**IQRA National University, Peshawar**


# Driving Negligence Dissuader System

***By***

**Mr. Hassan Mehdi** **INU/SP19 BCS 15453/PES**

***Supervisor***

**Ms. Samavia Tariq**

***Bachelor of Science in Computer Science (2019-2023)***

**The candidate confirms that the work submitted is their own and appropriate credit has been given where reference has been made to the work of others.**


## IQRA National University, Peshawar

# Driving Negligence Dissuader System

**A project presented to**

## IQRA National University, Peshawar

**In partial fulfillment**

**of the requirement for the degree of**

## *Bachelor of Science in Computer Science (2019-2023)*

**By**

**Mr. Hassan Mehdi INU/SP19 BCS 15453/PES**

# DECLARATION

I hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely based on our efforts. If any part of this project is proved to be copied out from any source or found to be the reproduction of some other. I will stand by the consequences. No portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Hassan Mehdi

# CERTIFICATE OF APPROVAL

It is to certify that the final year project of **BS (CS)** "**D**riving **N**egligence **D**issuader **S**ystem (**DNDS**)" was developed by **Mr. Hassan Mehdi (INU/091-19-115453 BS(CS) 15453/PES)** under the supervision of "**Ms. Samavia Tariq**" and that in her opinion; it is fully adequate, in scope and quality for the degree of Bachelor of Science in Computer Science.

---

**Supervisor**

---

**External Examiner**

---

**Head of Department**

**(Department of Computer Science)**

# Executive Summary

Many people die each year in roadway departure crashes caused by driver inattention. Traffic accidents affect countless lives every year. Most are caused by driver errors. The **N**ational **H**ighway **T**ransportation **S**afety **A**dministration (**NHTSA**) found that between **94%** and **96%** of all motor vehicle accidents are caused by some type of human error. For many decades, the road safety system was not among the major concerns. Though, it arguably is one of the imperative requirements that could avoid many traffic mishaps due to human negligence. When a driver is subject to changing and complicated driving tasks in traffic, they should be able to ensure driving authority to prevent potential hazards and mishaps. Thus, there is a dire need for a road safety system. A system that ensures the safety of drivers and vehicles on the road. In this context, this report presents a road safety recommendation system called **D**river **N**egligence **D**issuader **S**ystem (**DNDS**). This intelligent system has the ability to detect drowsiness, yawning, lane lines, and objects on the road and guarantees that the driver is paying their full attention to the road up ahead. **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) is a vehicle safety recommendation system that monitors a driver's facial behavior to detect the driver's drowsiness and yawning. The system also monitors the road in front to detect the road lanes, the lane curvature, and the vehicle center offset from lanes. In addition to lanes and drowsiness, the system also detects different objects on the road such as humans, animals, and other vehicles, etc. The system is built using **O**pen **C**omputer **V**ision **L**ibrary (**OpenCV**) to detect lanes. **H**aar **C**ascade **C**lassifiers (**HCC**) and **DLib** Shape predictor to detect drowsiness and yawning. And the **YoloV5** deep learning model trained on coco classes to detect the objects in front of the vehicle. The results show that the **D**rowsiness **D**etection **S**ystem (**DDS**), **Y**awning **D**etection **S**ystem (**YDS**), **L**ane **D**etection **S**ystem (**LDS**), and **O**bject **D**etection **S**ystem (**ODS**) are working with reasonable accuracy and effectiveness.

# Acknowledgment

Praise is due to Allah whose worth cannot be described by speakers, whose bounties cannot be counted by calculators, and whose claim (to obedience) cannot be satisfied by those who attempt to do so, whom the height of intellectual courage cannot appreciate, and the divings of understanding cannot reach; He for whose description no limit has been laid down, no eulogy exists, no time is ordained and no duration is fixed. He brought forth creation through His Omnipotence, dispersed winds through His Compassion, and made firm the shaking earth with rocks. All praise is to Him who bestowed upon me a minute portion of His boundless knowledge through which I was able to accomplish this challenging task.

I would like to express my sincere gratitude to several individuals and organizations for supporting me throughout my Undergraduate studies. Firstly, I wish to express my sincere gratitude to my respected supervisor, **Ms. Samavia Tariq**, for her enthusiasm, patience, insightful comments, helpful information, practical advice, and unceasing ideas that have helped me tremendously at all times in my project and writing of this report. Her immense knowledge in the field of Computer Science has enabled me to complete this project successfully. Without her support and guidance, this project would not have been possible.

I am also grateful to the following university staff: **HOD** of Computer Science **Dr. Atif Ishtiaq**, Coordinators of the Department of Computer Science **Sir. Muhammad Shakeel** and **Sir. Khalid Hamid** and Coordinator of the Final Year Projects **Sir. Shahab ul Islam** for their constant support and well wishes. They have been a great source of inspiration.

Most of all, I am most thankful to my beloved parents and family; they have been a constant source of support and encouragement for me, and brought me the values of honesty & hard work. Especially my brother **Mr. Asghar Abbas**, and my sisters **Dr. Shaista Parveen** and **Ms. Fatima Batool**. Their formal and informal support has enabled me to complete my studies and achieve my most desired goals.

Hassan Mehdi

# Abbreviations

| | |
|---|---|
| **DNDS** | Driving Negligence Dissuader System |
| **DDS** | Drowsiness Detection System |
| **YDS** | Yawning Detection System |
| **LDS** | Lane Detection System |
| **PDS** | Pedestrian Detection System |
| **ODS** | Object Detection System |
| **VCO** | Vehicle Center Offset |
| **CR** | Curve Radius |
| **CV** | Computer Vision |
| **NHTSA** | National Highway Transportation Safety Administration |
| **OpenCV** | Open Computer Vision |
| **HCC** | Haar Cascade Classifier |
| **DLib** | Digital Library |
| **YoloV5** | You Only Look Once Version 5 |
| **EEG** | Electroencephalographic |
| **EOG** | Electrooculographic |
| **IMU** | Inertial Measurements Units |
| **EAR** | Eye Aspect Ratio |
| **ECR** | Eye Close Ratio |
| **MAR** | Mouth Aspect Ratio |
| **SSD** | Single Shot Multi-Box Detector |
| **CNN** | Convolutional Neural Networks |
| **RNN** | Recurrent Neural Network |
| **RCNN** | Regions with Convolutional Neural Networks |
| **CRCNN** | Cascade Regions with Convolutional Neural Networks |
| **SCNN** | Spatial Convolutional Neural Networks |
| **RFCN** | Region-based Fully Convolutional Network |
| **FPN** | Feature Pyramid Network |
| **NAS** | Neural Architecture Search |
| **NAS-FPN** | Neural Architecture Search Feature Pyramid Network |

| | |
|---|---|
| **IoT** | Internet of Things |
| **IoV** | Internet of Vehicles |
| **GPS** | Global Positioning System |
| **SUV** | Sports Utility Vehicle |
| **PULP** | Parallel Ultra Low Power |
| **NSF** | National Sleep Foundation |
| **YawDD** | Yawning Detection Dataset |
| **NTHU-DDD** | NTHU Driver Drowsiness Detection |
| **ADAS** | Advanced Driver Assistance Systems |
| **DMS** | Driver Monitoring System |
| **GPU** | Graphics Processing Unit |
| **SLLF** | Sobel Lane Line Fitting |
| **CP** | Cubic Polynomial |
| **SC** | Spline Curve |
| **AC** | Arc Curve |
| **OSM** | One Stage Methods |
| **OSA** | One Stage Algorithms |
| **TSM** | Two Stage Methods |
| **TSA** | Two Stage Algorithms |
| **SDD** | Software Design Description |
| **RAD** | Rapid Application Development |
| **RoI** | Region of Interest |
| **CED** | Canny Edge Detection |
| **SWS** | Sliding Window Search |
| **SPP** | Special Pyramid Pooling |
| **PANet** | Path Aggregation Network |
| **IOU** | Intersection Over Union |
| **BDD** | Berkeley Deep-Drive |
| **CSPC** | Cross Stage Partial connections |

# Table of Contents

# List of Figures

# List of Tables

# List of Equations

# Chapter 1
# Introduction

## 1 Introduction

Road traffic damages cause substantial economic losses to individuals, their families, and nations as a whole. These losses stem from the expense of treatment as well as lost productivity for those killed or disabled by their injuries. And form family members who need to take time off work or academy to care for the injured. Road traffic collisions cost most countries **3.0%** of their **g**ross **d**omestic **p**roduct (**GDP**). The Lives of approximately **1.3M** people are cut short as a result of road traffic accidents every year [1]. Between **20M** and **50M** more people suffer from non-fatal injuries, with many incurring a disability as a result of their injury. The **N**ational **H**ighway **T**ransportation **S**afety **A**dministration (**NHTSA**) found that somewhere between **94%** and **96%** of all motor vehicle accidents are caused by some type of human error [2]. According to the **N**ational **S**leep **F**oundation (**NSF**), **41%** of drivers surveyed admitted that they have fallen asleep at the wheel at some point in their lives, and **10%** reported that they have done it [3] in the past year. In this paper, an IoT-based intelligent system called **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) is proposed that uses **O**pen **C**omputer **V**ision **L**ibrary (**OpenCV**), **DLib** Shape predictor, **H**aar **C**ascade **C**lassifiers (**HCC**), **YoloV5 D**eep **L**earning (**DL**) model trained on coco classes to detect drowsiness, yawning, road lanes, and objects. The results show that the **D**rowsiness **D**etection **S**ystem (**DDS**), **Y**awning **D**etection **S**ystem (**YDS**), **L**ane **D**etection **S**ystem (**LDS**), and **O**bject **D**etection **S**ystem (**ODS**) are working with reasonable accuracy and effectiveness.

### 1.1 Brief

The **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) is a combination of multiple intelligent systems (***D**rowsiness **D**etection **S**ystem (DDS), **Y**awning **D**etection **S**ystem (YDS), **L**ane **D**etection **S**ystem (LDS), **O**bject **D**etection **S**ystem (ODS)*) to work as a standalone powerful vehicle safety recommendation system. The individual systems are discussed further below.

### 1.1.1 Drowsiness Detection System (DDS)

Drowsiness detection is an important factor in road safety; in manual driving as well as in future semi-automated driving. In manual driving, about **20%** to **30%** of fatal road accidents are reported to be attributable to driver drowsiness. Two general approaches could be considered to detect driver drowsiness. Intrusive and non-intrusive. In intrusive approaches, the drowsiness state is analyzed using the processing of physiological outputs such as **E**lectroencephalographic (**EEG**) and **E**lectrooculographic (**EOG**) information [7]. High performance for drowsiness detection is achieved using intrusive methods but driver movements can negatively affect the reliability of the designed system. Non-intrusive methods provide an estimation for drowsiness using the facial features of drivers [8, 9] or vehicle-based measurements [10]. On one hand, these procedures have some imperfections. For instance, various lighting conditions can have disrupting effects on the performance of the system. On the other hand, vehicle-based measures reminisce the driving behavior, and therefore, can provide an acceptable estimation for drowsiness detection in an unobtrusive way [11]. Gas and brake pedal inputs, changes in vehicle speed, steering wheel angle and velocity signals, and lateral distance of the vehicle from the center of the line are typically used as vehicle-based measurements to detect drowsiness [12]. The **D**rowsiness **D**etection **S**ystem (**DDS**) employs the measurement of **E**ye **A**spect **R**atio (**EAR**), to determine if the driver is drowsy or not.

### 1.1.2 Yawning Detection System (YDS)

The **Y**awning **D**etection **S**ystem (**YDS**) is a complementary system to the **D**rowsiness **D**etection **S**ystem (**DDS**). Driver fatigue not only impacts the alertness and response time of the driver but also increases the chances of being involved in car accidents. Drowsiness is correlated with a variety of physiological variables, such as eye closing, yawning, head movements, pulse rate, eye twitch rate, etc. A warning sign that can be measured as an indication of driver drowsiness is yawning. Yawning detection is very important for the safety purpose of drivers as it will let the driver know if they are getting drowsy. Driving at that moment may not be safe. This is often ignored by the driver. Therefore, the use of assisting systems like yawning detection is crucial to increase overall system performance and prevent false positives, and in turn, prevent road accidents. These systems should then alert the driver in the case of drowsiness or yawning. Several automatic yawning detection techniques have been developed for the driver's drowsiness monitoring system. In almost all systems, where yawning is detected as a fatigue symptom, the recognition methods are built similarly. The camera (one or sometimes two) registers a head/face image. Using image analysis, the mouth

is recognized and its state (closed or open) is detected. In the case of an open state of mouth, we have to decide if it is yawning. Nevertheless, correctly detecting the yawning of the driver and predicting exhaustion in real-time situations is still a crucial challenge. The **Y**awning **D**etection **S**ystem (**YDS**) employs the measurement of **M**outh **A**spect **R**atio (**MAR**) to determine if the driver is yawning or not.

### 1.1.3 Lane Detection System (LDS)

**L**ane **D**etection **S**ystems (**LDS**) are useful in avoiding road accidents as safety is the main purpose of these systems. Lane detection is the process to locate lane markers on the road and then present these locations to an intelligent system. Such systems have the goal to detect lane marks and warning the driver in case the vehicle tends to depart from the lane. A lane detection system is an important element of many intelligent transport systems. Lane detection is a challenging task because of the varying road conditions that one can come across while driving. In the past few years, numerous approaches for lane detection were proposed and successfully demonstrated. With the rapid rise of urban traffic, traffic safety becomes more and more significant. Leaving the lane cause about **30%** of all accidents on the highway, and most of these result from the distraction and fatigue of the driver. Therefore, a system that could provide a warning to drivers of danger has great potential to save a large number of lives. Lane detection technology is mainly based on image processing algorithms to extract the features of lane lines, reduce the image channels, perform gray processing on the original image, and then use **The** **C**anny **A**lgorithm or **S**obel algorithm to edge the grayed image, extract some features of the acquired image, and then perform **L**ane **L**ine **F**itting (**SLLF**) after extracting the lane. Common lane fitting models include the **C**ubic **P**olynomial (**CP**), the **S**pline **C**urve (**SC**), and the **A**rc **C**urve [18–19]. In many proposed systems, lane detection consists of the localization of specific primitives such as road markings on the surface of the painted roads [13]. Various challenges like parked and moving vehicles, bad quality lines, shadows of trees, buildings, and other vehicles, sharper curves, irregular lane shapes, merging lanes, writings, and other markings on the road, unusual pavement materials, and dissimilar slopes cause problems in lane detection. The applications of a **L**ane **D**etection **S**ystem (**LDS**) could be as simple as pointing out lane locations to the driver on an external display, to more complex tasks such as predicting lane change in the instant future to avoid collisions with other vehicles. Keeping road safety in mind, the **L**ane **D**etection **S**ystem (**LDS)** uses **O**pen **C**omputer **V**ision **L**ibrary (**OpenCV**) to detect lane lines and **V**ehicle **C**enter **O**ffset (**VCO**).

### 1.1.4 Object Detection System (ODS)

**O**bject **D**etection **S**ystem (**ODS**) is an important part of the road safety system. To ensure the safe running of vehicles, real-time and accurate detection of all objects especially pedestrians on the road is of paramount importance. Object detection is the task of detecting instances of objects of a certain class within an image. This state-of-the-art method can be categorized into two main types. **O**ne **S**tage **M**ethods (**OSM**) and **Two-Stage M**ethods (**TSM**). **O**ne **S**tage **M**ethods (**OSM**) prioritize inference speed, and example models include **YOLO**, **SSD,** and **RetinaNet**. **T**wo **S**tage **M**ethods (**TSM**) prioritize detection accuracy, and example models include Faster **RCNN**, Mask **RCNN,** and **C**ascade **RCNN**. **T**wo **S**tage **A**lgorithms (**TSA**) usually have high accuracy but have a relatively slow detection speed. **O**ne **S**tage **A**lgorithms (**OSA**), such as **SSD** [16] and **YOLO** [17], perform classification and regression in just one stage. These methods generally have a low accuracy but a high detection speed. The main task of this system in our project is to accurately and quickly detect the vehicles, pedestrians, traffic lights, traffic signs, and other objects in front of the vehicles, to ensure safety in driving. The problem with most object detection algorithms is that large objects are easily detected, while small objects are often ignored by the detector. It is extremely dangerous to miss pedestrians, traffic lights, and traffic signs. In recent years, there are many feature fusion algorithms for small object detection [14–15]. The **O**bject **D**etection **S**ystem (**ODS**) uses the deep learning model **YoloV5** nano trained on coco classes to detect the object. The model is trained to detect upwards of **88** different objects including, humans and vehicles.

## 1.2 Relevance to Course Modules

The **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) project is related to multiple of our courses studied in the **BC(CS)** program. The individual courses equipped us with the necessary knowledge and skills to complete a portion and sometimes multiple portions of the project. Various statistical tasks and evaluation of the outcomes and results of the different systems included in the project are related to subjects such as **Probability and Statistics**. The basic understanding of artificial intelligence, machine learning, and deep learning in this project is related to the **Artificial Intelligence** course, which is the core part of this project. The **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) heavily depends on digital graphics and image processing, as the main function of this project is to process the incoming images from camera modules, process these images according to the needs of the used algorithms that make predictions and draw on the locations of where the predictions are happening. This part of the project is related to subjects such as **Computer Graphics** and **Digital Image Processing**. The programming languages and the coding structure used in this project are related to subjects like **Programming Fundamentals**, **Object Oriented Programming,** and **Modern Programming**. Coding is not everything. Using the correct coding techniques and achieving the desired task most efficiently and optimally is very important. Since our project is an IoT-based intelligent system that will be installed in vehicles for real-time detections, it is important to minimize the time complexity of the code as much as possible to save power and increase the performance of the system. These code optimization techniques and time complexity evaluations of the code are related to courses such as **Data Structures** and **Design & Analysis of Algorithms**. Every project is built with planning and preparations. The different preparation processes, the software development techniques, the software development life cycle, and the software development methodology used for this project are all related to **Software Engineering** and **Object-Oriented Analysis & Design** courses. Some data acquisition and processing techniques used in the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) are related to the subject of **Data Mining**. All of the mentioned subjects were of great help during the project planning, development, testing, and evaluation phases. The courses gave us a basic understanding of the different concepts and ideas used in this project. Further exploration and research in the fields improved and strengthen our knowledge base further, which allowed us to accomplish the challenging tasks of this project in the desired timeframe.

## 1.3 Project Background

**A D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) is the combination of multiple intelligent systems (***D**rowsiness **D**etection **S**ystem (**DDS**), **Y**awning **D**etection **S**ystem (**YDS**), **L**ane **D**etection **S**ystem (**LDS**), **O**bject **D**etection **S**ystem (**ODS**)*) to work as a standalone powerful vehicle safety system.

### 1.3.1 Drowsiness Detection System (DDS)

The phrase 'driver drowsiness detection' is the general term for systems that monitor driver attention. Drowsiness detection works to prevent accidents created by microsleep, fatigue, and lack of attention. Driver **D**rowsiness **D**etection **S**ystems (**DDS**) generally come as one tool, one part of **A**dvanced **D**river **A**ssistance **S**ystems (**ADAS**). These are various programs and technologies designed to make driving safer and lessen the chances of human error resulting in catastrophic road traffic incidents. These can range from warning drivers if there's something in their blind spot, to automatic emergency braking. Driver **D**rowsiness **D**etection **S**ystems (**DDS**) can use cameras, eye tracking sensors, and other hardware to monitor visual cues, where drowsiness can be detected through yawning frequency, eye-blinking frequency, eye-gaze movement, head movement, and facial expressions. The systems can also monitor driving input behavior to notice when there are erratic steering movements, pedal use, and lane deviations. Once the system has established that the driver's movements are erratic and they don't seem alert, other factors may then be taken into account. For example, the drowsiness detector might analyze the speed of the car, time of day, weather conditions, and so on. After doing so, it can calculate the 'tiredness index' of the driver and begin to take action, such as making an audible noise to alert the driver. Different car models have different systems, but in most cases, the driver will be alerted to their potential drift of attention with some sort of noise, or vibration of the steering wheel or seat. The system may also remind the driver to take a break, especially if they have been on the road for an extended period. When we have fully autonomous cars, we will be able to put our worries about drowsiness to one side. Until then, drowsiness at the wheel can be a fatal mistake that the **D**rowsiness **D**etection **S**ystems (**DDS**) works to prevent.

### 1.3.2 Yawning Detection System (DDS)

**Y**awning **D**etection **S**ystems (**YDS**) work as complementary systems to the **D**rowsiness **D**etection **S**ystem (**DDS**). **Y**awning **D**etection **Systems** (**YDS**) and **D**rowsiness **D**etection **S**ystems (**DDS**) sometimes come as a single intelligent system since the idea is to detect drivers' micro-sleeps and fatigue. **Y**awning **D**etection **S**ystem (**YDS**) also uses cameras, lips tracking sensors, and other hardware to monitor visual cues to detect yawning.

### 1.3.3 Lane Detection System (LDS)

**L**ane **D**etection **S**ystem (**LDS**) is designed to help drivers avoid crashes due to drifting or departing from their lane. These systems detect lane markers and alert the drivers when the vehicle drifts off to the side of the lane. The warning is usually a flashing indicator and sometimes it beeps from the corresponding side. In some systems, the steering wheel or driver's seat vibrates gently. Generally, **L**ane **D**etection **S**ystem (**LDS**) may not alert you when your turn signal is on. Despite the apparent simplicity of the white markings on a dark road, making it is very difficult to identify the markings on different types of roads. These difficulties are of an occlusion in the shadow of other vehicles, changes in the roadway itself, and different types of road markings. A lane detection system must collect all types of markers on the roads and give a reliable estimate of the path of the vehicle's position. Lane detection plays an important role in **D**river **A**ssistance **S**ystems (**DAS**). Lane detection algorithms detect lane markings and the edges of the road and estimate the vehicle position in the lane. Lane detection provides a framework for the support of many other single-camera-based Mobil eye functions as vehicle detection; in this case, it contributes to the correct position of the vehicle in the same lane. Provided that the road markings are visible and that their testimony is not hindered by the presence of clutter, acknowledge shadows, rain, snow, or other disturbances on the road. The **L**ane **D**etection **S**ystem (**LDS**) uses a camera to recognize lane markers. If the system detects that the car is too close to the left or right lane markings, and the vehicle turn signal is not on, a warning light, a vibration, or sound is activated to warn the driver. Sometimes the **L**ane **D**etection **S**ystems (**LDS**) is also able to control the vehicle and these systems will for a brief moment control the steering functionality and current the direction of the vehicle while alerting the driver of the situation.

### 1.3.4 Object Detection System (ODS)

Object detection and tracking is the main concept in advanced computer vision. It is one of the critical components to support autonomous driving. Autonomous vehicles rely on the perception of their surroundings to ensure safe and robust driving performance. This perception system uses object detection algorithms to accurately determine objects such as pedestrians, vehicles, traffic signs, and barriers in the vehicle's vicinity. Deep learning-based object detectors play a vital role in finding and localizing these objects in real-time. While it’s not entirely new, this feature is now fine-tuned to detect two and four-legged creatures, as well as bicyclists and bike riders. Initially, object detection could only recognize other vehicles, ensuring drivers didn’t sideswipe those in blind spots when trying to switch lanes. Researchers drive cars too though and know sideswiping cars is the least of driving worries when compared to others. The steady frequency of unsafe roads abounds, so researchers sought ways to avoid other things on the road that cause accidents. This kind of technology has been used in things like video surveillance and special effects for years, but it’s come a long way in cars. Using radar and cameras, object detection systems merge information and send a signal to the car, which then initiates a sequence of avoidance strategies. Some systems may incorporate a flashing notification on an in-car screen or use a loud noise to get the driver’s attention. Other systems automatically tighten a driver’s seatbelt and apply brakes. In many cases, it will also detect dogs, cats, rabbits, and other animals. Object detection is a well-researched area of computer vision with applications for autonomous vehicles and driver assistance systems. This blooming research in this area is partly due to human and economic losses from traffic accidents caused, the availability of possible technology accumulated over the last 30 years of research in computer vision, and the exponential growth of processor speed that has paved the way for the operation.

## 1.4 Literature Review

As accidents due to human neglect represent a significant portion of vehicle crashes in the world, researchers and automotive companies have considered different solutions ranging from finding patterns in the driving habits of the drivers to analyzing brain waves and vitals of the driver while driving. Most of these solutions are backed by predictive algorithms powered by statistics and machine learning.

### 1.4.1 Drowsiness Detection System (DDS)

The most common drowsiness detection techniques can be broadly divided into three categories as described in the following paragraphs.
One approach to detect if the driver is drowsy is to find changes in vehicle behavior, such as the one proposed by McDonald et al [20]. He created a contextual and temporal algorithm that utilizes the steering angle, vehicle speeds, and accelerator pedal positions. These values are passed into a Bayesian Network which determines if a driver shows characteristics of drowsy behavior. The algorithm was found to have lower false-positive rates than **PERCLOS** [21] methods, which predict drowsiness based on eyelid movements and patterns. The takeaway from this study was that to predict correctly, the context of the situation is crucial. The data that it captures over a previous 10-second period is vital in understanding whether the person is at risk of drowsiness related to lane departures.
A second approach is based on studies focused on using the driver's vitals, brain waves, and readings from **E**lectroencephalograms (**EEGs**) to make predictions. Wei et al [22] made comparisons between non-hair bearing **EEG B**rain-**C**omputer **I**nterfaces (**BCI**) that are easy to wear and less intrusive than the lab-based whole scalp **EEG** which are less comfortable. The study showed that non-hair bearing devices had no significant reduction in performance when compared to whole scalp **EEG**. Thereby with this finding one may develop less intrusive and comfortable headbands. **EEG** alone is unable to detect all stages of drowsiness, so Kartsch et al [23] used **EEG** with **I**nertial **M**easurements **U**nits (**IMU**) sensors to detect **5** levels of drowsiness with about **95%** accuracy. The team fused behavioral information from **IMU** and **EEG** information to detect drowsiness. Another drawback of the **EEG** system was the power requirements of these devices. Their technology has also facilitated the implementation of a **P**arallel **U**ltra-**L**ow **P**ower (**PULP**) platform on a microcontroller which extended the battery life to almost **46** hours, thereby creating devices that are always wearable and require low maintenance. Tateno et al [24] developed a system that just uses heart rate monitoring to detect

the respiration of a person and thereby detect drowsiness. The methodology was found to be an effective predictor of respiration and thereby drowsiness.
Yet another technique is to utilize the power of computer vision. The recent breakthroughs in Deep Learning have provided new tools for computer vision for detection and classification. Computer vision-related applications utilize these methods in object detection, health and wellness, and even agricultural applications too [25]. A big impact in this space has been on imaging data. Since a driver's facial features change significantly once he gets tired, computer vision scientists have attempted to capitalize on this and use it to provide solutions for drowsiness detection. Tayab Khan et al [26] proposed a solution to measure the angle of eyelid curvature and thereby identify if the eyes are closed or not. They achieved **95%** accuracy with this method, but the limitation is that there needs to be enough light for the method to work as it functions poorly at night. Shakeel et al [27] used **MobileNet-SSD** architecture to train a custom dataset of **350** images. The model was capable of achieving a Mean Average Precision of **0.84**. The system was cost-effective and efficient as the algorithm could be deployed on an Android device and the camera stream could be classified in real-time. Celona et al [28] proposed a vision-based Multitask Driver Monitoring Framework that analyzes the eyes, mouth, and the pose of the head simultaneously to predict the level of drowsiness. This study was conducted on the **NTHU** [29] dataset. Another study conducted by Xie et al [30] used transfer learning and sequential learning from yawning video clips to detect yawning on the **YawDD** and **NTHU-DDD** databases. This system was able to have higher precision and was robust to changes in the position and angle of the face to the camera. Mehta et al [31] developed an Android application that is capable of detecting facial landmarks and then computing the **E**ye **A**spect **R**atio (**EAR**) and **E**ye **C**losure **R**atio (**ECR**) to predict driver's drowsiness based on machine learning models with an accuracy of **84%**. In this project, we went with determining the **E**ye **A**spect **R**atio (**EAR**), with the help of **H**aar-**C**ascade-**C**lassifier (**HCC**) and **DLib**-**S**hape-**P**redictor to detect drowsiness.

### 1.4.2 Yawning Detection System (YDS)

The project uses similar principles to **D**rowsiness **D**etection **S**ystem (**DDS**) to detect yawning. The system determines **M**outh **A**spect **R**atio (**MAR**), with the help of a **H**aar-**C**ascade-**C**lassifier (**HCC**) and **DLib**-**S**hape-**P**redictor to detect yawning.

### 1.4.3 Lane Detection System (LDS)

The current methods for lane detection can be divided into three main categories: road-based models, road-based features, and neural network-based models. The detection method based on the road model mainly abstracts the lane lines into geometric shapes such as straight lines, curves, parabolas, and splines, and uses different two-dimensional or three-dimensional models to determine each model parameter.
Wang et al [32] first located the initial area of the lane line and then converted the lane line detection problem into the problem of determining the spline model in the initial area. Tan et al [33] proposed a robust curve lane detection method, which uses the improved river method to search for feature points in the far field of view, guided by a straight line detected in the near field or the curve of the last frame, and can connect dotted lane marks or fuzzy lane marks. Shin et al [34] used parallel features with a constant distance between the left and right lane lines in the top view, and use parallel lines to detect and track lane lines. The above method has high accuracy for detecting a specific lane line but needs to select an appropriate model and parameters. One model often has difficulty coping with multiple road scenarios, and its generalization ability and real-time performance are poor. The detection method based on road features uses lane line and environment differences to extract lane line features for recognition and segmentation. The main features include lane line edges, color, and geometric features. Yoo et al [35] proposed a gradient-enhanced conversion method for robust lane detection, which converts **RGB** space images into grayscale images and combines The Canny's algorithm to generate large gradients at the boundaries. Son et al [36] realized lane line detection by extracting white and yellow lane lines respectively. Jung et al [37] proposed an effective method for reliably detecting lanes based on spatiotemporal images, which improved the detection accuracy. Niu et al [38] proposed a lane detection method with two-stage feature extraction to solve the problem of robustness caused by the inconsistency of lighting and background clutter, where each lane has two boundaries. The above-mentioned detection methods are easy to implement, have low complexity, and can obtain high real-time performance, but are extremely susceptible to environmental influences. On rainy days or under insufficient light, misjudgment of lane lines is prone to occur.

