DRIVER'S DROWSINESS DETECTION

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

Of

Bachelor of Technology
In
Computer Science & Engineering

By:

Prabh Simran Singh (090/CSE/2024)
Mandeep Kaur (075/CSE/2024)
Jaskirat Singh Bedi (064/CSE/2024)
Kavya Bisht (073/CSE/2024)

Under the Guidance of
Dr. Aashish Bhardwaj

Department of Computer Science & Engineering
Guru Tegh Bahadur Institute of Technology
Guru Gobind Singh Indraprastha University
Dwarka, New Delhi
Year 2020-2024

DECLARATION

We hereby declare that all the work presented in the dissertation entitled DRIVER'S DROWSINESS DETECTION in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science, Guru Tegh Bahadur Institute of Technology, affiliated to Guru Gobind Singh Indraprastha University Delhi is an authentic record of our own work carried out under the guidance of Dr. Aashish Bhardwaj

Date:

Prabh Simran Singh
(090/CSE2/2024)

Mandeep Kaur
(075/CSE2/2024)

Jaskirat Singh Bedi
(064/CSE2/2024)

Kavya Bisht
(073/CSE2/2024)

CERTIFICATE

This is to certify that dissertation entitled “DRIVER'S DROWSINESS DETECTION”,which is submitted by Mr. Prabh Simran Singh, Miss Mandeep Kaur, Mr Jaskirat Singh Bedi and Miss Kavya Bisht in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science, Guru Tegh Bahadur Institute of Technology, New Delhi is an authentic record of the candidate’s own work carried out by them under our guidance. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Dr. Aashish Bhardwaj
(Project Mentor)

Dr. Aashish Bhardwaj
(Head of Department)
(Computer Science)

Date:

ACKNOWLEDGEMENT

We would like to express our great gratitude towards our supervisor, Dr. Aashish Bhardwaj who has given us support and suggestions. Without their help we could not have presented this dissertation up to the present standard. We also take this opportunity to give thanks to all others who gave us support for the project or in other aspects of our study at Guru Tegh Bahadur Institute of Technology.

Date:

Prabh Simran Singh
(090/CSE2/2024)
Prabhsimransingh142@gmail.com

Mandeep Kaur
(075/CSE2/2024)
Mkhunjan11@gmail.com

Jaskirat Singh Bedi
(064/CSE2/2024)
Jaskirat.bedi@gmail.com

Kavya Bisht
(073/CSE2/2024)
Kavya.bisht@gmail.com

ABSTRACT

Driving a car is a complex, multifaceted and potentially risky activity requiring full mobilization of physiological and cognitive resources to maintain performance over time. Car accident is the major cause of death in which around 1.3 million people die every year. Majority of these accidents are caused because of distraction or the drowsiness of driver. Construction of high speed highway roads had diminished the margin of error for the driver. The countless number of people drives for long distance every day and night on the highway. Lack of sleep or distractions like the phone call, talking with the passenger, etc may lead to an accident. To prevent such accidents we propose a system which alerts the driver if the driver gets distracted or feels drowsy.

So, our project aims to protect individual's life and to prevent accidents .It is a car safety technology which helps prevent accidents caused by the driver getting drowsy. Various studies have suggested that around 20% of all road accidents are fatigue-related, up to 50% on certain roads. We have used driver's face as well as his/her eyes to analyze the measure of drowsiness associated with it.

Facial landmarks detection is used with help of image processing of images of the face captured using the camera, for detection of distraction or drowsiness. This whole system is deployed on portable hardware which can be easily installed in the car for use. We have made a system that accurately manages the open as well as closed state of the driver's eyes in real time also facial recognition is also used to identify the driver. Eye aspect ratio is also used to determine the driver's level of drowsiness.

Our prime reason to work this project was to provide cheap solutions for individual to have safe and secure drive. Not everyone can afford expensive high tech cars like Tesla so using our software we ensure individual could have sense security at a minimal cost. Plus, it minimizes the accidents that happens due to negligence of drivers and help in ensuring their well-being. The project is favorable to all the users from different background.

LIST OF FIGURES AND TABLES

2.1 INTRODUCTION

This project aims to ensure the driver's safety by providing a software tool .It makes everyday life easy and have safety while driving. As we know frequent road accidents are happening nowadays. It is due to increased vehicle density , violation of rules and carelessness. India ranks fifth in road accidents over the world. We have the objective to minimize the road accidents due to above mentioned facts in real time using such technological platform in low cost It is a reliable and cost effective software.

In this project we create, delete or update driver's details of the user using tkinter and the data is stores in SQLite3.User can register as well as login his/her details. Then after logging his/her credentials our system can detect and recognize ,this is done by collecting the dataset of driver. This is done with the help of Then we have made measure the eye aspect ratio to detect the drowsiness. We also live stream the driving while driving and an alarm system have been included in the system so that the alarm alert the driver.

Further on the ,we have given a feature in the project in which when the driver is asleep a message would be sent to his/her family members so that they can make him a call to get him awake.Twilio is used for providing the messaging features,

Overall ,our program ensures the well being and security of the driver while driving .As getting drowsy accounts for a major amount of accident and if we provide this software to the masses it would benefit a lot of peoples in their day to day lives.

