SMART PARKING SOLUTIONS

MAJOR PROJECT REPORT

Submitted in partial fulfillment for the award

of the degree of

BACHELOR OF TECHNOLOGY IN

ELECTRONICS AND COMMUNICATION ENGINEERING

by

Anmol Deepak (08114802813)

Ashutosh Mishra (01714802813)

Gaurav Singh (01814802813)

Ronak Lalani (00714802813)

under the supervision of

Dr. Navneet Yadav, Assistant Professor

Maharaja Agrasen Institute of Technology

Guru Gobind Singh Indraprastha University, New Delhi

2013-17

CERTIFICATE

Certified that the major project report titled **"Smart Parking Solutions"** is a bonafide record of the work done by Mr. Anmol Deepak, Ashutosh Mishra, Gaurav Singh and Ronak Lalani**,** for the partial fulfillment of the requirements for the award of the four year degree of **Bachelors of Technology** in **Electronics and Communication Engineering**

Dr. Neelam Sharma
(Head of Dept, ECE)

Dr. Navneet Yadav
(Assistant Professor, ECE Dept)

Smart Parking Solutions

Abstract

With the increase in vehicle production and world population, more and more parking spaces and facilities are required. In world context, many approaches were suggested and implemented based on sensor technology, image processing and character recognition technology. Most of the parking areas today in India currently operate without or with a small computerized system using a small database. They usually require vehicle owners to roam around and manually check the occupancy of individual spots. The problem is usually encountered in metropolitan, where number of vehicles is higher as compared to the number of parking spaces available. Implementing proposed system will help to resolve the problem of finding a parking spot in a huge parking lot, the time wasted in finding the spot and help provide better public service. It will improve visitor's experience, increase parking utilization and prevent unnecessary capital investments.

The owners are concerned that they are not maximizing profit due the inefficient managements of parking slots. This project aims to examine the development of smart parking solution using image processing to detect the vacant spaces in the parking lot. In the system, CCTV cameras will be deployed in the parking lot, which detects and monitors the occupancy of the parking lot. Features of this system include vacant parking space detection, display of available parking spaces and different types of parking spaces namely vacant and occupied.

The solution consists of a Graphic User Interface which shows the status of vacant and filled parking spots in the parking lot. Upon entrance, the number plate of the vehicle will be used to extract the vehicle number of the vehicle. .Check-in and check-out time will be recorded and the customer will be charged accordingly through e-wallet/UPI/card payment. The feasibility of the proposed model has been analyzed through financial, technical and operational perspectives and it was discovered that digital image processing technology based solution has the possibility to rationalize the vehicle parking management system. Software used is Matlab.

Acknowledgment

We hereby take this opportunity to express our profound gratitude and deep regards to our mentor Dr. Navneet Yadav, Assistant Professor, ECE department for his cordial support, valuable information and guidance which helped us in completing the tasks through various stages.

We also take the opportunity to express our deep sense of gratitude to our families and friends who have been a constant source of inspiration. We are internally grateful to them for always encouraging us wherever and whenever we needed them.

Anmol Deepak
(08114802813)

Ashutosh Mishra
(01714802813)

Gaurav Singh
(01814802813)

Ronak Lalani
(00714802813)

Table of Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 Digital Image Processing Fundamentals

1.1.1 Image

An image may defined as a 2-dimensional function, f(x,y), where x & y are spatial (plane) coordinates, & the amplitude of at any pair of coordinates (x,y) is called the intensity or gray level of the image at that point. When x,y& the amplitude values of are all finite, discrete quantities, we call the image as Digital image.

A digital image differs from a photo in that the values are all discrete. A digital image can be considered as a large array of discrete dots, each of which has a brightness associated with it. These dots are called picture elements, or more simply pixels.

Fig 1.1 Image

The above figure is an example of digital image that you are now viewing on your computer screen. But actually, this image is nothing but a two dimensional array of numbers ranging between 0 and 255.

1.1.2 Digital Image Formation

Since capturing an image from a camera is a physical process. The sunlight is used as a source of energy. A sensor array is used for the acquisition of the image. So when the sunlight falls upon the object, then the amount of light reflected by that object is sensed by the sensors, and a continuous voltage signal is generated by the amount of sensed data. In order to create a digital image, we need to convert this data into a digital form. This involves sampling and quantization. (They are discussed later on). The result of sampling and quantization results in a two dimensional array or matrix of numbers which are nothing but a digital image

1.1.3 Image sharpening and restoration

Image sharpening and restoration refers here to process images that have been captured from the modern camera to make them a better image or to manipulate those images in way to achieve desired result. It refers to do what Photoshop usually does.

This includes Zooming, blurring, sharpening, gray scale to color conversion, detecting edges and vice versa , Image retrieval and Image recognition. The common examples are:

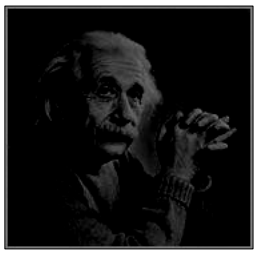


Fig 1.2 Normal Image

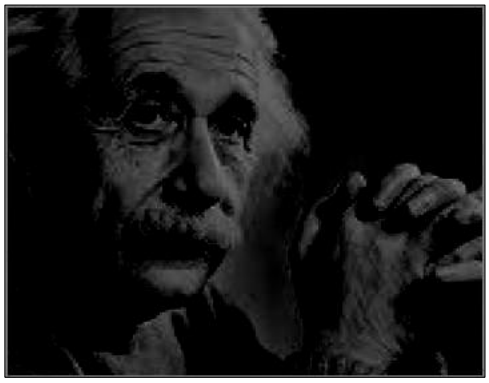


Fig 1.3 Zoomed Image

Fig 1.4 Blurr Image

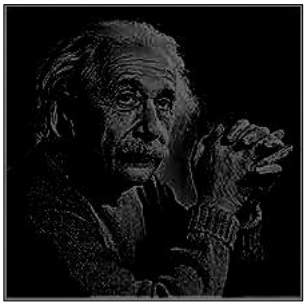


Fig 1.5 Sharp Image

Fig. 1.6 Edges

1.1.4 Applications of Digital Image Processing

Some of the major fields in which digital image processing is widely used are mentioned below

- Image sharpening and restoration
- Medical field
- Remote sensing
- Transmission and encoding
- Machine/Robot vision
- Color processing
- Pattern recognition
- Video processing
- Microscopic Imaging
- Others

The common applications of DIP in the field of medical science are

- Gamma ray imaging
- PET scan
- X Ray Imaging
- Medical CT

1.1.5 Types of Images

Binary: Each pixel is just black or white. Since there are only two possible values for each pixel (0, 1), we only need one bit per pixel.

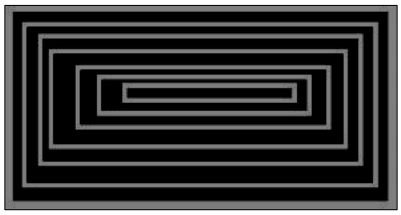


Fig 1.7 Binary Image

Grayscale: Each pixel is a shade of gray, normally from 0 (black) to 255 (white). This range means that each pixel can be represented by eight bits, or exactly one

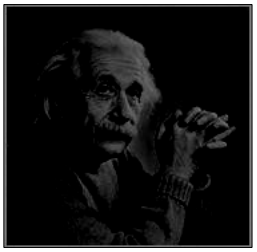


Fig 1.8 Grayscale Image

Byte: Other grey scale ranges are used, but generally they are a power of 2.

True Colour, or RGB: Each pixel has a particular color; that color is described by the amount of red, green and blue in it. If each of these components has a range 0–255, this gives a total of 2563 different possible colors. Such an image is a "stack"of three matrices; representing the red, green and blue values for each pixel. This means that for every pixel there correspond 3 values.

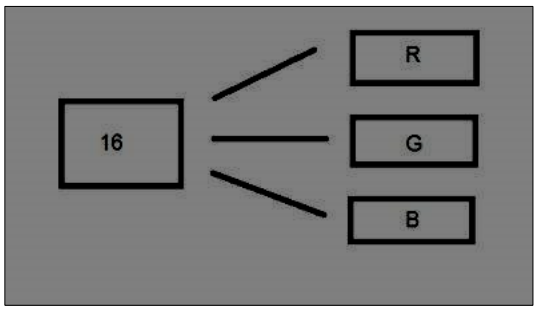


Fig 1.9 RGB Image – 1

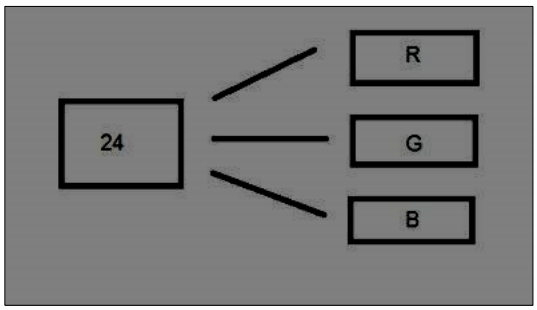


Fig 1.10 RGB Image- 2

1.2 Project Introduction

Problem Statement

The problems that have been identified are stated below:

- Driver needs some relevant information before entering the parking lot such as if there are available parking spaces in the parking lot or not.
- The current system used in parking lot is based on sensors installed at each space which is highly inefficient as the number of sensors increase with the increasing size of parking lot which would also contribute to increasing costs.
- Time wasted by driver in finding an available parking space. Imagine if the parking lot is multi-level, driver will have to sail through all parking spaces just to find an available parking space.