With the continuous development of technology and the continuous improvement of hardware equipment, neural network-based detection methods have made breakthrough progress. Using deep learning to detect lane lines can ensure good recognition accuracy in most scenarios [39]. Instead of relying on highly specialized manual features and heuristics to identify lane breaks in traditional lane detection methods, target features under deep learning can automatically learn and modify parameters during the training process. Liu et al [40] used the mobile edge computing framework to greatly reduce the time loss of learning. Qi et al [41, 42] significantly improved the data processing rate through the method of distributed location and multiple data sources. John et al [43] used the learned **CNN** model to predict the lane line position in the image, especially in the case of occlusion and missing, **CNN** can extract robust features from the road image, and then train an additional tree-like regression model directly from the features estimated by lane line position. Kim et al [44] used serial end-to-end transfer learning to directly estimate and separate the left and right lanes and collect a dataset that includes multiple road conditions to train a **CNN** model. Pan et al [45] proposed a **S**patial **CNN** (**SCNN**) and the convolution of the extended layer structure in the feature map is a slice structure convolution. In this way, the information transfer between pixels is performed between the rows and columns of each layer, which is suitable for strong spatial relationships, but long continuous-shaped structures or large targets lack apparent clues in this method, such as lane lines, walls, and pillars. Zhang et al [46] applied deep learning-based convolutional neural networks to instance segmentation of monocular vision to achieve segmentation of different objects in actual scenes, and achieved better segmentation results, but less robust. The use of deep learning methods for lane line detection has greatly improved the robustness and accuracy compared with traditional detection methods, but to train the network, a huge data set is required as support, and the requirements for hardware facilities are correspondingly increased. Xu et al [47, 48] proposed a video surveillance resource method when studying the **I**nternet **o**f **V**ehicles (**IoV**), which supports edge computing and can effectively use edge computing to solve the problem of lane image acquisition and training requirements and high network bandwidth, we combine this method in the data processing process to distribute computing power from the central node to devices with image acquisition and processing capabilities. At the same time, we choose to convert the lane line detection problem into an instance segmentation problem and combine edge computing to design a two-instance segmentation branch network, where the lane segmentation branch outputs the background or lane line, and the lane embedding branch separates the lane lines obtained by the segmentation branch into different lane examples to improve the accuracy of lane detection. At the same time, a custom

network is designed, and the training data is marginalized to enhance the real-time nature of the data processing. In this project, we are using the gradient and **HLS** thresholding are used to detect the lane lines on the roads.

### 1.4.4 Object Detection System (ODS)

Numerous kinds of research are done on different aspects of **A**dvanced **D**river **A**ssistance **S**ystems (**ADAS**) and Autonomous vehicles. The IoT-based occlusion technique called multiple targets tracking in occlusion areas with interacting object models in urban environments was used for autonomous vehicles to solve the problem of object detection by Chen et al by using a laser scanner [49]. The different observed shapes on each laser scan made it difficult to identify the object. Hence, the proposed system is developed using a machine learning approach using **YoloV5** which reduces the occlusion issue. **A**dvanced **D**river **A**ssistance **S**ystems (**ADAS**) also include driver monitoring systems. A **D**river **M**onitoring **S**ystem (**DMS**) helps in keeping track of various facial features of the driver like eyelid and mouth movement. One such system was proposed by Kato et al [50]. There is a lot of research done on object detection since it plays a crucial role in many of the technologies, to get a better understanding of state-of-the-art object detection techniques and models, cloud-based. Liu et al surveyed most of the research that provides a clear picture of these techniques. The main goal of this survey was to recognize the impact of deep learning techniques in the field of object detection which has led to many groundbreaking achievements. This survey covers many features of object detection ranging from detection frameworks to evaluation metrics [51, 52].

For many region-based detectors, like Fast **R-CNN** [53], a costly per-region subnetwork is applied several times. To address this, Girshick introduced **R-FCN** by proposing location-sensitive score maps to address a dilemma between translation-invariance in image classification and translation-variance in object detection [54]. One of the major challenges of object detection was to detect and localize multiple objects across a large spectrum of scales and locations, due to which the pyramidal feature representations were introduced. In this, an image is represented with multiscale feature layers. **F**eature **P**yramid **N**etwork (**FPN**), one such model to generate pyramidal feature representations for object detection, presents no difficulty and as well as effective but may not be the optimal architecture design. For image classification in a vast search space, the **N**eural **A**rchitecture **S**earch (**NAS**) algorithm demonstrates favorable results in the productive discovery of outstanding architectures. Hence, inspired by the modularized architecture proposed by Zoph et al, Dai et al proposed the search space of scalable

architecture that generates pyramidal representations. They proposed an architecture, called **NAS-FPN**, which provides a lot of flexibility in building object detection architecture and is adaptable to variety of backbone models, on a wide range of accuracy and speed tradeoffs [55]. Various detection systems repurpose classifiers by taking a classifier for an object and evaluating it at multiple locations and scales in a test image. For example, **R-CNN** uses region proposal methods to first produce bounding boxes that are likely to appear in an image and then, on these suggested boxes, run a classifier. These intricate pipelines were slow and hard to optimize. Hence, Ghiasi et al proposed you only look once (**YOLO**), an algorithm that is a single convolutional network that simultaneously predicts multiple bounding boxes and class probabilities for those boxes. Unlike **R-CNN** and other similar algorithms, **YOLO** is found to be extremely fast and sees the entire image during training and testing, hence making fewer background errors. When trained on natural images and tested on the artwork, **YOLO** outperforms other algorithms by a wide margin. But **YOLO** was yet found to lag behind state-of-the-art detection systems in accuracy and struggled to localize some objects precisely [56]. Redmon et al, focusing mainly on improving recall and localization while maintaining classification accuracy proposed **YOLOv2**. It was then found that detection methods are constrained to a small set of objects; hence, they as well proposed a joint training algorithm that allows one to train object detectors on both detection and classification data, using which they trained the **YOLO9000** algorithm which was built by modifying **YOLOv2** [57]. The majority of the accurate **CNN**-based object detectors required high **GPU** power and training to achieve their optimal accuracy. High **GPU** power is essential for achieving accuracy and speed in real-time since it is vital in a car collision or obstacle warning model. Redmon and Farhadi proposed a modified version of the state-of-the-art object detection model, **YoloV5**, with significant improvement in the speed and accuracy of the models. An impressive aspect of this model is that it can operate in real-time on a conventional **GPU** and training as well requires only a single **GPU**. Hence, using conventional **GPUs** such as **1080Ti** or **2080Ti**, we can train an accurate and extremely fast object detector [58]. Since **YoloV5** outperforms other frameworks, we have chosen to go with it. The proposed solution comes with the utilization of a conventional **GPU** power-based pipelined and accurate **YoloV5** framework for obstacle detection at a higher speed.

## 1.5 Analysis from Literature Review (in the context of your project)

The systems and techniques discussed in the literature have provided insight into road safety and vehicle hazard prevention methods. Many of the discussed systems performed reasonably well and provided acceptable results. The discussed methods each work well and have their strengths and weaknesses. The discussed systems in the literature review are mostly just a single system performing a very specific task. The type of vehicle the system is going to be used in is also a limiting factor. **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) on the other hand does four of the most important things include to road safety. The **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) incorporates drowsiness detection, yawning detection, lane detection, and object detection in a single intelligent system ready to be used in all kinds of vehicles. The purpose of the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) is not only to integrate the multiple methods to produce a single vehicle safety system performing multiple vehicle safety-related tasks but also to provide the general public with a system that can easily be installed in multiple types of vehicles.

For **D**rowsiness **D**etection **S**ystem (**DDS**), we have opted for the Machine Learning and Deep learning aided computer vision techniques as the recent breakthroughs in Deep Learning have provided new tools for computer vision for detection and classification. Since the driver's facial features change significantly during the driving sessions, we can use computer vision techniques to determine which type of human behavior is an indication of the driver being drowsy while driving. More specifically using **H**aar-**C**ascade-**C**lassifier (**HCC**) and **DLib**-**S**hape-**P**redictor to predict the driver's drowsiness.

For **Y**awning **D**etection **S**ystem (**YDS**), we are using a similar approach as the drowsiness **D**etection **S**ystem (**DDS**) The **Y**awning **D**etection **S**ystem (**YDS**) is also using **H**aar-**C**ascade-**C**lassifier (**HCC**) and **DLib**-**S**hape-**P**redictor to predict the yawning.

For **L**ane **D**etection **S**ystem (**LDS**) we are using Computer Vision and thresholding techniques. Preprocessing steps are taken to remove the noise from the frames. The gradient and **HLS** thresholding are used to detect the lane lines and vehicle center offset on the frames.

For **O**bject **D**etection **S**ystem (**ODS**) we are using the **YOLO-V5** model. The model has been significantly improved since the previous models and the speed and accuracy of the model are higher than that of its pre-requisites. Since **YoloV5** outperforms other frameworks using fewer resources, it is the preferred choice for both performance and resource vise.

## 1.6 Methodology and Software Lifecycle for This Project

A software development methodology is a splitting methodology of software development work into distinct phases (or stages) containing activities with the intent of better planning and management of the project. Currently, the common and traditional software development methodologies are Waterfall, Prototyping, Iterative and Incremental Development, Spiral Development, Rapid Application Development, and Extreme Programming methodologies as well as various types of Agile methodologies. The main aim of all of these software development methodologies is to produce high-quality software products. These methodologies comprise a set of processes that are designed to work rigidly and sequentially. This strong hierarchical approach is intended to guarantee a high level of control over software projects. Software developers seek to have high control over their projects to produce high-quality software products. Meanwhile, from the business perspective, high-quality software is important because it can best satisfy customer needs. On the other hand, there is another type of software development, which is the development of software for educational purposes. University students enrolled in programs of study in IT and computer science is required to take a compulsory subject called the Final Year Project or Graduate Project. To fulfill the requirements for passing this undergraduate project, students must develop a software program. Hence, the purpose of this software is completely different from that of other traditional software. In short, the undergraduate software project intends to achieve pedagogical and educational outcomes. Keeping the fact in mind that our project is based on educational backgrounds and our goal is to satisfy the learning objectives, we have chosen to go with the **R**apid **A**pplication **D**evelopment (**RAD**) model.

### 1.6.1 Rapid Application Development (RAD)

Rapid Application Development (**RAD**) is a development model that prioritizes rapid prototyping and quick feedback over long-drawn-out development and testing cycles. With rapid application development, developers can make multiple iterations and updates to software quickly without starting from scratch each time. This helps ensure that the outcome is more quality-focused and is in alignment with the end user's requirements. **R**apid **A**pplication **D**evelopment (**RAD**) was conceived in the 1980s, so it's not something new. But, unlike the waterfall model, it's not singular. It's a continuous evolution of development philosophies according to the requirement at that particular time. Initially, **R**apid **A**pplication **D**evelopment (**RAD**) took the shape of the spiral model, where one or more development models were used to work on a particular project.

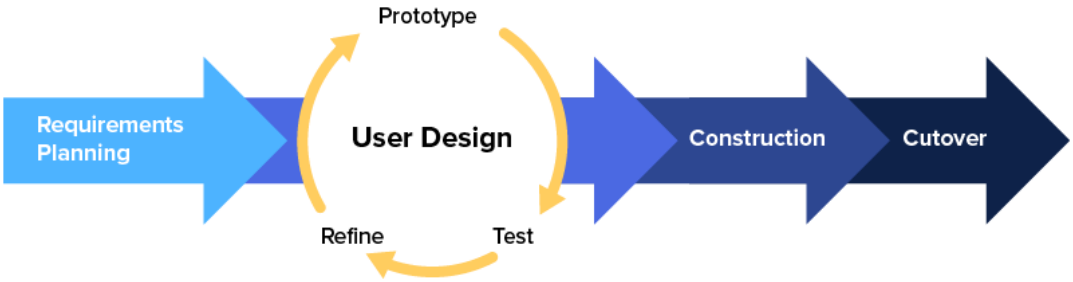



**Figure 1.1: Rapid Application Development Life Cycle**

### 1.6.2 Steps in Rapid Application Development

The four basic steps of **R**apid **A**pplication **D**evelopment (**RAD**) are as follows

### 1.6.3 Define the Requirements

At the very beginning, **R**apid **A**pplication **D**evelopment (**RAD**) sets itself apart from traditional software development models. It doesn't require us to sit with end users and get a detailed list of specifications; instead, it asks for a broad requirement. The broad nature of the requirements helps us take the time to segment specific requirements at different points of the development cycle.

### 1.6.4 Prototype

This is where the actual development takes place. Instead of following a rigid set of requirements, developers create prototypes with different features and functions as fast as they can. These prototypes are then shown to the clients who decide what they like and what they don't. More often than not, these prototypes are quickly made to work to showcase just the key features. This is normal, and the final product is only created during the finalization stage where the client and developer are in alignment with the final product.

### 1.6.5 Rapid construction and feedback gathering

Rapid construction is where application coding, system testing, and unit integration occur, converting prototype and beta systems into a working model. This phase may also be repeated as required, supporting new components and alterations. Generally, teams use low-code or **R**apid **A**pplication **D**evelopment (**RAD**) tools to quickly progress the application. Since the majority of user problems and client changes are addressed during the iterative prototyping phase, developers can construct a final working model faster than they would following a traditional development approach. Software and applications are thoroughly tested during this phase, ensuring the result satisfies client expectations and objectives.

### 1.6.6 Finalize product and implementation

The final phase of **R**apid **A**pplication **D**evelopment (**RAD**) is where developers address the technical debt accrued in early prototyping, optimizing implementation to improve stability and maintainability as they finalize the product for launch. Components are moved to a live production environment, where full-scale testing occurs to identify product bugs. The implementation phase is where development teams move components to a live production environment, where any necessary full-scale testing or training can take place. Teams write thorough documentation and complete other necessary maintenance tasks, before confidently handing the client a complete product.

### 1.6.7 The rationale behind Selected Methodology

A rapid application development approach has a plethora of benefits for us as academic users of this model. With speed and agility as a priority, we observe a boost in productivity that enhances project outcomes and enables our delivery in a matter of days or weeks. The following are some of the key advantages of using a **R**apid **A**pplication **D**evelopment (**RAD**) method.

### 1.6.8 Early system integration & risk reduction.

**R**apid **A**pplication **D**evelopment (**RAD**) allows us to quickly create and share working prototypes, enabling us to review functionality earlier in the software life cycle. The frequent iterations and nature of feedback allow all aspects to be easily evaluated, enabling measurable progress that determines if schedules and budgets are on track. Integration with other systems and services traditionally occurs at the end of a development life cycle, but rapidly developed applications are integrated almost immediately. Testing occurs during every iteration, enabling stakeholders to quickly identify and discuss errors, code vulnerabilities, or complications, and immediately resolve them without impacting the progress of development. A reduction in risk means a reduction in the cost and time of development of the system.

### 1.6.9 Adaptability and compartmentalization of system components

During the development phase, the software is malleable. Changing code can dramatically alter the entire system, and we can take advantage of this flexibility by iterating and prototyping potential concepts throughout development. This iterative nature encourages us to create components that are functional and independent. Each element is compartmentalized, which increases the reusability of components, and makes modification easily adaptable to the needs of software evolution and development.

### 1.6.10 Constant user feedback

**R**apid **A**pplication **D**evelopment (**RAD**)'s nature of easily and frequently obtaining relevant feedback from users who interact directly with applications during development and prototyping is invaluable. Regular communication and constant feedback increase the overall efficiency and quality of the system. The iterative design and access to UI/UX components of a system put feedback at the forefront of the process. Receiving feedback such as from fellow project members, supervisors and other concerned members of the project allows us to quickly adapt our system to the changes suggested by the reviewers and continue the development without a significant amount delay.

# Chapter 2
# Problem Definition

## 2 Problem Definition

Making efforts to achieve the ultimate road safety and coming up with genius and innovative technology-based answers to everyday road safety problems is very understandable. This chapter gives an overview of the problem used as the basis for the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) project. The actual problems faced are analyzed and a suitable solution is proposed. Furthermore, the deliverables and the development requirements are discussed. Additionally, an example current system is discussed in the hope to understand the current project in the context of why is it necessary to develop the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) if there is an alternative of alike already available.

### 2.1 Problem Statement

Ideally, the drivers would be alert and aware of their surroundings at all times and take control of the vehicle in case of an emergency. However, human beings are prone to make mistakes. Safety on the road is an important concern for all drivers and passengers. With the sharp increase of vehicles on the road, this has become increasingly important and challenging due to the higher probability of accidents. Many people die each year in roadway departure crashes caused by driver inattention.

#### 2.1.1 Statement

Keeping road safety in mind, the paper proposes using **O**pen **C**omputer **V**ision **L**ibrary (**OpenCV**) to detect lane lines and vehicle center offset and alert the driver in case they depart from their lane. **H**aar **C**ascade **C**lassifier (**HCC**) and **DLib** Shape predictor to detect drowsiness and yawning based on the Eyelids and lips distance and alert them in case of drowsiness and yawning detection. **YoloV5** nano deep learning model trained on coco classes to detect the object in front of the vehicle and label those objects. The project aims to minimize human casualties as a result of vehicle collisions by monitoring the driver's behavior and sending real-time alerts to get their attention back on the road in case any negligence is detected.

## 2.2 Deliverables and Development Requirements

The all-around objectives of this project are to take advantage of the technology at our disposal like Computer Vision, Machine learning, and Deep learning coupled with hardware like cameras and speakers to develop a vehicle safety system, equipped with drowsiness detection, yawning detection, lane detection and object detection that will allow the vehicles fitted with the systems to alert the careless drivers in real-time and defuse the danger of vehicle crash due to driver's negligence. To this end, the more specific objectives of this work are as under:

- To develop a system that can automatically detect drowsiness and alert the driver to be more attentive in real-time.
- To develop a system that can automatically detect yawning and alert the driver in real-time.
- To develop a system that can detect lane lines and vehicle center offset and alert the driver in case the driver wanders off the lane.
- To develop a system that can detect multiple objects including pedestrians and vehicles in front of the vehicle.
- To design a non-intrusive and effective road safety system for drowsiness, lane lines, and object detection.
- Reduce human casualties and injuries due to road accidents caused by driver negligence by providing organizations and individuals with a system to monitor drivers' behavior.
- Increase the quality of driving and road safety by alerting and notifying drivers as they feel drowsy, while also alerting them if they stay outside their lane for longer than usual or warning them to slow down if the system detects any objects up ahead.

## 2.3 Current System

There are several **A**dvanced **D**river **A**ssistance **S**ystems (**ADAS**) out in the market for users to buy. One such system is **ADAS+ Advanced Driver Assistance System ADAS-1100** sold by a company called **BRAND MOTION** for **$750**. The system has multiple useful features discussed as follows.

### 2.3.1 Integration

- Integrates into virtually any vehicle
- Versatile mounting and display options are available
- Multiple alert options are available
- 2-camera DVR system

### 2.3.2 System Features

- Automatic Calibration – quick and easy installation
- Forward Collision Alert – alert of an impending crash if the distance between vehicles becomes too close
- Lane Departure Warning – if lane departure is detected without the use of a turn signal, **ADAS+** will alert the driver
- Pedestrian Detection – **ADAS+** watches for pedestrian crosswalks and detects when people are crossing in front of traffic
- Forward Car Departure Alert – if traffic comes to a standstill, **ADAS+** will alert the driver of the flow of traffic
- Motorcycle Collision Alert – detects small vehicles to prevent collision
- Safe Distance Alert – **ADAS+** generates a safe distance in front of your vehicle to prevent tailgating
- Virtual Bumper – generates a virtual bumper length in front of your vehicle to prevent tailgating
- Automatic Event Recording – in the event of a collision, **ADAS+** will automatically record out of the front and rear camera with audio and record on the included 32GB SD card
- Manual Video Recording – recording can be triggered manually at the push of a button
- Password Protection Available – touch screen can be password protected to prevent disabling the system

### 2.3.3 System Specifications

- High-Quality Built-in Camera
- 1920 x 1080 Resolution 1080p 25 FPS Front and Rear Camera
- 16.5 Foot Rear Camera Harness
- 130¬∫ Horizontal Field of View
- 110¬∫ Vertical Field of View
- Continuously Recording with Automatic Event Save
- Built-In Microphone
- 3.5” TFT Touchscreen LCD Display
- 800 x 400 Resolution 400p Display
- GPS/GLDNASS Receiver Included
- 32 GB Micro SD Card Included (System Is Expandable To 128GB)
- Adjustable Speaker Volume
- 12-24V Operating Voltage Range
- Physical Dimensions of **ADAS+** Windshield Unit 4.5” x 2.5” x 1.4”

### 2.3.4 Packaging

- Length: 12 Inches
- Width: 9 Inches
- Height: 6 Inches
- Weight: 2 Lbs.
- UPC: 819779020818

# Chapter 3
# Requirement Analysis

## 3 Requirement Analysis

To understand and plan for the development of the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) it is essential to understand its requirement specifications. The parts of the **S**oftware **R**equirements **S**pecification (**SRS**) report included in this chapter are the Use Cases, functional and non-functional requirements of the project.

### 3.1 Use Case Diagrams

Multiple systems implement the Driving Negligence Dissuader System (DNDS). For understanding, each system will be presented with its designated Use Case diagram.

### 3.1.1 Drowsiness Detection System (DDS) & Yawning Detection System *(YDS)*

Since the **D**rowsiness **D**etection **S**ystem (**DDS**) and **Y**awning **D**etection **S**ystem (**YDS**) both work on the same methodologies, they are presented together as a single system.

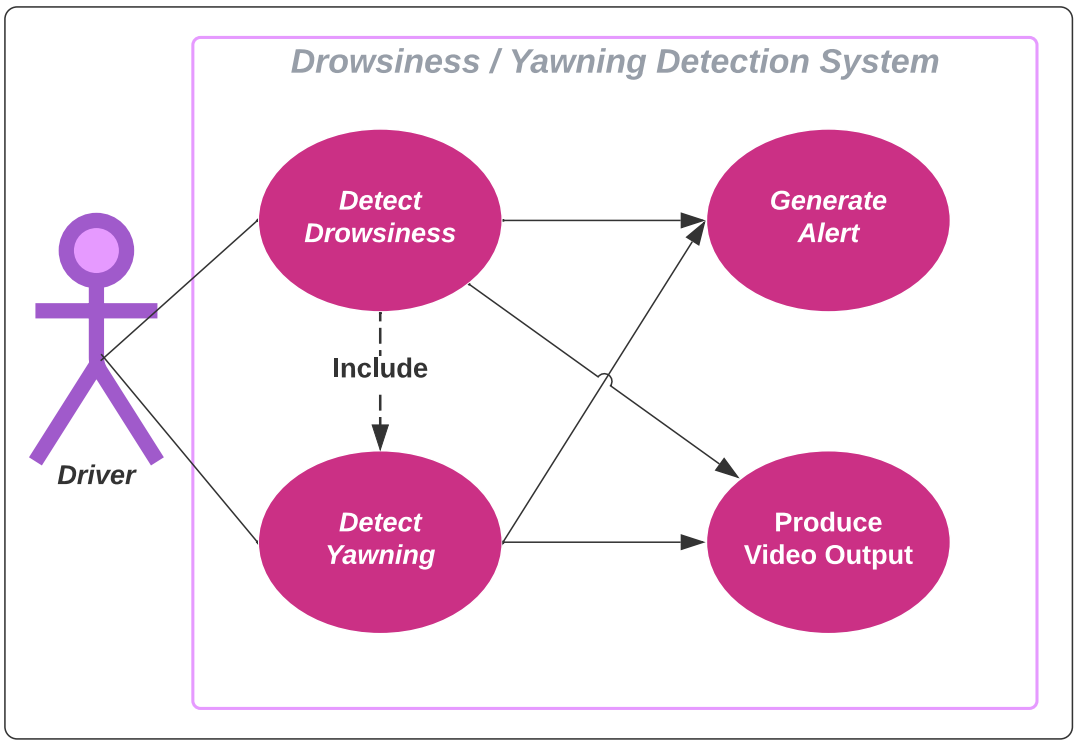



**Figure 3.1: DDS & YDS Use Case Diagram**

| Use Case | Description |
|---|---|
| Detect Drowsiness | The driver's drowsiness state is detected |
| Detect Yawning | Driver's Yawning is detected |
| Generate Alert | The Alarm or alert is generated upon any detections |
| Produce Video Output | Video output with detections is produced |

### 3.1.2 Lane Detection System (LDS)

The use case diagram for **L**ane **D**etection **S**ystems (**LDS**) is shown in **Figure 3.2**.

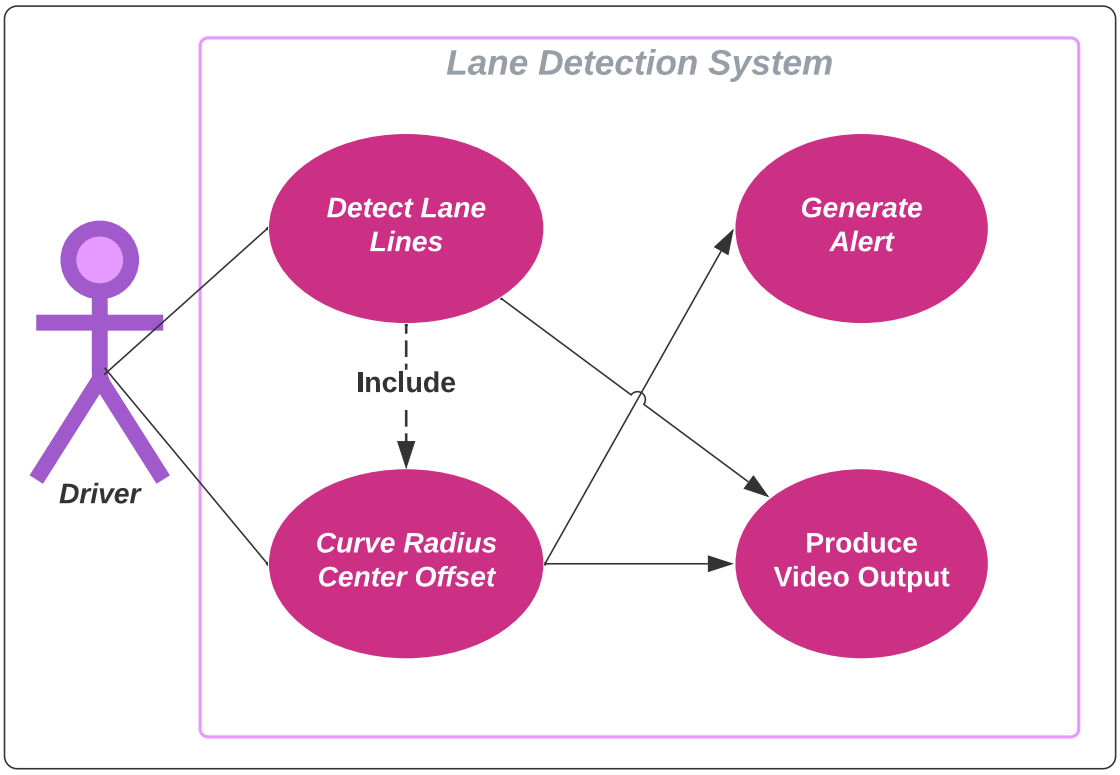



**Figure 3.2: LDS Use Case diagram**

| Use Case | Description |
|---|---|
| Detect Lane Lines | Lane lines are detected on the road |
| Curve Radius | Road Curve is determined using the sliding windows technique |
| Center Offset | The vehicle's center offset from the lanes is determined |
| Generate Alert | The alert is generated if the car wanders off its lane |
| Produce Video Output | Video output with detections is produced |

### 3.1.3 Object Detection System (ODS)

The Use case diagram for **O**bject **D**etection **S**ystems (**ODS**) is shown the **Figure 3.3**.

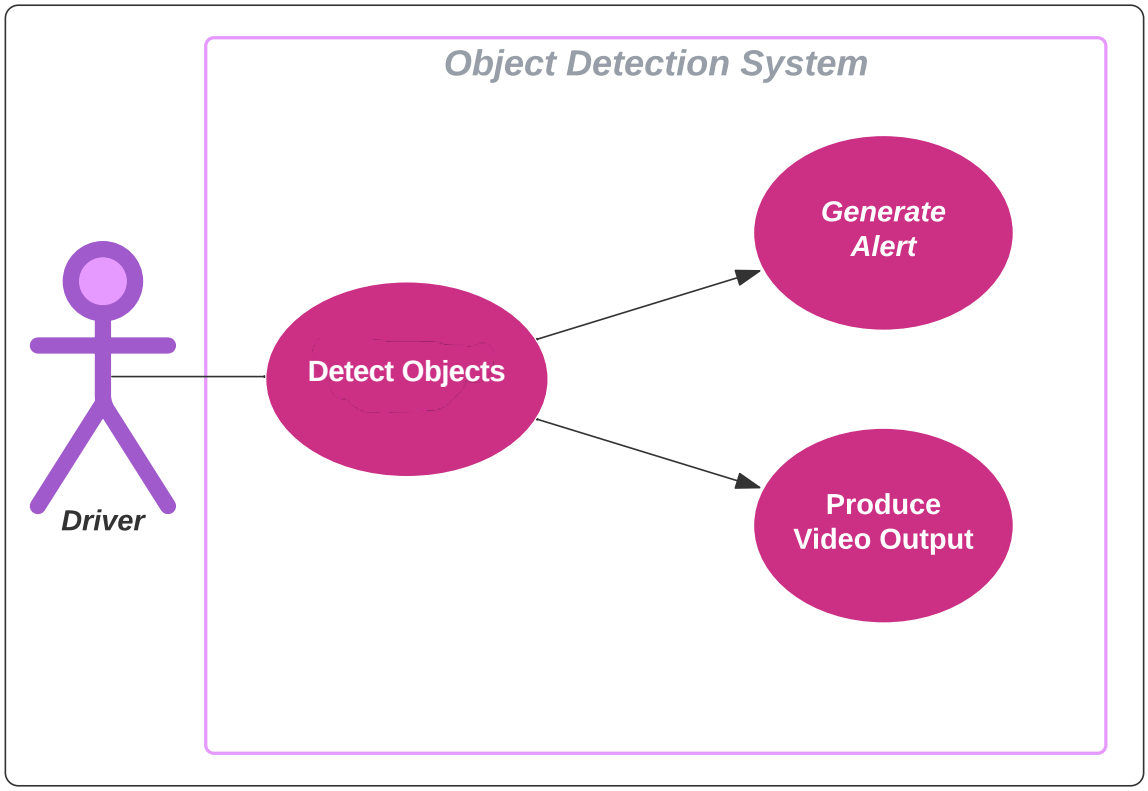



**Figure 3.3: ODS Use Case diagram**

| **Use Case** | **Description** |
|---|---|
| Detect Objects | Lane lines are detected on the road |
| Generate Alert | The Alarm or alert is generated upon detecting objects in close vicinity of the vehicle. |
| Produce Video Output | Video output with detections is produced |

## 3.2 Detailed Use Case

The Use Case diagrams in **Figures 3.1.1**, **3.1.2,** and **3.1.3** highlight the Use Cases of the project **D**riving **N**egligence **D**issuader **S**ystem (**DNDS)**. Drivers can use the system to detect their drowsiness state. In the case of drowsiness, the generated alert is used to alert the driver of their drowsiness. Likewise, the system also detects yawning. In the case of yawning detection, an alert is generated asking the driver to take a short break. The system can detect lane lines on the road and mark them as well as the vehicle center offset from the lane and can alert the driver in the case they wander off their lane. Lastly, the system can detect multiple objects including pedestrians, motorcycles, and other vehicles on the road up ahead. In case there are objects in close vicinity to the vehicle, an alert can be generated to warn the driver to be cautious. All of the detections are marked accordingly on the output video.

## 3.3 Functional Requirements

The **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) incorporates multiple modules. Each module is a standalone intelligent system that performs a specific type of detection based on the video footage it is fed as an input (Face-cam or dashcam) and gives outputs in the form of alert and processed video footage. The functional requirement of these modules is as follows.

### 3.3.1 Drowsiness Detection System (DDS) & Yawning Detection System (YDS)

The main goal of the **D**rowsiness **D**etection **S**ystem (**DDS**) and **Y**awning **D**etection **S**ystem (**YDS**) is to detect drivers' drowsiness and yawning. In case of drowsiness or yawning detection, the system must generate an alert, asking the driver to take a break o get some fresh air.

### 3.3.2 Lane Detection System (LDS)

The **L**ane **D**etection **S**ystem (**LDS**) must detect Lane lines on the road and determine the center offset of the vehicle. In case the driver wanders off their lane as determined by the center offset of the vehicle, the system must generate an alert to ask the driver to get back to their lane.

