International Journal of Current Research in Multidisciplinary (IJCRM) *ISSN: 2456-0979*
Vol. 8, No. 3, (March'23), pp. 09-36

TRAFFIC LIGHT TIMER ADJUSTMENT SYSTEM USING THRESHOLD EDGE DETECTION FOR IMAGE PROCESSING: A SOFTWARE ENGINEERING APPROACH

Ftoon Kedwan, PhD,
Assistant Professor, Software Engineering Department, University of Prince Mugrin, Medina, KSA

Abstract: *In more enhanced traffic light control, the road traffic in different directions is monitored by sensors and signals. The proposed method presents traffic light control that adapts to traffic density. There are huge number of traffic signaling variables and, therefore, the need for large computing efforts. Thus, the use of electronic sensors is not efficient because it has a high tendency to be affected by noise and it is expensive. The objective of this research project is to develop an integrated traffic light control system based on image processing to calculate the density of the vehicle on the road and set the timer based on the density. Hence, calculating time allocation for each lane based on density. The proposed solution develops a system for controlling the traffic light by image processing using the Threshold Edge Detection (TED) image matching method to identify the density of the traffic. The system will detect vehicles through images, which can then be used for calculating the density of vehicles on road. Cameras mounted on traffic lights will capture image sequences, which are analyzed using digital image processing to detect vehicles. According to road traffic conditions, traffic light timer is adjusted and controlled. Pixel differences are determined (via Pixel-based matching) between the reference image and real time image which is then converted into percentages and based on these percentage difference, different timings can be allotted for controlling the traffic. After TED procedure, both reference and real time images are matched, and traffic lights can be controlled based on difference of percentage in matching.TED Technique is used because it gives better results and higher accuracy as compared to other edge detection techniques when used for detecting vehicles density. It is less sensitive to noise, overcomes the streaking problem, better localization, and accurate edges detection compared to others. Besides, it is cost effective and less prone to error. The proposed solution has a direct effect on reducing the traffic congestion and avoiding the time being wasted by a green light on an empty road. It also helps in directing traffic on different routes without excessive congestion. Hence, provides travel time improvement leading to better driving experience for the community.*

Keywords: *Traffic Light Control; Threshold Edge Detection; Image Processing*

INTRODUCTION

One can hardly find a place where transportation via the means of cars is not present. Driving cars has become the main method of transportation, which lead to an exponential increase in the use of fuel having a direct relationship with traffic congestion and air pollution [1]. The current traffic control systems are not solving this traffic congestion build up. Therefore, this research paper employs image processing as the best way to determine the density of traffic. Targeted audience could be government Agencies or Companies working on Autonomous cars (i.e., Tesla), or just people owning cars.

1.1 Background

Traffic is a worldwide problem because of the incremental increase in the number of vehicles, particularly in large urban areas [2]. As the traffic congestion and the occurrence of road accidents increase, an advanced technology and equipment are very much needed to improve the traffic control systems. The simplest way of controlling a traffic light is using a timer for each phase. Vehicles could also be detected using electronic sensors which can produce a cycling signal as well. However, heavy traffic flow needs to be exceptionally managed more efficiently to make the best use of the current road capacity in a real time fashion and minimize traffic congestion.

Road accident is another main problem in the modern world. If we analyze the causes of road accidents, they are mainly due to narrow roads and rapid speed [3]. Traffic rules known by experienced drivers and traffic laws set by the government to regulate driving cars behaviors, road informative signs and traffic control systems are used to solve the previously mentioned traffic problems.

Image Processing is the analysis of photographs or frames of videos [4]. The output of image processing is either a picture or a category of attributes or parameters associated with the image. In most of the image processing techniques, images are treated as a two-dimensional signal and applies standard signal-processing techniques to that. Image processing generally specifies to digital image processing, however optical and

analog image processing are possible. Image Processing technique to strengthen raw images received from cameras/sensors placed on spy satellite, aircrafts or pictures taken in normal routine life for various applications is done using image processing. Image processing involves issues associated with image representation, compression techniques and various complex operations, which may be meted out on the image data. Image processing has various operations for image enhancement operations like sharpening, blurring, brightening, edge enhancement, etc. [4].

1.1.1 Traffic Light Systems

A traffic light system is an electronic device that assigns right of way at an intersection or street crossing by means of displaying the standard red, yellow, and green colored indications. This work aims at developing an intelligent traffic light control structure as an optimal solution for road traffic congestion and a smoother transportation channel.

The simplest way of controlling road intersections is the righthand rule or the roundabout with heavier traffic. A traffic policeman can also fill the purpose. However, there are situations where none of those ways can suffice, such as the existence of multilane roads at intersections, or the presence of railroad tracks. In such cases, a more advanced traffic light system is necessary. Yet, the most common way to handle such complex intersections is the conventional cyclic lights control [5]. In more enhanced and adaptive control systems, the road traffic in different directions is monitored by sensors and the signals obtain control on the traffic lights in an adaptive manner. However, the problem in this case is the huge number of traffic signaling variables and the need for large computing efforts.

The different mechanisms used for controlling traffic are:

1. *Manual Controlling:* It requires more manpower for the manual traffic controlling system. The traffic policers will have things like signboard, sign light and whistle to control traffic. In bigger cities, a better solution to control the traffic is needed.
2. *Automatic Controlling:* Automatic traffic light is controlled by timers and electrical sensors via a Traffic Control System [5]. In the traditional traffic light system, each phase has a constant numerical value loaded in the timer, which automatically switches the light ON or OFF upon a timer value change. This method is very prone to traffic congestion. Hence, instead of a defined timer, the density of the vehicles could be computed to set the timer. Electrical sensors could also be embedded in the pavement to detect vehicles and set a timer signal [6]. However, the latter method is expensive and has a high tendency to be affected by noise. Image processing is used to overcome all of these shortcomings.

1.1.2 Benefit of Traffic Light Controller

When traffic control signals are properly developed, implemented, and used, they shall provide the following advantages [7]:

- Organize and controls traffic flow on the road.
- Coordinated for continuous movement and minimum competition.
- Provide driver confidence by assigning right of way.

Traffic control signals are used to solve all traffic issues at intersections. Hence, they have been installed at many locations where they are not needed. Therefore, if the traffic control system is not properly designed, placed, operated, or maintained, it can have an adverse effect on the safety and efficiency of vehicular, bicycle, and pedestrian traffic [7].

1.1.3Types of Traffic Signals Components and Operations

The main components of traffic signals are:

- Main display with red, yellow, and green lights.
- Vehicle detection system and traffic signal controller placed in the traffic signal cabinet.
- Inductive loops or sensors.

Traffic signals can operate in either of the following three modes [5]:

1. *Fixed-time mode*: there are no detections for any approach. The signal cycling doesn't depend on the actual traffic demand situation.
2. *Semi-Actuated mode*: only a minor cross street traffic is detected. When vehicles are detected on a minor street, the major street green light is turned to red for the minor street traffic to merge into the

intersection.

3. *Actuated mode:* all approaches are detected. The traffic signal turns green only in the presence of vehicles.

1.1.4 Traffic Light System Using Image Processing

Image processing has proved to be the best way of controlling the traffic light signal change as it depends on analyzing an actual image of the road traffic in real time [8]. This seems to be more realistic compared to earlier means of vehicles presence detection using electronic sensors implanted in the pavement to detect of the vehicles' metal content. Image processing-based methods proved to solve traffic congestion issues by avoiding a situation of a green light on an empty road.

There is a great advantage using cameras beside de sensors as the camera-controlled traffic monitoring system gives us more information about the vehicles. For the detection purpose first, we have to define the object of interest (OOI). After identifying the OOI, itis isolated from the background for further image processing.

1.2 Problem Statement and Description

1.2.1 Problem Statement

The aim of our project is to determine the density of vehicles on road using image processing techniques and thereby controlling the switching of traffic lights.

1.2.2 Problem Description

Many researchers have proposed initial solutions for bettering the transportation system [8-12]. In order to control the flow of traffic on the road, fixed timer was used. Another method uses wireless sensor networks for the control of traffic, but these sensors can only detect the presence of vehicles. In the current research project, we propose a system for controlling the traffic light by image processing. The system will detect vehicles through images, which can then be used for calculating the density of vehicles on road. The image sequences are captured via a camera installed on the traffic light. Digital image processing will be used to analyze these image sequences to detect vehicles. Then, the traffic light signal can change according to traffic conditions on the road.

Traditional traffic light systems encounter many limitations, especially when traffic signaltimingisnotbasedonthenumberofvehicles.Thisleadstothefollowingissues [5]:

- *Heavy traffic jams*: With the increasing number of vehicles on the road, heavy traffic congestion occurs substantially in major cities. This usually happens at the main junctions, in the morning before office/school hours and in the evening after office/school hours. This causes people spending lots of time in traffic unproductively.
- *Green Light for an empty road*: Sometimes green light is on for a lane with no vehicles and in other lanes vehicles are queued and waiting for passage.
- N*o traffic, but the pedestrians still need to wait at certain junctions*. Sometimes, even if there is no traffic, pedestrians have to wait. Because the traffic light remains green for the present time period, the road users should wait until the light turns red.

1.3 Objectives

1.3.1 General Objectives

The general objective of this research project is to develop an integrated traffic light control system based on image processing to calculate the density of the vehicle on the road and set the timer based on the density.

1.3.2 Specific Objectives

This research project has some specific objectives as follows:

1. To capture an Image
2. RGB to gray scale conversion
3. Image Resizing
4. Image Enhancement Process
5. Edge Detection
6. Image matching and density calculation

7. Calculating time allocation for each lane based on density.

1.4 Organization of the Report

This research paper is organized as follows. Chapter 2 presents the literature survey reviews highlighting some important papers. Chapter 3 presents the system requirements specification (software and hardware) that are used in this research project. Chapter 4 presents the system design considerations, architecture of the proposed system and explains about the detail functionalities and description of each step of the flow diagram. Chapter 5 q1§explains the implementation, programming language selection and also coding lines for programming language used in the project. It also presents the pseudo code for main module and sub module along with Results. Chapter 6 explains the Advantages of the proposed system and also explains its various functional areas and applications. Chapter 7 contains the conclusion and the further future enhancements of the project.

REVIEW OF LITERATURE SURVEY

This section provides a direction in the research area and sets a goal for analysis, thus giving a problem statement. The following are some of the related research papers.