Objective

The main objectives behind developing the smart parking solutions using Image Processing are as follows:

- Count and display available parking spaces and their respective locations
- Assign the incoming vehicle, a parking spot nearest to their point of entrance.
- Use the check-in and check-out time to calculate the duration for which user need to be charged.
- Show the parking charges to be paid.

Assumptions

Some of the assumptions that are being taken while developing the prototype of the systems are:

- This system is just a prototype system using image processing techniques.
- Using image that is captured from smartphone camera 0.5 Megapixels.
- The position of the parked vehicle is dark as compared the vacant space which is light.
- The system can be used in daytime only without having a strong shadow.

Chapter 2

FILTERS AND ALGORITHMS USED

2.1 Filtering

Filters are used in filtering process to perform different kind of processing on an image. The processing include blurring an image, sharpening an image etc. Filtering is used in the project for the purpose of Blurring and Smoothing. Image filtering can be grouped into following types depending on the effects:

Low pass filters (Smoothing)

The most basic of filtering operations is called "low-pass". A low-pass filter, also called a "blurring" or "smoothing" filter, averages out rapid changes in intensity. The simplest low-pass filter just calculates the average of a pixel and all of its eight immediate neighbors. The result replaces the original value of the pixel. Low pass filtering, is employed to remove high spatial frequency noise from a digital image. The low-pass filters usually employ moving window operator which affects one pixel of the image at a time, changing its value by some function of a local region (window) of pixels. The process is repeated for every pixel in the image.

High pass filters (Edge Detection, Sharpening)

A high-pass filter can be used to make an image appear sharper. These filters emphasize fine details in the image – exactly the opposite of the low-pass filter. High-pass filtering works in exactly the same way as low-pass filtering; it just uses a different convolution kernel. If there is no change in intensity, nothing happens. But if one pixel is brighter than its immediate neighbors, it gets boosted. Unfortunately, while low-pass filtering smooths out noise, high-pass filtering does just the opposite: it amplifies noise. You can get away with this if the original image is not too noisy; otherwise the noise will overwhelm the image.

High-pass filtering can also cause small, faint details to be greatly exaggerated. An over-processed image will look grainy and unnatural, and point sources will have dark donuts around them. So while high-pass filtering can often improve an image by sharpening detail, overdoing it can actually degrade the image quality significantly.

Gaussian filter

In electronics and signal processing, a Gaussian filter is a filter whose impulse response is Gaussian function (or an approximation to it). Gaussian filters have the properties of having no overshoot to a step function input while minimizing the rise and fall time. This behavior is closely connected to the fact that the Gaussian filter has the minimum possible group delay. It is considered the ideal time domain filter, just as the sin is the ideal frequency domain filter. These properties are important in areas such as oscilloscopes and digital telecommunication systems. Mathematically, a Gaussian filter modifies the input signal by convolution with a Gaussian function. Due to the central limit theorem, the Gaussian can be approximated by several runs of a very simple filter such as the moving average.

Gaussian Low pass filter

The concept of Gaussian low pass filter and low pass filter remains the same, but only the transition becomes different and become smooth. There is a problem known as ringing effect which happens because at some points transition between one colour to the other cannot be defined precisely, due to which the ringing effect appears at that point. It minimizes the problem that occur in ideal low pass filter.

Gaussian high pass filter

Gaussian high pass filter has the same concept as ideal high pass filter, but again the transition is smooth as compared to the ideal one. There is a problem known as ringing effect which happens because at some points transition between one color to the other cannot be defined precisely, due to which the ringing effect appears at that point. It minimizes the problem that occur in ideal high pass filter.

2.2 Number Plate Recognition

This algorithm is used to recognize the vehicle number of the vehicle. It's a 4 step process which is explained as follows:

1. **Vehicle Image Captured By Camera**
   The image of the vehicle whose number plate is to be identified is captured using digital camera of 0.5 megapixel.

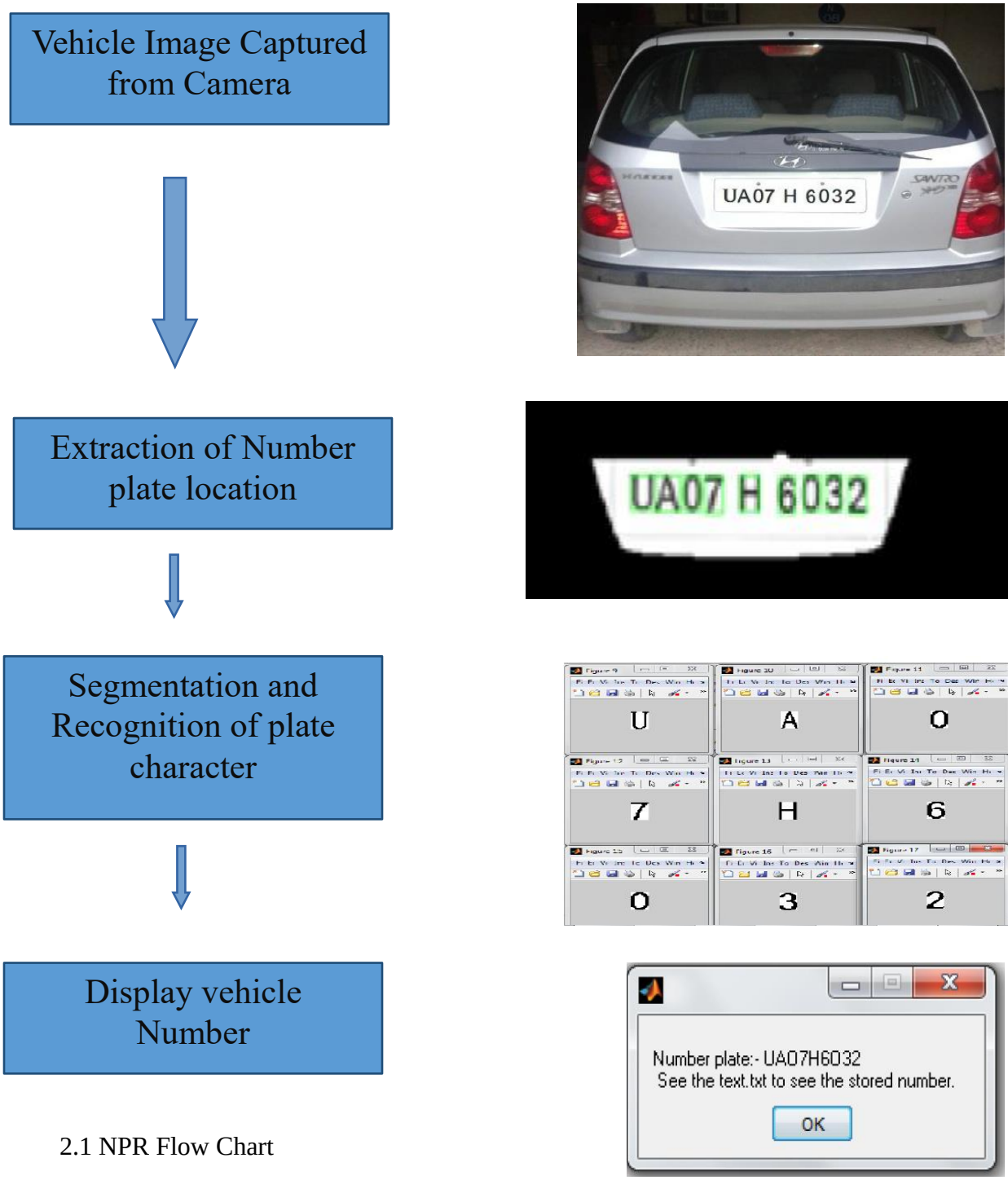
2. **Extraction of Number Plate Location**
   The basic step in recognition of vehicle number plate is to detect the plate size. In general number plates are in rectangular shape, hence it is necessary to detect the edges of the rectangular plate. In this step the number plate is extracted by firstly converting RGB Image i.e., the captured image to Gray Scale Image. Here mathematical morphology is used to detect the region and Sobel operator are used to calculate the threshold value. After this we get a dilated image. Then ***imfill*** function is used to fill the holes so that we get a clear binary image.

3. **Segmentation and Recognition of Plate Character**
   Segmentation is one of the most important processes in the number plate recognition, because all further steps rely on it. If the segmentation fails, a character can be improperly divided into two pieces, or two characters. The ultimate solution on this problem is to use bounding box technique. The bounding box is used to measure the properties of the image region. Once a bounding box created over each character and numbers presented on number plate, each character & number is separate out for recognition of number plate. The bounding box technique is used for segmentation. The bounding box is used to measure the properties of the image region.

