Help Me Park!

By

NC M. Azhar Munir
NC Sanwal Shabbir
NC Mazhar Abbas

Submitted to Faculty of Department of Computer Software Engineering National University of Sciences and Technology, Rawalpindi in partial fulfilment for the requirements of a B.E Degree in Computer Software Engineering, May 2017

In the name of Allah, the Most Beneficent, the Most Merciful

ABSTRACT

Help Me Park!

In urban areas, finding a vacant parking slot in parking garages or parking lots is time-consuming and a tedious task for drivers. A system to detect available parking spaces to route drivers efficiently to proper lots is desirable. Some systems have reached the market or are under research promising to support the driver by locating a vacant parking lot.

Help Me Park - a video-based system offers a proper alternative to deal with the classification problem. It is possible to combine low-cost hardware requirements with providing detailed occupancy maps for parking areas, which most of the current systems do not provide. Several image processing and machine learning algorithms including Artificial Neural Network, Decision Tree Algorithm, k – Nearest Neighbor, are employed to classify parking slots.

By using video-based systems several challenges occur especially on outdoor car-parks. Different weather and lighting conditions or objects occluding parking lots might influence the accuracy for the given task. Problem of shadow is tackled by using edge-detection technique. Help me park refreshes the state of parking lot every 7 – 9 seconds so users are always served with up-to-date information. Depending on the available training data, Help Me Park is capable of classifying slots with 94% accuracy.

CERTIFICATE FOR CORRECTNESS AND APPROVAL

It is certified that work contained in the thesis – Help Me Park carried out by M. Azhar Munir, Sanwal Shabbir, Mazhar Abbas under supervision of Dr. Naima Iltaf for partial fulfilment of Degree of Bachelor of Software Engineering is correct and approved.

Approved By

Dr. Naima Iltaf

Department of CSE, MCS

Dated: ___________________ ___________________

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

To great Muslim Scientists and Researchers of the history whose accomplishments have greatly contributed to the progress of this modern world and who are of great inspiration to us to excel in the fields of Science and Technology.

ACKNOWLEDGEMENTS

There is no success without the will of ALLAH Almighty. We are grateful to ALLAH, who has given us guidance, strength and enabled us to accomplish this task. Whatever we have achieved, we owe it to Him, in totality. We are also grateful to our parents and family and well-wishers for their admirable support and their critical reviews. We would like to thank our supervisor. Dr. Naima Iltaf, for her continuous guidance and motivation throughout the course of our project. Without their help we would have not been able to accomplish anything.

Table of Contents

List of Figures

Chapter 1. Introduction

1.1 Overview

Help Me Park is basically built to help the people who intend to park their vehicles in parking lots and don’t know where is any slot or space available to park. Usually, they enter the parking lot and consequently create or land into mess. This project will help them know about available slot beforehand.

1.2 Problem Statement

In urban areas, finding a vacant parking slot in parking garages or parking lots is time-consuming and a tedious task for drivers. A system to detect available parking spaces to route drivers efficiently to proper lots is desirable. Some systems have reached the market or are under research promising to support the driver by locating a vacant parking lot.

Existing solutions for real-time parking lot classification either rely on embedded and overhead sensors or use camera video streams to determine the occupancy status. Sensor based systems usually require labor-intensive and time-consuming installation of hardware and wiring for each single parking lot and, in addition, overhead technologies are difficult to install in large outdoor car-parks. These problems prevent the applicability of sensor-based systems in outdoor environments.

Video-based systems, on the other hand, have the potential to provide a cost-effective solution as they support large area observations, do not require tedious sensor installation (except camera sensor) and allow maintenance operation without disturbing the traffic flow.

1.3 Approach

Help Me Park - a video-based system offers a proper alternative to deal with the classification problem. It is possible to combine low-cost hardware requirements with providing detailed occupancy maps for parking areas, which most of the current systems do not provide. Several image processing and machine learning algorithms including Artificial Neural Network, Decision Tree Algorithm, k – Nearest Neighbor, are employed to classify parking slots.

1.4 Scope

The project Help Me Park will help the driver to find the available space for parking the vehicle in parking lots. The project has three modules. First module will acquire the footage to process, second module will process and identify the empty space and third will convey it to driver by displaying a comprehensive occupancy map.

1.5 Objectives

The main objective of this software system is to provide a facility to drivers, who intend to park their vehicles in parking lots, using which they'll be able to locate vacant parking slot easily. Following are the objectives that are kept in mind: -

- Develop a video based solution that can work using existing cameras mounted in parking lot.
- Acquire video footage from camera
- Detect/register slots from selected frame.
- Extract features against every slot.
- Classify the slot based on features as vacant / occupied.
- Generate occupancy map showing the state of parking slots.

1.6 Deliverables

Sr	Tasks	Deliverables
1	Literature Review	Literature Survey and Feasibility Analysis
2	Requirements Specification	Software Requirements Specification document (SRS)
3	Detailed Design	Software Design Specification document (SDS)
4	Implementation	Project demonstration
5	Testing	Evaluation plan and test document
6	Training	Deployment plan
7	Deployment	Complete application with necessary documentation

1.7 Overview of document

1.7.1 Purpose

This document covers the software requirement specifications for project Help Me Park! The idea of the project Help Me Park! is to ensure that the passenger who intend to park their vehicles in the parking lot know if there are any slots that are available to park. This document is meant to outline the features and requirements of Help Me Park, to serve as a guide to the developers on one hand and a software validation document for the prospective client on the other.

1.7.2 Document Conventions

Headings

Heading are prioritized in a numbered fashion, the highest priority heading having a single digit and subsequent headings having more numbers, per their level.

All the main headings are titled as follows: single digit number followed by a dot and the name of the section (All bold Calibri Light, size 18, Centered).

All second level sub headings for every sub section have the same number as their respective main heading, followed by one dot and subsequent sub heading number followed by name of the sub section (All bold Calibri Light, size 16).

Further sub headings, i.e. level three and below, follow the same rules as above for numbering and naming, but different for font (All bold Calibri, size 14).

Figures

All figures in this document have captions, and are numbered. Context and flow diagrams are based on UML standards.

Reference

All references in this document are provided where necessary, however where not present, the meaning is self-explanatory. All ambiguous terms have been clarified in the glossary at the end of this document.

Links to web pages

All links have been provided with underlined font, the title of the web page or e-book is written at the top of the link and the title may be searched on google to pinpoint to the exact address.

Basic Text

All other basic text appears in regular, size 12 Calibri Light. Every paragraph explains one type of idea.

1.7.3 Intended Audience and Reading Suggestions

This requirements document contains general information about Help Me Park App, use cases, functions, features and special technologies. It describes in detail all that an system needs to do in order to correctly convey the available parking slot. Functional and non-functional requirements are addressed separately. System features with use cases and constraints are discussed in detail. System interfaces are also discussed in detail.

For better understanding, the document is divided into chapters

- In chapter 1 & 2, an overall description of Application is provided. First product perspective is presented with product features and main functions. A complete research in project's domain is documented as a part of chapter 2. Then follow user classes and characteristics, operating environments that Application supports as well as design and implementation constrains. After all that, user documentation is presented and will provide you with more details about each feature's technology.
- In chapter 3 & 4 most important features are presented with detailed description, use cases and requirements.
- In chapter 5 testing and evaluation of the parking system is documented with detailed test cases for unit and integration testing.

This document is intended for:

- Developers: (Project Group)

  To be sure that they are developing the right project that fulfills the requirements provided in this document.
- Testers: (Project Group, Supervisor)

  To have an exact list of the features and functions that must respond according to requirements.
- Users: (Public/Drivers)

  To get familiar with the idea of the project and how to use/respond in failure situations and suggest other features that would make it even more functional.
- Project Supervisor: (Dr. Naima Iltaf)

  This document will be used by the project supervisor to check and guide the group about the understanding and implementation of the requirements properly and completely during the development lifecycle.
- Project Evaluators: (CSE Dept. MCS)

  To know the scope of the project and evaluate the project throughout the development for grading.

References

More about application can be retrieved from project development team.

M. Azhar Munir
BESE 19 MCS NUST
Group Leader – Dev. Team
Email: azharmunir123@gmail.com

Chapter 2. Literature review

2.0 Preamble

Some systems promising to improve the parking lot search and degree of capacity utilization have reached the market or are under research. We review two different types of systems, video-based and other sensor-based systems (like inductive loop detectors, ultra-sound, etc.). Most of video based systems are computationally so intensive that they need more 70 to 85 seconds to classify all slots. However, sensor based systems have been very popular due to accuracy but are very costly and difficult regarding maintainability.

2.1 Sensor Based Solutions

Concerning the sensor installation procedure, the sensor based systems can be divided into two categories: intrusive and non-intrusive sensor systems. While intrusive sensors are typically installed in the surface, by tunneling under the surface or anchoring to the surface, leading to invasive disruptions, non-intrusive sensors can be easily installed on the ground or the ceiling of a car park. The first kind of systems are also referred to as pavement embedded systems whereas the latter are sometimes called overhead technologies [1].

A common type of pavement embedded sensors are inductive loop detectors (ILDs), which are wire loops either installed at the car park entrance to count entering and leaving vehicles or at each parking space leading to expensive and disruptive maintenance work [2]. Moreover, the loops are subject to wear and tear due to stresses of traffic and temperature, and they are sensitive to water [1].

Another type of embedded systems uses magnetic field sensors, either magnetometers or magneto resistive sensors (e.g., anisotropic magnetoresistance sensors), that measure changes in the magnetic flux to detect parking vehicles [3]. Although these kinds of sensors are insensitive to weather conditions and certain types can be installed on the surface, they need to be employed at each parking lot which can be very costly as each sensing unit is usually attached with a processing unit and a transceiver [4].

Overhead occupancy detection methods are either based on light, sound, or radar sensor systems. Active and passive infrared sensors are usually installed above the vehicles, transmit and receive energy (active) or only receive energy (passive), and recognize changes in the characteristics of the received energy to detect the occupancy status of a parking lot [5]. The drawback of infrared sensors is their sensitivity towards environmental conditions such as heavy rain, dense fog, and blowing snow [6].

