# Informatics Simulation & Exploration of Mobile License Plate Detection Employing Infrared, Canny Edge Detection, Binary Threshold and Contour Detection for Submission in Limited Light Conditions

A thesis presented in partial fulfilment of the requirements for the degree of

Master of Engineering

In

Information and Communication

at Massey University, Albany,

New Zealand.

Binny Paul

2011

## Abstract

In modern societies, vehicle surveillance is considered as an important device that allows law enforcement to govern the society and ensure citizens' personal safety. Over the past three decades, there have been many submissions of the research into vehicle surveillance. Such research is typically broken down into two parts: license plate recognition and character recognition. This thesis focuses on licence plate recognition. As surveillance operates 24 hours a day, 365 days a year, which includes night time or limited light conditions, there has been no compelling evidence to show a mobile licence plate recognition system with optimum performance under limited light or night time conditions. Thus, the motivation of this thesis is to ascertain evidence of performance under limited light conditions. The CMOS camera, the infrared lens filter, and the infrared light source are included in the hardware design apparatus of the hardware architecture necessary for collecting image samples of real world settings. To locate the license plate, a software algorithm is envisaged as to reduce image noise and locate license plate in the image. To minimise image noise, the pre-processing techniques applied are the Gaussian blur, pyramid decomposition and up-sampling. To locate a license plate, the two unique techniques investigated are Canny edge detection and the binary threshold. Other algorithms included in the software design are dilation used for filling gaps in edge boundaries, followed by contour sets used for identifying unique object boundaries and simplifying boundary edges by means of the Dogual-Peucker algorithm. To sustain optimum performance, the Canny edge hysteresis threshold level was examined together with the iteration level of the binary threshold. The surveillance model was tested in four different environments, but only three were successful. The average accuracy across the three environment settings (1500 images) was 97.53%, which is above the accepted level in this industry.

## Acknowledgements

First and foremost, I want to thank my supervisors Dr Fakhrul Alam and Dr Johan Potgieter, whose encouragement, supervision and support from the preliminary to the concluding level enabled me to develop an understanding of the subject. It has been an honor to be their student. They have taught me, both consciously and unconsciously, how good research is done.

I owe my deepest gratitude to Nick Ewings and Dhiren Rugnathji, whose enormous effort was invaluable to me at the stage of the data collection.

I appreciate the contribution made by Oliver Grant and Yvette Hodgson, who spent their time proof reading the thesis.

I want to particularly acknowledge the unconditional support from my father Paul, and mother Parimala who made completion of this thesis possible.

# Table of Contents

# List of Figures

**Name** **Page**

# List of Tables

# Chapter 1

## Introduction to License Plate Surveillance System

Private and commercial motor vehicles are frequently used as the transport mechanism in today's society. To govern cities which use heavy motor vehicles, law enforcement authorities require a unique device to monitor or track vehicles randomly at a given location. The applications of such devices are located in the border crossing patrol, petrol station forecourts, automated toll ticketing and stolen vehicle detection (Ho et al., 2009). The New Zealand Police authorities reported that 18,768 vehicles were stolen between June 2009 and June 2010 (New Zealand Police, 2010). This figure averages 51 vehicles per day in addition to vehicles driven without registration or warrant, and disqualified vehicles.

The existence of a motor vehicle surveillance system was first recorded over two decades ago, and to date, vehicle surveillance technology has advanced enormously (Duan, Duc & Du, 2004). During the early development of vehicle surveillance, locations were in static areas of car parks, traffic junctions, motorway entrances and toll booths. In developed countries, surveillance systems are hidden from the public, and locations are unknown, but the coverage is limited in terms of location specifics. Recently, a new development of mobile surveillance was developed by an American commercial company with a costly solution for traffic control police officers (ELSAG, 2010). The adaptation of this device has been slow primarily due to technical challenges in design complexity and accuracy.

The research field in license plate surveillance has expanded in the last decade, and the outcomes of the research results are restricted in operation due to either hardware or software limitations. These limitations are found to be related to changeable indoor and outdoor settings, stationary and movable backgrounds, various types of illumination, varied vehicle speeds, titled camera angles with various distance between camera and vehicle (Duan et al., 2004), to mention a few. A typical mobile surveillance system operates 24 hours a day. A great deal of literature has been written on the day light surveillance. The night time or limited light surveillance has been studied, but no inclusive evidence has been

shown with regard to performance under various environmental conditions. Consequently, the motivation of this dissertation is to investigate a mobile surveillance model which could operate in limited light or at night time, and carry out real world simulation to improve the outcome of preceding research findings.

The mobile surveillance system model consists of two components that rely on each component performance to produce optimum end results. The components consist of hardware and software. The hardware component is outlined as an input device of a video stream of photos to be analysed by the software component.

The input images are relayed by the camera with an external illumination source for night time surveillance, for which an infrared spot light is usually utilised. The image quality is adapted by reducing external image noise (e.g. headlights, break lights, traffic lights), vibration from vehicle engine, illumination distortion, and weather conditions. The input image will influence efficiency of the software system in relation to precession and accuracy of the surveillance system.

The traditional software system is built from iterative complex algorithms to locate the license plate. This process is broken into two steps required to fulfil the requirements of the software system: firstly, the image is cleansed to remove as much noise interference from the image as possible, and secondly, the image is poised to identify the license plate. The challenges created by the software algorithm are a plate orientation, varied license plate sizes, a greyscale image (limited light conditions), a complex background, various image noise, as noted in relation to the hardware device, and most importantly high accuracy. The existing literature reveals models that use edge information, borderline mapping, wave transform and the AdaBoost machine vision to detect the license plate from the image. A more in depth analysis is carried out in Chapter 2, followed by the research model in Chapter 4.

The purpose of this research is to design a surveillance system able to operate in different locations and variable environment settings, with hardware filters to reduce object noise captured by the camera and image noise by using software filters. To determine the existence of a license plate, the image is scanned and traced for object border boundaries.

The boundaries are then mapped and validated to determine license plate detection. In this research study, the methodology was tested in four environmental settings with varied parameter values to find the optimum region, and accomplish a high license plate detection accuracy rate.

Having provided an overview of some of the challenges posed by license plate detection systems, Chapter 2 provides an outline of the current state of the art in static mobile day light systems with the development of hardware and software models. The hardware architecture, i.e. the setup, the coverage enhancement, the infrared filter, the light source and the camera, is proposed in Chapter 3, along with a software theory on image noise, i.e. the Gaussian blur, down-sampling, and up-sampling, and license plate detection, i.e. the edge trail using the Canny and binary threshold, border mapping using chain code contours, simplifying border detection by using the Douglas-Peucker algorithm, which is all documented in Chapter 4. The research objective, research questions, proposed methodology and preliminary results are given in Chapters 5 and 6. Finally, discussion of the research findings, the future model considerations and conclusions are documented in Chapter 7.

# Chapter 2

# Literature Review

## 2.1 Overview of surveillance systems

The growth in vehicle surveillances over the last three decades has encouraged researches to produce various surveillance system models. Although there are many models to choose from, care has been taken to research the literature completely and filter the models which do not show evidence of proper research techniques or produce elevated results without solid evidence of research work. However, there are commercial models not within the scope of this literature review due to privacy and secrecy of the model design to stay ahead from the competitors. In some cases, commercial products overestimate their surveillance model designs to increase sales.

## 2.2 Static vs. Mobile – (License Plate Recognition System)

Static surveillance systems, such as border crossing control, traffic management, identification of stolen vehicles, speed control, red light monitor and automated parking systems are integrated in a developed community (Mello & Costa, 2009). However, in 2008 one North American company moved away from static vehicle surveillance, and introduced mobile surveillance. Their website reported capturing 405,295 unique license plates in five months (ELSAG, 2010). The ability to covert mobile surveillance into static surveillance is achievable, but conversion of static systems to mobile surveillance results in performance deterioration. Thus, the ground design of surveillance system is built differently with advantage of mobile surveillance which can operate in various surroundings, wherever static surveillance is location specific.

Another benefit from mobile surveillance is its ability to operate without public awareness, and the location is different at different times. The faculty for covering a wide area enables the law enforcement to capture more vehicles that are illegally driven (ELSAG, 2010).

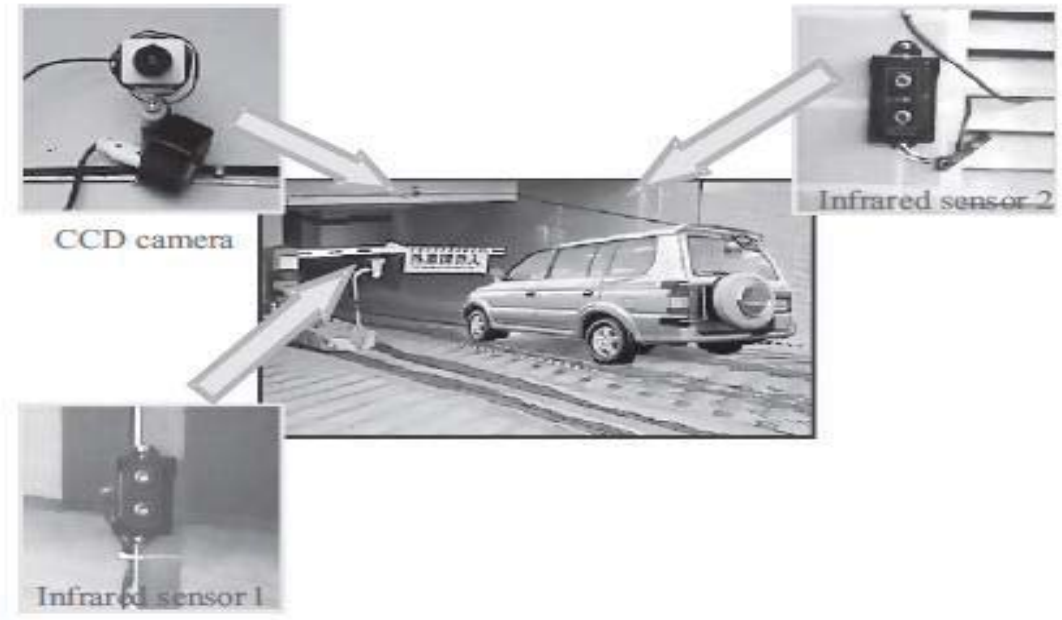



**Figure 2. 1: Static surveillance with infrared sensors and camera (Wu et al., 2007)**

**Figure 2. 2: Mobile surveillance camera (ELSAG, 2010)**

## 2.3 Hardware Components

Assembling a surveillance system requires hardware components. Each component plays a vital role in completing the system. The system components are a camera for capturing the image, an external light source to illuminate the surroundings, a camera position which enables the camera to effortlessly view the license plate, and a computer to feed in the image to process and locate the license plate. Each component is reviewed individually to address hardware materials used to complete a surveillance system.

### 2.3.1 Camera

A SONY XCD/X700 IEEE 1394 camera is used in the research carried out by Anagnostopoulos, Anagnostopoulos, Psoroulas and Loumous (2008) to collect sample images. The camera frame rate is 15fps, the shutter speed ranges from 1/100,000 to 2s, and

the range is divided into high, standard and low speed. Figure 2.3 shows a high response for wavelength around 500nm, but near the infrared wavelength (> 750 nm), the relative response deteriorates and is unusable in limited light conditions.

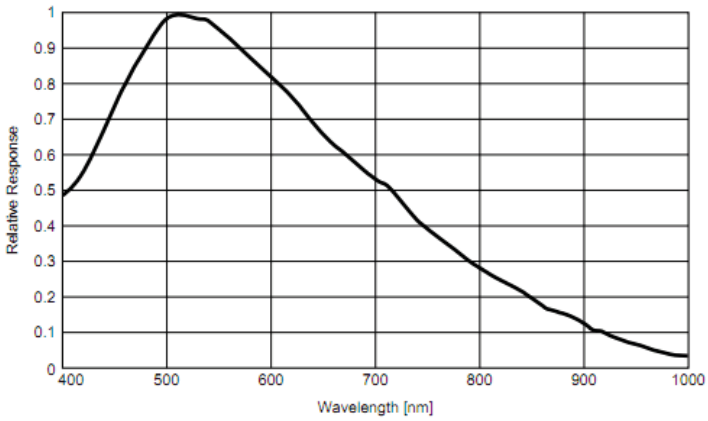



**Figure 2. 3: XCD / X700 Wavelength vs relative response (Sony Corporation, 2000)**

Duan et al. (2004) used Sony DC W350, which is a hand held digital camera. The camera was used to take sample photos manually, and with the distance range between 2- 5 meters.

Without specifying the camera details, other researchers recommend the CCD camera, and the shutter speed of 1/1000s (Naito et al., 2000) which will capture fast moving vehicles without blurriness.

### 2.3.2 External Light Source

The external light source illuminates the surrounding, and there are two techniques documented in different reviews. One is creating the infrared light source, and another is using synchronous flash lights.

The infrared light source is a popular solution for illuminating the surrounding in limited light conditions. The properties of the infrared light are suitable for license plate recognition because the frequency is not visible to human beings. A human eye will respond to wavelengths from 390 to 750nm, while the frequency for surveillance system is between 800 to 950nm (Starr, 2005).

The synchronous control flash light approach (Huansheng & Guoqiang, 2005) is applied in order to take a photo of a vehicle, which is triggered by the sensors. A series of multiple flash light shots needs to be completed to detect the licence plate. The technique then subtracts one flash shot image with a non-flash shot image, and by this process, the background is filtered along with most of the image of the motor vehicle's body, which results in an image with an isolated license plate, as shown in Figure 2.4.

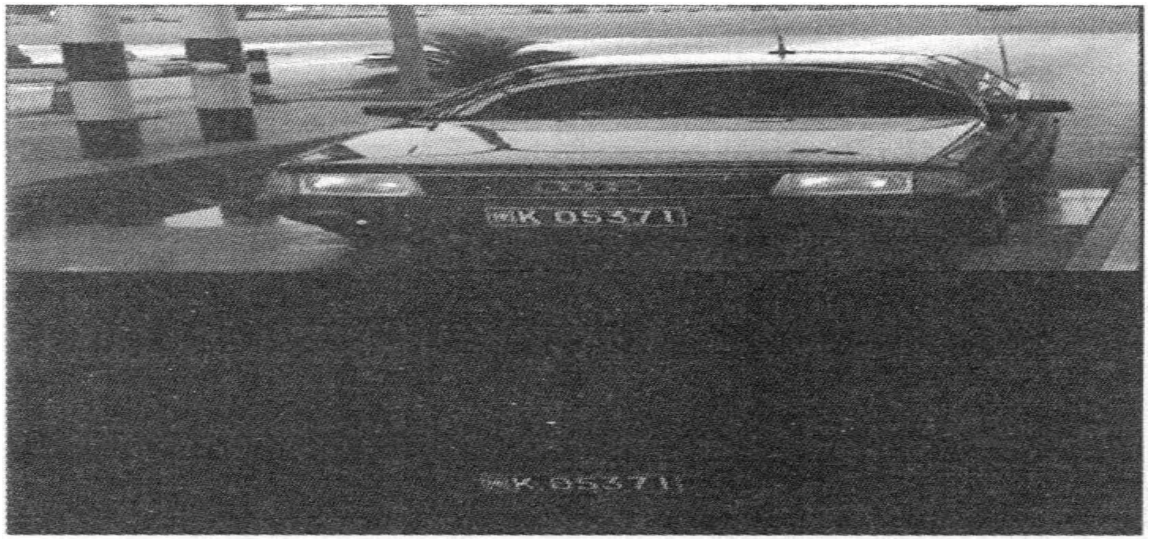



**Figure 2. 4: Top natural image, bottom result of subtract natural image with flash light image (Huansheng & Guoqiang, 2005)**

The external light source is required for limited light conditions, and it allows a choice of the infrared source light and the flash light. The infrared light source is invisible to the human eye so that the illumination will not distract drivers on the road. In contrast, the flash light is visible while in operation, and can be dangerous if drivers get distracted by it.

However, the reviews are not completely complacent with this information. The distance range for illumination settings, the effect of moving vehicles on illumination and the effect of weather conditions on illumination are not discussed in the literature. This area requires more in-depth analysis in order that the external light source is developed, and the coverage and limitations of such devices are determined, which will be discussed in Chapter 3.

### 2.3.3 Camera Position

The positions of the camera can vary depending on the requirements of the surveillance coverage. ELSAG's (2010) mobile surveillance places cameras on their front bonnets with two cameras tilted to either side of the vehicle with the aim of capturing as many license plates as possible. To steer and improve the coverage, the camera is positioned at different locations to cover all possible areas.

In mobile surveillance, the camera position is volatile, and hence the software algorithm is expected to work under various image angles and positions of the camera (ELSAG, 2010). However, the image captured by the camera needs to be realistic so that the visibility of the license plate is available for the software algorithms to detect the license plate.

### 2.3.4 Image Process Time

Software models for surveillance of license plates process an image within 50ms (Zheng, Zhao & Wang, 2005; Arth, Limberger & Bischof, 2007; Wang & Lee, 2007). During this period of time, the model recognises the license plate, and then completes the character recognition. During 50ms, a surveillance model can operate a camera video input of 20 fps. The surveillance model is expected to process a single image in 50ms or less. The time achieved with a good algorithm design and growth in computing process power is also an advantage.

## 2.4 Software Design –Image Processing Algorithms

This section will review the current and historical license plate detection algorithms, and attempt to highlight the areas relevant to this research study. The importance of the software design is critical to the success of the overall surveillance system. The license plate recognition algorithm has a growing community with different types of research and solutions to common problems, which will be researched in this thesis. However, as it is not possible to cover all the available literature, only the studies relevant to this research field have been selected.

Review of the existing literature has enabled the researcher to sum up the research intention, as presented in Table 2.1. The research is strictly measured against the specifications and requirements for a better understanding of the available literature and for the purpose of discovering gaps in it. While most systems for licence plate recognition have two software components, license plate location and character recognition, only the former is the scope of this review.

Under normal circumstances, the algorithm design is based on specifications or requirements. Therefore, in order to accurately examine the literature, the following specifications and requirements are addressed.

| SR[#] | (Specifications / Requirements)<br>The list of requirements for the software algorithm studied |
|---|---|
| 1 | Different plate sizes are detectable. The change in plate size is caused by distance variations between camera and the license plate. |
| 2 | The plate orientation in angles is between 0 and 360 degrees. Motor vehicles, motorbikes and other vehicles position the license plate in ways different from the standard horizontal position. |
| 3 | The infrared driven illumination produce a greyscale image captured on the camera. |
| 4 | The embedded image noise comes in different shapes and forms. The filter image noise can improve the license plate detection. |

| | |
|---|---|
| 5 | Adaptability to different environmental conditions. |
| 6 | Performance measure is preferably over the 90% accuracy rate. |

**Table 2.1 Algorithm specifications / requirements**

### 2.4.1 Model 1: Edge Detection, Greyscale Transformation, Mathematical Morphology, Window Size and Region of Interest (ROI)

A study conducted by Qui, Sun and Zhou (2009) was the first to convert the image to greyscale. After the conversion, according to assumption (1), a license plate has more vertical edges, and regions with cars without a license plate have more horizontal edges. By using the Sobel vertical edge detector, the image can be transformed. A weaker illuminated image after edge conversation encompasses illumination values that range from 60 to 120. However, it is not a clear definition of edge values, and the range is scaled up by using full greyscale values from 0 to 255. According to assumption (2), the license plate is generally located at the bottom in order to improve the computing process time, and start edge transformation from one third height of the whole image. Then by using the erosion and dilation operator of mathematical morphology, dilation is performed on the image to fill the holes found in the license plate in between the license plate numbers. The dilation operator is selected based on assumption (3) with regard to the width and the height of the license plate. The image remaining will have the object's blocks filled in with white pixels. Each individual block is analysed separately with the original illumination values by using the vertical histogram counting the pixels. The pixel count identifies blocks of characters, and then, if character blocks are found and meet the threshold limit of 4 to 6 characters, a license plate is identified.

After Sobel edge detection, another approach, applied by Rattanathammawat and Chlidabhongse (2004), was a window size scan. The image is scanned with a pre-set window size and examined with horizontal projections of the edges − if the horizontal projections match the license plate horizontal projection, then it is assumed that a license plate is detected. For further validation, one study examines four consecutive image frames, and performs similar window scans. As the car is moving during this time, the license plate

is located at a different part of the image. If four horizontal projections approximately match, a license plate is considered detected. However, this research is related to static surveillance.

The Model 1 technique demonstrates that it is suitable for the static surveillance system, i.e. assumption (1, 2) of plate location will be horizontal and located near the bottom of the image, and this does not comply with SRs [1 and 2]. Model 1 is permitted for the day light surveillance system, whereas the night time surveillance is not considered, and thus it does not meet SRs [3, 4, and 5]. However, the study produces 98.1% accuracy of the tested images. A similar study of Model 1 produces 83.5% accuracy (Faradji et al., 2007). The resulting inconsistency is explained by test samples used in each study, and hence, the image sample collection is another factor to consider in this research. These model test images are collected in good illumination conditions with no embedded noise, which is a rare occurrence in real life conditions.

### 2.4.2 Model 2: Hough Transform and Contour Algorithm

Duan et al. (2004) first apply the image pre-processing method, which is a process of converting the image from colour to greyscale. Similarly to the steps of Model 1, the image is then extracted of edges by using the Sobel detector, but after thresholding, the edges are embedded into binary values (black or white pixels).

The contour algorithm is completed to find closed boundary line objects; however, the consideration of a poor image quality is factored, and therefore Hough transform is used to locate any parallel lines that could possibly represent the border lines of the license plate. As this process is time consuming, Duan et al., (2004) managed to speed it up by only examining the found contour sets, and then by locating the Hough coordinates. This process eliminates Hough transform to scan the full image. Incorrect detection is high because objects similar to license plate boundaries appear in the image, such as the head light, break lights, and window glass, which have two parallel lines as well. To overcome this obstacle, the ratio matching is done so that the license plate width and height are the fixed ratio,

regardless of the size of the license plate. This ratio matching thus eliminates incorrect license plate matches.

Since the test sample produced 98.76% accuracy, Model 2 has solid techniques and satisfies SR [1], and SR [2] to some extent, while the test images have the maximum angle of 30 degrees. However, not only is Model 2 designed for the day light solution and the factors SR [4], but also SR [5] is not clearly defined, and the model is suited to the static system. It therefore performed well above the expected SR [6], but the computing process time increased by 0.62s.

### 2.4.3 Model 3: Colour Edge Detection, Colour Model Transform and Fuzzy Aggression

So far, Model 1 has searched for the license plate using edge information and aimed to match edge variations to represent the license plate. In Model 2, the boundary of the license plate is tracked, by using edge information and by matching parallel lines that could represent the license plate. A different approach is undertaken by Chang, Chen, Chung and Chen (2004); that is, instead of just looking for edges, it is uses colour classifiers. In a given image, if the license plate is present, it is represented by an alphanumerical identification. The classifier begins to search for particular colours and changes that occur between the characters of the license plate. However, this approach is less useful at night as the colour variation is limited to greyscale values. The edge information is also utilised by Chang et al. (2004), and combined with colour classifiers to sustain the fuzzy aggression rule engine. The success of the system depends on the rule engine to tune to particular edge and colour information; if the rule is contented then a license plate is detected. The high accuracy is achieved by very specific license plate tuning so that road signs or other characters in the image are not considered as a license plate.

Chang et al.'s (2004) approach produced the overall accuracy of 97.6%, with the computing process time of 2.4s. The complexity of fuzzy aggression is the rule engine, and the increased computation time is due to the intermediate steps of gathering edge and colour information. However, the test images were observed with a different plate size, and found

to satisfy SR [1], but the angle rotation was not properly handled, as the assumption was made on the positions of the license plate to be horizontal, which is not the case at all times. As a result, SR [2] performance was poor, the fuzzy aggression rule engine is dependant on colour information, and SR [3] is then unsuccessful. Concerning SR [4], embedded image noise is not addressed in the study, and the test images show a clear view of the license plate but with complex backgrounds. The accuracy may drop further if tested with a poor image quality where the license plate colours are mutilated compared with the normal values, and therefore SR [5] is not a success point.

### 2.4.4 Model 4: Histogram Equalisation, Median Filtering, Edge Detection, Gradient Filter and Horizontal Line check

Guo & Liu (2008) use image pre-processing before locating the license plate. In this case, histogram equalisation is done to improve image contrast when the illumination is low, followed by the median filter to remove noise but still preserve the details on the image. To eliminate unwanted information from the image, edge detection is done and then gradient of each edge is calculated. The gradient threshold is set at the beginning, and any gradients below the threshold is removed from the image so that remaining gradients are possible license plate candidates. To confirm the license plate location, a horizontal line is scanned through the original pixel values. The line measures pixel variations between the license plate numbers and letters, the variations is measured against the ratio. If the ratio and the measure are fairly accurate, the location of the license plate is found.

