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.

TensorFlow

TensorFlow is an open-source machine learning and artificial intelligence software library that is available for free. Although deep neural networks are capable of a wide range of tasks, training and inference are given particular emphasis. 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.

Calculations can be swiftly distributed across a variety of platforms (CPUs, GPUs, and TPUs), from desktops to clusters, thanks to its extensible armature..

CHAPTER 2

LITERATURE SURVEY

Data Collection and Processing Methods (2008) Recently, several product vehicles have been equipped with road departure discovery systems (RDDSs) to prevent road departure crashes. In order to enable and offer an objective and standardised performance evaluation of RDDSs, this study describes the development of the data collecting and data post-processing tools for testing RDDSs. Seven parameters are used to explain road departure test scripts. For the purpose of evaluating vehicle RDDSs, the general architecture and essential elements of the data collection and post-processing systems have been created and presented. The experimental results shown that a seeing system and data post-processing system could display the vehicle stirring characteristic from the testing vehicle right away and collect all necessary signals.

The University of Ottawa in Canada's Yue Chen and Azzedile Boukerche[1], 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.

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.

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).

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.

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 capture the proper images with good pixel quality.
- System should detect the road lane in which vehicle should move on.
- System should detect the signs on road

Notification Module

- System should notify to user through alarm when vehicle cross the lane.
- System should notify when cross the lane.
- System should notify when crosswalk is detected
- System should notify when sign is recognized.

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.

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

To address computer vision problems, OpenCV-Python, a set of Python programs, was developed. Python is a general-purpose programming language that was created by Guido van Rossum and quickly became well-known, partly because of its readability and simplicity.

Now, notions may be expressed by the programmer in fewer lines of code without compromising readability. Python is slower than C/C-like languages. However, C/ C can be easily integrated with Python, allowing us to write computationally demanding laws in C/ C and produce Python wrappers that may be utilized as Python modules. We gain from this in two ways. First off, the law is just as effective as the original C/C law because the real C law is running in the background.

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

System must be able to determine the proper detection lanes using high-quality cameras.

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.

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

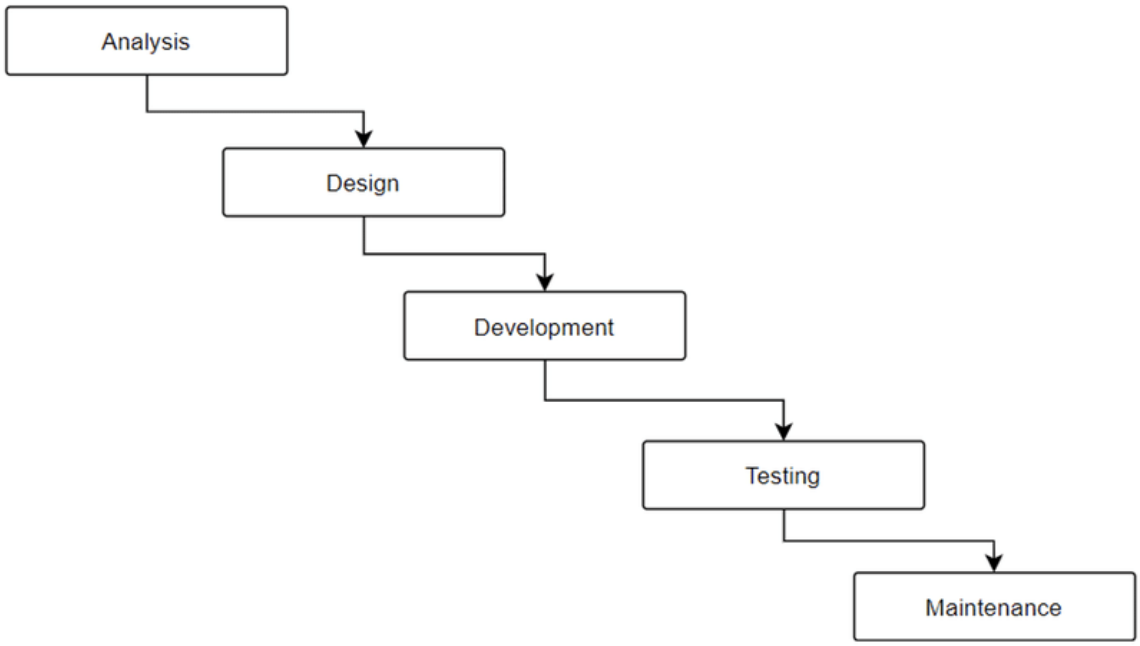
- Programming Language: Python
- OpenCV library

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

3.6 Models for analysis: SDLC to be used

Waterfall Model:



Software engineering and product development frequently use the Software Development Lifecycle (SDLC). The SDLC is approached directly and sequentially using the waterfall methodology. The waterfall model, which replicates the logical development of water cascading over a cliff's edge, is used to apply the SDLC process for a design. At each level of progress, it creates precise goals or criteria.

These endpoints or pretensions cannot be altered once they have been reached.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. Prior to beginning any building work, a structure's physical design is prepared..

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

3. The framework of the building is finalized before the walls are constructed.

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 have a high degree of process clarity and little to no inherent variability. Waterfalls are a fantastic alternative if money or time are constraints on the design.

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.

Because it sets out to do that, the waterfall is an effective software development approach. This is crucial if an application must work perfectly on the first try or risk losing users.

Consider the Agile management of projects and development process as an alternative. Agile methods utilise a methodology called continuous iteration, which consists of the creation, creation, and testing of software in a series of iterative phases 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.

