違規停車偵測系統結合 Web APP 推播

Detection System of Illegal Parking Using Web APP Notifications

指導教授:方瓊瑤 老師　　學生:李知穎

（一） 系統簡介

本研究擬開發一套違規停車偵測系統，並用 Web APP 推播提醒車主另擇停車位重新停車。本系統利用現有架設於各街道巷口的攝影機，來進行違規停車與否的偵測。當車輛進入街道巷弄停車時，系統會自動擷取停車影像，並偵測該車輛是否停於紅線禁停區或黃線臨停區。若該車輛停於紅線禁停區，則利用 Web APP 系統通知用戶該車輛目前違規停車中。若該車輛停於黃線臨停區，則利用 Web APP 系統在臨停限制時間過後通知用戶，提醒駕駛者臨停時間已到。若車主於系統內建的緩衝時間過後而未取車時，系統將通知警方前往取締。本系統的主要目的在協助民眾避免違規停車，輔佐警方進行違規停車的取締，並改善各街道巷口交通的狀況。

（二） 研究動機與研究問題

	90年	91年	92年	93年	94年	95年	96年	97年	98年1~11月
交通違規舉發總件數	4,998,139	3,558,562	2,630,983	3,030,470	3,108,407	3,006,283	2,931,547	2,898,321	2,471,310
一、動態違規舉發	3,164,145	1,902,655	1,348,763	1,691,138	1,864,837	1,713,166	1,665,465	1,757,841	1,553,284
1.闖停件數	1,081,120	825,957	593,248	696,068	668,321	627,259	672,448	698,990	604,541
2.逕舉件數	2,083,025	1,076,698	755,515	995,070	1,196,516	1,085,907	993,017	1,058,851	948,743
二、靜態違規舉發	1,354,375	1,062,835	734,297	795,577	810,054	897,853	923,004	823,912	663,673
1.違停取締	917,152	742,971	487,039	508,265	537,742	596,212	632,479	562,956	469,123
2.汽車拖吊	301,485	210,006	180,959	214,935	210,226	220,549	206,086	184,231	140,182
3.機車拖吊	135,738	109,858	66,299	72,377	62,086	81,092	84,439	76,725	54,368
動靜態違規合計	4,518,520	2,965,490	2,083,060	2,486,715	2,674,891	2,611,019	2,588,469	2,581,753	2,216,957
三、慢車違規	_	_	_	_	140	136	1,353	245	183
四、舉發道路障礙件數	291,575	342,140	377,608	307,711	204,886	188,373	163,702	152,647	131,780
五、行人違規件數	188,044	250,932	170,315	236,044	228,490	206,755	178,023	163,676	122,390

根據台北市政府交通大隊，從民國 90 年統計至 98 年的交通違規資料 (如上圖)，違規停車累積次數高達 5,453,939 次，罰鍰高達 4,908,545,100 元，在這麼多項交通違規中，違規停車更佔了 19.04%的違規數。如果能有效地偵測違規停車，一定能有效地改善交通的窘境。

此外，民眾常常違規停車而不自知，本系統可以協助民眾避免破財之災。另一方面，本系統同時可以及時彌補警力的不足，輔佐警方進行違規停車的取締，進而改善交通的現況！

再者，因為智慧型手機的蓬勃發展，以往需耗費大量成本的簡訊通知系統已不適用。本系統計畫使用 Web APP 的推播通知功能，通知民眾或警方，不只降低成本，也非常有效率！

本系統偵測違規停車的主要步驟有二，（一）定義違規區域（二）違規停車偵測。定義違規區域的主要目的在連續影像中找到違規停車的位置（如圖一的紅框處所示），由於影像中擁有道路標線的顏色（白、黃、紅）之 pixel 並不多，所以在尋找道路標線時，只需要篩選特定顏色的區域和相鄰邊緣即可。而違規停車偵測的目的則在正確的判斷出車輛是否有違規停車（如圖二）。

