

<http://researchcommons.waikato.ac.nz/>

## **Research Commons at the University of Waikato**

### **Copyright Statement:**

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

The thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of the thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from the thesis.

# **Real Time Vehicle License Plate Recognition on Mobile Devices**

This thesis is submitted in partial fulfillment  
of the requirements for the  
Degree

of  
**Master of Science in Computer Science**

at  
**The University of Waikato**

by  
**Mohammed Ali Alshahrani**



THE UNIVERSITY OF  
**WAIKATO**  
*Te Whare Wānanga o Waikato*

2013

# **Abstract**

Automatic license plate recognition is useful in many contexts such as parking control, law enforcement and vehicle background checking. The high cost and low portability of commercial systems makes them inaccessible to the majority of end users. However, current mobile devices now have processors and cameras that make image processing and recognition applications feasible on them. This thesis investigates high accuracy real-time license plate recognition on a smartphone, taking into account device limitations. It first explores how, using the minimal image processing and simple configurable heuristics based on plate geometry, license plates and their characters can be detected in an image. Then, using minimal training data, it shows that a character recognition package can achieve high levels of accuracy. This approach accurately recognized 99 percent of plates appearing in a test set of videos of vehicles with New Zealand license plates.

# **Acknowledgment**

My grateful and sincere thanks goes to my supervisor Associate Professor Steve Jones for his help and guidance during the practical work and the writing and for providing me with the application framework to start the implementation with. I really appreciate his help, guidance, valuable feedback and time throughout the time of this study.

My hearty thanks filled with gratitude to my parents for their love, kindness, care and support. Also my deepest thank to my siblings for their support and encouragement. To all of you my parents, brothers and sisters I am so sorry for missing many occasions and being away when you needed me.

I would like to take the opportunity to express my gratefully thanks to my lovely wife (Zainab) who have been besides me all the time.

# Table of Contents

Abstract .....	ii
Acknowledgment .....	iii
Table of Contents .....	iv
List of Figures .....	vi
List of Tables.....	viii
1 Chapter One: Introduction.....	1
1.1 Objectives.....	3
1.2 Research question.....	3
1.3 Approaches.....	4
1.4 Thesis structure .....	6
2 Chapter Two: Related Work .....	7
2.1 Historical view .....	7
2.2 Pre-processing .....	8
2.3 Features analysis .....	8
2.4 Finding plate region .....	9
2.5 Character segmentation.....	9
2.6 Character recognition.....	10
2.7 Accuracy of related work.....	11
2.8 Summary .....	12
3 Chapter Three: Plate and Character Detection.....	13
3.1 Implementation .....	13
3.2 Plate and character detection overview.....	13
3.3 Pre-processing .....	14
3.4 License Plate Detection.....	15
3.4.1 Find contours.....	16
3.4.2 Width and height.....	17
3.4.3 Area ratio.....	18
3.4.4 Aspect ratio .....	19
3.4.5 Rotation.....	20
3.4.6 Extract and store the estimate plate.....	21
3.4.7 Rejected Alternative solution in plate detection .....	21
3.5 Analysing Region Content: .....	23
3.5.1 Adaptive threshold within closing filtering.....	23
3.5.2 Rotate the image geometrically.....	25

3.5.3	Cleaning the plate border .....	26
3.5.4	Image processing prior to finding the character contours .....	27
3.5.5	Find and store characters contour .....	28
3.5.6	Reducing the number of non-character contours .....	28
3.5.7	Deciding whether the estimated rectangle is a plate .....	31
3.6	Alternative Procedure When No Plate Found .....	32
3.6.1	Find the plate in part (ROI) of the frame .....	33
3.6.2	Find the plate using the previous location coordinates .....	33
3.7	Plate Recognition .....	34
3.7.1	Show a copy of the original and the clean plate .....	34
3.7.2	Create a cleaner plate .....	35
3.7.3	Tesseract OCR .....	38
4	Chapter Four: Plate Character Recognition .....	41
4.1	Introduction .....	41
4.2	Choosing Tesseract .....	41
4.3	Training process .....	42
4.3.1	Installation .....	42
4.3.2	Create the training document .....	42
4.3.3	Train Tesseract .....	42
4.4	Training data and results .....	44
4.4.1	Standard English trained data .....	47
4.4.2	Plate font A-Z and 0-9 .....	50
4.4.3	Plate font + zero with an extended slash .....	52
4.4.4	Real plate characters and a zero with an extended slash .....	54
4.4.5	Multiple instances of the real plate characters .....	56
4.5	Best-trained data .....	58
5	Chapter Five: Evaluation and Experimental Results .....	60
5.1	Training and tested data .....	60
5.2	Plate detection results .....	61
5.3	Have the plate characters been recognized? .....	65
5.4	Number of frames for correct character recognition .....	69
5.5	Summary .....	70
6	Chapter Six: Conclusion .....	71
6.1	Strengths .....	72
6.2	Future work .....	73
7	References .....	75

# List of Figures

Figure 1.1-1 LPR unit attached to the police car roof.....	1
Figure 1.3-1 examples of New Zealand vehicle license plate.....	5
Figure 1.4-1 General license plate recognition techniques phases .....	7
Figure 3.2-1 overview of plate recognition process.....	14
Figure 3.3-1 using OpenCV Canny function to detect the edges.....	15
Figure 3.4-1 steps to detect the plate.....	16
Figure 3.4-2 Example of all the detected contours at the first step.....	17
Figure 3.4-3 rejected rectangles (width greater than height) in green colour .....	18
Figure 3.4-4 rejected rectangles in purple (large or small area ratio) .....	19
Figure 3.4-5 blue rectangles rejected because of the aspect ratio are not from 25 to 40 percent .....	20
Figure 3.4-6 yellow rectangles rejected because of the skew angle .....	21
Figure 3.4-7 variance identified the plate correctly .....	22
Figure 3.4-8 highest variance misclassified headlight as plate .....	22
Figure 3.5-1 steps to clean the estimated plate .....	23
Figure 3.5-2 some examples before and after filtering .....	24
Figure 3.5-3 the plates before and after rotation .....	25
Figure 3.5-4 examples before and after cleaning the plate border .....	27
Figure 3.5-5 plates with cleaned border, plates with colour swapped and clean plates. ....	28
Figure 3.5-6 eliminating small noise and avoiding eliminating another contour by mistake. ....	30
Figure 3.5-7 eliminating another contour accidentally because of the overlapping. ....	30
Figure 3.5-8 contour rectangle's width is greater than or equal to 50 percent of the plate rectangle .....	31
Figure 3.7-1 copies of the original and clean plate at the top left corner.....	35
Figure 3.7-2 Scaled cleaner plate at the bottom left.....	38
Figure 3.7-3 overall process diagram.....	40
Figure 4.4-1 mis-detected plate.....	45
Figure 4.4-2 two characters deleted during filtering.....	46
Figure 4.4-3 character E deleted during the border cleaning .....	46
Figure 4.4-4 Character W recognized as H .....	49
Figure 4.4-5 Q recognized as EJ .....	50
Figure 4.4-6 Licenz font A-Z and 0-9 .....	50
Figure 4.4-7 character W recognized correctly .....	52

Figure 4.4-8 Examples of: slashed zero at the left & extended slashed zero at the right .....	52
Figure 4.4-9 character 0 was recognized correctly .....	54
Figure 4.4-10 E was recognized as IZ.....	56
Figure 5.1-1 Plate in the rear of the vehicle recognized .....	61
Figure 5.1-2 Plate in the front of the vehicle recognized (the plate not in the center).....	61
Figure 5.3-1 Top right corner of the plate is damaged.....	67
Figure 6.2-1 diagram of how the car will be captured from the side .....	74

## List of Tables

Table 2.7-1 related work accuracy .....	11
Table 3.7-1 Character frequency over multiple frames .....	39
Table 4.4-1 Standard English trained data results.....	48
Table 4.4-2 plate font A-Z and 0-9 results .....	51
Table 4.4-3 plate font + zero with an extended slash.....	53
Table 4.4-4 real plate characters (one instance of each) + zero with an extended slash trained data results.....	55
Table 4.4-5 multiple instance of real plate characters trained-data test results ....	57
Table 4.5-1 the total and the percentage of the for each test data .....	59
Table 5.2-1 Contingency table shows the detection results without using previous plate location .....	62
Table 5.2-2 Contingency table shows the detection results with using the previous plate location .....	63
Table 5.2-3 Detection results for each video .....	64
Table 5.3-1 Character recognition.....	68
Table 5.3-2 Characters recognized correctly using the frequency in relation to the number of frames with detected plates.....	68
Table 5.3-3 Character recognition comparison .....	69
Table 5.4-1 First frame where the plate characters were recognized correctly.....	70

# 1 Chapter One: Introduction

This thesis is in the areas of image processing and computer vision. It investigates techniques for automatic vehicle license plate recognition on mobile devices.

Commercial license plate recognition systems are available, but their high cost and low portability makes them inaccessible to the majority of end users. Examples of such systems are ANPR (Automatic Number Plate Recognition) installed in police vehicles, traffic congestion charges and management, and parking access control. For instance, in London congestion charge the ANPR system is used to detect and recognize the car license plate in order to make the charge (Leape, 2006). Gordon and Wolf (2007) stated that license plate recognition is used by the police to help to fight against some crimes, like car theft. Figure 1.1-1 shows a police car with license plate recognition camera attached to the roof.



Figure 1.1-1 LPR unit attached to the police car roof<sup>1</sup>

---

<sup>1</sup> Gordon, A., & Wolf, R. (2007). License Plate Recognition Technology: Innovation in Law Enforcement Use. *FBI Law Enforcement Bulletin*, 76(3), 8-13.

However, license plate recognition has numerous applications on mobile consumer devices, and vehicle license plate recognition can be used in several situations: car plate monitoring, parking control and vehicle background checking. For example, in a small car park the low cost mobile phone system could be in a fixed place at the entrance to recognize the car when it entered and keep a record of the car license plate and the entry and exit times. The system will then compute the amount of time the car spent and the price. For parking control, this system will be beneficial for the wardens they in need to keep a record for the car license plate and the time, this mobile system can do this and it can keep a record of the location using the GPS locations. Vehicle background checking will become easier if the mobile application for the license plate recognition has been used to recognize the plate and give access to some databases to search for different background information such as the number of owners or a record of previous accidents. Also, it will benefit individuals like car dealers and traffic clerks. Mobile devices to recognize the vehicle license plate can be used in a fixed position like in a car parking entrance, or in a portable situation as when a person who wants to buy a new car. The numerous number of Smartphone users would be a motivation to implement such an application because it would help to solve some crimes by finding the stolen cars.

The license plate recognition mobile application might be linked to the websites or the databases that records the stolen cars like [spotter.co.nz](http://spotter.co.nz) in New Zealand. This system might be given the ability to recognize the car plate and the compare it the stolen car plates available in a database. If so, then it would do the next step like sending a text or email automatically to inform the people responsible. In addition, it can use the features of the phone like sending the GPS location and a

picture within the text message or email. The user of a website like spotter.co.nz is required to text a message to a specific number and wait for the response or search by himself online to see if a car is stolen, while such a website will be more powerful if there is an automatic check then the plate characters, GPS location, picture and time are sent. This will hasten the process of the searching, the sending and save the applicant time.

Current mobile devices now have processors and cameras that make image processing and recognition applications feasible on them, but are still limited compared to desktop computers. Therefore, the major objective for this research is to identify license plate recognition techniques that are accurate and efficient within the processor and memory limitations of mobile phones.

## **1.1 Objectives**

This research has two main objectives. The first objective is to investigate methods for high accuracy real-time plate detection and recognition on a Smartphone. This will take into account the accuracy of results in plate detection, in character recognition and speed in the detection and recognition process. The second objective is to create a completely free application for the normal end users. Therefore, in order to achieve these objectives, the following research questions should be answered.

## **1.2 Research question**

The aim of this project was to investigate methods for high accuracy real-time plate recognition on a Smartphone, involving application of computer vision and image processing techniques, taking into account device limitations and end user considerations. Six questions are investigated:

1. Is it possible to create a robust license plate recognition application that works in real time on a Smartphone?
2. Can the plate be identified in any position (front or rear), (cantered or offset) without using other identifying features on the car, like the brake lights?
3. How quickly in terms of number of camera frames and real time can the license plate be recognized?
4. Does recording the recognition results over multiple frames give better results?
5. How can accurate OCR be achieved without significant training data and using excessive effort?

### **1.3 Approaches**

The commercial systems are heavyweight approaches. They are very expensive and they need significant amount of time and data for training and for machine learning. They also require high quality cameras and large amount of memory.

This research aimed to avoid using heavyweight approaches in plate recognition, potentially using processors and memory intensive on the mobile device. The intention was to create a system that did not require extensive training.

Using the heuristic method and the Tesseract OCR produced accurate and quick license plate recognition. The research used New Zealand plates (Figure1.3-1), but the ease of use in other countries was also considered to be an essential outcome. This new system using the heuristic method provides a lightweight approach and to apply this approach on another countries car plates there is almost

no setup cost. The only effort to be expended is to grab a plate font file and train Tesseract.



Figure 1.3-1 examples of New Zealand vehicle license plate.

The heuristics method developed to detect the license plate depends upon mathematical relations like plate width and height ratio and the area ratio in relation to the whole frame and the characters' bounding rectangles' ratio compared to the candidate plate. By following such a method, the plate would be found wherever it was positioned and regardless of the surrounding colours. Also, the highest variance method investigated as if candidate plate with the highest variance of black and white is chosen to be the plate.

By training the Tesseract using New Zealand license plate font, an accurate result was produced with no need to gather a numerous data samples for machine learning. The character frequency counts have been used to increase the result percentage to reach the peak point in most of the tested videos.

## **1.4 Thesis structure**

Chapter Two will discuss prior work in the area of license plate recognition, including pre-processing, features analysis, finding plate region, character segmentation and character recognition. Chapter Three presents techniques adopted for plate detection and plate character recognition. Chapter Four shows how the Tesseract OCR has been trained and discuss the trained-data results and which were the best. Chapter Five presents and discusses the results of experiments that have been done using the implemented system on 2543 video frames. Finally, Chapter Six presents the thesis conclusion and the recommendations for future research.

## 2 Chapter Two: Related Work

This chapter will discuss some of the license plate recognition related work. Beginning from a historical perspective. Then license plate recognition techniques will be discussed. License plate recognition techniques, in general, are broken into five phases: pre-processing, features analysis, finding plate region, characters segmentation and character recognition (Figure 1.4-1). Those five steps are going to be discussed in this chapter.

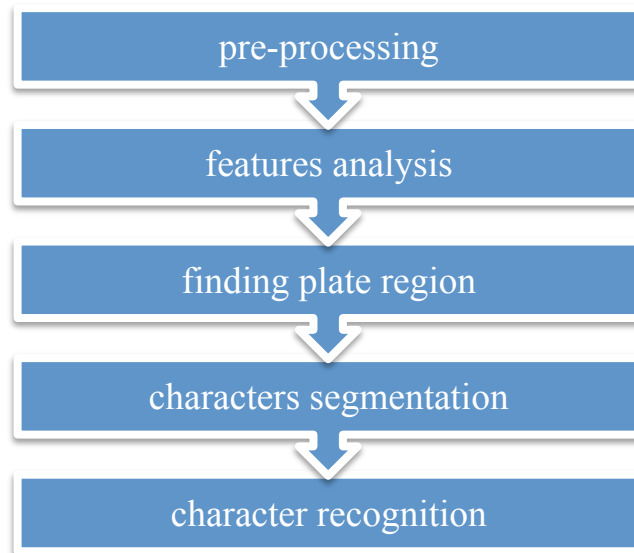


Figure 1.4-1 General license plate recognition techniques phases

### 2.1 Historical view

In 1967, the Automatic Number Plate Recognition (ANPR) was developed. Three years later the first prototypes of the developed system were tested in England (Sheldon, 2011). The inception use of the ANPR was in the 1980s. “The technology was used covertly and was very expensive”(Sheldon, 2011, p. 199). Since then license plate recognition has progressed and been used by police and other organizations for different purposes and has become cheaper and faster, but not cheap enough for private use. In the next sections, some of the recent methods

in the area of license plate detection and recognition will be discussed.