The distance between the camera and the plate is 2 – 5 meters, with the overall accuracy rate of 97.1%. This model also assumes that the license plate location is horizontal at all times, as it is the case in Model 1 and 3. The distance range is small, and does not guarantee SR [1], while SR [2] is unsuccessful in this model. However, SRs [3, 4, and 6] is completed to a satisfactory extent, image noise is considered and filtered by means of the median filter, and greyscale images are operable. The accuracy rate of 97.1% meets the standard of SR [6], but the test images are of a controlled nature due to distance limitation and environmental settings.

### 2.4.5 Model 5: Wavelet Transformation

Hung and Hsieh (2010) utilise mobile surveillance to operate their surveillance unit by means of a camera located on the front bonnet facing the road. To eliminate the background image, a restrictive filter transforms the colour format into YUV. This colour format allows scanning the image to locate distinct red lights. The red light represents the brake lights of a motor vehicle. When two brake lights are found by means of firm parameter settings, the vehicle is identified. Then the wavelet transform is applied to the image, which produces multi-resolution of the signal to acquire the relationship between frequency and location. Based on the high frequency information, a vertical and horizontal projection is measured (a similar method employed in Model 1, but by using edges instead of the wavelet frequency information). The assumption is that the location of the license plate is at the bottom, which is then examined to match the projection results with the known license plate − if the match is similar, the location is marked as a license plate.

Model 5 produces an overall accuracy rate of 88.71% although the limitation of the camera location is not ideal in real world conditions. The measured requirements set for this research show poor performance against all SRs [1, 2, 3, 4, 5, and 6]. The plate size is limited to the parameter values set to locate the rear red lights. The plate rotation is completely ignored and greyscale is not an option in this model. The image noise is not considered, nor is adaptation to various conditions applicable, and the accuracy of performance is below 90% par.

### 2.4.6 Model 6: AdaBoost – Machine Vision

Much research into AdaBoost algorithms demonstrates the effective license plate detection (Sun, Cui, Gu, Cai & Liu, 2009; Cui, Gu, Cai & Sun, 2009; Ho, Lim & Tay, 2009). The AdaBoost is short for adaptive boosting, and is a machine learning algorithms. The universal algorithm finds a number of weak classifiers through the iterative algorithm, and integrates these weak classifiers into strong classifiers according to their respective votes in a weighted manner (Cui et al., 2009). The AdaBoost is very suitable for character

recognition, but it can be trained to identify the license plate. The algorithm is very sensitive to noise and may result in miss detection.

Surveillance driven by AdaBoost has so far been implemented in static systems, as it is very reliant in regard to the license plate resemblance used for achieving peak performance. AdaBoost is capable of detecting various plate sizes, and hence meets the requirements of SR [1], but in relation to SR [2] it depends on the training image sets. If the license plate is upside down, AdaBoost will struggle to recognise the license plate. Greyscale images are compactable, but SRs [4, 5] are doubtful. The image noise is random although the common image noise can be trained, which is a difficult task. Adapting to different conditions is also dependant on the training set. However, the accuracy rate of a detectable license plate provided in the research is 98%.

## 2.5 Discussion

Over the past three decades, the research into vehicle surveillance has produced various surveillance models. However, there is no evidence of comprehensive research into the night and limited light conditions of licence plate recognition. Therefore, this research will be first of its kind to expand this field of licence plate recognition in a mobile environment.

Various models of license plate recognition have been developed to produce high performance accuracy results. The concern over the test sample images used by researchers is increasing. There is no industry standard for test methods, and researchers have taken advantage of this situation to improve their results by biasing the test samples within simple and easier real world settings. It is difficult to compare research results where no evidence of proper testing has been conducted. Thus, this research will explore real world samples in different environments with no biased settings in order to measure the performance of a surveillance model developed by the researcher.

The hardware components are a camera and external source light. As the industry in question uses the CCD camera as the first choice camera, and no other camera types have

been utilised in this field, this research will explore different camera types. The external illumination source is driven by infrared or flash light devices because the flash light allows easier identification of the licence plate. However, it is potentially dangerous for the public who can get distracted easily and cause motor vehicle accidents. The infrared light, as the external light source, is thus the alternative choice.

This literature review shows that although licence plate recognition is dependant on the software algorithm model, limited research has been carried out into license plate detection under limited light conditions. As the day light surveillance models are reviewed and six unique models investigated, it can be noticed that most studies employed the models whose availability is less restricted in operation under various conditions. However, this is not case with some of the popular models in the industry. The horizontal location of the license plate is assumed by Model 1, 3, 4, and 5, whereas Model 3 is driven by fuzzy rules and can adapt to different orientations if the rules are changed. The image noise is not considered in many models, except for Model 2 which acknowledges the image noise and uses parallel lines as the identifier of the license plate, but other objects in the image will have similar parallel lines and increase incorrect license plate detection. Model 6 can perform strong plate recognition, when the AdaBoost machine vision algorithm is trained for many license plate images. However, it is restricted to similar license plate images, very sensitive to noise, and limited to certain plate orientations. The commonly employed techniques are edge detection with vertical and horizontal projections of the edge information, or in some models, the frequency information. If the projection matches the license plate pattern, it is diagnosed as the license plate. Furthermore, this model is very sensitive to image noise.

To develop a suitable model for this research, Model 2 is considered because it can operate on a greyscale image and is not limited to plate orientation. The camera position can cause the license plate to be angled in any direction, and Model 2 is designed to operate in various conditions. However, parallel lines will not be considered as part of the design, and further image pre-processing is required to eliminate noise. The edge detection and border line contours are the key aspects that will be taken into consideration in this research.

# Chapter 3

# Mobile License Plate Surveillance Platform

## 3.0 Objectives of hardware architecture

The goal of object detection in machine vision is to make the important features of the object visible and suppress the undesired features of the objects (Guo & Liu, 2008). For that purpose, this chapter describes the hardware used to implement mobile license plate detection unit operated in limited light conditions. The following will be described: illumination under a limited light condition exposed by the emitted infrared light (Section 3.1); the camera which converts an analogue signal to a digital signal and transmits the image data via USB 2.0 to the computer (Section 3.2); the singular frequency camera lens filter for suppressing the image noise (Section 3.3); the camera positioned correctly to optimise license plate image visibility (Section 3.4); and a showcase output image from the hardware component (Section 3.6).

## 3.1 Emitted Infrared Light Source

Light is electromagnetic radiation of a certain range of wavelengths visible for humans (380-780nm). The light with shorter wavelengths is called ultraviolet light, and the light with longer wave lengths than the visible range is called infrared light. This research uses 850nm wavelength which is invisible for the human eye, and falls under the Infrared A category DIN5031 standards.

The light emitted diode (LED) is a light emitting semiconductor used as an infrared light source. The LED schematic consists of a chip of semi-conducting materials doped with impurities to create a p-n junction (the semiconductor material is gallium arsenide and aluminium gallium arsenide). The electric current flows from p-side to n-side but not in the reverse direction. The emission of light happens when an electron meets the hole which

falls into a lower energy level and releases energy in the form of a photon, as shown in Figure 3.1. The band gap required to produce infrared wavelength is 1.42eV.

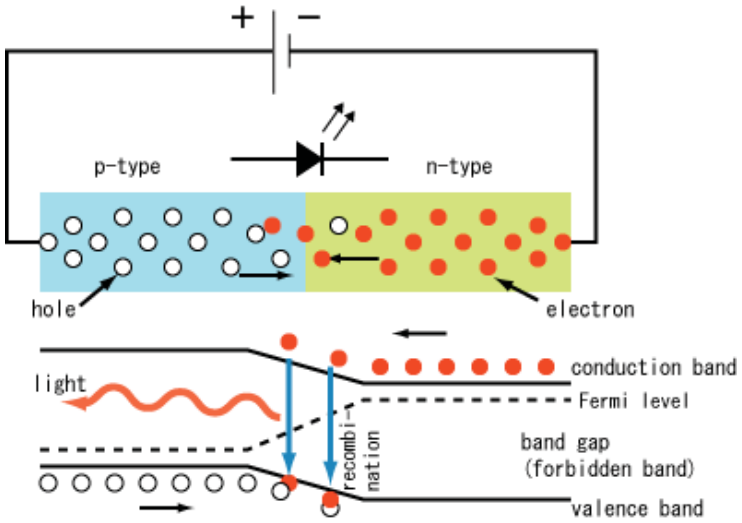



**Figure 3. 1: Pn junction – LED diode (Wikipedia, 2009)**

A set of 48 LED diodes is used as the infrared light source, and is activated by a photon resistor, as shown in Figure 3.2. It has the illumination distance up to 20 meters according to the manufacturer, and is powered by DC 12v 500mA via the motor vehicle battery. The surface area of the infrared light is 7cm in diameter, and 200mA is needed to power 48 LED Diodes. The photon resister in the middle behaves as a switch-on/off device. It is gradually turned on when the surrounding illumination drops from 10, 000 lux (the day light but not direct sunlight) to 5 lux (the clear night-sky darkness). In this research, the testing will be done in limited light conditions only, which means that the infrared light is required at all times. To enable the infrared light, the photon resistor is covered by glue tack, a material commonly used in classrooms to stick paper on the wall.

**Figure 3. 2: 48 LED infrared light source**

## 3.2 CMOS Camera

The purpose of the camera is to create an image from the light focused on the image plane by the lens (Guo & Liu, 2008). A digital sensor located in the camera converts the analogue image into the digital image format. There are two types of digital sensors, CCD (a charge-coupled device) and CMOS (a complementary metal-oxide- semiconductor). The primary difference between the sensors is the readout method used to convert information from analogue to digital. The Logitech Webcam Pro 9000 is selected for this research as it uses CMOS digital sensors.

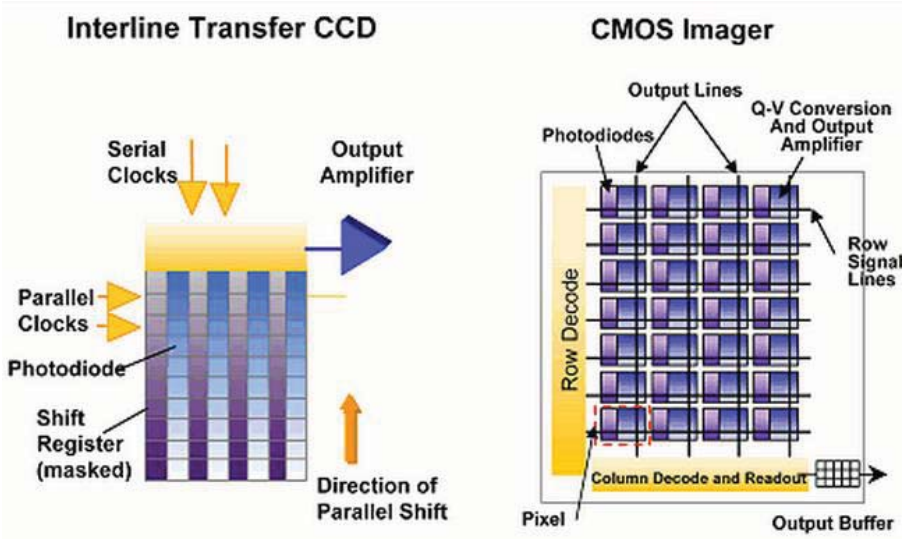



**Figure 3. 3: CCD and CMOS architecture (Titus, 2001)**

The internal operation of CCD and CMOS digital sensors will not be discussed in detail, and the distinctiveness of each sensor is explained. The CCD sensor puts the digital information in a serial array, whereas the CMOS sensor has the ability to put each pixel value separately (random access). Figure 3.3 draws the difference in architecture between the two digital sensors. The advantages of CCD sensors are that they create high quality low-noise images and have been mass-produced for a longer period of time. Therefore, a more mature technology can produce high quality pictures with more pixels. However, the disadvantages of CCD sensors are that they consume 100 times more power than equivalent CMOS sensors, and generate a slow process, compared with CMOS sensors, due to the serial readout.

The camera Logitech webcam Pro 9000 supports a high definition video (up to 1600 x 1200), i.e. up to 8 megapixel photos with 30-frames per second video. The connection to the computer is supported by hi-speed USB 2.0 certified connections. The Logitech webcam 9000 shutter speed is automatic, and its speed depends on the image resolution. For 640 x 480 resolution, the shutter speed is recorded as 1/33s, and for 1600 x 1200 resolution, it is as low as 1/13s. Thus, 640 x 480 resolution is more than sufficient to conduct the experiment. Other benefits added are an auto focus and steady shots.

**Figure 3. 4: Logitech Webcam Pro 9000**

## 3.3 Infrared Filter

Figure 3.5 shows an 850nm infrared filter which filters the white light frequency and allows the 850nm wavelength. The Logitech Webcam 9000 is capable of detecting the 850nm frequency range. The 55mm wide diameter is the exact size of the outer lens of the Logitech Pro 9000 Camera, and therefore no leakage of the outer light frequency can pass through. The function of the infrared filter is to remove irrelevant image objects, which is not part of the area of interest here (the license plate). Restricting the camera to detection of the 850nm frequency will allow capturing a controlled image and lowering the image noise. However, the natural light and other artificial light sources can also produce the 850nm wavelength. Some examples of artificial infrared lights are street lamps, motor vehicle headlights, motor vehicle break lights, and natural heat produced by motor vehicle engines, which is also inside the infrared frequency range. Under the natural circumstances, the listed infrared frequency will pass through the filter and will be seen as the image noise.


**Figure 3. 5: 850nm infrared filter**

## 3.4 Camera Position

The camera and infrared source positions are shown in Figure 3.6, and the position is located at the centre of the horizontal plane of the front bumper.


**Figure 3. 6: Camera and infrared light source position**

The camera positioned as shown in Figure 3.6 and 3.7 points outwards and parallel to the ground. The bottom frame of the camera lens captures at least 80cm of distance. At this distance, there is low probability of motor vehicle license to be located at the bottom of the image frame. To maximise the visibility of the camera lens, a minimum distance of 200cm is determined to be the bottom image frame, and thus the camera and the infrared light source need to be adjusted upwards. To calculate the angle, Pythagoras' theorem is used to calculate the angle (Ω), and the right angle is estimated from the position parallel to the ground. The camera is carefully placed and measured with a level gauge for the 90 angle degree accuracy. The distance of 46cm is measured from a flat ground surface to the bottom of the camera lens. Also, the distance of 80cm is measured from the camera to the first visible object seen at the bottom of the image frame. These two measurements allow θ to be found.

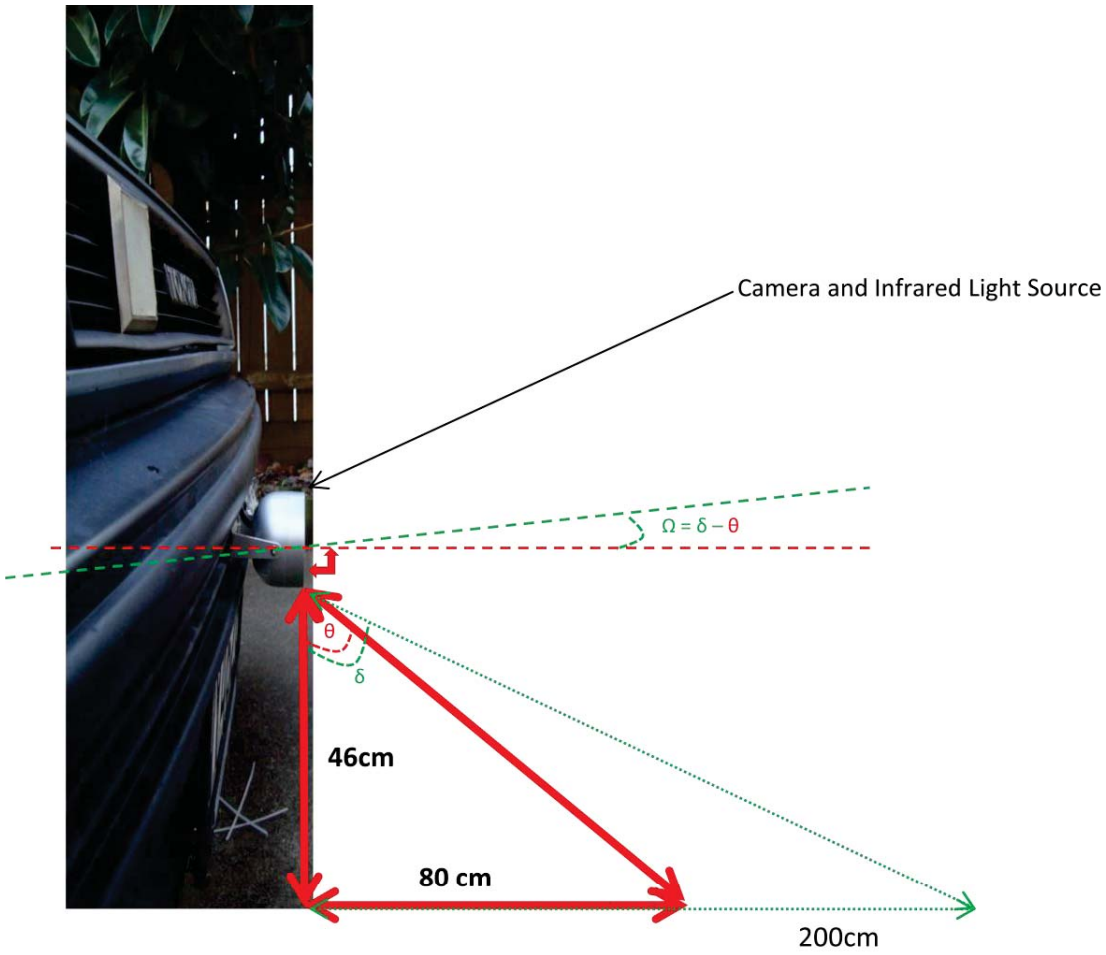



**Figure 3. 7: Camera angle drawing**

To calculate ∅:

$$\tan\theta = \frac{Oppsite\ (cm)}{Adjacent\ (cm)}$$

$$\emptyset = \tan^{-1} 80/46$$

$$\emptyset = \ 60.10\ (2dp)$$

To calculate $\delta$:

$$\tan\delta = \frac{Oppsite\ (cm)}{Adjacent\ (cm)}$$

$$\delta = \tan^{-1} 200/46$$

$$\delta = \ 77.05 \ (2dp)$$

To calculate $\Omega$:

$$\Omega = \ \delta - \ \emptyset$$

$$\Omega = \ 77.05 - \ 60.10$$

$$\Omega = \ 16.05^{o}$$

As shown in Figure 3.7, to move the camera lens visibility distance to 200cm, the camera angle needs to be adjusted by $16.05^{o}$ to a new angle, by following the diagonal green dotted line (not scaled). Both the camera and the infrared light sources are adjusted to this new angle.

## 3.5 Connection (Camera, Infrared Light Source and HP Notebook)

The camera and the infrared source light are attached to the front bumper of the motor vehicle. Cable ties are used to secure that the camera and the infrared light source are in the correct position as calculated. The infrared filter is placed over the camera lens, and then tightened with a kitchen glad wrap and tied by an elastic rubber band. The generic kitchen household material does not degrade the frequency signal. The cables from the camera and the infrared light source are tied together and fed into the passenger window, as shown in Figure 3.9.

**Figure 3. 8: Camera and infrared light source (USB and power cables)**

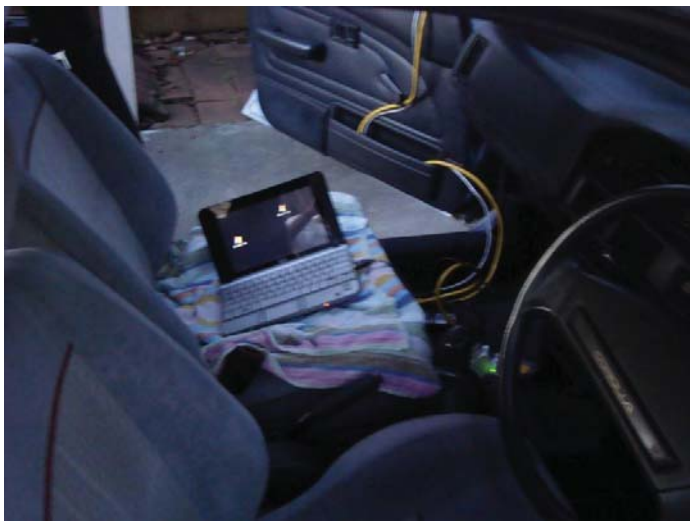


**Figure 3. 9: Cable connected to HP mini notebook**

Figure 3.9 also shows an HP 2133 mini notebook connected to the camera, and ready to capture and save the image. This notebook has an Intel mobile processor of 1.6 GHz, and operating windows vista. The processor's power is more than sufficient to capture the image and process the image, by using image processing algorithms, which will be explained in Chapter 4 and Chapter 5.

## 3.6 Output Image

The final image is greyscale as expected due to the presence of only infrared frequency which is not composed of any other colour frequency. Other objects, along with the motor vehicle license plate, can also be expected to be captured as explained previously in Section 3.3. Objects in the image other than the area of interest (the license plate) are classified as the image noise. The image noise objects are minimised via the infrared filter on the camera lens. The upcoming software image process will aim to remove the image noise, which will be discussed in Chapter 4. However, the selected sample is unique, i.e. each image will have a variable image noise and the license plate positioned differently. Figure 3.10 shows an example.

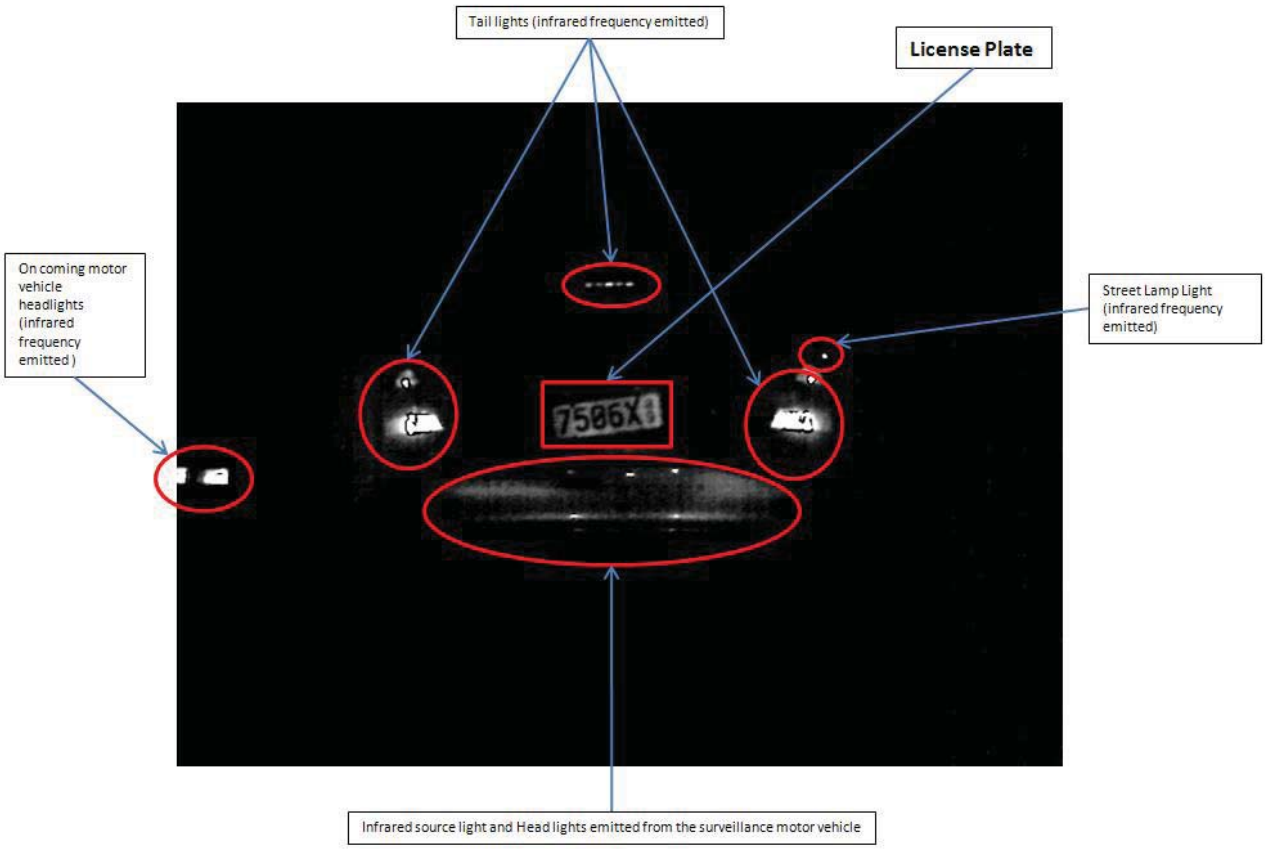



**Figure 3. 10: Camera view of single sample image**

# Chapter 4

# Software Theory on License Plate Detection

## 4.1 Objectives of software algorithms

To locate the licence plate, the image is processed for noise reduction by using the Gaussian and image pyramid algorithms so that edge detection techniques can extract the essential edges by using the edge and binary threshold detection algorithms. The established edges are connected by using contour chain code algorithm, and then the positions which can hold a license plate boundary are recorded. Various dimensions are matched with the standard license plate dimensions. If successful, a match is considered, and the license plate has been detected. The parameter optimisation and the design selection will be discussed in Chapter 5 (Methodology).