2.2 MOTIVATION

Our motivation arises from the necessity as India ranks 5th in the amount of road accident that happens. Our efforts will provide the effective solution to frequent drivers and help them to have a safe and sound journey .

Mostly accidents are caused in cases when drivers fell sleepy during the drive. The software monitors the drivers eyes during the ride and whenever he seems to be drowsy for a long while to make him conscious an alarm buzz is used. The family members are informed in the case the driver doesn't take rest when feeling extremely tired or restless. This thus will ensure that no accidents are caused due to drowsiness Even an individual with bare minimum level of knowledge regarding PCs will be able to interact and utilize the software easily. We will provide a low budget functionality to fleet owners and other people to manage their drives and moreover preventing accidents, thus enhancing the security.

2.3 Project Overview

As we know frequent road accidents are happening nowadays. It is due to increased vehicle density, violation of rules and carelessness. India ranks fifth in road accidents over the world. We have the objective to minimize the road accidents due to above mentioned facts in real time using such technological platform in low cost. Some of the main objectives are:-

- Face detection
- Recognize the driver
- Live video stream monitoring

- Drowsiness Detection of driver while driving.
- Alarm buzz to alert the driver
- Message alerts for informing the family members

2.4 MODULES

The device consists of the following software components:

1. Image Processing Algorithm: An image processing algorithm is used to analyze the images captured by the camera and detect the driver's drowsiness. The algorithm is designed to detect the following signs of drowsiness: • Slow eye blinking • Eye closure duration

2. Various libraries are being used here such as:

   • Tensorflow • Keras • OpenCV • NumPy • Random • OS • PyGame

3. Machine Learning Algorithm: A supervised machine learning algorithm is used to classify the level of drowsiness based on the features extracted from the image processing algorithm.

4. Alert System: An alert system is used to notify the driver when he/she is drowsy.

5. Sending alert to selected contacts for emergency cases.

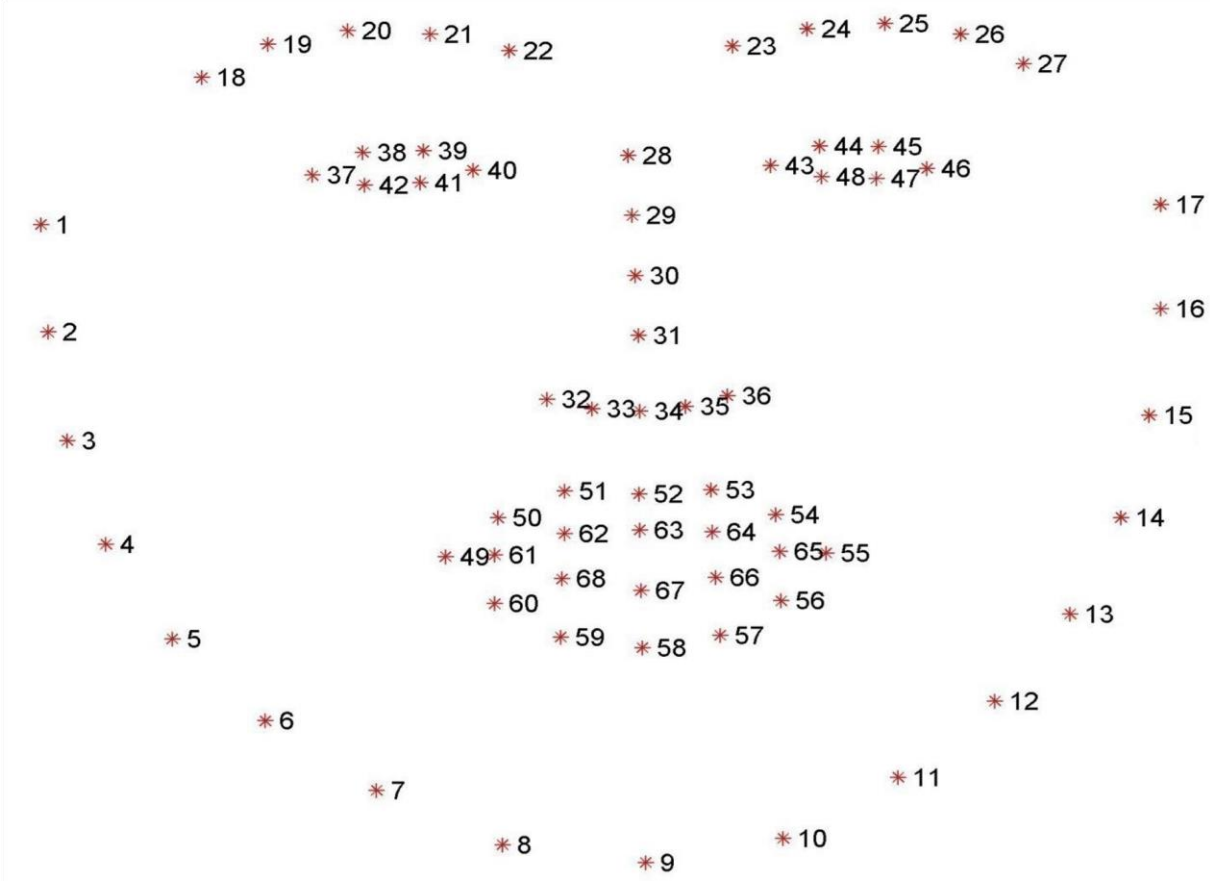

Fig 1

Blink Detection and Evaluation - The blinks are counted and calculate the Euclidean distance both horizontal and vertical between the eyes. In terms of blink detection, we are only interested in two sets of facial structures — the eyes.