## **2.2 Pre-processing**

From the license plate recognition pre-processing perspective, most commonly used solutions are divided into two steps. First, the input image is converted to gray scale format in order to provide a suitable image format for the next steps like finding features: this technique used by Waterhouse (2006). Second, comes the noise reduction, even though the techniques might vary in different studies. Two of the commonly used techniques to reduce the noise are the Gaussian filter and Median filter. These have been used by Sedighi and Vafadust (2011) and Songke and Yixian (2011) respectively. Additional approach that has been used by many researches is the contrast enhancement. For instance, it was used by (e.g. Abolghasemi & Ahmadyfard, 2009) .

## **2.3 Features analysis**

After the input image has been provided and pre-processed, some features have to be found to help in detecting the plate. A frequently used approach is to use edge detection and from that to find rectangles where the edges are connected under some criteria. Some researchers were looking for vertical lines. Others use horizontal lines to find those that may form part of a plate boundary. Others find sets of connected edges that enclose regions of image. Some use Canny, like Kong, Liu, Lu, and Zhou (2005), while, Sobel edge detection also used by some others like in Jiao, Ye, and Huang (2009). A different technique could also be used. The research by Huang, Chang, Chen, and Sandnes (2008) does not use edge detection but is looking for high contrast area because plates are generally higher in contrast than other segments of the image.

## **2.4 Finding plate region**

After the finding features step is completed, there might be several candidate plate regions as a result. Therefore those regions have to be filtered to reduce the number of candidate plates. A frequent approach used by more than one research study, is to use features of the candidate region's bounding rectangle, like size width, height aspect ratio and location for candidate filtering. For example, Yoon, Lee, and Lee (2009) used some calculations of the rectangles features to determine the plate. Also, based on some geometrical features of the candidate regions, Abolghasemi and Ahmadyfard (2009), Kong et al. (2005) and Tsai, Wu, Hsieh, and Chen (2009) have determined the candidate plate region in their research studies. Gazcón, Chesñevar, and Castro (2012) followed this with a further technique by assuming that the plate has six characters: if the candidate rectangle contains six identifiable sub-regions then this is the plate. Also, other research (Hung & Hsieh, 2010) has followed the geometrical features technique with a similar technique by taking the number of plate characters into consideration while filtering the candidate plates. Chen, Chen, Huang, and Wang (2011) adopted a slightly different approach by using the high density edges then horizontal projection techniques in their approach to determine which one of the candidate regions is the plate.

## **2.5 Character segmentation**

In some approaches characters have to be segmented in order to send them for recognition. This section will discuss some techniques used for this purpose by different researchers. Songke and Yixian (2011) and Huang et al. (2008) used the histogram approach to segment the license plate characters. This approach uses the accumulation or sum of the vertical and horizontal projections. An alternative

way to segment the plate characters is the connected-component labeling algorithm. This algorithm was used by Yoon et al. (2009) and Maglad (2012). This algorithm works in two steps. First it detects the connected blobs (characters), which is in black while the background is white. Then it labels the blobs (characters) with a bounding box. Liaqat (2011) used two algorithms, Canny to detect the edges and contour finding algorithm to find the connected edges that can be labelled using other methods, to segment the characters.

By this stage, some approaches use the results and information above for character recognition step. These characters and information can be provided to a mechanism to recognize them but a further judgment would be better. For example, at this stage in the research Tsai et al. (2009), the characters should pass some conditions to ensure that this is a plate before recognition. These conditions were that the width, and height of each character rectangle should be similar, the density of each character should be in a selected range, and the height must be greater than the width.

## **2.6 Character recognition**

Character recognition can be done in two different ways in regard to license plate recognition. In the first, the characters are provided to a recognition engine (package). For example, Liaqat (2011) is using the Open source Tesseract OCR. Tesseract will be provided with segmented character image and will recognize the characters with a textual output. In the second approach, bespoke recognition methods are implemented. Features are computed for each character, then when character recognition is needed, the feature of the tested character can be matched with the patterns implemented earlier. For instant, in order to recognize the characters the system will extract its features and compare them against patterns

that have been implemented earlier (Juntanasub & Sureerattanan, 2005). Alginahi (2011) has followed the same approach by extracting 88 features for each character and comparing these features with the earlier computed pattern features. In the research by Songke and Yixian (2011) The characters are extracted by histogram and recognized by using template matching. Another way is using a neural network (Ghosh, Sharma, Islam, Biswas, & Akter, 2011). The neural network is trained first and then provided with the character features to be recognized by comparing those features with the trained features. Tsai et al. (2009) found that recording the recognition results over the video sequence improved the final results.

## 2.7 Accuracy of related work

The accuracy for vehicle plate recognition systems can refer to multiple aspects of the process. First aspect is how accurately can the approach detect the license plate. Second aspect is how accurately can the approach recognize the license plate characters. The third aspect is the overall accuracy. Table ... shows the accuracy of some of the related work in regard to these aspects.

**Table 2.7-1 related work accuracy**

Authors	Detection accuracy	Recognition accuracy	Overall accuracy
Kong et al. (2005)	96.1 %		
Alginahi (2011)	98.3 %	98.63 %	94.9 %
Ghosh et al. (2011)	84 %	80 %	
Hung and Hsieh (2010)	95.33 %	88.71 %	
Chen et al. (2011)	Between 90.6 and 92.5 %		
Juntanasub and		92 %	

Sureerattanan (2005)			
Yoon et al. (2009)	96.5 %		
Huang, Chen, Chang, and Sandnes (2009)	96.7 %	97.1%	93.9%
Gazcón et al. (2012)		85.88 and 91.3 %	
Sedighi and Vafadust (2011)		90.5	
Liaqat (2011)		73 %	
Huang et al. (2008)		85 and 100 %	
Maglad (2012)	95 %	91 %	
Songke and Yixian (2011)	95.2 %	88.3%	80%
Jiao et al. (2009)	95.9%	92.3%	90%
Tsai et al. (2009)		92.68 %	
Waterhouse (2006)	96 %	83 and 93%	

## 2.8 Summary

A review of the current literature has shown that, using the edge detection approach will give useful information to find the license plate effectively. Using geometrical features on mobile phone will be fast and research shows that it seems to be an effective and appropriate approach. For one recognition solution researchers created their own system, others used the OCR engines. It was found that when using the Tesseract OCR with a pre-defined font extensive samples were not needed. Parts of some solutions are used in solving the problem addressed in this research, while others cannot be used.

## **3 Chapter Three: Plate and Character Detection**

This chapter describes the process by which a license plate and its constituent characters are detected in an image. Image may be sourced in a continuous stream from a device's camera or from pre-recorded video.

### **3.1 Implementation**

This solution implemented on galaxy nexus phone that is running an android 4.0.2. In the implementation, two libraries have been used: OpenCV libraries and Tesseract OCR libraries. Implementation of license plate detection and recognition was done in native C++ code for the android device.

### **3.2 Plate and character detection overview**

Overview of the process is shown in Figure 3.2-1 and it will be described in more detail in this chapter.

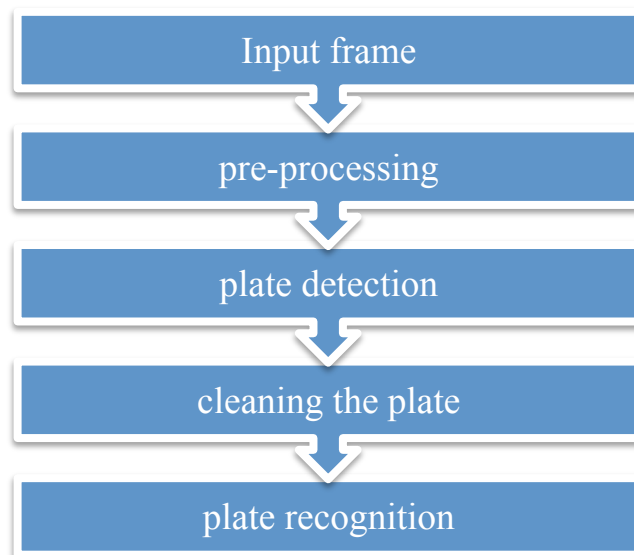
Input frame: captures image from camera proceeding /prerecorded video

Pre-processing: converts the source image into a format appropriate for feature analysis.

Plate detection: analyses image features to identify and select candidate plate regions

Analysis region content: Cleaning the plate and segment the character contours and to determine whether the candidate rectangle is a plate.

Plate Recognition: display copy of the plate image and generate a cleaner plate to be used for recognition then recognize the characters as textual.



**Figure 3.2-1 overview of plate recognition process**

### **3.3 Pre-processing**

Two copies of the frame image are created for different goals. The first is in RGBA colour format for the purpose of the output frames. The second is the same RGBA copy converted to gray scale format. This frame is used in image processing goals, starting with edge detection. Edge detector operators like Canny edge detector are susceptible to noise; therefore, to reduce the noise in the image, the Gaussian blur function has been used. Gaussian blur function enhances and smooths the image (frame) in order to prepare and clean the image to obtain a good edge detection when Canny edge detector is used. The Canny edge detector (Canny, 1986) is one of the best edge detection algorithms provided by OpenCV and FastCV. The Canny algorithm is used to find all the edges in the input image Figure 3.3-1.

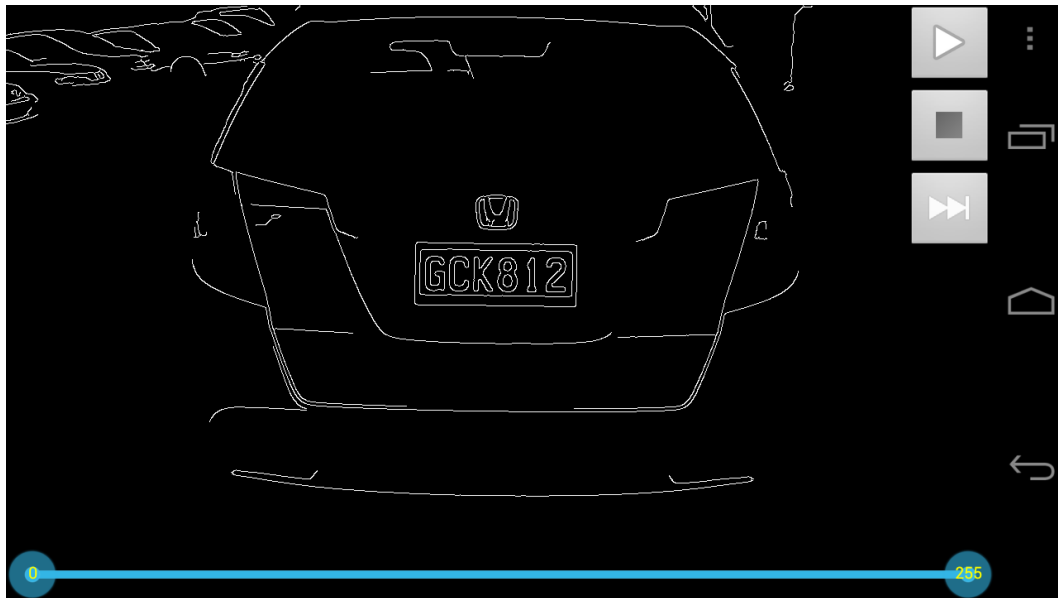


Figure 3.3-1 using OpenCV Canny function to detect the edges.

### 3.4 License Plate Detection

Related work shows many different methods have been used to detect the vehicle license plate. These approaches that use methods like machine learning don't necessary perform quickly enough. Therefore, This paper has followed a new method during the project implementation, relying on new developed heuristics that are fast/easy to compute, efficient, and yielded extremely effective results. In order to detect the plate, all the detected edges are tested in several steps (see Figure 3.4-1): find contours, correct the orientation, width and height, area ration, aspect ratio, rotation and extract and store the estimated plate.

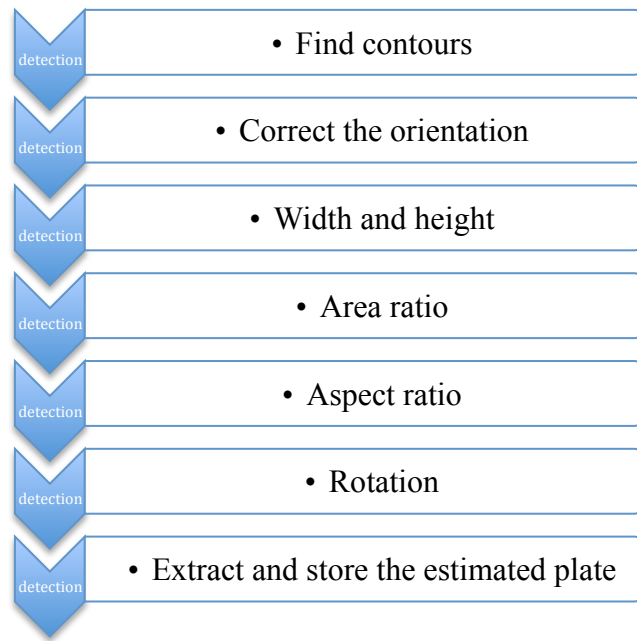


Figure 3.4-1 steps to detect the plate

### 3.4.1 Find contours

The output binary image created from Canny will be used as input binary image. This binary image of the detected edges (output from Canny) is processed to identify sets of connected edges that enclose a region of the image. Each set of connected edges is a contour. This is appropriate because the plates have well defined borders (edges). All the contours are detected regardless of the hierarchical relationship although it might detect contour that is inside another contour (Figure 3.4-2). The detected contours are stored in an array, as part of the function, in order to be used in the next steps.

Each detected contour is a candidate plate region. In order to reduce the number of the candidate plates, only those contours that match plate characteristics (i.e. rectangular, area ratio, aspect ratio, and rotation) will be selected as a potential plate.

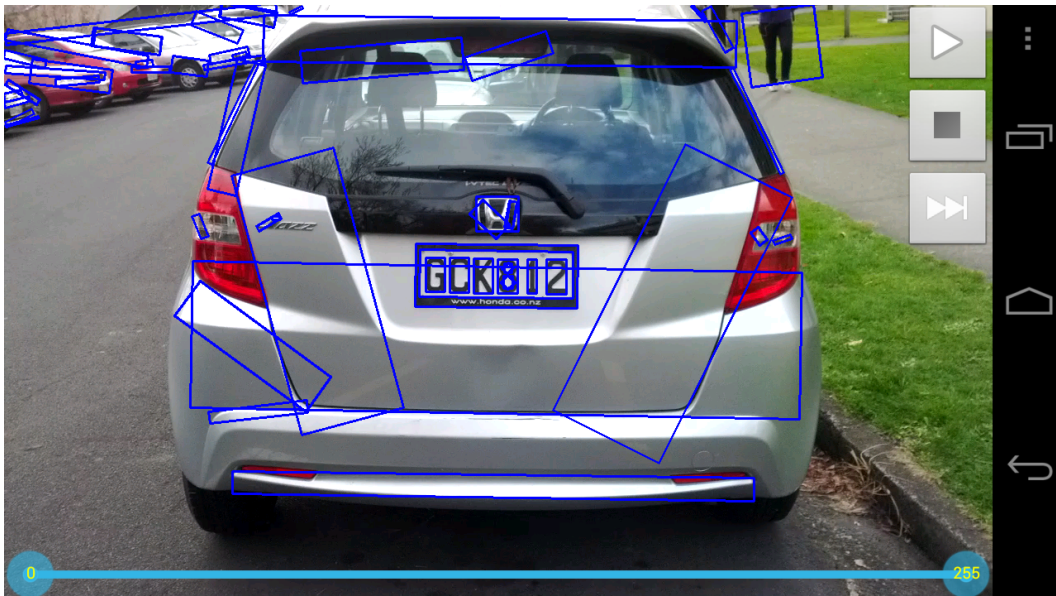


Figure 3.4-2 Example of all the detected contours at the first step.

At this point, the process loops through the stored contours, going through the steps from 3.4.2 to 3.4.6, in order to reduce the number of contours (estimated plates) to the minimum number. The minimum rotated bounding rectangle of each contour is calculated and generated. A bounding rectangle will then be created over the rotated bounding rectangle. The minimum rotated bounding rectangle and bounding rectangle parameters give the data needed in the mathematical and image processing computation in the following stages.

### 3.4.2 Width and height

In the loop of the contours, the issue with some rectangles mismatching width with height has been solved. It is time to eliminate the pointless contours. It is unusual in a New Zealand Vehicle license plate to have a square plate or a plate with the height greater than the width; therefore, if the current rectangle bounding the contour has a height greater than the width then it will be discarded, for instance, the green rectangles in Figure 3.4-3. Otherwise, the contour will go through to the next step.