## 4.2 Image Noise Minimisation

Every image exhibits image noise from the surrounding. Image noise minimisation identifies one popular image noise known in the image process industry, as well as plans to remove the noise via the industrial proven techniques modified to suit the application build for this research theory.

### 4.2.1 Load Image to Memory (Image Information)

At first, the image is to be loaded into the computer memory. Commencing from the hardware platform, image collections are saved into a folder directory using the jpeg format. Jpeg is an industrial image format for most digital cameras, and this research follows the traditional format. The image captured is then saved by using the RGB colour

scheme format. Although the image will be greyscale almost every time, and thus only single channel image frame is required, but in consideration for future aspirations the author allows the image to be saved in the three channel capacity using RGB component. By means of the openCV function cvLoadImage(), an input parameter, which is part of the location image directory, is passed. Throughout this chapter, a single image is loaded onto the memory as the first step.

### 4.2.2 Gaussian Blur

The image received from the hardware platform is intended to have optimum near image clarity. In the 'real world' application, however, a prefect image is not achievable. Thus, the image captured by the camera contains defects which occurred at acquisition during light transformation in digital sensors, optics failure and inadequate or non-uniform illuminations, and undesirable viewpoints. The listed characteristics are not apparent in every image, but the likelihood of image corruption by noise is high.

To eliminate the image noise, further research is required. While the existing literature considers many types of noise, it is not possible to cover all image noises in this research, and thus the focus is on one main type of noise.

Gaussian noise, also known as white noise, is observed in other aspects of engineering. One main characteristic of Gaussian is additive load, where small variant information (noise) is added to the original pixel value across the whole image frame. With this knowledge, an assumption is made about the noise levels, i.e. in the image, the noise stack is much smaller than any of its important details about the edges that represent the license plate boundary. Based on this assumption, image smoothing, also known as neighbourhood averaging, can be applied.

To reduce Gaussian noise, a probability distribution model is derived to the construct the noise magnitude residue. In isolation, Gaussian functions as a low pass filter, and thus a higher frequency component is eliminated. Gaussian noise probability can be calculated by

Equation 4.1 where $\sigma$ (standard deviation) is the smooth parameter, and the larger value of $\sigma$-, the greater extent of smoothing.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2}$$

**Equation 4.1: One-dimension Gaussian function**

The standard deviation calculation consists of the following steps:

1) Listing each pixel values (intensity), the total pixel value counts 640 x 480 = 307, 200 pixels;
2) Finding the mean, summing up the all the pixel values, and dividing the sum by the number of pixels;
3) Calculating deviation for each pixel value, and subtracting the mean value from it;
4) The calculated deviation value is then squared;
5) The squared deviation is summed up, and divided by total pixel minus one; and
6) The result from Step 5) is the standard deviation.

The value of the standard deviation is dependent on the image intensity values. The more detailed the image, in which the intensity values are spread across the range of values from 0 to 255, the stronger the return value of the Gaussian function. Similarly, the less spread the intensity values, the lower the value returned by the Gaussian function.

The equation $G(x)$ is graphed and shown in Figure 4.1. The universal shape follows the normal distribution bell curve whose characteristics are also recognised in other industries such as business, engineering and population study. Conversely, this research applies the probability distribution curve to envisage noise distribution.

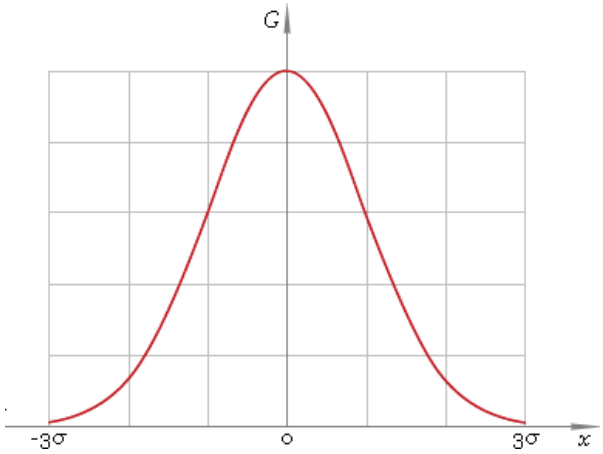



**Figure 4. 1: One-dimensional Gaussian distribution truncated at -3σ and 3σ**

The equation presented in Equation 4.1 allows the construction of the kernel window to be completed. The kernel is a grid of values calculated with the values from the Gaussian function, and can vary in dimensions from 3 x 3, 5 x 5, 7 x 7 and so on. The kernel dimensions always have an odd number so that a central point in the grid can be located equally across the whole grid.

The kernel grid is then populated with the values from the Gaussian function, and in this research, the 7 x 7 grid kernel is used. The coefficient obtained from the Gaussian formula represents the weighted value for each pixel in the dimension set. Figure 4.1 reveals that the central pixel value has a greater coefficient than the pixels value further spread from the centre. The weighted values are calculated by the ratio of the Gaussian curve, and then this value is transferred into the kernel position.

The Gaussian kernel has an anchor located at the centre of the kernel. The value of convolution at a particular point is computed by first placing the kernel anchor on top of a pixel in the image with the rest of the kernel, and then by overlaying the corresponding local pixels in the image. For each kernel point, the value of the kernel at that point and a value for the image at the corresponding image point are computed. By multiplying these together and by summing up the result, this result is then placed in the resulting image at the location corresponding to the location of the anchor in the input image. This process is

repeated for every point in the image by scanning the kernel over the entire image. The result of each pixel convolution is saved in a separate image frame, and the next convolution pixel will use the original pixel values for the convolution summation.

The computation of the Gaussian mathematical process will not be discussed in detail as the mathematical proof is sufficient, and knowledge of convolution is expected.

One-dimensional Gaussian function processes one pixel at a time, first horizontally and then vertically. This method is time consuming, and uses up the CPU resources. An improved process from one dimensional convolution is a two-dimensional Gaussian function, and the mathematical equation for this function is shown in Equation 4.2.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2+y^2 / 2\sigma}$$

**Equation 4.2 Two-dimensional Gaussian function**

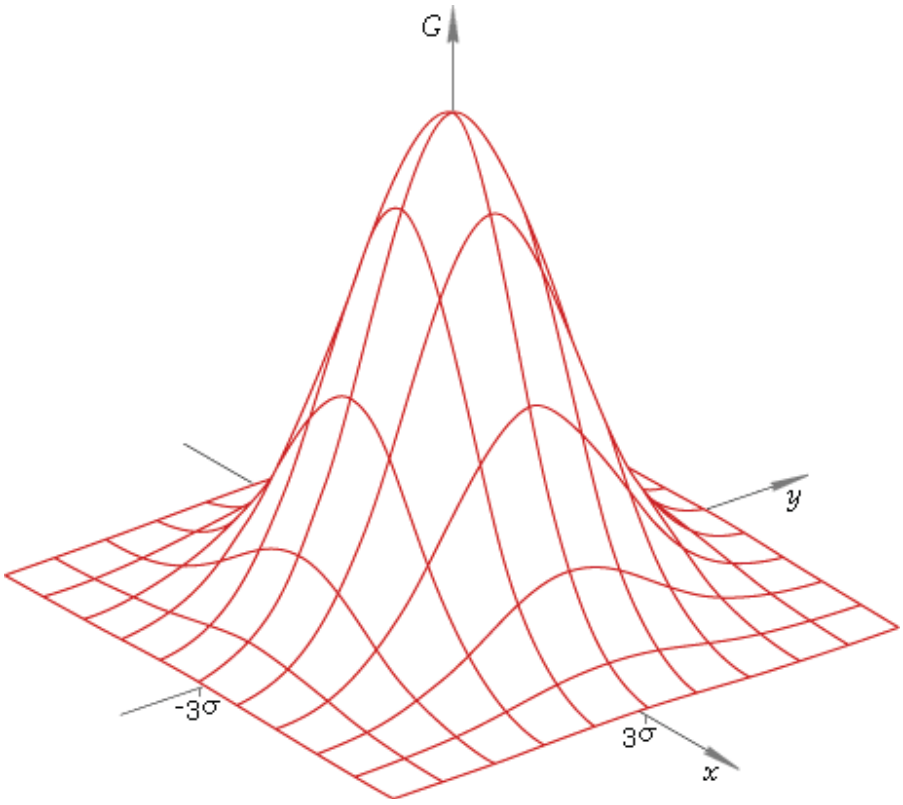



**Figure 4. 2: Two-dimensional Gaussian distribution truncated at -3σ and 3σ**

The advantage of the two-dimensional Gaussian blur reduces numerical calculations by half. Because of the reduction in separate horizontal and vertical calculations, the convolution process is faster, and obtains the same results as that of the one dimension. The kernel weighted values are now different as per calculation result of Equation 4.2.

The usefulness of the Gaussian function is subjective with regard to illuminations of edges of the image and other regions of the image. In the image, the license plate region is expected to be represented with strong edge bonds around its boundary. Other areas of the image with weak edges and non-edge image regions will be more affected by Gaussian, and the output is a blurry image.

The Gaussian function averages out the pixel value from its neighbouring pixels, which will affect small objects most. The edges of the image are also affected, but not to the extent as great as small isolated objects are.

### 4.2.3 Pyramid Decomposition

The direct approach to remove additional noise from the image is by reducing image resolution. Image resolution reduction is possible with the technique called pyramid decomposition. There are two types of pyramid decomposition. In this study, matrix pyramid (M-Pyramid) is explored. A matrix pyramid is a sequence of images, where the first sequence is the original image, and the next sequence is half the resolution of the original image and the image after is half the second image (Weeks & Myler, 1993). The image resolution is halved at each sequence. However, for the purpose of this study, the resolution is stopped at the second sequence, and the output resolution is 320 x 240.

In context of the image, the area of interest here is the license plate boundary. As the pixilation of the boundary is important, an assumption is made after pyramid decomposition of the image that the license plate boundary should stay intact and with enough pixels to be identified as the license plate.

The resolution reduction is applied evenly across the whole image frame. Even columns and rows removed from the original image frame remained behind the pixel and are then contracted to form a continuous pixel formation with a resolution of 320 x 240.

The function cvPyrDown in OpenCV constructs a Gaussian blur kernel, applies convolution to create a new image, and then applies pyramid decomposition. The original image shown in Figure 4.3 and Figure 4.4 illustrates the result of the Gaussian blur and pyramid decomposition.

**Figure 4. 3: Original image from hardware platform**

**Figure 4. 4: After Gaussian blur and pyramid decomposition**

To clearly see the evidence of the Gaussian blur and pyramid decomposition, the original image (Figure 4.5) and the image after application of Gaussian and pyramid decomposition (Figure 4.6) are zoomed by 600 percentage magnification with Microsoft Paint. Figure 4.6 has clearly distorted in pixilation and decreased the illumination values. The effect seen in Figure 4.6 is equally spread across the whole image, and intended to remove even pixel values. The content in the license plate is irrelevant at this stage as long as the license plate boundary is intact and has a rectangular shape.

**Figure 4. 5: Original image zoomed 600 %**

**Figure 4. 6: Gaussian blur and pyramid decomposition zoomed 600%**

### 4.2.4 Up-sampling and Gaussian Blur

The image received in Section 4.2.3 is set for a further image transformation. The retrieved image is half the resolution of the original image, and the illumination value for each pixel is convolved with the Gaussian kernel operator. The next step is to restore the image back to the original resolution. In the process of pyramid decomposition, the pixel values were removed from the image frame, and the deleted pixel values cannot be revived. To reconstruct the image resolution to the original size, the absent even row and column are filled in with values 0. The image resolution is thus restored to the original size. The pixel value of the image is further diluted from the original image, but the area of interest still

remains the ‘license plate’. However, the smaller details of the image are now being minimised as expected in this process.

After the injection of even rows and columns, the image is then again convolved with the Gaussian blur with the kernel size of 7 x 7. As explained in Section 4.2.3, the image neighbourhood values are averaged out by the coefficient of the Gaussian blur kernel. The image is even blurrier than before. This practice blurs the details from the image to flatten the pixel value as seen in Figure 4.7.

**Figure 4. 7: UP sample and Gaussian blur (7 x7) kernel**

The up-sampling and Gaussian blur processes are completed by the OpenCv function - cvPyrUP(), where the parameter input is an image and a Gaussian kernel (7 x7). The output of the function results in an image with resolution restored to the original size, which therefore doubles the input image resolution and then convolve with the Gaussian kernel.

## 4.3 Edge Detection Techniques

To this point, the image noise has diluted, and caused the image to be blurry. The image block that is of interest to this research is the license plate. Edge detection techniques explore the Canny edge detection and binary threshold algorithms to highlight strong edge points in the image in order to locate the license plate.

### 4.3.1 Edges Embedded in Noise

The rule proposed for an edge calculates the rate of change between one pixel values to the next neighbouring pixel value. The rate of change is the key to detection of the presence of an edge boundary. This concept is simple to interpret, but in the real world application, the pixels do not change at the rate of change required to be easily identified.

An example of pixel value variation is shown in Figure 4.8, and the top part of the figure shows three separate blocks of grey scale sections. If the first section is considered, from black to white, the transection periods of pixel values are close together (the transection rate is shown in second row of Figure 4.8). The graph measures the pixel value, the black parts representing the intensity value zero and the white parts the value 255. In the centre of the first section, the pixel value (from black to white) shows a vertical bar in the figure, which means that the rate of change is immediate in this case.

In the previous section, the image processes was set to reduce noise by means of the Gaussian blur technique, pyramid decomposition, up-sampling and then another Gaussian blur. The output comprises the image and the results in blurriness, and thus, the edges of the image object will spread across a wider magnitude of pixels. In Figure 4.8, the second section is relevant to this study because the pixel value changes from one extreme to another in a slow gradual process, and the graph indicates this ramp up to the final change. The gradient of the ramp reveals the information about the image edge, i.e. the steeper gradient shows a strong edge, and the smaller gradient shows a weaker edge. The third

section is another example of pixel intensity, and the graph demonstrates the rate of change. In this research, the edges are not expected to change so quickly.

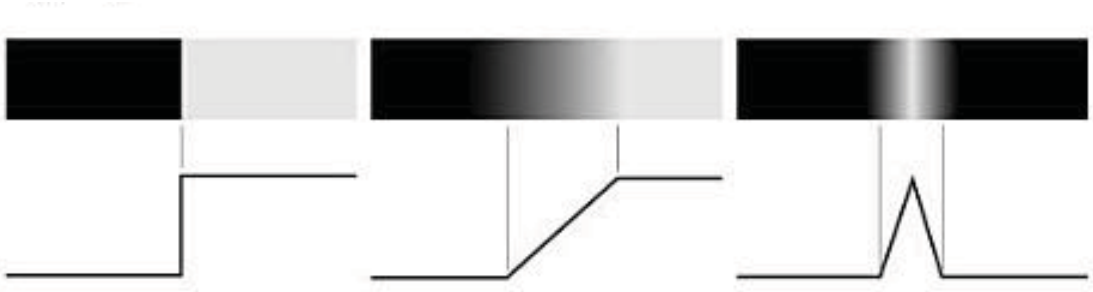


**Figure 4. 8: Model of digital edge (a, b, c from left to right) (Gonzalez & Woods, 2002)**

The rate of change in pixel intensity from black to white or vice versa is graphed to reveal the gradient. To find out the magnitude of the edge, a graph revealing the intensity of change is differentiated by using the function of calculus which results in a graph showing the rate of change. The magnitude value on the y-axis is evaluated as the strength of the edge.

The image noise, as an unwanted characteristic is present in all aspects of engineering. In the course of image processing, the image noise is a significant factor that needs inspection and acceptance across every image. To date, mathematically speaking, a number of equations have been designed to predict noise, and studied to eliminate the noise. However, many types of noise have been discovered so far. This study will not consider all types of noise, as it is time-consuming and unfeasible. One noise type discussed in the previous section is re-discussed in this section. Gaussian noise is present after the previous attempt to minimise it, and the influence of such noise is shown in Figure 4.9. The destructive effects of this additive image noise are unfavourable because they diminish the ability to detect weaker edge objects.

Figure 4.9, shows two columns. First column show pixel intensity values from left to right. The second column measures the rate of change of pixel intensity. From the top row,

Gaussian noise is introduced into the pixel sections blocks, and the noise intensity is increased in each row to a higher level. The ramp on the first column can be seen going up, but on a closer look, the ramp is variously uneven, which is clearly seen on the rate of change graph which is the second order of differentiation from the first column. The magnitude of noise is amplified in the second order, which demonstrates the additive Gaussian noise can be very destructive in calculation of weaker edges, which is case in the last row.

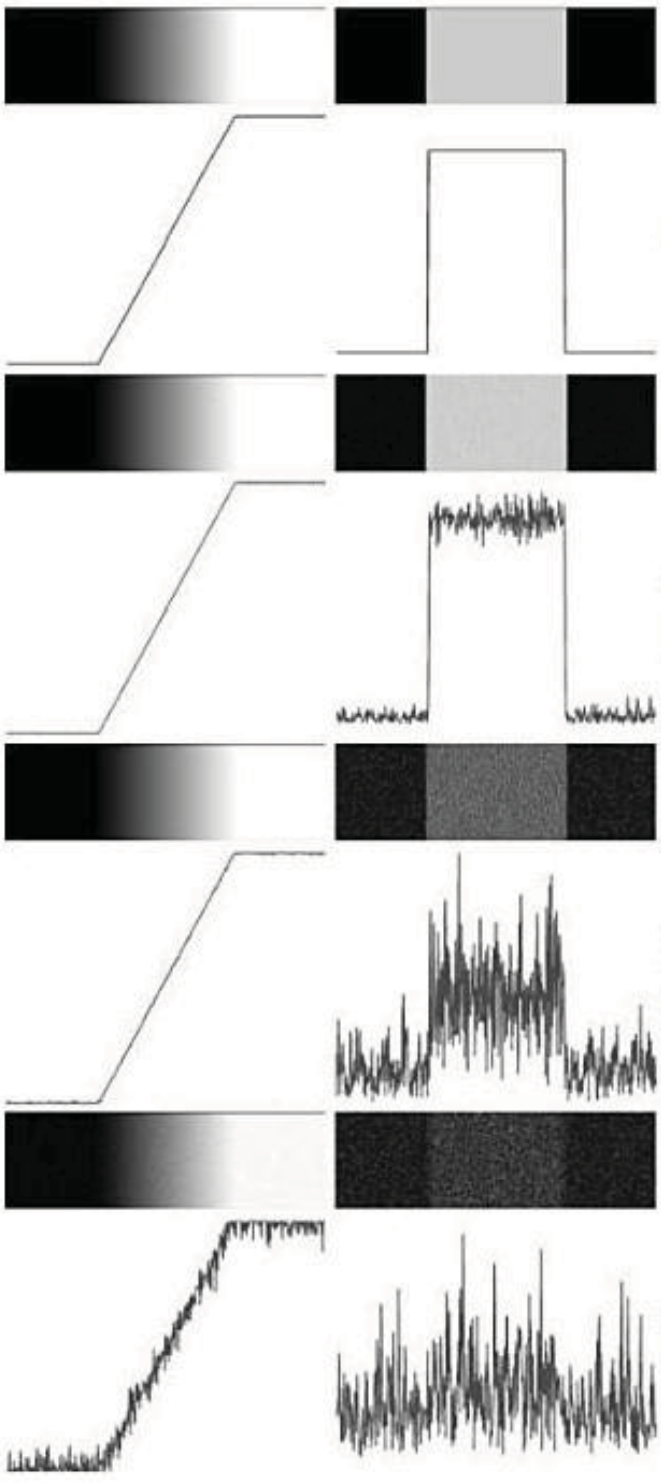


**Figure 4. 9: Image pixel intensity and first order differential with additive Gaussian noise (Gonzalez & Woods, 2002)**

### 4.3.2 Canny Edge Detection

The Canny edge detection algorithm was developed by John F. Canny in 1986, and devised according to the rules given below.

There are three criteria that an edge detector should fulfil.

1) It should have a good detection quality.

   There should be a low probability of false detection of an edge point, and a low probability of erroneous missing of an edge point.

2) The edge detector should have a good localisation quality.

   The extracted edges should be as close as possible to the true edges. This is formalised by minimising the variance of the extracted edge positions.

3) The edge detector should return only a single image edge for each object edge.

   It should avoid multiple responses.

The first criterion was applied in Section 4.2 by using Gaussian and image decomposition to smooth noise.

| | | |
|---|---|---|
| $(i-1, j-1)$ | $(i-1, j)$ | $(i-1, j+1)$ |
| $(i, j-1)$ | $(i, j)$ | $(i, j+1)$ |
| $(i+1, j-1)$ | $(i+1, j)$ | $(i+1, j+1)$ |

**Figure 4. 10: Neighbourhood of the pixel at point**

The second criterion involves computation of pixel magnitude gradients and of the direction, as well as application of non-maximal suppression.

To compute the values for gradients and the direction of gradient, the derivative of calculus measure function is utilised. Partial derivatives enable derivation of the following equations from the image neighbourhood in Figure 4.10.

$$\frac{dS}{dx}(i,j) = \frac{1}{2}[S(i+1,j) - S(i,j) + S(i+1,j+1) - S(i,j+1)]$$

**Equation 4.3 Derivative of function with respect to $i$**

$$\frac{dS}{dy}(i,j) = \frac{1}{2}[S(i,j+1) - S(i,j) + S(i+1,j+1) - S(i+1,j)]$$

**Equation 4.4 Derivative of function with respect to $j$**

To find the pixel gradient, Pythagoras theorem is used, and the gradient of the pixel is calculated by means of derivative Equation 4.3 and Equation 4.4.

$$G(i,j) = \sqrt{(\frac{dS}{dx}(i,j))^2 + (\frac{dS}{dy}(i,j))^2}$$

**Equation 4.5 Gradient magnitude $G(i,j)$ of the pixel**

The pixel gradient direction can be obtained from Equation 4.3 and Equation 4.4, and then the angle $\emptyset(\mathrm{i},\mathrm{j})$ can be calculated.

\

$$\emptyset(\mathrm{i},\mathrm{j}) = \arctan(\frac{\frac{dS}{dy}(i,j)}{\frac{dS}{dx}(i,j)})$$

**Equation 4.6 Gradient direction $\emptyset\ (\mathbf{i},\mathbf{j})$**

The final part of the second criterion is non-maximal suppression. The edge pixel gradient magnitude locates the local maximal in the gradient direction. To determine whether the gradient magnitude of the pixel at $(i, j)$ is local maximal or not, it is necessary to locate the two neighbouring pixels p1 and p2 of the pixel at point $(i, j)$, and to calculate the gradient magnitudes of the three pixels. It is supposed that pixels p1 and p2 are located at positions $(i1, j1)$ and $(i2, j2)$, respectively. If the gradient magnitude of the pixel at positions $(i, j)$ is maximum, it is an edge point, and the gradient magnitude is used as its intensity. Otherwise, the pixel is not an edge point, and its intensity is set to 0. The resulting image δ can be described as follows:

$$\delta(i,j) = \begin{cases} G(i,j), & If\ G(i,j) \geq G(i1,j1)\ \ and\ \ G(i,j) \geq G(i2,j2) \\ 0, & otherwise \end{cases}$$

**Equation 4.7 Local maximal**

Because the locations of pixels are discrete, gradient directions also need to be quantised. The 8-neighbouring domain, as shown in Figure 4.10, with the pixel at position (i,j) can be taken as an example.

Figure 4.11 presents the use of the following angle criteria:

a. If $-\frac{1}{8}\pi < \emptyset(i,j) \leq \frac{1}{8}\pi, \theta(i,j) qauntised\ as\ 0$
b. If $\frac{1}{8}\pi < \emptyset(i,j) \leq \frac{31}{8}\pi, \theta(i,j) qauntised\ as \frac{1}{4}\pi$
c. If $-\frac{3}{8}\pi < \emptyset(i,j) \leq -\frac{1}{8}\pi, \theta(i,j) qauntised\ as -\frac{1}{4}\pi$
d. $\frac{3}{8}\pi < \emptyset(i,j) \leq -\frac{1}{2}\pi\ or\ -\frac{1}{2}\pi < \theta(i,j) < -\frac{3}{8}\pi, \theta(i,j) qauntised\ as\ \frac{\pi}{2}$

The angle regions are represented by colour space where the four angles are shown as yellow (a), green (b), red (c) and blue (d).

