淡江大學資訊工程學系碩士在職專班

碩士論文

指導教授：洪文斌 博士

雲端車牌辨識系統之 Android App 設計

Designing Android App of Vehicle License Plate Recognition System Based on Cloud Computing

研究生：劉冠宏 撰

中華民國 101 年 6 月

誌　謝

能順利的完成論文，首先要感謝的是我的指導教授洪文斌老師，在這兩年求學過程中細心的教誨，學生才能夠克服初來時的徬徨無知，一步一步朝著全然陌生的影像處理這領域邁進，讓學生瞭解真正求學的意義與應有的態度，研究時發現，陌生的定義只是在於沒有真正試著去瞭解他，一些影像處理的原理，其實已經於高中甚至國中即接觸過，在影像處理中就是將這些原理以更生動、視覺性化的方式呈現，所知的技術理論不一定要很深奧，但是如何讓技術理論實用化是值得探討學習。更要感謝口試委員謝文恭老師與陳建彰老師，不辭辛勞給予學生論文許多的寶貴意見與指導。

在取得碩士過程中，公司主管張雅芬女士及承辦人朱弘如女士的大力支持，使學生能夠在求學與工作上順利並行，在課業上能追求自己感興趣的研究，工作上亦能得心應手並行，這樣的我何其幸運！不但讓我有時間培養另一方面的專長，又能面對現有競爭力的挑戰，畢業後更能投入職場面對不同挑戰。更要感謝華碩李政哲博士、華亞科技李价剛特助、衛展資訊朱家佑專案經理與承辦人李明珠女士推薦，讓我有機會逐漸朝向我的夢想實現。

最後，感謝我的父親劉忠銘先生、母親李美鶯女士、妹妹劉怡君，這兩年間家裡發生了許多事，因為你們一路的支持與愛護，我才可以無後顧之憂的完成學業，感謝。

劉冠宏　謹誌

101 年 6 月

論文名稱：雲端車牌辨識系統之 Android App 設計　　頁數：65

校系(所)組別：淡江大學資訊工程學系碩士在職專班

畢業時間及提要別：100 學年度第 2 學期碩士學位論文提要

研究生：劉冠宏　　指導教授：洪文斌 博士

論文提要內容：

智慧型手持裝置與生活應用的結合是近年來流行的議題，本論文主要嘗試提供一種智慧型手持裝置架構的實作，以照相與網路結合智慧型手持裝置運算功能，建置一個行動車牌辨識系統。系統主要採 Open Source 的 Android 平台裝置，及 Java 應用程式為基礎核心來建置雲端主機。手機部份提供 UI 畫面、方便的人機互動選取車牌、以及影像粗定位功能。雲端主機部份實現了 OCR 圖片處理、代登網站查詢、和資料紀錄功能。藉由雲端主機預先處理，僅將必要資訊回傳手機，以加快網路傳輸。此架構模式能有效降低機器的開發、管理、與維護升級成本，實現應用服務快速整合與資源有效利用的目標。

關鍵字：Android，車牌辨識，邊緣偵測，雲端

表單編號:ATRX-Q03-001-FM030-01

Title of Thesis: Designing Android App of Vehicle License Plate Recognition System Based on Cloud Computing

Total pages: 65

Key word: Android, License Plate Recognition, Edge Detection, Web Service

Name of Institute: Executive Master's Program, Department of Computer Science and Information Engineering, Tamkang University

Graduate date: 06/2012 Degree conferred: Master

Name of student: Kuan-Hung Liu 劉冠宏 Advisor: Dr. Wen-Bing Horng 洪文斌 博士

Abstract:

The integration of portable devices and our everyday life is a popular topic in recent years. This thesis attempts to provide an architectural implementation of portable devices, using photographs and networks to build a mobile license plate recognition systems. The system adopted the open source code of the Android platform device and Java applications in building a cloud server.

The portable device provides user interface (UI), easy selection of license plates through interaction, and rough positioning the plates via image processing algorithms. The cloud server provides OCR (optical character recognition) of license plate images, web site inquiry, and data recording. By the preprocessing of the cloud server, only necessary information will be sent back to the portable device in order to speed up network transmission. This architectural model is able to effectively reduce the cost of development, management, and maintenance. Furthermore, it also achieves the goal of rapid integration and effective utilization of resources.

目錄

第一章　緒論.....1

1.1　前言.....1
1.2　研究動機.....1
1.3　研究目的.....2
1.4　論文內容大綱.....2

第二章　相關理論.....4

2.1　Android 作業系統.....4
2.2　車牌定位.....5
2.3　車牌文字切割.....8
2.4　旋轉影像.....9
2.5　車牌辨識.....10
2.6　雲端服務.....11
2.7　HTML 解析.....13

第三章　系統架構與系統之設計.....16

3.1　智慧型手持裝置端.....16
3.1.1　手持設備照相.....18
3.1.2　手持設備車牌定位.....18
3.1.3　手持設備與雲端系統介接.....25
3.1.4　手持設備呈現雲端主機結果.....25
3.2　雲端主機服務.....27
3.2.1　雲端主機 Web Service.....29
3.2.2　雲端主機車牌辨識.....31
3.2.3　雲端主機代登入查詢.....37

第四章　系統實作.....41

4.1　系統環境.....41
4.1.1　手持裝置端.....41
4.1.2　雲端主機端.....44
4.2　用戶端程式介面說明.....45
4.3　實驗結果.....50

第五章　未來展望及討論.....53

參考文獻.....54

附錄－英文論文.....56

圖目錄

圖 2-1 利用 Sobel 與固定窗格定位車牌(圖片摘自[1])....6
圖 2-2 非彩色資訊圖(圖片摘自[2])....6
圖 2-3 群組化與侵蝕膨脹處理圖(圖片摘自[2])....7
圖 2-4 矩形區塊圖(圖片摘自[4])....7
圖 2-5 First-pass 結果圖....9
圖 2-6 旋轉變換....10
圖 2-7 瀏覽器主要架構(圖片摘自[6])....14
圖 3-1 系統架構圖....16
圖 3-2 智慧型手持裝置端架構圖....17
圖 3-3 智慧型手持裝置影像圖....19
圖 3-4 人機互動選取車牌範圍圖....19
圖 3-5 灰階影像圖....20
圖 3-6 二值化影像圖....20
圖 3-7 車牌水平投影圖....21
圖 3-8 車牌垂直投影圖....21
圖 3-9 複雜車牌影像圖....22
圖 3-10 車牌投影圖....22
圖 3-11 車牌連通區塊標記圖....23
圖 3-12 車牌粗定位位置圖....24
圖 3-13 車牌粗定位綠色矩形框圖....24
圖 3-14 手持裝置接收雲端主機之資料格式圖....27
圖 3-15 雲端主機架構圖....28
圖 3-16 雲端主機主機端 Web Service 圖....30
圖 3-17 雲端主機連通區塊標記圖....32
圖 3-18 雲端主機車牌字元區塊圖....32
圖 3-19 雲端主機車牌傾斜角判斷圖....33
圖 3-20 雲端主機車牌字元區塊圖....33
圖 3-21 Tesseract 訓練模式圖(圖片摘自[9])....34
圖 3-22 車牌字型圖....35
圖 3-23 車牌樣本圖....36
圖 3-24 車牌樣本字元標記圖....36
圖 3-25 車牌辨識圖....37
圖 3-26 車牌查詢相關網站圖....38
圖 3-27 環保署機車定期檢驗圖....38
圖 3-28 機車定期檢驗有資料圖....39
圖 3-29 機車定期檢驗無資料圖....39
圖 3-30 HTML DOM(圖片摘自[11])....40

圖 4-1 Samsung GALAXY Tab 10.1 畫面 41
圖 4-2 Opensignalmaps 統計畫面(圖片摘自[12]) 43
圖 4-3 Eclipse 畫面 44
圖 4-4 手機拍照畫面 46
圖 4-5 觸碰單點選取畫面 47
圖 4-6 觸碰兩點選取後畫面 47
圖 4-7 重新觸碰選取畫面 48
圖 4-8 雲端主機有資料畫面 48
圖 4-9 雲端主機無資料畫面 49
圖 4-10 雲端主機詳細資料畫面 49

表目錄

表 2-1 Android 作業系統版本更新參考[3] 4
表 2-2 連通元件法 8
表 2-3 各數字間相等關係 9
表 3-1 手持裝置傳送與雲端主機介接資料格式 25
表 3-2 手持裝置接收雲端主機之資料格式 25
表 3-3 查詢明細資料格式 26
表 4-1 手持裝置規格 42
表 4-2 雲端主機規格 44
表 4-3 用戶端按鈕功能 45
表 4-4 車牌辨識統計表 50
表 4-5 字元辨識統計表 50
表 4-6 辨識成功結果 51
表 4-7 辨識失敗結果 52

第一章　緒論

1.1　前言

近幾年來，因為科技的進步，智慧型手持裝置已由高昂的價格降低成大部份民眾可以負擔的選擇，加上政府近年來積極推行 WiFi 無線上網，去年台北市政府所規劃提供的 Taipei Free(臺北公眾區免費無線上網)，在訊號範圍內就可以免費上網，企圖打造一個網路台北城，但是，在如今行動網路的基礎環境愈趨成熟的情況下，與之相應的加值服務仍尚待發展，若能透過智慧型手持裝置與網路服務做一個結合，打造智慧服務，則能讓整個行動通訊的雲端服務變得更為全面化、生活化、在地化。

1.2　研究動機

現代化的都市必須有健全的交通網，在都市內最常見的交通工具就是汽車及機車，與之最為有關的應用就是車牌辨識的技術，其範圍像是停車場管理、車輛管制、交通監控、超速違規和贓車查緝，這些都與龐大的有車族群息息相關，其中最常見的是停車收費，在市區內收費停車員以 PDA 進行停車費開單，但這樣的機器使用方式非常繁瑣，且將功能綁定於某一機器上，生產成本不易降低，資料更新不易，未來功能變動需要升級或廠商不再維護時，機器可能就需汰換，收

費人員需重新適應，因此一個易上手，移植性高容易升級的車牌資料查詢軟體則顯得重要。

在收費停車員查詢這一個過程，是否可以不僅僅是收費開單，使用一鍵照相功能，針對車輛做車牌照相，並利用智慧型手持裝置及雲端網路，做一整合性的車牌 OCR、資料查詢及資料蒐集功能，使得停車收費這一動作不僅僅是收費，而是一個類似整合性的加值查詢，在做收費時候可同時至雲端主機查詢是否為贓車、有做排氣定檢等。

1.3　研究目的

綜合上一節的研究動機，本研究目的提出一個運用於智慧型手持裝置之雲端車牌辨識系統，透過網路服務與線上主機做一介接，即時查詢資料，採用的為 Android 系統，其開放源始碼的特性，及高市佔率，期能在不需要特定手機的情況下，結合手機與網路主機端，創造一智慧型手持裝置辨識平台，利用智慧型手持裝置的照相功能擷取車牌，方便的人機互動選取車牌粗定位及前置影像處理功能，再透過雲端主機實現 OCR 處理及網頁代登功能。

1.4　論文內容大綱

本論文共分為五章，各章節說明如下:

第一章:緒論，本章共三節，介紹研究動機，目的與論文架構。

第二章:相關理論，針對此研究運用到的相關資料及文獻做一介紹。

第三章:系統架構與此系統之設計，介紹系統設計與實作時所碰到的問題與解決方法。

第四章:系統實作，系統介紹及實作探討。

第五章:未來展望及討論。

TAMKANG UNIVERSITY

第二章　相關理論

2.1　Android 作業系統

為以 Linux 為基礎的作業系統，主要用於智慧型手持設備，由 Google 開發與領導。並提供 Android SDK「Android Software Development Kit」，供開發者以 Java 開發 Android 軟體。2011 年 8 月，Android 操作系統在全球智慧型手機操作系統的市場佔有率已達 48%，成為全球第一大智慧型手機操作系統。

至此系統完成時，Android 版本已至 4.0。如表 2-1 所示，列出與此系統照相功能、觸碰操作、顯示方面相關的 Android 作業系統版本更新參考。

表 2-1　Android 作業系統版本更新參考[3]

Platform Version	**API Level**	**Improvements**
Android 1.5	3	Faster Camera start-up and image capture
Android 2.0	5	(1)Built-in flash support (2)Digital zoom (3)Scene mode (4)White balance (5)Color effect

		(6)Macro focus
Android 2.3	9	(1) Includes a variety of improvements across the system that make common operations faster and more efficient for all applications。 (2) The plaform now handles touch and keyboard events faster and more efficiently, minimizing CPU utilization during event distribution。 (3) The platform now supports extra large screen sizes。

2.2　車牌定位

車牌定位是希望能透過影像處理的方式由影像中找尋出車牌的位置，進而做後續的車牌辨識，但往往因影像中的背景雜訊、車牌汙損、相似區域等干擾，增加了辨識出車牌位置的難度，車牌定位的研究，其實就如同人眼如何判斷出是否為車牌，依據的是車牌的特徵，在矩形範圍內有著英數字混合的區塊、車牌的背景顏色與字體等特徵結合起來，才認定出這就是車牌。

Chen[1]利用 Sobel Filter 強化影像中的垂直邊界，後再以固定窗格掃描，在邊緣密集處定位出車牌位置，如圖 2-1 顯示。

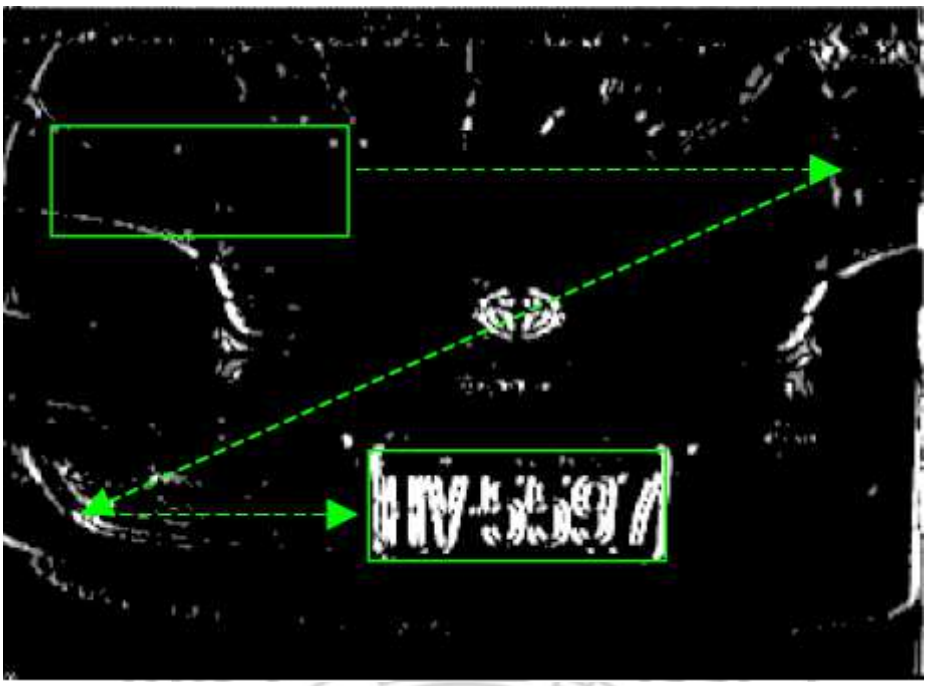


圖 2-1　利用 Sobel 與固定窗格定位車牌(圖片摘自[1])

Ye [2]利用車牌影像之邊緣與 HIS 轉換後之影像，混合取得非彩色資訊圖，如圖 2-2 顯示。接續做群組化後與侵蝕膨脹處理，如圖 2-3 顯示，再依特徵值投影來篩選車牌區域。

