# **A**

# **BE PROJECT REPORT ON**

AUTOMATIC ROAD ANALYSIS AND WARNING SYSTEM

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY IN THE

PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE

DEGREE

OF

**FINAL YEAR OF ENGINEERING**

**(COMPUTER ENGINEERING)**

**SUBMITTED BY**

1. **PRANALI SHIVALE 72173970K**
2. **NEHA SONAWANE 72173971H**
3. **SRUSHTI DHARMALE 72173972F**
4. **TAHA LOKHANDWALA 72034876F**

**UNDER THE GUIDANCE OF**

**Prof. Dr. M.B. Wagh**



**DEPARTMENT OF COMPUTER ENGINEERING**

**KJ EDUCATIONAL INSTITUTES**

# **TRINITY ACADEMY OF ENGINEERING**

SR.NO. 25 &27, KONDHWA-SASWAD ROAD, BOPDEV GHAT, PUNE 411 048

**SAVITRIBAI PHULE PUNE UNIVERSITY**

**2022 -2023**

**CERTIFICATE**

This is to certify that the BE Project report entitled

**“AUTOMATIC ROAD ANALYSIS AND WARNING SYSTEM”**

Submitted by

1. **PRANALI SHIVALE 72173970K**
2. **NEHA SONAWANE 72173971H**
3. **SRUSHTI DHARMALE 72173972F**
4. **TAHA LOKHANDWALA 72034876F**

is a genuine project that they completed under Dr. M.B. Wagh's supervision and that is submitted in order to satisfy the requirements of Savitribai Phule Pune University for the award of the degree of Final Year of Engineering (Computer Engineering).

**(Dr. M.B. Wagh)** **(Dr. M.B. Wagh)**

Guide HOD,

Computer Engineering

**(Prof. Dr. N.J. Uke)**

Principal,

**Trinity Academy of Engineering, Pune**

Place : Pune

Date :

# **ACKNOWLEDGEMENT**

We appreciate Prof. M.B. Wagh's assistance in helping us choose the topic and materials and for his insightful comments on the preparation of the BE project report and presentation.

Prof. M.B. Wagh, head of the computer department, is appreciated for providing a safe workplace and modern amenities. He allowed us to voice our worry and, by periodically extending his cooperation, sought to address it.

Our objectives drive our work. Goals are vital, but vision drives us to carry out our tasks in the best way possible so that they are all of the highest quality. We appreciate the support and vision of Principal Dr. N.J. Uke.

A constant boost at regular intervals is necessary for consistent achievement. We have had complete backing from management, which has encouraged us to be dependable and hit the goal. We appreciate the backing from management.

We appreciate the extended help and insightful advice from all of our colleagues. We want to express our gratitude to all of my friends for their unwavering support, assistance, and leadership.

# **NAME OF THE STUDENTS**

PRANALI SHIVALE 72173970K

NEHA SONAWANE 72173971H

SRUSHTI DHARMALE 72173972F

TAHA LOKHANDWALA 72034876F

**ABSTRACT**

Numerous methods for autonomous driving and Intelligent Transportation System (ITS) have been proposed in an effort to increase road safety and the effectiveness of the transportation system. The Lane Departure Warning System (LDWS) is one of them and is a significant problem. The primary purpose of LDWS is to alert the driver as soon as possible through various means, such as vibration or sound, when the vehicle being driven deviates too far from the centre of the lane. In light of this, precisely determine the road lane's centerline and the vehicle's trajectory's divergence from it. Numerous strategies based on computer vision have been suggested to enhance the performance of LDWS. By enhancing the approaches, we present a unique LDWS model in this study.

**TABLE OF CONTENT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No.** | | **Title of Chapter** | | | **Page No.** |
| **01** | | **Introduction** | | | 01 |
|  | 1.1 | Introduction | | |  |
|  | 1.2 | Overview | | |  |
|  | 1.3 | Motivation | | |  |
|  | 1.4 | Problem Definition and Objectives | | |  |
|  | 1.5  1.6 | Project Scope and Limitations  Methodologies for Problem solving | | |  |
| **02** | | **Literature Survey** | | | 05 |
| **03** | | **Software Requirements Specification** | | | 12 |
|  | 3.1 | Assumptions and Dependencies | | |  |
|  | 3.2 | Functional Requirements | | |  |
|  | 3.3 | External Interface Requirements (If Any) | | |  |
|  |  | 3.3.1 | | User Interfaces |  |
|  |  | 3.3.2 | | Hardware Interfaces |  |
|  |  | 3.3.3 | | Software Interfaces |  |
|  |  | 3.3.4 | | Communication Interfaces |  |
|  | 3.4 | Non-functional Requirements | | |  |
|  |  | 3.4.1 | | Performance Requirements |  |
|  |  | 3.4.2 | | Safety Requirements |  |
|  |  | 3.4.3 | | Security Requirements |  |
|  |  | 3.4.4 | | Software Quality Attributes |  |
|  | 3.5 | System Requirements | | |  |
|  |  | 3.5.1 | | Database Requirements |  |
|  |  | 3.5.2 | | Software Requirements (Platform Choice) |  |
|  |  | 3.5.3 | | Hardware Requirements |  |
|  | 3.6 | Analysis Models: SDLC Model to be applied | | |  |
| **04** | | **System Design** | | | 21 |
|  | 4.1 | System Architecture | | |  |
|  | 4.2 | Mathematical Model | | |  |
|  | 4.3 | Data Flow Diagrams | | |  |
|  | 4.4 | Entity Relationship Diagrams | | |  |
|  | 4.5 | UML Diagrams | | |  |
| **05** | | **Project Plan** | | | 32 |
|  | 5.1 | Project Estimate | | |  |
|  |  | 5.1.1 | Reconciled Estimates | |  |
|  |  | 5.1.2 | Project Resources | |  |
|  | 5.2 | Risk Management | | |  |
|  |  | 5.2.1 | Risk Identification | |  |
|  |  | 5.2.2 | Risk Analysis | |  |
|  |  | 5.2.3 | Overview of Risk Mitigation, Monitoring, Management | |  |
|  | 5.3 | Project Schedule | | |  |
|  |  | 5.3.1 | Project Task Set | |  |
|  |  | 5.3.2 | Task Network | |  |
|  |  | 5.3.3 | Timeline Chart | |  |
|  | 5.4 | Team Organization | | |  |
|  |  | 5.4.1 | Team structure | |  |
|  |  | 5.4.2 | Management reporting and communication | |  |
| **06** | | **Project Implementation** | | | 43 |
|  | 6.1 | Overview of Project Modules | | |  |
|  | 6.2 | Tools and Technologies Used | | |  |
|  | 6.3 | Algorithm Details | | |  |
|  |  | 6.3.1 | Convert to grayscale | |  |
|  |  | 6.3.2 | Convert original image to HLS colour space | |  |
|  |  | 6.3.3 | Gaussian Blur | |  |
|  |  | 6.3.4 | Canny Edge Detection | |  |
|  |  | 6.3.5 | Hough Transformation Lines Detection | |  |
| **07** | | **Testing** | | | 52 |
|  | 7.1 | Testing Information | | |  |
|  | 7.2 | Types of Testing | | |  |
| 7.3 | | Test cases and test results | | |  |
| **08** | | **Results** | | | 63 |
| 8.1 | | Outcomes | | |  |
| **09** | | **Conclusion** | | | 62 |
| 9.1 | | Conclusions | | |  |
| 9.2 | | Future work | | |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE** | **ILLUSTRATON** | **PAGE NO** |
| 3.3.2 | Raspberry Pi | 13 |
| 3.6 | Waterfall Model | 18 |
| 4.1.1 | System Architecture | 21 |
| 4.3.1 | (DFD 0)Data Flow Diagram | 25 |
| 4.3.2 | (DFD 1)Data Flow Diagram | 28 |
| 4.3.3 | (DFD 2)Data Flow Diagram | 29 |
| 4.5.1 | E-R Diagram | 30 |
| 5.3.2 | Class Diagram | 31 |
| 5.3.3 | Task Network | 40 |
| 6.3.1 | Timeline Chart | 41 |
| 6.3.2 | Grayscale Image | 45 |
| 6.3.4.1 | HLS Colour Images | 46 |
| 6.3.5 | Candy Edge Detection | 47 |
| 6.3.6 | Hough Transform | 48 |

**LIST OF TABLES**

**TABLE ILLUSTRATION PAGE NO.**

3.5.3 Hardware Requirements 17

5.1 Estimation of KLOC 33

5.2.1 Risk Table 36

5.2.2 Risk probability Definitions 36

5.2.3 Risk Impact Definitions 37

5.2.4 Overview of risk mitigation, monitoring, management 38

7.2 Test cases 61

**CHAPTER 1**

**INTRODUCTION**

**1.1 Introduction**

People all throughout the world are living more comfortably than they did in the past thanks to the recent rapid economic and technological development. These days, automobiles are an integral part of everyone's daily lives. The prevalence of cars, however, offers more than just benefits. Every year, as the number of cars grows, so does the frequency of auto accidents. The vehicles are made more advanced with greater infrastructure in intelligent transportation systems with upgraded technologies. But many automakers ignore the lane and object recognition component of how we travel on the highways, and there has been little progress in this area for many years. Accidents are greatly influenced by lane detection and object detection. In order to facilitate human eyesight.

**1.2 Overview**

A single-vehicle road departure collision can be avoided with the use of vision-based Lane Departing Detection Technology. In practice, it is extremely difficult to recognize lanes quickly and accurately due to a variety of complex noises, hence the main goal of this work is to develop a set of methods for image processing that can produce results quickly and accurately under circumstances that are not ideal. A lane-detecting method for the Lanes Departure Alert Systems is suggested. Based on the experiment's comparison, the Canny algorithm has been selected as the edge detection method, and the Hough transform, on the other hand, is chosen as the effective straight detection method. The area of focus is defined to reduce noise for accurate rising and improve processing to satisfy the real-time need.

**1.3 Motivation**

* To create a real-time lane detection system that is able to identify the locations of the lanes on the road.
* Monitor the car direction moving through and warn if driver is moving outside the lane.
* It will also identify road signs
* It will be featured with finding Zebra crossing as well.

**1.4 Problem Definition**

As there are more cars on the highway, there are also more car accidents occurring. True, even someone with experience can become preoccupied while driving, especially when traveling a distance. When a motorist is fatigued and preoccupied, it is very common for their car to veer off the road; in fact, when a driver is moving quickly, even a tiny deviation in direction can have disastrous results. Therefore, in order to achieve the desired level of road safety, we are working on advanced driver backing systems. One of the most difficult and challenging tasks for future road vehicles is to find the road's lanes, boundaries, and obstacles. In particular, finding moving objects is a key component of preventing crashes in driving backing systems.

We are utilizing the multi-stage Canny Edge Detection driver.