Figure 3.4-3 rejected rectangles (width greater than height) in green colour

### 3.4.3 Area ratio

The decision in this step is based on the area ratio in the range of the whole frame. The intended use of the technique is when user within 1.5 to 3 meters of vehicle, the plate ratio in relation to the image dimensions would be from 1 to 3 percent. This judgment was made after some experimentation in this regard. As a result of this step, Contours with bounding rectangle outside this range are rejected. For example, the purple rectangles in Figure 3.4-4 are rejected because their area ratios are not in the range of 1 to 3 percent in relation to the image dimensions.

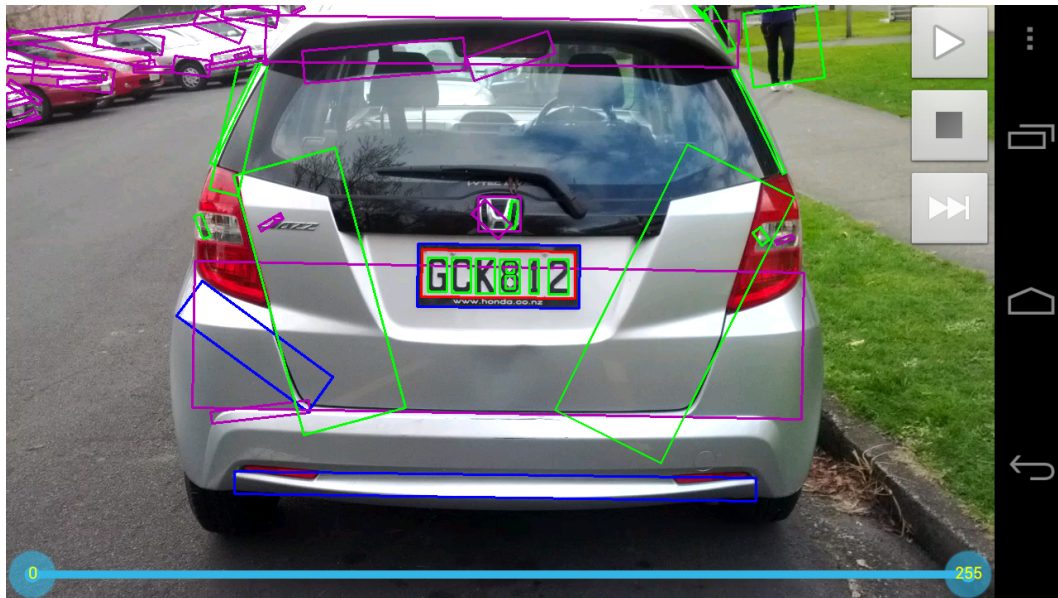


Figure 3.4-4 rejected rectangles in purple (large or small area ratio)

### 3.4.4 Aspect ratio

New Zealand vehicle's license plate has a fixed height and width, regardless of how many characters are on the plate. However, even though the previous step narrowed down the number of contours, focusing on the output result of the area ratio still has retained many rectangles in the area range and with width > height, but not necessarily dimensions corresponding to plate. Hence, the plate aspect ratio is taken into account in order to minimize the number of contours correctly. Consequently, the approximate ratio of the plate height over the width is determined as being greater than or equal to 25 percent and not greater than 40 percent. The decision made upon the bounding rectangle of the contour's aspect ratio within this approximation percentage will pass to the next step while reminders will be rejected such as the blue rectangles in Figure 3.4-5. As a result of this, many non-plate rectangles will be rejected and fewer contours will be obtained to focus on in the next step.



Figure 3.4-5 blue rectangles rejected because of the aspect ratio are not from 25 to 40 percent

### 3.4.5 Rotation

Given use case we expect plates to be approximately parallel with edge of image, but subject to minor rotation. The angle of the contour's rotated rectangle could be one of the conditions to ensure this is a plate. Normally, some parts of the car or in the around area could be detected as a rectangle (contour) and pass all the previous conditions. Figure 3.4-6 shows good examples in yellow colour, which are rejected because it has a large angle and they are obviously not a plate. Accordingly, the plate in the video would appear in a horizontal position; thus, the angle of the rotated angle for the estimated plate is set to be between 3 and -3 degrees. Non-plate regions that don't meet this expectation may remain (filtered by skew).

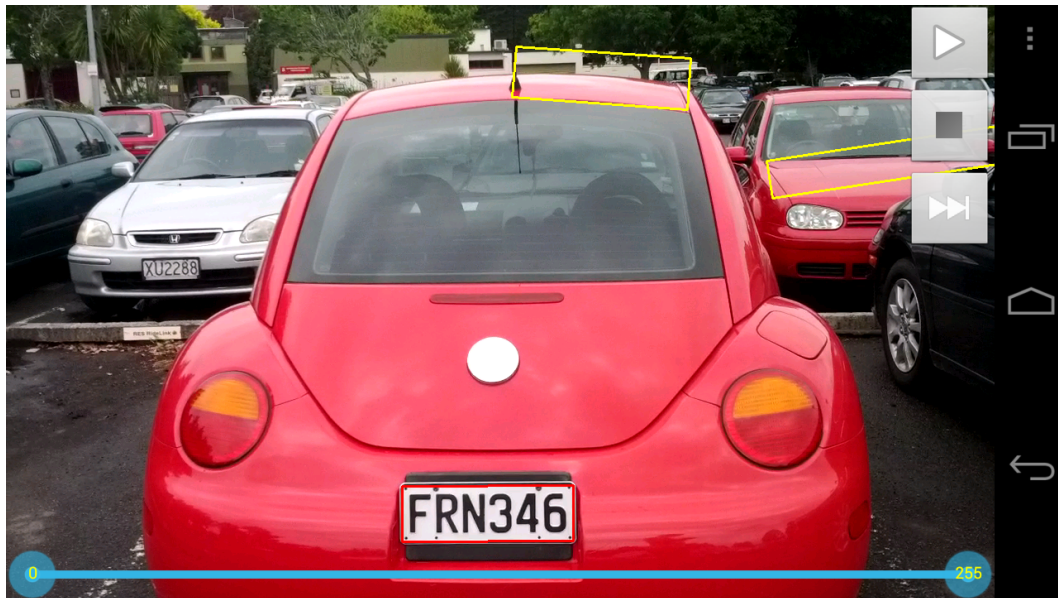


Figure 3.4-6 yellow rectangles rejected because of the skew angle

### 3.4.6 Extract and store the estimate plate

A large number of candidate regions have been reduced to a small number confirming to heuristics. The remaining contours that identified as potential plate will be used in deeper image processing to select the required license plate. Each remaining region is extracted from the source image for further processing. The next step analyses the features of each region to be accepted or rejected as plates.

### 3.4.7 Rejected Alternative solution in plate detection

By calculating the variance and looking to the histogram or calculating which one of the selected rectangles have the highest variance the plate can be identified because it generally has black characters and white background. Most of the other parts of the car will be colourful or plane black or white; therefore, the plate will be shown as the highest variance and it can be chosen as the only suitable rectangle. Figure3.4-7 shows the plate is chosen rather than the rectangle of the windscreen.

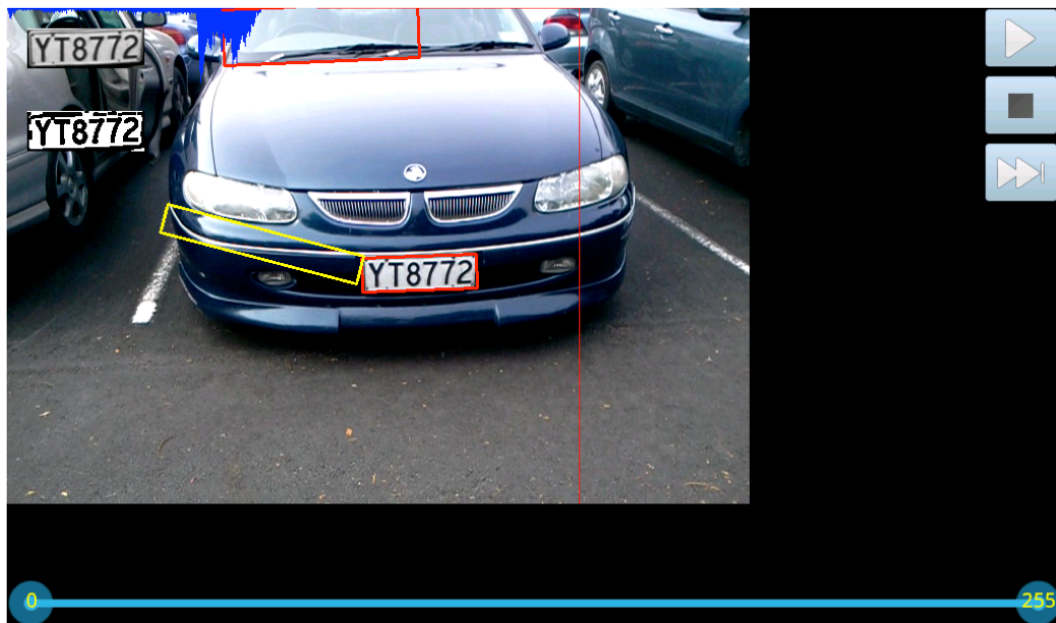


Figure 3.4-7 variance identified the plate correctly

However, Figure 3.4-8 shows unexpected results. In this case, the headlight is chosen as having the highest variance of black and white. In other cases depending on the light source, the grille or another rectangle may be selected as the estimated plate. The solution to this problem is to use a different process (as described above) which checks each rectangle in more detail for the preference of registration plate characters.

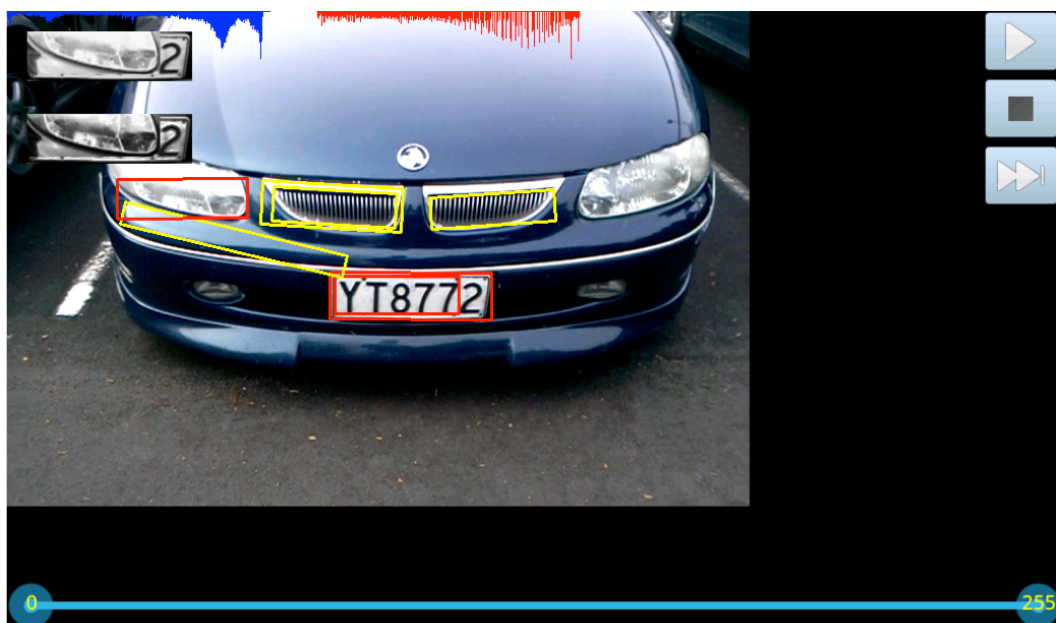


Figure 3.4-8 highest variance misclassified headlight as plate

### 3.5 Analysing Region Content:

The estimated plate or plates has been detected as a result of the previous section. Now, it is time to make image processing and to apply some condition to these estimated plates to narrow down the number of estimated plates has been detected. In order to do that, all the estimated plates are processed and tested in seven main steps (see Figure 3.5-1): adaptive threshold within closing filtering, rotate the image geometrically, cleaning the plate border, image processing prior to finding the character contours, find and store characters contour, reducing the number of non-character contours and deciding whether the estimated rectangle is a plate.

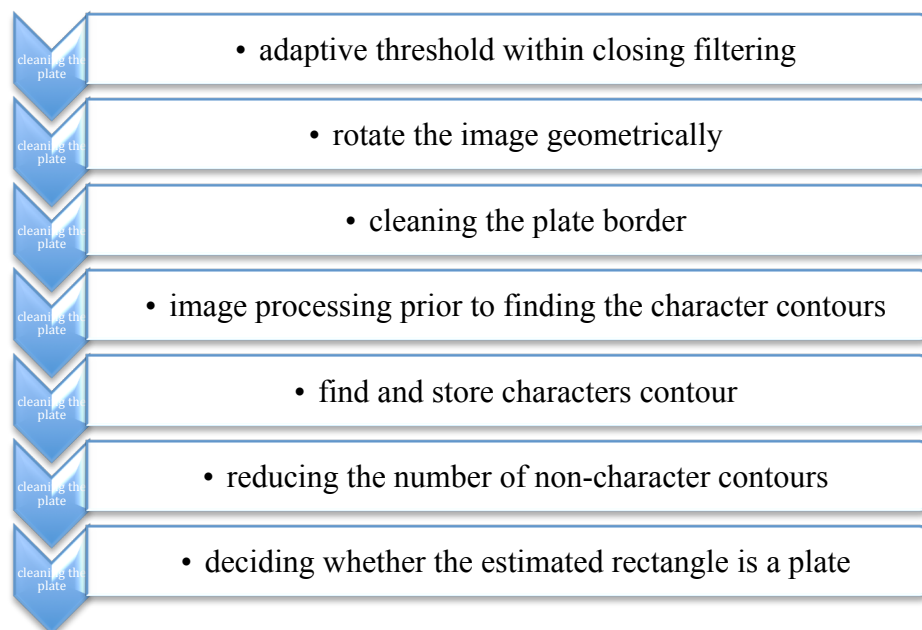


Figure 3.5-1 steps to clean the estimated plate

#### 3.5.1 Adaptive threshold within closing filtering

Closing filtering is a combination of two morphological transformations, dilate then erode. Bradski and Kaehler (2008, p. 121) have stated that closing is used “to reduce unwanted or noise-driven segments”. In this project, a combination of dilate, adaptive threshold and erode (adaptive threshold within closing filtering)

has been performed on the estimated plate to reduce the number of noise speckles. Dilation morphology helps to enlarge the white area of the plate by making the black holes smaller and destroying the little noise simultaneously, while the characters were big enough to remain. The adaptive threshold is applied next to smooth the plate image and generates sharp edges for the characters. Because of the different colour gradient in the image of the plate, the adaptive threshold is chosen to produce the whole plate after thresholding is processed (Bradski & Kaehler, 2008, p. 139). After the dilation and the adaptive threshold most of the noise in the plate has been destroyed while the characters become sharper. Then the erode function is applied to the output image to make the character (black holes) bigger and to round the sharp edges. See Figure 3.5-2 illustrates the detected license plate before and after this step.

	Before	After
1		
2		
3		
4		

Figure 3.5-2 some examples before and after filtering

### 3.5.2 Rotate the image geometrically

The estimated plate rectangle out of the plate detection the plate might be a skewed rectangle in some situations because it was allowed to pass from the previous section if the skew angle is between 3 and -3 degrees. In order for the plate characters to present as a straight line horizontally, the rotated contour's angle from the previous stage, and the centre of the estimated plate rectangle, is used to produce the rotation matrix. The rotation matrix is generated and applied to the rectangle using WarpAffine. After WarpAffine is applied, a non-skewed rectangle can be used in the next step since the plate is now horizontal. For instance, Figure 3.5-3 illustrates some examples of the plates before and after rotating the skewed angle.