圖 2-2　非彩色資訊圖(圖片摘自[2])

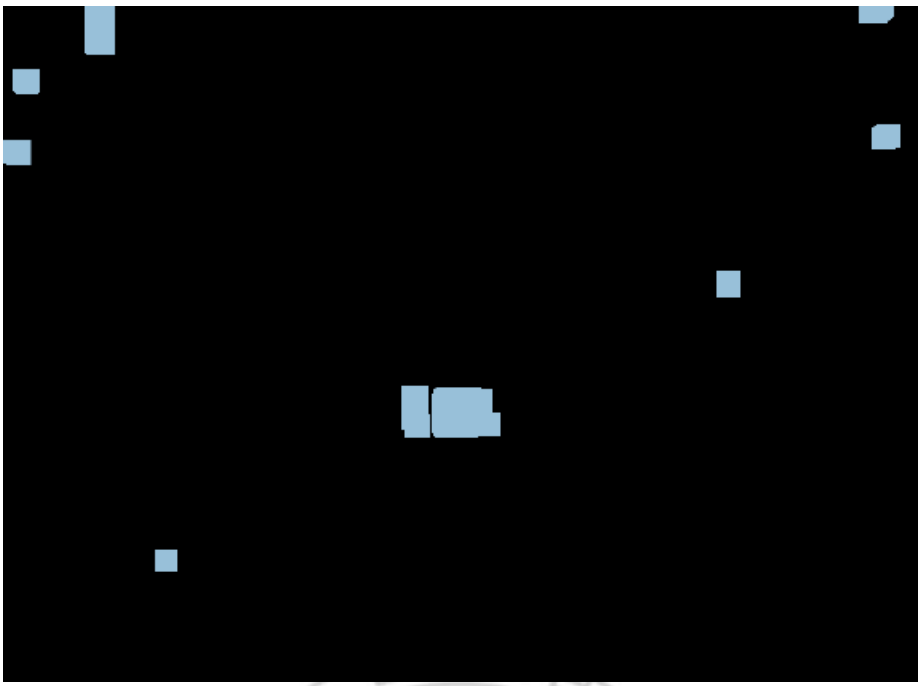


圖 2-3　群組化與侵蝕膨脹處理圖(圖片摘自[2])

Bellas 等人[4]利用 Connected Components Labelling 與 Bounding Box，找尋出矩形區塊，如圖 2-4 顯示。最後依區塊特性找尋出車牌所在位置。

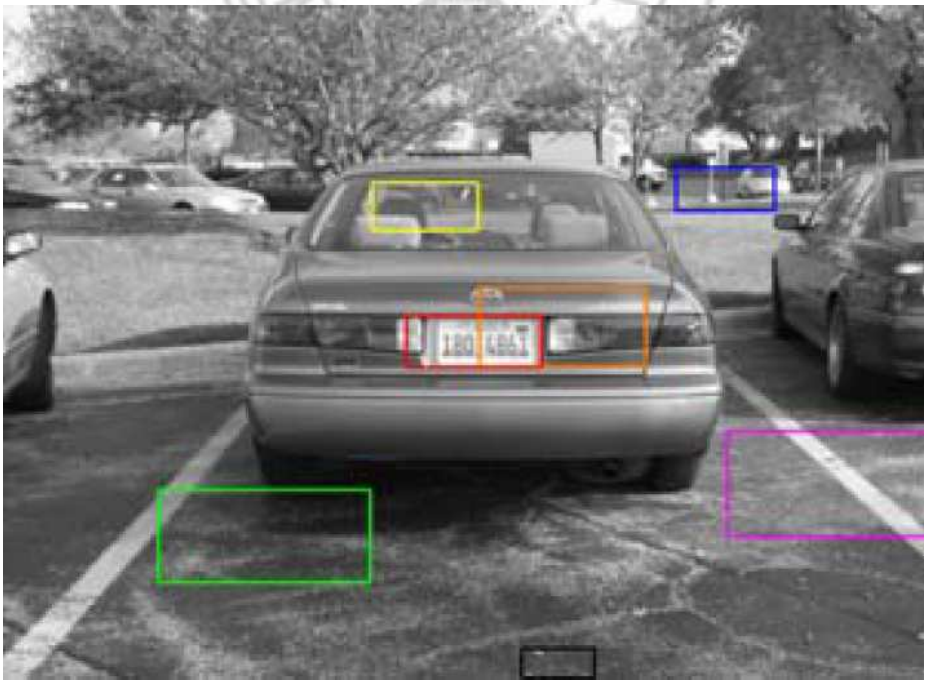


圖 2-4　矩形區塊圖(圖片摘自[4])

2.3　車牌文字切割

在取得車牌位置後，要辨識車牌字元，首先需要做的即是切割出車牌內的每個字元，一般最常見的方式為投影法，對於二值化後的車牌影像做水平投影與垂直投影，依字元內的區域投影量及字元間的間隔投影量找出特徵值輔以一些影像處理方式做切割，但此方式對於傾斜的車牌須先做旋轉校正的動作。

而另一種方式則為連通元件法(Connected-component labeling)，將影像經過二值化後分為前景及背景，做運算後可得相連的前景區塊，優點為對傾斜的車牌也不受影響，連通元件法通常可分為四連通及八連通，在此以四連通 two-pass 演算法為例。

假設 p 為目前像素，u 與 l 分別為其上及左之像素，如表 2-2 所示。

表 2-2　連通元件法

	u	
l	p	

1. first-pass 開始將影像由左至右，由上而下掃描:

   (1) 若 p 為背景，則繼續向下掃描。

   (2) 若 p 為前景，且 u 與 l 為背景則給與 p 一個新標記。

   (3) 若 p 為前景，且 u 或 l 為前景則給與 p 與前景像素一樣的標記。

   (4) 若 p 為前景，且 u 與 l 為前景，且 u=l，則標記 p=u 或 p=l。

(5) 若 p 為前景，且 u 與 l 為前景，且 u≠l，則標記 p=u 或 p=l，並註明 p=u=l。

掃描完成後，如圖 2-5 所示。而各數字間相等關係，如表 2-3 所示。

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	0	0	2	0	0
0	0	3	3	0	4	0	0	0	0	1	1	0	0	2	0	0
0	0	3	3	3	3	3	0	0	0	1	1	0	0	2	0	0
0	0	3	3	3	3	0	0	0	5	5	5	5	5	5	5	0
0	6	6	6	6	6	0	0	0	5	5	5	5	5	5	5	0
0	6	0	0	0	6	0	0	0	0	5	5	0	0	5	0	0
0	6	0	0	0	6	0	0	0	0	5	5	0	0	5	0	0
0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

圖 2-5　First-pass 結果圖

表 2-3　各數字間相等關係

Key 值	**相等之標記**
1	1,2.5
2	1,2,5
3	3,4,6
4	3,4,6
5	1,2,5
6	3,4,6

2. two-pass 做的即是將相等的標記做重新標記，使相同區塊內之數字為一致。

2.4　旋轉影像

於定位出車牌後，可能出現車牌歪斜的現象，關於歪斜車牌的處理原則為找出歪斜的角度，再做旋轉，如何找出歪斜的角度有許多研究，但於旋轉方面所用

的原理其實就是線性轉換(linear transformation)的方法，如圖 2-6 所示。將平面上的任一點 T(x,y)繞原點旋轉至 T’(x’,y’)。

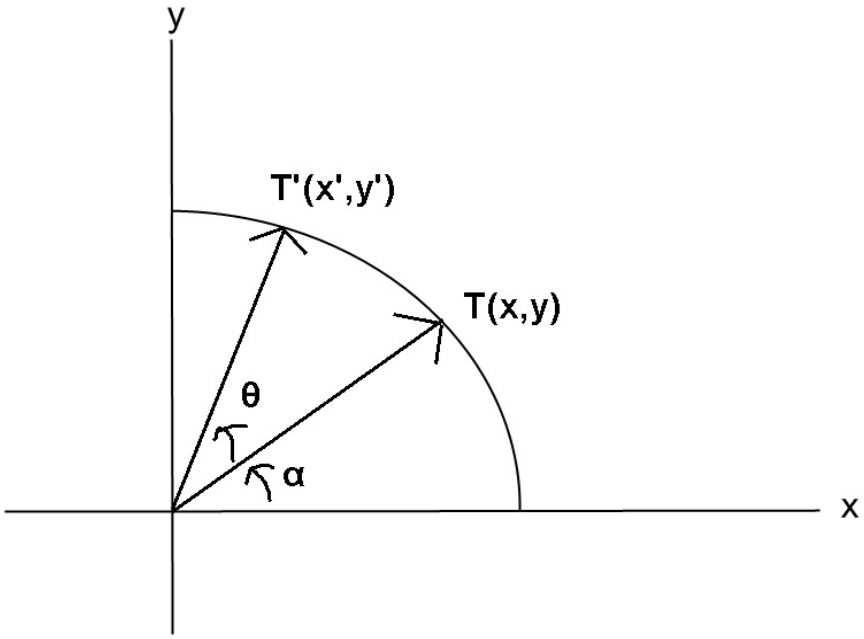


圖 2-6　旋轉變換

使用公式

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta - \sin\theta \\ \sin\theta \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2-1}$$

取得 T’(x’,y’)內之 x’= xcosθ-ysinθ，y’= xsinθ+ycosθ。依此方式可取得旋轉後之像素(x’, y’)與旋轉前像素(x, y)之對應。

2.5　車牌辨識

現行的車牌辨識的方式有許多種，機率神經網路（probabilistic neural network）、倒傳遞類神經網路(backpropagation)、支援向量機(support vector machine) 、多種分類器合併使用、樣板比對等方法，其中以樣板比對法具有程

式撰寫簡單且計算速度快等特性，由於車牌的字型固定，因此利用樣板比對法即可達到不錯的辨識率，Comelli 等人[5] 採用樣板比對法來做車牌辨識，先建立樣板字型，將樣板與待測字型以下方公式運算:

$$C_{fg} = \frac{\sum\sum(f-\overline{f})(g-\overline{g})}{\sqrt{\sum\sum(f-\overline{f})^2(g-\overline{g})^2}} \tag{2-2}$$

進行運算，其中待測字型與樣板大小是固定的。各符號間定義如下:

1. $\overline{g}$ 待測字型灰階平均值。
2. $\overline{f}$ 樣板灰階平均值。
3. g 待測字型灰階值。
4. f 樣板字型灰階值。
5. C_{fg} 正規化相關係數(normalized cross-correlation)值。

2.6　雲端服務

雲端服務是近年來熱門的話題，基本上不論是服務的類型，或者是服務的架構，只要能透過網際網路提供服務，讓使用者能經由網路連線使用，這就稱為雲端服務。

而常見的雲端服務內容，是透過 Web Service 的方式供提供，根據 W3C 的定義提到，Web Service 是一個軟體系統，設計用來支援網路間不同機器的互相

溝通的機制。透過 Web 通訊協定及開放式資料格式標準 WSDL、SOAP、XML、HTTP 等，來為其它的應用程式（或使用者）提供服務。略述如下:

1. WSDL：全名為 Web Services Description Language，是一個描述 Web Service 發佈的 XML 格式，定義服務提供之資源、使用的通訊協定、提供的操作、輸入輸出的訊息等。
2. SOAP：全名為 Simple Object Access Protocol，是一種以 XML 為基礎的訊息交換機制，以 XML 的方式來達成檔案、訊息交換與遠端程序呼叫。可以運行在 SMTP、HTTP、HTTPS 上。
3. XML：全名為 Extensible Markup Language，是一種標記語言。利用標記來定義各種屬性。是 SGML 簡化的子集，能夠描述各式不同的資料，主要用於跨平台傳遞資料、特別是能藉網路連結的平台。
4. HTTP : 全名為 Hypertext Transport Protocol，一種通訊協定，用於在網路上傳輸資訊，為網頁最常使用的通訊協定。

實現 Web Service 的服務架構一般有 RPC，SOA，REST，描述如下:

1. RPC : 全名為 Remote Procedure Call，是一種比較傳統的方式。通常在 WSDL 中對 RPC 介面進行定義，但有過於緊密耦合的缺點。因 RPC 之服務是利用一個簡單的映射，將用戶請求直接轉化成為一個特定語言編寫的函數或方法。
2. SOA : 全名為 Service-oriented Architecture，服務導向架構，是一種構造分散

式系統的方法，將服務元件化，服務間保持鬆散耦合，著重於種標準與互動性，使得服務提供者建立的服務可以被其他不同平台的需求者使用。

3. REST : 全名為 Representational state transfer。透過 HTTP 協議提供的 GET、POST、PUT 和 DELETE 來操作資源，連接協議具有無狀態性的特質。由資源的角度來觀察整個網路，分佈在各處的資源由 URI 確定，客戶端的應用通過 URI 來獲取資源。只要客戶端能夠模擬 HTTP 請求，通過標準的 HTTP 動作，都可以進行訪問。

2.7 HTML 解析

於網際網路上各式的網站提供著許多豐富的內容，供人們瀏覽。而一般最常見的是透過瀏覽器來瀏覽網頁，但是除非資料來源網站能夠主動提供資料存取 Web Service，否則，要不經由瀏覽器擷取出網頁內感興趣的內容是一件很困難的事情。

想想一般瀏覽器，能夠將網頁上的資訊，呈現出漂亮的頁面給使用者觀看，其實是透過瀏覽器分析解譯，將 HTML 做排版，而且因為 HTML 語法的不嚴謹性，瀏覽器還須設法將不合規格的 HTML 文件，轉換成瀏覽器能夠解讀的格式，瀏覽器運作流程，如圖 2-7 所示。

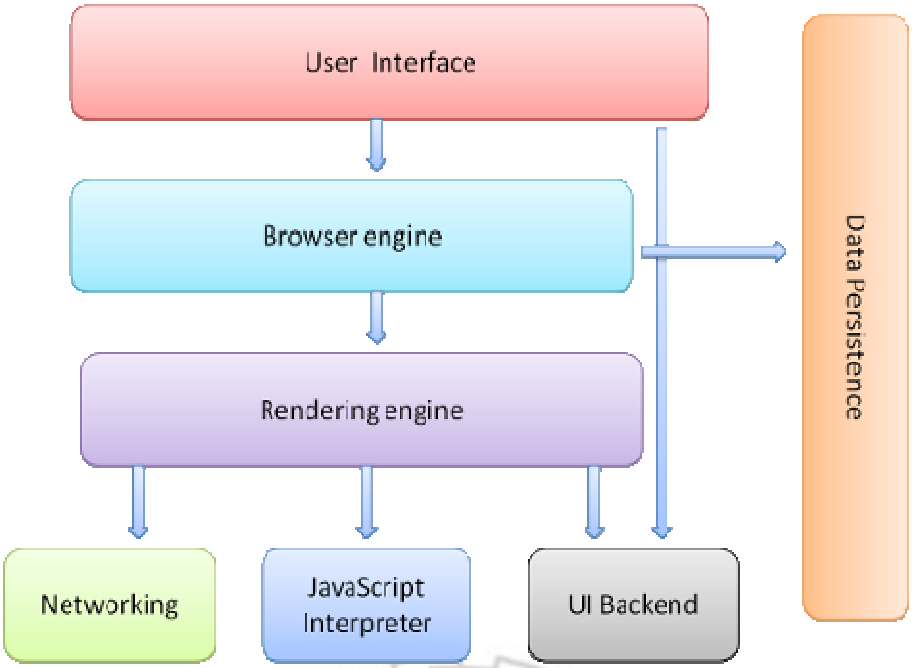


圖 2-7　瀏覽器主要架構(圖片摘自[6])

