

**UNRAVEL PARKING DIFFICULTY  
VIA  
RELATIVE CHAIN CODE ALGORITHM**

**By**

**CHAN HUEY MIN  
WEK020027**

A Thesis presented to the Faculty of Computer Science and  
Information Technology of University of Malaya in partial  
fulfillment of the Requirement for the Degree of

**BACHELOR OF COMPUTER SCIENCE**

**UNIVERSITY MALAYA**

**2004/2005**

## Abstract

Searching for a parking space is a familiar challenge to city-dwellers everyday. With the increasing numbers of vehicles, not only the roads are congested, it is also difficult to find parking space everywhere especially in shopping complexes. *Unravel Parking Difficulty via Relative Chain Code Algorithm* is a parking system which will help vehicles' owner to maneuver their means of transportation without the need of scrupulously locating a vacant parking space in the parking bay. This system will allow drivers to quickly locate an available parking space without hassle, thus they can save time from searching for vacant space. This project focuses on the image processing stage, where it will go through from the process of detecting empty space, character recognition, updating the database to notifying drivers about available car park.



## Acknowledgement

First and foremost, I would like to express my utmost gratitude to Mr. Mohd Yamani Idna bin Idris for willing to be my supervisor of the “Unravel Parking Difficulty via Relative Chain Code Algorithm” Project. Thank you for his supervision of my project progress from time to time. Despite my status as a student, I was given all possible help in designing the project and preparing the report. I would like to express my appreciation towards the guidance I received.

A special thanks to my moderator, Ms. Rafidah Md Noor for her advice and suggestion which is important for the success of the project.

Furthermore, I would like to thank my project partner, Ms. Chan Yuit Pui for her suggestions, help and willingness to share her knowledge for this project. Thank you for her support and without her I would not be able to complete my project and report on time.

Last but not least, I would also like to express my appreciation to the Faculty of Computer Science and Information Technology for facilitating the whole “Projek Ilmiah” program and given me the facility I needed to complete the project.

# Table of Contents

<b>Chapter 1: Introduction</b>	1
1.1 Overview	1
1.2 Project Motivation	3
1.3 Project Objective	4
1.4 Project Scope	5
1.4.1 Scope Allocation	7
1.5 Project Schedule	8
<b>Chapter 2: Literature Review</b>	9
2.1 System Background	9
2.1.1 Problems to Recover	11
2.2 Analysis Studies	13
2.2.1 Case Study 1- Optical Car Park Space Recognition	13
2.2.1.1 Analysis Result	13
2.2.2 Case Study 2 – Iris-Number Plate Recognition System	15
2.2.2.1 Analysis Result	15
2.3 Image Processing and Analysis	17
2.3.1 Digital Image Processing	17
2.3.2 Representation of Digital Images	19
2.3.3 Digital Image File Format	20
2.3.3.1 File Format	20
2.4 Image Digitization	22
2.5 Image Pre-Processing	22
2.6 Segmentation	24
2.6.1 Segmentation Methods	25
2.6.1.1 Thresholding	25
2.6.1.2 Edge-Based Segmentation	27
2.6.1.3 Region-Based Segmentation	32
2.6.1.4 Morphological Filtering	33
2.6.1.5 Matching	36
2.7 Recognition	36
2.8 Run Length Encoding	38
2.9 Chosen Algorithm	39
2.9.1 Noise Reduction	39
2.9.2 Gray-Scale Stretch	39
2.9.3 Edge-Smoothing	39
2.9.4 Threshold	40
2.9.5 Thinning	40
2.9.6 Edge Detection	40
2.9.7 Recognition	41
2.9.8 Run Length Encoding	41



<b>Chapter 3: Methodology and Techniques</b>	42
3.1 Methodology	42
3.2 Waterfall Model with Prototyping	43
3.3 Techniques Used to Gather Information	48
<b>Chapter 4: System Analysis</b>	50
4.1 Introduction	50
4.2 Functional Requirement	51
4.3 Language	52
4.3.1 MATLAB (MATrix LABoratory)	52
4.3.1.1 What is MATLAB?	52
4.3.1.2 The MATLAB System	53
4.3.1.3 MATLAB GUIDE	55
4.3.2 MATHEMATICA	57
4.3.2.1 Introduction to MATHEMATICA	57
4.3.3 Java	59
4.4 Chosen Language	60
4.5 Non-Functional Requirement	60
4.6 Hardware Requirement	64
4.7 Software Requirement	65
<b>Chapter 5: System Design</b>	66
5.1 Introduction	66
5.2 Conceptual Design	66
5.3 System Design	66
5.3.1 Input Digitization	68
5.3.2 Adding and Removing Noise	68
5.3.3 Image Pre-Processing Processes	70
5.3.3.1 Intensity / Brightness Adjustment	71
5.3.4 Segmentation	71
5.3.4.1 Threshold	72
5.3.4.2 Thinning	73
5.3.4.3 Edge Detection	74
5.3.5 Recognition	75
5.3.6 Knowledge Base	76
5.3.7 Run Length Encoding	77
5.4 Data Flow Diagram	77
5.5 User Interface Design	80
<b>Chapter 6: System Implementation</b>	86
6.1 Development Environment	86
6.1.1 Hardware in the Development Environment	87
6.1.2 Software in the Development Environment	87
6.2 Recognition Algorithm	88
6.3 Image Processing	89
6.4 Segmentation	90
6.5 Recognition	93



<b>Chapter 7: System Testing</b>	108
7.1 Introduction	108
7.1.1 Compilation and Execution	109
7.1.2 Debugging	109
7.1.3 Accuracy of Execution	110
7.2 Image Processing	111
7.2.1 Image Digitization	111
7.2.2 Threshold	112
7.2.3 Thinning	113
7.3 Segmentation	114
7.4 Recognition	116
<b>Chapter 8: Conclusion &amp; Discussion</b>	125
8.1 Conclusion	125
8.2 Problems Encountered	126
8.2.1 Inexperience in MATLAB	126
8.2.2 Understanding the Concept of Image Processing	126
8.2.3 Tight Schedule	127
8.3 System's Strengths	127
8.4 System's Constraints	129
8.5 Future Enhancements	130
<b>APPENDIX A: SOURCE CODE</b>	131
<b>APPENDIX B: Example of Generated Chain Codes</b>	142
<b>BIBLIOGRAPHY AND REFERENCES</b>	148
<b>USER MANUAL</b>	149

## List of Figures

Figure 2.1: 8-direction chain code	37
Figure 2.2: Comparison of Absolute and Relative Chain Code	37
Figure 3.1: Generic System Development Model	42
Figure 3.2: The Waterfall Model with Prototyping	47
Figure 4.1: MATLAB GUI Layout Editor	56
Figure 4.2: Java Image Processing	59
Figure 5.1: Process Flow of the Implementation Phases	67
Figure 5.2: Original Image	68
Figure 5.3: Gray-Scale Image	68
Figure 5.4: Horizontal/Vertical Line Preserving Neighborhood for Median Filtering	69
Figure 5.5: Numeric Character after adding Salt-and-Pepper Noise	70
Figure 5.6: Numeric Character after Removing Noise using Median Filter	70
Figure 5.7: Original Gray Scale Image	71
Figure 5.8: Image after Intensity Adjustment	71
Figure 5.9: Original Gray Scale Image	72
Figure 5.10: Image after Threshold	72
Figure 5.11: Image after Thinning	73
Figure 5.12: Image after Sobel Operation	74
Figure 5.13: 8-direction Chain Code Diagram	75
Figure 5.14: Data Flow Diagram of the Unravel Parking Difficulty via Relative Chain Code Algorithm System	79
Figure 5.15: Main page of the Unravel Parking Difficulty via Relative Chain Code Algorithm System	82
Figure 5.16: Analysis, Enhancement and Segmentation Process Page	83
Figure 5.17: Numeric Character Recognition of Unravel Parking Difficulty via Relative Chain Code Algorithm Page	84
Figure 5.18: Car Parking Vacant Map Page	85
Figure 7.1: Image of Number One	114
Figure 7.2: Image after Segmentation	115
Figure 7.3: Image that will be used for Recognition Process	116
Figure 7.4: Image of numerical character (number one) that is use to generate Chain Code	118
Figure 7.5: Image of numerical character (number five) that is use to generate Chain Code	119
Figure 7.6: Image of path for the robot navigation to search the position of the vacant parking space that is use to generate chain code	120
Figure 7.7: Possible pixel's value for the 8-connected neighborhood pixel	121
Figure 7.8: Pixels which in black boxes and directions of chain code that is impossible to be recognized.	124



## List of Tables

Table 1-1: Project Schedule	8
Table 5-1: Data Flow Diagram (DFD) Symbols	78
Table 7.1: Possible subsequent point of the 8-connected neighborhood pixel based on the initial point of (1, 6) and also the representation for the directions of the chain code.	122
Table 7-2: Possible subsequent point base on the previous code that has been read.	123



# Chapter 1: Introduction

## 1.1 Overview

Designing a system for automatic image content recognition is a non-trivial task that has been studied for a variety of applications. Computer recognition of specific objects in digital image has been put to use in various fields such as manufacturing industries, intelligence and surveillance, and image database cataloging.

In this project, a character recognition system which uses relative chain code algorithm to detect empty spaces in a car park will help to guide a vehicle owner to maneuver their vehicle to empty space, thus saving time and energy to locate empty parking lot manually. It can serve as an introduction for future work or research in improving the system to make it more sophisticated. This character recognition system will be implemented in a car park with integration of shortest path detection system and robot navigation system.

An empty parking space must be detected before a vehicle is able to park in it. One of the approach is to implement sensors such as laser range sensor in each parking space, where the sensors will detect any object that filled the area. Secondly is to use pressure sensor, where the sensors will detect the pressure given by the front and rear wheels to determine whether there is a vehicle on that specific car park space. The third approach is by using image processing technique similar to image recognition technique that has been implemented in various applications such as manufacturing, medical community and law enforcement. In this project, the third approach which is the image processing technique will be used since the vacant parking space can be detected in the range of the image captured and the camera can be used for surveillance to ensure a

secure environment in the car park. Furthermore, the project can be expanded into a more intelligent system in the time to come.

Each parking space will have their individual identity which is a unique numeric character on the floor of that space. The purpose of this numeric character is to allow the image of the character being captured rather than capturing the vehicle's image. The advantage of this method is that it does not require the time consuming process of training a neural network or computing distance measures between every possible region in the image. The additional purpose of utilizing the floor instead of the object (vehicle) is to discard the possibility of misdetection of vacant parking space since an error might occur if the object is not detected (the car may be too small and the vehicle's owner park the car too deep inside until the camera cannot detect it). If the floor is not located and detected, it will not jeopardize other vehicle from coming in to the occupied parking space.

In addition, for recognizing the path of the parking space, the system will use the approach of character recognition which will focus on the relative chain code algorithm to identify which parking space is empty after the empty space has been detected. After that, the information will go through another shortest path detection system to detect the shortest way for the vehicle to reach the destination. The numeric character on the floor of the parking space will need to go through basic digital image processing process, followed by segmentation to ensure the reliability of the end result. The image processing process will cover noise reduction, quality enhancement and segmentation where the image will be break up into its' own separate individual character.



## 1.2 Project Motivation

As the world economy become more flourishing and the standard of living keep on improving, vehicles have become an important means of transportation for city-dwellers to travel around. The consequences of more vehicles not only impact congestion on the road, but also a blow to the accessibility of parking spaces in one building especially in shopping complexes. The limited parking spaces have caused a number of difficulties not only to the management (to provide more parking spaces) also to the consumers (to search for an available parking space which would lead to a frustrating atmosphere). Because of the limited resources, most of the management did not undergo major revamp to provide more parking spaces since the average time for consumer to shop is about two to three hours. However, the management still trying to find suitable approach to attract their customer by providing them with the best services they could offer.

While on the customer side, the problem that usually faced by customers is that they are having difficulties to find vacant parking space in a over crowded shopping complex, but since they have paid the parking fees, they will just turn round and round the parking area just to find a vacant space. This will eventually result in more and more vehicles in the car park area because more and more car is coming in, to be congested with vehicles. This not only give a bad impression to customers that the particular car park operator can not provide enough parking spaces for its users and it is a wasting of time and money to enter into that car park.

An intelligent car parking system should be created to attend to this problem. This project is hoped to be one of the ways and solutions for future plan. For the above reason, it had motivates the writer and the writer's final year project partner, Ms Chan



Yuit Pui to study and research on a parking system which will guide a vehicle owner to maneuver their means of transportation without the needs of rigorously locating vacant parking space. While the system will be implementing the concept of image processing, shortest path recognition, robotics and control, this project will only be focusing on the image processing section.

### 1.3 Project Objective

The surface objectives of this project are as below:

- To help car park users to find an empty space in the car park more rapidly, thus saving time and money.
- To explore and study the image processing technology.
- To recognize the numeric character on the floor of the parking space.
- To transform the captured image from analog format to digital format.
- To improved the quality and well-defined the numeric character using enhancement technique.
- To perform segmentation process and comparison of the image with the existing data in the knowledge base in order to recognize the character.
- To build a knowledge base that is able to store all the chain codes and the digitized image.
- To recognize the captured image of the characters.
- To build an output map to show the result of the character recognition.

- To help the shortest path detection system to transform the captured image to relative chain code, in order for the system to find the shortest way to the vacant space.
- To test the Unravel Parking Difficulty via Relative Chain Code Algorithm.

#### 1.4 Project Scope

The Unravel Parking Difficulty via Relative Chain Code Algorithm is a software product mainly targeted to the car park's management or operator which will implement the system into their computer's in the security section, not only to monitor the security in the car park but also able to inform car park users where is the vacant parking space in the car park area. Its aim is to help car park users to find an empty parking space as soon as possible once they entered into the car park area.

Through using the Unravel Parking Difficulty via Relative Chain Code Algorithm, the management of the car park will be able to provide the best services to their customer. It will help customer to know whether there is any vacant parking space in the car park before they paid the parking fees and entered into the car park. As this is an improvement in the field of science and technology, it should attract more customers to the car park which implement this system. With the time of searching vacant parking space reduced, customers will tend to spend more time in shopping complexes to buy more things which will eventually bring in more profits to the management.



The main features provide by the system includes:

### 1. Image Digitization

Users are able to do the transformation. Images which are being captured by the camera will be digitized. The digitized image will be kept in JPEG or GIF file format.

### 2. Image Pre-Processing and Analyze

Users will be able to analyze and enhance the image before they can perform the segmentation and recognition process. Some of the processes for the image analysis and pre-processing algorithm are:

- Gray-scale
- Noise Reduction
- Brightness / Intensity Adjustment
- Edge Smoothing

### 3. Segmentation Process

Users are able to apply segmentation to the digitized image to make it easier for the character recognition process. Segmentation processes includes:

- Threshold
- Thinning
- Edge Detection



#### 4. Knowledge Base

It is a dictionary for recognition process to store all the relative chain code for the numeric characters of 0 to 9.

#### 5. Character Recognition using Relative Chain Code Algorithm

Users are able to recognize each character on the floor to know which parking space is empty. After that, it will translate the shortest path direction (which is done by the shortest path detection system) into chain code in the form of a map.

#### 6. Run Length Encoding (RLE)

Users are able to replace a long sequence of same symbol, in this case, the repetition of numbers, into a shorter sequence.

#### 7. Result / Output Map of the Empty Car Park Space

Users are able to identify which car park space is full and which one is vacant.

### 1.4.1 Scope Allocation

As this is a final year project for two students, thus the writer and the writer's fellow partner will divide the scope of project into two major parts. The writer will be focusing on the segmentation processes such as thinning, edge detection and the relative chain code algorithm. While Ms Chan Yuit Pui (WEK020028) will be focusing on the image pre-processing processes which includes gray-scale, noise reduction, brightness / intensity adjustment, edge-smoothing and also the run-length encoding (RLE).

Table 1-1: Project Schedule

Project Scope	Jun		July				August				September				October				November				December				January				February			
	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Preliminary Study and Planning																																		
Literature Review																																		
System Analysis																																		
System Design																																		
Prototype																																		
Development and Coding																																		
Unit Testing																																		
System Testing																																		
Documentation																																		
Implementation																																		



## Chapter 2: Literature Review

### 2.1 System Background

As the world economy become more flourishing and the standard of living in the city keep on improving, vehicles have become an important means of transportation for city-dwellers to travel around within the vicinity. The consequences of more vehicles not only result to the congestion on the road, but it is also a blow to the accessibility of parking spaces in one building especially in shopping complexes. The limited parking spaces have caused a number of difficulties not only to the car park operator (to provide more parking spaces) but also to the consumers (to search for an available parking space which would lead to a very annoying atmosphere). Because of the limited resources and funding, most of the car park operators did not undergo major revamp to provide more parking spaces since the average time for consumer to shop is about two to three hours. However, they are still trying to find a suitable approach to attract their customer to use their car park by providing them with the best service they could offer to them.

While on the customer side, the problem that usually faced by customers is that they are having difficulties to find vacant parking space, but since they have paid the parking fees, they will just go round and round the parking area just to find a vacant space. This will eventually resulted in more and more vehicles within the car park area to be congested with vehicles. This not only give a bad impression to customers that the particular car park operator can not provide enough parking spaces for its users and it is a wasting of time and money to enter into that particular car park.

In order to respond to the current problem, an intelligent parking system using character recognition approach is needed to accommodate the ever growing of vehicles and help drivers to maneuver around tight and restricted spaces in the car park. The system will display to the users the parking target position by adjusting and arranging vacant spaces on a screen (output map). Car park users will only need to maneuver their vehicle according to the numeric characters of the vacant space on the output screen and the output map will also present the shortest path to arrive to that particular vacant space via the Shortest Path Detection System.

2.2.2 In order to capture the images of the car park floors, there are only a few devices that are necessary, which is webcam, cables to connect from the webcam to the PC and computer. The camera will capture the floor of the car park which contains numeric characters on each individual car park space. The captured images will then be passed to the computer for transformation from analog to digital, image analysis, enhancement and recognition processes. Electronic transmission of image data may introduce noise and other undesired or unwanted element into the captured image. Furthermore, the lightness in the car park also decreased the quality of the image. Thus, there are a variety of image processing methods that offer different algorithms that cater and help to improve the quality of the captured image. Image analysis and enhancement processes attempts to correct or compensate for systematic errors and enhances image features that are important for further processing. Segmentation process which includes thresholding, thinning and edge detection aims to differentiate the foreground and background object where the image will be break into individual characters for better recognition. While for the recognition of the captured image, relative chain code algorithm will be implemented because of its advantages of recognizing character at different orientation.



Lastly, the recognized numeric character which will be the vacant space in the car park will be displayed on the screen. The output map will then be passed to the shortest path detection system to identify the shortest course for a vehicle to reach the destination parking space. Result of the shortest path recognition system has to go through the image processing processes to transform the path into chain codes. However, the scope and focus of this project is only on image processing and character recognition.

### **2.1.1 Problems to Recover**

The system requires webcams to be installed within the areas of the car park to capture the required images for recognition purposes. Noise may come from several sources during the process capturing the images. More often, camera captured images are in analog form and noise may appear in an image when the image is going through the electronic transmission. There is possibility that noise may appeared with the captured image according to the quality and types of the webcam. The problem of noise will be discussed during the explanation of the noise reduction algorithm.

Problems of insufficient lighting frequently occur in the car park. The lightsome of the car park will affect the quality of the captured image. The captured image may be presented in a blur and dark image which is complicated for the image processing processes. Intensity or brightness adjustment is needed to improve and enhance the image. The intensity and brightness adjustment algorithms will be discussed in the following sections.

Edge smoothing is another image enhancement technique that is used to give images a softer effect. It helps to improve the quality of the image thus make it easier

for the threshold process. Several smoothing algorithms are in consideration to apply smoothing to an image.

After the quality of the images has been improved, it is ready for distinguishing the numeric character that is needed from its background from a particular image. Thresholding will be use for this process, as it is able to separate out the regions of the image corresponding to the numeric character in which it is needed, from the regions of the image that correspond to the background. It is able to differentiate intensities or colors in the foreground and background regions of an image. The techniques for thresholding will be discussed in the later sections.

Although thresholding is able to differentiate the extract the numeric character from its background, but the result may be blurred and even unrecognizable. Therefore, there is a need to enhance the result by making the numeric character sharper and more recognizable. For this purpose, thinning which is a morphological operation has been chosen because of its ability to repeatedly remove boundary elements until a pixel set with maximum thickness of one or two is found. Thus makes the object (numeric characters) that is needed more prominent.

While thresholding produces a segmentation that yields all the pixels that belong to the object of interest in an image but it may not segment out the edges that form the structure of the object of interest. Therefore there is a need for edge detection to identify meaningful image features by finding for pixels that belong to the borders of the objects.

When the numeric characters are being transformed to a string of relative chain codes for recognition purposes, the sequence might become very long and it is time consuming to process the data. Thus there is a need to compress the string of chain codes to become shorter not only to speed up the processing time to match and recognize



the numeric characters but also to reduce the storage space for the relative chain codes. The recognition of the numeric characters using relative chain codes and the compression of the string of chain codes will be discussed in the subsequent sections.

## 2.2 Analysis Studies

### 2.2.1 Case Study 1- Optical Car Park Space Recognition

The optical car park space recognition uses video camera and with the help of mirrors that are placed at a 45 degrees angle above each space. The mirror is used to reflect the car park space at a camera somewhere nearby mounted on the ceiling. The algorithm is to take a picture of the car park subset visible from a camera when there are no cars. If any of the sum-of-squares is quite small. The space is empty; otherwise it is probable full. The camera are being use to capture images in the car park with the help of two different bordered mirrors on each car park space pointing to different cameras then splice off the video feed to pass into a computer to do all the calculations and data processing.

#### 2.2.1.1 Analysis Result

##### The Strengths

1. The use of two different bordered mirrors pointing to different cameras will help to do two different calculations, this is to make sure the car park space is really empty or not.
2. It helps the customers well informed of the vacant space in the car park, and it makes it easy for people to park.

3. The car park space is fully utilized, so the car park is sized for maximum capacity.
4. No need the turn round and round the car park to find for empty space which is a waste of time and money.

### **The Weaknesses**

1. Vandalism may occur in the car park, where vandals might damage the mirrors. This will cause the calculations to be not accurate and might pose dangers to people.

### **Conclusion**

The system is something new, economical and user friendly as it uses things that is available in the car park such as camera to capture the image, computer to do all the data processing and also mirrors (which is quite unusual) to help to capture image without installing too many camera in the car park. The system only need to calculate the sum-of-squares of the real time image and compare it with the image that may or may not have a car in it. Before the system can be implemented, the operator will need to run the system for a while to build up a statistical distribution of stay durations. This is to determine the sum-of-square need to be how large before the space can be declared as occupied. This information is important to be fed back into the algorithm, so that the system will know when the space can be declared as empty and when it is full.



### 2.2.2 Case Study 2 – Iris-Number Plate Recognition System

Automatic Number Plate Reader (ANPR) consists of cameras linked to a computer. As a vehicle passes, the ANPR equipment reads the number plate. ANPR is to capture the number plate of vehicles using image processing and recognize it from the database. The camera use direct show interface from the webcam. The system providing imaging solutions, from simple image capture and retrieval systems through to complex image processing solutions. Firstly, it will identify the image and locate in image file. The character will be extract from the number plate. The system will identify the character and each individual character will be displayed as separate image the screen. Recognition accuracy depends on several factors. Under the circumstance of an average parking or access control system/application, the recognition accuracy is steadily over 95%. 97-98% is frequently reached in 24 hours/day, 7 days/week operation.

#### 2.2.2.1 Analysis Result

##### The Strengths

1. ANPR has been uses for many different applications, such as access control car park, road tolling and security data for customer's side.
2. It is cost-effective as it only requires software-only system written in C++, standard PC and multi-camera input per PC.
3. Iris has the flexibility and speed to deal with virtually any requirement such as specialist infra-red (IR) cameras, high resolution digital camera and standard CCTV equipment in is standard configuration.

## **The Weaknesses**

1. The performance achieved will primarily depend on other system constraints, (e.g. is the illumination system biased to capturing images of the vehicle or for number plate recognition), as well as by the number plates passing the surveillance point. Some plates are very poor for recognition purposes (e.g. from Belgium).

## **Conclusion**

This system offers a few features that are appealing to users such as access control, security and surveillance, and traffic management. It uses the latest technology of modern world to recognize the car plate's number automatically without reading the license plate manually. This not only save time as the accuracy of the system is quite high but also slashing the overhead to maintain the system as it is done automatically.



## **2.3 Image Processing and Analysis**

Image processing and analysis can be defined as an act of examining image for the purpose of identifying a specific object and its' significance. Image analyst studies the remotely sensed data and attempt through logical process in detecting, identifying, measuring and evaluating the significance of physical and cultural objects, patterns and spatial relationship.

### **2.3.1 Digital Image Processing**

Image is a form of picture while digital image is an image which is digitized to convert it to a form that can be stored in a computer's memory or on some form of storage media such as a hard disk or CD-ROM. This digitization procedure can be done by a scanner, or by a video camera connected to a frame grabber board in a computer. Once the image has been digitized, it can be operated upon by various image processing operations.

Three major fields in digital image processing are image restoration, image enhancement and image compression. Image restoration is a process of taking an image with known or estimated degradation and in an attempt to refurbish the image back to its original appearance. Image enhancement corrects image defects which could be caused by the digitization process or by faults in the imaging set-up. Image compression involves reducing large amount of data that is needed to represent an image and also to reduce the amount of memory needed to store a digital image. Digital image processing can be divided into 3 types of computerized processes:

- Low-level process:
  - Primitive operations like noise reduction, edge extraction, contrast enhancement, image sharpening and etc.
  - Uses data which resemble the input image; input image which is captured by a camera is 2D in nature, being described by an image function whose value is usually brightness depending on two parameters, the coordinates of the location in the image.
  - Image which is digitized may be represented by a rectangular matrix with elements corresponding to the brightness at appropriate image locations.
  - Such matrices are inputs and outputs of low level image processing.
  - Input: image
  - Output: image
- Mid-level process:
  - Tasks like segmentation, representation, description.
  - Input: image
  - Output: attributes extracted from images
- High-level process:
  - “Making sense” of an ensemble of recognized objects.
  - Input: image (sequence)
  - Output: interpretation



### 2.3.2 Representation of Digital Images

Normally, digital image can be represented by  $I(r,c)$  where  $r$  and  $c$  is the position and pixel that shows the brightness for black and white images. However, representation of  $I(r,c)$  represented other function. There are a few types of digital image as shown below:

- Binary

The most easiest representation as it only have two values which is white or black, '0' or '1'. Often used in computer vision application.

- Gray-scale

A special representation where there is only brightness information available.

The number of bits will determine the level of brightness of an image. For example, 8 bits/pixel will have 256 (0-255) different brightness levels.

- Color

Color can be represented by three band of monochrome of image data where each band of data corresponds to different color. It is represent by RGB which is red, green, and blue. Every band has 8 bits per color so one pixel will have 24 bits per pixel.

- Multispectral

Multispectral contains information outside the normal human perceptual range. Some of the examples are infrared, ultrasound, x-ray and acoustic. Sources of multispectral images are from satellite system, underwater sonar system, and various types of airborne radar and infrared imaging system.

### 2.3.3 Digital Image File Format

It can divide into two categories:

- Bitmap image – represented by image model  $I(r,c)$
- Vector image – representing lines, curves, and shape by storing only key points. It will join key points and render to obtain image.

#### 2.3.3.1 File Format

There are a few types of file formats that are used to encode digital images:

- GIF (Graphic Interchange Format)
  - 8 bits/pixel (256 colors)
  - Allows only one type of compression
  - Contains only basic information
  - Ideal for - screen captures, line drawings, sharp-edged graphics and images with transparency
  - Not good for - photographic images and artwork with complex colors



- JPEG (Joint Photographic Experts Group)

- Compress image with JPEG algorithm
- User can define the amount of compression
- Requires less storage than GIF
- Ideal for - photographic images, images with rich color transitions
- Not good for - images with sharp edges, text, transparency

- TIFF (Tagged Image File Format)

- More sophisticated than GIF
- 24 bits/pixel
- Most comprehensive format
- Implemented on a variety of computer systems, usually used for archival scans

- PNG (Portable Network Graphics)

- Open format supported by most image creation programs.
- Supports millions of colors
- Ideal for – color complex-images and images with transparency

## 2.4 Image Digitization

“Image” is a representation of an object that is in a pictorial or visual form. This images can be machine-printed or photographic (film, prints or plates). “Digitization” is the process of creating a digital image and then presenting it on a computer. Image digitization systems workflow can be classified into two types:

- Scanning

It is an electronic photographic process to create digital reproductions from original prints, film, or plates. First-generation digital images are created at the scanner are referred to as “masters”, since these “masters” will be used to produce newer versions.

