Implementation of Smart Parking System Using Image Processing

A dissertation presented in partial fulfillment of the requirements

For the

Bachelor of Science (Special) Degree

In

Computer Science

Department of Statistics and Computer Science,

Faculty of Science,

University of Kelaniya,

Sri Lanka.

P.M.D.S Amarasooriya

2019

ABSTRACT

This paper aims to present a smart parking system for parking space detection based on the image processing technique. The proposed method can identify the free parking slots and the system contains the shortest path algorithm to help drivers in finding the nearest vacant parking space. In this project, the camera is acting as a sensor. The reason behind using a camera is with an image, it can detect the presence of many vehicles at once. Depending on the area to be covered, one or more cameras are used to process video clips. Since there are no sensors employed, the mechanical and electronic functionality of the system is reduced to a great extent. The smart parking system reduces the stress, waste of time associated with the parking vehicles and makes managing such parking areas more economical. Yolo algorithm which belongs to the Convolutional Neural Network family is used to detect vehicles in the parking area. The coordinates of the parking area are taken to an XML file when the parking area is empty. That XML file is used to identify the parking area. This proposed system is more efficient and less complex for drivers.

Keywords-Smart parking system, Image processing, Parking space detection, shortest path algorithm, Convolutional Neural Network

DECLARATION

The work reported in this dissertation was carried out by me in the Department of Statistics and Computer Science, University of Kelaniya as a partial requirement for the B.Sc. (Special) Degree in Computer Science. It has not been in any degree in this University or any other institution.

Full Name of the Student : P.M.D.S Amarasooriya

Student Number : PS/2014/002

Signature Date

....

I certify that the above statement is correct.

Date

....

Dr. (Mrs) M.A. Anusha Jayasiri
Senior Lecturer (Grade II)
Information Technology Centre
University of the Visual and Performing Arts
No.21, Albert Crescent,
Colombo 07,
Sri Lanka.

ACKNOWLEDGEMENT

At the very first I take this opportunity to thank the research project supervisor Dr. (Mrs) M.A. Anusha Jayasiri for guiding me throughout my research project by giving advice, opinions & suggestions as well as by encouraging me. And, I would like to thank all the other lecturers and staff members for helping me to carry out my research.

Also, I must express my very profound gratitude to my parents, older brother for providing me with unfailing support and continuous encouragement throughout my year of study and through the process of researching and writing this thesis.

Finally, I must thank my university colleagues for being with me even in difficult situations by doing crazy stuff to release stress and continuous encouragement thought my year of study. Also, I would like to thank all who gave support to me to finish my research successfully.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

LIST OF ABBREVIATIONS

CNN- Convolution neural networks

CCTV- Closed-circuit television

HSV- Hue and Saturation Variation

CHT- Circle Hough Transform

MCU- Microcontroller unit for communication

FOV- Field of view

ROI - Regions of interest

XML - Extensible Markup Language

ID- Identifier

RPN - Region Proposal Network

UFPR -Federal University of Parana

PUCPR-Pontifical Catholic University of Parana

Chapter 1-Introduction

1.1 Background

With the ever-increasing urban population and the improvement in living standards, usage of the vehicles have increased. Because of this reason, occurrences of traffic congestion are high in metropolitan areas and it is difficult to find a free parking slot. Hence the need for parking areas is increasing. Even though the parking areas are high, drivers cannot find an available parking spaces due to the insufficiency of the proper status of the parking area. The following figure 1 shows the increasing usage of vehicles in Sri Lanka from 2008 to 2106 according to the department of motor traffic in Sri Lanka. (Department of motor traffic Srilanka, 2015)

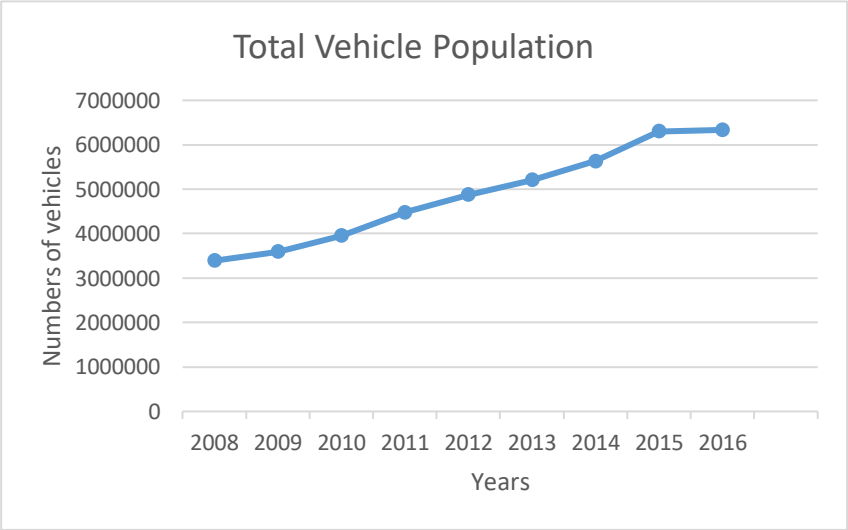


Figure 1: Total vehicle population in Sri Lanka

Nowadays most car parks are not run efficiently and most car parks systems have a manual process in Sri Lanka. Roads are full of traffic congestion as an average 250,000 vehicles, made up of 15,000 buses, 10,000 trucks and 225,000 private vehicles, enter Colombo daily. Because of that on busy days, drivers may take extra time driving around a car parking in order to find a free parking space. Failure to find a free parking slot can cause Congestion on the roads, CO_2 Emission, energy waste, road accidents and increase the stress level of drivers. Also, Vehicle owners often undergo an extremely frustrating process of driving. The main problem that always occurs at the car park is time being

wasted in searching for the available parking spaces. These problems are seriously affecting public health and wasting useful resources. The most urban cities the average search for the available parking place is up to 10 minutes. Every vehicle owner would wish to park the vehicle as closely as possible to his or her destination so as minimize his walking. (Edirisinghe, 2014)

To tackle these problems, the smart parking system can be implemented. The proposed system can identify the vehicle type and free parking slots. Also, the system contains the shortest path algorithm to help drivers in finding the nearest vacant parking space. In this project, the camera is acting as a sensor. The reason behind using a camera is with an image it can detect the presence of many vehicles at once. Also, I use machine learning algorithms to implement this system. Machine learning support for various fields like natural language processing, speech recognition, computer vision and etc.To implement this project I used yolov3 real-time object detection algorithm. This was followed by Faster R-CNN that used a RPN for identifying bounding boxes that needed to be tested.

1.2 Motivation Factors

1.2.1 **Resourcefulness**

In the early days processing a machine learning model is quite expensive compared to normal computer programs. Also, powerful hardware was hard to find or quite expensive. Early days it had been difficult to find good datasets. But in present researchers are able to find powerful hardware for cheap. Today, most of the systems of the world are automated and therefore their databases are vastly bigger. Researchers can find data sets from the internet easily. Kaggle, Imagenet, Quandle, YouTube, Facebook, Wikipedia are examples of online sources that provide free data of researchers. Now computers which are much cheaper and yet with more computation power. So it is much motivating to conduct a study using this approach.

1.2.2 Popularity

Machine learning has become famous today because the leading companies like Google, Facebook, YouTube and Amazon are using it to increase the functionalities of their

products. Also, Google Trends that tracks the popularity of search terms, suggests that searches for machine learning are about to out-pace the searches for artificial intelligence. So there many requirements that need to be addressed today with these technologies.

1.2.3 Accuracy

For this implementation Yolov3 object detection algorithm was used. Object detection is a task in computer vision that involves identifying the presence, location, and type of one or more objects in a given photograph. YOLO is based Convolutional Neural Network family of models for object detection. Since here we are using the Convolutional Neural Network approach we can check the accuracy from real world data before using the model for real world applications. Faster R-CNN has more accuracy. (Joseph Redmon, 2018)

1.2.4 Availability of Software and frameworks

Unlike the early days at present, there are lots of software and frameworks which support machine learning. Python has in-built packages such as pandas, numpy and scikit-learn etc covers the fundamentals so we can focus on the improvements without a hassle. MATLAB, OpenCV, Weka are good examples of software It makes easier to focus on the research, more than worrying about how to implement it.

1.3 Objectives

- Reduce the time of finding the parking slot.
- Reduce vehicle emissions and pollution.
- Provide better public service.
- Fuel-saving.
- Reduce high traffic congestion.
- Validation of vehicles that are entering through the parking area by using machine learning.
- Eliminates the unnecessary traveling of vehicles across the filled parking slots.
- Decreased Management Costs.
- Reducing stress while searching for a parking space.

1.4 Benefits of the system

- Cheaper compared to sensor-based systems.
- It can be used to avoid the collision in the parking area.
- Since open-source libraries are used, open to further improvements if required.
- It can be used for security purposes.

1.5 Overview of the Research

This research is done as a step to identify free parking slots before enter the parking area. Also, identify the vehicle type. For this purpose used real time object detection algorithm. The first step was to get the real time CCTV footage from a camera. Here I consider only one camera. Firstly, a camera placed in a fixed position above the vehicles will be used to acquire the image used to calculate the vacancy of the car parks. This camera should be in a position where can clearly see all the car parks and camera should be fixed more than ten feet above ground level because the mean pixel value of each parking slot must be captured any time. After identifying the vehicles in the parking area using the Yolov3 object detection algorithm. After getting the coordinate of different types of vehicles marked the bounding boxes. Compare the mean pixel value of each parking slot and coordinate of bounding boxes, can be determined the status of each parking slot. Compare the mean pixel value of each parking slot and coordinate of bounding boxes, can be determined the status of each parking slot. If the mean pixel value of the parking slot is inside the given bounding box then that parking slot is not empty. Otherwise empty. Finally, display the result entrance of the parking area.

Chapter 2- Literature Review

2.1 Overview of the Chapter

This chapter contains the knowledge gathered to conduct the research project. This Knowledge was gathered from studying theories of machine learning, studying existing methods for image processing and the ways for optimizing them.

2.2 Introduction

There are various methods to detect vehicles in a parking area. One method is using both Radio-Frequency Identification (RFID) and Internet of Thing (IoT) which can detect the available parking lot. This IoT based parking system is created by using controllers, sensors, servers, and cloud. Sensors will be placed in each parking slot to detect the presence of a car. The sensor will read the amount of vacant parking space and send the data to the microcontroller. Then the microcontroller will display the amount of vacant parking spot and display it to the LED display. Finally, drivers can be known available free parking slot connects to the servers before reaching the entrances of the parking area. Figure 2 shows the process of this system. (Aekarat Saeliw, 2019)

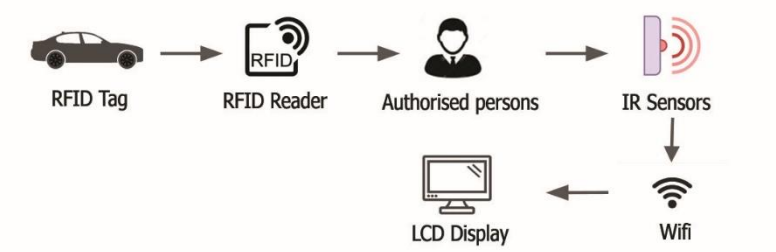


Figure 2: The architecture of Detecting Empty Parking Slot

The problem of this system is sensors cannot detect the presence of the vehicle in the bad weather condition. Each slot must have a sensor and having sensors in every parking slot is not cost-effective. Also, it is difficult to maintain a sensor-based parking system. There

is another system based on the effect of the Earth's magnetic field of magnetic sensors is harnessed to determine the vacancy status of parking spaces. (J. Wolff, 2006)

Another way is to identify the parking slot using image processing technologies. This system consists of some operations which are system initialization, image acquisition and processing, data interpretation and display updates. Cameras taking snapshots and passing them to the microcontroller unit (MCU) for communication at regular intervals. The controller forwards the collected data to the base station over the setup local area network. At the base station, the received images undergo a preprocessing stage. Using the method of Luminosity, Canny edge detection and other techniques display the result. The accuracy of this system depends on information that captured images of the parking space. (Benjamin Kommey, 2018) For implementation, this system used Yolov3 real time object detection algorithm.

