SHAROMIRA KARIMI

SC201/1149/2021

PROJECT PROPOSAL

A NEURAL VISION BASED SYSTEM FOR DRIVER'S SAFETY RISK MONITORING AND PREDICTION

DECLARATION

Declaration

I, Sharomira Karimi, hereby declare that the project titled *"A Neural Vision-Based System for Driver's Safety Risk Monitoring and Prediction"* is my original work. This report is submitted in partial fulfillment of the requirements for Bachelor of Science in Computer Technology at Murang'a University of technology.

All information, data, and work presented herein are the result of my own efforts, except where explicitly acknowledged. Any contributions, assistance, or external sources have been cited appropriately to maintain academic integrity.

I confirm that this project has not been previously submitted, either wholly or partially, for any degree, diploma, or qualification at this institution or any other.

Sign date

………………………………………………… ………………………………………

Full name

…………………………………………………………………………

Registration number

…………………………………………………………

Supervisor

Millicent Kathambi sign

…………………………………………

DEDICATION

I dedicate this report to my dear friends, whose unwavering support, encouragement, and companionship have been a source of strength throughout this journey. Your belief in me has been invaluable, and I am truly grateful for the joy and motivation you bring into my life.

I also dedicate this work to my school, Murang'a University Of Technology, for providing a nurturing environment that fosters learning, growth, and innovation. The resources, guidance, and opportunities I have received here have been instrumental in shaping my academic and personal development.

To all who have contributed to this journey, directly or indirectly, I extend my heartfelt appreciation. This achievement is as much yours as it is mine.

ACKNOWLEDEMENT

First, I would like to express my profound gratitude to Almighty God for His guidance, wisdom, and strength throughout this project. His blessings have been the cornerstone of my perseverance and success.

I am deeply thankful to my supervisor, Madam Millicent, for their invaluable guidance, constructive feedback, and unwavering support throughout this journey. Their expertise and encouragement have been instrumental in shaping this project and bringing it to fruition.

Finally, I extend my heartfelt appreciation to my parents, whose love, prayers, and sacrifices have always been a source of inspiration. Their unwavering belief in my potential has provided me with the confidence and motivation to overcome every challenge.

To all who have contributed in one way or another to this work, I am sincerely grateful.

Abstract

Road safety is a critical issue, with driver errors contributing to over 85% of traffic accidents. This project presents a neural vision-based system designed to enhance road safety by monitoring and predicting driver risks in real time. Using advanced computer vision and neural network techniques, the system analyzes driver behavior, vehicle dynamics, and environmental factors. Key features include fatigue detection through facial analysis, distraction recognition via gesture tracking, and predictive algorithms for hazard assessment.

The system aims to proactively reduce accidents caused by speeding, dangerous overtaking, impaired driving, and improper road use. By providing valuable insights into driver behavior and potential risks, it offers a robust solution for industries seeking to improve driver monitoring and accident prevention strategies. This innovative approach has the potential to significantly curb traffic accidents and contribute to safer transportation systems worldwide.

TABLE OF CONTENTS

Catalog

TABLE OF FIGURES

LIST OF TABLES

ABBREVIATIONS

CPU-	Central Processing Unit
GPU-	Graphical Processing Unit
OS-	Operating Systems
ADAS-	Advanced driver assistance systems
IVMS-	In Vehicle Monitoring System
RDBMS-	relational database management system
SQL-	structured query language
TPU-	Tensor Processing Unit
MSV-	Microsoft Visual C++
IDE-	Integrated Development Environment

CHAPTER ONE

1.1 BACKGROUND

Road safety is a critical concern worldwide, with traffic accidents remaining one of the leading causes of injury and death. Human factors such as fatigue, distraction, and risky driving behaviors significantly contribute to these incidents. Traditional approaches to monitoring driver safety, including manual observation and basic in-car sensors, offer limited insights and may fail to respond proactively to risky situations. To enhance safety, there is a need for more intelligent and proactive systems capable of assessing and predicting safety risks in real-time.

With advancements in machine learning, particularly in neural network and computer vision technologies, there is potential to build systems that can monitor driver behavior and detect early signs of risk. A neural vision-based system leverages deep learning models to interpret video and sensor data, enabling the system to detect dangerous behaviors like drowsiness, aggressive driving, or distractions by analyzing visual cues from the driver and the vehicle's surroundings.

This system, focusing on driver's safety risk monitoring and prediction, integrates real-time visual analysis with predictive capabilities to issue timely warnings or take preventive measures. By enhancing driver safety through predictive monitoring, this neural vision-based system can contribute significantly to reducing road accidents and improving overall driving safety.

1.2 PROBLEM STATEMENT

There are many factors behind road traffic crashes, but human error is the greatest, with over 85% of crashes caused by errors, such as speeding, dangerous overtaking, driving whilst drunk and poor use of the road. In order to curb the accidents, I want to develop a safety risk monitoring and prediction system the uses neural vision to scan for any human practice or behavior that might cause an accident. The behaviors include drowsiness, driving while drunk or driving while being on a phone call. If the system detects any of the malpractices, it warns the driver.

1.3 OBJECTIVES

1.3.1 GENERAL OBJECTIVE

To develop a neural vision based system for driver's safety risk and prediction

1.3.2 SPECIFIC OBJECTIVES

1. To develop a machine learning software that can detect changes in driver's facial expressions and behavior and predict if he is fit for driving
2. To develops a software that can notify the driver whenever their attention drifts from the road and notify them when they need a break
3. To test the developed system

1.4 PURPOSE, SCOPE AND APPLICABILITY

1.4.1 PURPOSE

The project, a neural vision system for driving safety risk prediction using driver's behavior is supposed to reduce the accidents caused by human error by providing a warning every time the driver's attention drifts away from the road. The hardware components required in the Driver Monitoring System are, sensors, CPU, GPU, OS and other devices that work together to assess the driver's condition. Cameras can detect eye movements, blink rates, and head position**.** Sensors monitor where the driver is looking, helping to identify distractions or lapses in attention. CPU executes instructions from programs through a cycle of fetching, decoding, and executing. GPU highly
efficient for tasks that require simultaneous processing of multiple data points, such as image and video processing.

