ABSTRACT

"In recent years, the issue of driver drowsiness has become a critical concern due to its potential to cause accidents and fatalities on roads worldwide. Various automated driver drowsiness detection systems have been developed to address this, utilizing advanced technologies such as computer vision, machine learning, and sensor integration. This abstract focuses on reviewing the methodologies and advancements in these systems, emphasizing their effectiveness in real-time monitoring of driver fatigue through facial recognition, eye-tracking, and physiological signal analysis. Furthermore, such systems' challenges and future prospects are discussed, highlighting the need for continuous innovation to enhance accuracy and reliability in preventing drowsiness-related accidents."

"Driver drowsiness remains a critical safety concern on roadways, necessitating effective and reliable detection systems. This abstract reviews a software-based driver drowsiness detection system designed to enhance road safety through real-time monitoring of driver alertness. The system integrates computer vision techniques for facial recognition and eye-tracking, coupled with machine learning algorithms to analyze driver behavior and physiological signals. By continuously assessing indicators such as eye closure, head position, and facial expressions, the system accurately identifies signs of drowsiness. This abstract explores the system's methodology, highlighting its potential to preemptively alert drivers and mitigate risks associated with fatigue-induced accidents. Furthermore, challenges in implementation and avenues for future advancements are discussed, emphasizing the ongoing efforts to refine and optimize the system's performance in diverse driving conditions."

Table of Contents

Acknowledgement i
Abstract ii
Table of Contents iii

CHAPTER	TITLE	PAGE NO.
1	**Introduction**	**1-5**
2	**Literature Survey** 2.1 Present System vs Proposed System 2.2 Problem Statement/Description 2.3 Objectives of the Project	**6-10**
3	**Requirement Specification** 3.1 Hardware Requirements 3.2 Software Requirements 3.3 Development Platform 3.4 Language used in coding 3.6 Functional Requirements	**11-20**
4	**Algorithm Design and Analysis** 4.1 Pseudo Code 4.2 Analysis	**21-25**
5	**Implementation** 5.1 Functions used 5.2 User defined functions	**26-30**
6	**Results and Discussion** 6.1 Discussion 6.2 Results 6.3 Snapshots	**31-38**
7	**Conclusion and Future Scope**	**39**
8	**Bibliography**	**40**

CHAPTER 1

INTRODUCTION

"In recent years, driver drowsiness has emerged as a significant factor contributing to road accidents globally. To mitigate this risk, advanced driver drowsiness detection systems leveraging software technologies have been developed. These systems employ computer vision and machine learning algorithms to monitor driver behavior and physiological signals in real-time. By detecting early signs of drowsiness such as eyelid closure and head position changes, these software-based solutions aim to alert drivers promptly, thereby reducing the incidence of accidents caused by impaired driving due to fatigue. This introduction provides an overview of how software-based approaches are crucial in enhancing road safety by addressing the challenge of driver drowsiness effectively."

The attention level of driver decreases because of less sleep, long continuous driving or any other medical condition like brain disorders etc. Several surveys on road accidents says that around 40 percent of accidents are caused by fatigue of the driver. When human drives for more than normal period for human then excessive fatigue is caused and also results in tiredness which drives the driver to sleepy condition or loss of consciousness [1].

Drowsiness is a complex phenomenon which states that there is degrade in alerts and conscious levels of the driver. Though there are no direct measures to detect the drowsiness but several indirect methods can be used for this purpose [2]. The consequences of a drowsy driver are very dangerous that can lead to loss of lives, casualties and vehicle damage. As the most important safety factor, it is necessary to make some serious measures, in order to improve working conditions of drivers, so that negative consequences subjected by a drowsy driver can be minimized.

Various different types of methods for measuring the drowsiness of the driver are mentioned which includes Vehicle based measures, Physiological measures, Behavioural measures [3]. Using these methods an intelligence system can be developed which would alert the driver in case of drowsy condition and prevent accidents. Depending on advantages and disadvantages the most precise method is chosen and proposed. Then the working for entire system development is explained. Then each frames are analysed to find face first. If a face is detected then the next task is to locate the eyes. After the positive result of detecting eye the amount of closure of eye is calculated and compared with the reference values for the drowsy state eye. If drowsy condition of driver is found out then driver is alarmed else repeatedly the loop of finding face and detecting drowsy condition is carried out [4].

When the driver is driving, the driver’s face is captured by a camera and it is converted into a video stream. The system then analyses the video to detect drowsiness and fatigue and also checks the level of drowsiness. In this stage, the most important parts which should be considered for analysis are: the driver’s face tracking, driver’s fatigue state, and recognition of key regions of the face based on eye closure. Finally, if the fatigue is detected.

Drowsiness is a state of near sleep, where the person has a strong desire for sleep. It has two distinct meanings, referring both to the usual state preceding falling asleep and the chronic condition referring to being in that state independent of a daily rhythm [16]. Sleepiness can be dangerous when performing tasks that require constant concentration, such as driving a vehicle. When a person is sufficiently fatigue while driving, they will experience drowsiness and this leads to increase the factor of road accident.

Figure : Examples of Fatigue & Drowsiness Condition

The development of technologies for detecting or preventing drowsiness while driving is a major challenge in the field of accident avoidance system. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects.

The aim of this project is to develop a simulation of drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes and mouth. By monitoring the eyes, it is believed that the symptoms of driver's drowsiness can be detected in sufficiently early stage, to avoid a car accident. Yawning detectionis a method to assess the driver's fatigue. When a person is fatigue, they keep yawning to ensure that there is enough oxygen for the brain consumption before going to drowsiness state [17]. Detection of fatigue and drowsiness involves a sequence of images of a face, and the observationof eyes and mouth open or closed duration. Another method to detect eye closure is PERCLOS. This detection method is based on the time of eyes closed which refers to percentage of aspecific time.

The analysis of face images is a popular research area with applications such as face recognition, and human identification and tracking for security systems. This project is focused on the localization of the eyes and mouth, which involves looking at the entire image of the face, and determining the position of the eyes and mouth, by applying the existing methods in image- processing algorithm. Once the position of the eyes is located, the system is designed to determine whether the eyes and mouth are opened or closed, and detect fatigue and drowsiness.

- ## Significance of This Project

Drowsiness and fatigue lead the cause of road accident in Malaysia. Thus, Driver Drowsiness Detection by Using Webcam is being introduced to minimize and reduce the number of accidentsinvolving cars, lorries and trucks. It detects the drowsiness signs and alerts drivers when they are in drowsy state.

- ## Problem Statement

Current drowsiness detection systems monitoring the driver's condition requires complex computation and expensive equipment, not comfortable to wear during driving and is not suitable for driving conditions; for example, Electroencephalography (EEG) and Electrocardiography (ECG), i. e. detecting the brain frequency and measuring the rhythm of heart, respectively.

A drowsiness detection system which use a camera placed in front of the driver is more suitable to be use but the physical signs that will indicate drowsiness need to be located first in order to come up with a drowsiness detection algorithm that is reliable and accurate. Lighting intensity and while the driver tilt their face left or right are the problems occur during detection of eyes and mouth region.

Therefore, this project aims to analyze all the previous research and method, hence propose a method to detect drowsiness by using video or webcam. It analyzes the video images that have been recorded and come up with a system that can analyze each frame of the video.

Objectives

The project focuses on these objectives, which are:

- To suggest ways to detect fatigue and drowsiness while driving.
- To study on eyes and mouth from the video images of participants in the experiment of driving simulation conducted by MIROS that can be used as an indicator of fatigue and drowsiness.
- To investigate the physical changes of fatigue and drowsiness.
- To develop a system that use eyes closure and yawning as a way to detect fatigue and drowsiness.

Scope of Study

In this project, the author will focus on these following procedures:

- Basic concept of drowsiness detection system
- Familiarize with the signs of drowsiness
- Determine the drowsiness from these parameters
  - Eye blink
  - Area of the pupils detected at eyes
  - Yawning
- Data collection and measurement.
- Integration of the methods chosen.
- Coding development and testing.

  Complete testing and improvement

Relevancy of the Project

This project is relevant to the implementation since fatigue and drowsiness drivers contribute to the percentage of road accidents. Many researches have been conducted to implement safe driving systems in order to reduce road accidents. Detecting the driver's alertness and drowsinessis an efficient way to prevent road accidents. With this system, drivers who are drowsy will be alerted by an alarm to regulate consciousness, attention and concentration of the drivers. Thiswill help to reduce the number of road accidents.

The aim of this project is to develop a simulation of drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes and mouth. By monitoring the eyes, it is believed that the symptoms of driver's drowsiness can be detected in sufficiently early stage, to avoid a car accident. Yawning detectionis a method to assess the driver's fatigue. When a person is fatigue, they keep yawning to ensure that there is enough oxygen for the brain consumption before going to drowsiness state [17]. Detection of fatigue and drowsiness involves a sequence of images of a face, and the observationof eyes and mouth open or closed duration. Another method to detect eye closure is PERCLOS. This detection method is based on the time of eyes closed which refers to percentage of specific.

The analysis of face images is a popular research area with applications such as face recognition, and human identification and tracking for security systems. This project is focused on the localization of the eyes and mouth, which involves looking at the entire image of the face, and determining the position of the eyes and mouth, by applying the existing methods in image- processing algorithm. Once the position of the eyes is located, the system is designed to determine whether the eyes and mouth are opened or closed, and detect fatigue and drowsiness.

Drowsiness and fatigue lead the cause of road accident in Malaysia. Thus, Driver Drowsiness Detection by Using Webcam is being introduced to minimize and reduce the number of accidentsinvolving cars, lorries and trucks. It detects the drowsiness signs and alerts drivers when they are in drowsy state.