2.3 Related Works

This project discusses image processing. Typically, this analysis is done by looking at the difference between consecutive video frames. It can be easily changed the region that a camera will scan by simply changing the camera's location. There are two ways to use this device either by applying the edge detection method for image detection module using boundaries condition method or by applying point detection (G and R 2016) using the canny operator method. Edge detection is the process by which sharp discontinuities are detected and located in an image. The discontinuities are sudden shifts in pixel intensity that define object boundaries in a scene. Edge detection is one of the most widely used digital image processing techniques. The boundaries of object surfaces in a scene frequently result in locally-focused changes in intensity. Edge detection (Igbinosa 2013) in noisy images is difficult to implement, as both noise and edges contain high-frequency content. Detection of the parking lot is achieved by defining the green rounded picture of each parking lot. Matlab is used as a base for the applications. If the device has adequate processing power to process color images quickly and efficiently, the RGB value can be used to determine where green circles are representing empty car spaces.

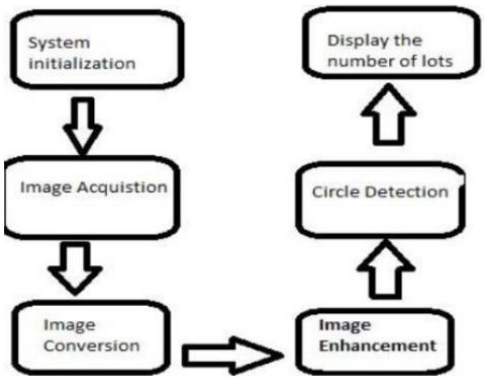


Figure 3: The process flowchart

Image conversion is performed to simplify processing while defining the various objects in the image. One way is to take a sample of the RGB image and see in a 3-dimensional graph where clusters are formed. Since this method is not feasible for each image frame in the framework, there is a need to consider another technique for this reason. A simpler and more convenient method was found to be to find the green dots by using the HSV. Detection of circles is important for the system as the number of circles is equal to the number of empty parking lots. Hough transform is used primarily in different image processing systems for the identification of circles. The CHT is a method for the extraction of features to detect circles. The number of circles implies the number of vacant spots.

This is another study of the smart parking system. For easy deployment of parking detection nodes, the parking area is logically demarcated into major sections. Every major area segment is fitted with a node for detection. A node consists of wide-field view cameras to take snapshots of the allocated minor parking segment at regular intervals and compliant IEEE 802.11 b /g /n MCU between the node and the base station. As shown in Figure 4, a node is mounted at a higher altitude close to the edge of its allocated section of parking so that the node overlooks the area and the cameras can capture their individual ROIs or small parts in their FOV.

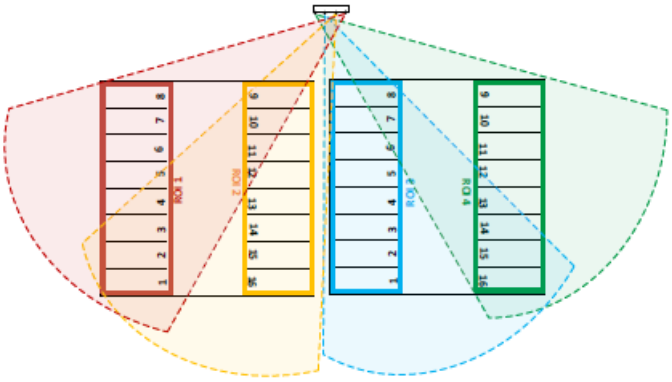


Figure 2.3.2: Camera positions with their respective FOVs and assigned ROIs.

The control unit of each node is built around the TI CC3200 microcontroller which operates on an ARM Cortex-M4 core with an industry-standard running at 80 MHz's The cameras are connected to the MCU through the peripheral interface of the strong parallel camera chip. The device of the image sensor receives recorded images from the camera as soon as it is shot. The controller forwards the collected data to the base station over the setup local area network. (Kommey, O., and S. 2018) The aerial cameras capture images while the field is empty from their assigned portion of the parking lot. The images are transmitted from the cameras to the MCU via the fast parallel image sensor interface. The MCU bundles the images received into TCP packets and transmits the data from the configured on-chip web socket (RFC 6455) HTTP server to the Web socket client application via Wi-Fi to the base stations. The images are extracted from the obtained packets and processed using the image processing algorithm mentioned in this paper's later section. Throughout subsequent image processing, coordinates of the designated individual parking slots and their boundaries are collected and retained for reference. These coordinates could be used as references as long as positions and orientations of the nodes are fixed.

2.3.1 Overview of Related Work

Most of the research used a canny edge detection algorithm for identifying the free parking slots. Parking areas in binaries images filled are marked with bounding regions based on the coordinates obtained from the initialization process. In each area, the pixel count is calculated and compared to the initial values stored. In the marked regions, parking spots with cars would have extra edges which increase the mean intensity within the regions. A substantial increase in the pixel count of an area allows the corresponding spot to be filled. This occupancy information is then sent to the display devices from the base station for incoming drivers to use.

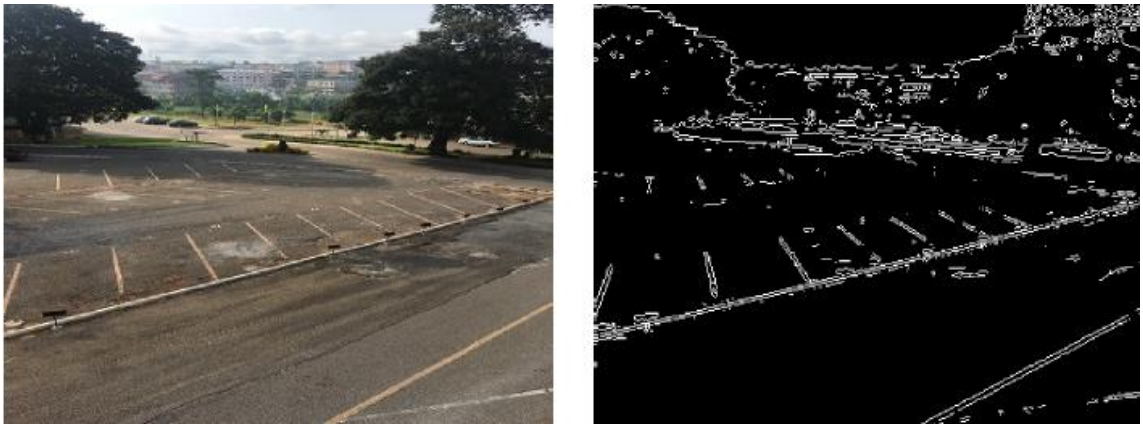


Figure 2.3.1.1: Strong edges identified using the canny detection algorithm

2.4 Machine Learning

Machine Learning has become one of the foundations of information technology over the past two decades, and with that, a rather central part of our lives, though usually hidden. With that amount of data being usable, there is reason to believe that smart data processing will become even more omnipresent as a necessary ingredient for technological advancement. Most security systems use face recognition as one of their elements, e.g. for access control. That is to know who this person is, provided a person's picture or video recording. In other words, the program has to assign the faces into one of several groups or conclude that they are an unknown face. The question of verification is related, but conceptually very different. (Baştanlar and Özuysal 2014) Even though ML is a computer science field, it differs from traditional approaches to computing. Algorithms are collections of specifically coded instructions used by computers to measure or solve problems in conventional computing. Alternatively, Machine Learning

algorithms allow computers to train on data inputs and use statistical analysis to produce values within a specific range. Machine learning, therefore encourages computers to create models from sample data to automate data-based decision-making processes (Tagliaferri, 2017).

2.4.1 Machine Learning With Image Processing

Images have always played a major role in human life as vision is probably the most important of human beings. As a result, there are numerous applications in the field of image processing. Nowadays and more than ever, images are everywhere and, thanks to the advances in digital technologies, it is very easy for everyone to generate a huge amount of images. Modern image processing methods have to deal with more complex issues with such image a profusion and have to be adaptable to human vision. Machine learning has emerged as a key component of smart computer vision systems when adaptation (e.g. face recognition) is required with vision being complex. (Lézoray et al. 2008)

Today, computers can not only identify images automatically but can also define the different elements in pictures and write short sentences explaining each section. The Deep Learning Network does this, which simply discovers patterns that occur naturally in images. ImageNet is one of the largest collections of labeled images to train Convolutionary Neural Networks using GPU-accelerated deep learning frameworks such as Caffe2, Chainer, Microsoft Cognitive Toolkit, MXNet, PaddlePaddle, Pytorch, TensorFlow, and TensorRT inference optimizers. (Image Processing With Deep Learning- A Quick Start Guide)

2.4.2 Canny Edge Detection

Canny Edge Detection is a common algorithm for edge detection. It was designed in 1986 by John F. Canny. It's a multi-stage algorithm, and we're going to go through every step. The purpose of edge detection, in general, is to significantly reduce the amount of data in

an image, while preserving the structural properties to be used for further image processing. The algorithm runs in 5 separate steps.(Malathy H Lohithaswa 2015)

- Smoothing

  Because edge detection is susceptible to noise in the image, with a 5x5 Gaussian filter, the first step is to remove the noise in the image.

- Finding gradients:

  The smooth image is then filtered in both horizontal and vertical directions with a Sobel kernel to get the first horizontal (G_x) and vertical (G_y) derivative. From these two images, the edge gradient and direction can be found as follows for each pixel.

$$\text{Edge\_Gradint}(G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle(\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

  The direction of the gradient is perpendicular to the bottom. It is rounded to one of four vertical, horizontal and two diagonal angles.

- Non-maximum suppression

  After the magnitude and direction of the gradient, a complete image scan is done to remove any unnecessary pixels that may not be the bottom. For this, pixels are tested at each pixel if they are a local limit in their neighborhood in the gradient direction.

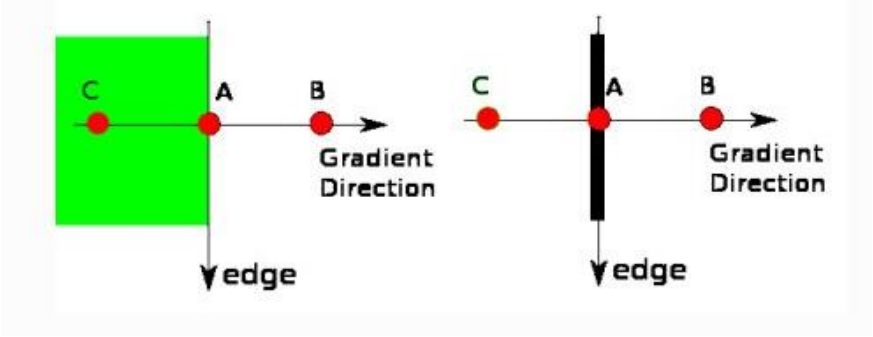


Figure 2.4.2.1: Illustration of non-maximum suppression.

Point A (in the vertical direction) is on the bottom. The direction of the gradient is natural to the bottom. Points B and C are in the direction of the gradient. So

point A is verified for a local limit with point B and point C. If so, the next stage will be considered otherwise, it will be suppressed (set to zero).In short, a binary image with "small edges" is the result you get.

- Hysteresis Thresholding

  This stage decides which edges are and which edges are not really. Two threshold values are required for this, minimum and maximum value. Those edges with more than maximum value gradient are sure to be edges, and those below minimum value are sure to be non-edges, so they are discarded. Those lying between these two thresholds are edges or non-edges classified based on their connectivity. They are considered part of the edges if they are connected to "sure-edge" pixels. Otherwise, they will be discarded as well. See the following picture.

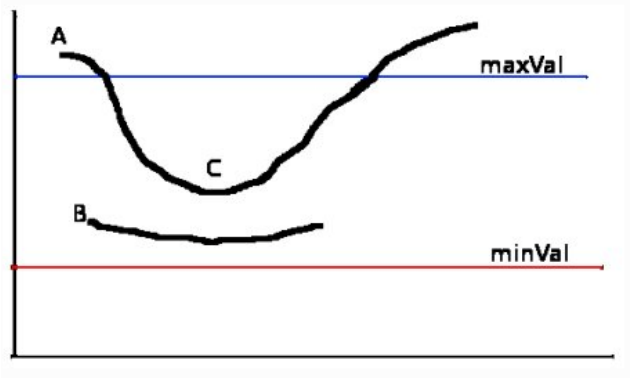


Figure 2.4.2.2: Hysteresis Thresholding Values

Edge A is above the maximum, which is therefore considered "sure-edge." Despite the fact that edge C is below maximum, it is connected to edge A, so it is still called a valid edge and we get the complete curve. Yet edge B, though above minimum and in the same region as edge C, is not connected to any ‘sure-edge’, so it is discarded. So it's very critical that we have to pick a minimum value and maximum value to get the right result. This stage also eliminates small pixel noises on the assumption that edges are long lines.