- Digital Image Processing

The digital conversion process to create “production” or “delivery” image optimized for stated uses, such as networked delivery to computer monitors, prints, color-corrected publication-quality reprints and also for character recognition in a computer.



## 2.5 Image Pre-Processing

Image pre-processing is the operations with images at the lowest level of abstraction – both input and output are intensity images. The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features important for further processing. Pre-processing consists of those operations that prepare the captured image for subsequent analysis that attempts to correct or compensate for systematic errors.

The digital imageries are subjected to several corrections such as geometric, radiometric and atmospheric, although not all these corrections is necessary to be applied in all cases. These errors are systematic and should be removed before they reach the users. After pre-processing is complete, the analyst may use feature extraction to reduce the dimensionality of the data. Thus feature extraction is the process of isolating the most useful components of the data for further study while discarding the less useful aspects (errors, noise, etc). Feature extraction reduces the number of variables that must be examined, thereby saving time and resources.

Four basic types of pre-processing methods are:

- Brightness transformations
  - Brightness corrections – modify pixel brightness taking into account its original brightness and its position in the image.
  - Gray-scale transforms – change brightness without regard to position in the image.

- Geometric transformations
  - This method permits the elimination of the geometric distortions that occur when an image is captured.
- Local neighborhood pre-processing
  - This method is uses a small neighborhood of a pixel in an input image to produce a new brightness value in the output image.
  - Smoothing
  - Edge detection
- Image restoration
  - Its aim is to suppress degradation using knowledge about its nature.

## 2.6 Segmentation

Image segmentation is one of the most important steps in leading to the analysis of recognizing image data. Its main aim is to divide an image into parts that have a strong correlation with objects or areas of the real world contained in the image. Segmentation results can be divided two which is:

- Complete Segmentation – where a set of disjoint regions uniquely parallel with objects in the input image. It is necessary to cooperate with high processing level which uses specific knowledge of the problem domain.



- Partial Segmentation – where regions do not correspond directly with the image objects. Images are divided into separate regions that are identical with respect to a chosen property such as brightness, color, reflectivity and texture.

### 2.6.1 Segmentation Methods

Segmentation methods can be divided into three groups according to the dominant features they employ:

- Global knowledge / Thresholding – an image or its part; the knowledge is usually represented by a histogram of image features.
  - Edge-based segmentations
  - Region-based segmentations
- } Characteristics may be used in this 2 methods, e.g. brightness and texture.

#### 2.6.1.1 Thresholding

Thresholding is a popular technique for image segmentation. Thresholding is the operation of converting a multi-level image into a binary image. In a binary image, each pixel value is represented by a single binary digit. It is a point based operation that assigns the values of 0 or 1 to each pixel of an image based on a comparison with the global threshold value. It can lead to significant reduction in data storage and results in binary images that are simpler to analyze. Binary images permit the use of powerful morphological operators for shape and structure-based analysis of image content. It is often useful to be able to see what areas of an image consist of pixels whose value lie within a specified range, or band of intensities (or colors).

Gray level thresholding is the most simple segmentation process. Many objects or image regions are characterized constant reflectivity or light absorption of their surfaces. A brightness constant or threshold can be determined to segment objects and background. Thresholding is computationally inexpensive, fast and provides a convenient way to perform this segmentation on the basis of the different intensities or colors in the foreground and background regions of an image. It is the oldest segmentation method and is still widely used in simple application.

Threshold detection methods which is used to determine the threshold automatically includes:

- Optimal Thresholding

- Based on approximation of the histogram of an image using a weighted sum of two or more probability densities with normal distribution represent a different approach.
- The threshold is set as the closest gray level corresponding to the minimum probability between the maxima of two or more normal distributions, which results in minimum error segmentation.

- Multi-Spectral Thresholding

- Multi-spectral or color images
- It determines thresholds independently in each spectral band and combines them into a single segmented image.



- Hierarchical Thresholding
  - It is based on local thresholding methods; the aim is to detect the presence of a region in a low-resolution image, and to give the region more precision in images of higher to full resolution.

#### 2.6.1.2 Edge-Based Segmentation

Edge-based segmentation represents a large group of methods based on information about edges in the image. Edge-based segmentation relies on edges found in an image by edge detecting operators – these edges mark image locations of discontinuities in gray-level, color, texture, etc.

However, image resulting from edge detection cannot be used as a segmentation result. Supplementary processing steps must follow to combine edges into edge chains that correspond better with borders in the image. The main aim is to reach at least a partial segmentation – that is, to group local edges into an image where only edge chains with a correspondence to existing objects or image parts are present. The most common problems of edge-based segmentation, caused by image noise or unsuitable information in an image, are

- An edge presence in locations where there is no border, and
- No edge presence where a real border exists.

There are a few types of edge-based segmentation which includes:

- Edge Image Thresholding
  - It is based on construction of an edge image that is processed by an appropriate threshold.

- Edge Relaxation

- This method uses crack edges (edges located between pixels), which produce some properties, although the method can work with other edge representations as well.
- It is an iterative method, with edge confidences converging either to zero (edge termination) or one (the edge forms a border).
- All the image properties, including those of further edge existence, are iteratively evaluated with more precision until the edge context is totally clear – based on the strength of edges in a specified local neighborhood, the confidence of each edge is either increased or decreased.

- Border Detection as Graph Searching

- It represents an extremely powerful segmentation approach.
- The border detection process is transformed into a search for the optimal path in the weighted graph.
- The aim is to find the optimal path (optimal border, with respect to some objective function) that connects two specified nodes or sets of nodes that represent the border's beginning and end.



- Hough Transform
  - It is applicable if objects of known shape are to be detected within an image.
  - The Hough transform can detect straight lines and curves (object borders) if their analytic equations are known.
  - One of the advantages of this approach is robustness of segmentation results; that is, segmentation is not too sensitive to imperfect data or noise.

There edge operators is purposeful in identifying meaningful image features on the basis of distributions of pixel gray-levels. The two categories of operators include:

- Edge Pixel Detectors, which assign a value to a pixel in proportion to the likelihood that the pixel is part of an image edge (i.e. a pixel that is on the boundary between two regions of different intensity values).
- Line Pixel Detectors, which assign a value to a pixel in proportion to the likelihood that the pixel is part of an image line. (i.e. a dark narrow region bounded on both sides by lighter version, or vice-versa).

Some of the popular methods for edge detection are as below:

- Canny Edge Detector

The Canny operator is designed to be an optimal edge detector. It takes as input of a gray scale image, and produces as output an image showing positions of tracked intensity discontinuities. It is a method that looks for the edges of objects, and it is very useful for images with solid regions, or

photographic images, where one will only want to vectorise the outlines. The Canny method employs more mathematics than the simple edge detection method, and there are optional settings which can improve the results. This method will attempt to find boundaries between poorly defined objects as well as hard edges. It uses a lot memory during processing, so it may not be appropriate for very large raster, or if memory is low.

- Sobel Edge Detector

The Sobel operator performs a 2-D spatial gradient measurement on an image and emphasizes regions of high spatial frequency that correspond to edges. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image.

The Sobel operator is slower to compute than the Roberts Cross operator, but its larger convolution kernel smooths the input image to a greater extent and so makes the operator less sensitive to noise. The operator also generally produces considerably higher output values for similar edges, compared with the Roberts Cross operator.

As with the Roberts Cross operator, output values from the operator can easily overflow the maximum allowed pixel value for image types that only support smallish integer pixel values (e.g. 8-bit integer images). When this happens the standard practice is to simply set overflowing output pixels to the maximum allowed value. The problem can be avoided by using an image type that supports pixel values with a larger range.



Natural edges in images often lead to lines in the output image that are several pixels wide due to the smoothing effect of the Sobel operator. Some thinning may be needed to counter this problem.

- Roberts Cross Edge Detector

The Robert Cross operator performs a simple and quick to compute 2-D spatial gradient measurement on an image. It highlights regions of high spatial frequency which is corresponding to the edges. In its most common usage, the input to the operator is a grayscale image, as is the output. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point.

The main reason for using the Roberts Cross operator is that it is very quick to compute. Only four input pixels need to be examined to determine the value of each output pixel, and only subtractions and additions are used in the calculation. In addition there are no parameters to set. Its main disadvantages are that since it uses such a small kernel, it is very sensitive to noise. It also produces very weak responses to genuine edges unless they are very sharp. The Sobel operator performs much better in this respect.

### 2.6.1.3 Region-Based Segmentation

While edge-based segmentation is to find borders between regions; region-based segmentation is to construct regions directly. It is easy to construct regions from their borders and it is easy to detect borders of existing regions. However, segmentations resulting from edge-based methods and region growing methods are not usually exactly the same, and a combination of results may often be a good idea. Region growing techniques are generally better in noisy images where edges are very difficult to detect.

Homogeneity is an important property of region and is used as the main segmentation criterion in region growing to divide an image into zones of maximum homogeneity. The criteria for homogeneity can be based on gray level, color, texture, shape, model and etc.

Three basic approaches to region growing exist:

- Region Merging
  - Specific methods differ in the definition of the starting segmentation and in the criterion for merging.
  - The result of region merging depends in which regions are merged.
  - The simplest methods to begin merging are by using regions of 2x2, 4x4, or 8x8 pixels.
  - Region description is then based on statistical gray level properties.
  - A region description is compared with the description of an adjacent region; if they match, they are merged into a larger region and a new region description is computer. Otherwise regions are marked as non-matching. The process will be continued until all image regions are marked.



- Region Splitting

- Region splitting is the opposite of region merging.
- It begins with the entire image represented as one region which does not usually satisfy the condition of homogeneity.
- The existing image regions are sequentially split to satisfy all given conditions of homogeneity.
- It does not result in the same segmentation even if the same homogeneity criteria are used.
- While region splitting uses similar criteria of homogeneity as region merging, they only differ in the direction of their application.

- Split-and-Merge

- It is a combination of splitting and merging, and it may result in a method with the advantages of both approaches.
- This approach uses pyramid image representations, where regions are square-shaped and correspond to elements of the appropriate pyramid level.

#### 2.6.1.4 Morphological Filtering

Morphology refers to the structure of the object. Morphological filtering is to simplify the segmented image to help to search for the object of interest. It is able to smooth out the object outlines, fill up small holes in the object and to eliminate small projections. Some of the morphological filtering is as follow:

- Dilation

The basic effect of the operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels, where it allows the object to expand. Thus areas of foreground pixels grow in size while holes within those regions become smaller and objects which are disjoint will be connected together.

- Erosion

The basic effect of the operator on a binary image is to erode away the boundaries of regions of foreground pixels. Thus the areas of the foreground pixels will shrink in size, and holes within those areas will become larger.

- Thickening

Thickening is used to grow selected regions of foreground pixels in binary images. It has several applications, which includes determining the approximate convex hull of a shape, and determining the skeleton by zone of influence. It is usually applied to binary images, and produces another binary image as output.

- Hit-and-Miss Transform

It is a general binary morphological operation that can be used to look for particular patterns of foreground and background pixels in an image. It is a basic operation of binary morphology as almost all other binary



morphological operators can be derived from hit-and-miss transform. This operator takes a binary image as input and a structuring element, and produces another binary image as its output.

- Thinning

Thinning is a morphological operation that is used to remove selected foreground pixels from binary images, which is black and white image. It can be used for several applications such as skeletonization where it is a process for reducing foreground regions in a binary image to a skeletal remnant that largely preserves the extent and connectivity of the original region while throwing away most of the original foreground pixels.

Thinning is used to tidy up the output of edge detectors by reducing all lines to single pixel thickness. Thinning is normally only applied to binary images, and produces another binary image as output. The behavior of the thinning operation is determined by a structuring element. The operation is calculated by translating the origin of the structuring element to each possible pixel position in the image, and at each such position comparing it with the underlying image pixels. If the foreground and background pixel in the structuring element matches the foreground and background pixels in the image, then the image pixel underneath the origin of the structuring element is set to background. Else, it is left unchanged (zero). The structuring element must always have a one or a blank at its origin if it is to have any effect.

One of the most common of thinning is to reduce the thresholded output of an edge detector such as Sobel operator, to lines of a single pixel thickness, while preserving the full length of those lines, where pixels at the extreme ends of lines should not be affected by the thinning operation.

#### **2.6.1.5 Matching**

Matching is also a basic approach to segmentation that can be used to locate known objects in an image, to search for specific patterns, etc. The best match is based on some criterion of optimality which depends on object properties and object relations. Matched patterns can be very small, or they can represent whole objects of interest. While matching is often based on directly comparing gray-level properties of image sub-regions, it can be equally well performed using image-derived features or higher-level image descriptors. In such cases, the matching may become invariant to image transforms. Criteria of optimality can compute anything from simple correlations up to complex approaches of graph matching.

### **2.7 Recognition**

Chain code is a sequential of numbers that correspond to the basic form of characters with given orientation which is based on the definition of direction diagram. It is very similar with the system of directions such as north, southeast and west; the points of a compass. Chain code can be categorized into Absolute Chain Code and Relative Chain Code. Absolute chain codes are created using absolute points of direction according to the points of the compass. Relative chain code uses the system of direction naming which is based on positions relative to the user's current orientation.



In relative chain codes, it is possible to base an eight-directional coding system, where each of the eight directions will be defined in relation to a moving perspective. In other words, the relative chain code of the directional code representing any particular section of line is relative to the directional code of the preceding line segment.

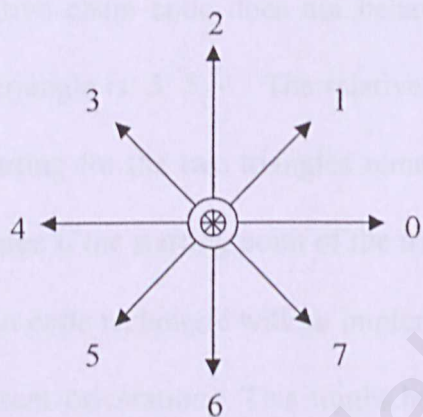


Figure 2.1: 8-direction chain code

One of the strong points of absolute chain code is that it provides information not only about the size and shape of the object but also about its rotation. Figure 2.2 gives an example of the strong points of absolute chain code.

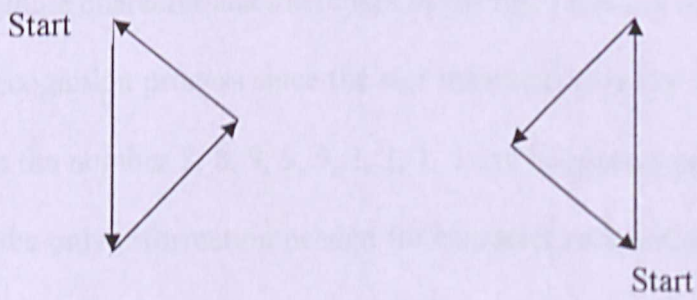


Figure 2.2: Comparison of Absolute and Relative Chain Code

The absolute chain code for the left triangle on the left is: 6, 1, 3, while on the right, it is: 2, 5, 7. These two triangles don't even have one number in common. The reason is that the rotation, size and shape of the triangle are being represented on the chain code.

On the other hand, relative chain code does not behave in such a way. The relative chain code for the left triangle is: 5, 5, 4. The relative chain code for the right triangle is also 5, 5, 4 if the starting for the two triangles remain the same. The chain code will only have a minor change if the starting point of the triangle changes.

In this research, the chain code technique will be implemented because it is able to recognize characters in different orientation. This might help in the recognition of two or more parking spaces that are opposite to each other without having to move the camera in a frame of time or having cameras installed in every direction of the car park.

## 2.8 Run Length Encoding

Run length encoding is the simplest data compression technique, by taking advantage of repetitive data. RLE consists of the process of searching of repeated runs (which is the repeating of characters) of a single symbol in an input stream, and replacing them by a single character and the length of the run. The use of RLE is able to assist the character recognition process since the size information is not required for this process. For instance the number 8, 8, 9, 9, 9, 1, 1, 1, 1 can be represented by 8X2, 9X3 and 1X4. However, the only information needed for character recognition in this case is 8, 9, 1. The longer and more frequent the runs are, the greater the compression that will be achieved. The second advantage is the shifting process of the chain code can be minimized therefore the time of recognition.



## **2.9 Chosen Algorithm**

### **2.9.1 Noise Reduction**

Median filter will be used to achieve noise reduction. The advantage of using the median filter is that it is capable of eliminating outliers such as the extreme brightness values in salt-and-pepper noise.

### **2.9.2 Gray-Scale Stretch**

Gray-scale function is capable of compressing uninterested and unwanted information and focus on the interested information. One advantage of using a grayscale image instead of a color image is because its file size is typically one-third the size of a comparable color image and making it easier for the image processing processes.

### **2.9.3 Edge Smoothing**

Gaussian filter for smoothing has become quite popular because of certain properties of Gaussian as well as several application areas. Some advantages of Gaussian filtering are rotationally symmetric (for large filters), filter weights decrease monotonically from central, and peak, giving most weight to the central pixels.

#### 2.9.4 Threshold

Optimal thresholding is chosen because it is easy and convenient to segment the image which is needed on the basis of the different intensities in the foreground and background regions of the captured image. It is able to separate out the regions that are needed with the regions of the image that correspond to the background. The threshold value is set as the closest gray level corresponding to the minimum probability between the maxima of the normal distributions, which will results in minimum error segmentation.

#### 2.9.5 Thinning

Thinning is able to obtain the 'skeleton' of an image. The 'skeleton' of a bitmap character can be obtained by peeling layers of border pixel from the image until the final object is only one pixel thick, where pixels at the ends of lines should not be affected by the thinning operation. This benefit is very important in image processing as the image will become more recognizable instead of viewing a blurred image.

#### 2.9.6 Edge Detection

The Sobel operator has been chosen for the reason that it is able to smooth the input image to a greater extent and makes it less sensitive to noise. This operator generally produces considerably higher output values for similar edges, compared with other edge detector such as the Roberts Cross operator and Canny operator.



### 2.9.7 Recognition

Relative chain code has been chosen to recognize the numeric character in the captured image. One of the advantages of relative chain code is that the chain code will only have a slight change if the starting point to generate the chain code changes. This enable the numeric characters to be recognize regardless of its orientation.

The focus of this project is to recognize characters using relative chain code algorithm. Secondly, after the output map gone through the shortest path recognition system, this system will need to recognize the shortest path direction in chain code as the direction to the destination of vacant car park space.

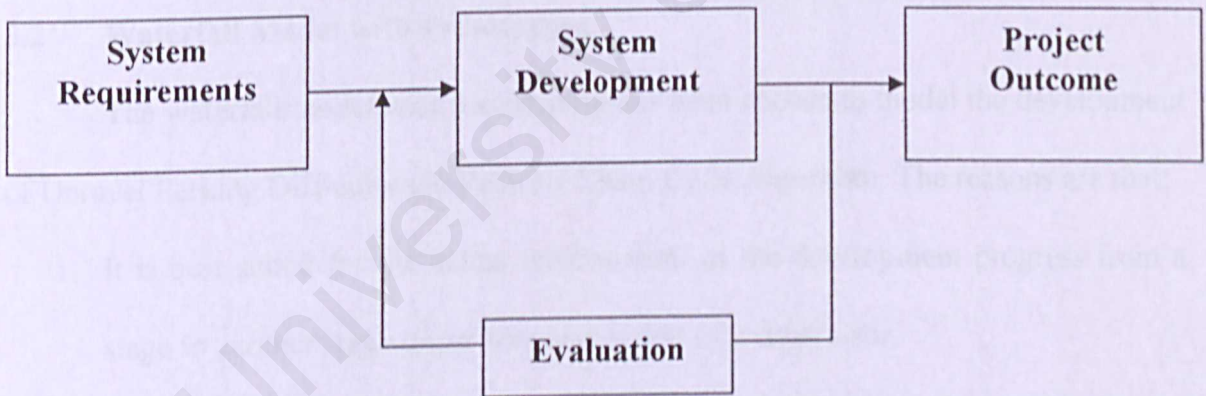
### 2.9.8 Run Length Encoding (RLE)

RLE is needed to replace a long sequence of same symbols by a shorter sequence after the system runs the relative chain code process. This process shortens the time to process the chain codes and reduces the size of storage in the knowledge base.

## Chapter 3: Methodology and Techniques

### 3.1 Methodology

The methodology adopted for the development process is very important as improper or inappropriate choice of methodology will eventually lead to the failure of the software project. There are two major factors of a software engineering project failure. The first problem is that are too many design flaws being discovered during the development of the project where it is complicated, expensive and sometimes impossible to rectify and revise it. The second problem is that the scope of the project seemed to expand rampantly and out of control as the project progresses.



**Figure 3.1: Generic System Development Model**

Every system development process model includes system requirements such as users, constraints (limitations), and resources as inputs and a fully developed system or software as the outputs. There are many popular software life-cycle models such as:



- Iterative-and-Incremental Model
- Evolution-Tree Model
- Code-and-Fix Model
- Operational Specification Model
- Waterfall Model
- Waterfall Model with Prototyping
- V Model
- Transformation Model
- Extreme Programming (XP) Model
- Synchronize-and-stabilize Model
- Spiral Model

### 3.2 Waterfall Model with Prototyping

The waterfall model with prototyping has been chosen to model the development of Unravel Parking Difficulty via Relative Chain Code Algorithm. The reasons are that:

1. It is best suited for the actual environment as the development progress from a stage to another stage under the supervision of a supervisor.
2. It allows part of the system to be developed and tested earlier to identify the problem.
3. It is easy to use and understand.
4. Validation can be done from time to time to ensure that the system has implemented all the requirements.
5. Verification ensures that each function work correctly.

Pfleeger (1991) extended the waterfall model by placing a larger emphasis on the testing component. In this model all steps are interconnected to allow a prototyping approach to be used in conjunction with the model. Every development stage should be completed before another the next begins. The eight stages are:

#### 1. Requirements Analysis

- Understand and determine the user's needs by having brainstorming, eliciting and analyzing system requirements.

#### 2. System Design

- Outlining system functionalities by having feasibility studies and case studies on current systems.
- Determine and specify hardware and software architecture.
- Verify the system design.

#### 3. Program Design

- Determine and specify the program design and database design.

#### 4. Coding

- Involve programming, personal planning, tool acquisition, database development and component level documentation.



## 5. Unit and Integration Testing

- Test the modules separately and integrate the tested modules.
- Test on the integrated modules.

## 6. System Testing

- Combine all the integrated modules into a system and test on the system.
- Specify and review the result of system test.
- Evaluate the system to meet the requirements.

## 7. Acceptance Testing

- Testing on system is completed. The system is delivered.

## 8. Operation and Maintenance

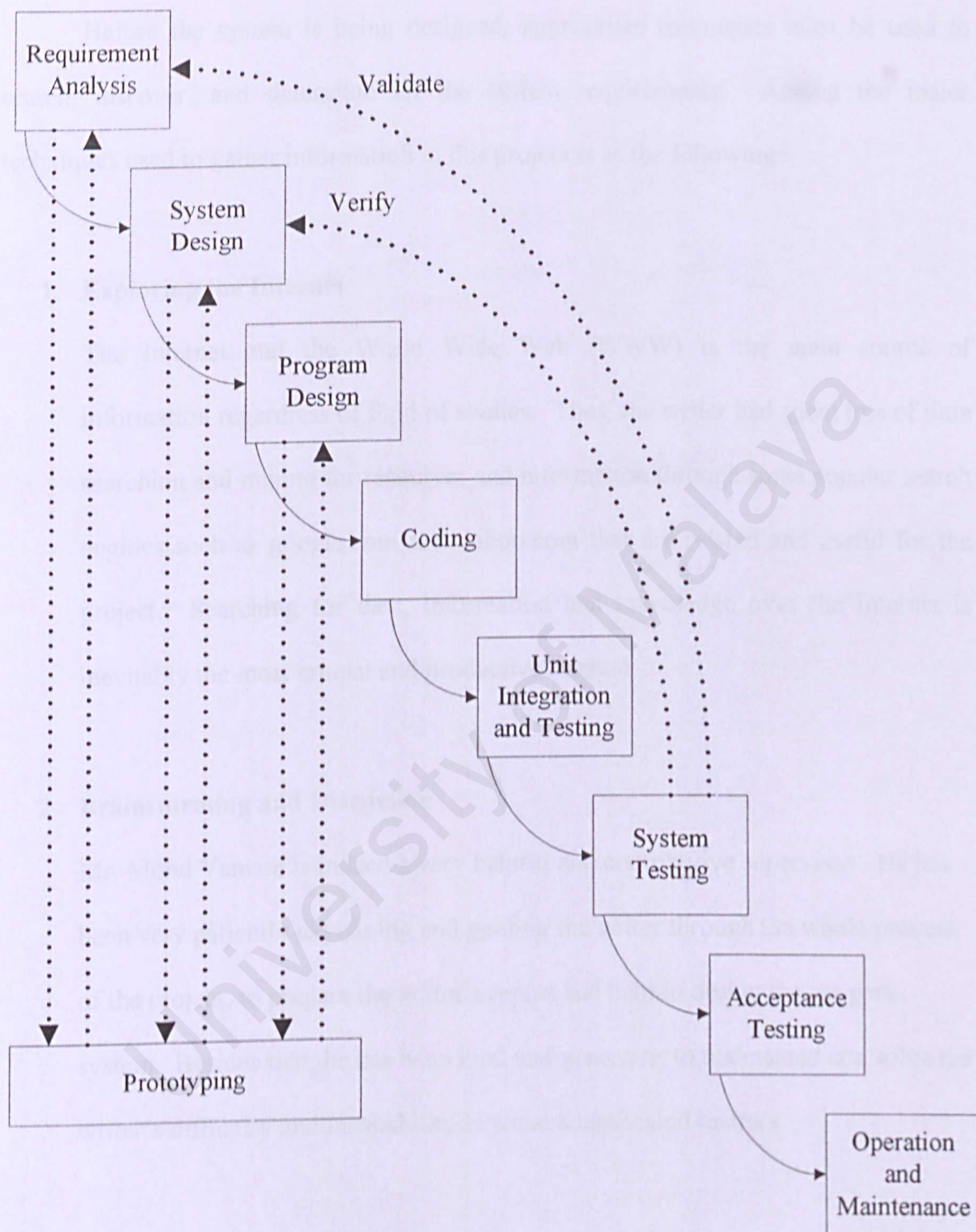
- Control and maintain the system which involves system modifications, perfecting existing acceptable functions and preventing system from degrading to an unacceptable level and revalidating of system.

The validation during system testing is to ensure that the system has implemented all the requirements, so that each system function can be tracked back to a particular requirement in the specification. As for the verification, it ensures that each function works correctly without any error and to check the quality of the implementation.

Prototyping is a sub-process and prototype is a partially developed product or a simple simulator of the actual system to examine the proposed system and the overview on the functionality of the system. The aim of prototyping is to enable input from the end-user at an early stage by giving them the look and feel of the application. This is achieved by modeling the user interface whilst having little or no content behind that interface. Prototyping is especially valuable where requirements cannot be specified clearly. Prototyping is important:

- To ensure the system achieves the performance goals or constraints.
- To ensure the system fulfills the functional requirements.
- To ensure the system is practical and flexible.
- To have an insight of how the module and sub-modules interact with each other.





**Figure 3.2: The Waterfall Model with Prototyping**

### **3.3 Techniques Used to Gather Information**

Before the system is being designed, appropriate techniques must be used to search, discover, and determine all the system requirements. Among the major techniques used to gather information in this project is as the following:

#### **1. Exploring the Internet**

The Internet and the World Wide Web (WWW) is the main source of information regardless of field of studies. Thus, the writer had spent lots of time searching and mining for resources and information through some popular search engines such as google.com and yahoo.com that are related and useful for the project. Searching for data, information and knowledge over the Internet is inevitably the most crucial and productive method.

#### **2. Brainstorming and Discussion**

Mr. Mohd Yamani is indeed a very helpful and contributive supervisor. He has been very patiently explaining and guiding the writer through the whole process of the project, to prepare the writer's report and help to design the car park system. Besides that, he has been kind and generous; to understand and solve the writer's difficulty and incapability in some complicated matters.