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 remainder of the region:

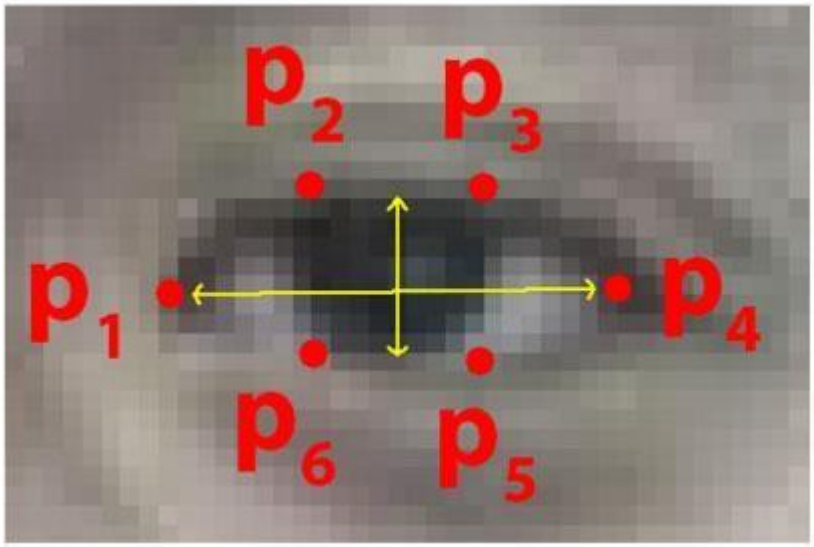

Fig.2

Based on this image, we should take away on key point:

There is a relation between the *width* and the *height* of these coordinates.

We can then derive an equation that reflects this relation called the *eye aspect ratio* (EAR):

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Eye aspect ratio equation.

Where *p1, …, p6* are 2D facial landmark locations.

The numerator of this equation computes the distance between the vertical eye landmarks while the denominator computes the distance between horizontal eye landmarks, weighting the denominator appropriately since there is only *one* set of horizontal points but *two* sets of vertical points.

Well, as we'll find out, the eye aspect ratio is approximately constant while the eye is open, but will rapidly fall to zero when a blink is taking place.Using this simple equation, we can *avoid image*

processing techniques and simply rely on the *ratio of eye landmark distances* to determine if a person blinking.

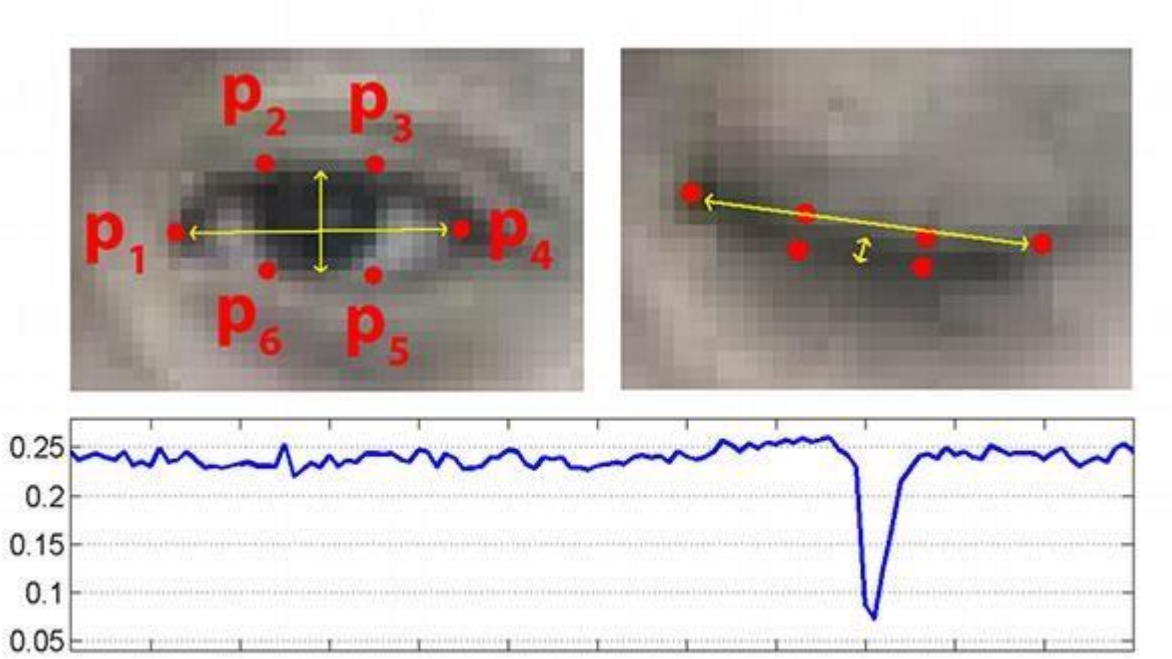

Fig 3

6. Alarm Alert – Driver is alerted with use of alarm to make him conscious and focus on the drive. The alarm is executed in a separate thread than the main thread of application thus using threading concept to increase the efficiency.

