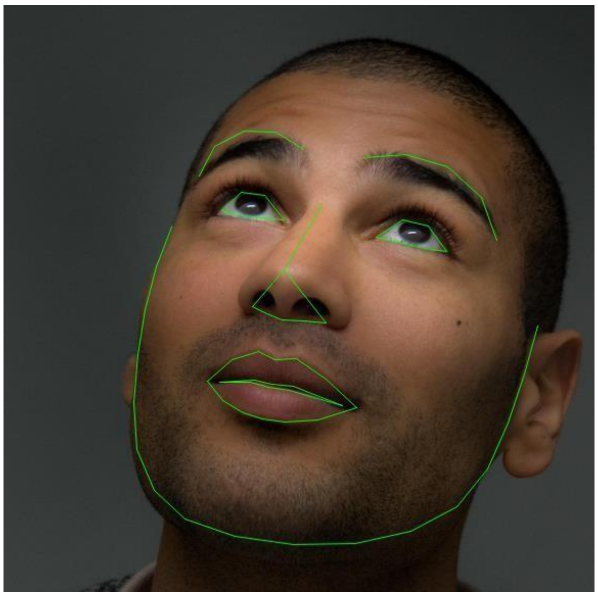


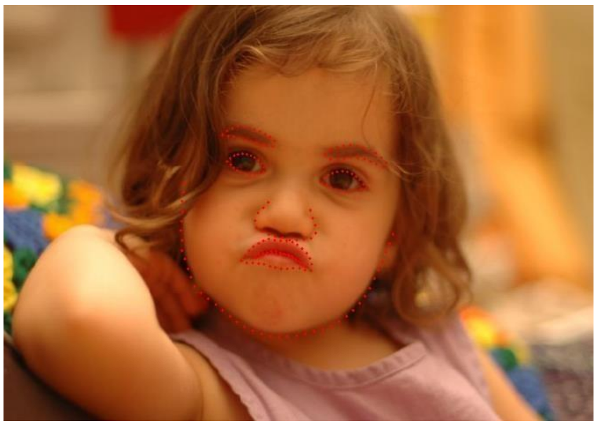
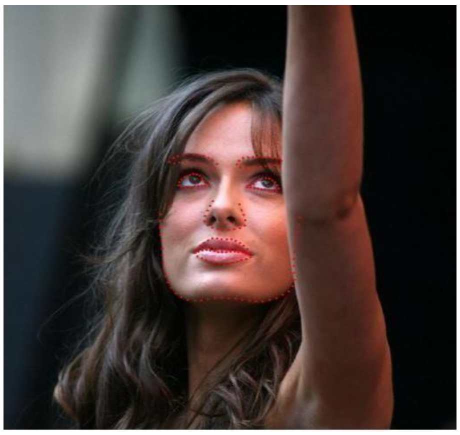
Figure 4: Block diagram

Chapter 8

Implementation

- In our program we used Dlib, a pre-trained program trained on the HELEN dataset to detect human faces using the pre-defined 68  landmarks.