	Before	After
1		
2		
3		

Figure 3.5-3 the plates before and after rotation

### 3.5.3 Cleaning the plate border

Having the estimated plate rectangle from the plate detection, the plate might be a skewed rectangle in some situations because it was allowed to pass from the previous section if the skew angle was between 3 and -3 degrees. While plate detection in progress the border is needed to detect the plate. After detecting the plate, it is time to remove the plate border because it might have an affect on the result in the next steps and in the character recognition step. The plate border will be removed in four different loops:

1. To remove the top border
2. To remove the bottom border
3. To remove the left border
4. To remove the right border

The process to remove the top border is to loop through every single pixel in the top row of the plate rectangle from the left to the right. If one pixel is found as not being white then the floodFill (OpenCV function) will be applied. When the floodFill function is performed on a seed pixel, it will fill this seed and all the connected black pixels with white. Removing the bottom border follows the same procedure but on the bottom row. The left and right borders also are removed by using the same process but only to the left column and right column respectively from the top to the bottom. After deleting the border using floodFill some noise of the border remains, for instance see the column named After in Figure 3.5-4. Thus, a closing filter, which is dilate then erode respectively, is used to eliminate the remaining border noise, for example see the third column in Figure 3.5-4. In the same Figure the column called before shows the plate with border before this step has begin.







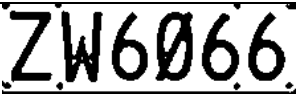
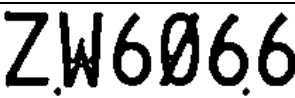
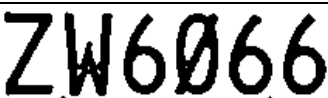
	Before	After	Border noise removed
1			
2			
3			

Figure 3.5-4 examples before and after cleaning the plate border

### 3.5.4 Image processing prior to finding the character contours

At this point regions may actually contain plates (e.g. Figure 3.5-4) or other non-plate (e.g. see the forth row in Figure 3.5-2) segments of the source image. Can analyse features to see if it has plate characteristics i.e. sub-regions that are likely plate characters. The regions at this point might still contain small black dots of noise (e.g. the attaching screws). In order to find the character contours accurately using FindContours, there is a method (CV\_RETR\_EXTERNAL) for retrieving the external boundaries but this only represents the outer edges of the holes (the white regions of the contour) (Bradski & Kaehler, 2008). This method is needed because many characters are recognized as two or more contours for instance number 8 the slashed 0 and the character B. When this method is used every character will be found as one contour; therefore, two copies of the estimated plate is needed: the first copy (called colour swapped plate e.g. Figure 3.5-5) will be used in finding the character contours. The characters need to be in white for the purpose of finding the external boundaries; thus, looping through every pixel and swapping its colour from white to black and vice versa. The second copy (clean border e.g. Figure 3.5-5) will keep the original but will set every pixel to the colour it belongs to either black (0 value) or white (255 value). There is no

gray scale colour remains, values from 0-127 becomes black (0) and values from 128 to 255 become white (255). The (clean plate) will be used for further processes such as output show and in the OCR process.

	The plate after cleaning plate border	Colour swapped plate	Clean plate
1	GCK812	GCK812	GCK812
2	FYR765	FYR765	FYR765
3	Z.W6066.	Z.W6066.	Z.W6066.

Figure 3.5-5 plates with cleaned border, plates with colour swapped and clean plates.

### 3.5.5 Find and store characters contour

Contours are identified within each region and the bounding rectangle of each is computed. For plate characters the contour defines the external edges of the character and the rectangle is the character's minimum bounding box. Which is expected to have characteristics that differ from non-character contours.

### 3.5.6 Reducing the number of non-character contours

In order to eliminate the non-character contour, three conditions will be applied to each contour from the previous step. Iterate over each contour bounding rectangle in the candidate plate image, if the current contour height is less than 55 percent compared to the whole plate rectangle's height, then the first condition is met and the whole rectangle will be filled with white (deleting the black contour), see the red contours (they were noise and screws) in the middle copied plate in Figure

3.5-6. Secondly, if the first condition is not met then the program will check whether the character rectangle width is greater than or equal to 25 percent of the plate rectangle's width or less than 50 percent of the plate rectangle. If these two conditions were met then it is not a character and floodFill function is applied to delete this contour. To avoid deleting some other contours like the blue rectangle in the middle copied plate in Figure 3.5-7, the floodFill function will be applied on the first black pixel it finds in the middle column of the current contour. The reason that the middle column has been chosen is that in some cases the bounding rectangle of a contour overlaps part of another contour. For instance, see the blue rectangle in the middle copied plate and the result in the bottom copied plate in Figure 3.5-6. The middle row has not been used so as to avoid deleting a character accidentally because the noise rectangle might be overlapping the character rectangle, which means the character is going to be deleted because it will be at the beginning of the noise rectangle from the horizontal view. Thirdly, if the first two conditions are not met then the program will check whether the contour rectangle's width is greater than or equal to 50 percent of the plate rectangle then the whole contour rectangle will be filled with white (deleted). For example, see blue rectangle in the middle copied estimated plate in Figure 3.5-8 it has been deleted. Otherwise, the reminding contours will be stored in an array to be used in the next step and with the clean plate (as defined earlier). The result of this step is an image for each candidate plate region in which only features that have characteristics of license plate characters remain.

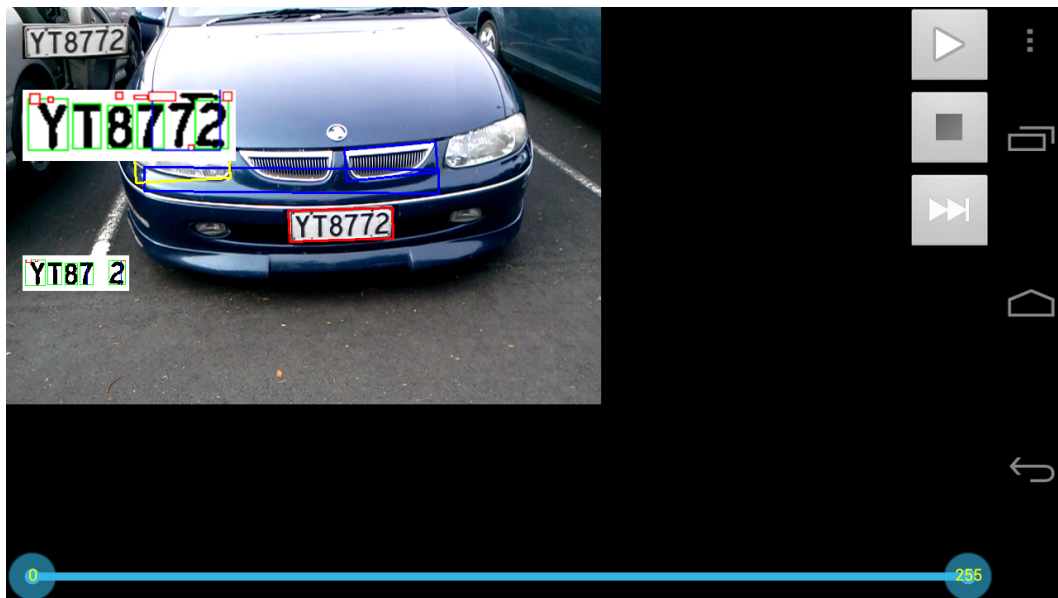


Figure 3.5-6 eliminating small noise and avoiding eliminating another contour by mistake.

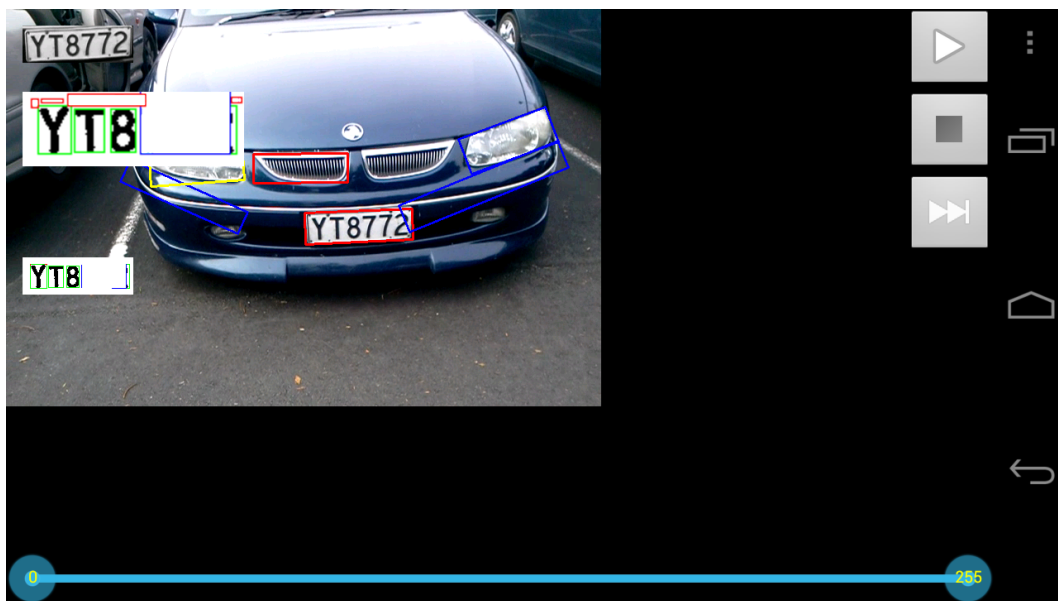


Figure 3.5-7 eliminating another contour accidentally because of the overlapping.



Figure 3.5-8 contour rectangle's width is greater than or equal to 50 percent of the plate rectangle

### 3.5.7 Deciding whether the estimated rectangle is a plate

Uses the positions of remaining features to determine if they match expectations for license plate layout. Based on the results of the contours stored in the array at the end of the previous step, some conditions will be applied to decide whether the rectangle is a plate. First, when the array of the contours only has one contour then the position of this contour will be checked according to the whole plate. In New Zealand a vehicle license plate might only contain one character or two, if so their position has to be in the middle of the plate. The position of the top left pixel of the contour rectangle has to be in the middle of the plate; otherwise if it is found to be in the first third or the last third of the plate then the estimated plate rectangle is not a plate and will be rejected. Second, if the contours array includes two contours then the position of these two has to be checked according to the whole plate. If the position of the top left pixels of these two contour rectangles are found to be in the first third or the last third of the plate then the estimated plate rectangle is not a plate and it will be rejected. If the character rectangles are in the middle of the plate then it is a plate. Finally, when the number of contours in the array is three or more then it is a plate. Now, the sub-images of the clean

plate and the original will be converted from gray scale to colour images to be used in the output show. The coordinates of the character contour rectangles will be applied to the clean plate to locate the characters, and then they will be stored in a new array. Meanwhile, the plate location coordinates according to the whole frame will be stored to be used in some situations in the next frame. At this point, there is no sequence of which seem to be a plate. If there is a plate recorded then proceed with step 3.7(plate recognition). If not then take the alternative action (step 3.6). Result is zero or more regions from source image with the visual characteristics of a plate. The candidate character regions are extracted from each for the character recognition step.

### **3.6 Alternative Procedure When No Plate Found**

In practice variations in the source image such as lightning, rotation, focus can affect ability to detect plates. For the intended use there is high likelihood that if the previous frame contained a plate, so will the current frame. This allows remedial action to be taken if no plate is found in the current frame .The result of the previous step (Deciding whether the estimated rectangle is a plate) dictates which of separate ways the next step will follow. When the results are positive then the process will continue to step 3.7 - Plate Recognition. Otherwise, this step the alternative procedure would be followed (activated) in order to find the plate in other ways. The alternative procedure is finding the plate rectangle in a smaller portion of the frame, and finding the plate by using the previous location coordinates. The former will be followed first, when it is not successful in finding the plate then the latter will be applied. These procedures will be described in more detail next.

### **3.6.1 Find the plate in part (ROI) of the frame**

When the plate has not been found at the earlier steps, the region of interest will be narrowed down using the previous location coordinates and enlarging it by 40 pixels to the height and the same to the width. The rectangle is enlarged by 20 pixels in each direction. The virtue of enlarging the previous plate rectangle by 40 pixels emerges as giving more flexibility for detecting the plate if the camera is moved gradually between the frames from one position to another (e.g. left to right or top to left). Meanwhile, by enlarging the ROI, the program is avoiding the loss of a character (when it touches the border pixels of the rectangle) in the case of a dramatic movement between frames. The ROI is subtracted from the frame copy that passed the Canny edge detection. On the smaller region, findContours is applied to find all the contours, then these contours will follow the same process as in 3.4 License Plate Detection and in 3.5 Analysing Region Content. If a plate is found then the plate is retained and the procedure continues to plate recognition in 3.7, otherwise it passes to the procedure that finds the plate by using the previous location coordinates.

### **3.6.2 Find the plate using the previous location coordinates**

Even though the coordinates are for the detected plate from the previous frame, it is necessary to ensure, when this is applied on the new frame, that it produces a plate. In order to ensure it is a plate, the rectangle that is subtracted according to the previous plate location will be tested. This new estimated plate will go through the conditions and image processing in cleaning the plate. This process will establish whether it is a plate or not. If the region is a plate then the plate is retained and the process will continue to plate recognition. Otherwise, the current frame is rejected, as not containing a plate.

## **3.7 Plate Recognition**

The next step in the process is to attempt OCR on the plate characters. When a plate has been identified, many different processes are used in order to recognize the license plate characters and show them with two copies of the plate, before and after processing, simultaneously. These processes includes fixing some issues with the sub-image copies of the plate, resizing and pasting these copies into the original frame, and sending and retrieving data to and from the OCR. Finally, the character's frequencies are kept and the final characters are showed as a typed letters on the original frame. This section will go through these processes in more detail in the following paragraphs.

### **3.7.1 Show a copy of the original and the clean plate**

Using the coordinates of the plate rectangle, the passed plate is copied from the plate of the original image (frame). Assuming that the top left corner would not contain a plate, on the output frame this position is set to paste the copy of the original plate on. This copy helps to show which part of the frame has been detected as a plate. After the image processing, the clean plate will be resized and copied to the output frame in a position underneath the original plate copy. The resizing is performed on the clean plate. (Figure 3.7-1) Shows an example of this step at the top left corner is a copy of the original plate and underneath that is a copy of the clean plate.

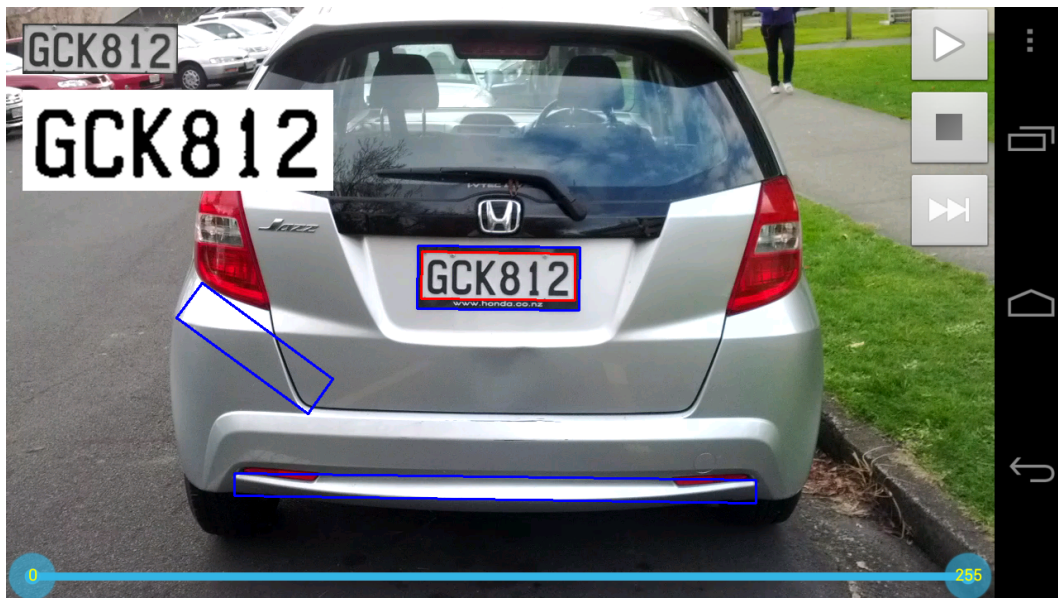


Figure 3.7-1 copies of the original and clean plate at the top left corner.

### 3.7.2 Create a cleaner plate

The character regions for each plate are stitched together and scaled to provide optimal input to the OCR process. Also, the other noises have to be removed, all the characters have to be aligned on the base line and the characters have to be straightened.