**1.5 Project Scope & Limitations**

* Road Lane Detection
* Zebra Crossing Detection
* Road Sign Detection

**Limitation**

* System is fully depended on good light condition and camera quality.

**1.6 Methodologies for Problem Solving**

**Hough transform**

The Hough transformation is an attribute extraction approach applied to digital image processing, computer vision, and image analysis. The method's goal is to use a voting process to identify instances of objects that belong to a particular class of shapes that are incorrect. It is possible to describe lines, circles, or other parametric angles using the Hough transform (HT). It was first presented in 1962 (Hough 1962) and first applied a decade later (Duda 1972) to locate lines in photos. Finding the location of lines in photos is the goal. It is crucial to perform edge discovery first in order to create an edge image that will also be used as input for the Hough Transform method. However, on a certain (, ) brace, their corresponding cosine angles will cross each other.

Because of this, the Hough Transform method finds lines by randomly selecting (, ) dyads with more corners than a predetermined threshold. Why does computer vision prefer the Hough transfigure? The Hough transfigure method's key benefit is that it is forgiving of gaps in point boundary definitions and is largely immune to image noise.

**YOLOv4**

It is a method for real-time object detection and recognition in photos, and its initials stand for You Only Look. In a work initially presented at the IEEE/ CVF Conference on Computer Vision and Pattern Recognition (CVPR) in 2015, Redmond et al. proposed the approach. A one-stage object detection approach called YOLOv4 builds on YOLOv3 by adding a number of modules and trick bags that have been discussed in the literature. The tips and modules utilised are described in the factors section below. Source YOLOv4 Optimal Object Detection Speed and Accuracy.

The YOLO algorithm divides the image into N grids, each of which has an equal-sized SxS region. These N grids are each in charge of finding and locating the thing they contain.

**Tensorflow**

A library of open-source software that is free for artificial intelligence and machine learning is called TensorFlow. Although it can be used for numerous different tasks, deep neural network training and inference are given special attention. A large number of languages used for programming, especially Python, JavaScript, C, and Java, can utilise TensorFlow. TensorFlow was first created for complicated mathematical calculations without taking advanced literacy into consideration. Large-scale, multi-layered neural networks are created using TensorFlow. Deep literacy or machine literacy issues like Bracket, Perception, Understanding, Discovering, Prediction, and Creation are largely addressed by TensorFlow.

With its adaptable armature, calculations may be quickly deployed across a range of platforms (CPUs, GPUs, and TPUs), from desktops to clusters.

**CHAPTER 2**

**LITERATURE SURVEY**

**Data Methods for Collecting and Analysing Data to Assess Vehicle Road Departure Detection Systems.**

Vehicle Road Departure Discovery Systems Evaluation: Data Collection and Processing Methods (2008) Recently, several product vehicles have been equipped with road departure discovery systems (RDDSs) to prevent road departure crashes. This paper details the creation of the data acquisition and data post-processing tools for testing RDDSs in order to support and provide a standardised and objective performance evaluation of RDDSs. Road departure test scripts are explained using seven parameters. The overall design and key components of the data gathering and post-processing systems for evaluating vehicle RDDSs have been developed and presented. According to the experimental findings, a seeing system and data post-processing system could immediately display the vehicle stirring characteristic from the testing vehicle and gather all required signals.

**Lane Departure Alert Systems Lane Detection Methodology**

The University of Ottawa in Canada's Yue Chen and Azzedile Boukerche undertook a study to investigate the efficacy of the vision-based Lane Departure Warning System (LDWS) in preventing Single Vehicle Road Departure Accidents. The main task is to create an image processing system that can produce results quickly and directly in less-than-ideal circumstances because in practise, complicated noise made it extremely difficult to discern lanes quickly. A lane-finding the mechanism for the Lane Departure Warning mechanism is suggested in this research. The Hough transfigure is identified as the most effective approach to discover the beeline, and the Canny algorithm is named the edge detection system based on the trial comparison.

**Vision-based Road Detection in Automotive Systems**

S.Berte and A. Broggi, In 1995 An essential component of raising driving safety is lane detection. is a successful means of assisting Single Vehicle Road Departure Accidents. It was extremely difficult to accurately describe lanes in practice due to a variety of complex noise, thus the main task was to create an image processing system that could still produce accurate findings under less-than-ideal circumstances. A lane-finding mechanism for the Lane Departure Warning mechanism is suggested in this research. The Hough transfigure is identified as the most efficient method to discover the beeline, and the Canny algorithm is named the edge detection system based on the trial comparison. The region of interest (ROI) is defined to reduce noise in order to meet the real-time requirement.

for rising directly, and to enhance the processing speed. At last, trial results indicate that this lane discovery system can prize lane information from road images

**Real-time Lane Marker Detection in Urban Streets**

M. Aly[6],2008,This research proposes a reliable vanishing point estimation-based lane detecting technique. Because similar lines converge on the evaporating point in a projected 2-D image, estimating an evaporating point can be useful in identifying lanes. Even so, it might be challenging to accurately estimate the evaporation point in images with intricate backgrounds. Therefore, a reliable evaporating point estimate system is suggested that makes use of crossroad points of line portions removed from an input image as the basis for a probabilistic voting process. Line member strength, which denotes the applicability of the uprooted line portions, is used to define the suggested voting mechanism. The seeker line components for lanes are then identified while taking into account geometric restrictions. The proposed score function, which is intended to eliminate outliers in the seeker line segments, is finally used to determine the host lane. Additionally, the estimated evaporation point and position thickness of the discovered host lane are taken into account while meliorating the detected host lane using inter-frame similarity. Similarly, a system utilising a lookup table is suggested to save computational costs in the evaporation point estimation process. The suggested approach effectively calculates the evaporation point and finds lanes in visually appealing environments, according to experimental results.

**Detection Monitoring Systems Methods for Data Collection and Processing to Assess Vehicle Road Departure**

Monitoring Systems Methods for Data Collection and Processing to Assess Vehicle Road Departure Recently, several product vehicles have been equipped with road departure discovery systems (RDDSs) to prevent road departure crashes. The creation of the data access and data post-processing systems for testing RDDSs is discussed in this study in order to assist and provide a standardised and objective performance evaluation of RDDSs.

Road departure test scripts are described using seven parameters. The overall design and key components of the data gathering and post-processing systems for evaluating vehicle RDDSs are prepared and presented. According to experimental findings, a seeing system and data post-processing system could directly display the vehicle stir profile from the testing vehicle and gather all required signals. The proposed data collection system's efficiency is demonstrated through test track testing with various scripts. A Lane Departure Warning System Lane Detection Method Single Vehicle Road Departure Accident can be prevented with the use of the vision-based Lane Departure Warning System (LDWS).

Road departure test scripts are described using seven parameters. The overall design and key components of the data gathering and post-processing systems for evaluating vehicle RDDSs are prepared and presented. According to experimental findings, a seeing system and data post-processing system could directly display the vehicle stir profile from the testing vehicle and gather all required signals. The proposed data collection system's efficiency is demonstrated through test track testing with various scripts. A Lane Departure Warning System Lane Detection Method Single Vehicle Road Departure Accident can be prevented with the use of the vision-based Lane Departure Warning System (LDWS). efficiently and directly acquired. For lane departure warning systems, real-time lighting steady lane finding is available. A key component of improving driving safety is lane discovery. We suggest a real-time, illumination-steady lane finding system for lane departure warning systems in this paper.

The suggested system performs admirably in a variety of lighting situations, including dimly lit rooms, heavy downpours, and at night. It has three main components. To reduce computing complexity, we first describe an evaporating point based on a voting chart and construct an adaptable region of interest (ROI). In order to ensure lighting steady lane marker seeker finding, we secondly exploit the distinctive quality of lane colours. Eventually, we use a clustering algorithm developed by the lane marker campaigners to locate the primary lane. Our system alerts the driver when there is a lane departure situation. Under conditions of colourful light, experimental results demonstrate good performance with an average discovery rate of 93. Additionally, the entire process only needs 33ms each frame.

An Effective Method for Robust Lane Detection Based on Evaporating Point Estimation and Line Parts This paper proposes a reliable lane discovery system based on evaporating point estimate. Because similar lines converge on the evaporating point in a projected 2-D image, estimating an evaporating point can be useful in identifying lanes. Even so, it might be challenging to accurately estimate the evaporation point in images with intricate backgrounds. Therefore, a reliable evaporating point estimate system is suggested that makes use of crossroad points of line portions removed from an input image as the basis for a probabilistic voting process.

Line member strength, which denotes the applicability of the uprooted line portions, is used to define the suggested voting mechanism. The seeker line components for lanes are then identified while taking into account geometric restrictions. The proposed score function, which is intended to eliminate outliers in the seeker line segments, is finally used to determine the host lane. Additionally, the estimated evaporation point and position thickness of the discovered host lane are taken into account while meliorating the detected host lane using inter-frame similarity. Similarly, a system utilising a lookup table is suggested to save computational costs in the evaporation point estimation process. The suggested approach effectively calculates the evaporation point and finds lanes in visually appealing environments, according to experimental results.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr No | Title | Authors | Abstract | Limitations |
| 1 | A Framework for Detecting and Recognising Real-Time Lane and Road Surface Markings Using Cameras  (IEEE Transactions on Intelligent Vehicles – 2015) | Any Gupta , Ayesha Choudhary | Vision based, integrated framework based on spatio-temporal incremental clustering coupled with curve fitting and Grassmann manifold learning technique for lane detection. | It is only limited by the type of road surface markings present in the training data. |
| 2 | IEEE Transactions on Intelligent Vehicles, 2018; A Novel Approach for Road Lane Detection and Tracking Based on a Vehicle's Forward Monocular Camera | Andrade, David C.; Bueno, Felipe; Franco, Felipe R. Silva, Rodrigo Adamshuk | A novel strategy for lane detection and tracking, which fits as a functional requirement to deploy DAS features like Lane Departure Warning and Lane Keeping Assist. | The visibility of the lane markings was compromised by several factors, such as the reflectivity of the lane, the camera’s glare, the presence of shadows and the deterioration of the paint. |
| 3 | A Portable Vision-Based Real-Time Lane Departure Warning System: Day and Night  (IEEE Transactions on Intelligent Vehicles – 2018) | Pei-Yung Hsiao; Chun-Wei Yeh; Shih-Shinh Huang; Li-Chen Fu | Embedded advanced RISC machines (ARM)-based real-time Lane Departure Warning System | This system will poorly perform if there is no sufficient illumination, |
| 4 | Automatic Detection and Classification of Road Lane Markings Using Onboard Vehicular Cameras  (IEEE Transactions on Intelligent Vehicles – 2018) | de Paula, Mauricio Braga; Jung, Claudio Rosito | New approach for road lane classification using an onboard camera using Bayesian classifier. | Limits the rapid detection of transitions and removal of isolated misdetections. |
| 5 | Advanced Driver Assistance Through Lane Departure Identification (IEEE Transactions on Intelligent Transport Systems, 2015) | Gaikwad, Vijay; Lokhande, Shashikant | This approach reduces the computational time required for the lane departure estimation and reduces the false warnings | Does not estimate the real-world coordinates of a vehicle with respect to both lane boundaries. |
| 6 | Robust Road Lane Detection from shape and color feature fusion for Vehicle Self-Localization  (2017 4th International Conference on Transportation Information and Safety (ICTIS)) | Cai, Hao; Hu, Zhaozheng; Huang, Gang; Zhu, Dunyao | To extract the region of interest. perform Hough transformation and calculate the vehicle position | The vehicle's controller limitations arise for a maximum speed of 70 km/h in sharp turns. |
| 7 | Lane Detection Algorithm using Vanishing point  (IEEE-International Conference on Machine Learning and Cybernetics, 2013) | Bounini, Farid; Gingras, Denis; Lapointe, Vincent; Pollart, Herve | Using Canny edge detection, kalman algorithm to extract the region of interest | The vehicle's controller limitations arise for a maximum speed of 70 km/h in sharp turns. |
| 8 | Lane Detection Algorithm using Vanishing point  (IEEE-International Conference on Machine Learning and Cybernetics, 2013) | Wang Jingyu, ; Duan Jianmin | Lane detection method using the vanishing point according to the perspective feature of the camera | Image processing takes a long time. It is a major factor affecting the system’s real-time feature. |