2.4.2.1 Advantages and Disadvantages of Canny Edge Detection

Table 1: Advantages and Disadvantages of Canny Edge Detection (Suryanshrao, 2017)

Advantages	Disadvantages
Detection of the Canny edge is adaptable to different environments.	The primary disadvantage of using Canny edge detector is that it consumes a lot of time due to its complex computation.
The effectiveness can be adjusted by using parameters.	It is difficult to implement to reach the real-time response.
Detects the edges in a noisy state by applying the thresholding method.	If the amount of smoothing required is important in the spatial domain it may be slow to compute.
The signal can be enhanced with respect to the noise ratio by non-maxima suppression method which results in one pixel wide ridges as the output.	It gives a bias towards vertical and horizontal edges and does not give a good approximation of rotational symmetry.

This method of tracking the dots based on their previous positions, and isolating them using edge detection and a flood fill has a number of drawbacks. This algorithm only tracks the dots, there is no provision for locating them within a scene. It is necessary to identify a point within each dot for the subsequent frame. This is predicted from the current frames computed center coordinates. The prediction algorithm used is quite simple: repeat the translation between the previous two frames. To account for changes in direction, and camera zoom would increase the complexity of the motion prediction considerably. Because of this reason canny edge detection is more complex and not suitable for real time video processing.

CHAPTER 3 – MATERIALS AND METHODS

3.1 Overview of the Chapter

This chapter contains the process of systematically planning the study. It also explains the concepts, algorithms, tools and frameworks that used this model to build.

3.2 Tools and Frameworks

3.2.1 Overview

Python is the programming language used in the Anaconda Navigator platform to build this project. Since this work is about machine learning, the python language has also made it easy to code. Python has several predefined libraries to ease coding Jupiter Notebook is the IDE used for writing python scripting. ASUS F556U Laptop Computer was used for hardware purposes in combination with all the above-listed equipment. A concise description of the resources and tools used in this research is given below.

3.2.2 Python

Python is a versatile programming language that's easy to learn. It has powerful, high-level data structures and an object-oriented programming style that is simple but effective. The elegant syntax and intuitive typing of Python, together with its interpreted nature, make it an ideal language for scripting and quick application development on most platforms in many areas.(Rossum and Drake 2010) Python also encourages the use of modules and bundles, ensuring that applications can be constructed in a modular manner, and code can be reused across a variety of projects. Once you have built a module or kit that you need, it can be scaled to be used in other projects and these modules can be easily imported or exported. Python is a programming language for general purposes, which is another way of saying it can be used for almost everything. Perhaps notably, it is an interpreted language, meaning that the written code at runtime is not directly converted into a computer-readable format. While this conversion is performed by most programming languages before the program is even running. Flowing figure shows the features of python language. (DataFlair, 2019)

Figure 4: Features of Python Language

- Easy to Code

  Coding in Python is simpler compared to other common languages such as Java and C++. Within just a few hours, everyone can know the syntax of Python. Mastering Python, though certain, requires learning about all of its advanced concepts and packages and modules. It takes time to do that. It is therefore programmer-friendly.

- Easy to Read

  Python programming is like English, being a high-level language. When someone look at it, he or she can say what to do with the file. It also requires indentation because it is dynamically typed. It helps to make information readable.

- Free and Open-Source

  Python is free to download from the website of Python. It's also open-source. It ensures that the public has access to its source code. It can be downloaded, modified, used and distributed. This is called FLOSS (Free / Free and Software for Open Source). As the world of Python, we're all moving toward one goal a Python that's getting better.

- Portable

  There's no need to write specific machine code. It makes Python a language that is portable. Nonetheless, in this case you need to avoid any system-dependent features.

3.2.3 Python Libraries

Some of the essential libraries that have been used throughout the project have been listed in this section.

3.2.3.1 Time

Python has a time-related feature to perform tasks. We must first import the module to use the functions specified in the module. The following figure shows the example of the time library. (Labs, n.d.)

```
In [7]: import time
        print("This is printed immediately.")
        time.sleep(2)
        print("This is printed after 2 seconds.")

        This is printed immediately.
        This is printed after 2 seconds.
```

Figure. 5: Time library code example

Essential facts about the time module describe follows.

- The time module wraps around the C runtime library to provide functions for time manipulation. It has a set of functions that you can call to deal with the system time.
- This time the library follows the EPOCH convention that applies to the starting point of the time. This supports the dates and times between the period and the year 2038.
- The UNIX EPOCH will be at midnight on January 1, 1970, at 12:00. Use the code below to calculate the EPOCH value on your machine.

3.2.3.2 Numpy

NumPy is python's open-source bundle. It can be used for scientific and numerical computing. It is mostly used to compute more effectively on arrays. It's written and based in both C and Python. It's a package of pythons and the term Numpy means Python number. It is mainly used to process the multidimensional array of homogeneity. Scientific computing is a core library. It, therefore has powerful multidimensional array objects and tools that are useful to work with these arrays. In almost every scientific python programming that includes machine learning, statistics, bioinformatics, etc., it is important. It offers some really good features, which is written very well and runs efficiently. It is mostly focused on performing mathematical operations on contiguous arrays similar to the arrays you have in languages of lower levels such as C. In other words, it is used in the manipulation of numerical data. (educba, 2019)The very first reason to pick the python numpy array is that it has less memory than the set. It is then very easy in terms of execution and at the same time working with numpy is very convenient.
Below are the python numpy operations. (aayushiedureka, 2019)

- ndim
  Used to find the dimension of array, whether it is a two-dimensional array or a single dimensional array.
- dtype
  Used to identify the element data form that is stored in the list. So, we want to know a particular element's data type, we can use the' dtype' feature to print the data type together with the size.
- itemsize
  Calculate the byte size of each element
- reshape
  Change the number of rows and columns which gives a new view to an object.
- Square Root & Standard Deviation
  There are various mathematical functions that python numpy can be used to run. We will consider the square root, the default array deviation.

```
In [5]:
1 import numpy as np
2 a = np.array([(1,2,3),(4,5,6)])
3 print(a.ndim)

2

In [6]:
1 a = np.array([(1,2,3)])
2 print(a.dtype)

int32

In [7]:
1 a = np.array([(1,2,3)])
2 print(a.itemsize)

4

In [8]:
1 a = np.array([(8,9,10),(11,12,13)])
2 print(a)
3 a=a.reshape(3,2)
4 print(a)

[[ 8  9 10]
 [11 12 13]]
[[ 8  9]
 [10 11]
 [12 13]]

In [9]:
1 a=np.array([(1,2,3),(3,4,5,)])
2 print(np.sqrt(a))
3 print(np.std(a))

[[1.         1.41421356 1.73205081]
 [1.73205081 2.         2.23606798]]
1.2909944487358056
```

Figure 6: Python Numpy Operations

3.2.3.2 OpenCv

OpenCV was introduced by Gary Bradsky at Intel in 1999 and the first version was released in 2000. To run the Russian OpenCV development team of Intel, Vadim Pisarevsky joined Gary Bradsky. For Stanley, the vehicle that won the 2005 DARPA Grand Challenge, OpenCV was used in 2005. Subsequently, under the sponsorship of Willow Garage, its active growth continued with Gary Bradsky and Vadim Pisarevsky leading the team. Right now, OpenCV supports a lot of computer vision and machine learning based algorithms and is expanding on day-by-day.(Mordvintsev and Abid 2017)OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language. All the OpenCV array structures are converted to-and-from Numpy arrays. So, OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems. The following functions are used to implantation this project.

- cv2.imread()

  Use this function to read an image. The image should be in the working directory or a full path of image should be given. Second argument is a flag which specifies the way image should be read.

  1. cv2.IMREAD_COLOR loads a color image. Any transparency of image will be neglected. It is the default flag.
  2. cv2.IMREAD_GRAYSCALE loads image in grayscale mode
  3. cv2.IMREAD_UNCHANGED loads image as such including alpha channel

  Instead of these three flags, you can simply pass integers 1, 0 or -1 respectively.

- cv2.imshow()

  This function is used to display an image in a window. The window automatically fits to the image size. First argument is a window name which is a string. Second argument is the image.

- cv2.waitKey()

  cv2.waitKey() is a keyboard binding function. Its argument is the time in milliseconds. The function waits for specified milliseconds for any keyboard event. If you press any key in that time, the program continues. If 0 is passed, it waits indefinitely for a key stroke.

- cv2.destroyAllWindows() and cv2.destroyWindow()

  cv2.destroyAllWindows () simply destroys all the windows we created. If you want to destroy any specific window, use the function cv2.destroyWindow() where you pass the exact window name as the argument.

```
1 import cv2
2 import os
3 fn = os.path.join(os.getcwd(), "./videos\images\sunny.jpg")
4 grayscaleimage = cv2.imread(fn,0)
5 cv2.imshow('image',grayscaleimage)
6 cv2.waitKey(0)
7 cv2.destroyAllWindows()
```

Figure 7: Load Image with gray scale mood

- cv2.VideoCapture()

  Often with a camera, we have to capture live stream. OpenCV is providing this with a very simple interface. To capture and display a video from the camera that will convert it to grayscale video. To capture a video, you need to create a VideoCapture object. Its argument can be either the device index or the name of a video file. The device index is just the number to specify which camera.

3.2.3.2 Minidom

Minidom is a simplified Document Object Model interface with related APIs in other languages. Use this library for doing some operation to xml file such as read content of XML file. Following example describe how to minidom library. (mkyong, 2019)

```
<data>
   <items>
      <item name="expertise1">SQL</item>
      <item name="expertise2">Python</item>
   </items>
</data>
import xml.etree.ElementTree as ET
tree = ET.parse('items.xml')
root = tree.getroot()

# all items data
print('Expertise Data:')

for elem in root:
   for subelem in elem:
      print(subelem.text)

Expertise Data:
SQL
Python
```

3.2.4 Anaconda Navigator

Anaconda Navigator is a graphical user interface (GUI) included in the Anaconda ®distribution that enables you to launch applications and navigate conda packages, environments and channels quickly without using command-line commands. Navigator can scan for packages from Anaconda Cloud or from a local repository of Anaconda. Windows, macOS, and Linux are available. Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. (docs.anaconda)

The following applications are available by default in Navigator.

- JupyterLab
- Jupyter Notebook
- QTConsole
- Spyder
- VSCode
- Glueviz
- Orange 3 App
- Rodeo
- RStudio

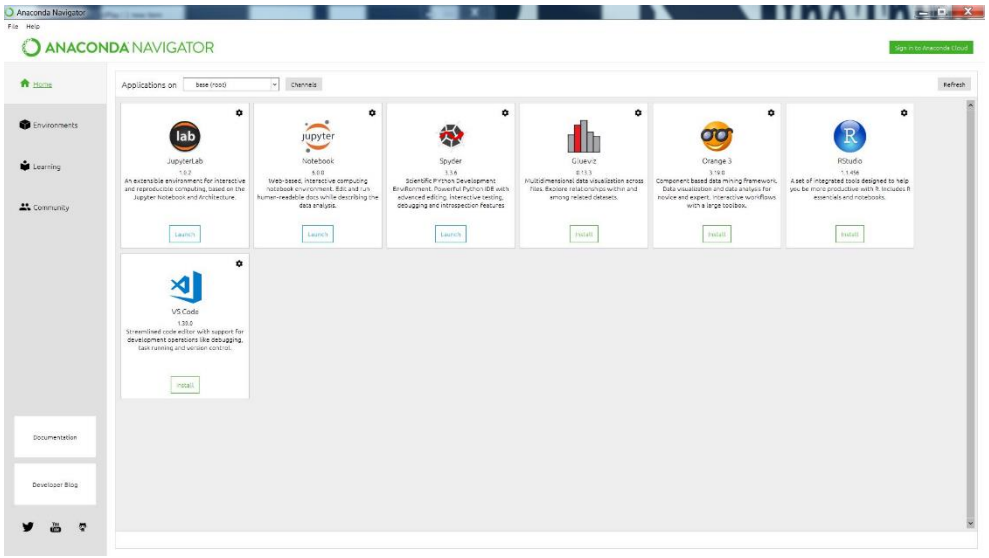


Figure 8: Preview of Anaconda Navigator

3.2.5 Jupyter Notebook 6.0.2

The Jupyter Notebook is an open-source web application that allows you to create and share live code, calculations, visualizations, and narrative text documents. Uses include data cleaning and transformation, numerical simulation, mathematical modeling, machine learning, data visualization, and more.

