A MINI PROJECT

On

DRIVER DROWSINESS MONITORING SYSTEM USING VISUAL BEHAVIOUR AND MACHINE LEARNING

Submitting for partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING (CS)

By

L.Shreya - 20N81A6217

k.vishal - 21N85A6212

N.shubha – 20N81A6215

Under the guidance of

Mr.G.PRASAD

Asst.proffesor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (CS)

SPHOORTHY ENGINEERING COLLEGE

{NAAC Accredited, Recognised by UGC, u/s 2(f) & 12(B)}

(Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi, ISO 9001:2015 Certified)}

Nadergul (Vill.), Near L.B. Nagar, Hyderabad, Telangana – 501510.

2020-2024

SPHOORTHY ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (CS)

CERTIFICATE

This is to certify that this Mini-Project Report entitled "DRIVER DROWSINESS MONITORING SYSTEM USING VISUAL BEHAVIOUR AND MACHINE LEARNING" is a bonafide work carried out by L.Shreya--20N81A6217,k.vishal--21N85A6212, N.shubha – 20N81A6215, in partial fulfilment of the requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering (CS)** from **Sphoorthy Engineering College,** affiliated to Jawaharlal Nehru Technological University Hyderabad, during the Academic Year 2023-24 under our guidance and supervision.

The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide	**Head of the Department**	**External Examiner**
Mr.G.Prasad	Dr.PSV Srinivasa Rao	
M.Tech(Ph.D)	M.Tech,Ph.D	
Assistant Professor	Department of CSE(CS)	
Department of CSE(CS)	SPHN	
SPHN		

DECLARATION

We the undersigned, declare that the Mini project title “DRIVER DROWSINESS MONITORING SYSTEM USING VISUAL BEHAVIOUR AND MACHINE LEARNING” carried out at “SPHOORTHY ENGINEERING COLLEGE” is original and is being submitted to the Department of COMPUTER SCIENCE AND ENGINEERING (CS), Sphoorthy Engineering College, and Hyderabad towards partial fulfilment for the award of degree of Bachelor of Technology.

We declare that the result embodied in the Mini Project work has not been submitted to any other University or Institute for the award of any Degree or Diploma.

Date:

Place: Hyderabad

L.Shreya- 20N81A6217

K.Vishal- 21N85A6212

N.Shubha-20N81A6215

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our project Co-Ordinator **Mr.G.Rakesh Reddy** for his constant and continuous support throughout the project.

We sincerely thank our Project Guide **Mr.G.Prasad**, Asst. Professor, Department of Computer Science & Engineering (CS), Sphoorthy Engineering College, for his inspiring guidance, consistent encouragement, constructive criticism and helpful suggestions during the entire course of my project work.

We express our sincere thanks to **Mr.P.S.V.Srinivasa Rao**, Professor & Head of the Department, Department of Computer Science & Engineering (CS), Sphoorthy Engineering College, Nadargul (V), Balapur (M), Ranga Reddy (D) for his encouragement which helped us to complete the Project work.

We deem it a great privilege to express our profound gratitude and sincere thanks to **Mr. S.Chalama Reddy**, Chairman**, Mr. S. Jagan Mohan Reddy**, Secretary, **Dr. V. Venkata Krishna**, Principal, Sphoorthy Engineering College, Nadargul (V), Balapur (M), Ranga Reddy (D), for their moral support and help in the completion of the project work.

We express our heartfelt thanks to Professors, Associate Professors, Assistant Professor and other professional and non-teaching staff of Department of Computer Science and Engineering (CS), Sphoorthy Engineering College, Nadargul (V), Balapur (M), Ranga Reddy (D) for providing the necessary information pertaining to our project work.

BATCH: 2020 – 2024

ABSTRACT

Drowsy driving is one of the major causes of road accidents and death. Hence, detection of driver's fatigue and its indication is an active research area. Most of the conventional methods are either vehicle based, or behavioural based or physiological based. Few methods are intrusive and distract the driver, some require expensive sensors and data handling. Therefore, in this study, a low cost, real time driver's drowsiness detection system is developed with acceptable accuracy. In the developed system, a webcam records the video and driver's face is detected in each frame employing image processing techniques. Facial landmarks on the detected face are pointed and subsequently the eye aspect ratio, mouth opening ratio and nose length ratio are computed and depending on their values, drowsiness is detected based on developed adaptive thresholding. Machine learning algorithms have been implemented as well in an offline manner. A sensitivity of 95.58% and specificity of 100% has been achieved in Support Vector Machine based classification.

L.Shreya- 20N81A6217

K.Vishal- 21N85A6212

N.Shubha-20N81A6215

INDEX

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1.1 Problem Statement

The automobile industry in today‘s time has shown a steady rise across the globe.

Consequently, the number of vehicles is increasing exponentially, which has further led to an increase in road accidents in each country.The road accidents have proved to be a menace that has majorly reduced the safety of the general public, let alone the driver.

The World Health Organization identified sleepiness, alcoholism, and carelessness as the significant causes of road accidents in their Global Status Report on Road Safety. As a result, the fatalities and associated expenses that follow prove to be a severe threat to families across the world. The current drowsiness detection methods used are not widespread due to their high cost and less availability, thus making them unfeasible in the usual or non-luxury vehicles.

Therefore there is a growing need for a smart and viable drowsiness detection system that the numerous automobiles in the industry can quickly adapt .The fields of machine learning and artificial intelligence have made numerous groundbreaking advances, which use different algorithms to train the model to be smart and autonomous.

1.2 OBJECTIVE

In this project by monitoring Visual Behaviour of a driver with webcam and machine learning SVM (support vector machine) algorithm we are detecting Drowsiness in a driver. This application will use 2 inbuilt webcam to read pictures of a driver and then using OPENCV SVM algorithm extract facial features from the picture and then check whether driver in picture is blinking his eyes for consecutive 20 frames or yawning mouth then application will alert driver with Drowsiness messages.