This project is relevant to the implementation since fatigue and drowsiness drivers contribute to the percentage of road accidents. Many researches have been conducted to implement safe driving systems in order to reduce road accidents. Detecting the driver's alertness and drowsinessis an efficient way to prevent road accidents. With this system, drivers who are drowsy will be alerted by an alarm to regulate consciousness, attention and concentration of the drivers. Thiswill help to reduce the number of road accidents.

CHAPTER 2
LITERATURE SURVEY

A literature survey on driver drowsiness detection systems using software reveals a diverse range of approaches and methodologies aimed at enhancing road safety by preventing accidents caused by driver fatigue. Several key studies highlight advancements in computer vision, machine learning, and sensor integration to develop effective detection systems.

1. Image Processing and Computer Vision Techniques:

- Many systems use cameras to monitor the driver's face and eyes for signs of drowsiness such as eye closure, drooping eyelids, and head pose.
- Techniques include facial landmark detection, eye tracking, and image segmentation to isolate the driver's face from the background.

2. Machine Learning and AI Algorithms:

- Supervised learning algorithms such as Support Vector Machines (SVM), Random Forests, and deep learning models (like Convolutional Neural Networks - CNNs) are commonly used to classify drowsiness based on features extracted from images or sensor data.
- Unsupervised learning approaches like clustering can also be applied for anomaly detection or pattern recognition.

3. Sensor Integration:

- Besides cameras, systems may incorporate other sensors such as EEG (Electroencephalography), ECG (Electrocardiography), steering angle sensors, and vehicle speed sensors to gather additional data for detecting drowsiness.
- Sensor fusion techniques are used to combine data from multiple sensors to improve accuracy.

4. Real-time Monitoring and Alert Systems:

- Algorithms are designed to operate in real-time to continuously monitor the driver's state and provide timely alerts.
- Alerts can be auditory, visual (e.g., warnings on dashboard), or haptic (vibrations in the steering wheel or seat).

5. Evaluation and Performance Metrics:

- Studies often evaluate the effectiveness of drowsiness detection systems using metrics such as sensitivity, specificity, accuracy, and reaction time to alarms.
- Field studies and driving simulations are used to validate these systems in real-world conditions.

6. Challenges and Future Directions:

- Challenges include robustness to varying lighting conditions, different driver characteristics (e.g., glasses, facial hair), and integration into vehicles with different dashboard designs.

2.1 Present System v/s Proposed System

Present System:

1. **Basic Alert Mechanism:**
   The present system likely includes basic drowsiness detection techniques such as monitoring steering wheel movements, lane deviation, or using a fatigue detection algorithm based on time-driven alerts.

2. **Limited Input Sensors:**
   It might rely on a limited number of input sensors such as a single camera for facial recognition or basic physiological sensors (like heart rate monitors).

3. **Alert Mechanism:**
   Alerts are typically in the form of audible warnings, vibration alerts, or visual prompts on the dashboard.

4. **Single-Mode Detection:**
   It operates primarily on a single mode of detection (e.g., facial recognition or steering behavior) without integrating multiple data sources.

5. **Accuracy Concerns:**
   There might be concerns regarding false positives or false negatives due to limited data inputs and less sophisticated algorithms.

Proposed System:

1. **Enhanced Sensor Integration:**
   The proposed system aims to integrate multiple sensors such as infrared cameras for eye tracking, additional steering sensors for more precise behavior analysis, and possibly even EEG (Electroencephalogram) sensors for brainwave monitoring.

2. **Advanced Algorithms:**
   It will utilize more advanced algorithms such as machine learning models (e.g., deep learning networks) to process data from multiple sensors simultaneously. These models can detect subtle patterns of drowsiness more accurately.

3. **Real-Time Monitoring and Feedback:**
   The proposed system includes real-time monitoring and feedback mechanisms that continuously assess the driver's condition and provide timely alerts. This could include adaptive alert systems that vary in intensity based on the severity of detected drowsiness.

4. **Integration with Vehicle Systems:**
   It may integrate more closely with vehicle systems, such as adjusting seat vibrations, cabin temperature, or even engaging semi-autonomous driving modes to assist the driver when necessary.

5. **Improved Accuracy and Reliability**: By combining data from multiple sensors and using advanced algorithms, the proposed system aims to significantly improve accuracy

6. **User Interface Enhancements:**
   There might be improvements in the user interface, such as more intuitive alert displays orintegration with smartphone apps for remote monitoring.

2.2 Problem Statement / Description

Problem Statement

Design and implement a software-based driver drowsiness detection system that utilizes computer vision techniques to analyze facial features and movements in real-time. The system should accurately detect signs of driver fatigue and issue timely alerts to mitigate the risks of accidents due to drowsy driving.

Steps to Develop the System:

1. **Data Collection:**
Gather a diverse dataset of images or videos containing drivers exhibiting various levels of drowsiness. This dataset will serve as the basis for training and testing the detection algorithms.

2. **Preprocessing:**
Preprocess the collected data to enhance the quality and suitability for analysis. This may involve tasks such as image resizing, normalization, and noise reduction.

3. **Feature Extraction:**
Utilize computer vision techniques to extract relevant features from the driver's face, such as eye closure patterns, head pose, and blink frequency. These features will be critical for identifying signs of drowsiness.

4. **Algorithm Development:**
Develop machine learning or deep learning algorithms capable of learning from the extracted features to classify the driver's alertness level. Popular approaches include convolutional neural networks (CNNs) for image analysis or simpler algorithms like SVM (Support Vector Machine) for classification tasks.

5. **Real-time Detection:**
Implement the trained algorithm into a real-time system that continuously processes video input from a camera facing the driver. This system should analyze facial expressions and movements frame by frame to detect any signs of drowsiness.

6. **Alert Mechanism:**
Design an alert mechanism that triggers when the system detects significant signs of driver drowsiness. Alerts can vary from audible alarms to visual warnings on the dashboard, depending on the severity of the detected fatigue.

Problem Description

Driver drowsiness is a significant cause of road accidents globally. Fatigued drivers exhibit reduced reaction times, impaired judgment, and diminished attention, leading to dangerous driving conditions. A Driver Drowsiness Detection System aims to mitigate these risks by monitoring and analyzing driver behavior and physiological signals to detect signs of drowsiness, thereby alerting the driver and preventing potential accidents.

2.3 Objectives of the Project

- **Detection Accuracy:** Accurately identify signs of drowsiness in real-time.
- **Real-Time Processing:** Ensure the system operates in real-time to provide timely alerts.
- **User-Friendly Interface:** Develop an intuitive interface for drivers to receive alerts.
- **Non-Intrusive Monitoring:** Implement a monitoring system that does not distract or discomfort the driver.

Key Components:

- **Camera Module:** Captures real-time video of the driver's face to monitor eye movements and facial expressions.
- **Sensor Module:** (Optional) Includes sensors like heart rate monitors to detect physiological signs of drowsiness.
- **Software Algorithm:** Processes data from the camera and sensors to detect signs of drowsiness using machine learning techniques.
- **Alert System:** Provides auditory, visual, or haptic feedback to alert the driver.

Functional Requirements

- **Face Detection:** Identify the driver's face in the video stream.
- **Eye Tracking:** Monitor eye movements and blinks to detect signs of drowsiness.
- **Head Position Analysis:** Track head movements and posture.
- **Facial Expression Analysis:** Detect yawning or other signs of fatigue.
- **Physiological Signal Monitoring:** Analyze data from heart rate or other sensors (if used).
- **Alert Generation:** Trigger alerts when drowsiness is detected.
- **Data Logging:** Record instances of detected drowsiness for analysis and improvement of the system.

Non-Functional Requirements

- **Performance:** The system should process data with minimal latency to provide real-time alerts.
- **Accuracy:** High accuracy in detecting drowsiness to minimize false positives and negatives.
- **Reliability:** The system should function consistently under different conditions (e.g., varying lighting conditions).
- **Scalability:** Ability to handle data from multiple sensors if integrated.
- **Usability:** The interface should be easy to understand and use by drivers.

Implementation Considerations

- **Machine Learning Models:** Use pre-trained models or train new models using datasets of drowsy and non-drowsy driver behaviors.
- **Edge Computing:** Implement processing on the edge device to reduce latency.
- **Integration:** Ensure compatibility with various vehicle models and existing infotainment systems.
- **Regulatory Compliance:** Adhere to automotive industry standards and regulations.

Challenges

- **Variability in Driver Behavior**: Different drivers exhibit drowsiness differently, requiring a robust and adaptable detection algorithm.
- **Environmental Factors:** Changing lighting conditions, vibrations, and other factors can affect the accuracy of detection.
- **Privacy Concerns:** Ensuring data privacy and security for the driver.

Expected Outcomes

- **Reduced Accidents:** Significant reduction in accidents caused by driver fatigue.
- **Enhanced Safety:** Improved overall safety for drivers and passengers.
- **Driver Awareness:** Increased awareness among drivers regarding their drowsiness levels and the importance of taking breaks.

• Yawning Detection Method

The author propose a method where drowsiness can be detected by mouth positioning and the images were process by using cascade of classifier that has been proposed by Viola-Jones for faces. g but yawning is definitely a sign of a person havingdrowsiness and fatigue.

After gone through the research papers and the existing methods, this project proposed that eyes and yawning detection method will be used. Eye blink duration gives the data that the longer the person's close their eyes, the drowsier it will be considered. It is because when a person is in drowsy state; its eyes will be closed longer than the normal eye blink. Other than that, yawning isone of the symptoms of drowsiness where it is a normal human response when yawning is the sign that they feel drowsy or fatigue.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Hardware Requirements:

1. Camera Module

Description: A high-quality camera capable of capturing real-time video of the driver's face.

Specifications:

Resolution: At least 720p for clear image capture.

Frame Rate: Minimum 30 fps to ensure smooth tracking of facial features.