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

CHAPTER 4
SYSTEM DESIGN USING HOUGH TRANSFORMATION AND CANNY EDGE DETECTION

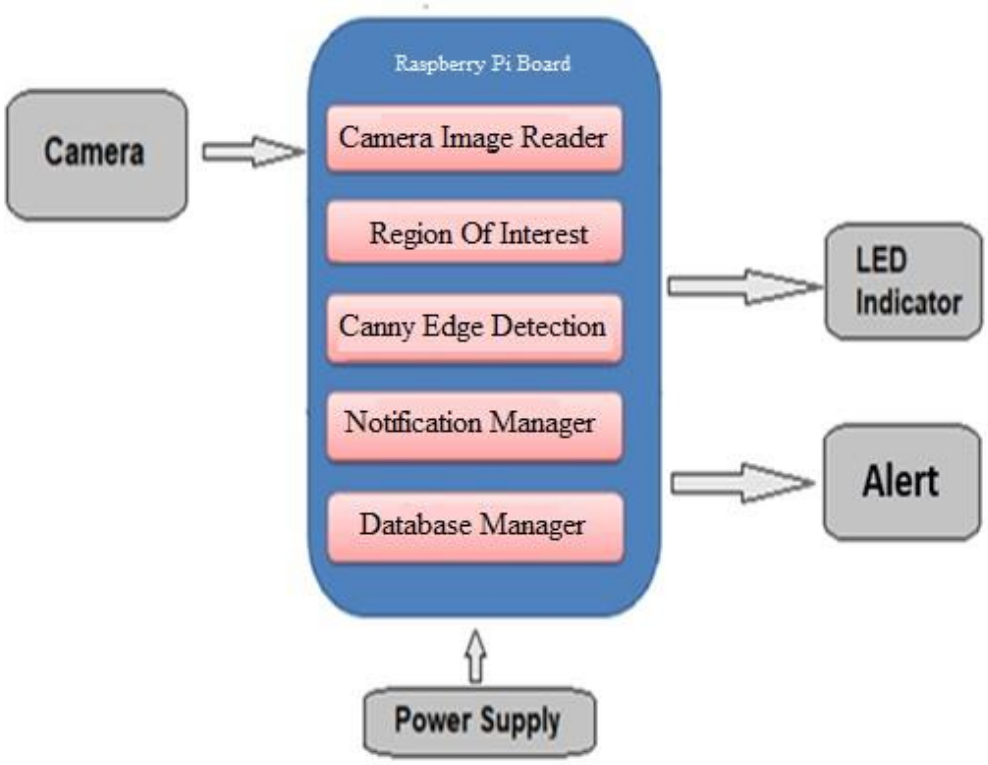


Fig.4.1 System Architecture

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.

A. 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)

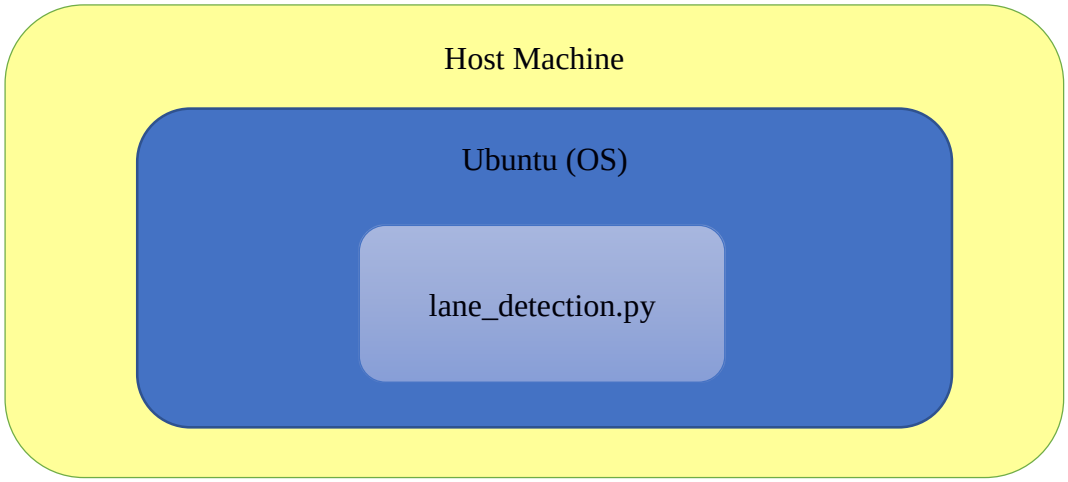


Figure 4.2: Deployment diagram

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

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 SxS region in each of the N grids created by the YOLO algorithm is the same size. Each of these N grids is responsible for finding and locating the object it contains.

The Transfer Learning Toolkit includes the object detection model YOLOv4.

Training the Model

Use the following command to train the YOLOv4 model:

```
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 location of the experiment output in that folder.
- `-k, --key`: the model's decryption key for encryption.
- `-e, --experiment_spec_file`: the file used to specify an experiment for the evaluation experiment. This should match the training-specification file exactly.

Optional Arguments

- `--gpus`: The default value for the number of GPUs to utilize for training in a multi-GPU scenario is 1.
- `--gpu_index`: You can specify the GPU indices to be used for training when the machine has multiple GPUs installed. This allows you to select specific GPUs for the training process.
- `--use_amp`: Use the flag to enable AMP training
- `--log_file`: The default path for the log file is not specified. `stdout`.
- `-h, --help`: Display the help message and terminate the program.

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

```
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 specification file used to configure the evaluation experiment should be identical to the training-specification file.
- -m, --model. Specify the path to the model file to be used for evaluation. The model file can be either a .tlt model file or a TensorRT engine.
- `-k, --key`: The key required to load the model is not necessary if the model is a TensorRT engine.

Optional Arguments

- `-h, --help`: To display the help message and exit the program, use the following command

- `--gpu_index`: You can specify the GPU index to be used for running the evaluation when the machine has multiple GPUs installed. It is important to note that evaluation can only be performed on a single GPU.
- `--log_file`: The path to the log file. The default path is `stdout`.

A corporate information system's data flow is visually represented using data flow diagrams (DFDs). DFDs are the methodologies employed within a system to facilitate the movement of data from input sources to file storage and the generation of reports.

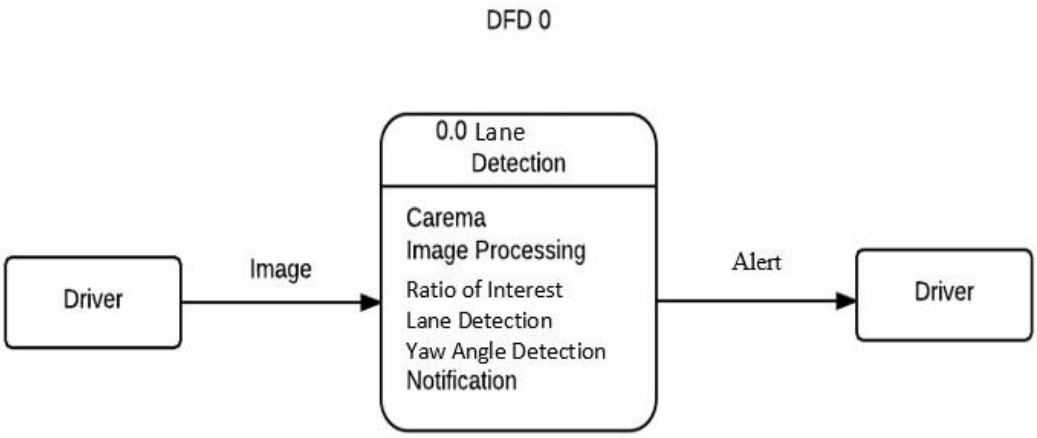


Figure 4.3: Data Flow Diagram (DFD 0)

In the data flow diagram, the entire system is represented by a single bubble, with incoming and outgoing arrows indicating input and output data respectively. In this particular scenario, the input data is provided by the camera's image, and the output is represented by a buzzer.