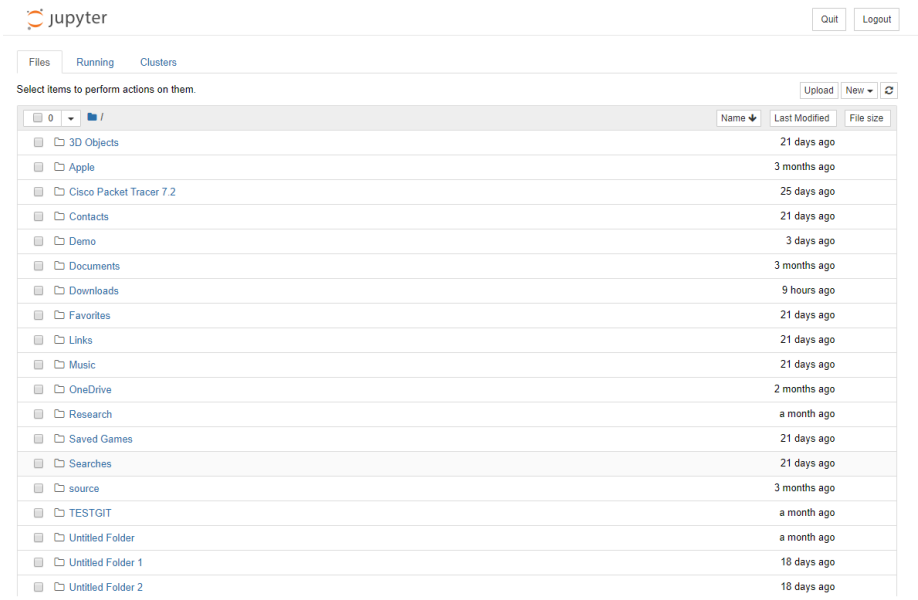


Figure 9: Start of jupyter note book

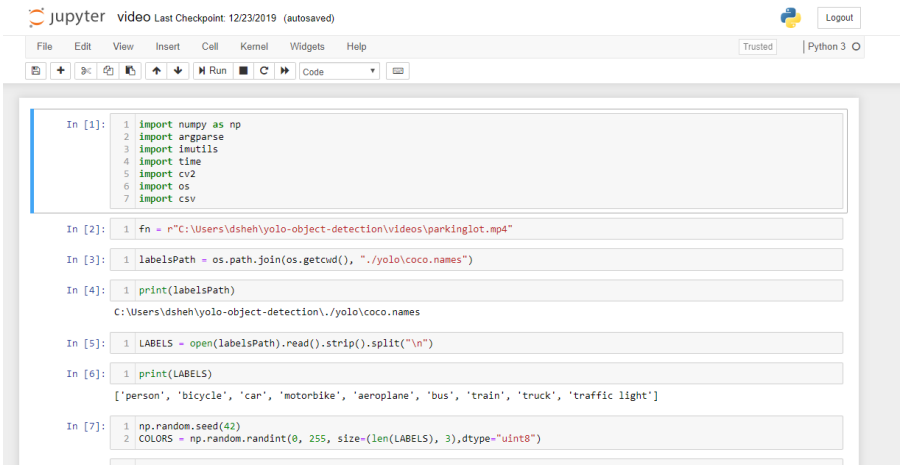


Figure.10: Edit preview of jupyter note book

3.2.6 ASUS NotebookSKU F556U Laptop computer

There should be a stable and well-organized programming machine with all the above tools. Below are the features I used for my work on my laptop computer.

Table 2: Laptop Details

• **Operating System**	Microsoft Windows 10 (64 bit) version
• **Processor**	Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz
• **Memory**	8 GB DDR3L-1600 SDRAM
• **Hard Drive**	1 TB 5600 rpm SATA
• **Screen**	15.6" HD (1366 x 768) resolution

3.3 Methodology

3.3.1 Data Collection

Using a reliable data set of high-quality images is an essential part of the workflow of computer vision research. The image data can be viewed from multiple cameras at different angles in different forms, such as video sequences. Because the object of this research is to identify the free parking slots, collecting real time video from parking areas is the data collecting part. There are two ways of gathering data. One way is getting the real time CCTV footage of parking areas from different places. Because of the security reason this method is not possible. Another way is collecting data from internet sources. The image data was acquired from the PKLots data set.

3.3.1.1 PKLots Dataset

Due to a large number of practical applications, outdoor parking lot vacancy detection systems have attracted a lot of attention in the last decade. The PKLot dataset includes

12,417 images of parking lots and 695,899 images of segmented parking spaces, checked and numbered manually. Both photographs were bought at the Federal University of Parana and the Pontifical Catholic University of Parana parking lots in Curitiba, Brazil. The dataset for parking slots is acquired by following the image acquisition process. This process was executed with a 5-min time-lapse interval for more than 30 days by means of a low cost full high definition camera (Microsoft LifeCam, HD-5000) positioned at the top of a building to minimize the possible occlusion between adjacent vehicles. Here is a short summary of this dataset. (De Almeida et al. 2015)

- Photos are taken under unregulated lighting that depicted various climatic conditions (sunny, rainy and overcast periods).
- Images were taken from different parking lots presenting distinct features.
- Images show a variety of issues, such as shadow effect, over-exposure to sunlight, low light in rainy days, the disparity in perspective, etc.

Also, an Extensible Markup Language (XML) file containing the position and location (vacant or occupied) of each parking space was created for each parking lot image.

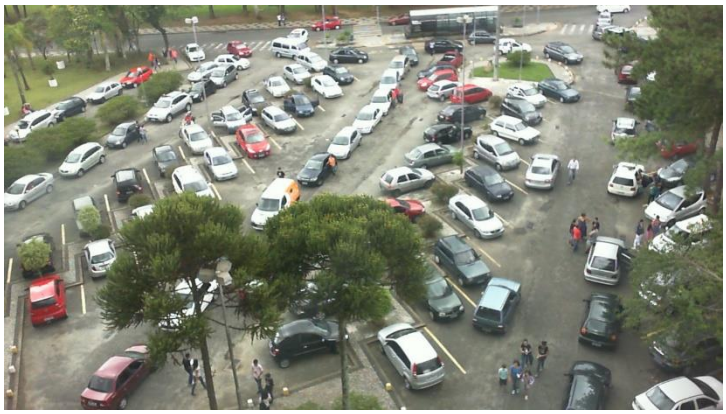


Figure 11: Image of PKLot dataset

Also, real time video from 'BLK-HDPTZ12 Security Camera Parking Lot Surveillance Video' is collected which includes include YouTube.

3.3.1 System Initialization

The vision-based method makes it possible to manage a large area by just using several cameras. This project aims is to identify the free parking slots. One or more cameras are used for image processing it is depending on the parking area. Here, one camera situation is considered. The process of this system shown in the below figure.

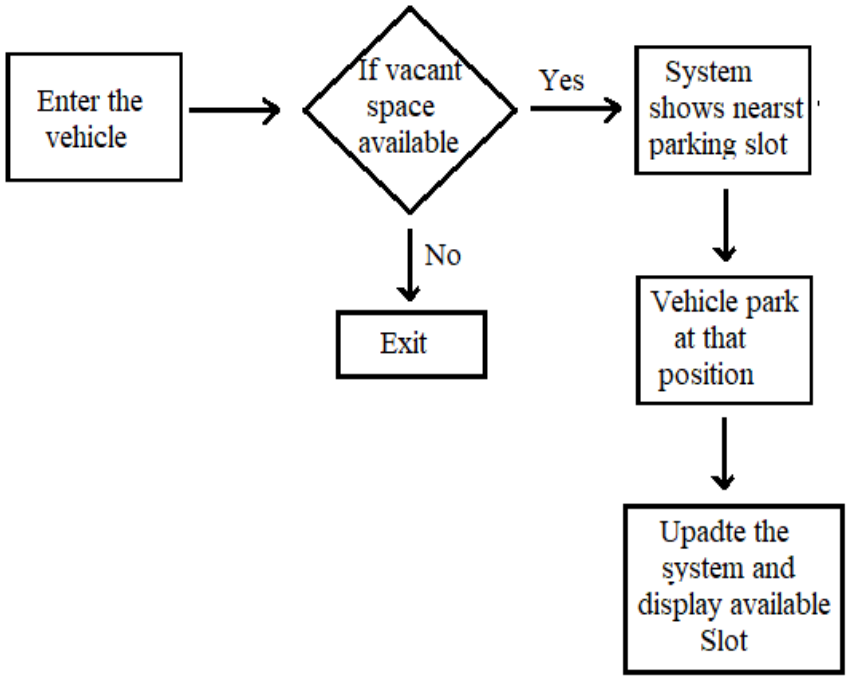


Figure 12: Block diagram of the process

The driver will see the free parking slots on the screen when the vehicle arrives at the entrance to the parking area. When the driver parked the vehicle at the free parking slots, the system is updated and display the newly available free parking slots.

To show the available free parking slots and the nearest parking slot used following the algorithm. If there are no free parking slots, the system displays the message parking area is full. Otherwise, the system shows free available parking slots. After the driver parking, the vehicle in the nearest or any free parking slot the system updates the status of each parking slot and displays the result at the vehicle park entrance.

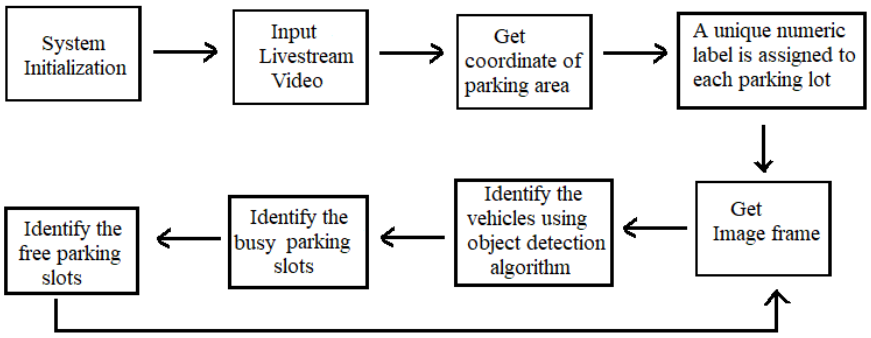


Figure 13: Algorithm of the system

The system initialization is a major part of this process. The first step is placing a camera in a fixed position above the vehicles, to acquire the video which used to get the vacancy of the car park. This camera should be in a position where that can see all parking lots. Also, the camera should be in a fixed position more than ten feet above the ground level because all vehicles must be clearly captured.

3.3.2 Input the Livestream video

CCTV system was used to get the parking area's real-time video. This system uses a fixed camera. Fixed cameras are mounted in a stationary position and focus on a single FOV, which is usually a particular area of interest. Fixed cameras are usually less expensive. If the parking area is very large high-quality image sensors must be used. The image sensor size and the focal length can be used to determine the FOV. As shown in the following figure the FOV is the area seen by the camera and lens.