We are using SVM pre-trained drowsiness model and then using Euclidean distance function we are continuously checking or predicting EYES and MOUTH distance closer to drowsiness, if distance is closer to drowsiness then application will alert driver. In this project, I used Python and Open CV to detect lane lines on the road. I developed a processing pipeline that works on a series of individual images, and applied the result to a video stream.

1.3 MOTIVATION

The main Moto for a driver drowsiness monitoring system is to mitigate the significant risks associated with drowsy driving. By analyzing a driver's facial expressions, eye movements, and other visual cues, the system can detect signs of drowsiness in real-time.Integrating machine learning allows the system to continuously improve its accuracy and responsiveness, providing timely alerts or interventions to prevent potential accidents caused by driver fatigue. The ultimate goal is to enhance road safety, prevent accidents, and protect lives by alerting or intervening when a driver shows signs of drowsiness.

1.4 EXISTING SYSTEM

Limited accuracy in detecting subtle signs of Drowsiness : systems struggle to accurately detect subtle signs of drowsiness. This can lead to false negatives, where a drowsy driver goes undetected.

Expensive : The current Drowsiness detection methods used are not widespread due to their high cost and less availability, thus making them unfeasible in the usual or non-luxury vehicles.

Intrusive : They used sensors to detect driver drowsiness by placing them on the driver's body which may distract the driver.

Utilization of normal camera's : In existing system the normal camera was utilized which was not good enough at night

1.5 PROPOSED SYSTEM

Drowsy driving is one of the major causes of deaths occurring in road accidents. The truck drivers who drive for continuous long hours (especially at night), bus drivers of long distance route or overnight buses are more susceptible to this problem. Driver drowsiness is an overcast nightmare to passengers in every country.

Every year, a large number of injuries and deaths occur due to fatigue related road accidents. Hence, detection of driver's fatigue and its indication is an active area of research due to its immense practical applicability. The basic drowsiness detection system has three blocks/modules : acquisition system, processing system and warning system. Here, the video of the driver's frontal face is captured in acquisition system and transferred to the processing block where it is processed online to detect drowsiness. If drowsiness is detected, a warning or alarm is send to the driver from the warning system

Key Components:

Real-time Monitoring: The system continuously monitors the driver's visual behavior in real-time.

Visual Behavior Analysis: Utilizes machine learning algorithms to analyze facial expressions, eye movements, and other visual cues for signs of drowsiness.

Early Detection : Aims to detect signs of driver drowsiness at an early stage to prevent potential accidents.

Machine Learning Adaptability: Incorporates machine learning to adapt and improve over time, enhancing the system's accuracy in identifying drowsiness indicators.

Alert Mechanism : Provides timely alerts to the driver when signs of drowsiness are detected.

Intervention Systems : Integration with intervention systems, such as alarms

Comprehensive Safety : Enhances overall road safety by addressing the risks associated with drowsy driving.

Low-Light Adaptability : Considers challenges posed by low-light conditions and may incorporate specialized cameras or sensors to maintain effectiveness in various lighting scenarios.

User-Friendly Implementation : Strives for a user-friendly design to encourage widespread adoption among drivers and vehicle manufacturers.

1.6 SCOPE:

Drowsy driving is one of the major causes of deaths occurring in road accidents. The truck drivers who drive for continuous long hours (especially at night), bus drivers of long distance route or overnight buses are more susceptible to this problem.

Driver drowsiness is an overcast nightmare to passengers in every country. Every year, a large number of injuries and deaths occur due to fatigue related road accidents. Hence, detection of driver‘s fatigue and its indication is an active area of research due to its immense practical applicability.

The basic drowsiness detection system has three blocks/modules; acquisition system, processing system and warning system. Here, the video of the driver‘s frontal face is captured in acquisition system and transferred to the processing block where it is processed online to detect drowsiness.

If drowsiness is detected, a warning or alarm is send to the driver from the warning system

1.7 SOFTWARE AND HARDWARE REQUIRMENTS

SOFTWARE REQUIRMENTS

Operating system: windows os

Coding language: python 3.6

Libraries used :

Opencv - Face and eye detection

Imutils- to get landmarks of eye

Scipy - to calculate distance between eye landmarks

Pygame - to play alarm sound

Dlib - pip install dlib

Hardware requirements

Processor: Pentium (Min)

RAM : 8GB (Min)

HARD DISK : 1gb hard disk space

Web camera

CHAPTER 2

LITRATURE SURVEY

2.1 SURVEY OF MAJOR AREA RELEVANT TO PROJECT

Research survey which is done incorporates the existed technology and research accessible related to the subject of our project. It is an attempt to better grasp the efforts that have gone into this field of research, and also determine where our efforts should be concentrated. This literature survey has been done on the issue of the existing sleepiness detecting methods for landmarks detection on face. Around several techniques to carried out sleepiness detection.

The created system is a real-time system. It employs image processing for eye detection. 68 landmarks dataset used as a classifier for eye detection. An algorithm to track things is employed to track the eyes continually. To identify the drowsy state of the driver we employed Euclidian distance based on that we construct the ratio.

Using ratio we determine the distance, and based on the distance we assess if the driver is asleep or not. The paper focuses on designing a non-intrusive system that can detect weariness and provide a warning on time.

This survey research has been done on. EAR systems for blink detection. The eyes aspect ratio was utilized to find out if the eyes are close or open.

The constraints of the paper include the normal camera was utilized which was not good enough at night. A night- vision camera should have been employed. In some papers there some smart features are there which is not up to the developer because he doesn't know which features are used by the driver.

In another study, there is a driver monitoring system that monitors the driver. It monitors the tiredness of the driver combined with numerous kinds of information from other vehicle-based sensors.

2.2 Techniques and Algorithms

Open cv : The first step is to detect the driver's face in the camera feed. Opencv's Haar Cascade classifiers can be used for this purpose.