Table 2.1 Literature Survey

**CHAPTER 3**

**SOFTWARE REQUIREMENTS SPECIFICATION**

**3.1 Assumptions and Dependencies**

The user is expected to have system with good camera which capture the images properly.

Car will have enough power to run system.

**3.2 Functional Requirements**

Hardware Module

* System should allow user to do authentication.
* System should allow user to start/stop the HW.
* System should capture the proper images with good pixel quality.
* System should detect the road lane in which vehicle should move on.

Notification Module

* System should notify to user through alarm when vehicle cross the lane.
* System should notify when cross the lane.

System Requirements

* System should be robust to handle image processing errors by itself.
* System should run continuously until user stop system.
* System should use less amount of memory and CPU while processing images.
* System should print debug messages which will be useful to debug any issue.
* System should report important events to user on console.

**3.3 External Requirements of Interface**

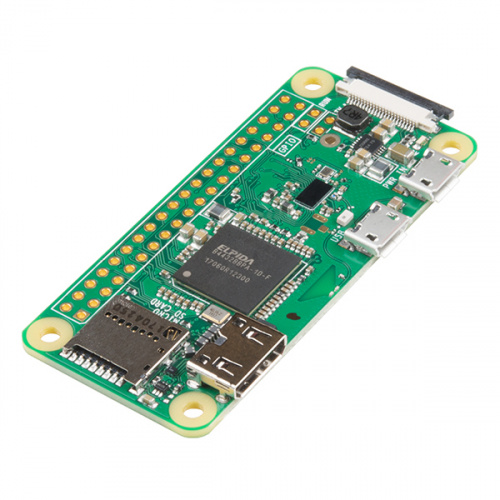
**3.3.1 User Interface**

* Console based user interface will be given for user to start stop the system.

**3.3.2 Hardware Interface**

* We use camera of 12 mega pixel which is connected to our Raspberry Pi. This will detect the road lane.
* System should allow user to select camera if more than 1 camera is connected with Hardware.

**Raspberry PI zero module**

Raspberry Pi Zero

Zero Raspberry PI module

Zero Raspberry Pi

Zero Raspberry PI module Zero Raspberry Pi Due to its ability to fit a whole computer on a single board, the Raspberry Pi is a well-known Single Board Computer (SBC). Many people may be familiar with the Raspberry Pi 3 and its predecessors because of their form factor, which has become largely recognizable. In fact, the Raspberry Pi has a smaller form size. The Raspberry Pi Zero's design allows for the bedding of a whole computer in actually more basic platforms. This companion will cover the Raspberry Pi Zero- Wireless, which has an integrated Wi-Fi module, which is the last iteration of the Zero product line. Although most interpretations and forms of the Pi should work with these instructions, the focus will be on the Pi Zero W.

Fig 3.3.2 Raspberry Pi

The specs of this board can be set up as follows:

1. 1 GHz, Single-core CPU
2. 512 MB RAM
3. Mini-HDMI harbourage
4. Micro-USB OTG harbourage
5. Micro-USB power
6. chapeau-compatible 40- leg title
7. Composite videotape and reset heads
8. CSI camera connector(v1.3 only)

At this point we shouldn't forget to mention that piecemeal from the boards mentioned before there are several other modules and factors similar as the Sense chapeau or Raspberry Pi Touch Display available which will work great for advance systems.

**3.3.3 Software Interface**

**OpenCV-Python**

A collection of Python scripts called OpenCV-Python was created to solve computer vision issues. Initiated by Guido van Rossum, Python is a general-purpose programming language that gained popularity quickly, largely due to its readability and simplicity. The programmer can now express concepts in fewer lines of code without sacrificing readability. Python is slower than languages like C/C. Nevertheless, Python can be easily expanded with C/ C, enabling us to create computationally ferocious laws in C/ C and generate Python wrappers that may be used as Python modules. This benefits us in two ways. First, because the actual C law is operating in the background, the law is as effective as the original C/C law.

**3.3.4 Communication Interface**

Then we will be using alarm system and going to produce our own communication protocol.

We'll use above announcement system to notify stoner.

**3.4 Non-functional Requirements**

**3.4.1 Performance Requirements**

he maximum amount of RAM should be used for our operation, and the garçon should be configured to just run the garçon process, for optimal performance.

**3.4.2 Safety Requirements**

System should be part of good package which can be installed inside vehicle.

**3.4.3 Security Requirements**

NA

**3.4.4 Software Quality Attributes**

Attributes The broad criteria that impact system design, run-time behaviour, and the marijuana user experience are quality attributes. They stand for areas of worry that could potentially have an impact on the entire business across all layers and categories. While some of these characteristics are specific to run time, design time, or stoner-centric difficulties, others are relevant to the overall system architecture. The success of the design and the overall quality of the software operation are determined by the amount to which the operation possesses the requested combination of high-quality characteristics, such as usability, performance, trustworthiness, and security.

**Reusability:**

Reusability is the capacity of components and subsystems to be suitable for usage in different contexts and applications. Reusability cuts down on component duplication and implementation time.

**Vacuity:**

The percentage of time that the system is operational and operating is known as vacuity. It can be calculated as the probability of a total system time-out over a set time frame. System crimes, structural issues, harsh attacks, and system freight will all have an impact on vacuity.

**Performance:**

Performance is an indication of how quickly a system will carry out any action in question. It can be quantified in terms of output or quiescence. Quiescence is the period of inaction before any action is initiated. The number of occurrences that occur during a particular quantum of time is known as the outturn.

**Reliability:**

The capacity of a system to continue operating over time is known as trust ability. The likelihood that a system won't fail to carry out its intended functions during a given time period is how trust ability is calculated.

**Scalability:**

Scalability is a system's capacity to either withstand increases in load without having an adverse effect on the system's performance or to be easily expanded.

**Testability:**

A system's testability refers to how simple it is to create test criteria for the system and its components and to carry out these tests to see if the criteria are met. It is more likely that a system's flaws can be insulated in a timely and efficient manner when it has good testability.

**Usability:**

Usability measures how well an operation satisfies the needs of the stoner and the consumer by being simple to use, easy to localize and globalize, providing decent access for drug users who are impaired, and operating well overall.

**3.5 System Requirements**

**3.5.1 Database Requirements**

We are using datasets of different sign boards which can help us to detect sing board on road in real time using image matching algorithms.

**3.5.2 Software Requirements**

* Programming Language: Python
* OpenCV library

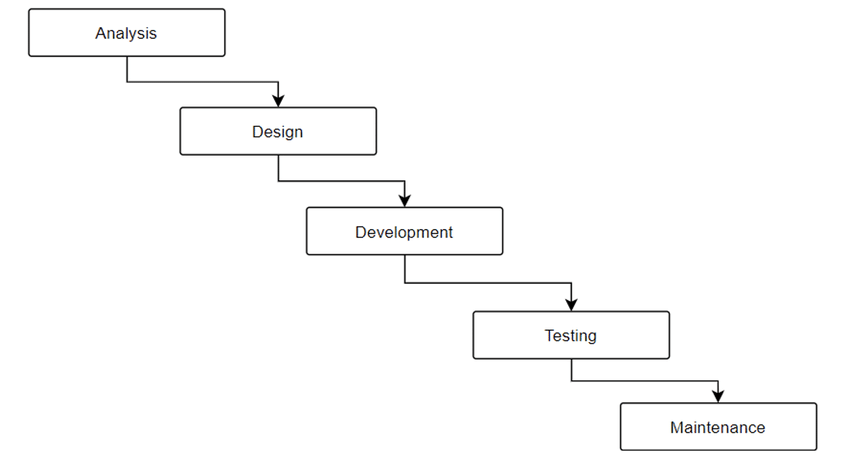
**3.5.3 Hardware Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr No. | Parameter | Minimum Requirements | Justification |
| 1 | CPU Speed | 2 GHz | Needed for High computation power |
| 2 | RAM | 3 GB | Needed for high Memory usage |
| 3 | Raspberry Pi | Pi Zero |  |
| 4 | Pi Camera | 12 mega pixel |  |
| 6 | Buzzer |  | Needed for Alerts |
| 7 | Power supply Unit |  | Needed for power supply |
| 8 | Led |  | Indication |

Table 3.5.3: Hardware Requirements

**3.6 Analysis Models: SDLC Model to be applied**

**Waterfall Model –**



The Software Development Lifecycle (SDLC) is commonly used in software engineering and product development. The Waterfall model is a direct, successional approach to the SDLC. The SDLC process for a design is implemented using the waterfall model, which follows a logical progression similar to the way water pours over a cliff's edge. It establishes clear objectives or standards at every stage of growth. After they are completed, those endpoints or pretensions cannot be changed.

In a paper written about his experience writing software for satellites and published in 1970, Winston W. Royce of the Lockheed Software Technology Centre suggested the idea. However, Royce did not refer to the waterfall; rather, he spoke of the attestation's downstream significance. Still being employed in artificial design activities is the waterfall paradigm. As the original software development approach, it is frequently mentioned. The approach is also employed more broadly as a high-position design and operation paradigm for intricate systems with many moving parts.

Design directors and brigades use the cascade approach to realise aspirations based on their company's needs. The approach is applied in a variety of design contexts, including architecture, manufacturing, information technology, and software development.

Each stage in a waterfall system is dependent on the outcome of the previous phase. How these systems develop follows a clear pattern.