Infrared Capability: For night-time operation.

Examples: Logitech C920, Intel RealSense Depth Camera.

2. Infrared Sensors

Description: Sensors to detect the driver's eye movements and facial features in low light conditions.

Specifications:

Wavelength: Typically around 850 nm for eye safety.

Range: Should cover the driver's face area.

Examples: Adafruit IR Sensor, Leap Motion Controller.

3. Microcontroller/Processing Unit

Description: A processing unit to handle data from the camera and sensors in real-time.

Specifications:

CPU: Multi-core processor, at least ARM Cortex-A53 or equivalent.

GPU: Necessary for running machine learning models efficiently.

RAM: Minimum 2GB for smooth operation.

Examples: Raspberry Pi 4, NVIDIA Jetson Nano.

4. Physiological Sensors (Optional)

Description: Sensors to monitor the driver's physiological signals like heart rate.

Specifications:

Accuracy: High accuracy in detecting heart rate.

Connectivity: Bluetooth or other wireless communication.

Examples: Polar H10 Heart Rate Sensor, Garmin Heart Rate Monitor.

5. **Alert System**

Description: Devices to alert the driver when drowsiness is detected.

3.2 Software Requirements:

1. Operating System

Description: The base system software that manages hardware resources.
Examples: Linux (e.g., Raspbian for Raspberry Pi), Windows, or a custom real-time operating system (RTOS) for embedded systems.

2. Machine Learning Frameworks

Description: Libraries and tools for developing and running machine learning models.
Examples: TensorFlow, PyTorch, OpenCV.

3. Face and Eye Tracking Software

Description: Software for detecting and tracking the driver's face and eyes.
Examples: Dlib (for facial landmarks), OpenCV (for image processing), MediaPipe.

4. Drowsiness Detection Algorithms

Description: Algorithms to analyze facial and physiological data to detect drowsiness.
Types:
Convolutional Neural Networks (CNN) for image-based detection.
Recurrent Neural Networks (RNN) for time-series data analysis.
Examples: Custom trained models using TensorFlow or PyTorch.

5. Data Processing and Analytics

Description: Software to preprocess, analyze, and interpret data from sensors.
Examples: NumPy and pandas for data manipulation, scikit-learn for additional machine learning algorithms.

6. User Interface

Description: Software to provide feedback to the driver and display system status.
Examples: Custom GUI applications built with frameworks like Tkinter (Python), Qt, or Electron.

7. Communication Protocols

Description: Software libraries for communication between various components.
Examples: MQTT for message queuing, Bluetooth libraries for sensor communication HTTP/REST for cloud connectivity.

8. Data Storage and Logging

Description: Systems to store and log data for future analysis.

Examples: SQLite for local storage, cloud storage solutions like AWS S3 for remote storage.

9. Integrated Development Environment (IDE)

Description: Tools for developing and debugging software.

Examples: Visual Studio Code, PyCharm, Eclipse.

10. Version Control Systems

Description: Tools for managing source code versions.

Examples: Git, GitHub.

11. Testing Frameworks

Description: Tools for automated testing of software components.

Examples: pytest (Python), JUnit (Java), Selenium for UI testing.

3.3 Development Platform:

Developing a Smart Dustbin Monitoring and Alert System using IoT involvesselecting appropriate development platforms for both hardware and software. Here'sa detailed outline of the development platforms typically utilized for such a system**:**

A. Hardware Development Platform:

- **Microcontrollers and Single-Board Computers:**
  Raspberry Pi: Suitable for high-level processing and integration with cameras and sensors.
  Arduino: Ideal for simpler, low-power applications, useful for basic sensor integration.
  NVIDIA Jetson Nano: Great for AI and machine learning applications, capable of running neural networks.

- **Cameras and Sensors:**
  Infrared (IR) Camera: Effective for detecting eye and facial movements in low-ligh conditions.
  Webcam: A basic option for capturing facial images and movements.
  Eye-Tracking Sensors: Specialized sensors for monitoring eye movements.
  Heart Rate Sensors: Detects changes in heart rate that may indicate drowsiness.
  Accelerometers and Gyroscopes: Measure head movements and orientation.

- **Development Kits:**
  Intel RealSense: Provides depth sensing and motion tracking capabilities.
  OpenCV and Dlib Libraries: Software libraries for computer vision tasks, useful for processing camera inputs.

- **Connectivity and Integration:**
  Bluetooth or Wi-Fi Modules: For wireless communication between components and central processing units.
  CAN Bus Modules: For integration with the vehicle's internal communication network.
  Power Supply:Ensure a stable and sufficient power supply compatible with the chosen components

B. Software Development Platform:

- **Machine Learning and AI Frameworks:**
  TensorFlow: A popular open-source framework for building and training machine learning models, especially neural networks.
  PyTorch: Another widely-used open-source deep learning framework, known for its flexibility and ease of use.
  Scikit-Learn: Useful for traditional machine learning algorithms and data preprocessing.

- **Computer Vision Libraries:**
  OpenCV: An open-source computer vision and machine learning software library with functions for real-time image and video processing.
  Dlib: A toolkit for machine learning and data analysis, with robust face detection and facial landmark detection capabilities.
  Mediapipe: Developed by Google, it provides pre-trained models for face and pose detection.

- **Integrated Development Environments (IDEs):**
  PyCharm: A powerful IDE for Python, providing tools for code analysis, graphical debugging, and integrated unit testing.
  Visual Studio Code: A lightweight but powerful code editor with support for debugging, version control, and extensions for various programming languages.

- **Data Processing and Analysis:**
  Pandas: A data manipulation and analysis library for Python, useful for handling and preprocessing data.
  NumPy: A fundamental package for scientific computing with Python, essential for handling arrays and numerical operations.

- **Development and Prototyping Platforms:**
  MATLAB: Provides tools for algorithm development, data visualization, and numerical computation, useful for prototyping and testing algorithms.
  Jupyter Notebooks: An open-source web application for creating and sharing documents that contain live code, equations, visualizations, and narrative text.

- **Embedded Software Development:**
  PlatformIO: An open-source ecosystem for IoT development with support for various embedded boards and platforms.
  Arduino IDE: An open-source platform for writing and uploading code to Arduino-compatible boards.

- **Data Storage and Management:**
  SQLite: A lightweight, disk-based database useful for local storage.
  Firebase: A cloud-based platform providing backend services such as real-time databases.

- **Cloud Services:**
  AWS (Amazon Web Services): Offers cloud computing services, including machine learning models, storage, and compute power.
  Google Cloud Platform (GCP): Provides cloud services including machine learning APIs, storage, and compute resources.

C. Development workflow

- **Research and Planning**
  Literature Review: Study existing methods and technologies for driver drowsiness detection.
  Requirements Gathering: Identify the needs and constraints of the system, including the hardware, software, and user requirements.
  Feasibility Study: Evaluate the technical and economic feasibility of the project.

- **Data Collection**
  Dataset Acquisition: Collect data on drowsiness, which can include images or videos of drivers, physiological data (e.g., heart rate), and vehicle data (e.g., steering patterns).
  Data Annotation: Label the collected data to distinguish between drowsy and alert states.

- **Preprocessing**
  Data Cleaning: Remove noise and irrelevant information from the data.
  Data Augmentation: Increase the size and variability of the dataset by applying transformations such as rotation, scaling, and flipping to images.
  Normalization: Standardize data to a consistent format and scale.

- **Feature Extraction**
  Image Processing: Extract features from images or videos using techniques like facial landmark detection to identify eyes, mouth, and head position.
  Physiological Signals: Extract relevant features from physiological data such as heart rate variability.
  Vehicle Dynamics: Analyze vehicle data to identify patterns associated with drowsy driving.

- **Model Development**
  Algorithm Selection: Choose appropriate machine learning algorithms (e.g., CNNs for image data, RNNs for sequential data).
  Model Training: Train the chosen algorithms on the preprocessed and labeled data.
  Model Evaluation: Assess the model's performance using metrics like accuracy, precision, recall, and F1-score. Use cross-validation to ensure robustness.

- **Integration and Testing**
  System Integration: Combine the trained model with other system components, such as cameras, sensors, and in-car displays.
  Testing: Perform extensive testing in simulated environments and real-world scenarios to validate system performance and reliability.

- **Deployment**
  Pilot Deployment: Roll out the system in a controlled environment to gather feedback and make necessary adjustments.
  Full Deployment: Deploy the system in the target environment (e.g., commercial vehicles, personal cars).

- **Monitoring and Maintenance**

Continuous Monitoring: Track the system's performance and accuracy over time.
Updates and Improvements: Regularly update the system with new data and retrain the model to improve accuracy and address emerging issues.
User Feedback: Collect and incorporate feedback from users to enhance the system's usability and effectiveness.

- **Tools and Technologies**

Hardware: Cameras, infrared sensors, physiological sensors.
Software: Python, OpenCV, TensorFlow, Keras, Scikit-learn.
Frameworks: Deep learning frameworks for model development, edge computing platforms for real-time processing.

3.4 Language used in coding:

Driver drowsiness detection systems often involve a combination of software and hardware, using various programming languages to implement different parts of the system. Here are some common languages and their typical uses in such systems:

- **Python:**

Machine Learning and AI: Python is widely used for developing machine learning models due to its rich ecosystem of libraries like TensorFlow, Keras, and scikit-learn.
Computer Vision: Libraries like OpenCV, Dlib, and Mediapipe facilitate image and video processing tasks, which are crucial for detecting facial features and expressions related to drowsiness.

- **Research Methodology**

Usually, research methodology refers to a set of procedures that will be used to carry out a certain research. In order to complete this project systematically within the specified time, there are some methodologies and activities that need to be planned and followed consistently.