### 3.3.3 Object Detection System (ODS)

The purpose of the **O**bject **D**etection **S**ystem (**ODS**) is to detect different kinds of objects on the road up ahead that could be potential road hazards. Objects include pedestrians, other vehicles, motorcycles, cyclists, large objects, etc.

## 3.4 Non-Functional Requirements

Both functional and non-functional requirements describe specific characteristics that a system must have to meet the needs of the project. Non-functional requirements of the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) are as follows.

### 3.4.1 Reliab*le*

**D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) must perform its intended tasks with a minimum failure rate. The system must be reliable to use most of the time.

### 3.4.2 Maintainable

The system must be easily maintainable. Adding new functionality to the system should be straightforward without requiring large changes to the system or the platform.

### 3.4.3 Non-Obtrusive

The driver must be focused and vigilant while driving since there is a lot at stake. **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) must not intrude in the process of driving and must work seamlessly without the driver noticing it other than when there is a need to alert the driver.

### 3.4.4 Universally Compatible

The system must be compatible with all sorts of vehicles i.e., SEDANs, COUPEs, SPORTS CARs, SUVs, etc.

### 3.4.5 Easy Installation*/Uninstallation*

The system must be easy to install and uninstall and should not require major modifications to the vehicle.

### 3.4.6 High performance, Low specifications

**D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) must perform smoothly and efficiently on minimum hardware requirements. The system should make detections in real-time without a considerable amount of lag.

# Chapter 4
# Design and Architecture

## 4 Design and Architecture

Documenting design requirements and creating software design documents is a must for every software project. It helps to ensure the design specifications of the software are well understood and clear to all. It specifies what is possible with the product and how it can be accomplished. It assures that the product is built to meet the needs and is on par with what was proposed to be delivered. The parts of **S**oftware **D**esign **D**escription (**SDD**) discussed in this chapter are the system architecture, data representation and description, process flow, and design models.

### 4.1 System Architecture

System architecture conveys the informational content of the elements consisting of a system, the relationships among those elements, and the rules governing those relationships. The following section describes the architecture of the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**).

#### 4.1.1 Drowsiness Detection System (DDS)

The **D**rowsiness **D**etection **S**ystem (**DDS**) uses **O**pen **C**omputer **V**ision **L**ibrary (**OpenCV**), **H**aar **C**ascade **C**lassifiers (**HCC**), and **DLib** Shape predictor to detect drowsiness. Initially, the face is detected and the region is cropped from the frame using **H**aar **C**ascade **C**lassifiers (**HCC**). After detecting the face of the driver, facial landmark detection is used to localize important regions of the face, including eyes, eyebrows, nose, ears, and mouth. Using the **DLib** face shape predictor, the **68** facial landmarks are obtained from the cropped face as shown in **Figure 4.1**.

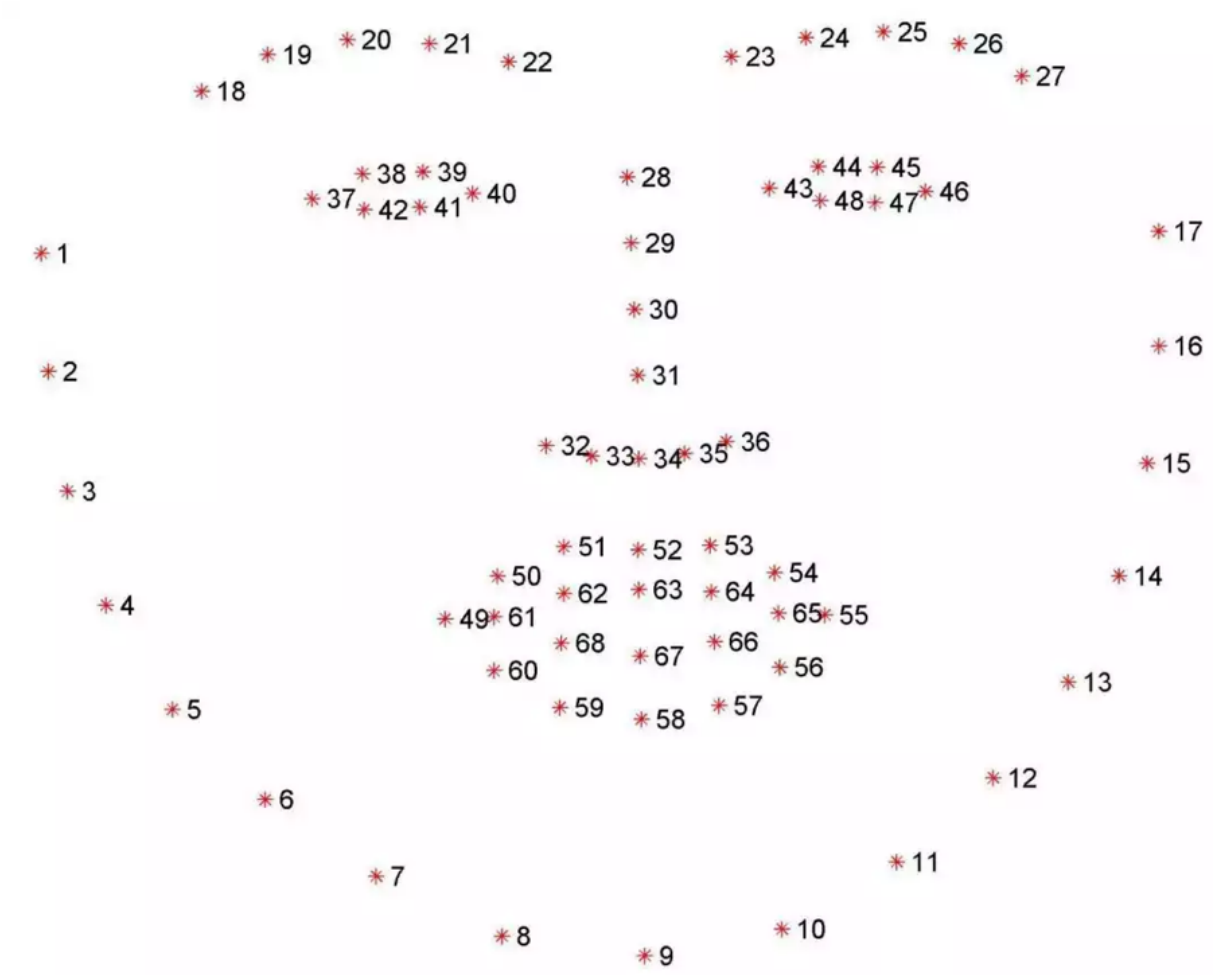



**Figure 4.1: Facial Landmark Points**

Of those **68** facial landmark points, points **37, 38, 39, 40, 41,** and **42** are selected for the right eye, and points **43, 44, 45, 46, 47,** and **48** are selected for the left eye. Once the points for eyes are acquired, the drowsiness is detected by measuring the **E**ye **A**spect **R**atio (**EAR**) of the driver using the function in **Equation 4.1**.

$$EAR = \frac{\left|\left|p2 - p6\right|\right| + \left|\left|p3 - p5\right|\right|}{2\left|\left|p1 - p4\right|\right|}$$

**Equation 4.1: Eye Aspect Ratio (EAR) function**

The **E**ye **A**spect **R**atio (**EAR**) function in **Equation 4.1** shows the eye aspect ratio formula where **p1, p2, p3, p4, p5,** and **p6** are the **2D** landmark locations of the eye. The **p2, p3, p5,** and **p6** are used to measure the height whereas **p1** and **p4** are used to measure the width of the eyes as shown in **Figure 4.2**. The calculation of the drowsiness level of the driver is based on eye close duration. The **E**ye **A**spect **R**atio (**EAR**) formula, in **Equation 4.1** can detect the eye blink using the scalar value. For instance, if the driver blinks their eyes for longer than usual (**1/3s** per blink), it means that the driver is in a state of drowsiness. Thus, it is necessary to detect the eye's shape accurately to calculate the eye blink properly. From the landmarks detected in the image with the face shown in **Figure 4.1**, the **E**ye **A**spect **R**atio (**EAR**) is used as an estimate of the eye openness state. For every video frame, the eye landmarks are detected between the height and width of the eye that had been computed.

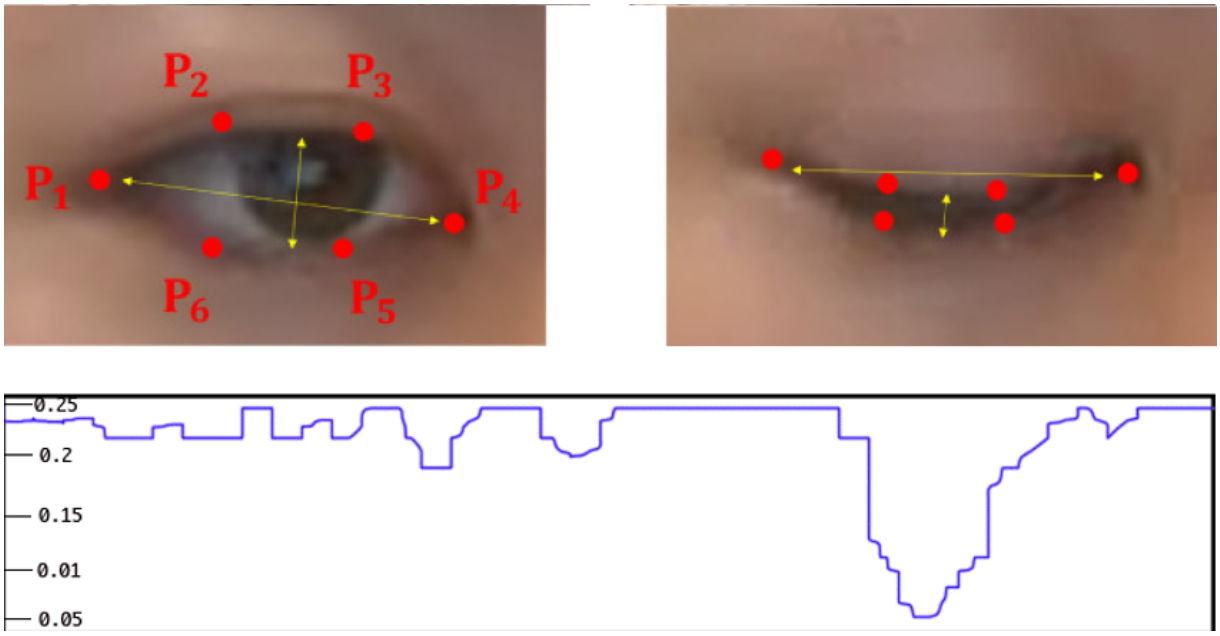



**Figure 4.2: Drowsiness Detection in action using EAR function**

The **E**ye **A**spect **R**atio (**EAR**) is a constant value when the eye is opened but rapidly falls approximately to zero (**0**) when the eye is closed as shown in **Figure 4.2**. Each eye is represented by **6 (x, y)** coordinates, starting at the left corner of the eye (as if you were looking at the person), and then working clockwise around the remainder of the region.

### 4.1.2 Yawning Detection System (YDS)

The **Y**awning **D**etection **S**ystem (**YDS**) works on the same principle as the **D**rowsiness **D**etection **S**ystem (**DDS**). It uses **O**pen **C**omputer **V**ision **L**ibrary (**OpenCV**), **H**aar **C**ascade **C**lassifiers (**HCC**), and **DLib** Shape predictor to detect yawning. The face is detected and the region is cropped from the frame using **H**aar **C**ascade **C**lassifiers (**HCC**). After detecting the face of the driver, the **68** facial landmarks are obtained from the cropped face as shown in **Figure 4.1**. Of those **68** facial landmark points, points **49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,** and **68** are selected for the mouth. Once the points for the mouth are acquired, the yawning is detected by measuring the **M**outh **A**spect **R**atio (**MAR**) of the driver using the function in **Equation 4.2**.

$$MAR = \frac{\left|\left|p2 - p8\right|\right| + \left|\left|p3 - p7\right|\right| + \left|\left|p4 - p6\right|\right|}{2\left|\left|p1 - p5\right|\right|}$$

**Equation 4.2: Mouth Aspect Ratio (MAR) function**

The **M**outh **A**spect **R**atio (**MAR**) function in **Equation 4.2** shows the mouth aspect ratio formula where **p1, p2, p3, p4, p5, p6, p7,** and **p8** are the **2D** landmark locations of the mouth. The **p2, p3, p4, p6, p7,** and **p8** are used to measure the height whereas **p1** and **p5** are used to measure the width of the lips in meters (**m**) as shown in **Figure 4.1**. The calculation of the yawning level of the driver is based on mouth open duration. The **M**outh **A**spect **R**atio (**MAR**) formula, in **Equation 4.2** can detect the mouth open using the scalar value. For instance, if the driver opens their mouth for longer than usual, it means that the driver is yawning. Thus, it is necessary to detect the mouth shape accurately. From the landmarks detected in the image with the face shown in **Figure 4.1**, the **M**outh **A**spect **R**atio (**MAR**) is used as an estimate of the mouth's open state. For every video frame, the mouth landmarks are detected between the height and width of the mouth that had been computed.

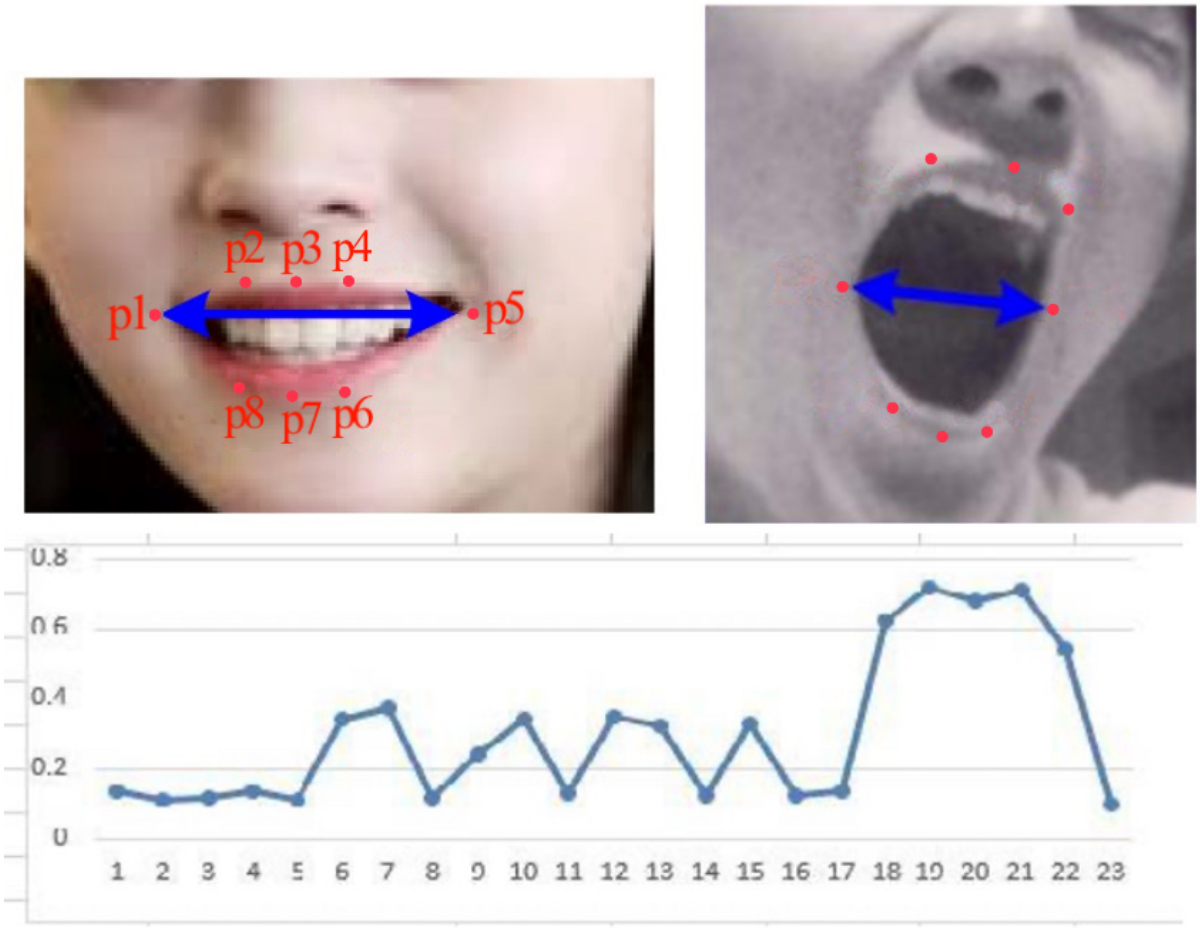



**Figure 4.3: Yawning Detection in action using the MAR function**

The **M**outh **A**spect **R**atio (**MAR**) value is relatively lower when the mouth is closed, but rapidly increases as the driver opens their mouth as shown in **Figure 4.3**. The mouth is represented by 8 **(x, y)** coordinates, starting at the left corner of the mouth and then working clockwise around the remainder of the region.

### 4.1.3 Lane Detection System (LDS)

**L**ane **D**etection **S**ystems (**LDS**) uses **O**pen **C**omputer **V**ision **L**ibrary (**OpenCV**) to detect lane lines, **V**ehicle **C**enter **O**ffset (**VCO**), and the **C**urve **R**adius (**CR**). **OpenCV** method uses the input frames from the camera module to find any lane lines and also for rendering out an illustration of the lane. The **OpenCV** tools like Color Selection, the **R**egion **o**f **I**nterest (**RoI**) selection, Grey Scaling, Gaussian Smoothing, **C**anny **E**dge **D**etection (**CED**), **S**liding **W**indow **S**earch (**SWS**), and Hough Transform are employed for detect lane lines detection. A Color Detection algorithm identifies pixels in a picture that matches a given color or color range (In this case white and yellow color). **R**egion **o**f **I**nterest (**RoI**) selection allows us to select a

rectangular area on the frames which will be the focus of interest for lane detection and the area outside this region will be cropped out. Grey Scaling is the method of changing an image from different color spaces e.g., **RGB**, **CMYK**, **HSV**, etc. to shades of grey. In the Gaussian Blur operation, the image is convolved with a mathematician filter rather than the box filter. The Gaussian filter could be a low-pass filter that removes the high-frequency elements. **C**anny **E**dge **D**etection (**CED**) is used to detect the edges in a frame. It accepts a grayscale image as input. The Hough Transform is a method that is used in image processing to detect any shape if that shape can be represented in mathematical form. The goal is to piece along a pipeline to detect the line segments within the frame, then average/extrapolate them and draw them onto the image for the show. In the first stage, the frames are undistorted so that it will restore the straightness of lines, helping to identify lane lines.

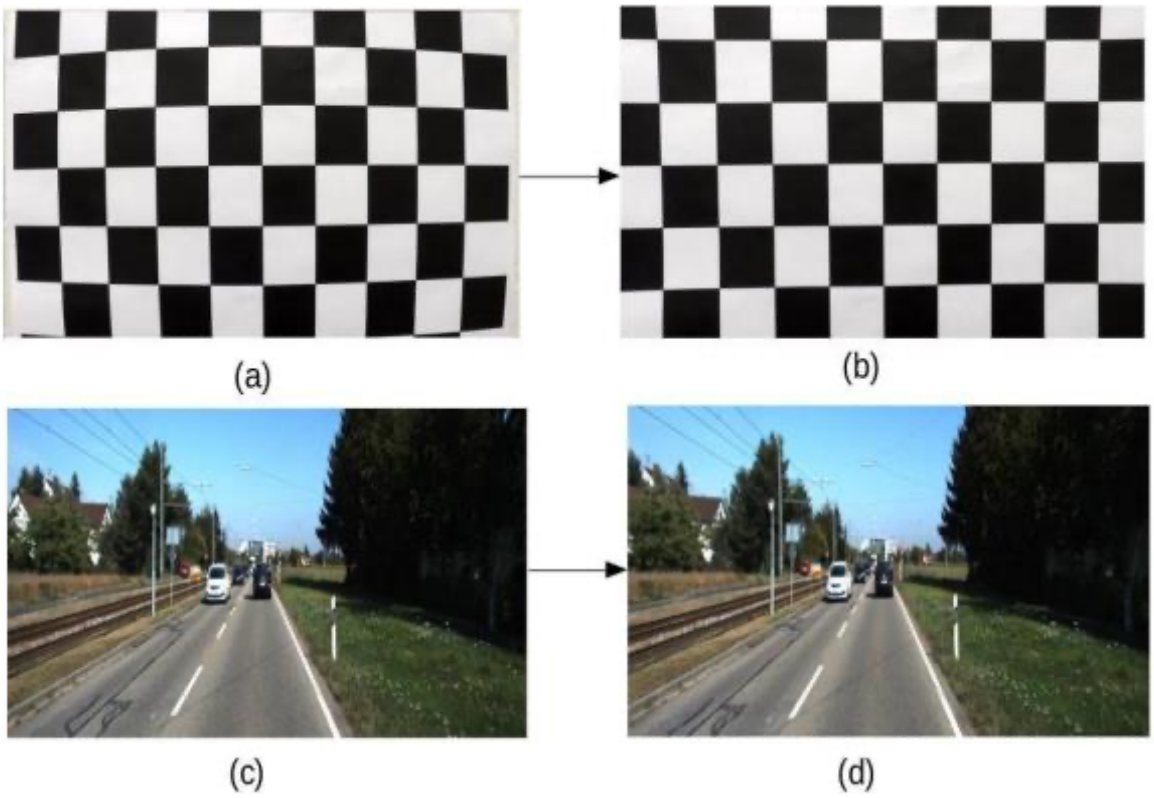



**Figure 4.4: Distorted Images (a, c) vs undistorted images (b, d)**

As shown in **Figure 4.4**. The variation between the distorted (original) and undistorted images is clear. The curved lines are now straight. The camera matrix and distortion coefficients using chessboard images are calculated in OpenCV. Here, it can be accomplished by gaining the inside corners within an image and utilizing that information to undistort the image.

Now lane line edges are isolated from the rest of the frame. To do this we define a function that will isolate a certain hard-coded region where we are likely to see lanes. It takes one parameter, the original image, and outputs the isolated region. **Figure 5.2** show the blueprint of the **R**egion of **I**nterest (**RoI**) function.

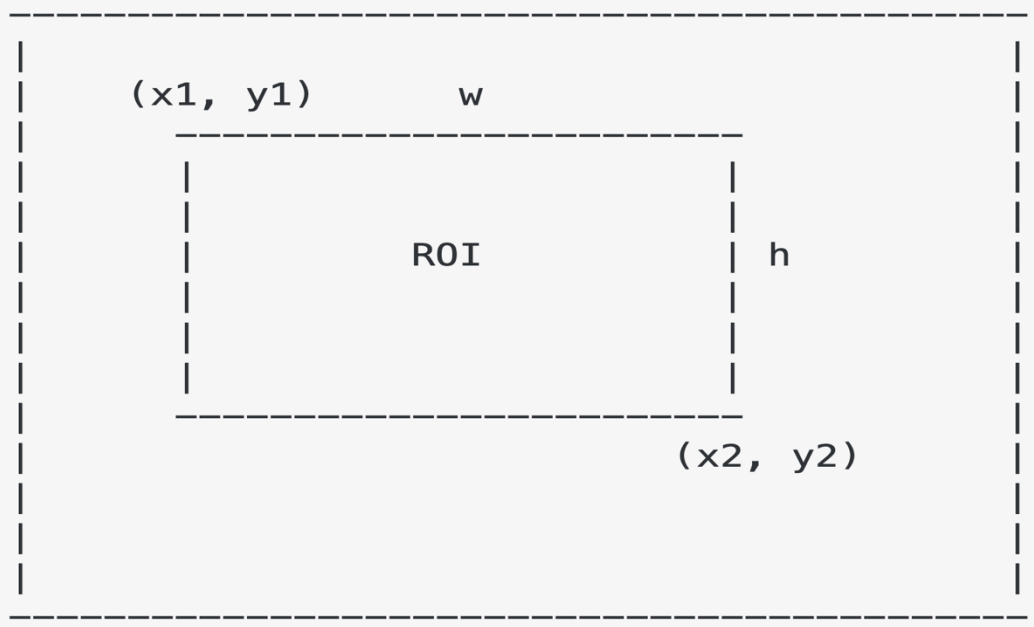



**Figure 4.5: RoI Function**

In **Figure 4.5** we are removing the unwanted part of the image such as the sides where we will not be focusing on detecting the lane lines. we only need to focus on the regions where we are likely to see lanes. For this reason, the cropping operation is done. After cropping the image, it is also converted into greyscale. Moving on we will be performing the rest of the operations on the cropped greyscale image.

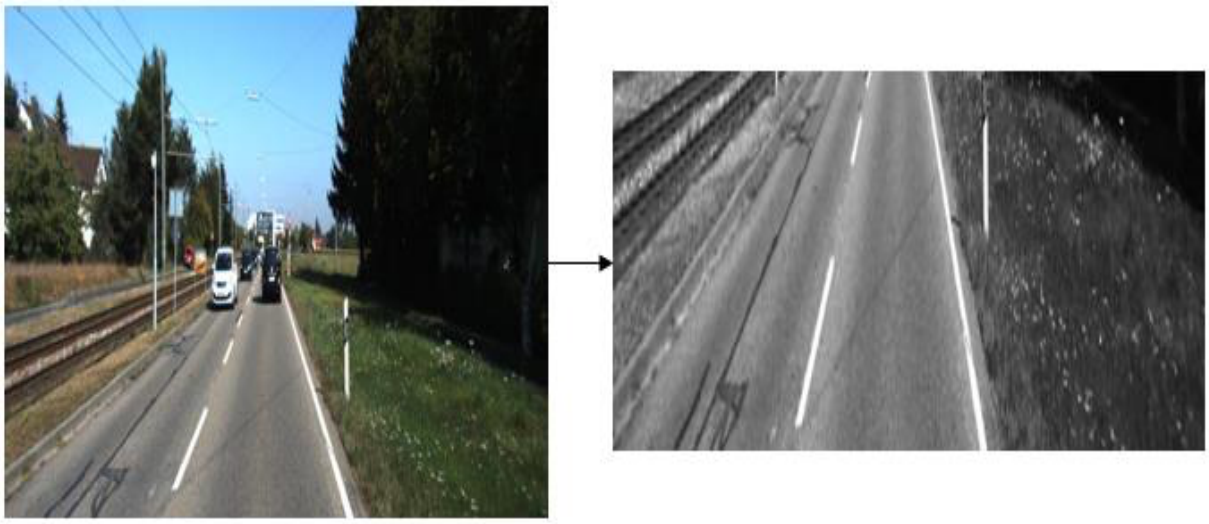


**Figure 4.6: Image after setting the RoI to the frame**

The next stage is to perform thresholding. This method is a kind of image segmentation that separates objects by altering grayscale images into binary images. The image thresholding technique is the most appropriate in images with high stages of contrast. The thresholding procedure is stated in **Equation 4.3**.

$$T = T[a, b, p(a, b), f(a, b)]$$

**Equation 4.3: Thresholding function**

Where **T** represents the threshold value, the coordinates of the threshold value are **(a, b)** and the grayscale image pixels are **p (a, b), f (a, b)**. We are using two types of thresholding. Gradient Thresholding and **H**ue **S**aturation **L**ightness **(HLS)** Thresholding. Sobel is a kernel for gradient thresholding in both and **x** and **y-axis**. Since the lane lines are probably going to be vertical, more weight on the inclination in a **y-axis** is given. **Figure 4.7 (b)** shows the Sobel image. **H**ue **S**aturation **L**ightness **(HLS)** color channel is used to handle cases when the road color is too bright or too light. The Lightness (**L**) channel threshold diminishes edges formed from shadows in the frame. The **S**aturation (**S**) channel threshold expands white or yellow lanes. Hue (**H**) toward the line colors. The outcome after applying **HLS** thresholding is depicted in **Figure 4.7 (c)**.

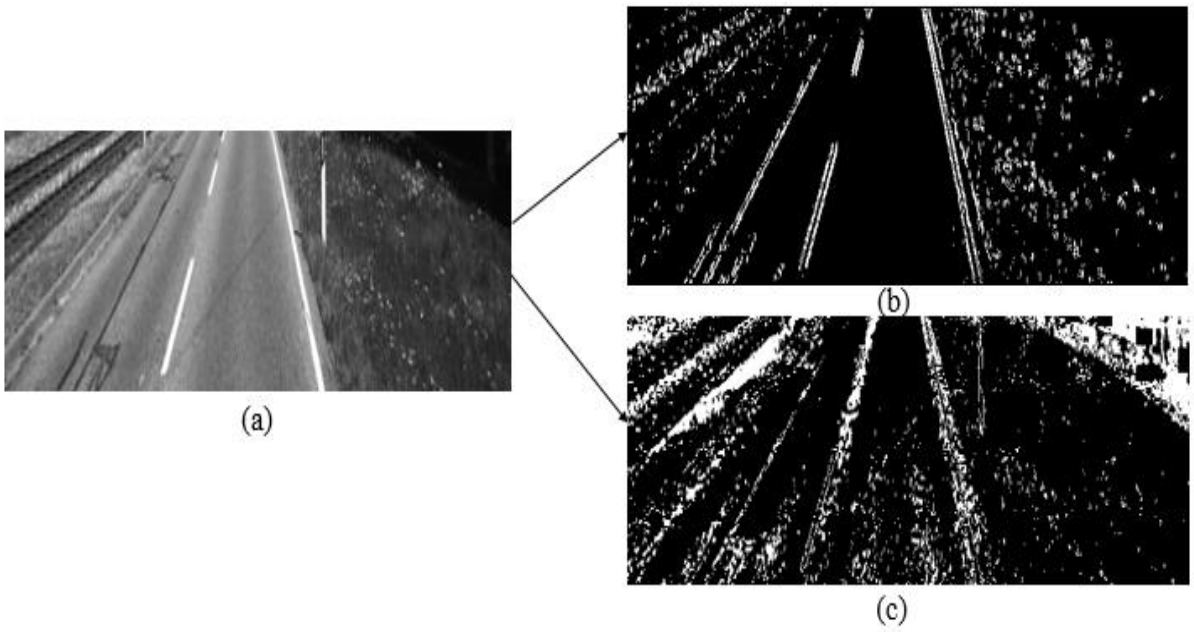



**Figure 4.7: Gradient and HLS Thresholding applied to the cropped image**

Then we are combining both the Gradient and **HLS** (color) thresholding into one for the final thresholding binary image that improves the overall results of the lane detection process. The output combining the Gradient and **HLS** thresholding is demonstrated in **Figure 4.8 (c)**.