One way to produce a cleaner plate is to insure that the background of the image is clear of noise (white). A new white image with the same size and type of the final processed plate (clean plate) is produced. Since the background of the new image now is clean (white), the character's contour rectangles will be pasted on it from the clean plate, as they have been stored in the array. During the testing some issues have been found while the character rectangle is being pasted on to the new image. These issues are with the coordinates of the characters rectangles. Some of the coordinate either becomes negative or out of the range and the most prominent cause is that the angle of the plate was skewed. To overcome this issue

each character rectangle in the array will pass through the following steps to ensure all the coordinates are in the range of the clean plate rectangle that will be copied from.

1. Top left x pixel has a negative value

The x value of the top left pixel of each character rectangle is checked: if it is negative then it will be set as zero; therefore, the width will be changed. To keep the original width and to avoid characters overlapping (because if the top left x value is negative this belongs to the first character rather than the others, and it might overlap with the second character) the negative value of the x (old value) will be added to the width.

2. Top left x pixel has a large value

The top left pixel of each character rectangle is checked: if its x value is larger than the clean plate number of columns then it is out of the range. To solve this issue, the top left x value will be set as the clean plate number of column minus 2; therefore, the width will be changed. To keep the original width and to avoid losing a part of the character (because if the top left x value is moved to the left then the top right x value will be less than the original) the old value of the x will be added to the width. Making the top left x value in the range while the top right x value is still the same solves the issue and insures that the whole character will be copied.

3. Top left y pixel has a negative value

The y value of the top left pixel of each character rectangle is checked: if it is negative then it will be set as zero; therefore, the height will be changed. To keep the original height, the negative value of the y (old value) will be added to the height.

4. Bottom right pixel values (x, y or x and y) are out of the range

Sometimes the changes of the WarpAffine affect the bottom right pixel values.

If the x value is out of the range (greater than the clean plate rectangle width) then the width of its contour rectangle has to be changed to be equal to the clean plate rectangle width minus its current top left x value, regardless of whether or not it has been changed in the steps above. If the y value is out of the range (greater than the clean plate rectangle height) then the height of its contour rectangle has to be changed to be equal to the clean plate rectangle height minus its current top left y value, regardless of whether or not it has been changed in the steps above.

When the character rectangle passes through these steps, it will be copied to the new white image on the current position. After all the characters have been copied, the new image will be enlarged to double size. After some testing it has been found that the resizing gives a better result in OCR with some characters, like W. The characters still have sharp edges and need smoothing to round the edges. Out of many filtering functions, median Blur with a 9 size of aperture liner has produced the best results for character smoothing in such a situation. Now, after smoothing the characters a scaled copy of the cleaner plate will be shown on the output frame, see the copied plate at the bottom left in Figure 3.7-2, while a gray-scaled image copy will be sent to the OCR because the OCR does only accept binary images.



Figure 3.7-2 Scaled cleaner plate at the bottom left.

### 3.7.3 Tesseract OCR

Predominantly, when an estimated plate has reached to this point it is a plate. The plate is still recognized as an image and it needs to be recognized as textual characters. For many reasons, Tesseract has been chosen to perform the OCR in this project. These reasons, the alternative training data, the training process, and the best-trained data for the purpose of New Zealand license plate recognition and why it was the best, will be illustrated more fully in the next chapter.

After the best trained-data has been trained and chosen, it will be placed in the sdcard folder in the android smartphone to be used in the recognition process. The Tesseract SetVariable should be set to include only the characters from A-Z and 0-9 because these are the only characters that need to be recognized in the license plate (there is no punctuation). Then the final filtered cleaner plate will be sent as a parameter to Tesseract for character recognition. Figure 3.7-3 shows the overall process diagram including the plate recognition at the lower section. The result from Tesseract in this project has used UTF8 text format. The recognized

characters are then typed on the output frame. To be aware of the misrecognized characters and the empty characters in some situations the character frequency will be computed during the process of each frame and stored in a global array as ASCII format by taking the placement of each character in the plate into consideration. The character with the highest frequency will be selected as being the correct character. If the empty character is the highest, the next highest will be chosen. When a new video is selected, the characters frequencies are set to zero.

**Table 3.7-1 Character frequency over multiple frames**

Frame Number	Characters in license plate					
	S	X	9	0	4	0
45	44	44	45	45	45	45
101	99	100	101	101	101	101
150	146	149	150	150	150	150
Majority choice	S	X	9	0	4	0
Highest frequency	150	150	150	150	150	150

Table 3.7-1 shows the characters frequency at three different frames during the recognition process. By frame 45, the character S has been misrecognized once. And the character X has been identified once as empty character. By frame 150, character S has been misrecognized four times; X has been identified as empty once. The majority choice can be made after every frame, even though in some frames some characters might be incorrectly recognized but with the majority choice they will be discarded. As the number of processed frames increases the correctly recognized characters become a significant majority.

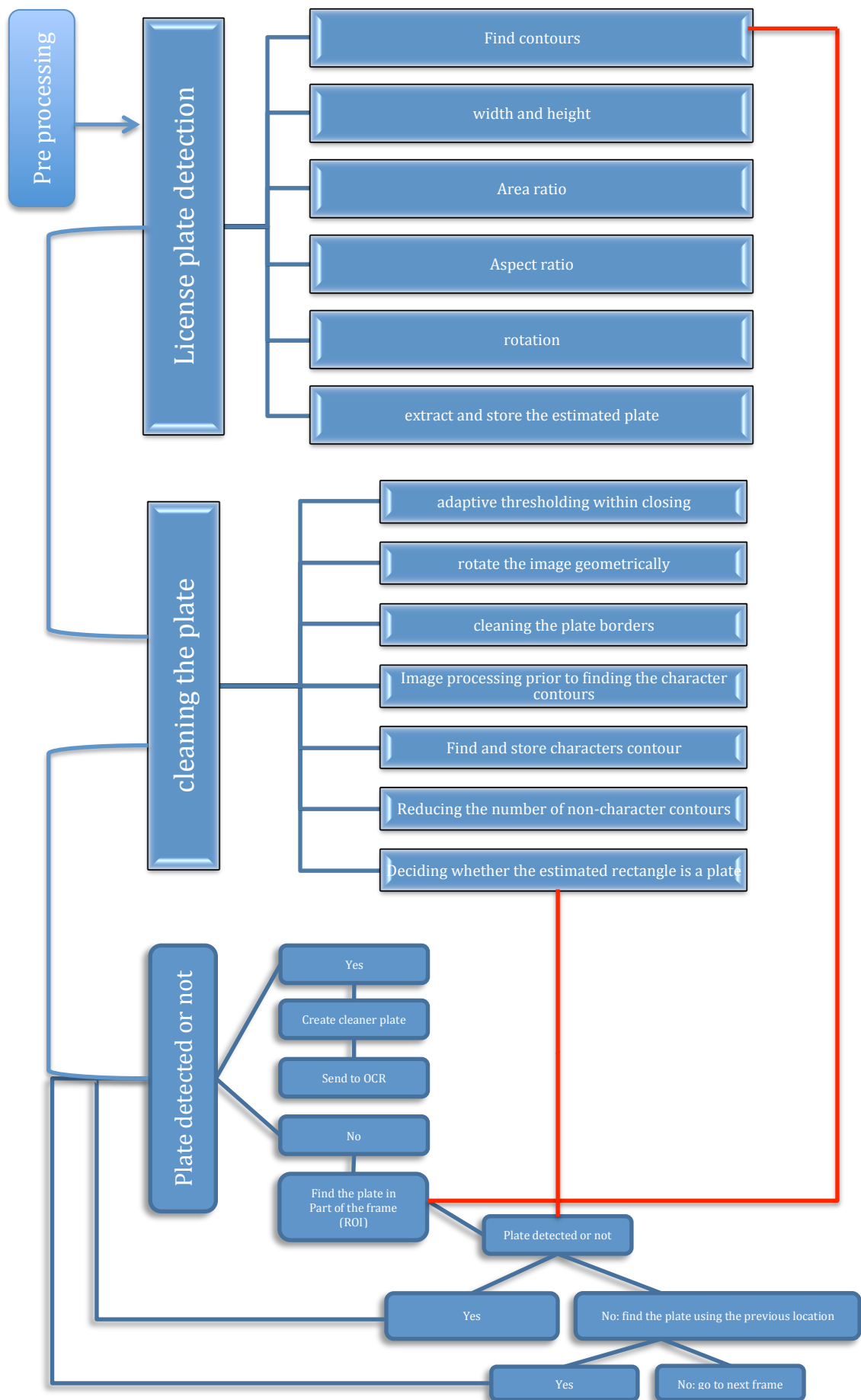


Figure 3.7-3 overall process diagram

## **4 Chapter Four: Plate Character Recognition**

### **4.1 Introduction**

When an image that contains characters requiring to be converted to a text the Optical Character Recognition (OCR) is used. In the previous chapter, the car license plate is detected and stored as a cleaned and smoothed image that only contains characters. Therefore, using OCR will produce the plate characters in text and make them searchable, more readable and storable.

### **4.2 Choosing Tesseract**

Tesseract is an open source OCR engine and that was one of the reasons why it has been chosen. It was originally developed by HP between mid-eighties and mid-nineties (Smith, 2007). In 2005 Tesseract became open source and in the next year was sponsored by Google. The second important reason for choosing Tesseract is that, Tesseract's performance of recognizing the characters was close to the commercial OCR systems (Breuel, 2008) in 2008 and it has been under development since then. Thirdly, Tesseract OCR provides an implement Tesseract-library that can be used as stand alone OCR in the android application using the Eclipse software. Finally, because Tesseract is an open source engine, new font or set of dirty images can be trained to gain the best-trained data that can be used in the Tesseract smart phone libraries in order to recognize the characters. The next step will describe how Tesseract has been trained using a new font or set of character images.

## **4.3 Training process**

### **4.3.1 Installation**

Tesseract engine and ImageMagick is installed using the command line in the Ubuntu desktop. ImageMagick software is open source and a command-line interface for converting image files. The box editor JTessBoxEditor ("jTessBoxEditor," n.d.) is installed to help in editing the box file during the Tesseract training process.

### **4.3.2 Create the training document**

The training document is created using the new font, a set of license plate character images or a combination of both as will be shown in section 4.4 Training data and results. After the training document has been created, in order to train the data in Tesseract the document has to be converted to tif format, which is done using ImageMagick. Once in the correct format (tif format), the training document should be named properly. Therefore, the name will include the language abbreviation from the SIL international standard (SIL International, 2012) followed by the font name (e.g. eng.Licenz.tif - eng refers to English language and Licenz refers to the new font).

### **4.3.3 Train Tesseract**

When the training document is ready, Tesseract engine is run over it. The result of this step will be in a box format. The box format is like the first draft in writing because it shows the first shot of the recognition result and it can be changed if any of the results are incorrect. For each recognized character, the box file provides its X, Y width and height coordinates according to its bounding rectangle position in the tif file. JTessBoxEditor is one of the available softwares that ease

the correction. JTessBoxEditor shows and enables editing the results of the box format file (recognized characters in the training image) in addition to, corresponding characters from the tif file that the Tesseract has started with, based on the coordinates in the box file. During the checking, if one character has been misrecognized then it should be corrected in the box file; for instance, in many cases the character 'W' in the New Zealand license plate was recognized as 'N'. Also, when two characters have been recognized as one then they should be broken up (given new coordinates) to ensure that they have their corresponding characters in the box file. One character might be recognized as two characters, therefore the coordinates of the character's bounding rectangles in the tif file should be merged (correct the coordinates) and it should be given the correct character in the box file. When all the misclassified characters have been corrected, then the box file should be sent back to Tesseract to continue the training process.

Once the box file has the correct data and the input image is in the right format, both the box file and the training image feed back to train in Tesseract using training mode. When the training is completed the characters features set should be computed and extracted. This project has only one set of character features extracted because it is doing the training for only one font. Also, to produce the final trained data, a file for the font properties needs to be created. In this case, the font properties file contains one line representing the box file name and the characteristics of the trained font. Then clustering data has to be created using the Font properties, the character set and the trained file to produce the prototypes. Four language files will be produced as a result of the previous step; in order to generate the trained-data file those language files should have the language prefix

in their file names. Now all the files with the name beginning with eng language prefix (includes: tif, box, tr, and the four files from clustering) are combined by running combine-tessdata. The combination order will create the trained-data file that can be used to recognize characters of the new font, which can be used in the android smartphone.

## **4.4 Training data and results**

The goal of this Tesseract font training is to make the recognition of the New Zealand license plate characters more accurate. However, five different trained-data were tested:

- ❖ Default Tesseract model using Standard English language font.
- ❖ The font that is used in New Zealand license plate (Buck, 2006).
- ❖ Images of characters captured from actual plates.
- ❖ The font that is used in New Zealand license plate with an image of the zero with an extended slash from actual plate.
- ❖ Multiple instances of images of characters captured from actual plates.

During this testing, it has been found that the number character 0 is still causing issues in recognition; therefore, the zero with an extended slash from actual plate added to the New Zealand license plate font. These trained-data source and results will be discussed next in more detail. Tables from 4.4-1 to 4.4-5 present the result of the character recognition using different Tesseract trained data on sixteen different test videos, which contain 2543 frames. These videos are named after the license plate number. In two videos there were only two misclassified plates, one frame for each. For instance, Figure4.4-1 shows the misclassified plate and how it

represents a non-plate character (these two misclassified plates are still the same in all the other test data results). In a few cases some of the characters were cleaned during the image processing filtering therefore the number for these character in the tables is less than the other characters in the same row; for instance, some frames in GKK709 and DEL311 videos (Figure4.4-2). In two videos (TO3145, DEL311 and FUE503) one of the characters was not recognized because it was connected to the border, which led to its deletion during the border cleaning Figure4.4-2. In two frames only in TO3145 a part of the left border remained which makes it recognized as the letter I and that causes mismatch recognition for the rest of the characters.

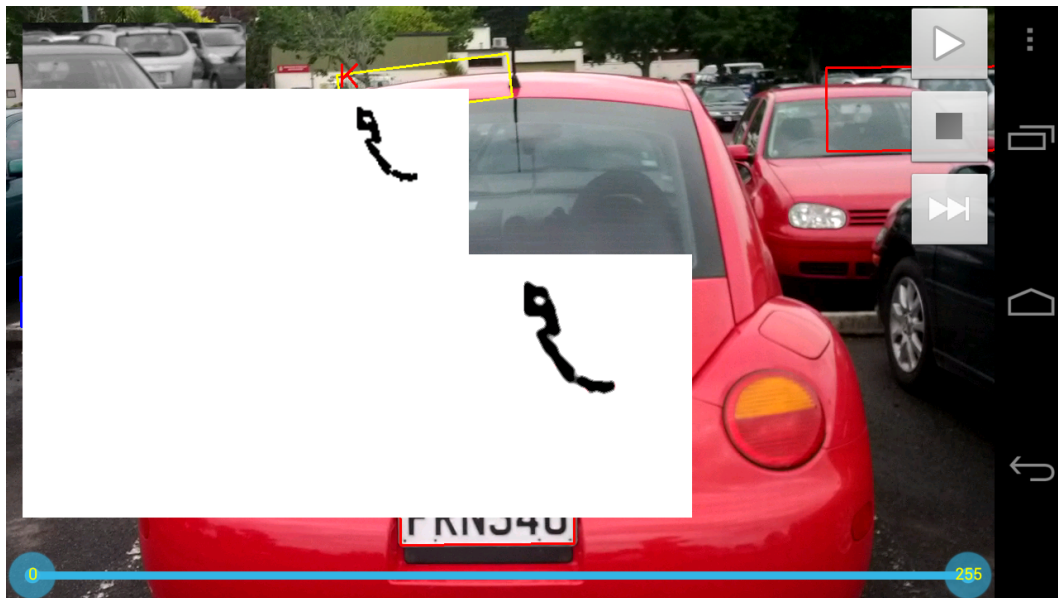


Figure 4.4-1 mis-detected plate



Figure 4.4-2 two characters deleted during filtering



Figure 4.4-3 character E deleted during the border cleaning

#### **4.4.1 Standard English trained data**

The tables 4.4-1 to 4.4-5 use the same abbreviations as column headings. They are listed in the following glossary.

