Algorithms and architectures for low-cost license plate recognition system

Ku, Wei Chiet

2012

Ku, W. C. (2012). Algorithms and architectures for low-cost license plate recognition system. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/50543>

<https://doi.org/10.32657/10356/50543>

**ALGORITHMS AND ARCHITECTURES
FOR LOW-COST LICENSE PLATE
RECOGNITION SYSTEM**

KU WEI CHIET

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Master of Engineering

2012

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Thambipillai Srikanthan for his invaluable guidance, support and suggestions throughout the research. He has always leaded me into the right direction when I was confused by my own thoughts. Without his assistance, I would not been able to accomplish my goals for this project.

I also want to thank my colleagues in Centre for High Performance Embedded Systems (CHiPES), for their generous help and support to complete this project. Last but not least, thanks to my parents for their understanding and encouragement at all time.

Table of Contents

Acknowledgements	2
Table of Contents	3
List of Figures.....	6
List of Tables	9
Abstract.....	10
Chapter 1 Introduction.....	12
1.1 Applications of License Plate Recognition System	12
1.2 Components of License Plate Recognition System	14
1.3 Project Objectives	15
1.4 Organization of the Thesis	16
Chapter 2 Literature Review	18
2.1 License Plate Localization	18
2.1.1 Binary/Edge Image-Based License Plate Localization.....	19
2.1.2 Grey Scale Image-Based License Plate Localization	21
2.1.3 Colour Image-Based License Plate Localization.....	24
2.1.4 Classifier-Based License Plate Localization.....	26
2.2 License Plate Character Segmentation.....	28
2.2.1 Binary Image-Based License Plate Character Segmentation.....	29
2.2.2 Grey Level Image-Based License Plate Character Segmentation	30
2.3 License Plate Character Recognition	32
2.3.1 Classifier-Based License Plate Character Recognition.....	32
2.3.2 Artificial Neural Network-Based License Plate Character Recognition	33
2.3.3 Pattern/Template Matching-Based License Plate Character Recognition ..	35
2.4 Commercial License Plate Recognition System	36
2.4.1 Fixed License Plate Recognition System.....	36
2.4.2 Notebook PC-Based Mobile License Plate Recognition System	38
2.4.3 Handheld Device-Based Mobile License Plate Recognition System	39

2.4.4 Vehicle Mounted-Based Mobile License Plate Recognition System	40
2.5 Summary	41
Chapter 3 License Plate Localization	43
3.1 Image Processing Functions	43
3.1.1 Convolution Operation.....	43
3.1.2 Edge Detection Convolution Filters.....	44
3.1.3 Binary Morphological Operations	45
3.2 Proposed Method for License Plate Localization	48
3.2.1 Vertical Edge Detection Using Sobel Filter.....	48
3.2.2 Connecting Vertical Edges Using Morphological Operations.....	50
3.3 Experimental Results of License Plate Localization Process	53
Chapter 4 License Plate Identification	64
4.1 Hough Transform-Based Line Detection.....	64
4.2 Removing Undesirable Area of License Plate Candidate.....	67
4.3 Verification by Horizontal Rule.....	70
4.4 Verification by Dip Rule.....	71
4.4.1 Horizontal Projection and Dips Detection	71
4.4.2 Elimination of Superfluous Dips	72
4.4.3 Applying Dip Rule.....	73
4.4.4 Character Segmentation	74
4.4.5 Verification by Dip Rule for License Plate with Dark Characters and Bright Background	75
4.5 Verification by Vertical Rule.....	76
4.6 Experimental Results of License Plate Identification Process.....	77
Chapter 5 Signature-Based Character Recognition Process	93
5.1 Character Enlargement for Scale Invariance	94
5.2 Horizontal Line Signature.....	94
5.3 Vertical Line Signature	98
5.3.1 Dual Segment-Based Symmetry Check.....	98
5.3.2 Quad Segment-Based Check.....	101

5.3.3 Separating Character ‘H’ From Character ‘W’	106
5.3.4. Verifying Character ‘1’	106
5.4 Experimental Results of Signature-Based Character Recognition	107
Chapter 6 FPGA-Based Hardware Implementation	114
6.1 Architecture of CSoC-Based System.....	114
6.2 Implementing Sobel Edge Detector	117
6.2.1 Avalon Slave Interface Block of Sobel Edge Detector.....	118
6.2.2 Input Buffer Block of Sobel Edge Detector.....	118
6.2.3 Sobel Vertical and Horizontal Filter Blocks.....	120
6.2.4 Thresholding Block.....	121
6.3 Implementing Morphological Operator	121
6.3.1 Avalon Slave Interface Block of Morphological Operator.....	122
6.3.2 Input Buffer Block of Morphological Operator.....	122
6.3.3 Dilation/Erosion Block	123
6.4 Implementing Hough Transform-Based Line Detector	124
6.4.1 Avalon Slave Interface Block of Line Detector.....	124
6.4.2 Hough Transform (HT) Processor	125
6.4.3 Line Pixels Generation Module	128
6.4.4 Line Segment Identification Software Function.....	130
6.5 Experimental Results of Hardware Implementation System.....	130
Chapter 7 Conclusions and Future Work	133
7.1 Conclusions.....	133
7.2 Future Work.....	135
References.....	137
Appendix.....	147
A.1 Characters Used in Singapore Vehicle License Plate	147

List of Figures

Chapter 1

Figure 1.1: Components in a typical license plate recognition system.....	15
---	----

Chapter 2

Figure 2.1: AutoPlate fixed license plate recognition system by PIPS Technology	37
Figure 2.2: License plate recognition system running on mobile computing device	38
Figure 2.3: Handheld license plate recognition system.....	39
Figure 2.4: Vehicle mounted license plate recognition system	40

Chapter 3

Figure 3.1: Convolution operation on a 3x3 image region.....	43
Figure 3.2: Sobel edge detection kernels.....	44
Figure 3.3: Dilation Operation	46
Figure 3.4: Erosion Operation	47
Figure 3.5: License plate localization process.....	48
Figure 3.6: Vehicle image and its vertical and horizontal edges.....	49
Figure 3.7: Series of morphological operations to locate license plate region - Part I.....	52
Figure 3.8: Series of morphological operations to locate license plate region - Part II	53
Figure 3.9: Samples of images used in license plate localization experiment - Part I.....	56
Figure 3.10: Samples of images used in license plate localization experiment - Part II	57
Figure 3.11: Samples of images used in license plate localization experiment - Part III.....	58
Figure 3.12: Samples of images used in license plate localization experiment - Part IV	59
Figure 3.13: Samples of images used in license plate localization experiment - Part V	60
Figure 3.14: Samples of images used in license plate localization experiment - Part VI.....	61
Figure 3.15: Samples of images used in license plate localization experiment - Part VII	62
Figure 3.16: Samples of images used in license plate localization experiment - Part VIII	63

Chapter 4

Figure 4.1: Flow chart of license plate identification process.....	64
Figure 4.2: Linear line representations in xy-plane and parameter space	65
Figure 4.3: Hough Transform parameter space.	66
Figure 4.4: Line detection by using Hough Transform	66
Figure 4.5: Removing undesirable area process on Candidate 1.....	68
Figure 4.6: Removing undesirable area process on Candidate 2 to Candidate 4	69
Figure 4.7: Verification by Horizontal Rule.....	70
Figure 4.8: Horizontal projection of Candidate 1.....	72
Figure 4.9: Segmentation lines plotted on horizontal line images.....	72
Figure 4.10: Horizontal projection with superfluous dips eliminated	73
Figure 4.11: Bounding boxes enclosing each character	74
Figure 4.12: Horizontal projection of license plate with dark characters and bright background	76
Figure 4.13: Horizontal projection of the complement image of license plate with dark characters and bright background.....	76

Figure 4.14: Verification by Vertical Rule.....	77
Figure 4.15: Result of applying license plate identification process on example images– Part I	80
Figure 4.16: Result of applying license plate identification process on example images – Part II	81
Figure 4.17: Result of applying license plate identification process on example images – Part III	82
Figure 4.18: Result of applying license plate identification process on example images – Part IV	83
Figure 4.19: Result of applying license plate identification process on example images – Part V	84
Figure 4.20: Result of applying license plate identification process on example images – Part VI	85
Figure 4.21: Result of applying license plate identification process on example images – Part VII	86
Figure 4.22: Result of applying license plate identification process on example images – Part VIII	87
Figure 4.23: Result of applying license plate identification process on example images – Part IX	88
Figure 4.24: Result of applying license plate identification process on example images – Part X	89
Figure 4.25: Result of applying license plate identification process on example images – Part XI	90
Figure 4.26: Result of applying license plate identification process on example images – Part XII	91
Figure 4.27: Result of applying license plate identification process on example images – Part XIII	92

Chapter 5

Figure 5.1: Signature-based character recognition process	93
Figure 5.2: Horizontal line signature of character ‘3’	94
Figure 5.3: Horizontal line signatures of characters ‘0’ to ‘H’	96
Figure 5.4: Horizontal line signatures of characters ‘J’ to ‘Z’	97
Figure 5.5: Vertical line signatures of characters ‘6’, ‘8’, and ‘9’	98
Figure 5.6: Vertical line signatures of characters ‘0’ to ‘N’ by using dual segment-based symmetry check.....	100
Figure 5.7: Vertical line signatures of characters ‘P’ to ‘Z’ by using dual segment-based symmetry check.....	101
Figure 5.8: Vertical line signatures of characters ‘0’, ‘C’, and ‘S’ by using quad segment-based check.....	103
Figure 5.9: Vertical line signatures of characters ‘U’, ‘V’, and ‘Y’ by using quad segment-based check.....	104
Figure 5.10: Vertical line signatures of characters ‘K’, ‘M’, ‘N’ and ‘X’ by using quad segment-based check without including long vertical lines	105
Figure 5.11: Characters ‘1’ with different vertical line signature based on dual segment-based check.....	106
Figure 5.12: Samples of character images used in experiment – Part I.....	108

Figure 5.13: Samples of character images used in experiment – Part II	109
Figure 5.14: Samples of character images used in experiment – Part III	110
Figure 5.15: Samples of character images used in experiment – Part IV	111
Figure 5.16: Samples of character images used in experiment – Part V	112
Figure 5.17: Samples of character images used in experiment – Part VI.....	113

Chapter 6

Figure 6.1: FPGA-based implementation of license plate recognition system	115
Figure 6.2: High-level block diagram of CSoC architecture.....	115
Figure 6.3: Block diagram of Sobel Edge Detector.....	118
Figure 6.4: Input buffer block of Sobel Edge Detector	119
Figure 6.5: Sobel edge detection kernels.....	120
Figure 6.6: Block diagram of Sobel Filters implementation	120
Figure 6.7: Block diagram of Morphological Operator.....	121
Figure 6.8: Block diagrams of Input Buffer and its output control logic	122
Figure 6.9: Block diagram of Dilation/Erosion.....	123
Figure 6.10: Block diagram of Hough Transform-Based Line Detector.....	124
Figure 6.11: Architecture of Hough Transform Processor block	125
Figure 6.12: Block diagram of a processing element in Hough Transform Processor block	126
Figure 6.13: Architecture of Line Pixels Generator	129

List of Tables

Table 5.1: Code words of all the characters based on their horizontal line signature	98
Table 5.2: Vertical line signatures of characters with horizontal line signature <i>010-000-010</i> by using quad segment-based check.....	102
Table 5.3: Vertical line signatures of characters with horizontal line signature <i>101-000-010</i> by using quad segment-based check.....	102
Table 5.4: Vertical line signatures of characters with horizontal line signature <i>101-000-101</i> by using quad segment-based check.....	102
Table 6.1: Success rate of hardware implementation	131
Table 6.2: Average execution time of one image on CSoC platform.....	131
Table 6.3: Logic elements utilization and estimated power consumption of CSoC system..	131

Abstract

License plate recognition system has its application in many areas such as recording parking statistics, identifying car thefts and monitoring traffic flow. Though several effective solutions have been proposed, the computation complexity has been one of the main limitations. Commercial mobile license plate recognition systems continue to suffer from the shortcomings of high cost and bulky implementation. This thesis presents several novel algorithms in an attempt to realize a low-cost license plate recognition system capable of real-time performance.

In this research, the license plate recognition process is divided into three stages: license plate localization, license plate identification and license plate character recognition. The license plate localization was realized by first finding the vertical edges in the grayscale vehicle image using Sobel edge detector. A series of morphological operation was then applied on the vertical edges to extract rectangular regions with dense concentration of vertical edges. An efficient technique was then introduced and tested for the removal undesirable license plate regions. This technique employs Hough Transform based line detector to detect vertical lines and horizontal lines in the license plate candidates. A unique relationship between the number of horizontal and vertical lines were established to isolate the license plate more accurately. This made it possible for the elimination of other candidates with similar edge characteristics.

The proposed license plate identification process involves the efficient analysis of both the horizontal and vertical planes. In particular, the analysis of the vertical plane examines for local minimums (dips) in the license plate's pixels intensity projection histogram. A technique was also introduced for the efficient segmentation of the characters of the license plate by relying on the uniform gaps between the characters. Lastly, unique codes were generated using a combination of vertical and horizontal line signatures to recognize the alpha numeric characters of the license plate.

The proposed techniques were fully evaluated in MATLAB using two hundred vehicle images. The proposed algorithms were subsequently implemented on an

Altera's Nios II-based Configurable System on Chip (CSoC) platform. Suitable partitioning of the application was performed to accelerate computation intensive image processing algorithms in FPGA hardware. In particular, Sobel edge detection, binary morphological operations and Hough Transform based line detection were ported to FPGA by exploiting the inherent parallelism for high performance. Experiment results show the hardware system achieved satisfactory performance with overall accuracy of 86%. With hardware accelerators, the execution speed was increased by significant 19x and the system took an average of 28s to recognize a vehicle from an image.

Chapter 1

Introduction

Many researches on Intelligent Transportation Systems (ITS) have been reported in recent years. ITS have had a wide impact in people's life by improving transportation safety and mobility. As one form of ITS technology, vehicle's license plate recognition system has been the subject of ongoing research for many years. The goal of the system is to distinguish a vehicle by recognizing the characters in the license plate. In a typical license plate recognition system, a computer processes a vehicle image captured by a camera and applying various image processing and optical pattern recognition techniques to recognize the information on the license plate. Since the license plate is already the unique identification for all vehicles, no additional transmitter or responder is required to be installed on the vehicle for the recognition process.

1.1 Applications of License Plate Recognition System

The fact that a license plate is universally required on all vehicles provides a key point of identification for the operator of that vehicle. License plate recognition system automates the process of gathering this information. The following list of application examples provides an overview of some areas where license plate recognition technologies are commonly deployed.

- **Parking Management:** License plate recognition system can be installed at the entrance of parking structure to monitor the passing cars. When a vehicle enters an input gate, its number plate is automatically recognized and stored in database. The number plate is read again upon the exit of the car and the difference in time is used to calculate the parking fee. The parking record statistics can also be used to display current parking spaces available.

- **Access Control and Security:** License plate recognition is used in many companies and facilities to grant access only to vehicles of authorized personnel. The license plate recognition system installed at the entrance gate can automatically open the gate for the returning tenants by identifying their vehicles stopping at the doorway. Compared with traditional approach where the company usually hiring security guards or installing special device on the vehicles for identification purpose, license plate recognition system provides a more efficient and flexible way of access control. Integrated license plate recognition software provides facility managers the ability to pre-define entry and exit regulations with different level of access, such as limiting admission to certain hours or days of the week.
- **Speed/ Law Enforcement:** Many cities have installed traffic cameras to monitor the movement and flow of vehicles around the road network. By using license plate recognition on these footages, it is possible to measure the amount of time a particular vehicle transitions between two locations. Since the distance between the two license plate recognition locations is static, the vehicle's average speed can be calculated based on the captured images. An automatic ticket can be generated and sent to the owner of the vehicle by cross-referencing the license plate recognition information with an existing database. It also can help law enforcement authorities to track and find suspect vehicles and illegal use of bus lanes in real time.
- **Tracking and Traffic Management:** In the case of a stolen vehicle or a car observed at the scene of a crime, license plate recognition systems distributed in an area can be used to track subject vehicles. Each license plate recognition system has a known location and can provide the exact time a specific vehicle is observed. From this data, a tracking map can be created to find the subject vehicle's location. Similarly, license plate recognition system can also provide a mapping

of traffic flow and be used as a car counting mechanism aiding in the development of traffic management data.

- **Customs/Immigration:** At border crossings and customs inspection areas, a large volume of vehicle traffic often occurs. License plate recognition system can be used to automatically scan the license plates of all vehicles approaching the border and search them against a database. The database could include stolen vehicles, related criminal histories, and other vehicles of interest. If something is found in the database, the vehicle can be routed to a different area for further investigation.
- **Mobile Vehicle Readers:** By installing a license plate recognition system in a patrol car, new opportunities are available to law enforcement officers. License plates can be identified at a distance while not distracting the driver. The officers can match the captured license plate image against a database of stolen or suspect vehicles. If there is a positive match, the system can alert the officer for further action.

1.2 Components of License Plate Recognition System

A typical license plate recognition system is composed of several hardware and software components that processes an input graphical signal like static snapshots or video sequences, and recognizes license plate characters from it. A fixed license plate recognition system consists of several units as illustrated in Figure 1.1.

When a vehicle approaches the secured area or gate that installed with a license plate recognition system, the sensor detects the presence of the vehicle and activates the camera to takes its pictures. Alternative to this solution is a software detection of an incoming vehicle, or continual processing of the sampled video signal. These approaches may consume more computation resources, but it does not need additional hardware equipment like the sensor or hardware trigger.

The captured image is sent to a stand-alone computer to analyze the image with special software recognition algorithms. The algorithm, which is the key part of whole license plate recognition system, detects the license plate position, segments the license plate character string, and identifies the characters in the plate. The extracted license plate information can be logged and stored along with the captured image in database, or used for authentication depending on the license plate recognition application.

Figure 1.1: Components in a typical license plate recognition system

1.3 Project Objectives

As will be explained in greater details in next chapter, majority of the commercial license plate recognition systems as well as those techniques reported in literature are using conventional PC based approach shown in Figure 1.1. These systems are expensive, power hungry and not highly portable. These reasons have motivated the need to devise a license plate recognition system without the shortcomings associated with these typical systems.

The main objective of this project is to propose efficient embedded computing techniques in an attempt to realize a low-cost license plate recognition system capable of real-time performance. It is envisaged that the license plate recognition system will be implemented as stand-alone smart camera system which can be easily deployed in

remote locations for the identification and wireless transmission of license plate information. Hardware efficient algorithms in various processing stages of the license plate recognition system are needed to achieve this goal.

The proposed algorithms need to be evaluated on an embedded platform in order to analyze their effectiveness and feasibility. Configurable System on Chip (CSoC) [1] is a suitable platform to serve this purpose. Processor, memories and peripheral interfaces are integrated in a single Field Programmable Gate Arrays (FPGA) chip. Designers are able to apply a combination of serial and parallel processing to perform complex functions on this platform. The processor can be used to execute high level sequential functions. By leveraging the configurability of the FPGA, computation intensive functions can be implemented as custom hardware to achieve high performance and low-power consumption.

1.4 Organization of the Thesis

This thesis is organized as follow:

- Chapter 1 (current chapter) introduces the license plate recognition system as well as objectives of this project.
- Chapter 2 provides literature review of the previous research on license plate recognition system. The advantages and limitation of these solutions for different processing steps of license plate recognition system are examined. Commercial license plate recognition system available in the market will also be presented.
- Chapter 3 explains the vertical edges based method to locate the rough location of license plate.
- Chapter 4 illustrates the identification rules to eliminate false candidates that have been extracted together with license plate in the previous localization process.
- Chapter 5 presents the character recognition algorithm based on linear line signatures of license plate characters.
- Chapter 6 describes the FPGA-based hardware implementation of the proposed license plate recognition system.

- Chapter 7 summarizes the thesis study with conclusion and discusses the direction of future research.

Chapter 2

Literature Review

In general, the license plate recognition algorithm running on the processing unit is the most important part of the system. The recognition process can be divided into 3 main tasks: 1) license plate localization from the captured image 2) license plate character segmentation 3) license plate character recognition. This chapter first examines the main techniques of these tasks that have been reported in the literature. The limitation of these techniques will also be discussed. This is followed by introduction of some of commercial license plate recognition systems available in the market.

2.1 License Plate Localization

Typically, an image acquisition system will result in an image of the vehicle with the license plate. The license plate may be located anywhere within the image. Prior to the character recognition, the license plate must be located from the background vehicle image. The localization phase extract the license plate from the background image by exploiting certain characteristic features of the license plate that will distinguish it from other regions in the image. Once the region of interest is located based on these features, it is extracted and further analysed to verify if it is a license plate. Some characteristics of the license plate such as the width to height ratio, the number of edges and colour may be applied during the verification process. If the candidate is found to be a valid license plate, it is processed further. Otherwise the search process is repeated on other regions in the image. If no region in the image passes the verification criteria, the picture is classified as having no license plate.