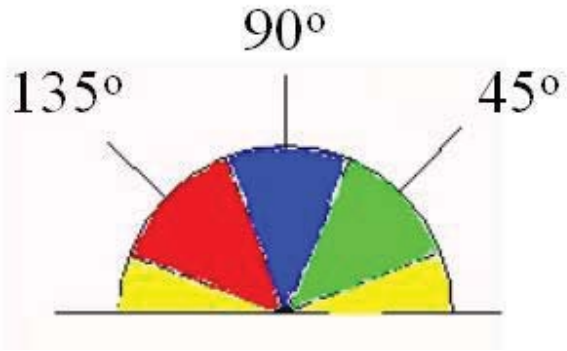



**Figure 4. 11: Edge directions (Green, 2002)**

Fulfilment of the third criterion is the threshold with hysteresis. Non-maximal suppression reduces the border of an object to the width of just one pixel. Due to the existence of noise and thin texture, this process may result in spurious responses, which leads to streaking problems. Streaking here means the breaking up of the edge contour caused by the operator fluctuating above and below the threshold. Figure 4.12 demonstrates the non-maximum suppression and hysteresis. An example of the edge gradient magnitude is drawn, along with two thresholds, one upper and one lower. The process of non-maximum suppression is selecting the points along the top of the ridge. Given that the top of the ridge initially exceeds the upper threshold, the threshold output is set to white until the peak of the ridge falls beneath the lower threshold. The threshold output is then set to black until the peak of the ridge exceeds the upper switching threshold.

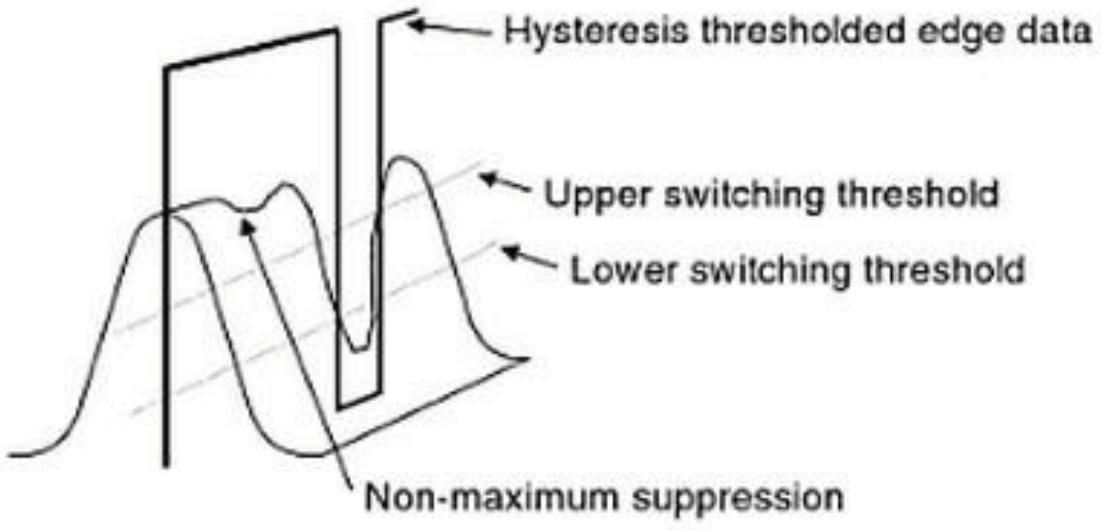



**Figure 4. 12: Hysteresis and non-maximum suppression (Cui et al., 2009)**

The Canny algorithm is applied to the original image from Chapter 3, as shown in Figure 4.13 and Figure 4.14. Subjective to the criteria of the Canny algorithm, the image is transformed into a single edge image. The edges are bounded by the non-maximum suppression regulation, and therefore the image is transformed from a greyscale to a black and white edge image. Figure 4.14 shows the close-up of the license plate, and the shape of the plate can be seen. The threshold hysteresis optimisation will be discussed in Chapter 5 (Methodology).

**Figure 4. 13: After Canny algorithm process, with lower threshold = 10, and upper threshold = 50.**

**Figure 4. 14: Zoom 600% of Figure 4.13, showing license plate edges after Canny process**

### 4.3.3 Mathematical Morphology – Dilation

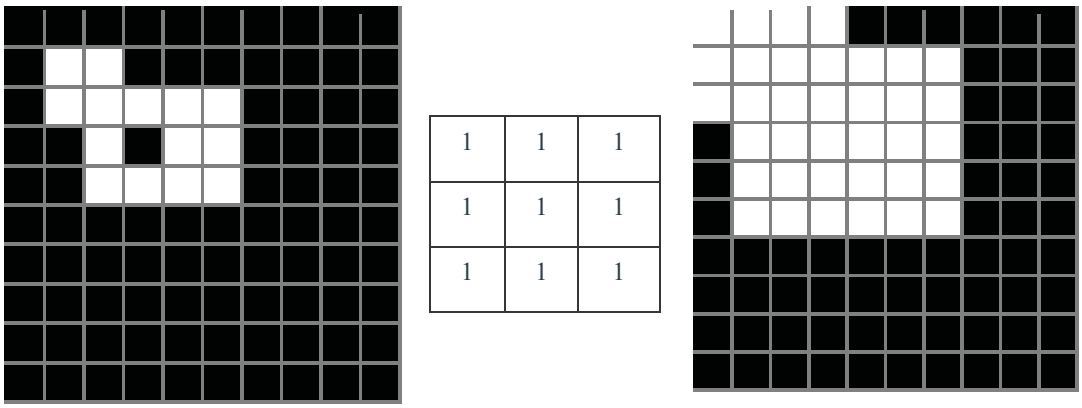
The dilation operation, a technique of mathematical morphology, is designed to find holes in the image object, and to fill them by using the structure element coordinates design. The need for dilation is obvious in the Canny process, i.e. the output image consists of a single edge boundary of the object shape. However, under the influence of image noise and other factors, the image boundaries can be broken, and therefore, gaps between the boundary outlines may occur. To amend this problem, dilation with a structure element is used to fill in object boundary holes, and produce a complete image object (Weeks & Myler, 1993).

Equation 4.8 shows dilation as a mathematical expression.

$$A \oplus E_1 = \bigcup_{b \in B} A_b$$

Equation 4.8 Dilation set A and structure element $E_1$

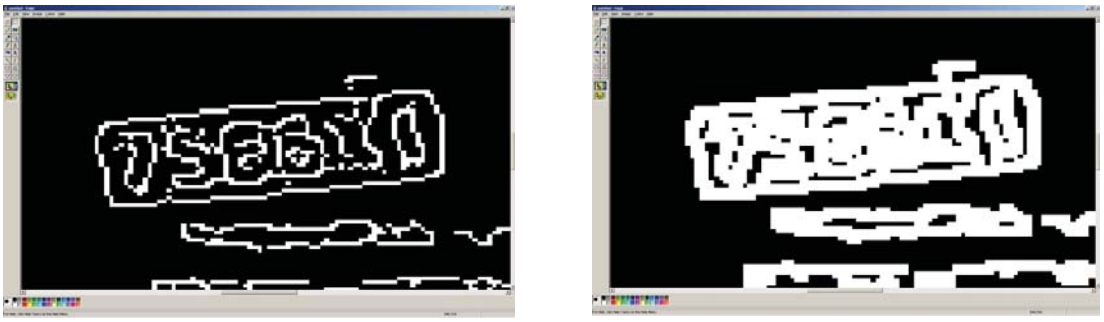
In the process of dilation, a binary image is treated as a set consisting of ordered pairs of coordinates of white pixels points in an image. An example of pixel coordination follows.




a. Example binary Image b. 3 x 3 structure element c. After Dilation

**Figure 4. 15: Binary image transformation of dilation operation (Code Source, 2005)**

The structure element passes a pixel at a time through the image. The structure element origin checks the image pixel, and if it is white, the image coordinates values are added to the structure element coordinates, which results in new coordinates. The new value of the coordinate pixel value is set to white, as seen in the example given in Figure 4.15c. This process is repeated to all the white pixels in the original image.



a. Canny image before dilation.

b. The license plate filled with white pixels after dilation using 3 x 3 structure element

**Figure 4. 16: Dilation applied to sample license plate image**

As mathematical dilation can be an iterative process, the same element design can be used to re-iterate through the image, but the image object will transform it into a bigger, less definite, object, and therefore, an inaccurate structural shape.

### 4.3.4 Binary Threshold Filter

There are several edge detection algorithms, and the Canny algorithm is proven to be the most efficient edge detection method. Nevertheless, the most efficient method does not always fulfil the task. For this reason, an alternative method is needed to deal with a failed Canny edge detection.

A simpler edge method for Canny edge detection is the binary threshold filter algorithm. This algorithm is a threshold filter driven by iterative variable threshold values as shown in Equation 4.9.

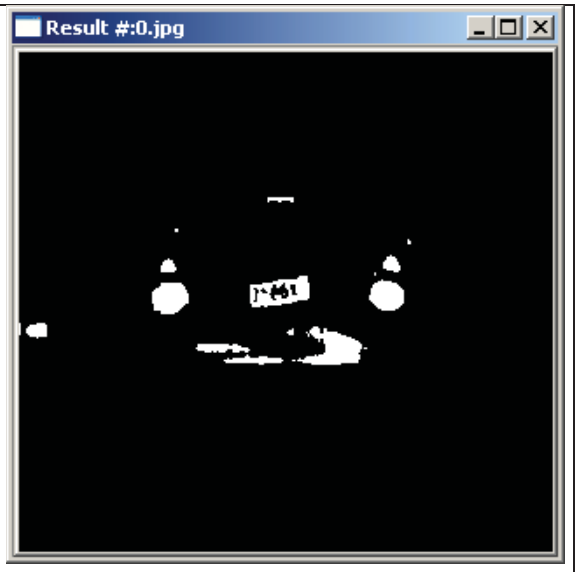
$$BTF = (L+1)*255/N$$

Equation 4.9 Binary threshold filter

In this equation, L is the interger value from 0 to N, after each iternation, and is incremented by 1, while N is the number of iternations. The value 255 is the mamixum pixel value in the image. For example, if N is set to 50, this will produce 50 edge threshold images to be processed by the detection algorithm (Wang & Lai, 2009).

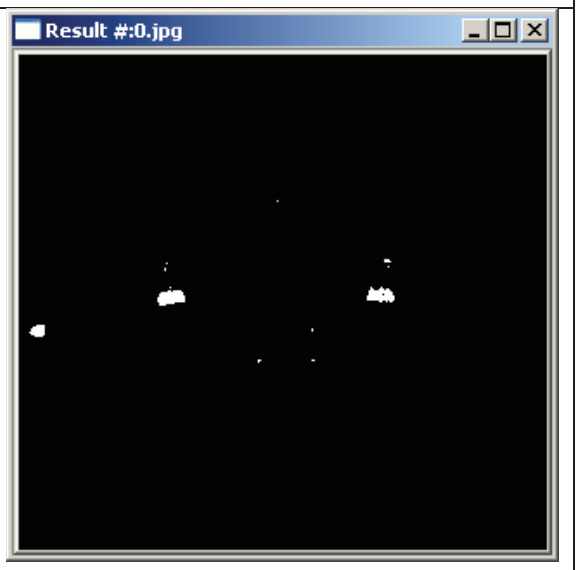
The image input is processed by the binary threshold algorithm, as part of the process setup, and every pixel in the image is examined. Beforehand, the image is expected to be pre-processed by the Gaussian smooth function, as explained in Section 4.2. In the first iteration process, the binary threshold filter value is calculated for each iteration level. The threshold value is the benchmark for every pixel value in the image. If the image pixel value is greater than the threshold value, the image intensity value is set to 255, which is otherwise 0. The final image output produces a black and white image.


A (Original image)




B (l = 10)


C(l = 20)




D (l =30)

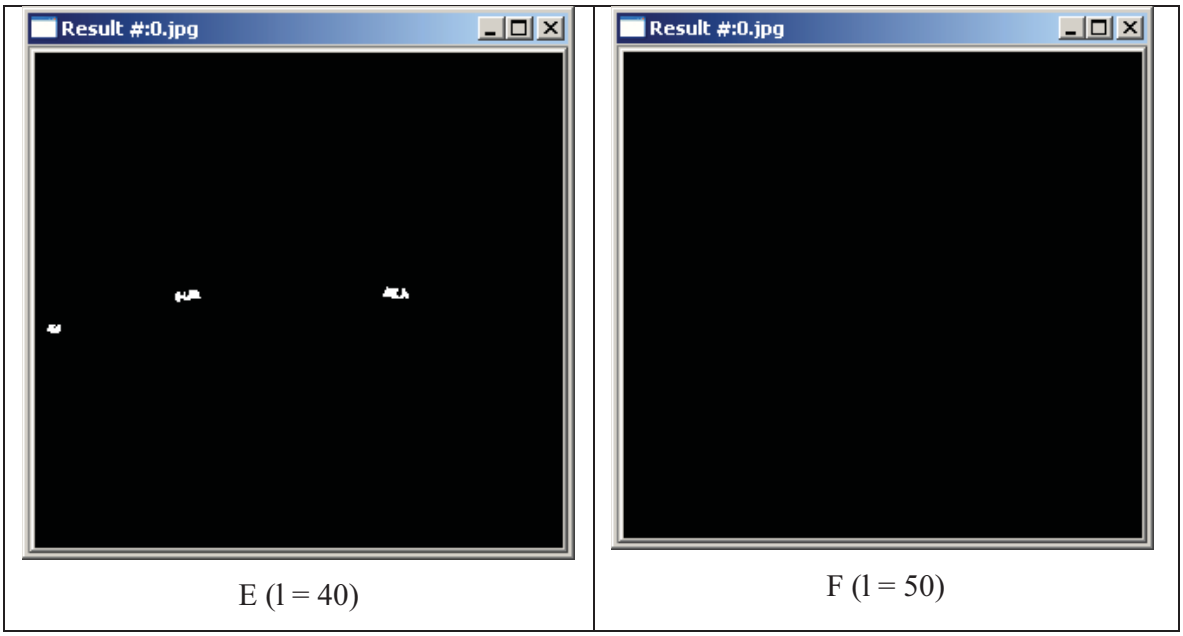



**Figure 4. 17: a – Original image, N = 50; b – Iteration 10, the license plate holds its square shape; c – Iteration 20, the license plate nearly gone; and d, e ,f – the license plate is completely filtered from the image**

Figure 4.17 shows the sequence of iteration, i.e. as the iteration gets bigger, the threshold values get bigger, and eventually the threshold value reaches 255 (the maximum pixel value). Figure 4.17F shows a blank image as there are no pixel values that are equal to 255.

The higher the value of N, the more the iterations, and therefore the better the opportunity for detecting the license plate. As seen in Figure 4.17, 80% of the iterations are wasted, as the license plate is filtered from the image. However, the percentage value changes for each image because the illumination intensity of the license plate for each image is different. The higher value of N is better for accurate detection, but it also amplifies the computation time. The optimisation of this process will be discussed in Chapter 5 (Methodology).

## 4.4 License Plate Detection

This section discusses the methods utilised to detect license plate from a given image. The detection method is split into three parts; that is, the first part marks the boundary and the region of objects in the image, the second part uses the chain code algorithm to code the image boundary, and the third part uses polygon approximation by means of the Douglas-Peucker algorithm to locate the four sided polygon and form a rectangle which is to be identified as the license plate.

As noted in Section 4.3, the pre-process output images are either in thick edge boundaries (Canny edges with dilation) or in object regions (the binary threshold).

### 4.4.1 Boundary and Region Detection

The objects in the image are represented as a collection of pixels in an image. To recognise a set of pixels as an object, the descriptor of an object can be coded by numbers. The coded numbers of the object can be compared with and recognised by other similar object code numbers. To successfully record the descriptors, the following sets of rules are applied.

1. Complete Set – Two objects must have the same descriptors only if they have the same shape.
2. Congruent Set – Similar objects are recognised when they have similar descriptors.
3. Invariant Properties – Rotation invariant descriptors will be useful for recognising objects regardless of their orientation.
4. Other Invariant Properties – Scaling, Position, Affine and perspective changes.

There are two forms of descriptors, i.e. the region and the shape. The focus of this research is on the shape boundary, which is coded by the chain code algorithm.

However, the region usually describes the contents surrounded by the boundary of an object, and this is called the region's contour. The point can be defined to be on the

boundary (the contour) if it is part of the region and there is at least one pixel in its surrounding (neighbourhood) that is not part of the region. The boundary is found by the first find which is one point on the contour, and then progresses round the contour either in a clockwise or anti-clockwise direction, finding the nearest contour point.

Connectivity can be defined as 4-way, in which only immediate neighbours are analysed, and 8-way, in which all the eight pixels around a selected pixel are analysed for connectivity. These two types of connectivity are illustrated in Figure 4.18 and Figure 4.19. In 4-way connectivity (Figure 4.18), a pixel has four immediate neighbours in its north, east, south and west direction. In 8-way connectivity (Figure 4.19), there are four extra neighbours in the directions north-east, south-east, south-west and north-west, or the points as the corners.

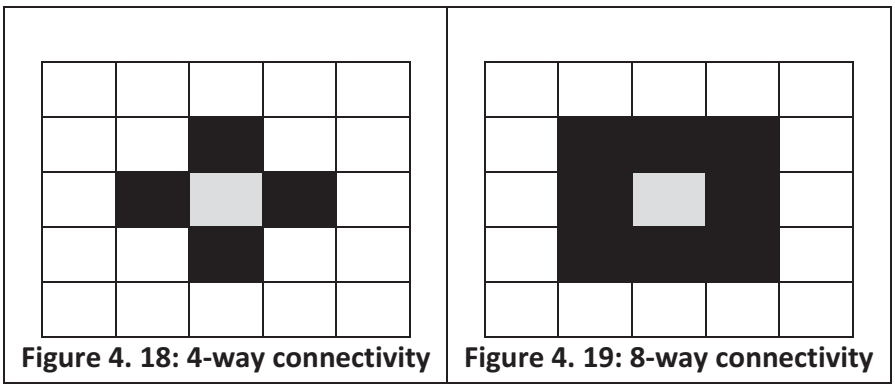
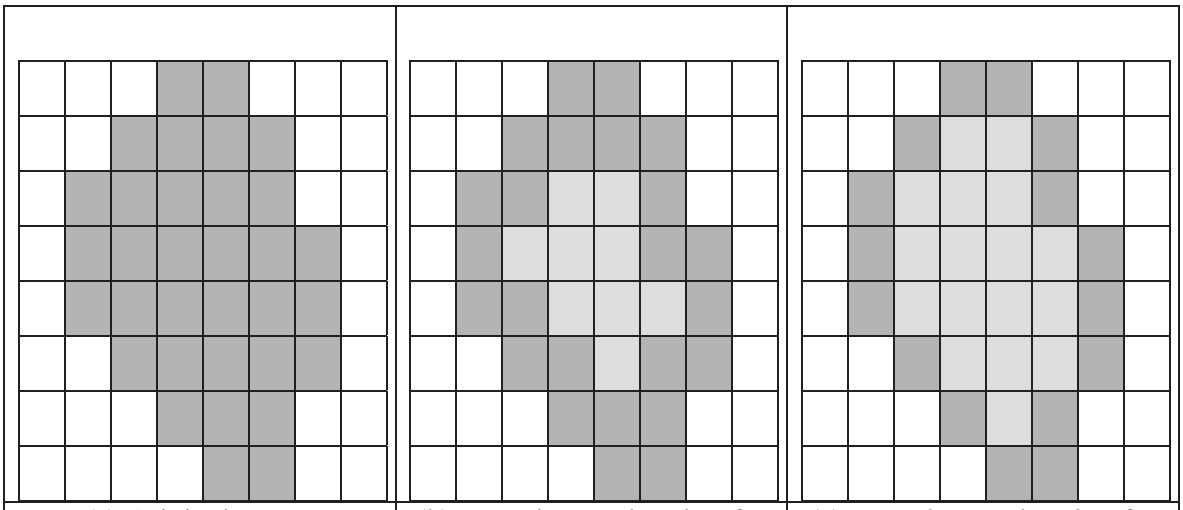


**Figure 4. 18: 4-way connectivity** **Figure 4. 19: 8-way connectivity**

The boundary and the region can be defined by using both types of connectivity, and they are always complementary. That is, if the boundary pixels are connected in the 4-way, then the region pixels will be connected in the 8-way, and vice versa. This relationship can be seen in the example shown in Figure 4.20, where the boundary is shown in dark grey, and the region in light grey. Observation of the diagonal boundary reveals that the 4-way connectivity gives a staircase boundary, whereas the 8-way connectivity gives a diagonal line formed from the points at the corners of the neighbourhood. It can be noticed that all the pixels that form the region in Figure 4.20 (b) have the 8-way connectivity, whilst the

pixels in Figure 4.20 (c) have the 4-way connectivity. This is complementary to the pixels on the border (Cui et al., 2009).



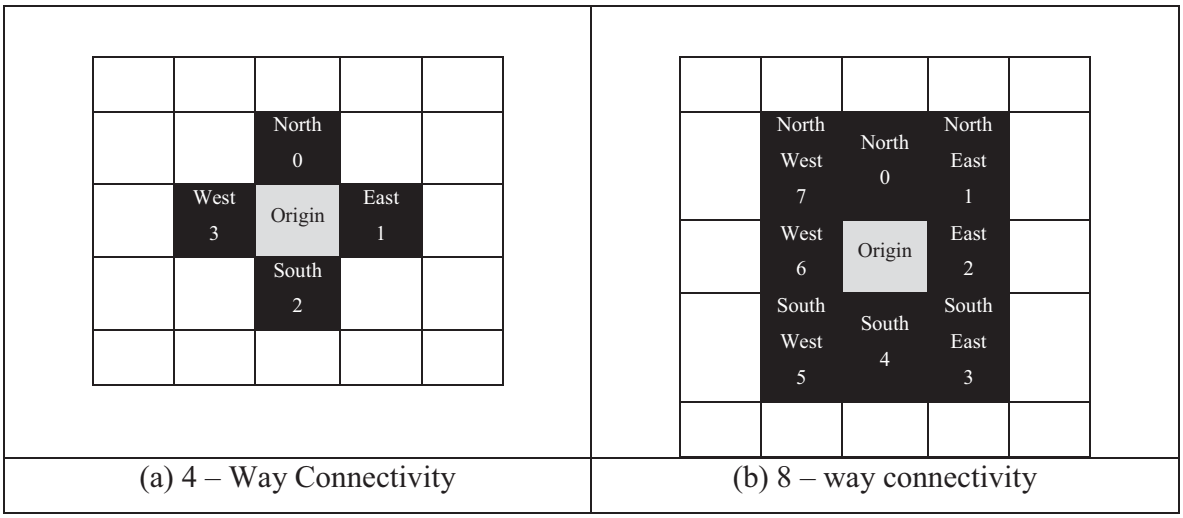
(a) Original Image | (b) Boundary and region for 4-way connectivity | (c) Boundary and region for 8-way connectivity

**Figure 4. 20: Edge connectivity map**

### 4.4.2 Chain Code Algorithm

The idea behind the chain code is to obtain a representation of the contour by simply storing the co-ordination of a sequence of pixels in the image. Alternatively, the relative position between consecutive pixels can only be stored. Essentially, the set of pixels on the border of a shape is translated into a set of connections between them. Given the complete border, and a set of the connected points, a researcher, starting from one pixel, needs to be able to determine the direction in which the next pixel is to be found. The next pixel is one of the adjacent points in one of the major compass directions. Thus, the chain code is formed by concatenating the number that designates the directions of the next pixel. The successive direction from one pixel to the next pixel becomes an element of the final code.

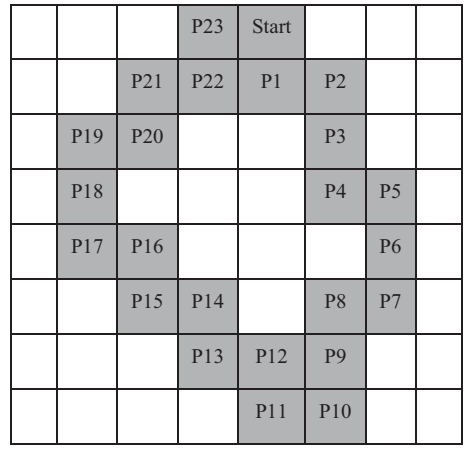
This repeats for each point until the starting point is reached when the closed shape is completely analysed.