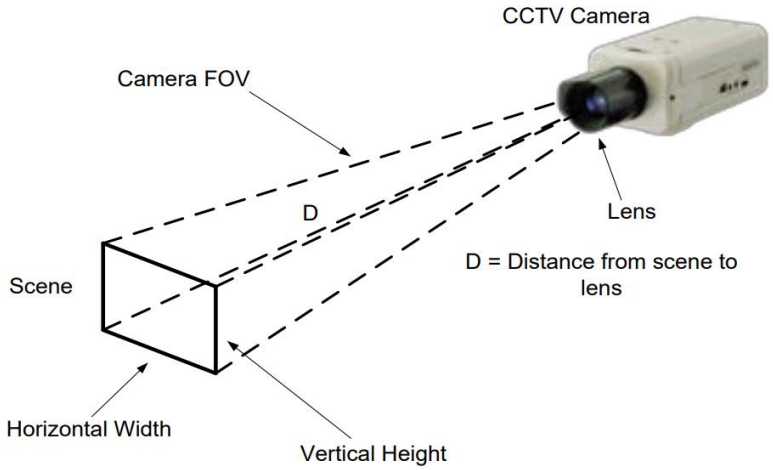


Figure 14: Calculating FOV

The following table shows how to calculate the horizontal width and vertical height of the FOV. The FOV is calculated using the image format information for the camera image size, along with the distance to the object of interest and the lens focal length. (The U.S. Department of Homeland Security 2013)

Table 3: Calculating the Horizontal and Vertical FOV

Imager Size	⅓ inch	½ inch	⅔ inch
Horizontal Format	4.4 millimeters	6.4 millimeters	8.8 millimeters
Vertical Format	3.3 millimeters	4.8 millimeters	6.6 millimeters
Horizontal Width Calculation: (Horizontal Format x Distance)/Focal Length			
Vertical Height Calculation: (Vertical Format x Distance)/Focal Length			
Example: Using a distance of 79 feet from a camera on a building overlooking a street, select a ⅓-inch format (imager size) camera with a 5-millimeter lens. **Horizontal width** = (4.4 mm x 79 ft)/5mm = 70 ft **Vertical height** = (3.3 mm x 79 ft)/5mm = 52 ft The camera has a coverage area of 70 feet x 52 feet at a distance of 79 feet.			

3.3.3 Identify the parking area

The individual parking slots must be identified after receiving the CCTV footage. For this step, the fist gets a video frame without any vehicles in the parking area. The following figure is shown in the parking area without any vehicles.

Figure 15: Example of parking Area

Using the image map creator identify the coordinates of each parking slot. (Image Mapping, n.d.)Using this tool get the coordinates of each parking slot. The following figure is shown marked polygon of some parking slots and gets the coordinates of that parking slots.

Figure 16: Marked area of parking slots

Below are the co-ordinates of these parking spaces.

```
<map name="map">
   <area shape="poly" coords="720, 549, 805, 665, 715, 698, 644,582" />
   <area shape="poly" coords="769, 451, 849, 534, 789, 586, 701, 489" />
   <area shape="poly" coords="806, 399, 879, 466, 823, 499, 747, 425" />
   <area shape="poly" coords="829, 323, 900, 383, 862, 430, 771, 356" />
</map>
```

Using this coordinate create an XML to identify the each slots. The XML file is shown below.

```
<? Xml version="1.0"?>
<parking id="ufpr04">
  <space id="0" >
     <point x="720" y="549" />
     <point x="805" y="665" />
     <point x="715" y="698" />
     <point x="644" y="582" />
  </space>
  <space id="1" >
     <point x="769" y="451" />
     <point x="849" y="534" />
     <point x="789" y="586" />
     <point x="701" y="489" />
  </space>
  <space id="2" occupied="0">
     <point x="806" y="399" />
     <point x="879" y="466" />
     <point x="823" y="499" />
     <point x="747" y="425" />
  </space>
  <space id="3" occupied="0">
     <point x="829" y="323" />
     <point x="900" y="383" />
     <point x="862" y="430" />
     <point x="771" y="356" />
  </space>
</parking>
```

3.3.3 Labeling parking slots

The above XML file has coordinates of each polygon. Used these polygon coordinates to find the middle of each polygon. After that, each car parking slot has been given a unique number. The following figure is shown parking slots with a unique id.

Figure 17: parking slots with unique ID

Identify all parking slots labeling every parking slot using the above steps. The following figure shows labeling different parking areas.

Figure 18: Labelling all parking slots

The maximum number of slots that could be labeled depends on the capture area of the camera and the training model.

3.3.4 Identify Vehicles

To identify the vehicles in the parking area, used the YoloV3 object detecting algorithm. After detecting the vehicles in the parking area get the mean pixel value of each vehicle in the parking area.

Figure 19: Identity the vehicle

Use the following equations to find the mean pixel values.

$$Mean\ pixel\ value\ of\ x = \frac{\sum x}{n}$$

$$Mean\ pixel\ value\ of\ y = \frac{\sum y}{n}$$

3.3.4.1 Yolo Object Detection Algorithm

The use of algorithms for object detection is becoming increasingly important in high accuracy object detection. Object detection is a computer vision activity involving the identification in a given photograph of the presence, position, and form of one or more objects. Deep learning techniques have obtained state-of-the-art results for object detection in recent years, such as on regular datasets for comparisons and in competitions for computer vision. Notable is the YOLO the Convolutional Neural Networks family that achieves almost state-of -the-art results with a single end-to-end model capable of

real-time object detection. (Brownlee, 2019) YOLO, the family of models are a series of end-to-end deep learning models designed for fast object detection, developed by Joseph Redmon, et al. and first described in the 2015 paper titled “You Only Look Once: Unified, Real-Time Object Detection”. YOLO at the time of writing, there are three major method variations; they are YOLOv1, YOLOv2, and YOLOv3. The first version proposed the general architecture, while the second version refined the design and used predefined anchor boxes to enhance the bounding box proposal, while the third version refined the architecture and training phase of the model. This is followed by Faster R-CNN that used an RPN for the identification of bounding boxes that needed to be tested. The first step is to download the pre-trained model weights. These were trained using the DarkNet code base on the MSCOCO dataset. COCO dataset is an outstanding dataset for object detection with 80 groups, 80,000 images for training and 40,000 images for validation. This is a mirror of that dataset, as uploading from their website is slow at times. (Farhadi) Next load the model weights. The model weights are stored in whatever format that was used by DarkNet. The output of the model is, in fact, encoded candidate bounding boxes from three different grid sizes, and the boxes have defined the context of anchor boxes, carefully chosen based on an analysis of the size of objects in the MSCOCO dataset.

OpenCV with Python bindings and necessary packages like Numpy, Os, Time, etc are used to implement this algorithm. Yolo is called YOLOV1, YOLOV2, YOLOV3 in three variants. YOLOV3 is used here because it was more accuracy compared to other models. YOLO divides the picture into a 13-by-13 cell grid. Each of these cells is responsible for predicting 5 bounding boxes. Bounding box describes the rectangle that encloses an object. YOLO also provides a confidence score that informs us how likely it is that some item is included in the expected bounding box. Previous detection systems are repurposing detection classifiers or localizers. At multiple locations and scales, they apply the model to a picture. Detections are considered as high scoring regions of the scene. YOLO's style is different. This refers to the whole image of a single neural network. This network divides the picture into regions and predicts boundary boxes and probabilities for each region. The expected probabilities weigh these bounding boxes. The YOLO V3 is more accurate. The following picture depicts the overall architecture for YOLOV3.

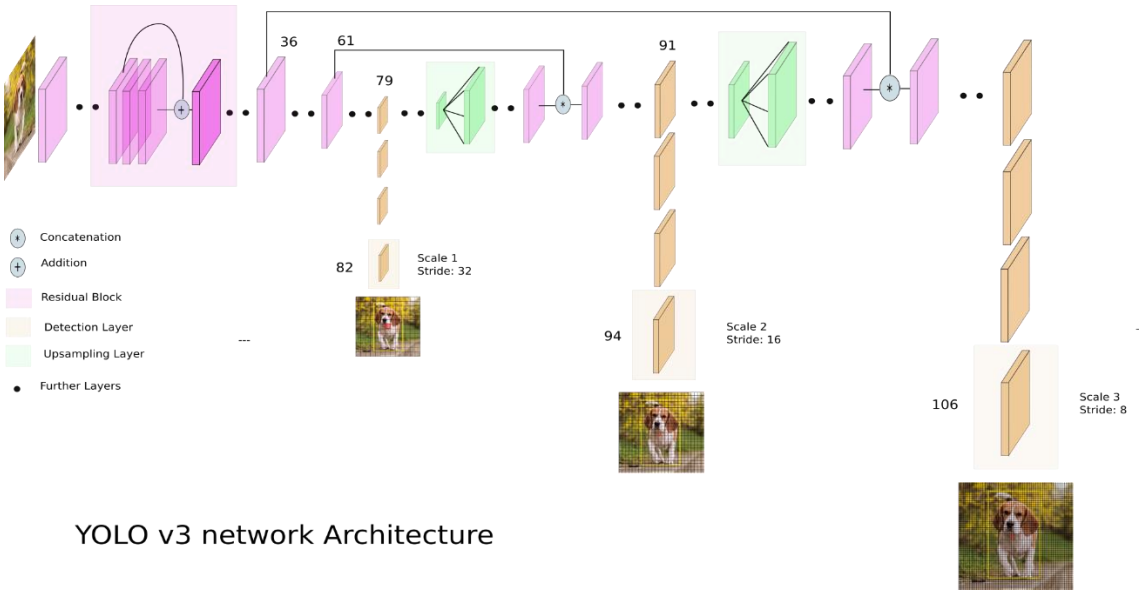


Figure 20: The architecture of YOLO V3

Also, convolutional layers are used by YOLO, rendering it a completely convolutional network (FCN). It has 75 layers of convolution, with skip links and layers of upsampling. No type of pooling is used, and the feature maps are downsampled using a convolutional layer with phase. It helps prevent the loss of commonly attributed low-level functionality to pooling. [10] The features learned by the convolutional layers are passed onto a classifier which makes the detection prediction. Our output is the first thing to notice is a feature map. The size of the prediction map is exactly the size of the function map before it. since it is used 1x 1 convolution. In YOLO v3, the understanding of this prediction map is that a fixed number of bounding boxes can be predicted by each cell. The following formulae describe how the network output is transformed to obtain bounding box predictions.

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

Figure 21: Formulae bounding box predictions

bx, by, bw, bh are the x,y center coordinates, width and height of our prediction. tx, ty, tw, th is what the network outputs. cx and cy are the top-left coordinates of the grid. pw and ph are anchors dimensions for the box. our center coordinates prediction through a

sigmoid function. This forces the value of the output to be between 0 and 1. The dimensions of the bounding box are predicted by applying a log-space transform to the output and then multiplying with an anchor. Object score represents the probability that an object is contained inside a bounding box. As it is to be interpreted as a probability, the objectness score is also passed through a sigmoid. Class confidences are the probabilities of the object detected belonging to a specific class (bicycle, truck, van, car, etc.). Until v3, YOLO used the class scores to softmax. The design option was dropped in v3, however, and instead, writers opted to use sigmoid. The reason is that the Softmaxing class scores assume the classes are excluded from each other. If an object belongs to one class in simple words, it is assumed that it can not belong to another class. This applies to the COCO database on which our detector will be based. YOLO v3 makes three different scales of prediction. For function maps of three different sizes, with steps 32, 16, 8 respectively, the detector layer is used to render detection. This means that we detect on scales 13x 13, 26x 26 and 52x 52 with an input of 416x 416. The network downsamples the input image to the first layer of detection where detection is achieved using a layer of stride 32 feature maps. Also, layers are upsampled by a factor of 2 and concatenated with feature maps of an earlier layer with similar feature map sizes. Another detection is now being rendered at the phase 16 layer. The same method for sampling is repeated and a final detection is made.

3.3.5 Identify the busy parking slots and free parking slots

The status of each parking slot can be determined by comparing the mean pixel value of each vehicle and the coordinates of bounding boxes. If the mean pixel value of the vehicle is inside the given bounding box then, that parking slot is not empty. Otherwise, it is empty. Sequentially update the status of each parking slot and display it. The following figure is shown by busy parking slots. The mean pixel value of the vehicle is inside the boundary area because of that parking is busy.

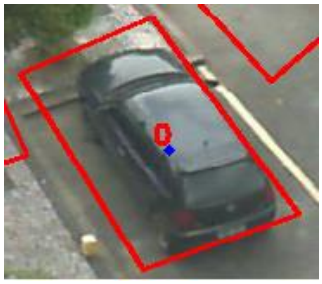


Figure 22: Busy Parking slot

The following figure shows the free parking slot. There are no mean pixel values of a vehicle inside the boundary area.

Figure 23: Free Parking slot

3.3.6 Identify the nearest parking slot

Numbering each parking slot is done by considering the distance from the entrance to the parking slot. Then XML file is created according to these numbering. The number zero is given the nearest parking slot. Finally, the system is displayed nearest parking firstly.