1. User Interface：即瀏覽器的操作介面，如網址欄輸入，上一頁，下一頁，我的最愛等。
2. Browser engine：即瀏覽器引擎，查詢與操作排版引擎的介面。
3. Rendering engine：即排版引擎。將 HTML 與 CSS 做排版並顯示於畫面上。
4. Networking：用於網路連結，如 HTTP 請求。
5. JavaScript Interpreter：用於解譯 JavaScript。
6. UI Backed：繪製簡易的元件，如 combo boxes and windows，不限於特定平台，可使用作業系統內用戶界面的方法。

以上是瀏覽器的主要架構，可知瀏覽器的解譯分析 HTML 是透過排版引擎，而現行有許多 Open Source 的排版引擎，如 Mozila Firefox 採用的 Gecko、Google Chrome 採用的 WebKit 等。

但是伺服器端如果不透過排版引擎核心來解譯 HTML，一般可使用 Open Source 的 HTML Parser，如 HTML Parser、Jtidy、NekoHtml 等，將 HTML 轉為 HTML DOM，再使用 Xpath（XML Path Language），一種基於 XML 的樹狀結構的語言，提供在樹中找尋節點的能力，由此來找出感興趣的內容，可採用來當作小型查詢語言使用。

第三章　系統架構與系統之設計

本系統可分為兩個部份，第一部份為智慧型手持裝置端，利用手持裝置取得車牌影像，將車牌做定位處理，第二部份為雲端主機端，利用 Web Service 與手持裝置達到一整合性的代登入查詢服務，圖 3-1 為系統架構圖。

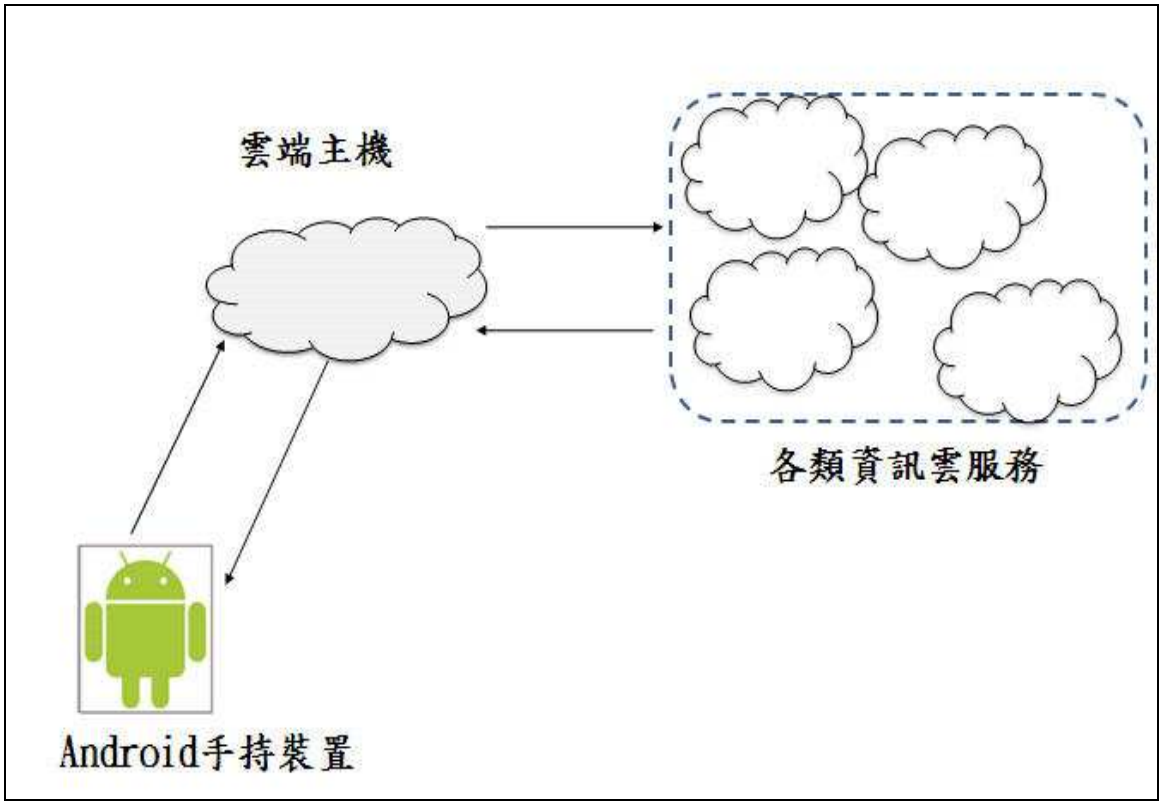


圖 3-1　系統架構圖

3.1　智慧型手持裝置端

透過智慧型手持裝置照相後，將影像做一些前置的影像處理，定位出車牌的位置，分割後傳至雲端主機。圖 3-2 為智慧型手持裝置端架構圖。

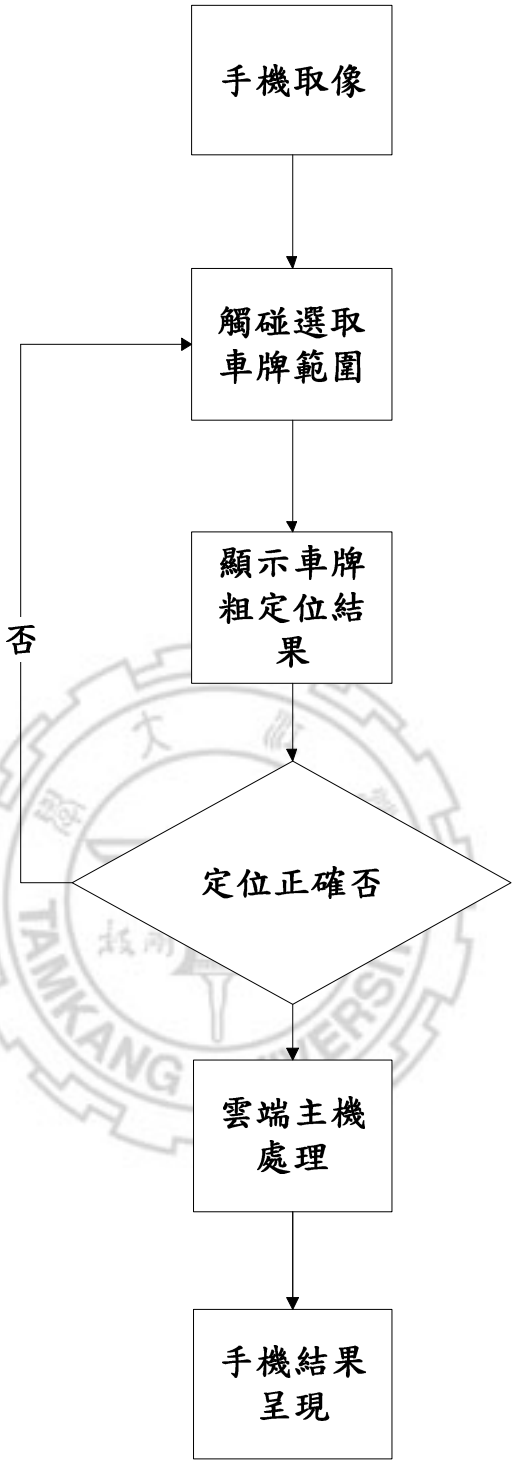


圖 3-2　智慧型手持裝置端架構圖

3.1.1　手持設備照相

現行手持設備照相的技術已相當成熟，像素愈來愈高，而且容易操作，只需將鏡頭瞄準畫面後按下快門鈕，即會自動完成所有的步驟，例如：逆光補償、消除紅眼、長時間曝光、重複曝光等。其中自動對焦及自動閃光更是必備的功能，因此在系統中針對拍攝出的影像無特別的針對成像品質不清晰的狀況做處理。

我們拍攝的影像，在 640x480 即可有不錯的辨識品質，故系統產生的影像設定為 640x480，如此可減少影像的大小及相對應的影像處理速度。

3.1.2　手持設備車牌定位

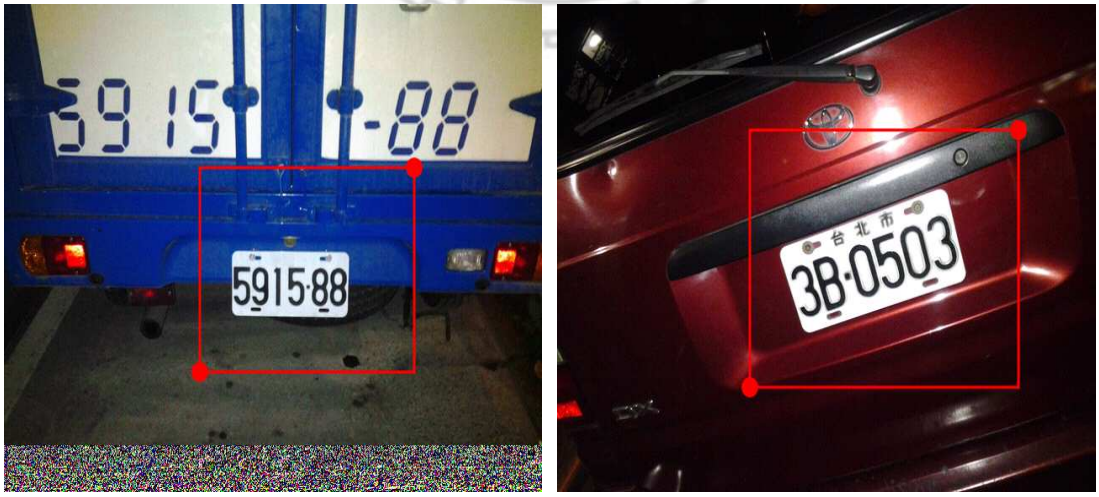
現行的研究，關於車牌定位方面可達到成功率 90%以上，於特定情況下更能達到 100%，但是否能夠減少手持裝置冗長的運算，而且藉由使用者端的確認後再將影像傳輸到雲端主機，減少不必要的查詢。圖 3-3(a)為一般標準的車牌影像，車牌未傾斜。圖 3-3(b)為車牌傾斜狀況。

(a)　　(b)

圖 3-3　智慧型手持裝置影像圖

本論文結合手持裝置觸碰功能，設計一套關於車牌定位的處理程序，步驟如下所述:

1. 於手持裝置照相後，使用者於畫面中使用觸碰的方式，同時或者分別選取 2 點，此 2 點需位於車牌左上及右下或者右上及左下，於手持裝置上將顯示兩點所組成的紅色矩形框。如圖 3-4 所示。



(a)　　(b)

圖 3-4　人機互動選取車牌範圍圖

2. 此時手持裝置也會於選取範圍內的紅框處做運算，將範圍內影像轉灰階，如

公式 3.1 所示。

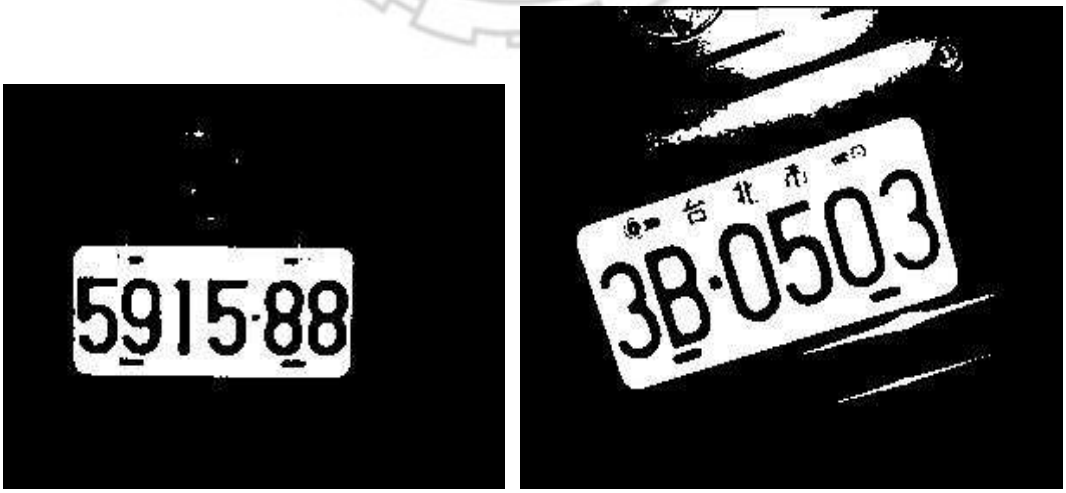
$$Gray=(Red\times0.299+Green\times0.587+Blue\times0.1114) \tag{3-1}$$

其轉換結果如圖 3-5 所示。

(a)　　(b)

圖 3-5　灰階影像圖

3. 影像二值化，因只針對選取範圍內的影像做二值化，全域二值化法即可得到較佳的速度及閥值，在此系統我們採用的為 Otsu 法，其結果如圖 3-6 所示。

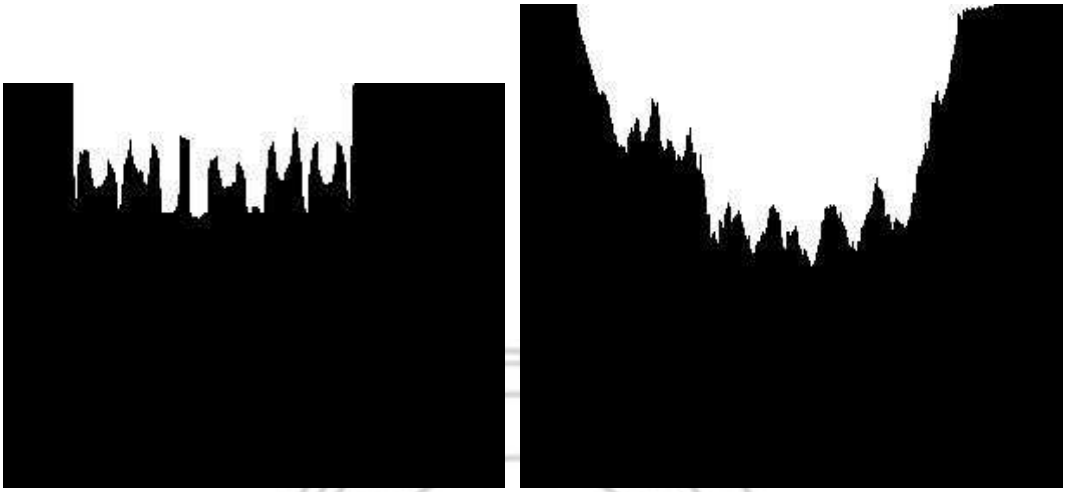


(a)　　(b)

圖 3-6　二值化影像圖

4. 車牌粗定位，由圖 3-6 可知不論水平或歪斜車牌影像，其於選取範圍內的影

像雜訊相較於整張圖片來的少，經由水平投影與垂直投影，如圖 3-7、圖 3-8 所示。可得知若將選取範圍內影像依水平及垂直投影，再加上一些閥值限制，即可找到車牌更進一步的位置。



(a)　　(b)

圖 3-7　車牌水平投影圖

(a)　　(b)