License plate localization process is considered as the most crucial step in a license plate recognition system, which influences the overall accuracy and processing speed of the whole system significantly. Successful extraction of the license plate primarily depends on two factors - quality of the image and the robustness of the extraction algorithm.

The quality of the image to a large extent is determined by the image acquisition module of the system. A well designed image acquisition system will generate high resolution images of vehicle with a clear view of the license plate. If the images are captured at a fixed distance from the vehicle every time, the size of the license plate in the image will not differ much. The extraction algorithm in this case will have to search only for license plates of specific dimensions at specific location in the image. If the camera is focused closer to the vehicle, it will not only ensure that there is just a single license plate in the image but will also eliminate most of the background. When the captured image has a limited amount of background, the probability of having other regions in the image that resemble the license plate is also less. Finally, the angle at which the image is captured is also crucial to the extraction process. If the license plate is too skewed in the image, the features used by the extraction process will have to be independent of license plate orientation.

Even with a well-designed image acquisition hardware that generates images of perfect quality, there are external factors that significantly affect the quality of images. The images of on-road vehicles have different amounts of illumination in them if they are captured under varying lighting conditions. Furthermore, higher speeds of the vehicle can introduce blurring in the license plate image. As a result, the image may require a considerable amount of pre-processing prior to extraction of license plate. Therefore, a well-designed license plate extraction algorithm should take into account all of these factors.

Several methods have been proposed to locate the license plate from the car images and each of them varies in their computational time, complexity and success rates. In the following, some of the general license plate localization methods are discussed and compared. The methods are grouped into several classes according to the type of images that are used to locate the license plate.

2.1.1 Binary/Edge Image-Based License Plate Localization

Edges represent boundaries in an image and are represented by sharp changes in intensity value from one pixel to another. Edge detection extracts the useful information from an image by preserving the basic structural details of an image while removing other information pertaining to colour, intensity and contrast. Many

license plate localization techniques are based on the edge properties of the license plate.

Based on the principle that the change of brightness in the license plate region is more remarkable and more frequent than elsewhere, the edge detected image of a vehicle will be characterized by strong edges near the license plate region. In [2] the vertical edge magnitude of an image was first computed, followed by a horizontal projection and vertical projection of the edge image. Because a license plate has many edge pixels, the projection counts of the license plate area will be high. The license plate was then extracted by searching of high count in the edge map projection histograms. Yang *et al.* [3] also use similar approach based on histogramming and mathematical morphology. The rough location of license plate is detected based on the horizontal projection of the vertical edge image and followed by defining the exact position of license plate by using mathematical morphology. Similar approach using gradient histogram also reported in [4] and [5]. Although this kind of approach is simple to be implemented, but it can hardly be applied to complex image since this is too sensitive to unwanted edges. Other objects in the image such as the radiator region in the front view of the car also show high magnitude of edge count.

In [6], the vertical edges of the car image were extracted by using image enhancement and Sobel operator, followed by removing most of the background and noise edges and searching for the plate region using a rectangle window. This method is based on the fact that license plate area contains rich edge and texture information. If the vertical edges are extracted from the car image and most of the background edges are removed, the plate area can be easily isolated from the whole edge image. However, some threshold values used by this method such as size of the searching rectangle window are relative to the estimated size of the license plate. Therefore this method can only be applied to extract license plate with certain fixed size.

Geometry features of license plate are frequently used to eliminate false candidates during extraction process. A method based on the magnitude of the vertical gradient is presented in [7]. Areas with large local variance in the vertical gradient image are identified as license plate candidates. These candidates are then evaluated based on three geometrical features: the ratio of the width and height, the size and the orientation. In [8], the authors combined the edge statistics and colour analysis to

extract license plate candidates. Incorrect candidates were then eliminated based on the geometry properties of license plate.

2.1.2 Grey Scale Image-Based License Plate Localization

Comelli et al. [9] presented a license plate localization module based on the structure of the Italian license plate, which is rectangular and contains black characters over a white background. By analysing the gradient of the whole image, the algorithm selects the area in the picture that presents the maximum local contrast that (possibly) corresponds to the rectangle that contains the license plate.

The approach used for the plate location in [10] is to scan the image horizontally looking for repeating contrast changes on a scale of 15 pixels and more. This approach used the assumptions that the contrast between the characters and the background of the plate is sufficiently good, that there are at least 3 to 4 characters on a plate and that the characters have a minimum vertical size of about 15 pixels. It should be noted that the particular value of 15 pixels is determined by the resolution of the camera/frame grabber used, the average distance of the vehicle from the camera and the real size of the characters. Similar scan line technique are also reported in [11] and [12], where the grey vehicle image was scanned row by row to locate the license plate region. This technique is based on the fact that the lines where the number plate are located in the image have a clear “signature” which makes it usually possible to distinguish them from other lines in the image. This technique features slow execution times as it needs to scan all rows of the image. It is also too simple to locate license plates in complex scenarios, and moreover, it is not size or distance independent.

In [2], the moving objects were first detected using block-difference method. Each image was tessellated into $M \times N$ blocks and categorized as three kinds: low-contrast, stationary and moving blocks. License plate was extracted by searching for peaks in the projection of edge magnitudes of the moving blocks. However, because not all blocks detected were license plate regions, further verification process was needed to eliminate false candidates.

An adaptive image segmentation technique sliding concentric windows (SCW) is introduced for license plate extraction in [13], [14] and [15]. The SCW method was developed to describe the “local” irregularity in the image. This method utilizes image

statistics such as the standard deviation and the mean value to locate possible license plate region. Two concentric windows A and B of different sizes ($X_1 \times Y_1$ and $X_2 \times Y_2$, respectively) are used to scan the image from left to right and from top to bottom and the mean value and the standard deviation are calculated. If the ratio of the statistical measurements in the two windows exceeds a threshold set by the user, then the central pixel of the concentric windows is considered to belong to a license plate. This process generates a binary image with all the redundant regions from the original image eliminated. The proposed technique was tested on a Pentium IV 3.0 GHz PC with 96.5% success rate.

A novel method for position estimation and tracking of license plates in 3-D from a monocular camera view has been recently reported in [16]. Given an initial estimate, the position of the plate is tracked in the successive video frames. To estimate the object and filter the measurements, the authors developed a new algorithm composed of the probability density propagation of the condensation algorithm [17] and a fine-scale optimization step according to the differential evolution (DE) algorithm [18]. Each sample of the DE-Condensation algorithm is an estimate for the license plate's position in 3D and projected to the image plane by perspective camera model for evaluation. This tracking based method is only suitable to extract license plate from video stream and requires long computation time.

Besides working on spatial image, image transformations were also widely implemented for license plate extraction. Gabor filters have been one of the major tools for texture analysis. To locate the license plate, the system in [19] analysed the image in certain directions and scales by utilizing the Gabor transform instead of traditional edge detection or thresholding approach. The results reported in [19] were encouraging (98% of success rate) when applied to digital images acquired strictly in a fixed and specific angle. However, this method was tested on small sample images, as the method is computationally expensive and slow for images with large analysis.

In the methods that use Hough transform (HT) [20], [21], [22], [23], [24], the edges in the input image are detected first. Then, HT is applied to detect the boundary lines of the license plate. The authors in [20] acknowledge that the execution time of the HT requires too much computation when applied to a binary image with great number of pixels. As a result, the algorithm they used was a combination of the HT

and a contour algorithm, which produced higher accuracy and faster speed so that it could be applied to real-time systems. Kamat and Ganesan in [21] considerably reduced the computational overhead of performing the transformation, by implementing a lookup table, limited angle, limited magnitude and limited area transformation on a window of interest. However, since this HT-based approach is very sensitive to boundary deformation, it only achieved very good results when applied on images with close shots of the vehicle.

A wavelet transform (WT)-based method is used in [25] for the extraction of important contrast features to be used as guides in searching for license plates. In the WT, there are four sub-images (sub-bands), namely LL, LH, HL, and HH, where L and H stand for low frequency and high frequency, respectively. According to [25], a reference line in the first-level LH sub-band (1LH) sub-image exactly above the plate is noticeable. Using the above reference line, a searching mask is created to locate the license plate. The proposed detection method is claimed to be able to locate multiple plates with different orientations in one image and works well in extracting differently illuminated license plates. Nevertheless, the method is unreliable when the distance between the vehicle and the acquisition camera is either too far or too close or the angle of viewpoint is wide.

Symmetry is also used as a feature for car license plate extraction. The generalized symmetry transform (GST) produces continuous features of symmetry between two points by combining the locality constraint and reflectional symmetry. In [26], the authors evaluate the symmetry of plate corners to extract car license plates captured from the arbitrary directions. A scan line decomposition method of calculating GST to achieve considerable reduction of the computational load was proposed. The computational speed of the proposed GST schema is approximately 30 times faster than the conventional GST. However, the effective distance is limited by the algorithm, as a closer view of the plate results to increased processing time. Moreover, this approach is insufficient in the case of slightly rotated or distorted plates. The performance was reported to be around 93% in 330 images.

2.1.3 Colour Image-Based License Plate Localization

Beside binary image and grey level image processing, many colour-based processing methods have also been proposed in the literature for license plate localization. These methods attempt to extract the license plate by using the expected plate appearance such as plate background and text colour in each country. Nevertheless, these methods do not provide a high degree of accuracy in natural scenery since they are highly sensitive to the changing of lighting conditions. In addition, these colours based method are country specific.

Based on the idea that colour combination of a plate background and character is unique and this combination occurs almost only in a license plate region, Shi *et al.* [27] devised a technique to extract Chinese license plate. As the Chinese license plates have specific formats, the authors proposed that all the pixels in the input image should be classified using the hue-lightness-saturation (HLS) colour model into the following 13 categories, i.e. dark blue, blue, light blue, dark yellow, yellow, light yellow, dark black, black, grey black, grey white, white, light white, and other. The license plate region is extracted by checking the histogram of the colour pixels. However, the process of converting input image from the red-green-blue (RGB) value to HLS value requires a lot of time. The sensitivity of colour-based solution under different illumination also causes it not suitable to be applied in natural scenery.

Fuzzy logic has been utilized in locating license plates [28], [29], [30]. The license plates are extracted by making some intuitive rules to describe the license plate and assigning some membership functions for the fuzzy sets “bright”, “dark”, “bright and dark sequence”, and “texture”. Zimic *et al.* [28] defines the following intuitive rules based on human perception for the object “number plate on a car”: 1) bright rectangle area within which there are some dark areas; 2) the border of the plate is bright; 3) it is located approximately in the middle or lower middle part of the image; and 4) the approximate dimension of the plate is 530×120 mm. The first two rules of the concepts of “brightness” or “darkness” are defined as a fuzzy set with trapezoidal membership functions on the interval $[0, 255]$, where 0 represents black, and 255 represents the white colour in grey scale. The input image (768×576 pixels) is first partitioned into sub-images (elements) of size 75×25 pixels and its fitness to the four intuitive rules is computed. The algorithm used 5s on an SG-INDIGO 2

workstation for every sample image with success rate 97%. However, rules 3) and 4) constrain the algorithm to identify license plates within a specific distance only.

In [29], the fuzzy rules of “yellowness” and “texture” were applied in license plate localization. The membership function for “yellowness” was determined by using a histogram-based method. First, the RGB values of pixels taken from a large number of hand-cut license plates were used to construct a frequency table. For each RGB value, the aforementioned table gives the number of times each particular colour occurs in the created set. The membership function was derived from this table by normalizing it so that the most frequently used colour has a membership degree of 1. On the other hand, the membership function for “texture” in a pixel was determined on the basis of the grey scale values of its 8-pixel neighbourhood. Finally, the segmentation is performed using a fuzzy *c*-means clustering algorithm with two clusters (license plate and non-license plate). The system was tested in a huge data set of approximately 10 000 images with only 75.4% of success rate.

The technique developed in [30] focus on extracting Korean license plates. The algorithm first use a colour edge detector which sensitive to only three kinds of edges, black–white, red–white, and green–white to create an initial edge image $\mathbf{E}$. All other colour tones beside white, black, red, and green are eliminated in $\mathbf{E}$. Next, the RGB model of the input colour image is transformed into the hue–saturation–intensity (HSI) space, and the respective $\mathbf{H}$, $\mathbf{S}$, and $\mathbf{I}$ maps of the initial image are generated. Since every map encodes some characteristics about the scene, the entry of any pixel in the map expresses the degree of the pixel possessing the property. This basic idea is used to generate a fuzzy map from a given one (e.g., $\tilde{\mathbf{H}}$ from $\mathbf{H}$). Each of the four fuzzy maps, i.e. $\tilde{\mathbf{H}}$, $\tilde{\mathbf{S}}$, $\tilde{\mathbf{I}}$, and $\tilde{\mathbf{E}}$, serves as a universal set of the complete fuzzy set and is defined with specific distinctive membership functions. The fuzzy maps $\tilde{\mathbf{H}}$, $\tilde{\mathbf{S}}$, $\tilde{\mathbf{I}}$, and $\tilde{\mathbf{E}}$ are integrated into a single map $\tilde{\mathbf{M}}$ and the areas with locally maximal values are defined as regions of interest. The algorithm was tested on a Pentium IV 1.6 GHz PC with success rate of 97.9%.

Mean shift algorithm is a non-parametric statistical method for seeking the main modes of a point sample distribution. Cheng in [31] introduced the mean-shift estimate of the gradient of a density function and the associated iterative procedure of mode seeking. In [32], Comaniciu proposed a practical method that employs mean

shift in the joint spatial-range domain of colour images for discontinuity-preserving filtering and image segmentation. Based on the above two works, a mean shift based colour segmentation [33] was applied to extract candidates that may include license plate regions. According to the statistical analysis performed in [33], license plates adhere to a unique feature combination of rectangularity, aspect ratio, and edge density when comparing to all other non-license plate regions. Based on these three features, a Mahalanobis classifier was then applied on each candidate regions to decide whether these regions represent a license plate or not. The proposed algorithm was merely tested on 57 vehicle images with 97.6 % success rate. The computation time also is considerably long since 6 seconds was needed to process a 324×243 colour images using a Pentium IV 1.60 GHz PC.

2.1.4 Classifier-Based License Plate Localization

Based on the work of Viola and Jones on Adaptive boosting (AdaBoost) algorithm using Haar-like features [34], AdaBoost was applied in license plate detection in [35], [36], [37] and [38]. The AdaBoost algorithm is used to select a small number of weak classifiers from a very large set of weak classifiers and construct a strong classifier, which makes classifications based on the sum of the weighted hypotheses of the selected weak classifiers. Each weak classifier is designed to produce a binary response on a single feature and needs only be slightly more accurate than random guessing. In [35], a total of 100 Haar-like features were applied on sub-regions with size of 45×15 pixels being scanned as expected license plate areas in the original image. In these features, 37 are based on variance, 40 on the x -derivative, 18 on the y -derivative, and 5 on the mean pixel intensity. The classifier was based on a conditional density function. Even though the proposed system achieved an encouraging result with 95.6% of success rate, the authors denote that since the classifier was applied to sub-regions of a specific dimension, the system could not detect plates of different sizes or images acquired from different views or distances without retraining. Similar works that implement Haar-like features in conjunction with cascade classifiers are reported in [36], [37] and [38].

Kim *et al.* [39] proposed a colour texture-based method for detecting license plates in images. The proposed method uses a small window to scan an input image and classify the pixel located at the centre of the window into plate or non-plate

(background) by analysing its colour and texture properties using a Support Vector Machine (SVM). After the classification, a license plate score image is generated where each pixel represents the possibility of the corresponding pixel in the input image being part of a plate region. Then, the license plate bounding boxes within the score image are identified by applying the continuously adaptive mean shift algorithm (CAMShift). This method was originally used to locate faces in a video stream by seeking the modes of flesh probability distribution [40]. The authors replaced the flesh probability with the license plate score which is obtained by performing colour texture analysis on the input image. The authors recognize that this combined method encountered problems when the image is extremely blurred or quite complex in colour.

In addition to using conventional image processing techniques, various computational intelligence architectures were proposed for license plate extraction, such as artificial neural networks (ANNs), genetic programming (GP), and genetic algorithms (GAs). In [41], the authors show that the most intensive computational steps in license plate recognition system could be accomplished by discrete-time cellular neural networks (DTCNNs). The authors first appointed two features, “greyness” and “texture” to each pixel in the image. The ranges for “greyness” and “texture” were determined by a histogram-based method. For the “greyness” feature, the grey values of pixels taken from a large number of exemplary license plates were used to construct a frequency table. Pixels that belong to fixed ranges were then identified by using several DTCNNs, whose templates were constructed by combining the appropriate morphological operations and traditional filter techniques (dilation, Sobel, and Laplacian operators). The results and computation time of the proposed method were not reported by the authors.

A Time Delay Neural Network (TDNN) technique was applied in [42] to find license plate in an image. TDNN is a multilayer feed forward network, whose hidden neurons and output neurons are replicated across time. TDNNs have the ability to represent relationships between events in time and to use them in trading relations for making optimal decisions [43]. Kim *et al.* used two TDNNs as filters for analyzing colour and texture properties of the license plate by examining small windows of vertical and horizontal cross sections of the image. The system obtained an accuracy rate of 97.5% with less than 1 second per image. It should be noted that the fast

execution time despite the complexity of the algorithm is due to the available of rough knowledge of plate location during the extraction process. A similar technique is presented in [44].

Many multilayer feed forward ANNs were employed to locate license plate. In [45], a backpropagation neural network of 24-30-4 topology was trained to classify the colour of a pixel into four distinct classes (green, red, white, and other), according to the classes of Korean license plates. The 24 input nodes correspond to the H, S, and I values of the eight neighbouring pixels in a 3×3 pixel neighbourhood. The license plate location was extracted by using horizontal and vertical colour histogram of green, red and white. The main drawback was reported to be the long execution time although the performance reached 90%. Similar technique was also used in [46] and [47] with different network topology. Whereas in [47], Oz and Ercal utilized the ANN to identify license plates as text (plate) regions in a grey-level image and classified all other pixels as non-text (non-plate).

A convolutional neural network (CNN) verifier that was originally introduced by Garcia and Delakis [48] was applied in a license plate detector [49]. The detector first repeatedly sub-sampled the input image by a factor of 1.2, and the CNN based verifier operated on each pyramid image to find possible text candidates that appear in the license plate regions. Then, the text candidates were fused and labeled to locate the license plates by using pyramid-based techniques and geometrical rules. Two geometrical constraints were designed to remove the false candidates. The width/height ratio of the candidate had to be larger than 1.5, and the size of a license plate had to be larger than 60×25 pixels. The detector achieved a 98% of success rate.

2.2 License Plate Character Segmentation

After successful extraction of the license plate region from the input image, the character string in the extracted image of the license plate is segmented into individual characters. The success of any license plate recognition system depends heavily on the accurate segmentation of the characters since incorrectly segmented characters cannot be recognized even with a powerful character recognition module.

There are several factors need to be considered in the character segmentation process. The edges of the license plate characters may touch each other if the license plate image is blurred due to the speed of the vehicle or poor image resolution. Insufficient or extra illumination of the image can cause the license plate looked faded and making isolation character pixels from the background a non-trivial task.

2.2.1 Binary Image-Based License Plate Character Segmentation

From the number of reported literature that that exploits vertical and horizontal projections of the pixels [27], [11], [20], [42], [22], [45], [50], [36], [51], [47], [52], [4], [53], [54], it is obvious that this method is the most common and simplest for character segmentation. After applying a threshold to covert grey level image to binary image, the image columns or rows are added up to obtain a projection vector. The local minimums of the projection indicate the gaps between the characters are used for segmentation. Although this method is relatively simple, its success rate hinges on the chosen binarization threshold value. If the binarization threshold is not selected properly, varying illumination of the image can result in broken characters or characters that touching each other and make the segmentation task complicated.

Connected component analysis (CCA) is also another technique that has been intensely applied to isolate the characters from the license plate image [55], [29], [46], [8], [56], [7]. CCA scans a binary image and labels its pixels into components based on pixel connectivity. Once all groups of pixels have been determined, each pixel is labeled with a value according to the component to which it was assigned. For license plate image, each character will be labeled with a unique number in ideal case. Binary object measurements such as height, width, area and orientation have also been used to determine whether the labeled component is a valid character. Since this method is applied on binarized image, it also has the drawback same as projection method.

To deal with the seriously degraded plate images, Nomura *et al.* [50] proposed a morphology-based adaptive approach for character segmentation task. At first, binary image is obtained by applying adaptive binarization [57] on colour license plate image, followed by thickening [58] and pruning algorithm [59] to isolate the noise. Based on the projection histogram, the algorithm automatically searches for natural segmentation points, detects fragments and merges fragments that belong to

the same character, as well as separates overlapping or connected characters. Prior knowledge of the maximum quantity of characters was employed in the aforementioned task to decide whether the merging or splitting is necessary. The proposed approach successfully segmented 1005 characters from a test set of 1189 degraded license plate images.