### **3. Discussion with partner**

As this is a two person project, the writer also spent time to discuss the project with the writer's partner, Ms Chan Yuit Pui. They would discuss on how they are progressing, what they had learnt and how to design the system. The two of them will share our knowledge on something which is related to the project and will try to solve problems or difficulties together or find help from other resources.

### **4. Studies of Existing Systems**

Through studying and surveying existing systems, it helps the writer to get a better idea on how to improve the system design and have a better understanding of the system's requirements. In the process of learning how other systems work, their functionalities and specifications, the writer have a clearer picture why the system works that way not the other way. As such, the writer knows the weaknesses and strengths of these systems and it helps the writer to improve the system to make it a better and proper one.

### **5. Written Materials**

Referring to written materials such as previous theses, computing reference books, journals and research papers either in electronic or hardcopy copy form has been a stepping stone for the writer to prepare the coming development phase. Through all these materials, it helps the writer to write the report and design the system.

## Chapter 4: System Analysis

### 4.1 Introduction

A requirement is a feature of a system or a description of what the system is capable to do in order to fulfill its purpose. Software requirement specification will give a detailed explanation on what the software is supposed to do not only the flow of information to and from the system, the transformation, processing of data by the system, but also constraints on the system's performance and capabilities. Requirement specification will help

- Software users to accurately describe what they wish to obtain;
- Allow developers and programmers to explain their understanding of how users want a system to work and function;
- Tell designers what functionality and characteristic the resultant system is to have;
- Tell developer what to demonstrate to convince users that the system is being delivered according to what was ordered.



## 4.2 Functional Requirement

Functional requirement specifies functions that a system or a system's component must be able to perform. These are software requirements that define behaviors of a system, that is, the fundamental process or transformation that software and hardware component of the system perform on input to produce the expected outputs.

As for the Unravel Parking Difficulty via Chain Code Algorithm, the functional requirements are as below:

- The input image which is captured by the camera should be in digital format, which is in GIF, JPEG, TIFF and HDF.
- Image pre-processing and enhancement includes suppress all irrelevant information of the image, perform noise reduction and smoothing processes, and ultimately to produce a better quality image.
- Segmentation process is being used to differentiate and separate the image into individual numeric character for easier recognition process.
- Character recognition is the last process where the images is being differentiated and recognize without having ambiguity.
- Run Length Encoding is used to compress a long sequence of relative chain codes into a shorter code that not only shortens the time to process it but to save the storage space of the knowledge base.

## 4.3 Language

### 4.3.1 MATLAB (MATrix LABoratory)

#### 4.3.1.1 What is MATLAB?

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows user to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of time it would take to write a program in a scalar noninteractive language such as C or FORTRAN.

The name MATLAB stands for *matrix laboratory*. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and



EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of add-on application-specific solutions called *toolboxes*. Very important to most users to MATLAB, toolboxes allow user to *learn* and *apply* specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logics, wavelets, simulation, and many others.

#### 4.3.1.2 The MATLAB System

The MATLAB system consists of five main parts:

- Development Environment

This is the set of tools and facilities that help user to use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, and browsers for viewing help, the workspace, files, and the search path.

- The MATLAB Mathematical Function Library

This is a vast collection of computational algorithms ranging from elementary functions such as sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

- The MATLAB Language

This is a high-level matrix/array language with control flow statement, functions, data structure, input/output, and object-oriented programming features. It allows both “programming in the small” to rapidly create quick and dirty throw-away programs, and “programming in the large” to create complete large and complex application programs.

- Graphics

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow user to fully customize the appearance of graphics as well as to build complete graphical user interfaces on MATLAB applications.



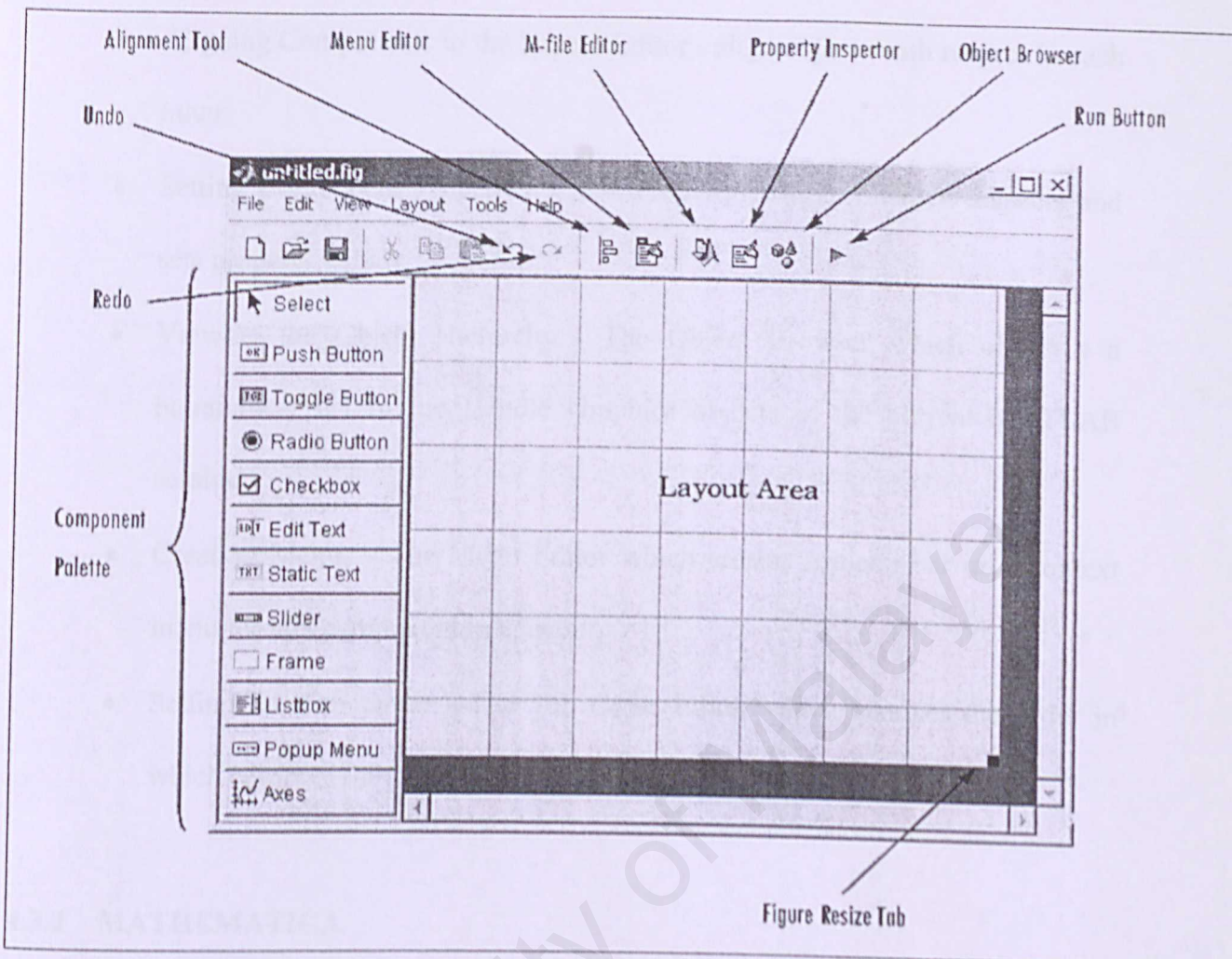
- The MATLAB Application Program Interface (API)

This is a library that allow user to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

#### **4.3.1.3 MATLAB GUIDE (Graphical User Interface Development Environment)**

GUIDE, the MATLAB Graphical User Interface development environment, provides a set of tools for creating GUIs. These tools greatly simplify the process of laying-out and programming a GUI.

When one opens a GUI in GUIDE, it is displayed in the Layout Editor, which is the control panel for all of the GUIDE tools. The Layout Editor enables one to lay out a GUI quickly and easily by dragging components, such as push buttons, pop-up menus, or axes, from the component palette into the layout area. The following picture shows the Layout Editor:



**Figure 4.1: MATLAB GUI Layout Editor**

Once one lays out his GUI and set each component's properties, using the tools in the Layout Editor, he can program the GUI with the M-file Editor. Finally, when one press the Run button on the toolbar, the functioning GUI appears outside the Layout Editor window.

The list of GUIDE toolsets is as below:

- Laying Out GUIs - The Layout Editor which adds and arranges objects in the figure window.



- Aligning Components in the Layout Editor - align objects with respect to each other.
- Setting Component Properties - The Property Inspector which inspects and sets property values.
- Viewing the Object Hierarchy - The Object Browser which observes a hierarchical list of the Handle Graphics objects in the current MATLAB session.
- Creating Menus - The Menu Editor which creates a menu bar or a context menu for any component in a layout.
- Setting the Tab Order - The Tab Order Editor which changes the order in which components are selected by tabbing.

## 4.3.2 MATHEMATICA

### 4.3.2.1 Introduction to MATHEMATICA

MATHEMATICA is a versatile, powerful application package for doing mathematics and publishing mathematical results. It runs on most popular workstation operating systems, including Microsoft Windows, Apple Macintosh OS, Linux, and other Unix-based systems.

MATHEMATICA is used by scientists and engineers in disciplines ranging from astronomy to zoology; typical applications include computational number theory, ecosystem modeling, financial derivatives pricing, quantum computation, statistical analysis, and hundreds more.

The best way to understand MATHEMATICA is to see it in action. The sections below describe three major categories of usage:

- User Tool: MATHEMATICA can be used to perform computations, either numeric or symbolic. Results can be visualized using 2-D and 3-D graphics.
- Programming Tool: MATHEMATICA provides a rich set of programming extensions to its end-user language. Programming can be done in procedural, functional, or logic (rule-based) style, or a mixture of all three. For tasks requiring interfaces to the external environment (such as extraction from a relational database) MATHEMATICA provides MathLink, which allows MATHEMATICA programs to communicate with external programs written in C, Java, or other languages.
- Publishing Tool: MATHEMATICA has extensive capabilities for formatting graphics, text, and equations. Documents, called notebooks, can be exported as PostScript, TeX, HTML, or a combination of HTML and MathML (Mathematical Markup Language).



4.3.3 Java

Algorithm developers and scientists want a tool that allows them to concentrate on their work (math, science), not on the tool (C, Mathematica, Java or Matlab). Java is an object-oriented and platform independent programming language introduced by Sun Microsystems in Jun 1995. Image in Java can be image class, imageProducer interface, imageConsumer interface, imageFilter class (implement imageConsumer interface), and filteredImage source class (implement imageProducer interface).

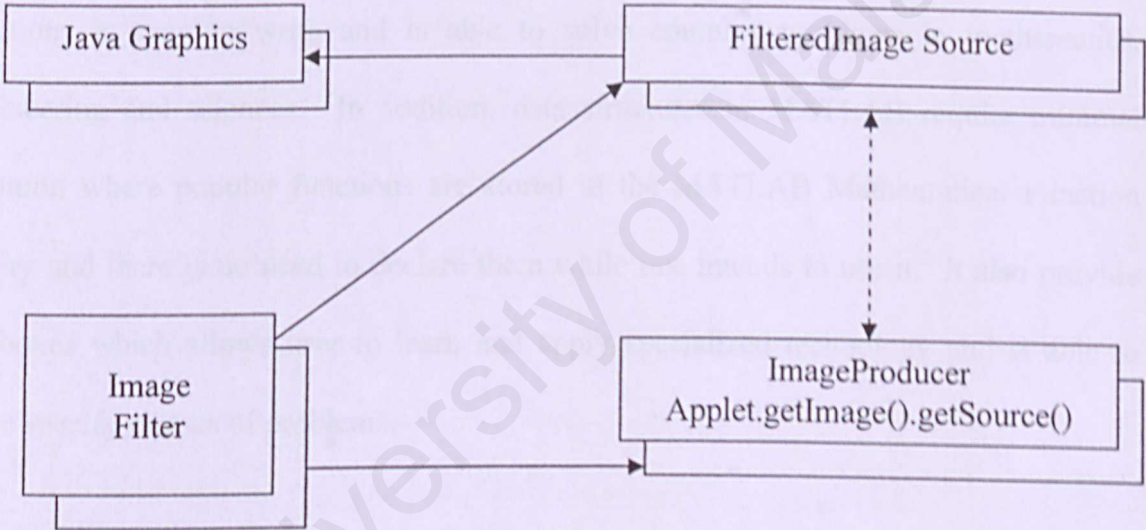


Figure 4.2: Java Image Processing

#### 4.4 Chosen Language

MATLAB has been chosen as the programming language because it is a powerful tool for general matrix manipulations and image processing. It is able to perform the image processing processes such as image pre-processing, enhancement, segmentation which includes thinning, thresholding, and edge detection. Because of the built-in functions, it will probably save a lot of time in the process of writing code and debugging.

MATLAB is very easy to learn as it is a high-performance language for technical computing which integrates computation, visualization, and programming in a user friendly environment. MATLAB code which is expressed in familiar mathematical notations is easy to write and is able to solve complex problems in mathematics, engineering and sciences. In addition, data structures in MATLAB require minimal attention where popular functions are stored in the MATLAB Mathematical Function library and there is no need to declare them while one intends to use it. It also provides toolboxes which allow users to learn and apply specialized technology and is able to solve specific classes of problems.

#### 4.5 Non-Functional Requirement

Non-functional requirement does not describe what a system or software will do rather HOW it does. For example, software performance requirements, some external interface requirements, software design constraints, and software quality attributes. Non-functional requirements are difficult to test; they are usually, or most of the time, evaluated subjectively. Non-functional requirements have been identified and acknowledged as a vital and crucial determinant to the success of many software projects.



For the Unravel Parking Difficulty via Relative Chain Code Algorithm, the following requirements have been set:

### **1. Reliability**

In this project, the reliability of the system is stress on the performance of the system to recognize the numeric character in a specific time frame and accurately. The reliability is affected by the performance of the system whether it is able to handle the workload during peak period of the car park. It is closely related on the system ability to recognize the numeric character without error and pass it to the shortest path system to determine the shortest way to arrive at the vacant space for the car park user. Thus it is important the whole system works accordingly to attain to its expectation.

### **2. Maintainability**

System maintenance often requires more effort and time if the system is not well planned and designed at the beginning. The system maintenance will allow users to make certain change or modification to the system according to the current needs and usage of the system.

### 3. Efficiency

Efficiency is the ability of a process procedure to be called or accessed unlimitedly to produce similar performance outcomes at an acceptable or credible speed. In this car park system, efficiency comes into picture as how fast the system can process the images that are being captured by the camera and then go through the image processing process. It is also about how well and accurate the system can inform users which car park is available for car park users to park their car.

### 4. Flexibility

In terms of flexibility, the car park system is a flexible system as it is a standalone system and can be actually executed over many Microsoft's, UNIX and Linux operating systems.

### 5. User Friendly

In this car park system, the user interface design aims is to fulfill three golden rules of user friendliness, which are:

- Place user in control

This will define interaction modes in a way that does not force a user into unnecessary or undesired actions or situations. Moreover, it also provides flexible interaction, for instance, via mouse movement and keyboard commands.



- Reduce the user's memory load

One of the principles that enable an interface to reduce the user's memory load is by reducing demand on short term memory. The user's interface should be designed to reduce and minimize the memory needed to load and execute the system.

- Make the interface consistent

The user's interface design should conform to consistent fashion where all visual information must be organized according to a design standard what is maintained throughout all screen displays. Apart from that, input mechanisms are restricted to limited sets that are used consistently throughout the application.

## **6. Correctness**

Correctness is the level for which the system performs according to its required function. To ensure that the Unravel Parking Difficulty via Relative Chain Code Algorithm system meet its requirements, the system will be reviewed from time to time together with the supervisor or moderator. This is important to assure the quality and maturity of the system.

#### 4.6 Hardware Requirement

Hardware includes any physical (that can be touched) device or peripheral that is connected to a computer and is controlled by the computer's microprocessor. Below are the hardware requirements for the system:

- Processor : Pentium II 166 MHz & above, AMD Athlon (TM) XP 2000
- Memory : 560 Mbytes of RAM, 256 or 512 Mbytes DDR RAM
- Hard Disc : At least 4.75 Gigabytes
- Graphic Adapter : 8-bits (for 256-simultaneous colors)
- Input Device : Keyboard, mouse.
- Output Device : Printer, monitor.



#### 4.7 Software Requirement Chapter 5: System Design

Software is a common term for a range of programs used to operate computers and its related devices. Software can be divided into system software and application software. System software is usually operating systems that support application software. In the meantime, application software is programs that do work that users are directly interested in. Below are the software requirements for the system:

- Operating System : Microsoft Windows XP Professional Version 2002 SP1, Windows 95, Windows NT 4.0 with Service Pack 3 or Linux, Macintosh (both 68000 and Power Macintosh) and workstations such as IBM RS/6000, HP 9000 PA-RISC, DEC Alpha (under DEC UNIX 4.0C) and Sun SPARC.
- Authoring Tool : MATLAB 6.0 or MATLAB 6.5 (Release 12 or Release 13)

## **Chapter 5: System Design**

### **5.1 Introduction**

System design is the specification of a detailed computer-based solution. The description of the system is called design where it focuses on the technical or implementation concerns of the system.

### **5.2 Conceptual Design**

The Unravel Parking Difficulty via Relative Chain Code Algorithm is design for car park operators to manage the car park efficiently thus attracting more customers to their car park. The system will help car park users to maneuver their vehicle to the vacant space with ease, therefore saving time and fuel in searching for a vacant space in a wide area of the car park manually. The car park operator is able to upload the latest output map to the screen of a monitor or television at the entrance of the car park. The output map will show the result of the recognition which consist of the vacant spaces in the car park that serves as a guideline to car park users to park their vehicle.

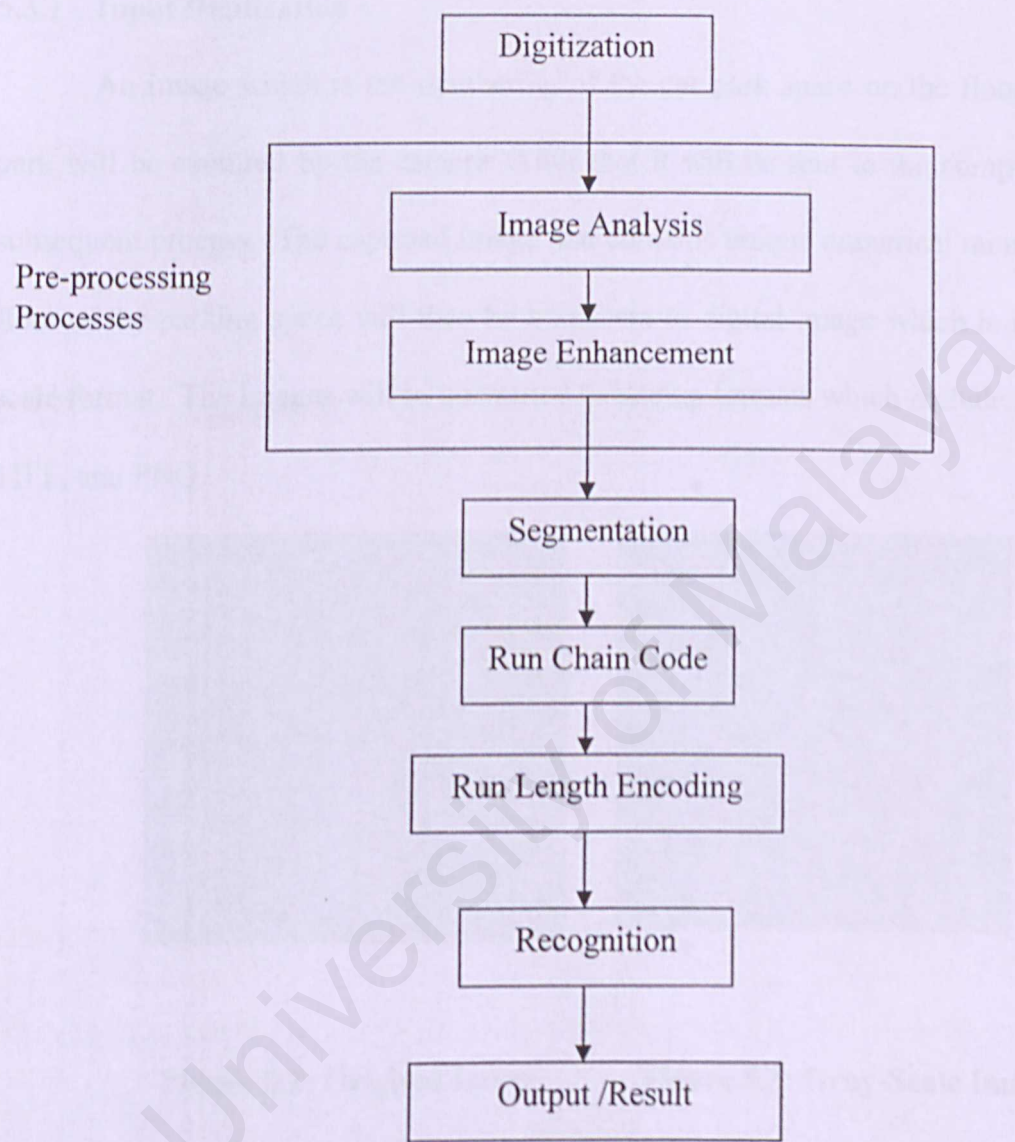
The system is also being designed for easier maintainability and allowed enhancement in the future. Additional features such as security issue conceptual from additional module can be implemented to the original version of the system.

### **5.3 System Design**

By comparing the existing car parking systems, languages and algorithms that are suitable to be used for Unravel Parking Difficulty via Relative Chain Code



Algorithm system, some decisions has been made on noise reduction, image enhancement, segmentation, recognition and compression processes.



**Figure 5.1: Process Flow of the Implementation Phases**

The following section will discuss in detailed concerning the chosen algorithm of the related processes.

### 5.3.1 Input Digitization

An image which is the numbering of the car park space on the floor of the car park will be captured by the camera. After that it will be sent to the computer for the subsequent process. The captured image that contains unique numerical numbers on the floor of the parking space will then be transform to digital image which is in the gray-scale format. The images will be converted to bitmap formats which include JPEG, GIF, TIFF, and PNG.

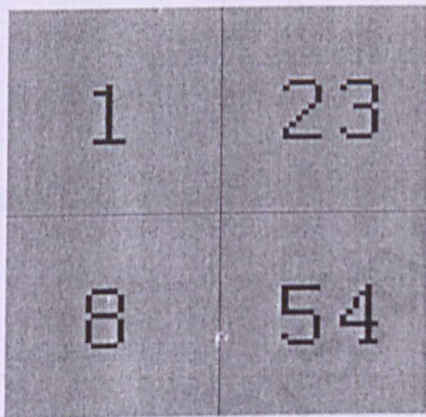


Figure 5.2: Original Image

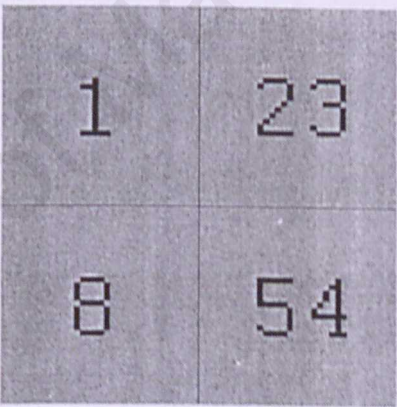


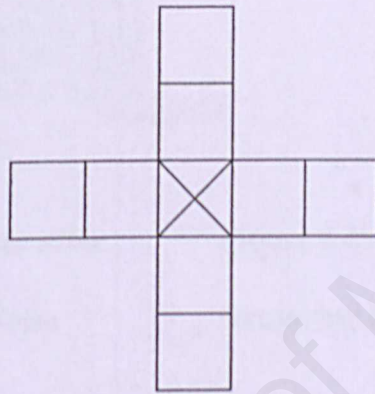
Figure 5.3: Gray-Scale Image

### 5.3.2 Adding and Removing Noise

In digital image processing, image smoothing, or more specific, noise removal is one of the most important elements in enhancement. However, most of the techniques have the side effect of blurring the image especially at the sharp edges that result in poor quality images.



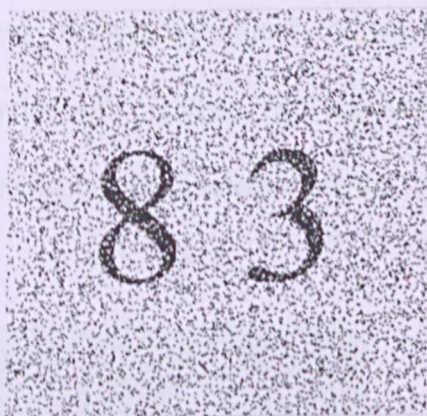
As such a non-linear filter-median filter has been chosen to achieve noise reduction. Sources of noise which is resulted from the CCD chip of a camera that is cause by the electronic signal fluctuations in the detector and also related to thermal energy. Average pixels in the image become corrupted by noise, as its fluctuations are fast and high in frequency. Median filter is able to produce better performance by retaining fine character of the image that may be smoothed by averaging.



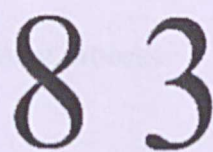
**Figure 5.4: Horizontal/Vertical Line Preserving Neighborhood for Median Filtering**

Median filter operates on a local neighborhood. Each pixel of the numeric character will take turns to check its neighbor pixel to get to know whether its neighbor pixel is disturbing the character or not. After the size of the local neighborhood is defined, the median value will be calculated starting from separating all neighboring pixel value into numbers in ascending orders. The median value is then assigned to the pixels in the character's image. For example, consider the set of numbers, 8, 23, 26, 2, and 15. These numbers being rearranged in an ascending order will result in 2, 8, 15, 23, 26, where 15 is the value of the median. Therefore, in median, filtering, the gray-level

of each pixel is replaced by the median of the gray level in a neighborhood of the pixel, instead of the average.



**Figure 5.5: Numeric Character after adding Salt-and-Pepper Noise**



**Figure 5.6: Numeric Character after Removing Noise using Median Filter**

### 5.3.3 Image Pre-Processing Processes

The digitized image will be enhanced to produce a more quality image. In the image pre-processing process, information that is not relevant to the specific image processing will be suppressed. Its aim is an improvement of the image data that suppresses undesired distortions or enhances some image features important for further processing.



5.3.3.1 Intensity / Brightness Adjustment

The captured image usually is not centered in intensity therefore the maximum number of bits is used to span the intensities of interest. This weakness can be corrected by adjusting the brightness and contrast of the image. The contrast may be adjusted linearly or logarithmically. This adjustment is important to provide brightness for the numeric character image to continue the following image processing process.

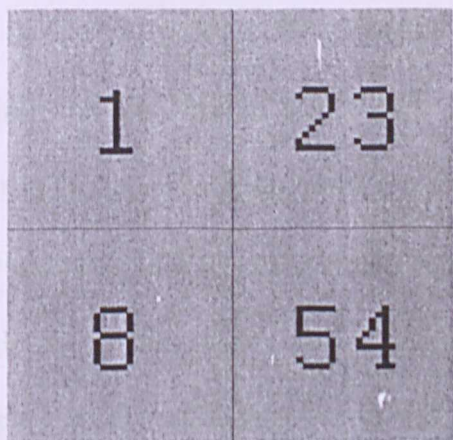


Figure 5.7: Original Gray Scale Image

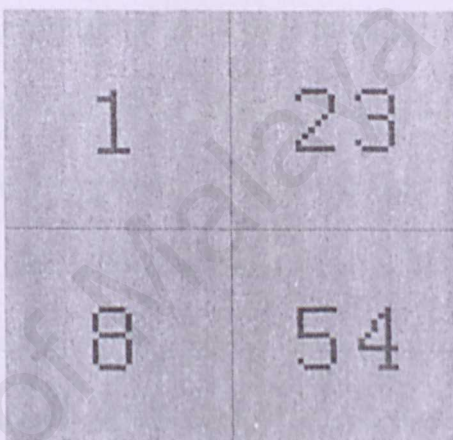


Figure 5.8: Image after Intensity Adjustment

5.3.4 Segmentation

Segmentation is to partition the image into several constituent components. Segmentation is an important process in the Unravel Parking Difficulty via Relative Chain Code Algorithm, because this technique enable user to distinguish between the objects of interest, in this case the numeric characters, with “the rest”. This latter group is also referred to as the background of the image.