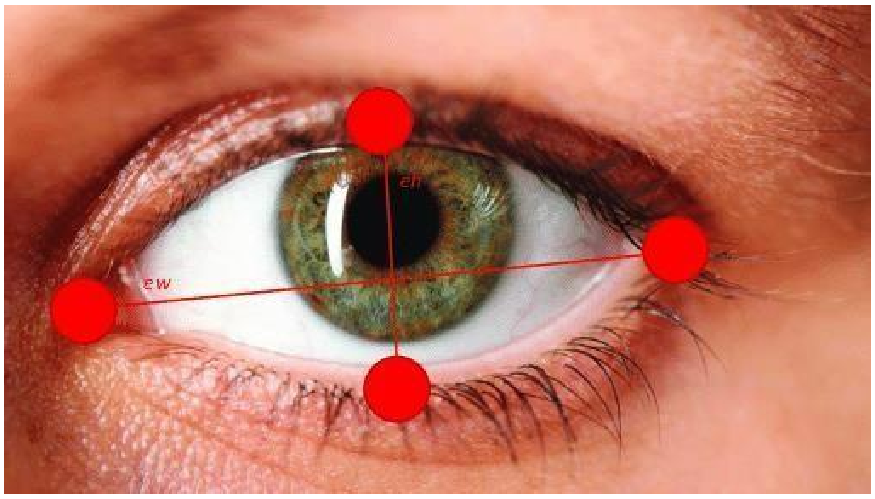
Landmarked Image of a person by Dlib

HELEN Dataset Samples

- After passing our video feed to the dlib frame by frame, we are able to detect left eye and right eye features of the face.

- Now, we drew contours around it using OpenCV.
- Using Scipy’s Euclidean function, we calculated sum of both eyes’ aspect ratio which is the sum of 2 distinct vertical distances betweenthe eyelids divided by its horizontal distance.



Eyes with horizontal and vertical distance marked for Eye Aspect Ratiocalculation.

- Now we check if the aspect ratio value is less than 0.25 (0.25 waschosen as a base case after some tests). If it is less an alarm is sounded and user is warned.

ALGORITHM

Recognition of Eye's State:The eye area can be estimated from optical flow, by sparse tracking or by frame-to-frame intensity differencing and adaptive thresholding. And Finally, a decision is made whether the eyes are or are not covered by eyelids. A different approach is to infer the state of the eye opening from a single image, as e.g. by correlation matching with open and closed eye templates, a heuristic horizontal or vertical image intensity projection over the eye region, a parametric model fitting to find the eyelids, or active shape models. A major drawback of the previous approaches is that they usually implicitly impose too strong requirements on the setup, in the sense of a relative face-camera pose (head orientation), image resolution, illumination, motion dynamics, etc. Especially the heuristic methods that use raw image intensity are likely to be very sensitive despite their real-time performance. Therefore, we propose a simple but efficient algorithm to detect eye blinks by using a recent facial landmark detector. A single scalar quantity that reflects a level of the eye opening is derived from the landmarks. Finally, having a per-frame sequence of the eye-opening estimates, the eye blinks are found by an SVM classifier that is trained on examples of blinking and non-blinking patterns. Eye Aspected Ratio Calculation: For every video frame, the eye landmarks are detected. The eye aspect ratio (EAR) between height and width of the eye is computed.

$$\mathrm{EAR} = ||p2 - p6|| + ||p3 - p5|| \ (1) \ 2||p1 - p4||$$

where p1, . . ., p6 are the 2D landmark locations, depicted in Fig. 1. The EAR is mostly constant when an eye is open and is getting close to zero while closing an eye. It is partially person and head pose insensitive. Aspect ratio of the open eye has a small variance among individuals, and it is fully invariant to a uniform scaling of the image and in-plane rotation of the face. Since eye blinking is performed by both eyes synchronously, the EAR of both eyes is averaged.