圖 3-8　車牌垂直投影圖

但若是使用者選取的範圍有較多干擾的話，如圖 3-9 所示。其水平投影與垂直投影，則會呈現較大波動，如圖 3-10 所示。

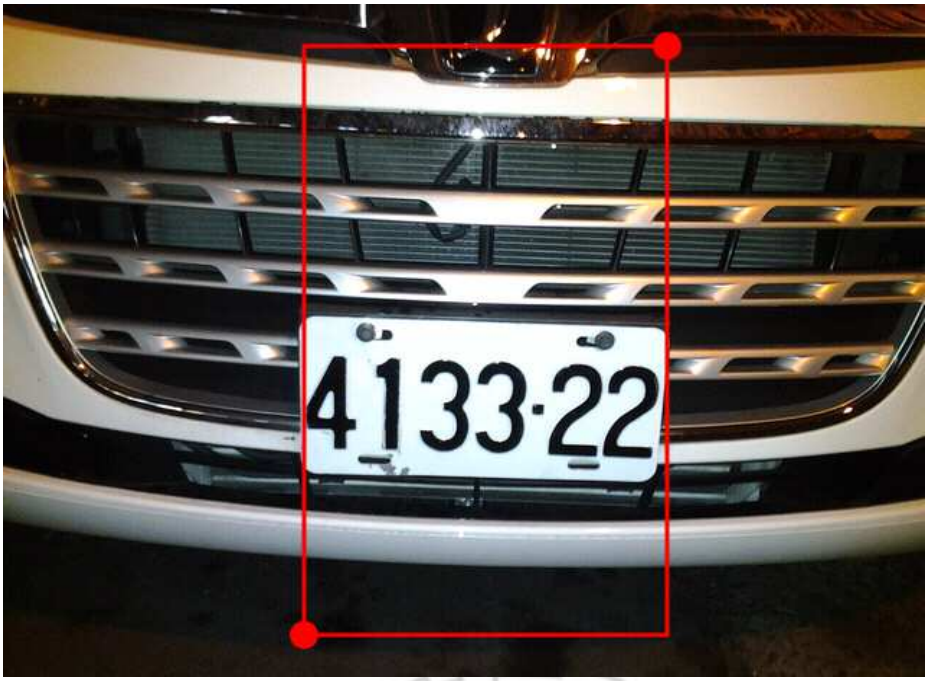
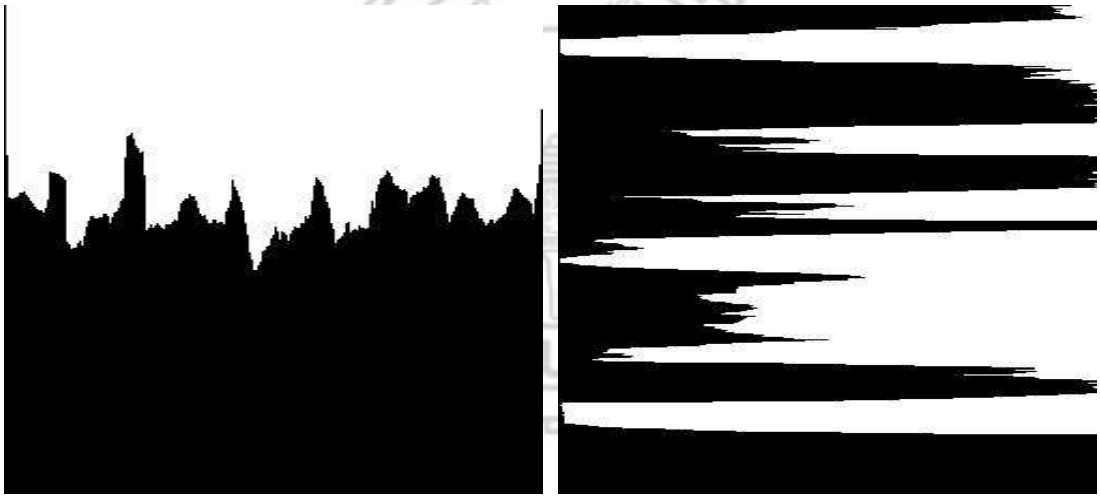


圖 3-9　複雜車牌影像圖



(a)　　(b)

圖 3-10　車牌投影圖

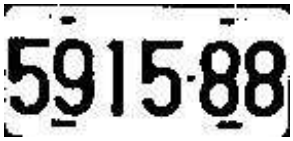
雖然仍可再加上更進一步的閥值取得車牌的約略位置，但是否有其他作法能夠簡單的達到選取範圍內找出車牌粗定位的要求，不需要設定過多的閥值，仍有車牌粗定位效果，因此我們採用的為連通元件法，將二值化後的影像，反過來以白色為前景、黑色為背景，找出影像範圍內為連接的區塊，並做標記，每個區塊取出其上、下、左、右位置，如圖 3-11 所示。最後直接將範圍內的最大區塊當做車牌粗定位結果，如圖 3-12 所示。

(a)

(b)

(c)

圖 3-11　車牌連通區塊標記圖



(a)

(b)

(c)

圖 3-12　車牌粗定位位置圖

為了能夠由使用者確認粗定位結果是否正確，再將取得選取範圍內與車牌相符的區域，取得此區域的上、下、左，右組成綠色矩形框，其結果如圖 3-13 所示。

(a)

(b)

(c)

圖 3-13　車牌粗定位綠色矩形框圖

5. 使用者同意定位結果，則按下執行，進行與雲端主機整合查詢功能，若不同意則使用觸碰的方式重新定位。

3.1.3　手持設備與雲端系統介接

在考量到手持裝置的易用性與通用性，採用的是現行手持裝置皆有提供的 HTTP 功能，以 Http Post 方式，傳送車牌影像及手持裝置相關資訊，如手持裝置機器識別碼，使用者資訊，GPS 定位資訊等。傳送相關資訊如表 3-1 所述(相關詳細說明可參照雲端主機部份)。

表 3-1　手持裝置傳送與雲端主機介接資料格式

屬性名稱	資料型態	欄位必填	屬性描述
userId	String	必填	手持裝置機器識別碼
type	String	必填	使用何種雲端主機服務
image	Byte	必填	車牌影像

3.1.4　手持設備呈現雲端主機結果

雲端主機查詢完畢後，將資料回傳給手持裝置，手持裝置接收到的為 XML 格式，手持裝置再將 XML 格式化及排版。接收之 XML 格式相關資訊如表 3-2、表 3-3、圖 3-14 所述(相關詳細說明可在參照雲端主機部份)。

表 3-2　手持裝置接收雲端主機之資料格式

屬性名稱	資料型態	欄位必填	屬性描述
license	String		車牌號碼
brandType	String		廠牌
engineCapacity	String		排氣量
strokecycle	String		行程別
birthDate	String		出廠日
useDate	String		發照日期

message	String		相關訊息
list	Array		查詢明細資料

表 3-3　查詢明細資料格式

屬性名稱	資料型態	欄位必填	屬性描述
checkPlace	String		檢測站號
checkType	String		檢測種類
checkHC	String		HC
checkCO	String		CO
checkCO2	String		CO2
checkSerial	String		序號
checkResult	String		檢測結果
checkDate	String		檢測日期時間
checkLable	String		檢測標籤

TAMKANG UNIVERSITY

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <epaAirBean>
    <license>CRN-588</license>
    <brandType>三騏</brandType>
    <engineCapacity>124</engineCapacity>
    <strokecycle>4</strokecycle>
    <birthDate>200109</birthDate>
    <useDate>20011213</useDate>
    <message>您應定檢的時間為：11 月 ~ 1 月</message>
  - <checkBean>
      <id>ken</id>
      <type>EPA_AIR</type>
      <ocrResult>CRN-588</ocrResult>
      <checkResult>true</checkResult>
      <errorMessage />
    </checkBean>
  - <list>
      <checkPlace>AH5</checkPlace>
      <checkType>定期檢驗</checkType>
      <checkHC>331.00</checkHC>
      <checkCO>1.03</checkCO>
      <checkCO2>12.42</checkCO2>
      <checkSerial>100010100</checkSerial>
      <checkResult>合格</checkResult>
      <checkDate>2011/01/25 20:23:56</checkDate>
      <checkLable>99-A395709</checkLable>
    </list>
  - <list>
      <checkPlace>AH5</checkPlace>
      <checkType>定期檢驗</checkType>
      <checkHC>323.00</checkHC>
      <checkCO>0.56</checkCO>
      <checkCO2>10.20</checkCO2>
      <checkSerial>98120084</checkSerial>
      <checkResult>合格</checkResult>
      <checkDate>2009/12/21 20:01:26</checkDate>
      <checkLable>98-A295850</checkLable>
    </list>
  - <list>
      <checkPlace>FG1</checkPlace>
      <checkType>定期檢驗</checkType>
      <checkHC>696.00</checkHC>
      <checkCO>0.28</checkCO>
      <checkCO2>9.73</checkCO2>
      <checkSerial>97110146</checkSerial>
      <checkResult>合格</checkResult>
      <checkDate>2008/11/15 18:07:45</checkDate>
      <checkLable>97-F835639</checkLable>
    </list>
  - <list>
      <checkPlace>AH5</checkPlace>
      <checkType>定期檢驗</checkType>
      <checkHC>212.00</checkHC>
      <checkCO>0.33</checkCO>
      <checkCO2>10.89</checkCO2>
      <checkSerial>49123</checkSerial>
      <checkResult>合格</checkResult>
      <checkDate>2008/01/30 21:34:25</checkDate>
      <checkLable>96-A342034</checkLable>
    </list>
  - <list>
      <checkPlace>A84</checkPlace>
      <checkType>定期檢驗</checkType>
      <checkHC>344.00</checkHC>
      <checkCO>2.21</checkCO>
      <checkCO2>13.01</checkCO2>
      <checkSerial>13705</checkSerial>
      <checkResult>合格</checkResult>
      <checkDate>2007/01/26 14:20:35</checkDate>
      <checkLable>95-A434002</checkLable>
    </list>
  - <list>
      <checkPlace>A16</checkPlace>
      <checkType>定期檢驗</checkType>
      <checkHC>24.00</checkHC>
      <checkCO>0.93</checkCO>
      <checkCO2>12.95</checkCO2>
      <checkSerial>32889</checkSerial>
      <checkResult>合格</checkResult>
      <checkDate>2006/01/24 10:53:00</checkDate>
      <checkLable>94-A025531</checkLable>
    </list>
  </epaAirBean>
```


圖 3-14　手持裝置接收雲端主機之資料格式圖

3.2　雲端主機服務

手持裝置透過網路呼叫雲端主機的 Web Service，雲端主機於接受到請求後，運用其強大的運算功能，辨識出車牌，並代為查詢車牌相關的資訊，匯整後傳回結果至智慧型手持裝置。圖 3-15 為雲端主機端架構圖。

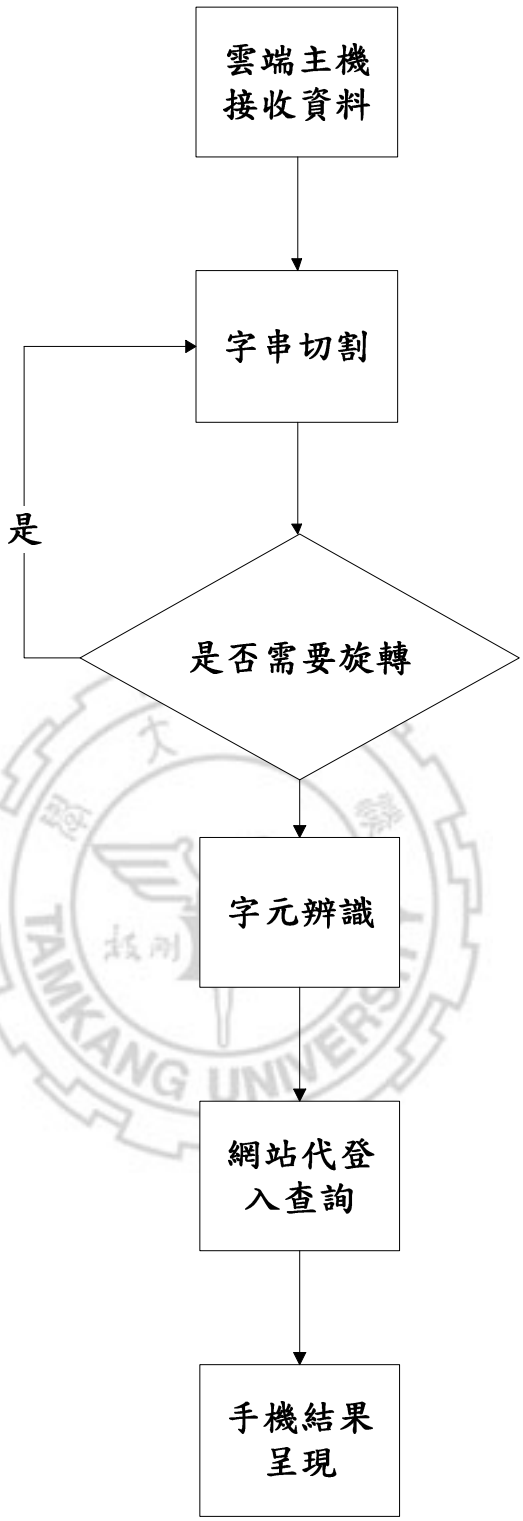


圖 3-15　雲端主機架構圖

3.2.1　雲端主機 Web Service

Apache CXF[7]是一個由 Apache 基金會開發、維護的 Web Service 服務框架，提供了各式的 Web 通訊協定及開放式資料格式標準 SOAP、XML/HTTP、RESTful HTTP、CORBA，經由 HTTP、JMS、JBI 傳輸 Web Service。

可知 CXF 是一種提供多樣化的 Web Service 服務的解決方式，目前的大型雲端服務提供者，如 Amazon、Google、Yahoo、IBM、Microsoft、Sun、GoGrid 等，都支援 RESTful 服務，甚至只提供 RESTful API。而且 Amazon 曾經對提供的兩種 Web Service 介面 SOAP 及 REST 做統計，發現使用情況以 REST 使用率較高，原因就是容易使用，雖然都是標準，但是對於 Client 端的使用方便與否，才是雲端服務的精神。

因此在雲端主機的服務，我們透過 CXF 實作基於 REST 為架構的 RESTful Web Service。提供標準的 HTTP 介接。因此可以很方便的與智慧型手持裝置介接，甚至只要能上網連接之裝置，只需符合 Web Service 規格即可與主機做一要求服務功能，如圖 3-16 所述。