3. REQUIREMENT ANALYSIS AND SPECIFICATIONS

3.1. Hardware Requirements

- 4 Gb RAM at least in Systems running Windows 7 Sp1 and above.
- Stable internet Connection via Ethernet or Wi-fi
- Proper working Webcam

3.2 Software Requirements

Python
OpenCV
Numpy
Tensorflow
Keras
OS
Random
PyGame
Time
SQLite3
Windows
Python IDLE (Integrated Development Environment)

3.3 Functional Requirement

Functional Requirements define specific behavior or functions. Functional requirement define what a system is supposed to and are usually in the form of "system shall do requirement", an individual action of part of the system.

- Register the driver based on details.
- Face detection.
- Recognize the driver as per the validated details.
- Live video stream monitoring Drowsiness .
- Detection of driver while driving.
- Alarm buzz to alert the driver.

3.4 Nonfunctional Requirement

A Non – Functional Requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. Functional requirements define what a system is supposed to do and nonfunctional requirements define how a system is supposed to be. Nonfunctional requirements are often called "quality attributes" of a system.

Some of the nonfunctional requirements are:

Availability: Friendly user interface making it easy for user to access everything.

Correctness: All the action taking place must contribute to an apt result without producing errors while performing operations.

Maintainability: it should be easy to maintain for future updating.

4. System Design

DATA FLOW DIAGRAM

Level 0:

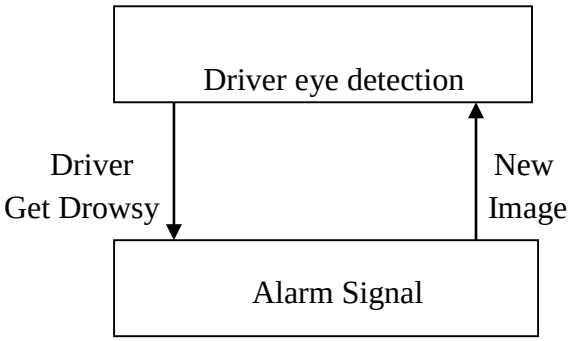

Fig 4

Level 1:

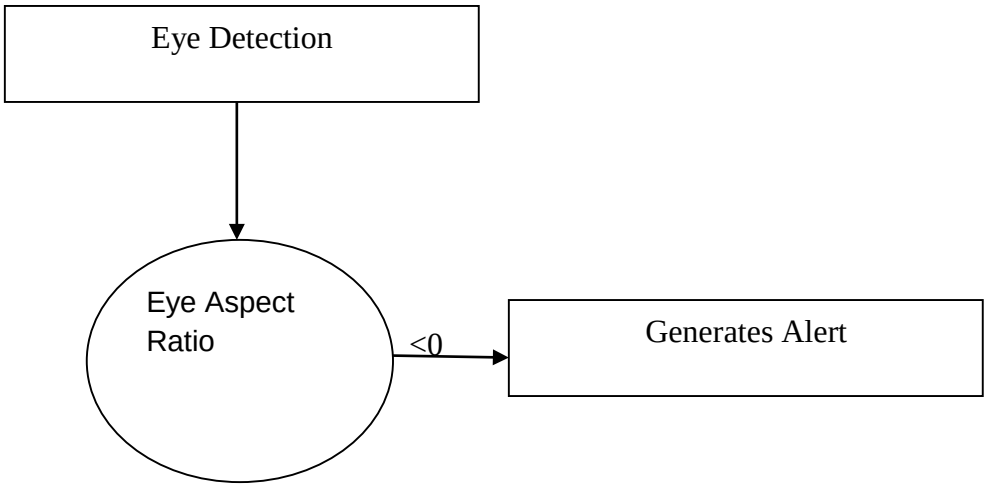

Fig 5

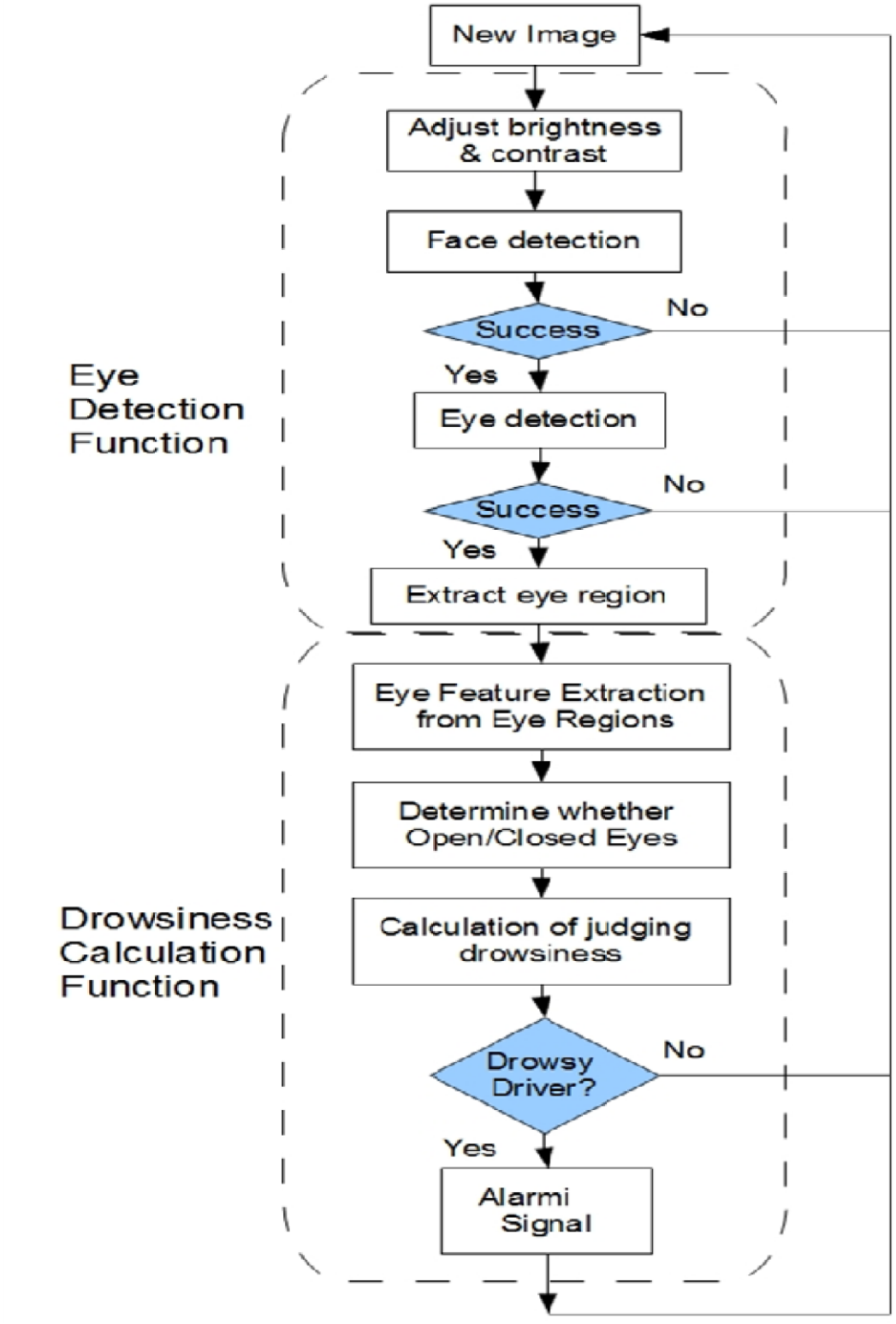
New Image
Adjust brightness & contrast
Face detection
Success
No
Yes
Eye detection
Success
No
Yes
Extract eye region
Eye Detection Function
Eye Feature Extraction from Eye Regions
Determine whether Open/Closed Eyes
Calculation of judging drowsiness
Drowsy Driver?
No
Yes
Alarmi Signal
Drowsiness Calculation Function

Fig 6

5. Body of Thesis

5.1 OpenCV

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDAand OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

5.2 PYTHON

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

5.3 TensorFlow

TensorFlow is an open-source machine learning library developed by the Google Brain team. It is designed to facilitate the development and deployment of machine learning models, particularly for deep learning tasks. TensorFlow provides a comprehensive set of tools and resources for building and training various types of neural network models.

1. Graph Computation:- TensorFlow uses a computational graph to represent and execute machine learning models. Nodes in the graph represent mathematical operations, and edges represent the data flowing between them.

2. TensorFlow Ecosystem:- TensorFlow has a vast ecosystem that includes TensorFlow Lite for mobile and embedded devices, TensorFlow Extended (TFX) for production deployment, and TensorFlow.js for browser-based applications.

3. High-Level APIs:- TensorFlow provides high-level APIs, such as Keras, which simplify the process of building and training neural networks. This makes it accessible to both beginners and experienced researchers.

4. Flexibility:- TensorFlow allows users to define and train custom machine learning models, making it suitable for a wide range of applications beyond image classification, such as natural language processing, reinforcement learning, and computer vision.

Uses in Drowsiness Detection Project:

1. Neural Network Development:- TensorFlow can be used to design and implement neural network architectures for drowsiness detection. Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) may be suitable for image or time-series data, respectively.