For instance, these three general approaches are typically used in construction:

1. Physical design of a structure is created before any construction work starts.

2. A structure's shell is built before the foundation is laid.

3. The building's framework is finished before the walls are built.

When building a product on a production line, steps are carried out sequentially in a predetermined order until the final deliverable is produced. Unlike other development approaches, the waterfall model does not incorporate a design's end user or client as much. Drug users are consulted early on in the process of gathering and defining conditions, and after that, customer input is taken into account. The development platoon advances quickly through the stages of a project by excluding the client from the primary portion of the waterfall process. For brigades and systems that seek to develop a design in accordance with fixed or unchanging conditions specified on the morning of the design, this process works well. Waterfall systems feature little to no inherent variability and a high degree of process definition. If the design is limited by money or time, waterfall is another excellent option.

Systems built on the cascade model have specific attestation, are clearly specified, and are predictable. They also exhibit the following traits:

• specific demands

· Generous coffers

• a predetermined schedule

• technology that is well-understood and unlikely to undergo major change.

Waterfall is a good software development process because it sets out to accomplish that goal, which is important if an application must function correctly on the first try or risk losing clients. As an alternative, consider the project management and development methodology of Agile. Continuous iteration, a strategy used by agile methodologies, entails designing, creating, and testing software in successive cycles that build on one another.

### Phases of the waterfall model

1. **Conditions.** A formal condition documents also known as a functional specification, is created by dissecting implicit conditions deadlines, and design requirements. The design is defined and planned at this stage of development ,but no specific process are mentioned.
2. **Analysis.** The system requirements are dissected to provide product models and business judgement to direct product.This is also time when the viability of financial and specialized coffers is evaluated.
3. **Design.** . To describe precise design requirements such as the programming language, approach, data sources, armature, and services, a design specification paper is prepared.
4. **Programming and application .** The source . The models, sense, and demand requirements

designated in the earlier steps are used to build the source legislation Before being assembled ,the system is typically enciphered in smaller elements or units.

1. **Testing.** At this point, problems that need to be fixed. are found through quality assurance,unit system and beta tests A forced repetition of the coding phase for debugging may result from this. If the system passes integration ,and testing the waterfall model resumes.
2. **Use and application.** The operation or product is placed in the environment and it is intended to be fully operational.
3. **Conservation**. To improve, modernise and enhance the product and its usefulness, corrective Corrective. adaptive, and perfective conservation is continuously practised. Release of new performances and patch updates may fall under this category.

**CHAPTER 4**

**SYSTEM DESIGN USING HOUGH TRANSFORMATION AND CANNY EDGE DETECTION**

**4.1 System Architecture**

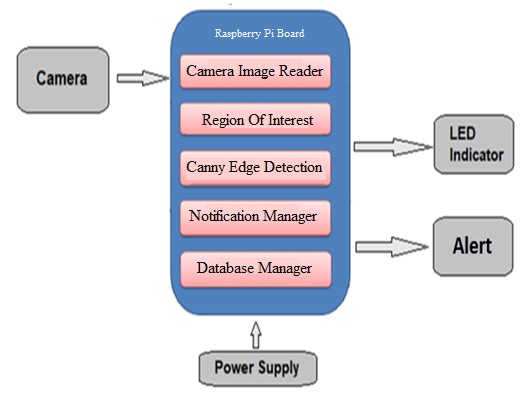


Fig.4.1 System Architecture

The CMOS camera, the image accession and processing module, and the interface module make up the LDWS, which is a element of the Vehicle Active Safety Perception Platform. Inside the test auto, a camera for recording frontal- view images are mounted between the frontal windscreen and the rear view glass. The digital media processor used by the image accession and processing module is grounded on Da Vinci TM technology. After decrypting from NTSC analogue to digital format, the CMOS camera may capture images of the road lanes and shoot them to a digital signal processor. To give the vehicle position and lane direction, the DSP core will identify and estimate using an internal algorithm. The system forecasts when the auto will arrive.

1. **Image Processing**

After gathering the picture data in front of the car, image processing is used to first prize the lane marking features and then fit the lane using these features. In reality, it's extremely delicate to describe lanes quickly and accurately due to light, dodging, and a variety of complex noise, so one of the crucial LDWS technologies is the development of an image processing system that can still produce results quickly and accurately under lower- than- ideal conditions[6].

* 1. Edge Detection
  2. Linear Model Fitting
  3. Setting of the Region of Interest (ROI)

Host Machine

Ubuntu (OS)



lane\_detection.py

Figure 4.1.1: Deployment diagram

The server and the client machine will be different and both will run there jar files as art of project.

**4.2 Mathematical Model**

YOLOv4 is an object detection model that is included in the Transfer Learning Toolkit.

**Training the Model**

Train the YOLOv4 model using this command:

tlt yolo\_v4 train [-h] -e <experiment\_spec>

-r <output\_dir>

-k <key>

[--gpus <num\_gpus>]

[--gpu\_index <gpu\_index>]

[--use\_amp]

[--log\_file <log\_file\_path>]

**Required Arguments**

* -r, --results\_dir: The path to the folder where the experiment output is written.
* -k, --key: The encryption key to decrypt the model.
* -e, --experiment\_spec\_file: The experiment specification file to set up the evaluation experiment. This should be the same as the training-specification file.

**Optional Arguments**

* --gpus: The number of GPUs to use for training in a multi-GPU scenario (default: 1).
* --gpu\_index: The GPU indices used to run the training. You can specify the indices of GPUs to use for training when the machine has multiple GPUs installed.
* --use\_amp: A flag to enable AMP training.
* --log\_file: THe path to the log file. The default path is stdout.
* -h, --help: Show this help message and exit.

**Input Requirement**

* **Input size**: C \* W \* H (where C = 1 or 3, W >= 128, H >= 128, W, H are multiples of 32)
* **Image format**: JPG, JPEG, PNG
* **Label format**: KITTI detection

**Sample Usage**

Here’s an example of using the train command on a YOLOv4 model:

tlt yolo\_v4 train --gpus 2 -e /path/to/spec.txt -r /path/to/result -k $KEY

**Evaluating the Model**

To run evaluation on a YOLOv4 model, use this command:

tlt yolo\_v4 evaluate [-h] -e <experiment\_spec\_file>

-m <model\_file>

-k <key>

[--gpu\_index <gpu\_index>]

[--log\_file <log\_file\_path>]

**Required Arguments**

* -e, --experiment\_spec\_file: The experiment spec file to set up the evaluation experiment. This should be the same as the training-specification file.
* -m, --model: The path to the model file to use for evaluation. The model can be either a .tlt model file or TensorRT engine.
* -k, --key: The key to load the model (not needed if the model is a TensorRT engine).

**Optional Arguments**

* -h, --help: Show this help message and exit.
* --gpu\_index: The GPU index used to run the evaluation. You can specify the index of a GPU to run evaluation when the machine has multiple GPUs installed. Note that evaluation can only run on a single GPU.
* --log\_file: The path to the log file. The default path is stdout.

**4.3 Data Flow Diagram**

A corporate information system's data flow is graphically represented using data flow diagrams. DFD is the term used to describe the procedures used in a system to move data from the input to file storage and report generation.

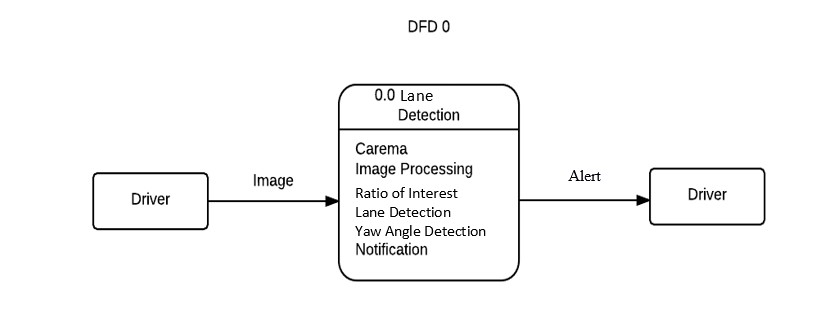


Figure 4.3.1: Data Flow Diagram (DFD 0)

It depicts the complete system as a single bubble with incoming/outgoing arrows designating input and output data. The camera's image serves as the input, and a buzzer is the result.

.