```
- <application xmlns="http://wadl.dev.java.net/2009/02" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  - <grammars>
    - <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified"
        elementFormDefault="unqualified">
        <xs:element name="checkBean" type="serviceCheckBean" />
        <xs:element name="epaAirBean" type="epaAirBean" />
      - <xs:complexType name="epaAirBean">
        - <xs:sequence>
            <xs:element minOccurs="0" name="license" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" name="brandType" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" name="engineCapacity" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" name="strokecycle" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" name="birthDate" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" name="useDate" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" name="isVerify" nillable="true" type="xs:boolean" />
            <xs:element minOccurs="0" name="message" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" ref="checkBean" />
            <xs:element maxOccurs="unbounded" minOccurs="0" name="list" nillable="true" type="epaAirDetailBean" />
          </xs:sequence>
        </xs:complexType>
      - <xs:complexType name="serviceCheckBean">
        - <xs:sequence>
            <xs:element minOccurs="0" name="id" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" name="type" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" name="ocrResult" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" name="checkResult" nillable="true" type="xs:boolean" />
            <xs:element minOccurs="0" name="errorMessage" nillable="true" type="xs:string" />
            <xs:element minOccurs="0" name="errorCode" nillable="true" type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      - <xs:complexType name="epaAirDetailBean">
        - <xs:sequence>
            <xs:element minOccurs="0" name="checkPlace" type="xs:string" />
            <xs:element minOccurs="0" name="checkType" type="xs:string" />
            <xs:element minOccurs="0" name="checkHC" type="xs:string" />
            <xs:element minOccurs="0" name="checkCO" type="xs:string" />
            <xs:element minOccurs="0" name="checkCO2" type="xs:string" />
            <xs:element minOccurs="0" name="checkSerial" type="xs:string" />
            <xs:element minOccurs="0" name="checkResult" type="xs:string" />
            <xs:element minOccurs="0" name="checkDate" type="xs:string" />
            <xs:element minOccurs="0" name="checkLable" type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      </xs:schema>
    </grammars>
  - <resources base="http://localhost:7080/cameraME/service/rest/">
    - <resource path="/imageService">
      - <resource path="/diyUpload">
        - <method name="POST">
          - <request>
            - <representation mediaType="multipart/form-data">
                <param name="request" style="plain" type="xs:string" />
              </representation>
            - <representation mediaType="multipart/form-data">
                <param name="request" style="plain" type="xs:string" />
              </representation>
            - <representation mediaType="multipart/form-data">
                <param name="request" style="plain" type="xs:string" />
              </representation>
            </request>
          - <response>
              <representation mediaType="application/xml" />
            </response>
          </method>
        </resource>
      - <resource path="/imageUpload">
        - <method name="POST">
          - <request>
            - <representation mediaType="multipart/form-data">
                <param name="request" style="plain" type="xs:string" />
              </representation>
            - <representation mediaType="multipart/form-data">
                <param name="request" style="plain" type="xs:string" />
              </representation>
              <representation mediaType="multipart/form-data" />
            </request>
          - <response>
              <representation mediaType="application/xml" />
            </response>
          </method>
        </resource>
      </resource>
    </resources>
  </application>
```


圖 3-16　雲端主機主機端 Web Service 圖

3.2.2　雲端主機車牌辨識

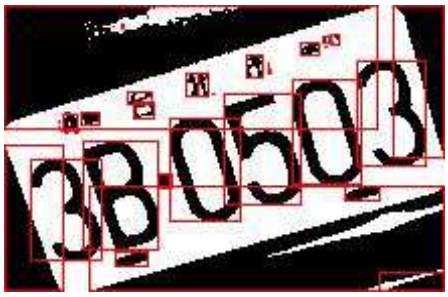
現行臺灣根據不同的車輛使用不同的車牌，交通部也已針對汽車、機車進行新式車牌規畫，未來新式車牌的大小樣式皆有可能做調整，甚至可以依民眾任意自行命名個人化的車牌，而且在體貼民眾兼具環保的政策思考下，後續新式車牌將只針對新領或重領車牌民眾進行核發。

在考量新舊車牌並行的情況下，車牌字型、長度、英數字命名方式等特徵型態將會有所不同，如何設計一個效率不錯，且未來容易針對辨識方面升級的系統是很重要的，因此我們將本系統車牌辨識部份做於雲端主機端，未來即可以很方便的透過雲端機器辨識模組更新，達到更高的辨識率。

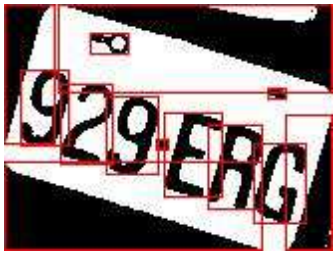
在車牌辨識處理方面，手持裝置傳送的為灰階影像，因考量手持裝置僅是輔助的功能，故於雲端仍有車牌定位與影像二值化功能。關於車牌辨識的處理程序，步驟如下所述:

1. 字串切割，在雲端主機接收到手持裝置之灰階影像後，取出車牌位置後，將影像做二值化，採連通元件法，將二值化後的影像，以黑色為前景、白色為背景，找出影像範圍內為連接的區塊找出並並標記，每個區塊取出其上、下、左、右位置，如圖 3-17 所示。

(a)



(b)



(c)

圖 3-17　雲端主機連通區塊標記圖

可得知連通元件法標記出的區塊可能含有雜訊干擾，在此我們採用閥值過濾方式，將各區塊依車牌字元特性，如長度，寬度，長寬比例，字與字之間的間隔，車牌與字元相對位置等，取出特徵值過濾車牌字元，如圖 3-18 所示。

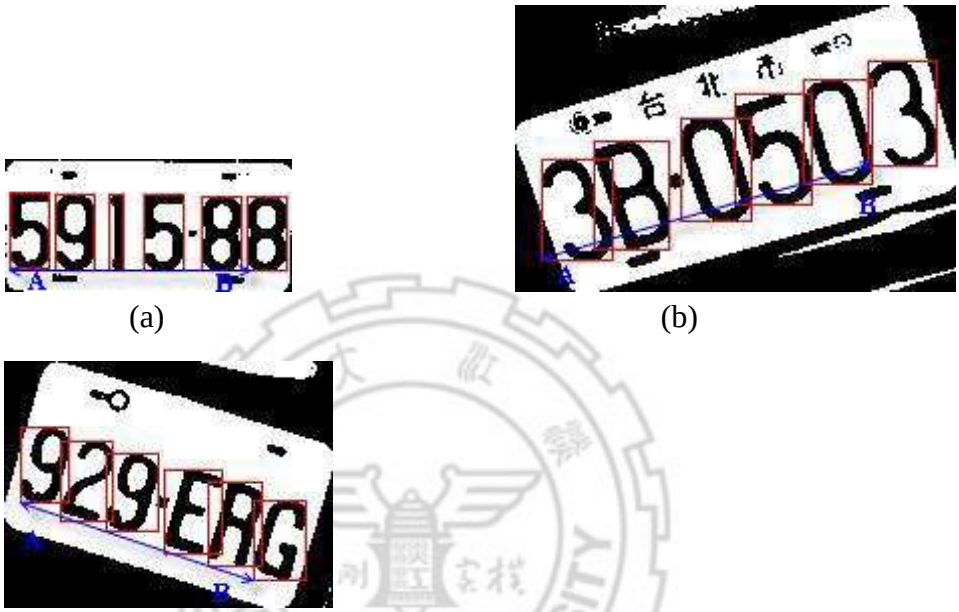
(a)

(b)

(c)

圖 3-18　雲端主機車牌字元區塊圖

2. 旋轉處理判斷，當取出車牌字元區塊後，因車牌可能會有歪斜的情況，於影像中由左至右，我們採用最左區塊與最右區塊左下角(A 與 B)相連直線，其與水平線夾角做是否傾斜超過 5 度判斷，如圖 3-19 所示。並依傾斜之角度做旋轉處理。



(a)　(b)

(c)

圖 3-19　雲端主機車牌傾斜角判斷圖

於做傾斜車牌旋轉後重新取得車牌字元區塊圖，如圖 3-20 所示。可知於左傾斜及右傾斜於正常範圍內(小於 45 度)，皆可正確校正，雖然校正過後影像仍略顯歪斜，但傾斜未超過 5 度，仍可正常辨識。

(a)　(b)

圖 3-20　雲端主機車牌字元區塊圖

3. 車牌辨識，Tesseract[8]是一個由 Google 維護的 OCR（optical character recognition）辨識軟體，基於 Apache License 為 Open Source，可運行於 Windows、Linux、Mac 等系統，它原先是由 HP 於 1985-1995 年之間所開發的。在 1995 年的時候於 UNLV Accuracy test 比賽中得到前三名。但是後來 HP 決定要放棄關於 OCR 方面的市場，因此 1995-2006 間 Tesseract 即很少被使用，後來由 Google 接續維護並 Open Source，整合 Leptonica Image Processing Library 使之可以支援更廣泛的影像格式及將之轉成超過 40 多種的語言，此外還提供訓練模式可供訓練新的字型，如圖 3-21 所示。

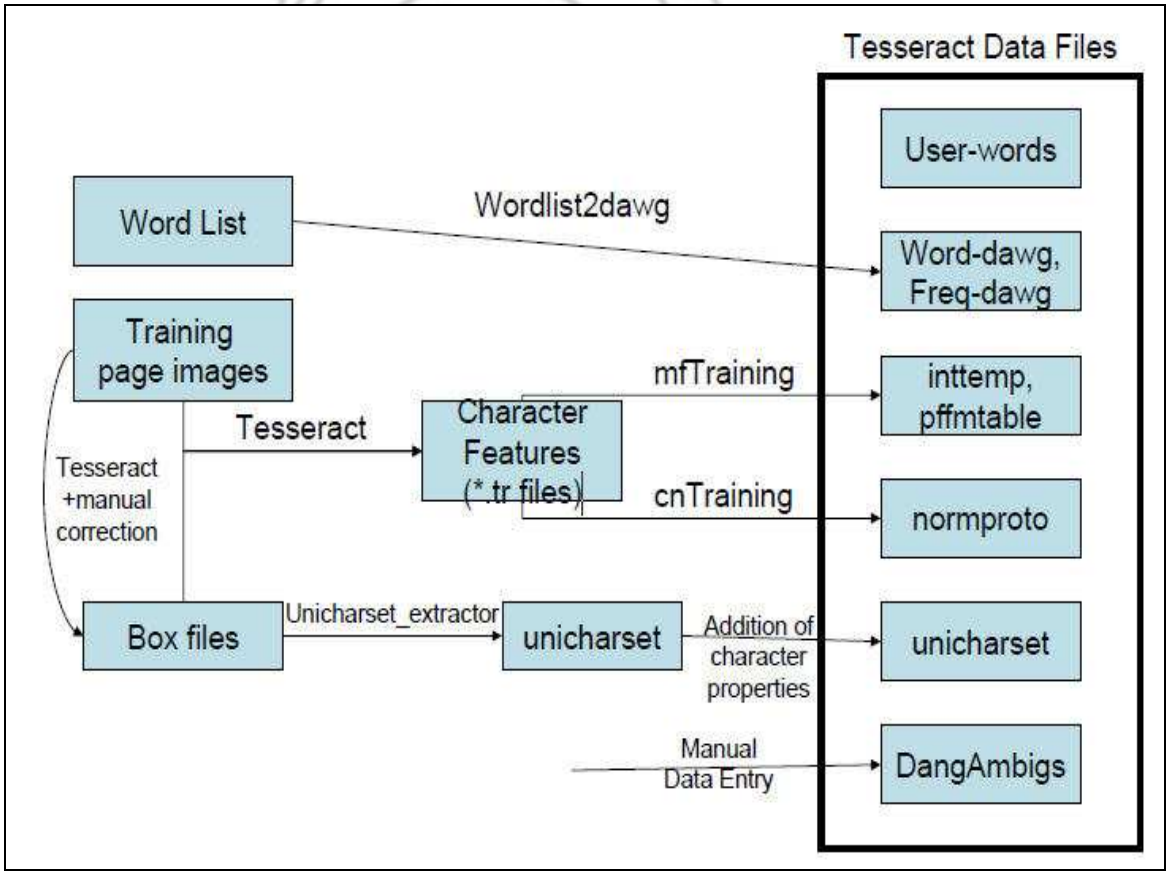
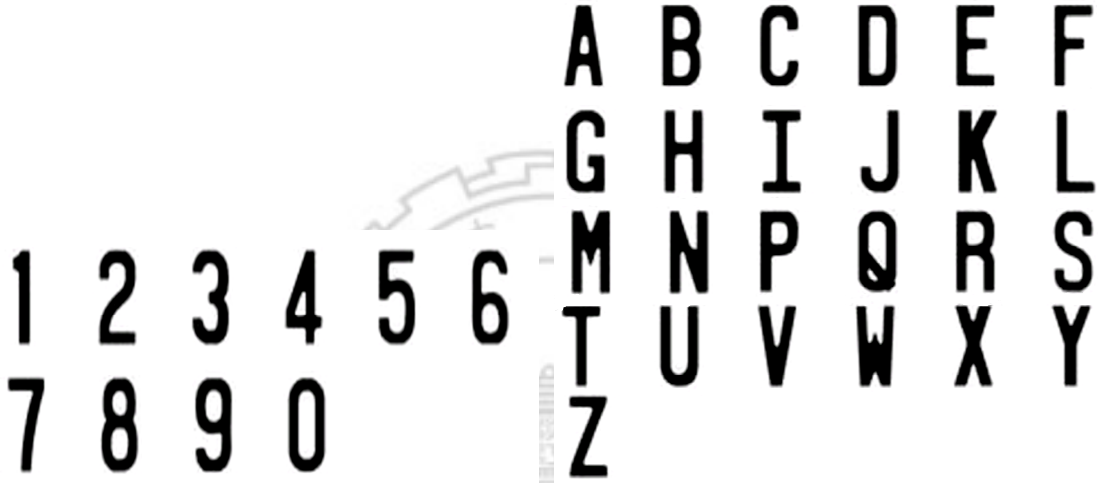


圖 3-21　Tesseract 訓練模式圖(圖片摘自[9])

因此在車牌辨識方面，我們採用 Tesseract 做為辨識的引擎，並使用 Tesseract 提供的訓練方式，針對取得的車牌字型做訓練，如圖 3-22 所示。其中 O 因為找不到樣本故未放入訓練，但不會造成影響，因交通部公路總局討論區[10]於 99 年 1 月 27 日回覆民眾 (路監牌字第 0990004334 號)所言，「因號牌英文字母 I 跟 O 易與阿拉伯數字 1 和 0 混淆，故自 85 年底起已不再製作含 I 及 O 之號牌」。



(a)　　(b)

圖 3-22　車牌字型圖

關於 Tesseract 的辨識程序，步驟如下所述:

1. Training Page Images，匯入車牌訓練樣本圖，匯入前需先將車牌字型圖內之字型做正規化，每個字元正規化為 35x17 像素，且字元與字元間需有空白，如圖 3-23 所示。

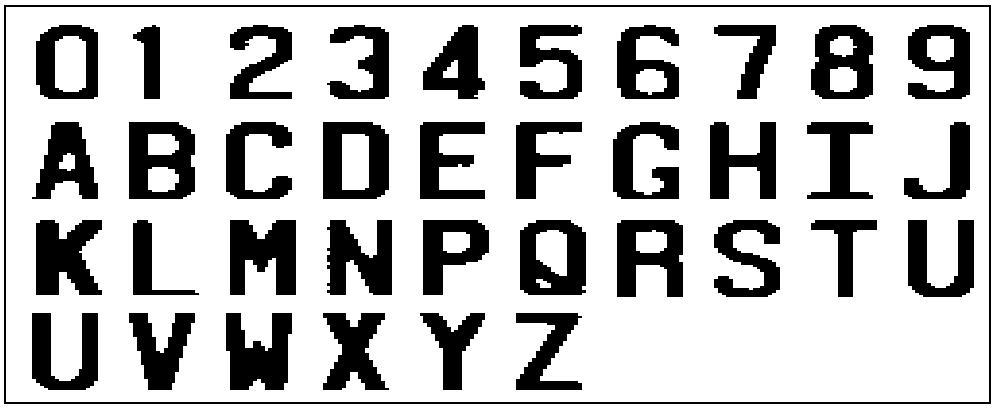


圖 3-23　車牌樣本圖

2. Box Files，使用 Tesseract makebox 產生 box file，標記出車牌樣本圖內每一個字元，如圖 3-24 所示。並將 box file 內每一個標示出來之字元位置輸入其相對應的英數字。

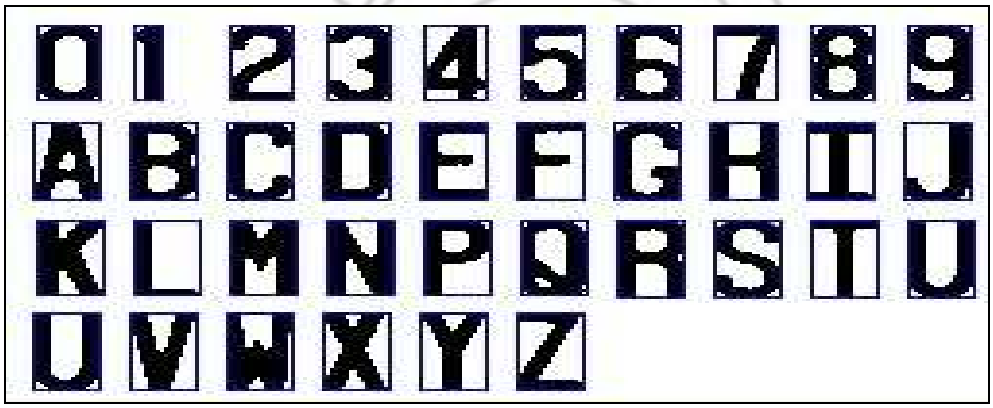


圖 3-24　車牌樣本字元標記圖

3. 接續使用 Tesseract 指令產生 Charater Feature 與 unicharset，經由 mfTaining、cnTraining 與 combine 指令完成訓練。

4. 於完成 1~3 之訓練樣本程序後，我們將經由前述影像處理過後的待辨識車牌做正規化動作，將車牌內字型使用內插法縮放為 35x17，使用 Tesseract 基於訓練的樣本庫來做辨識，於辨識完成後重新組成車牌號碼，如圖 3-25 所示。可辨識出 591588，而特殊符號－的辨識，則由連接區塊判斷前字元區塊的右

下角與後字元區塊左下角，差距最大的判定為一，因此再將車牌號碼修正為 5915-88。

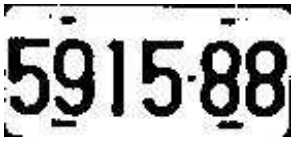


圖 3-25　車牌辨識圖

3.2.3　雲端主機代登入查詢

現行臺灣關於提供車牌相關的查詢服務於各機關管理單位皆或多或少有提供，但是各機關管理單位皆有自己一套的系統，功能散落於各個地方，且提供服務的方式並不一致，普遍性的更無可提供其他系統介接的 Web Service 服務。