2.2 Video Based Solutions

Besides sensor based systems, camera based systems have gathered great attention in recent years [7], since they have the potential to provide a cost-effective solution as they allow wide area detection and regular maintenance is possible without disturbing the traffic flow. This substantially reduces life cycle costs and increases detection flexibility [1]. Moreover, they can use existing visual surveillance infrastructure such as security cameras to capture images and videos [8]. Either static images or sequences of images (videos) are fed into computer vision based algorithms that are designed to classify the occupancy of a parking lot as vacant or occupied. Existing vision-based approaches can be roughly classified into three categories: vehicle-driven, space-driven and mixed methods.

2.2.1 Vehicle-driven method

Under vehicle-driven methods, parking vehicles are the major target and algorithms are developed to detect those. Based on the detection result, vacant parking spaces are determined. Many different vehicle detection algorithms have been proposed in recent years.

True [9] has combined interest point detection and color histogram classification using the k-nearest neighbor algorithm and support vector machines to detect vacant parking spaces. The limitations of this work are the relatively low detection accuracy (94%) and the “very slow” processing speed.

2.2.2 Space-driven methods

For space-driven methods, the characteristics of a vacant parking space are in the major focus such that available vacant parking spaces can be detected directly. For that purpose, common background subtraction algorithms are used that assume statistically stationary background variation. Unfortunately, this assumption might be violated in outdoor scenes, since passing clouds may suddenly change illumination.

Funck et al. [1] have presented a method to handle dynamic variations of an outdoor environment by creating an eigen-space representation that contains a huge set of background models. In this method, the occupancy estimate is determined by the vehicle to car park (empty lot) pixel ratio while compensating perspective distortion and occlusion. However, the high rate of error in the detection 10% can be seen as the major drawback.

2.2.3 Mixed and other methods

Mixed approaches try to combine both vehicle-driven and space-driven methods to improve the detection rate. Huang and Wang [7] have presented a Bayesian detection framework that considers both a ground plane model and a vehicle model to represent environmental changes, including perspective distortion, occlusion, shadows and lighting fluctuation. Their systems do not reach real-time performance.

2.3 Conclusion

Unfortunately, several solutions that are presented till now do not report on processing speeds. Also, most of researches have not shared their technical resources. We conclude from search that a cheap and fast system is desirable. That’s why we focused on three goals during the design of this solution: it is highly desirable that parking lot classification can make use of available surveillance cameras to reduce cost, required processing power should be minimum enough to work fine on mini-computer like Raspberry Pi leading to a firmware with maximum portability, applying a system at a new, unseen location should be easy, i.e., not require extensive adjustments like slot registration.

Chapter 3. Software Requirement Specification

3.1 Introduction

This chapter translates the project objectives in to functionalities by specifying the requirements of project. First, project is explored in detail by discussing perspective, functions and operating environment. Then approach for tackling the problem statement is explored by stating the function of system. These functions are then converted into precisely written Functional and Non-functional requirements. Moreover, this chapter also states the design and implementation constraints that should be kept in mind.

3.2 Overall Description

3.2.1 Product Perspective

Help Me Park is basically built to help the people who intend to park their vehicles in parking lots and don't know where is any slot or space available to park. Usually, they enter the parking lot and consequently create or land into mess. This project will help them know about available slot beforehand.

3.2.2 Product Functions

The main features of Help Me Park are highlighted below:

- Camera set-up in parking lot is attached to raspberry pi
- Raspberry pi receives feed from camera
- Video is processed and empty slot is identified
- Identified slot is then conveyed to driver

3.2.3 User Classes and Characteristics

Following are user classes and their brief description.

Tester (occasional user)

Tester will use this project to check for bug finding. They will also use the project to check if it's in accordance to the Software Requirements Specification document.

Project Supervisor (occasional user)

Project supervisor will also use the product to evaluate. They will use this product to find the accuracy and error in the output.

Driver (Regular user)

Driver looking for available space in parking lot will use the system to find the slot quickly and effortlessly.

3.2.4 Operating environment

Required operating environment for the application is listed below.

Hardware Requirements

- Camera Mounted in the Parking lot: The camera is used to make video of the parking lot in real time.
- Cable: The data (video feed) will be transferred from camera using cable to the raspberry pi.
- Raspberry Pi: Software installed which will process the data obtained through the camera and identify the available space.
- LCD Screens: LCD Screens will be used to convey the available space.

Software Requirements

- Linux: Raspbian Jessie
- OpenCV 3
- IDE: Python IDLE

3.2.5 Design and Implementation Constraints

Constraints of the product are given below:

- Help Me Park will only process the video when camera is aligned with the parking space.
- Parking lot should have slots clearly lined up and marked.
- Input may contain noise along with data.

3.2.6 User Documentation

A user manual will be provided to the users in which separate instructions will be given according to the user i.e. Regular user and the admin, developers and testers. It will include the details of the system’s working. Help documents will also be a part of the system.

The project report will also be available for the users which will highlight the system features, working and procedures.

3.2.7 Assumptions and Dependencies

- User must know the language and User Interface for the better performance of the product.
- Limitations of the product must be kept in mind by the user.
- Normal weather conditions have been assumed for our project. Our system depends upon the parking lot’s proper maintenance to work properly.

3.3 System Features

System features are organized by use cases and functional hierarchy so that the main functions of the system will be understandable. In the description of system features there are several references in various system interfaces. These interfaces are better explained in section 4.1 of this document.

3.3.1 Video Acquisition

Description

This feature enables the system to acquire video from camera mounted in parking lot.
This video will be fed into the system for further processing.

Stimulus/Response Sequences

Normal Path: Video sent for processing
Preconditions • The camera captures the video in real time.
Interactions • The captured video is sent to the system for processing.
Post conditions • Captured video is divided into frames
Categorization • **Criticality**: High • **Probability of Defects**: Medium • **Risk**: High

Exceptional Path: Beep Sound is produced
Preconditions • The camera is disconnected/malfunctioned.
Interactions An error signal is raised by the system
Post conditions • The beep sound is produced.
Categorization

• **Criticality**: High • **Probability of Defects**: Low • **Risk**: High

Functional Requirement

The system shall be able to acquire footage of parking lot in real time for further processing.

3.3.2 Video Processing

Description

This feature identifies best images from acquired video as input for further operations.

Stimulus/Request Sequences

Normal Path: Successfully divided into frames
Preconditions • The video is sent to the system for processing
Post conditions • Frames are used for identification of slots
Categorization • **Criticality**: Medium • **Probability of Defects**: Medium • **Risk**: Medium

Functional Requirement

1. System shall be able to effectively divide video in to frames.
2. System shall be able to select best image from divided frames.

3.3.3 Identification of Slots

Description

The images obtained from previous stage will be used to identify all the slots in parking lot.

Stimulus/Response Sequence

Normal Path: Slots identified
Preconditions • The image with appropriate data is identified.
Interactions • Slots identified using images.
Post conditions • Identified slots will be labeled
Categorization • **Criticality**: High • **Probability of Defects**: High • **Risk**: High

Exceptional Path: Slots not identified or Error occurred
Preconditions • The image identified is not suitable for classification.
Interactions An error signal is displayed on LCD
Post conditions • The beep sound is produced
Categorization • **Criticality**: High • **Probability of Defects**: Low • **Risk**: High

Functional Requirements

System shall be able to identify all the slots in parking area by interpreting boundary lines of slots defined on the ground.

3.3.4 Slots Labeling

Description

Identified slots from previous stage will be labelled as available or occupied.

Stimulus/Response Sequences

Normal Path: Slots are labeled
Preconditions • All the slots are identified
Interactions • Identified slots are labeled as available or occupied
Post conditions • Sketch is generated using labeled slots
Categorization • **Criticality**: High • **Probability of Defects**: Medium • **Risk**: High

Functional Requirement

1. System shall be able to label slots as available/occupied efficiently and correctly from identified slots (at 30 second refresh rate).
2. System shall be able to label slots as available/occupied efficiently and correctly from identified slots (signal from external system).

3.3.5 Mapping the Parking lot

Description

Labelled slots received will be processed to create sketch of complete parking area which will be sent for display.

Stimulus/Response Sequences

Normal Path: Sketch is generated
Preconditions

• All the identified slots are labeled
Interactions • All the labeled slots are combined to generate sketch
Post conditions • Sketch is displayed
Categorization • **Criticality**: High • **Probability of Defects**: Medium • **Risk**: High

Functional Requirement

System shall be able to combine all labeled slots to make fine sketch of parking area.

3.3.6 Display slots

Description

Sketch of parking area is displayed on an LCD at entry gate from where driver can easily find available slots and route to them.

Sequence/Response Sequences

Normal Path: Sketch is displayed
Preconditions • Sketch is generated
Post conditions • The driver will be able to see empty and filled slots
Categorization • **Frequency**: High • **Criticality**: High • **Probability of Defects**: Medium • **Risk**: High

Exceptional Path: Beep Sound is produced
Preconditions - Sketch is generated
Interactions LCD is disconnected
Post conditions - The beep sound is produced
Categorization - **Criticality**: High - **Probability of Defects**: Low - **Risk**: High

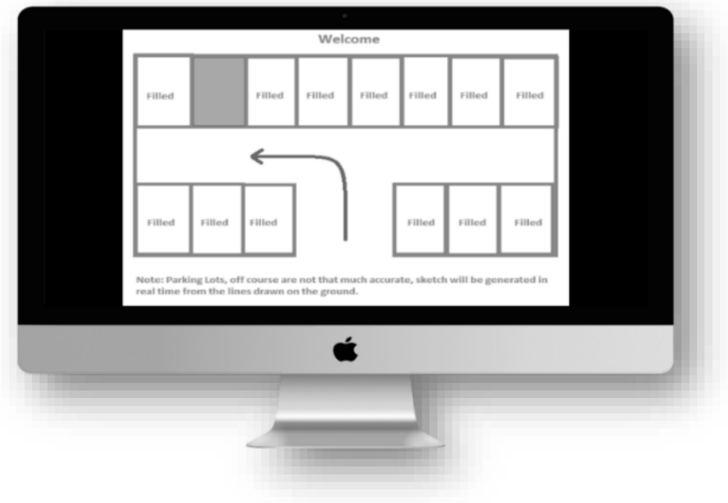
Functional Requirements

Right sketch of parking area containing information of occupied and available slots shall be displayed by the system.

3.4 External Interface Requirements

User Interfaces

Driver will be able to view state of parking slots.