4.5.2 Sequence Diagram

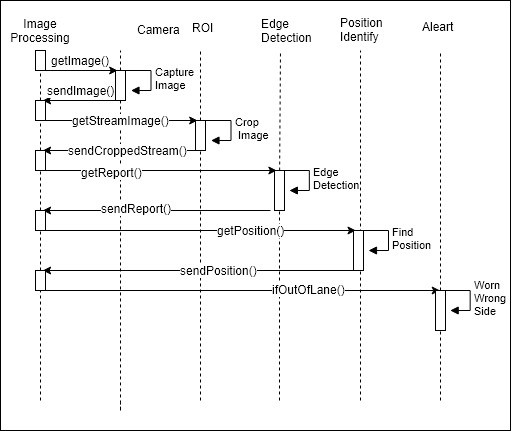


Figure 4.5.2: Sequence Diagram

4.5.3 Component Diagram

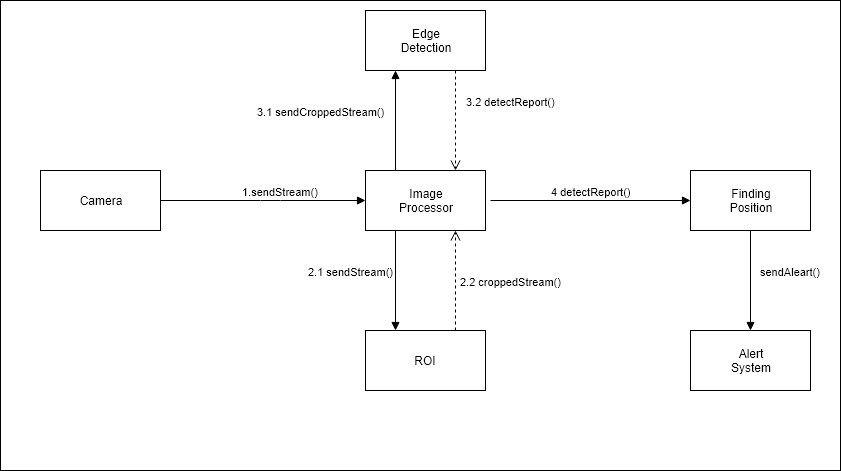


Figure 4.5.3: Component Diagram

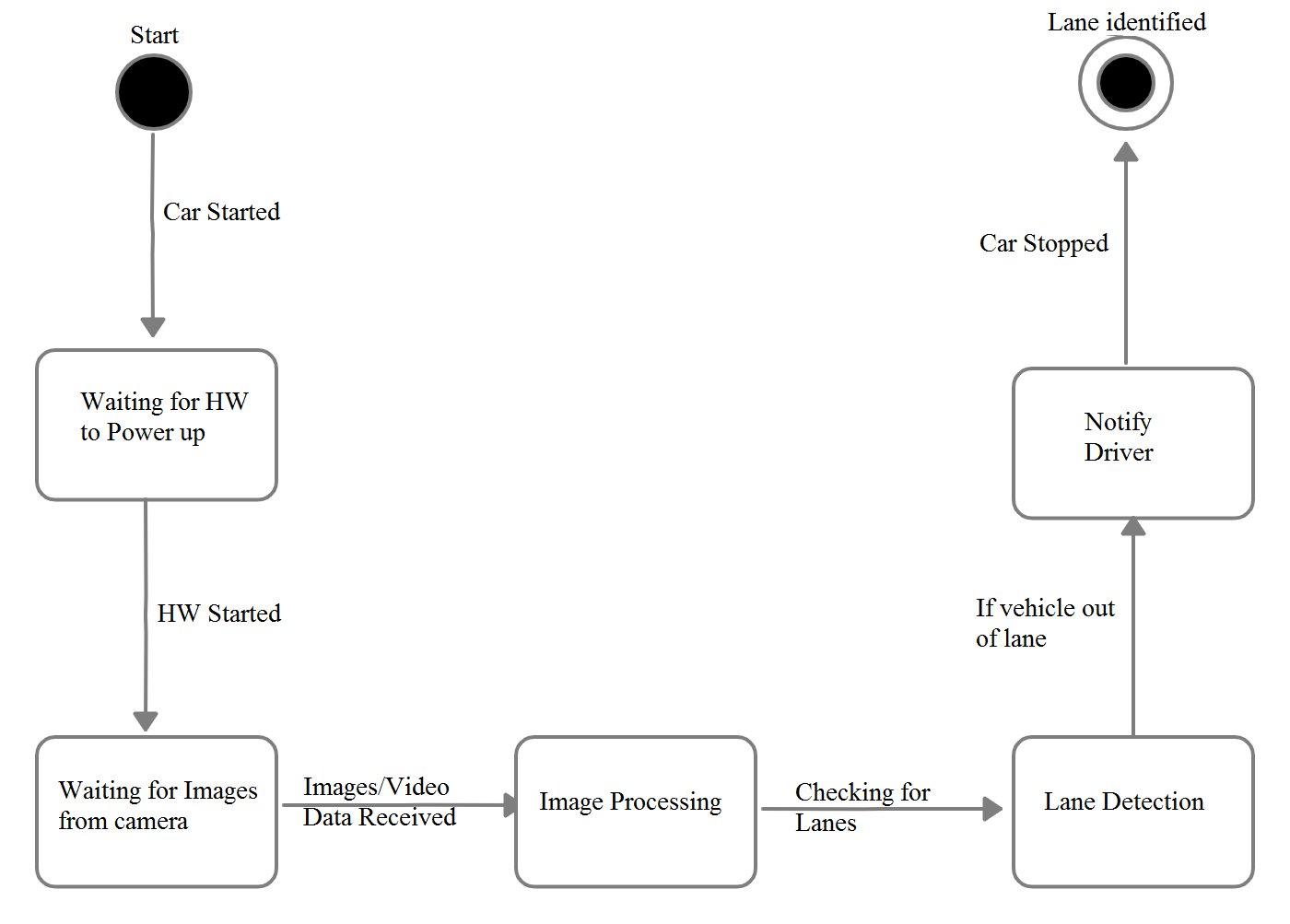
4.5.4 State Diagram ****

Figure 4.5.4: State Diagram

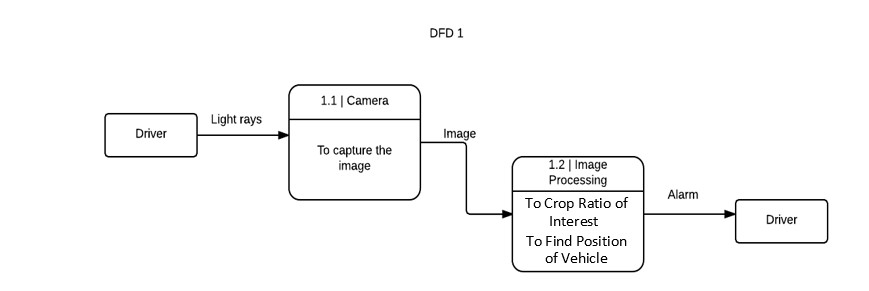


Figure 4.3.2: Data Flow Diagram (DFD 1)

The level-1 DFD highlights the main functions of the system and breakdown the high-level process of 0-level DFD into sub-processes.

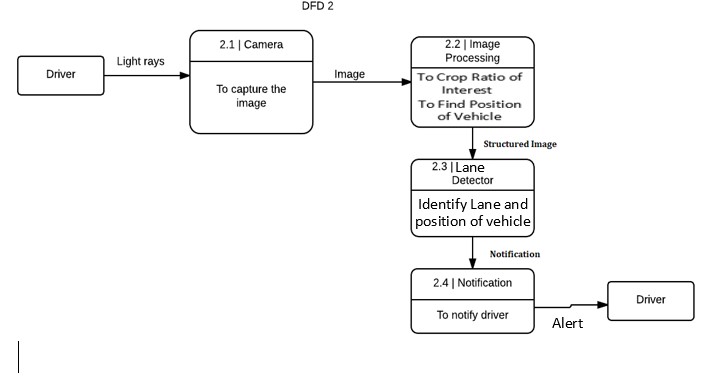


Figure 4.3.3: Data Flow Diagram (DFD 2)

Level-2 DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system’s functioning.

**4.4 Entity Relationship Diagram**

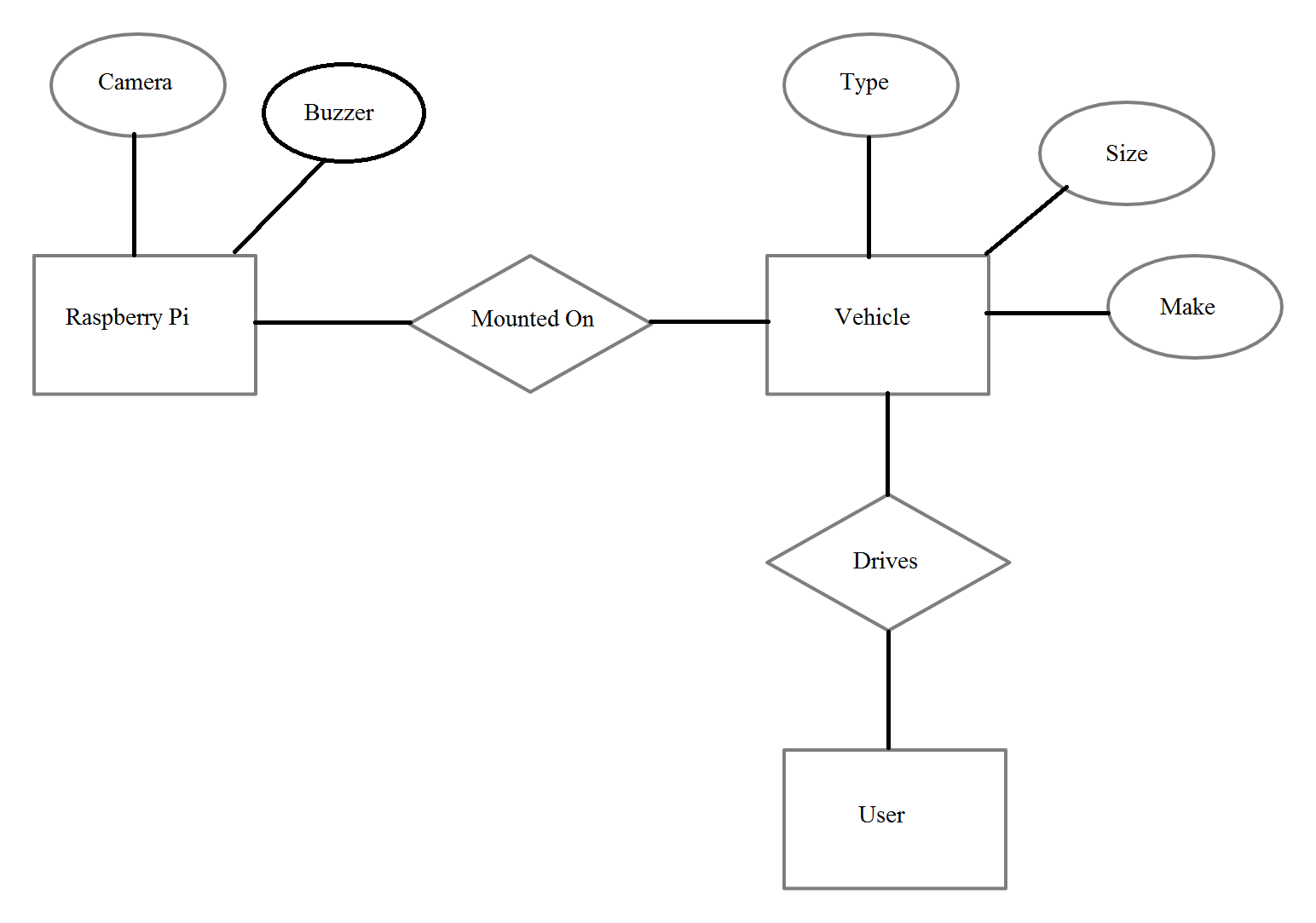
****

Figure 4.4: ER diagram

Entity Relationship Diagram, or ER Diagram for short, is a visual representation of the connections between reality sets that are maintained in a database. To put it another way, ER plates aid in describing the logical organisation of databases. Three original generalizations — reality, qualities, and connections form the base for the creation of ER plates. Blocks are used to represent realities, spheres are used to identify qualities, and diamond- shaped symbols are used to depict connections on ER plates.

**4.5 UML Diagrams**

4.5.1 Class Diagram

A class diagram in the Unified Modelling Language (UML) is a sort of static structural diagram used in software engineering that illustrates the classes, attributes, operations (or methods), and interactions between objects in a system to describe the system's structure.