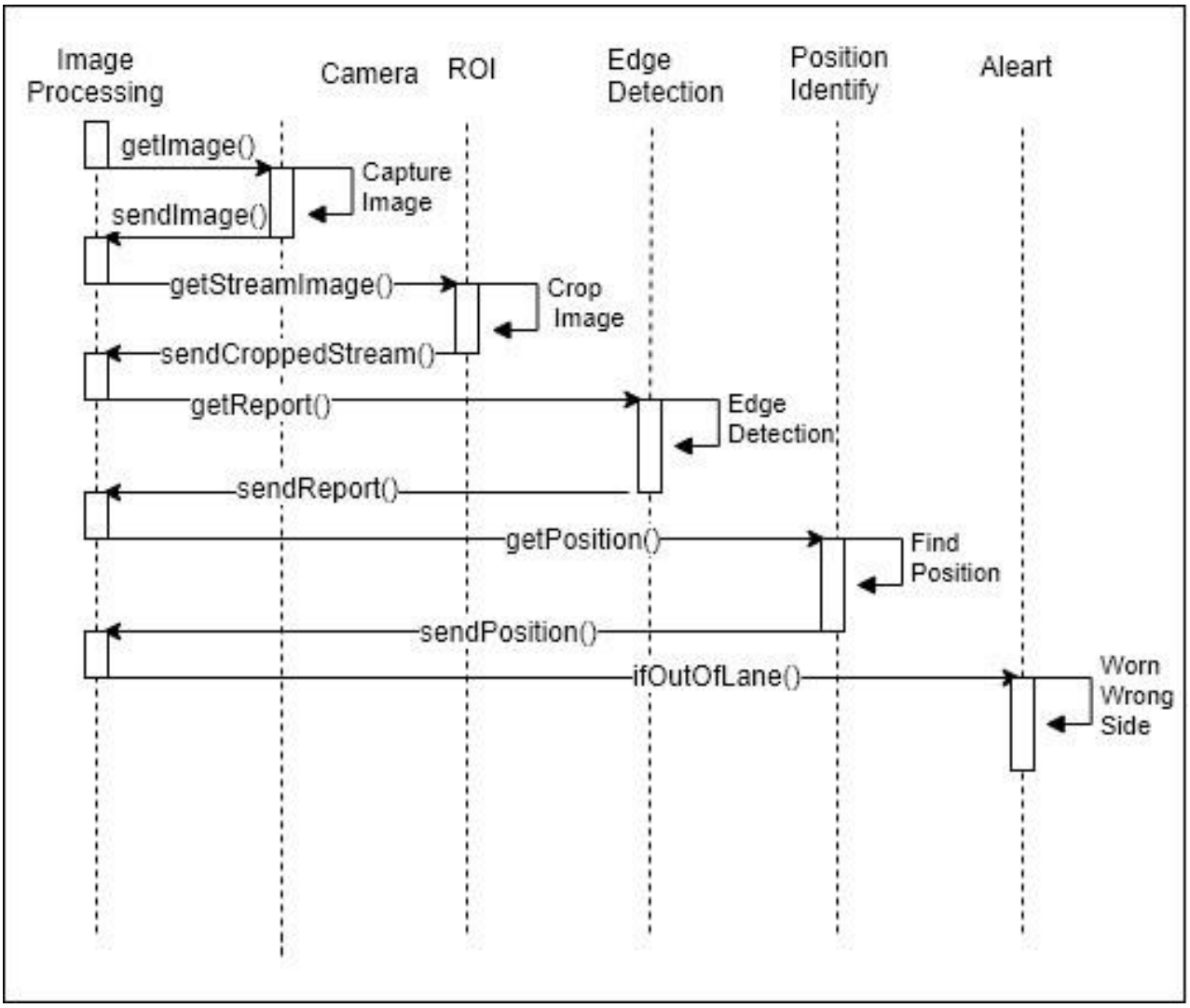


Figure 4.4: Sequence Diagram

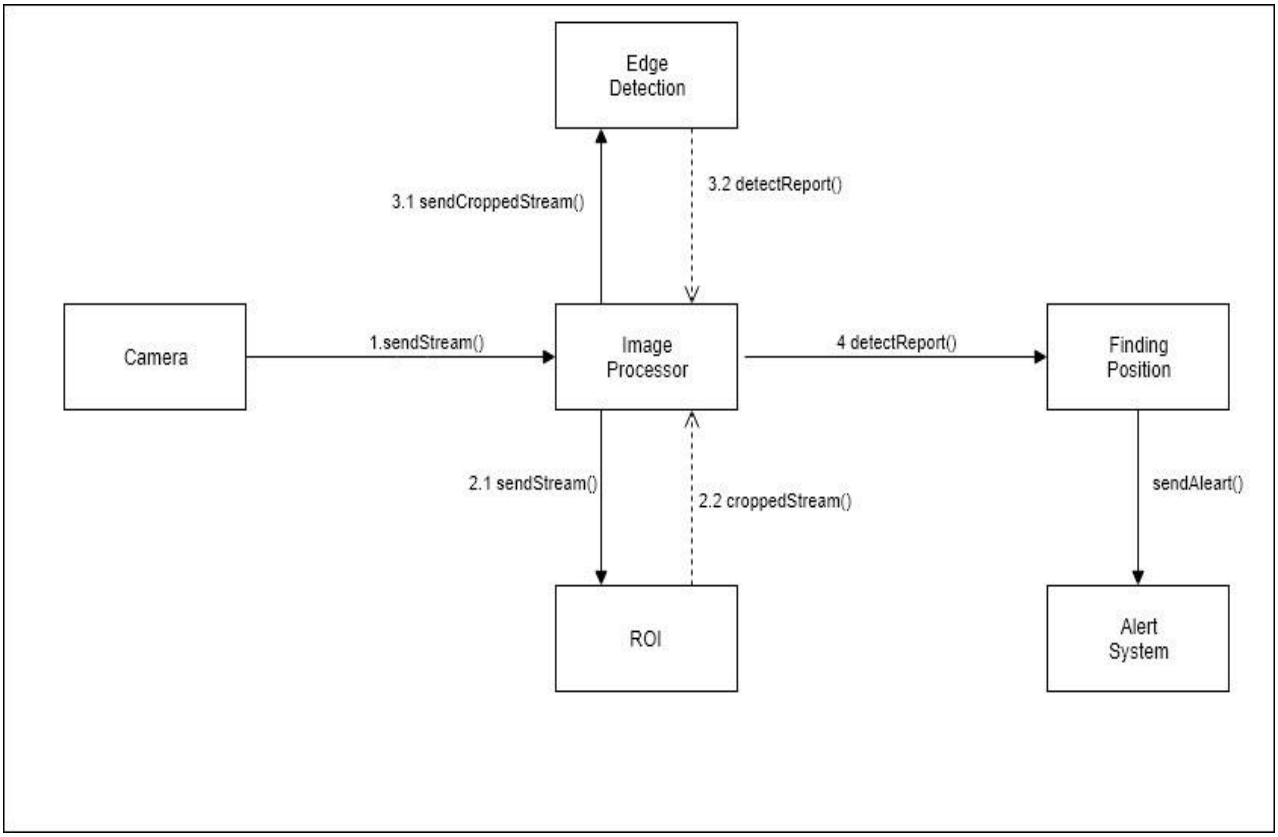


Figure 4.5: Component Diagram

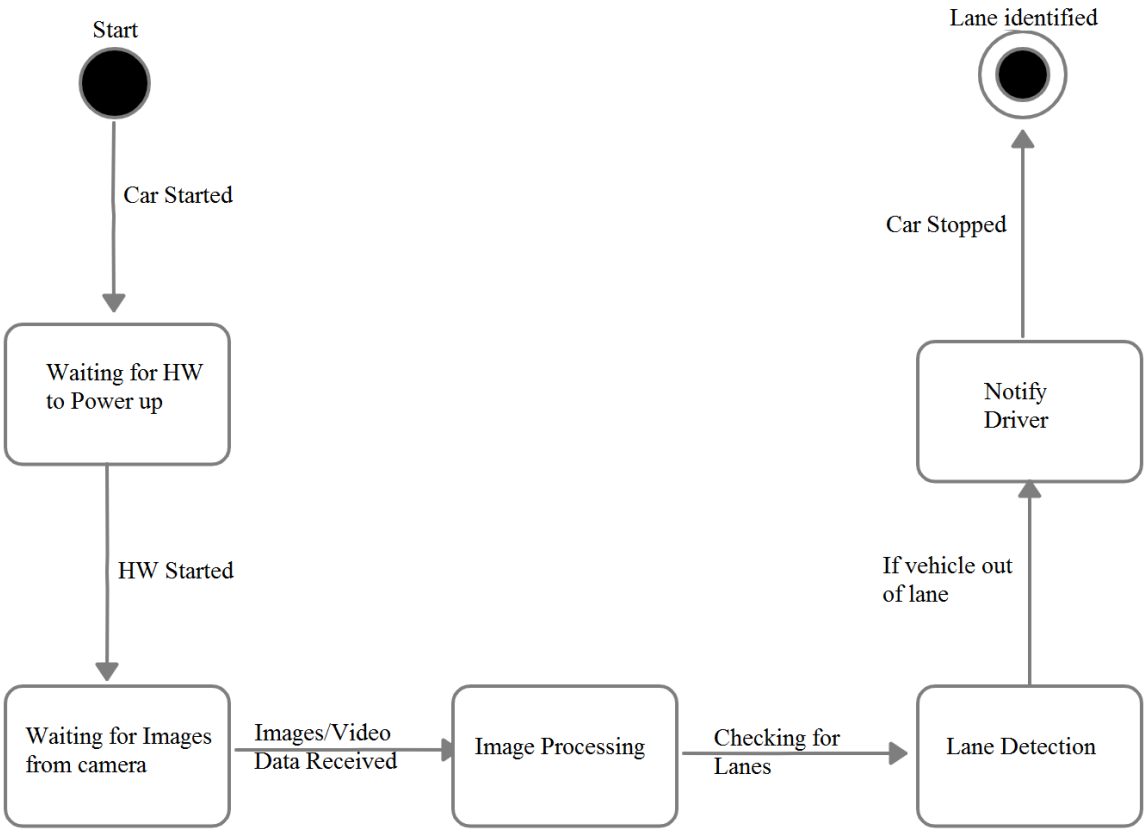


Figure 4.6: State Diagram

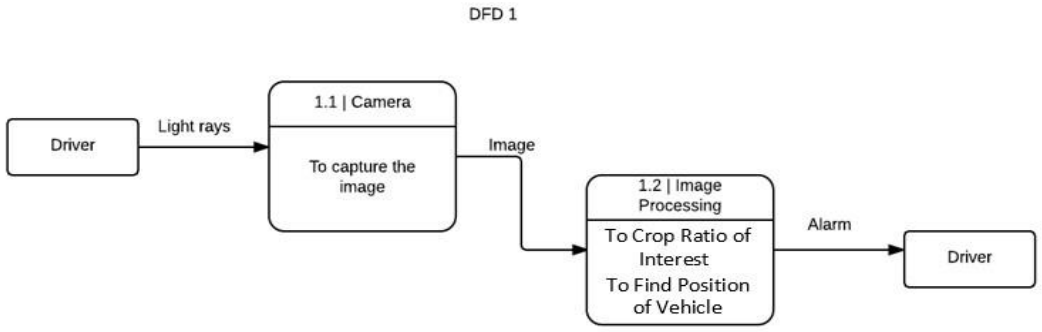