Based on a variational fast marching algorithm, Capar and Gokmen [60] proposed a shape-driven active contour model for character segmentation. First, coarse character locations are determined by an ordinary fast marching technique [61] combined with a gradient and curvature dependent speed function [62]. The exact boundaries of characters are then determined by means of a special fast marching methodology, which depends on gradient, curvature and shape similarity information. Shape similarity statistics were embedded into fast marching method for stopping the evolving front when the front resembles one of the trained shapes. The main advantage of this system is the capability of segmenting and recognizing the objects concurrently. However, segmentation on occluded objects is a limitation of the system, since the system needs the information of interested object shape at startup.

2.2.2 Grey Level Image-Based License Plate Character Segmentation

After many attempts and tests, Draghici [10] concluded that no unique thresholding technique can produce acceptable results in most situations. Therefore, the system proposed in [10] attempt to solve the character segmentation problem on a higher level by combining the results of various histogram thresholding techniques, such as intensity gradient scattergrams or finding a valley in the intensity distribution [63], as well as also using feedback from later stages of the system itself. Obviously this approach incurred high computation time owing to combining results of various techniques.

To overcome the problem associated with using one global binarization threshold, adaptive local binarization methods were implemented in many systems [64], [65], [66]. In these local binarization methods, an image is divided into $m \times n$ blocks, and a threshold is chosen for each block. The binarization in [64] adheres to the above block division technique that implements the dynamic binarization method of Otsu [67]. The expected size of characters in the images was estimated in advance

by utilizing the configurations of the camera and the distance between the camera and vehicles. In [65], the local threshold was computed by subtracting a constant c from the mean gray level in an $m \times n$ window centred in the pixel. Whereas in [66], the authors used the Niblack binarization [68] to calculate the threshold. For all these block-level local binarization methods, the size of the block window needs to be chosen carefully. Smaller size of block may produce desire output result, but at the expense of much longer computation time.

An improved local binarization method that determines a threshold for each character region was proposed in [69]. The projections of the pixels in horizontal and vertical direction are first examined to determine the positions and sizes of rectangles that possibly contain characters in a license plate. Based on the assumption that brightness changes mainly occur in a vertical direction, horizontal pixel accumulation histogram for every character region is checked to find out if a character is split or missing. If one character is divided into two parts, the thresholds are re-determined for these regions.

Based on a previous study in [70], Chang *et al.* [30] combined the HT and CCA to tackle the problem associated with non-uniform illumination. The aspect ratios of connected components are first calculated, followed by eliminating the components whose aspect ratios are outside the prescribed range. Then, an alignment of the remaining components is derived by applying the Hough transform to the centers of gravity of components. The components that disagree with the alignment are removed. The authors also introduced the operators of delete, merge, split and recover into the character segmentation procedure to determine if a component satisfies the structural constraints of a license plate character.

The method in [71] which simultaneously utilizes spatial and temporal information was developed for license plate character segmentation in video sequences. The extraction of characters is first modeled as a Markov Random Field (MRF), where the randomness is used to describe the uncertainty in pixel label assignment. MRF models can be used to incorporate prior contextual information or constraints in a quantitative way. Local spatial/contextual dependences can be utilized to perform binarization [72]. Therefore under the MRF modeling assumption, the extraction of characters can be modeled as the problem of maximizing a posteriori

probability based on given prior knowledge and observations. After that, a GA with a local greedy mutation operator is employed to optimize the objective function and speed up the convergence based on [73]. The authors acknowledged that the performance of the proposed method is still far from the real-time requirement.

In [74], a character segmentation method for noisy low-resolution license plate images was proposed by using Hidden Markov Chains to model a stochastic relation between an input image and the corresponding character segmentation. The segmentation problem was expressed as the maximum *a posteriori* estimation from a set of admissible segmentations. The proposed method exploits a specific prior knowledge such as the predetermined number of characters in the plate and their equal (but unknown) segmented width to reduce the segmentation error. An efficient algorithm to solve the problem based on dynamic programming was derived and tested on 1000 examples with an error rate of 3.3%.

2.3 License Plate Character Recognition

Following the extraction of a license plate and segmentation of the characters, recognition of characters is the last step in a license plate recognition system. Various Optical Character Recognition (OCR) methods have been considered to recognize the segmented characters. The first step in character recognition is the identification of the features that are unique for each character. Once these features are identified, a recognition engine based on these unique features is implemented for identifying the segmented characters.

Similar to localization and character segmentation of a license plate, poor quality of the image produced by the image acquisition system renders the character recognition process difficult. Varying illumination conditions, speed of the vehicle, presence of a noise (e.g., dust, soil materials) on the license plates are the factors that have significant impact on the accuracy of a character recognition process.

2.3.1 Classifier-Based License Plate Character Recognition

Hidden Markov Models (HMMs) were employed in [20] and [65] for character recognition. Duan *et al.* [20] used the ratio of foreground pixels in a window as the feature of the model. A window with size 9x9 was used to scan the image from

top left to bottom right to obtain the feature vector. The model was then trained to classify a character image into one of 36 classes. Similar approach was also reported in [75] with complex procedure of pre-processing and parameterization for the HMMs. The authors reported that the width of the plate in the image after rescaling lies between 25% and 75% of the original image width (between 200 and 600 pixels). This reveals the necessity for good character analysis when implementing HMMs, which poses a restriction on the effective distance of the license plate recognition system.

A Support Vector Machine (SVM) system [42] was designed to recognize Korean license plate characters. Each extracted character image was first histogram equalized and uniformly divided into 8x8 regions yielding 64 dimensional mesh vectors. Four SVM-based character recognizers were applied to recognize upper characters, upper numerals, lower characters, and lower numerals on the plate.

In order to achieve robust and high recognition performance, Pan *et al.* [76] proposed a two-stage hybrid recognition system that combines statistical and structural recognition methods. Before the recognition, skew images of car plates were corrected and normalized. In the first recognition stage, four statistical sub-classifiers (SC1, SC2, SC3, and SC4) independently recognize the input character, and the recognition results are combined using the Bayes method [83]. Sub-classifier SC1 uses the zoning density [77], SC2 uses the vertical projections, SC3 calculates the contour profile, and SC4 counts line segments in each row and column. Finally, if the output of the first (statistical) stage contains characters that belong to prescribed sets of similar characters, the second (structural) stage is initiated as a complement to the first. Structure features are obtained and then fed into a decision tree classifier. The proposed system works well on vehicles with running speed under 100 km/h.

2.3.2 Artificial Neural Network-Based License Plate Character Recognition

Multilayered feed-forward neural networks are found in many works of license plate character recognition [78], [29], [41], [47], [4], [79], [66], [80], [81]. Training a neural network involves computing weights so as to get an output response to the input within an error limit. The input and target vectors make up a training pair.

The classical training method for feed-forward neural networks is error backpropagation [82]. A backpropagation network algorithm includes the following steps:

1. Select the first training pair and apply the input vector to the net.
2. Calculate the net output.
3. Compare the actual output with the corresponding target and find the error.
4. Modify the weights so as to reduce the error.

The above steps are repeated until the error is within the accepted limits. The high accuracy of these methods comes at the cost of the training times. The network has to be trained for many training cycles to achieve good performance. The training process is time consuming and it is also not certain that the network will learn the training sample successfully. Moreover, the number of hidden layers, as well as the number of respective neurons, has to be defined after a trial-and-error procedure [67].

A multilayered perceptron (MLP) architecture that contains 24 input, 15 hidden, and 36 output neurons was designed in [29] to recognize 36 characters of the Latin alphabet. Unlike the approach presented in [78], where the classification was done on the basis of binary image input, the input neurons in [29] were fed with 24 features previously generated from a DTCNN. Meanwhile in [83], a three-layered MLP was specifically trained to solve problematic pairs (e.g., I/l, B/8, and O/0) or for the classification of border parts and legislation stamps that appear in license plates. During the training procedure, the problematic samples were trained more often, and special characters for borders and legislation stamps were added to the training sets.

In an attempt to improve classical multi-layered feed-forward networks, a non-standard feed-forward neural network based on the Adaptive Resonance Theory (ART) introduced by Grossberg [84] was examined in [75]. The proposed network is an unsupervised learning network, which dynamically creates the number of nodes in the hidden layer and guarantees the convergence to learning. The results reported in [75] show a slight improvement in the recognition performance of the ART model (95%) in a relatively small data set.

Chang *et al.* in [30] tackled the problems of noisy, deformed, broken, or incomplete characters by implementing self-organized neural networks based on Kohonen's self-organized feature maps. Their approach focuses on recognition accuracy at the expense of increased complexity and execution speed. To overcome misclassification of similar character pairs, such as (8/B), (0/D), and (O/D), an ambiguity set that contains these characters was defined. For each character in the set, the non-ambiguous parts of the character were specified. During character recognition, once an unknown character was classified as one of the characters in the ambiguity set, an additional minor comparison between the unknown character and the classified character was performed. The comparison then focused only on the non-ambiguous parts of the character.

Probabilistic neural networks (PNNs) first introduced by Specht [85] are neural-network implementation of kernel discriminate analysis. In both [13] and [86], two PNNs, i.e., one for alphabet recognition and the other for number recognition, were designed for OCR task. The PNNs were trained and tested in noisy, tilted, and degraded patterns with very encouraging recognition rates (over 90%). Additionally, in [87], the authors reported an impressive recognition rate that reaches 99.5%. PNN can be designed in a fraction of the time it takes to train standard feed-forward networks, as the hidden-layer neurons are defined by the number of the training patterns and are only trained once [88]. However this feature comes at the cost of larger memory requirements and slightly slower execution speed compared to conventional neural networks [40].

2.3.3 Pattern/Template Matching-Based License Plate Character Recognition

Template matching is a straightforward and widely used method for character recognition or classification [9], [89], [36], [51], [12]. This technique is suitable for recognition of single-font, non-rotated and fixed size license plate characters. For a standard template matching algorithm, a database that contains template images of each character that become the references for the comparison will be first created. The size of the candidate images is then normalized to the same with the template images in the database. After that, a continuous search process is done to find the similar

images in the database. The template image with highest correlation value with the candidate image is selected as the most matched image.

Although template matching method is preferably applied on binary image, some template matching method based on grey level [90] and color image [45] have also been reported. However, template matching methods by comparing the character pixels directly often causes misrecognition if the intensity value and the color of the objects vary under different illumination conditions. Hence, other features besides the character images themselves have also been used for template matching. For instance, using the comparison between the Hotelling transformed counterparts and the Hotelling transformed prototypes [53], using the linear sum of intensity projection and intensity variance of projection direction [42], and using histograms as features [54].

2.4 Commercial License Plate Recognition System

Beside various license plate recognition techniques that have been reported in literature, many commercial license plate recognition systems also available in the market. They can be broadly divided into two categories: fixed license plate recognition system and mobile license plate recognition system.

2.4.1 Fixed License Plate Recognition System

Majority of the license plate recognition system available in the market are stationary system installed at the gate or entrance of a building. Some of the systems include Fixed Plate Hunter-900 by ELSAG North America [91], AutoPlate by PIPS Technology [92], CarDetector by Vigilant Video [93], and SeeCar by Hi-Tech Solutions [94].

Fixed license plate recognition system is usually used for parking management and access control applications. Figure 2.1 shows an example of a fixed license plate recognition system installed at the entrance of a building. A vehicle image captured by the camera is transferred to the license plate recognition software running on a PC (Figure 2.1 (b)). The identified license plate information is sent to an existing parking management access control system to automatically permit entry/exit to authorized members. Captured information may be used to calculate overall parking time,

provide proof of parking in case of a lost ticket, determine traffic patterns in and out of an area, and automatic vehicle tolling systems.

(a) Fixed license plate recognition system installed at the entrance

(b) License plate recognition software running on a PC

Figure 2.1: AutoPlate fixed license plate recognition system by PIPS Technology [92]

A fixed license plate recognition system is usually equipped with a powerful processing unit such as PC or workstation to be capable of recognizing the license plate in real-time. For instance, the minimum hardware requirement of AutoPlate system [92] is a PC with processor speed 1GHz running on Microsoft Windows XP. Therefore, the system which is bulky and requires high power consumption is inflexible to be carried around for different type of mobile recognition applications. It is only suitable for application where the size and power consumption are not critical design issues. The cost of setting up this kind of system is high due to its high computation demands such as high performance processing unit, high-end camera and additional illumination. If the distance between the camera and the host computer is long, high speed communication channel such as Ethernet is also needed. It also requires complex installation and cannot be integrated easily into existing surveillance system.

A license plate recognition system installed at a fixed location has the advantage of knowing the environmental parameters in the design stage of the system, such as lighting conditions, plate location, expected plate size and etc. These parameters can be obtained from the site where the camera is mounted prior the deployment of the system. Since the provisional plate location is also known, the license plate reading procedure for a stationary system is expected to be fast and

highly accurate. However, it is not suitable for outdoor applications where complex environment exist. The system may fail to identify the vehicles if any calibration of the system is not done prior the recognition process.

Beside fixed license plate recognition system, many vendors also provide mobile license plate recognition systems for different kind of vehicle recognition applications. They can be broadly classified into three different groups: notebook PC-based, handheld device-based and vehicle mounted-based.

2.4.2 Notebook PC-Based Mobile License Plate Recognition System

(a) Motorola MW810 Mobile Workstation [95]

(b) AlertVU by L-3 Mobile Vision [118]

Figure 2.2: License plate recognition system running on mobile computing device

The most common type of mobile license plate recognition system available in the market is shown in Figure 2.2. This type of system is claimed to be portable by simply switching the operating platform from desktop PC or workstation in a normal fixed license plate recognition system to notebook PC or other means of mobile computing device. The working principle of this kind of system is roughly the same as fixed license plate system with the additional advantage of mobility.

Nevertheless, this kind of system suffers the same shortcomings of typical fixed license plate recognition system. For instance, the Motorola MW810 Mobile Workstation [95] is equipped with an Intel Core2 Duo processor. The cost of setting up this kind of system is comparable with or even higher than fixed license plate recognition system. Therefore, this kind of system is not the ultimate solution for

application that requires a low cost, low power and highly portable license plate recognition system with real-time performance.

2.4.3 Handheld Device-Based Mobile License Plate Recognition System

Instead of providing license plate recognition system with hardware equipment, some vendors offer license plate recognition software running on handheld devices such as smart phones and industrial handheld units equipped with camera module. It is designed to facilitate parking attendant or enforcement officer for parking management, scofflaw identification and others similar applications. Two examples are shown in Figure 2.3.

Figure 2.3: Handheld license plate recognition system

In order to correctly identify the vehicle using this kind of system, it requires the user to point the handheld unit towards the license plate location to scans and extracts the information. Any slight variation of the camera position may cause the system fail to recognize the license plate. Thus, this kind of system performs poorly when extracting license plate from complex environment. Moreover, smart phone and other handheld devices are not the optimized platform for license plate recognition system application. These devices are not designed specifically for recognition applications. They may not able to achieve real time performance requirement which is essential for many license plate recognition applications. The cost of utilizing these general purpose devices for large scale deployment of license plate recognition system is also high.

2.4.4 Vehicle Mounted-Based Mobile License Plate Recognition System

This type of system is specially designed for law enforcement officer to scan the license plate from the patrol car during daily patrol shift. It is permanently or temporarily mounted to a vehicle and comprised of a suite of components, including camera, dedicated license plate recognition processing unit and a display interface such as a notebook PC. The major vendors of this kind of system include PIPS Technology [96], Vigilant Video [97] and PlateScan [98]. Figure 2.4 illustrates a typical setting of such system.

Figure 2.4: Vehicle mounted license plate recognition system [99]

The camera mounted on the trunk of the police car captures the images of the vehicles while the car is patrolling on the street. These images are sent to the processing unit that is usually installed in the trunk of the car for recognition process. The identified plate information is sent and displayed to the user on the notebook PC that is placed beside the driver. The notebook PC which is also wireless connected to a database system can help the officer to identify any vehicles that are of interest to the authorities. If the system detects a match, it can alert the officer to take further actions.

Although the system is claimed to be very accurate, this come at the cost of expensive camera module and high performance processing unit. For instance, the mobile system provided by PIPS Technology is running on an Intel Dual Core CPU and the whole cost of such system is about US\$21,000 [96]. The size of the dedicated processing unit is also heavy and is purposely designed for mounting in the trunk of the vehicle. The camera, processing unit and the notebook PC are highly integrated modules and cannot be tailored easily for other type of application other than law enforcement.

In general, the high cost of this type of license plate recognition system is mainly due to its high NRE (Non-Recurring Engineering) cost. A lot of testing needs to be done to ensure the reliability of the product in wide variety of environment. Since police equipment is a small and difficult market, this reason also contributes to high unit cost of the product in order for the manufacturer to recoup the development and support costs.

2.5 Summary

In summary, existing techniques reported in literature are complex and time consuming, especially those using classifier and artificial neural network for the recognition task. Even though many of these techniques achieve high accuracy rate with more than 95%, it comes at the expense of high computing resources. As illustrated in previous sections of this chapter, these complex methods require high computing resources such as PC or workstation to attain short computation time. Therefore, the cost of applying these techniques is high and they are only suitable for certain license plate recognition applications where mobility does not appear to be the major design consideration.

Beside fixed license plate recognition system, many commercial mobile license plate recognition systems are also available in the market due to their highly demand in various law enforcement applications. Law enforcement professionals use these mobile systems in stolen vehicle recovery, speed enforcement and parking management. However, many kinds of these mobile systems which simply switching the computing platform from desktop PC to notebook PC are still facing the shortcoming of high cost and bulky implementations. Another type of the system that

use general handheld device has limited performance due to the hardware constraint. These handheld devices are not explicitly design for license plat recognition applications. In order to achieve good performance, the images captured by these handheld devices must only include exactly the vehicle license plate region without unnecessary outdoor environment information. All these mobile systems available in the market are not the optimal solution for low cost portable license plate recognition system.

These reasons have motivated the need to devise a real time capable portable license plate recognition system that is low cost and easy to use. A license plate recognition system based on a smart camera vision system is proposed in this project. In contrast to the PC-based system, smart camera architecture consists of a camera module and is coupled with an embedded processing unit for image processing purpose. It should be generally less expensive and unlike PC-based solutions, must be compact and real time capable.

Compared with PC-based solution, it would be ideal for a smart camera based system to be battery powered to facilitate remote deployments. There exists a need to develop suitable techniques that can lead to highly portable license plate recognition system. The proposed system are based on embedded platform which has limiting computing resources compared with PC or workstation. Hence, hardware efficient techniques in various stages of a license plate recognition system are required in order to realize a low cost and portable system with capable of real time performance. The following chapters in this thesis describe the proposed license plate recognition algorithms in an attempt to achieve this goal.

Chapter 3

License Plate Localization

License plate localization process is the first step as well as the most decisive step in a license plate recognition system. During the process, the license plate of the vehicle is extracted from the background image by exploiting certain characteristic features of the license plate that will distinguish it from other regions in the image. An efficient license plate localization algorithm can help to reduce the overall processing time of the system. The successful of the whole system is also highly hinged on the outcome of license plate localization process. In this chapter, the image processing functions used in the proposed algorithm are first introduced and followed by the detailed descriptions of the algorithm implementation.

3.1 Image Processing Functions

Edge detection and binary morphological are two main image processing functions utilized in the proposed license plate localization process. The working principles of these two functions are presented in this section.

3.1.1 Convolution Operation

Figure 3.1: Convolution operation on a 3x3 image region

The underlying operation of many image processing functions is convolution operation. Image convolution operation uses a wide variety of *masks*, also known as *kernels*, to calculate different output results. For example, certain masks yield smoothing, while others yield blurring or edge detection [59].

Figure 3.1 shows an image example, $z(x,y)$ and a mask, $m(x,y)$ to illustrate convolution process of a 3x3 region. The convolution operation, which is also widely known as sliding window operation, is performed by sliding the mask over the image. The resultant value R at the center of the mask, $(0,0)$ is calculated by sum of product of the mask coefficients and the corresponding image area spanned by the mask. Mathematically, it can be represented by the following equation:

$$R = (m_{-1,-1} \times z_{-1,-1}) + (m_{-1,0} \times z_{-1,0}) + \dots + (m_{1,0} \times z_{1,0}) + (m_{1,1} \times z_{1,1})$$

Equation (3.1)

3.1.2 Edge Detection Convolution Filters

Edges in an image are the area where strong intensity contrasts are present due to a sudden variation in the intensity from one pixel to the next. Detecting the edges of an image can significantly reduce the amount of processing data by filtering out the useless information while preserving the important structural properties of an image. The major edge detection methods can be grouped into two different categories, namely gradient-based and zero-crossing-based [59]. The gradient-based methods identify the edges by looking for the maximum and minimum in the first-order derivative of the image. The zero-crossing-based methods search for zero-crossings in the second-order derivative of the image to detect edges.