圖一、在影像中找到違規停車區域

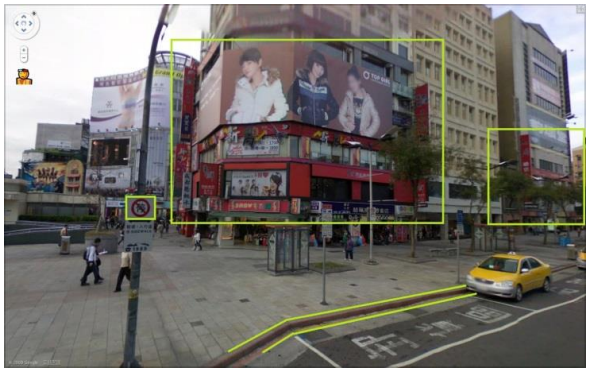
圖二、正確的判斷車輛是否有違規停車

本系統在進行車牌辨識時，因為影像擷取受到干擾的因素很多，將會面臨到許多應解決的困難，這些都是系統設計時應考慮的重點。

- 背景的影響

如圖三與圖四所示，由於有時街道，會有過多擁有類似顏色的雜訊，所以系統的設計應在不同背景的改變下，皆能準確的偵測違規停車。

圖三：背景的紅色招牌干擾



圖四：遠方的紅色房屋干擾

- 天氣的影響

街道設置的停車格，會因為某些天氣的改變，例如:下雨、起霧等，而增加影像辨識的困難度。

- 道路標線的完整性

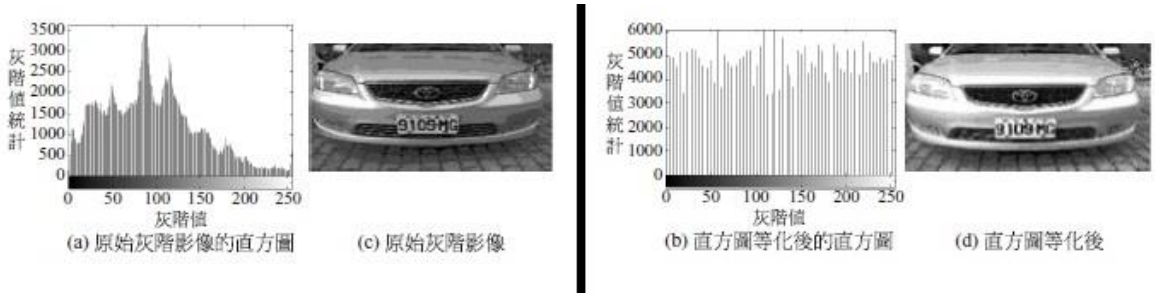
如果道路標線上有汙垢、毀損，皆會增加道路標線偵測的困難度，所以本系統將會運用一些方法來還原有缺陷的道路標線。

（三） 文獻回顧與探討

雖然更換了題目，但我認為原本的車牌辨識與現在的新題目，很多技術是可以共用的，所以決定延用一部分論文的技術，再追加一些新論文的技術。

國內外歷年來關於交通的研究相當多，但為了使用更新更快的方法，我選讀了 2010 年之後的相關論文，以下探討這些論文所使用的方法。

首先在影像光線調整上，李建興等人[1]的**即時動態車牌辨識**有提出一個程序。首先，將影像灰階化並作直方圖等化（Histogram Equalization），如下圖所示：



不但能提高影像亮度，更能提高影像的對比度，增加道路標線定位的成功率！

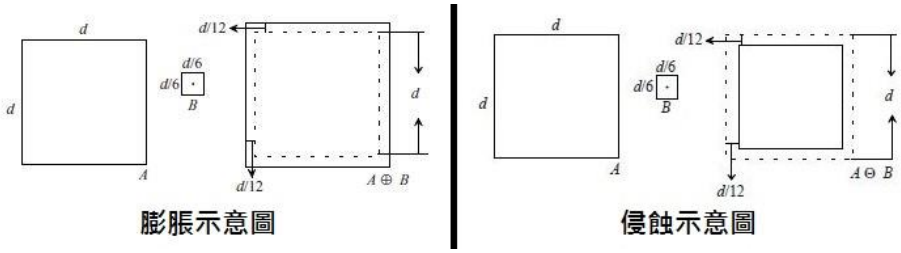
接著他使用 Sobel 遮罩來執行 Edge Detection，這個方法主要是利用車牌邊緣處灰階值差異較大的特性，且對 Pixel 正上下或正左右的加權來達到邊緣檢測的目標。因為道路標線也有與車牌邊緣相同的特性，所以也可以運用在尋找道路標線上，結果如圖五所示：

圖五：

原始車牌影像經Sobel遮罩所得到的邊緣影像

然後使用形態學運算過程裡的膨脹（Dilation）和侵蝕（Erosion），如圖六所示:

圖六：



膨脹：

其定義為 $A \oplus B = \{z|\hat{B}_z \cap A \neq \emptyset\}$，建立在獲得$B$ 對於其原點的反射並平移此反射z單位。A 藉由B膨脹為Z的所有位移 (使得$\hat{B}$ 和 A 的重疊至少一個元素) 的集合。

侵蝕:

其定義為 $A \ominus B = \{z|B_z \subseteq A\}$，指出$A$被$B$的侵蝕是使得$B$位移$Z$後仍包含在$A$中所有$Z$點的集合。

由膨脹與侵蝕執行的順序不同，又分為Closing和Opening：

- Closing：先膨脹在侵蝕，效果為雜訊去除。
- Opening：先侵蝕在膨脹，效果為連結區塊影像。

李建興等人[1]對上述得到的邊緣影像作Opening的形態學運算後得到成果，如圖七所示：

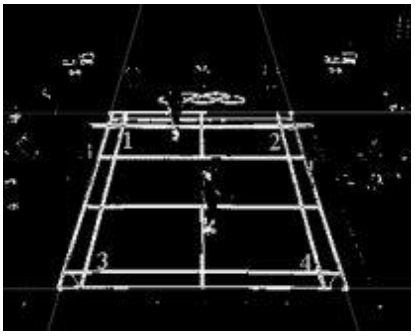
圖七：

對邊緣影像做Opening運算後的二值化影像

經過形態學處理後，雜訊已經被去除了，也連結了原本某些線段不完整的部分，接下來為偵測道路標線的部分。

張傑閔等人[7]使用了Hough Transform，來偵測羽球場的標線，我認為道路標線有異曲同工之妙，可以使用他們的方法來偵測，他們的成果如圖八所示：