4. **Display Vehicle Number**

   It is employed for the purpose of conversion of images of text into characters. Number plate recognition is now used to compare the each individual character against the complete alphanumeric database using template matching. The matching process moves the template image to all possible positions in a larger source image and computes a numerical index that indicates how well the template matches the image in that position. Matching is done on a pixel by pixel basis. The template is of size 42 × 24. Since the template size is fixed, it leads to accurate recognition. After undergoing the above steps the number plate is displayed in MATLAB window.

2.2.1 Procedure



2.1 NPR Flow Chart

Chapter 3

SMART PARKING IMPLEMENTATION

3.1 Objective

The whole project is divided into two modules. Some of the objectives behind developing the smart parking solutions using Image Processing are as follows:

- Capture and count number of vehicle already parked at the parking lot using image processing technique.
- Count and display number of available parking space and the location of the available parking spaces in parking lot.
- Assign the incoming vehicles, a parking spot nearest to their current location.

3.2 Assumptions

There are some assumptions that have been made in order to develop the system. Some of them are:

- This system is just a prototype system using image processing techniques.
- Using image that is captured from smartphone camera 0.5 Megapixels.
- The position of the parked vehicle is dark as compared the vacant space which is light.
- The system can be used in daytime only without having a strong shadow.

3.3 Flow chart

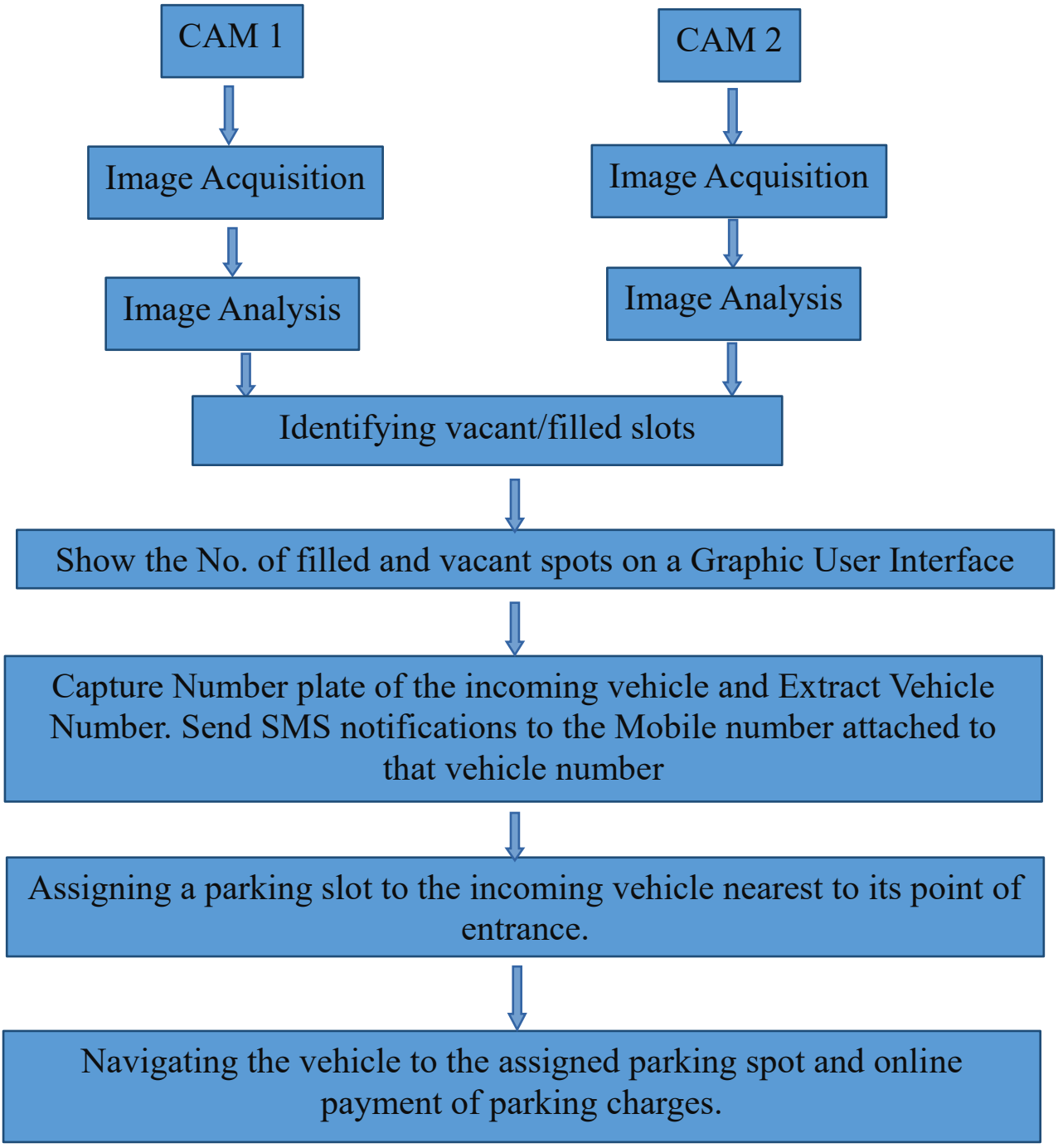


Fig 3.1 Overall Flow chart

3.4 Module 1: No. of Filled/Vacant Spaces

3.4.1 Flow Chart

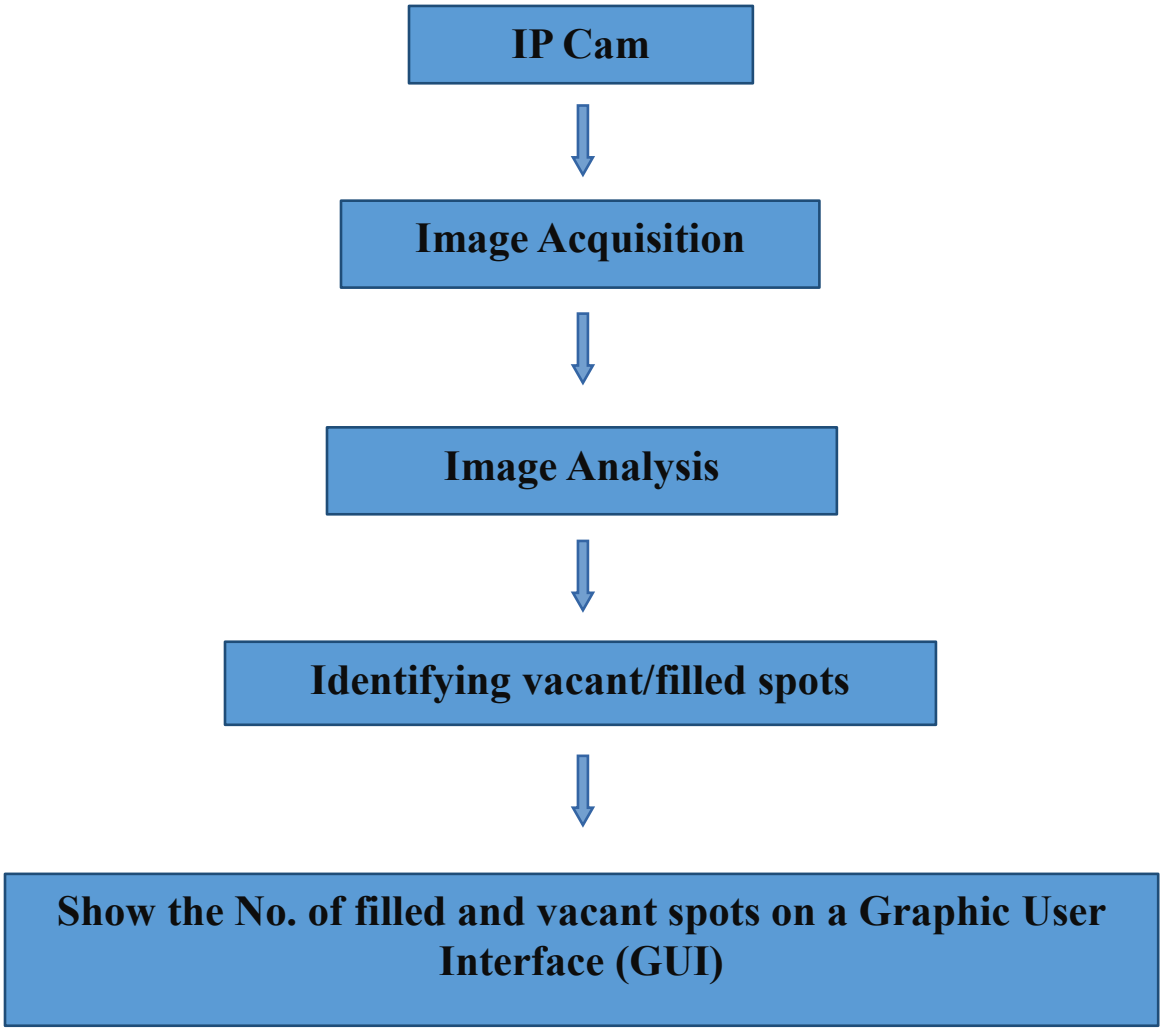


Fig 3.2 Flow chart (Module -1)