Note: Parking Lots, off course are not that much accurate, sketch will be generated in real time from the lines drawn on the ground.

Figure 1 – LCD Mounted to convey the real-time state

Hardware Interfaces

- Video input will be taken from camera in real time.
- Communication between camera and raspberry pi will be done using a cable.
- Raspberry pi will be connected to LCD screen where output will be displayed

Software Interfaces

- To convert images to sketch Raspberry pi will be used.
- OpenCV will be used to process images.
- Python packages for different operations
- Scikit-learn for machine learning algorithms

Communications Interfaces

Cable will be used as medium of communication between camera and software. This cable will transfer video from camera to Raspberry pi.

3.5 Other Nonfunctional Requirements

Performance Requirements

Application shall run on a minimal amount of memory and take up a small amount of disk space after install. Depending on the performance of the user’s computer, the communication might slow down the Application.

Safety Requirements

This application is a fast and responsive program. However as mentioned in section 5.1 working with large data may lead Application to become unresponsive or even crash. Network crash will also waste a lot of time and user may lose information about message transfer.

Security Requirements

User has direct access in this application. Password or username are not required except for admin panel. To be able to use application user should also allow application on firewall and use system and network resources. The user should also share his IP address only with other trusted users.

Software Quality Attributes

- Reliability

  Application should provide reliability to the user. The product will run stably with all the features mentioned above available and executing perfectly. It should be tested and debugged completely. All exceptions should be well handled.

- User Friendliness/Simplicity

  Application should have a graphical user interface with user friendly menu and options.

- Availability

  Application will be provided to field commander through proper military protocols.

- Accuracy

  To ensure reliability and correctness, there will be zero tolerance for errors in the algorithm that computes results.

Chapter 4. Software Design & Implementation

4.1 Introduction

This chapter describes the architecture and system design of project Help Me Park, release number 1.0. The chapter is meant to detail the design of features and requirements of Help Me Park, to serve as a guide to the developers on one hand and a software validation document for the prospective client on the other. It also includes classes and their inter-relationships, use cases with detailed descriptions, sequence diagrams and various flow charts.

For simplicity, the chapter is divided into various sections. Section 2 includes detailed description of the system architecture. This section includes all the architectural details of system under development. Section 3 describes all the modules and components of the system in detail one by one. Section 4 discusses relationships and interfaces in a system. Section 5 throws light on the design decisions and tradeoffs.

This document is intended for developers, testers, users, documentation writers, project clients, project supervisor and project evaluators. A copy of this document will be made available to all stakeholders.

4.2 System Architecture Description

This section provides detailed system architecture of Help Me Park. Overview of system modules, their structure and relationships are described in this section. User interfaces and related issues are also discussed.

4.2.1 Overview of Modules

Help Me park requires several modules to work. Following is the brief overview of all these modules. Detailed descriptions of these modules are presented in section 3.

Image Acquisition

This is the module from where the major functioning of application initiates. This module takes a frame from video enhances its quality and pass this frame to slot identifier module.

Slots Identification Module

When Image Acquisition module has completed its work and an enhanced frame has been obtained from video sequence, this module will identify the slots (*Regions of Interest*) from the frame based on lines drawn on ground.

Slots labeling / Classification Module

Classification module will classify the slots as vacant or occupied based on the processing through feature extraction and trained model.

Sketch Generation Module

Once slots are identified and successfully labeled as vacant or occupied the next step is to convey it to driver who intends to park. This module will generate a sketch from the slots identified as vacant or occupied and this will be displayed to driver through GUI.

Display Module

This module will display the up to date sketch over to LCD from where drivers would be able to choose which slot to park.

4.2.2 Structure and Relationships

This section covers the overall technical description of P2P Intranet App. It shows the working of application in perspective of different point-of-views and shows relationships between different components.

System Block Diagram

This diagram shows the higher-level description of the application. It shows all the modules of the system and their associations and flow of data between modules.

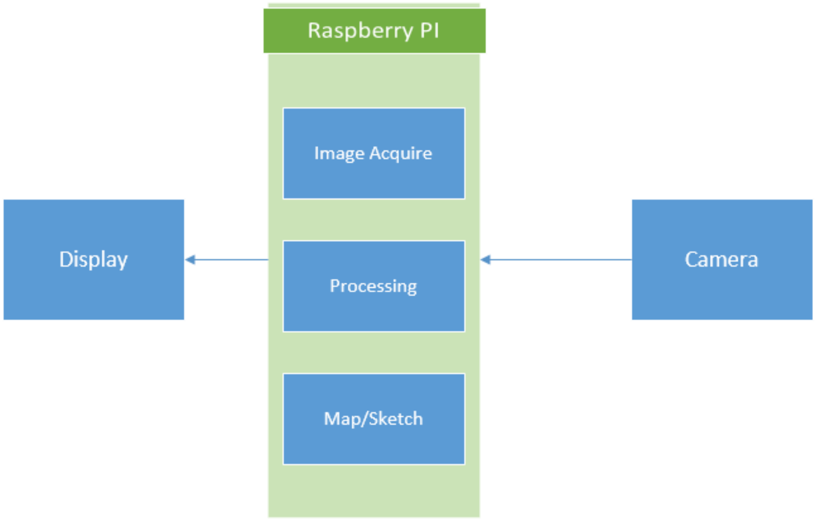


Figure 2 – Block Diagram

Image acquisition module would acquire the footage from camera module. It will then divide the footage into a sequence of images and then apply preprocessing step on this frame. This frame will then be processed to extract the information of parking slot to identify as vacant or occupied. Finally, this status of parking lot will be conveyed to driver.

User View (Use case diagram)

Following diagram shows course of events that take place when an actor (user and other allowed interactions) interacts with system.

Uses cases shown in above figure are described below in detail.

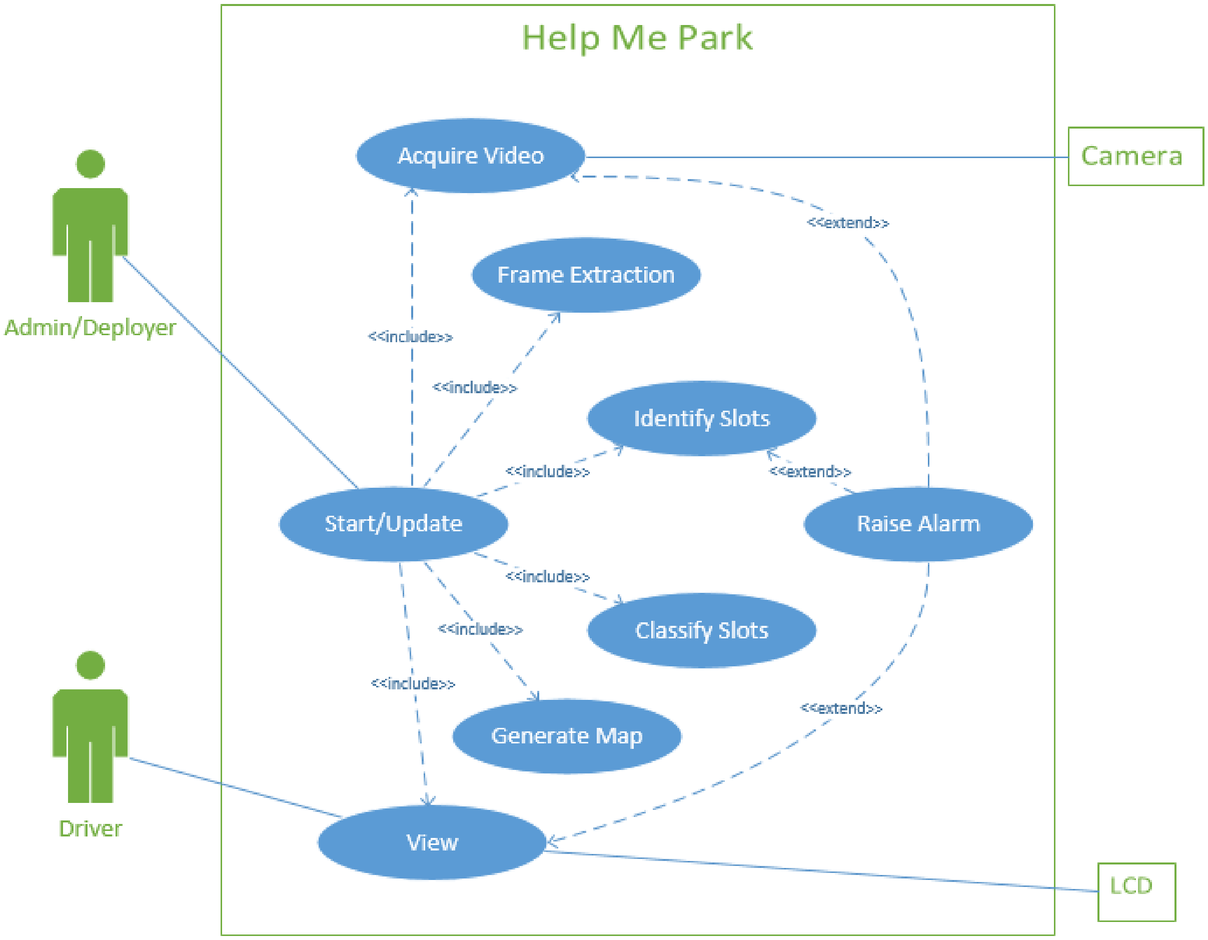
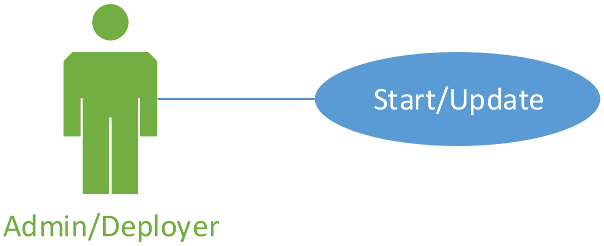


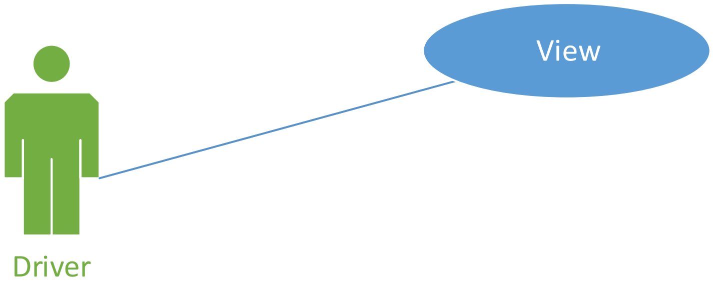
Figure 3 – System Use Case Diagram

Use Case 1



Use case name	Start/Update
Primary actor	Administrator/Deploying Person
Secondary actor	N/A
Normal course	- Successfully acquire video from camera - Video divided into frames and best frame is selected. - All slots in parking area are identified by interpreting boundary lines of slots defined on the ground. - Slots are labeled as available/occupied efficiently and correctly from identified slots - A sketch showing status of all parking slots is generated. - Right sketch of parking area containing information of occupied and available slots shall be displayed by the system.
Alternate course	- Error occurred due to misconfiguration or wrong calibration of camera consequently producing a beep.
Pre-condition	Raspberry pi is configured properly to run the application and camera is aligned to it.
Post-condition	Slots are classified as available or occupied successfully.
Extend	Raise Alarm
Include	Acquire Video, Frame extraction, Identify Slots, Classify Slots, Generate Map, View
Assumptions	Camera is properly aligned.