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

(a) Horizontal kernel

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

(b) Vertical Kernel

Figure 3.2: Sobel edge detection kernels

Sobel, Prewitt, and Robert are several well-known convolution kernels to compute the gradient of an image. The Sobel operator kernels that are used in this project are shown in Figure 3.2. These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations [59]. By applying the kernels separately to the input image, the gradient of an image $f(x,y)$ at location (x, y) is defines as the vector,

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad \text{Equation (3.2)}$$

where G_x and G_y are the gradient vectors at direction x and y respectively. The magnitude of the gradient is defined as:

$$|\nabla f| = \sqrt{G_x^2 + G_y^2} \quad \text{Equation (3.3)}$$

In practice, an approximate magnitude is computed using:

$$|\nabla f| \approx |G_x| + |G_y| \quad \text{Equation (3.4)}$$

By selecting an appropriate threshold T , the gradient image is converted to binary edge image BW_Edge by applying the following formula:

$$BW_EDGE(x, y) = \begin{cases} 0 \text{ (black)}, & |\nabla f|(x, y) < T \\ 1 \text{ (white)}, & |\nabla f|(x, y) \geq T \end{cases} \quad \text{Equation (3.5)}$$

3.1.3 Binary Morphological Operations

Mathematical morphology is a powerful tool for image analysis based on neighbourhood operations. Its common usages include noise removal, image enhancement and image segmentation [59]. The two basic operations in mathematical morphology are erosion and dilation. Both of these operations take two pieces of data as input: an image A to be dilated or eroded, and a structuring element B . Dilation is often used to fill the gaps or holes while erosion is used to remove irrelevant details from an image. The extent to which an object is dilated or eroded is determined by the shape and dimension of the structuring element.

[illegible]
$$\begin{array}{ccc} & & 1 \\ & \boxed{1} & \\ 1 & & \end{array}$$

The structuring element translated to these locations does not overlap any 1-valued pixels in the original image.

When the origin is translated to the “■” location, the structuring element overlaps 1-valued pixels in the original image.

[illegible]

(d) Output image

Figure 3.3: Dilation Operation [59]

46

value of every pixel covered by the structuring element is 1 (i.e., it does not overlap with any 0-valued pixels).

```

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

```

(a) Original image A with rectangular object

```

  1
 1
 1

```

(b) Structuring element B with the origin in a dark border

Output is zero in these locations because the structuring element overlaps the background.

Output is one here because the structuring element fits entirely within the foreground

(c) Structuring element translated to several locations on the image

```

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

```

(d) Output image

Figure 3.4: Erosion Operation [59]

Dilation and erosion operations are usually combined to form more useful functions. The morphological opening of A by B , denoted by $A \circ B$ is simply erosion of A by B , followed by dilation of the result by B :

$$\text{Opening Operation: } A \circ B = (A \ominus B) \oplus B \quad \text{Equation (3.6)}$$

The morphological closing of A by B , denoted by $A \bullet B$, is a dilation followed by erosion:

$$\text{Closing Operation: } A \bullet B = (A \oplus B) \ominus B \quad \text{Equation (3.7)}$$

Opening can remove the small regions from an image that cannot contain the structuring element, while preserving the shape and size of larger objects in the image. Closing can be used to fill the black holes in the image that does not fit within the structuring element and smooth the boundary of objects.

3.2 Proposed Method for License Plate Localization

In this project, a novel algorithm based on the edge information of the vehicle license plate was proposed for the license plate localization process. The algorithm utilizes the vertical edges and rectangular shape attribute of license plate to locate its rough location in an image. The proposed technique is efficient and straightforward to be implemented on hardware platform. Flowchart in Figure 3.5 illustrates the processing steps of the proposed algorithm.

Figure 3.5: License plate localization process

3.2.1 Vertical Edge Detection Using Sobel Filter

A license plate either consists of dark characters with light background or light characters with dark background. One of the prominent features of license plate region is its edges that exist due to the high contrast between the license plate characters and its background. This is a good feature that can be used to locate the license plate in an image. In this project, Sobel edge detector is used to extract the edge information from the vehicle image. Sobel edge detection is chosen over other

techniques such as Laplacian and Canny edge detector because of its simplicity of implementation.

Figure 3.6: Vehicle image and its vertical and horizontal edges

Vehicle image in Figure 3.6 (a) is used to illustrate the proposed technique for the license plate localization in this chapter. First, the Sobel edge detection method was used to find the gradient of the vehicle image. The vertical gradient of the vehicle image in Figure 3.6 (b) was obtained by convolving the image with a 3x3 vertical Sobel mask. The gradient image was then converted to the binary image in Figure 3.6 (c) according to Equation (3.5) by applying a threshold value. If a pixel's gradient value is greater than the threshold, the pixel value was set to 1 (white). Inversely, if it is smaller, the pixel value was set to 0 (black).

The vertical edges are chosen over horizontal edges because vertical edges of a vehicle image contain more relevant information regarding the license plate. Compared with horizontal edges of the vehicle image in Figure 3.6 (d), the vertical

edge detected image distinctly represents the edges of the license plate and has far less noise. Thus by extracting only vertical edges, most of the non-license plate regions which predominantly containing horizontal edges can easily be eliminated.

3.2.2 Connecting Vertical Edges Using Morphological Operations

As illustrated in Figure 3.6 (c), a vehicle image usually contains much noise from small irrelevant edges, especially if it is captured in outdoor environment. Moreover, the vertical edges are scattered over the license plate region and rarely connected. To remove noisy area and extract the exact license plate region, the vertical edges need to be connected to form useful license plate location information. One of the straightforward ways to achieve this objective is using morphological operations.

The following morphological-based method used to locate potential license plate candidates are adapted from similar technique reported in [100]. Let $SE_{m,n}$ denotes a structuring element (SE) with size $m \times n$ (height $\times$ width). A sequence of morphological operations using structuring elements $SE_{m,n}$ was applied on the vertical edge image to locate rectangular region with roughly the size of license plate. The sizes of the SEs have been determined by testing a large number of sample images in order to find the sizes that best fit most images. The series of morphological operation are organised in following steps:

Step 1: Apply a closing operation with $SE_{3,21}$ (where the width of the SE is the maximum space between characters) on the vertical edge image in Figure 3.7 (a) to fill the gaps between vertical edges of license plate characters. The initially scattered vertical edges are now connected in resultant image as shown in Figure 3.7 (b).

Step 2: Apply an opening operation with $SE_{11,3}$ (where the height of the SE is the minimum character height) on Figure 3.7 (b) to eliminate the region whose height is less than the minimum character height. Some small regions have been removed in the resultant image Figure 3.7 (c).

Step 3: Apply another opening operation with $SE_{41,3}$ to obtain regions with height taller than anticipated maximum license plate. The obtained output as shown in Figure 3.7 (d) contains the blobs taller than the expected license plate height.

Step 4: Subtract Figure 3.7 (d) from Figure 3.7 (c) to retain regions less than (or equal to) maximum license plate height. The resultant image in Figure 3.7 (e) contains only potential license plate regions with some small noise blobs.

Step 5: Apply an opening operation with $SE_{11,31}$ on Figure 3.7 (e) to eliminate the regions with width less than minimum width of license plate. The connected pixel groups in Figure 3.7 (f) are the resultant license plate candidates.

Step 6: Apply connected components labelling algorithm on Figure 3.7 (f) to obtain connected pixel groups of the image. The number of connected components along with the width and height of each connected regions are obtained after this process. The red rectangles are the minimum bounding boxes that enclose each region.

Occasionally, a car license plate is separated into several adjacent connected components due to the absent of vertical edges in some characters. Its vertical edges are not combined as a single region even after applying the morphological operations. It is worth to mention that these are very rare cases and only happen on license plate with consecutive characters without clear vertical edges like characters '7', 'A' and 'Z'. In order to crop a complete license plate region for these cases, a merging operation is performed to incorporate these separated regions together. During the merging operation, two regions can be combined together if their distance is bounded in a certain range and they have almost the same height. Such merging operation is applied repeatedly until there are no regions that can be merged. Figure 3.8 (g) is the resultant image after applying merging operation on Figure 3.7 (f).

Figure 3.8 (h) shows the final extracted regions from the original image after applying the morphological operations. Each region was labeled with a number, started with Candidate 1 and ended with Candidate 4. Even though license plate region with dense concentration of vertical edges were obtained after the process, other areas such as radiator grillers and headlights in front of the car with this similar characteristic were also acquired. To locate the license plate more accurately, license plate candidates form by these connected areas were continued with next identification process to eliminate non-plate regions.

(a) Vertical edges

(b) Edges merged into rectangle

(c) Regions with height less than license plate were eliminated

(d) Regions higher than expected license plate height

(e) Regions with height taller than license plate were eliminated

(f) Regions with width shorter than license plate were eliminated

Figure 3.7: Series of morphological operations to locate license plate region - Part I

Figure 3.8: Series of morphological operations to locate license plate region - Part II

3.3 Experimental Results of License Plate Localization Process

The proposed license plate localization method was implemented in MATLAB environment to evaluate its efficiency. The Image Processing Toolbox of MATLAB [101] provides many useful image processing functions such as edge detection, morphological operations and Hough Transform. Users can use these functions to quickly evaluate the concept of their algorithms before translating it to hardware implementation. It greatly saves the developers from the hassle of developing these image processing functions from scratch.

The proposed license plate localization method was tested by using 200 trial images with size of 480 x 640 (height x width). These images are comprised of indoor and outdoor images of stationary cars and were captured under good lighting conditions with a clear view of the license plate. The license plate localization method was considered success if it could include license plate in the image as one of its candidates. The proposed method achieved a success rate of 97% with an average of 0.2s processing time on a PC with a 2.4GHz Intel Core 2 CPU and 3GB memory. Figure 3.9 to Figure 3.16 illustrate some of the images used for the testing. Each example is presented with its original grey level image, vertical edge detected image, result after applying morphological operations on vertical edges and the final detected license plate candidates. These examples successfully demonstrate the capability of

the proposed license plate localization method to locate license plates in either indoor or outdoor images. Failure of localization step was found to occur primarily due the sizes of the license plates in the images are too small and the vertical edges of license plates are not evident enough to be regarded as candidate.

The proposed method usually detect the license plate as one of the candidates along with other areas with similar characteristic as license plate. The radiator of the vehicle which is located above the license plate is frequently included as one of the license plate candidates. This is due to the radiator of the car has strong vertical edges and possess height and width similar with the license plate. For instance, all radiators of the cars in Example 1 to Example 3 were detected as license plate candidates. Beside the radiator, the headlight of the car is also another common region that is usually picked as license plate candidate by the proposed license plate localization method. This is illustrated by the detected license plate candidates in Example 4 to Example 5.

From the experimental results, it is also noticeable that the number of detected license plate candidates in an image is highly depended on the image capturing environment. As compared with outdoor images such as Example 6 and Example 7, vehicle images captured in indoor environment such as Example 8 and Example 9 have significant less number of license plate candidates. In the outdoor images, license plate candidates are mainly contributed by the edges of the trees or their reflection on the windscreen of the vehicles. These candidates will contribute more processing time in the next license plate identification process.

Since the proposed license plate localization method is based on the feature of dense concentration of vertical edges in license plate region, it is expected that other alphanumeric areas in the image with similar feature will also be included as one of the candidates. In Example 10 and Example 11, the speed limit labels and maximum passenger capacity labels for lorry were detected as the license plate candidates. These are the area with numeric letters in the image. While in Example 12, both the vehicle manufacture's logo and its company's logo are the candidates. These alphanumeric candidates are difficult to be separated from license plate region during the localization process. Therefore, the license plate identification process which will

be described in next chapter is applied to identify true license plate among these candidates.

Example 1

(a) Grey level vehicle image

(b) Vertical edges detected image

(c) Resultant image after applying a series of morphological operations on vertical edges

(d) Detected license plate candidates

Example 2

(a) Grey level vehicle image

(b) Vertical edges detected image

Figure 3.9: Samples of images used in license plate localization experiment - Part I

Example 2 (continued)

(c) Resultant image after applying a series of morphological operations on vertical edges

(d) Detected license plate candidates

Example 3

(a) Grey level vehicle image

(b) Vertical edges detected image

(c) Resultant image after applying a series of morphological operations on vertical edges

(d) Detected license plate candidates

Figure 3.10: Samples of images used in license plate localization experiment - Part II

Example 4

(a) Grey level vehicle image

(b) Vertical edges detected image

(c) Resultant image after applying a series of morphological operations on vertical edges

(d) Detected license plate candidates

Example 5

(a) Grey level vehicle image

(b) Vertical edges detected image

Figure 3.11: Samples of images used in license plate localization experiment - Part III

Example 5 (continued)

(c) Resultant image after applying a series of morphological operations on vertical edges

(d) Detected license plate candidates

Example 6

(a) Grey level vehicle image

(b) Vertical edges detected image

(c) Resultant image after applying a series of morphological operations on vertical edges

(d) Detected license plate candidates

Figure 3.12: Samples of images used in license plate localization experiment - Part IV

Example 7

(a) Grey level vehicle image

(b) Vertical edges detected image

(c) Resultant image after applying a series of morphological operations on vertical edges

(d) Detected license plate candidates

Example 8

(a) Grey level vehicle image

(b) Vertical edges detected image

Figure 3.13: Samples of images used in license plate localization experiment - Part V

Example 8 (continued)

(c) Resultant image after applying a series of morphological operations on vertical edges

(d) Detected license plate candidates

Example 9

(a) Grey level vehicle image

(b) Vertical edges detected image

(c) Resultant image after applying a series of morphological operations on vertical edges

(d) Detected license plate candidates

Figure 3.14: Samples of images used in license plate localization experiment - Part VI

Example 10

(a) Grey level vehicle image

(b) Vertical edges detected image

(c) Resultant image after applying a series of morphological operations on vertical edges

(d) Detected license plate candidates

Example 11

(a) Grey level vehicle image

(b) Vertical edges detected image

Figure 3.15: Samples of images used in license plate localization experiment - Part VII

Example 11 (continued)

(c) Resultant image after applying a series of morphological operations on vertical edges

(d) Detected license plate candidates

Example 12

(a) Grey level vehicle image

(b) Vertical edges detected image

(c) Resultant image after applying a series of morphological operations on vertical edges

(d) Detected license plate candidates

Figure 3.16: Samples of images used in license plate localization experiment - Part VIII

Chapter 4

License Plate Identification

Many fake candidates, such as noisy environmental areas, headlights and radiator grillers of vehicles have abundance texture and line information with similar properties as true license plate. This chapter explains the identification process to eliminate candidates that are not license plate. By using the characteristics of the linear lines that exist in the license plate region, non-plate candidates are discarded after this process. This chapter starts with the introduction of Hough Transform based line detection that has been applied extensively during the identification process. It continues with the explanation of different stages of the identification process as shown in the following Figure 4.1.

Figure 4.1: Flow chart of license plate identification process

4.1 Hough Transform-Based Line Detection

Hough Transform is a well-known technique to detect straight lines in an image. Originally developed to recognize lines [102], Hough Transform has later been

generalized to cover arbitrary shapes [103], [104]. The following section describes how the Hough transform is able to detect straight lines.

Consider a general equation of a straight line in slope-intercept form $y_i = ax_i + b$ (Figure 4.2 (a)). Given this characterization of a line, there are infinite numbers of lines that pass through any given point (x_i, y_i) for varying values of a and b . By iterating through fixed values of a , the values of b can solve by equation $b = -ax_i + y_i$. This equation represents a single line in the a - b plane (also called parameter space) for a fixed point (x_i, y_i) in the image (Figure 4.2 (b)). All the points lying on the same line in the x - y plane have lines in the parameter space that intersect at the same point (a', b') .

Figure 4.2: Linear line representations in xy-plane and parameter space

However, this method for representation of a linear line is not very stable. As lines get more and more vertical, the magnitudes of a and b grow towards infinity. A more useful representation of a line is its normal form:

$$x \cos \theta + y \sin \theta = \rho \quad \text{Equation (4.1)}$$

The parameters θ and ρ is the angle of the line and the distance from the line to the origin respectively as shown in Figure 4.3 (a). The intersect point (ρ', θ') corresponds to the line that passes through both (x_i, y_i) and (x_j, y_j) covering the Hough space as illustrated in Figure 4.3 (b).

To determine the areas in the Hough space where most lines intersect, an accumulator cells covering the Hough space as illustrated in Figure 4.3 (c) is used. The range of θ values is $\pm 90^\circ$, and the range of the ρ -axis is $\pm \sqrt{D}$, where D is the distance between corners in the image. For every non-background point (x_k, y_k) in the

image plane, the θ is set to each of the allowed subdivision value on the θ -axis and the ρ values is solved by using the equation $x\cos\theta + y\sin\theta = \rho$. The corresponding (ρ, θ) in the accumulator cell is then incremented. The value of the bin $A(i,j)$ in the accumulator represents the number of the points in the xy -plane that lie on the line $x\cos\theta_j + y\sin\theta_j = \rho_i$ in the parameter space.

Figure 4.3: Hough Transform parameter space. (a) (ρ, θ) parameterization of line in the xy -plane. (b) Sinusoidal curves in the ρ - θ plane (c) Division of the ρ - θ plane into accumulator plane

Figure 4.4: Line detection by using Hough Transform

Figure 4.4 illustrates an example of line detection of a parallelogram using Hough Transform. The intensity of a pixel in Hough Transform plot corresponds to the number of votes it received. Hence, the brighter pixels represent coordinates in Hough space that got the most votes, and thus are more likely to generate lines that fit

many points. The lines in the image can be found by setting some threshold for the accumulator and interpret all values above the threshold as a line. The white square in the plot represent the detected peaks that correspond to the 4 lines of the parallelogram.

Compared with other methods like edge-linking and mask filtering, line detection using Hough Transform is robust against noise, break in the edge from non-uniform illumination and other effects that introduce spurious intensity discontinuities. Therefore, Hough Transform is chosen as the line detection method for license plate detection system of this project.

4.2 Removing Undesirable Area of License Plate Candidate

Sometimes, the rectangle that enclosing the license plate may include superfluous region other than the alphanumeric characters due to the vertical lines that contributed by the license plate frame. One of the examples is Candidate 1 obtained from license plate localization process in previous chapter. The license plate identification rules that will be described in Section 4.3 onwards are established based on license plate region that contains only alphanumeric characters. Hence, it is necessary to eliminate any undesirable area other than characters in license plate region before proceeding to the next phase of candidate verification step.

Appendix A.1 shows the characters used in Singapore vehicle license plate, along with the vertical lines and horizontal lines found in each character. From these images, it can be observed that most of the vertical and horizontal lines of the characters exist in pair form. Given a license plate region, undesirable area such as license plate frame can be removed by finding the area with cluster of horizontal and vertical line pairs.

Taking Candidate 1 as an example, Figure 4.5 (a) and (b) show its original image and vertical edge detected image respectively. Hough Transform which is described in previous Section 4.1 was used to detect straight lines of Candidate 1. The green lines are the detected vertical lines after applying Hough Transform on the vertical edge image. The lengths of the lines were chosen such that lines that exceed license plate characters height would not be detected. Next, for every two vertical lines with similar length and close to each other, they were grouped to form a pair of

lines. The red rectangle boxes in Figure 4.5 (b) mark the obtained group of paired vertical lines in Candidate 1. Likewise, the same process of Hough Transform and grouping of line pairs were repeated on the horizontal edge image of the candidate region to obtain groups of paired horizontal lines, as shown in Figure 4.5 (c). Finally, each vertical line pair and horizontal line pair that are overlapping or close to each other were merged together to form the final rectangle that enclosing the license plate. As shown in Figure 4.5 (d), the final rectangle in yellow box only includes license plate characters without unneeded frame area. This is due to the vertical lines and horizontal lines that contributed by the license plate frame would not form pair line groups during the process.

Figure 4.5: Removing undesirable area process on Candidate 1

The same process of finding the area with cluster of horizontal line pairs and vertical line pairs was applied on other license plate candidates. The resultant images are shown in Figure 4.6. For Candidate 2, the redefined area after combining the line pairs is too small to be regarded as license plate. While for Candidate 3 and Candidate

4, the number of line pair is too little and the line pairs are scattered over the region. Therefore, all candidates except Candidate 1 were discarded and would not be considered as license plate. All the left over candidates after the removing undesirable region process (Candidate 1 in this case) will proceed to next phase of identification process. A few identification rules based on distinctive features of the license plate are used to identify true plate among the remaining candidates.

Figure 4.6: Removing undesirable area process on Candidate 2 to Candidate 4

4.3 Verification by Horizontal Rule

Horizontal Rule: *If a license plate is divided equally into three horizontal segments, the numbers of horizontal lines in the top segment and bottom segment are usually more than the number of horizontal lines in the middle segment.*

Based on the images of characters in Appendix A.1, it is noticeable that the horizontal lines of license plate characters are approximately located in three horizontal portions: top, middle and bottom. For majority characters, the middle portion has less horizontal lines compared with their top portion and bottom portion. Since a license plate comprises a series of alphanumeric characters, the horizontal lines of a license plate are expected to be concentrated on the top portion and bottom portion of the license plate.