3.4.2 Number of Filled and Vacant Spots

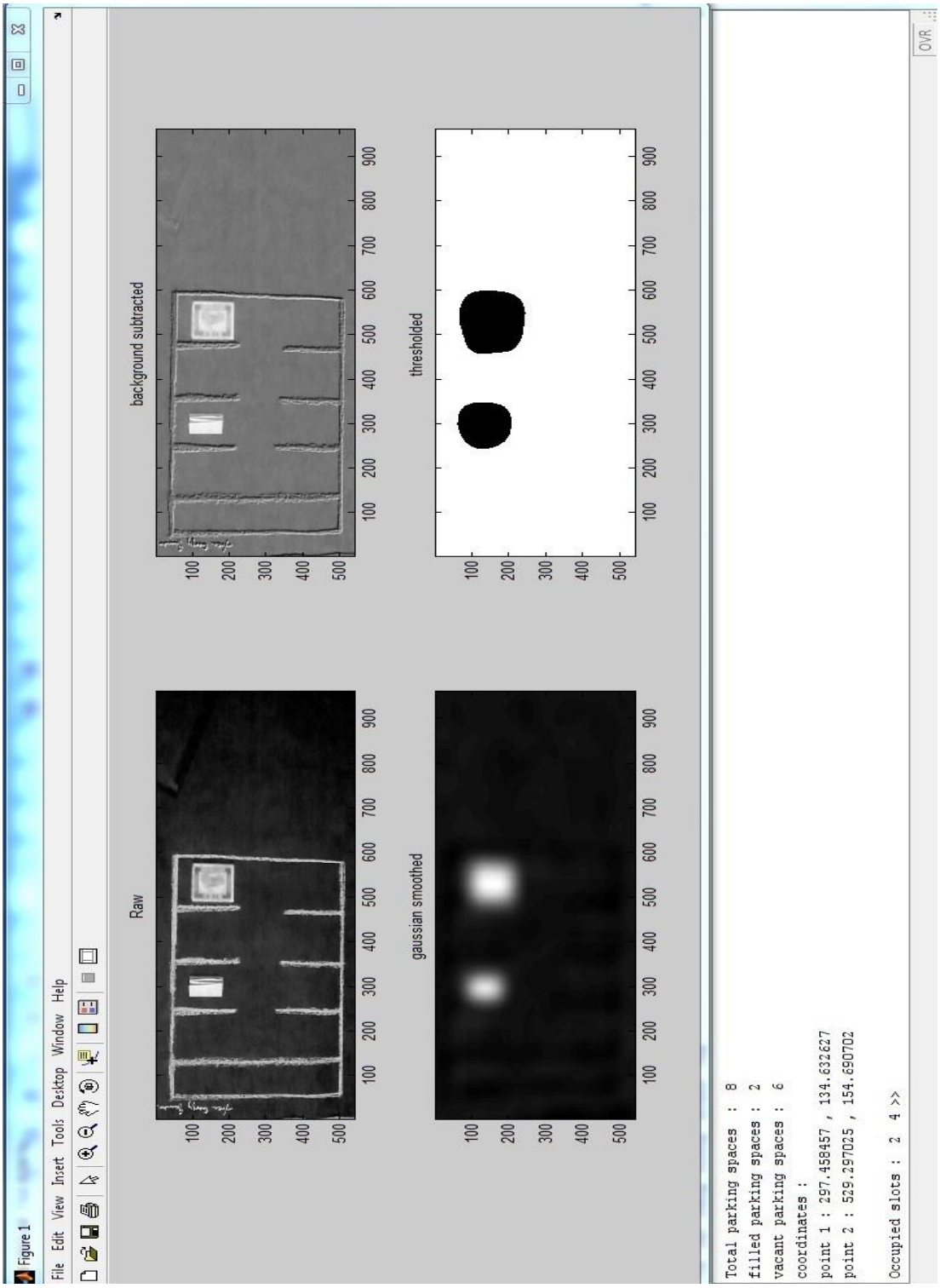


Fig 3.3 Filled/Vacant Spots

3.5 Module 2: Vehicle tracking

3.5.1 Flow Chart

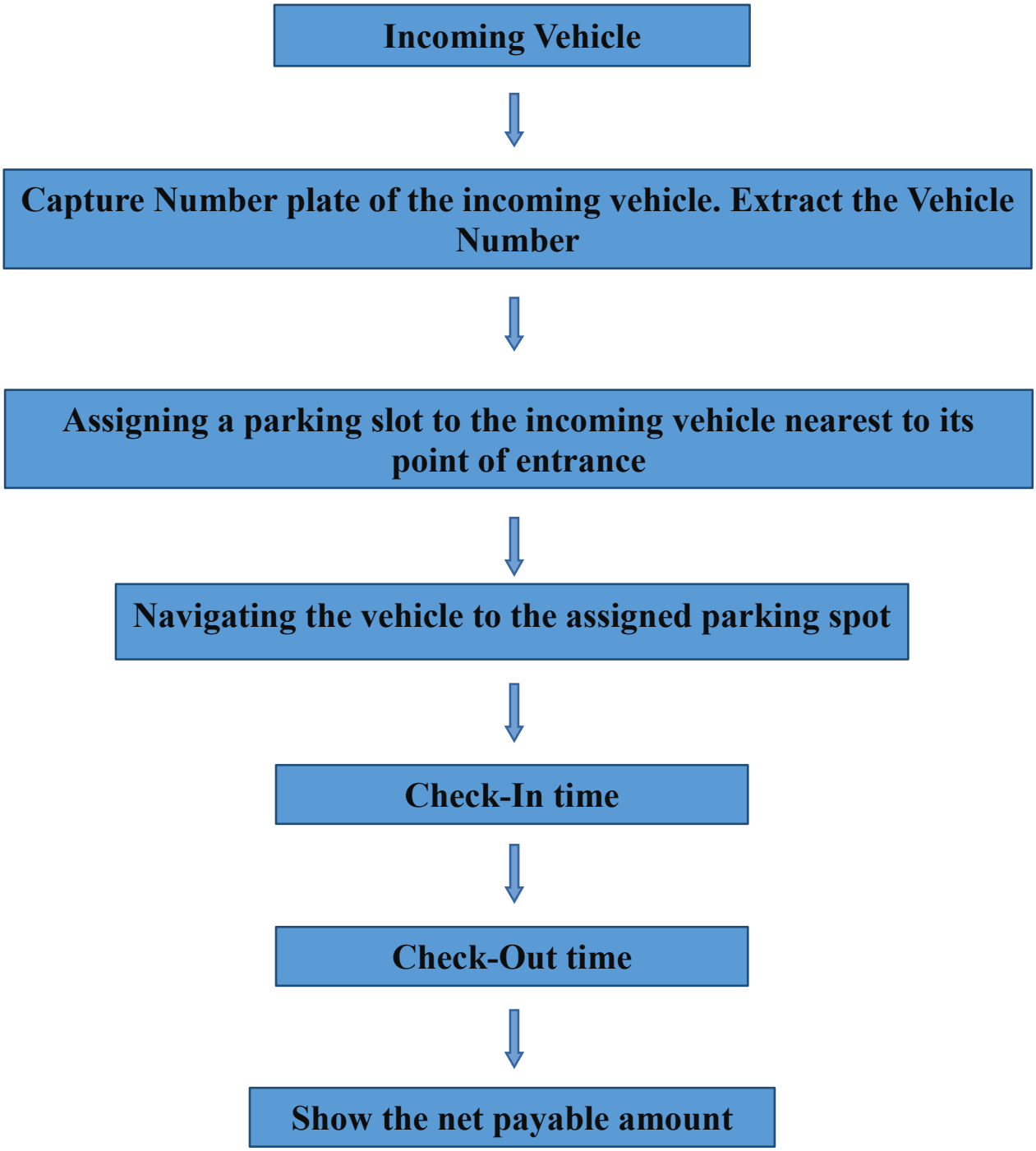


Fig 3.4 Flow chart (Module -2)