3.3.7 Identify the vehicles types

The system can identify different types of vehicles such as car, van, bus, and truck, etc. To identify the vehicle's type used YOLOV3 object detection algorithm. After identifying the vehicles in the parking area, display the vehicle types in each parking slot.

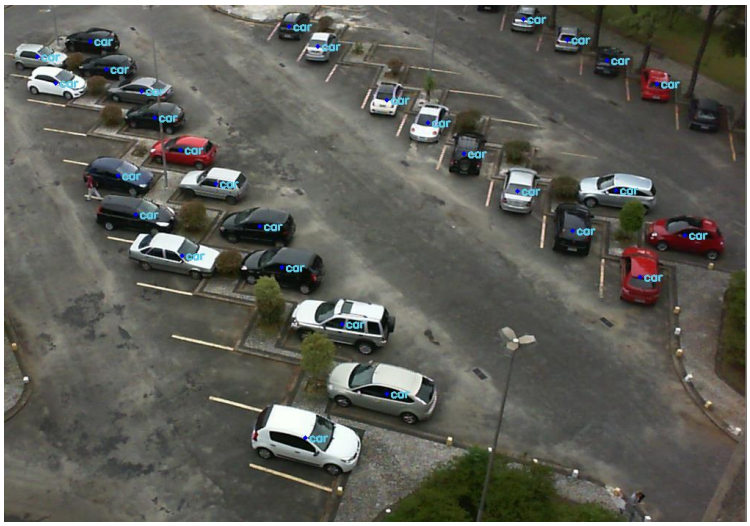


Figure 24 : Vehicles Types

CHAPTER 4 – RESULTS AND DISCUSSION

This section contains the results of this proposed model. Also, I discussed how I achieved the targets and objectives with the proposed solution. Moreover, accuracies were compared with other relevant solutions.

To test the accuracy of the system, used images at different parking areas in various weather conditions. The system is getting the image from video files. The performance of the proposed system is measured by using the following equation. (Bibi et al. 2017)

TPS = Total Parking slots

ANC = Actual Number of vehicles inside the parking slots

PNC = Detected Number of vehicles inside the parking slots

The percentage of error in the proposed system can be found by using this equation.

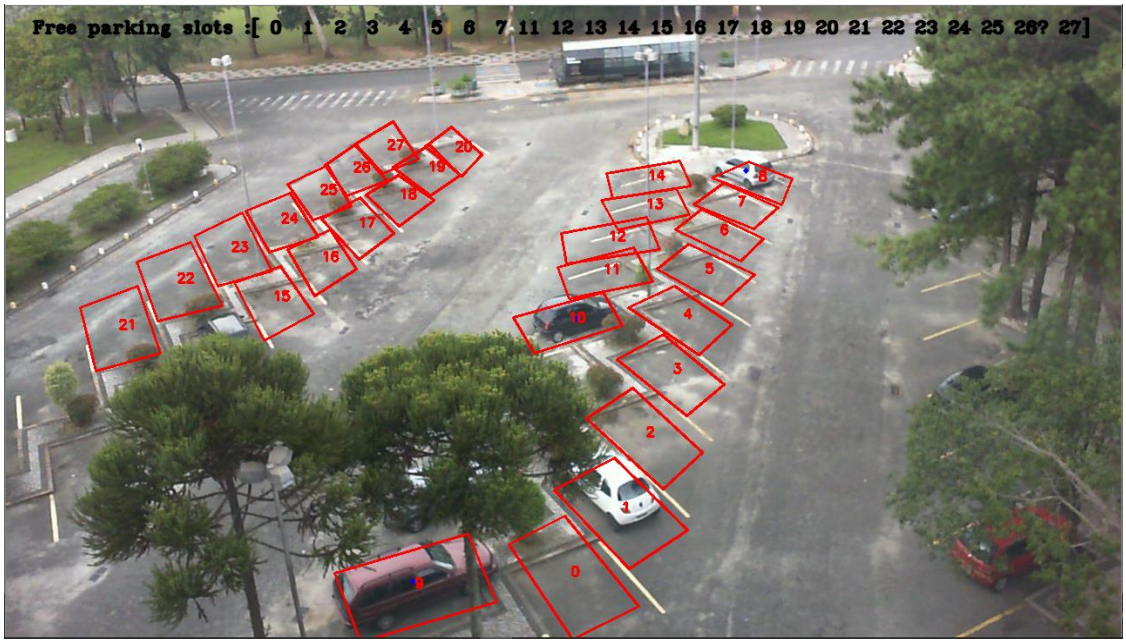
Percentage Error = $(\frac{ANC-PNC}{TPS})*100$

Twelve images of two different two parking areas and three weather conditions are used to test this system. Also, used a large parking area and real time video.

4.1 Test cases

4.1.1 Test case-01

The following figure shows UFPR04 parking area in cloudy weather condition.




```
Total Numbres of Parking Slots: 28
Total busy parking slots: 3
Busy parking slots :[8, 9, 10]
Free parking slots :[ 0  1  2  3  4  5  6  7 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
 27]
```

Figure 25: UFPR04 Area in cloudy weather condition

Total Numbers of Parking slots=28

Actual Number of vehicles inside the parking slots=4

Detected Number of vehicles inside the parking slots=3

Undetected number of vehicles inside the parking slots=1

Percentage Error= 3.571%

4.1.2 Test case-02

The following figure shows UFPR05 parking area in cloudy weather condition.

Figure 26: UFPR05 Area in cloudy weather condition

Total Numbers of Parking slots=40

Actual Number of vehicles inside the parking slots=23

Detected Number of vehicles inside the parking slots=23

Undetected number of vehicles inside the parking slots=0

Percentage Error= 0

4.1.3 Test case-03

The following figure shows the UFPR04 parking area in rainy weather condition.

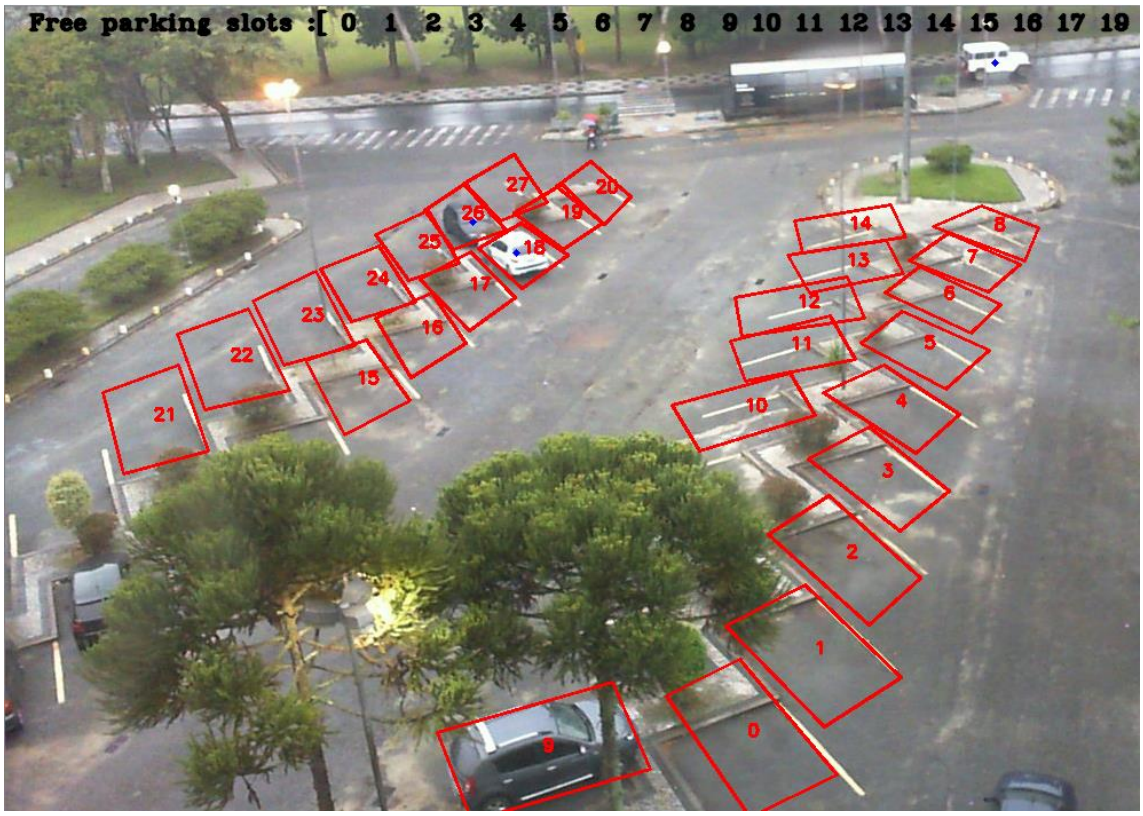


Figure 27 : UFPR04 Area in rainy weather condition

Total Numbers of Parking slots=28

Actual Number of vehicles inside the parking slots=3

Detected Number of vehicles inside the parking slots=2

Undetected number of vehicles inside the parking slots=1

Percentage Error= 3.571%

4.1.4 Test case-04

The following figure shows the UFPR05 parking area in rainy weather condition.

Figure 28: UFPR05 Area in rainy weather condition

Total Numbers of Parking slots=40

Actual Number of vehicles inside the parking slots=27

Detected Number of vehicles inside the parking slots=27

Undetected number of vehicles inside the parking slots=0

Percentage Error= 0

4.1.5 Test case-05

The following figure shows the UFPR04 parking area in sunny weather condition.