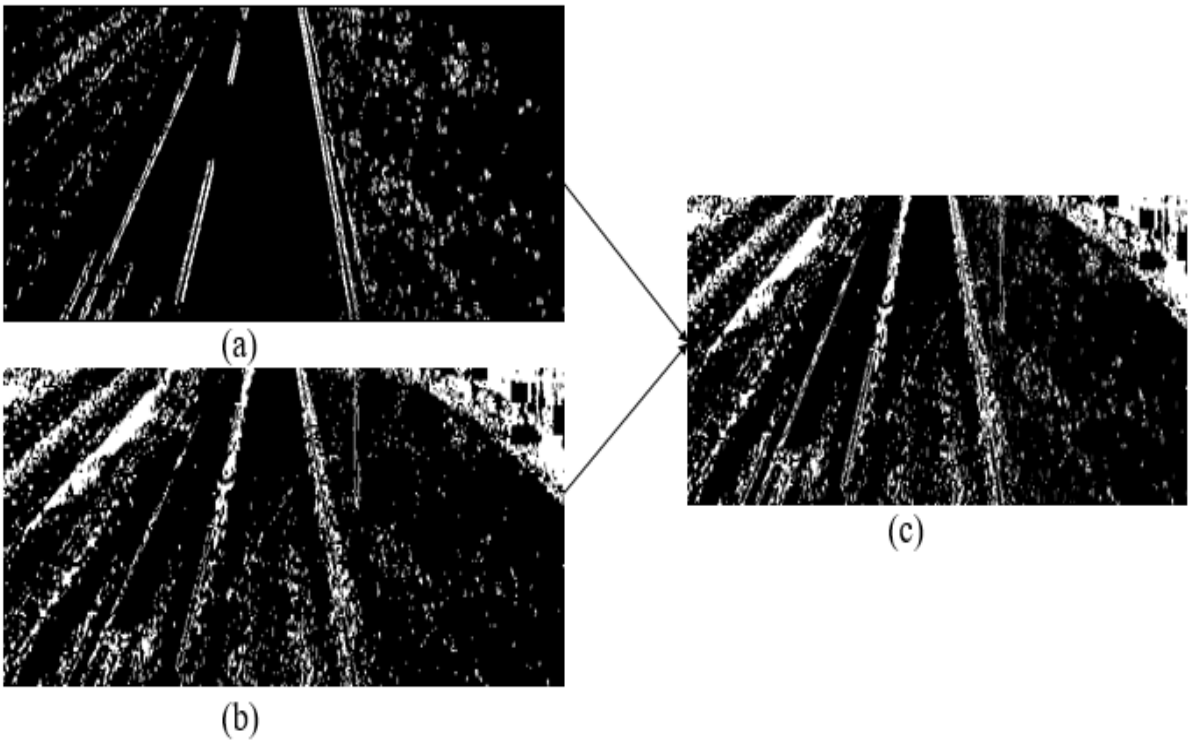



**Figure 4.8: Gradient and HLS images combined**

Once the thresholding process is complete. The perspective transformation is applied to the image. Perspective transformation is used to convert a **3d** world image into a **2d** image. While un-distorting and thresholding help to cover the crucial information, we can additionally divide that information by taking a drake at the part of the image of the road surface. To center around the road part of the image, we move our point of view to a top-down perspective of the street. While we don't obtain any more information from this step, it's enormously easier to isolate lane lines and measure things like curvature from this perspective. The Perspective Transform of the combined thresholding image is shown in **Figure 4.9**.

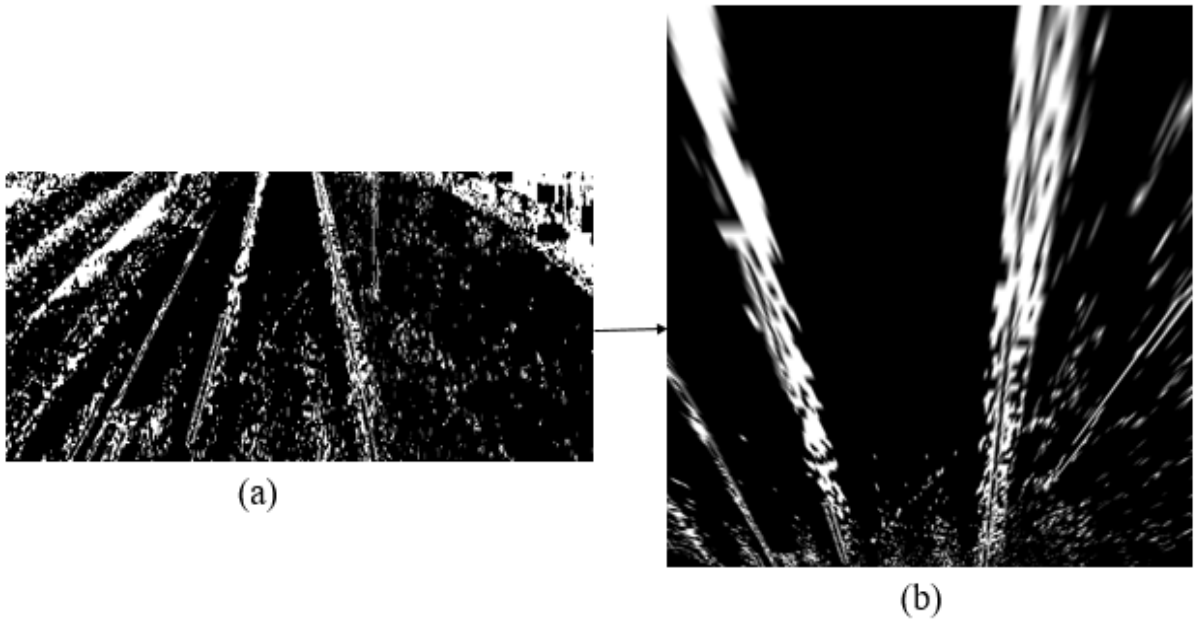



**Figure 4.9: Perspective Transformation applied to the image**

After perspective transformation, as the lane lines are already detected in an earlier frame, the information is used in a sliding window, placed around the line centers, to detect and track lane lines from the bottom to the top of the image. The result of the sliding window search is demonstrated in **Figure 4.10**.

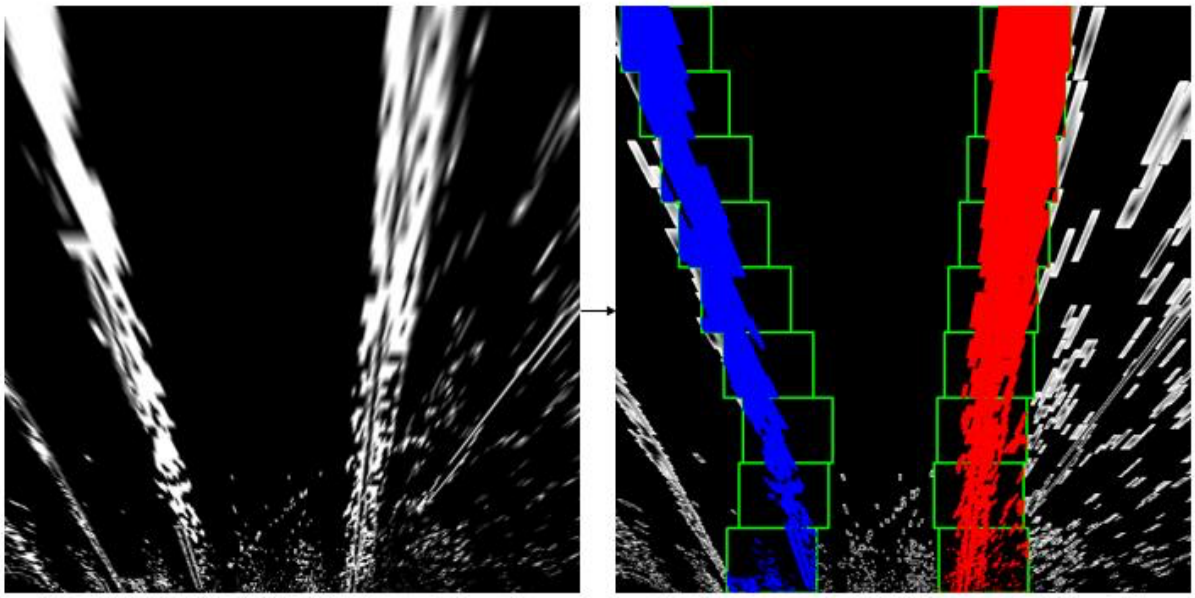


**Figure 4.10: Sliding Window search applied to the image**

This permits us to do a highly qualified search and saves a lot of processing time. To recognize left and right lane line pixels, their **x** and **y** pixel positions are used, to fit a second-order polynomial curve shown in **Equation 4.4**.

$$f(y) = ay^2 + by + c = 0$$

**Equation 4.4: Second Order Polynomial Curve function**

The **f(y)** is used, rather than **f(x)** because the lane lines in the warped image are approximately vertical and may have the same **x** value for more than one **y** value.

Once the sliding window search is performed, the lane lines are easily recognized from the top point of view. A sliding window search distinguishes the lane lines. The green boxes express to the windows where the lane lines are colored. The lane lines are then re-centered to the standard pixel position so that they resemble the lines. The shaded lines will be stepped back onto the original image. The result of illustrated lanes from the sliding window search image is shown in **Figure 4.11**.

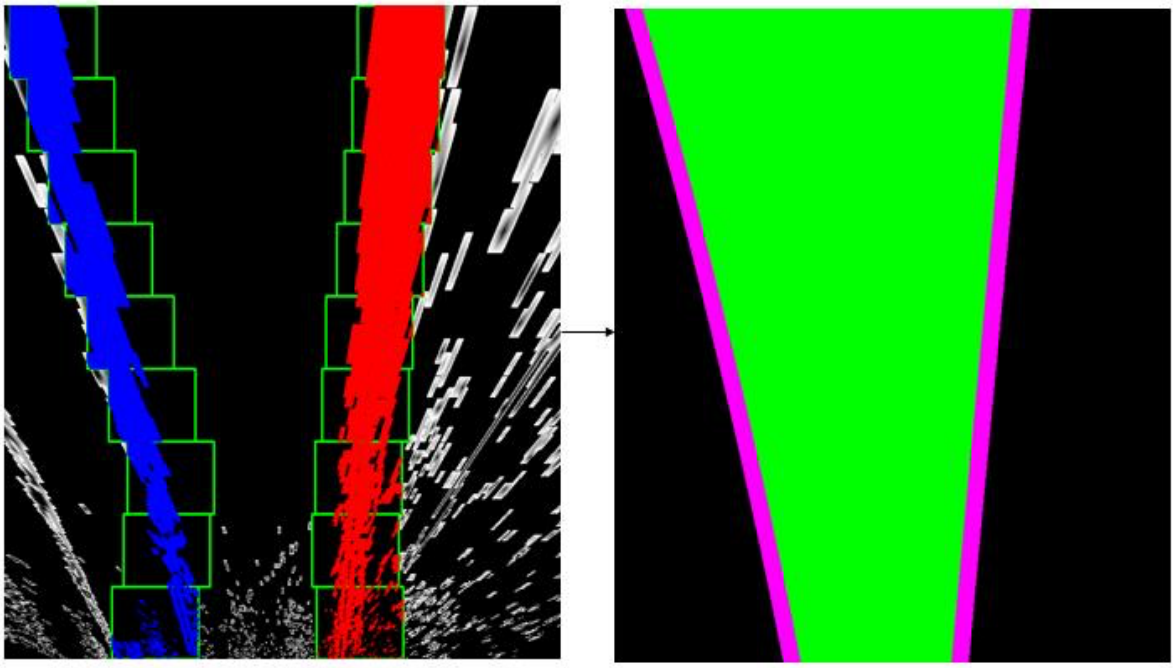


**Figure 4.11: Illustrated Lane from sliding window search**

After illustrating the lane lines, the warp and crop operations are performed for proper visualization of the image. The result of the warped and cropped image is shown in **Figure 4.12**.

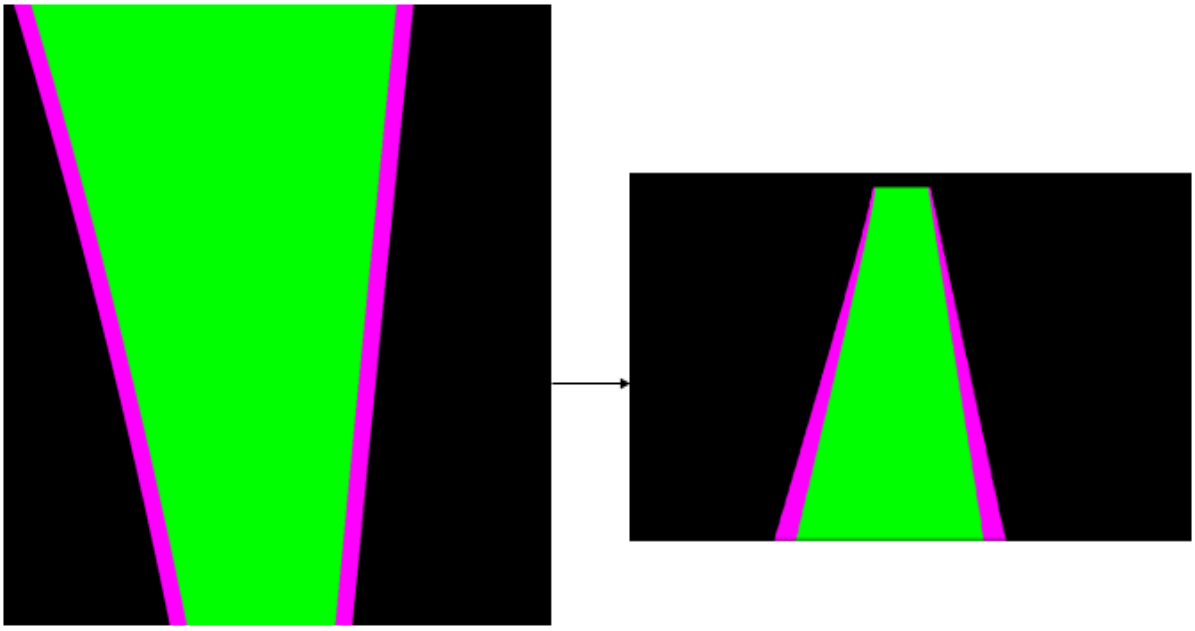


**Figure 4.12: Warped and cropped lane illustration**

Lastly, all this information is gathered and the results are drawn back onto the original image. The two pink lines which are recognized above are stated as lane lines, and the space between them is colored green to determine the road surface. While the pipeline prepares for a single image, it can easily be applied to processing many images to detect the lane line on the road surface. The final result of the proposed lane detection system is shown in **Figure 4.13**.

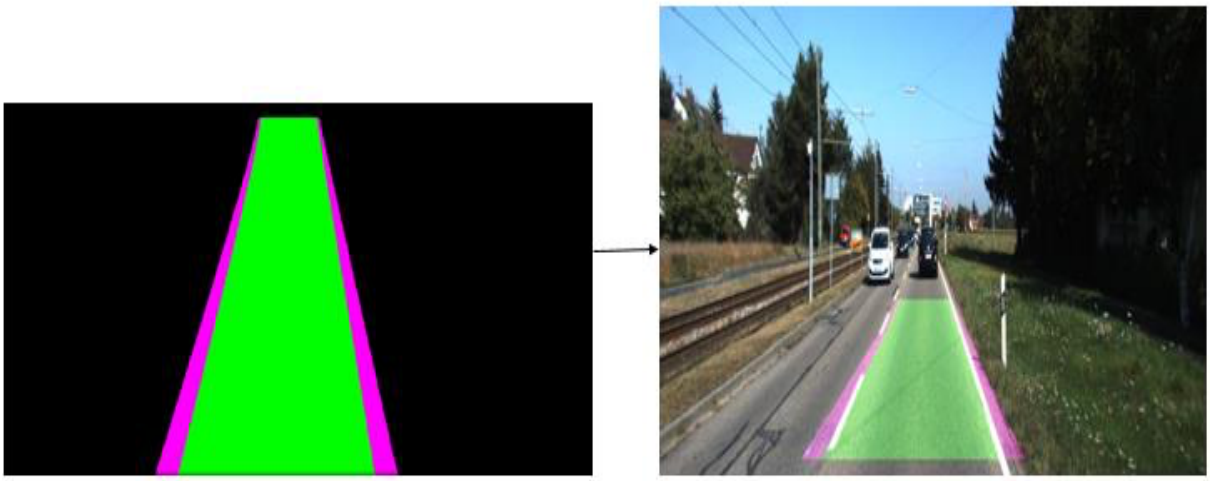


**Figure 4.13: Final result of Lane Detection System (LDS)**

### 4.1.4 Object Detection System (ODS)

**O**bject **D**etection **S**ystem (**ODS**) uses deep learning model **YoloV5** trained on coco classes to detect the objects. The model is trained to detect upwards of **88** different objects including, humans, animals, other vehicles, motorcycles, traffic signs, signals, etc. Each frame is processed along three stages in the algorithm. Namely, the **backbone**, **neck**, and **head** as illustrated in **Figure 4.14**.

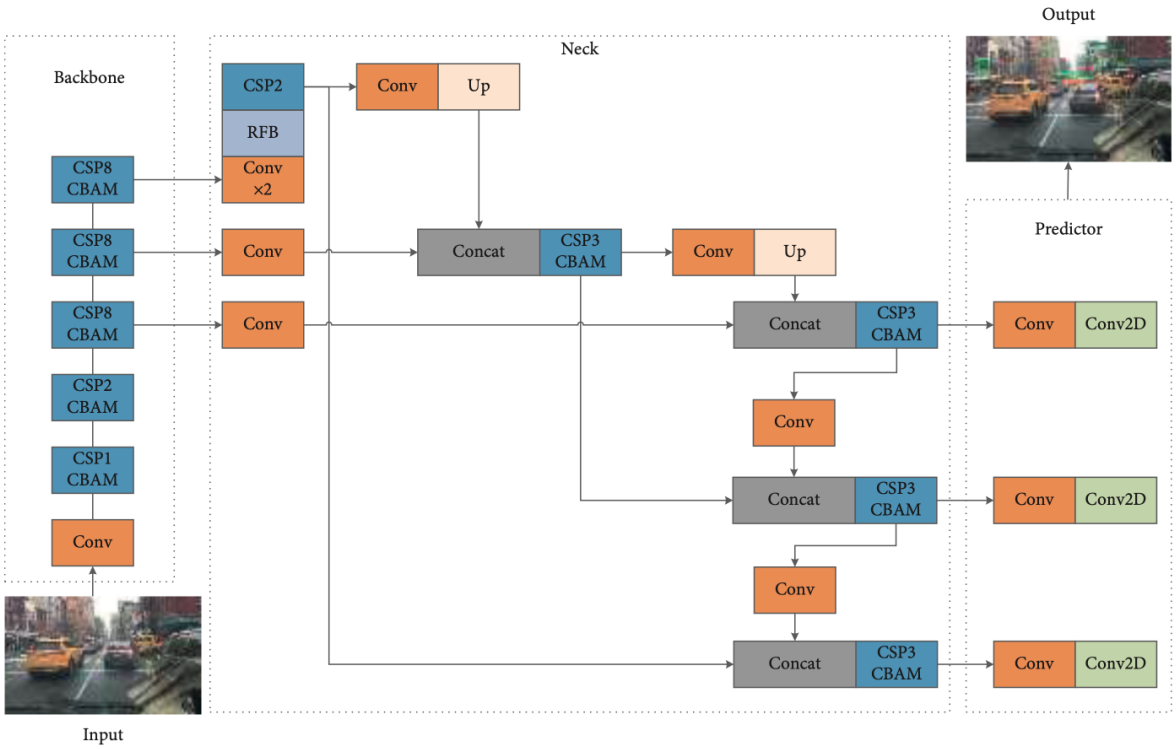



**Figure 4.14: YoloV5 model structure**

The structure of the **YoloV5** deep learning model is shown in **Figure 4.14**. The system is comprised of three major modules. The **backbone** is based on the **CSPDarkNet53**. The **Neck** is concatenated Path Aggregation Networks (**PANet**) with Spatial Pyramid Pooling (**SPP**) module, and the **head** is the **YOLO** layer of the system. The **Backbone** and **neck** are responsible for object extraction from the input frames. The head is responsible for object detection and tracking. **Darknet53** shown in **Figure 4.15** comprises **53** convolutional layers. These **53** layers are stacked onto the original architecture of **53** layers which gives us **106** layers of architecture. Initially, the video input is processed frame by frame. Then **C**ross-**S**tage-**P**artial-**C**onnections (**CSPC**) are used to eliminate duplicate gradient information that occurs while using conventional **DenseNet**.

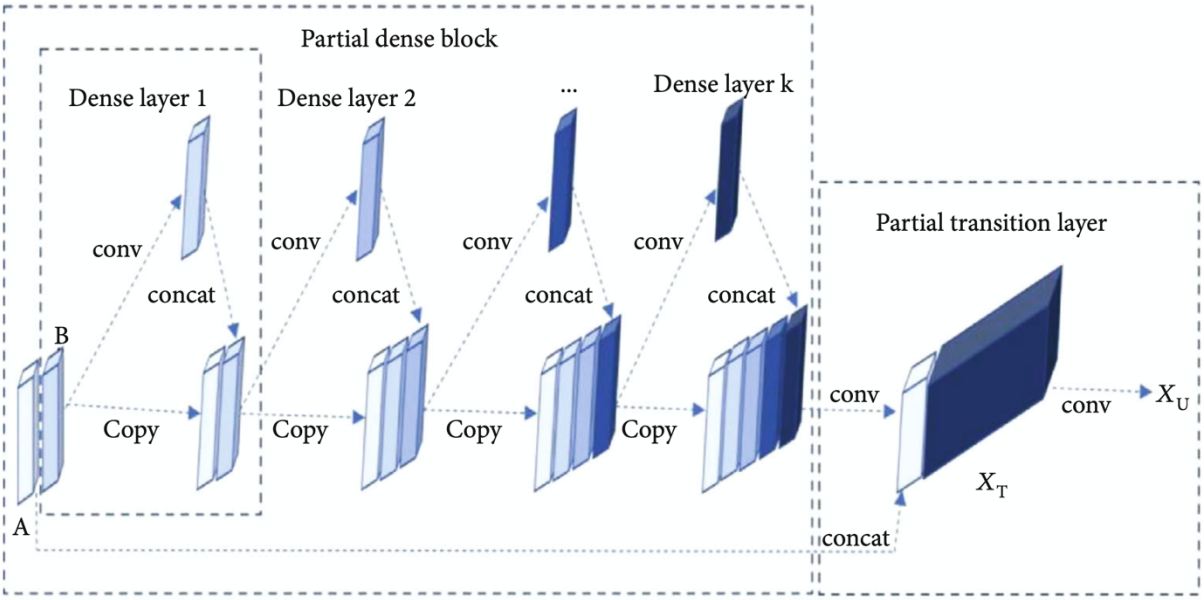



**Figure 4.15: CSPDarkNet53 overview**

In **CDPDarkNet53**, the base layer is divided into **2** parts, part **A** and part **B**. Part **B** goes into the original dense block and is processed accordingly. Part **A** will directly skip to the transition stage as shown in **Figure 4.15**. This results in no duplicate gradient information. It also reduces a lot of computations. Additional layers are added between the **backbone** and the **head** using the **neck**. To aggregate the information, the **YoloV5** algorithm applies a modified path aggregation network with a modified spatial attention module and a modified **S**patial **P**yramid **P**ooling (**SPP**). Concatenated path aggregation networks with additional **S**patial **P**yramid **P**ooling (**SPP**) modules are used to increase the accuracy of the detector. Each frame processed in the **backbone** and **neck** is then transferred to the **head** which involves the **YoloV5** algorithm. The **YoloV5** works using Residual blocks, Bounding box regression, **I**ntersection **O**ver **U**nion (**IoU**), and lastly the Final Detection. Initially, the input frame is divided into grids. Each grid cell is responsible for detecting the objects present in its cell. The **YOLO** algorithm runs such that bounding boxes as shown in **Figure 4.16**. Confidence scores are predicted around every object present in that particular grid. Every bounding box consists of width (**bw**), height (**bh**), bounding box center (**x,y**), and confidence score (**c**). The confidence score represents how confident and accurate the algorithm is of a particular object in that bounding box. Together with these attributes, **YOLO** uses a single bounding box regression to predict the probability of an object appearing in the bounding box.

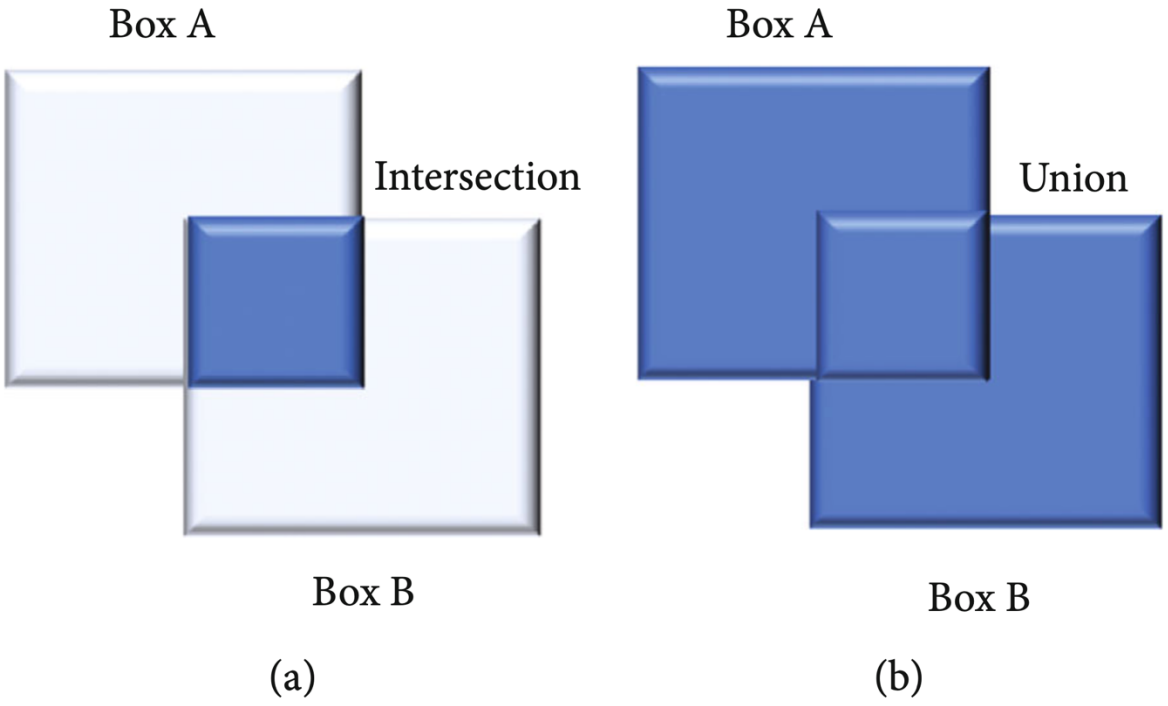



**Figure 4.16: The Intersection and Union of bounding boxes**

If no object exists in a grid cell, then the confidence score is zero; else, the confidence score must be equal to the **I**ntersection **O**ver **U**nion (**IoU**) between the predicted box and ground truth. Here, the ground truth boxes are manually predefined by the user; hence, greater **IoU** means a greater confidence score, which means higher accuracy of prediction by the algorithm. Filtration of those boxes with no objects is done based on the probability of objects in that box. Nonmax suppression processes eliminate the unwanted bounding boxes, and the box with the highest probability or confidence score will remain.

$$IoU = area\ of\ \frac{boxA\ \cap boxB}{boxA\ \cup boxB}$$

**Equation 4.5: Intersection Over Union (IoU) Function**

The **I**ntersection **O**ver **U**nion (**IoU**) Equation shown in **Equation 4.5** is used to measure the overlap between the two proposals. Nonmax suppression is used to find the appropriate bounding box among the predicted bounding boxes by the algorithms based on the confidence scores. This is represented in the Nonmax Suppression Algorithm in **Figure 4.17**.

**Input:** Set of proposal boxes A, corresponding confidence scores C and overlap threshold T.
**Output:** A list of filtered proposals F.
**procedure** Sup(A,m)
$A_{sup} \longleftarrow \varnothing$
**for** $a_i \in A$ do
***remove*** $\longleftarrow$ *false*
**for** $a_j \in A$ do
if ***match***$(a_i, a_j) > \alpha_{sup}$ *then*
if ***value***$(m, a_j) >$ ***value***$(m, a_i)$ *then*
*remove* $\longleftarrow$ *true*
**else** remove then
$A_{sup} \longleftarrow A_{sup} \cup a_i$
***return*** $A_{sup}$

**Figure 4.17: Nonmax Suppression Algorithm**

The algorithm detects the object and class probabilities with confidence scores. **Figure 4.18** show the **YoloV5** algorithm being run in real-time on a dashcam. The algorithm detected objects in the frames by indicating the classes they belong to and the confidence scores representing how sure it is of the objects.

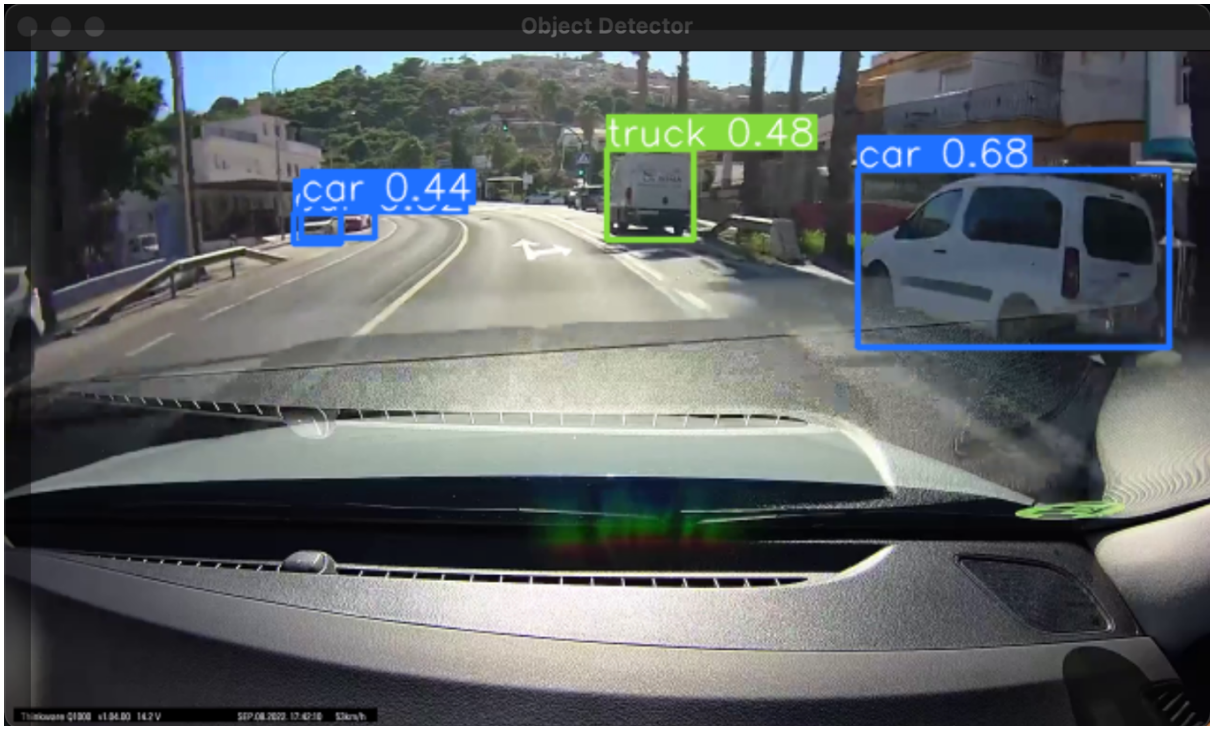



**Figure 4.18: YoloV5 running in real-time on a dashcam**

## 4.2 Data Representation [Diagram + Description]

Data Presentation plays an essential role in any project. The produced data from multiple intelligent systems implementing the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) can be presented in various patterns like text, table, or graph. Each pattern has some merits and demerits. The purpose of data representation is to record and transmit information across two or more ends and to determine the performance of the system in focus. In this case, the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**).