3.5.2 Vehicle Number Plate Recognition

3.5.2.1 Captured Image

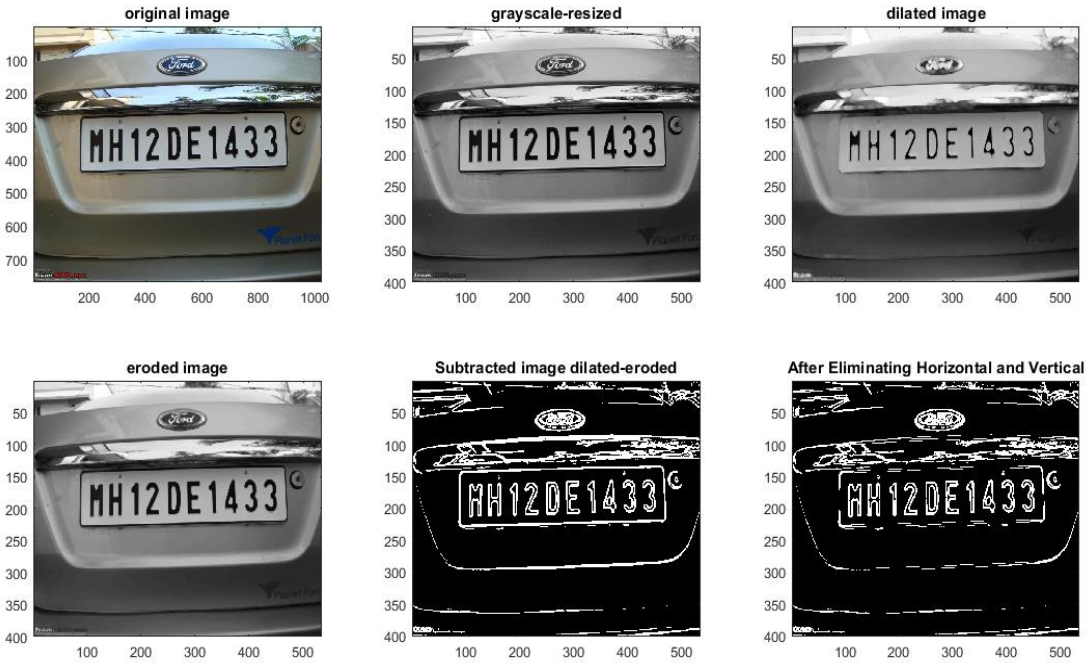


Fig 3.5 Captured Image

3.5.2.2 Extraction of Number Plate Location

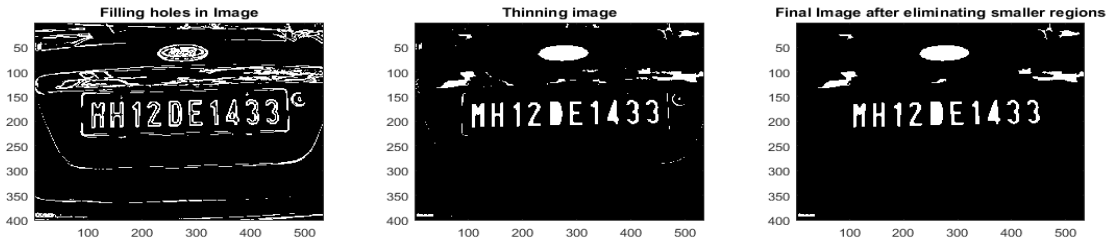


Fig 3.6 Number plate location extraction

3.5.2.3 Segmentation and Recognition of Plate Character

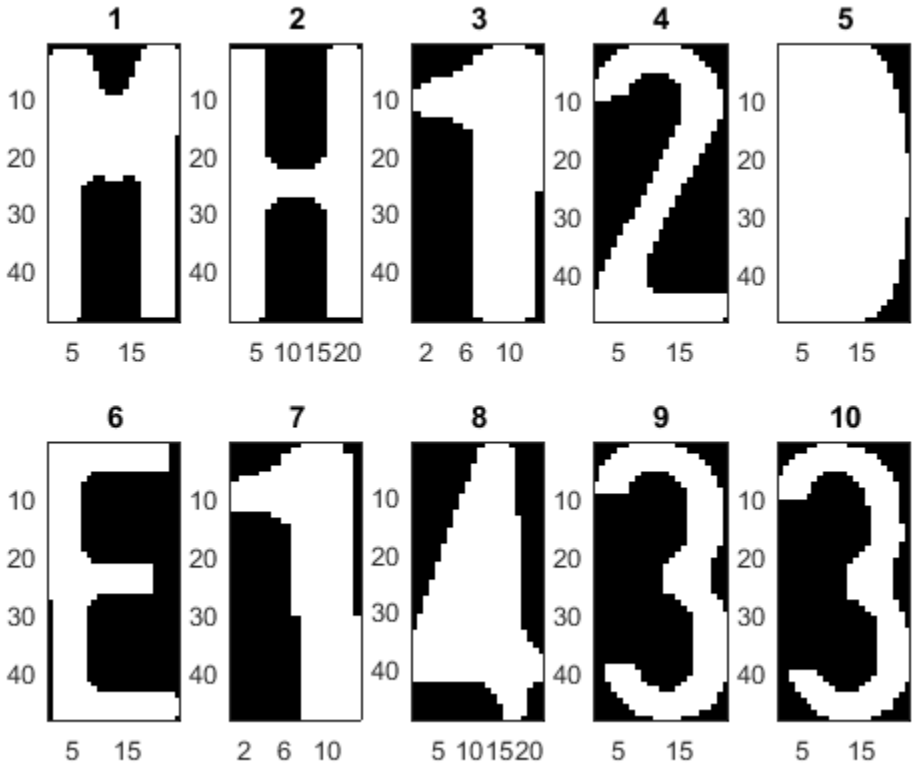


Fig 3.7 Segmentation & recognition

Chapter 4

GRAPHICAL USER INTERFACE

4.1 Front Screen

Fig 4.1 Front Screen

4.2 Live Camera Feed

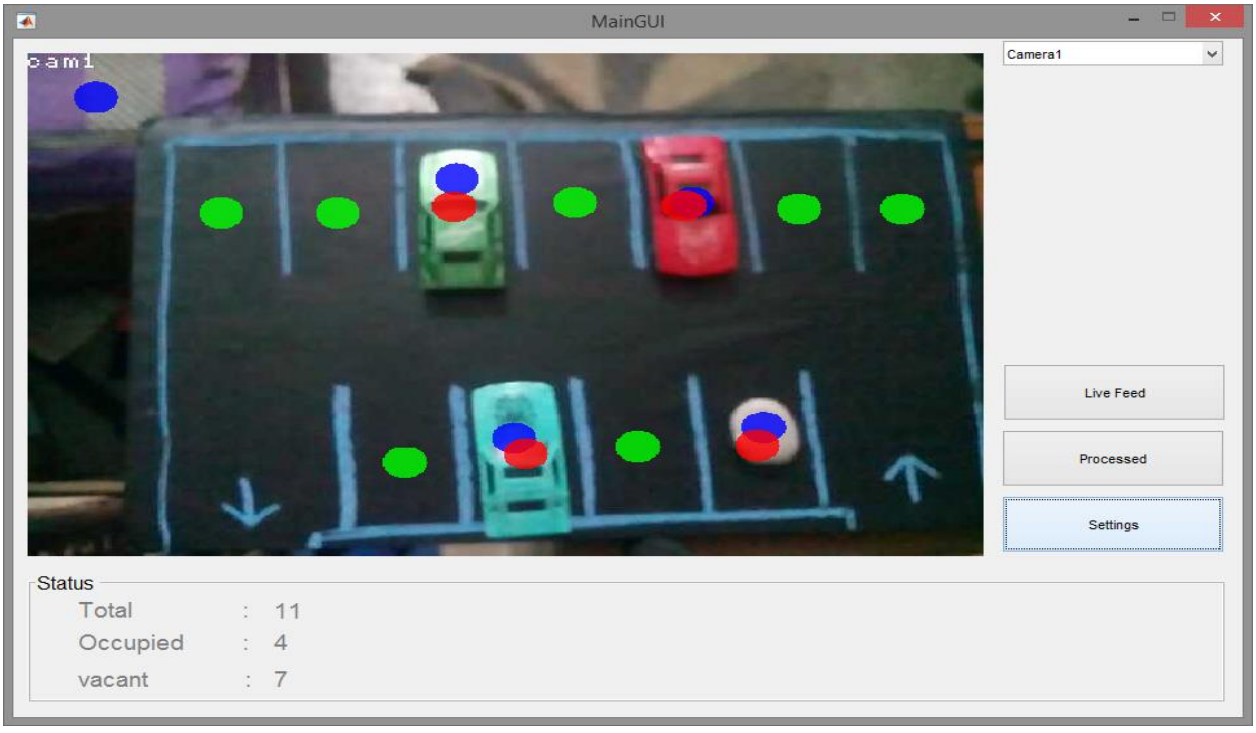


Fig 4.2 Live Camera Feed

4.3 Different Camera Options

Fig 4.3 Different camera options

4.4 Software Settings

Fig 4.4 Setting

Chapter 5

DATABASES

DB 1: Current

Sr. No	Vehicle Number	Type of Vehicle (Two- wheeler, Car, Electric, Cab, Auto)	Check- In (Time and Date) (yyyy-mm-dd-hh-mm)	Fixed Charges
1.	DL 3C 1995	Car	2017-03-15 16:43	50

Table 5.1 Database 1: Current

DB 2: Final

Sr. No	Vehicle Number	Type of Vehicle (Two- wheeler, Car, Electric, Cab, Auto)	Check- Out (Time and Date) (yyyy-mm-dd-hh-mm)	Final Charges
1.	DL 3C 1995	Car	2017-03-15 18:43	90

Table 5.2 Database 2: Final

Chapter 6

CONCLUSION

Smart Parking is at a very tender stage in India and people hardly know about the technology. They cannot distinguish between smart parking and automated parking which is already very prevalent. The number of people using their own vehicles has increased exponentially in the past ten or fifteen years. The vehicle parking has become an enormous matter especially in urban areas.

In view of increasing vehicle quantity, the proposed solution introduces a smart vehicle parking management based on Digital Image processing which makes parking much easier for drivers.

It will solve many problems like

1. Efficient use of available Space
2. Wastage of time
3. Wastage of fuel
4. Security to the vehicle.

The main contribution of this study is to introduce the most significant parking space problem (finding a vacant slot) and propose a solution. Digital image processing is used to detect parking spaces. The operator can handle the customer data. The proposed architecture for a parking detection system would decrease searching time for vacant spaces and direct the vehicle to the nearest vacant parking spot.