1.4.2 SCOPE

The development methodology used to develop the system is waterfall methodology. This is because it allows flexibility and adaptability of the driver monitoring system. The main issue I am covering in my project is curbing occurrence of road accidents. It is designed to detect practices that might cause an accident e.g. drunkenness, driving while on a phone call, the driver sticking his head outside the window and drowsiness- caused by boredom or fatigue. The system should notify the driver to keep their eyes open or keep their eyes on the road depending on the kind of driving malpractice they have done. The key functions of my project are - Driver identification and authentication, fatigue and drowsiness detection, distraction monitoring, emotion and stress analysis and gaze tracking and attention monitoring

1.4.3 APPLICABILITY

1.4.3.1 DIRECT APPLICATION

The system will be used in driver monitoring**-** real-time tracking of the driver's behavior such as speed, braking and idling, Compliance management**-** ensuring adherence to regulations and safety standards and Safety improvements**-** reducing the risk of accidents by monitoring and providing feedback on driver performance

1.4.3.2 Indirect application

The system will also help in monitoring **Operational efficiency.** Leading to improved driver behavior leads to better fuel efficiency and reduced vehicle wear and tear.

CHAPTER TWO

2.1 INTRODUCTION

In order to develop the neural vision based system, technologies to use are chosen based on machine learning capabilities, real-time data handling and scalability. Several programming languages and databases such as python, **c++**, JavaScript, Matlab, rust, MongoDB, PostgreSQL, influx DB, redis, Azure cosmos DB and java, meet the requirements to build such a system.

2.2 EXISTING SYSTEMS

[1] Advanced driver assistance systems (ADAS) focusing on "driver behavior" has been developed. Generally, an ADAS can include a set of active safety systems that work together to increase the safety of driver, passengers, pedestrians and other road users. The objective is to recognize critical driving situations by perception of the vehicle and the divers as internal parameters, road hazards as external parameters, and less highlighted by other researchers- the weather and lighting condition, as what we call circumferential parameters. Other systems are [2] Neonode which Uses AI MultiSensing technology to provide custom driver monitoring solutions. It can detect driver distraction, drowsiness, gaze direction, and hand positioning. [3] Samsara a fleet management system that offers driver management capabilities, including custom reports and alerts. It also has real-time tracking and automatic route optimization. [4]Azuga a telematics vehicle-tracking provider that also offers dashcams and eLogs. Driver management software, part of an IVMS, this software helps fleet managers improve driver and asset safety. It can measure driver performance and provide coaching tools. Automotive sensors used in conjunction with driver alertness systems, these sensors monitor driving habits and alert operators if they exhibit concerning behaviors.

2.3 EXISTING SOFTWARE DEVELOPMENT TOOLS

There are several software development tools and frameworks used in building neural vision-based driver safety risk monitoring and prediction systems. These tools help in image and video processing, deep learning model development, real-time data streaming, and analysis. Examples of these tools are python, **c++**, JavaScript, Matlab, rust, MongoDB, PostgreSQL. In addition, there are deep learning frameworks particularly for computer vision tasks like object detection, facial recognition, and behavior analysis such as tensor flow, pytorch and OpenCV.

Python

Python is a high-level, interpreted, and versatile programming language known for its simplicity and readability. It is used for data analysis, machine learning, web development, automation, scientific computing, and scripting. It is easy-to-read syntax, large standard library, extensive third-party libraries (e.g., NumPy, Pandas, TensorFlow).

C++

It is a high-performance, compiled programming language that extends the C language with object-oriented programming features. It is used for game development, system programming, real-time systems, embedded systems, and high-performance applications. It has high speed, memory control via pointers, and support for procedural and object-oriented programming paradigms.

JavaScript

Is a lightweight, interpreted language primarily used for creating interactive web applications. Runs in web browsers as well as on servers (e.g., Node.js). it can be used for web development (front-end and back-end), creating dynamic web pages, and building single-page applications. it is event-driven programming, asynchronous capabilities, and integration with HTML and CSS.

MATLAB

This is a proprietary programming language and environment designed for numerical computing and visualization. It is used in engineering simulations, data analysis, signal processing, control systems, and machine learning. It has built-in libraries for mathematical computations, easy visualization tools, and a specialized environment for technical computing.

Rust

A modern, high-performance programming language that prioritizes memory safety without a garbage collector. It is used in Systems programming, embedded systems, web assembly, and applications where performance and safety are critical. It has memory safety guarantees, concurrency support, and strong performance similar to C++.

MongoDB

this is a NoSQL database that stores data in flexible, JSON-like documents. It is best used in applications requiring high scalability, flexibility, and working with semi-structured or unstructured data. Common in web development and big data applications. It has schema-less design, horizontal scaling, and support for rich queries.

PostgreSQL

This is an advanced, open-source relational RDBMS that supports SQL and advanced features like JSON. It is used in applications requiring complex queries, data analytics, and transactional integrity. Often used in enterprise-level projects. It has strong ACID compliance, extensibility (e.g., custom functions), and support for both structured (SQL) and semi-structured (JSON) data.

2.4 CONCLUSION

In this case, I choose python as the programming language. Python is the best choice for machine learning, computer vision and neural network development due to its extensive libraries and frameworks. Popular libraries in python are tensor flow, Opencv which is used for image processing and PyTorch for neural network development and real-time inference. Since my project will involve predicting driver's safety using neural vision, python's libraries will be of great importance in ensuring that I can capture images, process them and provide a real-time feedback on whether the driver is at risk of getting an accident.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 INTRODUCTION TO RESEARCH METHODOLOGY

The main issue we are addressing is the unsafe driving behaviors that lead to accidents and endanger road safety. These behaviors include over speeding, dangerous overtaking, driving while intoxicated, and poor use of the road. The issue can be divided into several sub-problems such as driver drowsiness detection, which involves identifying when a driver is becoming drowsy or falling asleep at the wheel. Distraction Detection which helps in detecting when a driver is distracted by activities such as using a mobile phone, eating, or looking away from the road. Facial Expression Analysis, which involves analyzing the driver's facial expressions to assess their emotional state and stress levels. Over Speeding, which is driving at speeds higher than the legal limit or safe for the current road conditions. Driving Whilst Drunk, operating a vehicle under the influence of alcohol or drugs, impairs the driver's judgment and reaction times. Poor Use of the Road-failing to adhere to road rules, such as improper lane usage, not signaling, or ignoring traffic signs.