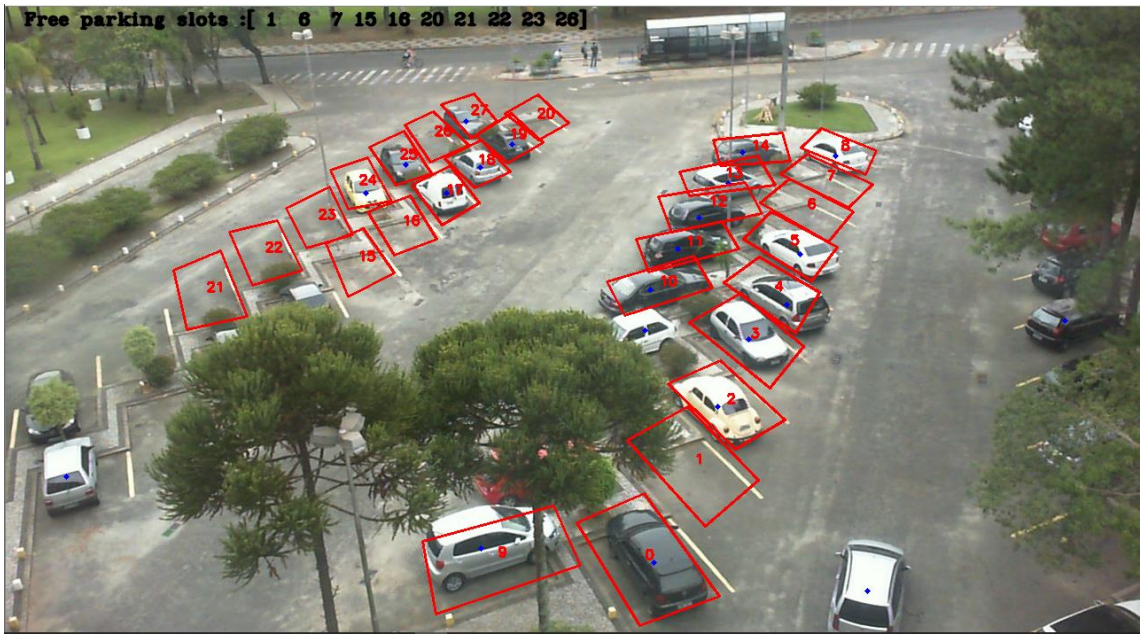


Figure 29:UFPR04 Area in sunny weather condition

Total Numbers of Parking slots=40

Actual Number of vehicles inside the parking slots=18

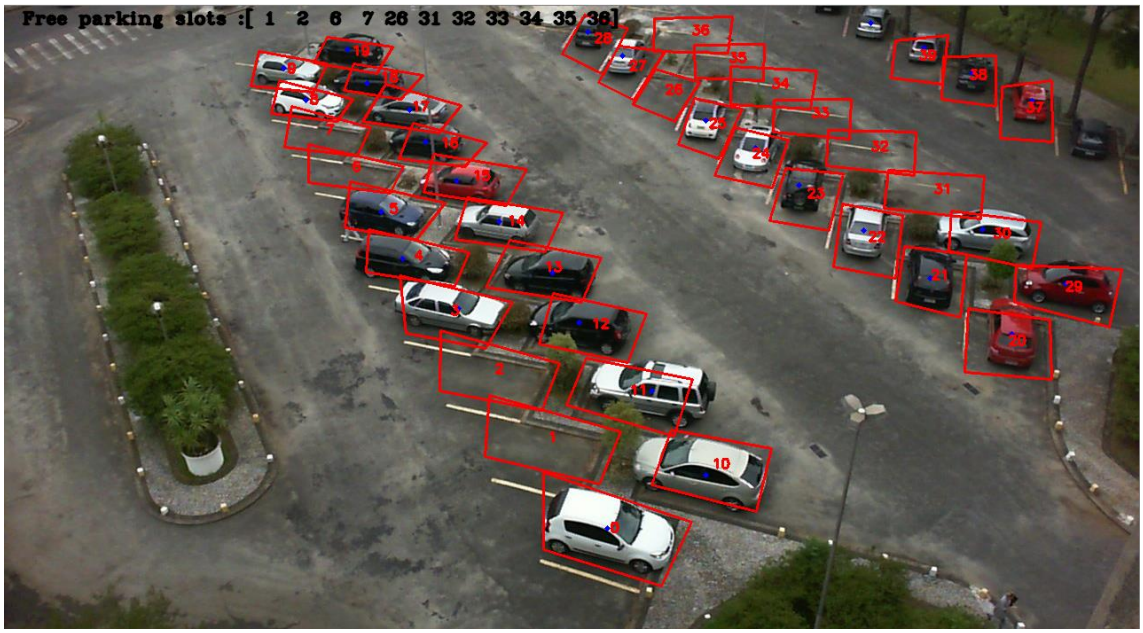
Detected Number of vehicles inside the parking slots18

Undetected number of vehicles inside the parking slots=0

Percentage Error= 0

4.1.5 Test case-06

The following figure shows the UFPR05 parking area in sunny weather condition.




```
Total Numbres of Parking Slots: 40
Total busy parking slots: 29
Busy parking slots :[0, 3, 4, 5, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 37, 38,
39]
Free parking slots :[ 1  2  6  7 26 31 32 33 34 35 36]
```

Figure 30:UFPR05 Area in sunny weather condition

Total Numbers of Parking slots=40

Actual Number of vehicles inside the parking slots=29

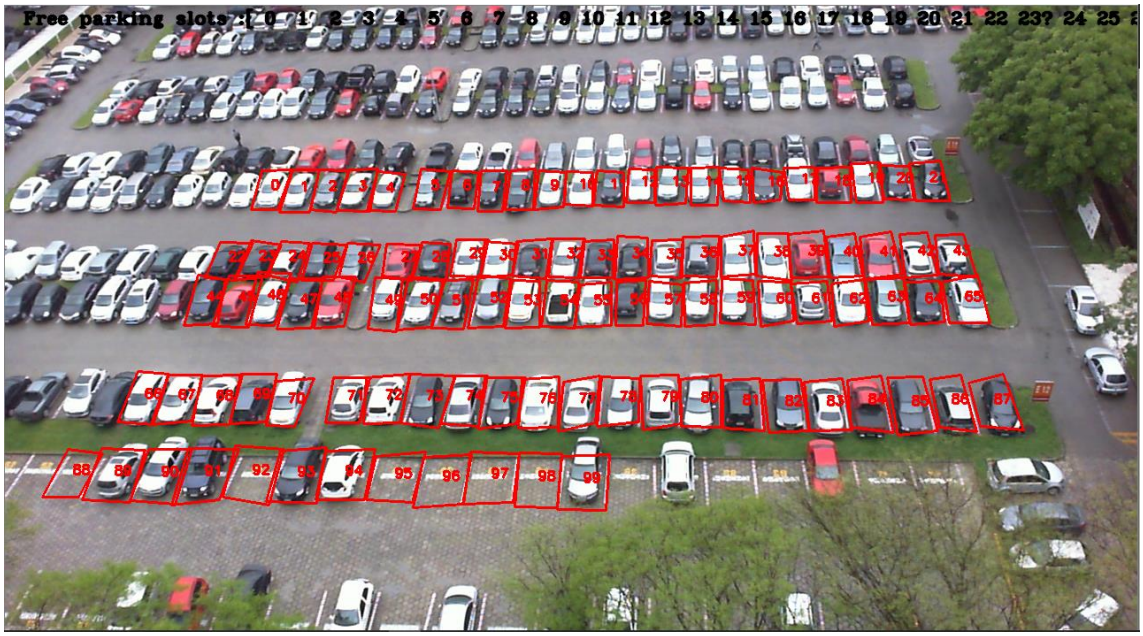
Detected Number of vehicles inside the parking slots=29

Undetected number of vehicles inside the parking slots=0

Percentage Error= 0

4.1.5 Test case-07

The following figure shows the PUCPR parking area in sunny weather condition.




```
Total Numbres of Parking Slots: 100
Total busy parking slots: 0
Busy parking slots :[]
Free parking slots :[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
 96 97 98 99]
```

Figure 31:UFPR05 Area in sunny weather condition

Total Numbers of Parking slots=100

Actual Number of vehicles inside the parking slots=71

Detected Number of vehicles inside the parking slots=0

Undetected number of vehicles inside the parking slots=71

Percentage Error= 71%

4.2 Test Real Time video

BLK-HDPTZ12 Security Camera Parking Lot Surveillance Video has been used for testing real time video analyzed. This video shows the quality of the 2.1-megapixel image and the 12 x optical zoom. (Supercircuits, 2012) After getting the video, it is grabbed into frames. Using the above same process for these frames to identifying the free parking slots.

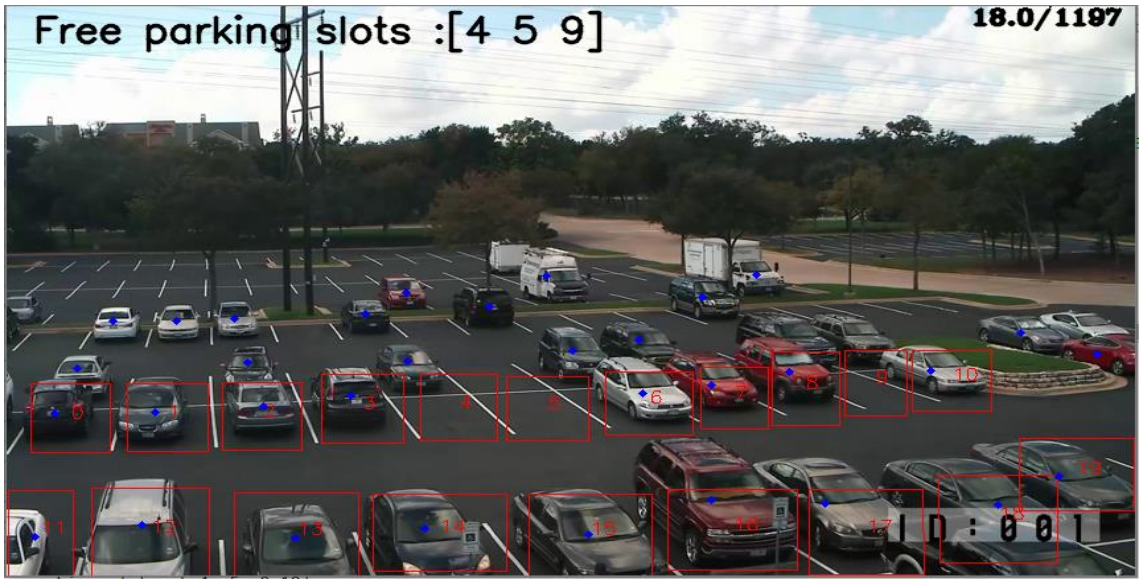


Figure 32: 18th frame of real time video

Total Number of frames in the video file=1197

Current video frame=18

Total Numbers of Parking slots=20

Actual Number of vehicles inside the parking slots=17

Detected Number of vehicles inside the parking slots=16

Undetected number of vehicles inside the parking slots=1

Percentage Error= 5%

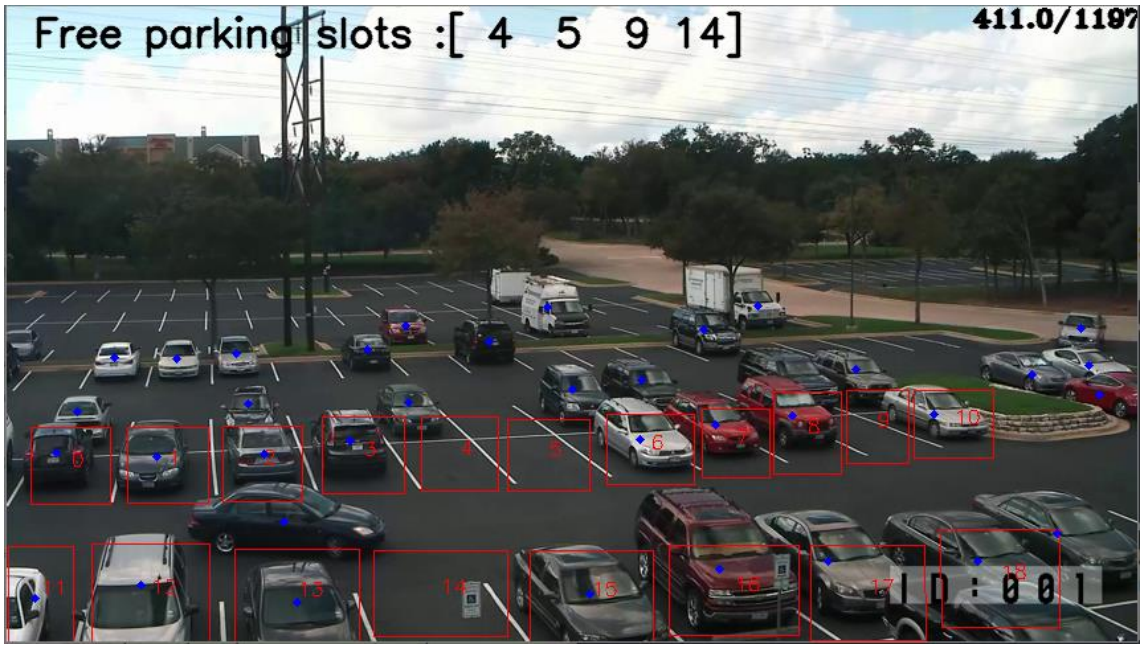


Figure 33:411th frame in real time video

Total Number of frames in the video file=1197

Current video frame=411

Total Numbers of Parking slots=20

Actual Number of vehicles inside the parking slots=16

Detected Number of vehicles inside the parking slots=15

Undetected number of vehicles inside the parking slots=1

Percentage Error= 5%

4.3 Overall Discussion

Over roll accuracy of this system is 95% approximately. When the parking area is large it is difficult to identify the vehicles in the parking area. The test case 07 shows the large parking area. In that case, the percentage error is 71%. These methods generally show a high accuracy but have a disadvantage of slow detection speed and lower efficiency. Because of that, it needs more hardware performances to capture the large parking area at once.

CHAPTER 5 – CONCLUSION

5.1 Overview

A summary of the research has been described in this chapter. In addition, by achieving goals, the conclusion of the proposed model has been described. Finally, the process of future development was discussed.

5.2 Overall Achievements

The Major objective of the proposed system is finding the free parking slots. This research is taken as an approach for a smart car parking system. The system was developed mainly using a Python object detecting algorithm to achieve this goal. The developed system was able to show an accuracy of 95% approximately.

5.3 Problems and Limitations Faced During the Project

During the data selection phase, it was difficult to find images and videos of the parking area. But a sufficient number of images were gathered from PKLots dataset and videos were gathered from Google search engines'. Also, it must be required more computational power for the implementation of this project because video processing has complex algorithms. The accuracy of this system depends on the object detection algorithm. If the parking area is large it is difficult to detect the vehicles. The camera should be in a position where can see all parking lots and camera should be fixed more than ten feet above the ground level. In order to capture the vehicles by the camera, there should not have obstacles between cameras and parking areas. If the camera position is changed, coordinates should be taken again to create the new XML file.