To apply this characteristic of alphanumeric characters region, Candidate 1 was first divided equally into three horizontal segments by the red lines as shown in Figure 4.7 (a). The green lines in Figure 4.7 (b) are the horizontal lines of the license plate found in each segment. To apply Horizontal Rule, the total numbers of horizontal lines pixels in each segment were summed up independently. Horizontal Rule was verified by observing the result of the summation in Figure 4.7 (c). If multiple license plate candidates exist, candidate that has most horizontal lines in top and bottom section is more likely to contain alphanumeric characters.

Figure 4.7: Verification by Horizontal Rule process

4.4 Verification by Dip Rule

***Dip Rule:** For license plate with bright characters and dark background, the y value of the dips and the interval between the dips in its pixel intensity projection are approximately the same.*

4.4.1 Horizontal Projection and Dips Detection

The license plate characters are separated in a regular distance. For license plate with bright characters and dark background like Candidate 1, the gaps in between the characters have lower intensity value compared with the characters. The boundary between the characters can be determined by searching for the lowest intensity points between the characters

Let the gray input image of a license plate candidate defined by a discrete function $I(x,y)$, where (x,y) represents the image coordinate with $0 < x < \text{image width}$ and $0 < y < \text{image height}$. Then, a horizontal projection p_x at a point x of that function is a summary of intensity magnitude in the x^{th} column.

$$p_x(x) = \sum_{j=1}^{\text{height}} I(x, j) \quad \text{Equation (4.1)}$$

By summing up the pixel value along each column, the horizontal projection of Candidate 1 together with its local minimums is plotted in Figure 4.8. The local minimums or the dips of the projection were found by comparing each projection $p_x(x)$ value with its neighbour value.

In Figure 4.8, the dips plotted in red squares represent the dips located at the space between the characters. These dips form the fundamental points for Dip Rule verification. However, some superfluous dips represented by the magenta asterisks were also discovered in the projection. These dips are usually located at the middle of some characters like 'M', 'H', 'K' and 'U' due to the lack of bright pixels at that area. These unwanted dips need to be identified and removed to preclude from being treated as character boundary dips.

Figure 4.8: Horizontal projection of Candidate 1

4.4.2 Elimination of Superfluous Dips

(a) Segmentation lines form by main dips do not cross over any horizontal lines

(b) Segmentation lines form by superfluous dips intersect with many horizontal lines

Figure 4.9: Segmentation lines plotted on horizontal line images

Two features based on horizontal lines location and pixel intensity variation of license plate were employed to accomplish the task of getting rid of redundant dips in Figure 4.8. First, the vertical segmentation lines based on x position of the projection dips were obtained. The segmentation lines based on main dips and superfluous dips are plotted in horizontal line images in Figure 4.9 (a) and Figure 4.9 (b) respectively. It is evident that segmentation lines in Figure 4.9 (a) are located exactly at the space between the characters and do not intersect with any green horizontal lines. However the segmentation lines form by superfluous dips in Figure 4.9 (b) cross over many

horizontal lines of the characters. Based on this observation, one can conclude that a given dip is located in between two characters if the segmentation line based on its location does not across any character horizontal lines.

On the other hand, assuming the x position of the dips is denoted by $x_1, x_2, x_3 \dots x_n$, where n is the number of detected dips. The pixel intensity variance σ_i^2 at column x_i is calculated by using the following equation:

$$\sigma_i^2 = \frac{1}{height} \sum_{y=1}^{height} (I(x_i, y) - \mu_i)^2 \quad \text{Equation (4.2)}$$

where μ_i is the average of pixel intensity at column x_i calculated using

$$\mu_i = \frac{1}{height} \sum_{y=1}^{height} I(x_i, y) \quad \text{Equation (4.3)}$$

The dips that are located at the space between the characters have low intensity variation owing to unvarying background intensity of license plate. Based on this information, a dip_i is discarded if its intensity variance σ_i^2 is greater than a threshold T . Threshold T is determined based on the results of running the algorithm on experiment image sets. By checking the presence of horizontal lines as well as intensity variation at the dips position, all the superfluous dips in Figure 4.8 were eliminated.

4.4.3 Applying Dip Rule

Figure 4.10: Horizontal projection with superfluous dips eliminated

Figure 4.10 shows the final remaining dips after getting rid of the superfluous points. For the regions with alphanumeric characters, the final dips are approximately located at the middle of the gap between the characters. The value of the dips and the distance between the dips are also approximately the same. Assuming the final dips are represented by $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ where n is the total number of dips and d_i represents the distance between two dips with $d_i = x_{i+1} - x_i$. The average distance between the dips $\bar{d}$, and the average y value of the dips $\bar{y}$, are calculated based on the following equations:

$$\bar{d} = \frac{\sum_{i=1}^{n-1} d_i}{n-1} \quad \text{Equation (4.4)}$$

$$\bar{y} = \frac{\sum_{i=1}^n y_i}{n} \quad \text{Equation (4.5)}$$

Based on Dip Rule, a candidate is verified as valid license plate by the condition in Equation (4.6):

$$\begin{cases} \text{valid,} & 0.8\bar{d} < d_i < 1.2\bar{d} \text{ AND } 0.8\bar{y} < y_i < 1.2\bar{y} \\ \text{invalid,} & \text{otherwise} \end{cases} \quad \text{Equation (4.6)}$$

Based on the horizontal projection in Figure 4.10, Candidate 1 with alphanumeric characters satisfies the Dip Rule.

4.4.4 Character Segmentation

The final dips of the projection in Figure 4.10 obtained in previous step are located at the middle of two characters. The segmentation lines that separating the characters were obtained by utilizing the dips location along the x axis in the horizontal projection. Figure 4.11 shows the bounding boxes that enfolding the characters.

Figure 4.11: Bounding boxes enclosing each character

Compared with character segmentation using the projection of binarized

image [27], [11], [20], [42], [22], [45], [50], this method is more robust against the value of binarization threshold, the influence of illumination, plate frame and rivet. The horizontal lines position information and density variation checking in previous steps have helped to exclude segmentation line that otherwise affected by dirt or rivet between the characters.

After the segmentation, some geometric characteristics of the characters are exploited for future verification:

- The height/width aspect ratio of the bounding rectangle is within the range of 1.0 to 4.5.
- The minimum of bounding box is 2 (minimum number of characters in the first row of two row format license plate).
- The maximum number of bounding box is 8 (maximum number of characters in one row format license plate).

4.4.5 Verification by Dip Rule for License Plate with Dark Characters and Bright Background

For license plate with dark characters and bright background, the gaps in between the characters have higher intensity value compared with the characters. The highest intensity points between the characters are the corresponding boundary between the characters. Thus, Dip Rule cannot be applied directly on the license plate with dark characters and bright background.

Figure 4.12 shows an example of license plate with dark characters and bright background and its corresponding horizontal projection. Many of the dips detected are not located at the space between the characters. Therefore, during the process of superfluous dips elimination, if more than half of the dips of the candidate are eliminated, the candidate is regarded as license plate with dark characters and bright background. Each pixel value of the image is subtracted from value 255 (the highest intensity value for 8-bit gray image) and the difference is used as the output value of the complement image. The process of horizontal projection, dips detection, superfluous dips elimination and Dip Rule verification are applied on the complement of the candidate image. Figure 4.13 shows the corresponding complement image and the horizontal projection of the license plate in Figure 4.12.

Figure 4.12: Horizontal projection of license plate with dark characters and bright background

Figure 4.13: Horizontal projection of the complement image of license plate with dark characters and bright background

4.5 Verification by Vertical Rule

Vertical Rule: *The vertical lines in the license plate region are only located at the border of the characters.*

Beside the horizontal line signature, the vertical lines of the license plate characters also possess a distinctive attribute that can be applied for verification task.

By examining the position of the vertical lines of the characters, one apparent characteristic is that these vertical lines are located at the left and/or right side of the characters. This feature is valid except for characters 'T' and 'Y' where their vertical lines are located at the middle of the characters. Since the license plate is formed by a combination of alphanumeric characters, it is safe to claim that the vertical lines in license plate region only locate at the border of its characters. The green vertical lines shown in Figure 4.14 of the edge detected image of Candidate 1 verify that these lines are located near the border of the character bounding boxes.

Figure 4.14: Verification by Vertical Rule. The vertical lines are located near the border of the character bounding boxes.

Taking into account these unique properties of alphanumeric characters region, non-plate candidates would be easily eliminated after the verification tasks. Candidate 1 is the final determined license plate after applying these constraints on all license plate candidates

4.6 Experimental Results of License Plate Identification Process

To evaluate the performance of the proposed method, the images with license plate identified as one of the candidates in previous license plate localization step were submitted to license plate identification process. The process was considered as successful if the license plate was identified among the candidates in an image and all its characters in the character string were segmented without any loss. From the result of previous license plate localization, there are 194 images with license plate located as one of the candidates. Among those images, 187 of the license plate (96.4%) were identified with all the characters successfully segmented. The average execution time of MATLAB implementation is 0.57s. Figure 4.15 to Figure 4.27 illustrates the results of applying the license plate identification process on some the image examples shown previous chapter.

The license plates obtained from previous localization step usually contain the license plate frame areas. The first step in the proposed license plate identification process is removing these undesirable regions before applying the identification rules. Example 1 in Figure 4.15 demonstrates one of the results of extracting the exact license plate location by searching the area with pairs of horizontal lines and pairs of vertical lines. For some candidates such as Candidate 1 of Example 3, the located license plate is pretty much free from the frame area. The removing undesirable area process has not adverse effect on the originally detected license plate image. Sometimes the character string of license plate is started or ended with characters without any pair of vertical or horizontal lines, such as characters ‘A’, ‘V’ or ‘X’. Candidate 1 of Example 4 and Example 5 illustrates these cases. The initial rectangle enclosing license plate that obtained by searching the area with cluster of vertical and horizontal line pairs may not contain these characters. To ensure that these characters are not missed out, the rectangle is extended to the end of any nearby horizontal lines.

The same removing undesirable area process was also applied on other license plate candidates such as radiator and headlight of the vehicle as well as tree and its reflection on the windscreen. For these candidates, the pairs of linear lines detected in these candidates are sparsely located and the number of these pair lines is insufficient to form an area with license plate width and height. For instance, only a few pair of horizontal lines was found in the radiator in Candidate 2 of Example 1. For some candidates such as Candidate 3 and Candidate 4 of Example 1, no long vertical lines were detected in their edge images. As a result, candidates other than license plate were rejected and only true license plate would proceed to the next verification process.

The candidates that survived from the removing undesirable region process were verified by the proposed three identification rules. For Horizontal Rule verification, the candidate was first divided into top, middle and bottom sections and the horizontal lines pixels in these sections were summed up individually. The plots of the summation value in each section verify that the license plate has more horizontal lines in the top and bottom sections compares with its middle section.

Next, horizontal projections of the candidates were calculated before applying Dip Rule verification. For license plate with bright background and dark characters

such as Candidate 1 in Example 4 (Figure 4.22), the horizontal projection plot was obtained from the complement of the image after performing the necessary check. The dips shown in the horizontal projections plots are the final dips after eliminating the superfluous dips which are not located at the middle of two characters. The y value of the dips and the distance of the dips satisfy the condition set in Equation (4.6) for Dip Rule verification. The locations of the dips were also used to segment the license plate into individual characters. The segmented characters of the license plates along with the detected vertical lines in their vertical edge images are also presented in Figure 4.15 to Figure 4.27. These vertical lines are located near the border of the characters as stated in Vertical Rule.

There are some rare cases where more than one candidate satisfies all the verification rules. Candidate 2 of Example 5 in Figure 4.25 is one of the cases. It is a vehicle's manufacture logo image comprised of alphabet letters. The characters used in the logo resemble the characters of license plate. This candidate possesses the characteristic of license plate such as having many pair of vertical lines and pair of horizontal lines. It also fulfills all the conditions of Horizontal Rule, Dip Rule and Vertical Rule. Therefore, this candidate was regarded as true license plate even it is not. This issue can be resolved by checking the alphanumeric pattern of the license plate candidate. Unlike license plate which is comprised of alphabet and numeric characters, all the characters in Candidate 2 of Example 5 are alphabet letters. After applying the character recognition process on its segmented characters and doing a check on the result of the recognition, candidate which does not have the alphanumeric pattern of license plate would be discarded.

Example 1

Detected license plate candidates

(a) Removing Undesirable License Plate's Frame Area Process

**Figure 4.15: Result of applying license plate identification process on example images—
Part I**

Example 1 (continued)

(b) Verification by Horizontal Rule

Candidate divided into three segments

Horizontal lines in each segment

Summation of horizontal line pixels in each segment

Figure 4.16: Result of applying license plate identification process on example images – Part II

Example 1 (continued)

(c) Verification by Dip Rule

Horizontal projection of Candidate 1

(d) Verification by Vertical Rule

Bounding boxes enclosing each character and detected vertical lines

Example 2

Detected license plate candidates

Figure 4.17: Result of applying license plate identification process on example images – Part III

Example 2 (continued)

(a) Removing Undesirable License Plate's Frame Area Process

Figure 4.18: Result of applying license plate identification process on example images – Part IV

Example 2 (continued)

(b) Verification by Horizontal Rule

Candidate divided into three segments

Horizontal lines in each segment

Summation of horizontal line pixels in each segment

(c) Verification by Dip Rule

Horizontal projection of Candidate 1

Figure 4.19: Result of applying license plate identification process on example images – Part V

Example 2 (continued)

(d) Verification by Vertical Rule

Bounding boxes enclosing each character and detected vertical lines

Example 3

Detected license plate candidates

(a) Removing Undesirable License Plate's Frame Area Process

Figure 4.20: Result of applying license plate identification process on example images – Part VI

Example 3 (continued)

(b) Verification by Horizontal Rule

Candidate divided into three segments

Horizontal lines in each segment

Summation of horizontal line pixels in each segment

(c) Verification by Dip Rule

Figure 4.21: Result of applying license plate identification process on example images – Part VII

Example 3 (continued)

(d) Verification by Vertical Rule

Bounding boxes enclosing each character and detected vertical lines

Example 4

Detected license plate candidates

(a) Removing Undesirable License Plate's Frame Area Process

Figure 4.22: Result of applying license plate identification process on example images – Part VIII

Example 4 (continued)

(b) Verification by Horizontal

Figure 4.23: Result of applying license plate identification process on example images – Part IX

Example 4 (continued)

(c) Verification by Dip Rule

(d) Verification by Vertical Rule

Bounding boxes enclosing each character and detected vertical lines
in both Candidate 1 and Candidate 2

Example 5

Detected license plate candidates

Figure 4.24: Result of applying license plate identification process on example images – Part X

Example 5 (continued)

(a) Removing Undesirable License Plate's Frame Area Process

Figure 4.25: Result of applying license plate identification process on example images – Part XI

Example 5 (continued)

(b) Verification by Horizontal Rule

Candidate 1 divided into three segments

Horizontal lines in each segment

Summation of horizontal line pixels in each segment

Candidate 2 divided into three segments

Horizontal lines in each segment

Summation of horizontal line pixels in each segment

Figure 4.26: Result of applying license plate identification process on example images – Part XII

Example 5 (continued)

(c) Verification by Dip Rule

Horizontal projection of Candidate 1

Horizontal projection of Candidate 2

(d) Verification by Vertical Rule

Bounding boxes enclosing each character and detected vertical lines of Candidate 1 and Candidate 2

Figure 4.27: Result of applying license plate identification process on example images – Part XIII

Chapter 5

Signature-Based Character Recognition Process

After the characters in the license plate have been successfully segmented, recognition of these characters is the last step in the license plate recognition system. Many Optical Character Recognition (OCR) of license plate recognition systems reported in literature are based on either classifier [20], [65], [76] or neural network [78], [29], [41], [47], [4], [79], [66] techniques. These techniques require high computation resources to achieve real time performance. These types of algorithms are not suitable to be applied in embedded application like this project. Therefore, this project employs a recognition algorithm based on linear line signatures of the characters.

Figure 5.1: Signature-based character recognition process

The alphanumeric characters used in Singapore vehicle license plate of in are shown in Appendix A.1. Alphabet characters ‘I’ and ‘O’ are no being used to avoid ambiguities with numeric characters ‘1’ and ‘0’. Figure 5.1 shows the flow chart of the character recognition process proposed in this project to recognize these 34 characters (24 alphabet characters and 10 numeric characters).

5.1 Character Enlargement for Scale Invariance

The success rate of the character recognition technique proposed in this project hinges on the detectable linear lines in the characters. Small image resolution has adverse effect on the result of line detection and consequently the outcome of character recognition process. To ensure that the segmented characters are not too small before feeding to the recognition engine, the size of each segmented character is checked. Based on the trial results, the minimum character size required by the recognition engine is 50x20 (height x width). If the size of a character is too small, an image enlargement process based on bilinear interpolation will be applied on that character. This process is to ensure that the size of the character meets the desirable minimum size required by the recognition process. Compared with character recognition methods used in other license plate recognition systems that require size normalization for every segmented character [9], [12], [29], [47], [79], [80], the recognition technique proposed in this project invokes size enlargement process only if it is necessary. After this process, the character proceeds to horizontal line signature test.

5.2 Horizontal Line Signature

Figure 5.2: Horizontal line signature of character ‘3’

Taking character '3' in Figure 5.2 as an example, the horizontal line signature-based recognition starts with dividing the character into 9 segments by two horizontal lines and two vertical lines. The two horizontal segmentation lines are placed at 30 percentile from the top and bottom of the character respectively. Similarly, the two vertical segmentation lines are placed at 30 percentile from the left and right of the character respectively. Hough Transform-based line detection is applied on the horizontal edge image of the character and the green lines in Figure 5.2 (b) are the detected horizontal lines of character '3'.

Next, the existence of pair of horizontal lines at top, middle and bottom sections is checked. If there is any pair of horizontal lines with similar length is located at these sections, each individual segment is future examined. For each individual segment, if there is any horizontal line from the pair of lines occupies at least 30% of the segment's width, the segment is marked with '1'. The top segments of character '3' are marked as *111* since it has a pair of horizontal lines that occupies all of its top three segments. If there is any pair of horizontal lines exists at the top, middle or bottom sections of the character, all short horizontal lines that are located within the same section as the pair of horizontal lines will not be marked. Therefore, the bottom segments of character '3' are marked as *010* since the short horizontal line near the pair of horizontal lines is ignored.

On the other hand, if no pair of horizontal lines is found, like in the case of the middle section of character '3', then each segment is checked for the presence of individual horizontal line. Similarly, the segment is marked with *1* only if the length of the horizontal line is at least 30% of the width of the segment where it is located. The middle segments of character '3' are marked as *010* as it only has one horizontal line located at second column of middle sections. Finally, the horizontal line signature of a character is form by joining the marking result of each segment in the form of *top-middle-bottom*. As a result, the code word of character '3' by using its horizontal line signature is *111-010-010* (Figure 5.2 (c)).

Figure 5.3 to Figure 5.4 show the horizontal line signatures of all characters along with their horizontal edges and horizontal lines. For clarity purpose, only horizontal lines that contribute to the code words of the characters are highlighted with green lines. Table 5.1 summarizes the code words associated with each character

after the horizontal line signature marking process. Character ‘M’ and character ‘W’ are linked to two code words due to a variation of their font as shown in Figure 5.4. From Table 5.1, it is noticeable that some characters like ‘Z’, ‘E’ and ‘F’ can be recognized immediately based on their unique code words. Other characters such as ‘6’, ‘9’, and ‘8’ have common code word and need the vertical line signature for them to be distinguished.

Figure 5.3: Horizontal line signatures of characters ‘0’ to ‘H’

Figure 5.4: Horizontal line signatures of characters 'J' to 'Z'

Horizontal Line Signatures			Characters	Horizontal Line Signatures			Characters
Top Segments	Middle Segments	Bottom Segments		Top Segments	Middle Segments	Bottom Segments	
111	000	101	W	101	000	101	K, X, N, M
111	010	111	E	101	000	010	U, V, Y
111	010	100	F	011	000	011	1
111	010	010	3, 5	010	011	111	4
111	000	111	Z	010	010	010	6, 8, 9
111	000	010	7, T	010	001	010	G
110	010	110	B	010	000	111	2
110	010	101	R	010	010	011	Q
110	010	100	P	010	000	010	0, C, S

110	000	110	D	100	000	111	L
101	010	101	H,W	010	010	111	A
101	000	111	M	001	000	110	J

Table 5.1: Code words of all the characters based on their horizontal line signature

5.3 Vertical Line Signature

5.3.1 Dual Segment-Based Symmetry Check

Figure 5.5: Vertical line signatures of characters '6', '8', and '9'

To search for vertical line signature of a character, the character is first going through the process of partitioning process similar with previous section. However, the character is only divided into left segment and right segment at this time. Figure 5.5 illustrates the partitioning of characters '6', '9', and '8' with green lines represent the detected vertical lines after applying Hough Transform on the vertical edge images of the characters.