5.3.4.1 Threshold

Gray-level thresholding is the simplest segmentation process and it is performed in the basis of the different intensities or colors in the foreground and background of an image. Numeric character of camera image which is characterized by constant reflectivity or light absorption of their surfaces: a brightness constant or threshold can be determined to segment the unique numeric character in a captured image from the car park.

The input for a thresholding operation is typically a gray-scale or color image (RGB) and the output is a binary image or black and white image representing the segmentation. The black pixels correspond to the background of the image and white pixels correspond to foreground (or *vice versa*). The segmentation is determined by a single parameter known as the *intensity threshold*. In a single pass, each pixel in the image is compared with this threshold. If the pixel's intensity is higher than the threshold, the pixel is set to white in the output. If it is less than the threshold, it is set to black.

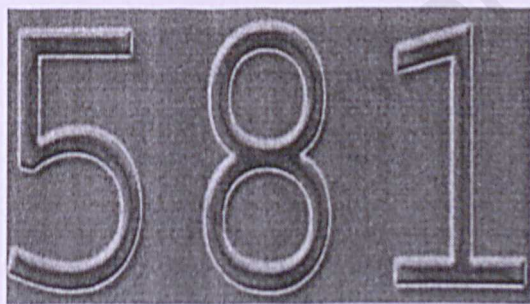


Figure 5.9: Original Gray Scale Image

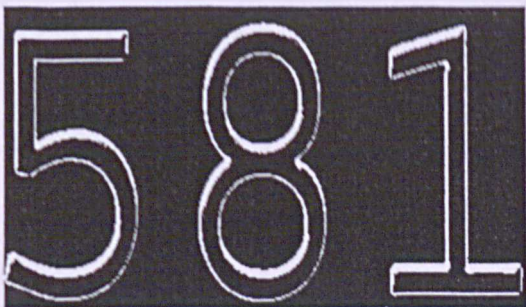


Figure 5.10: Image after Threshold



#### 5.3.4.2 Thinning

Image thinning is to reduce the size of the image while retaining the image characteristics. It is used to reduce the thresholded output of the Sobel operator, to lines of a single pixel thickness, while preserving the full length of those lines.

In the numeric character image which has been thresholded, all the pixels on the boundaries of foreground regions will be considered (*i.e.* foreground points that have at least one background neighbor). Points that have more than one foreground neighbor will be deleted, as long as it does not locally disconnect or split into two the region containing that pixel. Iterate until convergence. This procedure will erode away the boundaries of the foreground object—the numeric character, as much as possible, but does not affect pixels at the ends of lines.

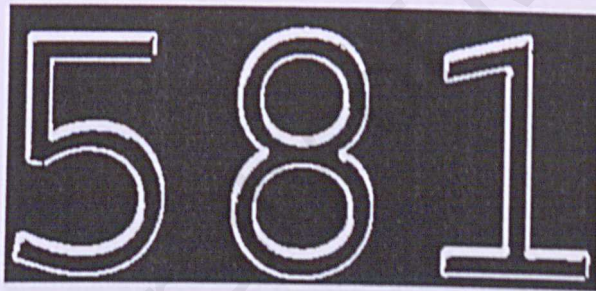
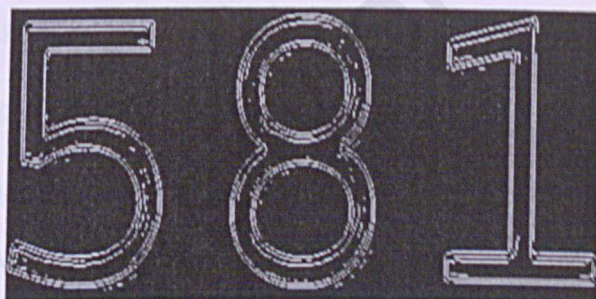


Figure 5.11: Image after Thinning

### 5.3.4.3 Edge Detection

Edge detection is used to find the edges of the numeric character in the captured image using edge detecting operators. These edges will mark image locations of discontinuities in gray-level, color, texture, etc.

The Sobel operator has been chosen to detect the edges of the numeric character in the captured image as it looks for an edge in horizontal and vertical direction. One of the advantages of Sobel operator is that it is able to smooth the input of the image to a greater extent and makes it less sensitive to noise. It performs a 2-D spatial gradient measurement on the captured image and so emphasizes regions of high spatial frequency that correspond to edges. It is used to find the approximate absolute gradient magnitude at each point in an input of gray-scale image.



**Figure 5.12: Image after Sobel Operation**



5.3.5 Recognition

Recognition is the most important process in the whole system, as it is crucial to recognize the numeric character which has been captured. The relative chain code algorithm will be used for the recognition process. Chain code is a sequential of numbers that represent the basic form of characters based on the definition of 8-direction diagram (refer to *Figure 5.13*), which is similar to the direction of the compass (north, south, northeast and etc). The relative chain code has been chosen because of the directional code representing any particular section of line is relative to the directional code of the preceding line segment. The 8-direction chain code diagram will be the direction guideline to calculate the relative chain code for the numeric characters. Images represented by the diagram will become smoother and complete through using the 8-direction chain code diagram. Each of the numeric character will be represented by the numbers and direction of the diagram. The method to read it is by anti-clockwise

After the chain code has been read and produced for the numeric character, it will be followed by the matching process where the chain code that has been read will be compared with the chain code in the knowledge base.

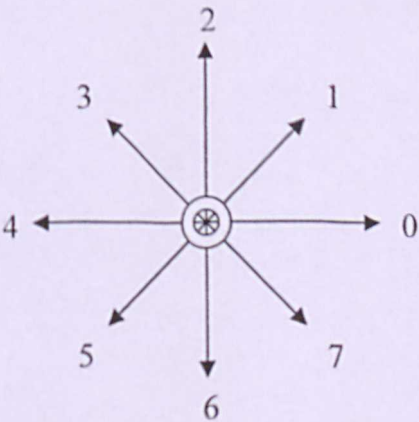


Figure 5.13: 8-direction Chain Code Diagram

### 5.3.6 Knowledge Base

Knowledge base is needed for the matching and recognition path process in this system. It will be build before input of all captured images goes through the processes in this system.

Knowledge base corresponds to a dictionary for the recognition process to store all the relative chain code of numeric characters from 0 to 9. Every input of data into the knowledge base is in relative chain code and it is not in the bitmap image file format. The knowledge base will be used in the recognition process for matching the related numeric character with the stored relative chain code of that particular numeric character. Matching will be done by comparing the relative chain code of the numeric character of the captured image with the relative chain code in the knowledge base.



### 5.3.7 Run Length Encoding (RLE)



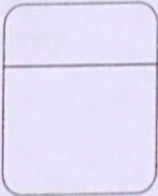
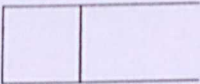
Run length encoding (RLE) is a very simple form of data compression in which runs of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run. It is based on the idea of replacing a long sequence of repeated symbol into a shorter sequence (run). In this process, RLE will replace sequences of consecutive repeated characters of the relative chain code into single character with the length of the run.

An example for the relative chain code for number 8 is 2, 2, 3, 3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1. 1, 1, 1 can be represented by 2X2, 3X6, 1X3. However, the information needed for the character recognition process is only 2, 3, and 1. Using RLE not only minimized the space to store the data but also shortened the time for the numeric character recognition process in the system.

### 5.4 Data Flow Diagram (DFD)

Data flow diagram is a process model used to depict the flow of data through a system and the work or processing performed by the system at any level of detail with a graphic network of symbols showing data flows, data stores, data processes, and data sources or destinations. DFD will depict the overview of the system inputs, processes and outputs. DFD is easy to understand as it has symbols that specify the physical aspects of the implementation. DFD is composed of four basic symbols as shown below.

**Table 5-1: Data Flow Diagram (DFD) Symbols**

Symbols	Attributes
	The External Entity symbol represents sources of data to the system or destinations of data from the system.
	The Data Flow symbol represents movement of data.
	The Process symbol represents an activity that transforms or manipulates the data.
	The Data Store symbol represents data that is not moving.

The purpose of the data flow diagram is to provide a semantic bridge between user and the system developers. The DFD is:

- Graphical which eliminates thousands of words;
- Logical representations, modeling WHAT a system does, rather than the physical model that shows HOW it does;
- Hierarchical which shows the system at any level of detail; and
- Jargonless which allows user to have a better understanding and reviewing.



The goal of DFD is to have a commonly understood model of the system. The diagram is the basis of structured system analysis.

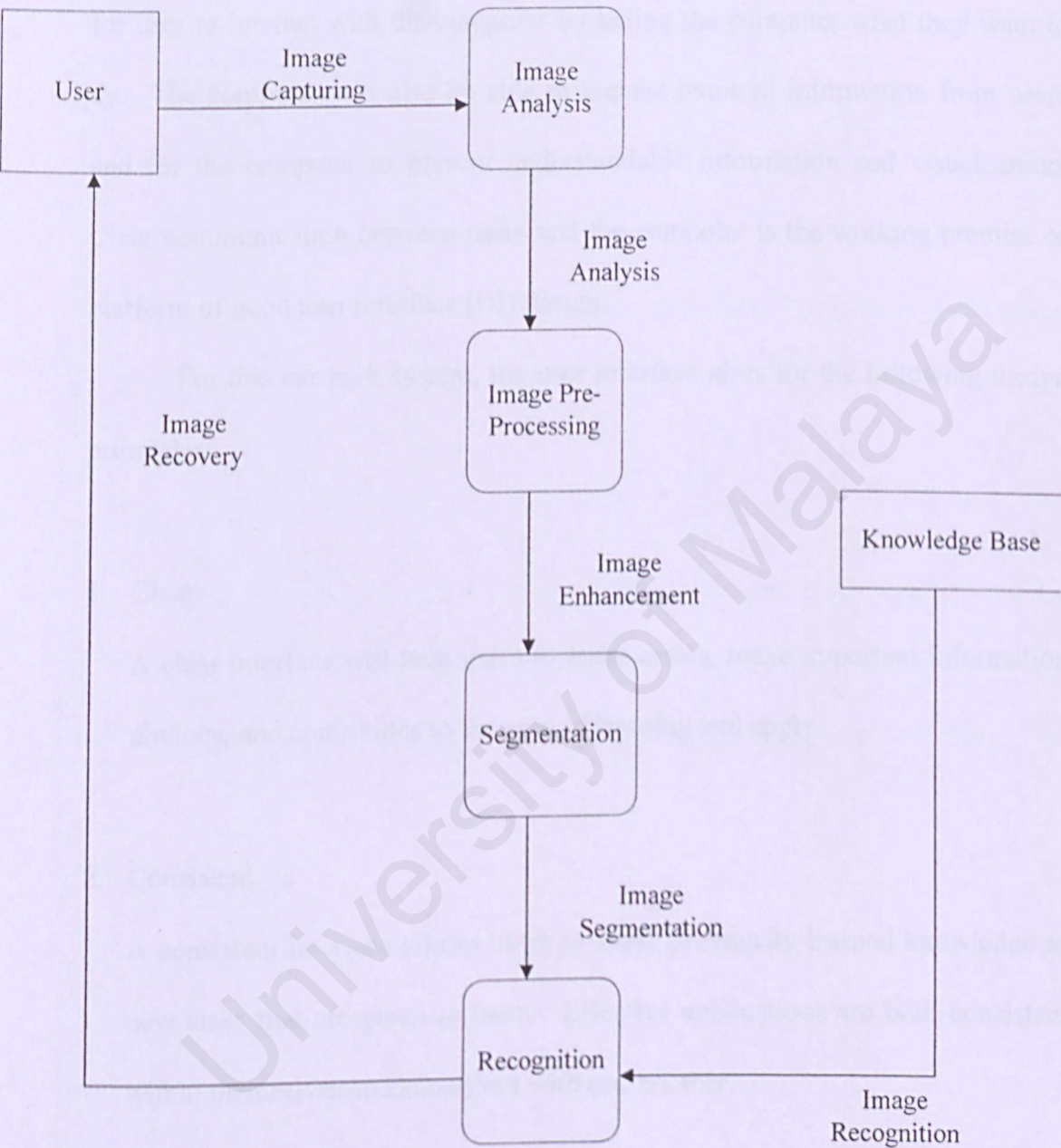


Figure 5.14: Data Flow Diagram of the Unravel Parking Difficulty via Relative Chain Code Algorithm System

## 5.5 User Interface Design

User interface design is concern about how effectively users are able to use a system and how they benefit from using it. A good interface makes it easy for user to interact with the computer by telling the computer what they want to do. The computer will also be able to request required information from users and for the computer to present understandable information and visualization. Clear communication between users and the computer is the working premise or platform of good user interface (UI) design.

For this car park system, the user interface aims for the following design principles:

### 1. Clear

A clear interface will help users to make errors, make important information obvious, and contributes to the ease of learning and apply.

### 2. Consistent

A consistent interface allows users to apply previously learned knowledge to new tasks that are given to them. Effective applications are both consistent within themselves and consistent with one another.



### 3. Simple

The best interface designs are simple and straightforward. Simple designs are easy to learn, use and also give the interface a consistent look. A design requires a good balance of maximizing functionality and maintaining simplicity through progressive disclosure of information.

### 4. User-Controlled

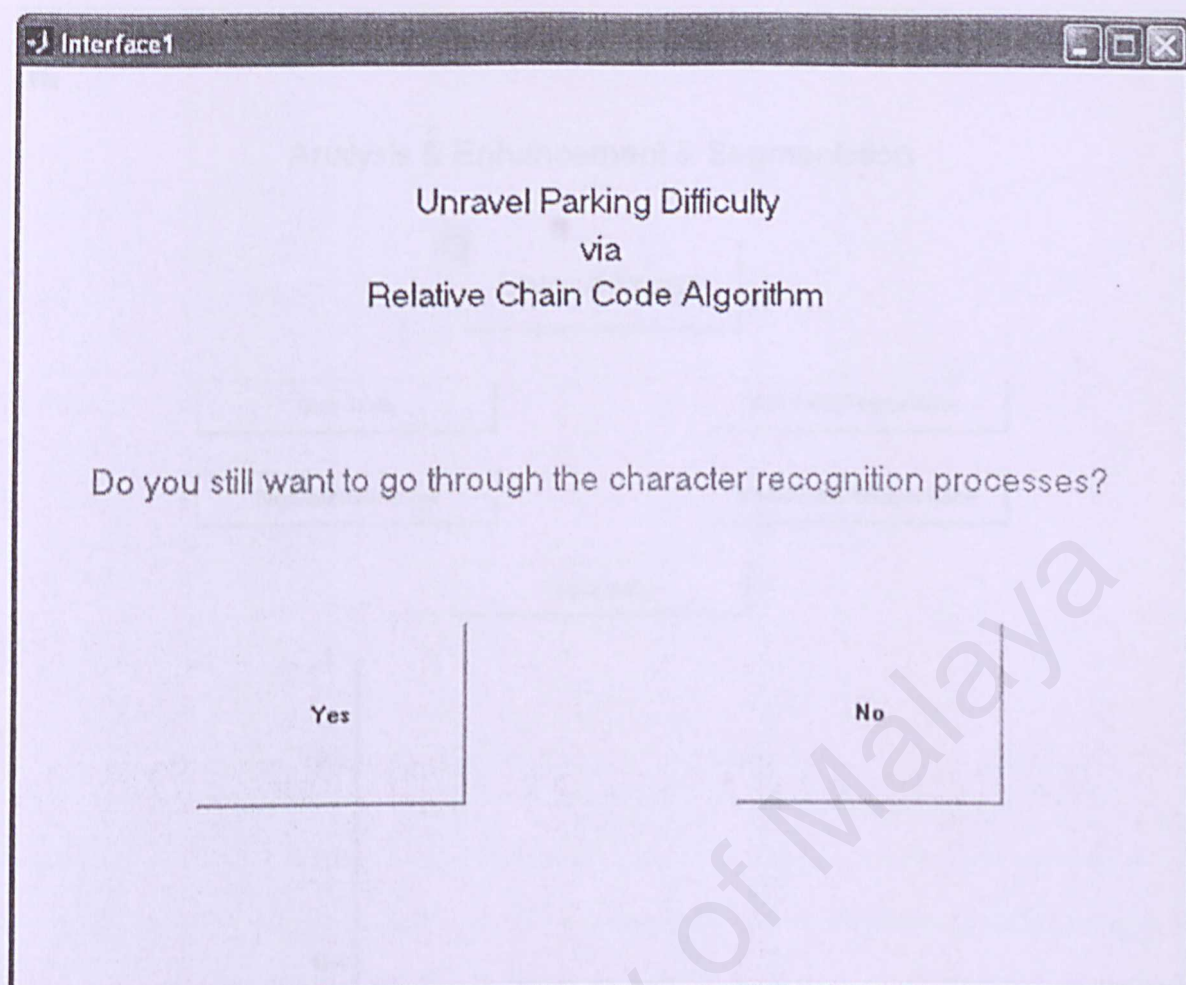
The users should be the one who initiate and control all actions and not the computer.

### 5. Direct

Users must be able to see the visible cause-and-effect relationships between the actions they take and the objects on the screen. This allows users to feel that they are in charge of the computer's activities and not the computer itself.

### 6. Aesthetic

Every visual element that appears on the screen potentially competes for the users' attention. It provides an environment that is pleasant to use and contributes to the users' understanding of the information and simulation presented.



**Figure 5.15: Main page of the Unravel Parking Difficulty via Relative Chain Code Algorithm System**



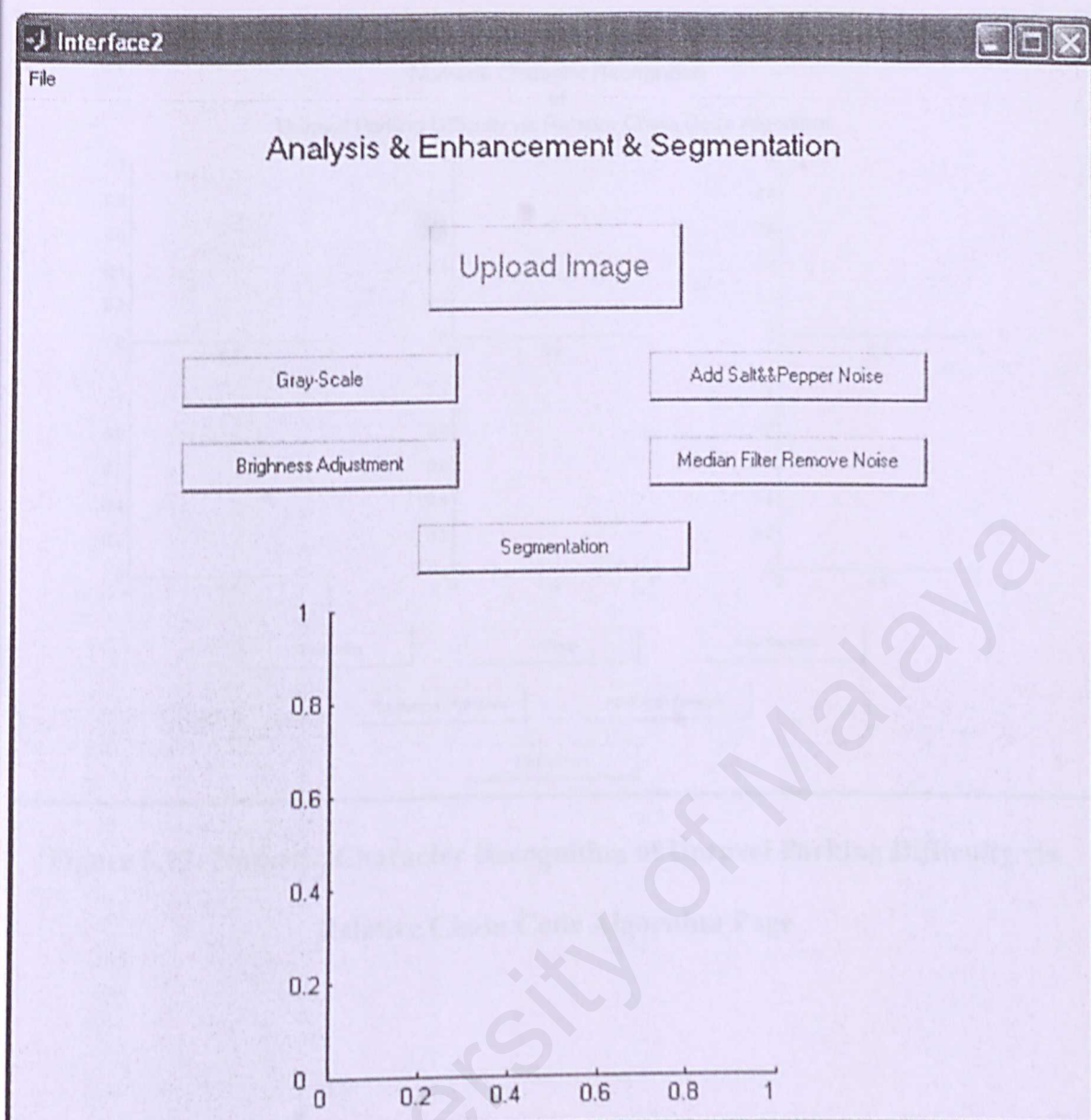
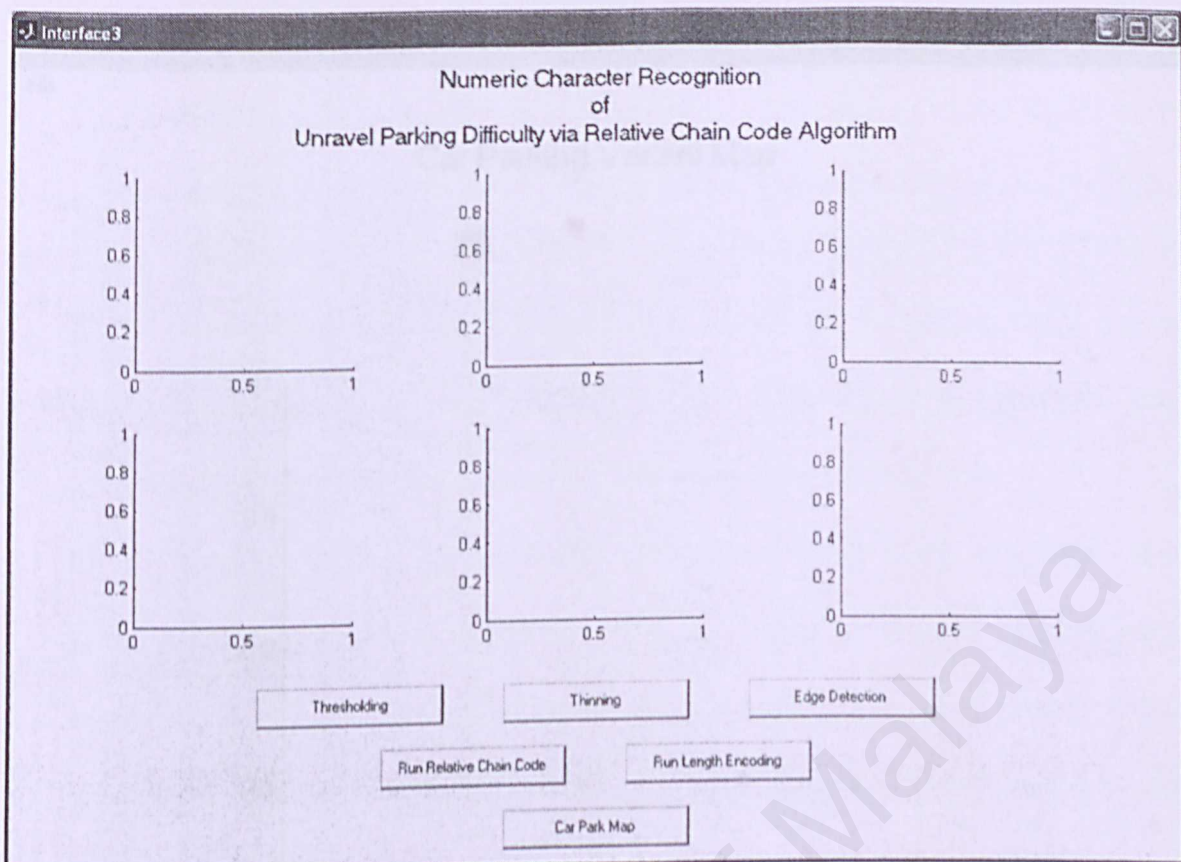


Figure 5.16: Analysis, Enhancement and Segmentation Process Page



**Figure 5.17: Numeric Character Recognition of Unravel Parking Difficulty via  
Relative Chain Code Algorithm Page**



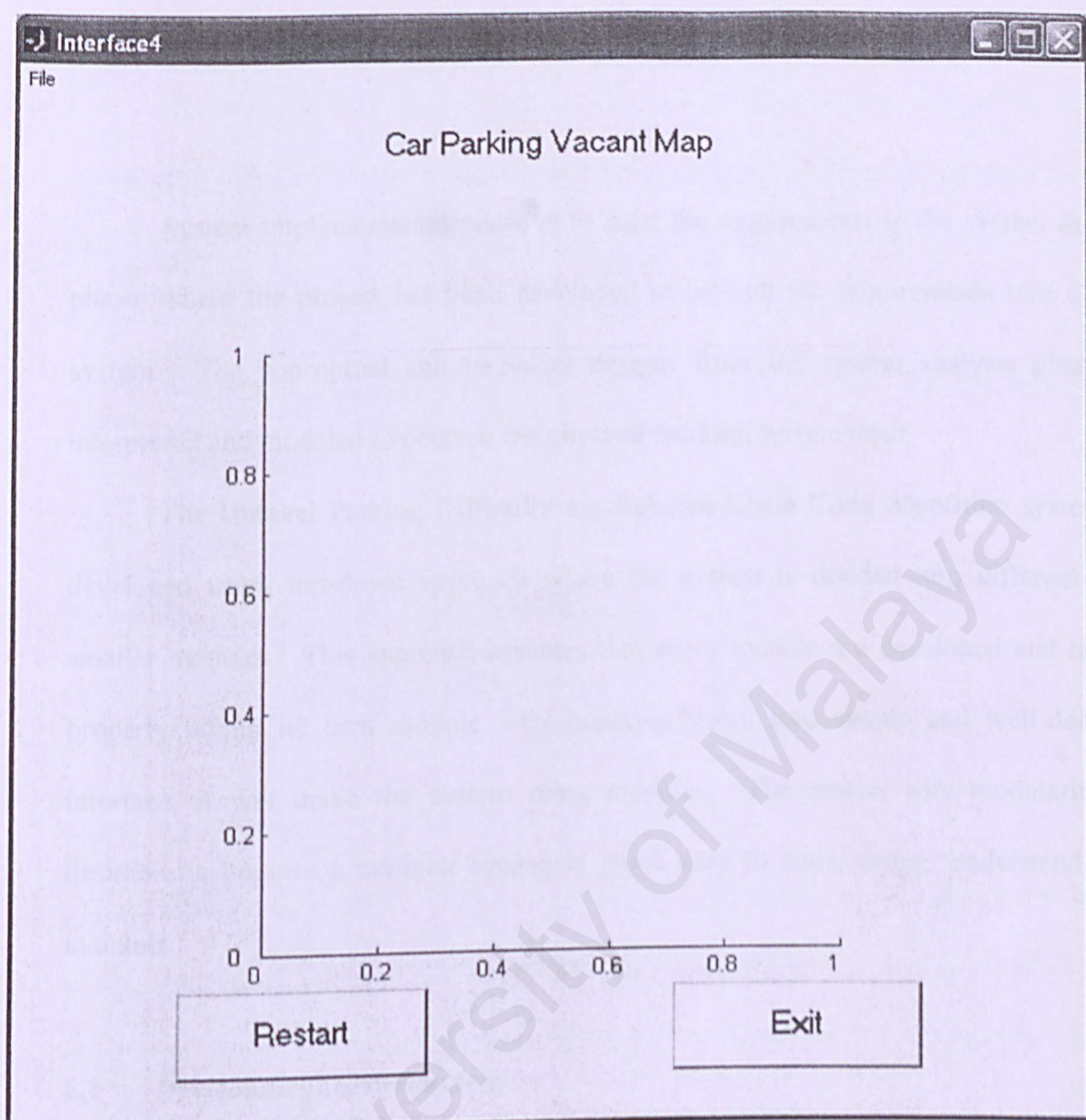


Figure 5.18: Car Parking Vacant Map Page

## Chapter 6: System Implementation

System implementation phase is to meet the requirements in the system design phase, where the project has been developed to turn all the requirements into a real system. The conceptual and technical designs from the system analysis phase is interpreted and modeled to become the physical working system itself.