(a) 4 – Way Connectivity (b) 8 – way connectivity

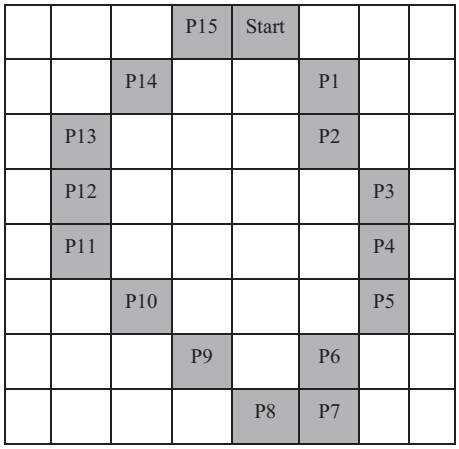
**Figure 4. 21: Chain code connectivity values**

Direction in the 4-way and the 8-way connectivity can be assigned as shown in Figure 4.21. Figure 4.22 shows the chain codes for the example region given in Figure 4.22(a). In case of the chain code for the 4-way connectivity, the direction from the starting point to the next is south, giving a code 2, and thus the first element of the chain code describing the shape is 2. The direction from point P1 to the next point, P2, is east, giving a code 1, and thus the next element of the code is 1. The next point after P2 is P3 which is south giving a code 2. This coding is repeated until P23 which is connected eastwards to the starting point, and thus the last element (the twenty-fourth element) of the code is 1. As shown in Figure 4.18 (b), while the code for 8-way connectivity is derived in an analogous way, the directions are defined according to the definition in Figure 4.17 (b). Since the number of boundary points is smaller for 8-way connectivity than it is for 4-way, the length of the code is shorter for 8-way connectivity (Aguado & Nixon, 2002).




Code =
{2,1,2,2,1,2,2,3,2,2,3,0,3,0,3,0,3,0,0,1,0,1,0,1}

(a) Chain code 4-way connectivity




Code = {3,4,3,4,4,5,4,6,7,7,7,0,0,1,1,2}

(b) Chain code 8-way connectivity

**Figure 4. 22: Chain code connectivity map**

It is logical to expect that this will be different when the starting point changes, and that the starting point invariance is needed. Thus, the elements of the code can be considered to constitute the digits in an integer. The digits can be then shifted cyclically, i.e. the least significant digit is replaced with the most significant one, and all other digits are moved one place to the left). The smallest integer is returned as the starting point invariant chain code description. Figure 4.23 shows that the original chain code is the one from the shape given in Figure 4.22. Figure 4.23 (b) shows the result of the first shift, which is equivalent to the code derived by using point P1 as the starting point. Figure 4.23(c) shows the result of two shifts, which, despite being the chain code equivalent to starting at point P2, is not a code corresponding to the minimum integer. The minimum integer code is the same as the chain code derived by starting at point P8.

Beside starting point invariance, the code can be obtained without being changed with rotation, but by expressing the code as a difference of chain code because rotation dependence is removed by relative descriptions. Change of scale may result in a set of

points, whose size is different from the original set, and thus the boundary needs to be re-sampled before coding. This can be complicated because of the effects noise can have. Although there might be difficulty with chain codes, they are simple and are generally used for shape description.

| | |
|---|---|
| Code = {3,4,3,4,4,5,4,6,7,7,7,0,0,1,1,2}<br>(a) Initial chain code | Code = {4,3,4,4,5,4,6,7,7,7,0,0,1,1,2,3}<br>(b) Result of one shift |
| Code = {3,4,4,5,4,6,7,7,7,0,0,1,1,2,3, 4}<br>(c) Result of two shifts | Code = {0,0,1,1,2,3,4,3,4,4,5,4,6,7,7,7}<br>(d) Minimum integer chain code |

**Figure 4. 23: Start point invariance in chain codes**

### 4.4.3 Douglas – Peucker Algorithm

The outputs of the chain code algorithm are chain codes which represent objects in the image. The object of interest in this study is the license plate and is identified by a rectangle shape. Simple rectangle coordinates only require knowing four vertices, which are yet unknown.

To this point, chain code algorithm will have coded other objects in the image which is not a license plate. To remove irrelevant objects and identify objects with four vertices, the Douglas-Peucker algorithm is used with the chain code algorithm to reduce the objects to four or more vertexes.

The Douglas-Peucker algorithm is a recursive algorithm. In the first pass, it creates the first approximation of the geometry that is to be simplified by joining the start and the end points directly with a straight line. In the second pass, if any of the points of the geometry lies further from this approximation, then the accepted tolerance, and the point that lies furthest away are added back, thus creating a more refined approximation. In subsequent passes, this process repeats, further refining the approximation in each pass until all of the

points within the original geometry lie within the accepted tolerance value from the approximation created.

Figure 4.24 demonstrates the four sequences from the original geometry to the final approximation in three steps. The grey shaded area is the set tolerance level. Figure 4.24 is a simple example selected to illustrate the working sequence of the algorithm, and not a live example license plate due to practicality.

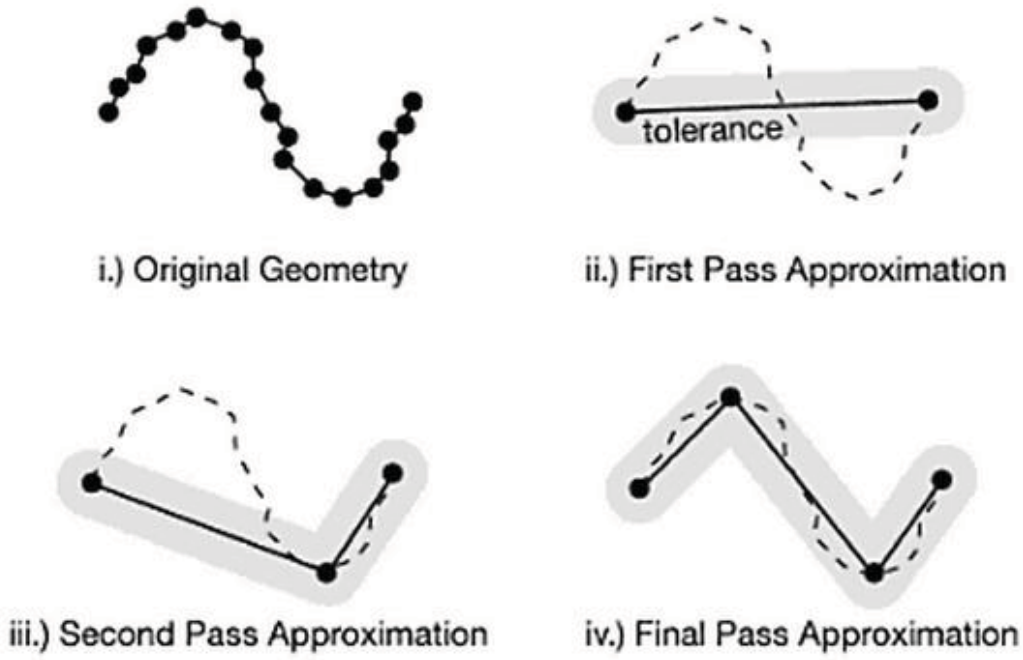



**Figure 4. 24: Douglas–Peucker approximation (Aitchison, 2008)**

## 4.5 Summary

In summary, this chapter has discussed the steps of image processing needed to acquire the licence plate from the image. The process is divided into three sections. In the first section, the enclosed image characteristics, and techniques such as the Gaussian blur, pyramid decomposition, and up-sampling were applied to combat the image noise. The second section identified image object borders by using edge boundaries found by Canny edge

detection and the binary threshold filter, and filled the broken edges with the mathematical dilation. In addition, how the image noise affects the performance of edge detection was shown. In the last section, the identified boundaries were transformed into the object boundary and regions by using the chain code algorithm, and later simplified by means of the Douglas-Peucker algorithm in order to find four vertices, which can be identified as the license plate. The concepts covered in this chapter will be discussed further in the methodology section which is set to identify the optimum settings as to achieve the high accurate license plate detection.

# Chapter 5

# Methodology

## 5.1 Methodological Design

The purpose of this experimental methodology is to simulate the mobile license plate detection system. This section discusses the environmental settings, sample collection, experimental design, and answers the questions raised by this research. The variable constraints of the design are tested and the results are presented in Chapter 6.

## 5.1 Environmental Settings

The simulation of the surveillance system was tested in real world surroundings on public roads of Auckland, New Zealand. The environment was grouped into four environment settings. Environment 1 was the near traffic junction, where the traffic vehicle speed is between 0km/h and 20km/h. Environment 2 was the residential area where vehicles travel between 20km/h and 50km/h. Environment 3 was inside and outside commercial car parks. Environment 4 was related to the travelling speed from 50km/h to 100km/h. The simulation was planned to be completed between sunset and sunrise. The sunset times were not recorded due to the fact sunset time and daylight saving vary between on different days of the year. However, the simulation should be sustainable any time after sunset and before sunrise.

Examination of each group category resulted in a different license plate size within each sample, and the license plate detection algorithm was scoped to handle such input images.

**Environment 1**

The vehicle was stationary, i.e. waiting at traffic junction point. Otherwise, it moved from a stop position, or slowed down to a junction point. The distance variation from the camera to

the vehicle in front varied depending on the speed of the vehicles. Assumption 1 was that the distance considered was approximate, and in this category, the maximum distance between the car and the surveillance camera was than 10 meters, while the minimum distance was 1 meter. Therefore, the license plate in each image varied in size, which will be discussed later in this chapter, in the research questions.

**Environment 2**

The surveillance vehicle in this category was travelling between 20km/h and 60km/h, which is the New Zealand residential speed limit, and the distance between the camera and the vehicle in front was estimated to be more than 6 meters. As distance may cause limitation, it was constrained to the camera and the infrared light source, as discussed later in the chapter. Thus, the license plate size variation was expected.

**Environment 3**

Most vehicles were stationary, and some vehicles were travelling at a slow speed below 10km/h. In most car park sites, there was significant artificial lighting. The distance between the camera and the vehicle in front varied between 1 and 5 meters.

**Environment 4**

The vehicle speed was between 50km/h and 110 km/h, and the distance exceeded 20 meters. The license plate in the image was significantly small.

The position of the camera, discussed in Chapter 3, was consistent across all four environments. The camera placement was not critical to the objective in this research, as the placement of the camera can be on the left side, the right side or the back side of the vehicle. Due to time limitation, the scope of this study limited the camera position to the front only.

## 5.2 Sample Collection

The sample images were collected in Auckland, New Zealand, in the areas of South Auckland, West Auckland and Auckland City. Each area was targeted for greater traffic population and location settings; for example, South Auckland has many shopping complexes, Manukau City and Mt Wellington Sylvia Park sustain over 2,000 car parks each. West Auckland has residence traffic with many traffic junction stops, and motor vehicles travelling at 50km/h or less. Auckland City was targeted for sampling as a metropolitan city, with a complex traffic environment at busy times of the night on Friday and Saturday. The southern motorway was also chosen collection of samples because many vehicles travel at 100km/h.

To eliminate overloading of unwanted sample images, the convenience sample method was utilised. The convenience model was grouped into four image environments, as noted in Section 5.1. The sample collection required two individuals, one driving the surveillance motor vehicle with the camera positioned on the front bumper, and the passenger taking images of motor vehicles as seen on the screen of the net-book computer, with the camera fed in via an USB cable, as explained in Section 3.6. For each environment introduced in Section 5.1, the sum of 500 sample images were saved in the notebook, with the final count of 2,000 images to be tested for all the environments.

Each sample environment holds various image characteristics, such as size, location, orientation of the license plate, and unwanted noise objects embedded in the image.

However, one condition must be met to incorporate an image in the sample. The sample needs to show some visibility of the license plate; however the quality of the image was not controlled and therefore varied depending on the surrounding. Conversely, in some samples, the illumination or contrast may be very low as visibility is faint. Other samples represent half of the license plates due to the direction of a motor vehicle in respect to the camera or the objects which partially blocked the view of the license plate, such as a car tow bar. Nonetheless, the sample collection was random in order to represent a nonbiased sample of the population.

The validity and reliability of the experiment is required to capture the intended sample images. The image samples were collected for periods of time, and consequently required each image to display a license plate, but the characteristics of the license plate, such as location, size, and orientation were not important. However, the limited image noise was ideal. For the examination of this research, a manual human operator was used to reduce non-license plate images. The consistency of the sample images was in control of the human operator.

## 5.2 Experimental Design

### 5.2.1 Hardware Design Simulation (Location, Camera, Infrared Source, and Infrared Filter)

The hardware simulation consisted of four components, and the location of the surveillance unit, as the topic of this thesis suggests, was set to mobile. This setup was more challenging in terms of designing of the algorithms to detect the license plate. The literature reviewed has also supported the idea that future surveillance will be mobile, and expressed the challenges in detection of the license plate in various surroundings. As a response, this research used the concept of placing the camera on the motor vehicle and scanning images while driving on the road.

The camera location on the front bumper was a responsible decision to improve the range and coverage of surveillance unit. The camera operation faced two challenges; one challenge was that the vehicle vibration could cause distorted pixilation of the image. The camera chosen for this task had steady still adjustment functionality (the camera specification is given in Section 3.2). Vibration of the motor vehicle during movement is not within the scope of this study, but a future study could investigate this area to improve performance. The second challenge was observed while collecting image samples. The camera in the horizontal position had a redundant image view of the road. To maximise this view, the camera was adjusted upwards by 16.05 degrees from the horizontal reference point. This adjustment improved the visibility of the license plate where the location of the

plate was on the high side of the vehicle instead of the normal bumper of the vehicle.

In limited light conditions, the camera requires an external light source to illuminate the surrounding environments. Section 3.1 explains how an infrared light source can supply the required light source. The wavelength of this light source is 850nm, and this light source is within the range of the heat emitting radiation of most electrical devices such as the light bulb and the engine heat. The ideal wavelength would be outside the range of heat radiation, but the maximum range of the camera detection is limited to 900nm wavelength, which is within the infrared range. The wavelength between 380 and 780nm is in the visible range seen by humans. Accordingly, this study was limited to the infrared frequency of 850nm. The manufacture label of the infrared light source reads the dispersion range is 20 meters. A real life testing showed that the illumination range degrades after 10 meters, and it limited to the head in focus, and the wide focus was limited to 5 -10 meters. However, infrared source adjustment is not within the scope of this study, but future work can focus on a better infrared light source to improve the visibility range.

The infrared filter glass (850nm) was inserted in front of the camera lens to stop white light frequency (all frequency ranges) to be detected by the camera. This filter process is important to reduce detection of noise objects by the camera. Section 3.3 explains the setup process and the integration into the surveillance system.

The hardware units were put together temporarily to allow the sample collection. Chapter 3 explains the setup process and integrates of the complete hardware system.

### 5.2.2 Software Design Simulation

The image algorithms discussed earlier were developed into native software algorithms to fulfil the requirements of this thesis. The task of this research is to simulate the image processing algorithms selected to test its validity and sustain results which can reliably answer the research questions. To complete the software simulation, the following

development tools were used to support the research findings: Visual Studio 2005, using C++ with an openCV image library.

### 5.2.3 Software Development Platform

OpenCV is an image library - developed by Intel, first released to public at an IEEE Conference on Computer Vision and Pattern Recognition in 2000. In 2006, version 1.0 was available for the public, and was adapted by this research. Version 1.0 was developed in C++; therefore, the simulations will also use C++.

### 5.2.4 Function: Main ()

The C++ code executes the first function tmain (), as shown in Figure 5.1. With no input parameter, the saved images are retrieved from a separate folder to be processed. This configuration was used for the simulation purpose only; however, in the real software package, the setup is different, and the camera is fed into a live image and processed immediately instead of being saved in the image files as the case was in this research.

```
1
2  #ifdef _CH_
3  #pragma package <opencv>
4  #endif
5
6
7  #include "stdafx.h"
8  #include "cv.h"
9  #include "highgui.h"
10
11 #include <math.h>
12 #include <string.h>
13 #include <stdlib.h>
14 #include <iostream>
15 #include <sstream>
16
17
18 IplImage *frame, *frame_copy, *step, *frame_segment =0;
19 CvMemStorage* storage = 0;
20 int imageCount =0;
21
22 char before[100];
23 char after[100];;
24
25 void capture();
26 void edgedetection();
27 void drawRectangles( IplImage* img, CvSeq* squares );
28 CvSeq* findLicensePlate( IplImage* img, CvMemStorage* storage );
29 void imageinfo();
30
31 int _tmain(int argc, _TCHAR* argv[])
32 {
33   capture();
34
35   return 0;
36 }
37
```

**Figure 5. 1: Main function with no input parameter**

### 5.2.5 Function Capture ()

The main function call is Capture (), in which the main loop goes through images to the directory and processes one image at a time. In Figure 5.2, Line 40 shows the counts from 0 to 20. If found in the directory, 20 images will be processed; in each loop, the function resets the parameter values and calls other functions. The function purposes to load the image into the memory and process the image by calling findLicensePlate () and then drawRectangles () which outlines the license plate if found. If the license plate is found, it

is shown in a separate window with a green border around the license plate. If the plate is not found, the image returns a black screen. Once the screen is closed, the next image is loaded and processed, and this process repeats until the FOR loop is completed.

```
void capture(){

    for (imageCount =0; imageCount <=20; imageCount++){

        char dirPath[100];
        char temp[100];
        char ext[20];

        strcpy(ext,".jpg");

        itoa(imageCount,temp,10);
        strcat (temp, ext);
        strcpy(dirPath,"C:\\Documents and Settings\\Study\\Desktop\\3.4.09\\");
        strcat (dirPath,temp);

        frame = cvLoadImage(dirPath);
        if (!frame){
            break;
        }
        storage = cvCreateMemStorage(0);

        strcpy(before,"Sample #:");
        strcat(before,temp);
        strcpy(after,"Result #:");
        strcat(after,temp);

        cvNamedWindow( before, 0);
        cvNamedWindow(after, 0);

        imageinfo();

        drawRectangles( frame, findLicensePlate( frame, storage ) );

        cvShowImage(before, frame );
        cvShowImage(after, frame_copy);

        int key = cvWaitKey(0);
        if( (cvWaitKey(10) & 255) == 27 ) break;
        cvReleaseImage(&frame_copy);
        cvReleaseImage(&frame);
        frame_copy = 0;
        frame =0;
        cvDestroyWindow(before);
        cvDestroyWindow(after);

    }
  cvReleaseMemStorage( &storage );
  return;
}
```

**Figure 5. 2: Main loop to process solitary image**

### 5.2.6 Function findLicensePlate ()

The objective of this function is to identify square objects in the image. The code specific instructions are continuation of Chapter 4 in interpreting the image processing algorithms into conceptual C++ coding, which is shown in Figure 5.3. In Line 118 to 125, local variables are declared, and the use of the variables is explained in the rest of this section.

Line 132 shows the function cvPyrDown (timg, pyr, 7) in which the input parameter is 'timg' in a grey illuminated image from the sample collection. The output, however, is 'pyr', which is half of the resolution of the original image. The function firstly applies the 2D Gaussian blur filter, with the kernel size 7 (Section 4.2.2), and then pyramid decomposition (Section 4.2.3) which removes even columns and rows from the image, and shrinks the image down to half of the original size. The image noise is firstly spread so that it is diluted, and then even rows and columns remove more information from the image and reduce the noise further.

In Line 132, the even rows and columns are removed from the previous cvPyrDown function, which, updated with value 0's into even rows and columns, brings back the resolution to the original size. After this process, the image is once again processed with the 2D Gaussian blur filter, with the kernel size 7. In completion, the image is further diluted from noise effects.

Noise removal is applied to all images, and the effect is seen by blurriness of the image. Diluting the noise impacted region will improve the image quality for the next image process, and ultimately improve the chance of detection of the license plate.

Although the input image is of grey scale, the image is thought be of the RGB format. In Line 139, the FOR loop is used to process the three image channels, and each channel is similarly processed, but the outcome can be different.

The next image process is the Canny algorithm, (explained in Section 4.3.2) which imprints only the edges of the image object. However, only the dominant edges are shown, while

weaker edges are filtered out. The final image singles out the edge borders around an image object. In Line 151, the function calls for cvCanny, where the parameters are the input image, the output image, the hysteresis threshold values and the Sobel kernel size. The hysteresis threshold values are set to 0's for design reasons, discussed in the answers to the questions raise in this section, and Sobel edge detection with the kernel 3 is selected. The selected kernel size is set to differentiate weaker edges from strong edges. The image quality is unknown, and thus the image process needs to be considered in case of the worst possible image quality.

The Canny algorithm is followed by mathematical operation dilation. Due to the singularity of the edge borders, the edge holes could cause incomplete border connection and later fail to detect the contour of the object. To fill the border gaps, image dilation is applied, and the edges are filled with pixel values to fill the holes near the edge boundary.

The Canny algorithm is one method of detecting the edges around the border, while another method is the binary threshold (see Section 4.3.4). The binary threshold is a simpler concept, but the process is intensive. The greyscale image is thought to have illumination values from 0 to 255, an object in the image is considered as a single block group of pixels with similar illumination values. In principle, a simple binary threshold can be applied as its effectiveness is shown in iterations. The maximum iteration possible is 244, which also means that 244 images will be produced and this is a waste of resources. The selected iteration for this research was 50, and hence 50 images were processed in addition to the Canny output image. Accordingly, 51 images needed to be processed by the license plate detecting algorithm, and the iteration levels were adjusted in the simulation run to discover the optimum value.

```
115 CvSeq* findLicensePlate( IplImage* img, CvMemStorage* storage )
116 {
117
118     CvSeq* contours;
119     int countRect, rgbLevel, countIterations, iterationLevel = 50;
120     CvSize sz = cvSize( img->width & -2, img->height & -2 );
121     IplImage* timg = cvCloneImage( img );
122     IplImage* gray = cvCreateImage( sz, 8, 1 );
123     IplImage* pyr = cvCreateImage( cvSize(sz.width/2, sz.height/2), 8, 3 );
124     IplImage* tgray;
125     CvSeq* result;
126
127
128     CvSeq* squares = cvCreateSeq( 0, sizeof(CvSeq), sizeof(CvPoint), storage );
129
130     cvSetImageROI( timg, cvRect( 0, 0, sz.width, sz.height ));
131
132     cvPyrDown( timg, pyr, 7 );
133
134     cvPyrUp( pyr, timg, 7 );
135
136     tgray = cvCreateImage( sz, 8, 1 );
137
138
139     for( rgbLevel = 0; rgbLevel < 3; rgbLevel++ )
140     {
141
142         cvSetImageCOI( timg, rgbLevel+1 );
143         cvCopy( timg, tgray, 0 );
144
145
146         for( countIterations = 0; countIterations < iterationLevel; countIterations++ )
147         {
148
149             if( countIterations == 0 )
150             {
151                 cvCanny( tgray, gray, 0, 0, 3);
152
153                 cvDilate( gray, gray, 0, 1);
154
155             }
156             else
157             {
158                 cvThreshold( tgray, gray, (countIterations+1)*255/iterationLevel, 255, CV_THRESH_BINARY );
159
160             }
161             cvFindContours( gray, storage, &contours, sizeof(CvContour),
162                 CV_RETR_LIST, CV_LINK_RUNS, cvPoint(0,0) );
163
```

**Figure 5. 3: findLicensePlate Function**

When the edge boundary is detected, this needs to be transformed into readable locations. The chain code algorithm interprets the boundary location and direction by using the 8-way connectivity model in order to map out the boundary of the image object (Section 4.1.1). However, an object is defined as a closed boundary loop, and if the boundary is not complete, then the chain code is dropped from the image. Via the chain code algorithm, the enclosed boundary edges is grouped and referenced as contour sets. Line 166 in Figure 5.4 shows this process completed.

In Figure 5.4, Line 169 – 176 shows process of the contour sets. Although the contour set identifies all the points in the chain code, in this research purpose, it was not necessary to know all the edge points. The objective here was to locate contours with quad corner coordinates or more, which is a property of the license plate. To reduce multiple edge points to four vertices coordinates, the Douglas-Peucker algorithm was applied to the approximate the chain code boundary. In Figure 5.4, Line 166 returns the number of points in the contour set. Douglas-Peucker approximation stops when no coordinate points on the approximate boundary lines is found above the threshold level and this threshold is 'cvContourPerimeter (contours) *0.08' (see Section 4.4.3 for more details).

After completion of Douglas-Peucker approximation, the new found vertices are processed if the contour has more than three vertices, and a further check is completed on the area of the contour. If the area of the contour is below the threshold limit, then it is not a license plate. The final check is done to make sure the object contour is inside the close loop system, instead of the outer closed loop system, in which case the area will be extremely large. Code implementation is illustrated in Line 170-171, in Figure 5.4.

The four coordinates are saved into the memory as a square object, by the FOR loop through the coordinate system as shown in Line 173 – 174 in Figure 5.4.