The system has several advantages such as high efficiency, low cost, high security etc. It solves all the issues related to vehicle parking such as finding free parking slots, improved demand system and certainly the security issues. So the main objective of this project is to implement a system to find parking slots in an efficient manner. Due to the difficulty of data handling process the proposed system will give a better solution for the all processing data which is integrated through a central server based database which can be accessed by the parking operator.

REFERENCES

[1] GGYU Gunasekara, ADAI Gunasekara and RPS Kathriarachchi (2015), 'A Smart Vehicle Parking Management Solution'. *Proceedings of 8th International Research Conference, KDU, Published November 2015*: 106-110.

[2] http://www.joneslanglasalleblog.com/realestatecompass/real-state/2014 / 06 / managing-parking-issues (automated- parking-solutions).

[3] Kastrinaki, V, Zervakis, M & Kalaitzakis, K 2003, A survey of video processing techniques for traffic applications, *Image Vision Computing*., 21: 359-381, viewed 10 March 2015.

[4] Khan, R, Shah, Y, Khan, Z, Ahmed, K, Manzoor M & Ali, A 2013, Intelligent Car Parking Management System On FPGA, *International Journal of Computer Science*, Vol. 10, Issue 1, no.3, 171- 174

[5] http://www.engineeringcivil.com/different-types-of-parking-spaces-and-multiple-level-car-parking.html

[6] http://smartcitiescouncil.com/article/smart-parking-rise-1-million-spaces-2020

[7] http://www.navigantresearch.com/newsroom/smart-parking-systems-will-reach-nearly-360-million-in-annual-revenue-by-2020

[8] http://indiatransportportal.com/2012/01/our-experience-on-parking-management-can-help-to-achieve-a-better-fluidity-of-the-traffic-s-sundar-ceo-smart-parking-india/

[9] http://www.parking-net.com/parking-industry/hectronic-india-retail-parking-automation-pvt-ltd

[10] W. Sanngoen, O. Akihisa and T. Takashi, "Parking Place Inspection System Utilizing a Mobile Robot with a Laser Range Finder", **(2012)**.

[11] N. Bandu, R. Ranjana and D. Pravin, "Performance Evaluation of Modern Sophisticated Parking Management System with Space Modeling", *International Journal of Innovative Research in Computer and Communication Engineering*, *vol. 2*,no. 11, (2014), p. 8.

[12] Z. Pala and N. Inan, "Smart Parking Applications Using RFID Technology", vol. 3, **(2007)**.

APPENDIX - 1

Matlab Code – Module 1

Main GUI

```
function varargout = MainGUI(varargin)
%% Initialization code
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @MainGUI_OpeningFcn, ...
    'gui_OutputFcn',  @MainGUI_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code

function MainGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for MainGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

if strcmp(get(hObject,'Visible'),'off')
    initial = imread('initialsp1.png');
    imagesc(initial);
    axis off;
end

function varargout = MainGUI_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

%Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

set(hObject, 'String', {'Incamera','Outcamera','Camera3',
'Camera4','Camera5','Camera6','Camera7'});

function popupmenu1_Callback(hObject, eventdata, handles)

function pushbutton1_Callback(hObject, eventdata, handles)
axes(handles.axes1);
cla;
load('slots.mat');
```

```
popup_sel_index = get(handles.popupmenu1, 'Value');


switch popup_sel_index
    case 1
        try
            vid = ipcam(camurl1);
            img = snapshot(vid);
            handles.vid=vid;
            hImage = image( zeros(size(img)) );
            preview(handles.vid, hImage)
        catch E
            msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
        end
        guidata(hObject, handles);

    case 2
        try
            vid = ipcam(camurl2);
            img = snapshot(vid);
            handles.vid=vid;
            hImage = image( zeros(size(img)) );
            preview(handles.vid, hImage)
        catch E
            msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
        end
        guidata(hObject, handles);

    case 3
        try
            vid = ipcam(camurl3);
            img = snapshot(vid);
            handles.vid=vid;
            hImage = image( zeros(size(img)) );
            preview(handles.vid, hImage)
        catch E
            msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
        end
        guidata(hObject, handles);

    case 4
        try
            vid = ipcam(camurl4);
            img = snapshot(vid);
            handles.vid=vid;
            hImage = image( zeros(size(img)) );
            preview(handles.vid, hImage)
        catch E
            msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
        end
        guidata(hObject, handles);

    case 5
        try
            vid = ipcam(camurl5);
            img = snapshot(vid);
            handles.vid=vid;
            hImage = image( zeros(size(img)) );
            preview(handles.vid, hImage)
        catch E
            msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
```

```
end
        guidata(hObject, handles);

    case 6
        try
            vid = ipcam(camurl6);
            img = snapshot(vid);
            handles.vid=vid;
            hImage = image( zeros(size(img)) );
            preview(handles.vid, hImage)
        catch E
            msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
        end
        guidata(hObject, handles);

    case 7
        try
            vid = ipcam(camurl7);
            img = snapshot(vid);
            handles.vid=vid;
            hImage = image( zeros(size(img)) );
            preview(handles.vid, hImage)
        catch E
            msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
        end
        guidata(hObject, handles);
end

function pushbutton2_Callback(hObject, eventdata, handles)
axes(handles.axes1);
cla;
load('slots.mat');
popup_sel_index = get(handles.popupmenu1, 'Value');
switch popup_sel_index
    case 1
        try
            vid = ipcam(camurl1);
            guidata(hObject, handles);
            img = snapshot(vid);
            handles.vid=vid;
            hImage = image( zeros(size(img)));
            preview(handles.vid, hImage)
            [numberplate,class] = numberPlateExtraction1(img);
            %set the status panel
            set(handles.text1, 'String','Vehicle No.              :');
            set(handles.text2, 'String','Vehicle Class         :');
            set(handles.text3, 'String','Time of Entry         :');
            set(handles.text4, 'String',numberplate);
            set(handles.text5, 'String',class);
            set(handles.text6, 'String',datetime('now'));

            imagesc(img);
            axis off;
            pause(5)

      catch E
            msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
        end

    case 2
        try
```

```
       %vid = ipcam('http://192.168.0.3:8080/video');
        vid = ipcam(camurl2);
        bck_image = double(imread('refcam2.jpg'));
        bck_img = bck_image(:,:,1);
        nodes = load('slots.mat', 'corcam2');
        nodes = nodes.corcam2;
        tps=size(nodes);
        tps=tps(1);

    catch E
        msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
    end

case 3
    try
        % vid = ipcam('http://192.168.0.2:8080/video');
        vid = ipcam(camurl3);
        bck_image = double(imread('refcam3.jpg'));
        bck_img = bck_image(:,:,1);
        nodes = load('slots.mat', 'corcam3');
        nodes = nodes.corcam3;
        tps=size(nodes);
        tps=tps(1);

    catch E
        msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
    end
case 4
    try
        % vid = ipcam('http://192.168.0.2:8080/video');
        vid = ipcam(camurl4);
        bck_image = double(imread('refcam4.jpg'));
        bck_img = bck_image(:,:,1);
        nodes = load('slots.mat', 'corcam4');
        nodes = nodes.corcam4;
        tps=size(nodes);
        tps=tps(1);

    catch E
        msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
    end
case 5
    try
        % vid = ipcam('http://192.168.0.2:8080/video');
        vid = ipcam(camurl5);
        bck_image = double(imread('refcam5.jpg'));
        bck_img = bck_image(:,:,1);
        nodes = load('slots.mat', 'corcam5');
        nodes = nodes.corcam5;
        tps=size(nodes);
        tps=tps(1);

    catch E
        msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
    end
case 6
    try
        % vid = ipcam('http://192.168.0.2:8080/video');
        vid = ipcam(camurl6);
        bck_image = double(imread('refcam6.jpg'));
```

```
            bck_img = bck_image(:,:,1);
            nodes = load('slots.mat', 'corcam6');
            nodes = nodes.corcam6;
            tps=size(nodes);
            tps=tps(1);

        catch E
            msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
        end
    case 7
        try
            % vid = ipcam('http://192.168.0.2:8080/video');
            vid = ipcam(camurl7);
            bck_image = double(imread('refcam7.jpg'));
            bck_img = bck_image(:,:,1);
            nodes = load('slots.mat', 'corcam7');
            nodes = nodes.corcam7;
            tps=size(nodes);
            tps=tps(1);

        catch E
            msgbox({'Configure The Cam Correctly!',' ',E.message},'CAM INFO')
        end

end
guidata(hObject, handles);
img = snapshot(vid);
handles.vid=vid;
hImage = image( zeros(size(img)));
preview(handles.vid, hImage)

processFlag=1;
if(popup_sel_index==1 || popup_sel_index==2)
    processingFlag=0;
end

    while processFlag==1
        %% from raw color image to binary image with highlighted occupied
spots
        himage2 = snapshot(vid);
        num3 = load('slots.mat', 'thres');num3 = str2double(num3.thres);
%threshold
        num4 = load('slots.mat', 'eliminate');num4=
str2double(num4.eliminate);   %bwareopen
        hsize = load('slots.mat', 'hsize');hsize= str2double(hsize.hsize);
%gaussian filter
        sigma = load('slots.mat', 'sigma');sigma=str2double(sigma.sigma);
        gaus_filt = fspecial('gaussian',hsize , sigma);

        img_tmp = double(himage2);                                          %load
image and convert to double for computation
        img2 = img_tmp(:,:,1);                                              %reduce
to just the first dimension
        sub_img = (img2 - bck_img);
%subtract background from the image
        gaus_img = filter2(gaus_filt,sub_img,'same');
%gaussian blurr the image
thres_img = (gaus_img < num3);
```

```
        thres_img = ~thres_img;


thres_img = bwareaopen(thres_img,num4);

        thres_img = ~thres_img;
        se2 = strel('disk',1);
        thres_img = imerode(thres_img,se2);

        %% counting no of blobs
        thres_img = ~thres_img;
        [L, num] = bwlabel(thres_img);
        stats = regionprops(L, 'Centroid');
        thres_img = ~thres_img;

        %% highlight strange objects
        img3 = himage2;
        % circle parameters
        r = 15;
% radius
        t = linspace(0, 2*pi, 20);
% approximate circle with 50 points

        num2 = 0;
        for k=1:1:tps
            c = [nodes(k,1) nodes(k,2)];                    % center
            BW = poly2mask(r*cos(t)+c(1), r*sin(t)+c(2), size(img,1),
size(img,2));% create a circular mask
            if thres_img(nodes(k,2), nodes(k,1))== 0
                clr = [255 0 0 ];            % Red color for circle
                num2 = num2 + 1;             % counting no of filled spots
            else
                clr = [0 255 0 ];            % Green color for circle
            end
            a = 0.8;                         % blending factor
            z = false(size(BW));
            mask = cat(3,BW,z,z); img3(mask) = a*clr(1) + (1-a)*img3(mask);
            mask = cat(3,z,BW,z); img3(mask) = a*clr(2) + (1-a)*img3(mask);
            mask = cat(3,z,z,BW); img3(mask) = a*clr(3) + (1-a)*img3(mask);
        end
        %% Results
        %set the status panel
        set(handles.text1, 'String','Total                :');
        set(handles.text2, 'String','Occupied         :');
        set(handles.text3, 'String','vacant              :');
        set(handles.text4, 'String',num2str(tps));
        set(handles.text5, 'String',num2str(num2));
        set(handles.text6, 'String',num2str(tps-num2));

        imagesc(img3);
        axis off;
        pause(.1)

    end

    function pushbutton3_Callback(hObject, eventdata, handles)
    %closePreview(vid);
    processFlag=0;
    Settings()
```