Use Case 2 (Interaction with LCD)



Use case name	View
Primary actor	Driver who is looking for available slot to park vehicle.
Secondary actor	LCD
Normal course	- Up to date information is displayed on LCD.
Alternate course	- Exception raised and system produced alert signal.
Pre-condition	Slots have been identified and classified successfully.
Post-condition	Created sketch is successfully displayed on LCD.
Extend	Raise Alarm
Include	Start/Update
Assumptions	N/A

Sequence Diagram

Following sequence diagrams show the sequence of activities performed in all use cases described above.

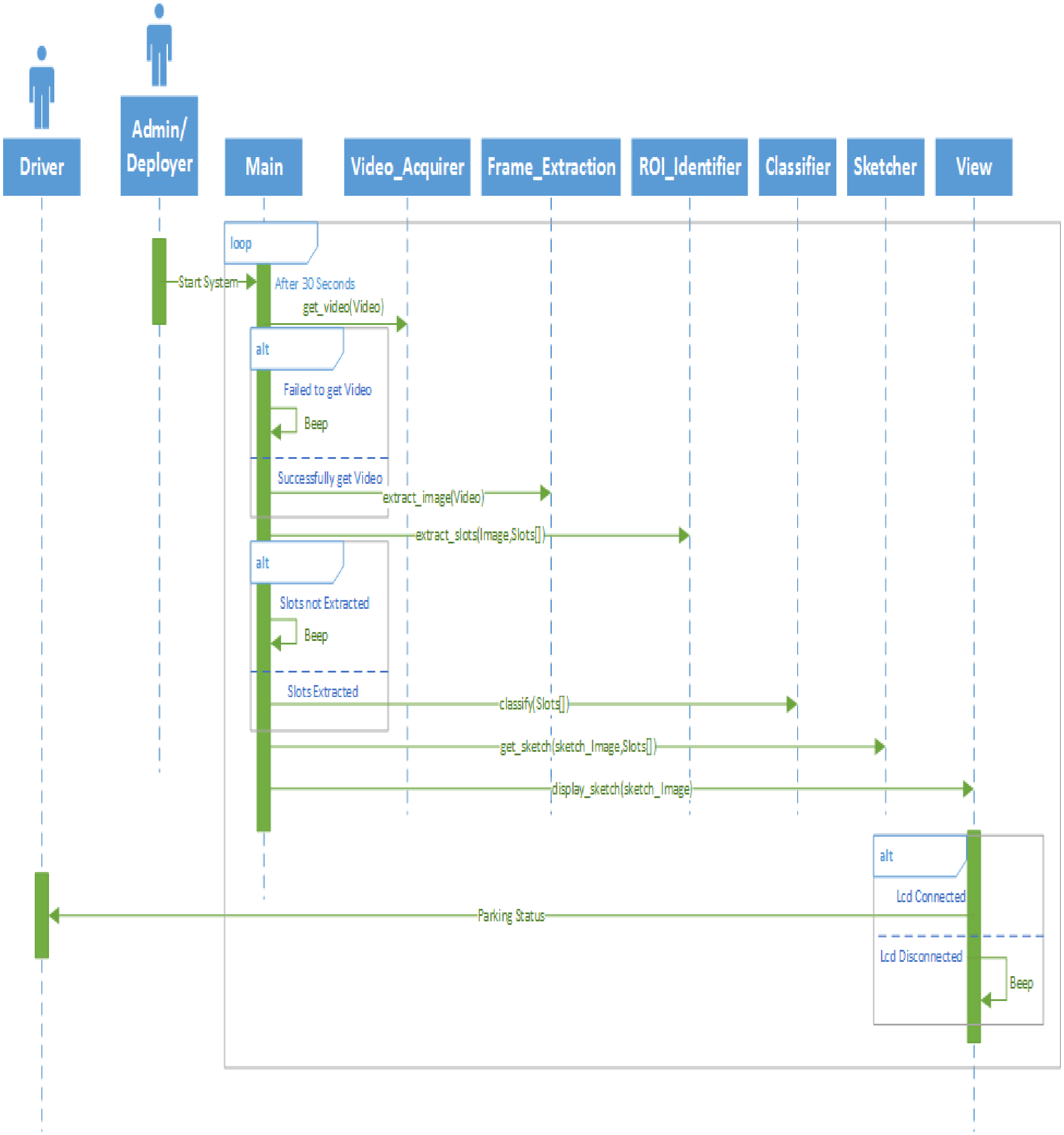


Figure 4 – Sequence Diagram

Implementation View (Class Diagram)

Class diagram shows all the classes of system and their relationship with one another. Following is the class diagram by following the MVC design pattern to implement event driven architecture.

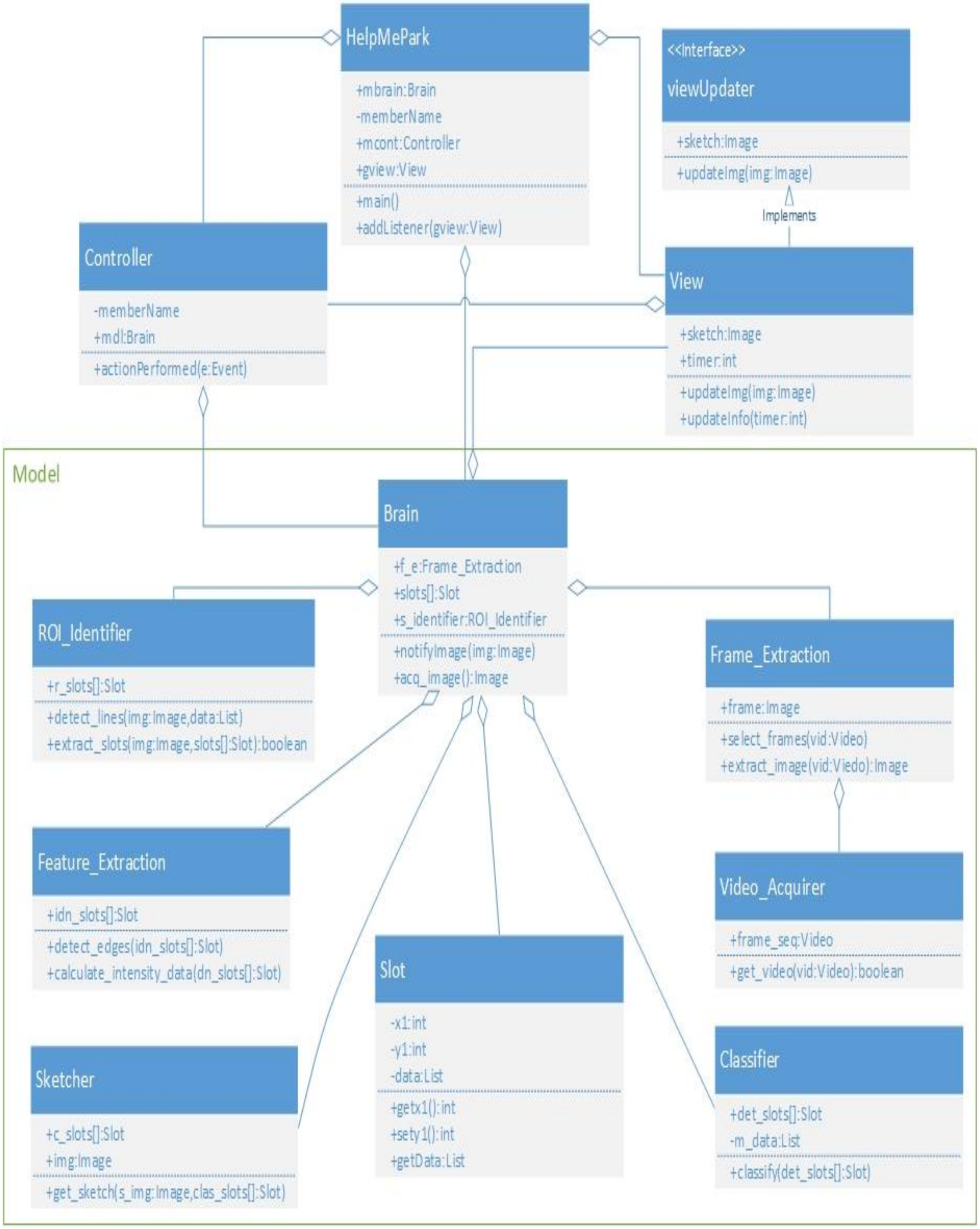


Figure 5 – Class Diagram

Class	Description
HelpMePark	This is main class of the System. It creates the objects of View, Controller and Brain classes. It implements the MVC design pattern.
Controller	Controller class here is performing the MVC's Controller class functionality. It gets actions from view and tell model to act accordingly. It invokes the events by making function calls to different methods in brain class of model at update or start actions.
View	This class plays the role of View class of MVC and generate view. Sketch is displayed by this class. Update calls after 30 seconds are made by this class.
Brain	Brain plays the main role of MVC's Model class. All the events are generated through its functions. It contains all the model classes objects to generate events and all the data that is required to create sketch and classify slots.
Video_Acquirer	This class get the video from camera. It belongs to model.
Frame _Extraction	This class is providing the functionality to extract frames from acquired video. Enhanced Image is produced by this class from extracted frames. It belongs to model.