After the above process of identification of suitable license plate objects, it is repeated for all the contour sets. This is an iterative process and contour sets vary depending on the image.

```
            while( contours )
            {
                result = cvApproxPoly( contours, sizeof(CvContour), storage,
                    CV_POLY_APPROX_DP, cvContourPerimeter(contours)*0.08, 0 );

                if( result->total >3 &&
                    fabs(cvContourArea(result,CV_WHOLE_SEQ)) > 1000 &&
                    cvCheckContourConvexity(result) )
                {
                for( countRect = 0; countRect < 4; countRect++ )
                    cvSeqPush( squares,(CvPoint*)cvGetSeqElem( result, countRect ));

                }
                contours = contours->h_next;
            }
        }
    }


    cvReleaseImage( &gray );
    cvReleaseImage( &pyr );
    cvReleaseImage( &tgray );
    cvReleaseImage( &timg );

    return squares;
}
```

**Figure 5. 4: Continuation of findLicensePlate Function**

### 5.2.7 DrawRectangles() and ImageInfo()

The found vertices are possible locations of the license plate, shown in the original image as an identified license plate. There are no validation checks of these coordinates at this point, and the function aims to draw the coordinates onto the original image. The copy of the original image is marked with green pixels by joining the vertices saved in the image. The capture () function shows the modified image of the license plate detected by marking green lines around it, as coded in Figure 5.5.

```
88  void drawRectangles( IplImage* img, CvSeq* squares )
89  {
90      CvSeqReader reader;
91      IplImage* cpy = cvCloneImage( img );
92      int countVertices;
93
94      cvStartReadSeq( squares, &reader, 0 );
95
96
97      for( countVertices = 0; countVertices < squares->total; countVertices += 4 )
98      {
99          CvPoint pt[4], *rect = pt;
100         int count = 4;
101
102         CV_READ_SEQ_ELEM( pt[0], reader );
103         CV_READ_SEQ_ELEM( pt[1], reader );
104         CV_READ_SEQ_ELEM( pt[2], reader );
105         CV_READ_SEQ_ELEM( pt[3], reader );
106
107
108         cvPolyLine( cpy, &rect, &count, 1, 1, CV_RGB(0,255,0), 3, CV_AA, 0 );
109         frame_copy = cvCloneImage( cpy );
110
111     }
112     cvReleaseImage( &cpy );
113 }
```

**Figure 5. 5: Draw Contour Boundary**

```
191 void imageinfo(){
192
193     int height, width, step, channels;
194
195     height = frame->height;
196     width = frame->width;
197     step = frame->widthStep;
198     channels = frame->nChannels;
199     printf("Processed image %dx%d with %d channels, Image number = %d\n", height, width,channels,imageCount);
200     return;
201 }
```

**Figure 5. 6: Image property information**

In Figure 5.6, the code displays the image information that is being processed by the software system. This information display is for simulation purpose only.

The experimental design embraces various algorithms to route an image to extract the license plate. The measured variables influence the result of the complex algorithm design. It is overwhelming to include all the variables; as a result, the thesis is limited to the controlled, driving and output variables.

The controlled variables considered not to change in the test apparatus are the following:

- Image resolution,
- The Gaussian blur kernel size (up and down pyramid composition), and
- The image sample set.

The driving variables considered to influence the output results are the following:

- The Canny hysteresis threshold levels,
- The binary threshold iteration, and
- The Douglas-Peucker tolerance level.

The output variables from each dataset of 500 images are the following:

- The total of license plates detected,
- The missed detections, and
- The total of incorrect detections.

## 5.3 Research Questions

1) What hardware\software limitations are experienced in the operation of the mobile system under limited light conditions?
2) What considerations should be taken into account with regard to the design algorithms in order to detect the license plate?
3) Is the license plate system optimised for all environmental settings?

**Answering Question One, Part One: What hardware limitations are experienced in the operation of the mobile system under limited light conditions?**

In this experimental simulation of the mobile license plate system, several physical hardware challenges were observed. Ideally, the hardware component operates equally in all environmental settings; however in 'real world' applications, such a setting is not possible, and these issues were addressed. Under the scope of this thesis, four

environmental settings were considered to be an unbiased population sample for the simulation.

One uncontrollable condition in this experiment is the variable - environmental change, and hence certain limitations were experienced and striking beyond the scope of this study.

- Distance of the license plate in the sample four (in which the vehicle speed is 50km/h – 110 km/h). The approximation distance between the camera and license plate was greater than 10 meters, and the visibility of the infrared light under this distance and speed conditions were poor. Therefore, the images in the sample were blank, or a block of noise was captured. The hardware used in this research selection was not considered to render high performance, nor in need of a high performance camera or a infrared light source because it could produce a near-perfect image quality that would not stretch the performance of software algorithms design to detect the license plate. Due to this limitation, the sample four was excluded from this experimental simulation, and left to be investigated further in a future study.

- The filter design, the infrared light source and the infrared filter were set to eliminate noise from the image, although noise was expected to be captured from other infrared light sources. The noise levels were higher than expected. Both sample one and two recorded noise from different sources. The subsequent noise seen in the image was traffic lights, tail lights, head lights, and lamp post light. Evidently, the list of noise types was emitting 850nm wavelength which was also used in this study. To overcome this problem, the research called for another frequency range that is not openly present in the open environment, and this investigation is also left to be considered in future research.

**Answering Question One, Part Two: What software limitations are experienced in the operation of the mobile system under limited light conditions?**

The software algorithm designed for this simulation experiment is for the worst case scenario in order to enable the system to operate in various environmental conditions. To achieve good outcomes, the controlled limitations were observed as follows:

- The process time is important for a surveillance system, which has also been addressed in the literature. The time monitored for performance as the threshold stands at 50ms or below, for resolution of an image of the size 640 x 480, processed in Pentium 4, 2.2 GHz. If resolution is decreased the process time also decreases, but if it increased, the process time increases. The relationship between process time and resolution was not measured in this study. A future study may change the resolution for better performance at a greater distance from the camera to the license plate.

- Concerning the plate visibility due to obstructions in the environment, the observed sample collection detected license plates with half a plate missing due to the angle, the tow bar or other obstacles. The design structure of the software algorithm is instructed to identify rectangular or quadrangular objects. There is no liberty for other criteria to be included in order to overcome object obstructions, which may lead to failure in identification of the license plate.

- The Canny edge detection algorithm is a key attribute of identification of the edges and marking of edge boundaries. One edge model is insufficient if the image characteristic does not favour the Canny edge model. Thus, a backup model (the binary threshold algorithm) was integrated into the software design in this research. However, the binary threshold is driven by iteration loops to produce numerous output images, and this puts load onto the contour chain code and Douglas-Peucker algorithms. Many images from binary threshold images were redundant, but 10% were suitable enough for plate detection. The ratio of the suitable image quality fluctuates depending on the input image quality. In this research, the maximum

iteration level was set to 50 to process a single image through this software simulation design.

**Answering Question Two: What considerations should be taken account with regard to the design algorithms in order to detect the license plate?**

The software design was confined by the top to bottom approach. The plate detection technique selected for the contour and the Douglas-Peucker algorithm compelled the design of the software noise filters and edge detection models. To fulfil the experimental simulation requirements, the following criteria were applied to satisfy the research design.

- Any plate size detectable

  The plate size varies in each image sample. The contour chain code algorithm is motivated to trace any border that is enclosed. The license plate size can be of any size to satisfy the chain code algorithm; however, the plate boundary needs to be enclosed to be able to trace the edge boundary. In this research, plate detection was completed by the vertices rule check. The enclosed boundary of the object after the Douglas-Peucker algorithm should have a total of 4 vertices. In some cases, the license plate can be screwed to the left or the right, and may have a total of 3, 4 or 5 vertices. The area of the enclosed boundary was also measured, as contour objects are not limited in size, and the algorithm picks up any image objects with 3, 4, or 5 vertices. To filter the license plate, the area of the enclosed boundary is required to be above the threshold value.

- Any plate orientation detectable

  A non-horizontal plate orientation is acceptable in the New Zealand motor vehicle law, and thus, vertical or non-horizontal plate locations can be used by motor bikes, trailers and other standard vehicles. To ensure that detection is possible in unlikely orientations, Canny edge detection does not work differently in detecting the edges, as the edges are detected regardless of the direction. The binary threshold algorithm is also resistant to different orientations, and any object with enclosed areas is a possible plate. As the contour chain code and the Douglas-Peucker algorithm are

also resistant, the object boundary edges can have any direction, and if four vertices are present, then the plate is detectable.

- Plate deformation – a trapezium object shape detectable

  Because of the open road environment, a defect image is possible in rare circumstances. Nevertheless, possible plate detection is limited. As one restriction is the boundary edges, the shape of the plate can be screwed to the right or left as long as the boundary edges are intact. If the boundary edges are missing or not connected, the contour chain code algorithm fails, and the object is dropped from the group. Such a licence plate may not be readable or recognised as a license plate in most circumstances, but in rare conditions such events can occur. The flexibility of detection of such an object is limited as the possibility of a non-license plate is high, such as advertising stickers, road marks and rectangular objects on motor vehicles. A special care must be taken to determine the correct detection of the plate.

- Reducing inaccurate detections

  The plate detection process is significantly open to license plate like-size objects. In open environment conditions, an object similar to a license plate opens the opportunity for incorrect plate detection. To reduce this occurrence, vertices angle measurement was considered in this study. As the angle of the four vertices points must be close to 90 degrees, in a perfect world condition all the license plates will be recorded with the 90-degree angle. However, in real world conditions, this is not possible. Thus, the angle filter is not an option for reducing inaccurate detection. The area of the license plate can be considered, but this is also limited. The plate size can be significantly small or very big, but there is limitation on how small a license plate can be to read the plate numbers or the character recognition algorithm. The limitation of recognisable characters should be the lowest threshold limit on the license plate size as to reduce inaccurate detection. Conversely, there is no limitation on the bigger license plates because such big plates are possible when the camera is very close to the license plate.

- Design - usable in any light environment

  Concerning the design methodologies for simulation in limited light conditions, the research design is usually scoped to test the consistency of the license plate detection under limited conditions. On the other hand, the software algorithm design is not limited to such a research design, but optimised for limited light conditions. The research therefore did not cover natural light conditions. This research design can be applied to a natural light environment with the same hardware setup, but the license plate detection accuracy may descend due to more visible objects seen in natural illumination. The usability of this research design in natural light is left to a future study to investigate.

**Answering Question Three: Is the license plate system optimised for all environmental settings?**

The objective of this research design is to locate the license plate in all environmental limited light settings. Initially, there were four categories, but one category was dropped due to insufficient hardware performance to process illumination of the infrared light and its inability to capture suitable images to process. The three remaining categories were examined for maximum performance. The software and hardware design was restricted to one design methodology to satisfy all the three sample categories.

Firstly, optimisation of the hardware design was carried out. The location of the camera and the infrared light source were suited for the environmental sample 1 and 2, but the sample 3 was located in a commercial car park. The front location of the camera was not ideal because of vehicles parked perpendicularly to the vehicle with camera, blocking the view of the license plate. However, the moving vehicles in the car park were in front of the camera at a close distance. The location of the camera was suited for majority environmental samples, and as such, it was the optimum location.

The camera angle view was optimised by reducing the camera view of the road. Section 3.5 measured the angle to improve the visibility of the licence plate. Increased visibility improves the chance of capturing license plate. The selected camera capability provided

steady shots because, despite the camera location on the front bumper, the constant vibration produced by the chassis of the vehicle did not reduce the image quality. The infrared light source was adjacently placed next to the camera to maximise the illumination range and visibility (Section 3.1). The infrared filter in the camera further optimised the hardware system by filtering noise objects before they could enter the camera lens (Section 3.3). In summary, the final hardware design was optimised to operate on a level required by this research design.

Secondly, with regard to the software algorithm, the approach of the algorithm design was the top to bottom approach. The license plate was located by means of the four edge boundary vertices found in the image objects, and the vertices position was exploited to optimise the license plate design.

The first image process to reduce the image noise was embedded in the image. To remove noise, the 2D Gaussian blur was completed. The Gaussian blur consists of a set of kernel values found in the normal distribution by calculating the mean illumination values in the image, which determines the standard deviation. The standard deviation controls the severity of the Gaussian blur by spreading the illumination values in the pixels to the neighbouring pixels. In this way, if noise was present in the image, it was diluted from strong to weak pixilation values. The standard deviation is dependent on the image if the image consists of varying illumination values ranging from 0 to 255, then standard deviations value is high. If the illumination values are less spread, then the values are similar, the standard deviations is small and the Gaussian blur kernel is less aggressive. However, the kernel of a fixed size (7 x 7) was chosen for this research design because a bigger kernel size would weaken the noise further by spreading the illumination values. The impact on kernel was the same across all the images. As the design needs to cater for a good and bad image quality, the approach chosen to reduce image noise was sufficient. The 2D Gaussian is a fast iteration process because it reduces computation time compared with the 1D Gaussian blur, and the final result is the same.

After completion of the Gaussian blur, the image was down-sampled by removal of even column and rows, which reduced the image resolution by half. Then the image was restored to the original resolution by injecting even columns and rows with the illumination value 0.

Next, another Gaussian blur (with the kernel size 7 x 7) was completed to finish the noise removal process. The image state was described as blurry; the noise removal operation was completed on all the images before proceeding to the next step. The steps taken so far to reduce noise benefited the edge detection process, and eliminated smaller noise objects.

Two edge techniques were applied to improve the detection of the license plate. The first technique was Canny edge detection, in which the output returns an image with a single edge boundary to represent the object found in the image (Section 4.3.2). As the Canny edge algorithm can be altered to detect strong edges and ignore the weak edges, it was controlled by the hysteresis threshold level. To optimise the plate detection, all the edge boundaries in the image should remain no matter how strong or weak the edge boundary seems. This configuration is only suitable if the image does not contain a large amount of information, which was the case in all the three simulation sample categories. If the image is overloaded with information, the detection process is overloaded and detection is possible, but with elevated false detection can also be expected. To reduce misdetection in this research, the detected edges in the Canny process were varied in hysteresis threshold levels.

The second edge technique was situated for the alternative method applied to the edges missed in Canny detection. The binary threshold method is an iterative process which produces many output images dependant on the iteration level. The higher the iteration level, the more output images are produced with the binary threshold levels. However, in this research, only 10 percentage of the output was of value. The other 90 percentage was out of range and insufficient to detect the license plate. It is not possible to determine the 10 percentage range due to various image samples which produce different illumination levels, and the ratio of the 10 percentage sample is different for each image. Ideally, the higher the iterations, the better detection, but it is not efficient. For this reason, different iteration levels were measured to determine the best value for the iteration level.

The next image process is the edge boundary found by Canny edge detection and the binary threshold. The contour technique explained in Section 4.4.2 produced chain code coordinates of all the bounded edges, and it was not possible to apply this standard to improve efficiency. However, the chain code coordinates were then processed by the

Douglas-Peucker algorithm which simplified the chain code coordinates to four vertices. Although other objects were also present, more vertices were discarded. The object-like license plate of interest had only four vertices. The Douglas-Peucker algorithm was an iterative approximation, whereby the algorithm stopped processing the bound object, when all the points in the boundary were within the tolerance level. High tolerance resulted in more detailed boundary points, where low tolerance was a rough estimate of the boundary points. The suited tolerance level for this research design was found by repeating calibration samples.

## 5.4 Data Collection / Procedures

The complexity of the algorithm design allowed many variables to be configured. The variables measured ranged from many values and thus approximate meaningful variables were selected to optimise the research design. The key variables influencing the performance of the research design were the two edge detection techniques, and thus nine simulations were conceived for testing.

Simulation Summary

Three environmental samples to process, 500 images each.

1) Sample 1: the vehicle Speed of 0 – 20km/h (at traffic junctions)
2) Sample 2: the vehicle speed of 20 – 60km/h (in residential areas)
3) Sample 3: the vehicle speed of 0 – 10 km/h (in car parks)

Each simulation to process three samples with a total of 1500 images.

The controlled variables:

- Image resolution (640 x 480)
- The Gaussian blur kernel size (7 x 7)
- The same samples for each simulation

<table>
<tr><td colspan="2">

- The Douglas-Peucker tolerance level: 8%
- The minimum threshold area > 1000 pixels
- The filter vertices of objects: 4

The output variables measured per image:

- The correct license plate detected
- The license plate not detected
- Incorrect detection

</td></tr>
<tr><td>Alpha</td><td>The driven variables:<br>• Only Canny edge detection<br>• Hysteresis threshold values:<br>Alpha 1: the low limit = 10, the high limit = 50<br>Alpha 2: the low limit = 20, the high limit = 40<br>Alpha 3: the low limit = 30, the high limit = 30<br>Alpha 4: the low limit = 0, the high limit = 0</td></tr>
<tr><td>Beta</td><td>The driven variables:<br>• Only the binary threshold<br>• Iteration levels:<br>Beta 1: iteration = 20<br>Beta 2: iteration = 30<br>Beta 3: iteration = 40<br>Beta 4: iteration = 50</td></tr>
<tr><td>Gamma</td><td>The driven variables:<br>• Combined Canny edge detection and the binary threshold<br>• The hysteresis threshold values: lower = 0, high limit = 0<br>• The binary threshold: 50</td></tr>
</table>

## 5.5 Sample Environment Settings

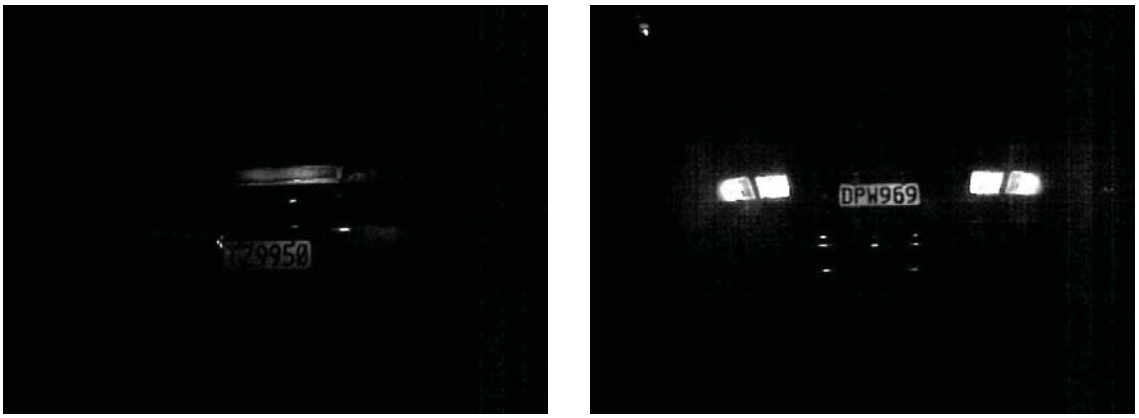



**Figure 5.7: Setting 1 Vehicle Speed 0 – 20 km/h**

Figure 5.7 shows two sample images captured in setting 1. The image on left is faded on the left side, but still detectable because of the border edge connection, where else image on the right shows clear image of the license plate but contains other image noise - visible brake lights.

**Figure 5.8: Setting 2 Vehicle Speed 20 – 50 km/h**

Figure 5.8 shows noisy background compared to figure 5.7; due to vehicle speeds and other external light frequency interferences. The license plate is detectable because the distinct

rectangular shape is visible in the image. The noisy background and other object like license plate will make the detection difficult.

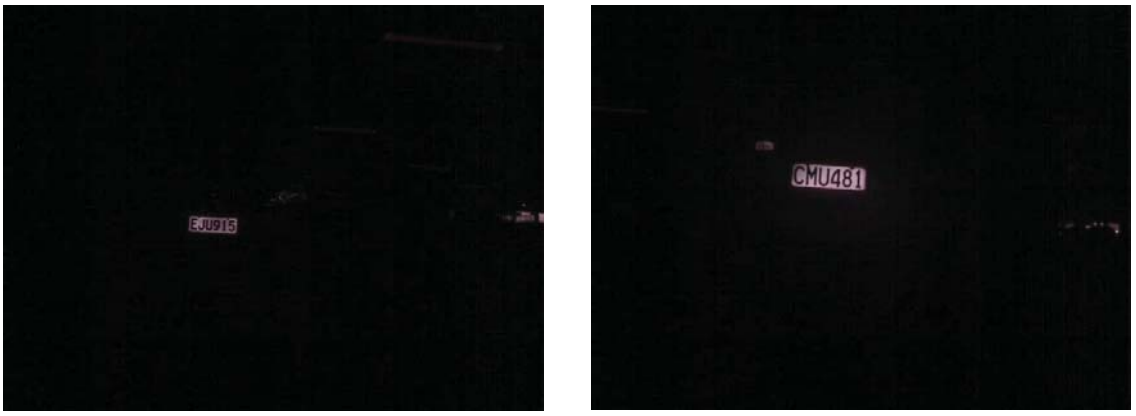



**Figure 5.9: Setting 3 Vehicle Speed 0 – 10km/h**

Figure 5.9 shows two clear sample images, this is possible because of little or no background noise. The surveillance vehicle is stand still and therefore the picture is in focus.

**Figure 5.10: Setting 4 Vehicle Speed 50 – 100km/h**

Figure 5.10 shows two sample images at distance as the vehicle is travelling at 50km/h or more. The image seen completely out of focus and it is impossible to make out the license plate location or the characters. Setting 4 is left out of this research study.

# 6 Results

## 6.1 Results Summary

To analyse and comprehend the research design, nine simulations have been proposed. In each simulation, the researcher attempted to locate the license plate from the image. Thus, each simulation analysed different areas of the software design, and aimed to optimise the settings in order to produce a high license plate recognition accuracy rate.

**Data Analysis**

Three image samples were tested on each simulation run, and samples named E1, E2 and E3. In each sample, there are 500 images.

| Environmental Samples | Summary |
|---|---|
| E1 | • Traffic junctions<br>• The speed: 0 km/h – 20 km/h |
| E2 | • Residential areas<br>• The speed: 20 km/h – 50 km/h |
| E3 | • Car parks<br>• The speed: 0 km/h – 10 km/h |

Table 6.1 Sample summary

| Results | Definition |
|---|---|
| Correct License Plate Detection | • If the license plate is found in the sample image, the counter is incremented. |
| License Plate Not Detected | • If there is presence of the license plate in the sample image and it is not detected, the counter is incremented.<br>(NB: The image includes distorted or partially visible |

| | |
|---|---|
| | plates, but does not include the license plates that are not recognisable to human eye.) |
| Incorrect Detection | • In each sample image, if the detection incorrectly marks the license plate, then the counter is incremented. If there are multiple incorrect detections in one sample image, the counter is only incremented by one. |

Table 6.2 Result definitions

The images in each environment sample are numbered from 0 to 499. Each image was iterated through a FOR loop, with two windows showing the license plate. The first window displayed the original image, and the second window displayed the image with marked locations of the license plate; if no plate was found, then a blank window could be seen. The observer of the test simulation manually recorded the result of each image onto a spreadsheet. The results were then tallied up into tables and bar graphs for each simulation, and shown as Alpha, Beta and Gamma.

## 6.2 Alpha Run

The Alpha run tested the ability of Canny edge detection to detect a license plate for different hysteresis threshold settings. The binary threshold filter was disabled for the purpose of testing only the Canny edge detection process. As a result, the different hysteresis settings were shown in Alpha 1, Alpha 2, Alpha 3 and Alpha 4.