3.2 DATA COLLECTION METHODS

The methods I used to collect data are :- observation- I noticed that most drivers like driving while on a phone call, sticking their heads out of the window while driving and driving while they are drunk. This behavior has caused very many road accidents in the past years. In addition, I used online content analysis where I accessed several online documents that indicated that they found that fatal accidents [5]were most often caused by mid-speed ranges of 30–60 km/hr. The reason was due to the drivers' inattention or distracted behavior, which made them careless in the mid-speed range. The causative factors identified were minimal yet essential, such as staring at something outside the car.

3.3 SOFTWARE DEVELOPMENT METHODOLOGIES

3.3.1 WATERFALL METHODOLOGY

Waterfall is a linear and sequential approach where each phase must be completed before the next one begins and it is simple to understand and manage.

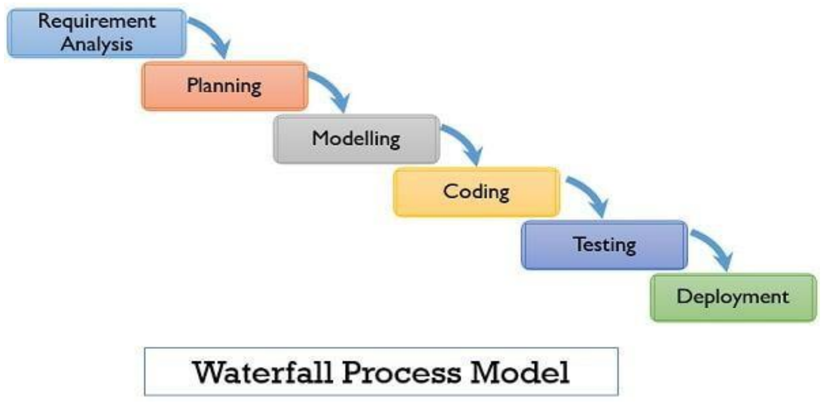


Figure 1

Advantages of Waterfall Model

- Waterfall model divides the entire process of software development into finite independent stages making controlling of each stage easier.
- Requirements are stable and known to the developer at the starting point of the project
- Only one stage is processed at a time thus avoiding confusion
- It's simple and easy to implement

3.4 SYSTEM REQUIREMENTS

3.4.1 HARDWARE REQUIREMENTS

The hardware required to develop the system are High-Resolution Cameras: For capturing clear images and videos of the driver. Graphics Card (GPU) NVIDIA GeForce RTX 3080 or higher**:** For handling the intensive computations required for neural network processing and real-time video analysis. Processor (CPU) Intel Core i7 or Core i5**:** To manage the overall system operations and support the GPU in processing tasks. Numeric Co-Processor like TPU**:** Optional, but beneficial for accelerating machine learning tasks. Memory (RAM), which is 8 or 16 GB DDR4, or higher**:** To ensure smooth multitasking and data processing. Storage 256GB SSD: For fast read/write speeds and sufficient storage for video data and software. Sensors : webcam**:** For eye-tracking and detecting driver alertness

3.4.2 Software requirement

Operating System such as Windows 10 or later**:** For compatibility with certain development environments and tools. Compiler such as Microsoft Visual C++**:** For compiling C/C++, code on Windows. Integrated Development Environment (IDE) like Visual Studio Code**:** A versatile and widely used code editor that supports multiple programming languages and extensions. Testing Tools such as webcam to enable you to view realtime detection. A testing framework for Python that allows you to write simple and scalable test cases. Linker such as Microsoft Linker: For linking compiled object files on Windows. Libraries like OpenCV: An open-source computer vision library that provides tools for image and video processing. TensorFlow: An open-source machine learning library for building and training neural networks and Git: A version control system for tracking changes in your codebase and collaborating with others.

3.5 SUMMARY

Developing a neural vision-based driver monitoring system addresses critical road safety concerns, including over speeding, dangerous overtaking, intoxicated driving, and poor road usage. Utilizing advanced hardware such as high-resolution cameras, GPUs, and sensors, combined with powerful software tools like OpenCV, TensorFlow, and Keras, enables real-time monitoring and analysis of driver behavior. The integration of machine learning and AI ensures accurate detection and timely alerts, enhancing road safety and reducing accidents, contributing to safer roads for all.

CHAPTER FOUR: System design, Implementation and Testing

4.1 INTRODUCTION

System design is the process of defining the architecture, product design, modules, interfaces, and data for a system to satisfy specified requirement

4.2 system design

The model is intended to monitor and predict driver's safety based on the behavioral practices of the driver such as yawning or using the phone while driving. To develop the model the following process of defining the architecture will be followed. The logical design of the neural vision-based road safety system provides an abstract representation of how data flows through different components of the system. It focuses on the structure, inputs, outputs, and processing steps without considering physical implementation details such as hardware or specific coding.

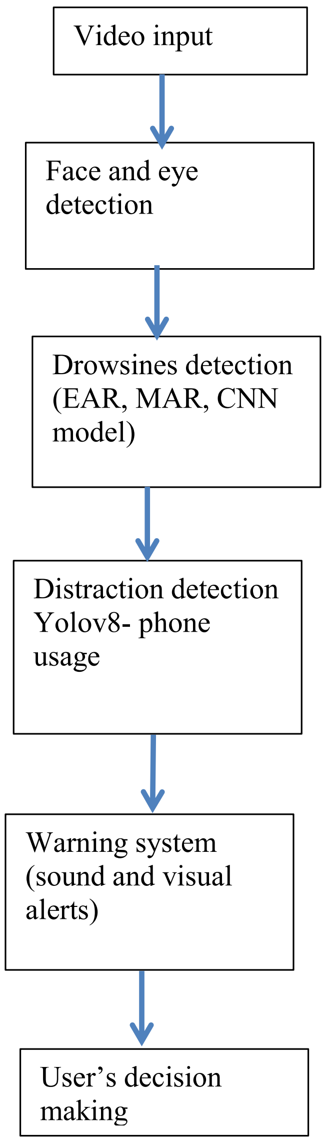

Video input
Face and eye detection
Drowsines detection (EAR, MAR, CNN model)
Distraction detection Yolov8- phone usage
Warning system (sound and visual alerts)
User's decision making