LP	Tested license plate.
T	Total number of frames.
NFDP	Number of how many frames has a detected plate.
NFMDP	Number of frames with missed Detected plate.
C1	How many times Character one recognized Correctly.
C2	How many times Character two recognized Correctly.
C3	How many times Character three recognized Correctly.
C4	How many times Character four recognized Correctly.
C5	How many times Character five recognized Correctly.
C6	How many times Character sex recognized Correctly.
#	Misrecognized character
' '	Empty character

**Table 4.4-1 Standard English trained data results**

LP	T	NF DP	NF M DP	C1	C2	C3	C4	C5	C6
FRN346	148	147	1	147 #K=1	147	147	147	147	147
EYG796	171	63	0	63	63	63	63	63	63
EWJ843	267	260	0	260	#H= 212 #N= 48	260	260	260	260
EPM11	153	16	0	16	16	16	16	16	
TO3145	161	156	0	152 #I =2 #1=1 #W= 1	12 #0= 141 #T=2	152 #0=1 #E=1 #O=1	151 #2=1 #3=2 #’ ‘ =1	134 #6=9 #1=11 #0=1	57 #S=2 #9=3 #7=1 #1=27
FUE503	157	150	0	146 #U=1	149 #E=1	149 #5=	97 #S=52 #B=1	#B=123 #8=10 #9=16 #3=1	114 #B=6 #I=13 #Q=2 #S=9
DSC825	155	154	0	151 #I=3	151 #J=3	151 #S=3	151 #C=3	150 #Z=1 #8=3	151 #2=3
GMA305	150	136	0	135 #B=1	136	136	58 #B=7 #C=3 #G=2 #H=1 #Q=21 #S=29 #I=15	#B=86 #8=6 #9=31 #E=2 #I=1 #J=10	67 #S=51 #B=16 #I=2
DQR871	160	160	0	160	130 #D=5 #E=14 #I=9 #O=2	137 #I=8 #J=15	137 #R=23	137 #8=23	137 #7=23
JEANEZ	148	148	0	148	148	148	148	148	148
GKK709	158	145	1	143 #K=1 #’ ‘=1	144 #I=1	143 #R=1	143 #A=1	#B=123 #6=1 #8=3 #9=15	142
SX9040	150	83	0	83	83	83	79 #9=3 #D=1	83	81 #9=2
DEL311	145	124	0	122 #3=1 #L=1	84 #’ ‘ =38 #3=1	122	122	122	93 #I=2 #’ ‘ =18
FSB290	156	47	0	47	47	47	47	47	#8=16 #9=9 #B=22
FHR274	147	140	0	140	140	140	140	140	140
LEVIT8	117	88	0	88	88	88	88	88	88

Table 4.4-1 presents the result of the character recognition using the Standard English trained data on sixteen different test videos. From Table 4.4-1, in video EWJ843 the letter W has not been recognized correctly, it has been recognized as H or N, for example see Figure4-4.4 the W is incorrectly recognized and displayed as H in red, at the too left of the car. In the same table, it can be seen that the number character 0 mostly was recognized as B, 8 or 9 instead of 0. In many situations, D, 3 and Q were recognized as two characters for example IJ, EJ or EI in the videos DSC825, GMA305 and DQR871 respectively (Figure4.4-5). Those misrecognitions had an effect on the rest of the characters. Because they are recognized as two characters, they affect the next characters and so the next character will move to the next position and so on. Table 4.4-1 shows many characters that were mis-recognized as well.

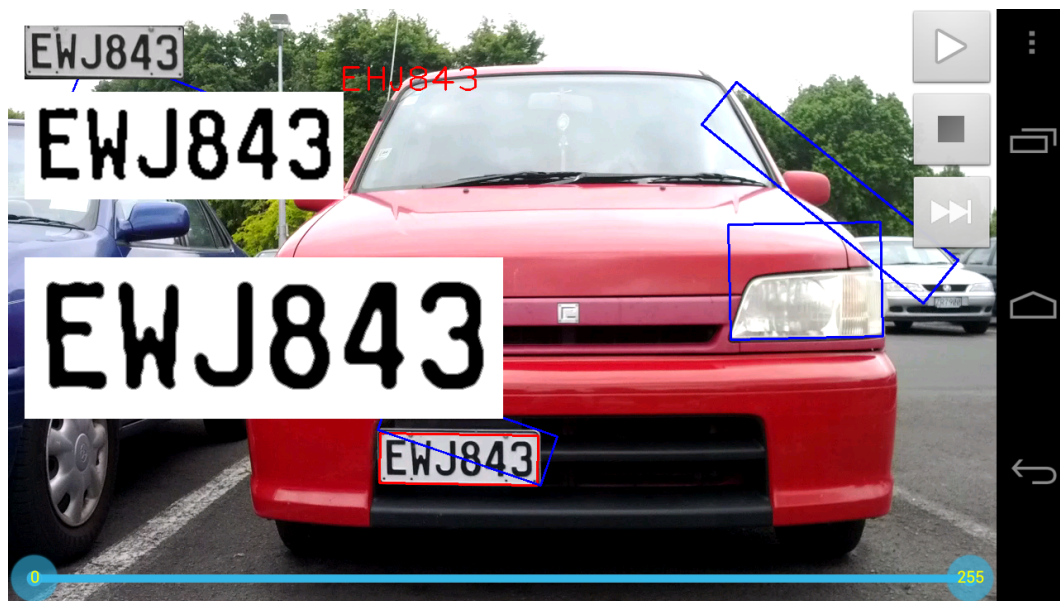


Figure 4.4-4 Character W recognized as H



Figure 4.4-5 Q recognized as EJ

#### 4.4.2 Plate font A-Z and 0-9

The (Licenz) New Zealand License plate font trained data has been used in order to improve the accuracy of the recognition results. In general, comparison to the results of the Standard English results in Figure4.4-1, the plate font A-Z and 0-9 (Figure 4.4-6) present better results (Figure4.4-2) with overall increase by 4.85 percent. Also, it is obvious that the letter character W was recognized correctly as in Figure 4.4-7 (Table 4.4-2) when the plate font A-Z and 0-9 trained-data was used, compared to the Standard English trained data results. Even though plate font A-Z and 0-9 trained data showed a big improvement in the results, the number character 0 is still not recognized well.

**ABCDEFGHIJKLMNOPQRSTUVWXYZ  
0123456789**

Figure 4.4-6 Licenz font A-Z and 0-9 <sup>2</sup>

<sup>2</sup> <http://www.leewardpro.com/articles/licplatefonts/font-licenz-nz.html>

**Table 4.4-2 plate font A-Z and 0-9 results**

LP	T	NF DP	NF MD P	C1	C2	C3	C4	C5	C6
FRN346	148	147	1	147 #K=1	147	147	147	147	147
EYG796	171	63	0	63	63	63	63	63	63
EWJ843	267	260	0	260	260	260	260	260	260
EPM11	153	16	0	16	16	16	16	16	
TO3145	161	156	0	153 #I=2 #G=1	153 #T=2	152 #E=1 #O=2	152 #3=2 #’ ‘=1	152 #O=1 #1=2	12 #9=69 #S=10
FUE503	157	150	0	149 #U=1	149 #E=1	149 #5=1	149 #Q=1	#Q=118 #G=16 #D=9 #6=6	144
DSC825	155	154	0	154	147 #5=7	154	154	154	154
GMA305	150	136	0	136	136	136	136	#Q=85 #G=40 #D=1 #8=1 #6=9	136
DQR871	160	160	0	160	160	160	160	160	160
JEANEZ	148	148	0	148	148	148	148	148	148
GKK709	158	145	1	141 #’ ‘=1 #6=2 #K=1	144 #3=1	143 #Z=1	143 #A=1	#6=15 #8=3 #B=6 #D=6 #G=25 #Q=87	142
SX9040	150	83	0	82 #5=1	83	83	10 #Q=73	83	#D=1 #O=22 #Q=60
DEL311	145	124	0	117 #3=1 #L=1 #O=5	84 #3=1 #’ ‘=38	122	122	122	95 #’ ‘=18
FSB290	156	47	0	47	47	47	47	47	#Q=41 #6=1 #G=5
FHR274	147	140	0	140	140	140	140	140	140
LEVIT8	117	88	0	88	88	88	88	88	88



Figure 4.4-7 character W recognized correctly

#### 4.4.3 Plate font + zero with an extended slash

In the previous step it was found that there was a big improvement in the results except with the number character 0; it was not recognized properly in most of the cases. In order to recognize the number character 0 correctly, a zero with an extended slash (Figure 4.4-8) was added to the plate font input image in the Tesseract training. Table 4.4-3 shows the results for the number character 0 were significantly improved in comparison to the previous two trained-data results (increased by 9.77 and 4.92 in relation to the Standard English language font and the New Zealand License plate Font). For example, see Figure 4.4-9 that shows the characters 0 recognized correctly.



Figure 4.4-8 Examples of: slashed zero at the left & extended slashed zero at the right

**Table 4.4-3 plate font + zero with an extended slash**

LP	T	NFDP	NFM DP	C1	C2	C3	C4	C5	C6
FRN346	148	147	1	147 #X=1	147	147	147	147	147
EYG79 6	171	63	0	63	63	63	63	63	63
EWJ843	267	260	0	260	260	260	260	260	260
EPM11	153	16	0	16	16	16	16	16	
TO3145	161	156	0	153 #I=2 #G=1	147 #T=2 #O=1 #Q=5	152 #2=1 #O=2	152 #3=2 #' '=1	152 #O=1 #1=2	3 #9=85 #S=3 #4=2
FUE503	157	150	0	149 #U=1	149 #E=1	149 #5=1	149 #0=1	138 #3=1 #G=11	144
DSC825	155	154	0	154	152 #5=2	154	154	154	154
GMA30 5	150	136	0	136	136	136	136	136 #G=7	136
DQR87 1	160	160	0	160	160	160	160	160	160
JEANE Z	148	148	0	148	148	148	148	148	148
GKK70 9	158	145	1	143 #' '=1 #K=1	144 #3=1	143 #0=1	143	142	142
SX9040	150	83	0	80 #5=3	83	83	83	83	83
DEL311	145	124	0	95 #3=1 #L=1 #O=2 7	84 #3=1 #' '=38	122	122	122	95 #' '=18
FSB290	156	47	0	47	45 #5=2	47	47	47	47
FHR274	147	140	0	140	140	140	140	140	140
LEVIT8	117	88	0	88	88	88	88	88	88



Figure 4.4-9 character 0 was recognized correctly

#### 4.4.4 Real plate characters and a zero with an extended slash

The input of this trained data was a combination of one instance of each character from a real plate except that for the character zero there were two instances (one zero with internal slash and one zero with an extended slash). Two zeros were added because there are two different zeros in the real plates and as that has been tested in the trained-data above, one of these zeros would not be enough to recognize all the zeros in the test videos. Table 4.4-4 shows the characters W and 0 were generally recognized correctly. However, in several videos (EWJ843, EPM11, FUE503 and JEANEZ) the character E was recognized as two characters IZ or I3 in many cases (Figure4.4-10) this caused many mismatches in the rest of the plate characters recognition when it occurred. Also, the character D in many situations was recognized as two characters (I3). In two videos, DSC825 and DQR871, the misrecognition of the letter character D made many mismatches in the rest of the plate characters recognition. In overall the performance of this trained-data was better than the default model using Standard English language font by 2.37 percent.

**Table 4.4-4 real plate characters (one instance of each) + zero with an extended slash trained data results**

LP	T	NF DP	NF M DP	C1	C2	C3	C4	C5	C6
FRN346	148	147	1	140 #P=7 #X=1	147	147	147	147	#G=128 #8=18 #0=1
EYG796	171	63	0	61 #I=2	61 #Z=2	61 #Y=2	61 #G=2	61 #7=2	#8=8 #9=2 #G=53
EWJ843	267	260	0	253 #I=7	253 #Z=7	253 #W=7	253 #J=7	253 #8=7	253 #4=7
EPM11	153	16	0	15 #I=1	15 #Z=1	14 #N=1 #P=1	15 #M=1	16	
TO3145	161	156	0	152 #I=2 #Q=1	105 #T=2 #D=47	91 #B=50 #C=1 #O=2 #S=9 #Z=1	150 # ' =1 #3=2 #U=1	151 #1=2 #O=1	2 #4=2 #8=1 #9=72 #S=15
FUE503	157	150	0	148 #P=1 #U=1	149 #E=1	147 #5=1 #I=2	147 #0=1 #Z=2	147 #3=1 #5=2	142 #0=2
DSC825	155	154	0	146 #0=5 #I=3	149 #3=3 #8=2	148 #0=2 #G=1 #S=3	151 #C=3	117 #8=3 #Z=3 4	151 #Z=1
GMA30 5	150	136	0	136	89 #N=47	136	136	136	136
DQR871	160	160	0	103 #0=32 #I=25	135 #3=25	135 #Q=25	135 #R=25	135 #8=2 5	135 #7=25
JEANEZ	148	148	0	148	146 #I=2	146 #Z=2	146 #A=2	146 #N=2	146 #E=2
GKK709	158	145	1	143 # ' =1 #K=1	144 #Z=1	143 #2=1	143 #A=1	142	142
SX9040	150	83	0	83	83	83	83	83	83
DEL311	145	124	0	1 #0=120 #3=1 #I=1 #L=1	83 # ' =38 #3=2	121 #E=1	121 #L=1	121 #3=1	95 # ' =18
FSB290	155	47	0	40 #P=7	45 #8=2	31 #8=16	33 #Z=14	47	47
FHR274	147	140	0	136 #P=4	135 #W=5	140	71 #Z=69	140	140
LEVIT8	117	88	0	88	88	88	88	88	88

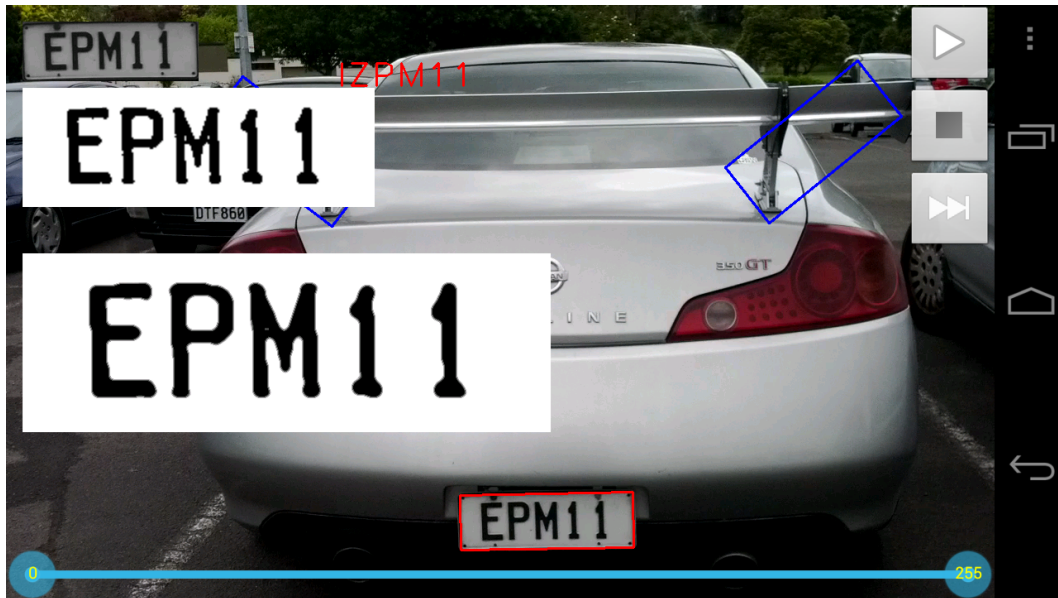


Figure 4.4-10 E was recognized as IZ

#### 4.4.5 Multiple instances of the real plate characters

By ensuring each character appears at least in one plate, multiple copies of plates were used as input data in this trained-data. Even though multiple instances of each character from real plates were used, character misrecognition still occurs (Figure 4.4-5). More than twenty times the letter character W was misrecognized as H in the test video EWJ843. Also, the issue of one character recognized as two occurred seven times in the same video. In general, this trained data presents good results (98.52%) similar to the results from the plate font with zero that has an extended slash trained-data results (98.56).