2.1 Implementation of Image Processing in Real Time Traffic Light Control

In this method [8], the desired output is obtained by distinguishing the presence and absence of vehicles on the road and signaling the traffic light to go red if the road is empty and signaling the traffic light to go green if vehicles are present on the road. The duration of green light is based on the number of vehicles on the road. Initially, a web camera handles the image acquisition. Obtained RGB image is converted to grayscale image and that image is enhanced using gamma correction [13]. Some of the fundamental functions which are used frequently are discussed here like power law transformations (gamma correction) and piecewise linear transformation functions [14]. After that, Prewitt edge detection method [15] - a gradient-based edge detection process - is used. Gradient operators require two masks to obtain the X and Y direction gradients, which are then combined to obtain a vector quantity. The magnitude of the vector quantity represents the strength of the edge gradient at a point in the image. The researchers inferred that image processing is a better technique to control a traffic light state change for its consistency in detecting vehicles on empty roads; since it uses actual traffic images.

2.2 Intelligent Traffic Signaling System

In this paper [9], a comparison has been given between traditional and modern traffic management methods such as manual controlling with the help of manpower and automatic controlling methods by using electrical sensors, magnetic loop detectors, inductive loop detectors, and light beams. The comparison has been discussed as to how the image processing methods is much more effective when compared to those traditional methods. Cameras are used for traffic density extraction, where image of foreground and background are acquired and processed frame by frame. Some preprocessing steps are employed to reduce camera noise. Then, Sobel edge detection, background subtraction, morphological image closing operation, and flood filling operation are performed. Finally, the image will be converted from Grayscale to Binary image. After all these steps, the estimation of signaling time for each road is carried out and the green light is turned on where there is more traffic. Advantages of this approach include a) The total amount of traffic on the road is estimated by measuring the total area occupied by vehicles on the road instead of vehicle count in terms of traffic density, b) This model could be extended to incorporate a large number of interconnected traffic junctions and using their traffic density to adjust adjacent junction's time allocation.

2.3 Real-time Area-Based Traffic Density Estimation by Image Processing for Traffic Signal Control System

In this paper [10], the complete process of getting output is carried out in 3 phases in which each phase has many steps to be followed. The first one is the analysis of empty roads; it consists of seven steps. That is the acquisition of empty road images, image cropping as a reference, RGB to grayscale conversion, removal of unwanted objects, Edge detection using Canny method, dilation of images, and area measurement of reference images. Next, roads with traffic are analyzed using the following steps: acquisition of road images with traffic, image cropping, RGB to grayscale conversion, removal of unwanted objects, edge detection

using Canny method, dilation of images and area measurement of live images. In the last phase, the decision-making process is carried out, which involves comparing the outputs of both the first and second phases which helps to determine the amount of time that should be allotted to the traffic signal to indicate green light. Advantages of this approach include a) a traffic density estimation technique using image processing is proposed based on the area occupied by the edges of vehicles. The proposed method can easily estimate traffic density according to experimental results, b) this area-based traffic density estimation method can be easily implemented in an intelligent traffic control system in a densely populated country.

2.1 Automatic Traffic Using Image Processing

In this system [11], authors discussed that they designed software that can generate the output by taking two types of files, video file and image file. In the detection of vehicles using video footage, first, the foreground objects are detected and enhanced and then the obtained image is processed for edge detection using the Canny edge detection technique. Advantages of this approach include a) it can be used to o control the traffic and count the number of dynamic vehicles that are passing on highways to know the density, b) the detection, tracking, and moving vehicle counting systems can be further improved to accommodate streaming video feeds. However, the limitation of this system is that it may fail in some of the lighting conditions.

2.2 Real-Time Traffic Light Control Using Image Processing

- In this paper [12], Edge Detection is used for every object which lies in the webcam range. This system is divided into 4 subsystems: Image acquisition, RGB to grayscale conversion, Image enhancement, and Image matching using Prewitt edge detection. Initially, image acquisition is done with the help of a camera. The image of the road is captured when there is no traffic on the road. This empty road's image is saved as a reference image. RGB to grayscale conversion is done on the reference image. Edge detection of these real-time images of the road is now done with the help of Prewitt edge detection operator. After edge detection procedure, both reference and real-time images are matched, and traffic lights can be controlled based on the percentage of matching. Advantages of this approach include:
- Magneticloopdetectorsarenotusedbecausetheirinstallationandmaintenance are inconvenient.
- It uses power law transformations for image enhancement.
- Image Processing is used as a better technique to control the traffic light state change.

This work has some limitations including:

- Web camera resolution should be high.
- γ (Gamma) is one of the parameters in the input gray level graph which cannot be used as a fractional value since these attempts show a reverse effect of brightening the image still further, which might be undesirable for the present case.
- Thissystemdoesnotachieve100%accuracy.

Overall, the above works described in the literature survey all achieved good outputs. However, the proposed system in the current research project achieved a higher accuracy than all of them for many reasons, but mainly for the use of threshold edge detection method.

SYSTEM REQUIREMENTS

1.5 Functional Requirements

Functional Requirements are the statements of services the system should provide and how the system reacts to particular inputs and how the system should behave in particular situations [16]. The requirement specifies a function that a system or component must be able to perform. These include inputs, outputs, calculations, external interfaces, communications, and special management information needs. A specification constrains the way a given task should be performed, and how the results are be obtained(speed, accuracy, etc.) [16]. It also constrains the elements of the functional entities involved (initiator, source, receptor, etc.). For the functional requirements, the following criteria should be met:

- High resolution camera should be used to detect the traffic density and traffic signal from a car, and to make sure vehicle acts as per the signal from a quite far distance.
- Images captured should be noise free.
- Hardware such as raspberry pi and a raspberry pi based camera should be chosen.

- Traffic images are stored in the system to verify with real time images to specify the density of traffic.
- Vehicle should have an ideal mount point for this hardware to capture the traffic signals precisely.
- Every time autonomous driving car detects a traffic pole signal, the camera should extract the exact color of the signal, signal should convey what actions need to be taken.
- Allocating time for each lane based on traffic density.
- Easy flow of traffic.

1.6 Non–Functional Requirements

Non-functional requirements (listed below) are requirements that specifies criteria that can be used to judge the operation of the system and how a system is supposed to be [16], rather than specific behaviors.

1. **Timeliness:** The system is expected to have reasonable short time response. As the camera is continuously recording the video in traffic signal, traffic lights should be changed on time without any delays.
2. **Reliability:** The system should be reliable. The output must be accurate at least for most of the times. Reliability is defined as a degree to which the result of a measurement, calculation, or specification can be relied on to be accurate [17].
3. **Availability:** Camera images must be always available for processing on time for time calculation.
4. **Performance:** The performance of the functions and every module must be high and satisfactory.

1.7 User Requirements

- Image needs to be captured using a camera interface.
- Traffic density needs to be measured.
- Traffic needs to be efficiently managed at junctions.
- Traffic monitoring system should achieve efficiency.
- Data should be stored in a centralized server.
- Driverless (autonomous) car should determine traffic signal.

1.8 Software Requirements

- Stored Images of the Traffic shall be converted from RGB to Gray Scale for ease of processing.
- Gray Scale Images shall be resized to a standard size since every camera has its resolution.
- Threshold Edge Detection techniques shall be applied on the Gray Scale Images.
- Check whether there are gaps in the image obtained from Edge Detection technique on the Gray Scale Image and Flood fill algorithm shall be used to fill the gaps in the Image.
- Access the images from the Camera, and RGB images shall be converted to Gray Scale Images.
- Gray Scale Images shall be resized to a standard size since every camera has its resolution.
- Edge based matching shall be carried on between the prestored applied with Threshold Edge detection and the Image we obtained from camera applied with prestored Grayscale Image.
- Reference Image and Real Time Image are matched Pixel by Pixel. Pixel-based matching shall be carried out.
- Pixel values of the converted real time image and converted reference image are stored in matrix, and both images shall be subjected to pixel matching.
- Counter shall be incremented on every pixel mismatch.
- The difference obtained after pixel matching shall be converted to percentage.
- Percentage shall be calculated by the formula number of pixels mismatch divided by total number of pixels.
- Different timings shall be allocated for controlling traffic lights.
- For traffic light detection, images shall be acquired from the camera.
- Gathered data shall be labelled and annotate the Images.
- To train the model, we shall convert our data into TFRecord format.
- TFRecord Format shall merge both the image file and YAML file required for training as an input.
- For training, ssd_inception_v2_coco and faster_rcnn_resent101_coco shall be used.
- Adjust the num_classes to be 4 and set path to PATH_TO_BE_CONFIGURED for model checkpoint, train and test data files and label Maps.
- The number of region proposals shall be reduced from 300 to 10 for faster RCNN and 100 to 50 for

ssd_inception.
- Other configurations like learning rate, batch size, default settings shall be used.
- Second_Stage_batch_size shall be less than or equal to max_total_detections.
- Label maps shall be created for each Class.
- For training on GPU, we shall need COCO pretrained network models, the TFRecord files, label map file for classes, Image dataset, TensorFlow model API.

1.9 Hardware Requirements
- Processor : Broadcom BCM2837 SoC 64 bit.
- Ram : 1 GB.
- Raspberry pi 4 model.
- Camera : Raspberry pi compatible camera.
- Breadboard, Resistors and Jumper wires.

SYSTEM DESIGN

4.1 System Design
In the below figure, Fig. 1, the block diagram for automatic density-based traffic light control is shown. This block diagram explains how image processing is used to control traffic.

4.1.1 Base Architecture
The base architecture consists of both traffic density detection system and Traffic light control system, which are mounted on Traffic Light poles, in addition to the Traffic Light detection system, which is mounted in a car. Although these two systems are not used together, they are both considered here because they are built the same way. However, the software differs for both systems. In the below Architecture diagram, Fig. 1, both systems are clubbed.