圖八：

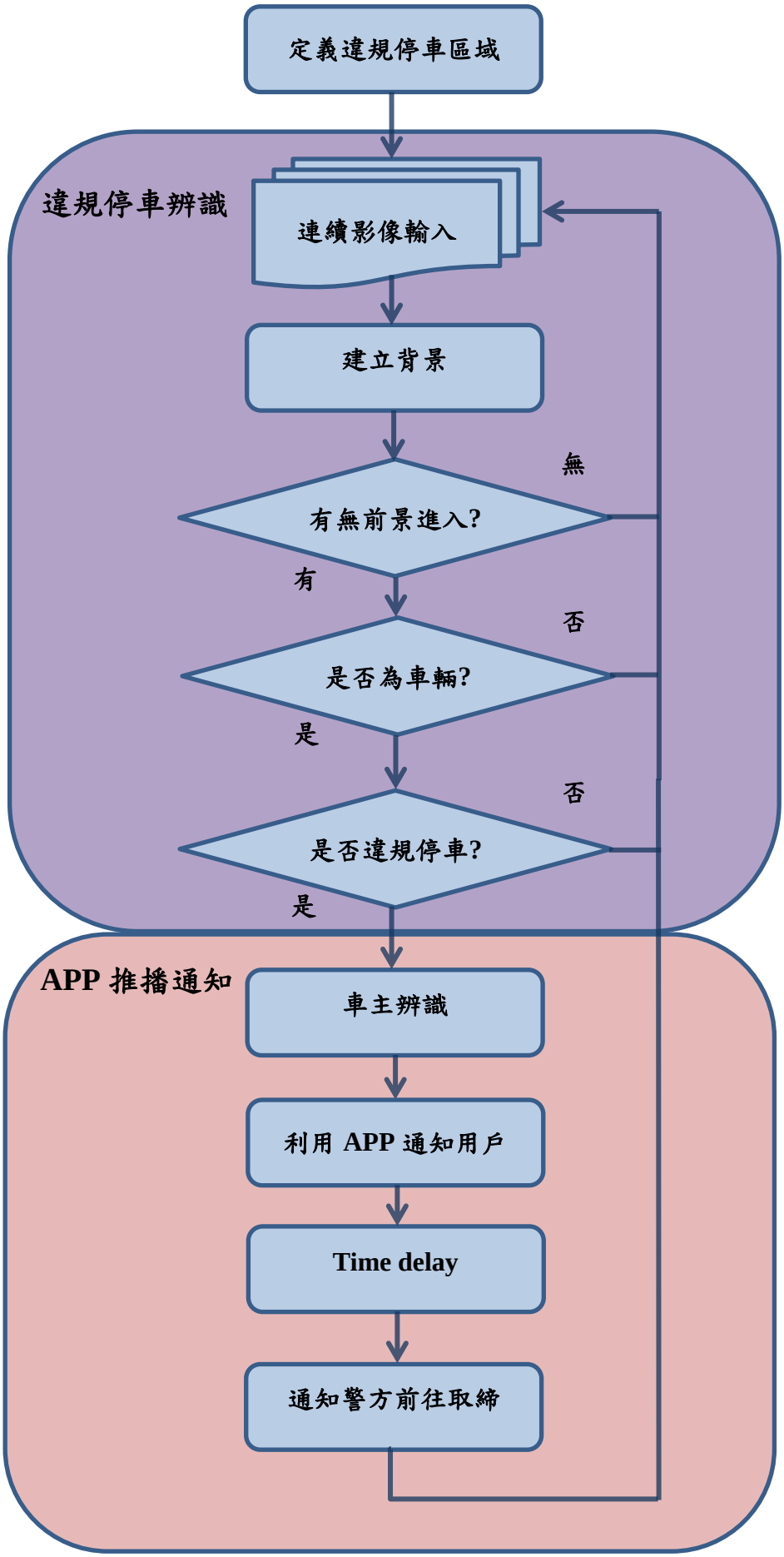


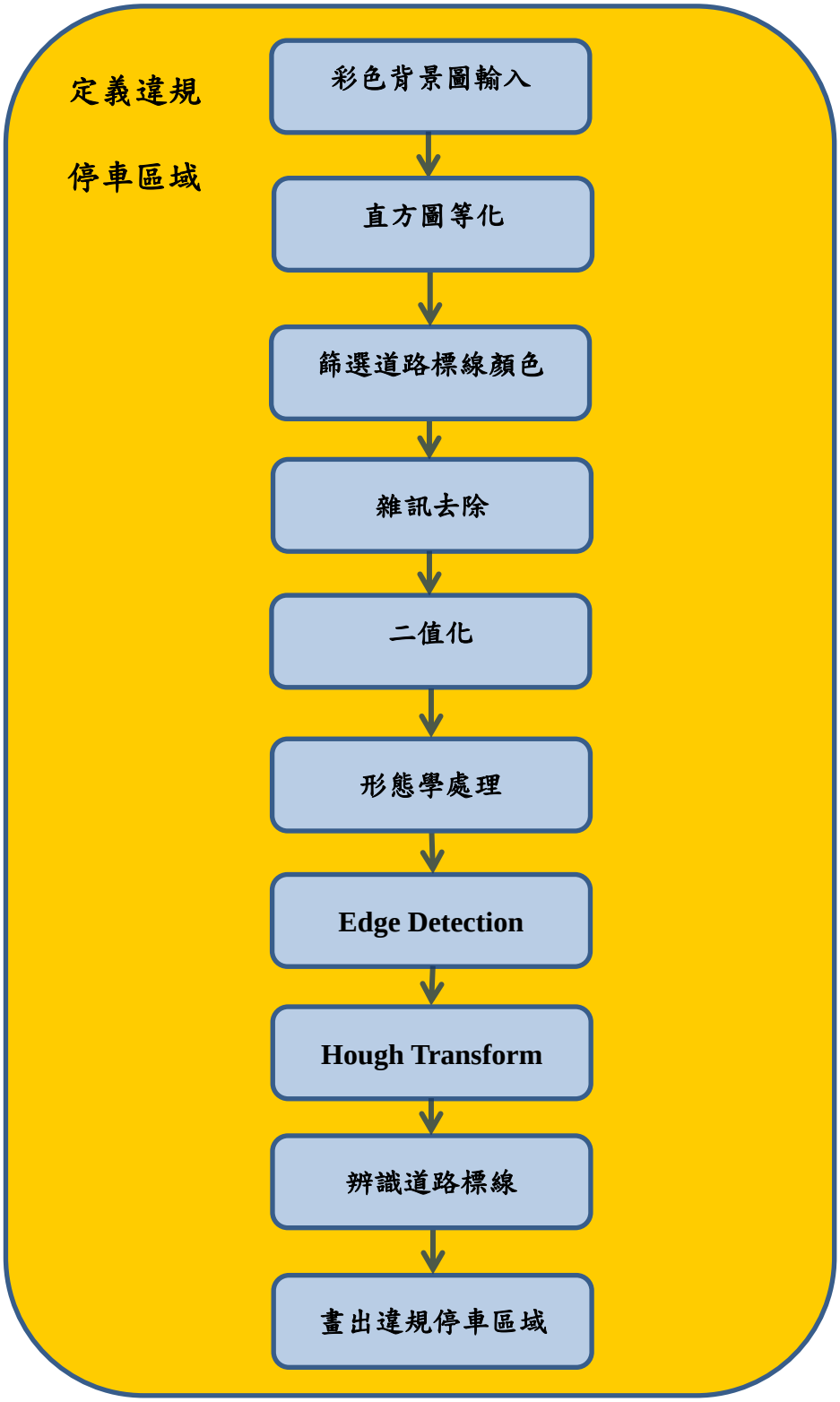
偵測完道路標線後，就可以根據道路標線的種類，來規劃不同的違規停車區域。

（四）開發工具與軟/硬體需求

本系統開發工具採用 Visual Studio 2010 並使用 OpenCV Library 來開發。且需要使用攝影機來擷取影像，與作業系統做聯結，將影像傳入電腦做處理。此外，也需要一台智慧型手機來接受推播通知，本研究使用了 Android 系統的智慧型手機。

（五）系統結構與操作流程


定義違規停車區域
違規停車辨識
連續影像輸入
建立背景
無
有無前景進入?
有
否
是否為車輛?
是
否
是否違規停車?
是
APP 推播通知
車主辨識
利用 APP 通知用戶
Time delay
通知警方前往取締


定義違規
停車區域
彩色背景圖輸入
直方圖等化
篩選道路標線顏色
雜訊去除
二值化
形態學處理
Edge Detection
Hough Transform
辨識道路標線
畫出違規停車區域