Compared with horizontal line signature in previous section that examines the presence of horizontal line in each divided segment, the vertical line signature only utilizes Hough Transform accumulator values. First, the total value of the peaks detected in the accumulator of left segment and right segment are summed up separately. These two values are compared subsequently for their disparity.

For each divided segment, the total value of the peaks in its Hough Transform accumulator roughly represents the total number of the line pixels. If significant

numbers of peaks are detected in both segments and the difference of their sum value is less than 30%, these two segments are regarded as symmetrical and the character's vertical line signature is marked with *11* (character '8' in Figure 5.5 (a)). Conversely, one segment is regarded as having stronger total Hough peaks value than the other segment if the difference is greater than 30%. In that case character '6' with more Hough peaks in the left segment and less Hough peaks in the right segment, its vertical line signature is *10* (Figure 5.5 (b)). The same check is applied on character '9' in Figure 5.5 (c) with *01* as the result. If no significant peaks are detected in both segments, *00* will be assigned as its vertical line signature.

Figure 5.6 to Figure 5.7 show the vertical lines of all characters highlighted in green color, along with the code word of their vertical line signature. By dividing a character into two segments and check the symmetry of their total Hough peaks value in each segment, the obtained vertical line signature can be used to separate characters with same horizontal line signature, such as characters in group '3, 5', group '7, T', or group '6, 8, 9' in Table 5.1. For characters with unique horizontal line signature, the vertical line signature can be used as supplementary signature to enhance the recognition certainty. However, for some characters with same horizontal line signatures, vertical line signature based on dual-segment symmetry check is insufficient to recognize these characters. For instance, both characters '0' and 'S' in the group with horizontal line signature *010-000-010* have the same vertical line signature *11*. Therefore, extra segment resolution for vertical line signature is needed for them be differentiated.

Figure 5.6: Vertical line signatures of characters '0' to 'N' by using dual segment-based symmetry check

Figure 5.7: Vertical line signatures of characters 'P' to 'Z' by using dual segment-based symmetry check

5.3.2 Quad Segment-Based Check

To recognize a character with horizontal line signature *010-000-010* (i.e. characters '0', 'C', and 'S'), the character is further divided into four segments as shown in Figure 5.8. For each segments, the total value of the peaks detected in the Hough accumulator bins are calculated. *1* is assigned to the segment if significant peaks value is obtained, *0* otherwise. By examining the strength of the peaks in these four segments, the vertical line signature of characters '0', 'C', and 'S' in Figure 5.8 are obtained. Based on their unique vertical line signature, the characters in this group is differentiable now. The same quad segment-based check is applied on characters 'U' and 'Y' with horizontal line signature *101-000-010* and vertical line signature *11*. Figure 5.9 shows the result of their quad segment-based vertical line signature.

Similarly, based on dual-segment-based symmetry check, vertical line signature *11* is assigned to character 'N' and one of the variations of character 'M'. These two characters belong to the character group with same horizontal line signature *101-000-101* (i.e. characters 'K', 'M', 'N', and 'X'). To identify these

characters, they are first divided into four segments as shown Figure 5.10. However, long vertical lines are excluded from the check this time by setting the Hough Transform module to detect vertical lines with length about the half of the character's height only. Figure 5.10 shows the obtained vertical line signatures of these four characters obtained using quad segment-based check but without including long vertical lines. These characters are differentiable now by their unique vertical line signature represented by strength of the peaks value in the four different segments.

Table 5.2 to Table 5.4 summarize the code word of the vertical line signature obtained by using quad segment-based check of each individual characters in the above mentioned three character groups.

Vertical Line Signatures (For Characters with Horizontal Line Signature <i>010-000-010</i>)				Characters
Segment 1	Segment 2	Segment 3	Segment 4	
1	0	0	1	S
1	0	1	0	C
1	1	1	1	O

Table 5.2: Vertical line signatures of characters with horizontal line signature *010-000-010* by using quad segment-based check

Vertical Line Signatures (For Characters with Horizontal Line Signature <i>101-000-010</i>)				Characters
Segment 1	Segment 2	Segment 3	Segment 4	
0	0	0	0	V
0	0	1	1	Y
1	1	1	1	U

Table 5.3: Vertical line signatures of characters with horizontal line signature *101-000-010* by using quad segment-based check

Vertical Line Signatures (For Characters with Horizontal Line Signature <i>101-000-101</i>)				Characters
Segment 1	Segment 2	Segment 3	Segment 4	
0	0	0	0	X
0	0	1	1	M
0	1	1	0	N
1	0	1	0	K

Table 5.4: Vertical line signatures of characters with horizontal line signature *101-000-101* by using quad segment-based check

Figure 5.8: Vertical line signatures of characters '0', 'C', and 'S' by using quad segment-based check

Figure 5.9: Vertical line signatures of characters 'U', 'V', and 'Y' by using quad segment-based check

Figure 5.10: Vertical line signatures of characters 'K', 'M', 'N' and 'X' by using quad segment-based check without including long vertical lines

5.3.3 Separating Character ‘H’ From Character ‘W’

Characters ‘H’ and one of the variations of character ‘W’ belong to the same character group with horizontal line signature *101-010-101*. Based on either dual segment-based check or quad segment-based check, both characters have the same vertical line signatures (*11* for dual segment-based check or *1111* quad segment-based check). Therefore, additional step is needed to distinguish these two characters.

Nevertheless, this issue can be resolved easily by checking the existence of a pair of horizontal lines in the middle section of the character. Character ‘H’ has a pair of horizontal lines in its middle section that differentiates it from character ‘W’. For that reason, if a character has horizontal line signature *101-010-101* and its middle segment signature is contributed by a pair of horizontal lines, the character is determined as character ‘H’. Otherwise, the character is classified as character ‘W’.

5.3.4. Verifying Character ‘1’

Figure 5.11: Characters ‘1’ with different vertical line signature based on dual segment-based check

Character ‘1’ has a unique horizontal line signature *011-000-011*. However, based on dual segment-based check, vertical line signature of character ‘1’ could result in two different code words due to the variation of its font. Figure 5.11 (a) is an example of character ‘1’ with vertical line signature *11* while Figure 5.11 (b) is another example of character ‘1’ with vertical line signature *01*. As a result, vertical line signature of character ‘1’ cannot be applied to identify this character. Two distinctive features of character ‘1’ are proposed here to verify the result of the recognition based on its horizontal line signature.

For any character with horizontal line signature *011-000-011*, the height/width ratio of the character is first calculated. Because alphabet character ‘I’ is not being

used in Singapore vehicle license plate, the only possible character with height/width greater than 2.0 is numeric character '1'. Secondly, the character is examined for the presence of a pair of vertical lines with similar length. Both variation of character '1' shown in Figure 5.11 satisfy this attribute. As a result, any character with horizontal line signature *011-000-011* and possess these two features is regarded as character '1'.

5.4 Experimental Results of Signature-Based Character Recognition

To evaluate the efficiency of the proposed signature-based character recognition, 1200 images of segmented character comprised all classes of license plate character were used in the experiment. Out of these characters, 1140 were successfully recognized providing an effectiveness of 95% with an average execution time of 0.34s per character in MATLAB environment. Figure 5.12 to Figure 5.17 show the samples of image of each class of character used in the experiment. Since the standard Singapore license plate follows a single font format, the proposed recognition method doesn't have the complexity that is usually presents in other unconstrained character recognition techniques. The experimental results demonstrate the effectiveness of the proposed signature-based character recognition method to recognize these characters.

The experimental results also revealed the wrong classified characters were occurred mainly in noisy images. Some characters such as character '5' and character 'S' share the same vertical line pattern but their horizontal line signature are considerably different. If the characters are near perfect (as in the high resolution images), the horizontal line signature is sufficient to differentiate between the two characters. If the images are noisy, characters extracted from license plates could have some information loss during the edge detection and line detection process. Such characters could be incorrectly identified if certain significant line information is absent.

Figure 5.12: Samples of character images used in experiment – Part I

Character '6'

Character '7'

Character '8'

Character '9'

Character 'A'

Character 'B'

Figure 5.13: Samples of character images used in experiment – Part II

Character 'C'

Character 'D'

Character 'E'

Character 'F'

Character 'G'

Character 'H'

Figure 5.14: Samples of character images used in experiment – Part III

Character 'J'

Character 'K'

Character 'L'

Character 'M'

Character 'N'

Character 'P'

Figure 5.15: Samples of character images used in experiment – Part IV

Character ‘Q’

Character ‘R’

Character ‘S’

Character ‘T’

Character ‘U’

Character ‘V’

Figure 5.16: Samples of character images used in experiment – Part V

Character ‘W’

Character ‘X’

Character ‘Y’

Character ‘Z’

Figure 5.17: Samples of character images used in experiment – Part VI

Chapter 6

FPGA-Based Hardware Implementation

After testing the efficiency of the proposed license plate recognition algorithms in MATLAB environment, the system is ready for deployment on hardware platform. A FPGA-based Configurable System on Chip (CSoC) platform [1] was chosen as prototyping platform for the proposed license plate recognition system. CSoC platform integrates the microprocessor, memories and peripheral interfaces into one single FPGA chip. The advantages of using CSoC concept include increased system performance, reduced system cost and reduced system power dissipation [1]. It leverages the design advantages of both conventional processor and programmable logic. In this project, the processor of the CSoC platform is used to execute normal sequential C++ application. The developed license plate recognition algorithms in MATLAB language were converted to C++ code by using Embedded MATLAB toolbox [105]. On the other hand, computation intensive image processing algorithms were implemented as custom hardware on the FPGA chip for optimal performance.

6.1 Architecture of CSoC-Based System

Figure 6.1 shows the basic set up of the FPGA-based hardware system. The Altera DE2-70 board [106] was chosen as the system implementation platform. DE2-70 has an Altera Cyclone II FPGA with approximately 70,000 logic elements. The rich set features of DE2-70 board are suitable for a wide range of design projects. The user can configure the FPGA to implement any system design since all connections to the peripherals on the board are made through the FPGA device. The proposed license plate recognition system was implemented based on Nios II system architecture [107] with the processor and the hardware accelerators running at 100MHz. Figure 6.2 illustrates the block diagram of the license plate recognition system implemented on DE2-70 board.

Figure 6.1: FPGA-based implementation of license plate recognition system. The system consists of a DE2 board and a LCD display.

Figure 6.2: High-level block diagram of CSoC architecture

The system was created by using SOPC Builder system development tool [108]. SOPC Builder allows the user to create a Nios II system complete with a customized set of system peripherals. It automatically generates the system interconnect fabric, which is the glue logic required to connect the design blocks

together. The system interconnect fabric manages design issues such as dynamic bus width matching, interrupt priorities and arbitration. All the peripherals communicate with the system interconnect fabric through Avalon interface ports [109]. The functions of each system component in Figure 6.2 are described as follow:

i. Nios II Soft Core Processor

Nios II processor is a general-purpose soft core processor implements a 32-bit instruction set based on RISC (Reduced Instruction Set Computer) architecture [107]. It is a highly customizable soft-core processor optimized for implementation in programmable logic chips. Many parameters of Nios II processor can be selected at design time, including the data and instruction cache size, number of pipeline stage and arithmetic logic unit implementation. It is used in the system to execute high level license plate recognition operations in C++ code.

ii. JTAG UART Core with Avalon interface

The JTAG universal asynchronous receiver/transmitter (UART) core acts as a communication interface of serial character streams between a host PC and the Nios II processor for software debugging purposes.

iii. DDR SDRAM Controller and Flash Interface Controller

These two controllers are used for accessing the DDR SDRAM and flash memory on DE2-70 board respectively. The SDRAM is used for storing the application's code, stack and heap during the run time execution. It also acts as frame buffer of the video pipeline subsystem. The flash memory is used to store the vehicle images during the experimental testing process.

iv. LCD Controller and Video Pipeline Subsystem

The execution results of various processing stages of the license plate recognition system are displayed on a graphical LCD module connected to the DE2-70 board. The LCD controller is used to interface the LCD module to configure the LCD for brightness, resolution, gamma curves and other parameters. The video pipeline subsystem is responsible for reading the pixel data generated by the Nios II processor and transfer them to LCD module for

rendering. The details implementation of the LCD controller and video pipeline subsystem can be found in the reference designs stated in [110].

v. DMA (Direct Memory Access) Controller

Three hardware accelerators were designed to accelerate the performance of time consuming image processing functions. The DMA controller is responsible for moving input and output image to and from the accelerators autonomously, without intervention from the Nios II processor. While the DMA controller performs memory transfers, the processor is free to perform other tasks in parallel.

vi. Hardware Accelerators

Sobel edge detection, binary morphological operations and Hough Transform-based line detection are the core image processing algorithms that have been used extensively in this project. It is envisioned that the computation time of these functions will affect the overall license plate recognition system performance. Therefore, these functions are implemented as hardware accelerators by using Altera DSP Builder [111] to speed up their processing time. Altera DSP Builder is a development tool that interfaces MATLAB with the Altera Quartus II development software. The tool greatly saves the development time by automatically generating the hardware description language (HDL) files of the image processing functions which were first designed in MATLAB environment. The generated HDL files are optimized for use in the Altera Quartus II software for rapid prototyping. The following sections describe the implementation details of these hardware accelerators.

6.2 Implementing Sobel Edge Detector

As described in previous chapters, this project utilizes the linear line features of license plate in various recognition processing stages. Sobel edge detection is the first process to convert the gray level vehicle image to binary image before the line detection process. Figure 6.3 is the block diagram of the FPGA-based hardware implementation of Sobel Edge Detector.

Figure 6.3: Block diagram of Sobel Edge Detector

The data flow of the Sobel Edge Detector is controlled by Nios II processor. DMA controller is used to transfer the pixel data between the external memory (DDR2 SDRAM) and the Sobel Edge Detector. The pixel data are streamed through the Sobel vertical filter and horizontal filter which function as co-processors to hardware accelerates the two-dimensional convolution operations. Thresholds are then applied on the generated gradient images to obtain the final binary edge images.

6.2.1 Avalon Slave Interface Block of Sobel Edge Detector

The Sobel Edge Detector is attached to the Nios II system as a co-processor. The Avalon Slave Interface block defines a collection of ports for connection to the SOPC Builder System. It is used to stream the input image to the Sobel Filter blocks as well as transfer the output back to the memory after computation. This block also receives the input threshold value used for image binarization from the user.

6.2.2 Input Buffer Block of Sobel Edge Detector

The input image is convolved with the Sobel filter mask to obtain the gradient image. As introduced in Chapter 3, convolution operation by using traditional computation approach requires a processor to read many non-adjacent pixels from the memory to compute the output of a single pixel. This method is not adapted to high throughput real-time execution, because several clock cycles are needed to compute one single pixel output.

$z_{1,1}$	$z_{1,2}$	$z_{1,3}$
$z_{2,1}$	$z_{2,2}$	$z_{2,3}$
$z_{3,1}$	$z_{3,2}$	$z_{3,3}$

(a) A 3x3 region of an image
window operation

(b) Delay-line architecture of sliding

Figure 6.4: Input buffer block of Sobel Edge Detector

The Input Buffer block uses delay-line architecture as shown in Figure 6.4 to implement the sliding window operation on a 3x3 region. The incoming pixels are shifted through line buffers that create a delay line. The main idea is to store pixels on the left and right of the kernel as long as they are to be used in future output calculations, i.e., all consecutive pixels from the upper left corner to the lower right corner of the kernel are stored in the memory chain. The buffer depth is 640 pixels, which is the number of pixels in each line of the frame. The number of buffer lines depends on the size of the convolution kernel and is 3 for the case of Sobel edge detection.

All pixels covered by the kernel are stored in flip-flops (FFs) and are thus individually accessible. All pixels in between two rows of the kernel are stored in line buffers. As an incoming pixel is shifted in, the oldest pixel currently in the architecture is shifted out. The major benefit of this architecture is its ability to support streaming input data and arbitrary shaped kernels. The output image can be computed by reading the input image only once because many redundant memory reading are eliminated by caching the input image using line buffers.

6.2.3 Sobel Vertical and Horizontal Filter Blocks

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

(a) Horizontal kernel

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

(b) Vertical Kernel

Figure 6.5: Sobel edge detection kernels

By applying Sobel vertical mask and horizontal mask in Figure 6.5 on a 3x3 region of image in Figure 6.4 (a), the gradient G_x and G_y at location $z_{2,2}$ of a 3x3 region can be summarized as following two equations:

$$G_x = (z_{3,1} + 2z_{3,2} + z_{3,3}) - (z_{1,1} + 2z_{1,2} + z_{1,3}) \quad \text{Equation (6.1)}$$

$$G_y = (z_{1,3} + 2z_{2,3} + z_{3,3}) - (z_{1,1} + 2z_{2,1} + z_{3,1}) \quad \text{Equation (6.2)}$$

(a) Sobel Horizontal Filter

(b) Sobel Vertical Filter

Figure 6.6: Block diagram of Sobel Filters implementation

The Sobel Vertical Filter and Horizontal Filter blocks in Figure 6.3 operate independently and in parallel to calculate the gradient of the image based on the above equations. Figure 6.6 shows the architecture of these two blocks. They are direct implementation of Equation 6.1 and Equation (6.2) by using adders, subtractors and amplifiers. The absolute values of the output are taken as the magnitude of the gradients.

6.2.4 Thresholding Block

The threshold input received from the user via Avalon Interface block is applied on the generated gradient to create a final binary image as the edge detection result. If the gradient of the pixel is above the threshold, it is considered as an edge pixel. If the gradient of the pixel is below the threshold then it is unconditionally set to zero. The filter is pipelined where a resultant pixel is calculated every clock cycle.

6.3 Implementing Morphological Operator

Binary morphological operations are used to connect the edges scattered over the license plate region. Morphological operations share similarity with sliding window operation, but use a different operator in their calculation. Figure 6.7 is the block diagram of the FPGA-based hardware implementation of Morphological Operator.

Figure 6.7: Block diagram of Morphological Operator

6.3.1 Avalon Slave Interface Block of Morphological Operator

Similar to the Avalon Slave Interface block of Sobel Edge Detector, this block defines the interaction between the operator and the interconnect logic that connects the operator to the rest of the system. Besides streaming the image between the operator and memory, it also receives the following two input parameters from user:

- i. *SE_Row_Size* and *SE_Column_Size*

The user controls the height and width of the operating structuring element (SE) by setting the value of these two signals. The accelerator is designed to support SE with rectangle size up to 40x40.

- ii. *Operation_Type*

The *Operation_Type* signal is used to set the type of morphological operation to be executed by the accelerator. These operations include dilation, erosion, closing and opening.

6.3.2 Input Buffer Block of Morphological Operator

Figure 6.8: Block diagrams of Input Buffer and its output control logic

The Input Buffer block of Morphological Operator in Figure 6.8 (a) has similar architecture with the corresponding block in Sobel Edge Detector. The FIFO line buffer depth is 640 pixels which is the width of input image, but the number of buffer lines is 40, the maximum allowable SE size.

To select the image pixels covered by the SE with variable size, the output of the buffers z_{ij} are governed by control logics as shown in Figure 6.8 (b). The *Erosion/Dilation* signal is derived from the user input *Operation_Type*. The *SE_Row_Size* and *SE_Column_Size* signals received from the user are translated to individual 40x40 enable signals of each SE cell, represented by SE_{ij_Enable} in the block diagram. These enable signals set the input pixels that are not covered by the SE to 1 (for erosion operation) or 0 (for dilation operation). This is to ensure that the pixels outside the SE coverage have no effect on the result of the operation. The output of the control logics r_{ij} are fed to the Dilation/Erosion block.

6.3.3 Dilation/Erosion Block

The main block of Morphological Operator is the Dilation/Erosion sub-block which performs one of the two basic morphological operations selected by the user. For dilation operation, the output is set to 1 if any of the neighboring pixels covered by the SE is 1. The erosion process on the other hand requires that all the values of the neighboring pixels covered by the SE to be 1 in order to output a 1. From these realizations, the dilation operation becomes an OR operation over the inputs that are covered by the SE. The erosion operation in contrast is a logical AND operation over the inputs covered by the SE. Figure 6.9 shows the implementation of the dilation and erosion operation based on this observation. The multiplexer select the final result pixel according to the operation type selection signal.

Figure 6.9: Block diagram of Dilation/Erosion. Dilation operation is OR operation over the inputs while Erosion is AND operation over the inputs.

As represented by previous Equation 3.6 and Equation 3.7, opening and closing operations can be formed by combining the dilation and erosion operations. In Morphological Operator block, these two operations are realized by simply cascading two Dilation/Erosion blocks. The *Operation_Type* signal selects the appropriate result at the multiplexer output as shown in Figure 6.7.