### 4.2.1 Drowsiness Detection System (DDS)

To evaluate the **D**rowsiness **D**etection **S**ystem (**DDS**) a total of **10** tests are performed, **5** tests in medium-lighting conditions and **5** in Dim-lighting conditions. Each test has **50** detections in total. Meaning during a single test, the driver shows drowsiness signs exactly **50** times. This equates to **500** total drowsiness detections; **250** in medium-lighting and **250** in dim-lighting conditions.

**Table 4.1: Drowsiness Detection System (DDS) evaluation Table**

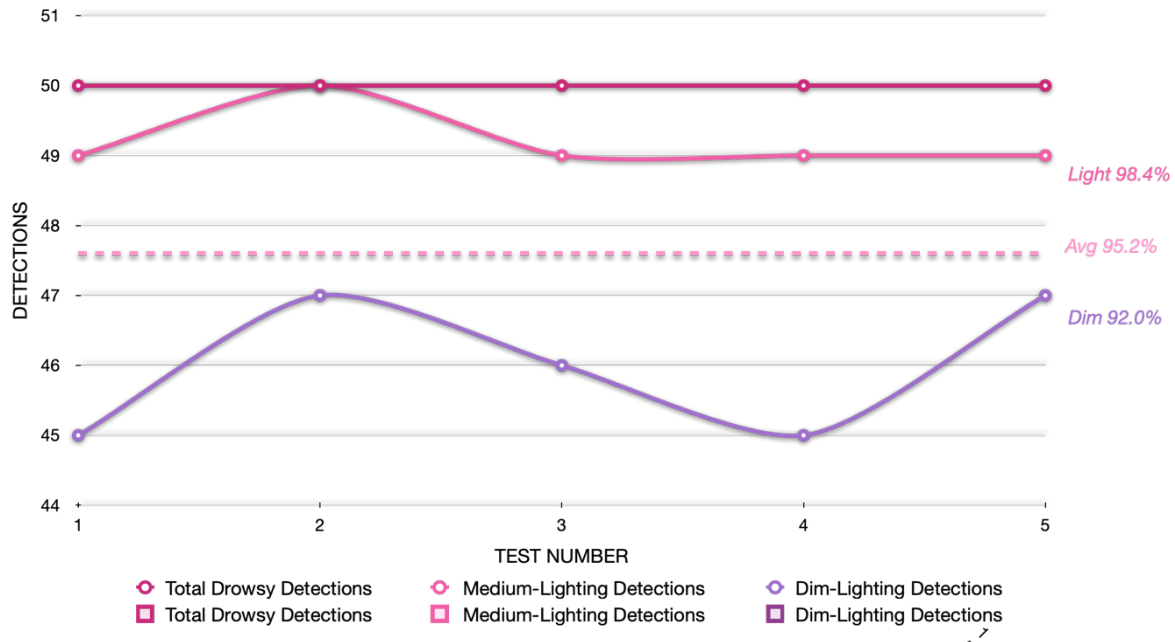
| **Test No** | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **Total Drowsy Detections** | 50 | 50 | 50 | 50 | 50 |
| **Medium-Lighting Detections** | 49 | 50 | 49 | 49 | 49 |
| **Dim-Lighting Detections** | 45 | 47 | 46 | 45 | 47 |
| **Avg Accuracy (%)** | 94% | 97% | 95% | 94% | 96% |

In **Table 4.1**, **Test No** show the test number of each of the **5** tests. **Total Drowsy Detections** show the total number of detections in each test. **Medium-lighting Detections** show the number of successful detections in each test performed in medium-lighting conditions. **Dim-lighting Detections** show the number of successful detections in each test performed in dim-lighting conditions. **Avg accuracy (%)** show the accuracy of the **D**rowsiness **D**etection **S**ystem (**DDS**) under medium and dim lighting conditions. The average accuracy of the system is calculated using the equation in **Equation 4.6**.

$$AvgDDSAccuracy = \frac{\left(\frac{light + dim}{2}\right)}{50} \times 100$$

**Equation 4.6: The average accuracy function of DDS**

The evaluation in **Figure 4.19** show that the average medium-lighting performance of the **D**rowsiness **D**etection **S**ystem (**DDS**) is **98.4%** for a total of **246** successful detections out of 250. The average performance of dim-lighting conditions is **92%** for a total of **230** successful detections out of **250**. The evaluation show that the **D**rowsiness **D**etection **S**ystem (**DDS**) overall performance is **95.2%** for a total of **476** successful detections out of **500** total detections in both medium-lighting and dim-lighting conditions. The evaluation also show that the **D**rowsiness **D**etection **S**ystem (**DDS**) performs **6.4%** better in medium-lighting conditions as appose to the dim-lighting conditions.




| TEST NO | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Total Drowsy Detections | 50 | 50 | 50 | 50 | 50 |
| Medium-Lighting Detections | 49 | 50 | 49 | 49 | 49 |
| Dim-Lighting Detections | 45 | 47 | 46 | 45 | 47 |

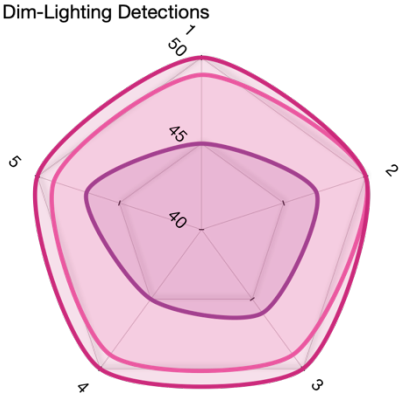



**Figure 4.19: Drowsiness Detection System (DDS) Performance evaluation**

The **D**rowsiness **D**etection **S**ystem (**DDS**) generates satisfying results with high accuracy. The experimental results show that the proposed technique identifies the drowsiness of the driver accurately in different circumstances of the environment, like weather conditions and various illumination.

### 4.2.2 Yawning Detection System (YDS)

Since **Y**awning **D**etection **S**ystem (**YDS**) is developed on the same methodology as the **D**rowsiness **D**etection **S**ystem (**DDS**), the evaluation method is almost the same. A total of **10** tests are performed, **5** tests in medium-lighting conditions and **5** in dim-lighting conditions just as in the **D**rowsiness **D**etection **S**ystem (**DDS**). Each test with **50** detections in total. Meaning during a single test, the driver yawns exactly **50** times. This equates to **500** total yawning detections; **250** in medium-lighting conditions and **250** in dim-lighting conditions.

**Table 4.2: Yawning Detection System (YDS) evaluation Table**

| **Test No** | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **Total Yawning Detections** | 50 | 50 | 50 | 50 | 50 |
| **Medium-Lighting Detections** | 49 | 48 | 48 | 49 | 47 |
| **Dim-Lighting Detections** | 46 | 47 | 44 | 45 | 46 |
| **Avg Accuracy (%)** | 95% | 95% | 92% | 94% | 93% |

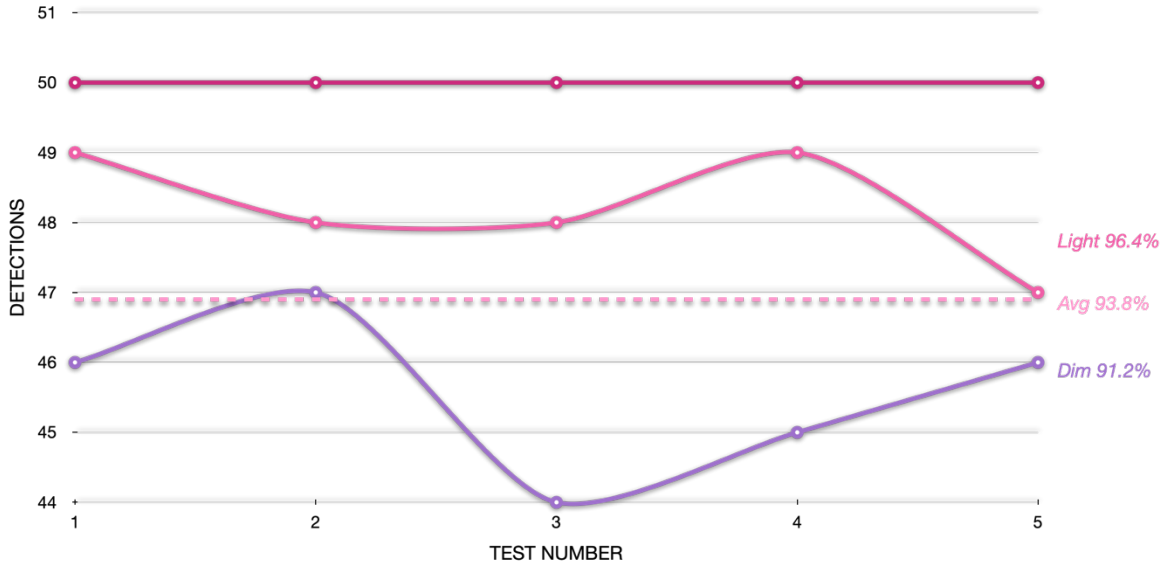
**Table 4.2 Test No** show the test number of each of the **5** tests. **Total Yawning Detections** show the total number of detections in each test. **Medium-lighting Detections** show the number of successful detections in each test performed in medium-lighting conditions. **Dim-lighting Detections** show the number of successful detections in each test performed in dim-lighting conditions. **Avg accuracy (%)** show the accuracy of the **Y**awning **D**etection **S**ystem (**YDS**) under medium and dim lighting conditions. The average accuracy of the system is calculated using the equation in **Equation 4.7**.

$$AvgYDSAccuracy = \frac{\left(\frac{light + dim}{2}\right)}{50} \times 100$$

**Equation 4.7: The average accuracy function of YDS**

The evaluation in **Figure 4.20** show that the average medium-lighting performance of the **Y**awning **D**etection **S**ystem (**DDS**) is **96.4%** for a total of **241** successful detections out of 250. The average performance of dim-lighting conditions is **91.2%** for a total of **228** successful detections out of **250**. The evaluation show that the **Y**awning **D**etection **S**ystem (**YDS**) overall

performance is **93.8%** for a total of **469** successful detections out of **500** total detections in both medium-lighting and dim-lighting conditions. The evaluation also show that the **Y**awning **D**etection **S**ystem (**DDS**) performs **5.2%** better in medium-lighting conditions as appose to the dim-lighting conditions.




| TEST NO | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Total Drowsy Detections | 50 | 50 | 50 | 50 | 50 |
| Medium-Lighting Detections | 49 | 48 | 48 | 49 | 47 |
| Dim-Lighting Detections | 46 | 47 | 44 | 45 | 46 |

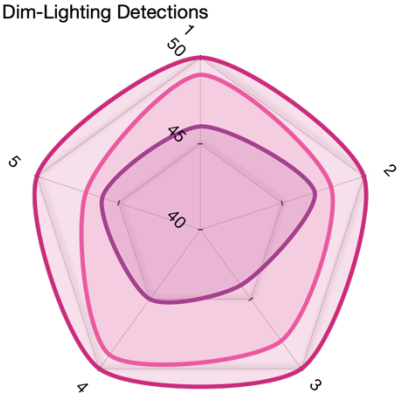



**Figure 4.20: Yawning Detection System (DDS) Performance evaluation Charts**

The **Y**awning **D**etection **S**ystem (**DDS**) generates satisfying results with acceptable accuracy. The experimental results show that the proposed technique identifies the yawning of the driver with decent accuracy in different circumstances of the environment, like weather conditions and various lighting conditions.

### 4.2.3 Lane Detection System (LDS)

To evaluate **L**ane **D**etection **S**ystem (**LDS**) **100** lane images are used from both medium-lighting and dim-lighting conditions. **50** images with visible lane edge under medium-lighting conditions and **50** images with visible lane edge under dim-lighting conditions respectively. Following the evaluation pattern for this project a total of **10** tests are performed, **5** tests in medium-lighting conditions and **5** in Dim-lighting conditions. Each test has **100** detections in total. Meaning during a single test, there are exactly **100** images with visible lane lines given as input to the **L**ane **D**etection **S**ystem (**LDS**). This equates to **500** total images with visible lane line edges; **250** in medium-lighting and **250** in dim-lighting conditions.

**Table 4.3: Lane Detection System (LDS) evaluation table**

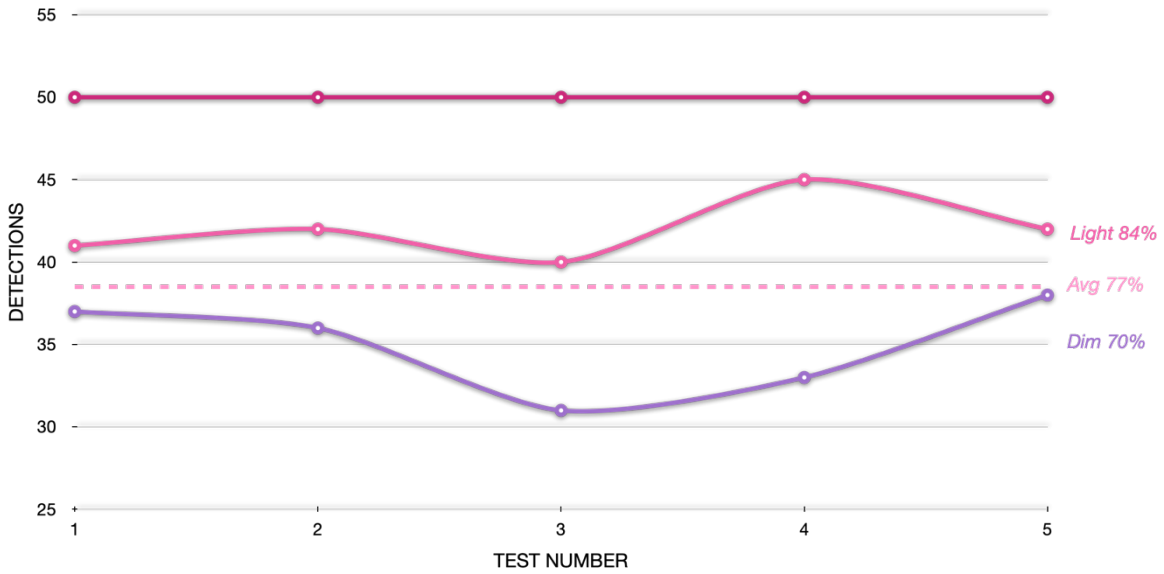
| **Test No** | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **Total Lane Detections** | 50 | 50 | 50 | 50 | 50 |
| **Medium-Lighting Detections** | 41 | 42 | 40 | 45 | 42 |
| **Dim-Lighting Detections** | 37 | 36 | 31 | 33 | 38 |
| **Avg Accuracy (%)** | 78% | 78% | 71% | 78% | 80% |

**Table 4.2 Test No** show the test number of each of the **5** tests. **Total Lane Detections** show the total number of detections in each test. **Medium-lighting Detections** show the number of successful detections in each test performed in medium-lighting conditions. **Dim-lighting Detections** show the number of successful detections in each test performed in dim-lighting conditions. **Avg accuracy (%)** show the accuracy of the **L**ane **D**etection **S**ystem (**LDS**) under medium and dim lighting conditions. The average accuracy of the system is calculated using the equation in **Equation 4.8**.

$$AvgLDSAccuracy = \frac{\left(\frac{light + dim}{2}\right)}{50} \times 100$$

**Equation 4.8: The average accuracy function of LDS**

The evaluation in **Figure 4.21** show that the average medium-lighting performance of the **L**ane **D**etection **S**ystem (**LDS**) is **77%** for a total of **210** successful detections out of 250. The average performance of dim-lighting conditions is **70%** for a total of **175** successful detections out of **250**. The evaluation show that the **L**ane **D**etection **S**ystem (**LDS**) overall performance is **77%** for a total of **385** successful detections out of **500** total detections in both medium-lighting and dim-lighting conditions. The evaluation also show that the **L**ane **D**etection **S**ystem (**LDS**) performs **14%** better in medium-lighting conditions as appose to the dim-lighting conditions.




| TEST NO | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Total Drowsy Detections | 50 | 50 | 50 | 50 | 50 |
| Medium-Lighting Detections | 41 | 42 | 40 | 45 | 42 |
| Dim-Lighting Detections | 37 | 36 | 31 | 33 | 38 |

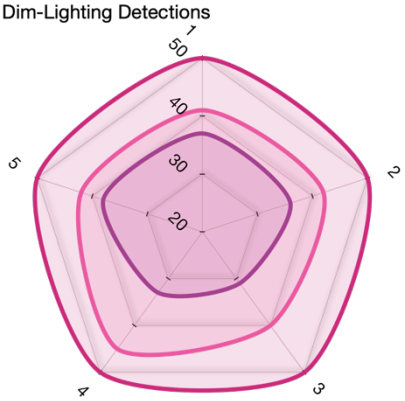



**Figure 4.21: Lane Detection System (LDS) Performance evaluation Charts**

The proposed system generates satisfying results with sharp curvature, sharp lane changes, hindrances, and lens flashes due to using the temporal consistency of lane width on each scan line. The experimental results show that the proposed technique identifies the lane accurately in different circumstances of environment like weather conditions and various illumination.

### 4.2.4 Object Detection System (ODS)

The **O**bject **D**etection **S**ystem (**ODS**) is evaluated using Berkeley DeepDrive dataset (**BDD**). A total of **3** tests are performed. Seven of the most important classes (**Peron**, **Car**, **Bus**, **Truck**, **Bike**, **Traffic Light**, **Traffic Sign**) of the **O**bject **D**etection **S**ystem (**ODS**) are used for this evaluation. At the end of each test, the average confidence of each class is recorded.

**Table 4.4: YoloV5 Model Specifications**

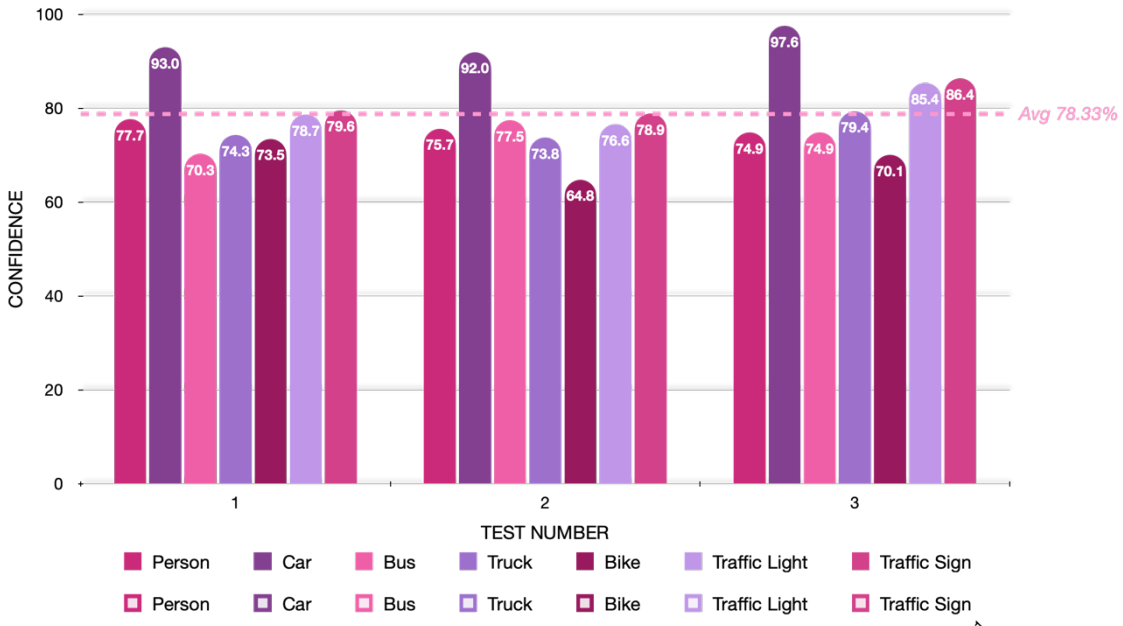
| **Model** | YoloV5n |
|---|---|
| **Pixel size** | 224 |
| **Avg Accuracy (%)** | 78(%) |
| **Total Epochs** | 90 |
| **Speed on CPU (ms)** | 3.3 ms |
| **Speed on NVIDIA tensorRT-v100 (ms)** | 0.5 ms |

**Table 4.5 Test No** show the test number of each of the **3** tests. **Person Detections, Car Detections, Bus Detections, Truck Detections, Bike Detections, Traffic Light Detections,** and, **Traffic Sign Detections** all show the average confidence level of each class during each test. **Avg accuracy (%)** show the accuracy of all seven classes of the **O**bject **D**etection **S**ystem (**ODS**) during each test. **Speed (FPS)** show the average frame rate of the system during each test. **Input Frame Size** show the video resolution on which the test is performed.

**Table 4.5: Object Detection System (ODS) evaluation table**

| **Test No** | 1 | 2 | 3 |
|---|---|---|---|
| **Person Detection** | 77.70 | 75.70 | 74.90 |
| **Car Detection** | 93.00 | 92.00 | 97.60 |
| **Bus Detection** | 70.30 | 77.50 | 74.90 |
| **Truck Detection** | 74.30 | 73.80 | 79.40 |
| **Bike Detection** | 73.50 | 64.80 | 70.10 |
| **Traffic Light Detection** | 78.70 | 76.60 | 85.40 |
| **Traffic Sign Detection** | 79.60 | 78.90 | 86.40 |
| **Avg Accuracy (%)** | 78.15(%) | 75.61 (%) | 81.24 (%) |
| **Speed (FPS)** | 48.56 | 57.67 | 41.20 |
| **Input Frame Size** | 512x512 | 512x512 | 704x704 |

The evaluation in **Figure 4.22** shows that the average accuracy of the object detection system for classes Person Detection is **76.1%**, Car Detection is **94%**, Bus Detection is **74.2%**, Truck Detection is **77%**, Bike Detection is **69.4%,** Traffic Light Detection is **80.2 %**, and Traffic Sign Detection is **81.6%**. The evaluation show that the **O**bject **D**etection **S**ystem (**ODS**) overall performance is **78.81%** for all **3** performed tests.




| Test No | 1 | 2 | 3 |
|---|---|---|---|
| Person | 77.70 | 75.70 | 74.90 |
| Car | 93.00 | 92.00 | 97.60 |
| Bus | 70.30 | 77.5 | 74.90 |
| Truck | 74.30 | 73.80 | 79.40 |
| Bike | 73.50 | 64.80 | 70.10 |
| Traffic Light | 78.70 | 76.60 | 85.40 |
| Traffic Sign | 79.60 | 78.90 | 86.40 |

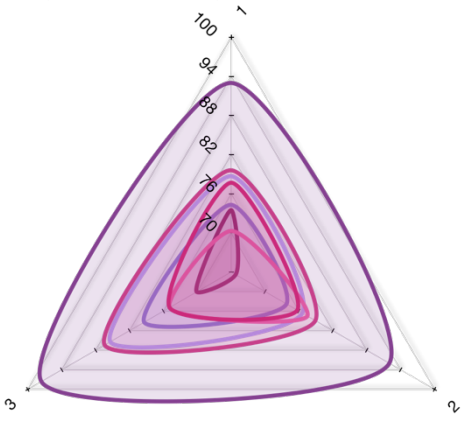



**Figure 4.22: Object Detection System (ODS) Performance evaluation**

The proposed algorithm generates acceptable results along all of the important classes. The experimental results show that the **O**bject **D**etection **S**ystem (**ODS**) identifies the objects of different classes adequately in different circumstances of environment like weather conditions and various illumination.

## 4.3 Process Flow/Representation

The process flow of the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) is illustrated in the flowchart diagram shown in **Figure 4.23** and the data flow diagram in **Figure 4.24**. The process starts with taking in the video frames from the video cameras. This will be the camera for **D**rowsiness **D**etection **S**ystem (DDS) and **Y**awning **D**etection **S**ystem (**YDS**) and the dash camera for **L**ane **D**etection **S**ystems (**LDS**) and **O**bject **D**etection **S**ystem (**ODS**).

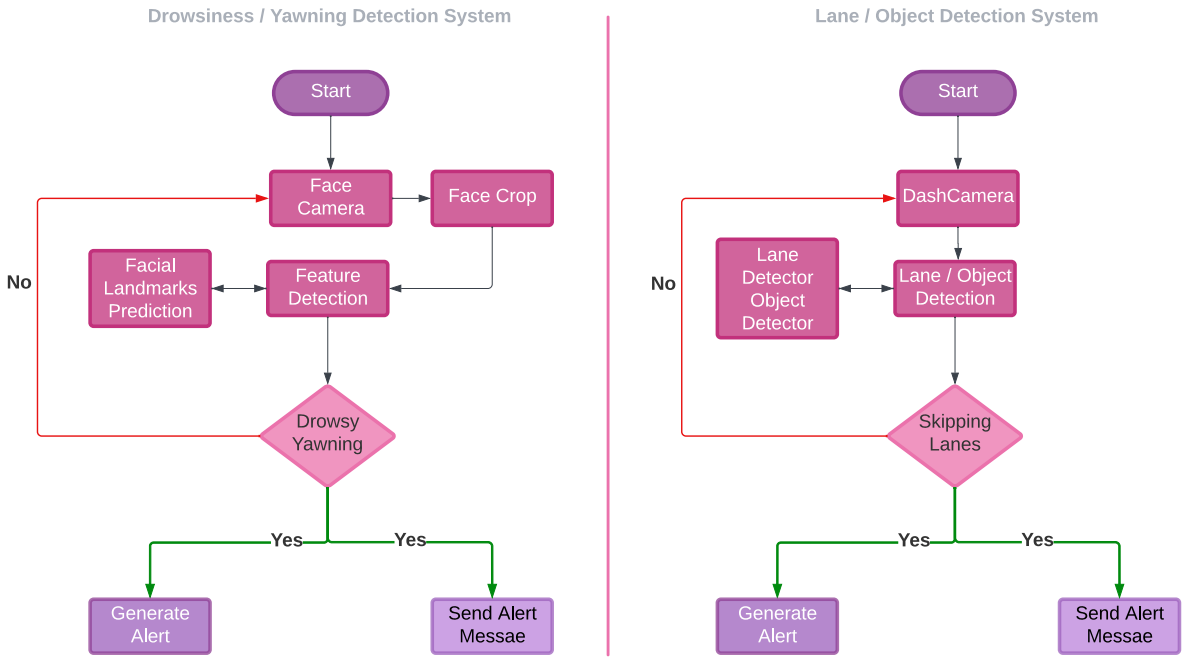



**Figure 4.23: Driving Negligence Dissuader System (DNDS) Flowchart Diagram**

Once received, the frames then pass through a series of steps for the system to produce the intoned output. In the case of the **D**rowsiness **D**etection **S**ystem (**DDS**) and **Y**awning **D**etection **S**ystem (**YDS**), the frames first go through some pre-processing steps where they are cropped, color transformed, and resized. From there the frames are fed to the feature detection unit where the system determines if the user is Drowsy, Yawning, or not based on the **E**ye **A**spect **R**atio (**EAR**) and **M**outh **A**spect **R**atio (**MAR**) values. In case of any positive detections, the system proceeds to generate an alert. In case of Negative detections, the system skips to the next frame in line. The process cycles back for every frame received from the camera.

In the case of **L**ane **D**etection **S**ystems (**LDS**) and **O**bject **D**etection **S**ystem (**ODS**), the process is somewhat similar. The frames are received from the input device, in this case, the dash camera. After some pre-processing, the frames are then sent to their designated intelligent system to get processed. This process continues for every frame received from the dash camera module.

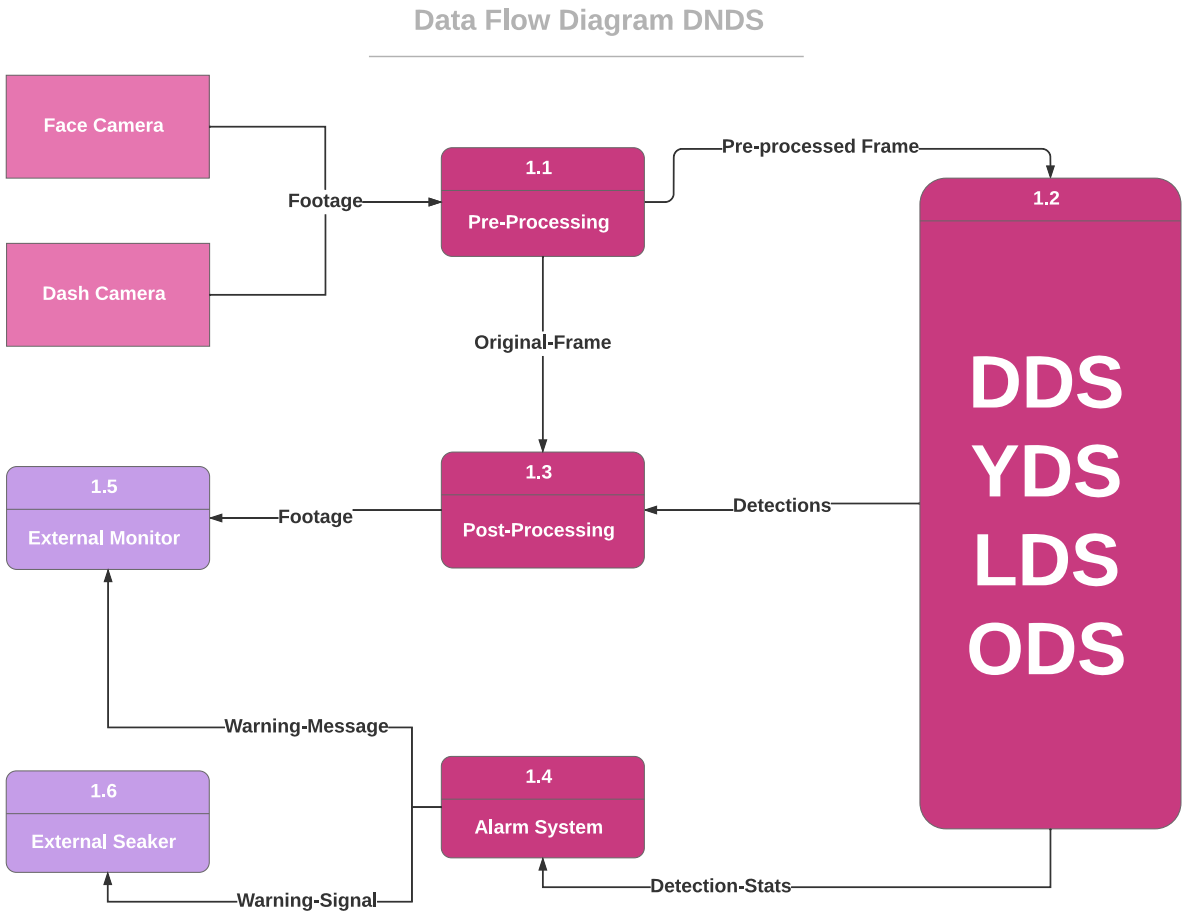



**Figure 4.24: Driving Negligence Dissuader System (DNDS) Data Flow Diagram**

## 4.4 Design Models [along with descriptions]

Design modeling will provide a variety of different views of the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**), like the architecture and component design. Designing a model is an important phase and is a multi-process that represents the data structure, program structure, and procedural details. Different methods like data-driven, pattern-driven, or object-oriented methods are used for constructing the design model. In **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) it represents the features of the system that helps us to develop it effectively such as the architecture, and the component level detail. **Figure 4.25** illustrates the system architecture design, while **Figure 4.26** illustrates the component-level design of the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**).