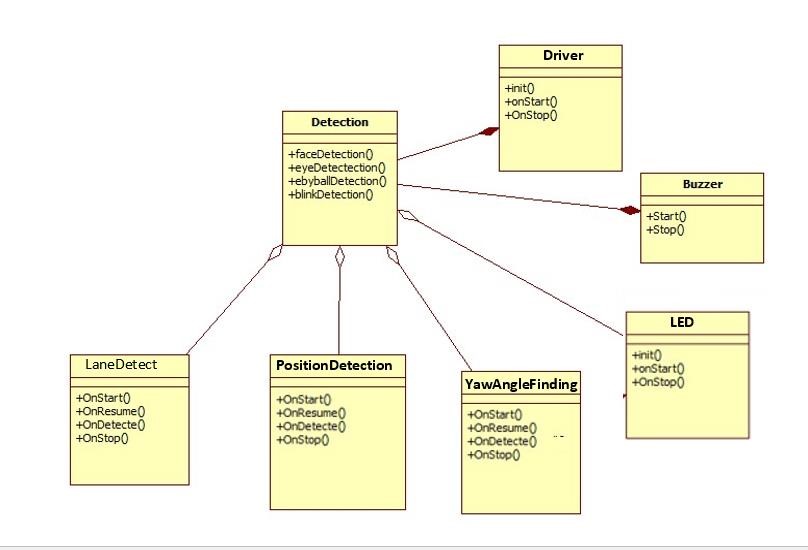


Figure 4.5.1: Class Diagram

**CHAPTER 5**

**PROJECT PLAN**

1. **PROJECT ESTIMATES**
2. **Reconciled Estimates**
3. **Cost Estimate**

The Constructive Cost Model (COCOMO) is generally used estimation measures of cost, project duration, man power, etc…

Like every other estimation model, COCOMO needs sizing information. This data can be specified in different ways

(OP)Object Point

(FP)Function Point (FP)

(KLOC)Lines of Source Code

In this project, we will use the sizing data in the form of Lines of source code.

**Equations**

* Equation for calculation of effort in person-month for the COCOMO model is:

E = a \* (KLOC) ^b

Where,

a=3.2

b=1.05, for semi-detached projects

E=Effort in person-months

D = E/N

Where,

E=Effort in person-months

N=Number of persons required

D=Duration of project in months.

**Estimation of KLOC**

KLOC according to module

|  |  |  |
| --- | --- | --- |
| Sr no. | Module | Estimated kloc |
| 1 | HW Connection Module | 0.7 |
| 2 | Authentication Module | 1.5 |
| 3 | Camera Module | 1.8 |
| 4 | Communication Module | 0.2 |
| 5 | Raspberry-Pi module | 1.2 |
| 6 | Road lane detection module | 1.5 |

Table 5.1: Estimation of KLOC

Total number of code required to estimate to be **6.2** KLOC.

1. **Time Estimates**

Efforts are calculated by using formula

E=3.2(KLOC) ^1.05… (Bohem simple model)

E=3.2(6.2) ^1.05

E=21.73 Person-month

**Development time:**

D=E/N

D=21.73 / 4

D=5.43 month

**Development time for Project**

* 1. Requirements analysis require 2 months
  2. Implementation and testing requires 3.43 months.

D= 5.43 months

1. **Project Resources**

* Developers - 2
* Testers -2
* Eclipse indigo Editor
* At least 2 laptops with Linux OS
* Internet connection
* Raspberry Pi Module with Camera

1. **RISK MANAGEMENT W.R.T. NP HARD ANALYSIS**

This part discusses the Project risks and the approaches in managing them.

1. **Risk Identification**

Following are the dome high level risk which wanted to highlight:

* Domain knowledge
* Technology will not Meet Expectations
* Lack of Development Experience
* Poor Quality Documentation
* Deviation from Software Engineering Standards
* Poor Comments in Code
* Changes in Requirements

1. **Risk Analysis**

The risks for the Project can be analysed within the constraints of time and quality

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Risk Defination | Prossibility | Impact | | |  |
|  |  |  |  |
| Agenda | Caliber | General |  |
|  |  |  |  |
|  |  |  |  |  |  |  |
| 1 | Domain knowledge | Low | Low | High | High |  |
| 2 | Technology will not Meet Expectations | Low | High | High | High |  |
| 3 | Lack of Development Experience | Medium | High | High | High |  |
| 4 | Poor Quality Documentation | Low | Low | Low | Low |  |
| 5 | Deviation from Software Engineering Standards | High | Low | High | High |  |
| 6 | Poor Comments in Code | Low | Low | Medium | Medium |  |
| 7 | Changes in Requirements | Medium | High | High | High |  |
|  |  |  |  |  |  |  |

Table 5.2.1: Risk Table

|  |  |  |
| --- | --- | --- |
| Probability | Value | Description |
|  |  |  |
| High | Probability of occurrence is | > 75% |
|  |  |  |
| Medium | Probability of occurrence is | 26 75% |
| Low | Probability of occurrence is | < 25% |
|  |  |  |

Table 5.2.2: Risk Probability descriptions

|  |  |  |  |
| --- | --- | --- | --- |
| Impact | Value | | Defintion |
|  |  |  |  |
| Very high | > 10% |  | Agenda impact or bad quality |
|  |  |  |  |
| High | 5 10% |  | Agenda impact or Some parts of the project have low Caliber |
|  |  |  |  |
|  |  |  |  |
| Medium | < 5% |  | Agenda impact or Barely noticeable degradation in caliber Low Impact on agenda or Caliber can be incorporated |
|  |  |  |  |
|  |  | Table 5.2.3: Risk Impact definitions | |

1. **Overview of Risk Mitigation, Monitoring, Management**

Following are the details for each risk.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk** | **Category** | **Possibility** | **Impact** | **RMMM plan (**Solution**)** |
| Computer Crash | Technical Issue | 65% | High | S3 |
| Domain knowledge | Business issue | 38 % | Medium | S2 |
| Technology will not Meet Expectations | Technology risk | 25 % | Low | S1, S6 |
| End Users Resist System | Business issue | 30 % | Low | S2, S4 |
| Changes in Requirements | Product size risk | 60 % | High | S5 |
| Lack of Development Experience | Technical Issue | 20 % | Low | S4 |
| Lack of Database Stability | Technical Issue | 40 % | Medium | S1 |
| Poor Quality Documentation | Business issue | 35 % | Medium | S6, S2 |
| Deviation from Software Engineering Standards | Process risk | 10 % | Low | S3 |
| Poor Comments in Code | Technical Issue | 50 % | Medium | S5,S1 |

Table 5.2.4: Overview of Risk Mitigation, Monitoring, Management

**Solution 1:**

When working on the product or documentation, the staff member should always be aware of the stability of the computing environment they’re working in. Any changes in the stability of the environment should be recognized and taken seriously.

**Solution 2:**

The schedule will be followed closely during all development stages. Steps have been taken to ensure a timely delivery by gauging the scope of project based on the delivery deadline.

**Solution 3:**

The customer meetings should make sure that both our company and the customer are clear on the needs of the product. If the development team discovers that the client's notion of the product specs differs from their own, they should notify the customer right away and take whatever steps are required to fix the issue.

**Solution 4:**

In order to prevent this from happening, the software will be developed with the end user in mind. The user-interface will be designed in a way to make use of the program convenient and pleasurable.

**Solution 5:**

Meetings with the customer will be arranged on a regular basis (both official and informal) to prevent this from happening. This guarantees that the product we are making and the client's needs are the same.

**Solution 6**

If commenting rules are more clearly stated, bad code commenting can be reduced. Although informal norms have been suggested, none are currently in place. To guarantee the calibre of comments in every code, a formal documented standard needs to be developed.

1. **PROJECT AGENDA**
2. **Project task set**

Major Tasks in the Project stages are:

Task 1: Communication

Task 2: Planning

Task 3: Risk Management

Task 4: Modelling

1. **Task network**

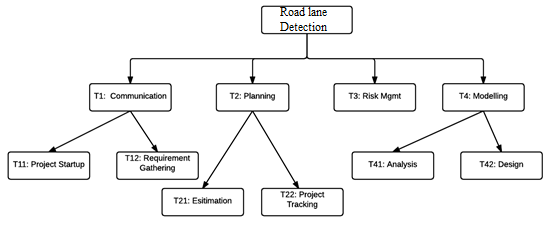
Here are the project tasks and their dependencies in this diagrammatic form.

Fig 5.3.2 Task Network

T-A: Communication

The communication between the client and the inventor is the first step in the software development process. We gathered the design-affiliated conditions in agreement with the design's requirements..

T-B: Planning

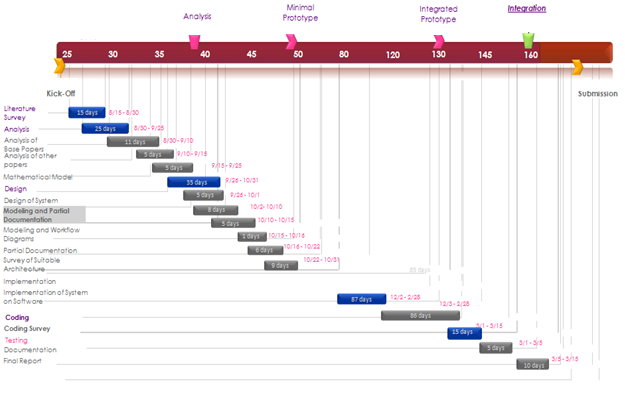
Entire estimating and scheduling are comprehended.

T-C: Risk Management

This entails feting hazards when they arise throughout design development and managing pitfalls that have an impact on design development.

T4: Modelling

This contains detailed requirement analysing and project designing.



1. **Timeline Chart**

Fig 5.3.3 Timeline Chart

**5.4 Team Association**

**5.4.1 Team Structure**

One of the major obstacles to a project's success is an effective project team structure. Without a productive and well-organized team, there is a higher likelihood that the project would fail right away since the team is initially unable to complete the task properly. People that make up a team will struggle to perform a number of distinct jobs and handle a variety of group and individual duties without proper collaboration organization. As a result, before beginning a new project, you must ensure that the project team is organized effectively through team building.

**Number of team members: 4**

Member 1: Coding and Testing

Member 2: Coding and Testing

Member 3: Testing and Documentation

Member 4: Testing and Documentation

**5.4.2 Management reporting and communication:**

The hierarchy of individuals, their purpose, workflow, and reporting system are all determined by the management reporting and communication setup. Each team member gives an update on the status of their allocated responsibilities during a weekly meeting. Determine the most effective solutions to a problem and eliminate high-impact risks as soon as possible. Where possible, look for methods to improve the algorithm or implementation. Assign new duties to each member and discuss the next stages.

**CHAPTER 6**

**PROJECT IMPLEMENTATION**

**6.1 Overview of Project Modules**

Overview of Project Modules relating lane road is truly usual task that mortal motorist performs. It is needed to keep the motor vehicle in the conditions of the lane. This is also veritably critical task for an independent vehicle to carry out. And veritably simple Lane Detection channel is possible with simple Computer Vision ways. This report will describe simple channel that can be used for simple lane discovery using ‘Python and OpenCV’. .