- **Flow Chart**

- **Background of Study**

Before starting any research or project, basic information of the related topic is required to ensure that the author understands what the project is all about. In this stage, the background of study helps the author understands the relation between drowsiness and fatigue. It also helps the author in understanding the seriousness of driving a motored vehicle in drowsiness condition. It is proven that driving the vehicle in fatigue and drowsiness condition is a lead factor to road accidents.

Figure : Feature for Face Detection

3.5 Coding in Python

Creating a driver drowsiness detection system in Python typically involves several steps, including data collection, feature extraction, model training, and real-time detection.Below is a high-level overview of how you might approach coding such a system in Python:

- **Data Collection**
  Collect a dataset that includes images or videos of drivers in various states of alertness and drowsiness.Possible sources of data include public datasets, custom recordings, or simulations.

- **Preprocessing**
  Image/Video Preprocessing: Resize, normalize, and augment the images or video frames to prepare them for model training.
  Face Detection: Use a library like OpenCV to detect and crop faces from the frames.

- **Feature Extraction**
  Extract relevant features such as eye aspect ratio (EAR), mouth aspect ratio (MAR), head pose, etc.
  Use libraries like Dlib or Mediapipe to detect facial landmarks and calculate these features.

- **Model Training**
  Use machine learning algorithms to train a model on the extracted features.
  You might use classifiers like SVM, Random Forest, or deep learning models like CNNs.

- **Real-time Detection**
  Implement the detection system to process real-time video feed and raise alerts when drowsiness is detected

3.6 Functional Requirements:

Functional requirements for a driver drownisness detection system outline the specific capabilities and behaviors the system must exhibit to meetits intended purpose effectively. Here's a structured list of functional requirements for such a system:

1. Fill Level Monitoring:

- **Requirement:** The system shall accurately monitor the fill level of the dustbin.
- **Description:** Use sensors (e.g., ultrasonic, load sensors) to measure and track the amount of rate
- **Acceptance Criteria:** The system shall provide real-time updates of the fill level with an accuracy of ±5%.

2. Alert Generation:

- **Requirement:** The system shall generate alerts when the dustbin reaches predefined fill levels.
- **Description:** Implement thresholds for different fill levels (e.g., 70%, 80%, 90%) and trigger alerts (e.g., visual, audible, SMS) when these thresholds are exceeded.
- **Acceptance Criteria:** Alerts shall be generated within 1 minute of reaching a predefined fill level.

3. Remote Monitoring:

- **Requirement:** The system shall allow remote monitoring of fill levels.
- **Description:** Enable users to access real-time data and receive alerts remotely via a web dashboard, mobile application, or SMS notifications.
- **Acceptance Criteria:** Users shall be able to view current fill levels and receive alerts on their preferred remote platform.

4. Automated Service Requests:

- **Requirement:** The system shall automatically initiate service requests based on fill level thresholds.
- **Description:** Integrate with service providers or municipal systems to automatically generate requests for waste collection or maintenance reaches capacity.
- **Acceptance Criteria:** Service requests shall include accurate location information and be triggered within 5 minutes of reaching a critical fill level.

1. User Authentication and Access Control:

- **Requirement:** The system shall implement user authentication and access control mechanisms.
- **Description:** Ensure secure access to monitoring and management functionalities, requiring user authentication (e.g., username/password, multi-factor authentication).
- **Acceptance Criteria:** Only authorized users shall have access to sensitive operations such as configuration changes and service requests.

2. Power Management:

- **Requirement:** The system shall optimize power usage to ensure continuous operation.
- **Description:** Implement power-saving techniques (e.g., sleep modes, low-power sensors) to extend battery life in battery-operated deployments
- **Acceptance Criteria:** The system shall operate continuously for at least 6 months on battery power with daily usage.

3. **Localization and Time Synchronization:**

- **Requirement:** The system shall support accurate localization and time synchronization.
- **Description:** Utilize GPS or Wi-Fi-based localization services to determine the dustbin's location and synchronize time with an accurate source (e.g., NTP server) for timestamping data.
- **Acceptance Criteria:** Location accuracy shall be within 5 meters, and time synchronization shall maintain a deviation of less than 1 second.

4. **Maintenance and Diagnostics:**

- **Requirement:** The system shall include maintenance and diagnostics capabilities.
- **Description:** Provide diagnostic tools (e.g., remote diagnostics, health checks) to monitor system health, detect sensor failures, and ensure timely maintenance.
- **Acceptance Criteria:** Maintenance alerts shall be generated for sensor failures within 24 hours, with detailed diagnostic information provided.

CHAPTER 4

ALGORITHM DESIGN AND ANALYSIS

4.1 Pseudo Code:

I. Segregation Code:

```
import cv2
import time
import os


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

eye_cascade = cv2.CascadeClassifier('haarcascade_eye_tree_eyeglasses.xml')

cap = cv2.VideoCapture(0)
a=0
b=0
c=time.time()
while 1:
ret, img = cap.read()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

if len(faces) == 0:
print("No face detected")

for (x,y,w,h) in faces:
cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

roi_gray = gray[y:y+h, x:x+w]
roi_color = img[y:y+h, x:x+w]

eyes = eye_cascade.detectMultiScale(roi_gray)

if (time.time()-c) >= 15:
  if a/(a+b)>=0.2:
    os.system('buzz.mp3')
    print("*ALERT**",a,b,a/(a+b))
    else:
```

```
print("safe",a//(a+b))
        c=time.time()



        a=0
        b=0

    if len(eyes) == 0:
        a=a+1
        print('no eyes!!!')
    else:
        b=b+1
        print('eyes!!!')

    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
    cv2.imshow('img',img)

    k = cv2.waitKey(1) & 0xff
    if k == 27:
    print("noeyes: ",a)
    print("total: ",(a+b))
    break

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

4.2 Analysis:

The provided code is designed to detect driver drowsiness by monitoring the presence of eyes in video frames captured from a webcam. Here's a breakdown and analysis of its functionality:

1. Imports and Initialization

```
import cv2
import time
import os

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye_tree_eyeglasses.xml')
cap = cv2.VideoCapture(0)

a = 0
b = 0
c = time.time()
```

cv2 for computer vision tasks.
Time for timing functionality.
os for executing system commands (though it's not the ideal choice for playing an alert sound).
Haar cascades for face and eye detection.
a and b are counters for eye presence/absence.
c is the timestamp for timing intervals.

2. Main Loop for Video Capture and Processing

```
while 1:
ret, img = cap.read()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

if len(faces) == 0:
    print("No face detected")
```

Captures a frame from the webcam.
Converts the frame to grayscale for easier processing.
Detects faces in the frame.
Prints a message if no faces are detected.

3. Face and Eye Detection within Each Frame

```
for (x, y, w, h) in faces:

    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

    roi_gray = gray[y:y+h, x:x+w]
```

```
    roi_color = img[y:y+h, x:x+w]

    eyes = eye_cascade.detectMultiScale(roi_gray)
```

Draws a rectangle around detected faces.
Defines regions of interest (ROI) for further eye detection within the face area.
Detects eyes within the ROI.

4. Drowsiness Detection Logic

```
    if (time.time() - c) >= 15:
        if a / (a + b) >= 0.2:
            os.system('buzz.mp3')
            print("*ALERT**", a, b, a / (a + b))
        else:
            print("safe", a // (a + b))
        c = time.time()
        a = 0
        b = 0
```

Checks if 15 seconds have passed since the last check.
Calculates the ratio of frames without eyes to total frames in the interval.
If the ratio is above 0.2, plays an alert sound and prints an alert message.
Otherwise, prints a "safe" message.
Resets the counters a and b, and the timer c.

5. Eye Detection and Counting

```
    if len(eyes) == 0:
        a += 1
        print('no eyes!!!')
    else:
        b += 1
        print('eyes!!!')

    for (ex, ey, ew, eh) in eyes:
        cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
```

Increments a if no eyes are detected, prints "no eyes".
Increments b if eyes are detected, prints "eyes".
Draws rectangles around detected eyes.

6. Displaying the Video and Handling Exit

```
cv2.imshow('img', img)

k = cv2.waitKey(1) & 0xff
if k == 27:
    print("noeyes:", a)
    print("total:", (a + b))
    break

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

Displays the video feed with detected faces and eyes.
Exits the loop if the 'Esc' key (ASCII 27) is pressed.
Prints final counts of frames with and without eyes.
Releases the video capture object and closes the display window.

Analysis and Improvements

- **Accuracy:**

The eye detection logic may not be very robust, especially under varying lighting conditions and angles.
Using more advanced techniques like deep learning-based face and eye detection models (e.g., Dlib, Mediapipe) can improve accuracy.

- **Alert Mechanism:**

os.system('buzz.mp3') is not an ideal way to play an alert sound. Libraries like playsound or pygame would be more appropriate and platform-independent.

- **Thresholds and Timing:**

The threshold ratio of 0.2 and the 15-second interval are arbitrary and might need tuning based on real-world testing and data.

- **Performance:**

Haar cascades are fast but not as accurate as modern methods. Using a GPU-accelerated deep learning model could provide better real-time performance and accuracy.

- **Modularity:**

Breaking the code into functions would make it more modular and easier to maintain.

CHAPTER 5
IMPLEMENTATION

5.1 Functions used

In this program there are several functions used. These include functions from the libraries you included as well as user-defined functions within the cv2.release() and cv2.destroyAllWindows() functions.

5.11 OpenCV Library Functions

- **cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'):**
  This function loads a pre-trained classifier for face detection. The CascadeClassifier is used to detect objects (in this case, faces) in video streams. The argument specifies the path to the XML file containing the pre-trained model.

- **cv2.VideoCapture(0):**
  This function initializes the video capture object. The argument 0 typically refers to the default camera of the system. It can also be a path to a video file.