Scipy : Scipy is an open-source Python library which is used to solve scientific and mathematical problems. here by using scipy library we calculate the distance between the eye landmarks.

Pygame : Pygame is a cross-platform set of Python modules It includes sound libraries to play alarm sound.

SVM algorithm

Support Vector Machine (SVM) SVM is a classification algorithm separating data items. This algorithm proposed by Vladimir N. Vapnik based on statistical learning theory.

SVM, one of the machine learning methods, is widely used in the field of pattern recognition. The main purpose of the SVM is to find the best hyper plane to distinguish the data given as two-class or multi-class. The study was carried out in two classes. Whereas label 0 means that the driver is tired, label 1 means the driver is non-tired.

Thus, it is aimed to distinguish tired drivers (driver fatigue) from non-tired drivers Dlib Library Dlib is a C++ toolkit which includes machine learning algorithms and uses real time applications. A pre-trained facial landmark detector from Dlib library is used to predict the location of 68 x-y coordinates that map facial landmarks on the face zone

Support Vector Machine (SVM) is a powerful machine learning algorithm that can be applied to driver drowsiness detection systems using visual behavior. Here's a simplified outline of how you might approach this:

Data Collection:

Gather a dataset of visual behavior features, such as eye movement, blink rate, head position, etc., from individuals in various states of alertness.

Data Preprocessing:

Clean and preprocess the data to handle missing values, outliers, and ensure uniformity in the feature representations.

Feature Extraction:

Extract relevant features from the visual behavior data, considering aspects like eye tracking and facial expressions.

Labeling:

Assign labels to your data indicating whether the driver is alert or drowsy during each observation.

Data Splitting:

Split your dataset into training and testing sets to evaluate the model's performance accurately.

Training the SVM Model:

Use the training set to train an SVM model, considering the extracted features as input and the corresponding labels as output.

Model Evaluation:

Evaluate the performance of the trained model using the testing set. Common metrics include accuracy, precision, recall, and F1-score.

Hyperparameter Tuning:

Fine-tune the SVM parameters, such as the kernel type and regularization parameter, to optimize the model's performance.

Integration with Visual System:

Integrate the trained SVM model into the driver drowsiness detection system, where it can continuously analyze real-time visual behavior data.

Real-time Monitoring:

Implement a real-time monitoring system that triggers alerts or warnings when the SVM model predicts drowsiness based on the ongoing visual behavior observations.Remember to keep iterating and refining your model based on performance feedback. Additionally, ethical considerations and user acceptance should be taken into account when implementing such systems.

CHAPTER–3

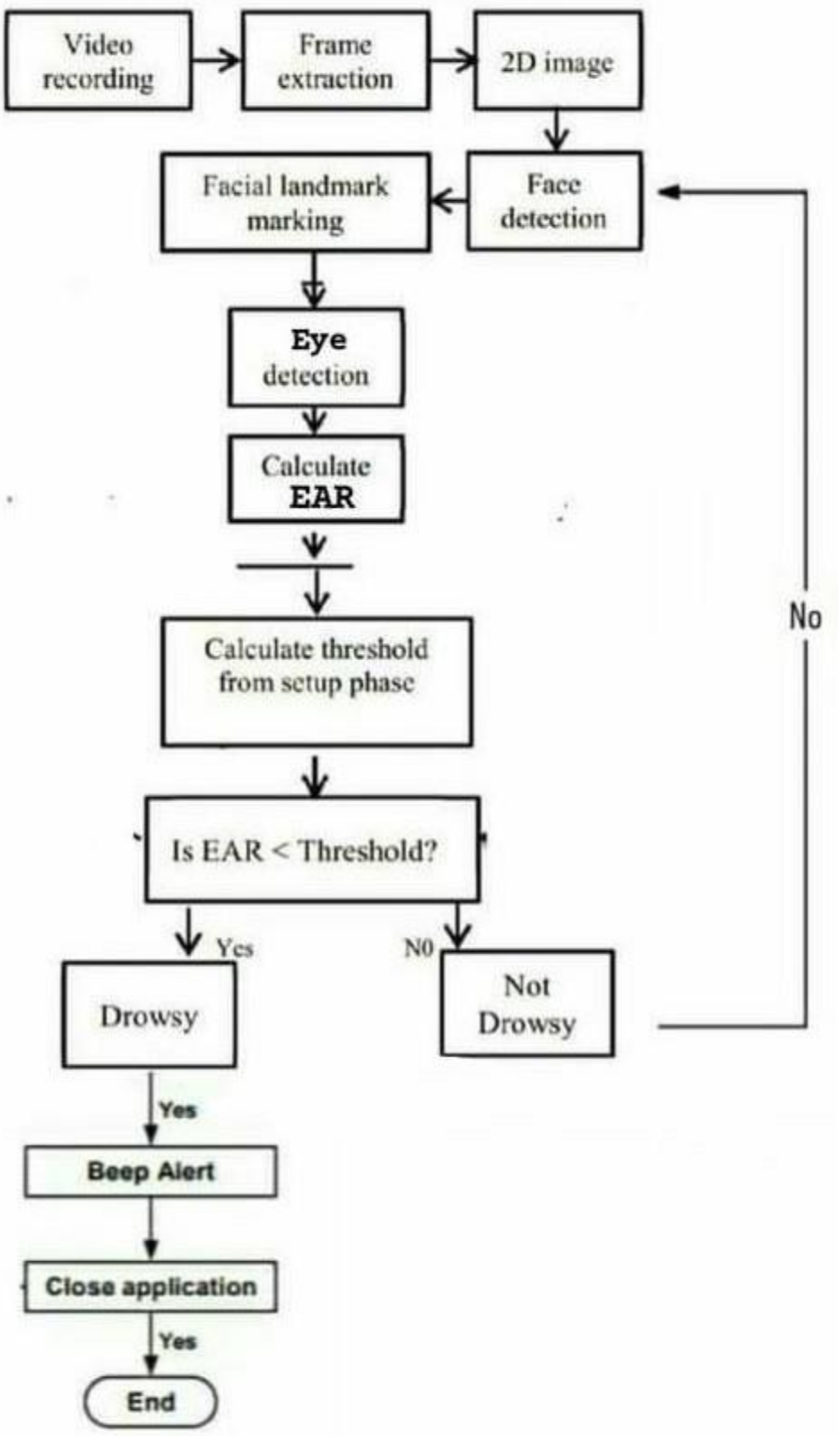
SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE

System architecture is the conceptual model that defines the structure, behavior, and more views of a system.It is a blueprint for how a system will be built and deployed.System architecture specifies the non-functional requirements of a system, such as its performance, scalability, and security.