（六）系統功能之分項說明

本系統分為三個部分，**（１）定義違規停車區域（２）違規停車辨識、（３）APP 推播通知**。（１）定義違規停車區域：找到道路標線，並規劃出違規停車的區域，（２）違規停車辨識：偵測是否有違規停車的情況發生。（３）APP 推播通知，主要是在偵測到違規停車的情況後，利用系統推播 APP，通知用戶違規停車。

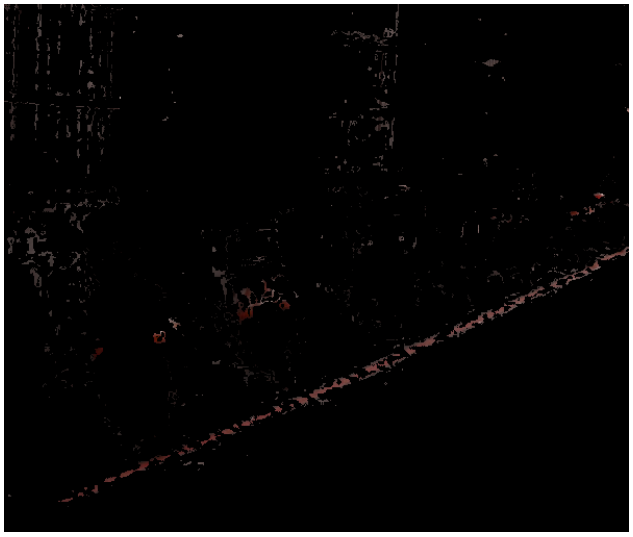
首先說明（１）定義違規停車區域的部分：

彩色背景圖輸入與道路標線顏色篩選：

由攝影機輸入背景圖，並篩選道路標線可能的顏色，主要的方法是將原本的 RGB 色彩空間影像，轉換為 HSV 色彩空間影像。使用 HSV 色彩空間來篩選顏色的原因，是因為 RGB 色彩空間利用三個 pixel 值來控制顏色，而 HSV 色彩空間則是只用了其中的 H（色相）來控制顏色，判別顏色的效果較為卓越，本研究經統計後，使用的值域為 0-30 與 330-360 度。而另外兩項 S（飽和度）與 V（明度），分別控制色彩的純度與亮度，主要受光線不同的影響，例如晴天與雨天此兩項值皆有不同的值域，所以本系統的顏色判斷並不受光線或天候的影響。判斷顏色的效果如圖九、圖十所示：