關於車牌查詢相關的網站，由於其功能性及目的性的考量，通常是簡易的查詢功能，如圖 3-26 所示。台北市停車繳費、警政署贓車查詢、環保署機車定檢，網頁操作皆不繁瑣、使用方式簡易，僅僅只需輸入車牌號碼按下查詢，就可得知網站回傳結果，而結果也總是為固定格式的資訊。

(a)

(b)

(c)

圖 3-26　車牌查詢相關網站圖

因此本系統嘗試提供一種代登入查詢的功能，由雲端主機彙整與使用者相關的服務資訊，做查詢後再將使用者感興趣的資料回傳給手持裝置，達到一個服務整合的方式。茲以環保署的機車定期檢驗的檢測資料查詢為例，如圖 3-27 所示。

圖 3-27　環保署機車定期檢驗圖

可發現為單純的輸入車號查詢網頁，輸入正確的車號後顯示相關查詢資訊，如圖 3-28 所示。輸入錯誤的車號顯示查無車號，如圖 3-29 所示。

圖 3-28　機車定期檢驗有資料圖

圖 3-29　機車定期檢驗無資料圖

因此我們可以在取得車牌號碼，由雲端主機模擬車牌查詢動作，得到網頁查詢結果，並於手持裝置呈現，雖然可直接將結果展示於手持裝置，但往往現在設計的網頁，於手持裝置顯示不夠友善，畫面過大與瀏覽不易，如果僅將查詢結果擷取出使用者感興趣的資料，而不是整個網頁的資訊，顯示畫面能以手持裝置習慣的方式來呈現，如此能夠使得查詢服務更加友善，且節省網路傳輸量。

在網頁格式固定情況下，採用 HTML 解析器解析出我們感興趣的內容，採用 NekoHTML 搭配 Xerces，實作 HTML 解析功能，將網頁資訊解析為 HTML DOM，如圖 3-30 所示。對於網頁資料操作轉化為對這棵樹的操作，接著再使用 Dom4j 輸入要查詢資料的 Xpath 路徑，擷取出需要的資料，再將之組成 XML 透過網路回傳給手持裝置。

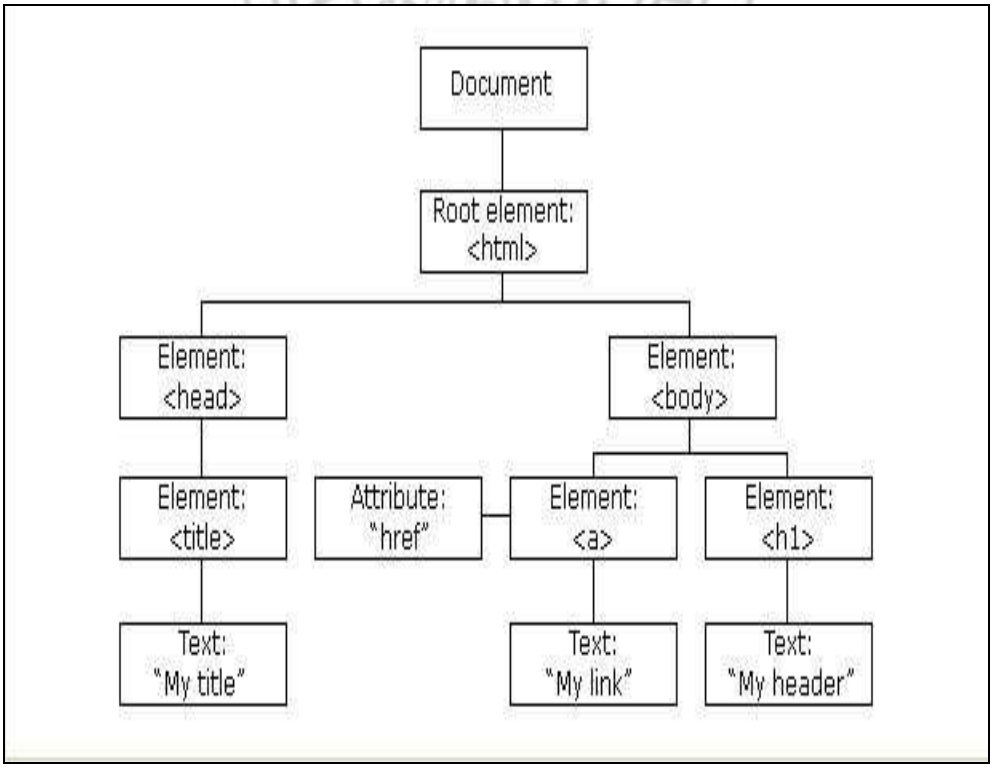


圖 3-30　HTML DOM(圖片摘自[11])

第四章　系統實作

在本章節中，將實作此系統，並分別為系統環境及操作介面展示。

4.1　系統環境

系統環境方面可分為手持裝置端及雲端主機端兩部分。

4.1.1　手持裝置端

手持裝置端是以 Android 平台開發，實作部份採用 Samsung GALAXY Tab 10.1 平板電腦，如圖 4-1 所示。規格如表 4-1 所示。

圖 4-1　Samsung GALAXY Tab 10.1 畫面

表 4-1　手持裝置規格

處理器	NVIDIA Tegra 2 雙核心處理器
作業系統	Android 3.1
記憶體	RAM :1GB ROM:16GB
螢幕	螢幕大小:10.1 吋 TFT 觸控螢幕 螢幕解析度:1,280 x800pixels
網路	支援 Wi-Fi 802.11 a/b/g/n 無線網路
相機	300 萬畫素相機鏡頭、LED 閃光燈、自動對焦
其他	內建陀螺儀感應器、三軸加速器、數位電子羅盤、距離感應器

1. 程式開發方面，採用的版本為 Android 2.3，因本系統所會用到的觸碰及照相功能於此版本已有很大的改善，直至 Android 4.0 差異不大，根據 Opensignalmaps 於 2012 年 4 月調查結果顯示，如圖 4-2 所示。可知高達 55.4%的用戶還在使用 Android 2.33+，次之則仍有 20.5%在使用 Android 2.2 。

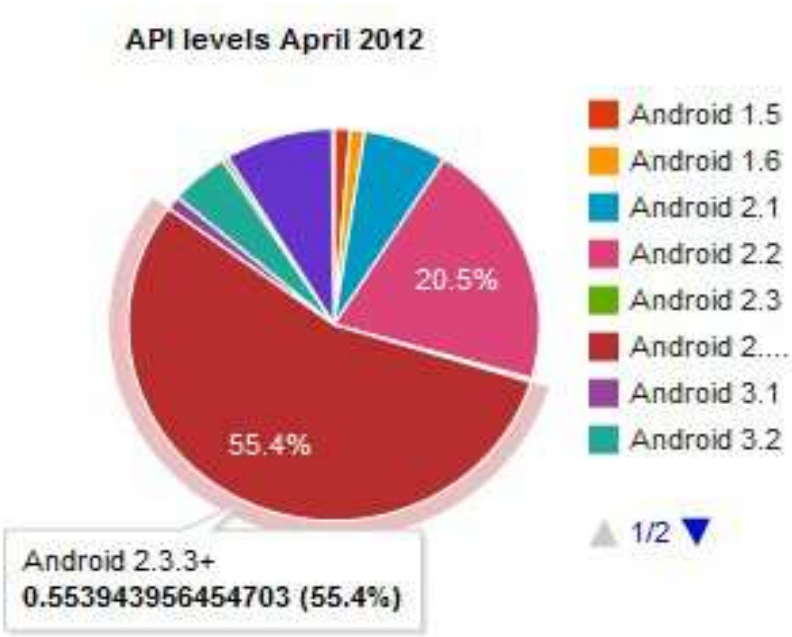


圖 4-2　Opensignalmaps 統計畫面(圖片摘自[12])

2. 開發的環境則以 Eclipse 來開發，如圖 4-3 所示。Eclipse 是著名的跨平台的免費開發環境（IDE）。最初主要用於 Java 開發，但亦可通過外掛程式使其作為 C++、Python、PHP、Android 等其他語言的開發工具。

3. 我們於開發環境設定並安裝如下所需軟體:

   (1) Eclipse SDK 3.5.2。

   (2) Android SDK。

   (3) Java JDK 6。

   (4) Samsung GALAXY Tab 10.1 USB 驅動程式。

   (5) Eclipse 安裝 Andoid plugin 套件 Eclipse ADT。

   (6) Eclipse 安裝 Samsung GALAXY Tab Addon 套件。

而後透過電腦之 usb 與平板相連接來開發。

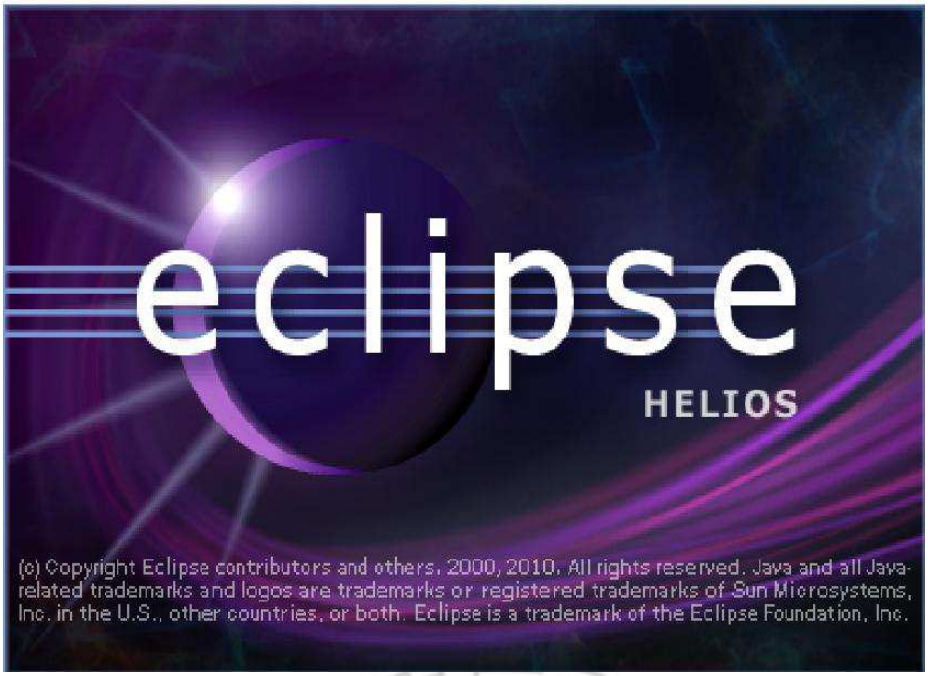


圖 4-3　Eclipse 畫面

4.1.2　雲端主機端

雲端主機端為 Windows，主機採用一般家用型電腦，規格如表 4-2 所示。

表 4-2　雲端主機規格

處理器	Intel Core i5-760
作業系統	Windows xp
應用伺服器	Apache Tomcat 6.0
JAVA 版本	JDK (Java Development Kit) 6
網路	ADSL

1. 程式開發方面，以 Java 為應用程式開發的基礎核心，可以在任何具有 Java VM 的環境下執行。

2. 應用伺服器，是一個由 Apache 基金會開發、維護的一個 Servlet 容器，按照 Sun Microsystems 提供的技術規範，實現對於 Servlet 和 JavaServer Page（JSP）的支持，由於 Tomcat 是使用 Java 開發的，可以在任何一個具有 Java VM 的環境下執行。

3. 開發的環境則以 Eclipse 來開發，如圖 4-3 所示。

4.2　用戶端程式介面說明

用戶端程式介面依按鈕可分為拍照、執行、詳細資料、返回四個功能，規格如表 4-3 所示。

表 4-3　用戶端按鈕功能

按鈕	功能
拍照	做拍照動作
執行	執行雲端主機整合查詢
詳細資料	顯示詳細雲端主機整合查詢結果
返回	返回至第一頁

而相關的操作步驟如下所述。

1. 手持裝置照相，如圖 4-4 所示。當使用者按下右下角拍照按鈕，拍照完畢後，若影像正確則接續做觸碰選取，反之則再按拍照鈕，重新照相。

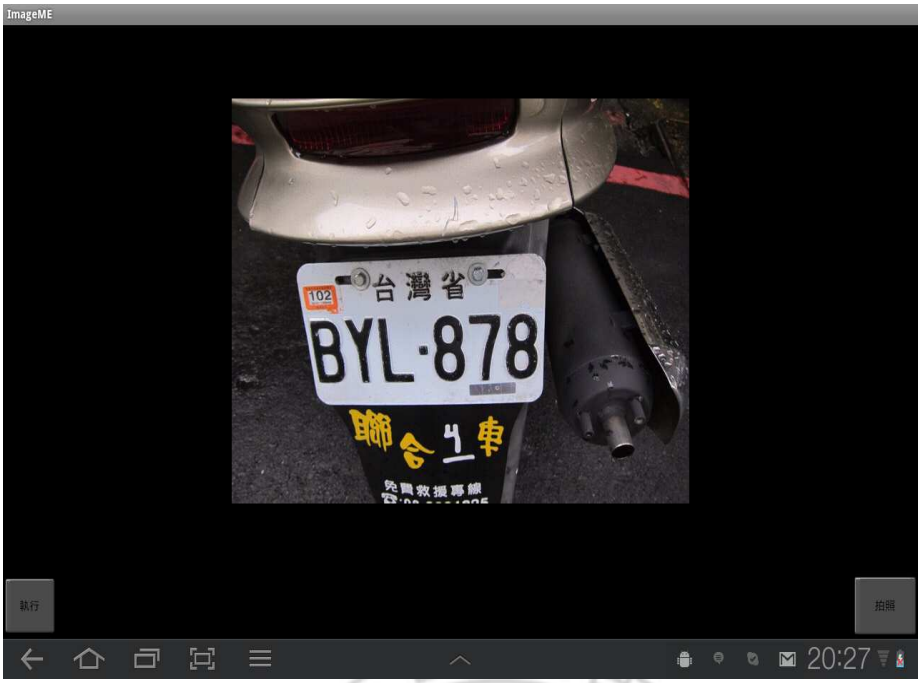


圖 4-4　手機拍照畫面

2. 觸控選取，於拍照完成的畫面使用觸碰方式做車牌定位操作，需經由觸碰選取兩點，依操作喜好可選擇兩點單獨觸碰選取或兩點同時觸碰選取，茲以分別觸碰選取為例，先於手持裝置輕觸一點，此時於觸碰點會呈現紅色圓點，如圖 4-5 所示。再繼續觸碰另一點出現另一紅色圓點，手持裝置會依兩點所在位置組成紅色矩形框，並同時於紅色矩形框內做運算，求出矩形框內車牌位置，並以綠色矩形框標注，如圖 4-6 所示。

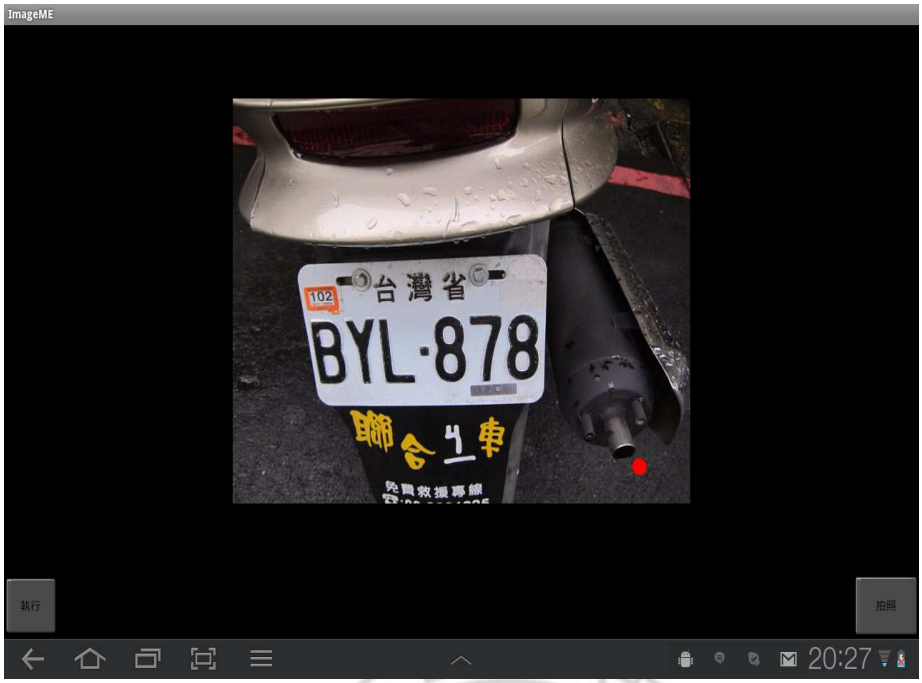


圖 4-5　觸碰單點選取畫面

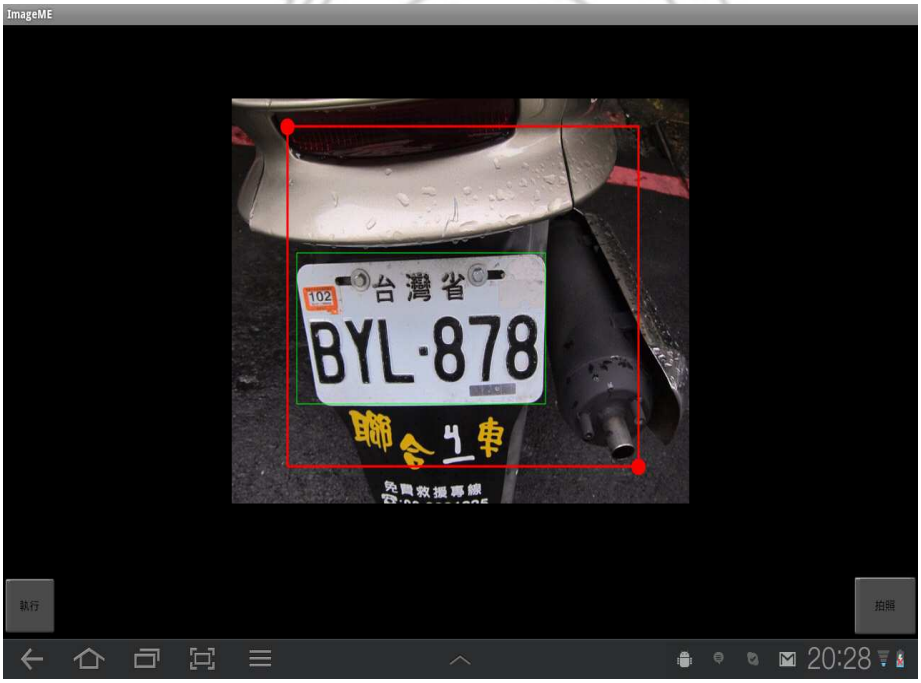


圖 4-6　觸碰兩點選取後畫面

若不滿意所選擇的紅色矩形框，可重新觸碰選取兩點，如圖 4-7 所示。若滿意定位選取畫面，則按下左邊執行按鈕。

圖 4-7　重新觸碰選取畫面

3. 執行，按下左邊執行則會開始與雲端主機做介接，執行完畢則回傳結果，如圖 4-8 所示。顯示雲端主機 OCR 之結果於中央，若代登查詢有資料則可點取畫面左邊的詳細資料按鈕。

圖 4-8　雲端主機有資料畫面

若查詢無資料，如圖 4-9 所示。

圖 4-9　雲端主機無資料畫面

4. 詳細資料，按下詳細資料按鈕則顯示雲端主機代登查詢的詳細資料，如圖 4-10 所示。查詢完畢可按右邊返回按鈕回到前一頁。

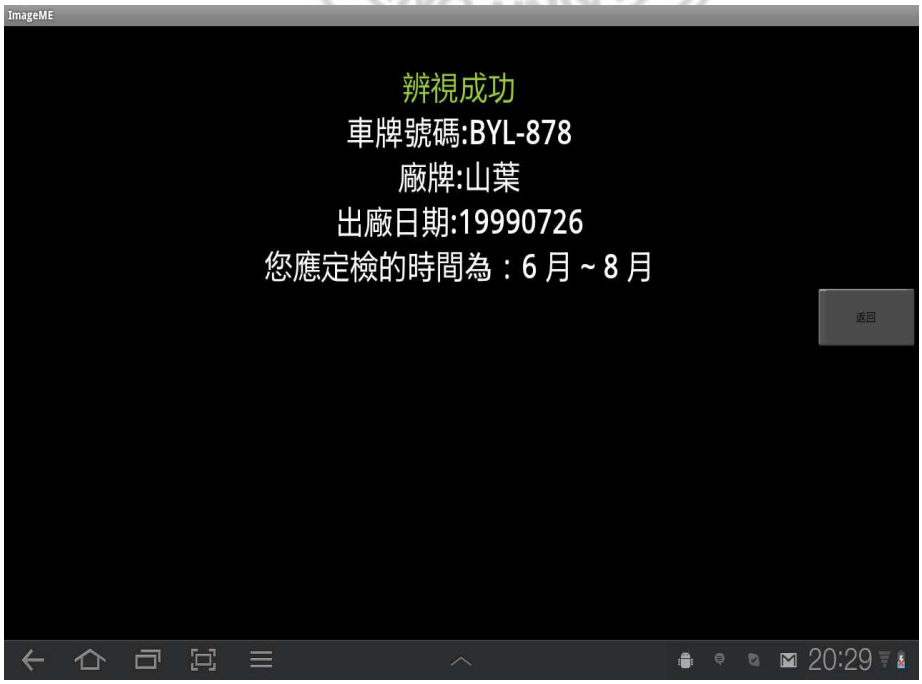


圖 4-10　雲端主機詳細資料畫面

4.3　實驗結果

我們隨機測試 330 輛機車來驗證系統，車牌辨識成功的有 271 張，如表 4-4 所示。

另再細分車牌內個別字元的辨識率可達 87%，如表 4-5 所示。

表 4-4　車牌辨識統計表

車牌辨識成功	271 張
車牌辨識失敗	59 張
成功率	82%

表 4-5　字元辨識統計表

辨識成功字元數	1769 字元
辨識失敗字元數	211 字元
成功率	87%

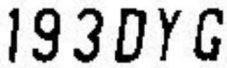
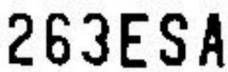
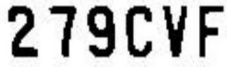
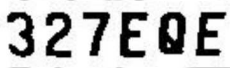
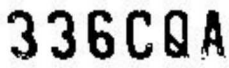
辨識的成功因素，結果如表 4-6 所示。總結如下:

1. 車牌字元清晰。

2. 車牌無污損或雜訊(如貼紙) 。

3. 若有易誤認字元(0，D ，8 ，B ，A ，4)，則車牌盡量不傾斜。

表 4-6　辨識成功結果

車牌影像	處理後影像	車牌號碼	辨識結果
		193-DYG	193-DYG
		263-ESA	263-ESA
		279-CVF	279-CVF
		2GN-163	2GN-163
		327-EQE	327-EQE
		336-CQA	336-CQA
		339-EUV	339-EUV

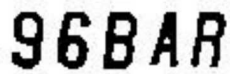
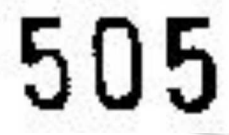
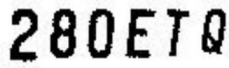
辨識的失敗因素，結果如表 4-7 所示。總結如下:

1. 車牌字元模糊。

2. 車牌污損或雜訊(如貼紙) 。

3. 有易誤認字元(0，D ，8 ，B ，A ，4)，車牌傾斜。

表 4-7　辨識失敗結果

車牌影像	處理後影像	車牌號碼	辨識結果	錯誤分析
		060-ERC	D60-ERC	車牌傾斜，0 辨識為 D
		096-BAR	96-BAR	字元無法辨識，0 上方貼紙雜訊
		0BW-462	DBW-462	車牌傾斜，0 辨識為 D
		ALC-585	AL-85	字元無法辨識，C-5 下方貼紙雜訊
		505-KCZ	505	字元無法辨識，-KCZ 下方貼紙雜訊
		280-ETQ	28D-ETQ	車牌傾斜，0 辨識為 D
		287-KFA	287-KF4	車牌傾斜，A 辨識為 4

第五章　未來展望及討論

本研究所提出的運用於智慧型手持設備之雲端車牌辨識系統,結合手持裝置特有的觸控功能，藉由人機互動與車牌做定位，此方式能減少運算時間而且提高正確率。

雲端車牌辨識方面，雲端主機先做車牌字元的定位及旋轉校正，再使用 Tesseract OCR 做字元辨識，因論文是以提出這個系統架構做貢獻，未特別著重於辨識處理這一塊，未來期望能於改善車牌雜訊處理方面做加強，及辨識方面能以 Tesseract 為基礎下加強關於車牌辨識的方法。

雲端車牌資訊查詢方面，現行為整合環保署排氣檢測網站做一代登入查詢，未來可以增加代登入查詢服務的網站以增加系統之功能性。

雲端主機及手持裝置方面介接方面，能夠增加互動學習、紀錄功能，手持裝置使用者可以將每次判別結果做簡單的手動確認,再傳與雲端主機做一影像辨識修正學習。

參考文獻

[1] 陳奕志，“建構嵌入式車牌辨識系統”，淡江大學資訊工程系，碩士論文，2005 年。

[2] 葉本源，“適用於臺灣各種車輛之車牌辨識系統”，中原大學電子工程系，碩士論文，2006 年。

[3] Android SDK, URL: http://developer.android.com/sdk/index.html.

[4] N. Bellas, S.M. Chai, M. Dwyer, and D. Linzmeier, “FPGA implementation of a license plate recognition SoC using automatically generated streaming accelerators,” in: *Proceedings of the 20th International Parallel and Distributed Processing Symposium*, Rhodes Island, April 25-29, 2006.

[5] P. Comelli, P. Ferragina, M.N. Granieri, and F. Stabile, “Optical recognition of motor vehicle license plates,” *IEEE Transactions on Vehicular Technology*, vol. 44, no. 4, November 1995, pp.790-799.

[6] How browsers work: Behind the scenes of modern web browsers, URL: http://taligarsiel.com/Projects/howbrowserswork1.htm#3_6.

[7] Apache CXF, URL: http://cxf.apache.org/.

[8] Tesseract-ocr, URL: http://code.google.com/p/tesseract-ocr/.

[9] R. Smith, “An overview of the Tesseract OCR engine,” in *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, vol. 2, Parana,

September 23-26, 2007, pp.629-633.

[10] Directorate General of Highways, MOTC, URL:

http://www.thb.gov.tw/tm/menus/new_english/index.htm.

[11] W3schools.com, URL: http://www.w3schools.com/.

[12] Opensignalmaps, URL: http://opensignalmaps.com/.

附錄一英文論文

The Android License Plate Recognition Based on Cloud Computing

Wen-Bing Horng, Kuan-Hung Liu
Computer Science and Information Engineering, Tamkang University
horng@mail.tku.edu.tw

Abstract- *The integration of portable devices and our everyday life is a popular topic in recent years. This thesis attempts to provide an architectural implementation of portable devices, using photographs and networks to build a mobile license plate recognition systems. The system adopted the open source code of the Android platform device and Java applications in building a cloud server.*