Figure 4.7: Data Flow Diagram (DFD 1)

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

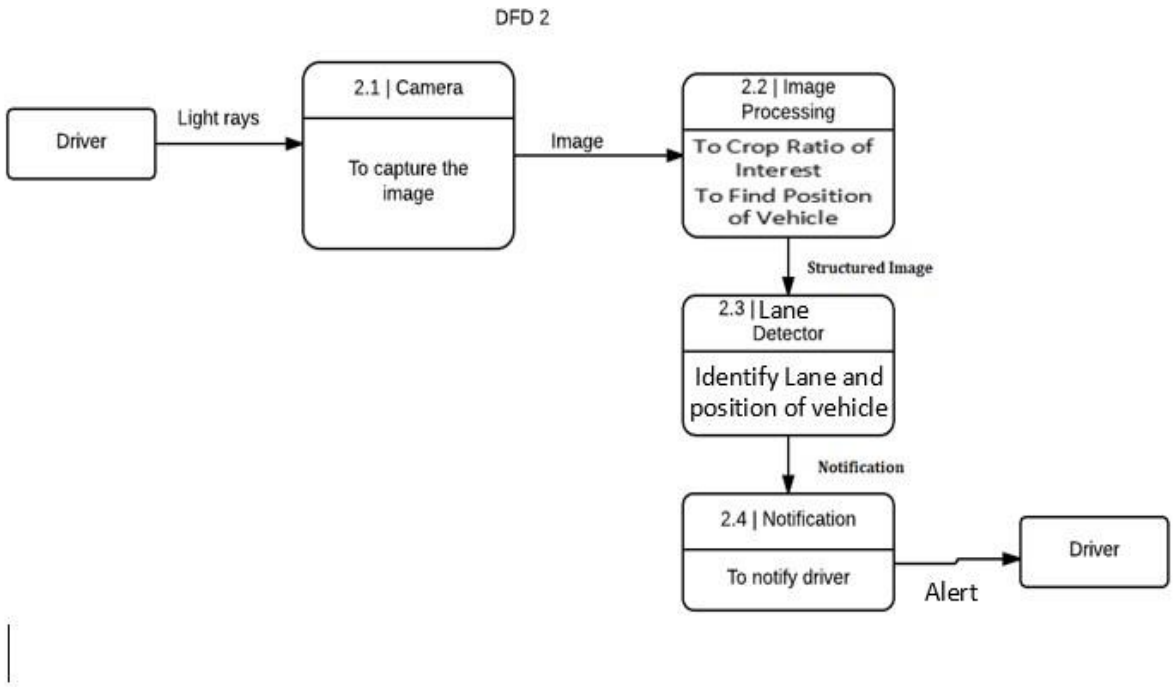


Figure 4.8: Data Flow Diagram (DFD 2)

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.