圖九、處理前：

圖十、處理後：



此操作是先將 RGB 色彩空間利用特定公式，將影像轉換為 HSV 色彩空間，接著判斷整張影像，符合本研究統計出來的值域之 Pixel，處理完後即為圖十的結果。

直方圖等化：

將 Poor Contrast Image 轉換成 High Contrast Image，如圖十一、圖十二所示：

圖十一、處理前：

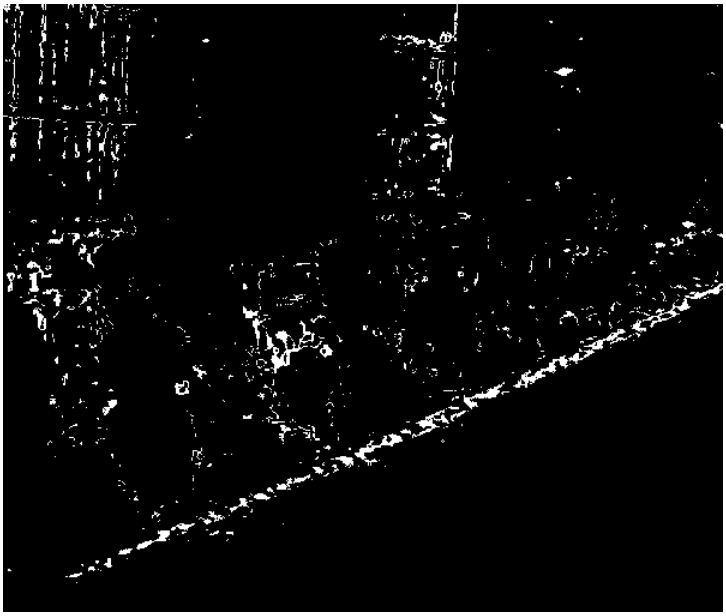
圖十二、處理後：

此操作將原本低對比的影像（即 pixel 值分布不平均，太集中在某一區塊），將其 pixel 值平均分布在 0-255，成為高對比影像。

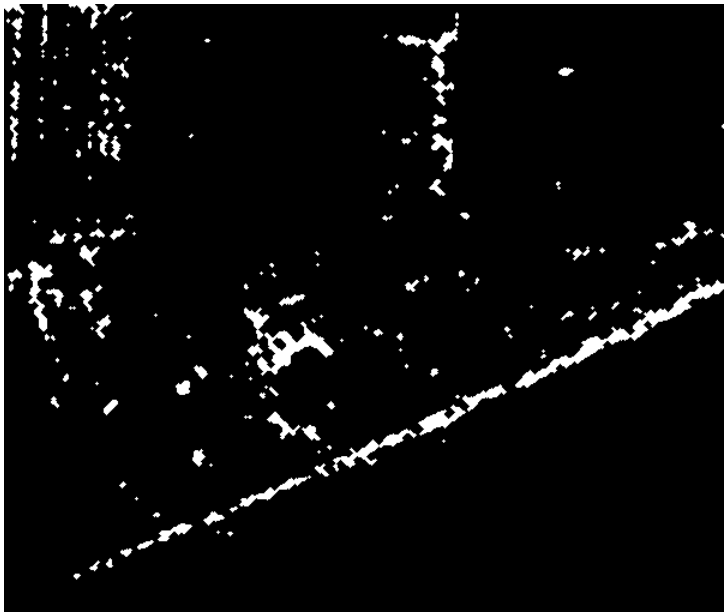
雜訊去除：

本系統結合連通物件法(Connected component)與形態學(Morphology)進行 Noise removement 處理，本研究發現，如果只使用形態學進行雜訊處理，將可能使原本需要的有用資訊一併去除，所以必須結合連通物件法進行處理。主要的想法是先用形態學將較細微的 noise 去除，接著再使用連通物件法將較巨大的 noise 去除，這樣不只能完善的去除雜訊，也能同時保留有效資訊，使得後續影像處理更為順利，如圖十三、圖十四所示：

圖十三、處理前：



圖十四、處理後：



二值化：

採用的是 Otsu's method，將影像的所有 pixel，變成 0 或 255，這個操作主要是去除顏色因素，讓後續處理的重心，從顏色判斷轉變到影像中的形狀辨識上，如圖十五、圖十六所示：