The Unravel Parking Difficulty via Relative Chain Code Algorithm system is developed using top-down approach where the system is divided into different and smaller modules. This approach assumes that every module are developed and tested properly within its own module. By breaking down into simple and well-defined interface, it will make the system more modular. The reason why modularity is desirable is because a modular system is much easy to code, debug, understand and maintain.

### 6.1 Development Environment

Development environment has a significant influence in the process of developing the system. System development can be paced up significantly by utilizing the appropriate software and hardware.



### 6.1.1 Hardware in the Development Environment

Hardware which is configured for the development environment is the underlying element for the entire system. The hardware used in the implementation phase of the system plays an important role in realizing the finale of the system's architecture.

The hardware configuration of the development environment is as the following:

- Intel Pentium 4 Processor 1.80GHz
- Memory – 256MB PC2100 of DDR RAM
- Storage – 4 GB of Hard Disk space is reserved
- Other standard desktop PC component

### 6.1.2 Software in the Development Environment

Hardware and software are tightly coupled that operates in accord to performed programmed tasks. Without the proper and correct software, the biggest, fattest or even the most powerful computer cannot perform up to expectation rather it is inoperative and idle without any use.

The software tools that are being used in the development environment are as the following:

- Operating System – Microsoft Windows XP Professional Service Pack 1
- Web Browsers – Internet Explorer 6.0, Firefox 1.0
- Matlab 6.5.0.18091 3a Release 13
- Notepad
- Documentation – Microsoft Office XP

## 6.2 Recognition Algorithm

The process of character recognition is divided into 4 modules, which are:

### i. Image processing

In image processing, the image which is kept in the directory will be opened and displayed. The chosen image will go through the threshold and thinning process.

### ii. Segmentation

The segmentation involves segmenting out the wanted numerical character with others that may appear in the image. It will then recognize the main numerical character that will go through the recognition process.

### iii. Recognition

The recognition process involves determining the starting point, reading the chain code and matching the chain code of the image with the one in the knowledge base. It is necessary to determine the starting point of the image as the system needs to know where it should start to read the chain code for that particular image. After the chain code has been read, it will then be kept and compared with the knowledge base that has been built, in order to identify the numerical character.



### 6.3 Image Processing

- Open and display the file which is being stored in the directory

```
[imagefile, imagepath] = uigetfile ('*.*', 'Please Select Your Image');
```

```
[I,map] = imread ([imagepath,imagefile]);
```

```
image (I);
```

- Executing the thresholding process

```
level = graythresh (I);
```

```
threshold = im2bw(I, level);
```

- Executing the thinning process

```
SE = strel ('rectangle',[2 1]);
```

```
thinning = imdilate (threshold, SE);
```

#### 6.4 Segmentation Process

```
[m,n]=size(I);
```

```
bw=I;
```

```
length_array=length(bw);
```

```
array=ones(1,length_array);
```

```
M=cell(length_array,1); %creates a cell array that is the same size array M.
```

```
%-----Insert image (binary=0) into cell array M-----%
```

```
for i=1:m
```

```
total=0;
```

```
for j=1:length_array
```

```
if bw(i,j)==1 %Determine the quantity of j=1
```

```
total=total+1;
```

```
end
```

```
end
```

```
if total==length_array %j=1 is same as length_array
```

```
total=0;
```

```
elseif total~=length_array %j=1 is not the same as length_array
```

```
for a=1:length_array
```

```
array(1,a)=bw(i,a); %Insert the image into the array
```

```
end
```

```
M{i}=array; %Store array into cell array M
```

```
total=0;
```

```
end
```

```
end
```



temp=0;

temp1=0;

temp2=0;

temp3=0;

temp4=0;

temp5=0;

for i=1:m

see=isempty(M{i});

if see==1 %if M{i} do not contain value 0

temp=temp+1;

temp1=temp; %temp1 return rows that do not contain value 0

elseif see==0 %if M{i} contain value 0

temp=temp+1;

temp2=temp; %temp2 return rows that contain value 0

break

end

end

temp3=temp2; %temp3 return rows that contain value 0

array2=ones(1,n);

char1{temp3-1}=array2; %add the default value of array to the highest row

for i=temp3:m

see=isempty(M{i});

```

if see==0

    char1{i}=M{i};

else

    char1{i}=array2;    %add default value of the array into the lowest row

    temp4=temp4+i; %temp4 return rows that contain the same value as 1

    break

end

end

temp5=temp4;

%display char1

y=cell2mat(char1);

figure(1);

imshow(y);

title('Segmented Character');

handles.image_value2=y;

guidata(hObject,handles)

```



## 6.5 Recognition

```
[k,l]=size(y);

%-----Determine initial point for image-----%

length_array=length(y);

for i=1:k

    for j=l:-1:1 %for j=l:-1:1

        if y(i,j)==0

            start1=i;

            start2=j;

            disp('Image=')

            disp(y)

            fprintf(1,'Initial point for the image is at pixel(%d,%d)\n',start1,start2)

            break;

        end

    end

end

if y(i,j)==0

    break;

end

end

prev_code=8;
```

```

%-----Determine the chain code image-----%

store_code=[];           %declare store_code array to store the code

increment=1;

while start1~=0 & start2~=0

    if prev_code==8           %start to read the code

        if y(start1+1,start2-1)==0       %code=5

            start1=start1+1;

            start2=start2-1;

            store_code{increment}=5;

            prev_code=5;

        elseif y(start1+1,start2)==0       %code=6

            start1=start1+1;

            start2=start2;

            store_code{increment}=6;

            prev_code=6;

        elseif y(start1-1,start2)==0       %code=2

            start1=start1-1;

            start2=start2;

            store_code{increment}=2;

            prev_code=2;

```



```

elseif y(start1,start2+1)==0    %code=0

    start1=start1;

    start2=start2+1;

    store_code{increment}=0;

    prev_code=0;

elseif y(start1-1,start2+1)==0    %code=1

    start1=start1-1;

    start2=start2+1;

    store_code{increment}=1;

    prev_code=1;

elseif y(start1-1,start2-1)==0    %code=3

    start1=start1-1;

    start2=start2-1;

    store_code{increment}=3;

    prev_code=3;

elseif y(start1,start2-1)==0    %code=4

    start1=start1;

    start2=start2-1;

    store_code{increment}=4;

    prev_code=4;

elseif y(start1+1,start2+1)==0    %code=7

    start1=start1+1;

    start2=start2+1;

    store_code{increment}=7;

```

```

    prev_code=7;

else

    start1=0;

    start2=0;

end

increment=increment+1;      %move up the position of the array

else      %if prev_code is exist-after defining the previous code

    if prev_code == 5      %if prev_code==5

        if y(start1+1,start2+1)==0 %code=7 -prev_code=5

            start1=start1+1;

            start2=start2+1;

            store_code{increment}=7;

            prev_code=7;

        elseif y(start1+1,start2)==0 %code=6 -prev_code=5

            start1=start1+1;

            start2=start2;

            store_code{increment}=6;

            prev_code=6;

        elseif y(start1+1,start2-1)==0 %code=5 -prev_code=5

            start1=start1+1;

            start2=start2-1;

            store_code{increment}=5;

            prev_code=5;

```



```

elseif y(start1,start2-1)==0 %code=4 -prev_code=5

    start1=start1;

    start2=start2-1;

    store_code{increment}=4;

    prev_code=4;

elseif y(start1-1,start2-1)==0 %code=3 -prev_code=5

    start1=start1-1;

    start2=start2-1;

    store_code{increment}=3;

    prev_code=3;

elseif y(start1-1,start2)==0 %code=2 -prev_code=5

    start1=start1-1;

    start2=start2;

    store_code{increment}=2;

    prev_code=2;

else

    start1=0;

    start2=0;

end

increment=increment+1;

elseif prev_code==6 %if prev_code==6

    if y(start1+1,start2+1)==0 %code=7 -prev_code=6

        start1=start1+1;

        start2=start2+1;

```

```

    store_code{increment}=7;

    prev_code=7;

elseif y(start1+1,start2)==0 %code=6 -prev_code=6

    start1=start1+1;

    start2=start2;

    store_code{increment}=6;

    prev_code=6;

elseif y(start1+1,start2-1)==0 %code=5 -prev_code=6

    start1=start1+1;

    start2=start2-1;

    store_code{increment}=5;

    prev_code=5;

elseif y(start1,start2-1)==0 %code=4 -prev_code=6

    start1=start1;

    start2=start2-1;

    store_code{increment}=4;

    prev_code=4;

elseif y(start1-1,start2-1)==0 %code=3 -prev_code=6

    start1=start1-1;

    start2=start2-1;

    store_code{increment}=3;

    prev_code=3;

```



```

else

    start1=0;

    start2=0;

end

increment=increment+1;

elseif prev_code==2      %if prev_code==2

if y(start1-1,start2-1)==0    %code=3 -prev_code=2

    start1=start1-1;

    start2=start2-1;

    store_code{increment}=3;

    prev_code=3;

elseif y(start1-1,start2)==0    %code=2 -prev_code=2

    start1=start1-1;

    start2=start2;

    store_code{increment}=2;

    prev_code=2;

elseif y(start1-1,start2+1)==0 %code=1 -prev_code=2

    start1=start1-1;

    start2=start2+1;

    store_code{increment}=1;

    prev_code=1;

elseif y(start1,start2+1)==0 %code=0 -prev_code=2

    start1=start1;

    start2=start2+1;

```

```

    store_code{increment}=0;

    prev_code=0;

elseif y(start1+1,start2+1)==0 %code=7 -prev_code=2

    start1=start1+1;

    start2=start2+1;

    store_code{increment}=7;

    prev_code=7;

else

    start1=0;

    start2=0;

end

increment=increment+1;

elseif prev_code==0          %if prev_code==0

if y(start1-1,start2+1)==0    %code=1 -prev_code=0

    start1=start1-1;

    start2=start2+1;

    store_code{increment}=1;

    prev_code=1;

elseif y(start1,start2+1)==0 %code=0 -prev_code=0

    start1=start1;

    start2=start2+1;

    store_code{increment}=0;

    prev_code=0;

elseif y(start1+1,start2+1)==0 %code=7 -prev_code=0

```



```

    start1=start1+1;

    start2=start2+1;

    store_code{increment}=7;

    prev_code=7;

elseif y(start1+1,start2)==0 %code=6 -prev_code=0

    start1=start1+1;

    start2=start2;

    store_code{increment}=6;

    prev_code=6;

elseif y(start1+1,start2-1)==0 %code=5 -prev_code=0

    start1=start1+1;

    start2=start2-1;

    store_code{increment}=5;

    prev_code=5;

else

    start1=0;

    start2=0;

end

increment=increment+1;

elseif prev_code==1 %if prev_code==1

if y(start1-1,start2-1)==0 %code=3 -prev_code=1

    start1=start1-1;

    start2=start2-1;

    store_code{increment}=3;

```

```

    prev_code=3;

elseif y(start1-1,start2)==0 %code=2 -prev_code=1

    start1=start1-1;

    start2=start2;

    store_code{increment}=2;

    prev_code=2;

elseif y(start1-1,start2+1)==0 %code=1 -prev_code=1

    start1=start1-1;

    start2=start2+1;

    store_code{increment}=1;

    prev_code=1;

elseif y(start1,start2+1)==0 %code=0 -prev_code=1

    start1=start1;

    start2=start2+1;

    store_code{increment}=0;

    prev_code=0;

elseif y(start1+1,start2+1)==0 %code=7 -prev_code=1

    start1=start1+1;

    start2=start2+1;

    store_code{increment}=7;

    prev_code=7;

elseif y(start1+1,start2)==0 %code=6 -prev_code=1

    start1=start1+1;

    start2=start2;

```



```

    store_code{increment}=6;

    prev_code=6;

else

    start1=0;

    start2=0;

end

increment=increment+1;

elseif prev_code==3      %if prev_code==3

if y(start1+1,start2-1)==0    %code=5 -prev_code=3

    start1=start1+1;

    start2=start2-1;

    store_code{increment}=5;

    prev_code=5;

elseif y(start1,start2-1)==0    %code=4 -prev_code=3

    start1=start1;

    start2=start2-1;

    store_code{increment}=4;

    prev_code=4;

elseif y(start1-1,start2-1)==0    %code=3 -prev_code=3

    start1=start1-1;

    start2=start2-1;

    store_code{increment}=3;

    prev_code=3;

elseif y(start1-1,start2)==0    %code=2 -prev_code=3

```

```

    start1=start1-1;

    start2=start2;

    store_code{increment}=2;

    prev_code=2;

elseif y(start1-1,start2+1)==0 %code=1 -prev_code=3

    start1=start1-1;

    start2=start2+1;

    store_code{increment}=1;

    prev_code=1;

elseif y(start1,start2+1)==0 %code=0 -prev_code=3

    start1=start1;

    start2=start2+1;

    store_code{increment}=0;

    prev_code=0;

else

    start1=0;

    start2=0;

end

increment=increment+1;

elseif prev_code==4 %if prev_code==4

if y(start1+1,start2-1)==0 %code=5 -prev_code=4

    start1=start1+1;

    start2=start2-1;

    store_code{increment}=5;

```



```

    prev_code=5;

elseif y(start1,start2-1)==0 %code=4 -prev_code=4

    start1=start1;

    start2=start2-1;

    store_code{increment}=4;

    prev_code=4;

elseif y(start1-1,start2-1)==0 %code=3 -prev_code=4

    start1=start1-1;

    start2=start2-1;

    store_code{increment}=3;

    prev_code=3;

elseif y(start1-1,start2)==0 %code=2 -prev_code=4

    start1=start1-1;

    start2=start2;

    store_code{increment}=2;

    prev_code=2;

elseif y(start1-1,start2+1)==0 %code=1 -prev_code=4

    start1=start1-1;

    start2=start2+1;

    store_code{increment}=1;

    prev_code=1;

else

    start1=0;

    start2=0;

```

```

end

increment=increment+1;

elseif prev_code==7      %if prev_code==7

if y(start1-1,start2+1)==0  %code=1 -prev_code=7

    start1=start1-1;

    start2=start2+1;

    store_code{increment}=1;

    prev_code=1;

elseif y(start1,start2+1)==0 %code=0 -prev_code=7

    start1=start1;

    start2=start2+1;

    store_code{increment}=0;

    prev_code=0;

elseif y(start1+1,start2+1)==0 %code=7 -prev_code=7

    start1=start1+1;

    start2=start2+1;

    store_code{increment}=7;

    prev_code=7;

elseif y(start1+1,start2)==0 %code=6 -prev_code=7

    start1=start1+1;

    start2=start2;

    store_code{increment}=6;

    prev_code=6;

elseif y(start1+1,start2-1)==0 %code=5 -prev_code=7

```



```

    start1=start1+1;

    start2=start2-1;

    store_code{increment}=5;

    prev_code=5;

elseif y(start1,start2-1)==0 %code=4 -prev_code=7

    start1=start1;

    start2=start2-1;

    store_code{increment}=4;

    prev_code=4;

else

    start1=0;

    start2=0;

end

increment=increment+1;

end %end for prev_code=5

end %end for prev_code=8

end %end for while loop

S=cell2mat(store_code);

disp('Chain Code that has been read is:');

disp(S)

handles.image_value=S;

guidata(hObject,handles)

```

## Chapter 7: System Testing

### 7.1 Introduction

System testing is an important phase in the development of the Unravel Parking Difficulty via Relative Chain Code Algorithm system. This is because system testing tells whether the coding has been successfully implemented, all the functions is working accordingly, and whether the code need to be modified, enhanced or debugged to produce the correct result from the system.

Furthermore, testing is the process of establishing the existence of errors. A good test is one that has a high probability of finding an undiscovered error. Nevertheless, testing cannot prove the absence of defects, as it can only show the software defects that are present. Therefore, an appropriate approach must be chosen to reduce the possibility of errors on the program.

The testing process for the system is divided into a few modules that involve the execution of a series of functions. This is to check whether the system will perform accordingly through execution and implementation of the functions.



### 7.1.1 Compilation and Execution

Once the coding for the system is completed, the MATLAB source code needs to be compiled to check whether there are any bugs or error in the coding.

If the MATLAB application can be compiled successfully without error, then the testing phase can proceed with the execution of the application. Otherwise, the testing phase will have to go through the debugging phase to fix the errors, before it can be recompiled again.

### 7.1.2 Debugging

During the compilation of the application, if error(s) are found, the error message(s) need to be scrutinized to identify where the errors have occurred in the source code. The errors might be syntax error, such as missing semi-colons, curly braces and undefined function. Some other errors might be logical errors, such as errors in referencing, errors in passing arguments and errors in calling methods.

The process of debugging is to check the occurred errors and correcting them. It is a process of eliminating errors or bugs from the source code in order for the system to be compiled successfully. This is an iterative process, where it is very common in the programming environment. Only a successful compiled piece of source code is able to proceed to the execution process in the testing phase.

### 7.1.3 Accuracy of Execution

When the source code is freed from all errors and bugs, it will then be executed or in other words run. The aim of execution is to allow users to use the system or to interact with the system through the system's interface.

For the Unravel Parking Difficulty via Relative Chain Code Algorithm system, the compiled source code is executed to check and verify the correctness and accuracy of the interface, the response of the system when user click on one of the buttons, whether the system produced accurate relative chain code for the specific number when it is input into the system.

If the system does not generate the correct chain code according to the algorithm or response to user's click, the testing phase will go back to the debugging-compiling-executing process until the system is able to produce accurate chain code according to the chain code algorithm.

If the functions does not perform as expected or produce good result, the code will be modified and debugged in order to come out with better results. Some other modifications other than the ones mentioned above, that can be done here would include modifying the appearance of the interface such as colors of the background and also the setting of layout and panel.



## 7.2 Image Processing Module

Image pre-processing processes include a few functions that are being used to produce a more quality image. One of the advantages of using MATLAB is because it has a built-in Image Processing Toolbox. The Image Processing Toolbox is a collection of functions that extend the capability of the MATLAB numeric computing environment. The toolbox supports a wide range of image processing operations. Some of the functions that are being used are discussed in the following sections:

### 7.2.1 Image Digitization

- `[imagefile, imagepath] = uigetfile ('*. *', 'Please Select Your Image')`

It returns the name and path of the file that has been chosen from the dialog box. When the open image button is being pressed, *imagefile* will contain the name of the file that has been chosen and *imagepath* will have the name of the path that has been chosen. If the cancel button is being pressed or if an error occurs, *imagefile* and *imagepath* are set to 0.

*uigetfile* displays a dialog box used to retrieve a file. The dialog box lists the files and directories in the current directory.

- $[I, \text{map}] = \text{imread}([\text{imagepath}, \text{imagefile}])$

Reads the indexed image in *imagepath* into *I* and its associated colormap into *map*. *I* is of class `uint8`, and *map* is of class `double`. The colormap values are rescaled to the range  $[0,1]$ . *imagepath* is a string that specifies the format of the file. The file must be in the current directory or in a directory in the MATLAB path. If *imread* cannot find the *imagepath* file, it looks for a file named *imagepath.fmt*.

- $\text{delete}(\text{handles.figure})$

Delete the current figure.

- $\text{set}(\text{handles.figure}, 'Visible', 'off')$

Make the figure invisible.

- $\text{set}(\text{handles.figure}, 'WindowStyle', 'normal')$

Change the figure's *WindowStyle* property to normal.

### 7.2.2 Threshold

- $\text{level} = \text{graythresh}(I)$

Computes a global threshold (*level*) that is used to convert an intensity image into binary image with *im2bw*. *level* is a normalized intensity value that lies in the range of  $[0,1]$ .



- $threshold = im2bw(I, level)$

Converts the intensity image  $I$  to black and white.  $im2bw$  produces binary image from indexed, intensity, or RGB images. It converts the input image to grayscale format (if it is not already an intensity image), and then converts this grayscale image to binary by thresholding.  $level$  is in the range  $[0,1]$ .

### 7.2.3 Thinning

Thinning is an operation where the value of the output pixel is the maximum value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1.

- $SE = strel('rectangle',[2 \ 1])$

Creates a structuring element,  $SE$ , of the type specified by shape.  $strel$  function will execute according the shape and parameters that has been chosen. In this testing,  $SE$  creates a flat, rectangle-shaped structuring element, where  $[2 \ 1]$  is the size of the rectangle. The size must be a two-element vector of nonnegative integers. The first element which is 2 signifies there are two rows in the structuring element neighborhood; the second element which is 1 signifies there is one column.

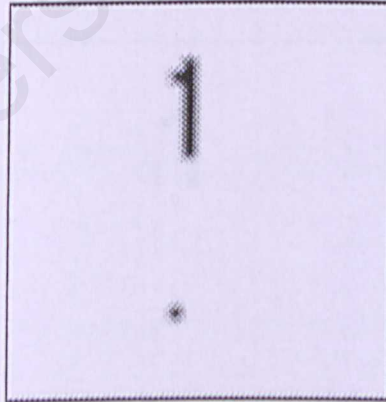
- *thinning = imdilate (threshold, SE)*

This function dilates the grayscale, binary, or packed binary image threshold, returning the dilated image, thinning. The argument *SE* is a structuring element object, or array of structuring element objects, returned by the *strel* function.

The input for this module is from the threshold module where the pixel value is 0 or 1, thus if the pixel equals to 1, the output for that particular pixel will be set to 1 also.

### 7.3. Segmentation

Segmentation process is the process of separating the image that is wanted for recognition with other unwanted image that may appear in the captured image. The numerical character that has been chosen to be tested is number 1. The image of number 1 that will be tested in the segmentation module is as shown in Figure 7.1.



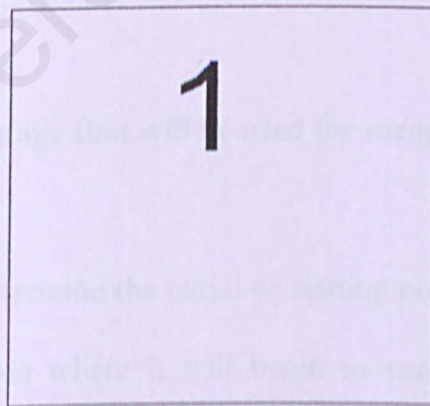
**Figure 7.1: Image of Number One**



The segmentation process begins when the first row of the image that has been detected with zero in at least of the pixel. When that particular row has been identified, the entire pixel's value of that particular row will be stored into cell and continue with the next row, and so on, until the segmentation module can find the first row where all the pixel's value equal to one. Image that is being stored in cell represent the image that will be recognized in the recognition module.

Rows that have all pixels' value equal to one suggest that there is empty space in the image. All this pixels' value will be ignored as it only contains image that is useless in the image recognition process, as the segmentation module will only focus on the image that is required. The segmentation module has been tested successfully as it is able to separate out an image that is sought after with other unnecessary extra image.

After the segmentation process only the important image will remain and the other unnecessary images will be discarded for easier recognition in the following testing process.

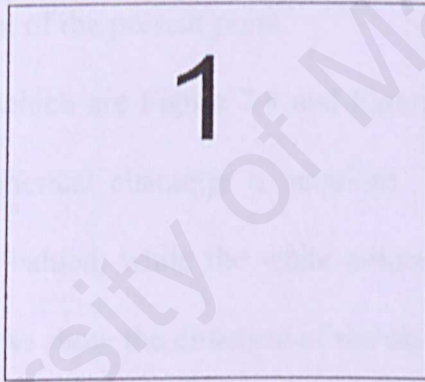


**Figure 7.2: Image after Segmentation**

#### 7.4 Recognition

In the recognition process, it will focus on testing the chain code algorithm. The main reason of the testing is to check whether the chain code is working properly, according to the chain code algorithm and produce correct result while it is being tested on images. The generated chain code will be used as representations of a particular image and in this case the numerical characters.

The chain code will be read in an anti-clockwise manner and this causes each pixel will have eight possibility in every generated chain code. The images that have been segmented will be used for recognition purposes as shown in Figure 7.3.



**Figure 7.3: Image that will be used for recognition process**

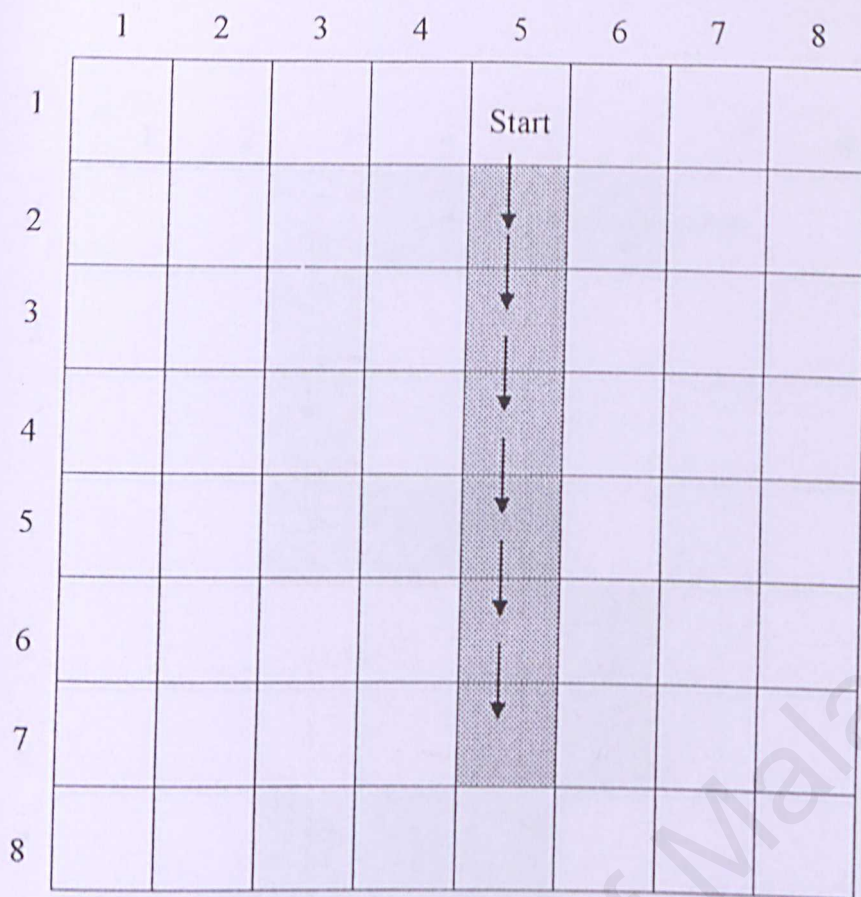
It is a very crucial to determine the initial or starting point of the image that will be recognized, as it will decide where it will begin to read the chain code of the particular image. The numerical character is being read in matrix  $(i \ j)$  where  $i$  represent rows and  $j$  represent columns of the image. The initial point will be determined from the right to left and from top to bottom.



When the recognition process has been activated, the algorithm will start searching for the first pixel that is zero valued in the rows and columns of  $(i, j)$  and return the starting point of the image. The algorithm will stop the searching process once the first zero valued pixel is found. After the preliminary point of the image has been confirmed, it will then start generate the chain code.

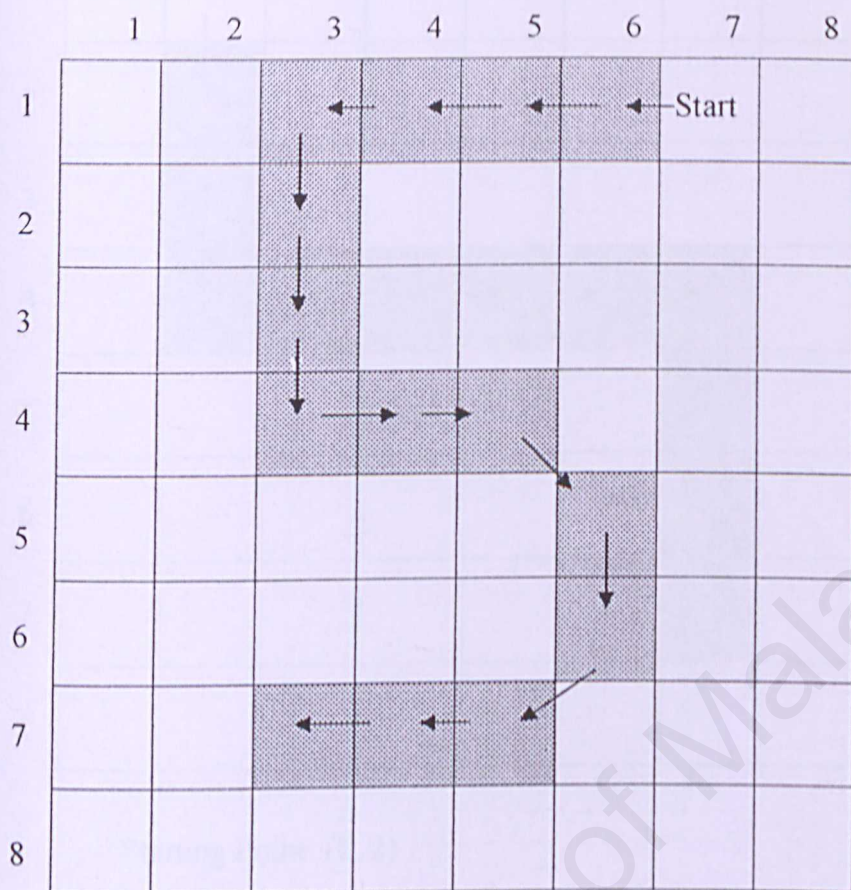
As it has been stated, the chain code is being generated based upon the initial point that has been determined and the direction of the subsequent point. The chain code algorithm will start with the initial point and continue with the next point by examining all possible movement that consists of the value of the 8-connected neighborhood. The value of the point will change every time there is a new point being discovered during the movement of the present point.