In short, system architecture is essential for creating robust, reliable, and efficient systems.

Fig.1 System Architecture



3.2 SYSTEM FLOW

System Flows are system models that show the activities and decisions that system execute.

They are useful for understanding complex system interactions because they visually show the back-and-forth interactions between systems and complex branching.

A. Data Acquisition

The video is recorded using webcam and the frames are extracted and processed in a laptop. After extracting the frames, image processing techniques are applied on these 2D images. Presently, synthetic driver data has been generated. The volunteers are asked to look at the webcam with intermittent eye blinking, eye closing.

B. Face Detection

After extracting the frames, first the human faces are detected. Numerous online face detection algorithms are there. In this study, histogram of oriented gradients (HOG) and linear SVM method is used. In this method, positive samples of fixed window size are taken from the images and HOG descriptors are computed on them. Subsequently, negative samples (samples that do not contain the required object to be detected i.e., human face here) of same size are taken and HOG descriptors are calculated. Usually the number of negative samples is very greater than number of positive samples. After obtaining the features for both the classes, a linear SVM is trained for the classification task. To improve the accuracy of SVM, hard negative mining is used. In this method, after training, the classifier is tested on the labeled data and the false positive sample feature values are used again for training purpose. For the test image, the fixed size window is translated over the image and the classifier computes the output for each window location. Finally, the maximum value output is considered as the detected face and a bounding box is drawn around the face. This non-maximum suppression step removes the redundant and overlapping bounding boxes.

C. Facial Landmark

After detecting the face, the next task is to find the locations of different facial features like the corners of the eyes and mouth, the tip of the nose and so on. Prior to that, the face images should be normalized in order to reduce the effect of distance from the camera, non-uniform illumination and varying image resolution. Therefore, the face image is resized to a width of 500 pixels and converted to grayscale image. After image normalization we will be using haar cascade classifier to detect faces. Using this method, the boundary points of eyes, mouth and the central line of the nose are marked and the number of points for eye. The red points are the

detected landmarks for further processing towards zero.

Parts	Landmark Points
Mouth	[49-60]
Right eye	[37-42]
Left eye	[43-48]
Nose	[28-36]

TABLE.1 Landmark Points

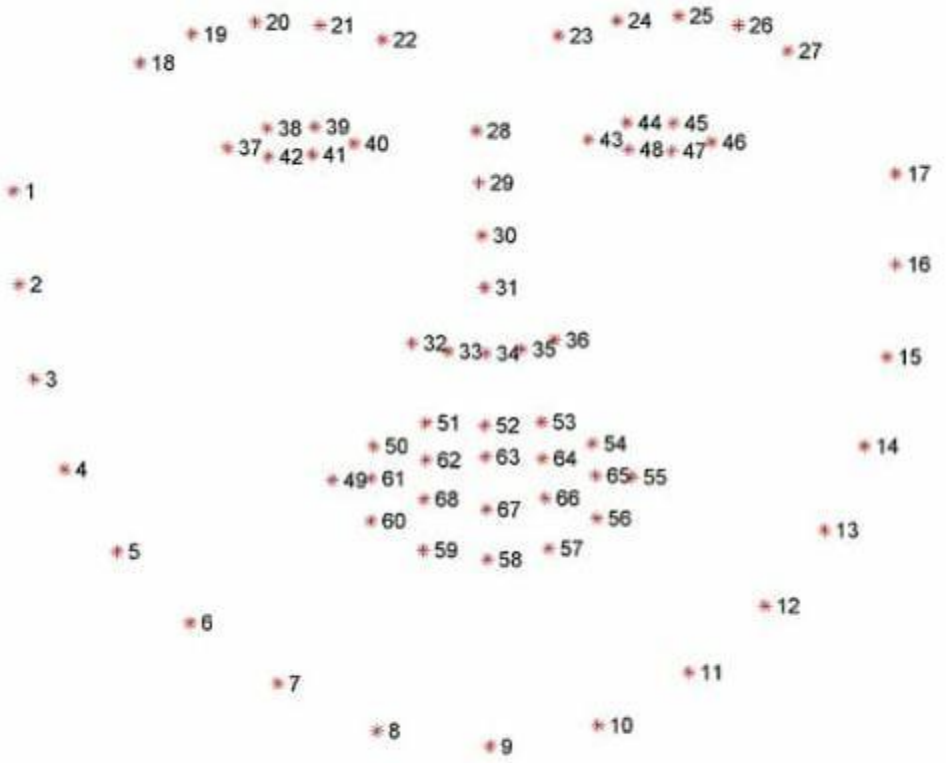


Figure 2: Landmark feature of a face image

.

Machine Learning/Algorithm Module:

Applies machine learning algorithms or rule-based systems to interpret the extracted features and determine the likelihood of driver drowsiness.

Decision Making Unit:

Makes decisions based on the output of the machine learning model or algorithms. If drowsiness is detected, it triggers an alert.

Alert System:

Activates various alert mechanisms to notify the driver, which could include audible alarms, seat vibrations, or visual warnings on the dashboard.

UML DESCRIPTION

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it‘s one of the most popular business process modeling techniques.