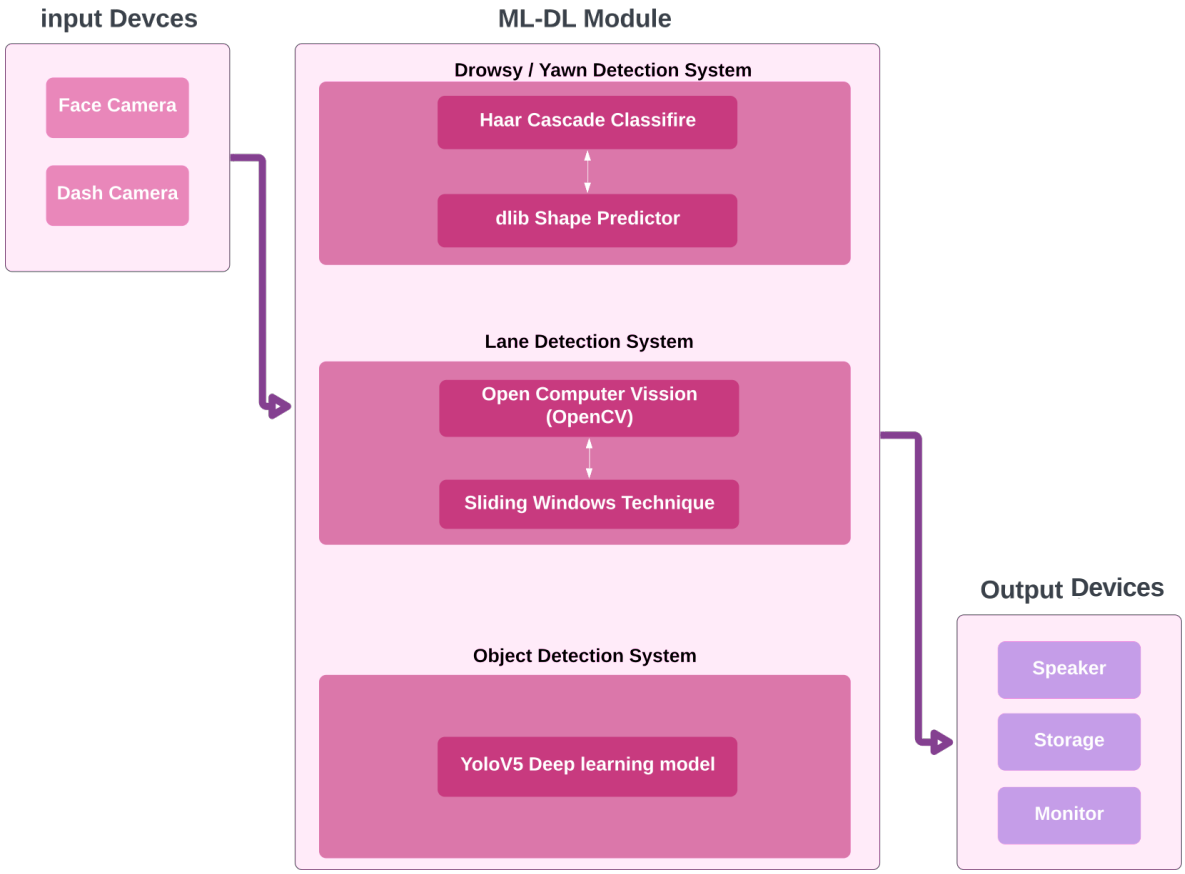



**Figure 4.25: Driving Negligence Dissuader System (DNDS) Architecture**

| Item | Description |
|---|---|
| Input Devices | Live footage is recorded through input devices |
| ML – DL Module | Multiple systems to detect Drowsiness, Yawning, Lane, and objects |
| Output Devices | Takes the processed information and store and/or display it. |

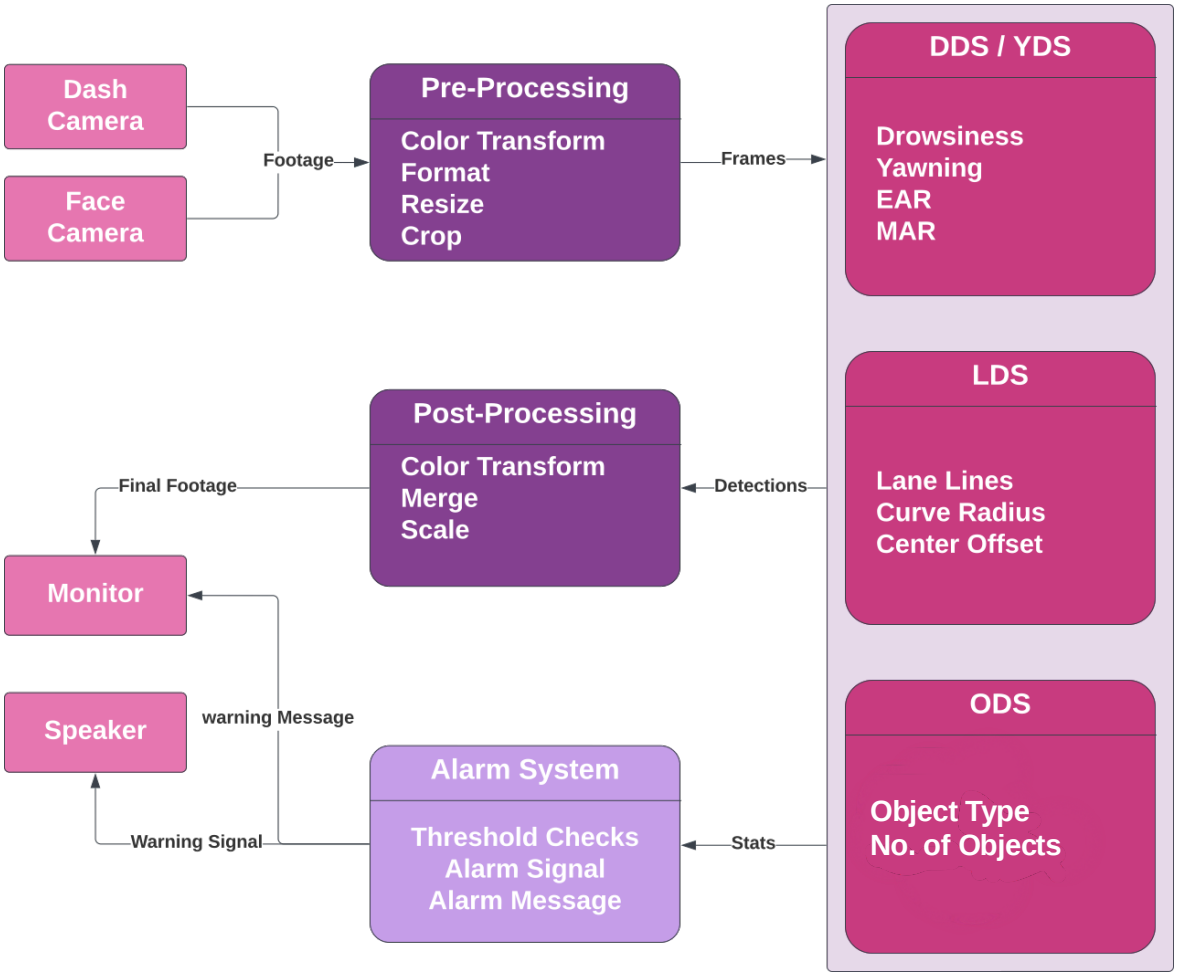



**Figure 4.26: Driving Negligence Dissuader System (DNDS) Components**

| Component | Description |
|---|---|
| Face Camera | Captures video of the driver |
| Dash Camera | Captures video of the road ahead |
| Pre-Processor | Prepare the frames (Color, size, form) for each detection system |
| DDS / YDS | Drowsy / Yawn detector that detects drowsiness and yawning |
| LDS | Detects lane lines, road curvature, and vehicle center offset |
| ODS | Detects objects in front of the vehicle |
| Post-Processor | Re-scales and overlays detections on the original footage |
| Alarm System | Checks for any detections and send out warning signal and message |
| Monitor | Display the final footage, detection information, and warnings |
| Speaker | Warns the driver of the detections when a warning signal is received |

# Chapter 5
# Implementation

## 5 Implementation

The strategic planning outlined for the system helps us achieve our project objectives. In this chapter, we discuss how we put the project plan for the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) into action to produce the deliverables, otherwise known as project implementation. We discuss the major algorithms used along with their pseudo code as well as the external APIs used in the system. Lastly, we discuss the user interface and its design.

### 5.1 Algorithm

The major algorithms used in the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) are **D**rowsiness **D**etection **S**ystem (**DDS**), **Y**awning **D**etection **S**ystem (**YDS**), **L**ane **D**etection **S**ystem (**LDS**), and **O**bject **D**etection **S**ystem (**ODS**). The pseudo-code is used to plan the algorithms by sketching out the structure of the program before the actual coding takes place. The algorithms are illustrated using Algorithm Flowcharts to give us an idea of how the different processes of the system work together to achieve the desired goal.

### 5.1.1 Drowsiness Detection System (DDS) / Yawning Detection System (YDS)

The **D**rowsiness **D**etection **S**ystem (**DDS**) and **Y**awning **D**etection **S**ystem (**YDS**) share most of the basic processes. Hence the designed algorithm is utilized in a way to perform both tasks at once. The algorithm flowchart for **D**rowsiness **D**etection **S**ystem (**DDS**) and **Y**awning **D**etection **S**ystem (**YDS**) is illustrated in **Figure 5.1**. The pseudo-code is as follows.

*Initialize the system*
*Load Haar Cascade Classifier and DLib Shape predictor*
*Open the attached face camera and start the Video Stream*
*If the camera is opened*
    *Read video frame(s)*
    *Pre-process the Frame*
    *Scan frame for facial features to detect face*
    *If a face is Detected*
        *Crop Face region*
        *Detect facial landmarks with DLib shape predictor*
        *Calculate EAR and MAR using Facial Landmarks*
        *If EAR or MAR is above the threshold*
            *Alert the driver*
            *Increase Detection count by one*
        *Else*
            *Read Another Frame*
    *Else*
        *Read Another Frame*
*Else*
    *Close the System*

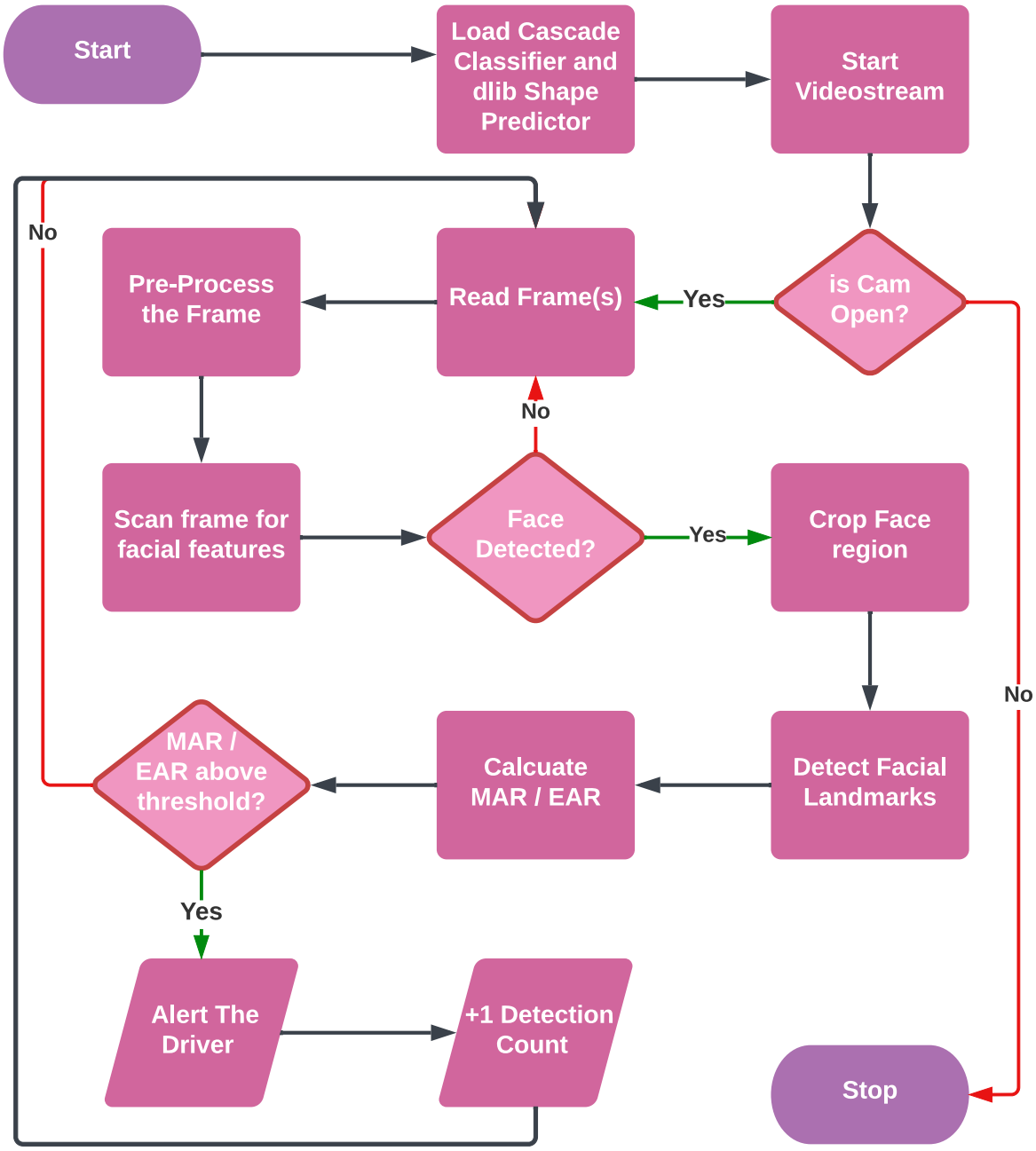



**Figure 5.1: DDS / YDS Algorithm Flowchart**

### 5.1.2 Lane Detection System (LDS)

The **L**ane **D**etection **S**ystem (**LDS**) is using **O**pen **C**omputer **V**ision Library (**OpenCV**). The individual frames go through a series of image processing techniques powered by **OpenCV** to make out the lane lines from the image. The algorithm flowchart for **L**ane **D**etection **S**ystem (**LDS**) is illustrated in **Figure 5.2**. The pseudo-code is as follows.

*Initialize the system*
*Open the attached dash camera and start the Video Stream*
*If the camera is opened*
    *Read video frame(s)*
    *Pre-process the Frame*
    *Set ROI on the frame*
    *Apply HLS Transform to the frame*
    *Apply Perspective transform on the frame*
    *Perform a Sliding Window Search on the Frame*
    *If Lane is Detected*
        *Highlight the Lane Lines form Sliding Window*
        *Warp, Crop the frame*
        *Overlay the Lane frame onto the original frame*
    *Else*
        *Read Another Frame*
*Else*
    *Close the System*

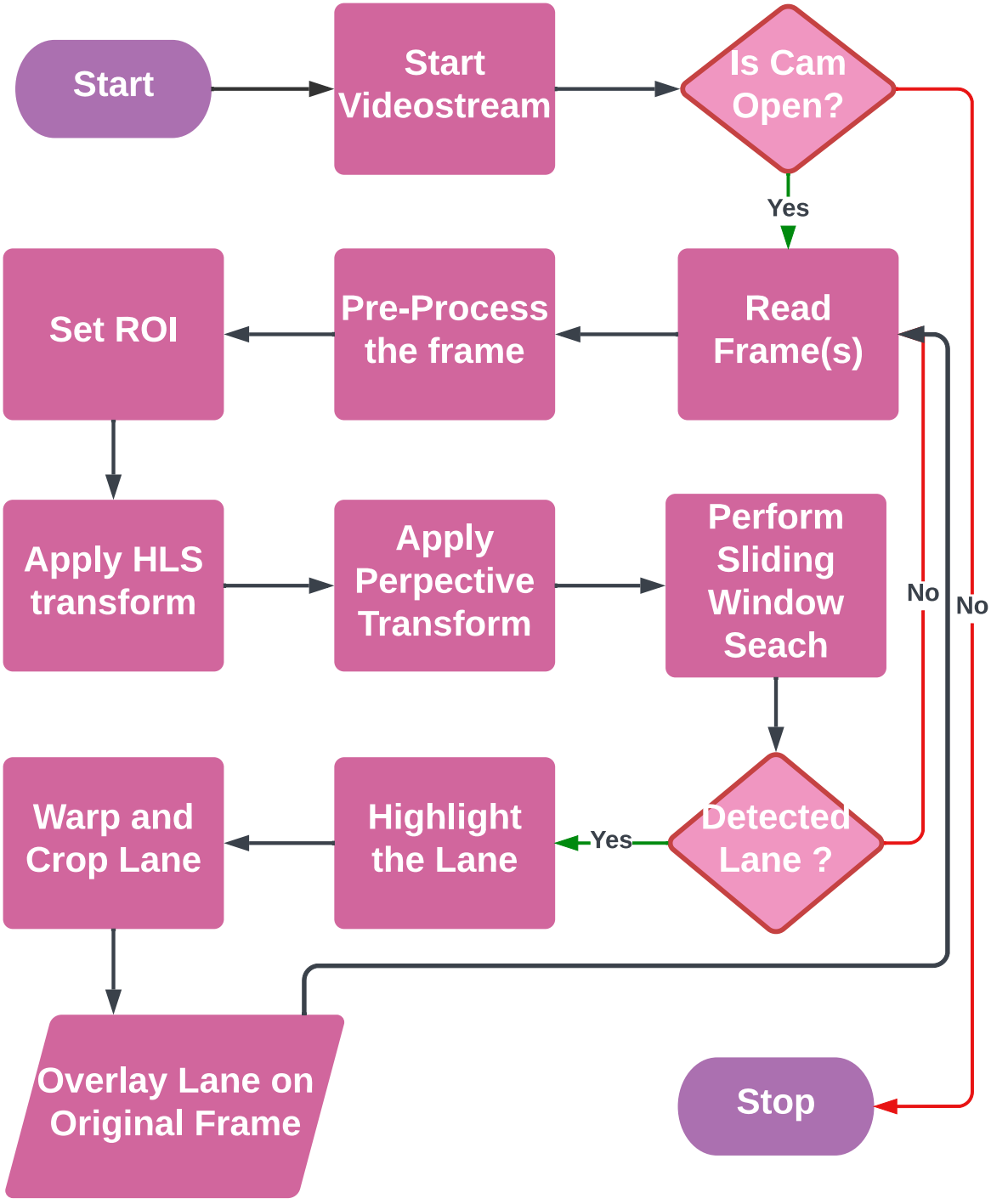



**Figure 5.2: LDS Algorithm Flowchart**

### 5.1.3 Object Detection System (ODS)

The **O**bject **D**etection **S**ystem (**ODS**) is using **YoloV5** deep learning model trained on coco classes. The model can detect up to 80 different classes. In the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) however, we are only focusing on the major classes that have to do with roads and traffic like Humans, Cars, Bikes, Trucks, Busses, Stop signs, etc. The algorithm flowchart for **O**bject **D**etection **S**ystem (**ODS**) is illustrated in **Figure 5.3**. The pseudo-code is as follows.

*Initialize the system*
*load local YoloV5 Object Detection model*
*Open the attached dash camera and start the Video Stream*
*If the camera is opened*
    *Read video frame(s)*
    *Pre-process the Frame*
    *detect Objects in the frame with the YoloV5 model*
    *If Objects are Detected*
        *Enclose objects in bounding boxes*
    *Else*
        *Read Another Frame*
*Else*
    *Close the System*

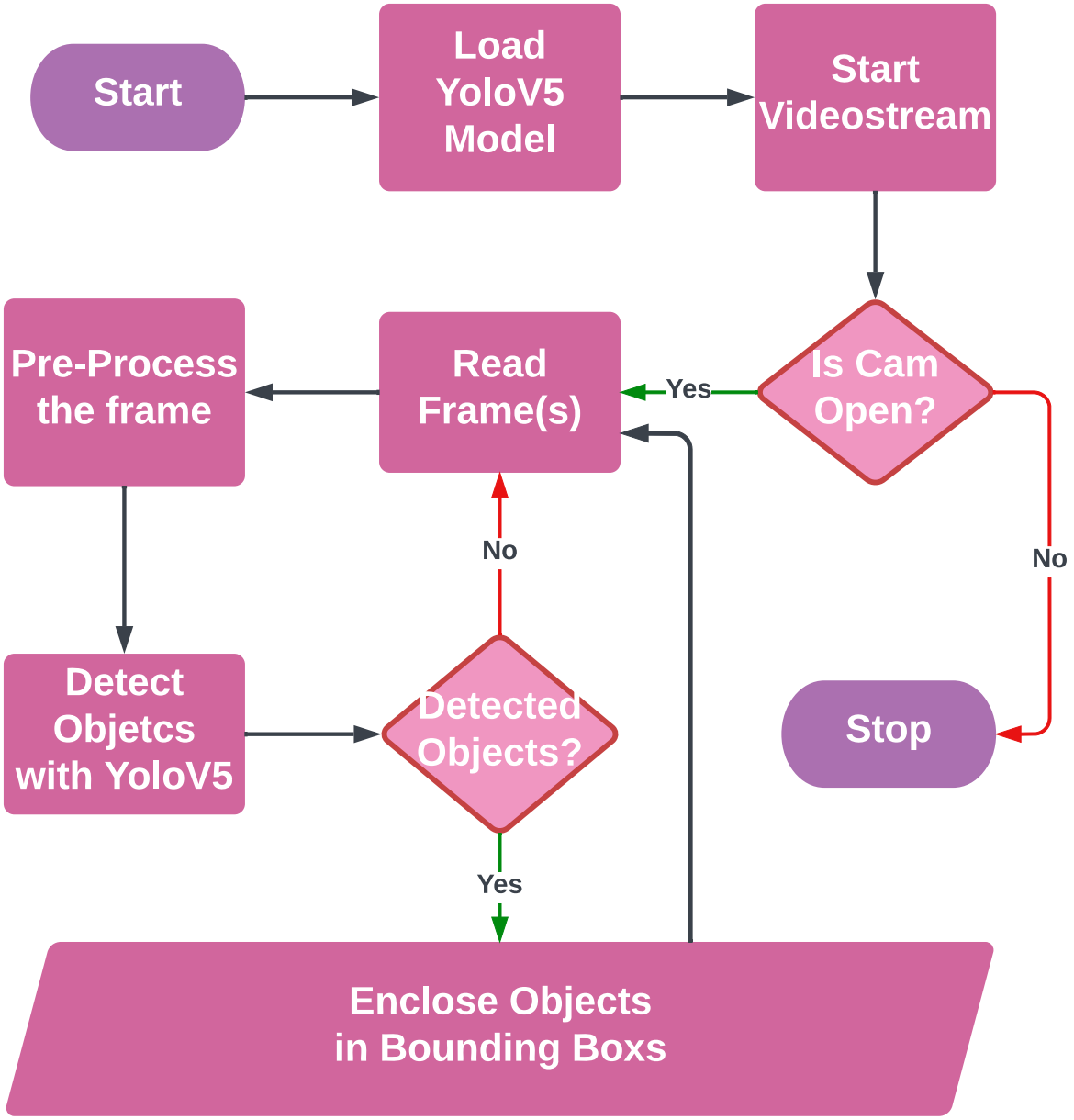



**Figure 5.3: ODS Algorithm Flowchart**

## 5.2 External APIs

The API used in the project **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) is called **PyQt** API. **PyQt** is a set of Python bindings for The **Qt** Company's **Qt** application framework and runs on all platforms supported by **Qt** including **Windows**, **macOS**, **Linux**, **iOS**, and **Android**. **PyQt6** supports **Qt v6**, **PyQt5** supports **Qt v5** and **PyQt4** supports **Qt v4**. The bindings are implemented as a set of Python modules and contain over 1,000 classes.

**Table 5.1: Details of APIs used in the project**

| **Name of APIs** | **Description of API** | **Purpose of usage** | **List down the function/class name in which it is used** |
|---|---|---|---|
| PyQt | PyQt API is a set of modules containing a large number of classes and functions. While the **QtCore** module contains non-GUI functionality for working with files and directories etc. **QtGui** module contains all the graphical controls. In addition, there are modules for working with XML **(QtXml)**, SVG **(QtSvg)**, SQL **(QtSql)**, etc. | PyQt is used to build the user interface for the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**). | • __init__.py<br>• Class DNDS<br>• __name__ == '__main__'<br>• drowsy_yawn_detection.py<br>• class StartDDS<br>• lane_detection.py<br>• class StartLDS<br>• object_detection.py<br>• class StartODS<br>• pedestrian_detection.py<br>• class StartPDS |

## 5.3 User Interface

The user interface is one of the most essential parts of any software project, and in the case of our project, there is no exception. The user interface for the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) is a graphics-based based system interface that uses icons, buttons, labels, and a mouse to manage interaction with the system. The **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) start at the home page illustrated in **Figure 5.4** and **5.5**.

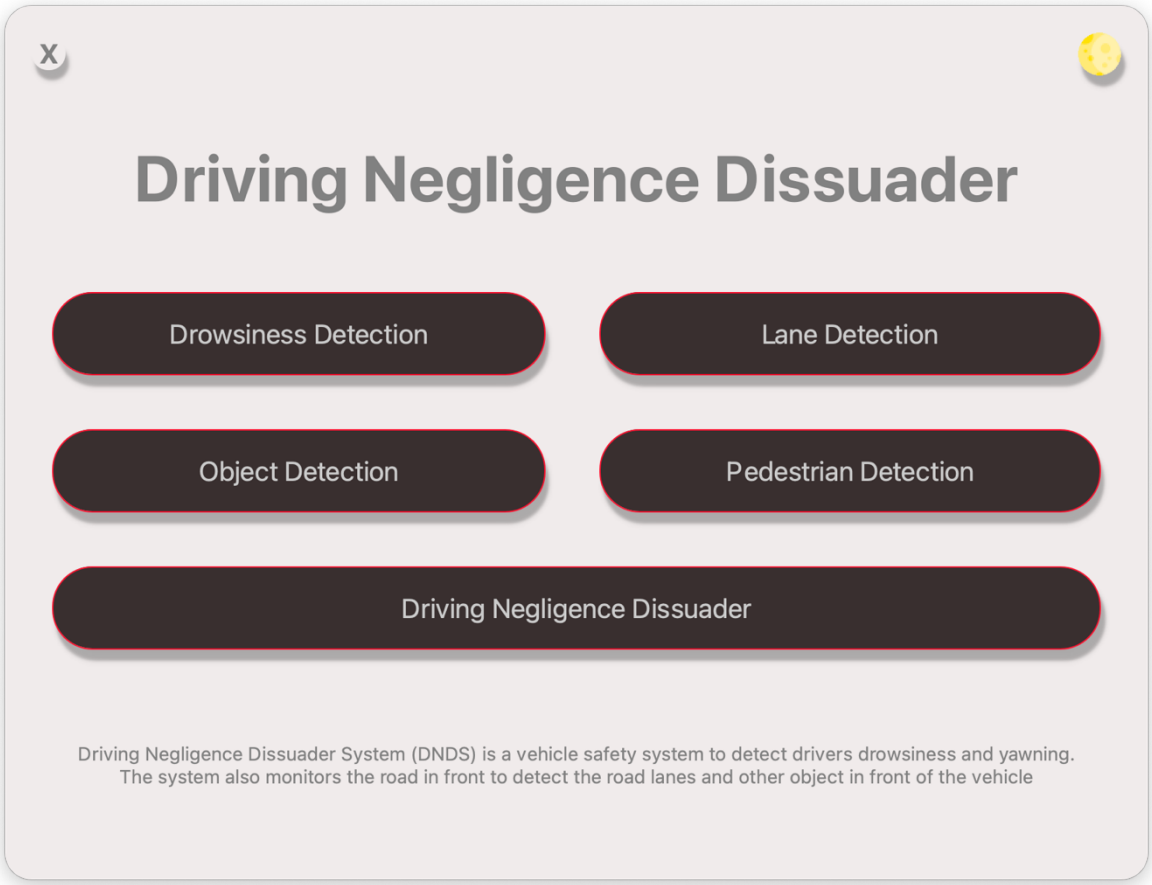



**Figure 5.4: Driving Negligence Dissuader System (DNDS) Homepage**

The homepage is where the user interacts with multiple functionalities that the system provides such as **D**rowsiness **D**etection **S**ystem (**DDS**), **Y**awning **D**etection **S**ystem (**YDS**), **L**ane **D**etection **S**ystem (**LDS**), and **O**bject **D**etection **S**ystem (**ODS**), etc. There are multiple components for the user to interact with. Most notably the **Drowsiness Detection**, **Yawning Detection**, **Object Detection**, **Pedestrian Detection.** and **Driving Negligence Dissuader** buttons. Users can start any of the four available systems either individually or simultaneously.

| Element | Description |
|---|---|
| Cross Button | The user is able to reset the application with this button |
| Moon/Sun Icon | The user is able to through Light/Dark mode |
| Header | Displays the title of the application |
| Drowsiness Detection Button | Starts the **D**rowsiness **D**etection **S**ystem (**DDS**) |
| Lane Detection Button | Starts the **L**ane **D**etection **S**ystem (**LDS**) |
| Object Detection Button | Starts the **O**bject **D**etection **S**ystem (**ODS**) |
| Pedestrian Detection Button | Starts the **P**edestrian **D**etection **S**ystem (**PDS**) (*Optional*) |
| DND Button | Starts **DDS**, **LDS**, and **ODS** on a separate page |
| Footer | Displays description of the software system |

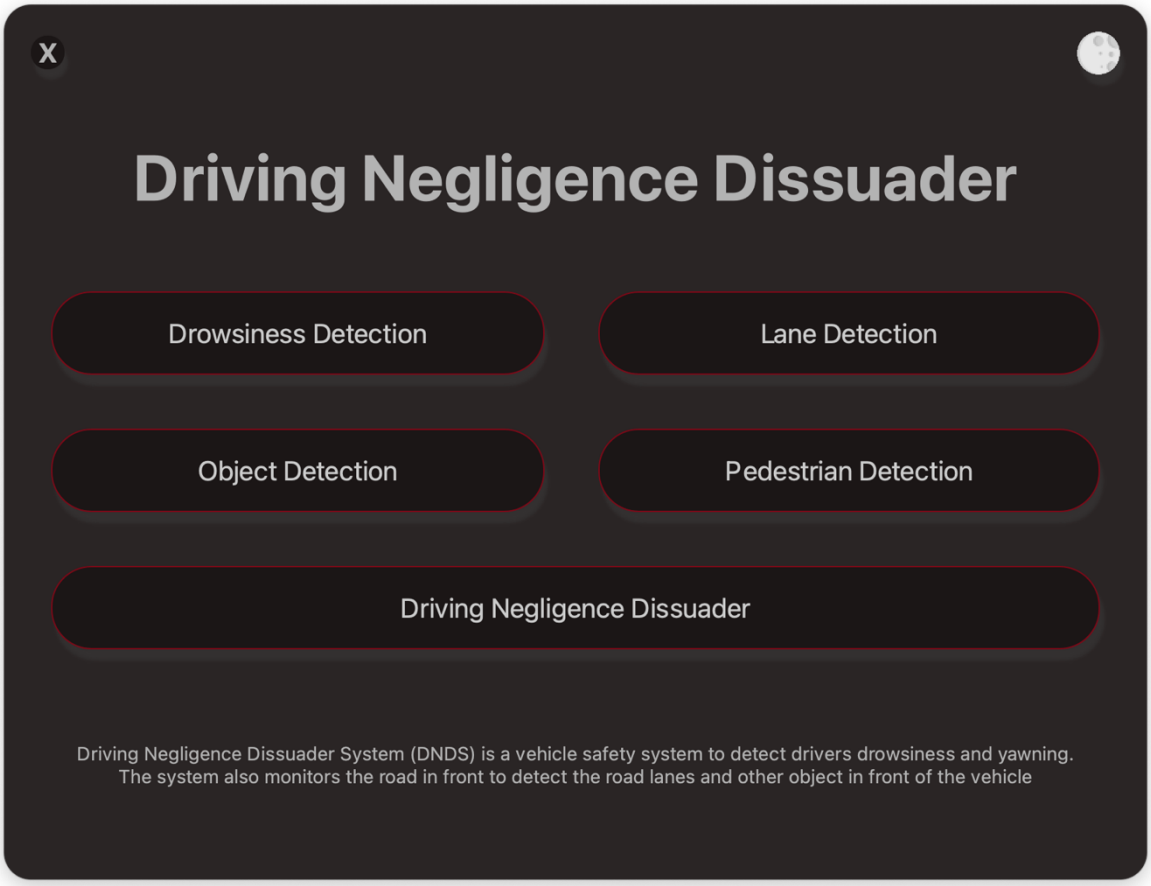



**Figure 5.5: Driving Negligence Dissuader System (DNDS) Homepage (Dark)**

When a user decides to click the **Drowsiness Detection** button on the home page, they are switched to the **DDS/YDS** page illustrated in **Figures 5.6** and **5.7**. Here the user is facilitated with information about **E**ye **A**spect **R**atio (**EAR**), **M**outh **A**spect **R**atio (**MAR**), and the number of detections. A live video session shows the drowsiness and yawing detections in real-time. The user is warned of any detections by an audio message as well as a warning message below the video footage when the system either detects drowsiness or yawning.