2. Model Training:- TensorFlow facilitates the training of machine learning models using optimization algorithms. You can feed labeled datasets into your model to enable it to learn patterns indicative of drowsiness.

3. Integration with Computer Vision Libraries:- TensorFlow can be integrated with computer vision libraries like OpenCV for image processing tasks. This is useful for tasks such as face detection and tracking in drowsiness detection projects.

4. Real-time Inference:- Once a model is trained, TensorFlow allows for the deployment of models for real-time inference, enabling the detection of drowsiness in live video streams.

5. Transfer Learning:- Transfer learning, a technique where a pre-trained model is adapted to a new task, can be easily implemented using TensorFlow. This can be useful when working with limited labeled data.

6. TensorFlow Lite for Edge Devices:- TensorFlow Lite allows for the deployment of models on edge devices, making it suitable for implementing drowsiness detection systems on devices with limited computational resources.

7. Model Evaluation and Fine-Tuning:- TensorFlow provides tools for evaluating the performance of trained models and fine-tuning them to improve accuracy and robustness.

TensorFlow is a powerful library for developing and deploying machine learning models, and its versatility makes it suitable for a variety of applications, including drowsiness detection in real-world scenarios.

5.4 NUMPY

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code

useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. NumPy is licensed under the BSD liscence, enabling reuse with few restrictions.

SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.[3]

SciPy is also a family of conferences for users and developers of these tools: SciPy (in the United States), EuroSciPy (in Europe) and SciPy.in (in India).

The SciPy library is currently distributed under the BSD license, and its development is sponsored and supported by an open community of developers. It is also supported by Numfocus which is a community foundation for supporting reproducible and accessible science.

5.5 KERAS

Keras is an open-source high-level neural networks API written in Python. It acts as an interface for building, training, and deploying deep learning models, providing a user-friendly and modular approach. Keras supports various backends, with TensorFlow being the most commonly used backend.

Simplified Model Development:

Keras simplifies the process of designing neural network architectures. It provides a high-level abstraction that allows developers to define models using concise and intuitive code, making it easier to experiment with different architectures.

Ease of Prototyping:

Drowsiness detection models often require experimentation with different architectures and configurations. Keras's user-friendly interface facilitates rapid prototyping, allowing developers to iterate quickly and find the optimal model for the task.

Integration with TensorFlow:

Keras is integrated with TensorFlow, a popular deep learning framework. This integration combines the simplicity of Keras with the computational efficiency of TensorFlow, providing a powerful toolset for building and training complex models.

Flexibility in Model Design:

Drowsiness detection may involve using various neural network architectures, such as convolutional neural networks (CNNs) for image processing. Keras allows developers

to flexibly design and customize models to suit the specific requirements of the drowsiness detection task.

Community Support:

Keras has a large and active community of developers and researchers. This community support translates into extensive documentation, tutorials, and a wealth of resources. Developers working on drowsiness detection can benefit from the shared knowledge and solutions within the community.

Compatibility with Image Processing Libraries:

Drowsiness detection often involves image processing tasks. Keras can be easily integrated with popular computer vision libraries like OpenCV, facilitating the handling and processing of image data within the drowsiness detection workflow.

Transfer Learning Capabilities:

Keras supports transfer learning, allowing developers to leverage pre-trained models on large datasets. This is advantageous in drowsiness detection, where obtaining a large labeled dataset may be challenging. Transfer learning helps to improve model performance with limited labeled data.

Keras serves as a valuable tool in drowsiness detection by providing a high-level and user-friendly interface for developing, training, and deploying deep learning models.

5.6 OS

OS in Python refers to the operating system module, which provides a way to interact with the underlying operating system that Python is running on. The os module is versatile and can be used for various tasks related to file and directory manipulation, environment variables, and more.

1. File and Directory Operations:

The drowsiness detection project may involve managing files and directories, such as saving or loading trained models, handling datasets, or creating logs. The os module provides functions like os.path.join(), os.listdir(), and os.makedirs() to work with paths and directories efficiently.

2. System Calls and Commands:

Drowsiness detection applications might need to interact with the operating system for system-specific tasks. The os.system() function allows execution of system commands,

which can be useful for tasks like launching external processes or managing system resources.

3. Environment Variables:

The os.environ dictionary in the os module provides access to environment variables. This can be beneficial for configuring the drowsiness detection application based on environmental settings, such as adjusting paths or behavior depending on the system.

4. Platform Information:

The os module allows you to retrieve information about the underlying operating system using functions like os.name and os.uname() or platform.system(). This can be useful for adapting the drowsiness detection code to be platform-independent.

Here's a simple example of how the os module might be used in drowsiness detection:

```
python
import os

# Example: Create a directory to store model checkpoints
checkpoint_dir = 'model_checkpoints'
os.makedirs(checkpoint_dir, exist_ok=True)

# Example: Execute a system command
os.system('echo "Drowsiness detection system starting..."')

# Example: Access environment variables
data_path = os.environ.get('DROWSINESS_DATA_PATH', '/default/data/path')

# Example: Get platform information
platform_info = os.uname() if os.name == 'posix' else os.name
print(f"Platform Information: {platform_info}")
```

OS module is a versatile tool for interacting with the operating system, and its use in drowsiness detection can range from managing files and directories to executing system commands and adapting the application based on the underlying operating system's characteristics.