ROI_Identifier	ROI stands for region of interest. The role of this class is to identify slots in the parking area from enhanced image. It belongs to model.
Feature_ Extraction	This class extract data from Identified Slots which is further used in classifying the identified slots. It belongs to model.
Classifier	Classifier class classify the identified slots as available or Occupied using data extracted by Feature_Extraction class. It belongs to model.
Sketcher	Sketch is generated by this class. It returns the created sketch as image to brain class. It belongs to model.
Slot	This class contain the attributes which are required to save data required by one slot. For each identified slot an object of this class is created. It belongs to model.

Dynamic View (Activity Diagram)

In activity diagram, the dynamic view of the system is shown. All the activities are shown concurrently with their respective start and end states.

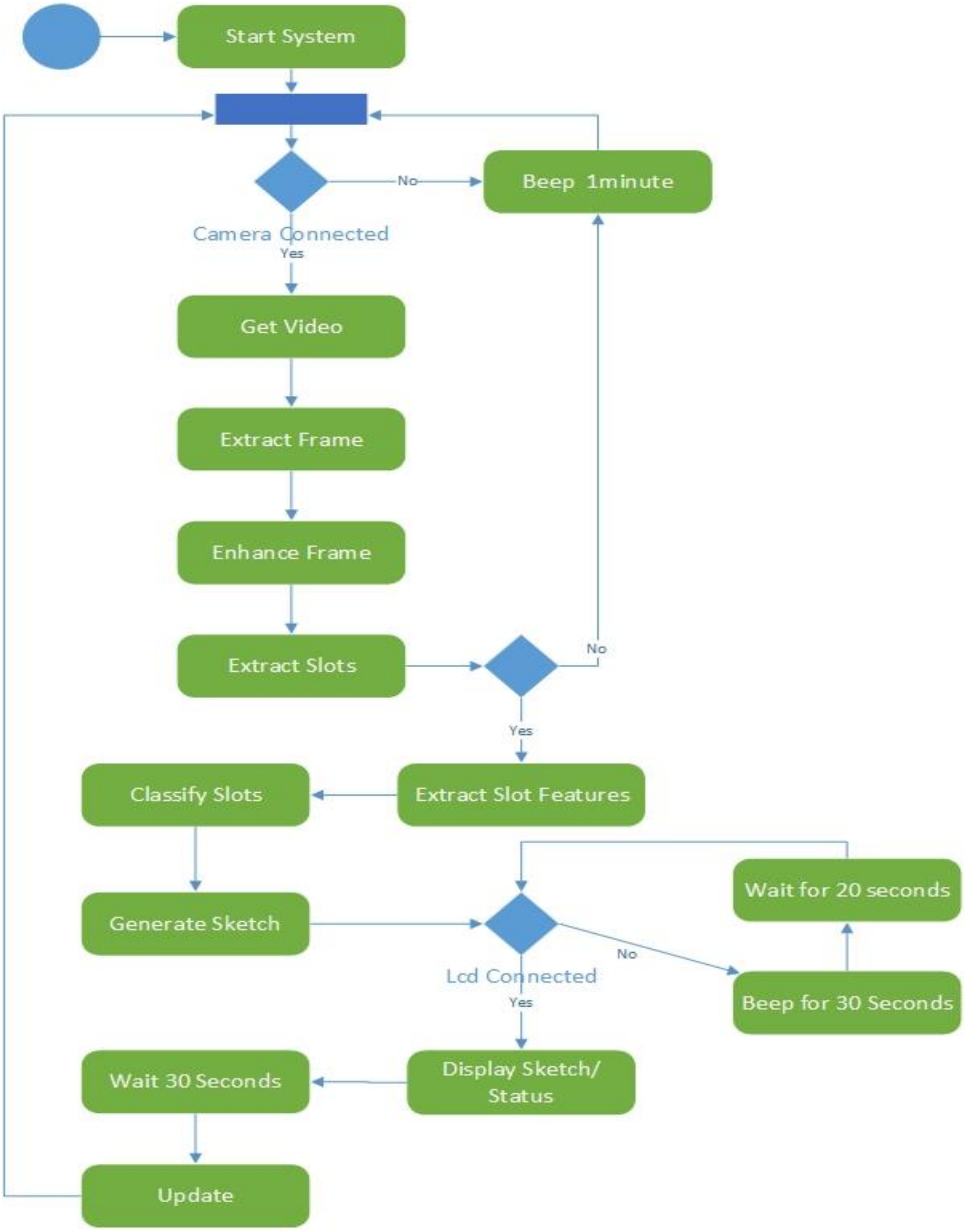


Figure 6 – Activity Diagram

Logical View (State Diagram)

Following is the state diagram of Help Me Park showing all the states that the system goes through during action.

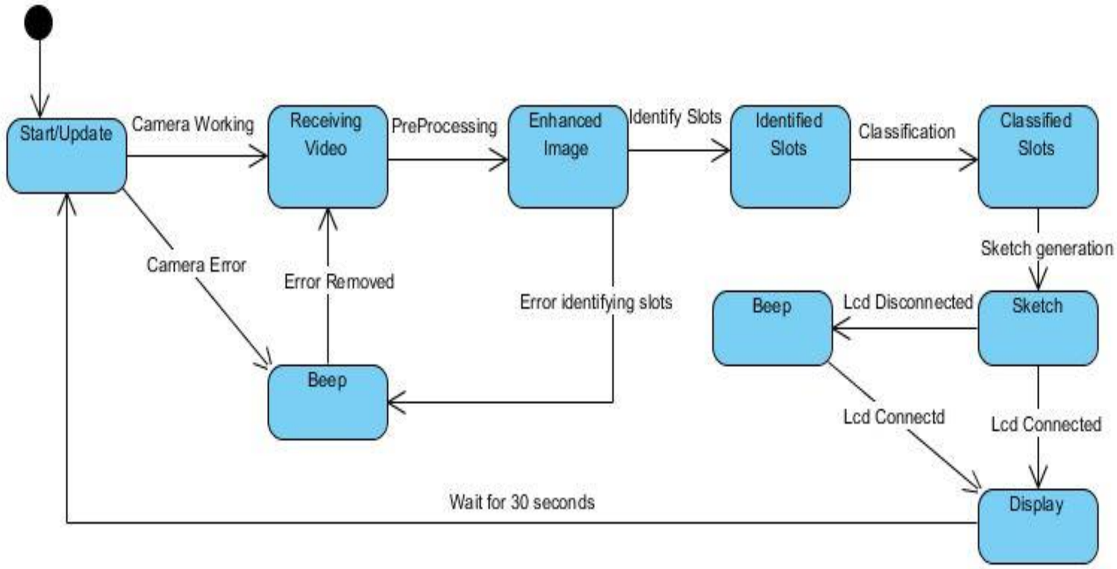


Figure 7 – State Transition Diagram

User Interfaces

Following diagrams show user interfaces and screens for Help Me Park App.

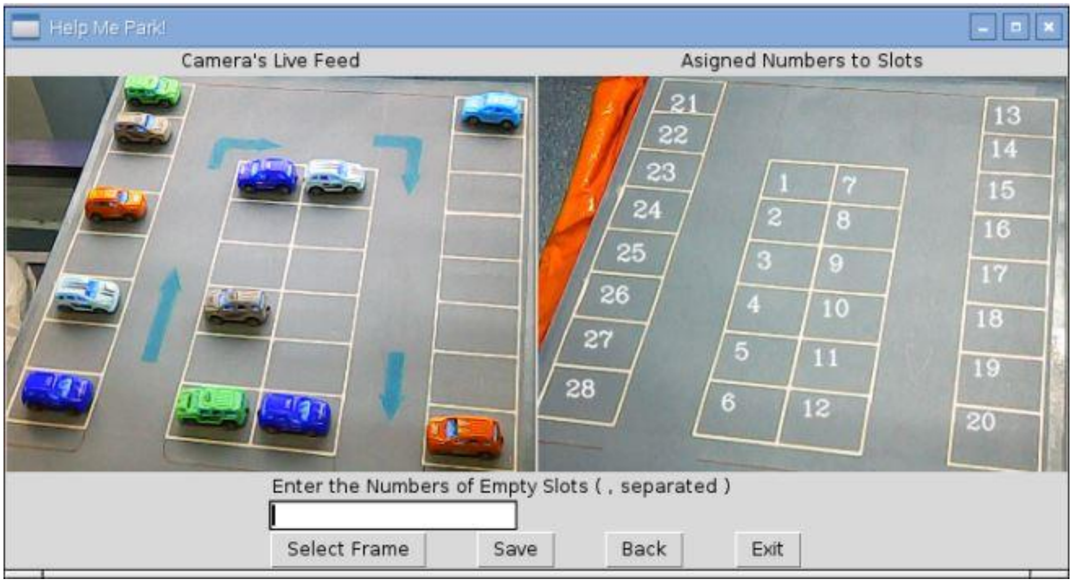


Figure 8 – Graphical User Interface of Training Mode

Occupancy Map (Sample)

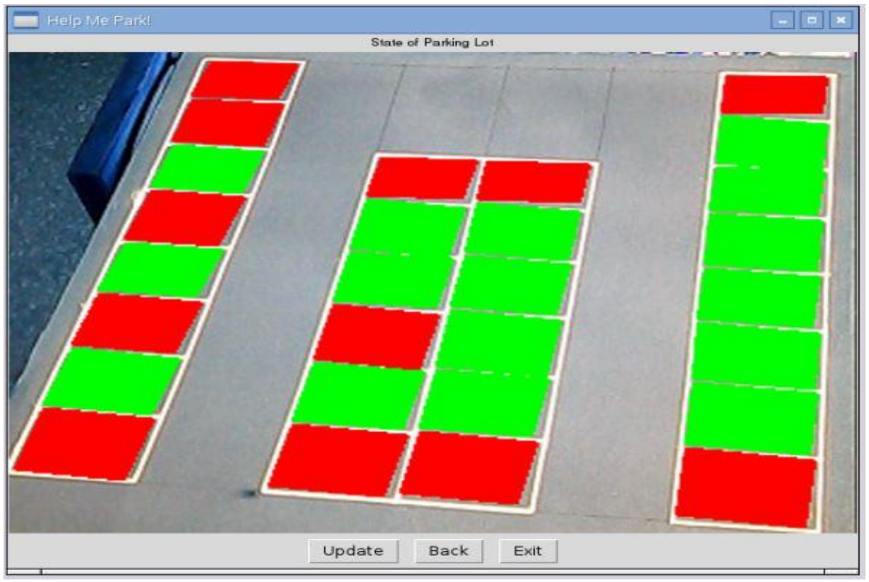


Figure 9 – Tentative Occupancy Map

4.3 Detailed Description of Components

This section describes in detail all the modules of Help Me Park. These modules have been assigned responsibilities. Modules are further sub classified into components.