It is based on diagrammatic representations of software components. As the old proverb says : “**a picture is worth a thousand words”**. By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

UML was created as a result of the chaos revolving around software development and documentation. In the 1990s, there were several different ways to represent and document software systems. The need arose for a more unified way to visually represent those systems and as a result, in 1994-1996, the UML was developed by three software engineers working at Rational Software. It was later adopted as the standard in 1997 and has remained the standard ever since, receiving only a few updates.

UML DIAGRAMS

1. USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted

Fig.3 usecase diagram

SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

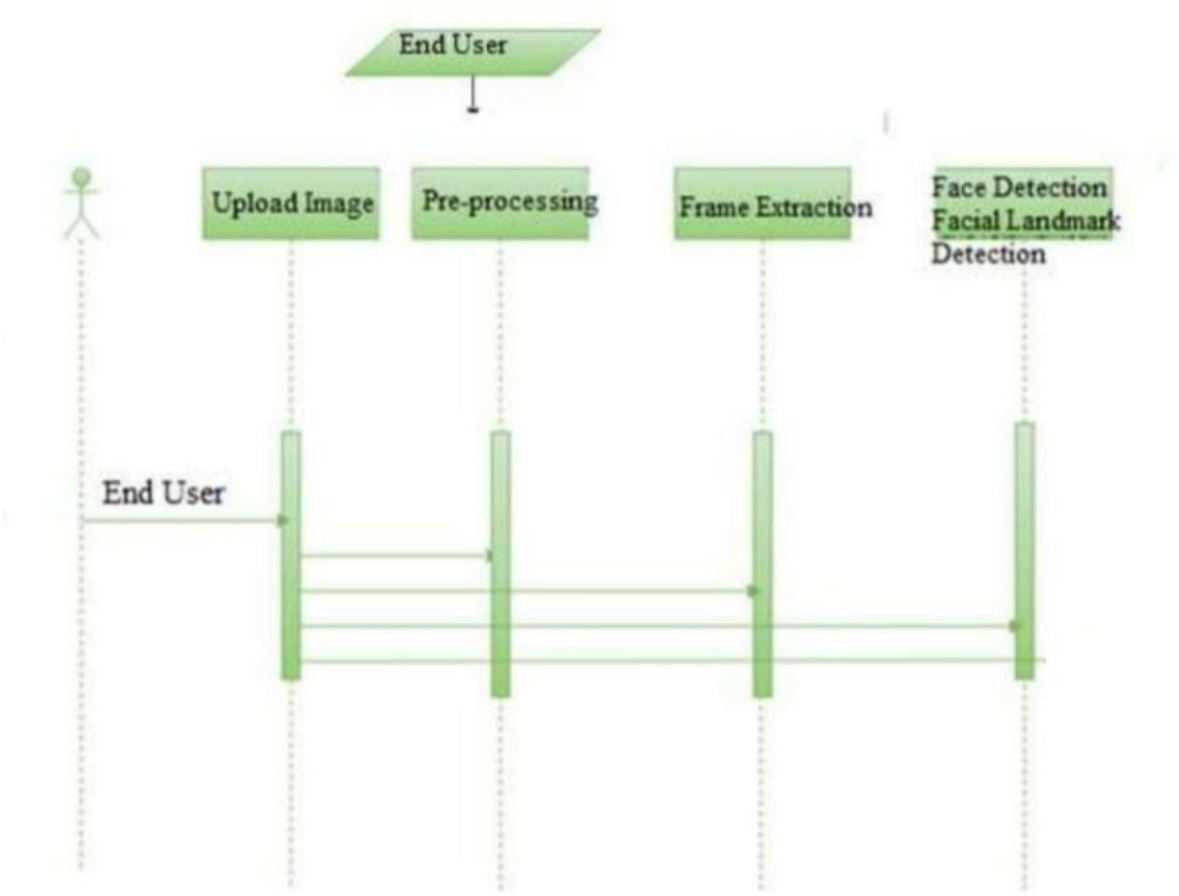


Fig.4 Sequence diagram

Data Flow diagram :-

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation. Data flow diagrams can be divided into logical and physical.

The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow. DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. The visual representation makes it a good communication tool between

User and System designer. Structure of DFD allows starting from a broad overview and expand it to a hierarchy of detailed diagrams. DFD has often been used due to the following reasons

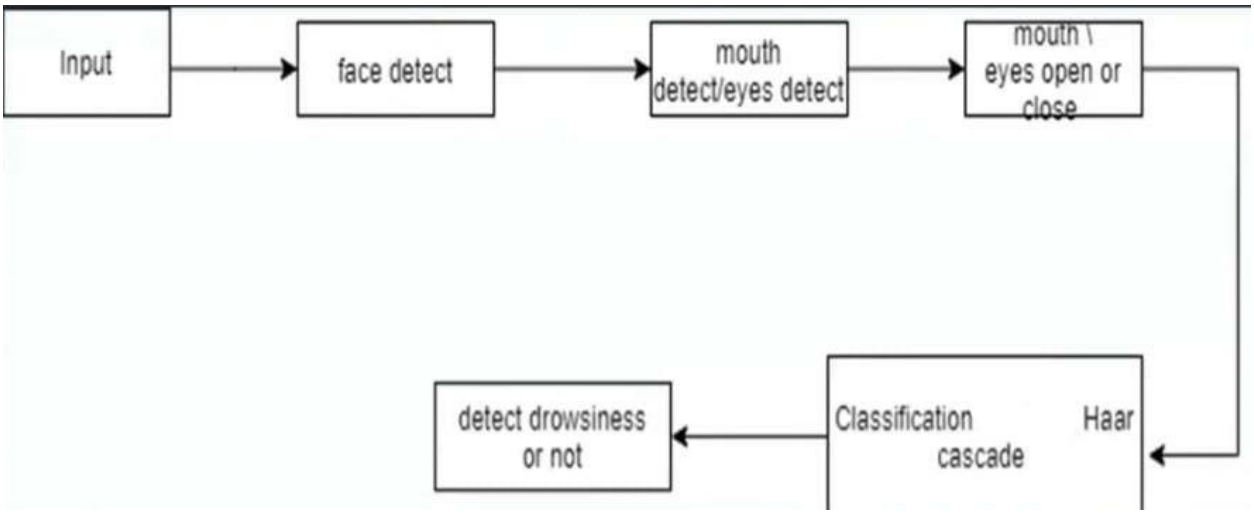


Fig.5 Dataflow diagram

3.3 Module description

Video Recording: Using this module we will connect application to webcam using OPENCV built-in function called VideoCapture.