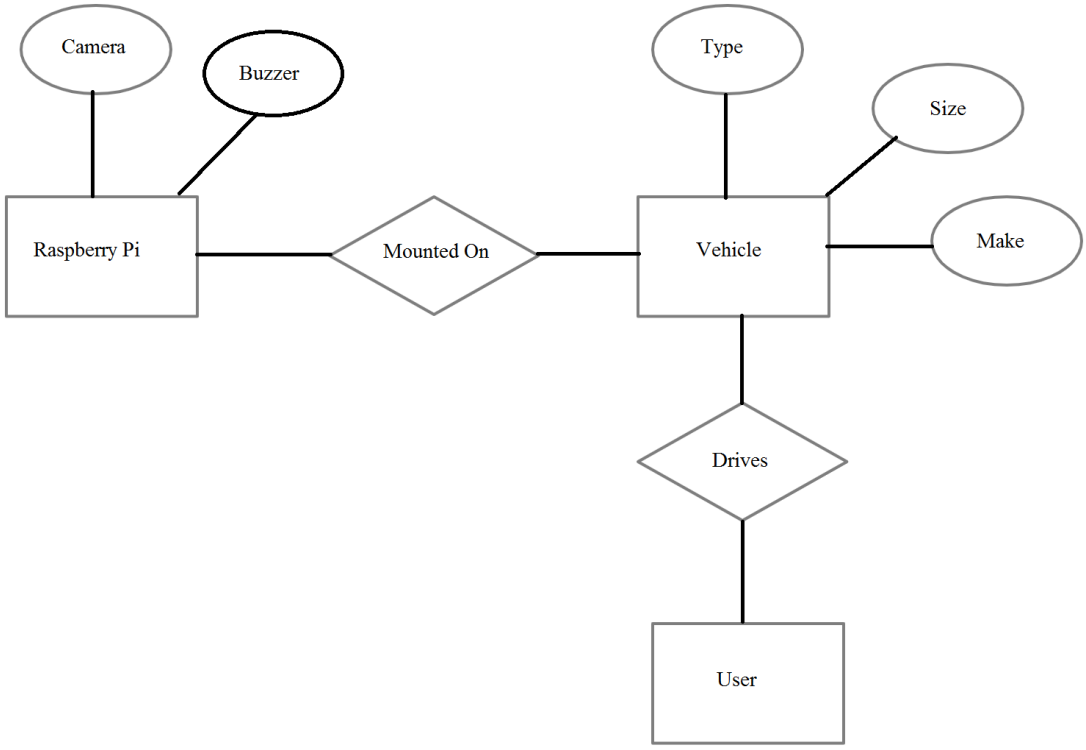


Figure 4.9: ER diagram

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.

A class diagram in the Unified Modeling Language (UML) is a type of static structural diagram utilized in software engineering. It visualizes the classes, their attributes, operations (or methods), and the relationships and interactions between objects within a system. The purpose of a class diagram is to describe the structure of the system being modeled.

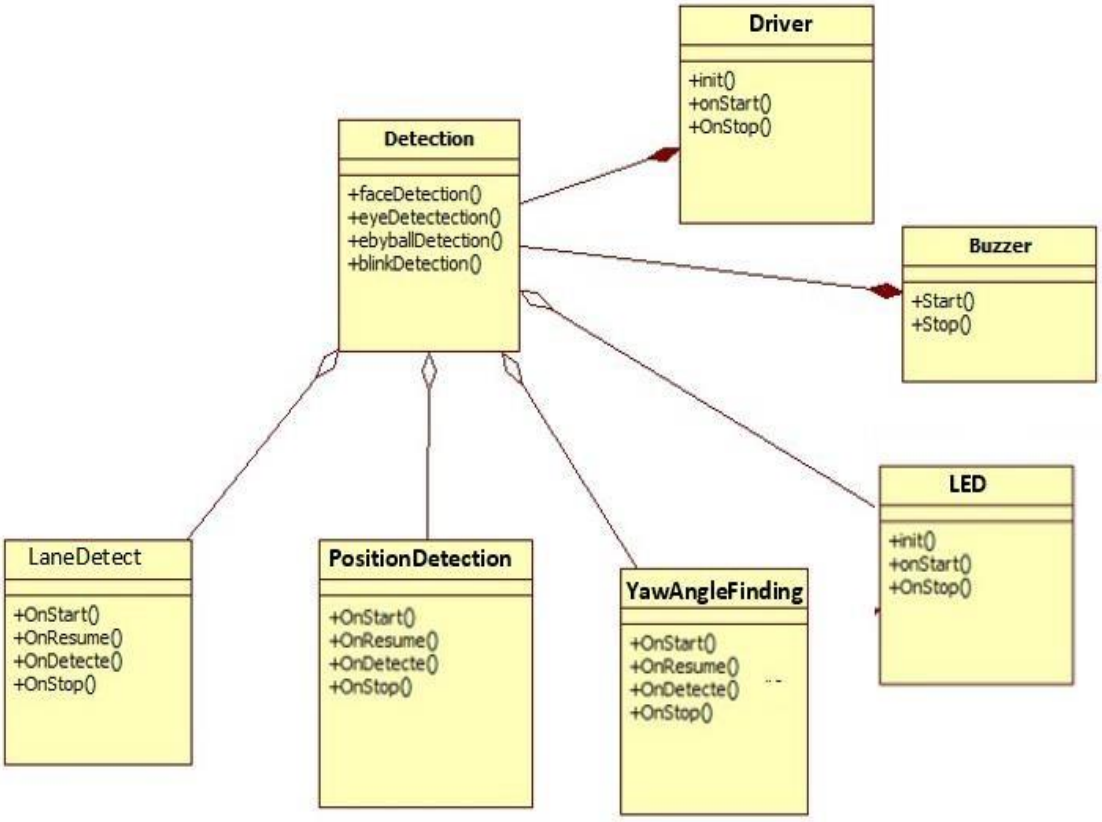


Figure 4.10: Class Diagram

CHAPTER 5

PROJECT PLAN

5.1 PROJECT ESTIMATES

5.1.1 Reconciled Estimates

5.1.1.1 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 codes required to estimate to be **6.2** KLOC.