5.7 RANDOM

In Python, the random module is used to generate pseudo-random numbers. It provides functions for generating random numbers, selecting random elements from sequences, and shuffling sequences.

In the context of drowsiness detection, randomness might not be directly related. Drowsiness detection typically involves analyzing facial features, eye movements, or other physiological signals to determine if a person is becoming drowsy. While randomness itself may not be a direct tool for this task, machine learning algorithms or statistical models can be employed, and randomization might play a role in data preprocessing or model training for better generalization.

If you have a specific use case in mind within drowsiness detection where randomness is mentioned, please provide more details for a more accurate explanation

5.8 PYGAME

Pygame is a set of Python modules designed for writing video games. It includes computer graphics and sound libraries. While Pygame itself is not typically used for drowsiness detection, it can be utilized for creating graphical interfaces or visualizations if you're building a drowsiness detection system with a graphical user interface (GUI).

For drowsiness detection, you would more likely use computer vision libraries such as OpenCV to analyze video feeds from cameras, detect facial features, and monitor signs of drowsiness like eye movements. Pygame might come into play if you want to display the video feed or detection results in a window with graphical elements.

If you have a specific use case or scenario in mind, feel free to provide more details so I can offer more targeted guidance.

5.9 TIME MODULE

The time module in Python provides various time-related functions, including measuring time intervals, formatting dates, and working with timestamps. In the context of drowsiness detection, the time module might be used to track the duration of certain events or to set up intervals for checking the occurrence of specific conditions.

For example, you might use time to measure the time elapsed between blinks or eye movements in a drowsiness detection system. By tracking time intervals, you can analyze patterns and identify deviations that may indicate drowsiness.

5.10 SQLite

SQLite is an in-process library that implements a self-contained, zero-configuration, serverless, transactional SQL database engine. The source code for SQLite exists in the public domain and is free for both private and commercial purposes.

SQLite has bindings to several programming languages such as C, C++, BASIC, C#, Python, Java and Delphi. The COM (ActiveX) wrapper makes SQLite accessible to scripted languages on Windows such as VB Script and JavaScript, thus adding capabilities to HTML applications. It is also available in embedded operating systems such as iOS, Android, Symbian OS, Maemo, Blackberry and WebOS because of its small size and ease of use.

SQLite is atomicity, consistency, isolation, durability (ACID) compliant. This embedded relational database management system is contained in a small C programming library and is an integral part of client-based applications. SQLite uses a dynamic SQL syntax and performs multitasking to do reads and writes at the same time. The reads and writes are done directly to ordinary disk files.An SQLite library is called dynamically and application programs use SQLite functionality through simple function calls, reducing latency in database access. These programs store entire databases as single cross-platform files on host machines. This simple design is implemented by locking the entire database file during a write SQLite implements the SQL-92 standard for SQL and uses an unusual system for SQL compatible database management systems. Types are assigned to individual values, adding flexibility to columns when bound to dynamic scripting languages. Full unicode support in SQLIte .

6. RESULT S AND FUTUTRE SCOPE

In this proposed system method for detecting driver drowsiness is implemented, where face is detected after that calculation of the eye map. For any input image it will check the eye by comparing the eye ratio of query image with the stored image mouth map in dataset. If the eye ratio is less than zero face it will generate the sound alert otherwise no alert for blinking of eyes. The aim to build this system is to provide a product which will assist driver while driving such that though driver is drowsy we can alert the driver at earlier stage to avoid further mishaps. 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 driver's alertness level on the basis of eye behavior.

FUTURE SCOPE

As we know there is always a scope for further improvement same thing will be applicable over here. There is vast scope for this system of drowsiness detection. As there are several signs by which we can say the person is feeling drowsy. To detect the drowsiness several alternatives are available like eye detection, Iris detection, pupil detection, and yawing detection. Out of these options proposed system uses eye detection for drowsiness detection. For making the system more strong and efficient one can check the two different behavior together so chances of false identification may reduce. This system can be introduced in an organization like security system, Toll collection counters, importantly at check post.

7. SUMMARY AND CONCLUSION

Our drowsiness detector hinged on two important computer vision techniques:

- Facial landmark detection
- Eye aspect ratio

Facial landmark prediction is the process of localizing key facial structures on a face, including the eyes, eyebrows, nose, mouth, and jawline.

Specifically, in the context of drowsiness detection, we only needed the eye regions.

Once we have our eye regions, we can apply the eye aspect ratio to determine if the eyes are closed. If the eyes have been closed for a sufficiently long enough period of time, we can assume the user is at risk of falling asleep and sound an alarm to grab their attention.