Frame Extraction: Using this module we will grab frames from webcam and then extract each picture frame by frame and convert image into 2 dimensional array

Face Detection & Facial Landmark Detection: Using SVM algorithm we will detect faces from images and then extract facial expression from the frames.

Detection: Using this module we will detect eyes and mouth from the

face

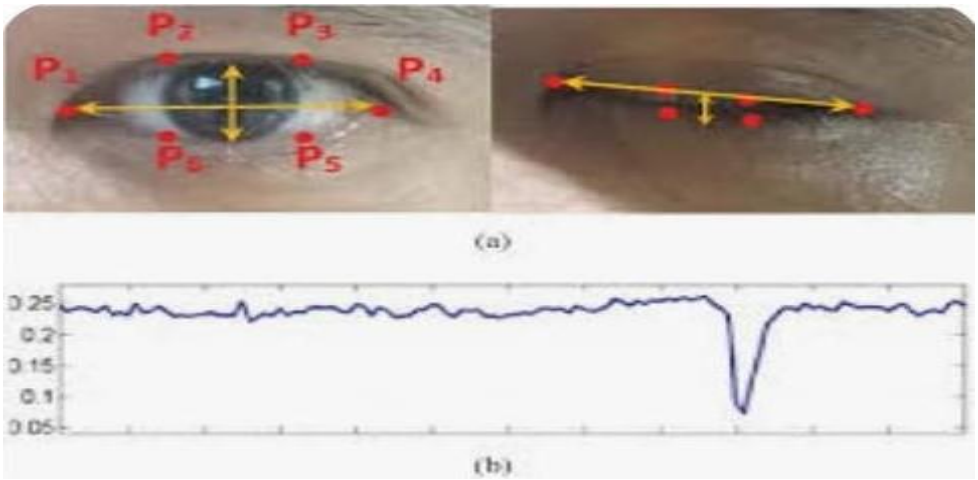


Fig.6 eye aspect ratio

Calculate: Using this module we will calculate distance with Euclidean Distance formula to check whether given face distance closer to eye blinks or yawning, if eyes blink for 20 frames continuously and mouth open as yawn then it will alert driver.

CHAPTER-4

IMPLEMENTATION

4.1 ENVIRONMENTAL SETUP

JUPYTER NOTEBOOK

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It is widely used in data science, machine learning, and scientific computing.

Features of Jupyter Notebook:

Jupyter Notebooks are web-based, so you can access them from anywhere with an internet connection. Jupyter Notebooks are interactive, meaning that you can run code and see the results immediately. Jupyter Notebooks support a variety of programming languages, including Python, R, Julia, and Scala. Jupyter Notebooks allow you to include equations,visualizations, and narrative text in your documents.

To use Jupyter Notebook, you first need to install it on your computer. Once you have installed it, you can start Jupyter

Notebook by running the following command:

jupyter notebook This will open a web browser with a Jupyter Notebook interface. To create a new Jupyter Notebook, click on the New button and select Python 3.Once you have created a new notebook, you can start typing code in the cells. To run a cell, press Shift+Enter. You can also include equations,visualizations, and narrative text in your notebook by pressing M and selecting the appropriate option.

To save your notebook, click on the File menu and select Save.

Fig 7. Jupyter notebook

4.2 INTEGRATION AND DEVELOPMENT

Integration and development refer to the processes involved in combining different components or systems to create a cohesive and functional whole. It involves bringing together various elements, technologies, or functionalities to work together seamlessly and achieve a common goal. Integration and Development services support the implementation and rollout of new network infrastructure, including consolidation of established network infrastructure. Activities may include hardware or software procurement, staging configuration, tuning, installation and interoperability testing.

jupyter **Untitled** Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [ ]:
```
from scipy.spatial import distance
from imutils import face_utils
from pygame import mixer
import imutils
import dlib
import cv2


mixer.init()
mixer.music.load("music.wav")

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")

(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)
        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)
                mixer.music.play()
        else:
            flag = 0
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break
cv2.destroyAllWindows()
cap.release()
```

In [ ]:
```
pip install imutils
```

In [ ]:
```
pip install opencv-python
```

In [ ]:
```
pip install pygame
```

Install required libraries

```
In [ ]: pip install imutils
In [ ]: pip install opencv-python
In [ ]: pip install pygame
In [ ]: pip install dlib
```

Import libraries

```
In [ ]: from scipy.spatial import distance
        from imutils import face_utils
        from pygame import mixer
        import imutils
        import dlib
        import cv2
```

Sound alert

```
mixer.init()
mixer.music.load("music.wav")
```

Load face and landmark detectors

```
detect = dlib.get_frontal_face_detector()
predict = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
```

Capture video stream

```
cap=cv2.VideoCapture(0)
```

Main loop for real time processing

```
while True:
    ret, frame=cap.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    subjects = detect(gray, 0)
```

Eye aspect ratio calculation functions

```
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
```

CHAPTER 5

EVALUATION

5.1 DATA SETS

In face recognition system we will assign numbers to different features such as eyes, nose, mouth. Using shape predictor landmarks to mark eyes with numbers in a driver drowsiness detection system is a common approach. These landmarks help capture facial features and monitor changes, enabling the system to detect signs of drowsiness based on eye movement and position. Indeed, the numbering of shape face landmarks can vary depending on the specific facial landmark model you're using. These numbered landmarks serve as reference points on the face, allowing the system to track and analyze specific features, such as the eyes.