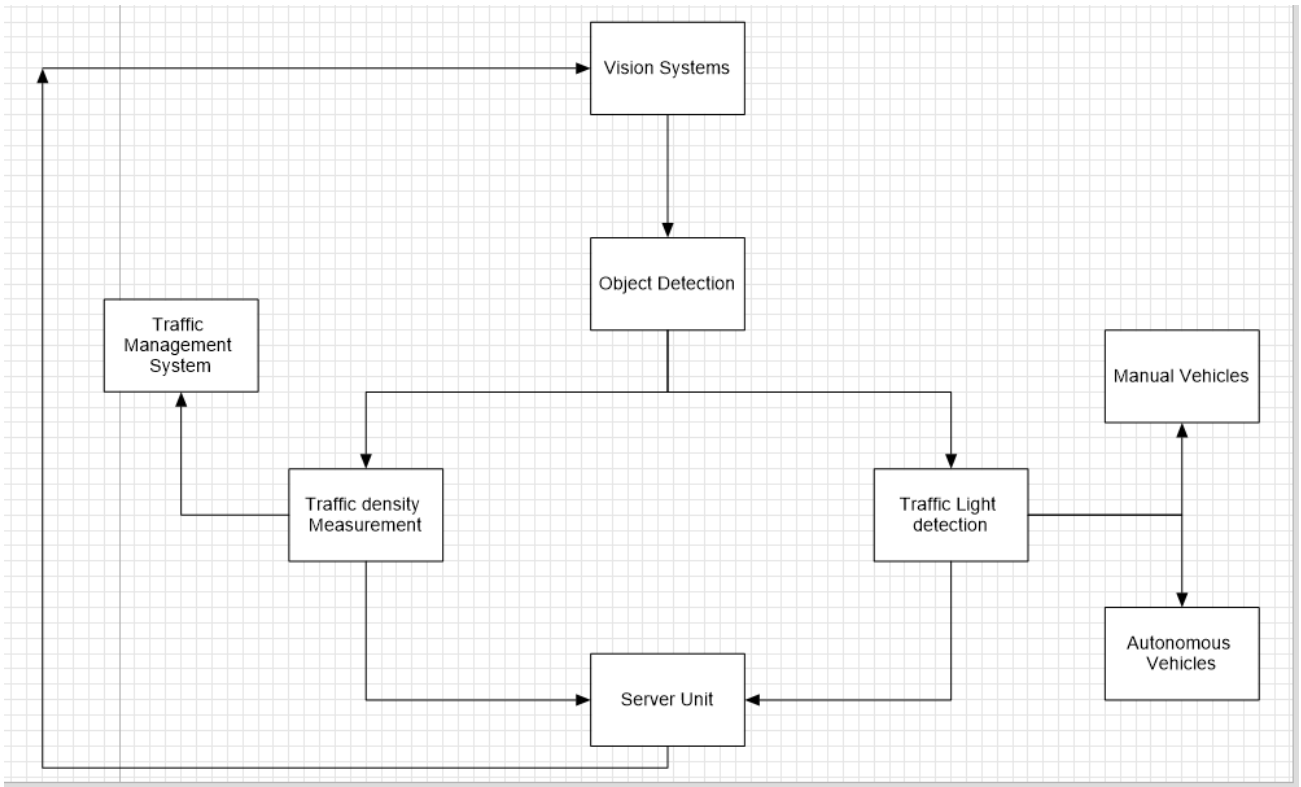

Figure 1 Block Diagram for Automatic Density-Based Traffic Light Control

- Vision system comprises of Camera.
- Object detection is mandatory for both systems.

- Traffic density Measurement measures the density of traffic and operates the traffic lights accordingly and the data is being sent to the server.
- Traffic light detection system helps detect the color of the traffic light, which can be further used in Manual or Autonomous vehicles.
- Light detection data can also be sent to a local server by the Traffic light detection unit present in cars for future data access.

4.1.2 System Architecture for Traffic Density detection

Detailed procedures are implemented in three phases.

Phase1:

- First, an image of an empty road is captured and saved as a reference image.
- The captured RGB image is then converted into a gray scale format.
- The threshold edge detection technique is used for edge detection and Flood filling of the reference image.

Phase2:

- Images of the road with traffic are captured.
- The captured sequence of RGB image is then converted into a gray scale format.
- The threshold edge detection technique is used for edge detection and Flood filling of the real time images.
- Image processing tools are used to calculate the vehicles count on each lane.
- Different traffic signal timings will be assigned according to the vehicles' count.

Phase3:

- After edge detection procedure, both reference and real time images are matched to calculate the density of vehicles. Traffic lights can be controlled based on the density.

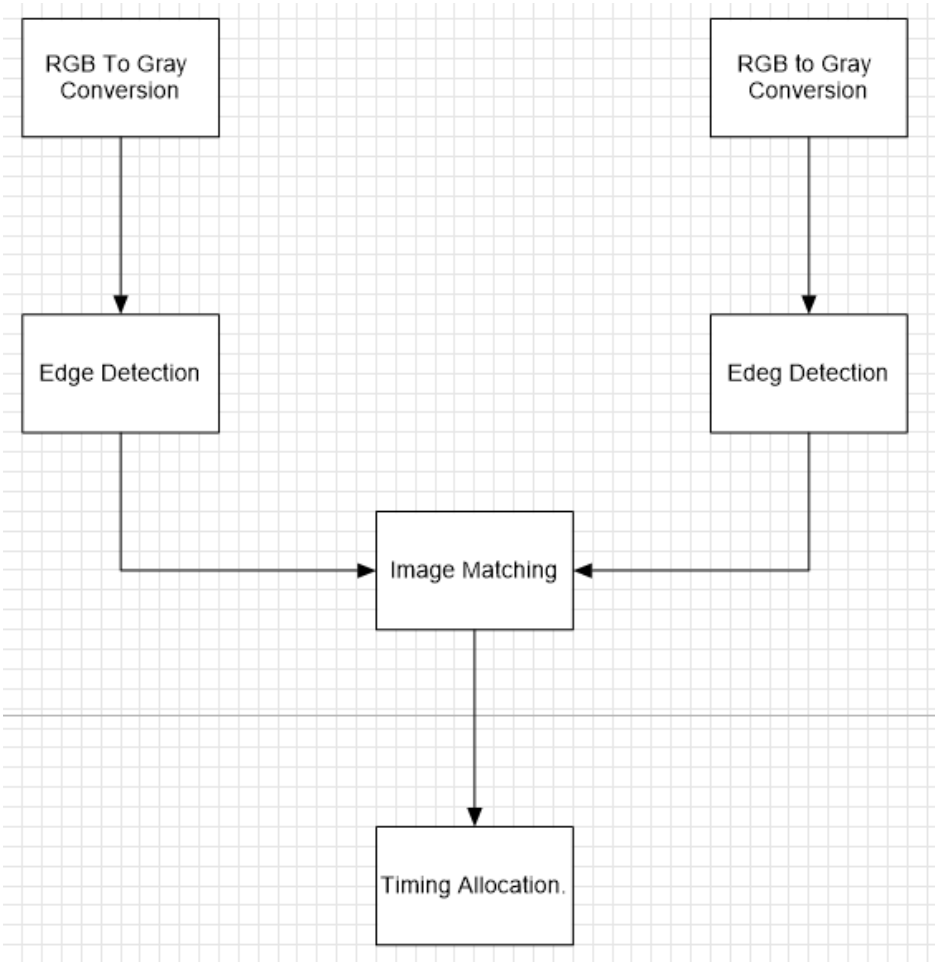

Figure 2 Image Processing Steps

As in Fig. 2 above, the following are the generalized main steps involved in the image processing:

1. Image acquisition
2. RGB to grayscale conversion
3. Edge Detection and flood fill
4. Image matching and time allocation

4.1.2.1 Image Acquisition

In image processing operations, the first step is the image acquisition, usually by digital photography using a digital camera. Then, further image processing can be applied for various different tasks. In the current project, Raspberry Pi 5MP camera is used, which plugs directly into the CSI connector on the Raspberry Pi [18]. For the simulation purpose, sample traffic images from the internet are used.

4.1.2.2 RGB to Grayscale Conversion

After the images are acquired, they are converted to gray scale in order to ease the image processing since less information needs to be provided for each gray scale image pixel, which is intensity information where black has the weakest intensity and white has the strongest. The images are then resized; because every camera has its own specific resolution specification.

4.1.2.3 Edge Detection and Flood fill

Edge detection is used for feature detection and extraction [15]. It is used to identify image points where brightness changes sharply (discontinuous) to indicate objects' boundaries. This is highly important to reduce the amount of processed data and to simplify the subsequent task of information contents

interpretation.
The common use of Flood fill operation is to fill holes in images, particularly the binary or grayscale images [15]. When boundary is of many colors and interior is to be filled with one color, we use this operation.

4.1.2.4 Image matching and time allocation

Edge detection image matching method is used in the current research project to identify the density of the traffic. It follows twosteps.
Step1: Edge Detection - Among the key features of an image, i.e., edges, lines, and points, edge is used in the present work which can be detected from the abrupt change in the gray level. An edge essentially demarks between two distinctly different regions, which means that an edge is the border between two different regions, i.e., the edge of the cars in our case. As such, the pixels of the car objects are located via edge detection methods. Hence, (a) The result is a binary image with the detected edge pixels. (b) Common algorithms used are Sobel Edge Detection Technique, Perwitt Edge Detection, Roberts Edge Detection Technique, Zero cross Threshold Edge Detection Technique and Canny Edge Detection Technique. In the current project, Threshold Edge Detection Technique is used because it gives better result as compared to others with some positive points. In addition, Threshold Edge Detection proved to be less sensitive to noise and adaptive in nature. It also overcomes the streaking problem and provides good localization. It also detects sharper edges as compared to others. Hence, it is considered as an optimal edge detection technique.
Step 2: Image Matching - Edge based matching is when the edges of similar objects are paired together [15].Edge detection of reference and real time images have been compared, evaluated, and matched pixel by pixel using threshold edge detection technique. Pixels are stored and converted in matrices in the memory, and their pixel values are compared and matched to decide if they represent the same object (car). The number of pixel value mismatches are counted and then converted into percentages. Based on the difference percentage, different traffic light timings can be assigned for controlling the traffic. Percentage of matching is expressed using below equation.
%match=No.pixelsmatchedsuccessfully/totalno.ofpixels
After edge detection procedure, both reference and real time images are matched, and traffic lights can be set and controlled based on difference of percentage in matching.

- If the difference is between 0 to 20% - green light is on for 15 seconds.
- If the difference is between 20 to 40% - green light is on for 25 seconds.
- If the difference is above 40% - green light is on for 35 seconds.

Though there are some disadvantages related to pixel-based matching, it is one of the best techniques for decision making.

4.2 Design Issues/limitations

- If removal of noise and detection of objects is not performed correctly, the system is prone to errors and not efficient.
- The weather conditions are not considered which may affect the image quality when it becomes foggy or in heavy rains.
- After 6 PM, if there is a power cut in streetlights then the system will revert back to normal cycle of controlling the traffic signals.

SYSTEM ARCHITECTURE

4.3 System Architecture for Traffic Light detection

5.1.1 Gather the data

- The required data are gathered from the digital camera. These data will be used to train the machine learning model.

5.1.2 Label and Annotate the Images

Next, the acquired images will be manually annotated and a yaml file will be created.

5.1.3 Training the Model

Here, mainly ssd_inception_v2_coco and faster_rcnn_resnet101_coco network models are trained and run on Amazon GPU by constructing a Pipeline with the required parameters. Gathered data are converted into the

TFRecord format to be able to train the model via a TensorFlow model API. TFRecord format combines the gathered images and the yaml file of annotations into one input file for model training. Next, an object detection pipeline is built. To do this, two models are used, ssd_inception_v2_coco and faster_rcnn_resnet101_coco [19-23] which are needed to adjust the number of classes (num_classes) to 4 and also set the path for the model checkpoint. The image data sets for training and testing data, and the label map files are created for each class. The number of region proposals are reduced from 300, to 10 for faster_rcnn and from 100 to 50 for ssd_inception. In terms of other configurations like the learning rate, batch size and etc., default settings are used.