圖十五、處理前：

圖十六、處理後：

形態學處理：

對影像進行 Closing 或 Opening 處理，使得某些線段能更完整，或者去除一些雜訊處理忽略的小雜訊，這邊的形態學處理，與雜訊處理中的形態學處理有些微的不同，這邊的形態學次數相對於雜訊處理中的形態學次數較多，主要的概念是要將原本已經找出來的區塊結合，使得道路標線的特徵更為明顯，而不夠明顯的特徵則去除如圖十九、圖二十所示：

圖十九、處理前：

圖二十、處理後：

很明顯的，在紅色框框的範圍，原本是非常複雜，會嚴重影響後續處理的效率與準確度，經過形態學初步處理後，已經篩選掉不夠明顯的特徵了，經過後續的微量調整後，效果將會更卓越。而在綠色框框的部分，原本線段特徵不太明顯，經過形態學初步處理後，使得線段特徵變得非常明顯，對後續的處理非常有利！

Edge Detection：

進行 Edge Detection，方便後續辨識與分析線段，如圖十七、圖十八所示：

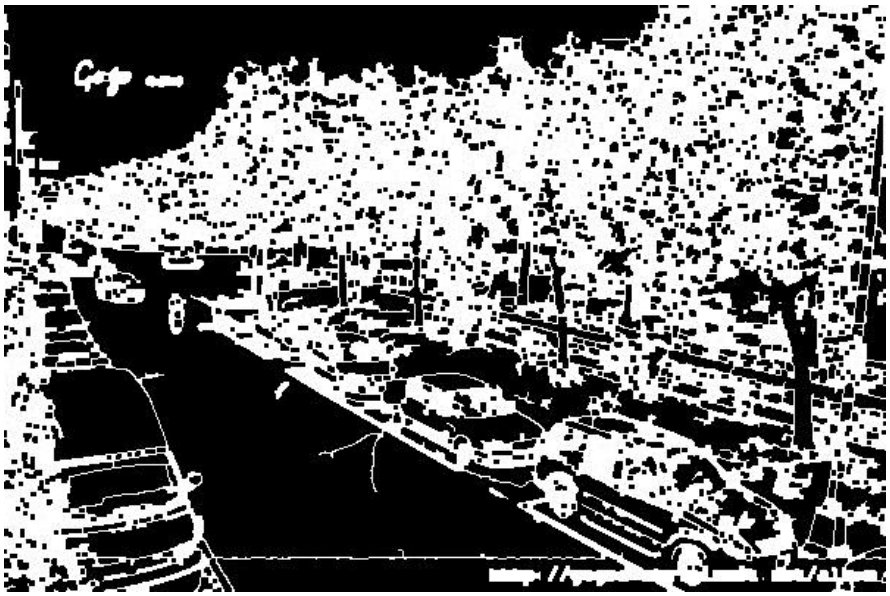
圖十七、處理前：　　圖十八、處理後：

本系統使用的方法，是將原本的影像去與經 Erosion 處理的影像相減，以下為示意圖：

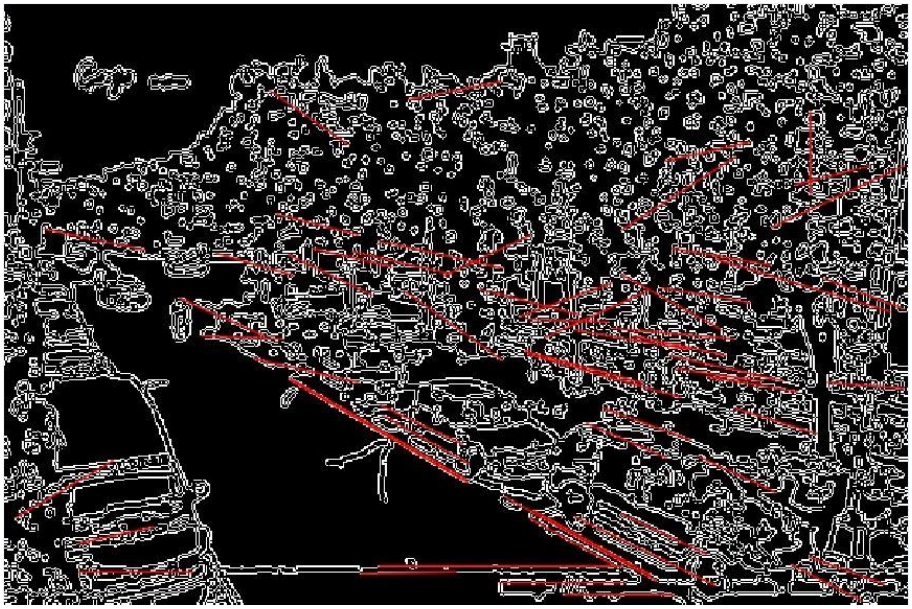
Hough Transform：

利用 Hough Transform，並設定特定條件（例如：線段的長度、線段的寬度…等等），進而偵測出影像中所有符合條件的線段。主要的原理是將原圖的 *x,y* 空間座標公式：*y=ax+b*，轉換為 *a,b* 空間座標公式：*b=-ax+y*，並在 *a,b* 空間座標系統的每個點給予 *counter*，只要有 *x,y* 空間的點轉換過來則 *counter* 值加一，最後找出 *counter* 值夠大的點，將其 *a*、*b* 值帶回 *y=ax+b*，即為原本在 *x,y* 空間的直線。如圖二十一、圖二十二所示：