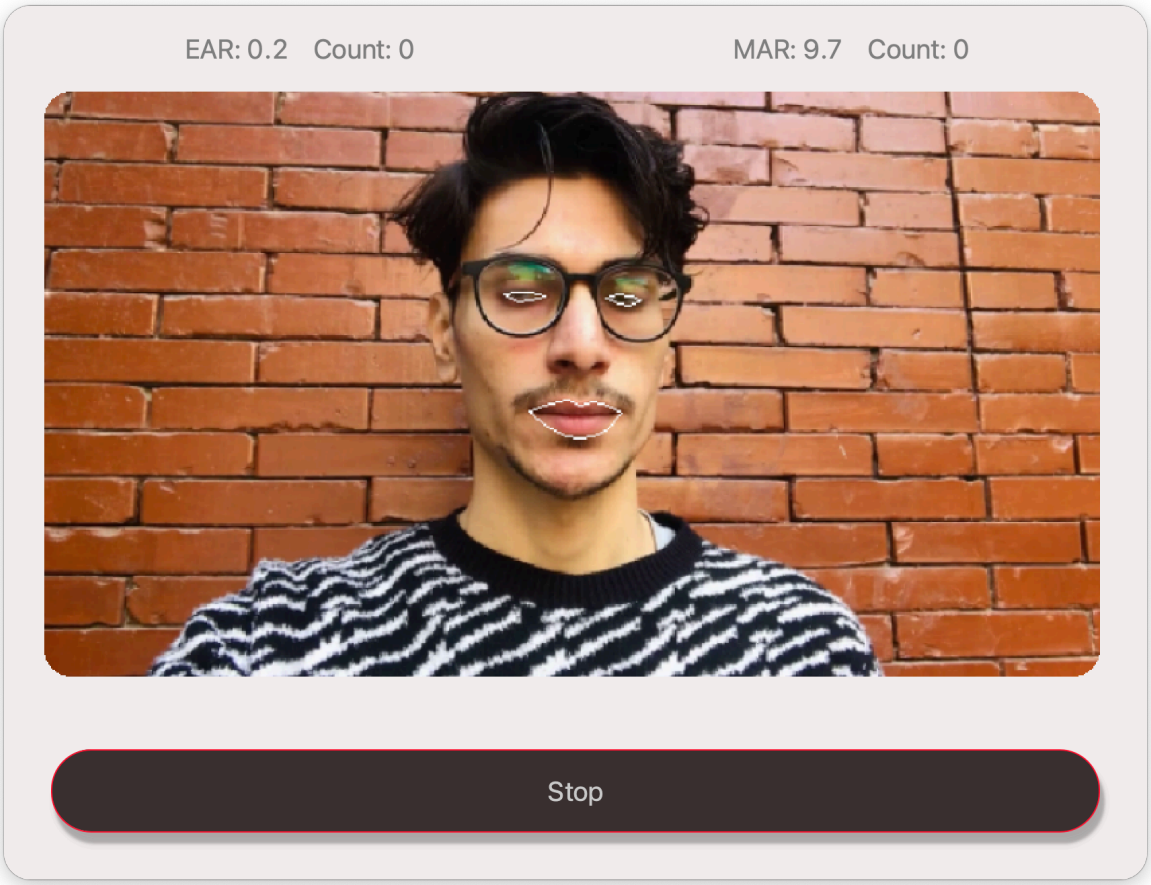



**Figure 5.6: Drowsiness Detection System (DDS)**

| Element | Description |
|---|---|
| EAR and Count | Shows Eye Aspect Ratio and Drowsy Detections count |
| MAR and Count | Shows Mouth Aspect Ratio and Yawning Detections count |
| Video Player | Displays Video footage showing detections in real-time |
| Stop Button | Stops the system and returns to the homepage |

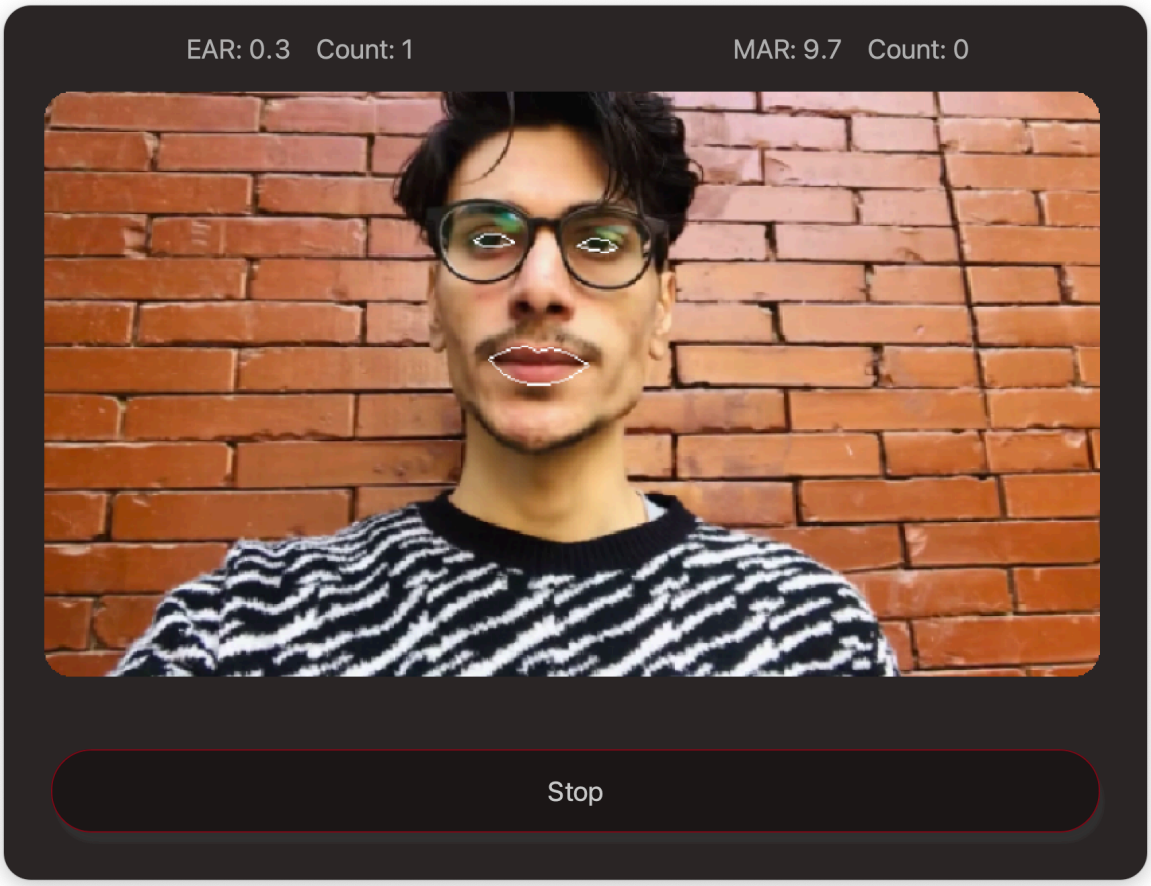



**Figure 5.7: Drowsiness Detection System (DDS) (Dark)**

When a user decides to click the **Lane Detection** button on the home page, they are switched to the **YDS** page illustrated in **Figures 5.8** and **5.9**. Here the user is facilitated with information about the **C**urve **R**adius (**CR**) and the **V**ehicle **C**enter **O**ffset (**VCO**) of the vehicle from the lane. A live video session shows the highlighted lane in real-time. The user is warned of any detections by an audio message as well as a warning message below the video footage when the system the vehicle wandering off their lane.

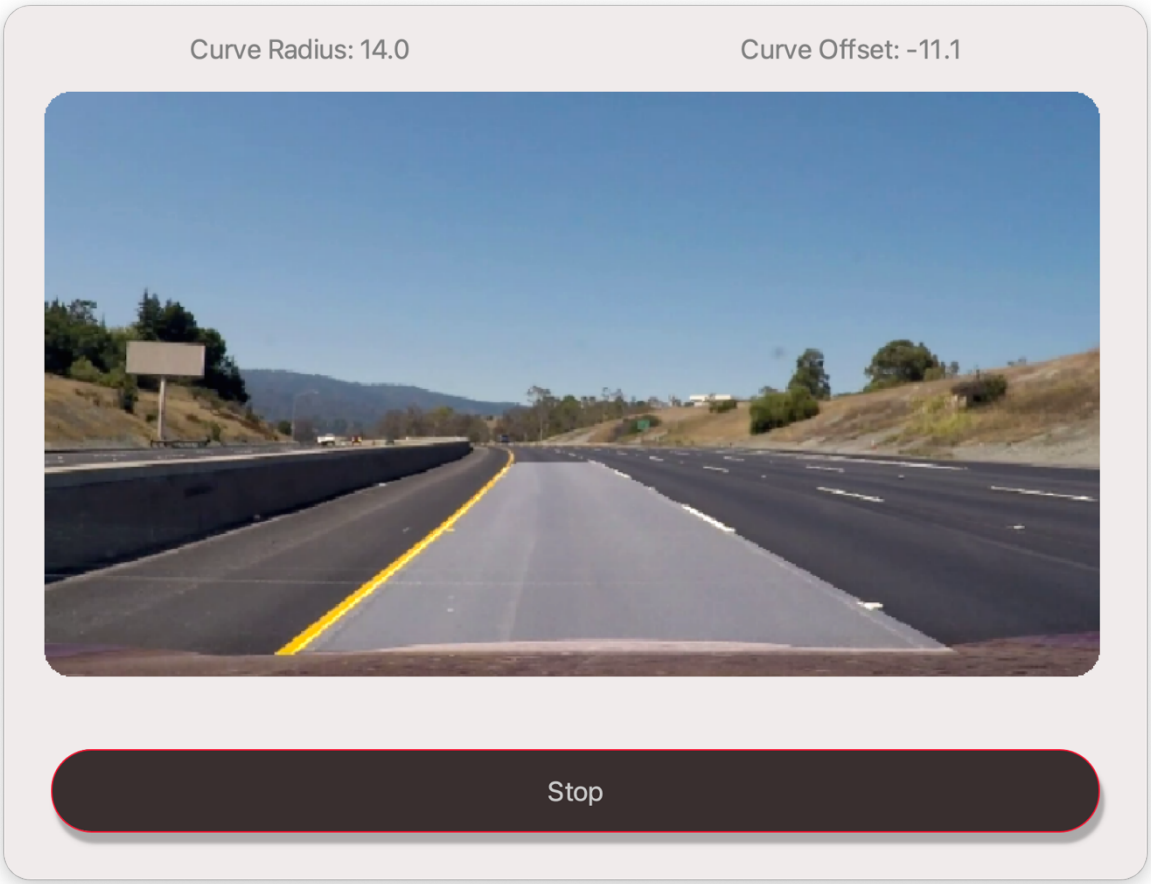



**Figure 5.8: Lane Detection System (LDS)**

| Element | Description |
|---|---|
| Curve Radius | Shows the road curvature value |
| Curve Offset | Shows the vehicle position from the center of the lane |
| Video Player | Displays Video footage showing highlighted lane in real-time |
| Stop Button | Stops the system and returns to the homepage |

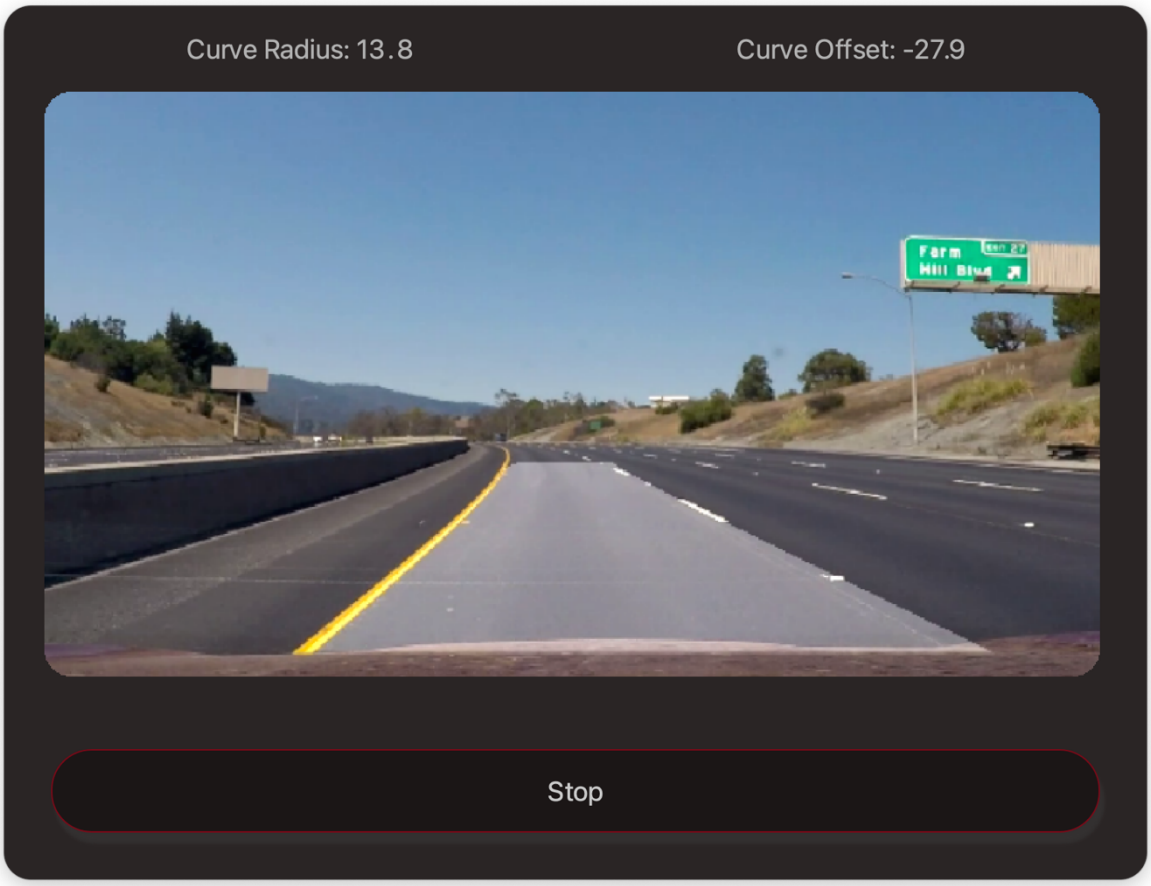



**Figure 5.9: Lane Detection System (LDS) (Dark)**

When a user decides to click the **Object Detection** button on the home page, they are switched to the **ODS** page illustrated in **Figures 5.10** and **5.11**. Here the user is facilitated with information about the detected objects and the number of items of the same class. A real-time video session shows the objects enclosed in bounding boxes and the detection confidence for that object.

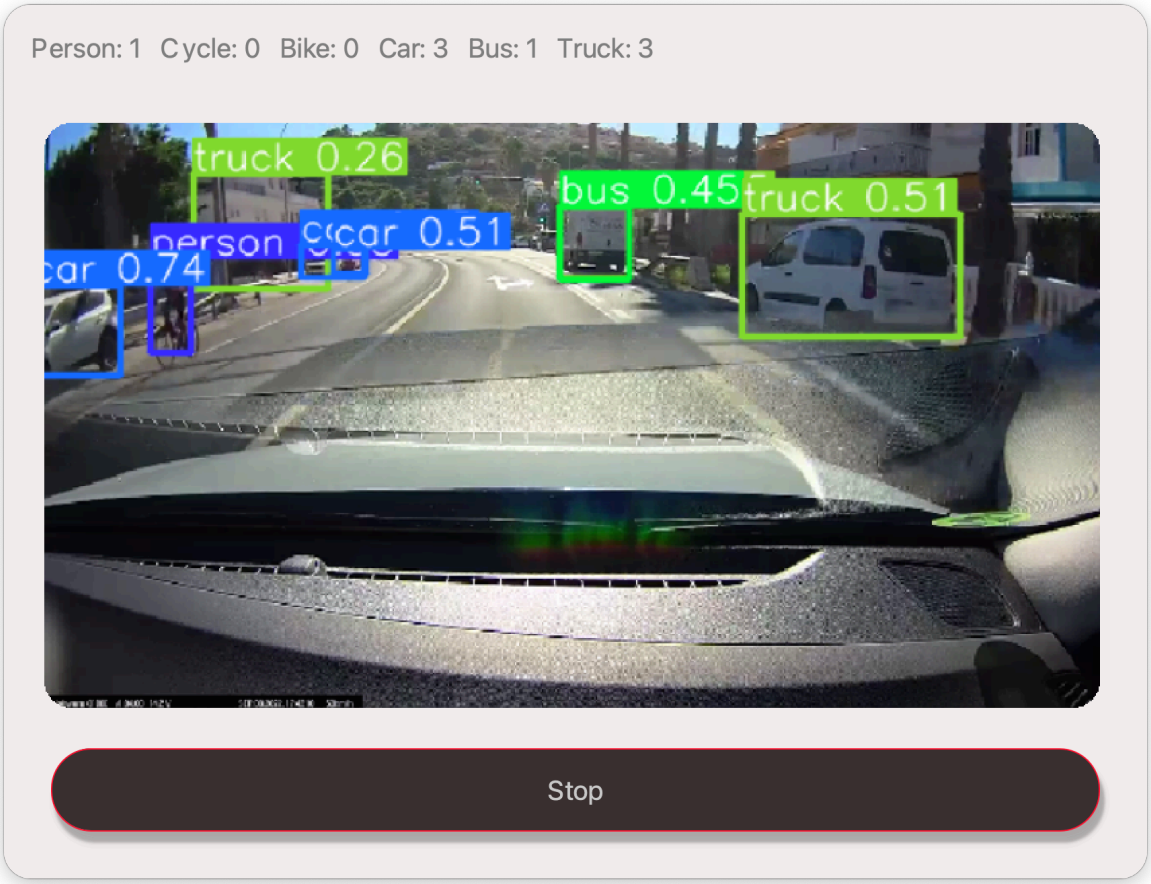



**Figure 5.10: Object Detection System (ODS)**

| Element | Description |
|---|---|
| Person | Shows the number of persons detected on the road |
| Cycle | Shows the number of cyclists detected on the road |
| Bike | Shows the number of Bikes detected on the road |
| Car | Shows the number of Cars detected on the road |
| Bus | Shows the number of Busses detected on the road |
| Truck | Shows the number of trucks detected on the road |
| Video Player | Displays Video footage showing detected objects in real-time |
| Stop Button | Stops the system and returns to the homepage |

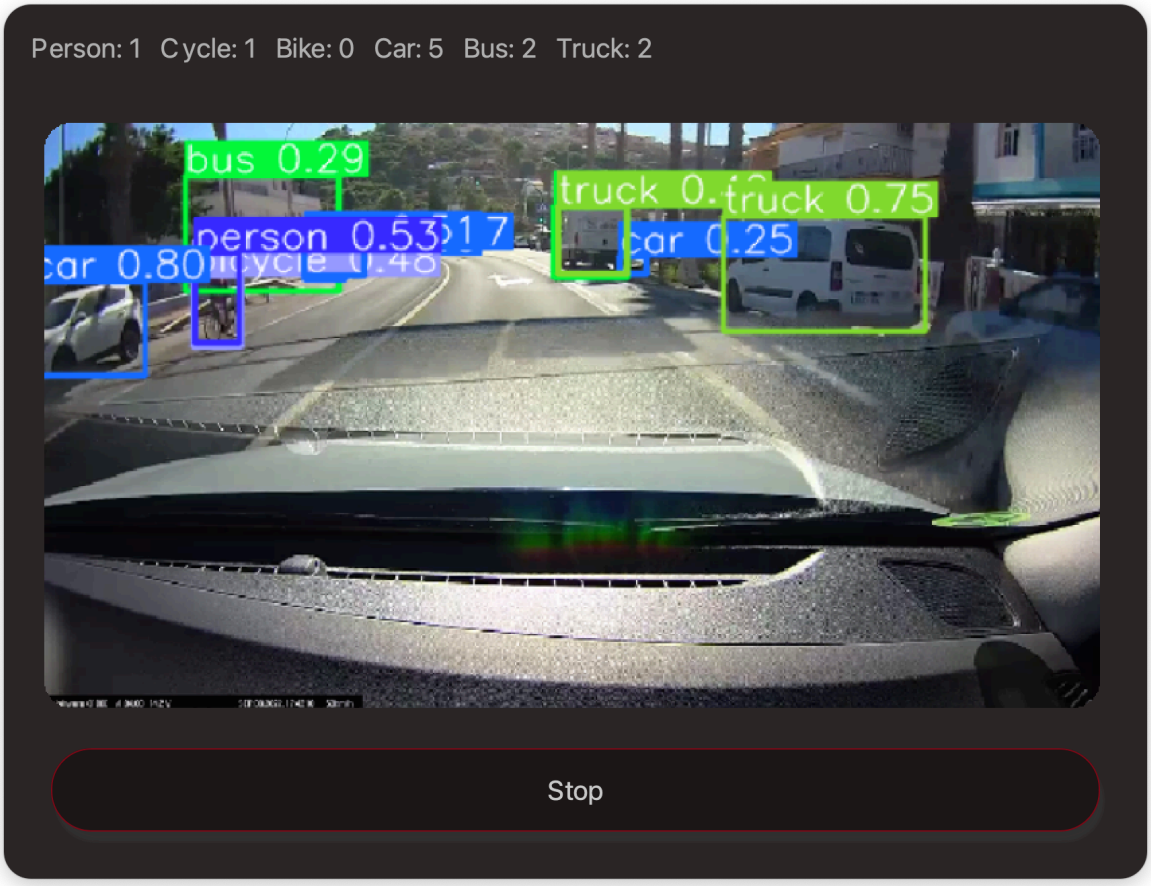



**Figure 5.11: Object Detection System (ODS) (Dark)**

**P**edestrian **D**etection **S**ystem (**PDS**) was originally a part of the **D**riving **N**egligence **D**issuader System (**DNDS**) but was later replaced by **O**bject **D**etection **S**ystem (**ODS**). Nonetheless, it is still kept for demo and comparison purposes. When a user decides to click the **Pedestrian Detection** button on the home page, they are switched to the **PDS** page illustrated in **Figures 5.12,** and **5.13**. Here the user is facilitated with information about the number of pedestrians detected on the road. A live video session shows the detected pedestrians enclosed in a numbered bounding box.

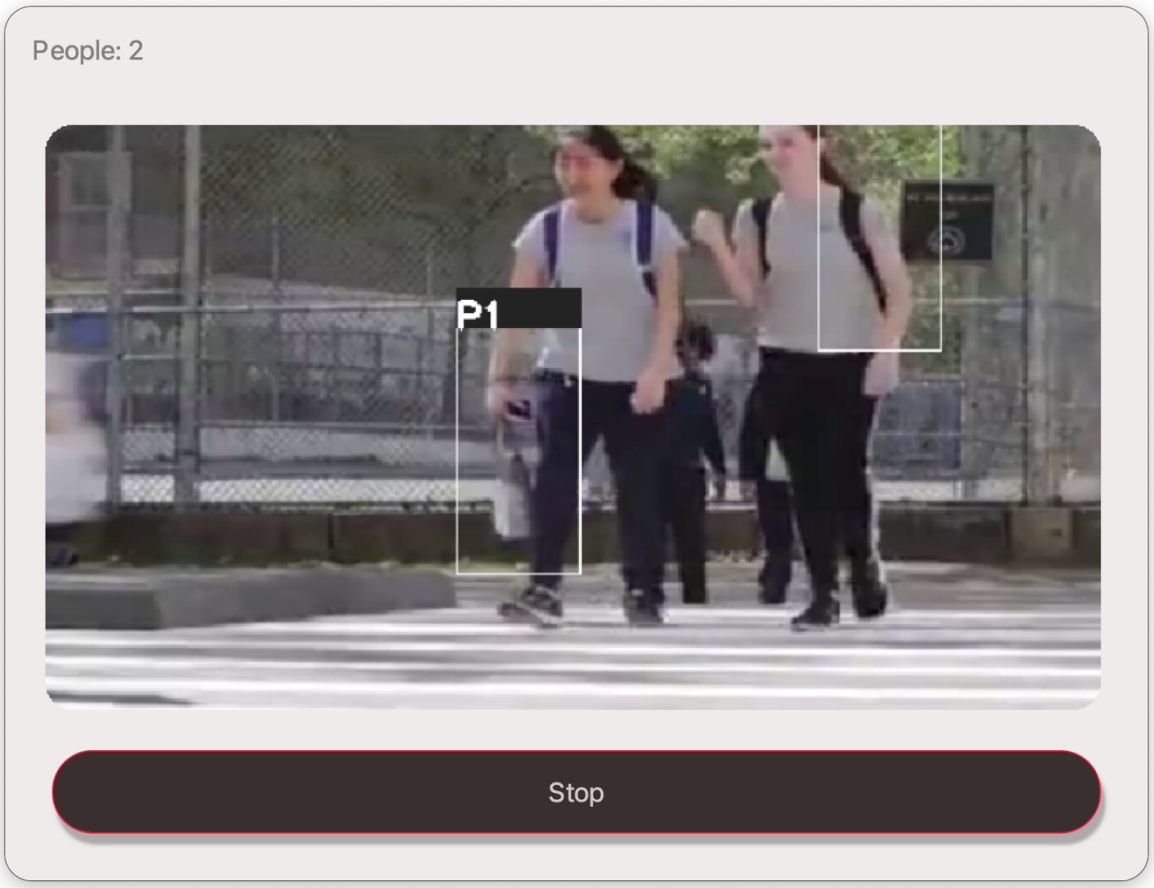



**Figure 5.12: Pedestrian Detection System (PDS) (Dark)**

| Element | Description |
|---|---|
| People | Shows the number of pedestrians detected |
| Video Player | Displays Video footage showing highlighted lane in real-time |
| Stop Button | Stops the system and returns to the homepage |


**Figure 5.13: Pedestrian Detection System (PDS) (Dark)**

When a user decides to click the **Driving Negligence Dissuader** button on the home page, they are switched to the **DND** page illustrated in **Figures 5.14** and **5.15**. Here the user is facilitated with all the information about each detection system in their designated corner. Live video sessions for each system show the detections in real-time. The user is warned of any detections by an audio message as well as a warning message below the video footage of the system where the warning is generated upon detection.