5.2 System Architecture for Traffic Density Detection

The above figures, Fig. 3 and 4, illustrate the detailed system architecture implemented to detect traffic density.

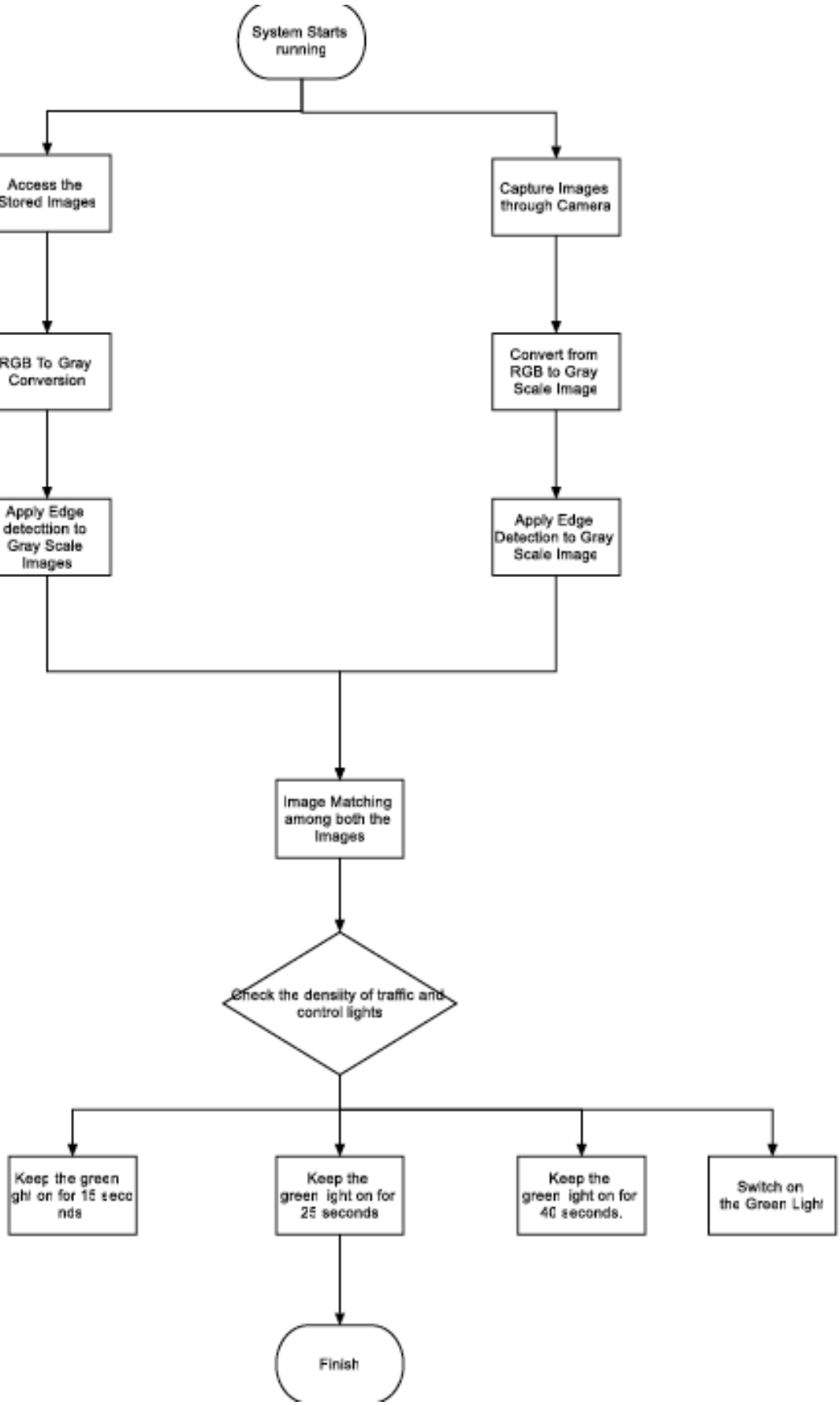

Figure 3 Traffic Density Detection Architecture

Figure 4 Traffic Density Detection Prediction and Prediction Architecture

5.3 Architecture Diagram for Traffic Light Control

The below figures, Fig. 5-8, illustrate the detailed traffic light control system architecture adopted in the current research project.

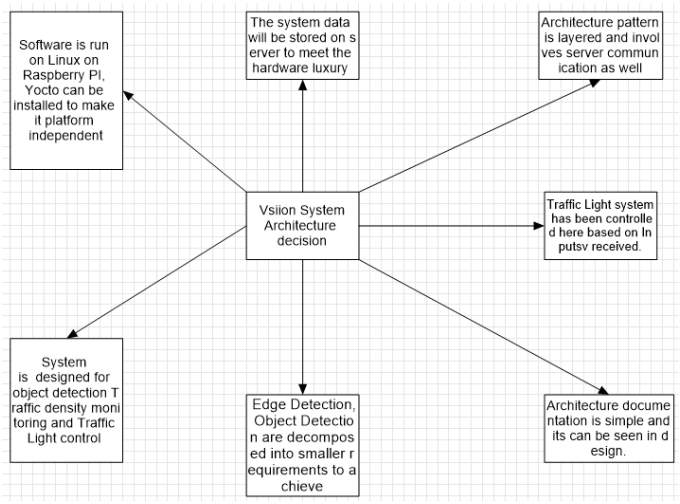

Figure 5Architecture Design Decisions

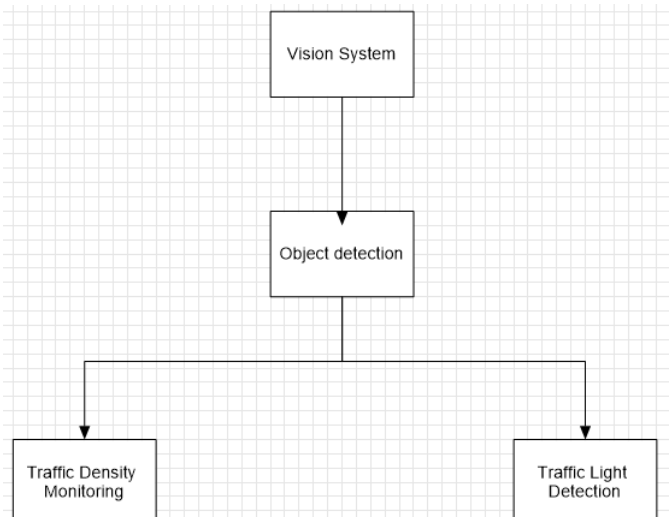

Figure 6 Layered Architecture Model Diagram

In Fig. 7 and 8, scenarios about modules that the Traffic density detection system interacts with (i.e., environment) are mentioned.

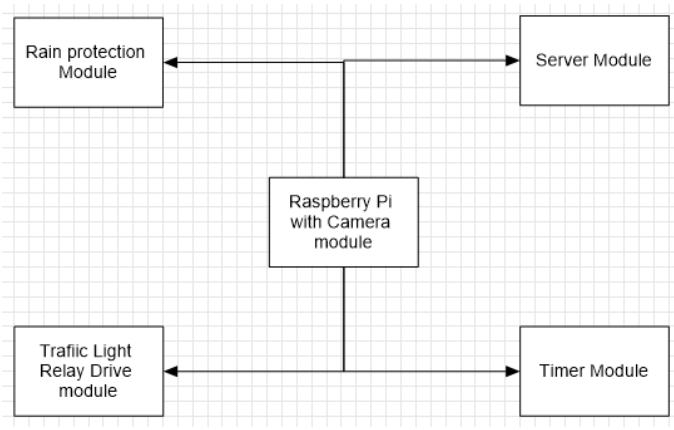

Figure 7 Context Models Traffic Density

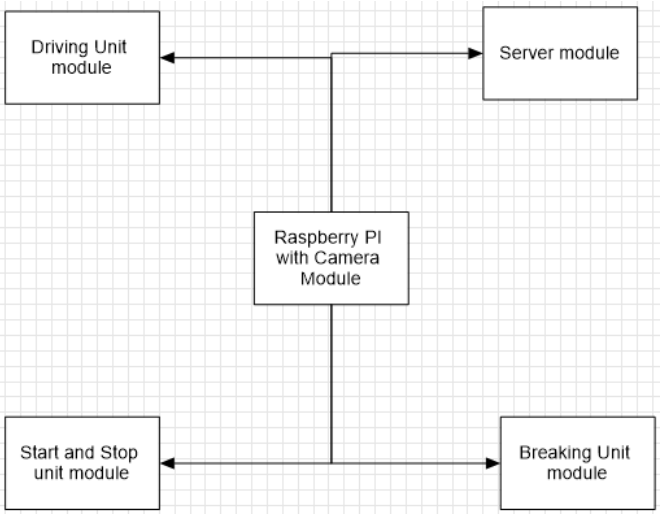

Figure 8 Context Model Traffic Light Detection

5.3.4 Process model

In Fig. 9, the entire flow of traffic density detection module is mentioned.

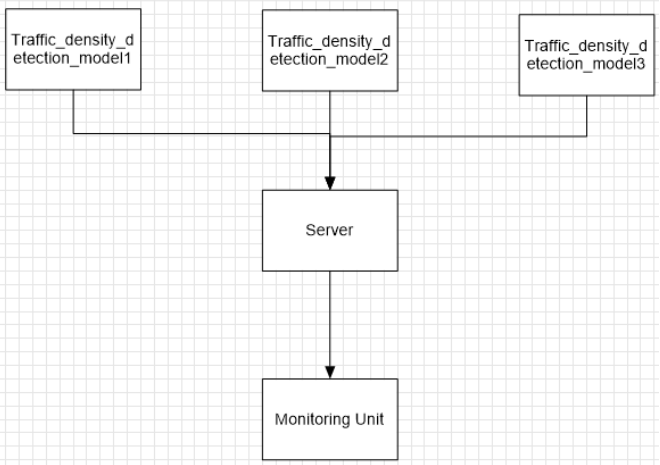

Figure 9 Traffic Density Detection Process Model

In Fig. 10, the entire flow of traffic Light detection module is mentioned.