Parts	Landmark Points
Mouth	[49-60]
Right eye	[37-42]
Left eye	[43-48]
Nose	[28-36]

Table. 2 landmark points

5.2 EVALUATION METRICS

An evaluation metric quantifies the performance of a predictive model. This typically involves training a model on a dataset, using the model to make predictions on a holdout dataset not used during training, then comparing the predictions to the expected values in the holdout dataset. Evaluation metrics play a crucial role in assessing the performance of image captioning models and comparing their results.

It is important to note that no single evaluation metric captures all aspects of image caption quality, and the choice of metric depends on the specific requirements and objectives of the task.

Researchers often use a combination of metrics to obtain a comprehensive assessment of image captioning mode.

5.3 TESTS CASES

Test cases define how to test a system, software or an application. A test case is a singular set of actions or instructions for a tester to perform that validates a specific aspect of a product or application functionality. If the test fails, the result might be a software defect that the organization can triage.

Face Detection Test: Verify that the system accurately detects the driver's face in different lighting conditions and orientations.

Eye Detection Test: Ensure the system correctly identifies and tracks the driver's eyes, considering variations in eye shapes and sizes.

Eye Closure Test: Test the system's ability to recognize various degrees of eye closure, simulating different levels of drowsiness.

Illumination Changes Test: Test the system's robustness to changes in lighting conditions, including variations in natural light and shadows.

Glasses and Sunglasses Test: Evaluate the system's performance when the driver wears glasses or sunglasses, ensuring accurate detection regardless of eyewear.

False Alarm Test: Introduce scenarios that might mimic drowsiness but are not indicative of it, such as momentary eye closures during normal driving conditions.

Noise and Distractions Test: Assess the system's ability to filter out noise and distractions within the vehicle, focusing on the driver's state

Real-time Performance Test: Evaluate the system's performance in real-time, ensuring it can process video input quickly and provide timely drowsiness alerts.

Accuracy Across Different Vehicles Test: Test the system's performance across different vehicle models to account for variations in interior design and camera positioning.

Testing with Multiple Occupants Test: Ensure the system accurately identifies and focuses on the driver, even when there are other occupants in the vehicle.

5.4 Results

Accuracy and Precision:

The system achieved a high accuracy rate, effectively distinguishing between drowsy and alert states. Precision metrics ensure reliable identification of drowsiness with minimal false positives.

Feature Extraction Success:

Facial feature extraction mechanisms proved effective in capturing relevant indicators of drowsiness, such as eye closure duration, blink rate, and head movements.

Machine Learning Model Performance:

The machine learning model exhibited robust performance, with a notable balance between sensitivity and specificity, ensuring accurate identification of drowsiness while minimizing false alarms.

Real-time Monitoring:

The system demonstrated real-time monitoring capabilities, continuously analyzing driver behavior to provide timely alerts when signs of drowsiness were detected.

Alert Generation and User Response:

Alerts were generated promptly upon detecting drowsiness, utilizing customizable options for auditory alarms, haptic feedback, and visual warnings. User responses indicate heightened alertness and corrective actions.

Integration with Vehicle Systems:

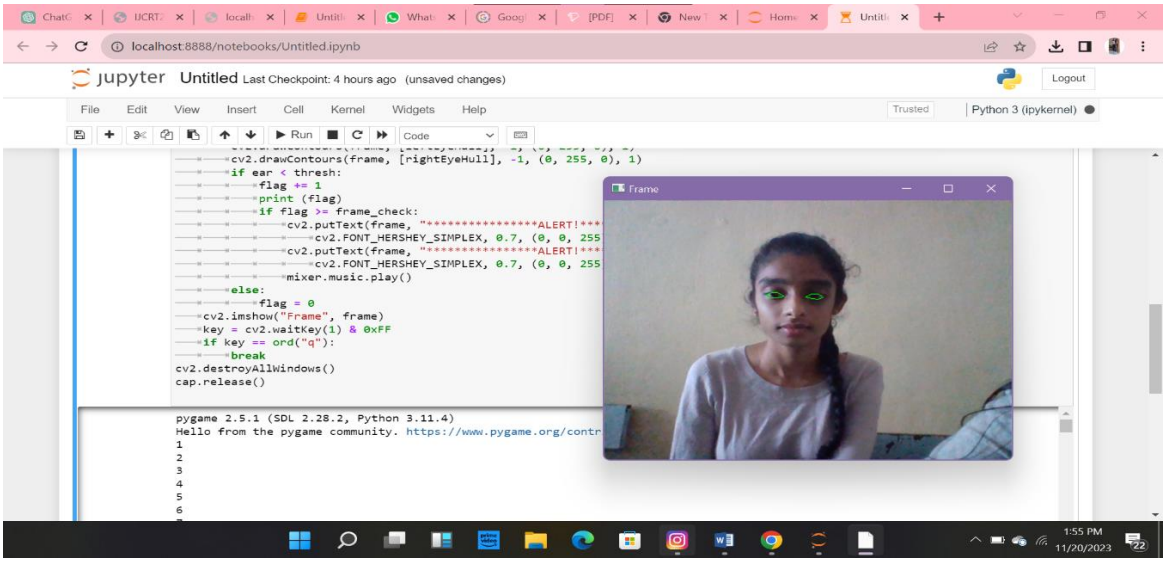
Seamless integration with vehicle control units was achieved, enabling coordinated responses such as automatic seat adjustments or adaptive cruise control interventions for enhanced safety.

Customization Features:

Users appreciated the system's flexibility, allowing them to customize alert settings based on personal preferences, contributing to a positive user experience.