6.4 Implementing Hough Transform-Based Line Detector

The Hough Transform-Based Line Detector (abbreviated as Line Detector in the following sections) is one of the main components in this license plate recognition system. The system makes use of the linear lines discovered by the Line Detector in several recognition phases. The Line Detector was created to improve the performance of the line detection process. Figure 6.10 below shows the architecture of the Line Detector. Beside standard Avalon Interface block for any component connected to SOPC Builder system, it has another two main modules: Hough Transform (HT) Processor and Line Pixels Generator. The former is used to accelerate the computation of Hough Transform which is originally a time consuming process, while the latter is used to recover the lines in the image based on the detected peaks in Hough Transform space.

Figure 6.10: Block diagram of Hough Transform-Based Line Detector

6.4.1 Avalon Slave Interface Block of Line Detector

The Avalon Interface block is responsible for receiving the edge image input for line detection process and transferring the detected lines information back to user. It also receives the following parameters from the user:

- i. *Image Size*
The height and width of the input edge image for line detection process.
- ii. *Direction*
The direction (vertical or horizontal) of the lines to be detected.
- iii. *Peak Threshold*
The threshold at which values in Hough Transform space are considered to be peaks.

6.4.2 Hough Transform (HT) Processor

The HT Processor is used to accelerate the computation of Hough Transform which is originally a time consuming process. It can be divided into four sub-blocks: coordinate counter, processing element array, accumulator memory and peak detector.

Figure 6.11: Architecture of Hough Transform Processor block. For each incoming pixels, the PEs calculates four ρ simultaneously and store them in accumulator memory.

6.4.2.1 Coordinate Counter Block

After initialization, the input edge image is transferred in raster scan sequence to HT processor starting from top left position. Coordinate Counter block takes the image size as the input and tracks the position of the pixel that is transferred from the memory at the moment. It contains two counters: horizontal position counter and vertical position counter to provide the value of x and y parameter in the Hough Transform equation.

The horizontal position counter and vertical position counter are initialized to zero at the starting. The horizontal position is increased by one when there is a valid incoming input. The horizontal position counter will set to zero and trigger the vertical position counter to be increased by one when the counter value is equal to the maximum image width. Both counters will stop counting when they reach image bottom right pixel position.

6.4.2.2 Processing Element Array

Since the license plate recognition algorithms proposed in this project only rely on vertical line and horizontal line features, the range of theta angle for Hough Transform computation are restricted to 10 angles for horizontal line detection (-4° to $+5^\circ$) and another 10 angles for vertical line detection (-90° to -86° and $+86^\circ$ to $+90^\circ$). The Line Detector contains 10 processing elements (PEs), each dedicated for different theta (θ) angle calculation. These PEs operate in parallel to calculate 10 values of rho (ρ) for either vertical line or horizontal line detection simultaneously.

Figure 6.12: Block diagram of a processing element in Hough Transform Processor block

Each PE calculates the ρ value according to Equation 4.1. As shown in Figure 6.12, each PE consists of two multipliers and an adder. During the initialization process, Line Detector is first configured to detect either vertical line or horizontal line. The pre-computed $\cos \theta$ and $\sin \theta$ values which are stored in a lookup table are loaded to the multipliers of each PE. As the edge pixels are sequentially transferred in, the ρ values for different angles θ are generated by the PEs at the same time and fed to the address line of the accumulator memory.

6.4.2.3 Accumulator Memory

Hough Transform needs accumulator memory to store the voting count in each probable parameter point. To avoid access contention, the accumulator memory is partitioned into blocks with each RAM module is associated with one PE in the system (Figure 6.11). By assuming the maximum input image is 240x320 (half of size of original vehicle image), the size of each RAM module is 799 words (the range of ρ calculated based on the distance between corners in the image).

The accumulator memory has two operation modes: clear mode and normal mode. When working in clear mode during the initialization, an external address generator supplies the address to the accumulator memory. The supplied address starts from zero and is sequentially increased to the maximum size of the RAM module. Value 0 is written to the location pointed by the generated address.

In normal mode, the address of each RAM block is provided by the ρ value computed by their associated PE. If the image input value is 1 (indicates an edge point), the adder at the output of RAM will be activated and the count value in each RAM as pointed by the ρ address will be increased by one. Otherwise, the content of the accumulator memory remain unchanged.

6.4.2.4 Peak Detector

In a normal line detection application, a peak of the Hough Transform will be detected by searching for local maximum with respect to its neighbors in the accumulator memory, as well as having counter value over a certain threshold. However, the peak detection process in this design only exports the first output the peak candidates that have value above the given threshold. This is due to the concern

of the complexity of the peak extraction circuits, and to reduce the data amount in the peak reporting process.

The output of each RAM module in the accumulator memory is connected to a comparator that compares its value with the preset threshold value. When the updated RAM content value reaches the threshold, the system raises a flag to indicate a peak is detected for that θ bin. The Peak Output Transfer Unit will check the flag signal status of each θ bin when each ρ computation is finished. If the corresponding flag signal of a θ bin is high, the transfer unit will take the ρ value and the θ bin to form the peak output to be transmitted out. Considering the complexity of the I/O interface by adopting parallel transfer of all the peaks output, the transfer unit is constrained to output one set of peak data one after another in a daisy chain format. If more than one peak is detected for the same (x, y) coordinate, the flag status of the comparator will remain high until all of the peaks have been transmitted out by the transfer unit. The flag will then be set to clear status by the system.

6.4.3 Line Pixels Generation Module

Line Pixels Generation module is used to recover the lines in the image based on the detected peaks in Hough Transform space. It was implemented by using a combination of hardware components and software function. In Hough Transform-based line detection algorithm, locating the peaks in Hough parameter space is just half the job done. The (x, y) coordinates that correspond to the peaks detected in (ρ, θ) space need to be found. The simplest method to achieve this purpose is stated in following algorithm:

```

for all detected peaks  $(\rho_i, \theta_i)$  where  $1 \leq i \leq \text{number of detected peaks}$ 
    for all edge pixels  $(x_j, y_j)$  where  $1 \leq j \leq \text{number of edge pixels}$ 
        calculate  $\rho_j = x_j \cos \theta_i + y_j \sin \theta_i$ 
        if  $\rho_j = \rho_i$ , then  $(x_j, y_j)$  is one of the points contribute to line segment of
peak  $(\rho_i, \theta_i)$ 
    end for
end for

```

From the above procedure, it is evident that locating the edge pixels that contribute to the line segments of a peak could be a time consuming task. For each

detected peak with value θ_i , the ρ values of each edge pixel needs to be computed and compared with the peak ρ_i value. To accelerate the whole process, the Line Pixels Generator module in Figure 6.13 contains four processing elements that do the ρ calculating task for 4 peaks simultaneously. Its process is very similar to HT Processor that calculates the ρ value of each incoming edge pixel.

Figure 6.13: Architecture of Line Pixels Generator. For each incoming pixels, the PEs calculates four ρ_j simultaneously and compare them with the ρ of the peaks.

Let assuming there are 4 peaks, (ρ_i, θ_i) detected by the Peak Detector module, where $1 \leq i \leq 4$. Each PE in block Line Pixels Generator module is set to determine the (x, y) coordinates that contribute to each peak individually. After setting the peak (ρ_i, θ_i) parameter for each PE, the Avalon Slave Interface receives and transfers the edge pixels to the 4 HT calculators simultaneously. The coordinate counter keeps track of the current image pixel (x_j, y_j) coordinates.

If the incoming pixel is an edge point (its value is 1), the calculators are activated to calculate the ρ_j value of the edge pixel and compare with their allotted ρ_i value. If both ρ values are matched, the current (x_j, y_j) coordinate will be transfer out

as line pixel by the output port. The line pixels corresponding to each peak are stored in different section of the memory. The whole process is repeated until there is no more incoming image pixel. In the end, the edge pixels that contribute to line segment of the 4 peaks are obtained.

6.4.4 Line Segment Identification Software Function

In real world application, some long straight line may be broken into different line segments due to noise. In that case, the line pixels obtained by the Line Pixel Generator module may result in fragmented lines which are too small for a particular purpose. This issue is resolved by a software function, Line Segment Identification that connects the line pixels to generate useful line segments.

For each detected peak in Hough Transform space, the function takes two parameters from the user, *fill_gap* and *min_length* together with the corresponding coordinates of the peak's edge pixels. For every two adjacent line segments, the parameter *fill_gap* determines the limit of the number of missing points in between them. If the number of missing points in between the two lines is less than *fill_gap*, the intermediate missing points will be filled up and both the segments will be treated as a single continuous line segment. After that, the length of all the line segments including the newly merged one is compared with *min_length* value. The line segment is kept if its length is greater than *min_length* value. Otherwise, all short line will be discarded. The final outputs are the detected lines in the edge image with minimum length of *min_length* value.

6.5 Experimental Results of Hardware Implementation System

The performance of the license plate recognition system implemented on the CSoC platform was evaluated by using 200 vehicle images. To signify the performance boosted by the dedicated hardware accelerators, the same vehicle images were also tested by using a Nios II system with hardware accelerators being replaced by their corresponding software version. Table 6.1 shows the recognition results of the experiment while Table 6.2 shows the average execution time taken by the systems to process an image. The number of programmable logic elements (LEs) of

FPGA utilized as well the power consumption of the systems are also being compared in Table 6.3. The Cyclone II FPGA of the DE2 board has total of 68,416 LEs.

Stage	Accuracy (%)
License Plate Localization	96%
License Plate Identification	95.3%
License Plate Character Recognition	94%
Overall	$96\% \times 95.3\% \times 94\% = 86\%$

Table 6.1: Success rate of hardware implementation

Stage	Average Execution Time (s)		Performance Speed Up by Hardware Accelerators
	Nios II System Without Hardware Accelerators	Nios II System With Hardware Accelerators	
License Plate Localization	224.868s	2.184s	103x
License Plate Identification	32.666s	2.808s	12x
License Plate Character Recognition	269.006s	22.852s	12x
Total	526.54s	27.844s	19x

Table 6.2: Average execution time of one image on CSoC platform

System	Number of Logic Elements Utilization	Estimated Power Consumption
Nios II System Without Hardware Accelerators	15,197 of 68,416 (22%)	828.34 mW
Nios II System With Hardware Accelerators	36,094 of 68,416 (53%)	1120.11 mW

Table 6.3: Logic elements utilization and estimated power consumption of CSoC system

The license plate recognition system achieved an overall success rate of 86%. For the system with hardware accelerators, it took an average of 28s to recognize a vehicle license plate. The hardware accelerators increased the logic elements utilization by 31%, but the execution speed was increased by significant 19x compared with the Nios II system without hardware accelerators. The hardware accelerators had greatly reduced the execution time that would otherwise be spent in excessive memory reading and writing operations involved in edge detection and binary morphological operations. The parallel execution of Hough Transform equation by using processing element array in Figure 6.11 also contributes to this speed up.

The power consumption of the system was estimated by using Altera PowerPlay Power Analyzer [112] tool. Because accurate signal-activity data for the nodes of the design were not being used during the analysis, the obtained results are just rough estimation of the power consumption of the system. Nevertheless, the system with hardware accelerators only increased the total power consumption by 35% compare with the system without hardware accelerators.

From Table 6.1, it can also be noticed that there are small discrepancies between the success rate obtained by the hardware implementation and the results obtained in MATLAB environment as shown in previous chapters. This is due to the output of the MATLAB functions of edge detector, morphological functions and Hough Transform are not exactly the same as the output of their corresponding hardware accelerators. Nevertheless, these discrepancies are negligible.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, several novel algorithms have been proposed for the implementation of a low-cost license plate recognition system with real-time performance capability. The implementation of the system was divided into three stages – extraction of the license plate, identification of license plate among candidates and character recognition. The performance of the algorithms implemented in each stage was evaluated individually in MATLAB environment and a success rate more than 90 % was realized in each of the stage.

The extraction of license plate was performed by searching for area with dense concentration of vertical edges in the image. The proposed technique was able to extract license plate even from complex background without any *a priori* knowledge regarding the foreground and background colors of the license plate. However, multiple license plate candidates with similar characteristic were always extracted along with license plate after this process, especially for cases where the images were taken outdoor. The more number of candidates extracted from an image, the more time the system would spend in the subsequent process to identify true license plate among these candidates.

Besides removing undesirable frame area of license plate, the process of finding the area with cluster of horizontal line pairs and vertical line pairs in license plate had also helped to eliminate many of the non-plate candidates. According to the experimental results, most of the candidates such as the radiator and headlights of the vehicles with none or limited number of horizontal line pairs and vertical line pairs were discarded after this process. The remaining candidates except true license plate were then rejected by the three verification rules which were defined based on unique alphanumeric characters features of license plate. The verification however failed when the image was seriously tilted as the distribution of horizontal lines in tilted license plate didn't satisfy the condition of Horizontal Rule. Moreover, dips in the projection histogram of tilted license plate couldn't accurately represent the location

of the gaps between characters. The characters in the license plate were not segmented precisely due to the segmentation lines based on the inaccurate dips locations.

To reduce the overall complexity of the system, the features of the license plate's straight lines are further exploited in character recognition module. The experimental results show that the implementation of the signature-based recognition module was simple, yet effective for recognizing characters in vehicle license plate of Singapore. More than half of the characters could be recognized immediately based on their unique horizontal line signature. The vertical line signatures were used to differentiate the rest of the characters that have the same horizontal line signatures. It also increased the confidence on the recognition results obtained by using horizontal line signatures. The recognition module however could not differentiate between characters that are very similar in appearance such as characters 'D' and '0', characters 'B' and '8', when these characters were excessively degraded by noise. If the images were noisy, some meaningful signatures pertaining to the characters could have been lost during the edge detection and line detection process.

After testing the effectiveness of the algorithms in MATLAB environment, a hardware prototype for the proposed license plate recognition system was built on an Altera's FPGA-based CSoC platform. The minor discrepancies of less than 2% between the success rate obtained by the hardware implementation and the results obtained in MATLAB environment implies there was no significant loss of accuracy when converting the original algorithms in MATLAB language to C++ codes. This also means that the results obtained by the hardware accelerators are comparable with their corresponding software implementation. Experiment results reveal that the system spent most of the execution time in license plate character recognition process due to the repeated invocation of Hough Transform for each character. The hardware accelerators increased the execution speed by 19x with an average of 27s to process an image when compared to the entire solution running on Nios II processor. This was possible due to the high levels of parallel processing data-flow structures supported by FPGA, which are important for efficient implementation of image processing algorithms.

7.2 Future Work

The set of images used in experiment were captured under good condition with clear view of license plate. In real world application, there are variations in the image quality such as blurred and skewed images, or dust, shadows, and strong lights on license plates. The edges and the straight lines of the license plate characters are not clearly presented in these low quality images. The proposed system which is based on these two important features of license plate is expected to fail to recognize the license plate in these images. To increase the robustness of the system to deal with real world conditions, the future work should include some considerable level of pre-processing on the captured images. These pre-processing steps could enhance the contrast or sharpen the captured images prior the recognition process. This can ensure that the quality of the images is sufficient for the rest of the algorithms to work.

Some ambiguous characters such as ‘B’ and ‘8’, ‘D’ and ‘0’ are difficult to recognize under the presence of noise. The future work should include finding ways to distinguish these characters even in the presence of noise. One possible method could be utilizing the curve line signatures or diagonal line signatures found on these characters. It is envisaged that this method will not add too much complexity to the current system because it can be built on top of Hough Transform based line detection engine that has already been utilized broadly by the system. On the other hand, alphabets and numbers that are similar in appearance can be distinguished by using the context in which they appear on the plate. This would entail finding the position of the character in the license plate and knowledge about the preceding and succeeding characters. For example case of characters ‘8’ and ‘B’, if the preceding and succeeding characters are alphabets, then the character is most likely a character ‘B’ instead of character ‘8’.

Image with large tilt in the license plate could affect the result of license plate identification process. The license plate in a capture image could be tilted depending on the angle between the camera and the license plates at the time of image capture. Even though the license plate region could still be located, but seriously skewed image would not satisfy the Horizontal Rule or Dip Rule conditions. Characters in tilted license plate could not be segmented without any loss. For this reason, it is necessary to determine the orientation of the license plate and correct the skewed

image before applying the identification rules. During the process of removing undesirable area of license plate, Hough Transform based line detection is applied to detect the horizontal lines within the license plate candidate. Therefore, it is feasible to obtain the tilt angle of the license plate by checking at which angle most of the horizontal lines are detected. Tilt correction such as rotating the image back by the tilt angle can be subsequently applied on the image.

The proposed license plate recognition algorithms in this thesis were designed to detect Singapore plates. License plates of other nationalities have not been included in the testing images. The future work will include the assessment of the portability of the system to license plate of different nationalities. Some algorithms such as Horizontal Rule which was designed in the beginning based on the characteristics of Singapore plate may need to be re-examined and revised to accommodate license plate of other countries.

Compared with Nios II system without hardware accelerators, the hardware accelerators consumed additional 31% of the logic element utilization. Nevertheless, there are remaining 47% of logic elements left in the Cyclone II FPGA on the DE2-70 board. To further boosting the performance of the system, a thorough profiling on the system could find out which portion of the computation process consume most of the time. The bottleneck of the system could be solved by implementing additional hardware accelerators or custom instructions of Nios II processor [1] on these remaining FPGA resources. Moreover, Nios II is a highly configurable soft-core processor implemented on FPGA [2]. The number of Nios II processor that can be implemented on a FPGA is only restricted by its available resources. A multicore Nios II processor system can execute different software tasks in parallel and greatly improve the performance of the system. For instance, the execution time of license plate character recognition process could be shorten by assigning dedicated Nios II processor and Hough Transform based line detector to recognize each character in parallel. It is also possible to increase the current frequency of Nios II processor beyond 100MHZ at the expense of increasing resource usage.

References

- [1] Frantz, G.A., "System on a chip: a system perspective," in *Proceedings of Technical Papers of the 2001 International Symposium on VLSI Technology, Systems, and Applications*, Hsinchu, Taiwan, 2001, pp. 1-5.
- [2] H. Lee, Y. Chen and S. Wang, "Extraction and recognition of license plates of motorcycles and vehicles on highways," in *Proc. 17th ICPR 4*, 2004, pp. 356-359.
- [3] F. Yang and Z. Ma, "Vehicle license plate location based on histogramming and mathematical morphology," in *Proc. 4th IEEE Workshop Autom. Identification Advanced Technol.*, 2005, pp. 89-94.
- [4] D. Hamad, and G. Tilmant A. Taleb-Ahmed, "Vehicle license plate recognition in marketing application," in *Proc. Intell. Vehicles Symp*, 2003, pp. 90-94.
- [5] M. Hongqing, L. Jilin, and L. Langang D. Yan, "A high performance license plate recognition system based on the web technique," in *Proc. Conf. Intell. Transp. Syst.*, 2001, pp. 325-329.
- [6] D. Zheng, Y. Zhao, and J. Wang, "An efficient method of license plate location," *Pattern Recognit. Lett.*, vol. 26, no. 15, pp. 2431–2438, Nov 2005.
- [7] S. Wang and H. Lee, "Detection and recognition of license plate characters with different appearances," in *Proc. Conf. Intell. Transp. Syst.*, vol. 2, 2003, pp. 978-984.
- [8] S. Kasaei, F. Dorri, and F. Dorri H. Mahini, "An efficient features-based license plate localization method," in *Proc. 18th ICPR*, vol. 2, Hong Kong, 2006, pp. 841-844.
- [9] P. Ferragina, M. N. Granieri, and F. Stabile P. Comelli, "Optical recognition of motor vehicle license plates," *IEEE Trans. Veh. Technol.*, vol. 44, no. 4, pp. 790–799, Nov 1995.
- [10] S. Draghici, "A neural network based artificial vision system for license plate recognition," in *Int. J. Neural Syst*, vol. 8, Feb. 1997, pp. 113–126.
- [11] F.-C. Ni, K.-T. Li, and Y.-P. Chen T.-H. Wang, "Robust license plate recognition based on dynamic projection warping," in *Proc. IEEE Int. Conf. Netw., Sens. Control*, 2004, pp. 784–788.
- [12] J. Barroso, E. L. Dagless, A. Rafael, J. Bulas-Cruz, "Number plate reading using computer vision," in *Proc. IEEE Int. Symp. Ind. Electron.*, 1997, pp. 761–766.