The following figures, which are Figure 7.4 and Figure 7.5, show that how the chain code for a particular numerical character is acquired. The gray colored boxes represent pixels that is of zero valued, while the white colored boxes represent pixels which is equal to one. The arrows show the direction of the chain code of the image that is being read.



**Figure 7.4: Image of numerical character (number one) that is use to generate chain code.**

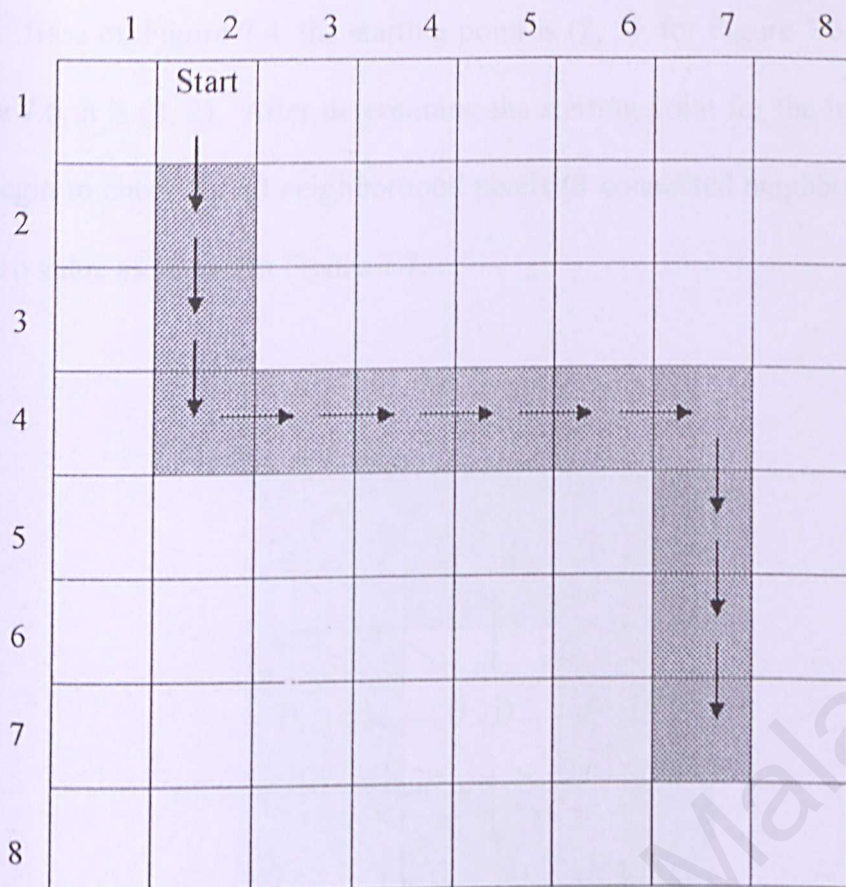




Starting Point: (1, 6)

Chain Code: 4, 4, 4, 4, 6, 6, 6, 0, 0, 7, 6, 5, 4, 4

**Figure 7.5: Image of numerical character (number five) that is use to generate chain code.**



Starting Point: (2, 2)

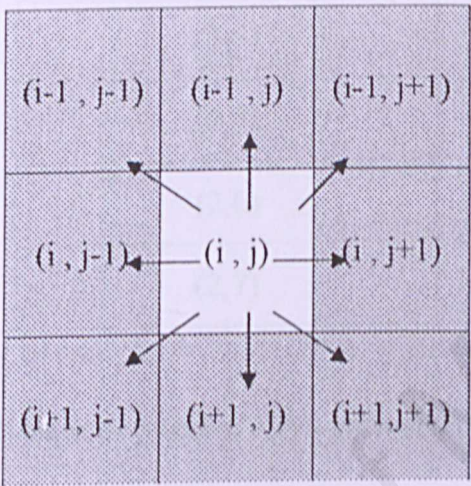
Chain Code: 6, 6, 6, 0, 0, 0, 0, 0, 6, 6, 6

**Figure 7.6: Image of path for the robot navigation to search the position of the vacant parking space that is use to generate chain code.**

According to Figure 7.6, the system is able to recognize the path that leads to the vacant car park; it will help the robot navigation to determine the direction and distance for the robot to calculate the position of the empty space. This will help in solving the problem of where is the empty space as the robot navigation does not have the function to determine the position of the car park space. The number of the chain code will show the direction while how many times the number appears will determine the distance of it.



Base on Figure 7.4, the starting point is (2, 5); for Figure 7.5, it is (1, 6) and for Figure 7.6, it is (2, 2). After determining the starting point for the image, the algorithm will begin to check for all neighborhood pixels (8-connected neighborhood) that contain the zero value as shown in Figure 7.7.



**Figure 7.7: Possible pixel's value for the 8-connected neighborhood pixel**

The table below, Table 7.1, displays all possible subsequent point of the neighborhood pixel based on the initial point of (1, 6) and also the representation for the directions of the chain code.

Initial Point ( $i, j$ )	Subsequent Point ( $i, j$ )	Chain Code Direction
(1,6)	(1,7)	0 →
	(0,7)	1 ↗
	(0,6)	2 ↑
	(0,5)	3 ↖
	(1,5)	4 ←
	(2,5)	5 ↓
	(2,6)	6 ↘
	(2,7)	7 →

**Table 7.1: Possible subsequent point of the 8-connected neighborhood pixel based on the initial point of (1, 6) and also the representation for the directions of the chain code.**

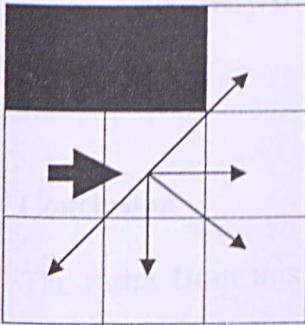
However, there is a constraint for the algorithm as it can only read certain direction of the chain code only and this is based on the starting point of the image. There are some codes that cannot be read and generated to become a chain code. The constraint is being discussed in detail in Table 7.2 and Figure 7.9.



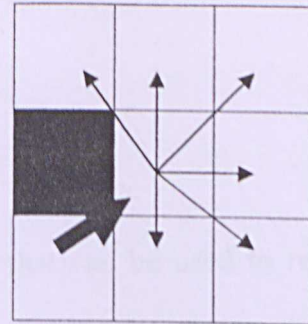
Previous Point	Next Point (Anti-clockwise)
0	5,6,7,0,1
1	6,7,0,1,2,3
2	7,0,1,2,3
3	0,1,2,3,4,5
4	1,2,3,4,5
5	2,3,4,5,6,7
6	3,4,5,6,7
7	4,5,6,7,0,1

**Table 7.2: Possible subsequent point base on the previous code that has been read.**

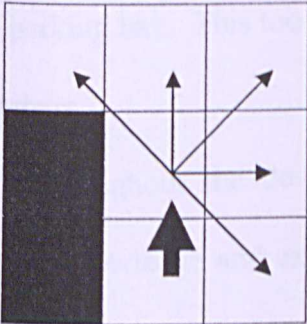
Figure 7.9 shows the black colored boxes that are impossible to contain an image. The arrow illustrates the possible subsequent point in an anti-clockwise manner as in Table 7.1.



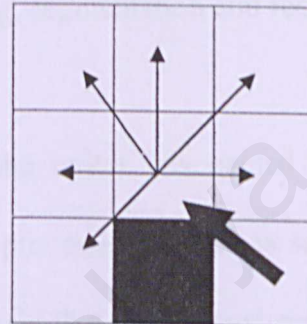
Previous direction = 0



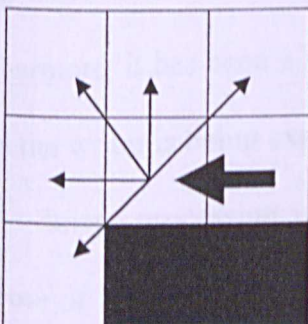
Previous direction = 1



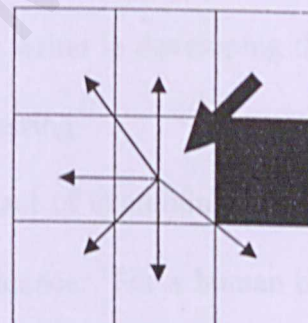
Previous direction = 2



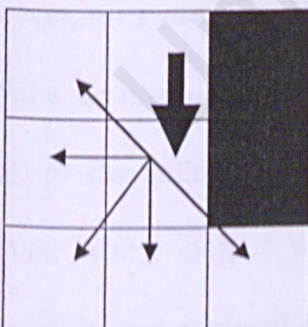
Previous direction = 3



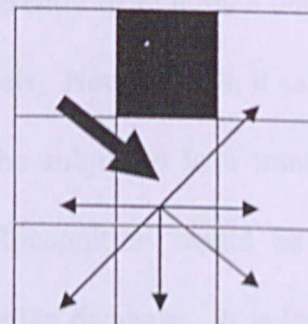
Previous direction = 4



Previous direction = 5



Previous direction = 6



Previous direction = 7

**Figure 7.8: Pixels which in black boxes and directions of chain code that is impossible to be recognized.**



## Chapter 8: Conclusion & Discussion

### 8.1 Conclusion

The result from this final year project is a tool that can be used to recognize character in the parking lot, in order to help car park users to quickly locate vacant space in the parking bay. This tool consists of image processing, segmentation and recognition of numbers.

Throughout the development of this project, the writer has gained a lot of priceless knowledge and exposure in regards of image processing, which is something very new and unfamiliar to me. Image processing is a field that is very vast and there is much more that can be explore as it is still quite new in the computing environment. Furthermore, it has been a wonderful experience for the writer in developing the project since the writer is being exposed to the idea image processing.

Image processing which can be defined as the act of examining images for the purpose of identifying objects and judging their significance. For a human being with normal vision, it is easy to recognize and to interpret the image that they have seen. However, for a machine, it recognizes image quite differently from human beings. To a machine, an image is just an unrelated collection of pixels. Nevertheless, it can store the image as the collection of pixels, or the image can be subjected to a transformation function whose output would be stored instead. Recognition would be based on comparison and retrieval from stored data in a knowledge database. It is important to have the right tool that is able to detect, identify, classify, measure, and evaluate the significance of physical and cultural objects, their patterns and spatial relationship.

## **8.2 Problems Encountered**

The success of the Unravel Parking Difficulty via Relative Chain Code Algorithm system does not come without problems and constraints. Much effort has been placed into understanding, eradicating, and solving problems through the whole course of the system development.

### **8.2.1 Inexperience in MATLAB**

Though the writer had chosen MATLAB as the project's development tool, the writer does not have any knowledge or experience in using the software. Therefore the writer started to learn the language and its usage whenever time permits and during the semester break. Through this, it has given the writer some concept and knowledge about MATLAB. The writer also downloaded some electronic books for off-line reference as the library is lack of reference books that is related to MATLAB. Internet is indeed a good source for faster learning process as there are many up-to-date resources available for any types of programming language.

### **8.2.2 Understanding the Concept of Image Processing**

As image processing is something new to the writer, the writer need to study the fundamental concepts of it and try to understand the processes it involves in order for the machine to understand the image itself. Since there is many methods involves in image processing, the writer also need to choose the methods that are suitable for the system in turn to attain the objectives of the system. In addition, the writer also need to learn about the Relative Chain Code algorithm that includes how it works and why it is better



than the ordinary chain code algorithm. It took the writer some time to understand it and be able to work out the relative chain code for the numbers.

### 8.2.3 Tight Schedule

The project time frame is very short and tight since all the development and documentation need to be finished in less than five months. As the writer still needs to attend lectures and to do assignments, the time allocated for the development process is very short. For this reason, the writer had planned a development schedule for each module. The writer also tried to jot down inspiration for the project from time to time and write comment blocks within the coding which help the writer to continue with the coding without wasting time to recall and review the code again.

### 8.3 System's Strengths

Some of the strong points of the Unravel Parking Difficulty via Chain Code Algorithm system's are as the following:

- It is a tool that is specifically for image processing in the Unravel Parking Difficulty via Relative Chain Code Algorithm system that includes noise reduction, improving the quality of the image by adjusting the brightness of the image as the captured may be not bright enough to be recognized, thresholding process to highlight the regions of the image that are needed for the recognition process and thinning by reducing the all lines into single pixel thickness for easier recognition.

- This system is able to segment the character that is needed to be recognized without regard where the character is located. The image can be located in the middle, left or right but the system is able to segment the character that we are interested to perform the recognition process.
- This system also has the advantage of determining the starting point of the image that will then go through the recognition process. It is an important process as the starting point of the image will be used to implement the chain code algorithm. The chain code algorithm that has been built is being used to read the code of the intended image in order for the recognition process to be successfully performed.
- In addition, this system is equipped with the run length encoding feature that is being use the compress the generated chain code while recognizing the intended image. In this feature, it is able to increase performance of the system as shorter code will be used for recognition and to store the interpreted chain code.
- Lastly, not only the system is capable of recognizing the character from the image, it is also able recognize the route that is needed for the robot to know the path to reach the vacant car park space. This is an added feature that is very useful and beneficial in the shortest path detection system to transform the captured image into relative chain code.



## 8.4 System's Constraints

Although the Unravel Parking Difficulty via Chain Code Algorithm system can be categorized, overall, as successful, it still has some weak points / limitations which are:

- The sending parameter from the image processing module is not complete that causes the image that has gone through earlier processes cannot be used for the thinning process. The reason is that the image from previous processes is not in a one pixel mode and may produce none wished-for result in the recognition stage.
- Furthermore, the noise reduction, brightness adjustment, and thresholding functions can only prove the flow of the entire system but it cannot be send to be used for the following of the functions.
- The recognition process is intended to identify the images that are in one pixel mode that represents one straight line. The chain code generated from none one pixel image will issue in incorrect code or even worse the chain code recognition process cannot be carried out successfully.
- Testing has only been able to perform in some numerical characters only. Hence the testing does not cover all numbers where some of the numbers may not be able to be recognized by the system.

## 8.5 Future Enhancements

As this system still comes with a number of limitations, there are still a lot of areas that allows for further enhancement. One the area is to make sure that in the future, the images that has gone through earlier image processing processes can be in use in the thinning process in order to produce better result.

Besides that, the chain code algorithm that is being used in the recognition process should be fault tolerance where it can still recognize image that is not in a single pixel mode. The chain code that has been generated will be read in detail to determine whether it really represent the actual character.

The writer's suggestion is that this project should be continued in the future with the purpose of improving the system to repair the existing weakness and add in more new functions so as to produce an operative, efficient and reliable system for the usage of car park operators and users. Research concerning this system ought to be carried out to create a system that can perform more effectively.

Last but not least, the completed system is just a beginning to build a more sophisticated system that can really help car park users to find vacant car park space with ease, consequently reducing the traffic in the car park and encourage visitors to spend more time in shopping complexes. It is very important for functions such as thinning process, segmentation process and recognition process to be implemented successfully as these processes are considered as core process for the system. Again, all these are just brilliant ideas, there are much more effort, research and development to be funded and carried out in reaching the goal of the Unravel Parking Difficulty via Chain Code Algorithm.



## Appendix A: Source Code

```
%-----carpark_main.m-----%
%Faculty of Computer Science and Information Technology
%University of Malaya (UM), Kuala Lumpur
%WXES 3182 - Final Year Project
%Unravel Parking Difficulty via Relative Chain Code Algorithm
%Student Name: CHAN HUEY MIN
%Matric No: WEK 020027
%
%Supervisor: Mr. Mohd Yamani Idna bin Idris
%-----%
function varargout = carpark_main(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @carpark_main_OpeningFcn, ...
    'gui_OutputFcn', @carpark_main_OutputFcn, ...
    'gui_LayoutFcn', [], ...
    'gui_Callback', []);
if nargin & isstr(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 carpark_main is made visible.
function carpark_main_OpeningFcn(hObject, eventdata, handles, varargin)

% Determine the position of the dialog - centered on the callback figure
% if available, else, centered on the screen
FigPos=get(0,'DefaultFigurePosition');
OldUnits = get(hObject, 'Units');
set(hObject, 'Units', 'pixels');
OldPos = get(hObject,'Position');
FigWidth = OldPos(3);
FigHeight = OldPos(4);
if isempty(gcbf)
    ScreenUnits=get(0,'Units');
    set(0,'Units','pixels');
    ScreenSize=get(0,'ScreenSize');
    set(0,'Units',ScreenUnits);

    FigPos(1)=1/2*(ScreenSize(3)-FigWidth);
    FigPos(2)=2/3*(ScreenSize(4)-FigHeight);
else
    GCBFOldUnits = get(gcbf,'Units');
```

```

set(gcbf,'Units','pixels');
GCBFPos = get(gcbf,'Position');
set(gcbf,'Units',GCBFOLDUnits);
FigPos(1:2) = [(GCBFPos(1) + GCBFPos(3) / 2) - FigWidth / 2, ...
              (GCBFPos(2) + GCBFPos(4) / 2) - FigHeight / 2];
end
FigPos(3:4)=[FigWidth FigHeight];
set(hObject, 'Position', FigPos);
set(hObject, 'Units', OldUnits);

% Choose default command line output for carpark_main
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = carpark_main_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

%OPEN IMAGE
% --- Executes on button press in open_image_button.
function open_image_button_Callback(hObject, eventdata, handles)
[imagefile,imagepath]=uigetfile('*.*','Please Select Your Image'); %----set the file, then the file's
path. Get the file
[l,map]=imread([imagepath,imagefile]);%----Input the file and display the path using imread.
Locate the path, then the file
image(l);
handles.image_value=l;
guidata(hObject,handles)

%THINNING
% --- Executes on button press in thinning_button.
function thinning_button_Callback(hObject, eventdata, handles)
BW=handles.image_value;
SE=strel('rectangle',[2 1]); %rectangle',[2 1]
N=imdilate(BW,SE);
imshow(N),title('Dilation Image')
handles.image_value=N;
guidata(hObject,handles)

%THRESHOLDING
% --- Executes on button press in threshold_button.
function threshold_button_Callback(hObject, eventdata, handles)
l=handles.image_value;
l=rgb2gray(l);
level=graythresh(l);
BW=im2bw(l,level);
imshow(BW),title('Thresholded Image') %title(['Thresholding', num2str(level)])

handles.image_value=BW;
guidata(hObject,handles)

```



```

%SEGMENTATION
% --- Executes on button press in segmentation_button.
function segmentation_button_Callback(hObject, eventdata, handles)

l=handles.image_value;
[m,n]=size(l);

bw=l;
length_array=length(bw);
array=ones(1,length_array);
M=cell(length_array,1); %creates a cell array that is the same size as another array, M.
char1=cell(length_array,1);

%-----Insert image(binary=0) into cell array M-----%
for i=1:m
    total=0;

    for j=1:length_array

        if bw(i,j)==1 %j=1
            total=total+1;
        end
    end

    if total==length_array %j=1 is same as length_array
        total=0;

    elseif total~=length_array %j=1 is not the same as length_array

        for a=1:length_array
            array(1,a)=bw(i,a); %Insert the image into array
        end

        M{i}=array; %Keep array into cell array M
        total=0;

    end
end

%-----Segmentation Process-----%
temp=0;
temp1=0;
temp2=0;
temp3=0;
temp4=0;
temp5=0;

for i=1:m
    see isempty(M{i});

    if see==1 %if M{i} do not contain value 0
        temp=temp+1;
        temp1=temp; %temp1 return rows that do not contain value 0

    elseif see==0 %if M{i} contain value 0
        temp=temp+1;
    end
end

```

```

    temp2=temp; %temp2 return rows that contain value 0
    break
end
end

temp3=temp2; %temp3 return rows that contain value 0
array2=ones(1,n);
char1{temp3-1}=array2; %add in the default value for array all ones into the topmost row
for i=temp3:m
    see isempty(M{i});
    if see==0
        char1{i}=M{i};
    else
        char1{i}=array2; %add in the default value for array all ones into the most bottom row
        temp4=temp4+i; %temp4 return all rows that contain the same value as 1
        break
    end
end
temp5=temp4;

%display char1
y=cell2mat(char1);
figure(1);
imshow(y);
title('Segmented Character');

handles.image_value2=y;
guidata(hObject,handles)

%-----CHAIN CODE-----%
% --- Executes on button press in chain_code_button.
function chain_code_button_Callback(hObject, eventdata, handles)
y=handles.image_value2;
[k,l]=size(y);

%-----Determine initial point for image-----%
length_array=length(y);
for i=1:k
    for j=l:-1:1 % for j=l:-1:1
        if y(i,j)==0
            start1=i;
            start2=j;
            %disp(y)
            %fprintf(1,'Starting point for the image is at pixel(%.1d,%.1d)\n',start1,start2)
            break;
        end
    end
    if y(i,j)==0
        break;
    end
end
prev_code=8;

```



```

%-----Determine the chain code image-----%
store_code=[]; %declare store_code array to store the code
increment=1;

while start1~=0 & start2~=0
    if prev_code==8 %-----start to read the code,prev_code is not found for point that is
        firstly found
            if y(start1+1,start2-1)==0 %code=5
                start1=start1+1;
                start2=start2-1;
                store_code{increment}=5;
                prev_code=5;
            elseif y(start1+1,start2)==0 %code=6
                start1=start1+1;
                start2=start2;
                store_code{increment}=6;
                prev_code=6;
            elseif y(start1-1,start2)==0 %code=2
                start1=start1-1;
                start2=start2;
                store_code{increment}=2;
                prev_code=2;
            elseif y(start1,start2+1)==0 %code=0
                start1=start1;
                start2=start2+1;
                store_code{increment}=0;
                prev_code=0;
            elseif y(start1-1,start2+1)==0 %code=1
                start1=start1-1;
                start2=start2+1;
                store_code{increment}=1;
                prev_code=1;
            elseif y(start1-1,start2-1)==0 %code=3
                start1=start1-1;
                start2=start2-1;
                store_code{increment}=3;
                prev_code=3;
            elseif y(start1,start2-1)==0 %code=4
                start1=start1;
                start2=start2-1;
                store_code{increment}=4;
                prev_code=4;
            elseif y(start1+1,start2+1)==0 %code=7
                start1=start1+1;
                start2=start2+1;
                store_code{increment}=7;
                prev_code=7;
            else
                start1=0;
                start2=0;
            end
            increment=increment+1; %naikkan kedudukan array
        else %-----if prev_code is exist-after defining the previous code
            if prev_code == 5 %-----if prev_code==5
                if y(start1+1,start2+1)==0 %----code=7 -prev_code=5

```

```

    start1=start1+1;
    start2=start2+1;
    store_code{increment}=7;
    prev_code=7;
elseif y(start1+1,start2)==0 %----code=6 -prev_code=5
    start1=start1+1;
    start2=start2;
    store_code{increment}=6;
    prev_code=6;
elseif y(start1+1,start2-1)==0 %----code=5 -prev_code=5
    start1=start1+1;
    start2=start2-1;
    store_code{increment}=5;
    prev_code=5;
elseif y(start1,start2-1)==0 %----code=4 -prev_code=5
    start1=start1;
    start2=start2-1;
    store_code{increment}=4;
    prev_code=4;
elseif y(start1-1,start2-1)==0 %----code=3 -prev_code=5
    start1=start1-1;
    start2=start2-1;
    store_code{increment}=3;
    prev_code=3;
elseif y(start1-1,start2)==0 %----code=2 -prev_code=5
    start1=start1-1;
    start2=start2;
    store_code{increment}=2;
    prev_code=2;
else
    start1=0;
    start2=0;
end
increment=increment+1;

elseif prev_code==6 %-----if prev_code==6
if y(start1+1,start2+1)==0 %----code=7 -prev_code=6
    start1=start1+1;
    start2=start2+1;
    store_code{increment}=7;
    prev_code=7;
elseif y(start1+1,start2)==0 %----code=6 -prev_code=6
    start1=start1+1;
    start2=start2;
    store_code{increment}=6;
    prev_code=6;
elseif y(start1+1,start2-1)==0 %----code=5 -prev_code=6
    start1=start1+1;
    start2=start2-1;
    store_code{increment}=5;
    prev_code=5;
elseif y(start1,start2-1)==0 %----code=4 -prev_code=6
    start1=start1;
    start2=start2-1;
    store_code{increment}=4;
    prev_code=4;

```



```

elseif y(start1-1,start2-1)==0 %----code=3 -prev_code=6
    start1=start1-1;
    start2=start2-1;
    store_code{increment}=3;
    prev_code=3;
else
    start1=0;
    start2=0;
end
increment=increment+1;

elseif prev_code==2 %-----if prev_code==2
if y(start1-1,start2-1)==0 %----code=3 -prev_code=2
    start1=start1-1;
    start2=start2-1;
    store_code{increment}=3;
    prev_code=3;
elseif y(start1-1,start2)==0 %----code=2 -prev_code=2
    start1=start1-1;
    start2=start2;
    store_code{increment}=2;
    prev_code=2;
elseif y(start1-1,start2+1)==0 %----code=1 -prev_code=2
    start1=start1-1;
    start2=start2+1;
    store_code{increment}=1;
    prev_code=1;
elseif y(start1,start2+1)==0 %----code=0 -prev_code=2
    start1=start1;
    start2=start2+1;
    store_code{increment}=0;
    prev_code=0;
elseif y(start1+1,start2+1)==0 %----code=7 -prev_code=2
    start1=start1+1;
    start2=start2+1;
    store_code{increment}=7;
    prev_code=7;
else
    start1=0;
    start2=0;
end
increment=increment+1;

elseif prev_code==0 %-----if prev_code==0
if y(start1-1,start2+1)==0 %----code=1 -prev_code=0
    start1=start1-1;
    start2=start2+1;
    store_code{increment}=1;
    prev_code=1;
elseif y(start1,start2+1)==0 %----code=0 -prev_code=0
    start1=start1;
    start2=start2+1;
    store_code{increment}=0;
    prev_code=0;
elseif y(start1+1,start2+1)==0 %----code=7 -prev_code=0
    start1=start1+1;

```

```

    start2=start2+1;
    store_code{increment}=7;
    prev_code=7;
elseif y(start1+1,start2)==0 %----code=6 -prev_code=0
    start1=start1+1;
    start2=start2;
    store_code{increment}=6;
    prev_code=6;
elseif y(start1+1,start2-1)==0 %----code=5 -prev_code=0
    start1=start1+1;
    start2=start2-1;
    store_code{increment}=5;
    prev_code=5;
else
    start1=0;
    start2=0;
end
increment=increment+1;

elseif prev_code==1 %-----if prev_code==1
if y(start1-1,start2-1)==0 %----code=3 -prev_code=1
    start1=start1-1;
    start2=start2-1;
    store_code{increment}=3;
    prev_code=3;
elseif y(start1-1,start2)==0 %----code=2 -prev_code=1
    start1=start1-1;
    start2=start2;
    store_code{increment}=2;
    prev_code=2;
elseif y(start1-1,start2+1)==0 %----code=1 -prev_code=1
    start1=start1-1;
    start2=start2+1;
    store_code{increment}=1;
    prev_code=1;
elseif y(start1,start2+1)==0 %----code=0 -prev_code=1
    start1=start1;
    start2=start2+1;
    store_code{increment}=0;
    prev_code=0;
elseif y(start1+1,start2+1)==0 %----code=7 -prev_code=1
    start1=start1+1;
    start2=start2+1;
    store_code{increment}=7;
    prev_code=7;
elseif y(start1+1,start2)==0 %----code=6 -prev_code=1
    start1=start1+1;
    start2=start2;
    store_code{increment}=6;
    prev_code=6;
else
    start1=0;
    start2=0;
end
increment=increment+1;