**Table 4.4-5 multiple instance of real plate characters trained-data test results**

LP	T	NFDP	NF MD P	C1	C2	C3	C4	C5	C6
FRN346	148	147	1	147 #K=1	147	147	147	147	147
EYG796	171	63	0	63	63	63	63	63	63
EWJ843	267	260	0	260	239 #H=21	260	260	260	260
EPM11	153	16	0	16	16	16	16	16	
TO3145	161	156	0	152 #I=2 #H=1	91 #T=2 #D=55 #Q=7	102 #B=17 #O=2 #2=14 #5=20	151 #' #=1 #3=2 #B=1	151 #1=3 #O=1	76 #4=3 #B=3 #9=10 #7=1
FUE503	157	150	0	149 #U=1	149 #E=1	149 #5=1	149 #0=1	149 #3=1	144
DSC825	155	154	0	154	154	154	154	154	154
GMA305	150	136	0	136	136	136	136	136	136
DQR871	160	160	0	160	160	160	160	160	160
JEANEZ	148	148	0	148	148	148	148	148	148
GKK709	158	145	1	143 #' #=1 #K=1	144 #2=1	143 #B=1	143 #R=1	142	142
SX9040	150	83	0	83	83	83	83	83	83
DEL311	145	124	0	122 #3=1 #L=1	84 #' #=38 #3=1	122	122	122	95 #' #=18
FSB290	155	47	0	47	47	47	47	47	47
FHR274	147	140	0	140	140	140	140	140	140
LEVIT8	117	88	0	88	88	88	88	88	88

## 4.5 Best-trained data

A comparison between those five different Tesseract OCR trained data, has been found that the trained data Plate font + 0 with an extended slash (A-Z and 0-9 and 0 with an extended slash) was the best according to its' high percentage of the correctly recognized characters (see Table4.5-1). This table shows the total and the percentage out of the test data according to the correctly recognized characters. Real plate characters (multiple instances of each character) trained data test results had the second highest percentage while the difference between the first and second highest percentages was insignificant. However, several aspects have been identified as the drawbacks of the other three trained data: Firstly, the characters (W and 0) were hard to recognize correctly. Secondly, in many cases one character (like E and D) was recognized as two characters like IJ, which made many mismatches.

Having tested five sets of trained data, it was noticeable that the plate font was reasonably accurate when the other sample of dashed zero had been added to its train data image. It was; therefore, decided to use this trained-data for the project. The good thing about this result is that high accuracy with no effort in capturing real-world training data.

**Table 4.5-1 the total and the percentage of the for each test data**

LP	T	TD1	TD1/ T = %	TD2	TD2/ T = %	TD3	TD3/ T = %	TD4	TD4/ T = %	TD5	TD5/ T = %
FRN346	882	882	100.00	882	100.00	882	100.00	728	82.54	882	100.00
EYG796	378	378	100.00	378	100.00	378	100.00	305	80.69	378	100.00
EWJ843	1560	1300	83.33	1560	100.00	1560	100.00	1518	97.31	1539	98.65
EPM11	80	80	100.00	80	100.00	80	100.00	75	93.75	80	100.00
TO3145	867	658	75.89	774	89.27	759	87.54	651	75.09	723	83.39
FUE503	894	655	73.27	740	82.77	878	98.21	880	98.43	889	99.44
DSC825	924	905	97.94	908	98.27	922	99.78	862	93.29	924	100.00
GMA305	816	532	65.2	680	83.33	809	99.14	769	94.24	816	100.00
DQR871	960	838	87.29	960	100.00	960	100.00	778	81.04	960	100.00
JEANEZ	888	888	100.00	888	100.00	888	100.00	878	98.87	888	100.00
GKK709	861	715	83.04	713	82.81	857	99.54	857	99.54	857	99.54
SX9040	498	492	98.8	341	68.47	495	99.4	498	100.00	498	100.00
DEL311	670	665	99.25	662	98.81	640	95.52	542	80.9	667	99.55
FSB290	282	235	83.33	235	83.33	280	99.29	243	86.17	282	100.00
FHR274	840	840	100.00	840	100.00	840	100.00	762	90.71	840	100.00
LEVIT8	528	528	100.00	528	100.00	528	100.00	528	100.00	528	100.00
T  %	11928	10591		11169		11756		10874	91.16	11751	
			88.79		93.64		98.56				98.52

Table4.5-1 uses abbreviations as column headings. They are listed in the following glossary

- TD1            Standard English trained data
- TD2            Plate font (Licenz) A-Z and 0-9
- TD3            Plate font + 0 with an extended slash (A-Z and 0-9 and 0)
- TD4            Real plate characters (one instance of each) (A-Z and 0-9 and 0 with an extended slash)
- TD5            Real plate characters (multiple instances of each Character).
- T                Total number of the detected characters discarding the empty ones

## **5 Chapter Five: Evaluation and Experimental Results**

Once the implementation and the best-trained Tesseract data have been completed and chosen, some tests have to be performed to show the accuracy of the results. Previously during the training, many videos of different vehicles were used. Now, in order to show the experimental results sixteen different videos were tested. This chapter will explain these videos in more details. Also, the results will be discussed from various points of view in this chapter.

### **5.1 Training and tested data**

While the implementation was under process and the Tesseract trained-data was under training and testing, 35 samples of videos were needed. Those videos included all the characters from A-Z and from 0 to 9 in their plates, including the extended slashed zero. However, since the implementation has been completed and the best-trained data for Tesseract has been chosen, accuracy results have to be computed. In order to compute the accuracy results, new samples of videos for experimental results are needed. Therefore, sixteen new videos, which have 2543 frames in total, have been captured for this purpose. These videos were captured during daytime, mostly on sunny days, with some captured in cloudy days. These videos are in different lengths of frames. These videos include both the front and the rear of the vehicles (see Figures 5.1-1 and 5.1-2 respectively). Also, the plate will be recognized wherever it is positioned, it does not have to be in the centre of the width of the car. For example see Figure 5.1-2 shows that the plate is on the right side of the front of the car.



Figure 5.1-1 Plate in the rear of the vehicle recognized

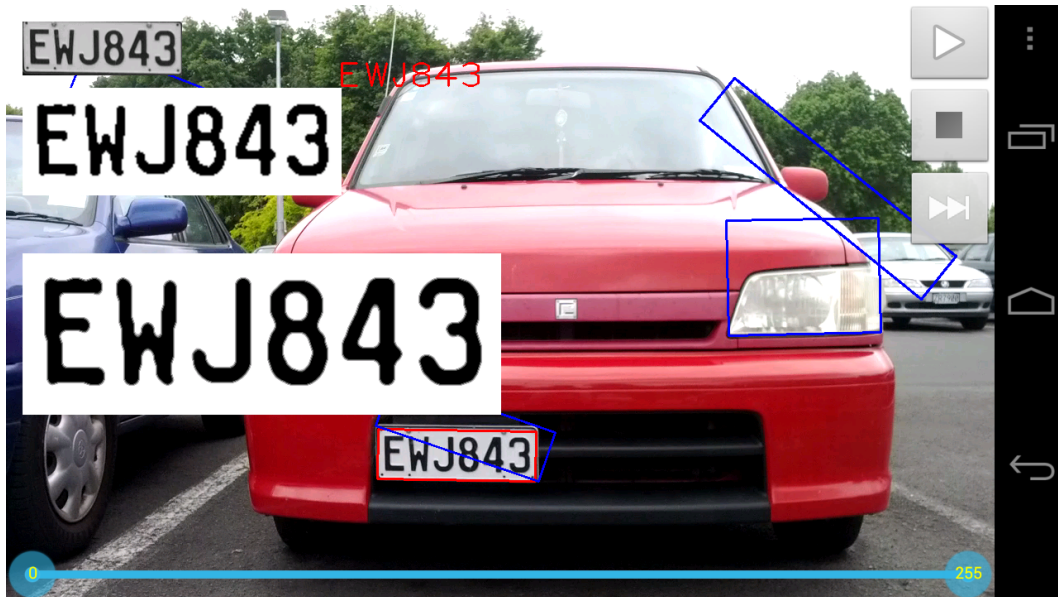


Figure 5.1-2 Plate in the front of the vehicle recognized (the plate not in the center)

## 5.2 Plate detection results

To represent the results for plate detection, the implemented method and the test data has been used. All the 2643 test video frames include vehicle with license plate. The results were computed for both without and with the previous plate location as described in section 3.6. In general only two incorrect regions were detected as plates out of 2543 frames (Table 5.2-3).

The implemented method without using previous plate location was tested first. In this case, the plate was detected in 2017 frames out of the total (2543)(Table 5.2-1). Table 5.2-1 shows 20.60 percent (524 frames) of the frames, which includes plates, have not detected the plate.

**Table 5.2-1 Contingency table shows the detection results without using previous plate location**

Before alternative procedure processed	Image contains plate	Percentage in relation to total number of frames
Total number frames (images)	2543	
Plate detected	2017	79.32%
No plate detected	524	20.60%
Incorrect region detected as plate	2	0.08%
Percentage		100.00%

The result for the implemented method with the previous plate location shows better results. After the previous plate location has been applied in the implemented method, a high percentage (98.90 %: 2515 / 2543 frames) of the frames has been detected the plate correctly. In the case where no plate had been detected, there was a significant improvement after the previous plate location had been applied (see Table 5.2-2). In only 26 frames the plate was not detected.

**Table 5.2-2 Contingency table shows the detection results with using the previous plate location**

After alternative procedure processed	Image contains plate	Percentage in relation to total number of frames
Total number frames (images)	2543	
Plate detected	2515	98.90%
No plate detected	26	1.02%
Incorrect region detected as plate	2	0.08%
Percentage		100.00%

**Table 5.2-3 Detection results for each video**

Video number	License Plate characters	Total number of frames	Total detected plates Before alternative procedure	% Of detected plates Before alternative procedure	Total detected plates after alternative procedure	% Of detected plates after alternative procedure	Incorrect region detected as plate
1	FRN346	148	147	99.32	147	99.32	1
2	EYG796	171	63	36.84	164	95.91	
3	EWJ843	267	260	97.38	267	100.00	
4	EPM11	153	16	10.46	149	97.38	
5	TO3145	161	156	96.89	161	100.00	
6	FUE503	157	150	95.54	157	100.00	
7	DSC825	155	154	99.35	155	100.00	
8	GMA305	150	136	90.67	150	100.00	
9	DQR871	160	160	100.00	160	100.00	
10	JEANEZ	148	148	100.00	148	100.00	
11	GKK709	158	145	91.77	157	99.37	1
12	SX9040	150	83	55.33	150	100.00	
13	DEL311	145	124	85.52	143	98.62	
14	FSB290	156	47	30.13	145	92.95	
15	FHR274	147	140	95.24	147	100.00	
16	LEVIT8	117	88	75.21	115	98.29	
Total		2543	2017		2515		2

Tables 5.2-1, 5.2-2 and 5.2-3 show the final results of both procedures. Although the results of the implemented method before using the previous plate location were reasonable (approx. 79%), there was a significant improvement of about 20 percent after the implemented method with the previous plate location had been applied (Table 5.2-2). Simultaneously, by inverse relationship proportion the percentage in the case of no plate having been detected had been gradually decreased by 19.4 percent to reach 1.02 percent (see Table 5.2-2).

### 5.3 Have the plate characters been recognized?

The alternative procedure made a huge improvement in the detection results. Therefore, in the character recognition results the implemented method with the alternative procedure will be performed. This experimental results used the Tesseract best-trained data (as discussed in Chapter Four) which is the Plate font + 0 with an extended slash (A-Z and 0-9 and 0 with an extended slash). Table 5.3-1 illustrates the results of the character recognition. While Table 5.3-2 illustrates the results of each character recognition after using the frequency counts and Table 5.3-3 shows the comparison between the average of the total recognized characters and the results after applying the frequency method. This result will be discussed in the next two paragraphs.

Table 5.3-1 shows the percentage of plate characters that have been correctly recognized. The table shows that six license plates have been recognized correctly after detection. Table 5.3-1 shows that except in two cases, the misrecognized character percentage in any character is less than 10 percent. For the ones with lower percentage the errors occurred because of the misrecognition or the spaces where character did not find. Misrecognition instance is that, in the video of the license plate **TO3145**, the top right corner of the plate is damaged (Figure 5.3-1) which causes the last number character **5** to have an extended black squiggle; therefore, this has significantly affected the recognition of this character but not the other characters in the plate. The following paragraph will show how the frequency counts of the recognized characters helped to increase the accuracy of these results.

The average total of the recognized characters is shown in Table 5.3-3. By computing the percentages of these values in relation to the total frames with detected plates, it was found that except two plates all the percentages are above 92 percent. The first plate is **DEL311** with about 84 percent; it was affected because of not being able to find where the character is. This made the OCR to recognize them as spaces but they were less than a third of the total percentage. The second plate is **TO3145**. The damage on the plate caused the problem not the detection or the OCR. However, there are only six plates 100 percent correctly recognized at this stage. When the percentage of the average total of the recognized characters is computed in relation to the total number of frames there are slight decreases in the percentages, concluding with only four plates having a full percentage. The recognized character frequencies were applied to increase the accuracy of the plate recognition. When the frequency counts kept for each character then a decision made at the end of the video based on these frequency counts the results were impressive (see Table 5.3-2). Table 5.3-3 shows that, there is a sharp increase in the percentage in the average of the total frequency of all characters that are recognized correctly over the total number of frames with detected plates. Fifteen plates have achieved the peak point (100 percent) while the one remaining is the plate with the damaged corner, (**TO3145**) which has increased by approximately 3 percent. The last column in Table 5.3-3 shows the percentage of the average of the total frequency of all characters that were recognized correctly in relation to the total number of frames. The percentage has decreased slightly, but is still about 97.82 percent on average. The major reason for this drop, while the character frequency is applied, is that in some videos the plate has not been detected at the first couple of frames. This will be discussed in the following results step. To sum up, the total average percentage of characters

accurately recognized is 95.51 percent. This percentage is increased to 97.82 percent when the frequency method is applied.

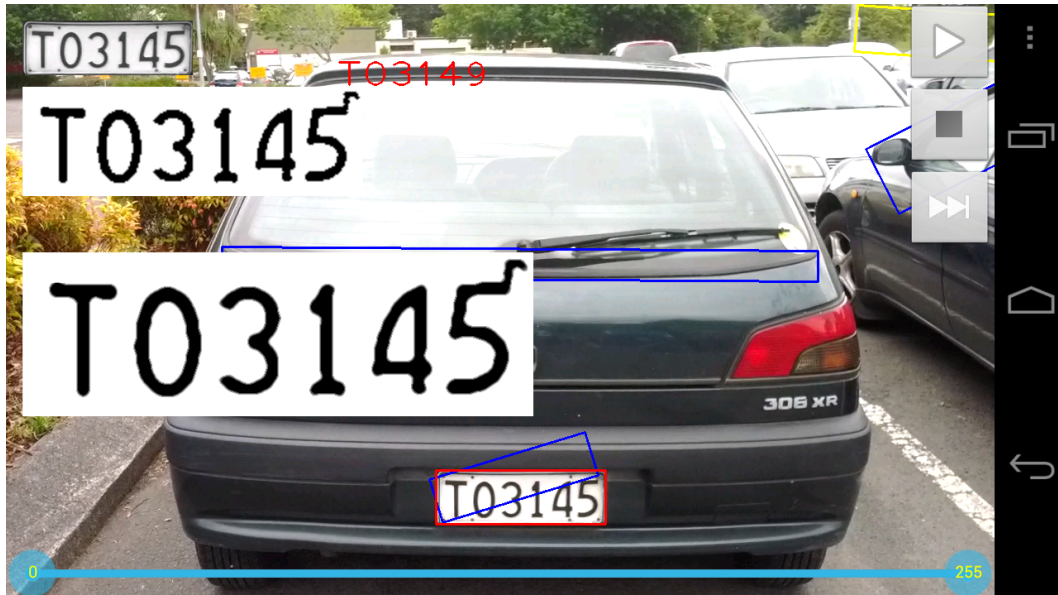


Figure 5.3-1 Top right corner of the plate is damaged

**Table 5.3-1 Character recognition**

* LP	C1 %	C2 %	C3 %	C4 %	C5 %	C6 %
FRN346	100.00	100.00	100.00	100.00	100.00	100.00
EYG796	93.29	93.29	92.68	92.68	90.85	90.85
EWJ843	100.00	100.00	100.00	100.00	100.00	100.00
EPM11	95.97	95.30	95.30	95.30	99.33	
TO3145	97.51	93.78	96.89	96.89	96.89	1.86
FUE503	99.36	99.36	99.36	99.36	92.36	96.81
DSC825	100.00	98.71	100.00	100.00	100.00	100.00
GMA305	100.00	100.00	100.00	100.00	94.67	100.00
DQR871	100.00	100.00	100.00	100.00	100.00	100.00
JEANEZ	100.00	100.00	100.00	100.00	100.00	100.00
GKK709	100.00	100.00	99.36	99.36	98.73	98.73
SX9040	97.33	99.33	100.00	100.00	100.00	100.00
DEL311	75.52	62.24	96.50	96.50	96.50	76.92
FSB290	97.93	96.55	97.24	95.86	96.55	96.55
FHR274	100.00	100.00	100.00	100.00	100.00	100.00
LEVIT8	100.00	100.00	100.00	100.00	100.00	100.00

\*Glossary of column headings.