Figure 2

Drowsiness design

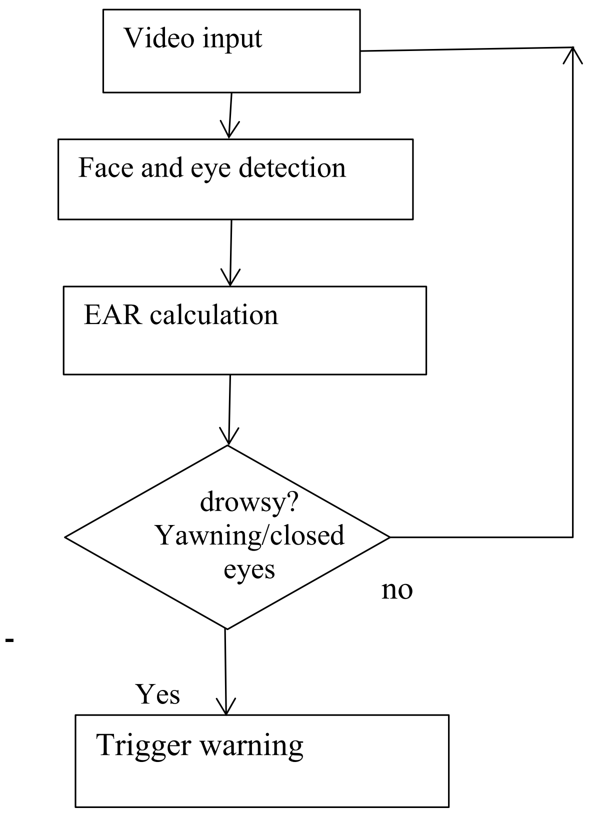


Figure 3

phone usage design

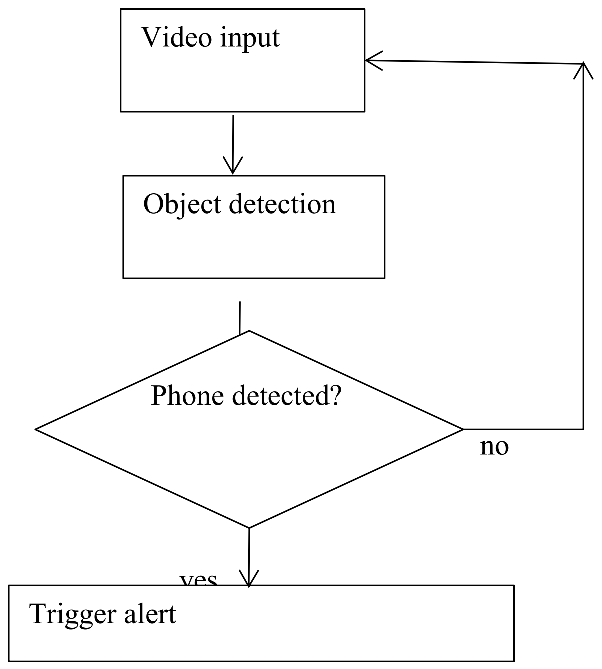


Figure 4

4.3 Implementation Approaches

The implementation of the **neural vision-based road safety system** follows a structured approach, ensuring efficient model performance, accurate detection, and real-time processing.

4.3.1 implementation plan

4.3.1.1 PHASE 1: DATASET COLLECTION & PREPROCESSING

The first thing I did was collect the dataset. The dataset used was Pre-existing datasets from Kaggle (e.g., drowsiness datasets).Custom dataset collection for Open_eyes and Closed_eyes categories.Face Detection was done using OpenCV's Haar Cascade. I used Eye Region Extraction to focus on key facial features. Then I Normalized and Resized images to match the input size of MobileNetV2. aftewards, I used Data Augmentation to improve model generalization.

4.3.1.2 PHASE 2: MODEL DEVELOPMENT & TRAINING

During training I Used MobileNetV2 for drowsiness detection due to its lightweight architecture and efficiency on CPUs and fine-tuned MobileNetV2 on the customized drowsiness dataset. The model's training conducted using TensorFlow/Keras, then I did Optimization using Adam optimizer and binary cross-entropy loss function. The evaluation metrics for Accuracy, Precision, Recall, and F1-score to assess model performance were Confusion Matrix for evaluating classification results.

4.3.1.4 PHASE 3: REAL-TIME DETECTION SYSTEM

Real time detection was done by capturing real-time video using OpenCV, detecting face and eyes using dlib's 68-point landmark model and predicting drowsiness based on extracted eye region (ROI).Phone Usage Detection was done usingYOLOv8 for better detection accuracy and faster inference then training YOLOv8 on a custom dataset with labeled phone usage scenarios. And moving average filter and mediapipe for mplementing head pose estimation and yawning detection

4.3.1.5 PHASE 4: ALERT SYSTEM & DEPLOYMENT

Real-Time Warnings are audio alerts (beeping sounds) and visual indicators (color-coded messages) Generated when drowsiness or distractions are detected.

Running the system locally on Windows OS.Using VS Code for development and debugging.Ensuring real-time processing on a CPU-based setup.

4.3.2 Standards Used in Implementation

To ensure **accuracy, efficiency, and reproducibility**, the following standards and best practices were followed:

4.3.2.1 MACHINE LEARNING AND DEEP LEARNING STANDARDS

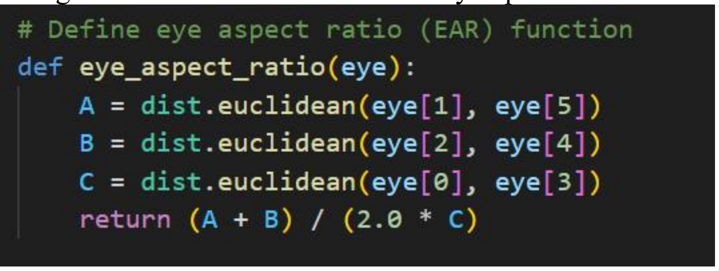
MobileNetV2 was chosen for its balance between speed and accuracy. YOLOv8 was selected for object detection improvements. Data augmentation was applied using TensorFlow's ImageDataGenerator. Image normalization ensured consistency in model input. Confusion Matrix for performance assessment.Precision, Recall, and F1-score to evaluate misclassifications.

4.3.2.2 SOFTWARE & CODE STANDARDS

PEP 8 coding standards followed for Python scripting. Modular programming for maintainability and scalability. Open CV for real-time image processing. Dlib for facial landmark detection.TensorFlow/Keras for deep learning. Model quantization techniques considered for CPU efficiency.Multi-threading in OpenCV to improve real-time processing.