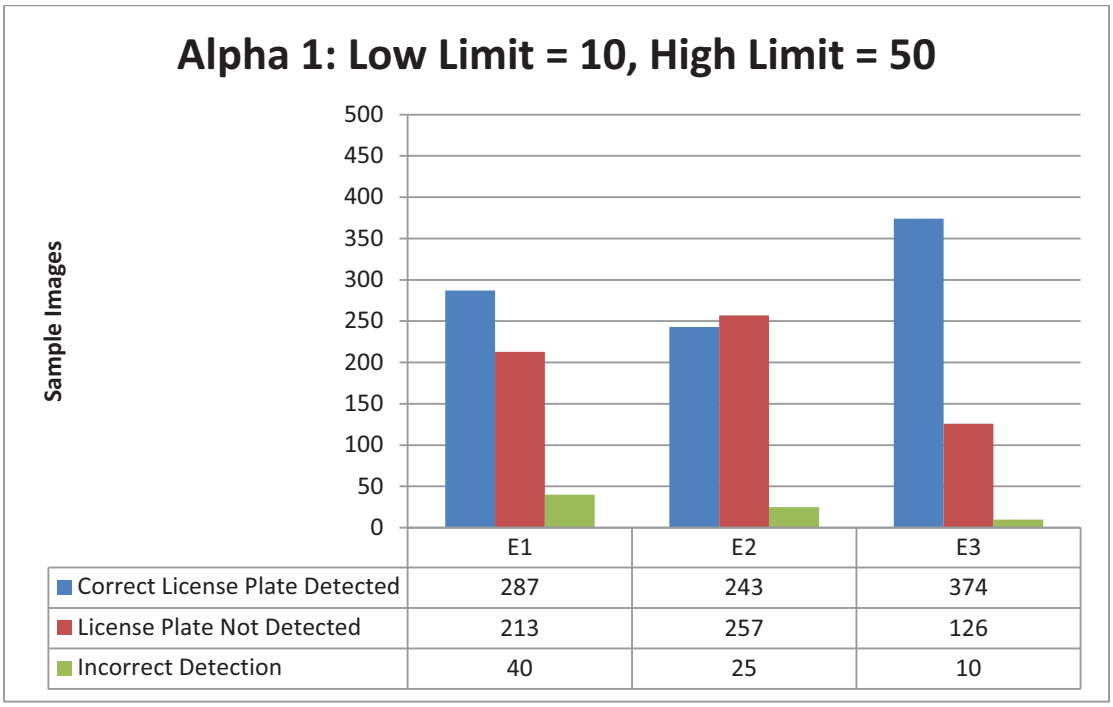



**Figure 6. 1: Alpha 1 results**

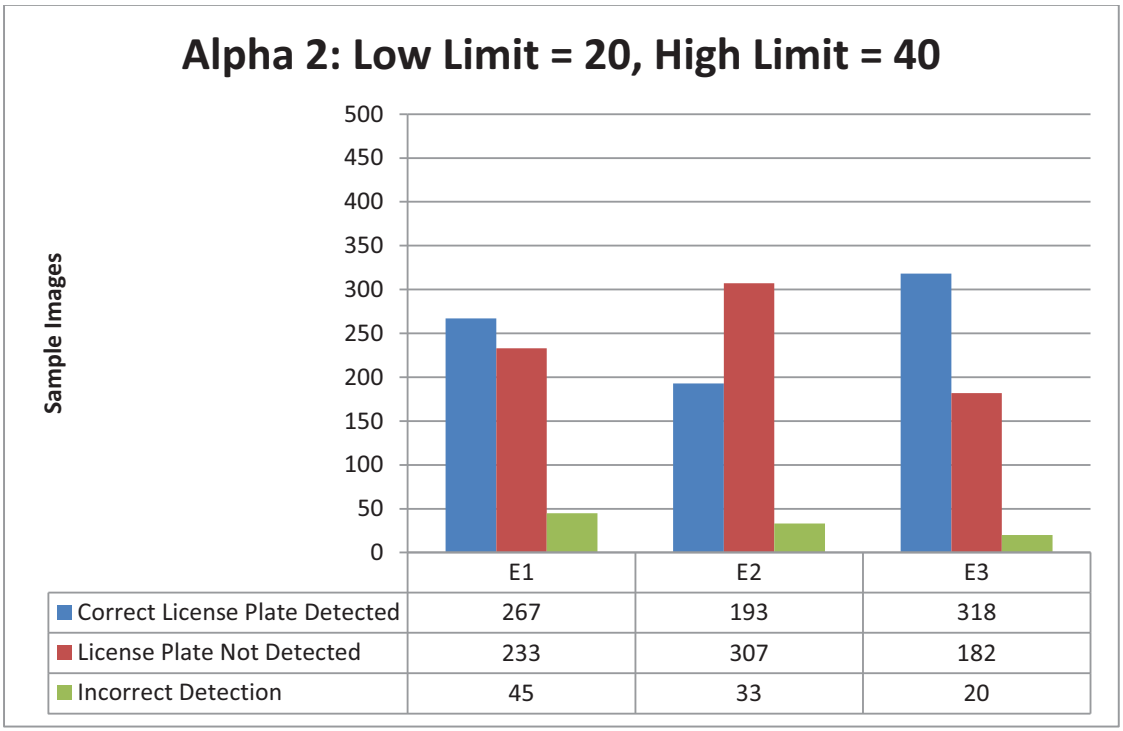



**Figure 6. 2: Alpha 2 results**

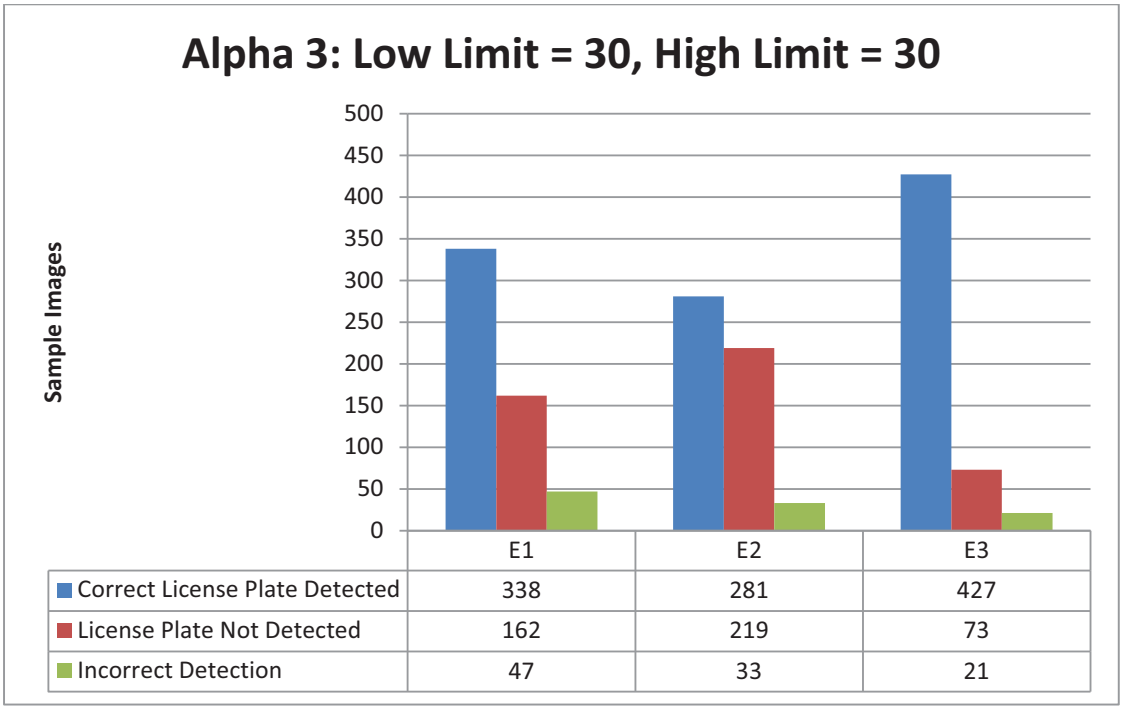



**Figure 6. 3: Alpha 3 results**

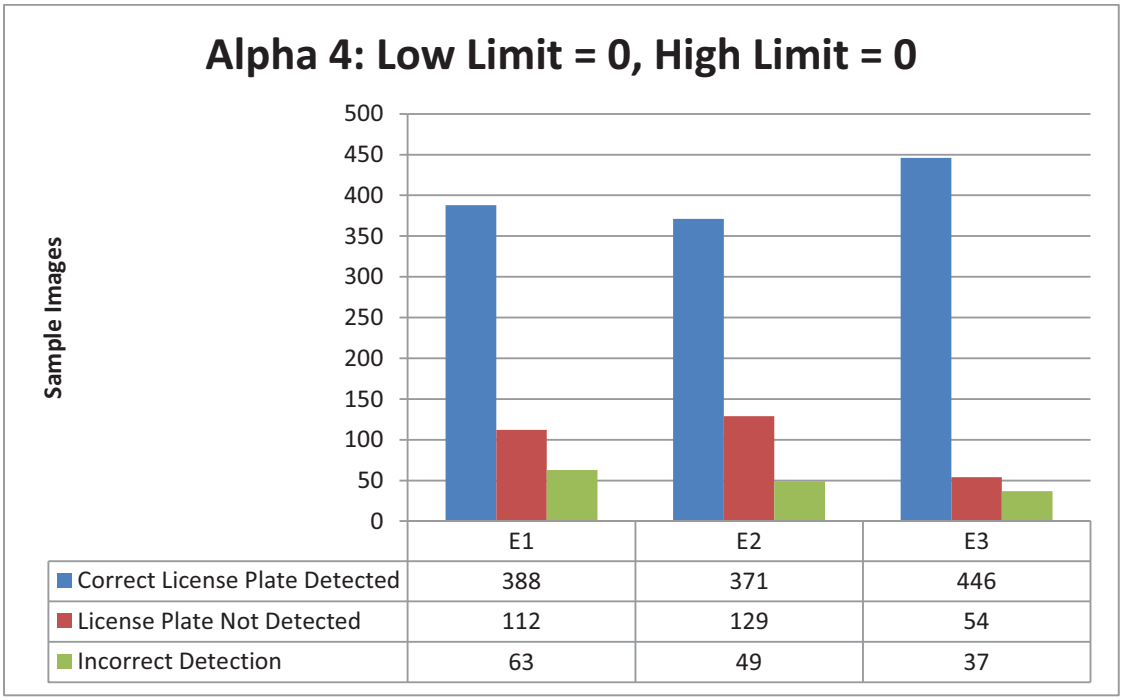



**Figure 6. 4: Alpha 4 results**

## 6.3 Beta Run

In the Beta simulation run, the Canny detection algorithm was disabled, and the binary threshold algorithm was activated and tested on different iteration levels, as revealed by Beta 1, Beta 2, Beta 3 and Beta 4.

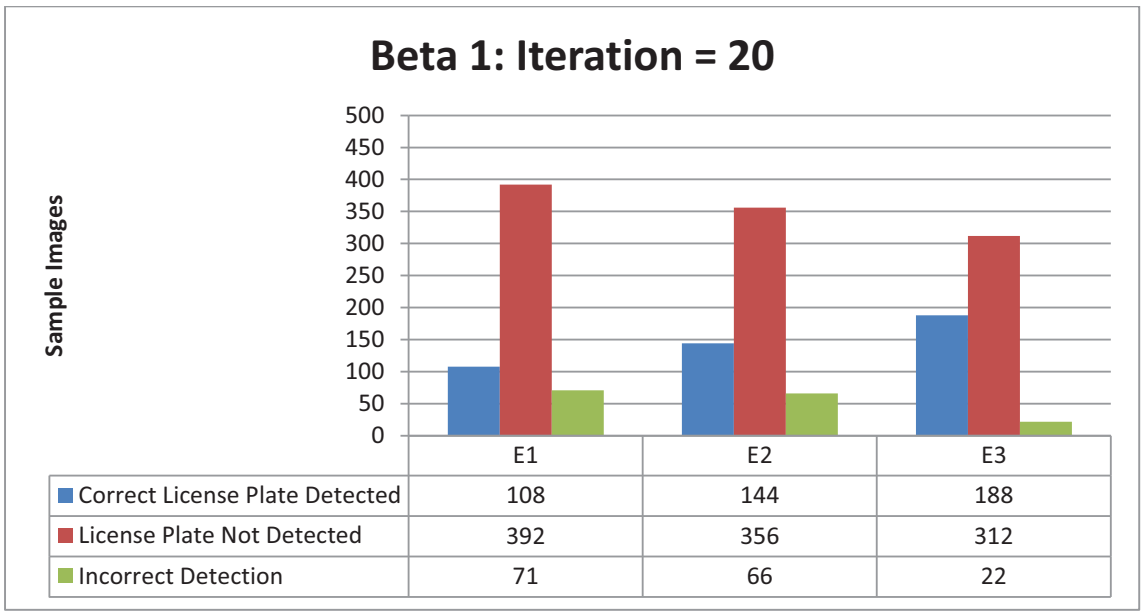
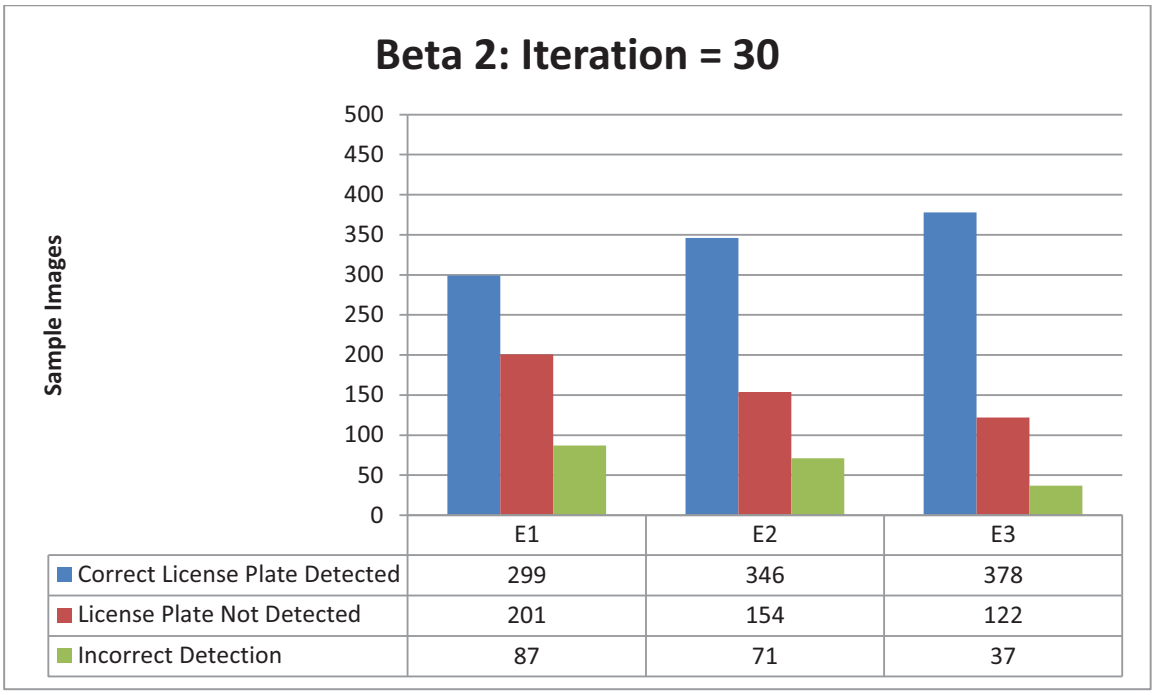
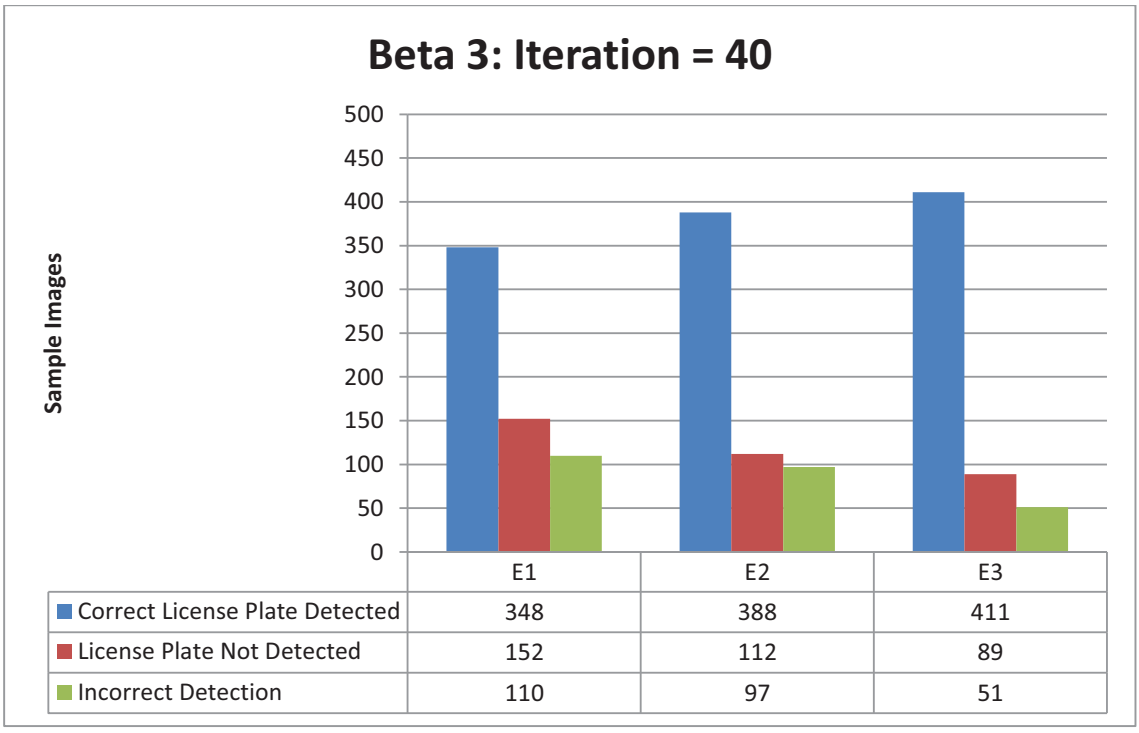



**Figure 6. 5: Beta 1 results**




| | E1 | E2 | E3 |
|---|---|---|---|
| Correct License Plate Detected | 299 | 346 | 378 |
| License Plate Not Detected | 201 | 154 | 122 |
| Incorrect Detection | 87 | 71 | 37 |

**Figure 6. 6: Beta 2 results**




| | E1 | E2 | E3 |
|---|---|---|---|
| Correct License Plate Detected | 348 | 388 | 411 |
| License Plate Not Detected | 152 | 112 | 89 |
| Incorrect Detection | 110 | 97 | 51 |

**Figure 6. 7: Beta 3 results**

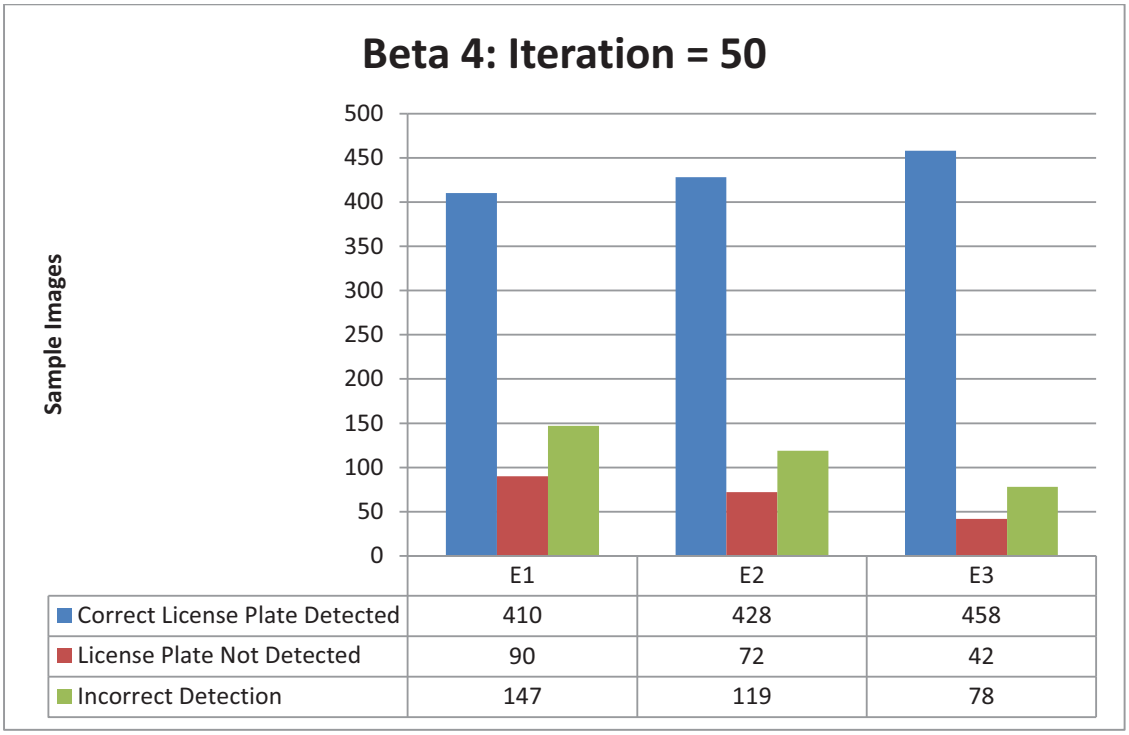



**Figure 6. 8: Beta 4 – results**

## 6.4 Gamma Run

Alpha and Beta simulations were completed separately for the Canny edge and binary threshold. The optimum performance of each algorithm was tested in Gamma simulation by combing both algorithms and using optimum values found in Alpha and Beta simulations.

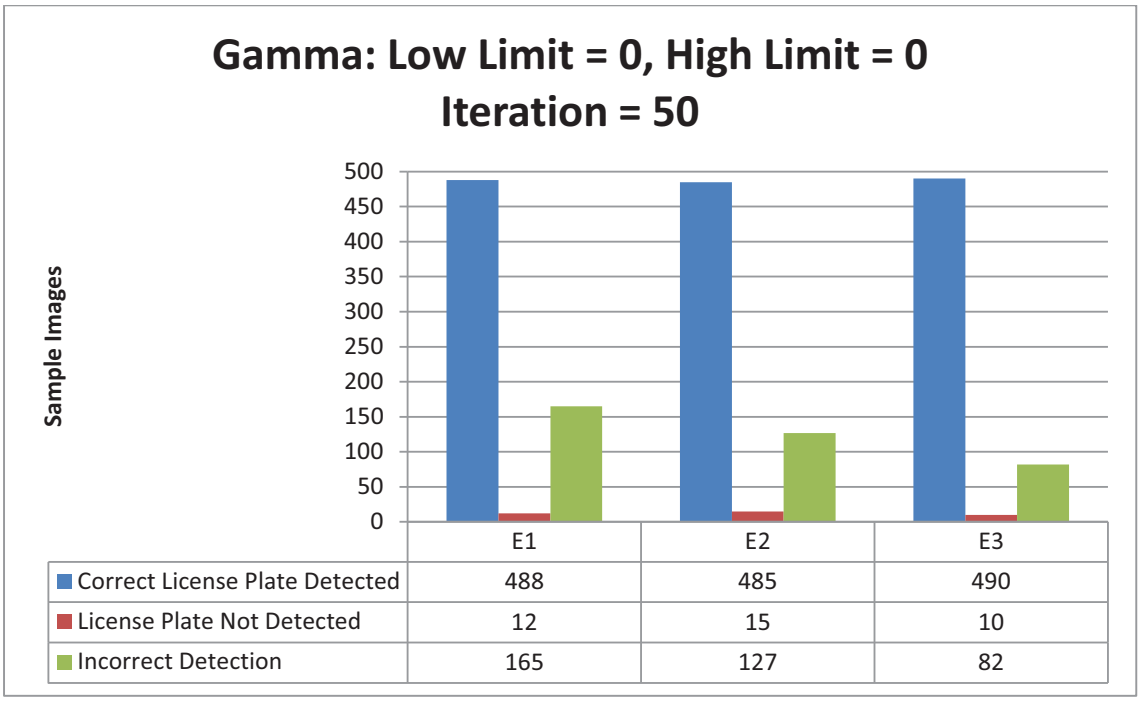



**Figure 6. 9: Gamma results**

# 7 Discussion and Conclusion

## 7.1 Discussion

The simulation runs were motivated by two edge detection techniques that were examined with different parameter value settings to find the optimum performance of the methodological design. Within the simulation, there were various variables that were defined as constants in the research experiment. The results were influenced by the setting of constant variables and acknowledged in the result of the simulation.

The constants variables are:

- The hardware setup, the infrared frequency, the camera plate distance variations, and the illumination spread.
- Noise filters: the image pyramid, down-sampling, up-sampling, and Gaussian blur settings.
- The contour: boundary detection using the chain code algorithm, and 8-way connectivity.
- The vertices extract: the Douglas-Pecuker algorithm, and the tolerance level.
- Contour filters: vertices > 3, the contour area > 1000 pixels, and area convexity.

The sample collections were completed in a period of several weeks. Each sample collection required the re-assembly of the camera and the infrared light source. The image quality and the setup apparatus required to stay constant throughout the experiment. However, in the setup, small variations were expected to occur in the configuration angle of the camera and the position of the infrared light source. Thus, after careful observation of the image quality, no unusual outer layers were found, it was acceptable to conclude that the sample quality was not compromised.

In the experimental design, the infrared frequency wavelength was set to 850nm and the infrared lens filters were set to 850nm. Both setting values remained constant throughout the experiment, and enabled the camera to capture illuminated pictures of the surroundings with inclusion of the license plate. In this mobile experiment, it was expected that the

distance between the camera and the license plate would be different for every image, and this distance variation was incorporated in the scope of the research settings. Due to the distance variations, illumination weakened as the distance increased. The illumination variations were expected but not measured in the research apparatus. The relationship between the distance and illumination spread is left to be investigated in a future study. The simulation reproduced real world settings by means of distance and illumination variations, and resulted in a variable image quality.

The noise filter functionality is essential for reducing the visibility of the noise. The down-sampling and up-sampling with the Gaussian blur dilutes the concentration of the illumination values dictated by the kernel size. Accordingly, the kernel size 7 was opted for in both sampling techniques. The kernel size spreads the illumination values in the image to the wider neighbourhood pixels and results to noise minimisation. A smaller kernel size would be less effective, and the relationship between the kernel size and plate detection was not tested due to the time requirements for completion of this research. However, the kernel size applied is a constant to all simulations.

Canny edge detection utilises the chain code algorithm to map the image object boundaries. The chain code algorithm requires a connectivity mode to define the relationship between pixel neighbourhood connectivity. The connectivity mode selected for this research design was 8-way connectivity. The 8-way connectivity model allows the pixel connection to be horizontal, vertical and diagonal and the existence of license plate edge boundaries can be found in any directions within 360 degrees.