LP License plate characters

T Total number of frames in the video

TD Total number of frames with detected plates

C1 – C6 Character 1 to 6

**Table 5.3-2 Characters recognized correctly using the frequency in relation to the number of frames with detected plates**

* LP	C1 %	C2 %	C3 %	C4 %	C5 %	C6 %
FRN346	100.00	100.00	100.00	100.00	100.00	100.00
EYG796	100.00	100.00	100.00	100.00	100.00	100.00
EWJ843	100.00	100.00	100.00	100.00	100.00	100.00
EPM11	100.00	100.00	100.00	100.00	100.00	
TO3145	100.00	100.00	100.00	100.00	100.00	0
FUE503	100.00	100.00	100.00	100.00	100.00	100.00
DSC825	100.00	100.00	100.00	100.00	100.00	100.00
GMA305	100.00	100.00	100.00	100.00	100.00	100.00
DQR871	100.00	100.00	100.00	100.00	100.00	100.00
JEANEZ	100.00	100.00	100.00	100.00	100.00	100.00
GKK709	100.00	100.00	100.00	100.00	100.00	100.00
SX9040	100.00	100.00	100.00	100.00	100.00	100.00
DEL311	100.00	100.00	100.00	100.00	100.00	100.00
FSB290	100.00	100.00	100.00	100.00	100.00	100.00
FHR274	100.00	100.00	100.00	100.00	100.00	100.00

\*Glossary of column headings.

LP License plate characters

T Total number of frames in the video

TD Total number of frames with detected plates

C1 – C6 Character 1 to 6

**Table 5.3-3 Character recognition comparison**

* LP	T	TD	AvgTR	AvgTR/TD=%	AvgTR/T=%	F/TD=%	F/T=%
FRN346	148	147	147	100.00	99.32	100.00	99.32
EYG796	171	164	151.33	92.28	88.50	100.00	95.91
EWJ843	267	267	267	100.00	100.00	100.00	100.00
EPM11	153	149	143.4	96.24	93.72	100.00	97.38
TO3145	161	161	129.83	80.64	80.64	83.33	83.33
FUE503	157	157	153.5	97.77	97.77	100.00	100.00
DSC825	155	155	154.67	99.78	99.78	100.00	100.00
GMA305	150	150	148.67	99.11	99.11	100.00	100.00
DQR871	160	160	160	100.00	100.00	100.00	100.00
JEANEZ	148	148	148	100.00	100.00	100.00	100.00
GKK709	158	157	156	99.36	98.73	100.00	99.37
SX9040	150	150	149.17	99.44	99.44	100.00	100.00
DEL311	145	143	120.17	84.03	82.87	100.00	98.62
FSB290	156	145	140.33	96.78	89.96	100.00	92.95
FHR274	147	147	147	100.00	100.00	100.00	100.00
LEVIT8	117	115	115	100.00	98.29	100.00	98.29
%				96.59%	95.51%	98.96%	97.82%

\*Glossary of column headings.

LP	License plate characters.
T	Total number of frames in the video.
TD	Total number of frames with detected plates.
AvgTR	Average of total of all characters recognized correctly.
AvgTR/TD	Average of total of all characters recognized correctly / total number of frames with detected plates.
AvgTR/T	Average of total of all characters recognized correctly / total number of frames.
F/TD	Average of total of all characters recognized correctly after applying the frequency / total number of frames with detected plates.
F/T	Average of total of all characters recognized correctly after applying the frequency / the total number of frames.

## 5.4 Number of frames for correct character recognition

Even though in the previous step, character recognition results has been discussed, this stage will show how many frames are needed to recognize all the characters from a plate correctly at the first time. Table 5.4-1 shows that in 11 videos all the license plate characters have been recognized correctly at the first frame. Two other plate characters have been recognized correctly for the first time at the third and fourth frame. At the fifth frame, the whole plate has been recognized correctly in two other videos (Table 5.4-1). Whereas all the characters were

recognized at the first frame in the longest video, the second longest did not recognize all the plate characters correctly in the first seven frames (Table 5.4-1). By calculating the normalized average of those sixteen videos, it was found that this process would recognize all the characters correctly within 1.45 frames.

**Table 5.4-1 First frame where the plate characters were recognized correctly**

LP	Frames	
	Total	First correct recognition
FRN346	148	1
EYG796	171	8
EWJ843	267	1
EPM11	153	5
TO3145	161	5
FUE503	157	1
DSC825	155	1
GMA305	150	1
DQR871	160	1
JEANEZ	148	1
GKK709	158	1
SX9040	150	1
DEL311	145	1
FSB290	156	4
FHR274	147	1
LEVIT8	117	3

## 5.5 Summary

While, using this process, one plate took eight frames to correctly recognize all characters, the average is 1.44 frames to detect all the character correctly. The approach of reusing previous plate location and the frequency counts improve plate detection and characters recognition. Therefore, very few frames are required to recognize the plate correctly.

This was the experimental result; the future work and the conclusion will be discussed in the following chapter.

## **6 Chapter Six: Conclusion**

At the beginning of this thesis, the problem and the objective were discussed. The problem identified was the need for portable license plate recognition system in real time, which was cheap or free solutions. The objective was mainly to solve these problems by providing a new solution that recognize the license plate accurately which could be used on smart phone devices.

The second chapter of this thesis has discussed the related work. In this discussion the related work focused on the recently used techniques in license plate recognition. The related solution was divided into five frequent steps to encompass the solutions in each step. The first four steps (pre-processing, feature analysis, finding plate region and character segmentation) were part of the detection. The last step was discussing the solutions from the character recognition point of view.

The middle chapters of this thesis have investigated a new technique to recognize the vehicle license plate using smart phone devices. This solution provides an effective, appropriate and free application for private and small organization use. Primarily, this solution is based on geometrical features associated with image processing to detect the plate and prepare the detected image for the next phase (character recognition). During the detection phase, sequence of filters are applied to ensure accurate detection. From the recognition point of view, this solution has used one of the most effective open source OCRs. Tesseract is the OCR used in this solution: it was trained with New Zealand license plate font and real plates images. The license plate font after the training was found to be the most accurate trained-data for OCR for the purpose of plate recognition.

Chapter Five in this thesis illustrated and discussed the experimental results. The experimental results have proven the high accuracy and effectiveness of the solution in license plate detection and recognition. It was found that the final plate detection solution gave 98.90 percent correctly detected plate. The incorrectly detected plates were only 0.08 percent; in the remaining percentage (1.20%) there was no plate detected. The percentage of all characters recognized correctly out of the detected plates was 98.96 percent.

## **6.1 Strengths**

The strength of this solution has seven aspects. Firstly, this solution is a lightweight approach to be used on mobile phone devices. Secondly, it has a high accuracy in license plate detection; the incorrect detected regions were less than one percent (2 out of 2543). Thirdly, this solution can, with minor adjustments, be adopted in countries other than New Zealand to detect the plate. Fourthly, in the recognition phase there is no need for extensive training data for training, only the country's font to train the Tesseract. Fifthly, this system has the ability to recognize the plate regardless of the plate colour, position (high, low, cantered, left or right) and whether it is in the rear, or front. Sixthly, recording the recognition results over multiple frames has improved the results. Finally, in terms of number of camera frames and real time, the license plate characters could be recognized correctly on average within the first two frames. From the user's perspective this is almost instantaneous in most cases. The recognition rate is 98.96 percent and by referring to Table 2.7-1 this approach seems to be better than most of the other studies.

## 6.2 Future work

This thesis has investigated a lightweight approach that recognizes the license plate accurately in real time. However, a further investigation is recommended to increase the range of viewing angles from which it is effective. The recommended solution is to give the user of the phone the ability to capture the car, in order to recognize the plate, from a side angle (Figure 6.2-1). One possible solution to do this is that, the technique of using the bounding rectangle will still be used but some of the values need to be modified, like width, height and aspect, this will be done in the next steps. In the filtering stage, after the candidate region has been determined as plate then perspective transform has to be applied. Perspective transform will use the four corner points of the bounding box (candidate plate) in order to generate new corner points which transform the bounding box to a rectangle. Using the transformed rectangle and features, this plate will then go through the rest of the detection and recognition processes. This possible solution will enable the system to recognize the plate for a car that is captured from a side angle and would not have much effect on the straight captured video license plate recognition because the determined region here is in a rectangular shape therefore; the transform here will make a minimal transform.

In this thesis approach only recognizes one vehicle license plate per frame. Therefore, the second recommendation is that, to extend this approach to be able to recognize more than one car per frame. A possible solution to do this is to keep a list of the best potential plate regions and provide them to the OCR to be recognized.

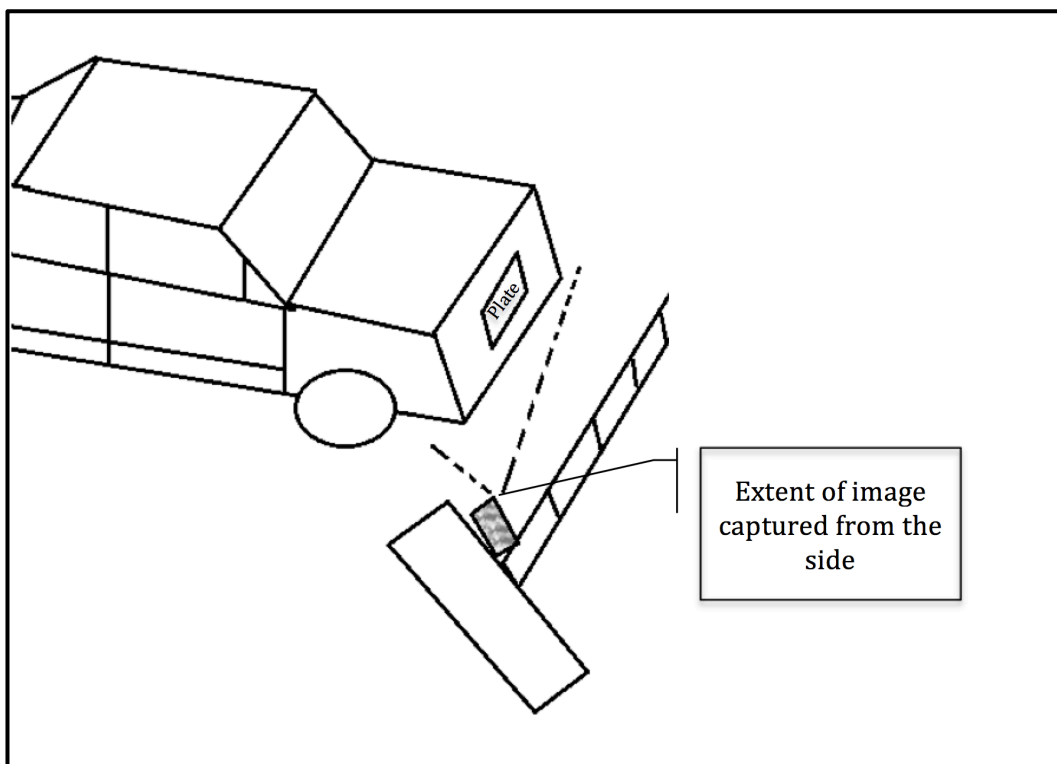


Figure 6.2-1 diagram of how the car will be captured from the side

## 7 References

- Abolghasemi, V., & Ahmadyfard, A. (2009). An edge-based color-aided method for license plate detection. *Image and Vision Computing*, 27(8), 1134-1142.
- Alginahi, Y. M. (2011). Automatic Arabic License Plate Recognition. *International Journal of Computer and Electrical Engineering*, 3(3), 454-460.
- Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Incorporated.
- Breuel, T. M. (2008). The OCRopus open source OCR system *In Electronic Imaging 2008 . International Society for Optics and Photonics*. (pp. 68150F-68150F): doi:10.1117/12.783598
- Buck, D. (2006). *Number plate fonts of Australasia: New Zealand*. Retrieved from <http://www.leewardpro.com/articles/licplatefonts/font-licenz-nz.html>
- Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 679-698. doi:10.1109/TPAMI.1986.4767851
- Chen, C.-H., Chen, T.-Y., Huang, C.-M., & Wang, D.-J. (2011). License plate location for vehicles passing through a gate *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2011 Seventh International Conference on* (pp. 340-343): IEEE.
- Gazcón, N. F., Chesñevar, C. I., & Castro, S. M. (2012). Automatic vehicle identification for Argentinean license plates using intelligent template matching. *Pattern Recognition Letters*
- Ghosh, K., Sharma, S. K., Islam, M. N., Biswas, S., & Akter, S. (2011). Automatic License Plate Recognition (ALPR) for Bangladeshi vehicles. *Global Journal of Computer Science and Technology*, 11(21)
- Gordon, A., & Wolf, R. (2007). License Plate Recognition Technology: Innovation in Law Enforcement Use. *FBI Law Enforcement Bulletin*, 76(3), 8-13.
- Huang, Y.-P., Chang, T.-W., Chen, Y.-R., & Sandnes, F. E. (2008). A back propagation based real-time license plate recognition system. *International Journal of Pattern Recognition and Artificial Intelligence*, 22(02), 233-251.
- Huang, Y.-P., Chen, C.-H., Chang, Y.-T., & Sandnes, F. E. (2009). An intelligent strategy for checking the annual inspection status of motorcycles based on license plate recognition. *Expert Systems with Applications*, 36(5), 9260-9267.

- 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.
- Jiao, J., Ye, Q., & Huang, Q. (2009). A configurable method for multi-style license plate recognition. *Pattern Recognition*, 42(3), 358-369.
- jTessBoxEditor (Version 2) [Recognition software]: Apache License. Retrieved from <http://sourceforge.net/projects/vietocr/files/>
- Juntanasub, R., & Sureerattanan, N. (2005). Car license plate recognition through Hausdorff distance technique *Tools with Artificial Intelligence, 2005. ICTAI 05. 17th IEEE International Conference on* (p. 5 pp.): IEEE.
- Kong, J., Liu, X., Lu, Y., & Zhou, X. (2005). A novel license plate localization method based on textural feature analysis *Signal Processing and Information Technology, 2005. Proceedings of the Fifth IEEE International Symposium on* (pp. 275-279): IEEE.
- Leape, J. (2006). The London Congestion Charge. *Journal of Economic Perspectives*, 20(4), 157-176. doi: 10.1257/jep.20.4.157
- Liaqat, A. G. (2011). *Mobile real-time license plate recognition* (Unpublished master's thesis). Linnaeus University, Kalmar, Sweden.
- Maglad, K. W. (2012). A Vehicle License Plate Detection and Recognition System. *Journal of Computer Science*, 8(3), 310-315.
- Sedighi, A., & Vafadust, M. (2011). A new and robust method for character segmentation and recognition in license plate images. *Expert Systems with Applications*, 38(11), 13497-13504.
- Sheldon, B. (2011). Camera surveillance within the UK: Enhancing public safety or a social threat? *International Review of Law, Computers & Technology*, 25(3), 193-203. doi:10.1080/13600869.2011.617494
- SIL International. (2012). *SIL International*. Retrieved from <http://www.sil.org/>
- Smith, R. (2007). An Overview of the Tesseract OCR Engine. *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on* (pp. 629-633). Curitiba, Parana, Brazil: IEEE.
- Songke, L., & Yixian, C. (2011). *License plate recognition*. University of Gävle. Retrieved from <http://hig.diva-portal.org/smash/record.jsf?pid=diva2:422755>
- Tsai, I.-C., Wu, J.-C., Hsieh, J.-W., & Chen, Y.-S. (2009). Recognition of vehicle license plates from a video sequence. *IAENG International Journal of Computer Science*, 36(1), 26-33.
- Waterhouse, N. (2006). *Vehicle license plate recognition on mobile devices* (Unpublished master's thesis). University of Waikato, Hamilton, New Zealand.

Yoon, H.-S., Lee, H.-C., & Lee, J.-Y. (2009). Automatic number plate detection for Korean vehicles *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human* (pp. 1191-1195): ACM.