4.4 Coding Details and Code Efficiency

Using euclidean distane to determine eye openness inorder to detect drowsiness



```
# Define eye aspect ratio (EAR) function
def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    return (A + B) / (2.0 * C)
```

Mouth Aspect Ratio (MAR) for Yawning Detection

```
# MAR threshold for yawning
MAR_THRESHOLD = 0.5
FRAME_THRESHOLD = 2  # Number of frames for yawn detection
```

While loop for starting drowsiness, phone and distraction detection process

```
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
```

Landmarks for detecting a drooping head to detect fatigue

```
# Get 2D facial points for head pose estimation
        image_points = np.array([
            (landmarks.part(30).x, landmarks.part(30).y),  # Nose tip
            (landmarks.part(8).x, landmarks.part(8).y),    # Chin
            (landmarks.part(36).x, landmarks.part(36).y),  # Left eye left corner
            (landmarks.part(45).x, landmarks.part(45).y),  # Right eye right corner
            (landmarks.part(48).x, landmarks.part(48).y),  # Left mouth corner
            (landmarks.part(54).x, landmarks.part(54).y)   # Right mouth corner
        ], dtype=np.float32)
```

4.5 Testing Approach

Testing the neural vision-based road safety system involved multiple stanges to ensure accuracy, efficiency and real-time performance.

4.5.1 State Machine-Based Testing

This test focuses on ensuring **correct state transitions**. It ensures that the alerts do not overlap and each feature is monitored independently

The system transitions between different **states**:

1. **Awake** → Normal state, no alerts.
2. **Drowsy** → When EAR is below the threshold for a given time.
3. **Yawning** → When MAR exceeds the yawning threshold.
4. **Distracted** → If YOLO detects phone usage or head movement.
5. **Warning Issued** → When any condition persists beyond the threshold.

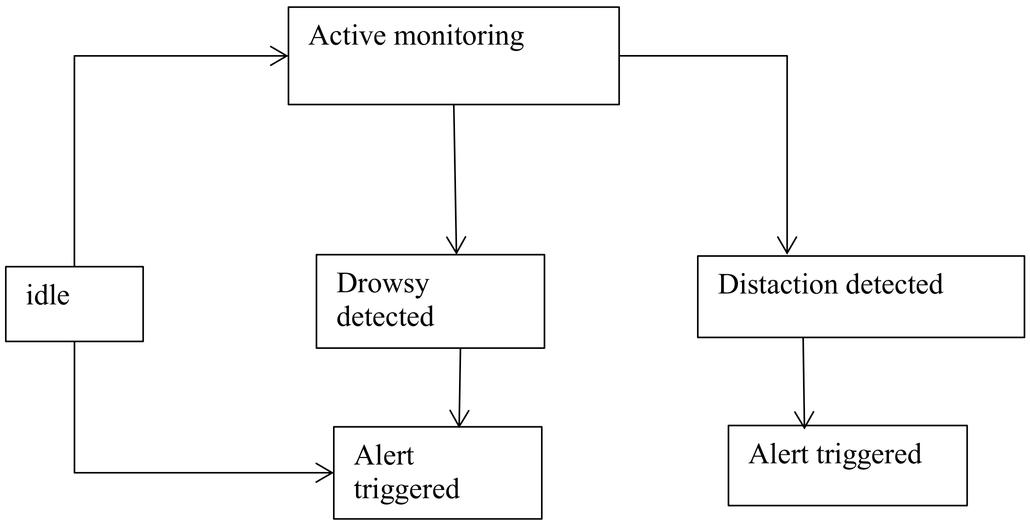


Figure 53

<u>4.5.3 Functional Testing</u>

It is a type of software testing that validates the software system against the functional requirements/specifications.

<u>Test cases for drowsiness detection</u>

Table 1

Test case	Input condition	Expected output	Actual output	Passs/fail
Normal eye state	Open eyes	No alert	No alert	pass
Drowsy state	Closed eye for >20 frames	Drowsy alert!	Drowsy alert!	pass
Partial aye closure	Half-closed eyes	No alert	No alert	pass
Face tilted	Tilted head with visible eyes	Correct detection	Correct detection	pass