5.4 Future Work and Conclusion

This study's main contribution is to optimize the identification of available parking slots to potentially reduce parking area congestion. As a result of advances in machine learning and vision-based technology, cost-effective automatic parking systems make it easier for drivers to locate available parking spaces. This method is capable of managing large areas

by just using several cameras. This system is developed by using an integrated image processing approach to reduce the cost of sensors and the wiring hassle. It is cheap and easy-installed because of the simple equipment. The camera position could be adjusted to improve performance. Always a camera must capture the whole area of every parking slot for the highest accuracy. Drivers can get useful real-time parking lot information from this system by the guidance information display. Future researchers can focus on the allocation of specific locations and security parking system using an online parking management system by registered customers.

References

aayushiedureka. (2019, 11 26). *Python NumPy Tutorial – Learn NumPy Arrays With Examples.* Retrieved from edureka: https://www.edureka.co/blog/python-numpy-tutorial/

Aekarat Saeliw, W. H. (2019). Smart Car Parking Mobile Application based on RFID and IoT. *International Journal of Interactive Mobile Technologies (iJIM)*, 1-10.

Benjamin Kommey, A. S. (2018). A Smart Image Processing-based System for Parking Space Vacancy Management-J. *International Journal of Computer Applications*.

Brownlee, J. (2019, 05 27). *How to Perform Object Detection With YOLOv3 in Keras.* Retrieved from https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/

DataFlair. (2019, 12 05). *Features of Python – Review the irresistible Python attributes*. Retrieved from https://data-flair.training/blogs/features-of-python/

Department of motor traffic Srilanka. (2015). Retrieved from http://www.motortraffic.gov.lk/

docs.anaconda. (n.d.). *Anaconda Navigator.* Retrieved from https://docs.anaconda.com/anaconda/navigator/

Edirisinghe, D. (2014). *Managing traffic congestion in Colombo and its suburbs.* Retrieved from Sri Lanka Institute Of Development Administration Colombo Sri Lanka: http://www.slida.lk/slidatest/application_resources/images/mpm_reso/policypaper/pp5.pdf

educba. (2019). *Introduction to NumPy*. Retrieved from https://www.educba.com/what-is-numpy/

Farhadi, A. (n.d.). *Common Objects in Context Dataset Mirror.* Retrieved from https://pjreddie.com/projects/coco-mirror/

Image Mapping. (n.d.). Retrieved from https://summerstyle.github.io/summer/

Image Processing With Deep Learning- A Quick Start Guide. (n.d.). Retrieved from https://becominghuman.ai/image-processing-with-deep-learning-a-quick-start-guide-38e166340200

J. Wolff, T. H. (2006). Parking monitor system based on magnetic eld sensors. *IEEE Conference on Intelligent Transportation Systems*, (pp. 1275–1279).

Joseph Redmon, A. F. (2018). *YOLOv3: An Incremental Improvement.* Retrieved from https://pjreddie.com/media/files/papers/YOLOv3.pdf

Labs, P. (n.d.). *programiz*. Retrieved from Parewa Labs Pvt.: https://www.programiz.com/python-programming/time

mkyong. (2019, 12 04). *Python XML Parser Tutorial: Read xml file example(Minidom, ElementTree).* Retrieved from guru99: https://www.guru99.com/manipulating-xml-with-python.html

Supercircuits. (2012). *YouTube*. Retrieved from YouTube: https://www.youtube.com/watch?v=U7HRKjlXK-Y&t=9s

Suryanshrao. (2017, 10 13). *brainly.* Retrieved from https://brainly.in/question/1586608

Tagliaferri, L. (2017, 09 28). *An Introduction to Machine Learning*. Retrieved from https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning

De Almeida, Paulo R.L. et al. 2015. "PKLot-A Robust Dataset for Parking Lot Classification." *Expert Systems with Applications* 42(11): 4937–49. http://dx.doi.org/10.1016/j.eswa.2015.02.009.

Baştanlar, Yalin, and Mustafa Özuysal. 2014. "Introduction to Machine Learning." *Methods in Molecular Biology* 1107: 105–28.

Bibi, Nazia, Muhammad Nadeem Majid, Hassan Dawood, and Ping Guo. 2017. "Automatic Parking Space Detection System." *Proceedings - 2017 2nd International Conference on Multimedia and Image Processing, ICMIP 2017* 2017-Janua(October): 11–15.

G, Nithinya, and Suresh Kumar R. 2016. "Design and Implementation of an Intelligent Parking Management System Using Image Processing." *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)* 5(1): 95–100.

Igbinosa, Ireyuwa. 2013. "Comparison of Edge Detection Technique in Image Processing Techniques." *Information Technology & Electrical Engineering* 2(1): 25–29.

Kommey, Benjamin, Ernest O., and Andrew S. 2018. "A Smart Image Processing-Based System for Parking Space Vacancy Management." *International Journal of Computer Applications* 182(5): 1–6.

Lézoray, Olivier, Christophe Charrier, Hubert Cardot, and Sébastien Lefévre. 2008. "Machine Learning in Image Processing." *Eurasip Journal on Advances in Signal Processing* 2008(June 2014): 1–3.

Malathy H Lohithaswa. 2015. "Canny Edge Detection Algorithm on FPGA." *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)* 10(1): 15–19. http://www.iosrjournals.org/iosr-jece/papers/Vol. 10 Issue 1/Version-1/C010111519.pdf.

Mordvintsev, Alexander, and K. Abid. 2017. "OpenCV-Python Tutorials Documentation." *OpenCV Python Documentation*: 1–269. https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf.

Rossum, Guido Van, and Fred L Drake. 2010. "Python Tutorial." *History* 42(4): 1–122. http://docs.python.org/tutorial/.

The U.S. Department of Homeland Security. 2013. "CCTV Technology Handbook." (July): 66.

APPENDICES

Python Codes

Lording necessary library files

```
import numpy as np
import argparse
import imutils
import time
import cv2
import os
from xml.dom import minidom
import xml.etree.ElementTree as ET
```

Capture the video file

```
fn = r"C:\Users\dsheh\yolo-object-detection\videos\test.mp4"
vs = cv2.VideoCapture(fn)
writer = None
(W, H) = (None, None)
```

Load the labels in the train model

```
labelsPath = os.path.join(os.getcwd(), "./yolo\coco.names")
LABELS = open(labelsPath).read().strip().split("\n")
print(LABELS)
```

```
['bicycle', 'car', 'van', 'bus', 'train', 'truck']
```

Load the Yolo weights and model configuration

```
weightsPath=os.path.join(os.getcwd(), "./yolo\yolov3.weights")
configPath=os.path.join(os.getcwd(), "./yolo\yolov3.cfg")
```

Load Yolo Object detector

```
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

Determine the total number of frames in the video file

```
try:
	prop = cv2.cv.CV_CAP_PROP_FRAME_COUNT if imutils.is_cv2() \
		else cv2.CAP_PROP_FRAME_COUNT
	total = int(vs.get(prop))
	print("[INFO] {} total frames in video".format(total))
except:
	print("[INFO] could not determine # of frames in video")
	print("[INFO] no approx. completion time can be provided")
	total = -1
```

```
[INFO] 1197 total frames in video
```

Get the coordinates of each parking slots to the array

```
xmlpath=os.path.join(os.getcwd(), "./videos\images\stestlotsvedio.xml")
xmldoc = minidom.parse(xmlpath)
itemlist = xmldoc.getElementsByTagName('space')
totals="Total Numbres of Parking Slots: "+str(len(itemlist))
print(totals)
arrayy=[]
arrayx=[]
root = ET.parse(xmlpath).getroot()
for type_tag in root.findall('space/contour/point'):
    x = type_tag.get('x')
    arrayx.append(x)
for type_tag in root.findall('space/contour/point'):
	y = type_tag.get('y')
	arrayy.append(y)
def divide_chunks(l, n):
       # looping till length l
    for i in range(0, len(l), n):
        yield l[i:i + n]
xlist = list(divide_chunks(arrayx, 4))
ylist= list(divide_chunks(arrayy, 4))
```

```
Total Numbres of Parking Slots: 20
```

Check mean pixel value of vehicles inside the boundary area

```
def check_boundary(x,y,poly):
    n = len(poly)
    inside = False
    p2x = 0.0
    p2y = 0.0
    xints = 0.0
    p1x,p1y = poly[0]
    for i in range(n+1):
        p2x,p2y = poly[i % n]
        if y > min(p1y,p2y):
            if y <= max(p1y,p2y):
                if x <= max(p1x,p2x):
                    if p1y != p2y:
                        xints = (y-p1y)*(p2x-p1x)/(p2y-p1y)+p1x
                    if p1x == p2x or x <= xints:
                        inside = not inside
        p1x,p1y = p2x,p2y

    return inside
```

Mark the bounding box of each parking slots and give unique ID

```
def draw_bounding_box(frames):
    for i in range(len(itemlist)):
        x1=xlist[i][0]
        x2=xlist[i][1]
        x3=xlist[i][2]
        x4=xlist[i][3]
        y1=ylist[i][0]
        y2=ylist[i][1]
        y3=ylist[i][2]
        y4=ylist[i][3]
        pts = np.array([[x1,y1],[x2,y2],[x3,y3],[x4,y4]], np.int32)
        point.append(pts)
        cv2.polylines(frames, [pts], True,(0,0,255), 1)
        xsum=(int(x1)+int(x2)+int(x3)+int(x4))/4
        ysum=(int(y1)+int(y2)+int(y3)+int(y4))/4
        cv2.putText(frames,str(i),(int(xsum),int(ysum)),cv2.FONT_HERSHEY_SIMPLEX,0.5, (0,0,255), 1)
```

Detect the vehicles and get mean pixel vales

```
for output in layerOutputs:
    for detection in output:
        scores = detection[5:]
        classID = np.argmax(scores)
        confidence = scores[classID]

        if confidence > 0.5:
            box = detection[0:4] * np.array([W, H, W, H])
            (centerX, centerY, width, height) = box.astype("int")

            x = int(centerX - (width / 2))
            y = int(centerY - (height / 2))

            boxes.append([x, y, int(width), int(height)])
            confidences.append(float(confidence))
            classIDs.append(classID)
idxs = cv2.dnn.NMSBoxes(boxes, confidences, 0.5,0.3)
if len(idxs) > 0:
    for i in idxs.flatten():
        if i==0:
            break
        (x, y) = (boxes[i][0], boxes[i][1])
        (w, h) = (boxes[i][2], boxes[i][3])

        color = [int(c) for c in COLORS[classIDs[i]]]
        text = "{}: {:.4f}".format(LABELS[classIDs[i]], confidences[i])
        cv2.putText(image, "", (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX,0.5, color, 2)

        xmean=int(x+(w/2))
        ymean=int(y+(h/2))

        xvalues.append(xmean)
        yvalues.append(ymean)

        image = cv2.circle(image, (xmean,ymean), 1, (255, 0, 0), 3)
```

Identify the free parking slots

```
w=0
car=[]
for i in range(len(xvalues)):
    for u in range(len(itemlist)):
        ss=ray_tracing(xvalues[i],yvalues[i],point[u])
        if(ss):
            #print(str(i)+" Mean value Inside ----------------------------------------------------------ID "+str(u))
            w=w+1;
            car.append(u)
            break

print(total)
car.sort()
print('Total busy parking slots: '+str(w))
print('Busy parking slots :'+str(car))
result='Free parking slots :'+str(np.setdiff1d(parkingID, car))
print(result)
    
# show the output image

image=cv2.putText(image, result, (200, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2, cv2.LINE_AA)
cv2.imshow("Image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
Total Numbres of Parking Slots: 28
Total busy parking slots: 36
Busy parking slots :[0, 0, 2, 2, 3, 3, 4, 4, 5, 5, 8, 8, 9, 9, 10, 10, 11, 11, 12, 12, 13, 13, 14, 14, 17, 17, 18, 18, 19, 19,
24, 24, 25, 25, 27, 27]
Free parking slots :[ 1  6  7 15 16 20 21 22 23 26]
```