The chain code map is simplified by the Douglas-Peucker algorithm which limits the number of edge points mapped to represent object boundary. The tolerance level set on the Douglas-Peucker algorithm controls boundary details. In this research, the tolerance level was found by calculating 8% of the chain code length. As the tolerance level changes depending on the size of the contour object, whereas the 8% ratio remains the same, in the mapping of the license plate in this study, 8% ratio of contour perimeter was found to be the optimum value. As the optimum setting in this case would be a loose term after

numerous runs of the setting, 8% was found to be the best setting to produce the optimum result. This setting should be further investigated to find the actual optimum value.

To simplify the mapping coordinates, Dogulas-Peucker minimised the details of the border by using the tolerance level to find critical vertices that defined the object shape. As the practical license plate has 4 vertices, the object detected in the image was defined to have four vertices, but due to image corruption, the plate could be deformed and may have three, four or five vertices. Another filter setting is the area taken up by the contour object, which is rejected if it is less than 1000 pixels.

The research simulation was completed successfully and meaningful data were collected to analyse the system performance. The performance of the nine simulation runs could have positive or negative results depending the on external settings of the research design. Due to the time limitation, no further analysis was done to understand the impact of the other configuration settings.

### 7.1.1 Simulation Run Alpha

To determine the performance of the Canny edge detection algorithm, four Alpha simulations were completed, and in each simulation the hysteresis levels were adjusted (Alpha 1: L=10, H=50; Alpha 2: L=20, H=40; Alpha 3: L=30, H=30; and Alpha 4: L=0, H=0). Alpha 4 preformed the highest correct license plate detection across all the three environments with E1= 388, E2 = 371 and E3 = 446, and equated to an average of 80.33% license plate detection rate. The inconsistency in results between the three samples demonstrates that the embedded noise levels of the sample sets were different, but E1 and E2 were similar, while E2 showed the poorest performance.

The E3 sample produced the highest performance across all Alpha simulations. The reason for this is that complexity of the image in E3 was greatly reduced because there was no external light source that produced infrared frequency, whereas in E1 and E2 external infrared frequency was accountable for unwanted image noise. Therefore, the best performance using the Canny algorithm was indoors and in the car park environments with

the license plate image clearly seen and with limited image noise. The distance range between the camera and the license plate was another factor for the advantage of E3 over E1 and E2 – the distance in E3 was approximately 2-4 meters.

The hysteresis levels control threshold values for edges need to be detected. The illumination value 0 represents black and 255 white pixels. Ideally, the illumination value zero represents black edges around the license plate; however this will never be the case in real world conditions. Thus, hysteresis allows range of illumination values to be included in the zero range, as explained in Section 4.3.2.

Therefore, the results of the Alpha simulation can be used to find the optimum hysteresis settings that are equally effective across the three environment samples. Alpha 1 shows E3 as the most favourable, and then the E1 and E2 sample sets. This demonstrates that the illumination values of the border line of the license plates are spread out, and are rather different for each E1, E2 and E3. Due to the random nature of the image quality, it is impossible to predict the illumination value spread. Accordingly, the Alpha 2 simulation was run to optimise hysteresis performance. However, Alpha 2 performed poorly compared with Alpha 1 across all the environmental samples, and is thus no longer of interest. In the Alpha 3 simulation run, the hysteresis levels configured differently from those in Alpha 1 and 2. In Alpha 3, the low and high hysteresis level was set to an equal value of 30. This setup is similar to the binary threshold, and if the illumination values are below 30, then they are considered as black or white pixels. Alpha 3 results are better than those of Alpha 1 and 2, but average a 69.73% correct accuracy rate. Alpha 3 results are also inconsistent across the three samples. After observation of Alpha 1, 2, and 3, it was decided that fixed hysteresis values could never be optimised across three environment samples. An alternative to the hysteresis threshold values was to set both low and high sets to zero. Alpha 4 produced an accuracy rate of 80.33%, which is an increase of 10.6% compared with Alpha 3. With no hysteresis levels set, all the edge points can be detected including the higher illumination value edges. The Alpha 4 setting also increased incorrect detection to 9.93%, whereas other Alpha simulations produced 6.73% (A3), 6.53% (A2) and 5% (A1). Due to the time limitation, it is left for future research to investigate the process of reducing the incorrect detection rate to a minimum.

Alpha 4 is of interest to this research. To investigate the missed license plate detection, one example license plate is shown in Figure 7.1 (the original image). The plate detection failure is due to the formation of the plate boundary which got broken after dilation, and therefore, the chain code algorithm failed to complete the closed boundary. By taking one sample image, the original image, the license plate with character X and the boundary edge is established. Because of this contact, as Figure 7.1 shows, Alpha 1, 2, 3 and 4 failed to close the outer boundary, and thus the chain code algorithm followed the edge leading into the characters of the plate, which is incorrect for the detection purpose. Under these circumstances, Canny edge is not an ideal edge detection technique when the edge boundary is broken by internal or external boundary objects.

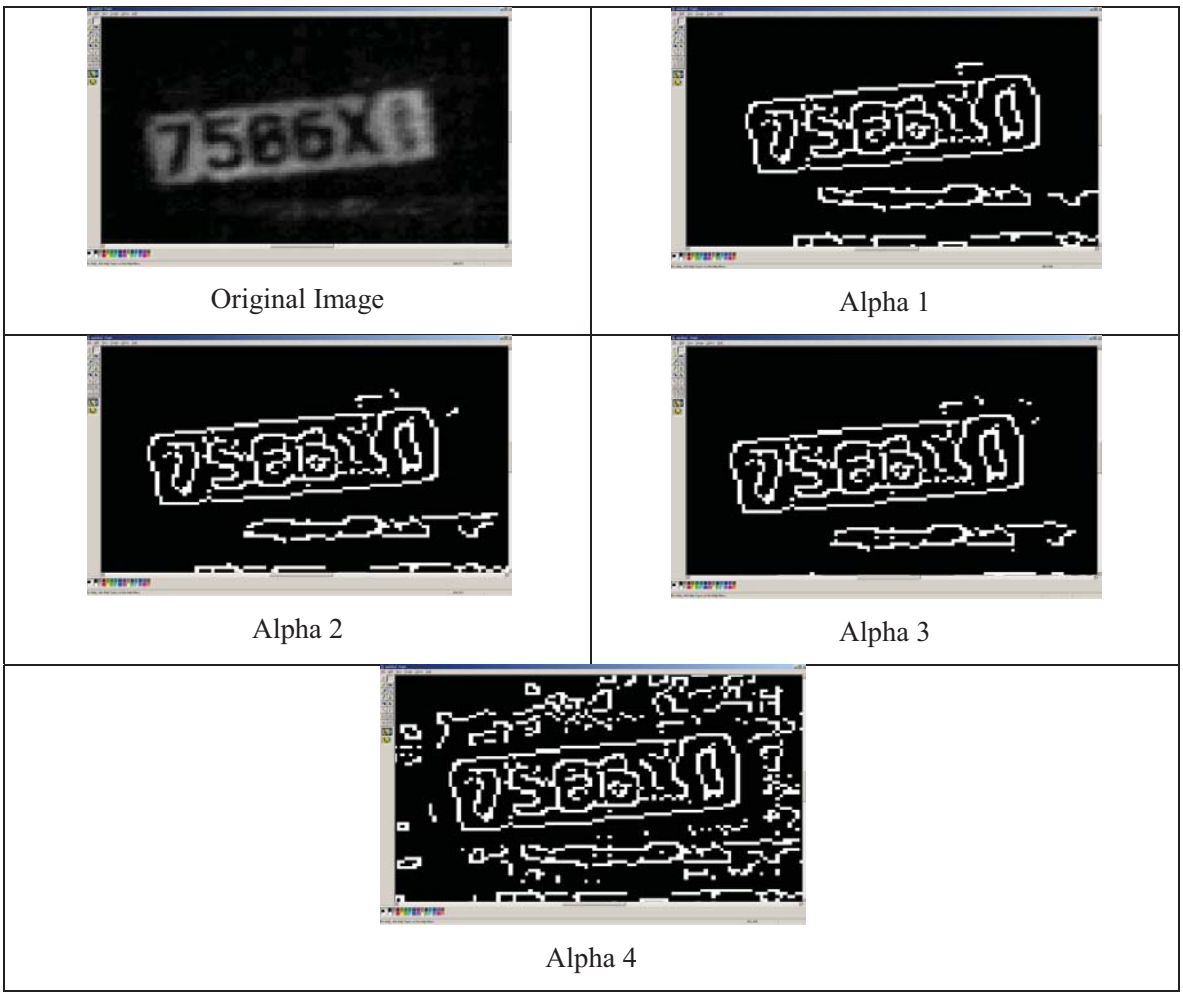



**Figure 7. 1: Alpha simulation output image after Canny edge detection**

Further observation demonstrates similarities (Figure 7.1), i.e. different threshold levels in each Alpha simulation produced similar plate edge boundaries. However, at a closer look of the plate surroundings made up of different edges sums, in Alpha 4 all the edge objects are shown.

Alpha 4 reveals the vast improvement in plate detection. As the low limit and the high limit were set to zero, the hysteresis did not filter weak edges nor filtered out strong edges. All the edges in the image were located, but the Alpha 4 incorrect detection increased significantly compared with Alpha 1, 2 and 3. This result was expected and that other object like license plate boundaries would be present in the image which is incorrectly detected.

To summarise, Alpha 4 produced the highest accurate rate, but also the highest incorrect detection rate of 9.93%, of all the Alpha simulations. Alpha 1, 2 and 3 fell short by a 10% accuracy rate produced by Alpha 4. However, Alpha 4 correct detection rate of 80.33% was still below the expected accuracy rate of 90% or more. The Beta and Gamma simulation runs aimed to improve this accuracy.

### 7.1.2 Simulation Run Beta

The Beta simulation run was set to test the performance of the binary threshold effectiveness in the licence plate detection, and Canny edge detection was disabled for this purpose. As the binary threshold operates on iteration levels, four Beta simulations were completed on different iteration levels. The external setting of the simulation remained the same as Alpha simulation run.

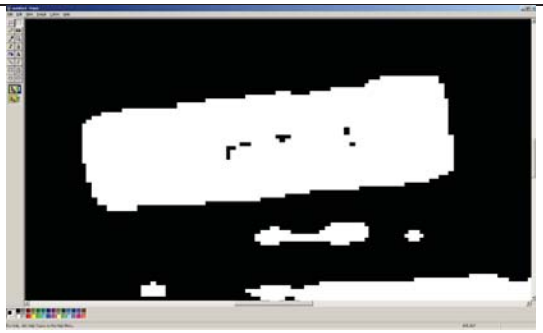
As the binary threshold can be a substitute for Canny edge detection, the edge detection process was done differently by block filling objects with the illumination value 255 using different binary threshold levels, as explained in Section 4.3.4.

Examination of Beta 1 reveals that, as the iteration level was set to 20, 20 iterations completed the image which produced the weakest average detection rate of 29.33% across

the three environment settings. However, the trend appears to be in favour of E3, which produced a high accuracy rate compared with E1 and E2, across all the Beta simulation runs, and this trend is similar to the Alpha simulations, followed by Beta 2 accuracy of 68.2%, Beta 3 of 76.46%, and Beta 4 of 86.4% across the three environmental settings.

As noted above, the binary threshold is an alternative object detection method to Canny edge detection. Figure 7.2 demonstrates the binary threshold with the original image shown in Figure 7.1.

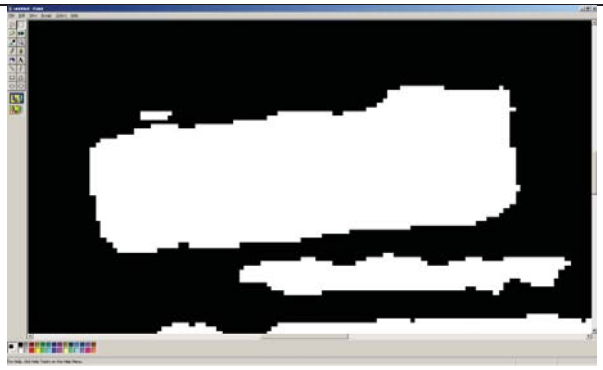
Original Image



Beta 1

Beta 2

Beta 3



Beta 4

**Figure 7. 2: Beta simulation output image after binary threshold**

Examination of Figure 7.2 reveals that, at each iteration level, the detail of the image object blob also improved. Nevertheless, it can be seen that Beta 1 shows as a significantly good representation of the original license plate model as that shown by Beta 4. The original image was measured to be a good sample image, but the earlier Alpha Canny detection failed to detect this license plate due to the boundary breakage.

This was not a problem for Beta simulation because the binary threshold operated as dilation and filled the object space with the illumination value 255, which allowed the boundary edges clearly to be defined and mapped by the chain code algorithm.

The binary threshold had the maximum iteration level of 254, but it was not practical to set iteration level to 254 because it needed to process 254 images separately by the chain code algorithm. Canny edge detection operated on gradient change of pixel values, and the boundary of the license plate needed to be intact for successful detection using the contour chain code algorithm. As the binary threshold depended on the object separation, the license plate needed to be cleansed from other objects and from noise patches. If the presence of noise or other objects were in the similar illumination values as the license plate and in close proximity, then the blob also included that noise or those objects as part of the single object and failed to retain the rectangular shape required for Douglas – Peucker to differentiate from other objects in the image.

Beta 4 shows the optimum performance (86.4%) out of the Beta simulation runs. Further examination reveals that Beta 4 produced the highest incorrect detection rate of 22.93%. This could be expected at a high iteration level which allows the opportunity for other objects to be considered as the license plate. However, in the Alpha simulation, incorrect detection was 9.93%, and Beta 4 average was 22.93%, which is 10% more than that of Alpha and Beta simulation.

Beta simulation provided a suitable alternative option of the Alpha simulation. However, both simulations showed weaknesses in different areas. The binary threshold showed stability when Canny edge detection was weak in the border edge breakage. The binary threshold was also vulnerable to noise and near objects, while Canny edge detection was less vulnerable to near objects and noise.

The binary threshold by itself falls short of 90% accuracy level. To improve this research model another simulation was completed (Gamma), in which the optimum performance of Alpha and Beta was combined to produce Gamma simulation (Section 7.1.3).

### 7.1.3 Simulation Run Gamma

Gamma simulation combined Canny edge detection and the binary threshold to optimise plate detection. Observation of the Alpha and Beta simulation reveals that both edge techniques have strengths and weaknesses in different areas of the image quality. The results show improvement in detection rates across E1, E2 and E3, with respect to individual simulation runs from Alpha and Beta. The integration of both algorithms was required to improve the plate detection, but as expected, incorrect detection was significantly high. The correct detection rate of 97.53% with incorrect detection of 24.93% was the result of Gamma simulation.

The research was successful in achieving a 97.53% accuracy rate, and this accuracy rate is well within the industry standard. However, the incorrect detection rate also increased to 24.93%. This increase in the incorrect detection is due to other license plate similar objects which appeared in the image, and Canny edge detection with the binary threshold did not

filter out the weaker objects in case of not detecting the noisy license plate. Future research in this field can investigate this problem as to reduce the incorrect detection rate without reducing the correct detection rate.

### 7.1.4 Discussion Conclusion

The research methodology was set out to test two unique edge detection techniques (Canny edge detection and the binary threshold). Both techniques were individually tested to find the optimum values, and a total of nine simulation runs were completed. The Alpha simulation tested Canny edge detection by varying hysteresis levels, while the Beta simulation tested the binary threshold levels by varying iteration levels. The binary threshold achieved license plate accuracy (86.4%) higher than Canny edge detection (80.33%). However, it was recognised that both simulations had weaknesses and strengths, and that both could benefit from each other's algorithm design. Accordingly, a third simulation, Gamma, was conducted to combine both algorithms. The Gamma simulation achieved 97.53% accuracy and the research model was deemed a success. However, the incorrect detection rate was found to be 24.93%, and this area requires further refining. Other structure parameter values were set constant during the simulation runs, and the set values were an educated guess. As this, however, can influence the final results, it is not possible to simulate all parameter values to optimise the system performance.

## 7.2 Recommendations & Conclusion

### 7.2.1 Further Model Development

In order to continue improvement of the current model, several areas of the license plate detection model can be refined. The existing system is a prototype model which ensures that the research methodological design is simulated, and is able to prove its theoretical

findings. However, further development is necessary for this design to be considered as an industry approved product.

In order to complete the material setup, a standard webcam with CMOS sensor chip was selected, but the research did not include investigation of the camera capabilities, and also chose a non-industry standard infrared light source. The research intension was more focused on the software design, and the time was too limited for investigation of the material design. A CCD camera would be a better option if this research was to be completed again. Because of practicality, this research model used broadly available webcams which meet the requirements at the time of the research. An enhanced camera selection would have improved the quality of the image significantly, and consequently the detection process of the license plate. The infrared light source underperformed in the simulation runs. The capability of the infrared source light was limited to the hardware specifications. The applied infrared source appeared to be underpowered, and resulted in limited distance coverage, thus reducing the ability to investigate a vehicle speed environment greater than 50km/h. An industrial infrared source would have improved the physicality of the model to a great extent.

The software architecture allowed one image per 50ms to be processed. The load on the software unit was intensive, and a further image analysis would require additional CPU power. One such analysis is a character recognition system (converting the image into the ASCII code). Also, the CPU utilisation can be improved by reducing the load of the image input rate. To achieve this, a physical sensor pointed outward from surveillance system would trigger the camera when a vehicle or object is in surveillance range; otherwise the system would stay idle. Another advantage of the sensor is expected to be a drop in the incorrect detection rate.

As an expected outcome for any sample image, there is one license plate to be detected. The current model maps the contours of all possible objects in an iterative nest. The required contour is a single object that represents the license plate. A refined model would check each mapped contour after it is found; thus, if the contour is a license plate, then no further contour is required. Character recognition based check is suitable under these circumstances, and the incorrect detection rate is expected to drop.

Static variable settings are found throughout the research model, and the image quality varies in-between images. To consider optimum settings, the image quality needs be measured, and suitable license plate detection algorithms need to be applied. The research was not allowed the time for analysing the image quality, and was therefore adapted to a static variable choice throughout the system design. A future design should refrain from using static variables which limit the system ability to stand against different environment conditions such as wet weather, fog, snow, and windy and sunny day conditions.

### 7.2.2 Conclusion

This thesis set out to examine a mobile license plate detection system via simulation, with intention of operating in limited light conditions to produce accurate detection of the license plate detected in the image.

Simulations were devised to investigate two edge detection techniques (Canny edge detection and the binary threshold) in three different operating environments. Two individual edge detections were simulated to establish the ability to detect the license plate with different parameter settings in order to find the optimum parameters. In the simulations at a later stage, the two edge detections were combined to advance the license plate detection capability.

As a result of these simulations, some individual edge detections were unsuccessful in reaching the near optimum accuracy detection rate of 100% set to be achieved in this thesis. However, the simulation runs in Alpha 4 and Beta 4 produced optimised results which allowed development of a model which combines both edge detection processes in order to increase the accuracy rate up to 97.53% (the Gamma simulation) averaging across the three different environmental settings.

This research successfully developed a mobile surveillance system to be operated in limited light conditions for which the previous research had not shown evidence or ability to detect a license plate at the accuracy level of 97.53%. As the research performance is restricted to

limited light conditions, future research can refine this research findings for a system operable 24 hours a day.

## References


Aguado, A. & Nixon, M. (2002). *Feature Extraction & Image Processing.* (2nd Ed.). London: Academic Press

Aitchison, A. (2008). *Beginning Spatial with SQL Server 2008.* New York: Apress

Anagnostopoulos, C.N.E., Anagnostopoulos, I.E., Psoroulas, I.D. & Loumous, V. (2008). License Plate Recognition from Still Image and Video Sequence: A Survey. *IEEE Transactions on Intelligent Transportation Systems 1(3)*, 377-391

Arth, C., Limberger, F. & Bischof. (2007). Real-time license plate recognition on an embedded DSP-platform. *Proceeding IEEE conference CVPR* 1-8

Chang, S.L., Chen, L.S., Chung, Y.C. & Chen, S.W. (2004). Automatic License Plate Recognition. *IEEE Transactions on Intelligent Transportation System 5(1),* 42-53

Code Source (2005). *Dilation Operation on Binary & Graylevel images in C#.* Retrieved July 2, 2010 from http://www.codersource.net/microsoft-net/c-image-processing/dilation-in-image-processing-using-c.aspx

Cui, D., Gu, D., Cai, H. & Sun, J. (2009). License Plate Detection Algorithm Based on Gentle AdaBoost Algorithm with a Cascade Structure. *Proceedings of the 2009 IEEE, International Conference on Robotics and Biometrics*

Duan, D.T., Duc, D.A. & Du, T.L.H. (2004). Combining Hough Transform and Contour Algorithm for Detecting Vehicles' License-Plates. *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing* 747 – 750

ELSAG − North America Law Enforcement Systems (2010). *The most advanced automatic license plate recognition technology available.* Retrieved June 12, 2010, from www.elsag.com/about.htm

Faradji, F., Zezaie, A.H. & Ziaratban, M. (2007). A Morphological Based License Plate Location. *International Conference on Image Processing*

Gonzalez, R.C. & Woods, R.E. (2002). *Digital Image Processing*. (2nd Ed.). New Jersey: Prentice Hall.

Green, B. (2002). *Canny Edge Detection Tutorial*. Retrieved June 15, 2010 from http://www.pages.drexel.edu/~weg22/can_tut.html

Guo, J.M. & Liu, Y.F. (2008). Licence Plate Localisation and Character Segmentation With Feedback Self-Learning and Hybrid Binarization Techniques. *IEEE Transaction on Vehicle Technology 57(3),* 1417-1424

Ho, W.T., Lim, H.W. & Tay, Y.H. (2009). Two – stage License Plate Detection using Gentle Adaboost and SIFT – SVM. *Asian Conference on Intelligent Information and Database Systems*

Huansheng, S. & Guoqiang, W. (2005) The High Performance Car License Plate Recognition System and its Core Techniques. *Proceedings of the 2005 IEEE*

Hung, K.M. & Hsieh, C.T. (2010). A Real-Time Mobile Vehicle License Plate Detection and Recognition. *Tamkang Journal of Science and Engineering (13) 4,* 433-442

Mello, C.A.B. & Costa, D.C. (2009). A complete system for Vehicle License Plate Recognition. *IEEE Transaction on Urban Traffic Controls Vol (1)*, 1-4

Naito, T., Tsukada, T.,Yamada, K., Kozuka, Kazuhiro, K. & Yamamoto, S. (2000). Robust License Plate Recognition Method for Passing Vehicle Under Outside Environment. *IEEE Transactions on Vehicular Technology 49(6),* 2301-2319

New Zealand Police (2010). *New Zealand Crime Statistics*. Retrieved August 5, 2010 from http://www.police.govt.nz/sites/default/files/services/statistics/00-national-09-10-official-stats_asoc.pdf

Qui, Y., Sun, M. & Zhou, W. (2009). License Plate Extract Based on Vertical Edge Detection and Mathematical Morphology. *International Conference on Image Processing. 1(3)*

Rattanathammawat, P. & Chlidabhongse, T.H. (2004). A Car Plate Detector using Edge Information. *International Symposium on Communications and Information Technologies*

Sony Corporation (2000) *XCD-SX900, XCD-X700 User's Guide (V 1.0)*. [Department of Computer Science, University of North Carolina] Retrieved March 8, 2009 from http://www.cs.unc.edu/Research/stc/FAQs/Cameras_Lenses/Sony/XCD700-900UserGuide.pdf

Starr, C. (2005). *Biology: Concepts and Applications*. (7th Ed.). CA, USA: Thomson Brooks & Cole

Sun, J., Cui, D., Gu, D., Cai, H. & Liu, G. (2009). Empirical Analysis of AdaBoost Algorithm on License Plate Detection. *Proceedings of the 2009 IEEE International Conference on Mechatronics and Automation*

Titus, H. (2001). Imaging Sensors That Capture Your Attention. *Sensors Magazine.* Retrieved March 20, 2009 from http://www.siliconimaging.com/ARTICLES/sensors.htm

Wang, M. & Lai, C.H. (2009). *A concise Introduction to Image Processing using C++.* (Vol. 5). London: CRC Press.

Wang, S.Z. & Lee, H.J. (2007). A cascade framework for a real-time statistical plate recognition system. *IEEE Transaction Information Forensics Security 2(2),* 267 – 282

Weeks, A.R. & Myler, H.R. (1993). *Computer Imaging Recipes in C.* New Jersey: Prentice Hall.

Wikipedia (2009). *Light-emitting diode.* Retrieved February 18, 2009 from http://en.wikipedia.org/wiki/LED_diode

Wu, B.F., Lin, S.P. & Chiu, C.C. (2007). Extracting characters from real vehicle license plates out of doors. *Journal of Intuition of Engineering and Technology Computer Vision 1(1),* 2 – 10

Zheng, D., Zhao, Y. & Wang, Y. (2005). An efficient method of license plate location. *International Conference on Image Processing 26(15),* 2431 – 2438.