Logical testing

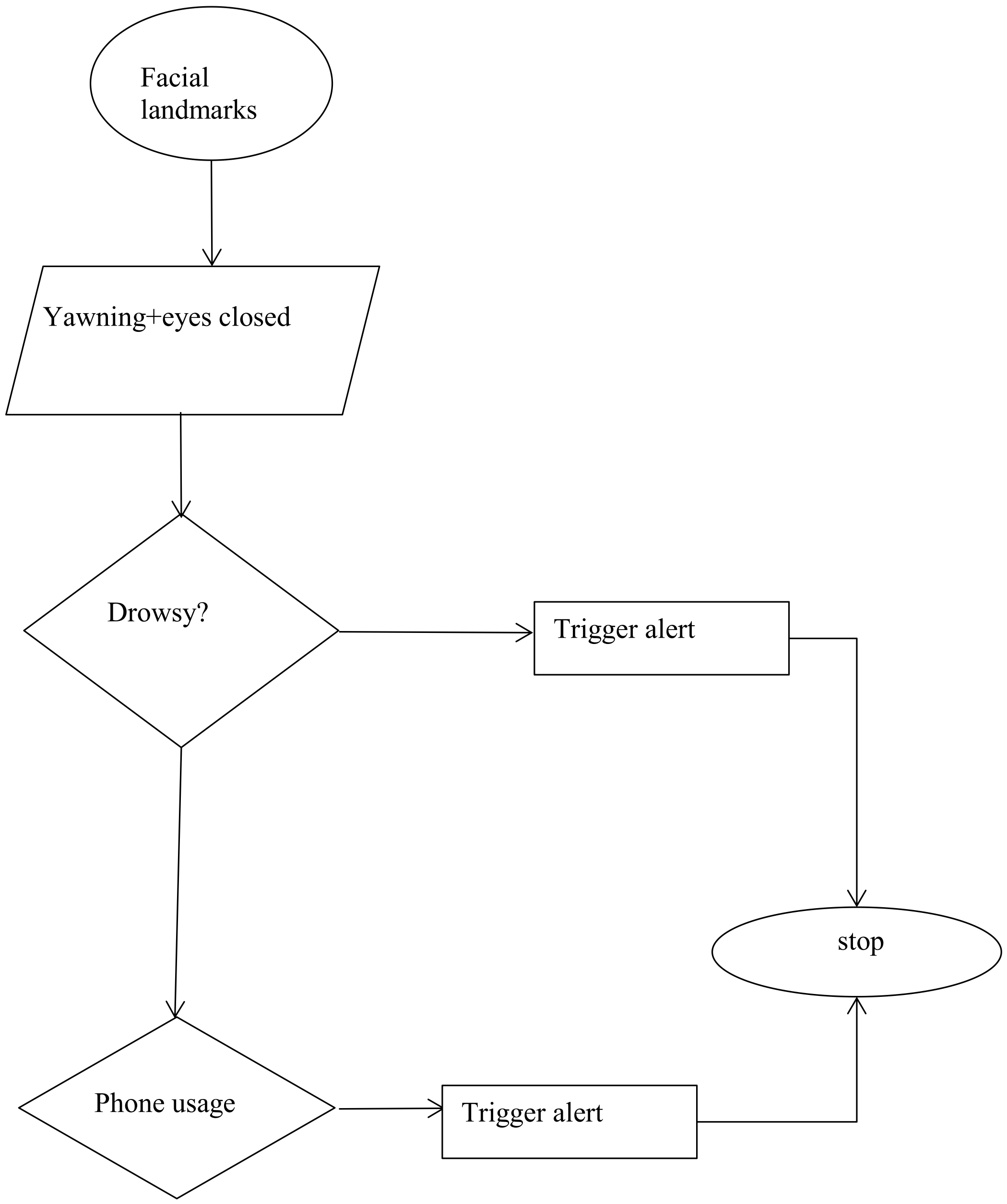


Figure 6

Test cases for yawning detection

Table 2

Test case	Input condition	Expected output	Actual output	Passs/fail
Normal mouth position	Closed mouth	No alert	No alert	pass
yawning	MAR>0.5 for >15 frames	Yawning alert	Yawning alert	pass
smiling	Wide smile	No alert	No alert	pass
talking	Speaking normally	No alert	No alert	pass

Phone usage alert

Table 3

Test case	Input condition	Expected output	Actual output	Passs/fail
No phone in hand	No phone visible	No alert	No alert	pass
Phone in hand	Phone detected	Phone usage alert	Phone usage alert	pass
Holding an object	Object detected (not a phone)	No alert	No alert	pass
Phone on the side	Phone not in view	No alert	No alert	pass

User accecptance testing
the System was deployed on a laptop with webcam then I documented feedback on false alerts & missed detections

Table 4

User feedback area	Rating 1-5	remarks
Accuracy of detection	4.6	Works well, minor tuning needed
alert effectiveness	4.9	Audible and visible alerts are clear
System lag	4.3	Real-time but slight delays sometimes
Ease of use	4.8	Simple and automatic

4.6. Modifications and Improvements

once I tested the system I realized that object detection datasets yolov3 was too slow hence I had to transit to yolov8 which was faster and it prevente freezing during real-time detection. Also I detected frame sampling delay therefore I increased frame sampling rate and optimized MobileNetV2 model which improved the response time. Also I added start and stop buttons so that users can easily control the system. For frequent false yawning alerts, I increased mouth aspect ratio (MAR) threshold for yawning detection eliminating false yawning alerts when talking