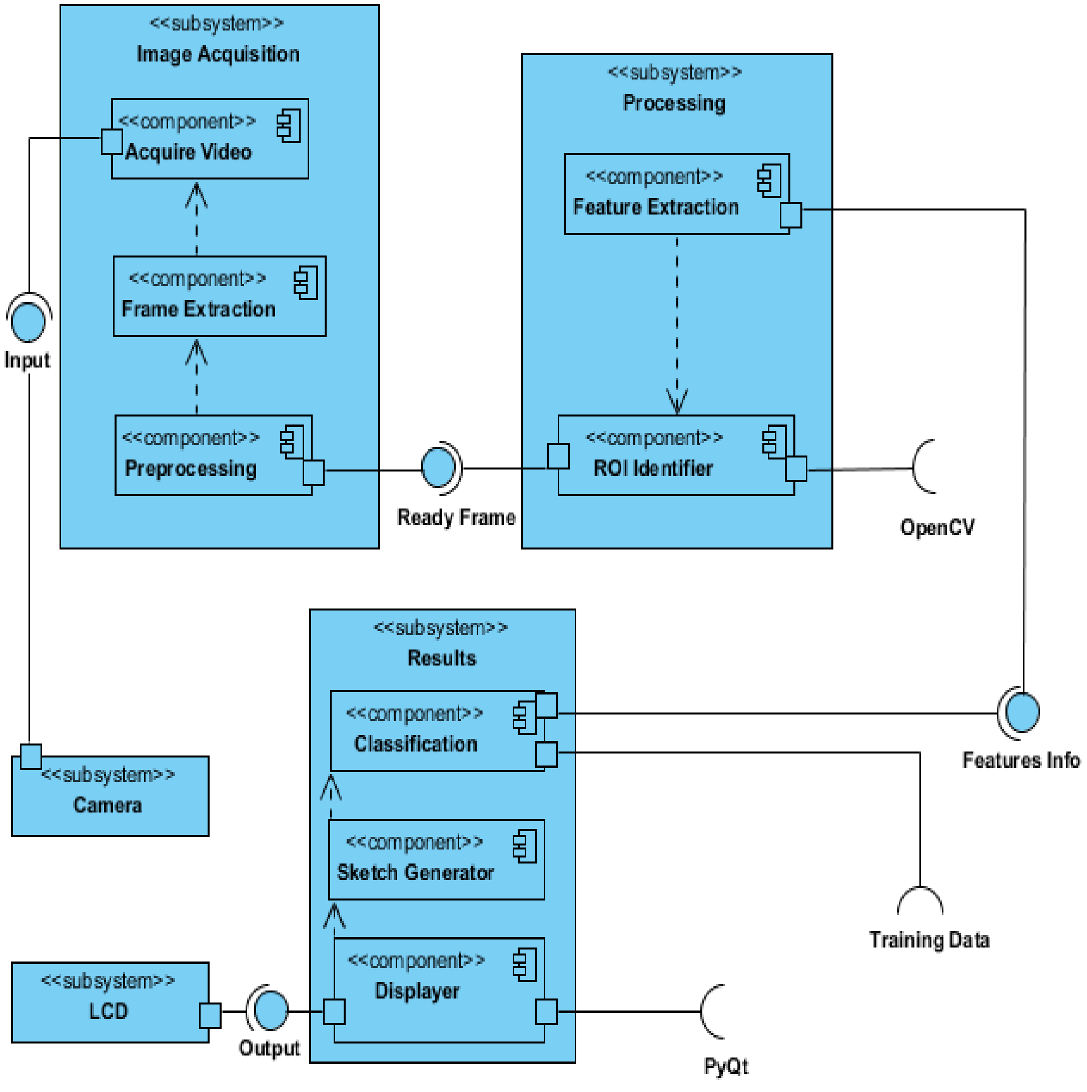


Figure 10 – Component Diagram

4.3.1 Image Acquisition Module

This module performs all the preprocessing for Help Me Park which includes receiving footage from camera, frame extraction and frame enhancement techniques. This module provides the base for successful working of other modules.

Acquire Video

Identification	Name: Acquire Video Location: Image Acquisition Module
Type	Component
Purpose	This component fulfils following requirement from Software Requirements Specification Document: Video Acquisition Requirement The system shall be able to acquire footage of parking lot in real time for further processing. Description This feature enables the system to acquire video from camera mounted in parking lot. This video will be fed into the system for further processing.
Function	This component of system interfaces with camera to obtain footage for further processing.
Subordinates	It has two subordinates: Frame Extraction: Extract frames for processing Preprocessing: Calibration and enhancement
Dependencies	This component is independent module and runs in parallel to entire application.
Interfaces	This component has following interfaces: **Camera Interface:** For getting video input as live feed from camera.
Resources	**Hardware:** Camera, Raspberry Pi, Network cable

	Software: Raspberry pi camera interface
Processing	Acquire video component would receive real time footage of camera which will be used for further processing.
Data	This component uses following information of the application: - Time of recording as sequence identifier

Frame Extraction

Identification	Name: Frame Extraction Location: Image Acquisition Module
Type	Component
Purpose	This component fulfils following requirement from Software Requirements Specification Document: **Video Processing** **Requirement** System shall be able to effectively divide video in to frames. **Description** This feature identifies best images from acquired video as input for further operations.
Function	This component of system divides the video into sequence of images.
Subordinates	It has one subordinate: Preprocessing (Calibration and enhancement):
Dependencies	This component is dependent on Video Acquisition module.

Interfaces	None
Resources	**Hardware:** Camera, Raspberry Pi, Network cable **Software:** Raspberry pi camera interface
Processing	Video is sequence of frames coming at a certain number of frame per second. We need this module to transform a video to a sequence of frames.
Data	This component uses following information of the application: - Time of recording as sequence identifier

Preprocessing

Identification	Name: Preprocessing Location: Image Acquisition Module
Type	Component
Purpose	This component fulfils following requirement from Software Requirements Specification Document: **Video Processing** **Requirement** System shall be able to select best image from divided frames. **Description** This feature identifies best images from acquired video as input for further operations.
Function	This component of system performs preprocessing steps like calibration and image enhancement.

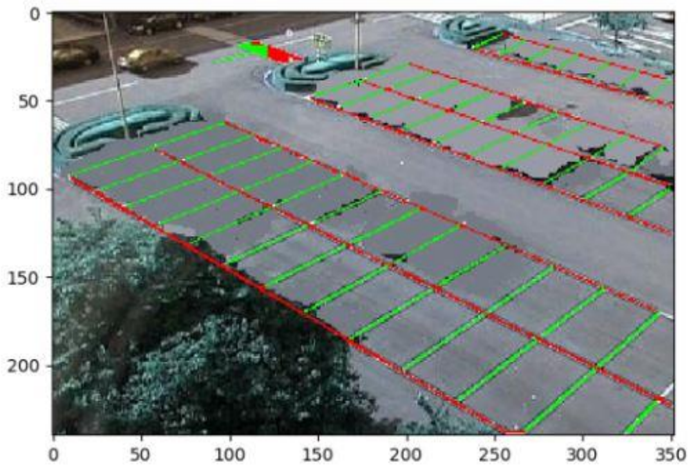
Subordinates	It has one subordinate: Slot Identification: Registering coordinates of slots with the system.
Dependencies	This component is dependent on frame extraction module.
Interfaces	None
Resources	**Hardware:** Camera, Raspberry Pi, Network cable **Software:** Raspberry pi camera interface
Processing	Video is sequence of frames coming at a certain number of frame per second. We need to enhance the frames for better results.
Data	This component uses following information of the application: - Time of recording as sequence identifier

4.3.2 Processing Module

This module performs all the processing related to identification and feature extraction of slots from an image. Feature extraction is the main input for classification module.

ROI Identifier

Identification	Name: ROI Identifier Location: Processing Module
Type	Component
Purpose	This component fulfils following requirement from Software Requirements Specification Document: **Identification of Slots** **Requirement**

	System shall be able to identify all the slots in parking area by interpreting boundary lines of slots defined on the ground. Description The images obtained from previous stage will be used to identify all the slots in parking lot. Regions of Interests are identified based on an image of parking lot that is empty with lines visible clearly. We used a combination of Depth-first-search and Breadth-first-search to traverse the image to locate points/coordinates of rectangular slot. Point slope formula is used to draw the rectangular region of parking slot for rectification of perspective distortion.  0 50 100 150 200 0 50 100 150 200 250 300 350
Function	This component of system will identify the objects in the image which, in our case, are line drawn on the ground. The information retrieved by this component would be used for registering image.
Subordinates	It has one subordinate: Slot Labeling/Classification: Based on features extracted, model will classify slots as vacant or occupied.
Dependencies	This component is dependent on image preprocessing because feature rich image would lead us to better results.
Interfaces	None

Resources	Hardware: Camera, Raspberry Pi, Network cable Software: OpenCV, Python
Processing	This component will identify objects in an image by using intensity based/feature based edge detection techniques.
Data	This component uses following information of the application: Time of recording as sequence identifier, coordinate information of slots in an image

Feature Extraction

Identification	Name: Feature Extraction Location: Processing Module
Type	Component
Purpose	This component helps in fulfilling the following requirement from Software Requirements Specification Document: **Slots labeling** **Requirement** System shall be able to label slots as available/occupied efficiently and correctly from identified slots (at 30 second refresh rate). **Description** Identified slots from previous stage will be labelled as available or occupied.
Function	This component of system will find and calculate the features of Regions of Interest.
Subordinates	It has one subordinate: Classification: labelling of slots

Dependencies	This component is dependent on ROI Identifier component.
Interfaces	None
Resources	Hardware: Camera, Raspberry Pi, Network cable Software: OpenCV, Python
Processing	Features are the characteristics which are used to classify slots as empty or occupied. This component will get these calculated for every slot.
Data	This component uses following information of the application: Time of recording as sequence identifier, Intensity levels/features of objects detected(lines)

4.3.3 Results Module

This module will conclude the working of other two modules by finally identifying the slots based on the features extracted.