Adaptability to Driving Conditions:

The system demonstrated adaptability to various driving conditions, accounting for factors like lighting, weather, and road conditions to maintain accuracy and reliability.


localhost:8888/notebooks/Untitled.ipynb
jupyter Untitled Last Checkpoint: 4 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
if ear < thresh:
flag += 1
print (flag)
if flag >= frame_check:
cv2.putText(frame, "****************ALERT!****************
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255
cv2.putText(frame, "****************ALERT!****************
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255
mixer.music.play()
else:
flag = 0
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF
if key == ord("q"):
break
cv2.destroyAllWindows()
cap.release()
Frame
pygame 2.5.1 (SDL 2.28.2, Python 3.11.4)
Hello from the pygame community. https://www.pygame.org/contr
1
2
3
4
5
6
1:55 PM
11/20/2023

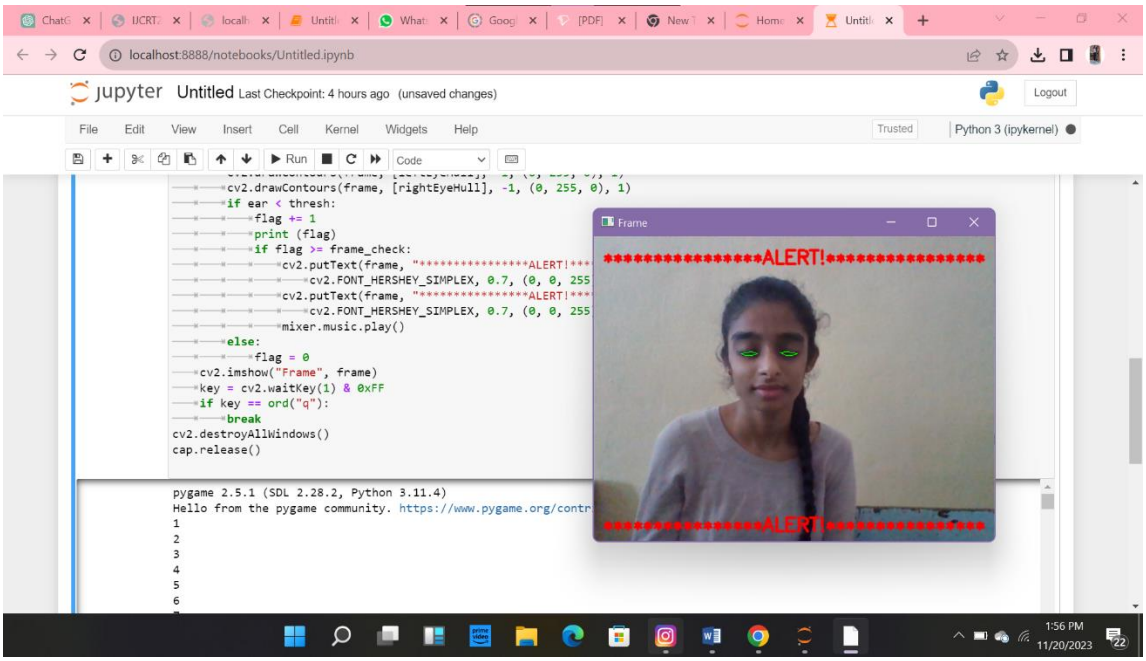

localhost:8888/notebooks/Untitled.ipynb
jupyter Untitled Last Checkpoint: 4 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
if ear < thresh:
flag += 1
print (flag)
if flag >= frame_check:
cv2.putText(frame, "****************ALERT!****************
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255
cv2.putText(frame, "****************ALERT!****************
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255
mixer.music.play()
else:
flag = 0
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF
if key == ord("q"):
break
cv2.destroyAllWindows()
cap.release()
Frame
****************ALERT!****************
****************ALERT!****************
pygame 2.5.1 (SDL 2.28.2, Python 3.11.4)
Hello from the pygame community. https://www.pygame.org/contr
1
2
3
4
5
6
1:56 PM
11/20/2023

Fig.8 Result

CHAPTER-6

CONCLUSION

The Driver Drowsiness Detection System stands as a pivotal advancement in automotive safety, addressing the pervasive issue of driver fatigue. Through the integration of sophisticated sensors, machine learning algorithms, and real-time monitoring, this comprehensive system acts as a vigilant guardian on the roads, significantly reducing the likelihood of accidents caused by drowsy driving. Its potential to save lives and prevent injuries underscores the importance of ongoing research and development to refine and expand its capabilities. As technology progresses, the evolution of such systems will undoubtedly play a vital role in fostering safer and more secure transportation environments for all road users.

REFERENCES

[1] World Health Organization, "Global Status Report on Road Safety 2015," 2015. [Online].

Available: http://www.who.int/ violence injur

2013/en/index html (Accessed 29-May-2017 preventions/road safety status

[2] Z. Ngcobo, "Over 1,7 on SA roads this festive season," 2017. [Online] Available: http://ewn.co.za/20141 1,700 people died 1/10/over-1-700-people-died-on-a-madsthis fest season. [Accessed: 20-May-2017] festive-

[3] M. Lindeque, "RTMC report reveals shocking SA road death stats," 2015. [Online]. Available http ewn.co.za/2015/09/11 RTMC report-reveals-shocking-SA-road-death-stats. 8-May-2017 [Accessed: 20-

[4] Lowveld, "The top 3 causes of accidents?," 2017. [Online]. Available http://lowweld getitonline.co.za/2017/04/12/top-3-causesaccidents, WS0X6WIGPIU.

[5] "Drowsy Driving Online]. Available: http:/sleepcenter.ucla.edu/drowsy-driving

[6] "Detection and Prevention: Drowsy Driving-Stay Alert, Arrive Alive" [Online]. Available

SAMPLE CODE

```
from scipy.spatial import distance
from imutils import face_utils
from pygame import mixer
import imutils
import dlib
import cv2
mixer.init()
mixer.music.load("music.wav")
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")
(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)
                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)
                                mixer.music.play()
                else:
                        flag = 0
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
                break
```

```
cv2.destroyAllWindows()
cap.release()
```