- **cap.read():**
  This function reads a frame from the video capture object. It returns two values: a boolean (ret) indicating if the frame was read successfully, and the frame itself (img).

- **cv2.cvtColor(img, cv2.COLOR_BGR2GRAY):**
  This function converts the image from one color space to another. Here, it's converting the image from BGR (Blue, Green, Red) to grayscale, which is necessary for the face and eye detection algorithms.

- **face_cascade.detectMultiScale(gray, 1.3, 5):**
  This function detects objects (faces) in the image. Parameters:
  gray: The input image in grayscale.
  1.3: The scale factor that compensates for faces appearing larger or smaller.
  5: The minimum number of neighbors each rectangle should have to retain it. This helps in reducing false positives.

- **cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2):**
  This function draws a rectangle around the detected face. Parameters:
  img: The image on which to draw.
  (x, y), (x + w, y + h): The coordinates of the top-left and bottom-right corners of the rectangle.
  (255, 0, 0): The color of the rectangle (blue in BGR).
  2: The thickness of the rectangle lines.

- **eye_cascade.detectMultiScale(roi_gray):**
  This function detects eyes within the region of interest (ROI) defined by the face rectangle. Similar to face detection, it uses a pre-trained classifier for eyes.

- **cv2.imshow('img', img):**
  This function displays the image in a window. The first argument is the window name, and the second is the image to display.

- **cv2.waitKey(1) & 0xff:**
  This function waits for a key event for a specified amount of time (1 millisecond in this case). The & 0xff ensures it only considers the least significant byte of the key value, making it compatible across different platforms.

- **cap.release():**
  This function releases the video capture object, freeing the camera for other applications.

- **cv2.destroyAllWindows():**
  This function closes all OpenCV windows that were opened.

- **time.time():**
  This function returns the current time in seconds since the Epoch (January 1, 1970). It's used here to measure time intervals.

- **os.system('mpg123 buzz.mp3'):**
  This function runs a system command. In this case, it plays an audio file (buzz.mp3) using the mpg123 command-line audio player. Make sure mpg123 is installed on your system, or use another method to play audio.

5.2 User-Defined Functions:

1. Intialization

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye_tree_eyeglasses.xml')
cap = cv2.VideoCapture(0)
a = 0
b = 0
c = time.time()
```

- **Face and Eye Classifiers:** These lines load the Haar cascade models for face and detection.
- **Video Capture:** Initializes the webcam capture.
- **Counters and Timer:** Initializes counters a and b for tracking the number of frames without and with eyes detected, respectively, and sets a starting time c.

2. Main Loop

```
while 1:
ret, img = cap.read()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
if len(faces) == 0:
print("No face detected")
```

- **Capture Frame:** Captures a frame from the webcam.
- **Convert to Grayscale:** Converts the captured frame to grayscale.
- **Face Detection:** Detects faces in the grayscale image.
- **No Face Detected:** Prints a message if no faces are found.

3. Face and Eye Detection:

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

- **Draw Rectangle Around Face:** Draws a rectangle around each detected face.
- **Region of Interest (ROI):** Defines the ROI in both grayscale and color images for further processing.
- **Eye Detection:** Detects eyes within the face ROI.

4. Alert Logic

```
if (time.time() - c) >= 15:
if a / (a + b) >= 0.2:
os.system('buzz.mp3')
print("*ALERT**", a, b, a / (a + b))
else:
print("safe", a // (a + b))
c = time.time()
a = 0
b = 0
```

- **Check Time Interval:** Checks if 15 seconds have passed since the last check.
- **Calculate Eye Absence Ratio:** If the ratio of frames without eyes detected is greater than or equal to 20%, triggers an alert by playing a sound and printing an alert message.
- **Reset Timer and Counters:** Resets the timer and counters for the next interval.

5. Eye Presence Check

```
if len(eyes) == 0:
    a = a + 1
    print('no eyes!!!')
else:
    b = b + 1
    print('eyes!!!')
for (ex, ey, ew, eh) in eyes:
    cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 2)
```

- **Count Frames with/without Eyes:** Increments counters based on whether eyes were detected.
- **Draw Rectangle Around Eyes:** Draws a rectangle around each detected eye.

6. Display and Exit