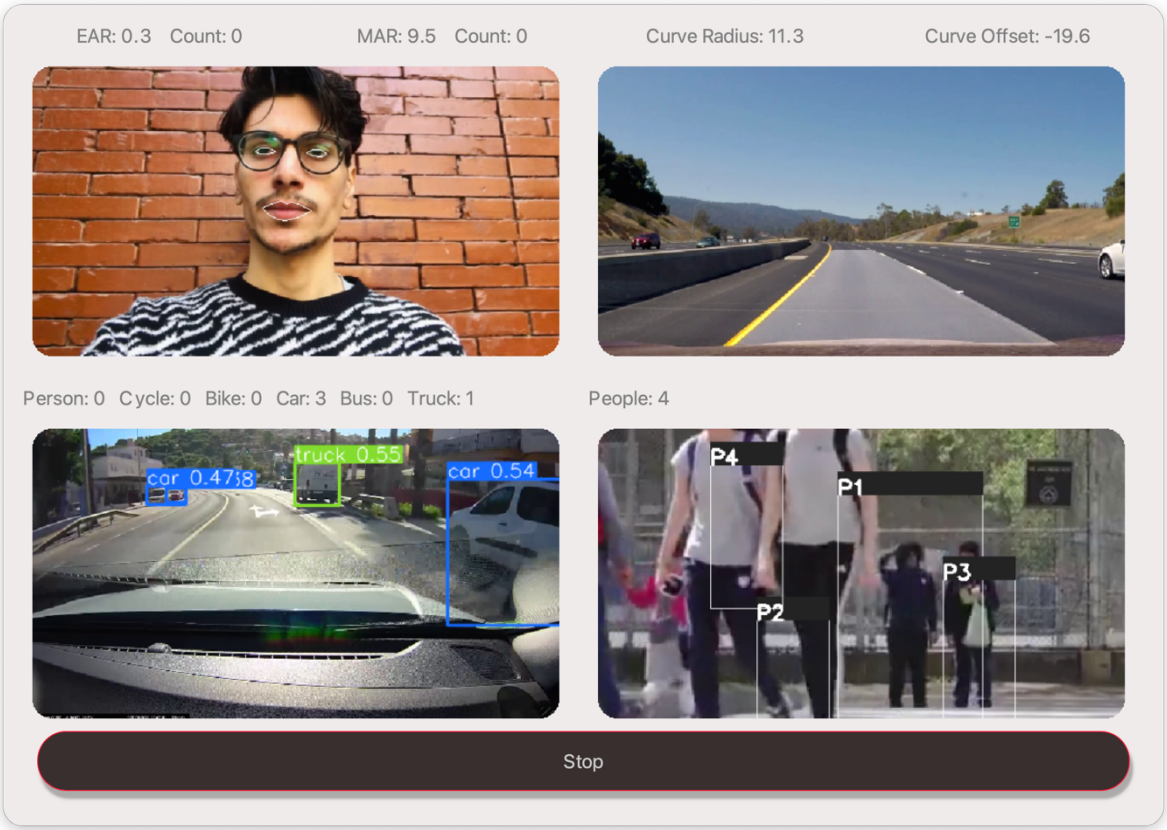



**Figure 5.14: Driving Negligence Dissuader System (DNDS)**

| Element | Description |
|---|---|
| EAR and Count | Shows Eye Aspect Ratio and Drowsy Detections count |
| MAR and Count | Shows Mouth Aspect Ratio and Yawning Detections count |
| Curve Radius | Shows the road curvature value |
| Curve Offset | Shows the vehicle position from the center of the lane |
| Person | Shows the number of persons detected on the road |
| Cycle | Shows the number of cyclists detected on the road |
| Bike | Shows the number of Bikes detected on the road |
| Car | Shows the number of Cars detected on the road |
| Bus | Shows the number of Busses detected on the road |
| Truck | Shows the number of trucks detected on the road |
| Video Players | Displays Video footage showing detections in real-time |
| Stop Button | Stops the system and returns to the homepage |

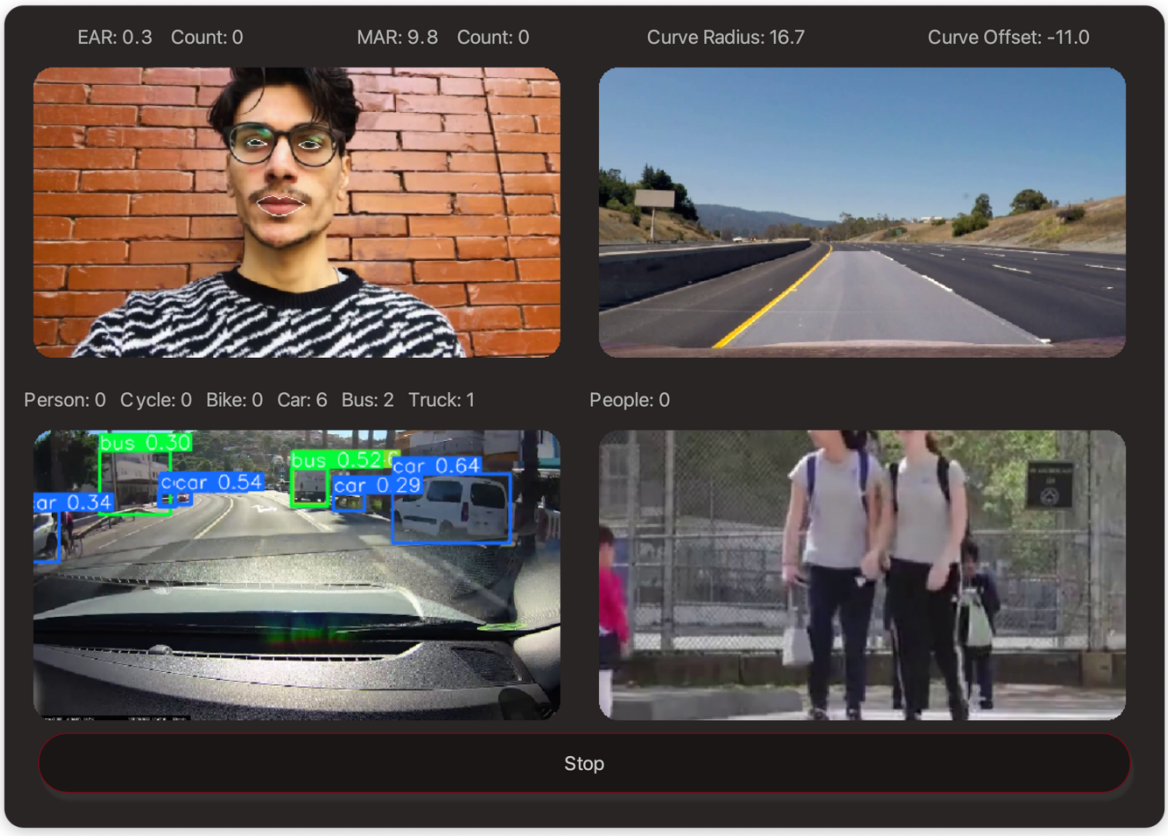



**Figure 5.15: Driving Negligence Dissuader System (DNDS) (Dark)**

# Chapter 6
# Testing and Evaluation

## 6 Testing and Evaluation

Having completed the development of the project **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**), the next stage is to test the finished product and see how well it performs. At this point, we revisit the requirements and test the operation of the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) according to the requirements both manually and by using automation tools.

### 6.1 Manual Testing

While tedious and time-consuming, manual testing is one of the essential steps of the **S**oftware **D**evelopment **L**ife **C**ycle (**SDLC**). Manual testing is very hands-on and requires proper analysis of the system to conduct. It is a highly involved testing method, from test case creation to actual test execution which allows for in-depth testing of the software system, in this case, **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**).

#### 6.1.1 System testing

With the project successfully developed, it is now time to test it to ensure that the system working as intended. We test the system to make sure it meets the requirements stated earlier. System testing will also help in finding the errors that may be hidden from the user. There are a few types of testing which include unit testing, functional testing, and integration testing.

#### 6.1.2 Unit Testing

In unit testing, we test the individual units of the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) to ensure each unit is performing its task as intended. In our case, this will be the subroutines, and functions of the system **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**).

**Unit Testing 1:** Launch **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**).

**Testing Objective:** To ensure the application launches with the appropriate theme and page.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Launch the application in light mode. | **OS** theme: Light<br>Target: Dark Stylesheet | The application launches on the homepage in light mode | Pass |
| 2. | Launch the application in dark mode. | **OS** theme: Dark<br>Target: Light Stylesheet | The application launches to the homepage in dark mode | Pass |

**Unit Testing 2:** Change the application theme from light mode to dark mode and vice versa.

**Testing Objective:** To ensure the theme change function is working as intended.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Change the application theme from light mode to dark mode by pressing the theme button. | App theme: Light<br>Target: Dark Stylesheet | The theme changes from light mode to dark mode on button press. | Pass |
| 2. | Change the application theme from dark mode to the light mode by pressing the theme button. | App theme: Dark<br>Target: Light Stylesheet | The theme changes from dark mode to light mode on button press. | Pass |

**Unit Testing 3:** Start **D**rowsiness and **Y**awning **D**etection **S**ystem (**DDS/YDS**).

**Testing Objective:** To ensure the **D**rowsiness and **Y**awning **D**etection **S**ystem (**DDS/YDS**) runs.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Start the **D**rowsiness and **Y**awning **D**etection **S**ystem (**DDS/YDS**) by pressing the **Drowsiness Detection** button. | Video Source: Face Camera | The application switches to **DDS/YDS** page and starts detection. | Pass |

**Unit Testing 4:** Start **L**ane **D**etection **S**ystem (**LDS**).

**Testing Objective:** To ensure the **L**ane **D**etection **S**ystem (**LDS**) runs.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Start the **L**ane **D**etection **S**ystem (**LDS**) by pressing the **Lane Detection** button. | Video Source: Dash Camera | The application switches to the **LDS** page and starts detection. | Pass |

**Unit Testing 5:** Start **O**bject **D**etection **S**ystem (**ODS**).

**Testing Objective:** To ensure the **O**bject **D**etection **S**ystem (**ODS**) runs.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Start the **O**bject **D**etection **S**ystem (**ODS**) by pressing the **Object Detection** button. | Video Source: Dash Camera | The application switches to the **ODS** page and starts detection. | Pass |

**Unit Testing 6:** Start **P**edestrian **D**etection **S**ystem (**PDS**).

**Testing Objective:** To ensure the **P**edestrian **D**etection **S**ystem (**PDS**) runs.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Start the **P**edestrian **D**etection **S**ystem (**PDS**) by pressing the **Pedestrian Detection** button. | Video Source: Dash Camera | The application switches to the **PDS** page and starts detection. | Pass |

**Unit Testing 7:** Start **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**).

**Testing Objective:** To ensure the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) runs.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Start the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) by pressing the **Driving Negligence Dissuader** button. | Video Source 1: Face Camera<br>Video Source 2: Dash Camera | The application switches to the **DNDS** page and starts detection. | Pass |

**Unit Testing 8:** Close or Reset the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**).

**Testing Objective:** To ensure the application close and reset function is working as intended.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Restart the application by pressing the **Close/Reset** button and choosing **Reset**. | Input: Click **Reset** | The application restarts on the homepage. | Pass |
| 2. | Close the application by pressing the **Close/Reset** button and choosing **Close**. | Input: Click **Close** | The application terminates. | Pass |

### 6.1.3 Functional Testing

In functional testing, we test the functionality of the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) against the functional requirements of the project stated earlier. We test each module (Comprised of multiple units working together to make it function) and make sure it performs as intended.

**Functional Testing 1:** Detect user Drowsiness and Yawning.

**Objective**: To ensure the **D**rowsiness and **Y**awning **D**etection **S**ystem (**DDS/YDS**) is performing its intended functions.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Start the **D**rowsiness and **Y**awning **D**etection **S**ystem (**DDS/YDS**) and detect drowsiness and yawning. | Video Source: Face Camera | The system successfully detects user drowsiness and yawning. And displays the detection information and live video on the **DDS/YDS** page. | Pass |

**Functional Testing 2:** Detect Road lanes.

**Objective**: To ensure the **L**ane **D**etection **S**ystem (**LDS**) is performing its intended function.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Start the **L**ane **D**etection **S**ystem (**LDS**) and detect Road lanes. | Video Source: Dash Camera | The system successfully detects lanes on the road. And displays the detection information and live video on the **LDS** page. | Pass |

**Functional Testing 3:** Detect objects on the road.

**Objective**: To ensure the **O**bject **D**etection **S**ystem (**ODS**) is performing its intended function.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Start the **O**bject **D**etection **S**ystem (**ODS**) and detect objects on the road. | Video Source: Dash Camera | The system successfully detects objects of multiple classes on the road. And displays the detection information and live video on the **ODS** page. | Pass |

**Functional Testing 4:** Detect pedestrians on the road.

**Objective**: To ensure the **P**edestrian **D**etection **S**ystem (**PDS**) is performing its intended function.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Start the **P**edestrian **D**etection **S**ystem (**PDS**) and detect pedestrians on the road. | Video Source: Dash Camera | The system successfully detects pedestrians on the road. And displays the detection information and live video on the **PDS** page. | Pass |

**Functional Testing 5:** Perform all detections at the same time.

**Objective**: To ensure the **DNDS** is performing its intended functions.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Start the Start **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) and detect user Drowsiness and yawning. Also detect lanes, objects, and pedestrians on road. | Video Source 1: Face Camera<br>Video Source 2: Dash Camera | The system successfully detects user drowsiness and yawning. It also detects road lanes, objects, and pedestrians on the road. And displays the detection information and live video on **DNDS** page. | Pass |

### 6.1.4 Integration Testing

In integration testing, we test the different units, modules, and components of the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) as a combined entity to ensure the different components of the system are integrated and working properly.

| No. | Test case/Test script | Attribute and value | Expected result | Result |
|---|---|---|---|---|
| 1. | Launch **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**). | **OS** theme: Light/Dark Target: Dark/Light Stylesheet | The application launch is successful and the theme is set according to the operating system. (If the **OS** theme is Light, the application launches in light mode. If the **OS** theme is Dark, the application launches in Dark, mode). All the Buttons, Icons, and labels are loaded successfully and are positioned properly. | Pass |
| 2. | Change application theme. | **OS** theme: Light/Dark Target: Dark/Light Stylesheet | The application theme changes successfully and the button icon also changes according to the set theme. | Pass |
| 3. | Start the **D**rowsiness and **Y**awning **D**etection **S**ystem (**DDS/YDS**), detect drowsiness and yawning, and stop the system to go back to the homepage. | Video Source: Face Camera | The system starts successfully by pressing the **Drowsiness Detection** button, detects user drowsiness and yawning, displays the detection information and live video on the **DDS/YDS** page, | Pass |

| | | | | |
|---|---|---|---|---|
| | | | and returns to the homepage when stopped by pressing the **Stop** button. | |
| 4. | Start the **L**ane **D**etection **S**ystem (**LDS**), detect lanes on the road, and stop the system to go back to the homepage. | Video Source: Dash Camera | The system starts successfully by pressing the **Lane Detection** button, detects lanes on the road, displays the detection information and live video on the **LDS** page, and returns to the homepage when stopped by pressing the **Stop** button. | Pass |
| 5. | Start the **O**bject **D**etection **S**ystem (**ODS**), detect objects of multiple classes on the road, and stop the system to go back to the homepage. | Video Source: Dash Camera | The system starts successfully by pressing the **Object Detection** button, detects objects of multiple classes on the road, displays the detection information and live video on the **ODS** page, and returns to the homepage when stopped by pressing the **Stop** button. | Pass |
| 6. | Start the **P**edestrian **D**etection **S**ystem (**PDS**), detect pedestrians on the road, and stop the | Video Source: Dash Camera | The system starts successfully by pressing the **Pedestrian Detection** button, detects pedestrians on the road, displays the detection information and | Pass |

| | | | | |
|---|---|---|---|---|
| | system to go back to the homepage. | | live video on the **PDS** page, and returns to the homepage when stopped by pressing the **Stop** button. | |
| 7. | Start the Start **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**), detect user Drowsiness and yawning. Detect lanes, objects, and pedestrians on the road, and stop the system to go back to the homepage. | Video Source 1: Face Camera<br>Video Source 2: Dash Camera | The system starts successfully by pressing the **Driving Negligence Dissuader** button, detects user drowsiness and yawning, detects road lanes, objects, and pedestrians on the road, displays the detection information and live video of each detection system on the **DNDS** page, and returns to the homepage when stopped by pressing the **Stop** button. | Pass |

## 6.2 Automated Testing:

Automated testing helps improve the quality of software development by reducing the amount of time required to test every piece of code. In most cases, automation can be used in combination with manual validation to provide a complete software development process. In automated testing, we test and validate the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) using automation tools.

**Tools used**

| **Tool Name** | **Tool Description** | **Applied on [list of related test cases / FR / NFR]** | **Results** |
|---|---|---|---|
| **PyTest** | Pytest is a Python testing framework that originated from the **PyPy** project. It can be used to write various types of software tests, including unit tests, integration tests, end-to-end tests, and functional tests. Its features include parametrized testing, fixtures, and assert re-writing | test_dnds.py::test_widgets_dictionary | [ 3%] Pass |
| | | test_dnds.py::test_light_stylesheet_exists | [ 7%] Pass |
| | | test_dnds.py::test_dark_stylesheet_exists | [ 11%] Pass |
| | | test_dnds.py::test_face_landmark_detectors | [ 15%] Pass |
| | | test_dnds.py::test_drowsy_alert | [ 19%] Pass |
| | | test_dnds.py::test_yawn_alert | [ 23%] Pass |
| | | test_dnds.py::test_faces | [ 26%] Pass |
| | | test_dnds.py::test_facial_landmarks | [ 30%] Pass |
| | | test_dnds.py::test_eye_aspect_ratio | [ 34%] Pass |
| | | test_dnds.py::test_mouth_aspect_ratio | [ 38%] Pass |
| | | test_dnds.py::test_draw_eyes_lips | [ 42%] Pass |
| | | test_dnds.py::test_lane_alert_left | [ 46%] Pass |
| | | test_dnds.py::test_lane_alert_right | [ 50%] Pass |
| | | test_dnds.py::test_lane_lines | [ 53%] Pass |
| | | test_dnds.py::test_perspective_transform | [ 57%] Pass |
| | | test_dnds.py::test_histogram | [ 61%] Pass |
| | | test_dnds.py::test_lanes_overlay | [ 65%] Pass |
| | | test_dnds.py::test_curvature | [ 69%] Pass |
| | | test_dnds.py::test_center_offset | [ 73%] Pass |
| | | test_dnds.py::test_yolo_model | [ 76%] Pass |
| | | test_dnds.py::test_detect_objects | [ 80%] Pass |
| | | test_dnds.py::test_extract_detection_data | [ 84%] Pass |

| | | | |
|---|---|---|---|
| | | test_dnds.py::test_squeeze_frame<br>test_dnds.py::test_hogcv<br>test_dnds.py::test_detect_persons<br>test_dnds.py::test_draw_bounding_boxes | [ 88%] Pass<br>[ 92%] Pass<br>[ 96%] Pass<br>[100%] Pass<br>26 tests passed in 23.09s |

# Chapter 7
# Conclusion and Future Work

## 7 Conclusion and Future Work

The developed project has achieved its desired objectives set at the beginning of the project. The developed systems perform their tasks reasonably well. In this chapter, we conclude our findings in the conclusion section. We also discuss the potential additions and other things can be added and improved on in the future for the **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**).

### 7.1 Conclusion

The **D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) is a real-time system. It processes the given footage and produces results in real-time without any stutters or delays. The purpose of developing the system was to produce a vehicle safety system that detects drivers' negligence and alerts them in order to reduce the chances of any mishap on the road. The system utilizes popular technologies like computer vision, machine leering, and deep learning to perform the intended tasks. The **D**rowsiness **D**etection **S**ystem (**DDS**) and **Y**awning **D**etection **S**ystem (**YDS**) based on **H**aar **C**ascade **C**lassifier (**HCC**) and **DLib** shape predictor were implemented successfully with exceptional results. The **L**ane **D**etection **S**ystems (**LDS**) based on Computer vision and sliding window technique was implemented with reasonable results. And lastly, the **O**bject **D**etection **S**ystem (**ODS**) based on the **YoloV5** deep leering model was also successfully implemented and executed with acceptable results. **D**rowsiness **D**etection **S**ystem (**DDS**) performed the best among all with an average accuracy of **95.2%**. The **Y**awning **D**etection **S**ystem (**YDS**) performed the second best with an average accuracy of **93.8%**. **L**ane **D**etection **S**ystems (**LDS**) performed with an acceptable **77%** accuracy. The **O**bject **D**etection **S**ystem (**ODS**) performed with an average of **78.81%** accuracy in all classes. The interactive graphical user interface lets users test and interacts with available features. User has the ability to test the individual systems and view the detections in real-time. The user has no issue interacting with the system. The I/O operations work as intended. The overall performance of the system is very reasonable but can still be improved upon further by using better tools.

## 7.2 Future Work

**D**riving **N**egligence **D**issuader **S**ystem (**DNDS**) just like any other system has its limitations. The **D**rowsiness **D**etection **S**ystem (**DDS**) and **Y**awning **D**etection **S**ystem (**YDS**) although performing exceptionally well can use some improvement. The system currently only relies on **MAR** and **EAR** values. An addition to the system should be the ability to detect driver's fatigue using facial features. The **L**ane **D**etection **S**ystem (**LDS**) is the weakest part of the developed system. It currently uses computer vision to detect lanes. Although adequate for marked roads and highways, the system is rendered useless when faced with unmarked urban roads and pathways. This can be fixed by leveraging machine learning and deep learning technologies for lane detection. The **O**bject **D**etection **S**ystem (**ODS**) can detect objects on the road but there is no way of telling how far the object is. An addition to the **O**bject **D**etection **S**ystem (**ODS**) should be to detect the distance between the object and the car. This can be used to alert the driver of upcoming objects and suggest them to slow down. The user interface uses a mouse and keyboard as input devices to interact with the system. A worthy inclusion to the user interface will be to add touch screen functionality. The overall performance of the system can be improved by further calibration and adjustment. Furthermore, the system can be improved by adding footage and logs saving functionality. An automatic braking system can be added by leveraging data from the **O**bject **D**etection **S**ystem (**ODS**). Some honorable mentions are remote access to the logs, footage and settings via smartphone, collision detection, GPS tracking, parental controls, fatigue detection. Special person controls and Bluetooth pairing.

# Chapter 8
# References

## 8 References


**1.** World Health Organization (WHO), Road traffic injuries, 21 June 2021

**2.** Bailey Javins & Carter, Percentage of Auto Accidents are Caused by Human Error, 08 Sep 2020.

**3.** Bailey Javins & Carter, Percentage of Auto Accidents are Caused by Human Error, 08 Sep 2020.

**4.** R. Jabbar, M. Shinoy, M. Kharbeche, K. Al-Khalifa, M. Krichen, and K. Barkaoui, "Driver drowsiness detection model using convolutional neural networks techniques for android application," in 2020 IEEE International Conference on Informatics, IoT, and Enab

**5.** S. Arefnezhad, S. Samiee, A. Eichberger, and A. Nahvi, "Driver drowsiness detection based on steering wheel data applying adaptive neuro-fuzzy feature selection," Sensors, vol. 19, no. 4, p. 943, 2019.

**6.** K. Mutya, J. Shah, A. D. McDonald, and J. Jefferson, "What are steering pictures worth? using image-based steering features to detect drowsiness on rural roads," Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 63, no. 1, pp. 2041-2045, 2019

**7.** Li, G.; Chung, W.Y. A context-aware EEG headset system for early detection of driver drowsiness. Sensors 2015, 15, 20873–20893. [CrossRef] [PubMed]

**8.** Manu, B.N. Facial features monitoring for real-time drowsiness detection. In Proceedings of the International Conference of Innovations in Information Technology (IIT), Al-Ain, UAE, 28–30 November 2016. [CrossRef]

**9.** Selvakumar, K.; Jerome, J.; Rajamani, K.; Shankar, N. Real-time vision-based driver drowsiness detection using partial least squares analysis. J. Signal Process. Syst. 2016, 85, 263–274. [CrossRef]

**10.** Naurois, C.J.D.; Bourdin, C.; Stratulat, A.; Diaz, E.; Vercher, J. Detection and prediction of driver drowsiness using artificial neural network models. Accid. Anal. Prev. 2017. [CrossRef]

**11.** Daza, I.G.; Bergasa, L.M.; Bronte, S.; Yebes, J.J.; Almazan, J.; Arroyo, R. Fusion of optimized indicators from advanced driver assistance systems (ADAS) for driver drowsiness detection. Sensors 2014, 14, 1106–1113. [CrossRef] [PubMed]

**12.** Meng, C.; Shi-wu, L.; Wen-cai, S.; Meng-zhu, G.; Meng-yuan, H. Drowsiness monitoring based on steering wheel status. Transp. Res. Part D Transp. Environ. 2018. [CrossRef]

**13.** K. Ghazali, R. Xiao and J. Ma, "Road Lane Detection Using H-Maxima and Improved Hough Transform", Fourth International Conference on Computational Intelligence, Modelling and Simulation, pp: 2166-8531, 2011.

**14.** K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp. 1904–1916, 2015. View at: Publisher Site | Google Scholar

**15.** S. Liu, Di Huang, and Y. Wang, "Learning spatial fusion for single-shot object detection," 2019, https://arxiv.org/abs/1911.09516. View at: Google Scholar

**16.** W. Liu, D. Anguelov, D. Erhan, et al, "SSD: single shot multibox detector," in Proceedings of the European Conference on Computer Vision, pp. 21–37, Springer, Amsterdam, Netherlands, October 2016. View at: Publisher Site | Google Scholar

**17.** R. Joseph, S. Divvala, R. Girshick, and F. Ali, "You only look once: unified, real-time object detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788, Las Vegas, NV, USA, June 2016. View at: Google Scholar

**18.** Smuda P, Schweiger R, Neumann H, Ritter W (2006) Multiple cue data fusion with particle filters for road course detection in vision systems. In: 2006 IEEE Intelligent Vehicles Symposium. IEEE. pp 400–405. https://doi. org/10.1109/ivs.2006.1689661

**19.** Teng W, Wang Y, Yu B, Liu J (2020) Icciu: A new real-time lane-level positioning algorithm. IEEE Access. https://doi.org/10.1109/access.2020.2970503

**20.** A. D. McDonald, J. D. Lee, C. Schwarz, and T. L. Brown, "A contextual and temporal algorithm for driver drowsiness detection," Accident Analysis & Prevention, vol. 113, pp. 25–37, Apr. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457518300058

**21.** U. S. F. M. C. S. A. T. Division, "PERCLOS: A Valid Psychophysiological Measure of Alertness As Assessed by Psychomotor Vigilance," October 1998.

**22.** C.S.Wei, Y.T.Wang, C.T.Lin, and T.P.Jung, "Toward Drowsiness Detection Using Non-hair-Bearing EEG-Based Brain-Computer Interfaces," IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2018.

**23.** V. J. Kartsch, S. Benatti, P. D. Schiavone, D. Rossi, and L. Benini, "A sensor fusion approach for drowsiness detection in wearable ultra-low-power systems," Information Fusion, vol. 43, pp. 66–76, Sep. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253517306942

**24.** S. Tateno, X. Guan, R. Cao, and Z. Qu, "Development of Drowsiness Detection System Based on Respiration Changes Using Heart Rate Monitoring," in 2018 57th Annual Conference of the Society of Instrument and Control Engineers of Japan, SICE 2018, 2018, pp. 1664–1669.

**25.** A. Kamilaris and F. X. Prenafeta-Bold, "Deep learning in agriculture: A survey," Computers and Electronics in Agriculture, vol. 147, pp. 70 – 90, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169917308803

**26.** M. Tayab Khan, H. Anwar, F. Ullah, A. Ur Rehman, R. Ullah, A. Iqbal, B.-H. Lee, and K. S. Kwak, "Smart Real-Time Video Surveillance Platform for Drowsiness Detection Based on Eyelid Closure," Wireless Communications and Mobile Computing, vol. 2019, pp. 1–9, 2019.

**27.** M. F. Shakeel, N. A. Bajwa, A. M. Anwaar, A. Sohail, A. Khan, and H. ur Rashid, "Detecting Driver Drowsiness in Real-time Through Deep Learning

Based Object Detection," 2019, pp. 283–296. [Online]. Available: http://link.springer.com/10.1007/978-3-030-20521-8{ }24

**28.** L.Celona, L.Mammana, S.Bianco, and R.Schettini, "A multi-task CNN framework for driver face monitoring," IEEE International Conference on Consumer Electronics Berlin, ICCE-Berlin, vol. 2018-Septe, pp. 1–4, 2018.

**29.** C.-H. Weng, Y.-H. Lai, and S.-H. Lai, "Driver drowsiness detection via a hierarchical temporal deep belief network," in Asian Conference on Computer Vision. Springer, 2016, pp. 117–133.

**30.** Y. Xie, K. Chen, and Y. L. Murphey, "Real-time and Robust Driver Yawning Detection with Deep Neural Networks," Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018, pp. 532–538, 2019.

**31.** S. Mehta, S. Dadhich, S. Gumber, and A. Jadhav Bhatt, "Real-Time Driver Drowsiness Detection System Using Eye Aspect Ratio and Eye Closure Ratio," SSRN Electronic Journal, pp. 1333–1339, 2019.

**32.** Wang Y, Teoh EK, Shen D (2004) Lane detection and tracking using b-snake. Image Vis Comput 22(4):269–280

**33.** Tan H, Zhou Y, Zhu Y, Yao D, Li K (2014) A novel curve lane detection based on improved river flow. In: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC). IEEE. pp 133–138. https://doi.org/10.1109/itsc.2014.6957679

**34.** Shin B-S, Tao J, Klette R (2015) A superparticle filter for lane detection. Pattern Recogn 48(11):3333–3345

**35.** Yoo H, Yang U, Sohn K (2013) Gradient-enhancing conversion for illumination-robust lane detection. IEEE Trans Intell Transp Syst 14(3):1083–1094

**36.** Son J, Yoo H, Kim S, Sohn K (2015) Real-time illumination invariant lane detection for lane departure warning system. Expert Syst Appl 42(4):1816–1824

**37.** [37] Jung S, Youn J, Sull S (2015) Efficient lane detection based on spatiotemporal images. IEEE Trans Intell Transp Syst 17(1):289–295

**38.** Niu J, Lu J, Xu M, Lv P, Zhao X (2016) Robust lane detection using two-stage feature extraction with curve fitting. Pattern Recogn 59:225–233

**39.** De Brabandere B, Van Gansbeke W, Neven D, Proesmans M, Van Gool L (2019) End-to-end lane detection through the differentiable least-squares fitting. arXiv preprint. arXiv:1902.00293

**40.** Liu L, Chen X, Lu Z, Wang L, Wen X (2019) Mobile-edge computing framework with data compression for the wireless network in energy internet. Tsinghua Sci Technol 24(3):271–280

**41.** Qi L, Zhang X, Dou W, Ni Q (2017) A distributed locality-sensitive hashing-based approach for cloud service recommendation from multi-source data. IEEE J Sel Areas Commun 35(11):2616–2624

**42.** Qi L, Zhang X, Dou W, Hu C, Yang C, Chen J (2018) A two-stage locality-sensitive hashing-based approach for privacy-preserving mobile service recommendation in cross-platform edge environment. Futur Gener Comput Syst 88:636–643

**43.** John V, Liu Z, Guo C, Mita S, Kidono K (2015) Real-time lane estimation using deep features and extra trees regression. In: Image Video Technol. Springer. pp 721–733. https://doi.org/10.1007/978-3-319-29451-3_57

**44.** Kim J, Park C (2017) End-to-end ego lane estimation based on sequential transfer learning for self-driving cars. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp 30–38. https://doi.org/10.1109/cvprw.2017.158

**45.** Pan X, Shi J, Luo P, Wang X, Tang X (2018) Spatial as deep: Spatial CNN for traffic scene understanding. In: Thirty-Second AAAI Conference on Artificial Intelligence

**46.** Zhang Z, Schwing AG, Fidler S, Urtasun R (2015) Monocular object instance segmentation and depth ordering with CNN. In: Proceedings of the IEEE International Conference on Computer Vision. pp 2614–2622. https://doi.org/10.1109/iccv.2015.300

**47.** Xu X, Fu S, Yuan Y, Luo Y, Qi L, Lin W, Dou W (2019) Multiobjective computation offloading for workflow management in cloudlet-based mobile cloud using nsga-ii. Comput Intell 35(3):476–495

**48.** Xu X, Liu X, Xu Z, Dai F, Zhang X, Qi L (2019) Trust-oriented IoT service placement for smart cities in edge computing. IEEE Internet Things J. https://doi.org/10.1109/jiot.2019.2959124

**49.** J.F.Chen, C.C.Wang, and C.F.Chou, "Multiple target tracking in occlusion area with interacting object models in urban environments," Robotics and Autonomous Systems, vol. 103, pp. 68–82, 2018.

**50.** S.Kato, E.Takeuchi, Y.Ishiguro, Y.Ninomiya, K.Takeda, and T. Hamada, "An open approach to autonomous vehicles," IEEE Micro, vol. 35, no. 6, pp. 60–68, 2015.

**51.** S. Shaily, S. Krishnan, S. Natarajan, and S. Periyasamy, "Smart driver monitoring system," Multimedia Tools and Applications, vol. 80, no. 17, pp. 25633–25648, 2021.

**52.** L. Liu, W. Ouyang, X. Wang, et al, "Deep learning for generic object detection: a survey," International Journal of Computer Vision, vol. 128, no. 2, pp. 261–318, 2020.

**53.** Z.Zou, Z.Shi, Y.Guo, andJ.Ye, "Object detection in 20 years: a survey," 2019, http://arxiv.org/abs/1905.05055.

**54.** R. Girshick, "Fast RCNN," in Proceedings of the IEEE international conference on computer vision, pp. 1440–1448, 2015.

**55.** J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: object detection via region-based fully convolutional networks," In Advances in neural information processing systems, vol. 29, pp. 379–387, 2016.

**56.** G. Ghiasi, T. Y. Lin, and Q. V. Le, "Nas-fpn: learning scalable feature pyramid architecture for object detection," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7036–7045, 2019.

**57.** J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788, 2016.

**58.** J. Redmon and A. Farhadi, “YOLO 9000: better, faster, stronger,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7263–7271, 2017.