5.1.1.2 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 require 3.43 months.

   D= 5.43 months

5.1.2 Project Resources

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

5.2 RISK MANAGEMENT W.R.T. NP HARD ANALYSIS

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

5.2.1 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

5.2.2 Risk Analysis

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

Table 5.2: Risk Table

ID	Risk Definition	Possibility	Impact		
			Agenda	Calibre	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 5.3: Risk Probability Descriptions

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

Impact	Value	Definition
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 5.4: Risk Impact definitions

5.2.3 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 5.5: 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 need to be developed.

5.3 PROJECT AGENDA

5.3.1 Project task set

Major Tasks in the Project stages are:

Task 1: Communication

Task 2: Planning

Task 3: Risk Management

Task 4: Modelling

5.3.2 Task network

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

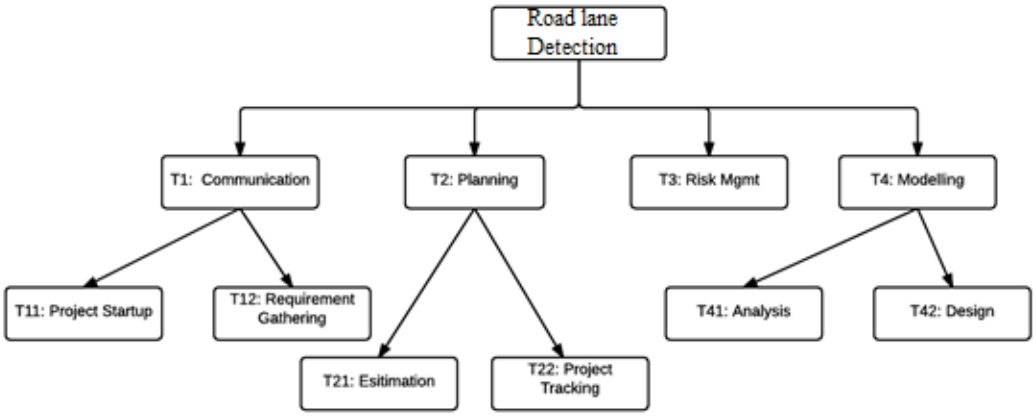


Fig 5.6: Task Network

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.

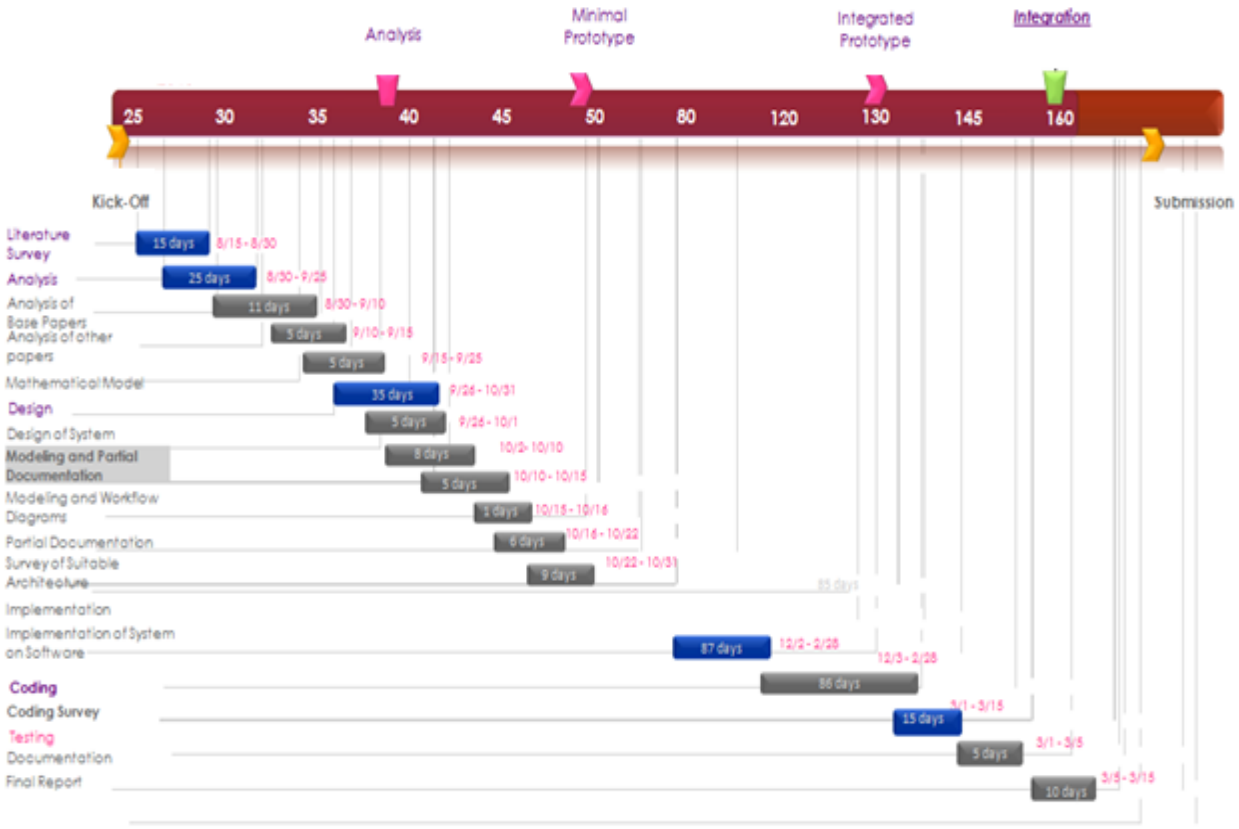
T-C: Risk Management

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.

5.3.3 Timeline Chart



5.4 Team Association

5.4.1 Team Structure

An efficient project team structure is one of the key factors influencing the success of a project. Without a productive and well-organized team, the chances of project failure increase significantly, as the team may struggle to complete tasks efficiently. The team members will face challenges in fulfilling various roles and responsibilities, both as individuals and as a group, without proper collaboration and organization. Therefore, it is crucial to establish an effective project team through team building activities before embarking on a new project.

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

The typical work for a mortal driver is to review the project modules related to a lane road. For the car to stay in the lane's conditions, it needs to be kept there. For an independent vehicle to complete, this is also a seriously important task. And with only a few easy computer vision techniques, lane detection is actually quite straightforward. In this study, "Python and OpenCV" will be utilised to describe a straightforward channel that can be used for lane detection.

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 6.1 Basic 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 6.2 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 and 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.

.