Source Code

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Drowsiness Detection OpenCV\n",
    "\n",
    "\n",
    "This code can detect your eyes and alert when the user is drowsy.\n",
    "\n",
    "## Applications\n",
    "This can be used by riders who tend to drive for a longer period of time that may
lead to accidents.\n",
    "\n",
    "### Algorithm\n",
    "\n",
    "Each eye is represented by 6 (x, y)-coordinates, starting at the left-corner of the eye
(as if you were looking at the person), and then working clockwise around the eye:.\n",
    "\n",
    "<img src=\"eye1.jpg\">\n",
    "\n",
    "### Condition\n",
    "\n",
    "It checks 20 consecutive frames and if the Eye Aspect ratio is lesst than 0.25, Alert
is generated.\n",
    "\n",
   "#### Relationship\n",
   "\n",
   "<img src=\"eye2.png\">\n",
   "\n",
   "#### Summing up\n",
   "\n",
   "<img src=\"eye3.jpg\">\n",
   "\n",
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 1,
  "metadata": {},
  "outputs": [],
  "source": [
   "from scipy.spatial import distance\n",
   "from imutils import face_utils\n",
   "import imutils\n",
   "import dlib\n",
   "import cv2"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 2,
  "metadata": {},
  "outputs": [],
  "source": [
   "def eye_aspect_ratio(eye):\n",
   "\tA = distance.euclidean(eye[1], eye[5])\n",
   "\tB = distance.euclidean(eye[2], eye[4])\n",
   "\tC = distance.euclidean(eye[0], eye[3])\n",
   "\tear = (A + B) / (2.0 * C)\n",
   "\treturn ear"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 3,
  "metadata": {},
  "outputs": [],
  "source": [
   "thresh = 0.25\n",
   "frame_check = 20\n",
   "detect = dlib.get_frontal_face_detector()\n",
   "predict = dlib.shape_predictor(\"shape_predictor_68_face_landmarks.dat\")"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 4,
  "metadata": {},
  "outputs": [],
  "source": [
   "(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS[\"left_eye\"]\n",
   "(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS[\"right_eye\"]"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 5,
"metadata": {},
"outputs": [],
"source": [
 "cap=cv2.VideoCapture(0)\n",
 "flag=0\n",
 "while True:\n",
 "\tret, frame=cap.read()\n",
 "\tframe = imutils.resize(frame, width=450)\n",
 "\tgray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)\n",
 "\tsubjects = detect(gray, 0)\n",
 "\tfor subject in subjects:\n",
 "\t\tshape = predict(gray, subject)\n",
 "\t\tshape = face_utils.shape_to_np(shape)#converting to NumPy Array\n",
 "\t\tleftEye = shape[lStart:lEnd]\n",
 "\t\trightEye = shape[rStart:rEnd]\n",
 "\t\tleftEAR = eye_aspect_ratio(leftEye)\n",
 "\t\trightEAR = eye_aspect_ratio(rightEye)\n",
 "\t\tear = (leftEAR + rightEAR) / 2.0\n",
 "\t\tleftEyeHull = cv2.convexHull(leftEye)\n",
 "\t\trightEyeHull = cv2.convexHull(rightEye)\n",
 "\t\tcv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)\n",
 "\t\tcv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)\n",
 "\t\tif ear < thresh:\n",
 "\t\t\tflag += 1\n",
 "\t\t\t#print (flag)\n",
 "\t\t\tif flag >= frame_check:\n",
 "\t\t\t\tcv2.putText(frame, \"****************ALERT!****************\", (10, 30),\n",
 "\t\t\t\t\tcv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)\n",
 "\t\t\t\tcv2.putText(frame, \"****************ALERT!****************\", (10,325),\n",
 "\t\t\t\t\tcv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)\n",
 "\t\telse:\n",
    "\t\t\tflag = 0\n",
    "\tcv2.imshow(\"Frame\", frame)\n",
    "\tkey = cv2.waitKey(1) & 0xFF\n",
    "\tif key == ord(\"q\"):\n",
    "\t\tcv2.destroyAllWindows()\n",
    "\t\tcap.release()\n",
    "\t\tbreak\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.6.4"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
from scipy.spatial import distance
from imutils import face_utils
import imutils
import dlib
import cv2

def eye_aspect_ratio(eye):
        A = distance.euclidean(eye[1], eye[5])
        B = distance.euclidean(eye[2], eye[4])
        C = distance.euclidean(eye[0], eye[3])
        ear = (A + B) / (2.0 * C)
        return ear

thresh = 0.25
frame_check = 20
detect = dlib.get_frontal_face_detector()
predict = dlib.shape_predictor(".\shape_predictor_68_face_landmarks.dat")# Dat file is
the crux of the code

(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["right_eye"]
cap=cv2.VideoCapture(0)
flag=0
```