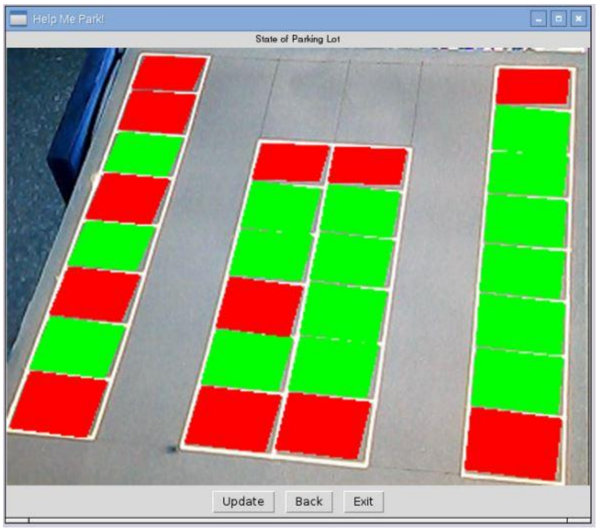
Classification

Identification	Name: Classification Location: Results Module
Type	Component
Purpose	This component helps in fulfilling the following requirement from Software Requirements Specification Document: **Slots labeling** **Requirement** System shall be able to label slots as available/occupied efficiently and correctly from identified slots (at 30 second refresh rate). **Description**

	Identified slots from previous stage will be labelled as available or occupied.
Function	This component of system will mine through the training data to determine the class of slot.
Subordinates	It has one subordinate: Sketcher: Map the slots to show occupancy state of parking lot.
Dependencies	This component is dependent on feature extraction component.
Interfaces	None
Resources	Hardware: Camera, Raspberry Pi, Network cable Software: OpenCV, Python
Processing	Features are the characteristics which are used to classify slots as empty or occupied. This component will take these features of every slot and determine it class, i.e. vacant or occupied.
Data	This component uses following information of the application: Time of recording as sequence identifier, Intensity levels/features of objects detected(lines), status/class of parking slot.

Sketcher

Identification	Name: Sketcher Location: Results Module
Type	Component
Purpose	This component helps in fulfilling the following requirement from Software Requirements Specification Document:

	Slots labeling **Requirement** System shall be able to combine all labeled slots to make fine sketch of parking area. **Description** Labelled slots received will be processed to create sketch of complete parking area which will be sent for display. This component will use point-slope formula to reconstruct the slots as occupied / available.  Help Me Park! State of Parking Lot Update Back Exit
Function	This component of system will map the ROIs to make a sketch showing the status of every parking slot.
Subordinates	It has one subordinate: Display: Displaying the screen so driver can see.
Dependencies	This component is dependent on Classification component.
Interfaces	None
Resources	Hardware: Camera, Raspberry Pi, Network cable Software: OpenCV, Python

Processing	All the ROIs identified as vacant or occupied are now needed to present in a user-friendly way. This component will create a sketch in a way that vacant and occupied slots are clearly distinguishable.
Data	This component uses following information of the application: Coordinate details of slots, Status of slots represented on a image(sketch).

Displayer

Identification	Name: Displayer Location: Results Module
Type	Component
Purpose	This component fulfills the following requirement from Software Requirements Specification Document: **Display Slots** **Requirement** Right sketch of parking area containing information of occupied and available slots shall be displayed by the system. Req. [4.6] **Description** Sketch of parking area is displayed on an LCD at entry gate from where driver can easily find available slots and route to them.
Function	This component of system will display the sketch in GUI consequently onto the LCD.
Subordinates	None
Dependencies	This component is dependent on Sketcher component.
Interfaces	None

Resources	Hardware: Camera, Raspberry Pi, Network cable, LCD Software: OpenCV, Python
Processing	Features are the characteristics which are used to classify slots as empty or occupied. This component will get these calculated for every slot.
Data	This component uses following information of the application: Coordinate info.

4.4 Reuse and Relationships to other Products

Help Me Park implements the intelligent parking space detection mechanism by first registering the image for regions of interest and then extracting features for classifying slots as vacant or occupied. Various systems for vacant space detection have been proposed using sensor based systems but these systems are difficult to install and maintain.

Help Me Park is simple and install to go system that can reduce the effort required to look for an available slot in a densely-populated parking area.

4.5 Design Decisions and Tradeoffs

Help Me Park is component based system that is driven by demand. Every component has been assigned with the responsibility to do a task. Mainly there are three modules; video acquisition module will acquire the footage and divide into frames. These frames are then needed to be enhanced to be processed. Next module will register the regions of interest that are the slots in our case. Finally, features are extracted from ROIs and then classified as vacant or occupied.

Clearly, components can do their work independently, but, in a certain flow (data as well as control). That lead us to **high cohesion.**

Moreover, component don’t have much interaction, once a component has completed its work system will generate an event for further action, consequently, the component registered for that event will come into action. This leads us to **low coupling**.

System’s control flow, data flow, high cohesion and low coupling lead us to Event Driven architecture (Implicit Invocation).

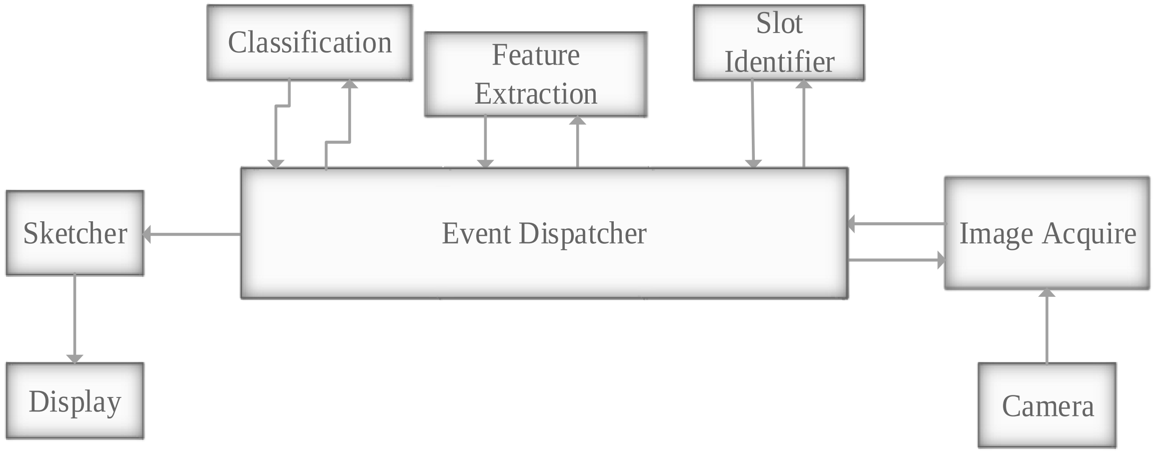


Figure 11 – System Architecture Diagram

Chapter 5. Project Test and Evaluation

5.1 Introduction

This chapter provides the test documentation for project Help Me Park, version 1.0 that will facilitate the technical tasks of testing including the detailed test cases for black box testing. Each test case specifies who will be performing the test, the preconditions required to execute each test case, the specific item to be tested, the input, expected output or results, and procedural steps where applicable. By providing detailed test information, we hope to reduce the probability of overlooking items and improve test coverage. Testers will be able to use each test case provided in this document to move forward and begin testing.

5.2 Test Items

Based on the requirements of project Help Me Park following are the major modules/ functionalities that should be considered during the testing process: -

- Video Acquisition
- Video Processing
- Identification of Slots
- Slots Labeling
- Mapping the Parking Lot
- Display Slots

5.3 Features to Be Tested

Following features are being tested:

1. The system shall be able to acquire footage of parking lot in real time for further processing.
2. System shall be able to effectively divide video in to frames.
3. System shall be able to select best image from divided frames.
4. System shall be able to identify all the slots in parking area by interpreting boundary lines of slots defined on the ground.
5. System shall be able to label slots as available/occupied efficiently and correctly from identified slots (at 5-7 second refresh rate).
6. System shall be able to label slots as available/occupied efficiently and correctly from identified slots (signal from external system).

7. System shall be able to combine all labeled slots to make fine sketch of parking area.
8. Right sketch of parking area containing information of occupied and available slots shall be displayed by the system.

5.4 Assumptions

Item Pass/Fail Criteria

Details of the test cases are specified in the section Test Deliverables. Following the principles outlined below, a test item would be judged as pass or fail.

- Preconditions are met
- Inputs are carried out as specified
- The result works as what specified in output => Pass
- The system doesn't work or not the same as output specification => Fail

Suspension Criteria and Resumption Requirements

Testing will be suspended when a defect is introduced/found that cannot allow any further testing. Testing will be resumed after defect removal.

5.5 Detailed Test Strategy

The project Help Me Park is a computationally intensive system that is why systems modules should be developed independently and then these modules should be integrated. Overall strategy comprises of Unit Testing using White Box and Black box testing. Integration testing is performed to successfully integrate the system.

Unit Testing

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test functions or code modules. The unit test cases shall be designed to test the validity of the programs correctness.

White Box Testing

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level in functions and the results are compared according to requirements. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. The test cases that have been generated shall cause each condition to be executed at least once. To ensure this happens, we are applying

Basis (alternative) Path Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.

Black Box Testing

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would.

Integration Testing

Integration testing is the part where we will test all the previous tested modules in a way that they are functioning normally when they are combined.

Incremental Testing

There are five primary modules that are required to be integrated. These components, once integrated, will form the complete application testing. The following describes these modules as well as the steps that will need to be taken to achieve complete integration. We will be employing an incremental testing strategy to complete the integration. The integration testing will be performed by the development team.

Image Acquisition

This is the module from where the major functioning of application initiates. This module takes a frame from video enhances its quality and pass this frame to slot identifier module. This module is developed independently and the tested first separately and then combined with Slot Identification module, by passing image to Slot Identification module.

Slots Identification Module

When Image Acquisition module has completed its work and an enhanced frame has been obtained from video sequence, this module will identify the slots (Regions of Interest) from the frame based on lines drawn on ground. This module is developed independently and the tested first separately and then combined with Slot Classification module.