```
cv2.imshow('img', img)
k = cv2.waitKey(1) & 0xff
if k == 27:
print("noeyes: ", a)
print("total: ", (a + b))
break
cap.release()
cv2.destroyAllWindows()
```

- **Display Image**: Displays the frame with rectangles drawn around detected faces and eyes.
- **Wait for Key Press:** Waits for a key press for 1 millisecond. If the 'Esc' key (ASCII 27) is pressed, prints the counts and breaks out of the loop.
- **Release Resources:** Releases the webcam and destroys all OpenCV windows.

Program Flow

1.Intialization

- **Import Libraries:**

  cv2: OpenCV library for computer vision tasks.

  time: Time library for handling time-related tasks.

  os: OS library for interacting with the operating system.

- **Load Haar Cascades:**

  Load pre-trained classifiers for face and eye detection using Haar cascades. These classifiers are used to detect faces and eyes in the input frames.

- **Initialize Video Capture:**

  Initialize video capture to use the default camera (usually the webcam).

- **Initialize Counters and Timer:**

  Initialize counters a and b to count frames without eyes and with eyes, respectively.

  Initialize a timer c to keep track of time for periodic checks.

2.Main Loop

The main loop continues indefinitely until the 'Esc' key is pressed.

- **Capture Frame:**Capture a frame from the video feed.
- **Convert to Grayscale:**Convert the captured frame to grayscale to simplify processing.
- **Detect Faces:**Detect faces in the grayscale frame using the face cascade classifier
- **Check for Faces:**Print a message if no faces are detected.
- **Process Each Detected Face:**For each detected face, process the region of interest (ROI) to detect eyes.
- **Periodic Check for Eye Absence:**Every 15 seconds, check the ratio of frames without eyes to total frames processed.If the ratio of frames without eyes is 20% or higher, trigger an alert by playing a sound and printing a message.Reset the timer and counters after the check.
- **Draw Rectangles Around Eyes:**Draw rectangles around detected eyes within the face ROI.
- **Display Frame:**Display the processed frame with rectangles drawn around detected faces and eyes.
- **Check for 'Esc' Key Press:**Wait for a key press for 1 millisecond. If the 'Esc' key (ASCII 27) is pressed, print the final counts and break the loop to exit.
- **Release Resources:**Release the video capture object and destroy all OpenCV windows.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 Discussion

- **Detecting Face**

To detect the face, author use the algorithm that are part of the Computer Vision Toolbox System which is Vision Cascade Detector. It creates a system object detector that detects object using Viola-Jones method. By default, the detector is configured to detect faces. the command script and the result of the face detection algorithm.

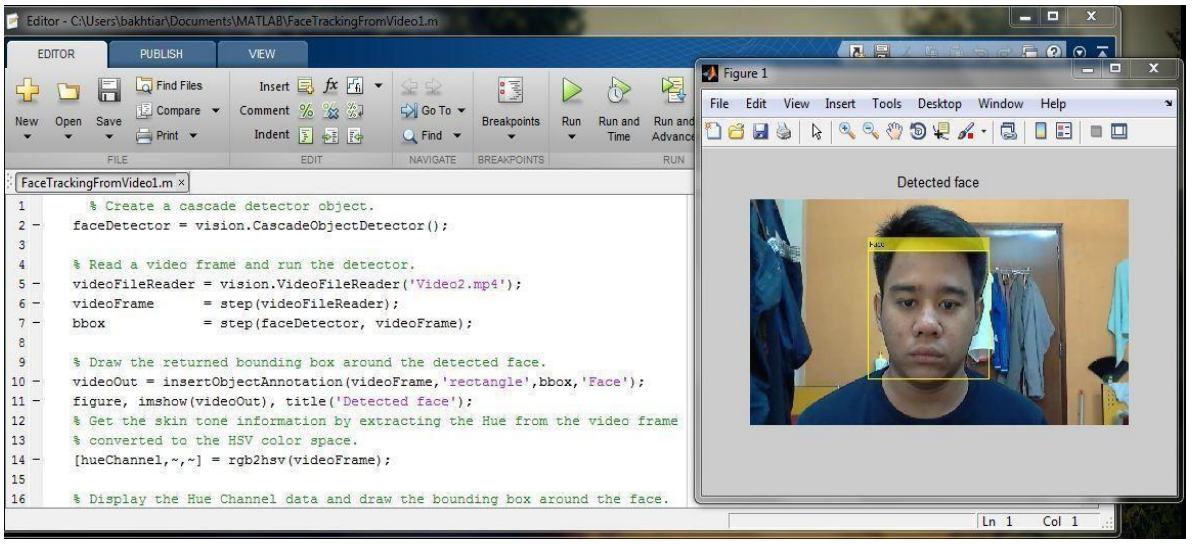

Figure : Face Detection Result

The features to detect face area are as below:

1. Define and setup the cascade object detector using the constructor. The constructor uses built-in Viola-Jones algorithm to detect faces, noses, eyes, mouth and upper body.
2. Read the video or the image selected and run the face detector.
3. Draw the bounding box around the detected face. The bounding box is the area of the desired detected face. Here, the bounding box is around the face area.

- **Detecting Eyes**

The eyes must be detected separately because when the drivers tilt their face, it still can be detected. The author uses the same algorithm as the face detection but here it changes the object to detect eyes. The resulting output of the eye detection algorithm . When author tries using different videos, a problem occurs; other parts in the video are detected as eyes. If the detected region is within the eye area, it is consider as True Positive. But during the trial, other parts in the video were detected as eyes far from the eye region which it is considered as False Positive. Definition for False Positive is a result that is erroneously positive when a situation is normal.