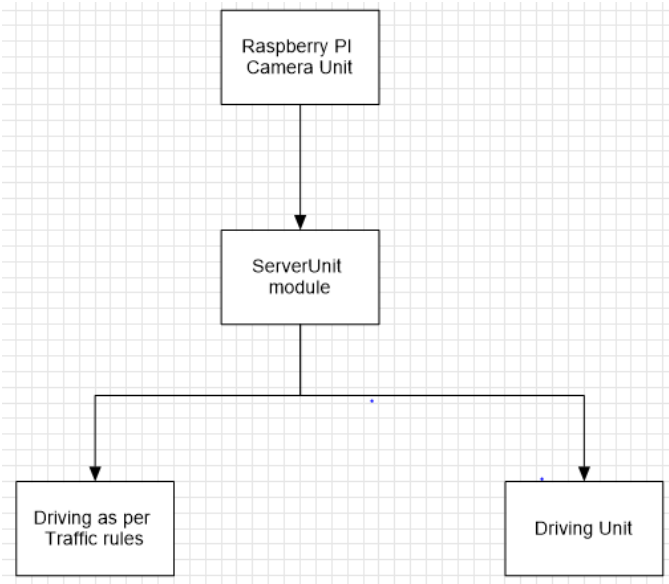

Figure 10 Traffic Light detection Process Model

5.4 State Diagrams

In Fig. 11, the transitions through which Traffic density detection system traverses are mentioned.

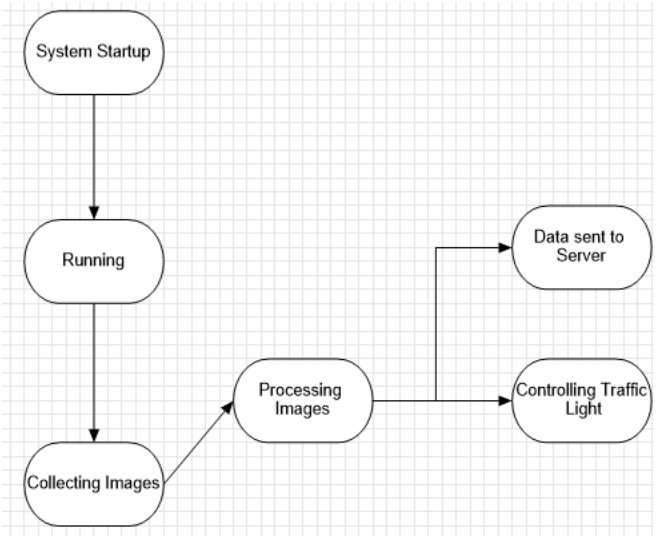

Figure 11 Traffic Density Detection State Diagram

In Fig. 12, the transitions through which Traffic Light detection system traverses are mentioned.

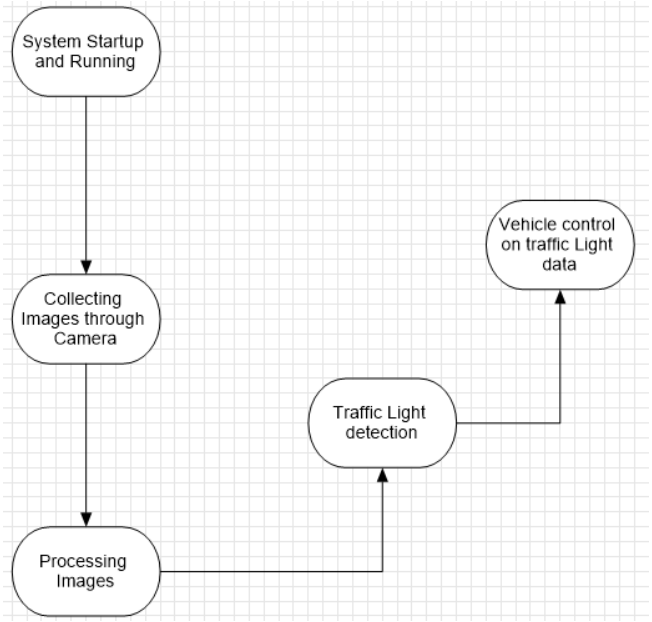

Figure 12 Traffic Light detection State Diagram

5.5 UML Sequence Diagram

In Fig. 13, the UML sequence diagram is presented to illustrate the interactions between traffic density detection system and the traffic light control system.

5.5.1 Traffic Density Detection

- RGB Image of Traffic Density is captured.
- RGB Image is converted to Grayscale Image.
- Threshold Edge Detection Algorithm is applied.
- The count of cars is extracted based on Pixel matching.
- Captured Images are compared with prestored images and checked for Pixel mismatches. The count of Pixel mismatches is considered to get the count of vehicles on the lane.

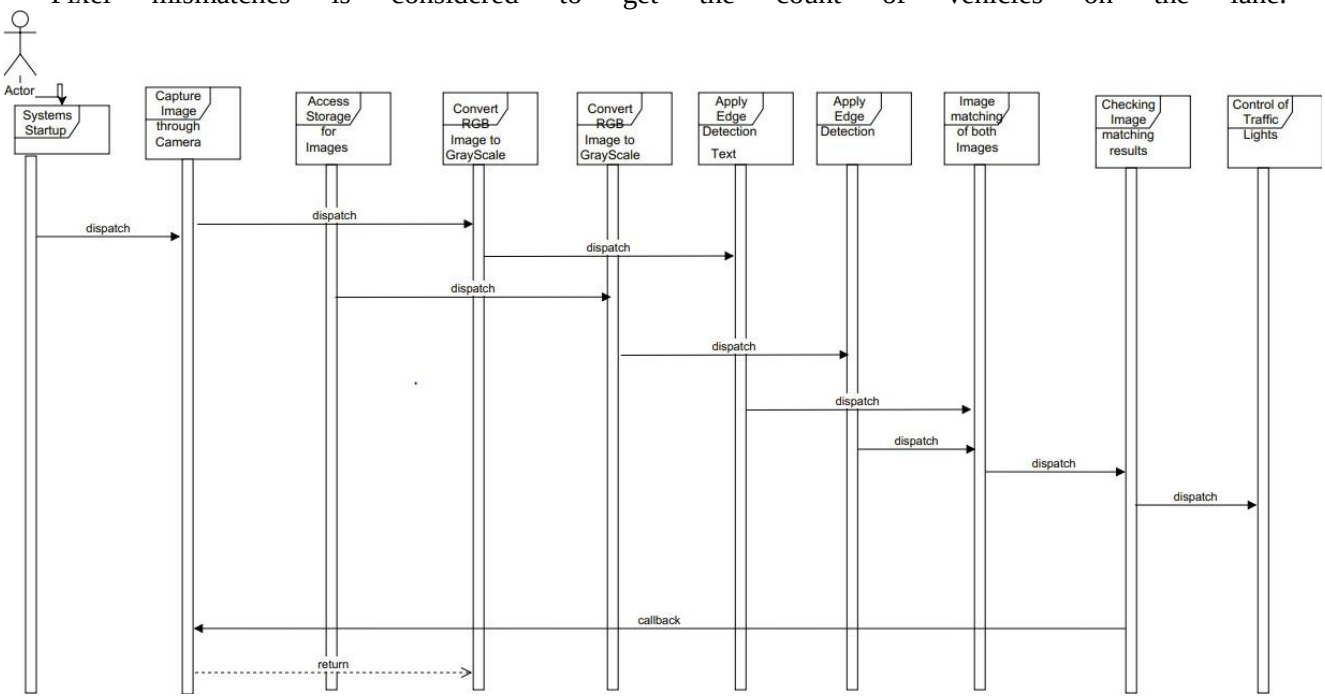

Figure 13 Traffic Density detection and Traffic Light Control

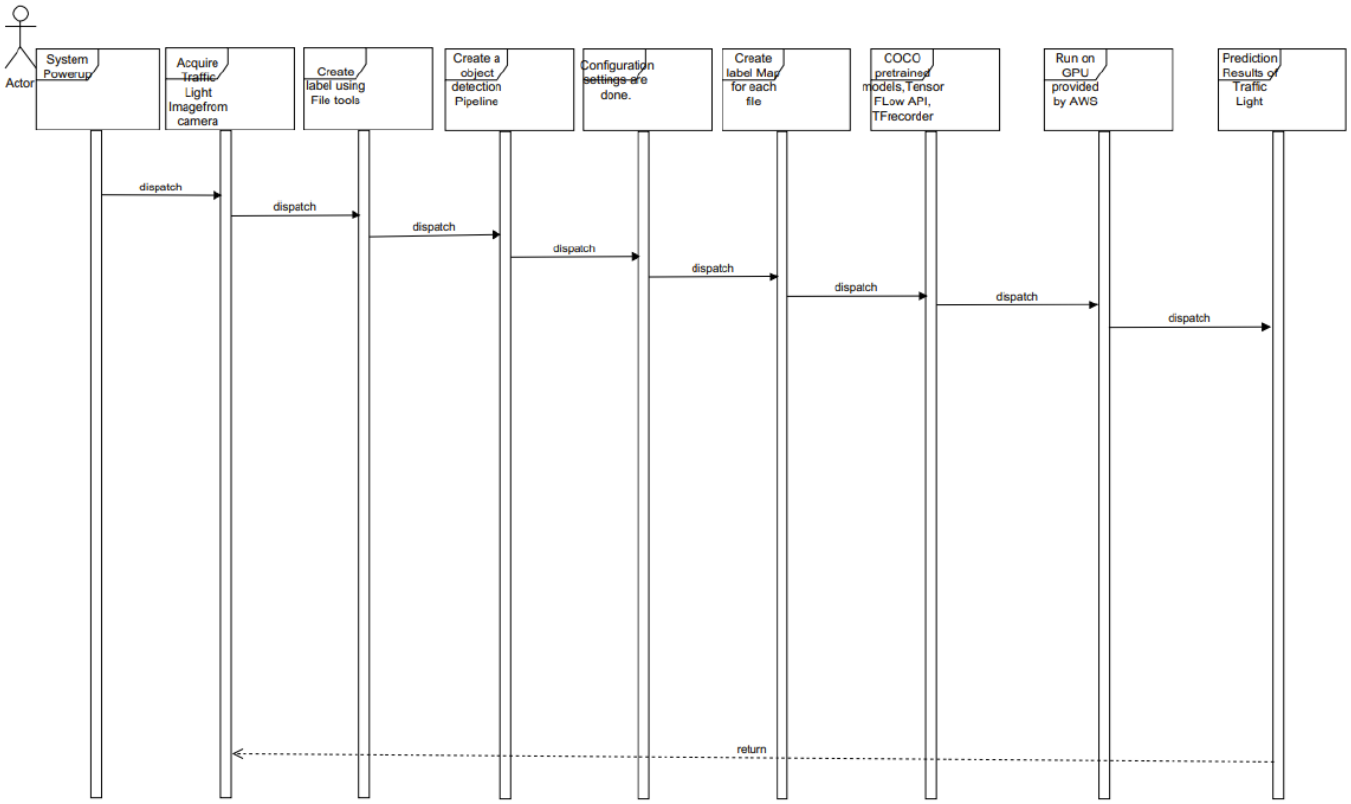

Figure 14 Traffic Light Detection

5.5.2 Traffic Light Detection

Fig. 14 illustrates the interactions involved in the traffic light detection system.