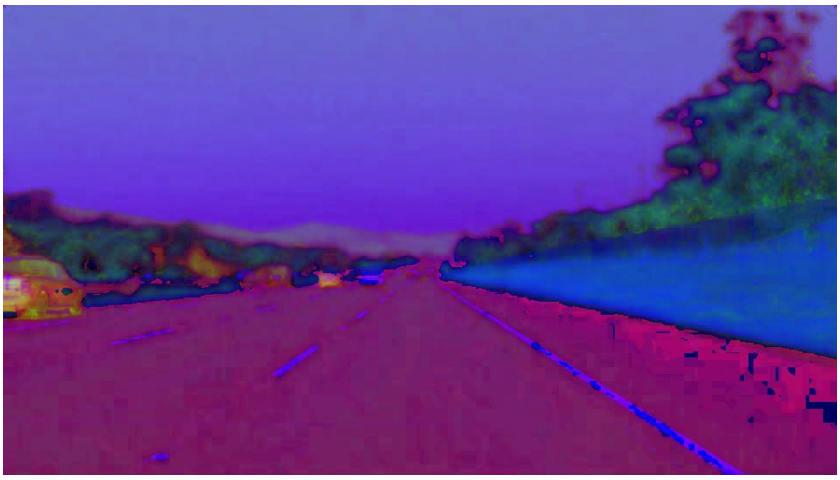


Fig 6.3 HLS Colour Images

Choosing 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 masks 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.

6.3.4 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 includes-

Changing Intensity grade of the Image:

- Non-maximum repression
- Hysteresis Thresholding

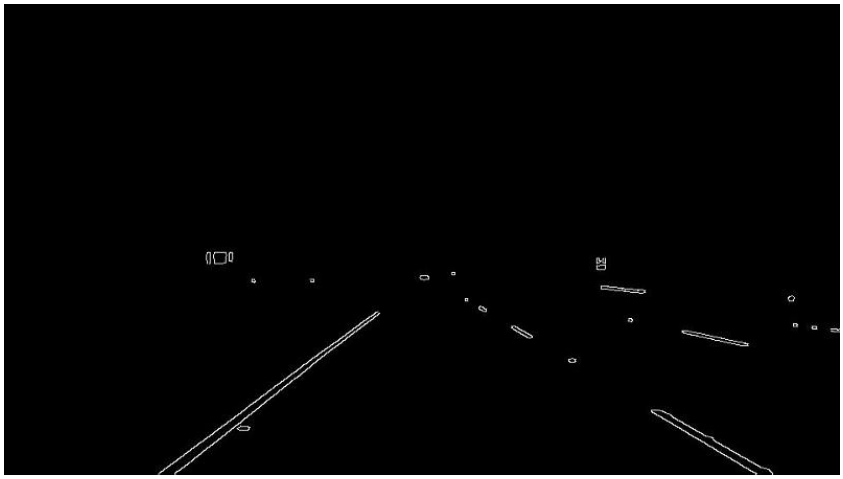


Fig 6.4: Canny Edge

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.

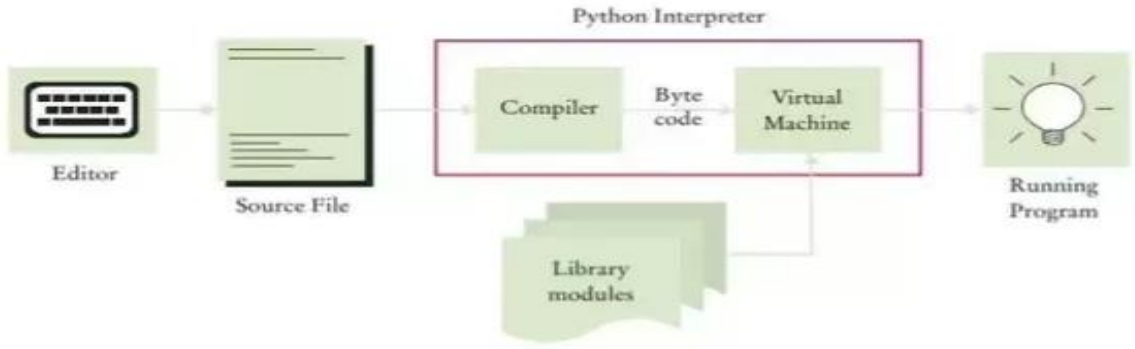
Fig 6.5 Hough Transform

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. The late 1980s saw the creation of Python, a widely used computer language that retains Monty Python's name. It is utilised by hundreds of people to run Instagram, evaluate Intel microprocessors, and develop video games using the PyGame package. It is compact, shares many characteristics with English, and has a considerable number of outside libraries. The language Python is dynamic, interpreted, and bytecode-collected. The source text lacks any declarations for variables, parameters, functions, or style types. As a result, the original statute's collect-time type checking is lost, and the law is made more concise and adaptable. Python constantly checks the values' types and alerts users when any code does not make sense.

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 widely used programming language that was developed in the late 1980s and is named after Monty Python. It is used by hundreds of people to power Instagram, test microchips at Intel, and build video games using the PyGame package. It is compact, really resembles English in many ways, and has a large number of third-party libraries. Python is a dynamic, bytecode-collected, interpreted language. Variable, parameter, function, and style type declarations are absent from source legislation. As a result, you lose the collect-time type checking of the original legislation and the law becomes brief and flexible. Python keeps track of the types of all valuations in real time and alerts any code that does not make sense.



.

Fig 6.6 Python Interpreter

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. It lets the programmer to write code in fewer lines without sacrificing readability to express ideas.

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. Due to the fact that the factual C law is operating in the background, this gives us two benefits: first, the law is just as quick as the original C/C law, and second, Python is easier to decode than C/C. The original OpenCV offender has been wrapped in Python by OpenCV- Python.

NumPy, a heavily optimised library for numerical operations with a MATLAB-like syntax, is used by OpenCV-Python. Each and every OpenCV array structure is converted into and out of a NumPy array. Additionally, this makes it simpler to interact with other NumPy-using libraries, such 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 fulfils 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 a method of testing that ignores the internal workings of the system and focuses only on the results of any input and system execution. occasionally known as functional testing.

Whitebox Testing:

"White box testing" is a testing method that considers a system's internal activities. It goes by the terms structural testing and glass box testing, among others.

In contrast to white box testing, which is often used for verification, black box testing is typically used for validation.

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:

A lot of sections are put together during integration testing to create an output. Integration testing looks at how software and hardware interact if there is a connection between them. Black box testing as well as white box testing may be used.

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. In an entirely operational environment, system testing is performed. It is included in the category 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 is categorized as black box testing.

Acceptance Testing:

Clients frequently do acceptance testing to ensure that the given product meets their standards and performs as they anticipated.