CHAPTER FIVE: REULTS AND DISCUSSIONS

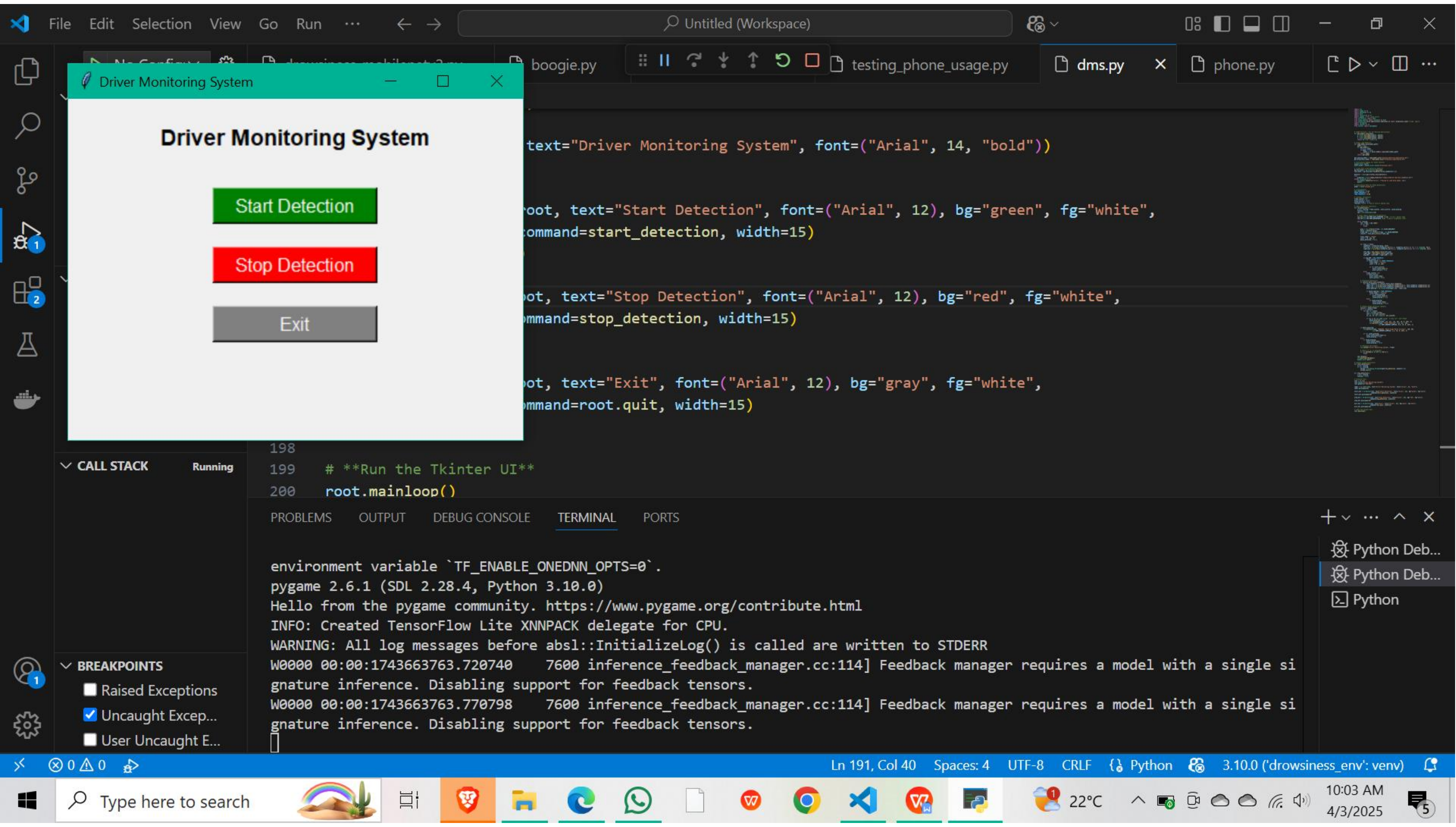
5.1test reports

Test case	Input condition	Expected output	Actual output	Passs/fail
Phone usage detection	Driver holding phone near ear	Alert triggered: WARNING: phone usage while driving	WARNING: phone usage while driving	pass
Phone usage detection	Driver using a phone for scrolling	Alert triggered: "WARNING: Phone Usage While Driving!"	WARNING: phone usage while driving	pass
Drowsiness Detection (Closed Eyes)	Driver closes eyes for > 2 sec	Alert triggered: "WARNING: Drowsiness Detected!"	Alert triggered: "WARNING: Drowsiness Detected!"	pass
Drowsiness Detection (Yawning)	Driver opens mouth wide for > 1.5 sec	Alert triggered: "WARNING: Yawning Detected!"	Alert triggered: "WARNING: Yawning Detected!"	pass
Drowsiness Detection (Head droop)	Driver tilts head excessively (>15°)	Alert triggered: "WARNING: Drowsiness Detected		
Partially closed eyes	Driver squints their eyes	awake	awake	pass
Wide smile	Driver smiles widely	No alert	No alert	pass

Table 5

5.2 user documentation

STARTING THE PROGRAM



```
text="Driver Monitoring System", font=("Arial", 14, "bold"))

oot, text="Start Detection", font=("Arial", 12), bg="green", fg="white",
ommand=start_detection, width=15)

ot, text="Stop Detection", font=("Arial", 12), bg="red", fg="white",
mmand=stop_detection, width=15)

ot, text="Exit", font=("Arial", 12), bg="gray", fg="white",
mmand=root.quit, width=15)

# **Run the Tkinter UI**
root.mainloop()
```

```
environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
pygame 2.6.1 (SDL 2.28.4, Python 3.10.0)
Hello from the pygame community. https://www.pygame.org/contribute.html
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
W0000 00:00:1743663763.720740    7600 inference_feedback_manager.cc:114] Feedback manager requires a model with a single si
gnature inference. Disabling support for feedback tensors.
W0000 00:00:1743663763.770798    7600 inference_feedback_manager.cc:114] Feedback manager requires a model with a single si
gnature inference. Disabling support for feedback tensors.
```

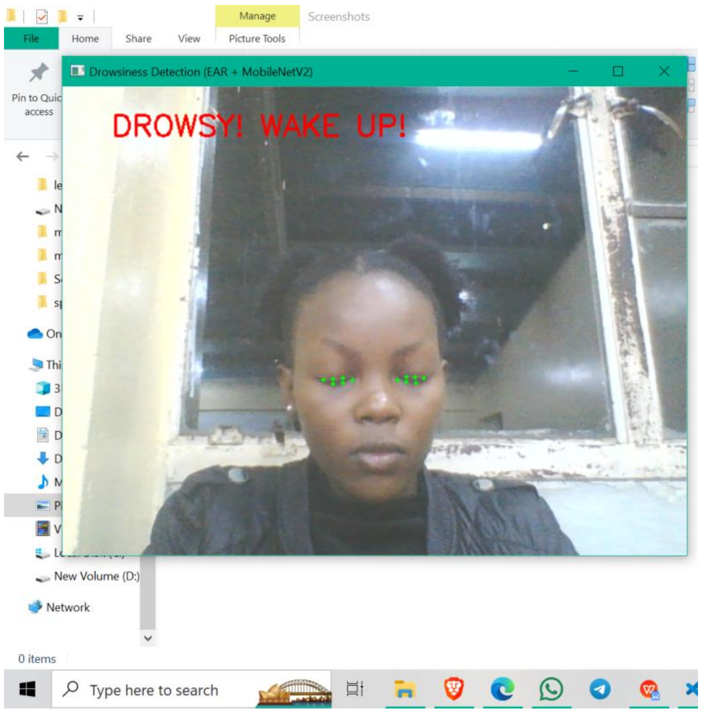
To start the program press **start detection**
To stop the program but not necessarily exit press **stop detection** button
To exit press **exit**

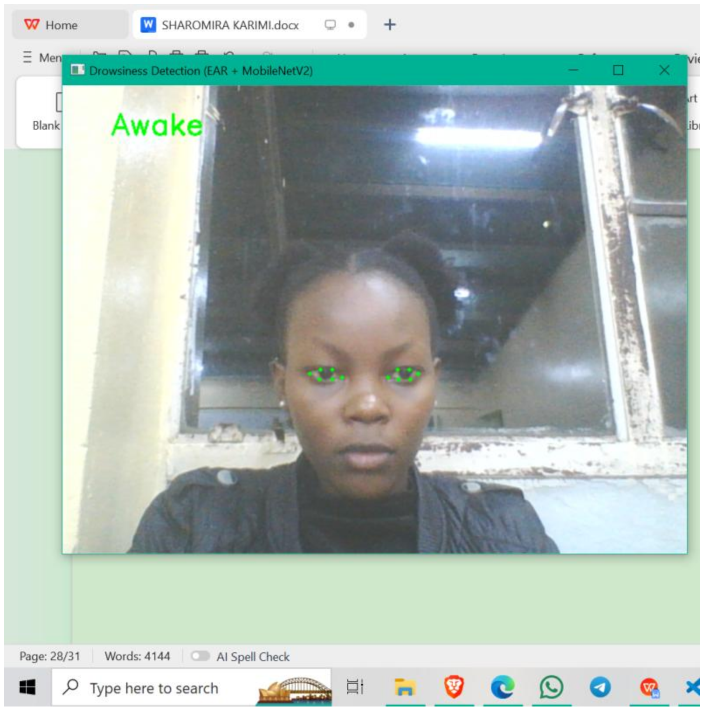
Drowsiness detection

Detects driver drowsiness by analyzing eye aspect ratio **(EAR)**. If eyes remain closed for a set number of frames, a drowsiness alert is triggered.