```
while True:
        ret, frame=cap.read()
        frame = imutils.resize(frame, width=450)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        subjects = detect(gray, 0)
        for subject in subjects:
                shape = predict(gray, subject)
                shape = face_utils.shape_to_np(shape)#converting to NumPy Array
                leftEye = shape[lStart:lEnd]
                rightEye = shape[rStart:rEnd]
                leftEAR = eye_aspect_ratio(leftEye)
                rightEAR = eye_aspect_ratio(rightEye)
                ear = (leftEAR + rightEAR) / 2.0
                leftEyeHull = cv2.convexHull(leftEye)
                rightEyeHull = cv2.convexHull(rightEye)
                cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
                cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
                if ear < thresh:
                        flag += 1
                        print (flag)
                        if flag >= frame_check:
                                cv2.putText(frame, "****************ALERT!****************", (10,
30),
                                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                                cv2.putText(frame, "****************ALERT!****************",
(10,325),
                                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                                #print ("Drowsy")
                else:
                        flag = 0
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
                Break
from scipy.spatial import distance
from imutils import face_utils
import imutils
import dlib
import cv2

def eye_aspect_ratio(eye):
        A = distance.euclidean(eye[1], eye[5])
        B = distance.euclidean(eye[2], eye[4])
        C = distance.euclidean(eye[0], eye[3])
        ear = (A + B) / (2.0 * C)
        return ear

thresh = 0.25
frame_check = 20
detect = dlib.get_frontal_face_detector()
predict = dlib.shape_predictor(".\shape_predictor_68_face_landmarks.dat")# Dat file is
the crux of the code

(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["right_eye"]
cap=cv2.VideoCapture(0)
flag=0
while True:
        ret, frame=cap.read()
        frame = imutils.resize(frame, width=450)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        subjects = detect(gray, 0)
```

```
    for subject in subjects:
        shape = predict(gray, subject)
        shape = face_utils.shape_to_np(shape)#converting to NumPy Array
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2.0
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
        if ear < thresh:
            flag += 1
            print (flag)
            if flag >= frame_check:
                cv2.putText(frame, "****************ALERT!****************", (10,
30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                cv2.putText(frame, "****************ALERT!****************",
(10,325),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                #print ("Drowsy")
        else:
            flag = 0
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break
```

Conclusion and Future Scope

10.1 Conclusion

It completely meets the objectives and requirements of the system. The framework has achieved an unfaltering state where all the bugs have been disposed of. The framework cognizant clients who are familiar with the framework and comprehend it's focal points and the fact that it takes care of the issue of stressing out for individuals having fatigue-related issues to inform them about the drowsiness level while driving.

10.2 Future Scope

The model can be improved incrementally by using other parameters like blink rate, yawning, state of the car, etc. If all these parameters are used it can improve the accuracy by a lot.

We plan to further work on the project by adding a sensor to track the heart rate in order to prevent accidents caused due to sudden heart attacks to drivers.

Same model and techniques can be used for various other uses like Netflix and other streaming services can detect when the user is asleep and stop the video accordingly. It can also be used in application that prevents user from sleeping.

References

[1] COMPUTATIONALLY EFFICIENT FACE DETECTION; B. SCHLKOPF-A. BLAKE, S. ROMDHANI, AND P. TORR.

[2] USE OF THE HOUGH TRANSFORMATION TO DETECT LINES AND CURVES IN PICTURE; R. DUDA AND P. E. HART.

[3] JAIN, “FACE DETECTION IN COLOR IMAGES; R. L. HSU, M. ABDEL-MOTTALEB, AND A. K. JAIN.

[4] OPEN/CLOSED EYE ANALYSIS FOR DROWSINESS DETECTION; P.R. TABRIZI AND R. A. ZOROOFI.

[5] http://ncrb.gov.in/StatPublications/ADSI/ADSI2015/chapter1A%20traffic%20accidents.pdf

[6] http://www.jotr.in/text.asp?2013/6/1/1/118718

[7] http://dlib.net/face_landmark_detection_ex.cpp.html