- Images are captured using Camera.
- Label is created for the Images.
- YAML file is created.
- Images and YAML file are converted to TFRecorder format.
- Object pipeline is created.
- Finally, Ssd_inception_v2_coco model and faster_rcnn_resnet101_coco is applied on an Amazon GPU and traffic light prediction is done.

IMPLEMENTATION AND RESULTS

The image processing techniques are implemented on Raspberry pi 4 model using Python and OpenCV.

6.1 Implementation Tools and Techniques

6.1.1 Implementation Tools

The implementation development Host is a Linux based PC and Target device is Raspberry PI execution platform with machine learning libraries from OpenCV for object detection.

6.1.1.1 Raspberry pi

The Raspberry Pi is a series of small single-board computers with no peripherals (i.e., keyboards) or cases [18]. The following Figure, Fig. 15, taken from [18] illustrates the technical specifications of Raspberry Pi board.

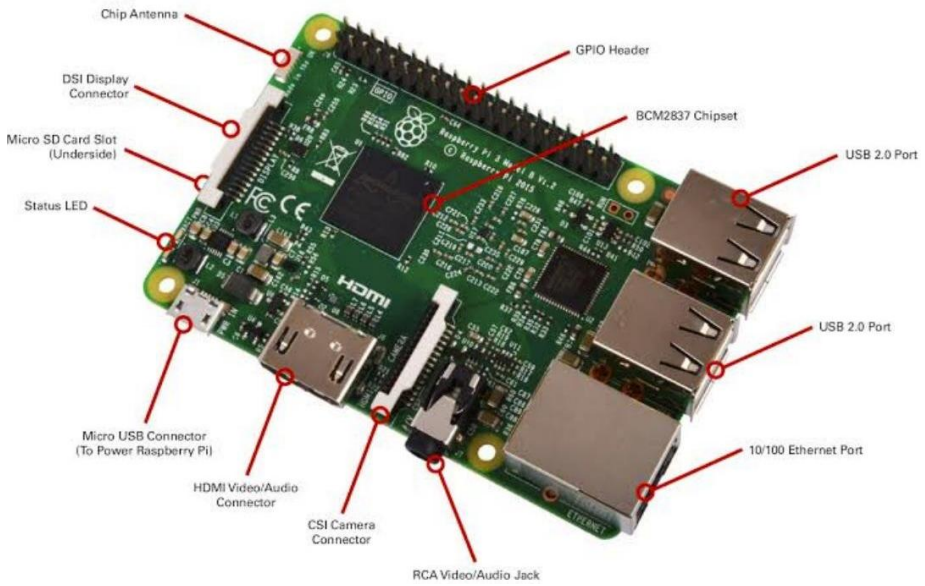

Figure 15Raspberry Pi 3b Board Components

6.1.1.2 Python

Python is a high-level general-purpose programming language [24]. It is used to implement the current research project for the following useful features [24]:

- Easy to learn and use and more expressive, understandable, and readable, especially with the use of whitespace.
- Code is executed line by line making it interpretable.
- Compatible interpreters on different platforms, i.e., Windows, Linux, Unix, and Macintosh.
- Freely available at official web address. The source-code is also available. Therefore, it is open source.
- Code can be compiled using other languages (i.e., C/C++).
- Has a large and broad comprehensive standard library and provides rich set of module and functions for rapid application development.
- Supports graphical user interfaces design.
- Integrated with OOP, C, and C++ languages.
- It has a garbage collection system. Hence, reference cycles can be collected.

5.1.3 Open CV

Open-Source Computer Vision Library (OpenCV)[25] has more than 2500optimized computer vision and machine learning algorithms. An algorithm from this library is used in the current research project to detect and identify objects (cars).

6.1.2 Implementation Techniques and Results

6.1.2.1 Reading Images

The code snippet for reading images is as follows:

edges=cv2.imread("/home/pi/Desktop/Traffic1/ref.png",cv2.IMREAD_GRAYSCALE); cv2.imshow('img', edges)

cv2.waitKey(0)

Below is an explanation for the used functions, objects, and arguments in the above snippet:

- cv::imread - reads a stored reference image "ref.png" from the OpenCV samples set.
- "/home/pi/Desktop/Traffic1/ref.png" - the specified file path.
- IMREAD_GRAYSCALE- converts the colored reference image into grayscale image as in the below figures, Fig. 16-18.

- cv::Mat- the object storing the read image.
- cv::imshow- displays the image.
- 'img' -the title of the window.
- Edges – the cv::Mat object that will be shown.
- cv::waitKey(0) - a function that holds on the window view until the user presses a key on the keyboard. Zero makes it wait forever.

Figures 17 and 18 show the colored reference image with vehicles and its grayscale converted output.

Figure 16 GrayScale Image of Reference Image without Vehicles

Figure 17 Color image of reference image with vehicle took from the webcam in real time

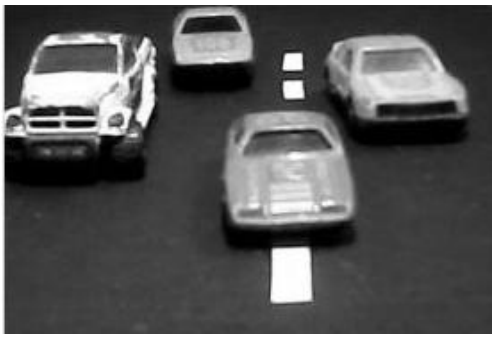

Figure 18 Gray Scale Image of reference Image with vehicle after conversion to grayscale

6.1.2.2 Capturing Image from Camera

Open CV captures a live stream using a digital camera via an interface. The code snippet for reading video frames is as follows:

video=cv2.VideoCapture(0);

check, frame = video.read() cv2.imshow('img', frame) cv2.waitKey(0) video.release()

The following is an explanation for the used functions, objects, and arguments in the above snippet:

- cv::VideoCapture- creates a video object.
- video.read- reads and stores a video object as individual frames.
- cv::imshow- displays a video frame
- cv::videorelease- clears the output and reads new image frames.

6.1.2.3 Edge Detection and Flood

The code snippet for edge detection (using the threshold edge detection technique) and flood filling the boundaries of objects and the holes of video frames is as follows:
th,im_th = cv2.threshold(edges,120,255,cv2.THRESH_BINARY_INV)
cv2 . imshow (' img ' , im _t h) cv2.imshow('img',im_th) cv2.waitKey(0)

Figures 19 and 20 are the Edge detected and flood filled Images converted from Grayscale.

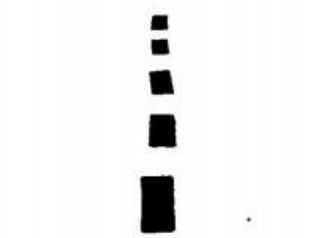

Figure 19 Edge detected and Flood Filled Image without Vehicles

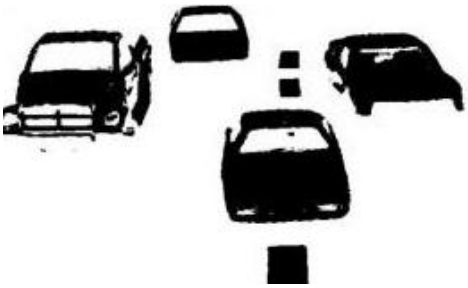

Figure 20Edge detected and Flood Filled Image with vehicle converted from grayscale

6.1.2.4 Comparing the Images and Allocating the Time
Next, the below code snippet compares the reference and captured images, after edge detection and flood filling, to find the pixel differences which various traffic light time periods are assigned based on.

```
Dif=(np.count_nonzero(res)*100)/res.size print(dif)
if(dif<20):
print("time allocated_is"+"15_sec")
time2=15;
elif(dif>20 and dif <40):
print( '"time_allocated_is"+"25_sec")
time2=25;
 else:
print("time_allocated_is"+"35_sec")
time2 =35;
```

6.1.2.5 Operating the LED lights
LED traffic lights are operated using the below code snippet:
import RPi.GPIO as GPIO GPIO.setwarnings(False) GPIO.setmode(GPIO.BOARD)
GPIO.setup(8,GPIO.OUT,initial=GPIO.HIGH) GPIO.setup(5,GPIO.OUT,initial=GPIO.LOW)
GPIO.output(8,GPIO.LOW) GPIO.output(5,GPIO.HIGH)

The traffic light has two conditions, ON or OFF. The OFF condition (red LED light) is represented in the code as high with output pin number 8. The ON condition (green LED light) is represented in the code as low with output pin number 5.

6.1.2.6 Camera Blockage Alert Message
The below code snippet uses Twilio API to send an alert message when the camera view is blocked.
from twilio.rest import Client account_sid="AC1bcfeff26338562e51c7d48a82d838a8" auth_token =

```
"d3c3f95e91b4422abd8fac7313a20575" client = Client(account_sid, auth_token)
message= client.api.account.messages.create( to="+966569894720", from_="+966547023434",
body="CAMERA VIEW BLOCKED AND TIME SET TO DEFAULT VALUE")
```

The message will be sent to a given number; therefore, an authentication key and an account key are needed. The default message view time is35 secs.

6.1.2.7 Traffic Density Detection and Traffic Light Control

The code snippets for traffic density detection and traffic light control are appended in Appendix 1.

ADVANTAGES AND APPLICATIONS

This section discusses the advantages and application areas of the developed traffic light timer adjustment system using threshold edge detection for image processing.

7.1 Advantages

- Captures various and large amounts of data(images) at the same time at traffic speeds, which can also be stored for other uses.
- Energy and environment related benefits.
- Reduces some accident cases, i.e., of right angles accidents.
- Improved system throughput and time consumption.
- Allows drivers to move confidently, especially with autonomous driving cars.
- Provides a better travel time and an orderly movement of traffic with timely interception of heavy traffic to allow others to cross the road intersection safely.
- Controls the vehicles' speed and eases congestion by directing traffic on different routes.

7.2 Applications

An Automatic Density based Traffic management system has several operational roles, including monitoring traffic in real time, analyzing video footage of traffic, and actively managing traffic. In addition, the following practical applications are good possibilities:

- Live public transport updates: Automatic traffic management systems give the public the power to monitor public transportation vehicles' arrival time with the help of the installed GPS devices.
- Re-routing: With the aid of advanced traffic management systems and sensors embedded in roads, drivers gain access to accurate route conditions, information they can use to avoid traffic jams.
- Prompt response by authorities: Advanced traffic management systems allow transportation management authorities to attend to incidents swiftly by having all needed information.
- Traffic safety and control: IoT applications integrate into advanced traffic management systems, e.g., traffic light control systems that help prevent some accidents and reduce traffic congestion.
- Transparency: Data from advanced traffic management systems can promote efficiency in daily transport operations, optimize mobility routes, and help develop new traffic applications.
- Highway advisory radio: Traffic information gathered by licensed AM radio stations can be streamed into the Automatic traffic management system. Official bodies—transport officials, airports, government departments, parks, colleges, and events and destination—frequently inform the public of local travel conditions and warn them of dangers. This data can be gathered systematically via radio stations and integrated into the traffic management system.
- Ramp meters: Automatic Traffic Controlling System helps control and manage the flow of vehicles allowed access to the freeway through ramp meters. Monitoring on-ramp traffic signals enable real-time optimization of freeway congestion.