How it works

Detects the driver's facial landmarks.Computes EAR (Eye Aspect Ratio) to check if eyes are closed.Triggers an alert if the driver's eyes stay closed beyond a threshold.





Yawning detection

Detects yawning based on the mouth aspect ratio (MAR).If MAR exceeds the threshold for a given time, the system alerts the driver.

How it works

Detects the driver's mouth region using facial landmarks.Computes MAR (Mouth Aspect Ratio) to check if yawning is occurring.If the driver yawns continuously for more than 15 frames, a "Yawning Alert" is triggered.

<u>Phone usage detection</u>
Uses **YOLOv8 object detection** to identify if a driver is using a **phone while driving**.If a phone is detected, the system **issues an alert**.
<u>How it works</u>

- The **YOLOv8 model** scans for **mobile phones** in the driver's hand. If a phone is detected, a **"Distraction Alert"** is triggered.

CHAPTER SIX: CONCLUSIONS

6.1 conclusion and future works

6.1.1 CONCLUSION

The neural vision-based road safety system developed in this project successfully monitors driver behavior in real-time to detect potential distractions and fatigue. By integrating deep learning models (YOLOv8, MobileNetV2) and computer vision techniques (MediaPipe, OpenCV), the system identifies:

Phone Usage Detection – Recognizes when a driver is using a phone by detecting both hand near ear and finger scrolling gestures.
Drowsiness Detection – Analyzes eye closure (EAR), yawning (MAR), and head pose estimation to assess fatigue levels.
Real-Time Alerts – Provides on-screen warnings and audio alerts to notify drivers of unsafe behavior.

The performance enhancements and modifications implemented during testing significantly improved the system's accuracy, efficiency, and usability. The GUI integration also makes it more accessible for users, allowing easy activation and control.

Overall, the system demonstrates a practical approach to improving road safety, particularly in reducing accidents caused by driver distraction and drowsiness.

6.1.2 future works

Implementing the system on edge devices (Raspberry Pi, Jetson Nano) for real-time processing in vehicles. Expanding the dataset with more diverse driving scenarios (different lighting conditions, driver positions, and vehicle types). Developing a mobile version that allows users to receive notifications and performance reports. Exploring transformer-based vision models (Vision Transformers, EfficientNet) for more precise drowsiness detection. Integrating with IoT and cloud services to track driver behavior trends over time and provide remote monitoring for fleet management. Implementing self-learning AI that adapts to each driver's unique behavior, reducing false alerts and improving accuracy. Future improvements in impairment detection systems may be expected through a combination of improved hardware, physiological measures from unobtrusive sensors and wearables, and the intelligent integration of environmental variables like time of day and time on task

APPENDICES

6.2 APPENDIX 1: BUDGET

Table 1

ITEM	QUANTITY	UNIT PRICE	TOTAL(Ksh)
Printing and binding		5000	5000
Laptop	1	40000	40000
Software	3	40000	120000
Internet		6000	6000
Miscellaneous		15000	15000
TOTAL (ksh)			186,000

6.3 APPENDIX 2: SCHEDULE

Table 2

ACTIVITY	SEPTEMBER	OCTOBER	NOVEMBER	DECEMBER	JANUARY	FEBRUARY	MARCH
Project identification	🟦						
System analysis		🟦					
System Design		🟦					
Coding and Testing			🟦	🟦			
Implementation					🟦		
Documentation						🟦	
Project submission							🟦

REFERENCES

[1] F. Rundo, R. Leotta and S. Battiato, "Real-Time Deep Neuro-Vision Embedded Processing System for Saliency-based Car Driving Safety Monitoring," pp. 218-224, Malaysia , 2021.

[2] "neonode.com," noenode, [Online]. Available: e86378bdd59f6769&sxsrf=ADLYWIJtOvxognbJBjUWok-XQODpcIlGbg:1732194794847&source=lnms&fbs=AEQNm0AbGf0PdIf3AgPCbPKuvq3LsKshsZAx84Zxp11idRJxGfP-6NEi7rHAWuISAhkPt6M3_Xrj0YdsWvdt-fLnMenxutvtEkWBPOVz4uyMeQhFcrpXFgw18XslQWcyw52EcEDwHgQymWNARns6D0hoHswUJGf9ItjVMyE.

[3] "samsara," samsara ventures, [Online]. Available: https://www.samsara.com/.

[4] "azuga," bridgestone company, [Online]. Available: https://www.azuga.com/.

[5] K. Alkaabi, "Identification of hotspot areas for traffic accidents and analyzing drivers’ behaviors and road accidents," ISSN 2590-1982, United Arab Emirates, 2023.

```python
import cv2
import numpy as np
import mediapipe as mp
import dlib
import tensorflow as tf
import pygame  # For sound alerts
import threading
import tkinter as tk
from tkinter import messagebox
from ultralytics import YOLO
from scipy.spatial import distance as dist
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input # type: ignore

# **Lazy Load Models**
def load_model_once(model_path):
    model = None
    def get_model():
        nonlocal model
        if model is None:
            model = tf.keras.models.load_model(model_path)
        return model
    return get_model

get_yawning_model = load_model_once("yawning_detection_mobilenetv2.h5")
get_drowsiness_model = load_model_once("drowsiness_mobilenetv2.h5")

# **Class Labels**
class_labels = ["No Yawn", "Yawn"]

# **Initialize Pygame for Sound Alerts**
pygame.mixer.init()
alert_sound = pygame.mixer.Sound("drowsiness.wav")

# **Initialize Face Mesh & Hand Tracking**
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(refine_landmarks=True)

# **Load Dlib Face Detector & Landmark Predictor**
detector = dlib.get_frontal_face_detector()
try:
    predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
except Exception as e:
    print(f"Error loading Dlib model: {e}")
    exit()

# **Eye Aspect Ratio (EAR) Function**
def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    return (A + B) / (2.0 * C)

# **Reduce Exposure Function**
def reduce_exposure(image, alpha=0.5, beta=0):
    adjusted = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
    return adjusted

# **Landmark Indices for Eyes**
LEFT_EYE = list(range(42, 48))
RIGHT_EYE = list(range(36, 42))

# **Thresholds**
EAR_THRESHOLD = 0.20
FRAME_THRESHOLD = 5
YAWN_THRESHOLD = 0.08

# **Initialize Counters**
```