```



```

elseif prev_code==3 %-----if prev_code==3
if y(start1+1,start2-1)==0 %----code=5 -prev_code=3
    start1=start1+1;
    start2=start2-1;
    store_code{increment}=5;
    prev_code=5;
elseif y(start1,start2-1)==0 %----code=4 -prev_code=3
    start1=start1;
    start2=start2-1;
    store_code{increment}=4;
    prev_code=4;
elseif y(start1-1,start2-1)==0 %----code=3 -prev_code=3
    start1=start1-1;
    start2=start2-1;
    store_code{increment}=3;
    prev_code=3;
elseif y(start1-1,start2)==0 %----code=2 -prev_code=3
    start1=start1-1;
    start2=start2;
    store_code{increment}=2;
    prev_code=2;
elseif y(start1-1,start2+1)==0 %----code=1 -prev_code=3
    start1=start1-1;
    start2=start2+1;
    store_code{increment}=1;
    prev_code=1;
elseif y(start1,start2+1)==0 %----code=0 -prev_code=3
    start1=start1;
    start2=start2+1;
    store_code{increment}=0;
    prev_code=0;
else
    start1=0;
    start2=0;
end
increment=increment+1;

elseif prev_code==4 %-----if prev_code==4
if y(start1+1,start2-1)==0 %----code=5 -prev_code=4
    start1=start1+1;
    start2=start2-1;
    store_code{increment}=5;
    prev_code=5;
elseif y(start1,start2-1)==0 %----code=4 -prev_code=4
    start1=start1;
    start2=start2-1;
    store_code{increment}=4;
    prev_code=4;
elseif y(start1-1,start2-1)==0 %----code=3 -prev_code=4
    start1=start1-1;
    start2=start2-1;
    store_code{increment}=3;
    prev_code=3;
elseif y(start1-1,start2)==0 %----code=2 -prev_code=4
    start1=start1-1;
    start2=start2;

```

```

    store_code{increment}=2;
    prev_code=2;
elseif y(start1-1,start2+1)==0 %----code=1 -prev_code=4
    start1=start1-1;
    start2=start2+1;
    store_code{increment}=1;
    prev_code=1;
else
    start1=0;
    start2=0;
end
increment=increment+1;

elseif prev_code==7 %-----if prev_code==7
if y(start1-1,start2+1)==0 %----code=1 -prev_code=7
    start1=start1-1;
    start2=start2+1;
    store_code{increment}=1;
    prev_code=1;
elseif y(start1,start2+1)==0 %----code=0 -prev_code=7
    start1=start1;
    start2=start2+1;
    store_code{increment}=0;
    prev_code=0;
elseif y(start1+1,start2+1)==0 %----code=7 -prev_code=7
    start1=start1+1;
    start2=start2+1;
    store_code{increment}=7;
    prev_code=7;
elseif y(start1+1,start2)==0 %----code=6 -prev_code=7
    start1=start1+1;
    start2=start2;
    store_code{increment}=6;
    prev_code=6;
elseif y(start1+1,start2-1)==0 %----code=5 -prev_code=7
    start1=start1+1;
    start2=start2-1;
    store_code{increment}=5;
    prev_code=5;
elseif y(start1,start2-1)==0 %----code=4 -prev_code=7
    start1=start1;
    start2=start2-1;
    store_code{increment}=4;
    prev_code=4;
else
    start1=0;
    start2=0;
end
increment=increment+1;

end %end for prev_code=5
end %end for prev_code=8
end %end for while loop

S=cell2mat(store_code);

```



```

disp('Chain Code that has been read is:');
disp(S)
handles.image_value=S;
guidata(hObject,handles)

```

```

% --- Executes on button press in Initial_point_button.
function Initial_point_button_Callback(hObject, eventdata, handles)
% hObject    handle to Initial_point_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
y = handles.image_value2;
[k,l]=size(y);

```

```

% ----- Determine initial point for image ----- %
length_array=length(y);
for i=1:k
    for j=l:-1:1
        if y(i,j)==0
            start1=i;
            start2=j;
            disp('Image=')
            disp(y)
            fprintf(1,'Starting point for the image is at pixel (%d , %d)\n',start1,start2)
            break;
        end
    end
    if y(i,j)==0
        break;
    end
end

```

```

% --- Executes on button press in exit_button.
function exit_button_Callback(hObject, eventdata, handles)
% hObject    handle to exit_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

delete(handles.carpark_main);

```

## Appendix B: Example of Generated Chain Codes

### Number 1

Columns 1 through 16

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1
1	1	1	1	1	1	1	1	1	0	1	0	0	1	1	1
1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Starting point for the image is at pixel (2 , 13)

Chain Code that has been read is:

Columns 1 through 16

5 7 6 6 6 6 6 6 6 6 6 4 2 2 2 2

Columns 17 through 24

2 2 2 2 2 3 5 5



Number 1 (Thin)

Columns 1 through 16

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1
1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Starting point for the image is at pixel (2 , 14)

Chain Code that has been read is: 6 6 6 6 6 6 6 6 6 6 6 6

Number 5

Columns 1 through 17

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1
1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1
1	1	1	1	1	1	1	1	1	0	0	1	1	0	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1
1	1	1	1	1	1	1	1	1	0	0	1	1	0	0	1	1
1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Starting point for the image is at pixel (2, 14)

Chain Code that has been read is:

Columns 1 through 17

4	4	5	5	6	7	0	0	7	7	6	6	6	5	5	4	4
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Columns 18 through 34

4	3	0	7	0	1	2	2	2	2	2	3	4	5	4	2	2
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Columns 35 through 41

2	1	2	0	0	0	0
---	---	---	---	---	---	---



Number 5 (thin)

Columns 1 through 18

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Starting point for the image is at pixel (2 , 16)

Chain Code that has been read is:

Columns 1 through 18

4	4	4	5	6	6	6	6	7	0	0	7	6	6	6	6	6	5
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Columns 19 through 23

4	4	3	2	2
---	---	---	---	---

### Path Route

Columns 1 through 24

[illegible]





## Bibliography and References

1. MathWorks' MATLAB:  
<http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml> .
2. Sommerville, Ian. *Software Engineering, 5<sup>th</sup> Edition*. Harlow: Addison-Wesley Pub, 1995.
3. MATHEMATICA Homepage: <http://documents.wolfram.com> .
4. Milan Sonka, Vaclav Hlavac & Roger Boyle. *Image Processing, Analysis, and Machine Vision*. PWS Publishing, USA, 1998.
5. Introduction to Chain Codes; The Mind Project:  
<http://www.mind.ilstu.edu/curriculum2/perception/chaincode1.html>
6. Pfleeger, Shari Lawrence. *Software Engineering: Theory and Practice*. New York: John Wiley, 2000.
7. Source code resources: <http://www.sourceforge.net> .
8. Blobworld Source Code: <http://elib.cs.berkeley.edu/src/blobworld>
9. Computer Vision Source Code:  
[www.cs.cmu.edu/afs/cs/project/cil/ftp/html/txtv-source.html](http://www.cs.cmu.edu/afs/cs/project/cil/ftp/html/txtv-source.html)
10. Basic image processing demos showing some basic image processing filters: Thresholding, Gaussian filter, and Canny edge detector using MATLAB:  
<http://robotics.eecs.berkeley.edu/~sastry/ee20>
11. Edge Detection:  
[www.cs.cf.ac.uk/Dave/Vision\\_lecture/node24.html#SECTION00150000000000000000](http://www.cs.cf.ac.uk/Dave/Vision_lecture/node24.html#SECTION00150000000000000000)



# ABOUT UNRAVEL PARKING DIFFICULTY VIA RELATIVE CHAIN CODE ALGORITHM

## **Unravel Parking Difficulty via Relative Chain Code Algorithm USER MANUAL**

# ABOUT UNRAVEL PARKING DIFFICULTY VIA RELATIVE CHAIN CODE ALGORITHM

## Preparation

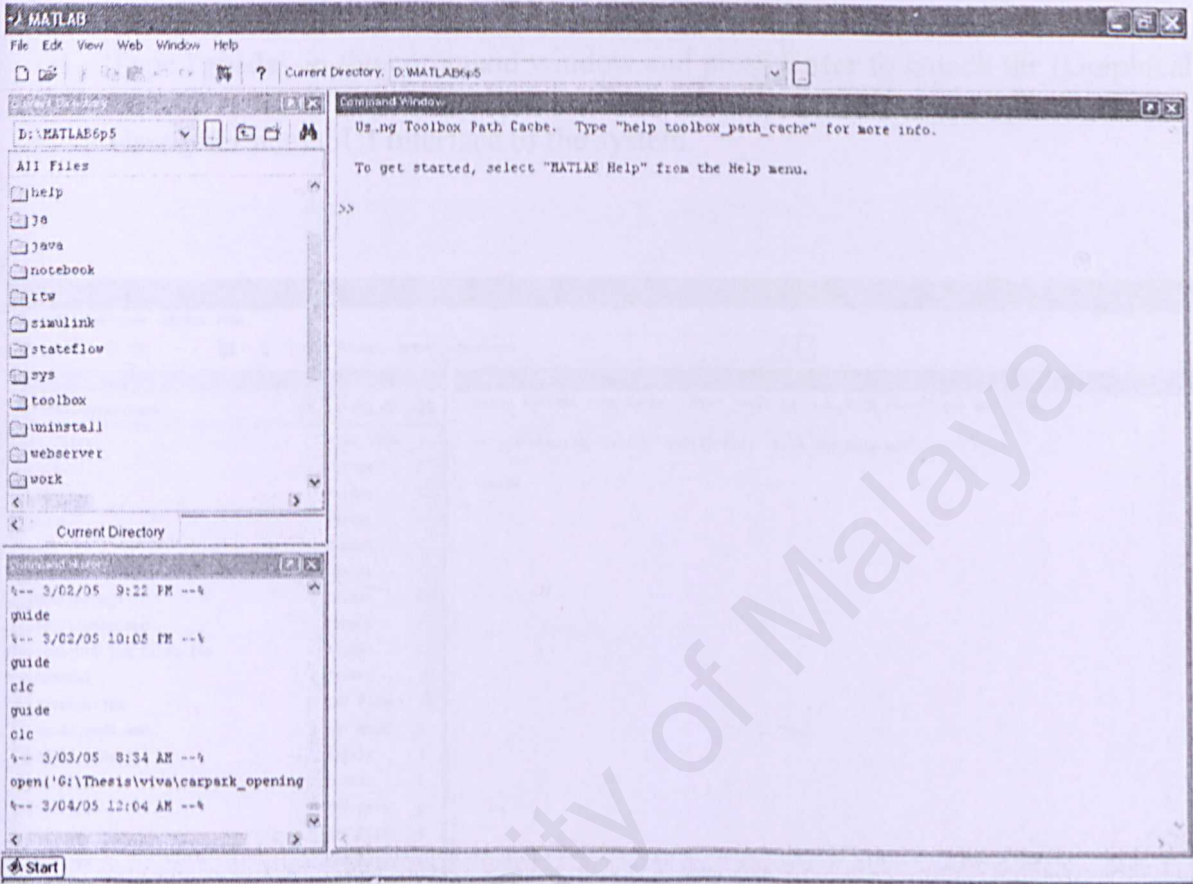
Before you can start using the Unravel Parking Difficulty via Relative Chain Code Algorithm system, a few preparations is needed in order for the system can run in a computer. Firstly, you will have to make sure that the MATLAB software, preferably MATLAB 6.5/Release 13 being installed in your operating system.

At the time of the development of the system, the version of for MATLAB is 6.5/R13. Newer version may have been developed and are usually backward compatible with older version. Thus, the Unravel Parking Difficulty via Relative Chain Code Algorithm system should be able to run in newer MATLAB version although it is being developed using the 6.5/R13 version.

Make sure that you have installed the complete version of the MATLAB software and test whether the software can function properly when it is being executed.

When MATLAB is successfully being executed, you will be brought to the desktop layout of MATLAB and you will see three different windows, which is the current directory window, command history window and the command window.

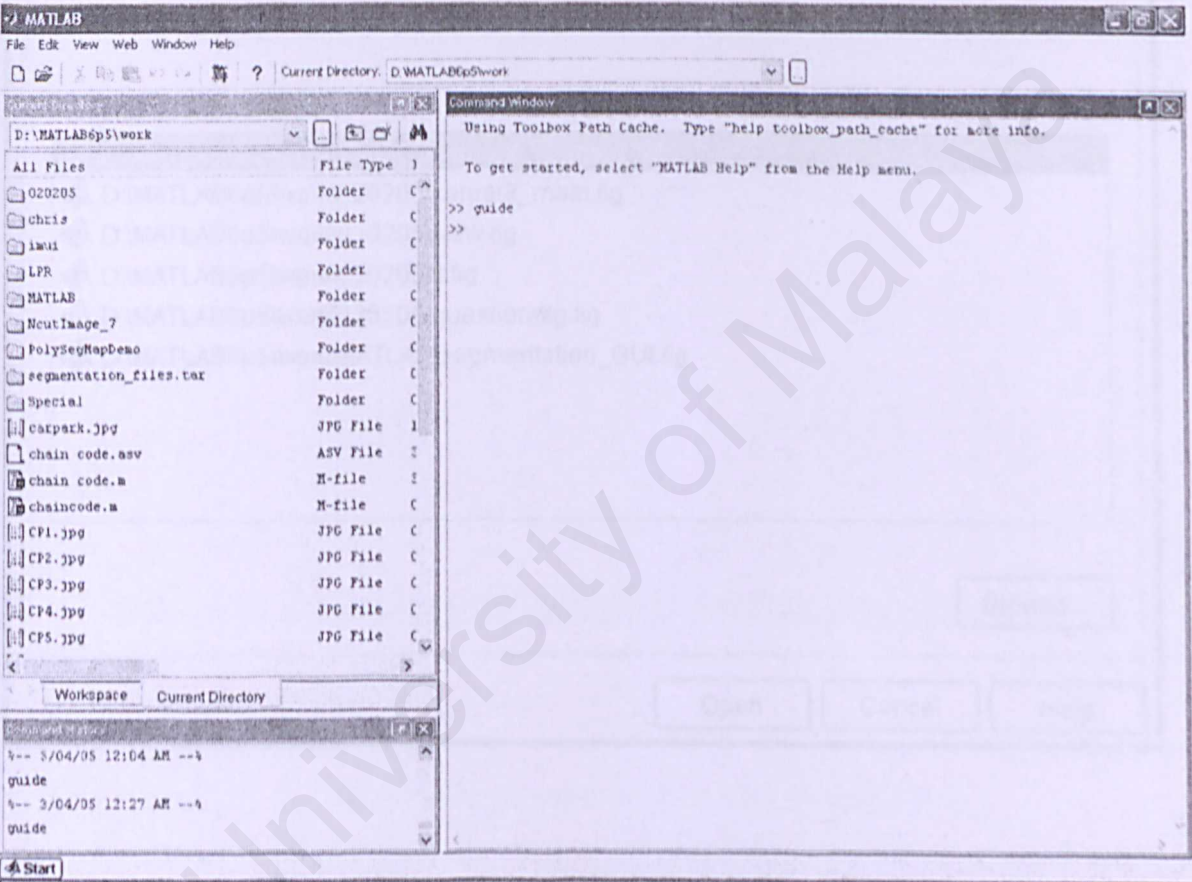




Manual Figure B1: Desktop Layout of MATLAB

# Execution of Unravel Parking Difficulty via Relative Chain Code Algorithm System

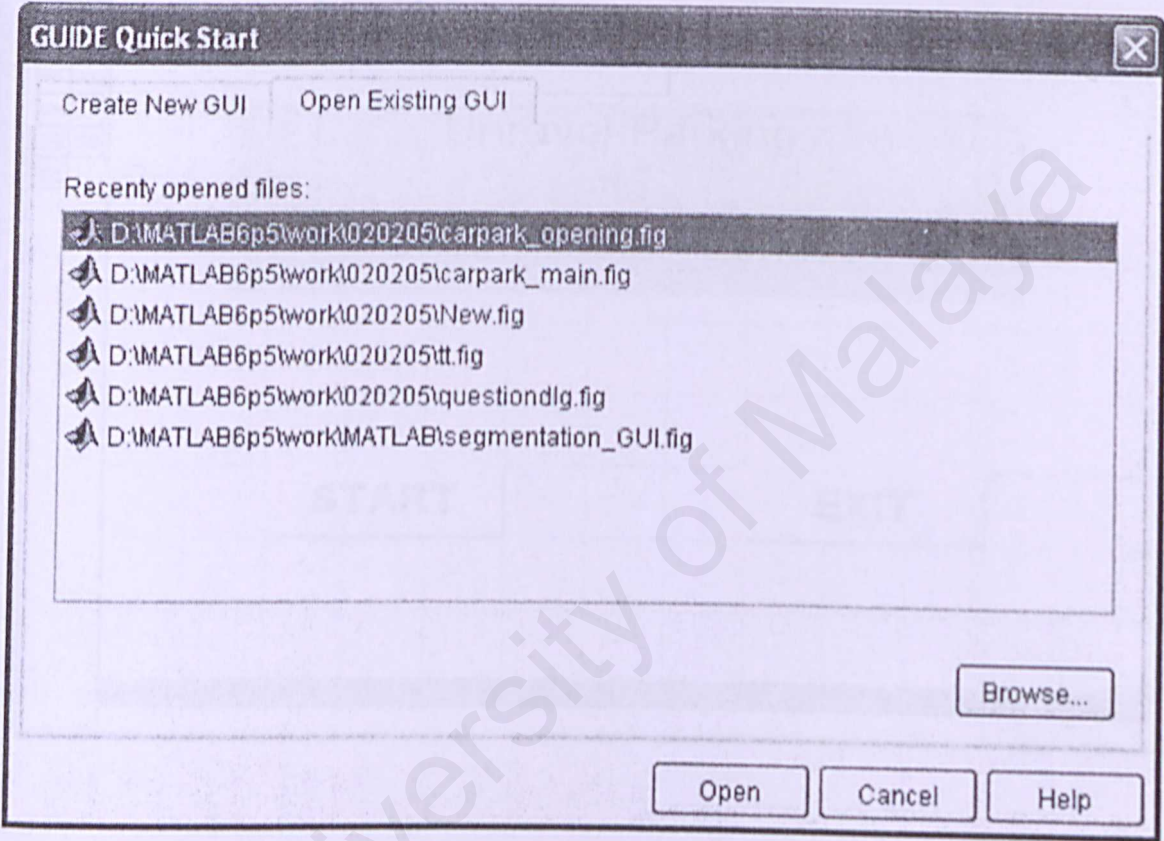
1. Type “guide” in the command window and press **Enter** to launch the (Graphical User Interface) GUI interface of the system.



Manual Figure B2: Type “guide” to launch the GUI of the system

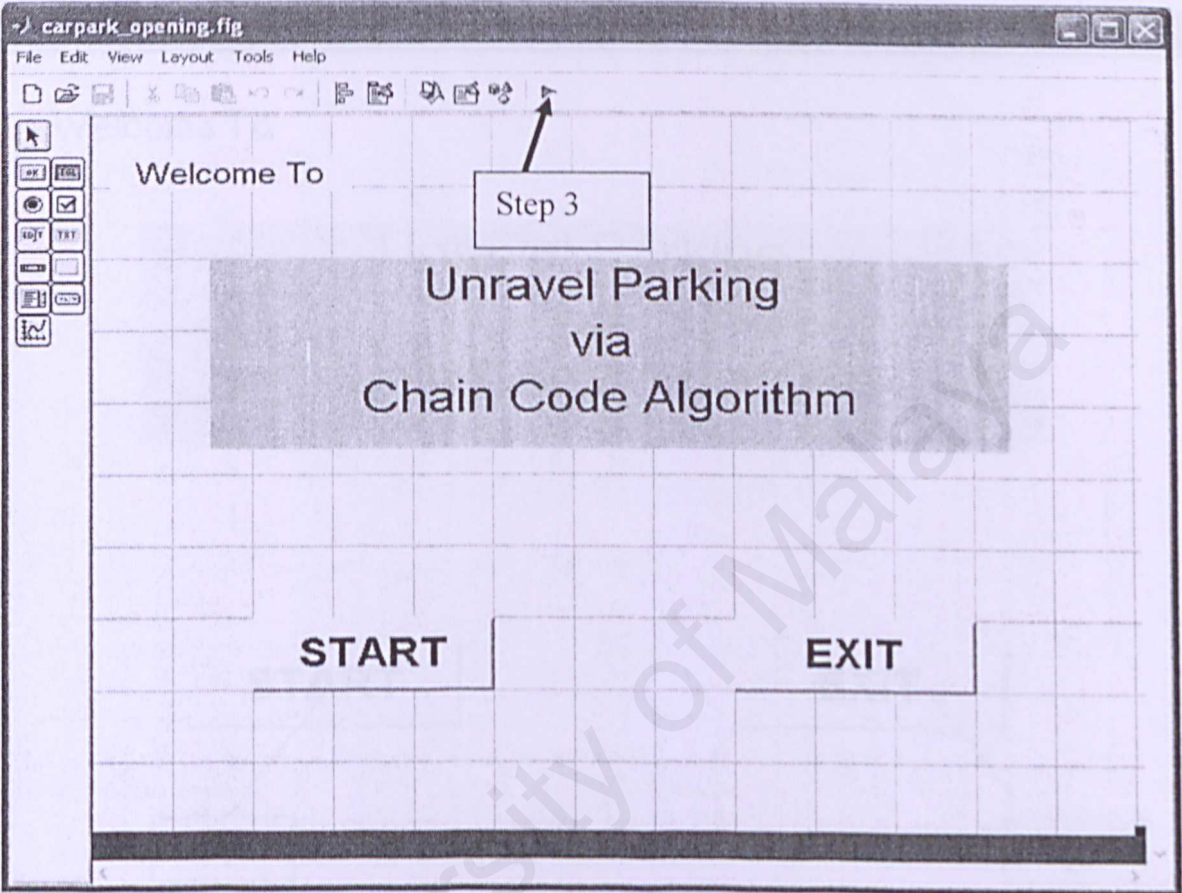


2. The GUIDE Quick Start window will appear. Go to Open Existing GUI and choose the figure file of the system where it is usually located in the **work** folder of folder **MATLAB6p5** to open the figure file of the system.



Manual Figure B3: GUIDE Quick Start window

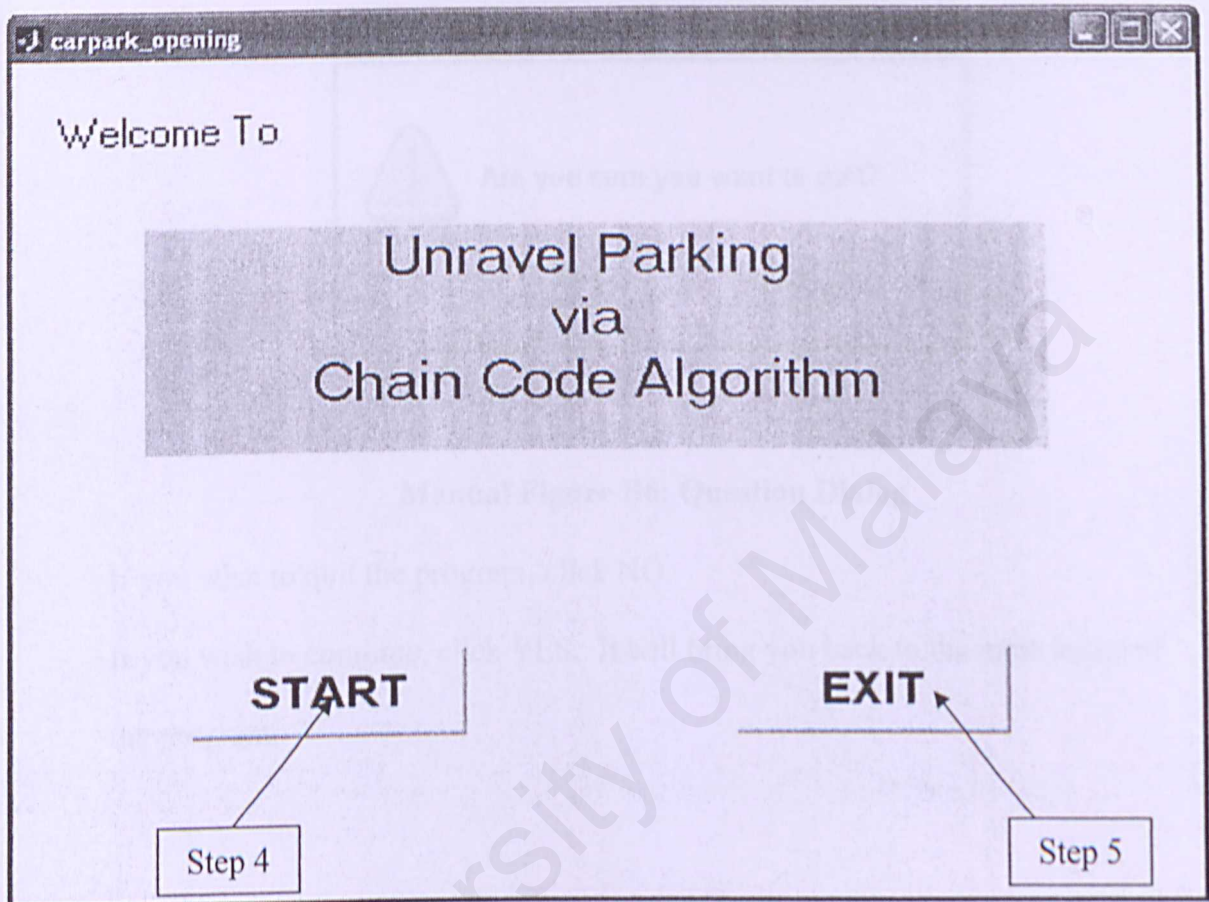
3. You will come to the figure file of the system. Click the arrow button to run the system.



Manual Figure B4: Figure File of the System

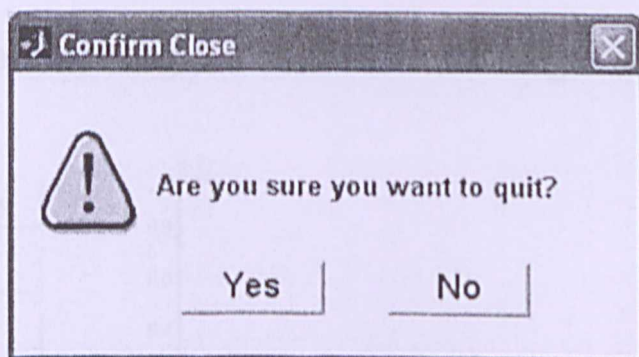


4. Click on the START button to begin the image processing process.



Manual Figure B5: Opening Page of the System

5. If you do not wish to continue, click on the EXIT button. A question dialog window will pop-up asking user whether they really want to quit the system.