Camera Configuration

```
function varargout = cameraconfig(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @cameraconfig_OpeningFcn, ...
                   'gui_OutputFcn',  @cameraconfig_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before cameraconfig is made visible.
function cameraconfig_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for cameraconfig
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes cameraconfig wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = cameraconfig_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
try
a=load('slots.mat', 'camurl1');
set(hObject,'String',a.camurl1)
end

function edit2_Callback(hObject, eventdata, handles)
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end
try
a=load('slots.mat', 'camurl2');
set(hObject,'String',a.camurl2)
end

function pushbutton1_Callback(hObject, eventdata, handles)
camurl1 = get(handles.edit1,'String');
save('slots.mat', 'camurl1','-append');
camurl2 = get(handles.edit2,'String');
save('slots.mat', 'camurl2','-append');
camurl3 = get(handles.edit3,'String');
save('slots.mat', 'camurl3','-append');
camurl4 = get(handles.edit4,'String');
save('slots.mat', 'camurl4','-append');
camurl5 = get(handles.edit5,'String');
save('slots.mat', 'camurl5','-append');
camurl6 = get(handles.edit6,'String');
save('slots.mat', 'camurl6','-append');
camurl7 = get(handles.edit7,'String');
save('slots.mat', 'camurl7','-append');
close(cameraconfig);



function edit3_Callback(hObject, eventdata, handles)
function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
try
a=load('slots.mat', 'camurl3');
set(hObject,'String',a.camurl3)
end



function edit4_Callback(hObject, eventdata, handles)
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
try
a=load('slots.mat', 'camurl4');
set(hObject,'String',a.camurl4)
end



function edit5_Callback(hObject, eventdata, handles)
function edit5_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
try
```

```
a=load('slots.mat', 'camurl5');
set(hObject,'String',a.camurl5)
end



function edit6_Callback(hObject, eventdata, handles)
function edit6_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
try
a=load('slots.mat', 'camurl6');
set(hObject,'String',a.camurl6)
end



function edit7_Callback(hObject, eventdata, handles)
function edit7_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
try
a=load('slots.mat', 'camurl7');
set(hObject,'String',a.camurl7)
end
```

APPENDIX - 2

Matlab Code – Module 2

Number Plate Extraction

```
function [noPlate,VehicleClass] = numberPlateExtraction1(img)
%% Input Image
%f=imread('10.jpg');
f=img;
figure(1)
subplot(231);imagesc(f);
title('original image');

% Resizing the image keeping aspect ratio same.
f=imresize(f,[400 NaN]);

% Converting the RGB (color) image to gray (intensity).
g=rgb2gray(f);
colormap(gray)
subplot(232);imagesc(g);
title('grayscale-resized');
% Median filtering to remove noise.
g=medfilt2(g,[3 3]);
%% Morphological processing
se=strel('disk',1);
% Dilation
gi=imdilate(g,se);
gi=imdilate(gi,se);
%gi=imdilate(gi,se);
subplot(233);imagesc(gi);
title('dilated image');
% Erosion
ge=g;
%ge=imerode(ge,se);
subplot(234);imagesc(ge);
title('eroded image');

% Morphological Gradient for edges enhancement.
gdiff=imsubtract(gi,ge);
% Converting the matrix to double image.
gdiff=mat2gray(gdiff);
% Convolution of double image for brightening edges.
gdiff=conv2(gdiff,[1 1;1 1]);
% Adjust the contrast of image, specifying contrast limits. Intensity scaling
between the range 0 to 1.
gdiff=imadjust(gdiff,[0.5 0.7],[0 1],0.1);
% Conversion from double to binary.
B=logical(gdiff);
% Here B is the final b/w image of car for extracting number plate
subplot(235);imagesc(B);
title('Subtracted image dilated-eroded');

er=imerode(B,strel('line',50,0));
% deletiong horizontal lines from b/w image B
out1=imsubtract(B,er);
% locating vertical lines
er=imerode(out1,strel('line',50,90));
```

```
% Filling all the regions of the image.
F=imfill(out1,'holes');
figure(2)
colormap('gray')
subplot(231);imagesc(out1);
title('Filling holes in Image');

% Thinning the image to ensure character isolation.
H=bwmorph(F,'thin',1);
H=imerode(H,strel('line',3,90));
subplot(232);imagesc(H);
title('Thinning image');

% Selecting all the regions that are of pixel area more than 100.
final=bwareaopen(H,100);
subplot(233);imagesc(final);
title('Final Image after eliminating smaller regions');

% Bounding boxes are acquired.
Iprops=regionprops(final,'BoundingBox','Image');

% Selecting all the bounding boxes
NR=cat(1,Iprops.BoundingBox);
figure(3)
colormap('gray')
% Calling of controlling function.
r=controlling(NR);
if ~isempty(r) % If succesfully indices of desired boxes are achieved.
    I={Iprops.Image}; % Cell array of 'Image'
    noPlate=[];
    for v=1:length(r)
        N=I{1,r(v)}; % Extracting the binary image corresponding to the
indices in 'r'.
        subplot(2,5,v);imagesc(N);
        title(v);
        letter=readLetter(N); % Reading the letter corresponding the binary
image 'N'.
        while letter=='O' || letter=='0'
            if v<=3
                letter='O';
            else
                letter='0';
            end
            break;
        end
noPlate=[noPlate letter]; % Appending every subsequent character in noPlate.
        if v==1
            CityCode=letter;
        end
        if v==2
            CityCode=[CityCode letter]
        end

        if v==5
            VehicleClass=letter
            %vehicle class codes for delhi
            if CityCode == 'DL'
                if VehicleClass == 'S'
                    VehicleClass='Two Wheeler';
                elseif VehicleClass == 'C'
```

```
VehicleClass='Car';
                elseif VehicleClass == 'E'
                    VehicleClass='Electric';
                elseif VehicleClass == 'P'
                    VehicleClass='Passenger';
                elseif VehicleClass == 'R'
                    VehicleClass='Rickshaw';
                elseif VehicleClass == 'V'
                    VehicleClass='Van';
                elseif VehicleClass == 'T'
                    VehicleClass='Taxi';
                end
            elseif CityCode == 'RJ'
                if VehicleClass == 'S'
                    VehicleClass='Two Wheeler'
                elseif VehicleClass == 'C'
                    VehicleClass='Car'
                end
            elseif CityCode == 'HH'
                if VehicleClass == 'S'
                    VehicleClass='Two Wheeler'
                elseif VehicleClass == 'D'
                    VehicleClass='Car'
                end
            end
        end
    end

    clc
    fprintf('Number plate  :  %s\n',noPlate)



    %inserting VehicleRecords
    InOut='In'
    a = clock;
    if InOut=='In'
        %insert into table- checkIn details
        conn = sqlite(dbfile);
        VehicleData = {noPlate,VehicleClass,a(1),a(2),a(3),a(4),a(5)} ;

insert(conn,'VehicleRecords',{'VehicleNumber','VehicleClass','CheckInYear','C
heckInMonth','CheckInDay','CheckInHour','CheckInMinute'},VehicleData)
        close(conn)

    elseif InOut=='Out'
        %insert into table- CheckOut details
        conn = sqlite(dbfile);
        %assumption same day checkIn and CheckOut
        if VehicleClass == 'Two Wheeler'
            %amt = (((checkouthour-checkinhour)*60+(checkoutMin-
checkInMin))/60)*5+10
        elseif VehicleClass == 'Car'
        %amt = (((checkouthour-checkinhour)*60+(checkoutMin-
checkInMin))/60)*5+10
        VehicleData = {noPlate,VehicleClass,a(1),a(2),a(3),a(4),a(5),0} ;

insert(conn,'VehicleRecords',{'VehicleNumber','VehicleClass','CheckOutYear','
CheckOutMonth','CheckOutDay','CheckOutHour','CheckOutMinute','Amount'},Vehicl
eData)
```

```
        close(conn)
    end

else
    % If fail to extract the indexes in 'r' this line of error will be
displayed.
    numberPlateExtraction2()
    fprintf('Unable to extract the characters from the number plate.\n');
    fprintf('The characters on the number plate might not be clear or
touching with each other or boundries.\n');
    noPlate = '0';
    class = '0';
end
end
```