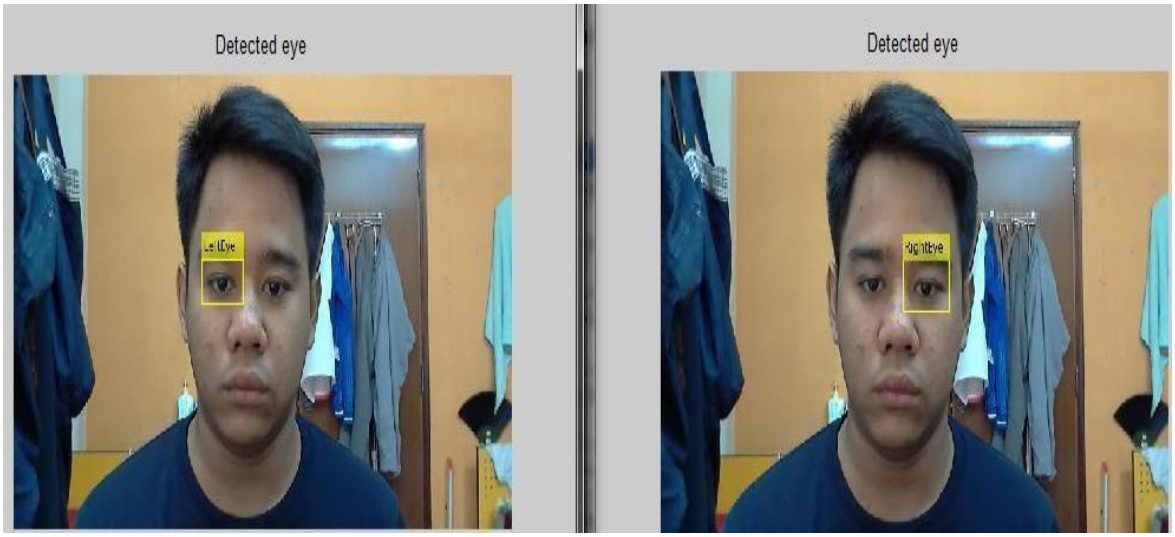

Figure : Result of Eyes Detection

Figure : False Negative Eyes Detection

- **Detecting Mouth**

Detecting mouth objective is to detect the drowsiness symptom which is yawning. To detect mouth, the cascade object detector that use Viola-Jones algorithm has been used by detecting object in rectangle shape.

The features to detect mouth area are as below:

1. Define and setup the cascade object detector using the constructor. The constructor uses built-in Viola-Jones algorithm to detect faces, noses, eyes, mouth and upper body.
2. Read the video or the image selected and run the face detector.
3. Draw the bounding box around the detected mouth. The bounding box is the area of the desired detection. But the algorithm to detect mouth is not as efficient as it detects other parts in the video as mouth.

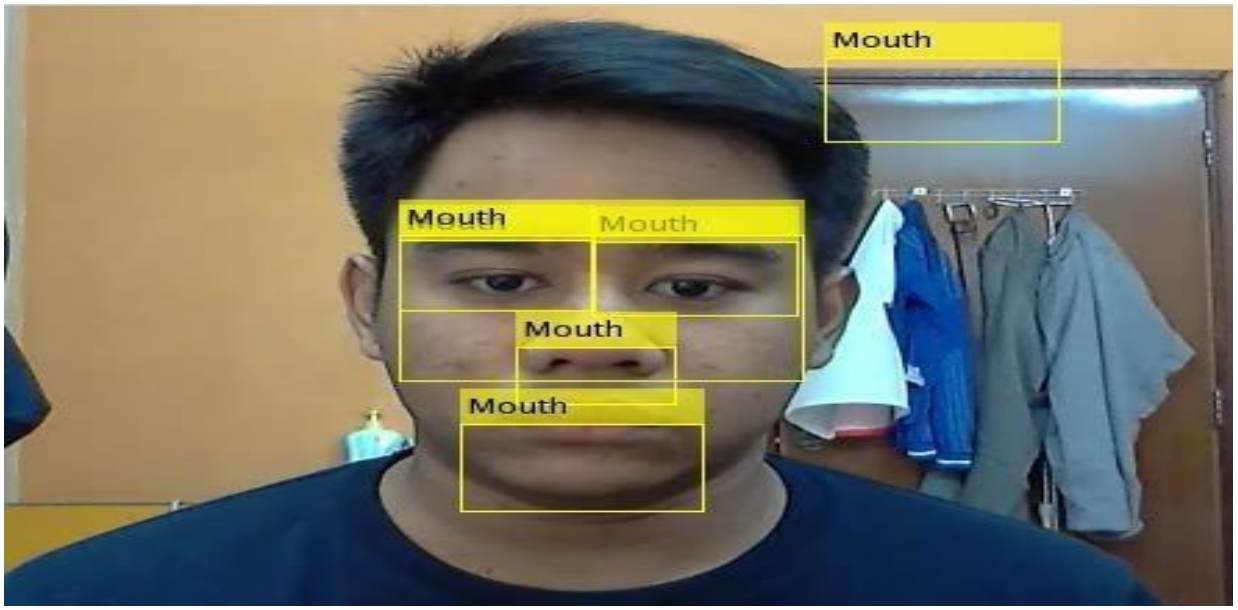

Figure : Fail Negative Mouth Detection

6.2 Results

- **Modelling**

After the algorithm has been developed during the experimentation part, it needs improvement for the system meet the objectives of the project. the flow process of the improved algorithm that has been developed. The algorithm successfully detects the eyes and mouth in the video; hence the result can be obtained.

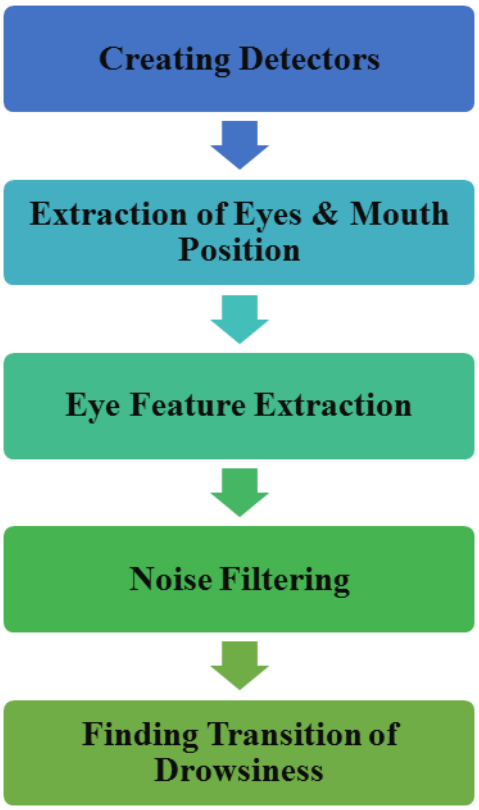

Figure : Flow Process of Detection Algorithm

- **Extraction of Eyes & Mouth Position**

After creating the detectors, bounding box of face, each eye and mouth areas is obtained. Each bounding box contains the parameters of the detection part. Cascade filters were used to extract the detected facial features by getting the centre gravity of the face bounding box. It is obtain by dividing the face bounding box (x, y) value by 2 and marked by red dash-line in the image . To require the True Positive detection, the centre gravity of face bounding box (x, y) value is used and it is divided by the algorithm to obtain the centre gravity of the face.

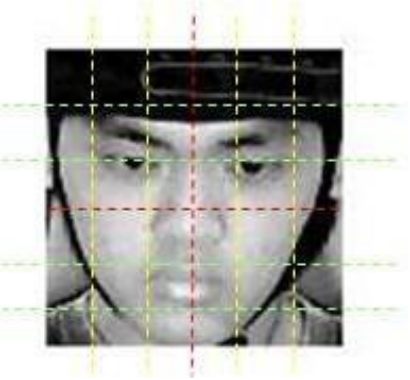

Figure : Cascade Filters of Face Image

The extraction is done to ensure the eyes detection bounding box are at the upper part of face and the mouth detection bounding box is below nose or at lower part of face.

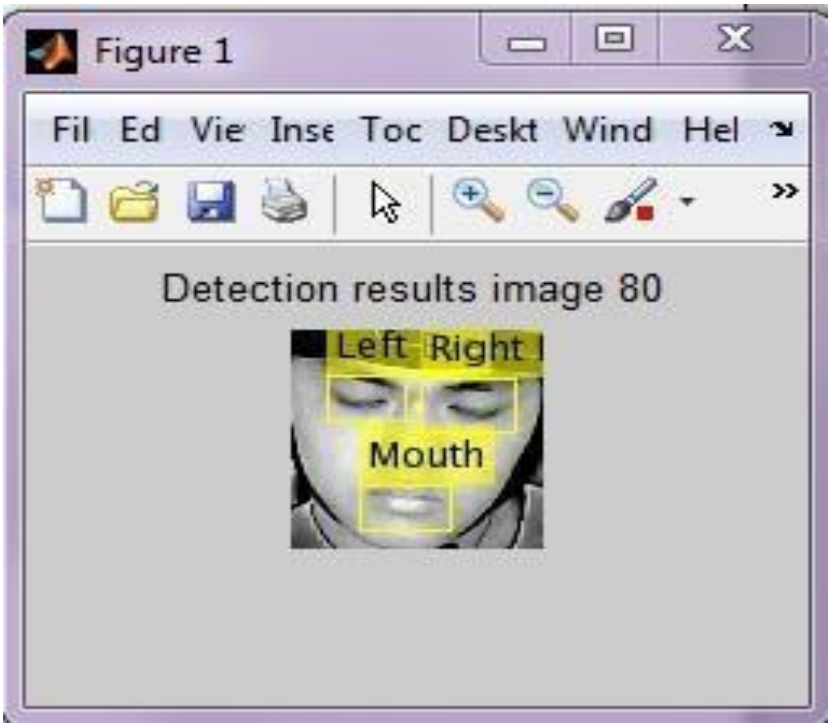

Figure : True Positive Eyes and Mouth Detection.

- **Noise Filter**

For data collection, median filter is used in the algorithm to filter the noise for the final graph is smooth. After that, global parameter method is used to get the transition from eye open to eye close, done by the product energy of each region. Lastly, the global parameter is derived by using DoG. The same methods are used in the algorithm to detect the opening of the mouth. the full process of the drowsiness detection system.

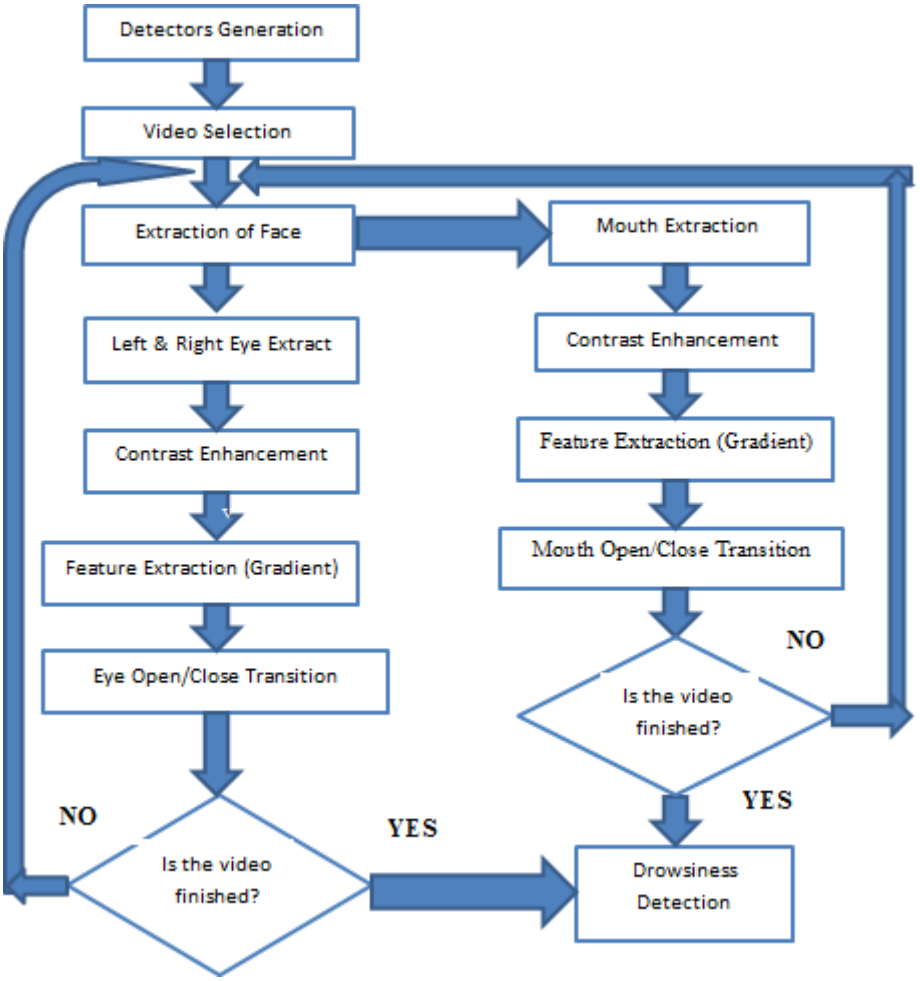

Figure : Flow Process of Drowsiness Detection System

- **Result:**

After completing the algorithm modelling, the simulation of algorithm was performed by using the video from the experiment conduct by MIROS. Due to the video length cannot be run in MATLAB® because the processor have not enough memory, the video were separated into 3 minutes each video. By doing that, each video contains not more than 400 images per video. There a 10 participants who complete the driving experiment but as to show in the result, we only take a participant.

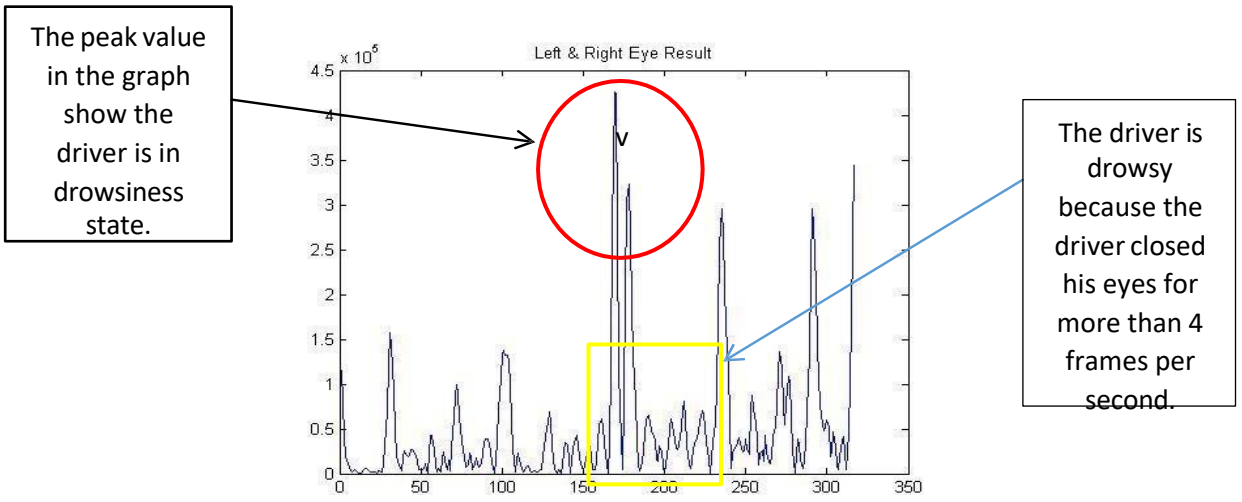

Figure : Drowsiness Detected in Graph

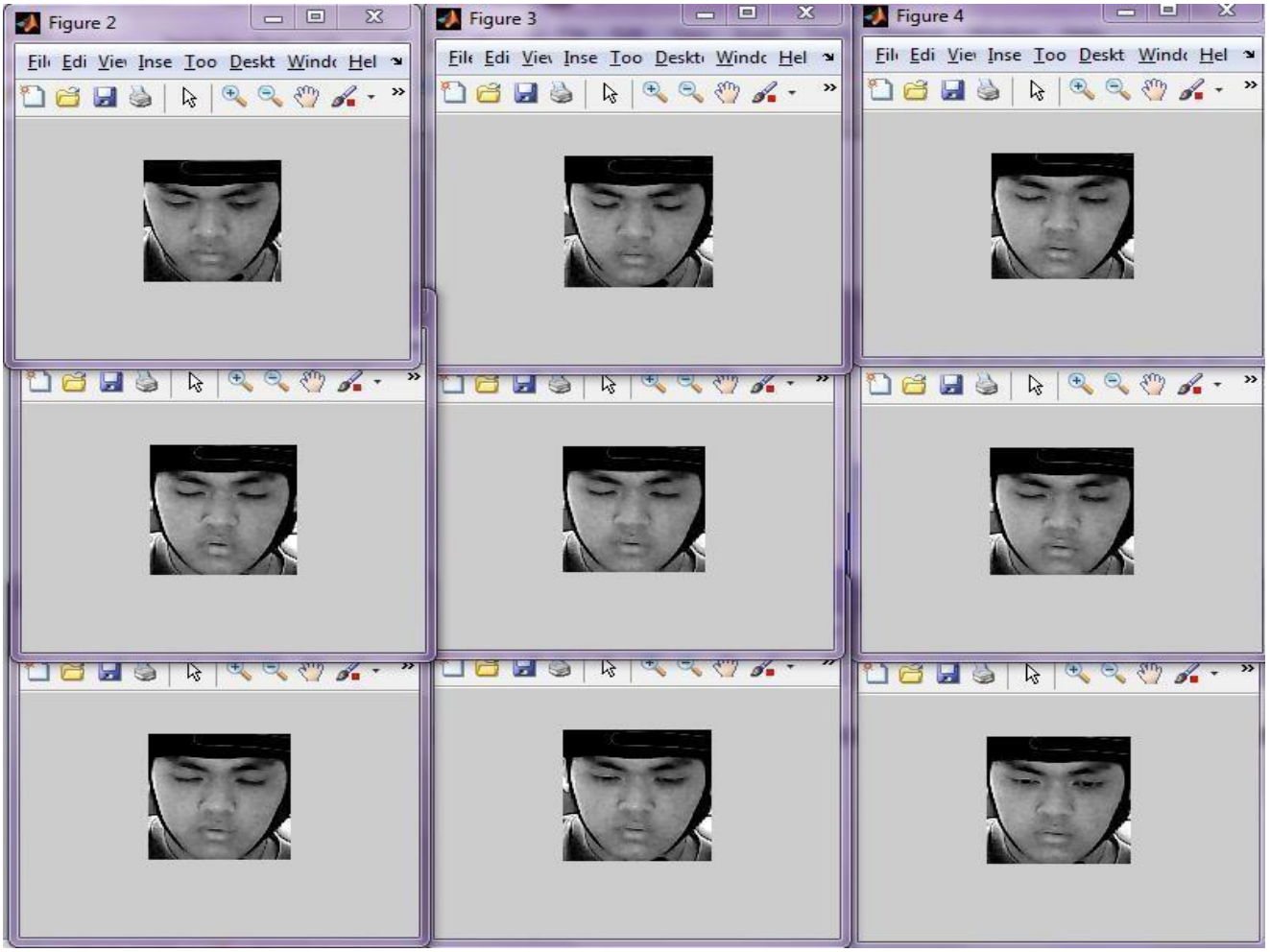