- [13] C. Anagnostopoulos, I. Anagnostopoulos, E. Kayafas, and V. Loumos, "A license plate recognition system for intelligent transportation system applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 3, pp. 377-392, Sept 2006.
- [14] C. Anagnostopoulos, T. Alexandropoulos, S. Boutas, V. Loumos, and E. Kayafas, "A template-guided approach to vehicle surveillance and access control," in *Proc. IEEE Conf. Advanced Video Signal Based Surveillance*, 2005, pp. 534-539.
- [15] C. Anagnostopoulos, I. Anagnostopoulos, D. Vergados, E. Kayafas, and V. Loumos, "Sliding windows: A software method for real-time inspection in textile surface," *Textile Res. J.*, vol. 74, no. 7, pp. 646-651, Jul 2004.
- [16] I. K. Yalcinn and M. Gokmen, "Integrating differential evolution and condensation algorithms for license plate tracking," in *Proc. 18th ICPR*, Hong Kong, 2006, pp. 658-661.
- [17] M. Isard and A. Blake, "Condensation—Conditional density propagation for visual tracking," *Int. J. Comput. Vis.*, vol. 29, no. 1, pp. 5-28, Aug 1998.
- [18] R. Storn and K. Price, "Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces," *Int. Comput. Sci. Inst.*, Berkeley, CA, Tech. Rep. TR-95-012, 1995.
- [19] F. Kahraman, B. Kurt, and M. Gokmen, "License Plate Character Segmentation Based on the Gabor Transform and Vector Quantization," *Lecture Notes In Computer Science*, vol. 2869, pp. 381-388, 2003.
- [20] T. D. Duan, T. L. H. Du, T. V. Phuoc, and N. V. Hoang, "Building an automatic vehicle license-plate recognition system," in *Proc. Int. Conf. Comput. Sci. (RIVF)*, 2005, pp. 59-63.
- [21] V. Kamat and V. S. Ganesan, "An efficient implementation of the Hough transform for detecting vehicle license plates using DSP'S," in *Proc. Real-Time Tech. and Appl. Sym.*, 1995, pp. 58-59.
- [22] Y. Cheng, J. Lu, and T. Yahagi, "Car license plate recognition based on the combination of principal component analysis and radial basis function networks," in *Proc. ICSP*, 2004, pp. 1455–1458.
- [23] C. R. Jung and R. Schramm, "Rectangle detection based on a windowed Hough transform," in *Proc. 17th SIBGRAPI*, 2004, pp. 113-120.
- [24] M. Rouhani, "A fuzzy feature extractor neural network and its application in license plate recognition," in *Computational Intelligence, Theory and Application*, Berlin, Germany, 2006, pp. 223-228.

- [25] C.-T. Hsieh, Y.-S. Juan, and K.-M. Hung, "Multiple license plate detection for complex background," in *19th International Conference on Advanced Information Networking and Applications*, 2005, pp. 389 - 392.
- [26] D.-S. Kim and S.-I. Chien, "Automatic car license plate extraction using modified generalized symmetry transform and image warping," in *Proc. ISIE*, 2001, pp. 2022-2027.
- [27] X. Shi, W. Zhao, and Y. Shen, "Automatic License Plate Recognition System Based on Color Image Processing," *Lecture Notes in Computer Science*, vol. 3483, pp. 307-314, 2005.
- [28] N. Zimic, J. Ficzkowski, M. Mraz, and J. Virant, "The fuzzy logic approach to the car number plate locating problem," in *Proc. IIS, Grand Bahama Island*, 1997, pp. 227-230.
- [29] J. A. G. Nijhuis, M. H. ter Brugge, K. A. Helmholt, J. P. W. Pluim, L. Spaanenburg, R. S. Venema, and M. A. Westenberg, "Car license plate recognition with neural networks and fuzzy logic," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 5, 1995, pp. 2232–2236.
- [30] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, "Automatic License Plate Recognition," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 1, pp. 42-53, Mar 2004.
- [31] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790-799, Aug 1995.
- [32] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603-619, May 2002.
- [33] W. Jia, H. Zhang, X. He, and M. Piccardi, "Mean shift for accurate license plate localization," in *Proc. 8th Int. IEEE Conf. Intell. Transp. Syst.*, Vienna, Austria, 2005, pp. 556-571.
- [34] P. Viola and H.J. Lee, "Robust Real-Time Face Detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137-154, May 2004.
- [35] L. Dlagnekov, "License Plate Detection Using AdaBoost," Comput. Sci. Eng. Dept., Univ. California, San Diego, May 2004.
- [36] LC On, CH Weng, TS Kuan, K Ng C Wu, "A Macao license plate recognition system," in *Proc. 4th Int. Conf. Mach. Learn. Cybern*, Guangzhou, China, Aug. 18–21, 2005, pp. 4506–4510.

- [37] H. Zhang, W. Jia, X. He, and Q. Wu, "Learning-based license plate detection using global and local features," in *Proc. 18th ICPR*, vol. 2, Hong Kong, 2006, pp. 1102–1105.
- [38] S.Z. Wang and H. J. Lee, "A cascade framework for a real-time statistical plate recognition system," *IEEE Trans. Inf. Forensics Security*, vol. 2, no. 2, pp. 267–282, Jun 2007.
- [39] K. I. Kim, K. Jung, and J. H. Kim, "Color Texture-Based Object Detection: An Application to License Plate Localization," in *Pattern Recognition with Support Vector Machines.*: Springer, 2002, vol. 2388, pp. 321-335.
- [40] G.R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," in *Proc. IEEE Workshop Appl. Comput. Vis.*, 1998, pp. 214-219.
- [41] M. H. ter Brugge, J. H. Stevens, J. A. G. Nijhuis, and L. Spaanenburg, "License plate recognition using DTCNNs," in *Proc. IEEE Int. Workshop Cellular NNs Appl.*, 1998, pp. 212-217.
- [42] K.K. Kim, K.I. Kim, J.B. Kim, H.J. Kim, "Learning-based approach for license plate recognition," in *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop*, vol. 2, 2000, pp. 614 - 623.
- [43] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust., Speech Signal Process*, vol. 37, no. 3, pp. 328-339, 1989.
- [44] S. H. Park, K. I. Kim, K. Jung, and H. J. Kim, "Locating car license plates using neural networks," *Electron. Lett.*, vol. 35, no. 17, pp. 1475–1477, Aug 1999.
- [45] P. K. Kim, and H. J. Kim E. R. Lee, "Automatic recognition of a car license plate using color image processing," in *Proc. IEEE ICIP*, vol. 2, Nov. 13–16, 1994, pp. 301-305.
- [46] G. Li, R. Zeng, and L. Lin, "Research on vehicle license plate location based on neural networks," in *Proc. 1st ICICIC*, Beijing, China, 2006, pp. 174-177.
- [47] C. Oz and F. Ercal, "A practical license plate recognition system for real-time environments," in *Proc. IWANN*, 2005, pp. 881-888.
- [48] C. Garcia and M. Delakis, "Convolutional face finder: a neural architecture for fast and robust face detection," *IEEE Trans. Pattern Anal. Mach Intell.*, vol. 26, no. 11, pp. 1408-1423, Nov 2001.
- [49] Y.-N. Chen, C.-C. Han, C.-T. Wang, B.-S. Jeng, and K.-C. Fan, "The application

of a convolution neural network on face and license plate detection," in *Proc. 18th ICPR*, Hong Kong, 2006, pp. 552-555.

- [50] Nomura, K. Yamanaka, O. Katai, H. Kawakami, and T. Shiose, "A novel adaptive morphological approach for degraded character image segmentation," *Pattern Recognit.*, vol. 38, no. 11, pp. 1961–1975, Nov. 2005., vol. 38, no. 11, pp. 1961-1975, Nov 2005.
- [51] N. Bellas, S. M. Chai, M. Dwyer, and D. Linzmeier, "FPGA implementation of a license plate recognition SoC using automatically generated streaming accelerators," in *Proc. 20th IPDPS*, Nice, France, 2006, pp. 8-15.
- [52] J. Kong, X. Liu, Y. Lu, and X. Zhou, "A novel license plate localization method based on textural feature analysis," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol.*, Athens, Greece, 2005, pp. 275-279.
- [53] Hegt, H.A., de la Haye, R.J. and Khan, N.A., "A high performance license plate recognition system," in *IEEE International Conference on Systems, Man, and Cybernetics*, 1998, pp. 4357 - 4362.
- [54] A. Rahman, W. Badawy, and A. Radmanesh, "A real time vehicle's license plate recognition system," in *Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance*, 2003., 2003, pp. 163 - 166.
- [55] M. Garcia, and J. L. Alba F. Martin, "New methods for automatic reading of VLP's (Vehicle License Plates)," in *Proc. IASTED Int. Conf. SPPRA*, 2002.
- [56] V. Shapiro and G. Gluhchev, "Multinational license plate recognition system: Segmentation and classification," in *Proc. 17th ICPR*, Cambridge, U.K., 2004, pp. 352–355.
- [57] S. Nomura, K. Yamanaka, O. Katai, and H. Kawakami, "A new method for degraded color image binarization based on adaptive lightning on grayscale versions," *IEICE Trans. Inf. Syst*, vol. E87-D, no. 4, pp. 1012– 1020, Apr. 2004.
- [58] P. Soille, *Morphological Image Analysis: Principles and Applications*. Berlin, Germany: Springer-Verlag, 1999.
- [59] R. C. Gonzalez and R. E. Woods, *Digital image processing (2nd ed.)*. Boston, MA, USA: Addison-Wesley, 1993.
- [60] A. Capar and M. Gokmen, "Concurrent segmentation and recognition with shape-driven fast marching methods," in *Proc. 18th ICPR*, vol. 1, Hong Kong, 2006, pp. 155-158.

- [61] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," in *Proc. Nat. Acad. Sci.*, vol. 93, 1996, pp. 1591–1595.
- [62] P. Stec and M. Domanski, "Efficient unassisted video segmentation using enhanced fast marching," in *Proc. ICIP*, vol. 3, 2003, pp. 427-430.
- [63] J. Kittler, J. Illingworth, and J. Foglein, "Threshold selection based on a simple image statistic," *Comput. Vis. Graph. Image Process.*, vol. 30, no. 2, pp. 125-147, May 1985.
- [64] T. Naito, T. Tsukada, K. Yamada, K. Kozuka, and S. Yamamoto, "Robust license-plate recognition method for passing vehicles under outside environment," *IEEE Trans. Veh. Technol.*, vol. 49, no. 6, pp. 2309-2319, Nov 2000.
- [65] D. Llorens, A. Marzal, V. Palazon, and J. M. Vilar, "Car License Plates Extraction and Recognition Based on Connected Components Analysis and HMM Decoding," in *Lecture Notes on Computer Science.*, 2005, vol. 3522, pp. 571–578.
- [66] C. Coetzee, C. Botha, and D. Weber, "PC based number plate recognition system," in *Proc. IEEE Int. Symp. Ind. Electron.*, 1998, pp. 605-610.
- [67] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 1, pp. 62-66, Jan 1979.
- [68] Niblack, W., *An Introduction to Digital Image Processing*. Englewood Cliffs: Prentice Hall, 1986.
- [69] B. R. Lee, K. Park, H. Kang, H. Kim, and C. Kim, "Adaptive Local Binarization Method for Recognition of Vehicle License Plates," in *Combinatorial Image Analysis.*: Springer, pp. 646-655.
- [70] Y. Nakagawa and A. Rosenfeld, "Some experiments on variable thresholding," *Pattern Recognit.*, vol. 11, no. 3, pp. 191-204, 1979.
- [71] Y. Cui and Q. Huang, "Extracting characters of license plates from video sequences," *Mach. Vis. Appl.*, vol. 10, no. 5/6, pp. 308-320, Apr. 1998.
- [72] K. V. Mardia and T. J. Hainsworth, "A spatial thresholding method for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 6, pp. 919-927, Nov 1988.
- [73] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 96-101, Jan. 1994.
- [74] V. Franc and V. Hlavac, "License Plate Character Segmentation Using Hidden Markov Chains," in *Pattern Recognition*. Berlin, Germany: Springer-Verlag, 2005,

pp. 385-392.

- [75] K. B. Kim, S. W. Jang, and C. K. Kim, "Recognition of car license plate by using dynamical thresholding method and enhanced neural networks," in *Computer Analysis of Images and Patterns*. Eds. New York: Springer-Verlag, 2003, vol. 2756, pp. 309-319.
- [76] X. Pan, X. Ye, and S. Zhang, "A hybrid method for robust car plate character recognition," *Eng. Appl. Artif. Intell.*, vol. 18, no. 8, pp. 963-972, Dec. 2005.
- [77] O. D. Trier, A. K. Jain, and T. Taxt, "Feature extraction methods for character recognition—A survey," *Pattern Recognit.*, vol. 29, no. 4, pp. 641-661, Apr. 1996.
- [78] A. Broumandnia and M. Fathy, "Application of pattern recognition for Farsi license plate recognition," in *Proc. Int. Conf. GVIP*, Cairo, Egypt, 2005.
- [79] W. Wei, Y. Li, M. Wang, and Z. Huang, "Research on number-plate recognition based on neural networks," in *Proc. IEEE Signal Process. Soc. Workshop Neural Netw. Signal Process. XI*, 2001, pp. 529-538.
- [80] T. Sirithinaphong and K. Chamnongthai, "The recognition of car license plate for automatic parking system," in *Proc. 5th Int. Symp. Signal Process. Appl.*, 1999, pp. 455-457.
- [81] T. Sirithinaphong and K. Chamnongthai, "Extracting of car license plate using motor vehicle regulation and character pattern recognition," in *Proc. IEEE Asia-Pacific Conf. Circuits Syst.*, 1998, pp. 559-562.
- [82] D. E. Rumelhart and J. L. McClelland, "Parallel distributed processing," in *Explorations in the Microstructure of Cognition*. Cambridge, MA: MIT Press, 1986.
- [83] M. Raus and L. Kreft, "Reading car license plates by the use of artificial neural networks," in *Proc. 38th Midwest Symp. Circuits Syst.*, 1995, pp. 538-541.
- [84] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cogn. Sci.*, vol. 11, no. 1, pp. 23-63, Jan./Mar. 1987.
- [85] D. F. Specht, "Probabilistic neural networks," *Probabilistic neural networks*, vol. 3, no. 1, pp. 109-118, 1990.
- [86] C. Anagnostopoulos, I. Anagnostopoulos, G. Tsekouras, G. Kouzas, V. Loumos, and E. Kayafas, "Using sliding concentric windows for license plate segmentation and processing," in *Proc. IEEE Workshop Signal Process. Syst. Des. Implementation*, Athens, Greece, 2005, pp. 337-342.

- [87] Y. Hu, F. Zhu, and X. Zhang, "A Novel Approach for License Plate Recognition Using Subspace Projection and Probabilistic Neural Network," in *Advances in Neural Networks*, J. Wang, X. Liao, and Z. Yi, Eds.: Springer-Verlag, 2005, pp. 216-221.
- [88] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York: Oxford Univ. Press, 1997.
- [89] Y.-P. Huang, S.-Y. Lai, and W.-P. Chuang, "A template-based model for license plate recognition," in *Proc. IEEE Int. Conf. Netw., Sens. Control*, 2004, pp. 737-742.
- [90] Young-Mo Kim Mi-Ae Ko, "License plate surveillance system using weighted template matching," in *Proceedings 32nd Applied Imagery Pattern Recognition Workshop*, 2003, pp. 269 - 274.
- [91] Fixed Plate Hunter-900 - ELSAG North America. [Online]. <http://www.elsag.com/fixed.htm> Accessed [27 Apr. 2012].
- [92] AutoPlate - PIPS Technology. [Online]. http://www.pipstechnology.co.uk/products.php?section_id=7&article_id=37 Accessed [27 Apr. 2012].
- [93] CarDetector - Vigilant Video. [Online]. <http://www.vigilantvideo.com/prod-detectr.htm> Accessed [27 Apr. 2012].
- [94] SeeCar - Hi-Tech Solutions. [Online]. <http://www.htsol.com/Products/SeeCar.html> Accessed [27 Apr. 2012].
- [95] MW810 Mobile Workstation - Motorola Solutions USA. [Online]. http://www.motorola.com/Business/US-EN/Business+Product+and+Services/Mobile+Computers/Vehicle-mounted+Computers/MW810+Mobile+Workstation_US-EN Accessed [27 Apr. 2012].
- [96] Pips Technology. [Online]. <http://pipstechnology.com> Accessed [27 Apr. 2012].
- [97] CarDetector - Mobile LPR Systems. [Online]. http://www.vigilantvideo.com/CarDetector_Mobile.htm Accessed [27 Apr. 2012].
- [98] PlateScan LLC. [Online]. <http://www.platescan.com/> Accessed [27 Apr. 2012].
- [99] To Catch a Car Thief ... Up to 3600 a Minute, Actually. [Online]. http://images.businessweek.com/mz/10/45/1045_40techlowenforcement.pdf?channel=magazine+channel_news+-+technology Accessed [27 Apr. 2012].

- [100] P. V. Suryanarayana, S. K. Mitra, A. Baneree, and A. K. Roy, "A Morphology Based Approach for Car License Plate Extraction," in *Proc. IEEE Indicon*, Chennai, India, 2005, pp. 24-27.
- [101] Image Processing Toolbox. [Online]. <http://www.mathworks.com/products/image/> Accessed [27 Apr. 2012].
- [102] P.V.C. Hough, "Method and means for recognizing complex patterns," 3069654, 1962.
- [103] D.H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes ," *Pattern Recognition*, vol. 13, no. 2, pp. 111-122, 1981.
- [104] PE Hart RO Duda, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11-15, January 1972.
- [105] About Embedded MATLAB. [Online]. <http://www.mathworks.com/products/featured/embeddedmatlab/index.html> Accessed [27 Apr. 2012].
- [106] Altera DE2-70 Board. [Online]. <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=53&No=226> Accessed [27 Apr. 2012].
- [107] Nios II Processor Reference Handbook. [Online]. http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf Accessed [27 Apr. 2012].
- [108] SOPC Builder User Guide. [Online]. http://www.altera.com/literature/ug/ug_socp_builder.pdf Accessed [27 Apr. 2012].
- [109] (2011, May) Avalon Interface Specifications - Altera. [Online]. www.altera.com/literature/manual/mnl_avalon_spec.pdf Accessed [27 Apr. 2012].
- [110] AN 527: LCD Controller Replacement - Altera. [Online]. www.altera.com/literature/an/an527.pdf Accessed [27 Apr. 2012].
- [111] DSP Builder Handbook Volume 1: Introduction to DSP Builder (PDF). [Online]. http://www.altera.com/literature/hb/dspb/hb_dspb_intro.pdf Accessed [27 Apr. 2012].
- [112] PowerPlay Power Analysis - Altera. [Online]. http://www.altera.com/literature/hb/qts/qts_qii53013.pdf Accessed [27 Apr. 2012].
- [113] Nios II Custom Instruction User Guide - Altera. [Online]. www.altera.com/literature/ug/ug_nios2_custom_instruction.pdf Accessed [27 Apr. 2012].

2012].

- [114] D. Kim, Y. Ryu, and G. Kim, S. Kim, "A robust license-plate extraction method under complex image conditions," in *Pattern Recognition Proceedings. 16th International Conference*, 2002.
- [115] Guanghua Zhang, Bin Liu Xiaobo Lu, "Localization of Vehicle License Plate Based on Gray Level Variation," in *International Conference on ITS Telecommunications Proceedings*, Chengdu, 2006, pp. 1220-1223.
- [116] Liu Changping Bai Hongliang, "A hybrid license plate extraction method based on edge statistics and morphology," in *Proc.ICPR*, 2004, pp. 831- 834.
- [117] D. Tran Le Hong D. Duong Anh, "Combining Hough transform and contour algorithm for detecting vehicles' license-plates," in *ISIMP*, 2004, pp. 747-750.
- [118] J Yu-Shan, H Kuo-Ming H Ching-Tang, "Multiple license plate detection for complex background," in *19th International Conference on Advanced Information Networking and Applications*, 2005, pp. 389 - 392.
- [119] (2011, May) Avalon Interface Specifications - Altera. [Online]. www.altera.com/literature/manual/mnl_avalon_spec.pdf Accessed [27 Apr. 2012].
- [120] Optasia IMPS. [Online]. <http://www.singaporegateway.com/optasia/imps.html> Accessed [27 Apr. 2012].
- [121] AlertVU - L-3 Mobile Vision. [Online]. http://www.mobile-vision.com/pdf/L-3MV_ALPR_datasht.pdf Accessed [27 Apr. 2012].
- [122] Optasia IMPS. [Online]. <http://www.singaporegateway.com/optasia/imps.html> Accessed [27 Apr. 2012].

Appendix

A.1 Characters Used in Singapore Vehicle License Plate

Character	Gray Level Image	Vertical Lines in the Character	Horizontal Lines in the Character
0
1
2
3
4
5
6

Character	Gray Level Image	Vertical Lines in the Character	Horizontal Lines in the Character
7
8
9
A
B
C
D
E

Character	Gray Level Image	Vertical Lines in the Character	Horizontal Lines in the Character
F
G
H
J
K
L
M (Variant 1)
M (Variant 2)

Character	Gray Level Image	Vertical Lines in the Character	Horizontal Lines in the Character
N
P
Q
R
S
T
U
V

Character	Gray Level Image	Vertical Lines in the Character	Horizontal Lines in the Character
W (Variant 1)
W (Variant 2)
X
Y
Z