The portable device provides user interface (UI), easy selection of license plates through interaction, and rough positioning the plates via image processing algorithms. The cloud server provides OCR (optical character recognition) of license plate images, web site inquiry, and data recording. By the preprocessing of the cloud server, only necessary information will be sent back to the portable device in order to speed up network transmission. This architectural model is able to effectively reduce the cost of development, management, and maintenance. Furthermore, it also achieves the goal of rapid integration and effective utilization of resources.

Keyword: Android, License Plate Recognition, Edge Detection, Web Service.

1. Introduction

In recent years, most people could posses a portable device because its price is much affordable than before. In addition, government in recent years promoted the wireless Internet access. Last year, Taipei City Government, came up with “Taipei Free”, a free Internet connection service which attempts to cover the whole Taipei City. However, the framework of mobile network environment is getting more and more matured, but value-added services seemed under-developed. If we can combine portable device with Web services, we can then create a more comprehensive and localized smart cloud service integrated into our daily life.

Cars and motorcycles are the most common means of transportation in the city nowadays; and its most related application would be the license plate recognition technology. The scope of this technology covers systems such as parking management, vehicle control,

traffic monitoring, and speeding violations and stolen vehicle. PDA's are mostly used in billing parking fees, however, the device is very complicated, and the function is bound to a specific device. Its production is costly and also difficult to update. Therefore, a portable and upgradeable license plate data query software is seemingly important.

In this paper, we use photography and Internet with smart computing functions to build a mobile license plate recognition system. The system adopted open source code of the Android platform device, and Java applications in building Cloud Server.

2. Related Works

Android[1] is a Linux-based operating system developed by Google mainly used in smart portable devices. It has an Android SDK(Android Software Development Kit) for Android software developers. In August 2011, the Android operating system has reached 48% market share in the global smart devices.

License plate positioning system aims to find out the license plate region through image processing, and then performs license plate recognition. However, interference of background noise in the image, region similarity, and un-removable stain on the license plate, made it difficult achieve our goal. In fact, human eye has the same problem with those interferences mentioned above. Human eye determines a license plate mainly based on characteristics of the license plate, license plate background color and front characteristics. Chen[2] used the sobel filter to enhance the vehicle edge, and affixed the car window to find the license plate in an intensive area. Ye[3] uses the edge of license plate in the image and the image processed by HIS conversion, combined together to obtain a monotoned information graph, then by grouping and erosion, the projection is then used to get the region of the license plate region. Bellas[4] used connected component and bounding box to find the license plate region.

3. System Architecture and Design

The system can be divided into two parts, as shown in Figure 1. The first part is the portable device side. The use of portable device to catch the license plate image, and the license plate region detector. The other part is the Cloud Server side, we use the Web Service and portable device to do a multifunctional query service.

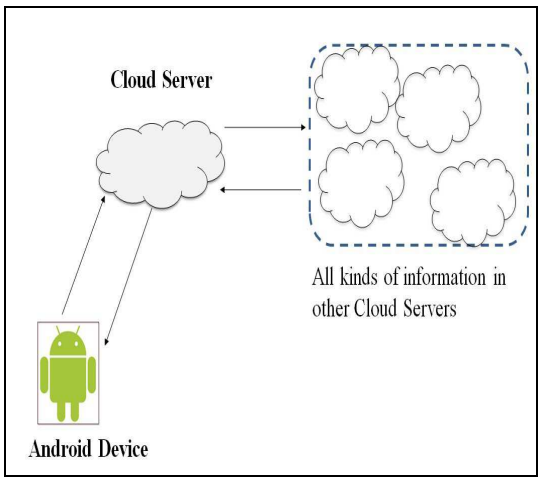


Figure 1 System architecture and design

3.1. Portable Device

After the portable device take the picture, we then perform some pre-image processing, detect the license plate region, and splitting the region send to Cloud Server side as shown in Figure 2.

The images we shot was in 640x480 and have a good recognition.

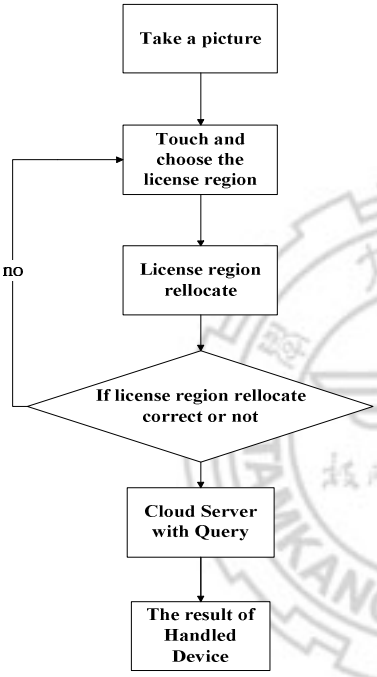


Figure 2 Portable Device architecture

3.1.1. Detect the license plate region

This paper used the touch functions in portable device, the steps described below:

1. Take the pictures, after taking the vehicle pictures, user touches the upper left and lower right or upper right and lower left, two touch composed of a red rectangle, as shown in Figure 3.
2. The portable device will transfer the red box image to grayscale.
3. We use Otsu to get the binary image.
4. To get the rough position of the vehicle license, we use connected-component method. The images processed after binarization, turns white as the foreground, black as the background, find the connected block, and mark each block, as shown in Figure 4, then we mark the max block as green rectangle, as shown in Figure 5.
5. If user confirms the green rectangle include the vehicle license, then touches the confirm button, if not can reTouch the picture to reDetect the vehicle license.

Figure 3 The red rectangle

Figure 4 Connected block

Figure 5 Mark the max block

3.1.2. Portable device with Cloud Server

We use Http Post to transfer the vehicle license and basic info to Cloud Server, as shown in Table 1.

name	type	description
userId	String	Portable device id
type	String	service type
image	Byte	vehicle license

Table 1 The form to Cloud Server

3.1.3. The result of Portable Device

After Cloud Server end the query, the Cloud Server return the information as XML format, then portable device parse the xml and show in the screen, the XML format shown in Table 2.

name	type	description
license	String	
brandType	String	
engineCapacity	String	engine displacement
strokecycle	String	cycle
birthDate	String	
useDate	String	Licensing Date
message	String	

Table 2 The XML format

3.2. Cloud Server

The portable device calls the Cloud Server via Web Service, as shown in Figure 6.

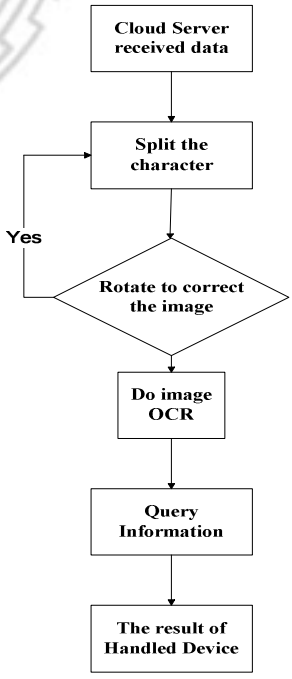


Figure 6 Cloud Server architecture

3.2.1. The Cloud server Web Service

We use CXF in implementing REST-based Web Service.

3.2.2. OCR in Cloud Server

Portable device transfers the license plate, gray scale image, to Cloud Server, The Cloud Server OCR steps described below:

1. Split the character. First we process image to get the binary image, then we use connected-component with black as the foreground, white as the background, to get the connected-block and mark it, as shown in Figure 7. Then we use threshold to filter the character, such as length, width, aspect ratio, the spacing between words and so on, as shown in Figure 8.