We have provided following features into our system:

1. Tackle can capture the images
2. Tackle can descry the road lanes
3. Tackle can on buzzer when cross the road lane
4. Tackle can on led when cross the lane

# **6.2 Tools and Technologies Used**

**Tools and Technologies Used:**

**Tools:**

* Raspberry-Pi Board
* WinScp
* Putty
* Remote desktop connection

**Technologies:**

* Python
* Open CV

# **6.3 Algorithm Details**

Lane Detection Pipeline:

1. Change initial picture to grayscale.
2. Darkening of the grayscale picture (this help in reducing contrast from discoloured regions of road)
3. Change initial picture to HLS colour space.
4. Separate yellow from HLS to get yellow mask. ( yellow lane)
5. Separate white from HLS to get white mask. (white lane)
6. Do Bit-wise OR yellow and white masks to get common mask.
7. Do Bit-wise AND mask with darkened picture .
8. Then Apply slight Gaussian Blur.
9. Application of canny Edge Detector (adjust the thresholds — trial and error) to get edges.
10. Define Region of Interest. It will result in weeding out unwanted edges detected by canny edge detector.
11. Recoup Hough lines.
12. Consolidate and conclude the Hough lines and draw them on initial picture.



Fig 6.3 Original Image

#### **6.3.1 Convert to grayscale**

Changing the initial picture to grayscale has its benefits. Finding yellow and white lanes, and changing initial picture to grayscale will increase the contrast of lanes, respectful to road.



Fig 6.3.1.1 Grayscale Image

**6.3.2 Change initial picture to HLS colour space**

Initial pictures are in RGB, but we must also explore other colour spaces like [HSV](https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_hsv) and [HLS.](https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_hls) Side-by-side, when looked at ,it can easily be seen that we can get better colour contrast in HLS colour space from road. This may help in good colour selection and lane detection.

.

Fig 6.3.2 HLS Colour Images

#### **Chossing Colour**

Also, we use OpenCV’s in range to receive mask between the thresh hold values. Later, after some trial and error, we can now find out range for threshold.

Unheroic mask:

1. Hue value was used between 10 and 40.
2. We use advanced achromatism value (100 – 255) to avoid unheroic from hills.

white mask:

1. We use advances lightness value (200–255) for white mask.

Then We bit-wise OR both mask to get combined mask.

The pictures Below show combined mask being bit-wise AND with darkened image.

#### **6.3.3 Gaussian Blur**

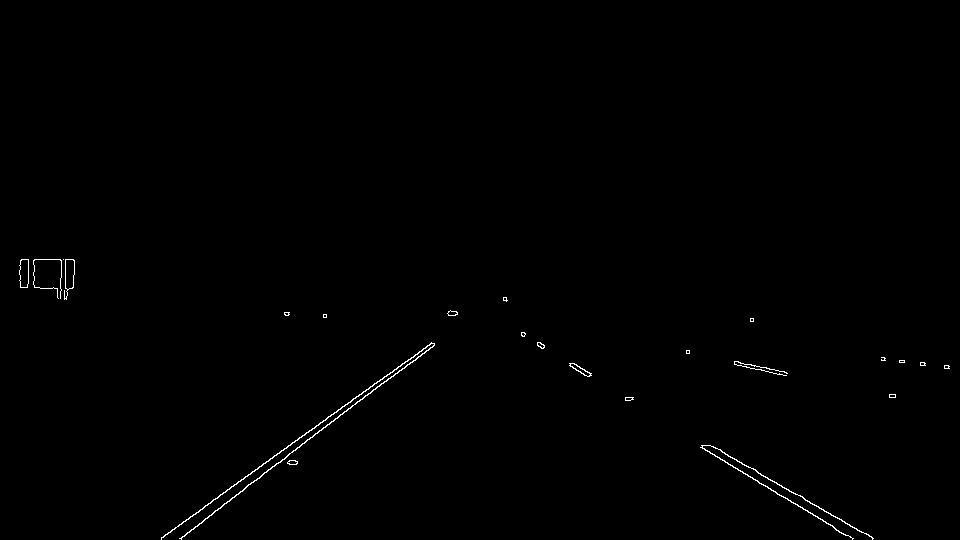
It is a pre-processing step we use to reduce the noise from picture (or to smooth the picture). It is used when pre-processing step to remove numerous detected edges and only keep the most visual edges from the picture.

* + 1. **Canny Edge Detection**

Application of Canny edge detection to these Gaussian blurred pictures. It is an algorithm that detects edges grounded on grade change. Note: the first step of Canny Edge detection is picture smoothing with dereliction kernel size 5, we still have to apply unequivocal Gaussian blur in former step. Other way include-

Chancing Intensity grade of the Image

* Non-maximum repression
* Hysteresis Thresholding



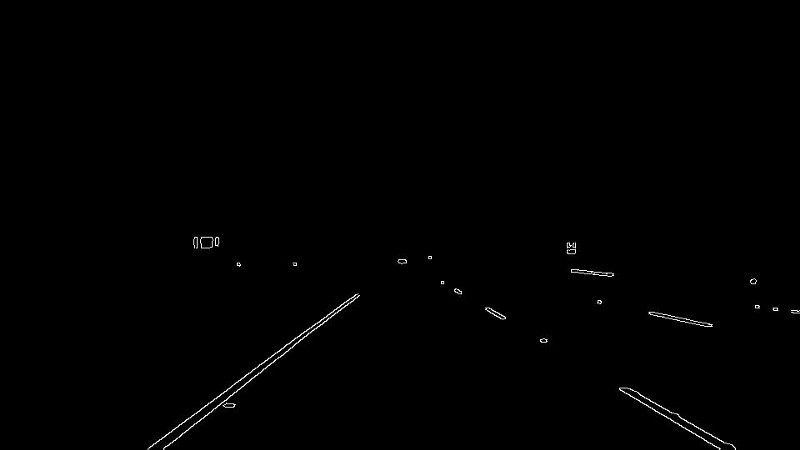


Fig 6.3.4.1 Canny Edge Detection

**Choose (ROI) Region of Interest**

After applying Canny Edge Detection, there are many edges that are detected which aren’t lanes. ROI is a polygon that describes area in the picture, from where edges we are interested in.

#### **6.3.5 Hough Transformation Lines Detection**

It is the technique to find out lines by getting to know all points on the line. This can be done by representing a line as a point. Whereas points are represented as lines. And if multiple lines pass through the point, we can then deduce that these points lie on the same line.



Fig 6.3.5 Hough Transform

**Python**

It is a major high- position, object- oriented programming language created by Guido van Rossum. It has plain effortless- to- use syntax, making it the complete language for someone trying to learn computer programming for the first occasion. Python is a common idea programming language created in the late 1980s, and named after Monty Python, that’s exercised by thousands of people to do things from testing microchips at Intel, to powering Instagram, to putting up video games with the PyGame library. It’s small, truly nearly resembles the English language, and has hundreds of being third- party libraries. Python is a dynamic, interpreted( bytecode- collected) language. There are no type declarations of variables, parameters, functions, or styles in source law. This makes the law short and flexible, and you misplace the collect- time type checking of the origin law. Python tracks the types of all valuations at runtime and flags crack that does not form sense as it runs.

**Libraries**

It is a major high- position, object- oriented programming language created by Guido van Rossum. It has plain effortless- to- use syntax, making it the complete language for someone trying to learn computer programming for the first occasion. Python is a common idea programming language created in the late 1980s, and named after Monty Python, that’s exercised by thousands of people to do things from testing microchips at Intel, to powering Instagram, to putting up video games with the PyGame library. It’s small, truly nearly resembles the English language, and has hundreds of being third- party libraries. Python is a dynamic, interpreted( bytecode- collected) language. There are no type declarations of variables, parameters, functions, or styles in source law. This makes the law short and flexible, and you misplace the collect- time type checking of the origin law. Python tracks the types of all valuations at runtime and flags crack that does not form sense as it runs.

.

**Community**

Python has stoner groups far and wide, generally called prize fighters, and does major conferences on every mainland other than Antarctica. PyCon NA, the largest Python conference in North America, sells out its 2,500 tickets this time. And, reflecting Python’s commitment to diversity, it had over 30 women speakers. PyCon NA 2013 also started a trend of offering “ youthful Coder ” shops, where attendees tutored Python to kiddies between 9 and 16 times of age for a day, getting them familiar with the language and, eventually, helping them hack and mod some games on the jeer Pis they were given. Being part of a similar a positive community does a lot to keep you motivated. Check out PyCon to find your neareth Python conference.

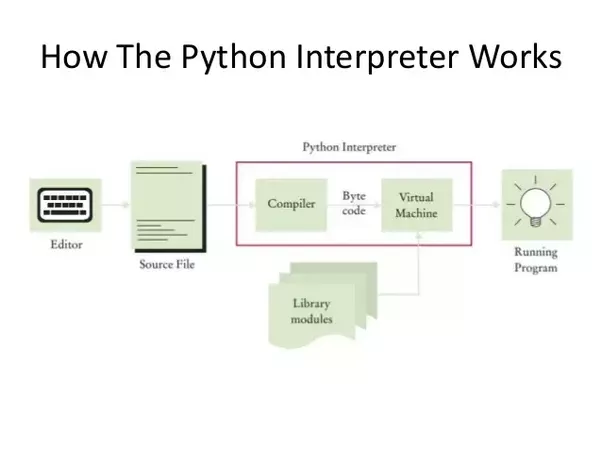


Fig 6.3.5.1 Python Interpreter

**OpenCV-Python**

OpenCV- Python is a library of Python tapes designed to break computer vision problems.

Python is a general purpose programming language started by Guido van Rossum that came veritably popular veritably snappily, substantially because of its simplicity and law readability. It enables the programmer to express ideas in smaller lines of law without reducing readability.

Compared to languages like C/ C, Python is slower. That said, Python can be fluently extended with C/ C, which allows us to write computationally ferocious law in C/ C and produce Python wrappers that can be used as Python modules. This gives us two advantages first, the law is as presto as the initial C/ C law (since it's the factual C law working in background) and second, it easier to decode in Python than C/ C. OpenCV- Python is a Python wrapper for the initial OpenCV perpetration.

OpenCV- Python makes use of NumPy, which is a largely optimized library for numerical operations with a MATLAB- style syntax. All the OpenCV array structures are changeed to and from NumPy arrays. This also makes it easier to integrate with other libraries that use NumPy similar as SciPy and Matplotlib.

# 

# **CHAPTER 7**

# **SOFTWARE TESTING**

**7.1 Testing Information**

**Software Testing:**

Software testing is a critical step in the creation of software that involves assessing the quality, performance, and functionality of an application. In order to make sure the software satisfies the given criteria and expectations, it seeks to find flaws, faults, and holes in it. Important details concerning software testing Unit testing, integration testing, system testing, acceptance testing, performance testing, security testing, regression testing, usability testing, and exploratory testing are all types of software testing that are carried out at different phases of the software development life cycle( SDLC). Depending on the complexity and requirements of the product, it can be done manually or through automated testing methods.

**Verification:**

Verification and validation are the two main procedures involved in software testing. Despite the fact that these words are frequently used interchangeably, they have different connotations when referring to software testing.