Slots labeling / Classification Module

Classification module will classify the slots as vacant or occupied based on the processing through feature extraction and trained model. This module is developed independently and the tested first separately and then combined with Slot Identification, Image Acquisition and sketch generation module.

Sketch Generation Module

Once slots are identified and successfully labeled as vacant or occupied the next step is to convey it to driver who intends to park. This module will generate a sketch from the slots identified as vacant or occupied and this will be displayed to driver through GUI. Functionality of this module.

Display Module (GUI)

This module will display the up to date sketch over to LCD from where drivers would be able to choose which slot to park. GUI is designed independently and then integrated with whole system.

5.6 System Testing

In the end, system testing will ensure that all the modules are working, separately and together combined. Then only the outcome of the program will decide the correctness of whole system.

Performance testing

This test will be conducted to evaluate the fulfillment of a system with specified performance requirements. It will be done using black-box testing method. And this will be performed by:

- Checking out the response time of the system
- Memory management of the program

5.7 Test Deliverables

Following are the test cases:

Test case name	Acquire Footage
Test Case Number	1
Description	This feature enables the system to acquire video from camera mounted in parking lot. This video will be fed into the system for further processing.
Testing Technique used	White Box Testing
Preconditions	Camera should be configured properly with the raspberry pi.
Input	Camera's footage feeding into this module
Steps	Open the video capture mode using OpenCV function
Expected output	Function returned True and populate the list with frames provided as argument
Alternative path	**Cause:** Function returned False **Corresponding Output:** Error Message Displayed
Actual output	Confirmed

Test case name	Select Best Frame
Test Case Number	2
Description	This feature identifies best images from acquired video as input for further operations.

Testing Technique used	White Box Testing
Preconditions	Frames have been successfully fed in to this function, Test Case 1 satisfied
Input	Frames obtained from Acquire footage module as input
Steps	Convert to gray scale image, apply Laplacian filter and calculate variance
Expected output	Image with highest variance (least blur) is selected and returned
Alternative path	N/A
Actual output	Confirmed

Test case name	**Enhance Image**
Test Case Number	3
Description	Image is pre-processed before any further computation
Testing Technique used	White Box Testing
Preconditions	Test Case 1, 2 are satisfied
Input	Frame to be enhanced
Steps	Calculate histogram and apply gamma correction if image is brighter or darker than usual
Expected output	Enhanced Image
Alternative Path	N/A

Actual output	Confirmed

Test case name	Identification of Slot
Test Case Number	4
Description	The images obtained from previous stage will be used to identify all the slots in parking lot.
Testing Technique used	White Box Testing
Preconditions	Slots are registered from an image of properly laid out parking slot. Test Case 1-3 satisfied
Input	Enhanced Frame
Steps	1. Using Breadth First Search find the rectangle corners 2. Apply the depth first search in order to find all rectangles
Expected output	Populates the list with slot objects passed to this function with coordinates of each slot.
Alternative Path	**Cause:** Function returned False **Corresponding Output:** Error Message Displayed
Actual output	Confirmed

Test case name	Classification of Slots
Test Case Number	5

Description	Identified slots from previous stage will be labelled as available or occupied.
Testing Technique used	White Box Testing
Preconditions	Slots have successfully identified and fed into the classification function. Test Case 1-4 satisfied
Input	Slots objects List
Steps	Using weighted average of KNN, ANN and Decision Tree algorithm, predict the class of every slot.
Expected output	Slots are labeled
Alternative Path	N/A
Actual output	Confirmed

Test case name	**Sketch Generation**
Test Case Number	6
Description	Labelled slots received will be processed to create sketch of complete parking area which will be sent for display.
Testing Technique used	White Box Testing
Preconditions	Every slot's label is predicted successfully Test Case 1-5 satisfied
Input	Slot objects list

Steps	Show the corresponding slot on the sketch by red or green color based on occupied or vacant state respectively
Expected output	Sketch should be generated
Alternative Path	N/A
Actual output	Confirmed

Test case name	Graphical User Interface
Test Case Number	7
Description	This module is related to the design of dynamic and responsive user interface.
Testing Technique used	Black Box Testing
Preconditions	Nil
Input	Program Initiation will be treated as input to the system
Steps	Create and layout the widgets in the root window
Expected output	Main Screen displayed on screen
Alternative Path	N/A
Actual output	Confirmed

Test case name	Camera Posture Adjustment
Test Case Number	8

Description	This test case tests the functionality of camera setting option in which user will be displayed with the footage for proper alignment of camera mounted in parking lot with the system.
Testing Technique used	Black Box Testing
Preconditions	Camera is interfaced properly and Test Case 1 through 7 are already satisfied
Input	Operational Button is clicked
Steps	Fetch the frame and update in GUI
Expected output	Screen with image is displayed on screen
Alternative Path	**Cause:** Camera 's footage could not be fetched and displayed **Corresponding Output:** Image will not be visible rather a text 'Image will be displayed here' displayed.
Actual output	Confirmed

Test case name	Display State
Test Case Number	9
Description	This test case evaluates the functionality of system for displaying real time state of parking lot.
Testing Technique used	Black Box Testing
Preconditions	Camera is already calibrated
Input	Start Button is clicked

Steps	Process the frame and update in GUI
Expected output	Screen with image and corresponding state is displayed on screen
Alternative Path	**N/A**
Actual output	Confirmed

Test case name	**Refresh State**
Test Case Number	10
Description	This test case tests the system's functionality in which state is update at 5 – 6 second refresh rate so driver is always entertained by up-to-date information.
Testing Technique used	Black Box Testing
Preconditions	Camera is interfaced properly and Test Case 1 through 9 are already satisfied
Input	Internal Refresh Trigger
Steps	Fetch the frame and update in GUI
Expected output	Screen with image is displayed on screen
Alternative Path	**Cause:** Camera 's footage could not be fetched and displayed **Corresponding Output:** Image will not be visible rather a text 'Image will be displayed here' displayed.
Actual output	Confirmed

Test case name	Exit
Test Case Number	11
Description	This test case tests the functionality of exit button displayed on every screen.
Testing Technique used	Black Box Testing
Preconditions	Test Case 7-9 are satisfied
Input	Exit Button is clicked
Steps	Nil
Expected output	System will exit.
Alternative Path	N/A
Actual output	Confirmed

5.8 Test Environment

Hardware

- Raspberry Pi 3 Model B
- Camera
- LCD

Software

- Raspbian jessie up and running on Raspberry Pi
- Python 3.4.0 installed
- OpenCV 3.0.0
- Scikit-Learn installed

Staffing and Training Needs

Basic knowledge of testing strategies and techniques is needed for the testing of project. Techniques such as Black Box testing, integration testing should be known to developers. All the developers will be testing each other's work and will be actively participating in the development and testing of the project simultaneously.

5.9 Risk and Contingencies

Efforts have been made to remove all and every chance of failure but there are certain unpredictable factors such as network issues, corrupt input data, or system failure that may lead to some issues. Error handling will be applied more deeply to cover all these issues but unforeseen circumstances may happen.

Schedule Risk

The project might get behind schedule. So, to complete the project on time, we will need to increase the hours/day.

Budget Risk

The budget will be compensated by using less costly alternatives to fit the budget requirements.

Chapter 6. Future Work

We plan to use this system in parking garages even though it was only tested on outside parking lots. Due to the lack of space, more cameras would be needed to cover a large range of lots. Furthermore, the system's parameters must be adopted to changed lighting conditions to gain a sufficient performance.
Further improvements can be achieved by minimizing the influence of adjacent cars parking left and right of the regarded parking lot. This already poses a problem on the current setup where the camera distance is still very large.
Therefore, we aim at a 3D-estimation of parking cars to obtain the occluded space of nearby parking lots. Additionally, we currently work on an embedded system. It will make it possible to create *smart cameras* which are already equipped with a small processing unit to calculate the occupancy map directly. This will enhance the portability of our system because this approach only needs a *WiFi* connection to provide the occupancy map to a server. A smart-phone application will then visualize the current car park situation.

References

[1] K. C. Mouskos, M. Boile, and N. Parker, “Technical solutions to overcrowded park and ride facilities, university transportation research center region 2,” City College of New York, Final Report FHWA-NJ-2007-01, 2007.

[2] T. Ristola, “Parking guidance system in tapiola,” in Proceedings of the IEEE Conference on Road Traffic Monitoring, 1992, pp. 195 – 198.

[3] J. Wolff, T. Heuer, H. Gao, M. Weinmann, S. Voit, and U. Hartmann, “Parking monitor system based on magnetic field sensors,” in Proceedings of the IEEE Conference on Intelligent Transportation Systems, 2006, pp. 1275–1279.

[4] D. B. L. Bong, K. C. Ting, and K. C. Lai, “Integrated approach in the design of a car park occupancy information system (COINS),” IAENG International Journal of Computer Science, vol. 35, no. 1, pp. 7–14, 2008.

[5] S. Funck, N. Mhler, and W. Oertel, “Determining car-park occupancy from single images,” in Proceedings of the IEEE Intelligent Vehicles Symposium, 2004, pp. 325–328.

[6] M. Y. I. Idris, Y. Y. Leng, E. M. Tamil, N. M. Noor, and Z. Razak, “Car park system: A review of smart parking system and its technology,” Information Technology Journal, vol. 8, no. 2, pp. 101 - 113, 2009.

[7] C.-C. Huang and S.-J. Wang, “A hierarchical bayesian generation framework for vacant parking space detection,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 20, no. 12, pp. 1770–1785, 2010.

[8] A. Nallamuthu and S. Lokala, “Vision based parking space classification,” Clemson University, Department of ECE, Report, 2008.

[9] N. True, “Vacant parking space detection in static images,” University of California San Diego, Report, 2007.

Appendix A. System Operational Requirements

Hardware Requirements

- **Camera Mounted in the Parking lot:** The camera is used to make video of the parking lot in real time.
- **Cable:** The data (video feed) will be transferred from camera using cable to the raspberry pi.
- **Raspberry Pi**: Software installed which will process the data obtained through the camera and identify the available space.
- **LCD Screens:** LCD Screens will be used to convey the available space.

Software Requirements

- Linux: Raspbian Jessie
- OpenCV 3
- IDE: Python IDLE