2. Rotate the image. After we get the connected-block, in the image from left to right, the left-most block and right block of the lower left corner (A and B) connected to a straight line, as shown in Figure 9, If more than 5 degrees to the horizontal angle, we rotate the image, as shown in Figure 10. Although the image is still slightly askew, but is not more than 5 degrees, so we can do OCR.
3. The OCR. We use Tesseract[5] as the OCR engine, first we train the template, as shown in Figure 10, each character normalize to 35x17, second use Tesseract makebox to produce box file, the box file mark all characters, as shown in Figure 11, and we enter the corresponding character, third use Tesseract command generates the the character feature and unicharset, then use mfTaining and cnTraining complete the training.
4. Finally. We take the processed images, normalized to 35x17, then recognized by Tesseract, get the output String 591588, as shown in Figure 12, special symbols – we can identify it by the max gap between each connected block, so we get 591-88.

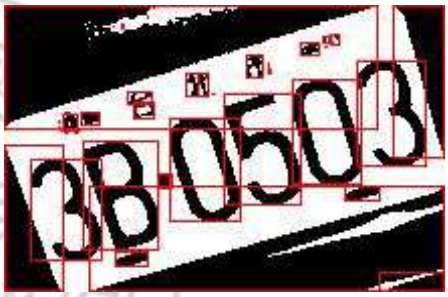


Figure 7 Connected-block

Figure 8 License character

Figure 9 Straight line

Figure 10 After rotate correction

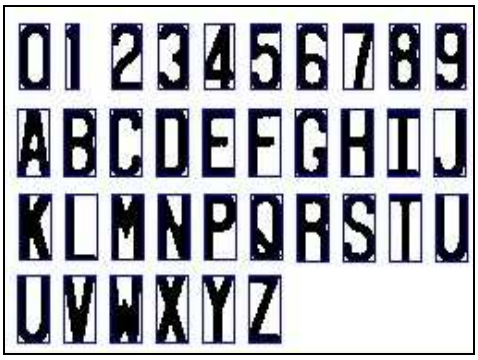


Figure 11 OCR template

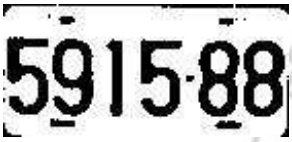


Figure 12 OCR image

3.2.3 Cloud Server with proxy query

The existing website about license plate, usually provides simple query function, as shown in Figure 13.

Figure 13(a) Simple page 1

Figure 13(b) Simple page 2

Figure 13(c) Simple page 3

Therefore, this system attempts to provide a query service, the Cloud Server organize and query information that user may be interested, and returned the brief information to portable device. We use Taiwan Government's website, and locomotive exhaust periodic inspection service as an example, as shown in Figure 14. This is a simple search website, where in user inputs the correct plate number, then related result will be given. If the input is incorrect, a notice of no result message will be outputted, as shown in Figure 15.

Figure 14 Correct query

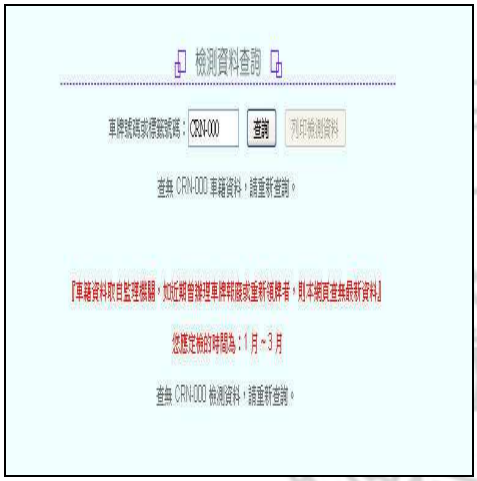


Figure 15 Fail query

If the website is simple and regular, we can easily use the HTML parser to parse and get the message that we are interested. We use NekoHTML with Xerces to implement parse function, parse HTML as HTML DOM, as shown in Figure 16.

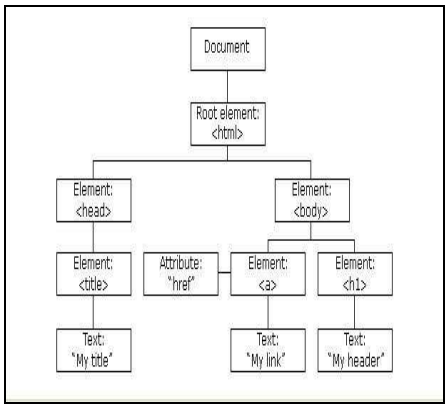


Figure 16[6] HTML DOM

4. System Implementation

This is section, shows the system environment and user interface.

4.1. System Enviroment

In Portable device we use Android platform Samsung Galaxy Tab 10.1, as shown in Table 3. In Cloud Server side we use personal computer, as shown in Table 4.

Processor	1 GHz nVidia Tegra 2 dual-core mobile processor
Operating System	String
Display	10.1 HD touch screen, 1280 x 800 WXGA TFT LCD
Connectivity	Wi-Fi 802.11 a/b/g/n
Camera	Rear-facing 3-megapixel auto-focus camera with LED flash

Table 3 Samsung Galaxy Tab 10.1

Processor	Intel Core i5-760

Operating System	Windows xp
Application Server	Apache Tomcat 6.0
Java version	JDK (Java Development Kit) 6
Network	ADSL

Table 4 Cloud Server

4.2. User Interface

Client program interface can be divided into four functions, as shown in Table 5.

Take a picture	Take a picture
Execute	Query to the Cloud Server
Detail	Get more details
Return	To previous page

Table 5 Client program function

The steps are described below:

1. Take a picture. When the user clicks the lower right corner of camera button to take a picture. If the photo taken is correct, the user then touches the screen and selects the license region, otherwise press the camera button and re-take a picture, as shown in Figure 17.
2. Touch and Select. User touches two point in the screen, then we can get the license region as red rectangle, At the same time, some computation is done in red rectangle to get the green rectangle with license plate region, as shown in Figure 18. User can re-touch to select two points, or press the Execute button on the left.
3. Execute. Pressing the left button will begin query to the Cloud Server, the result will be returned after finishing, as shown in Figure 19. User can press the left button Detail for more detail.
4. Detail. Press the left button to get more details, as shown in Figure 20, or press right button return to previous page.

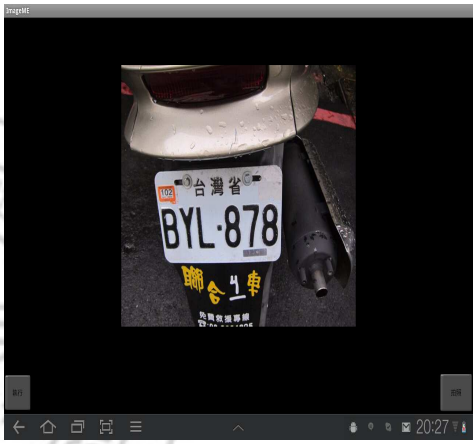


Figure 17 Take a picture

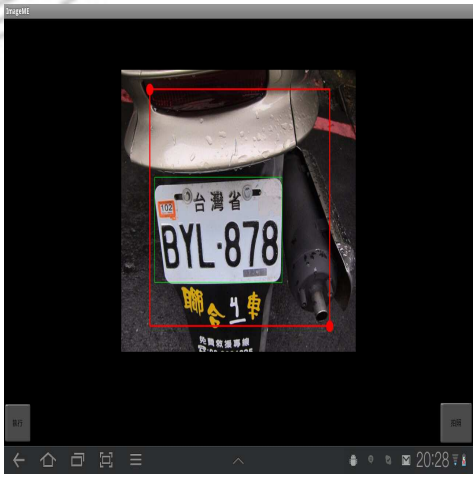


Figure 18 Touch and Select

Figure 19 Execute

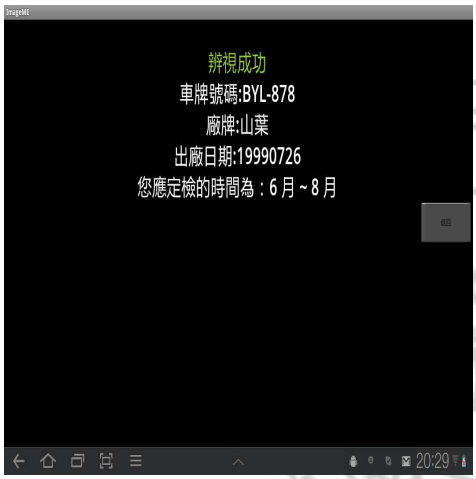


Figure 20 Detail

4.3. Result

We randomly tested 330 pictures and had 271 successful license plate recognition, as shown in Table 6. License plate character recognition rate is up to 87%, as shown in Table 7.

License plate recognition successful	271 pictures
License plate recognition failure	59 pictures
The success rate	82%

Table 6 License plate recognition

License characters recognition successful	1769 characters
License characters recognition failure	211 characters
The success rate	87%

Table 7 License characters recognition

Recognition of the success factors, the results shown in Table 8. and summarized as follows:

1. license plate characters should be clear.
2. The license plate without deface or noise (such as label).
3. Some characters (0, 8, B, A, and 4) can easily cause OCR failure, therefore it is suggested not to tilt the license plate.

License	Process	Result
	193DYG	193-DYG
	263ESA	263-ESA
	279CVF	279-CVF
	2GN163	2GN-163
	327EQE	327-EQE
	336CQA	336-CQA
	339EUV	339-EUV

Table 8(a) OCR Success

License	Process	Result
	060ERC	D60-ERC
	96BAR	96-BAR
	OBW462	DBW-462
	AL85	AL-85
	505	505
	280ETQ	28D-ETQ
	287KFA	287-KF4

Table 8(b) OCR Fail

5. Conclusion

This article presented a system for Vehicle License Plate Recognition based on Cloud Computing which integrated portable device and touch function with Human-machine interface that can locate license region. This can reduce the response time and improve the correctness rate. In the future, we hope to enhance our License plate recognition and offers a variety of query services by implement of this technology.

6. Reference

[1] Android SDK, URL: http://developer.android.com/sdk/index.html.

[2] I-Chih Chen, "Constructing Embedded Car License Plate Recognition System," *Tamkang University Department of Computer Science and Information Engineering*, Taiwan, 2005.

[3] Been-Yuan Yeh, "A Universal Plate Recognition System for All Vehicle Types in Taiwan," *Chung Yuan Christian University Department of Electronic Engineering*, Taiwan, 2006.

[4] N. Bellas, S.M. Chai, M. Dwyer, and D. Linzmeier, "FPGA implementation of a license plate recognition SoC using automatically generated streaming accelerators," in: *Proceedings of the 20th International Parallel and Distributed Processing Symposium*, Rhodes Island, April 25-29, 2006.

[5] Tesseract-ocr, URL: http://code.google.com/p/tesseract-ocr/.

[6] W3schools.com, URL: http://www.w3schools.com/.