Verification is concerned with assessing the software at every stage of the development process to make sure it complies with the requirements. Does it solve the issue of "Are we building the software right?"? Static analysis, walkthroughs, inspections, and reviews are examples of verification activities. It seeks to verify that the software complies with the established requirements, specifications, and laws

**Validation:**

Validation focuses on assessing the software during or after the development process to make sure it fulfills the intended usage and meets the client's expectations. Is the software we're creating the proper kind? is a question that is addressed. Dynamic testing, user feedback, and comparing the software to user requirements are all part of the validation process.

**Software testing basics:**

Blackbox testing and whitebox testing are the two fundamental types of software testing.

**Blackbox Testing:**

Black box testing is an approach to testing that focuses on the outputs of any input and system execution while ignoring the internal operations of the system. sometimes referred to as functional testing.

**Whitebox Testing :**

A testing technique called "white box testing" takes into account a system's internal operations. Other names for it include structural testing and glass box testing.

Black box testing is typically used for validation whereas white box testing is typically utilised for verification.

**7.2 Types of testing:**

There are many types of testing like

* Unit Testing
* Integration Testing
* Functional Testing
* System Testing
* Stress Testing
* Performance Testing
* Usability Testing
* Acceptance Testing
* Regression Testing
* Beta Testing

**Unit Testing:**

A single unit or a group of connected units is examined during unit testing. White box testing is the category in which it falls. The programmer frequently checks to see if the unit he or she developed is providing the desired results given the input.

**Integration Testing:**

Integration testing involves combining a number of components to produce an output. If there is a relationship between software and hardware components, integration testing also examines how the two interact. Both white box testing and black box testing may apply.

**Functional Testing:**

The operational of the functionality specified in the system requirements is confirmed by a functional test. It falls under the umbrella of black box testing.

**System Testing:**

System testing entails putting the software in different conditions (like those created by different operating systems) to verify if it still works. System testing is done in a fully functional system environment. It falls under the umbrella of black box testing.

**Stress Testing:**

The evaluation of a system's performance under stressful conditions is known as stress testing. Testing is conducted above and above what is necessary. It falls under the umbrella of black box testing.

**Performance Testing:**

Performance testing is the process of determining whether a system meets performance standards in terms of speed, efficiency, and ability to deliver outcomes in a set amount of time. It falls under the umbrella of black box testing.

**Usability Testing:**

In order to determine how user-friendly the GUI is, usability testing is done from the client's point of view. How fast is the customer able to pick things up? When a customer has mastered the use, what level of skill is possible? Its use is how aesthetically pleasing? This falls under the heading of black box testing.

**Acceptance Testing:**

Acceptance testing is commonly done by the client to make sure the delivered product conforms with the specifications and functions as the customer intended.

**Regression Testing:**

Regression testing involves assessing a system, component, or group of connected units after a change has been done to ensure that the change is successful and hasn't damaged or compelled other modules to produce unanticipated results. It falls under the umbrella of black box testing.

**Beta Testing:**

Software testing, known as beta testing," is carried out by a small number of end users or other external stakeholders who are not members of the development team. It is usually carried out following the conclusion of internal beta testing, in which the development team tests the product.

The main goal of beta testing is to get input from actual users and find any faults, bugs, or usability concerns that might have gone unnoticed in previous testing stages. Developers can learn more about how users engage with the software during this phase, collect ideas for enhancements, and evaluate the software's performance in various settings.

**7.3 Test Cases & Test Results**

|  |  |
| --- | --- |
| Test Case ID | 1 |
| Test Case Description | Hardware should start. |
| Steps | 1.Start Vehicle  2.Start Hardware |
| Test Case Result | Hardware will start the camera. |
| Action Result | Hardware will start the camera. |
| Status | PASS |

|  |  |
| --- | --- |
| Test Case ID | 2 |
| Test Case Description | System should allow camera to capture images. |
| Steps | 1.Start Vehicle  2.Start Hardware  3. Start camera.  4.Capture the image |
| Test Case Result | System will start camera and capture the image. |
| Action Result | System will start camera and capture the image. |
| Status | PASS |

|  |  |
| --- | --- |
| Test Case ID | 3 |
| Test Case Description | System should verify the quality of capture image. |
| Steps | 1.Start Vehicle  2.Start Hardware  3. Start camera.  4. Capture the image.  5. Verify quality of image. |
| Test Case Result | System will capture the image and verify the quality of image. |
| Action Result | Application will capture the image and verify the quality of image. |
| Status | PASS |

|  |  |
| --- | --- |
| Test Case ID | 4 |
| Test Case Description | Quality of captured image will be bad |
| Steps | 1.Start Vehicle  2.Start Hardware  3. Start camera  4. Capture Image  5.verify the quality of image if it is bad  6. Capture image again. |
| Test Case Result | System will capture image if image quality is bad. Application will again capture the new image. |
| Action Result | System will capture image if image quality is bad. Application will again capture the new image. |
| Status | PASS |

|  |  |
| --- | --- |
| Test Case ID | 5 |
| Test Case Description | Quality of captured image will be good |
| Steps | 1.Start Vehicle  2.Start Hardware  3.Start camera  4. Capture image  5. Verify the quality of image  6. Use for image processing |
| Test Case Result | System will verify that quality of image will be good. |
| Action Result | System will verify that quality of image will be good. |
| Status | PASS |
| Test Case ID | 6 |
| Test Case Description | System should detect the road lane |
| Steps | 1.Start Vehicle  2.Start Hardware  3.Start camera  4.Capture image  5.verify that quality of image will good  6.detect the road lane |
| Test Case Result | System will verify the image quality and detect the road lane |
| Action Result | System will verify the image quality and detect the road lane |
| Status | PASS |

|  |  |
| --- | --- |
| Test Case ID | 7 |
| Test Case Description | System should On buzzer when vehicle cross the lane |
| Steps | 1.Start Vehicle  2.Start Hardware  3.Start camera  4. Capture image  5. verify image quality  6.detect road lane  7. If cross lane buzzer on |
| Test Case Result | System can on buzzer when vehicle cross the lane |
| Action Result | System can on buzzer when vehicle cross the lane |
| Status | PASS |

|  |  |
| --- | --- |
| Test Case ID | 8 |
| Test Case Description | System should on led when vehicle cross the cross the lane |
| Steps | 1.Start Vehicle  2.Start Hardware  3. Start camera  4. Capture image  5. Verify image quality  6. Detect lane  7. If cross the lane the on led |
| Test Case Result | System can on led when vehicle cross the cross the lane |
| Action Result | System can on led when vehicle cross the cross the lane. |
| Status | PASS |

|  |  |
| --- | --- |
| Test Case ID | 9 |
| Test Case Description | System should not on led when vehicle does not cross the cross the lane |
| Steps | 1.Start Vehicle  2.Start Hardware  3.Start camera  4.Capture image  5.verify that quality of image will good  6.detect the road lane  7. led off |
| Test Case Result | System cannot on led when vehicle does not cross the cross the lane |
| Action Result | System cannot on led when vehicle does not cross the cross the lane |
| Status | PASS |

|  |  |
| --- | --- |
| Test Case ID | 10 |
| Test Case Description | System should not on led when vehicle does not detect zebra crossing |
| Steps | 1.Start Vehicle  2.Start Hardware  3.Start camera  4. Capture image  5. verify image quality  6.detect lane  7. led off |
| Test Case Result | System cannot on led when vehicle does not detect zebra crossing |
| Action Result | System cannot on led when vehicle does not detect zebra crossing |
| Status | PASS |

Table 7.2 Test Cases

# **CHAPTER 8**

# **RESULTS**

**8.1 Outcomes**

The implementation of Road Lane Detection, Zebra Crossing Detection, Road Sign Detection, and Pedestrian/Object Detection has been effective. As anticipated, our model's accuracy is greater than that of the models that are already in use and the models that we have evaluated. The model passes each of the test cases as described in Chapter 7 with success. Using the proper notifications, the hardware successfully notifies the user via Bluetooth speaker, the built-in speaker of the device to which it is connected, or via Buzzer. Since we need a quick response when a driver changes lanes or travels outside of his present lane, the speed of execution on real-time data is good. The Zebra Crossing/Crosswalk Detection Module can identify several crosswalks in a video, if there are any. Tensorflow has been utilised by us to identify.

**CHAPTER 9**

**CONCLUSIONS**

**9.1 Conclusions**

Intelligent transport system lane detection has shown to be a good accident prevention technique. Our strategy will increase the accuracy of the system's lane detecting. Based on the findings of the investigations, our technique succeeds in achieving its objective in support of lane change and warning. In good weather, the suggested technique is expected to detect over 98% of the details in a picture and over 96% in less than perfect conditions.

**9.2 Future Work**

The visibility of roads is impaired when the weather is foggy, this aspect can be worked upon.

A database can be used to store the record of already visited potholes, blockages on the road which may have been visited in the past while passing through the same route. And aim to reduce the number of accidents caused on the roads and also to improve the seriousness of such accidents.

**References**

1. Yue Chen,Azzedine Boukerche Paradise Research Lab,EECS,University of Ottawa,Canada,”Novel Lane Departure Warning System for Improving road safety ”,1 Aug 2020.
2. S. Srivastava, R. Singal and M. Lumb, “ Efficient Lane Detection Algorithm using Different Filtering Techniques”, International Journal of Computer Applications, vol. 88, no.3, pp. 975-8887, 2014.
3. A. Borkar, M. Hayes, M.T. Smith and S. Pankanti , “A Layered Approach To Robust Lane Detection At Night” , In IEEE International Conference and Exposition on Electrical and Power Engineering, Iasi, Romania, pp. 735 - 739, 2011.
4. 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.
5. Z. Kim, “Robust Lane Detection and Tracking in Challenging Scenarios”, In IEEE Transactions on Intelligent Transportation Systems, vol. 9, no. 1, pp. 16 - 26, 2008.
6. M. Aly, “Real time Detection of Lane Markers in Urban Streets”, In IEEE Intelligent Vehicles Symposium, pp. 7 - 12, 2008.
7. J.C. McCall and M.M. Trivedi, “Video-based Lane Estimation and Tracking for Driver Assistance: Survey, System, and Evaluation”, IEEE Transactions on Intelligent Transportation Systems, vol.7, pp.20-37, 2006.
8. Y.Wang, E. K.Teoh and D. Shen, “Lane Detection and Tracking Using B-snake,” Image and Vision Computing, vol. 22, pp. 269-280, 2004.
9. A. Broggi and S. Berte, “Vision-based Road Detection in Automotive Systems: a Real-time Expectation-driven Approach”, Journal of Artificial Intelligence Research, vol.3, pp. 325-348, 1995.
10. M. Bertozzi and A. Broggi, “GOLD: A Parallel Realtime Stereo Vision System for Generic Obstacle and Lane Detection”, IEEE Transactions of Image Processing, pp. 62-81, 1998.