Figure : Transition Image of Driver Drowsy

6.3 Snapshots

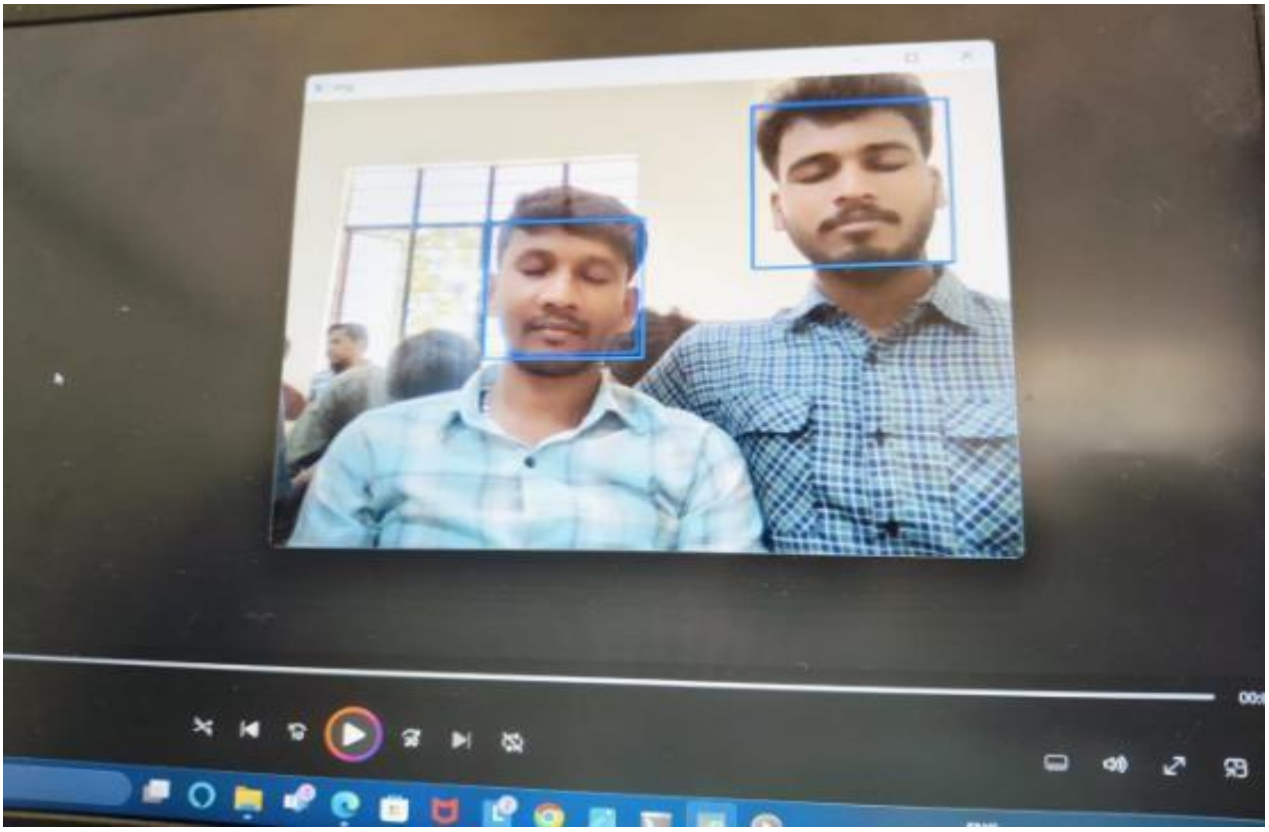

Figure : Result of Eyes Detection

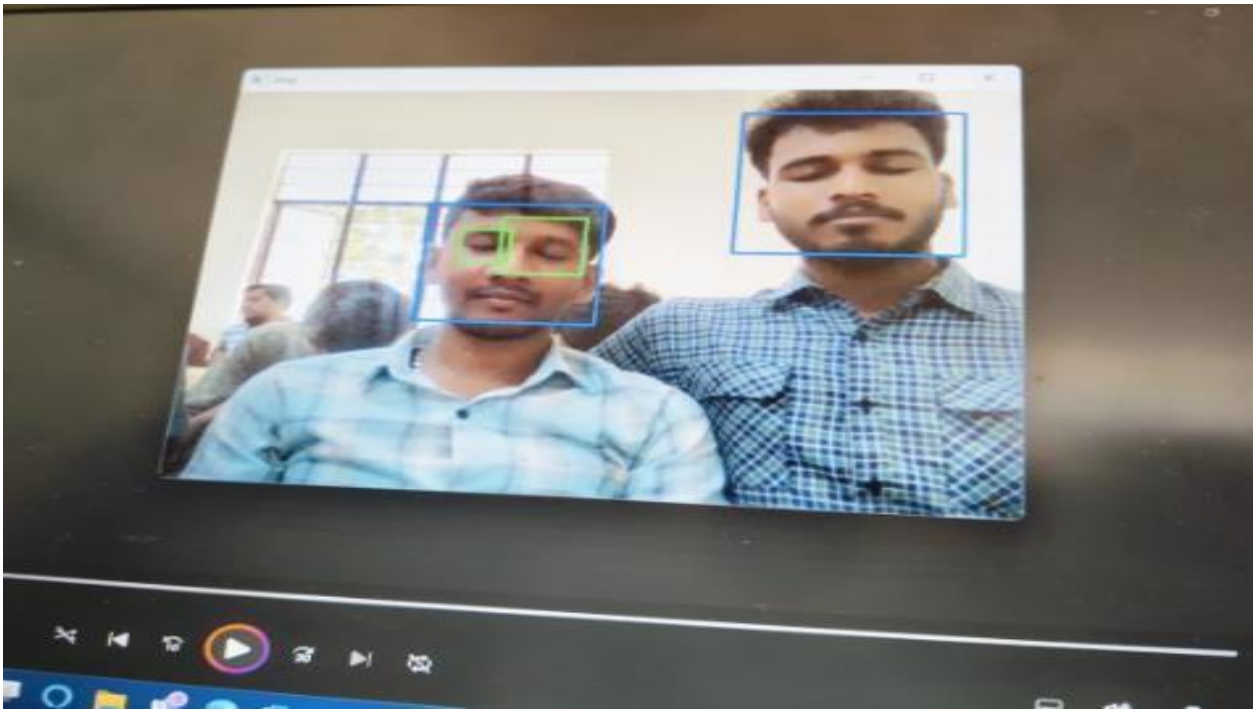

Figure : Result of Eyes Detection

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

Previously, the author focuses on developing the algorithm or command to detect drowsiness. The developments of the algorithm takes time due to the authors only have basic skill in using opencv2. The author learns about the opencv2 commands by developing the algorithm only with the help from Computer Vision Toolbox System that is already built-in in opencv2 software and also by trial and error of the shared file from the MathWorks MathWorks® is where all the high skills of MATLAB users from all over the world share their works on algorithms.

Investigating the drowsiness signs and collecting the data from the video of the experiments have been the main job scope. It will be used as parameters to develop the simulation system in detecting drowsiness. Until now, one of this semester's project objectives has been achieved which is to study the video images of participants in the experiment of driving simulation conducted by MIROS. Several techniques to develop the simulation system have been discovered. There are also other objectives that this project needs to achieve.

The author started developing the algorithm to detect the drowsiness. Few techniques have been implemented in this project which has been found through the previous researches. Further adjustment of the algorithm and the techniques need to be done in order to meet the requirement of this project and to finish it within the given time frame.

BIBLIOGRAPHY

[1] Z. Ahmad Noor Syukri, M. Siti Atiqah, L. Fauziana, and A. Abdul Rahmat, "MIROS crash investigation and reconstruction: annual statistical 2007-2010," 2012.

[2] A. Picot, S. Charbonnier, and A. Caplier, "On-line automatic detection of driver drowsiness using a single electroencephalographic channel," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, 2008, pp. 3864-3867.

[3] G. Borghini, L. Astolfi, G. Vecchiato, D. Mattia, and F. Babiloni, "Measuring neurophysiological signals in aircraft pilots and car drivers for the assessment of mental workload, fatigue and drowsiness," *Neuroscience & Biobehavioral Reviews*, 2012.

[4] B. T. Jap, S. Lal, P. Fischer, and E. Bekiaris, "Using EEG spectral components to assess algorithms for detecting fatigue," *Expert Systems with Applications*, vol. 36, pp. 2352- 2359, 2009.

[5] D. Liu, P. Sun, Y. Xiao, and Y. Yin, "Drowsiness Detection Based on Eyelid Movement," in *Education [1]Technology and Computer Science (ETCS), 2010 Second International Workshop on*, 2010, pp. 49-52.

[6] H. Seifoory, D. Taherkhani, B. Arzhang, Z. Eftekhari, and H. Memari, "An Accurate Morphological Drowsy Detection," ed: IEEE, 2011.

[7] D. J. McKnight, "Method and apparatus for displaying grey-scale or color images from binary images," ed: Google Patents, 1998.

[8] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," *ACMTransactions on Graphics*, vol. 21, pp. 277-280, 2002.

[9] I. Garcia, S. Bronte, L. Bergasa, N. Hernandez, B. Delgado, and M. Sevillano, "Vision- based drowsiness detector for a realistic driving simulator," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, 2010, pp. 887-894.

[10] T. Danisman, I. M. Bilasco, C. Djeraba, and N. Ihaddadene, "Drowsy driver detection system using eye blink patterns," in *Machine and Web Intelligence (ICMWI), 2010 International Conference on*, 2010, pp. 230-233.

[11] D. F. Dinges and R. Grace, "PERCLOS: A valid psychophysiological measure of alertness assessed by psychomotor vigilance," *Federal Highway Administration. Officeof motor carriers, Tech. Rep. MCRT-98-006*, 1998