Traffic Light detection helps in Autonomous car Driving since there would be no drivers, traffic light would be detected, and car would stop, move, or accelerate depending on the predictions from traffic light detection system.

7.3Agile Example of Product Backlog and Story Creation

Monday.com [26] is used to extract the below pictures. Fig. 21 is just an example of how Product Backlog looks like and how the stories are created within a sprint.

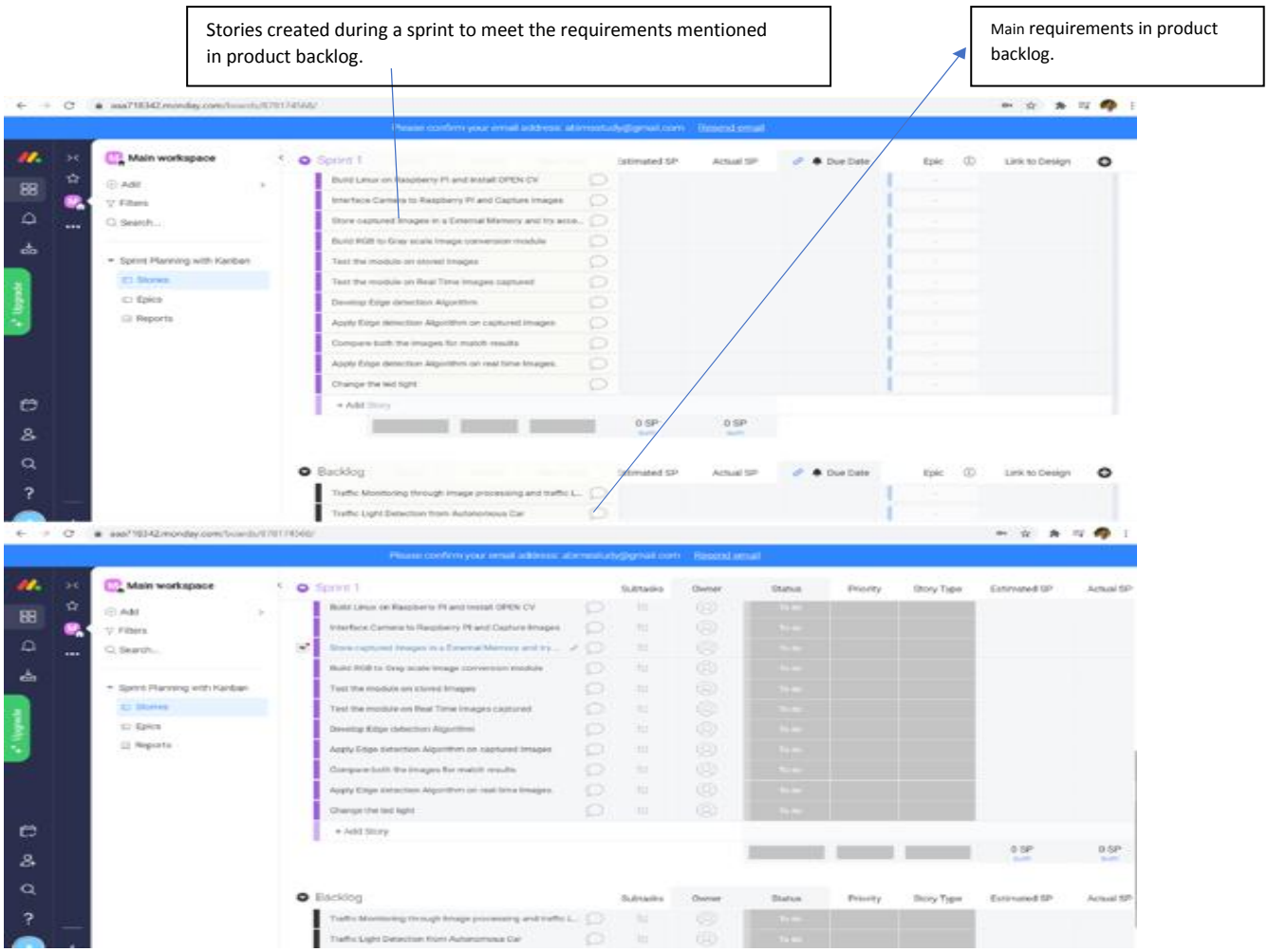

Figure 16 Product Backlog and Story Creation Example Window

IMPLEMENTATION VALIDATION AND DEVELOPMENT TESTING

8.1 Validation Testing

The proposed system has been tested to determine the Traffic density and how long the traffic light glows.

- Different set of Images are provided and checked whether traffic density is calculated in a correct manner and the traffic light glowed for a correct duration.
- Real Time Images are captured in many iterations and checked whether traffic density is calculated properly.
- Different scenarios are considered such as during rain, night, cloudy, snowy and the predictions are monitored, hence, system working is ensured.

The system has also been tested to determine Traffic Light colors.

- Different set of Images are provided and checked whether Traffic Light is predicted properly.
- Real Time Images are captured in many iterations and checked whether Traffic Light prediction is correct.
- Different scenarios are considered such as during rain, night, cloudy, snowy and the predictions are monitored, hence system working is ensured.

8.2 Defect Testing.

Test cases are designed to expose any system's defects.

- *Traffic density is calculated* with different kinds of Images to make sure only Car and Vehicle images are considered. Animal Images or thing images should not affect traffic density calculation.
- Different scenarios are considered such as during rain, night, cloudy, snowy and the predictions are monitored, hence, system working is ensured.
- System's communication with the server is tested and checked whether the system stops working if not able to communicate with the server.
- Traffic is increased/decreased and tested to verify the system working.
- Traffic density is tested at different places with proper roads and Improper roads to check the weather traffic density is calculated properly and object detection works properly.
- Traffic density is tested at Rural, Urban, desert, hilly, with different road inclinations and different weather conditions and verified the system works properly.
- *Traffic Light prediction* is done with different kind of Images to make sure only traffic light post images are considered and similar lights present on road are not considered while predicting traffic light.
- Different scenarios are considered such as during rain, night, cloudy, snowy and the predictions are monitored, hence system working is ensured.
- Traffic light numbers are increased/decreased and tested to verify the system working.
- Traffic light heights and traffic light positions(few may be vertical, and few may be horizontal) are changed and tested to verify the system working.
- Traffic Light prediction are tested at Rural, Urban, desert, hilly, with different traffic light pole structures and different weather conditions and verified the system works properly.

8.3 Verification and Validation:

Verification: "Are we building the product right": Hence System is tested to make sure the product is developed and working as per the System and Software requirements mentioned earlier.

Validation: "Are we building the right product": After product development system is tested using Validation testing and Defect testingscenarios,andthesystemisensuredtoworksafelyduringallthescenarios,hencewemade sure we built the right product.

Verification and Validation are done to assure:

- Software purpose of Traffic density detection and Traffic Light prediction is achieved.
- User expectations of Traffic density detection and Traffic Light prediction is achieved.
- Marketing can be achieved early.

8.4 Development Testing

8.4.1 Unit Testing

- Individual program Units for Traffic density detection and Traffic Light prediction are tested.

8.4.2 Component Testing

- Traffic Density Calculation Unit and Server Unit and Traffic Light setup are combined and tested.
- Traffic Light detection and Autonomous driving unit are combined and tested.
- Interface Testing are achieved here in Component Testing such as the following Procedural Interfaces:
  - Sub-system encapsulates set of procedures to be called, such as traffic density detection and Traffic light control are encapsulated while testing.
  - Stress testing is done with High density traffic to assure the system works as per designed.
  - Stress testing is done with frequent Light poles within certain duration of time and the system is verified whether system prediction is correct and as per design.

8.4.3 System Testing

- *Traffic Density Calculation*: whole system is integrated and tested for its functionality.
  - Interaction between the server and Traffic density detection system is tested and made sure there are no failures, and if there are failures, they are handled safely.
  - Sequence diagrams associated with Traffic density detection is compared to make sure the execution flow remains the same as per the UML sequence diagrams.
- *Traffic Light detection*: whole system is integrated and tested for its functionality.
  - Interaction between Traffic Light detection system and GUI is tested and made sure there are no failures, and if there are failures, they are handled safely.
  - Sequence diagrams associated with Traffic Light detection is compared to make sure the execution flow remains the same as per the UML sequence diagrams.

8.5 Code Coverage Testing

Traffic density detection system is tested for unreachable code. There are many code coverage tools in market, i.e., froglogic, which can be used to check whether all portions of code are getting executed considering all the scenarios. There shouldn't be unreachable code.

8.6 Simplified Debugging

Specifying why the system stopped working or what happened before a system failure becomes very crucial to determine the failure conditions of a system. Hence Logging becomes a very crucial part of the system that is being developed. Specific API should be developed to write important event of actions into a log file. During the system failure, these Log files can be accessed, and system checking can be verified.

8.7 Performance Testing

Traffic Light detection and Traffic Density detection system is tested for performance evaluation. Traffic Light detection is tested how accurately it predicts the traffic light where it scored an accuracy of 99.6%, while traffic density detection is tested how accurately traffic density is calculated and whether traffic lights are controlled precisely where it scored an accuracy of 98.3%.

8.8 User Testing

User or customer tests the software.

- *Alpha testing*: Users test with Traffic density detection and Traffic Light detection development team at developers' site.
- *Beta Testing*: Traffic Light detection software and Traffic density detection software are released to customers to test and raise concerns with system developers.
- *Acceptance Testing*: Traffic Light detection software and Traffic density detection software are tested and accepted whether the system can be deployed in real world.

8.9SystemDocumentation

Traffic Density detection system and Traffic Light detection are documented for all the code work, working of the system, enhancement features with existing code, failure conditions, and Different testing conditions to verify system functionality.

8.10 Implementation Limitations

This system can be reused on any Linux based system with Yocto installed on it; to make it compatible with all Linux based systems. The following are system configurations management guidelines used to develop the model:

- Git can be used for configuration management purpose; since Git is open source and widely used across many companies. There are 4 fundamental configuration Management activities: Version management, System Integration, Problem Tracking, and Release Management. All are taken care of by Git.
- Change proposal or Change management can be done using Jira.
- Development Platform can be considered as Linux based PC, Execution platform Raspberry PI 4 is considered.