Template Creation

```
%CREATE TEMPLATES
%Letter
A=imread('A.bmp');B=imread('B.bmp');
C=imread('C.bmp');D=imread('D.bmp');
E=imread('E.bmp');F=imread('F.bmp');
G=imread('G.bmp');H=imread('H.bmp');
I=imread('I.bmp');J=imread('J.bmp');
K=imread('K.bmp');L=imread('L.bmp');
M=imread('M.bmp');N=imread('N.bmp');
O=imread('O.bmp');P=imread('P.bmp');
Q=imread('Q.bmp');R=imread('R.bmp');
S=imread('S.bmp');T=imread('T.bmp');
U=imread('U.bmp');V=imread('V.bmp');
W=imread('W.bmp');X=imread('X.bmp');
Y=imread('Y.bmp');Z=imread('Z.bmp');
Afill=imread('fillA.bmp');
Bfill=imread('fillB.bmp');
Dfill=imread('fillD.bmp');
Ofill=imread('fillO.bmp');
Pfill=imread('fillP.bmp');
Qfill=imread('fillQ.bmp');
Rfill=imread('fillR.bmp');

%Number
one=imread('1.bmp');  two=imread('2.bmp');
three=imread('3.bmp');four=imread('4.bmp');
five=imread('5.bmp'); six=imread('6.bmp');
seven=imread('7.bmp');eight=imread('8.bmp');
nine=imread('9.bmp'); zero=imread('0.bmp');
zerofill=imread('fill0.bmp');
fourfill=imread('fill4.bmp');
sixfill=imread('fill6.bmp');
sixfill2=imread('fill6_2.bmp');
eightfill=imread('fill8.bmp');
ninefill=imread('fill9.bmp');
ninefill2=imread('fill9_2.bmp');

letter=[A Afill B Bfill C D Dfill E F G H I J K L M...
    N O Ofill P Pfill Q Qfill R Rfill S T U V W X Y Z];
number=[one two three four fourfill five...
```

```
    six sixfill sixfill2 seven eight eightfill nine ninefill ninefill2 zero
zerofill];
character=[letter number];
NewTemplates=mat2cell(character,42,[24 24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24]);
save ('NewTemplates','NewTemplates')
clear all
```

Letter Reading

```
function letter=readLetter(snap)

load NewTemplates % Loads the templates of characters in the memory.
snap=imresize(snap,[42 24]); % Resize the input image so it can be compared
with the template's images.
comp=[ ];
for n=1:length(NewTemplates)
    sem=corr2(NewTemplates{1,n},snap); % Correlation the input image with
every image in the template for best matching.
    comp=[comp sem]; % Record the value of correlation for each template's
character.
end
vd=find(comp==max(comp)); % Find the index which correspond to the highest
matched character.
if vd==1 || vd==2
    letter='A';
elseif vd==3 || vd==4
    letter='B';
elseif vd==5
    letter='C';
elseif vd==6 || vd==7
    letter='D';
elseif vd==8
    letter='E';
elseif vd==9
    letter='F';
elseif vd==10
    letter='G';
elseif vd==11
    letter='H';
elseif vd==12
    letter='I';
elseif vd==13
    letter='J';
elseif vd==14
    letter='K';
elseif vd==15
    letter='L';
elseif vd==16
    letter='M';
elseif vd==17
    letter='N';
elseif vd==18 || vd==19
```

```
    letter='O';
elseif vd==20 || vd==21
    letter='P';
elseif vd==22 || vd==23
    letter='Q';
elseif vd==24 || vd==25
    letter='R';
elseif vd==26
    letter='S';
elseif vd==27
    letter='T';
elseif vd==28
    letter='U';
elseif vd==29
    letter='V';
elseif vd==30
    letter='W';
elseif vd==31
    letter='X';
elseif vd==32
    letter='Y';
elseif vd==33
    letter='Z';
    %*-*-*-*-*
% Numerals listings.
elseif vd==34
    letter='1';
elseif vd==35
    letter='2';
elseif vd==36
    letter='3';
elseif vd==37 || vd==38
    letter='4';
elseif vd==39
    letter='5';
elseif vd==40 || vd==41 || vd==42
    letter='6';
elseif vd==43
    letter='7';
elseif vd==44 || vd==45
    letter='8';
elseif vd==46 || vd==47 || vd==48
    letter='9';
else
    letter='0';
end
end
```

Determining the values of indices

```
function r=takeboxes(NR,container,chk)
takethisbox=[]; % Initialize the variable to an empty matrix.
for i=1:size(NR,1)
    if NR(i,(2*chk))>=container(1) && NR(i,(2*chk))<=container(2) % If
Bounding box is among the container plus tolerence.
        takethisbox=cat(1,takethisbox,NR(i,:)); % Take that box and
concatenate along first dimension.
    end
end
```

```
r=[];

for k=1:size(takethisbox,1)
    var=find(takethisbox(k,1)==reshape(NR(:,1),1,[])); % Finding the indices
of the interested boxes among NR
    if length(var)==1                                  % since x-coordinate
of the boxes will be unique.
        r=[r var];
    else                                               % In case if x-
coordinate is not unique
        for v=1:length(var)                            % then check which box
fall under container condition.
            M(v)=NR(var(v),(2*chk))>=container(1) &&
NR(var(v),(2*chk))<=container(2);
        end
        var=var(M);
        r=[r var];
    end
end
end
```

Controlling

```
function r=controlling(NR)
[Q,W]=hist(NR(:,4)); % Histogram of the y-dimension widths of all boxes.
ind=find(Q==6); % Find indices from Q corresponding to frequency '6'.

for k=1:length(NR)            % Taking the advantage of uniqueness of y-co
    C_5(k)=NR(k,2) * NR(k,4); % ordinate and y-width.
end
NR2=cat(2,NR,C_5');           % Appending new coloumn in NR.
[E,R]=hist(NR2(:,5),20);
Y=find(E==10);                 % Searching for six characters.
if length(ind)==1 % If six boxes of interest are succesfully found record
    MP=W(ind);    %  the midpoint of corresponding bin.
    binsize=W(2)-W(1); % Calculate the container size.
    container=[MP-(binsize/2) MP+(binsize/2)]; % Calculating the complete
container size.
    r=takeboxes(NR,container,2);
elseif length(Y)==1
    MP=R(Y);
    binsize=R(2)-R(1);
    container=[MP-(binsize/2) MP+(binsize/2)]; % Calculating the complete
container size.
    r=takeboxes(NR2,container,2.5); % Call to function takeboxes.
elseif isempty(ind) || length(ind)>1 % If there is no vlaue of '6' in the Q
vector.
    [A,B]=hist(NR(:,2),20); % Use y-coordinate approach only.
    ind2=find(A==10);
    if length(ind2)==1
        MP=B(ind2);
        binsize=B(2)-B(1);
        container=[MP-(binsize/2) MP+(binsize/2)]; % Calculating the complete
container size.
        r=takeboxes(NR,container,1);
    else
        container=guessthesix(A,B,(B(2)-B(1))); % Call of function
guessthesix.
        if ~isempty(container) % If guessthesix works succesfully.
            r=takeboxes(NR,container,1); % Call the function takeboxes.
```

```
    elseif isempty(container)
        container2=guessthesix(E,R,(R(2)-R(1)));
        if ~isempty(container2)
            r=takeboxes(NR2,container2,2.5);
        else
            r=[]; % Otherwise assign an empty matrix to 'r'.
        end
    end
end
```