Regression Testing:

Regression analysis entails evaluating a system, element, or collection of interconnected units after an alteration has been made to make sure the modification was effective and didn't harm or force other modules to generate unexpected outcomes. It is included in the category 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 and Test Results

Table 7.2 Test cases

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	The system will activate the camera and take the picture.
Action Result	The camera will fire up and take the picture.
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	The quality of the picture will be checked by the system.
Action Result	The system is going to inspect in order to make sure that the quality of the picture is 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

CHAPTER 8
RESULTS

Outcomes

The implementation of Road Lane Detection System is very useful and effective. In this system the road lane analysis is done where we get a voice message as lane changed and it is understood by the driver and it avoids the road accidents in large amount.it works mostly on highways where two road lines are mandatory for detection. Good image quality helps in getting better results.

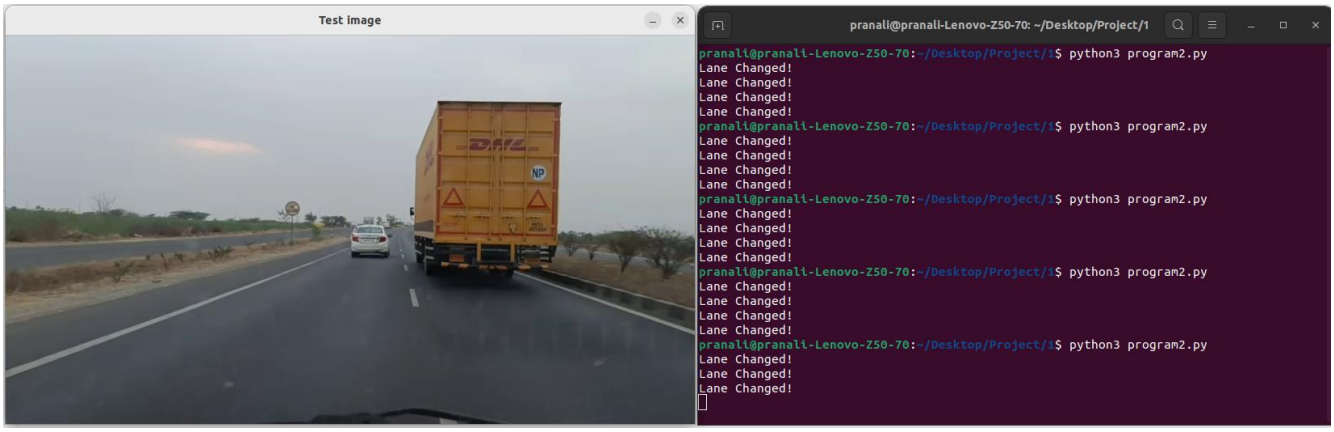


Fig 8.1 Output image one

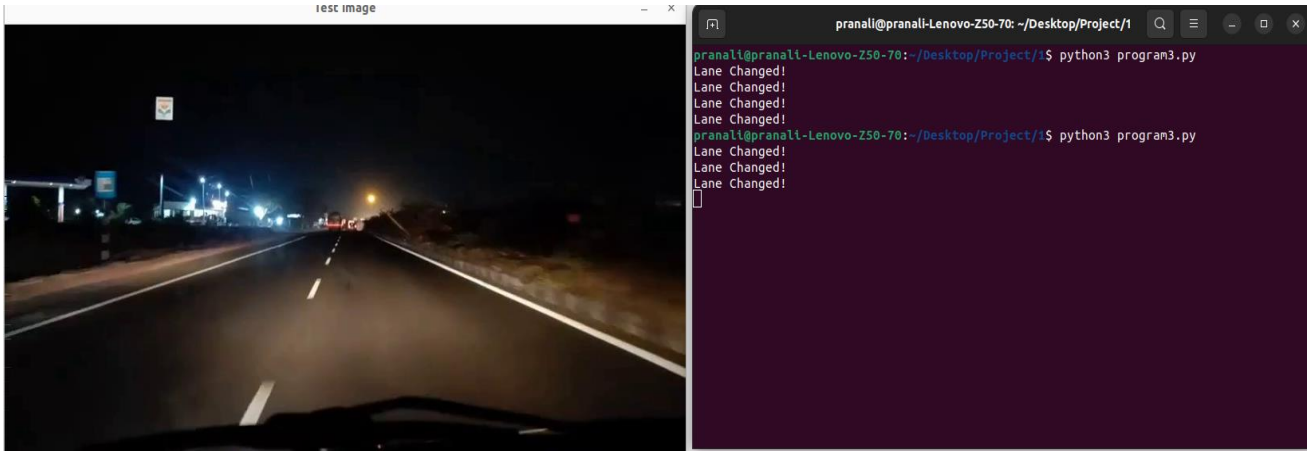


Fig 8.2 Output image two

Crossing Detection, Road Sign Detection, and 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.

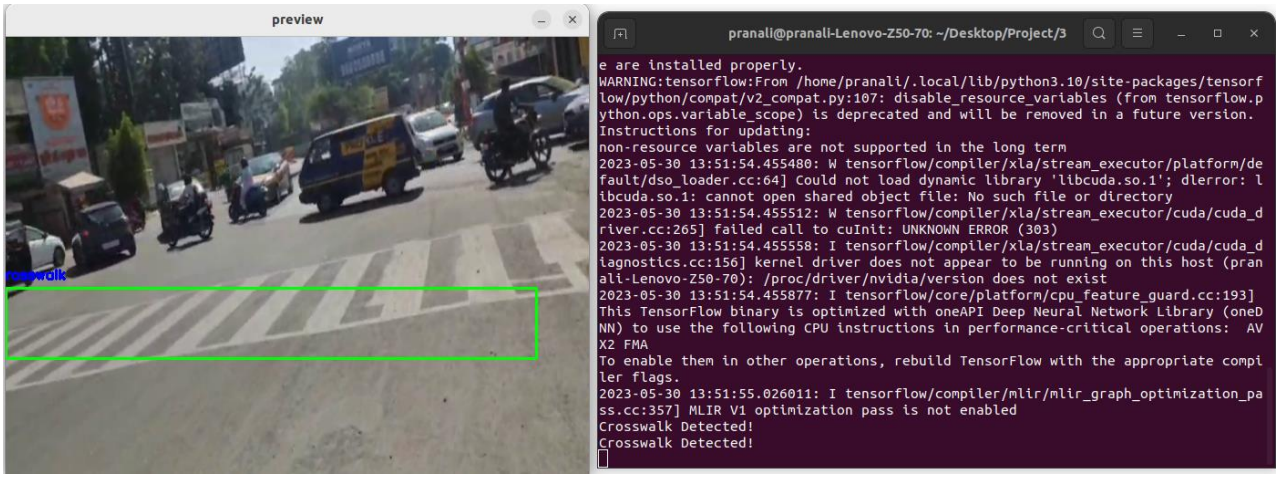


Fig 8.3 Output image three

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.