CONCLUSION

Traffic control using image processing techniques overcome the limitations of the formally used techniques used for controlling the traffic. Earlier in automatic traffic control use of timer had a drawback that the time is being wasted by green light on an empty road. Upon comparison with various edge detection algorithms, it was inferred that Threshold Edge Detector technique for image processing is the most efficient one for traffic control compared to traditional techniques. Also, it is more effective than the density-based system since it is cost effective and less prone to error. This technique removes the need for extra hardware such as sound sensors & magnetic loop embedded in pavements. In addition, the allocated traffic signal timings vary drastically depending on the traffic density detection using Image matching. The accuracy in the calculation of time due to single moving camera depends on the registration position while facing road every time.
The state change of the traffic light is proved to be best controlled using image processing techniques. Various advantages of the proposed system were discussing, including the reduction of traffic congestion, time saving, and better consistency at vehicle presence detection. These benefits are mainly due to the use of actual traffic images as a form of reality visualization, giving it advanced performance compared to systems relying on the detection of the vehicles' metal content. Overall, the system's performance is the best compared to other similar systems in the literature survey. However, further work on the system is recommended to capture and process traffic images under unpredictable lighting conditions.

REFERENCES

[1] Gaffney, J. S., & Marley, N. A. (2009). The impacts of combustion emissions on air quality and climate–From coal to biofuels and beyond. Atmospheric Environment, 43(1), 23-36.

[2] Buchanan, C. (2015). Traffic in Towns: A study of the long-term problems of traffic in urban areas. Routledge.

[3] Aarts, L., & Van Schagen, I. (2006). Driving speed and the risk of road crashes: A review. Accident Analysis & Prevention, 38(2), 215-224.

[4] Furht, B., Smoliar, S. W., & Zhang, H. (2012). Video and image processing in multimedia systems (Vol. 326). Springer Science & Business Media.

[5] Tari, T., Kóczy, L. T., Gáspár, C., & Hontvári, J. (2006, July). Control of traffic lights in high complexity intersections using hierarchical interpolative fuzzy methods. In 2006 IEEE International Conference on Fuzzy Systems (pp. 1045-1048). IEEE.

[6] Han, B., Zhang, K., Yu, X., Kwon, E., & Ou, J. (2011). Nickel particle-based self-sensing pavement for vehicle detection. Measurement, 44(9), 1645-1650.

[7] Chowdhury, M. A., & Sadek, A. W. (2003). Fundamentals of intelligent transportation systems planning. Artech House.

[8] Choudekar, P., Banerjee, S., & Muju, M. K. (2011, April). Implementation of image processing in real time traffic light control. In 2011 3rd International Conference on Electronics Computer Technology (Vol. 2, pp. 94-98). IEEE.

[9] Lakshmi, C. J., & Kalpana, S. (2017, March). Intelligent traffic signaling system. In 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT) (pp. 247-251). IEEE.

[10] Uddin, M. S., Das, A. K., & Taleb, M. A. (2015, May). Real-time area-based traffic density estimation by image processing for traffic signal control system: Bangladesh perspective. In 2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT) (pp. 1-5). IEEE.

[11] Akoum, A. H. (2017). Automatic traffic using image processing. Journal of Software Engineering and Applications, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958,Volume.8,pp67-71,2017.

[12] Choudekar, P., Banerjee, S., & Muju, M. K. (2011). Real time traffic light control using image processing. Indian Journal of Computer Science and Engineering (IJCSE), 2(1), 6-10.

[13] Amiri, S. A., & Hassanpour, H. (2012). A preprocessing approach for image analysis using gamma correction. International Journal of Computer Applications, 38(12), 38-46.

[14] Tsai, C. M., & Yeh, Z. M. (2008). Contrast enhancement by automatic and parameter-free piecewise linear transformation for color images. IEEE transactions on Consumer Electronics, 54(2), 213-219.

[15] Joshi, S. R., & Koju, R. (2012, November). Study and comparison of edge detection algorithms. In 2012 Third Asian Himalayas international conference on internet (pp. 1-5). IEEE.

[16] Wiegers, K., & Beatty, J. (2013). Software requirements. Pearson Education.

[17] Mohajan, H. (2017). Two criteria for good measurements in research: Validity and reliability. Annals of Spiru Haret University Economics Series, (4).

[18] Upton, E., & Halfacree, G. (2014). Raspberry Pi user guide. John Wiley & Sons.

[19] Possatti, L. C., Guidolini, R., Cardoso, V. B., Berriel, R. F., Paixão, T. M., Badue, C., ... & Oliveira-Santos, T. (2019, July). Traffic light recognition using deep learning and prior maps for autonomous cars. In 2019 international joint conference on neural networks (IJCNN) (pp. 1-8). IEEE.

[20] De Charette, R., & Nashashibi, F. (2009, October). Traffic light recognition using image processing compared to learning processes. In 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 333-338). IEEE.

[21] Tran, T. H. P., Pham, C. C., Nguyen, T. P., Duong, T. T., & Jeon, J. W. (2016, October). Real-time traffic light detection using color density. In 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia) (pp. 1-4). IEEE.

[22] Prakash, U. E., Vishnupriya, K. T., Thankappan, A., & Balakrishnan, A. A. (2018, July). Density based traffic control system using image processing. In 2018 International Conference on Emerging Trends and Innovations in Engineering and Technological Research (ICETIETR) (pp. 1-4). IEEE.

[23] Hasan, M. M., Saha, G., Hoque, A., & Majumder, M. B. (2014, May). Smart traffic control system with application of image processing techniques. In 2014 International Conference on Informatics, Electronics & Vision (ICIEV) (pp. 1-4). IEEE.

[24] Van Rossum, G., & Drake, F. L. (2009). Python 3 reference manual. CreateSpace.

[25] Bradski, G., &Kaehler, A. (2000). OpenCV. Dr. Dobb's journal of software tools, 3, 120.

[26] Monday, A Platform Built for a New Way of Working, link: https://monday.com/, last accessed: Aug. 13th, 2022.

Appendix
Appendix 1

```
import numpy as np
import time
import RPi.GPIO as GPIO
import cv2
from RPLCD import CharLCD
from twilio.rest import Client

account_sid="AC1bcfeff26338562e51c7d48a82d838a8"
auth_token = "d3c3f95e91b4422abd8fac7313a20575"

client = Client(account_sid, auth_token)
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8,GPIO.OUT,initial=GPIO.HIGH)
GPIO.setup(5,GPIO.OUT,initial=GPIO.LOW)
lcd = CharLCD(cols=16, rows=2, pin_rs=37, pin_e=35, pins_data=[33, 31, 29, 38], numbering_mode = GPIO.BOARD)
i=2;
p="/home/pi/Desktop/Traffic1/tled.png";
edges = cv2.imread("/home/pi/Desktop/Traffic1/ref.png");
cv2.imshow('img',edges)
cv2.waitKey(0)
lcd.clear()


edges = cv2.imread("/home/pi/Desktop/Traffic1/ref.png", cv2.IMREAD_GRAYSCALE);
saved=edges;
cv2.imshow('img',edges)
cv2.waitKey(0)

th, im_th = cv2.threshold(edges,120, 255, cv2.THRESH_BINARY_INV);
cv2.imshow('img',im_th);
cv2.waitKey(0)
cv2.imwrite(p,im_th)
n=20
while n>=0:
        lcd.clear()
        lcd.write_string(str(n))
        time.sleep(1)
        n=n-1
while i<5:
    i=i+1;

    video = cv2.VideoCapture(0);

```

```
while i<5:
    i=i+1;

    video = cv2.VideoCapture(0);


    check, frame = video.read() #reading the video object and storing the individual frames inside the loop
    cv2.imshow('img',frame)
    cv2.waitKey(0)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #coverting the frame to gray scale
    cv2.imshow('img',gray)
    cv2.waitKey(0)
    p="/home/pi/Desktop/Traffic1/t1ed1.png"
    th, im_th = cv2.threshold(gray,120, 255, cv2.THRESH_BINARY_INV);
    cv2.imshow('img',im_th);
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    cv2.imwrite(p,im_th)


    img1 = cv2.imread('/home/pi/Desktop/Traffic1/t1ed.png', 0)
    img2 = cv2.imread('/home/pi/Desktop/Traffic1/t1ed1.png', 0)
    cv2.imshow('img',saved);
    cv2.waitKey(0)
    cv2.imshow('img',gray);
    cv2.waitKey(0)
    res1 = cv2.absdiff(saved, gray)
    visible=1
    #--- convert the result to integer type ---
    res1 = res1.astype(np.uint8)

     #--- find percentage difference based on number of pixels that are not zero ---
    dif1 = (np.count_nonzero(res1) * 100)/ res1.size
    print(dif1)
    if(dif1>95):
        visible=0
        message= client.api.account.messages.create(
            to="+918277790482",
            from_="+12029536098",
            body="CAMERA VIEW BLOCKED AND TIME SET TO DEFAULT VALUE")
    #--- take the absolute difference of the images ---
    res = cv2.absdiff(img1, img2)

    #--- convert the result to integer type ---
```


```
while i<5:
    i=i+1;

    video = cv2.VideoCapture(0);


    check, frame = video.read() #reading the video object and storing the individual frames inside the loop
    cv2.imshow('img',frame)
    cv2.waitKey(0)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #coverting the frame to gray scale
    cv2.imshow('img',gray)
    cv2.waitKey(0)
    p="/home/pi/Desktop/Traffic1/t1ed1.png"
    th, im_th = cv2.threshold(gray,120, 255, cv2.THRESH_BINARY_INV);
    cv2.imshow('img',im_th);
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    cv2.imwrite(p,im_th)


    img1 = cv2.imread('/home/pi/Desktop/Traffic1/t1ed.png', 0)
    img2 = cv2.imread('/home/pi/Desktop/Traffic1/t1ed1.png', 0)
    cv2.imshow('img',saved);
    cv2.waitKey(0)
    cv2.imshow('img',gray);
    cv2.waitKey(0)
    res1 = cv2.absdiff(saved, gray)
    visible=1
    #--- convert the result to integer type ---
    res1 = res1.astype(np.uint8)

     #--- find percentage difference based on number of pixels that are not zero ---
    dif1 = (np.count_nonzero(res1) * 100)/ res1.size
    print(dif1)
    if(dif1>95):
        visible=0
        message= client.api.account.messages.create(
            to="+918277790482",
            from_="+12029536098",
            body="CAMERA VIEW BLOCKED AND TIME SET TO DEFAULT VALUE")
    #--- take the absolute difference of the images ---
    res = cv2.absdiff(img1, img2)

    #--- convert the result to integer type ---
```