8. REFERENCES

1. https://www.w3schools.com/python/
2. www.realpython.com
3. https://www.scipy.org/
4. http://pythongeek.com

9. APPENDIX A(SCREENSHOTS)

10. APPENDIX B (CODE)

#Drowsiness Code

```
import cv2
import os
from keras.models import load_model
import numpy as np
from pygame import mixer
import time
from geopy.geocoders import Nominatim
from twilio.rest import Client



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

face = cv2.CascadeClassifier('Trash\haarcascade_frontalface_alt.xml')
leye = cv2.CascadeClassifier('Trash\haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('Trash\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=(150,150))
left_eye = leye.detectMultiScale(gray)
right_eye =  reye.detectMultiScale(gray)

cv2.rectangle(frame, (0,height-50) , (450,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)
    predict_x = model.predict(r_eye)
    rpred = np.argmax(predict_x,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)
    predict_y = model.predict(l_eye)
    lpred = np.argmax(predict_y,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:
    if (score>20):
        score=score-5
        cv2.putText(frame,"Open",(10,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)
    else:
        score=score-2
        cv2.putText(frame,"Open",(10,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)


if(score<0):
    score=0
cv2.putText(frame,'Risk%:'+str(score),(100,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)
cv2.putText(frame,'TroubleShooters',(220,height-20), font, 1,(255,0,0),1,cv2.LINE_AA)
if(score>8):
    #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<2):
        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 score > 100 driver response 0
if (score>50):
    #location scrapping
    geoloc=Nominatim(user_agent="GetLoc")
```

```
        loc_name=geoloc.reverse("23.3158,85.3740")
        locations=loc_name.address
        #location conversion
        #sms sending
        account_sid = 'AC623a70940a5970405c8a8950107961f4'
        auth_token = '9ab4b8bdf0774f03019621f056c859c2'
        client = Client(account_sid, auth_token)
        message = client.messages.create(
            from_='+16206789260',
            body =locations,
            to ='+916204864673'
        )
        print(message.sid)
        break
        #sms sending ends
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

#Model Code

```
import os
from keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
from keras.utils.np_utils import to_categorical
import random,shutil
from keras.models import Sequential
from keras.layers import Dropout,Conv2D,Flatten,Dense, MaxPooling2D, BatchNormalization
from keras.models import load_model


def generator(dir, gen=image.ImageDataGenerator(rescale=1./255),
shuffle=True,batch_size=1,target_size=(24,24),class_mode='categorical' ):
```

```
    return
gen.flow_from_directory(dir,batch_size=batch_size,shuffle=shuffle,color_mode='grayscale',class_mode=
class_mode,target_size=target_size)

BS= 32
TS=(24,24)
train_batch= generator('data/train',shuffle=True, batch_size=BS,target_size=TS)
valid_batch= generator('data/valid',shuffle=True, batch_size=BS,target_size=TS)
SPE= len(train_batch.classes)//BS
VS = len(valid_batch.classes)//BS
print(SPE,VS)


# img,labels= next(train_batch)
# print(img.shape)

model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(24,24,1)),
    MaxPooling2D(pool_size=(1,1)),
    Conv2D(32,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(1,1)),
#32 convolution filters used each of size 3x3
#again
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(1,1)),

#64 convolution filters used each of size 3x3
#choose the best features via pooling

#randomly turn neurons on and off to improve convergence
    Dropout(0.25),
#flatten since too many dimensions, we only want a classification output
    Flatten(),
#fully connected to get all relevant data
    Dense(128, activation='relu'),
#one more dropout for convergence' sake :)
    Dropout(0.5),
#output a softmax to squash the matrix into output probabilities
    Dense(2, activation='softmax')
])
```

```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

model.fit_generator(train_batch, validation_data=valid_batch,epochs=15,steps_per_epoch=SPE
,validation_steps=VS)

model.save('models/cnnCat2.h5', overwrite=True)
```

#Matrix Code

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from keras.models import load_model
import cv2,os

face = cv2.CascadeClassifier('Trash\haarcascade_frontalface_alt.xml')
leye = cv2.CascadeClassifier('Trash\haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('Trash\haarcascade_righteye_2splits.xml')

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=(150,150))
    left_eye = leye.detectMultiScale(gray)
    right_eye =  reye.detectMultiScale(gray)
```

```
    cv2.rectangle(frame, (0,height-50) , (450,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)
        predict_x = model.predict(r_eye)
        rpred = np.argmax(predict_x,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)
        predict_y = model.predict(l_eye)
        lpred = np.argmax(predict_y,axis=1)
        if(lpred[0]==1):
            lbl='Open'
        if(lpred[0]==0):
            lbl='Closed'
        break

# Example ground truth and predicted labels

r_eye = np.expand_dims(r_eye,axis=0)
```

```
rpred = np.argmax(predict_x,axis=1)

# Create confusion matrix
cm = confusion_matrix(r_eye,rpred)

# Define class labels
lbl=['Close','Open']

# Plot confusion matrix
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(lbl))
plt.xticks(tick_marks, lbl)
plt.yticks(tick_marks, lbl)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')

# Add counts to the confusion matrix plot
for i in range(len(lbl)):
    for j in range(len(lbl)):
        plt.text(j, i, str(cm[i][j]), ha='center', va='center')

plt.show()
```