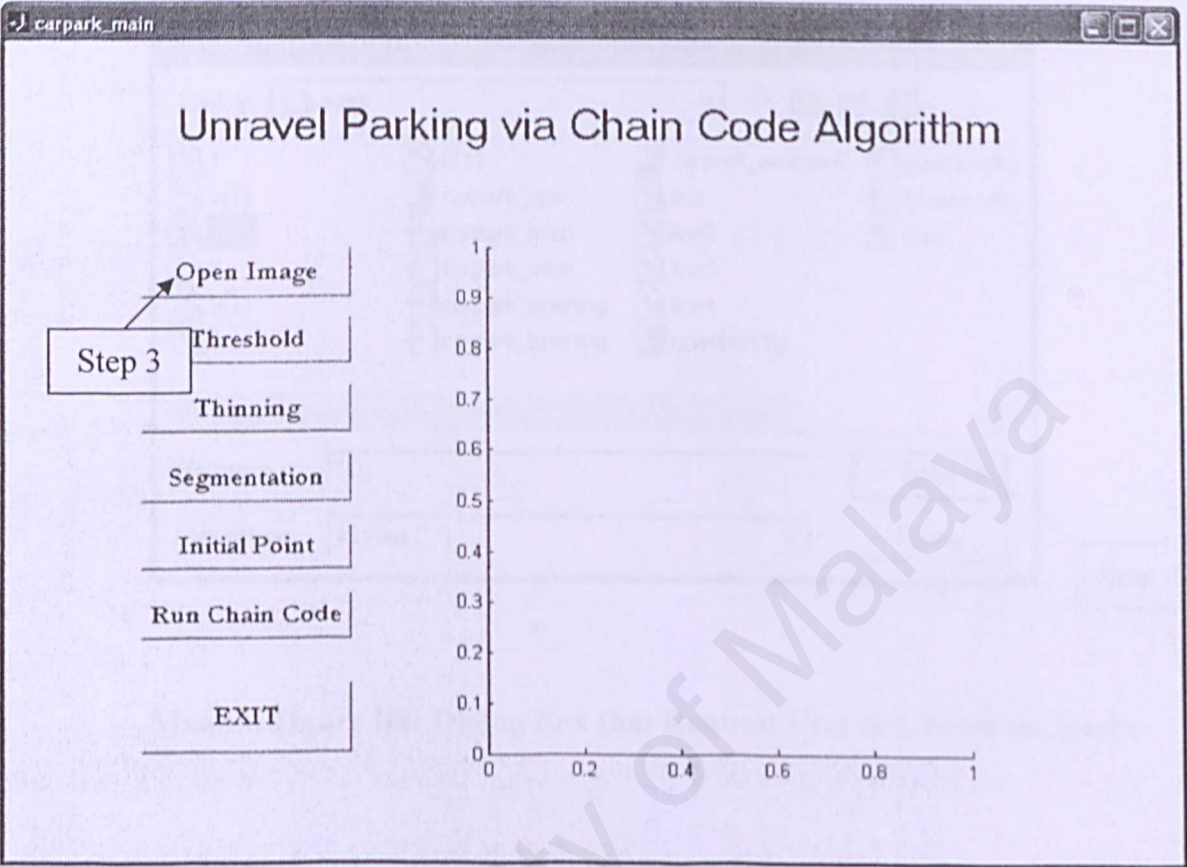
**Manual Figure B6: Question Dialog**

If you wish to quit the program, click NO.

If you wish to continue, click YES. It will bring you back to the main menu of the program.

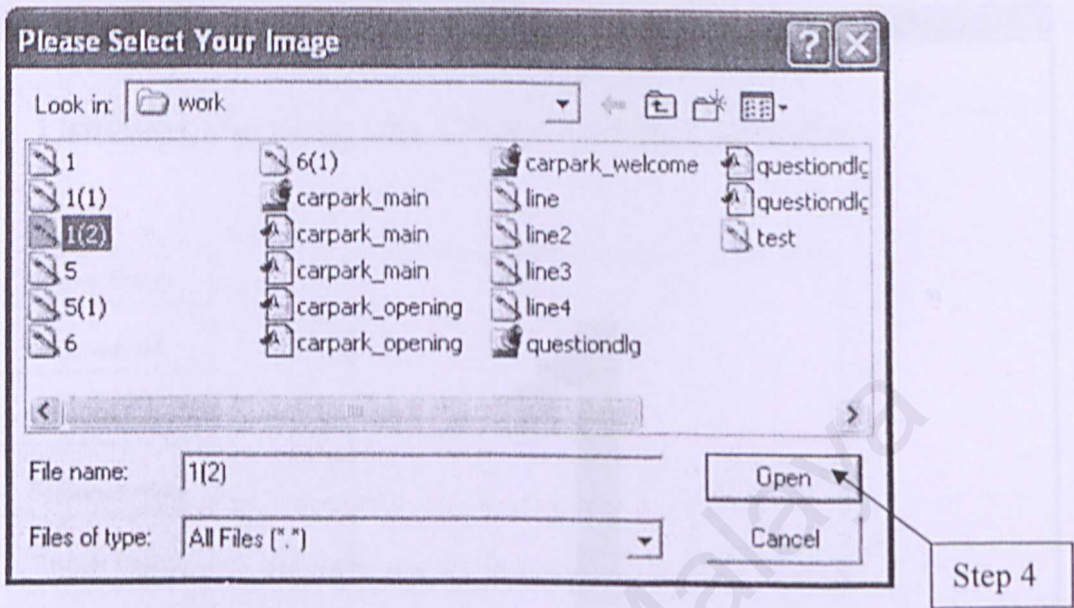


6. Click on the *Open Image* button and a pop-window will be displayed.



Manual Figure B7: Main Page of the System

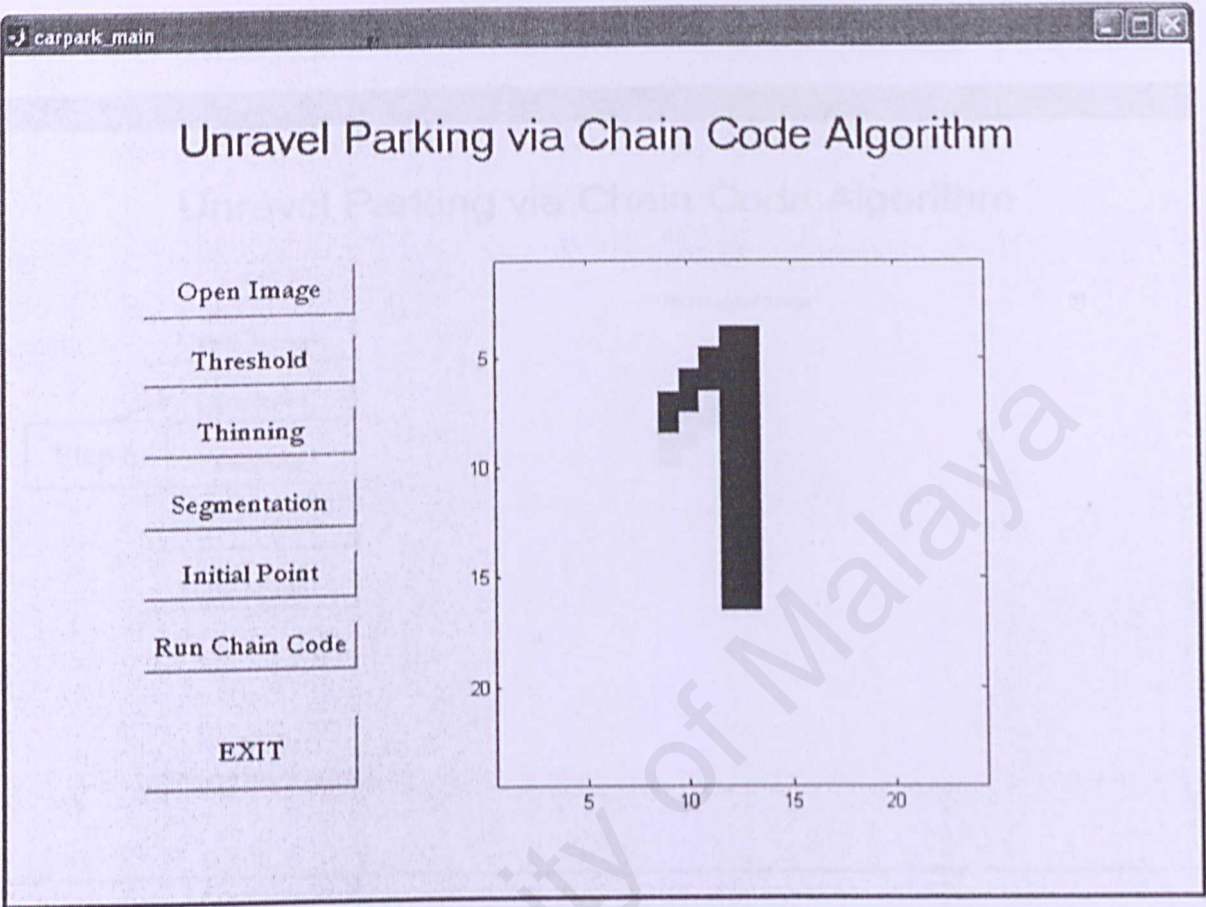
7. Choose an image file from the directory and click *Open*.



**Manual Figure B8: Dialog Box that Request User to Choose an Image**

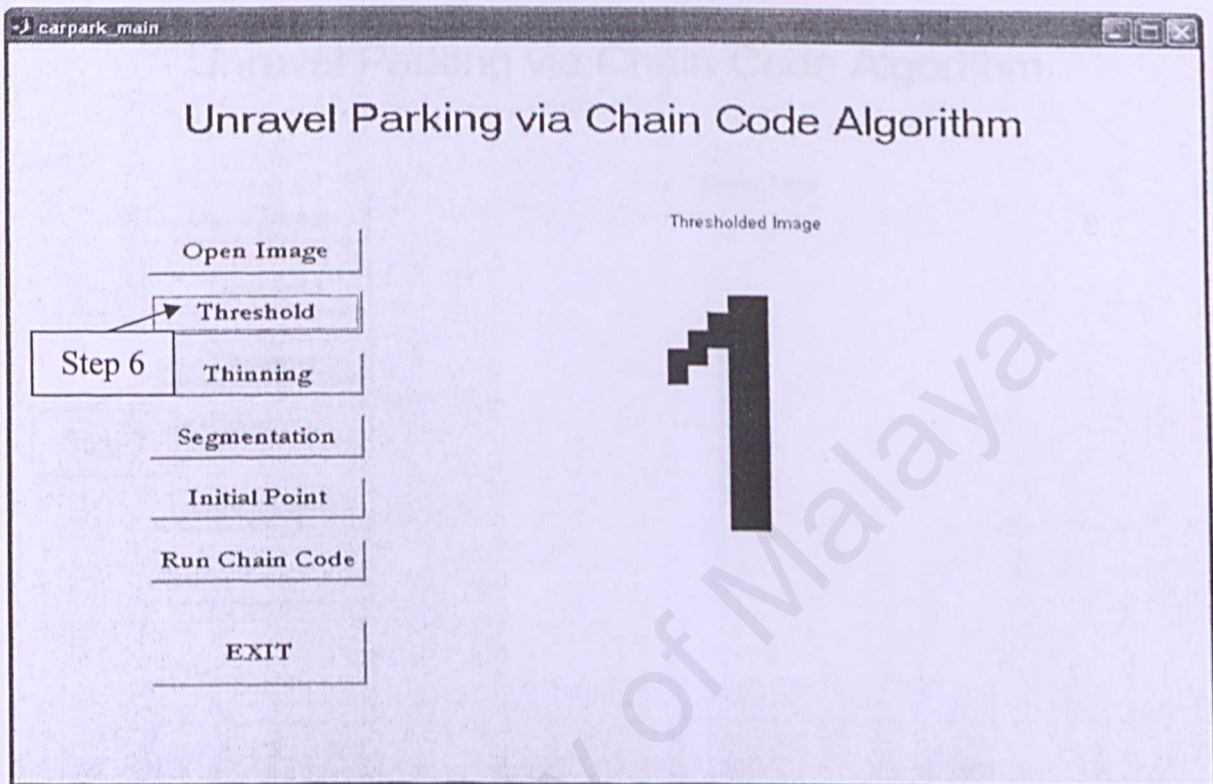


8. The chosen image file will be displayed on the screen.



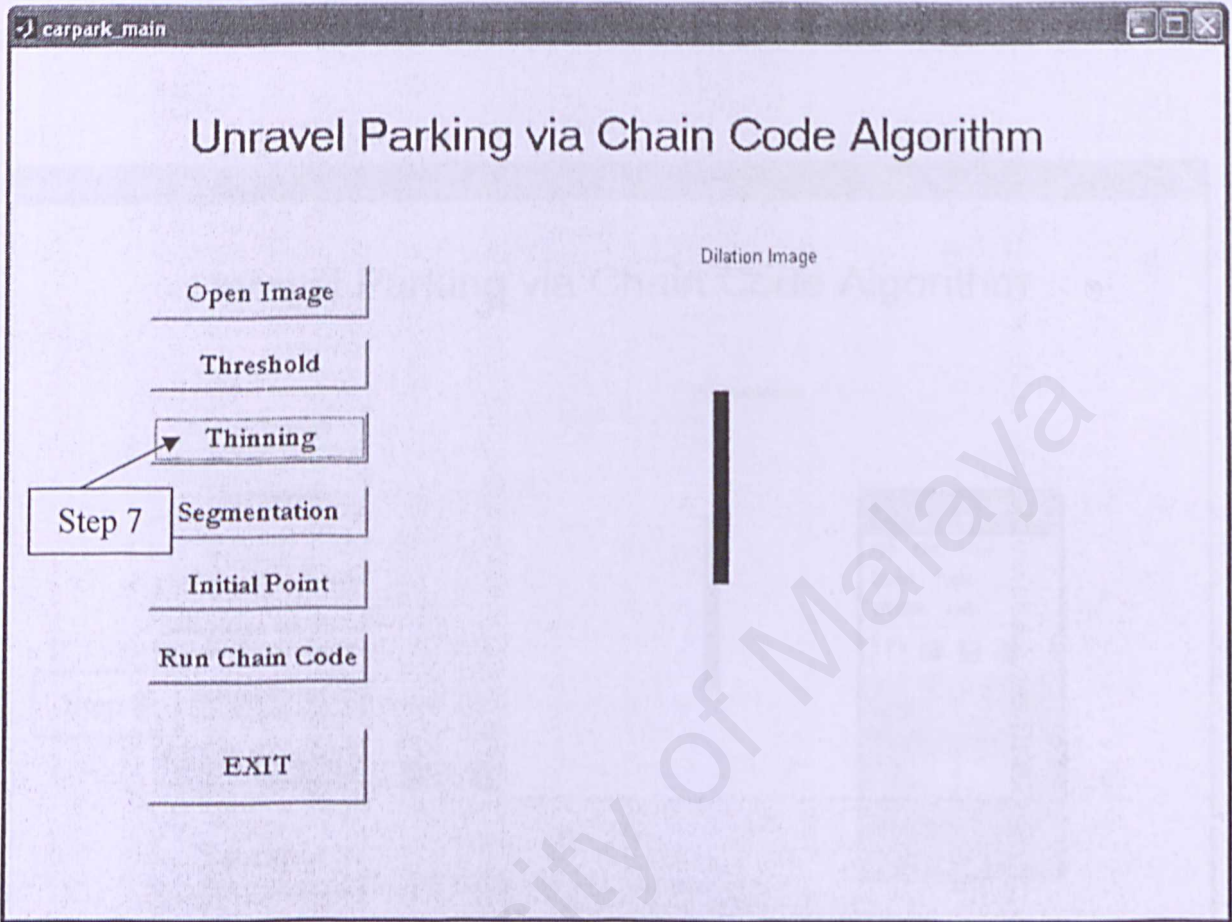
Manual Figure B9: Chosen Image Being Displayed

9. Click on the *Threshold* button and the image that has been threshold will be displayed.



Manual Figure B10: Thresholding Process

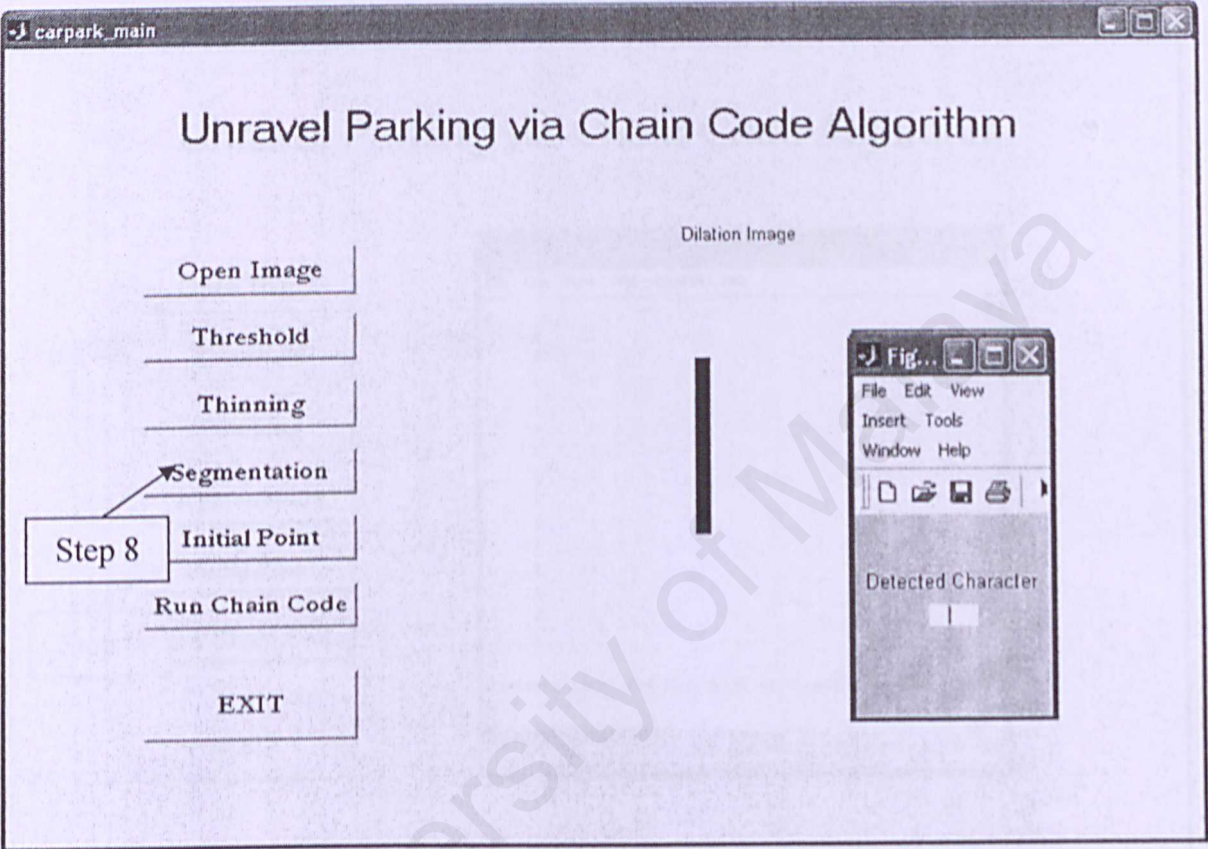
10. Click on the Thinning button to view the image that has been make it thinner.



Manual Figure B11: Thinning Process

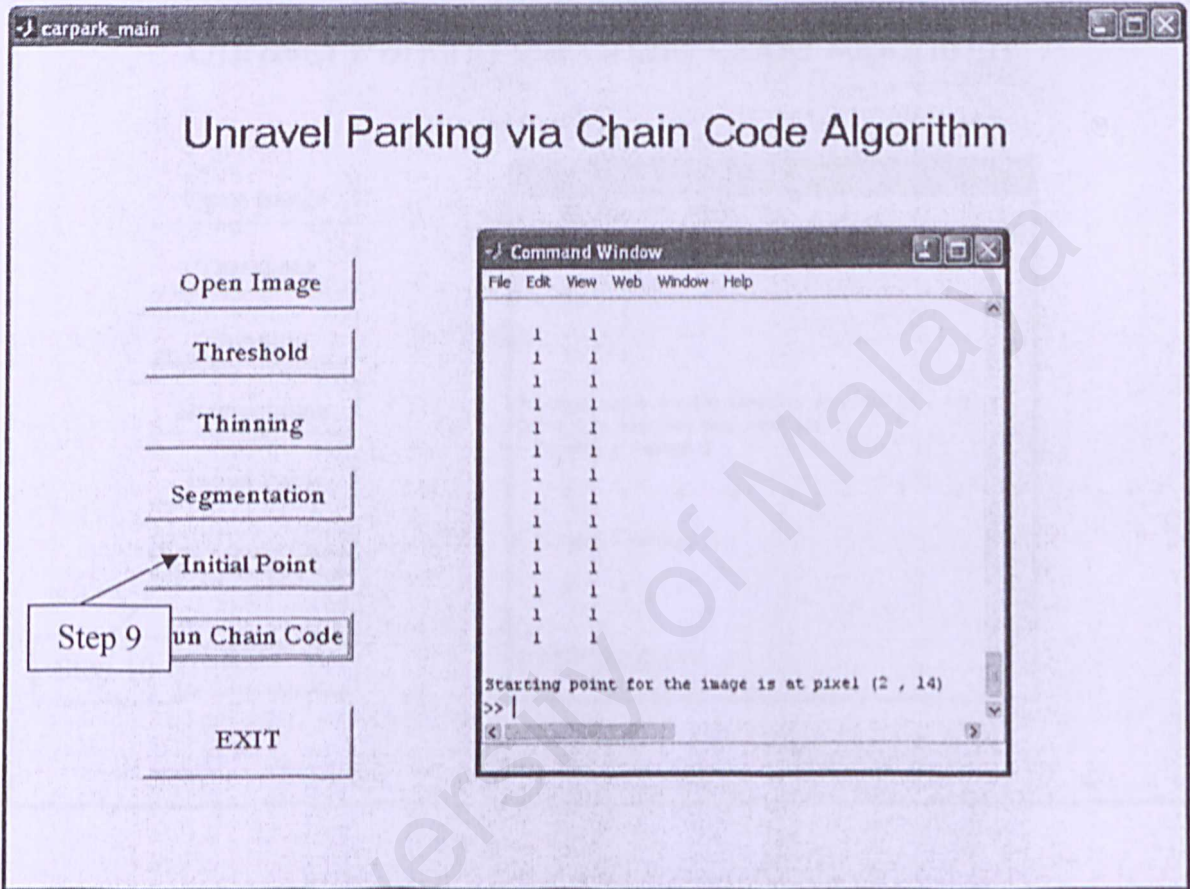


11. Next is the segmentation process. Click on the Segmentation button and you will see a clear image which is free from other unwanted/extra image.



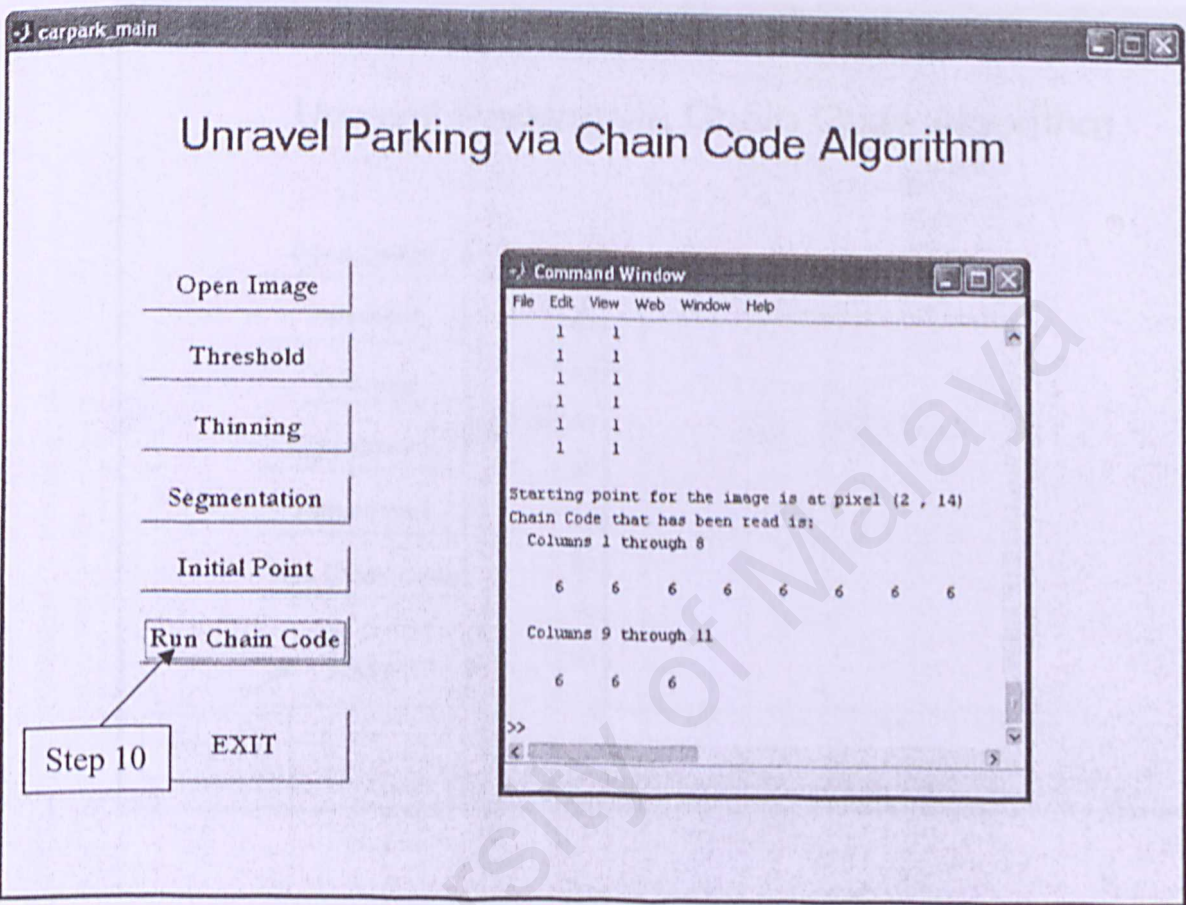
Manual Figure B12: Segmentation Process

12. Click on the Initial Point button to determine the initial point for the image to be recognized.



Manual Figure B13: Determine the Initial Point of the Image

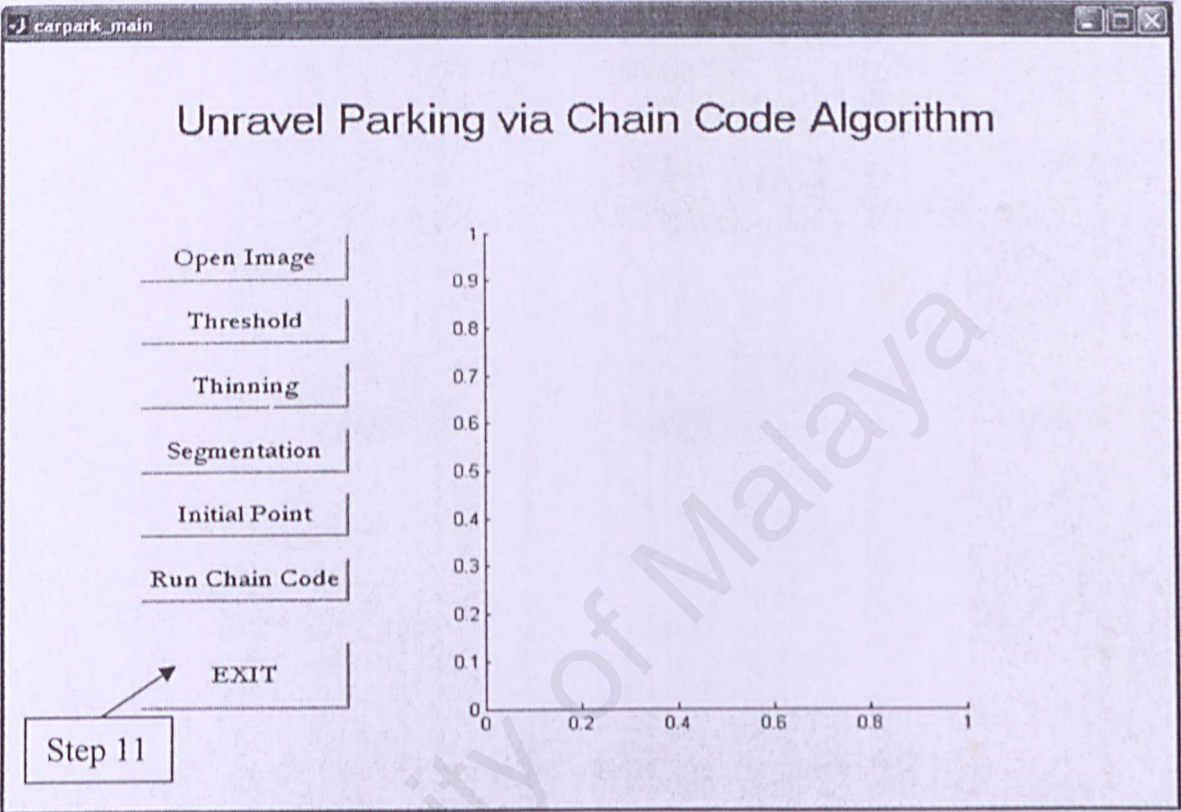
13. Click on the Run Chain Code button to see the generated code for the image.



Manual Figure B14: Generate Chain Code of the Image



14. Click on the EXIT button to quit the system.



Manual Figure B14: Exit the System