圖二十一、處理前：



圖二十二、處理後：



辨識道路標線：

從 Hough Transform 找到的所有線段中，辨識出道路標線。可以利用每個線段之間的距離，或者線段本身的寬度及長度來辨識出道路標線，另外也可以透過收集夠多的影像後，利用統計的結果，彙整出道路標線在影像中位置，進而用此位置篩選出道路標線。本系統找出的道路標線位置，如圖二十三所示：

圖二十三、綠色線段為找到的道路標線。

道路標線種類判斷：

判斷找到的所有道路標線之種類，下表為主要的道路標線種類：

道路標線種類	可能之顏色	代表意義
紅色道路標線	鮮紅色、暗紅色	禁止停車
白色道路標線	白色	可以停車
黃色道路標線	鮮黃色、黃色	只能臨時停車

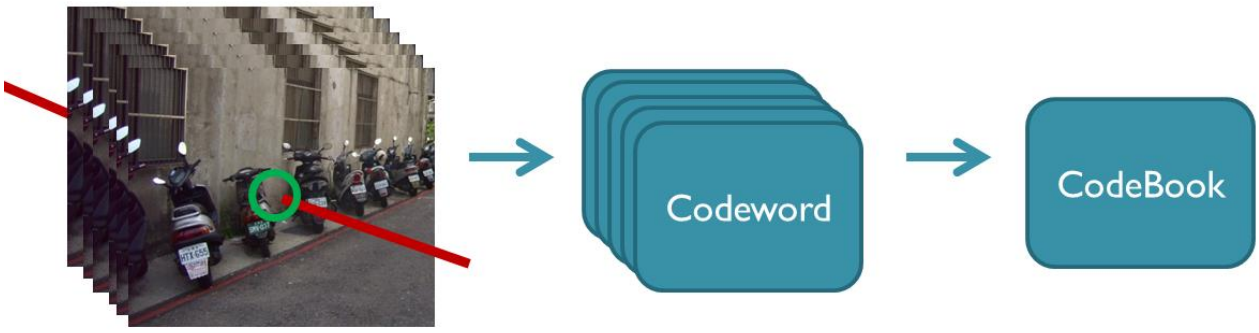
畫出違規停車區域：

利用辨識出的道路標線種類，規劃出違規停車區域，以利後續系統判斷，如圖二十四所示：

圖二十四、紅色區域為依照道路標線規劃的違規停車區域。

（２）違規停車辨識：

首先，本系統使用 Codebook 資料結構，來輔佐背景的建立與前景的判斷。Codebook 資料結構與建構方法的說明如下圖所示：



首先本系統會記錄連續影像中，每個 frames 中，同個位置不同時序的 pixel 值，接著將其在 RGB 座標空間上的分布利用 K-Means 進行分群，每一群都是一個 Codeword，裡面記錄了同一時序的 pixel 值之細微分布變化，接著將這些 Codewords 彙整為一個 Codebook，最後

依照每個 pixel 自己的 Codebook，建立出背景，此背景將能有效降低光線與氣候的影響，也能降低細微 Pixel 值的變化，例如雜草的擺動等因素影響。

連續影像輸入：

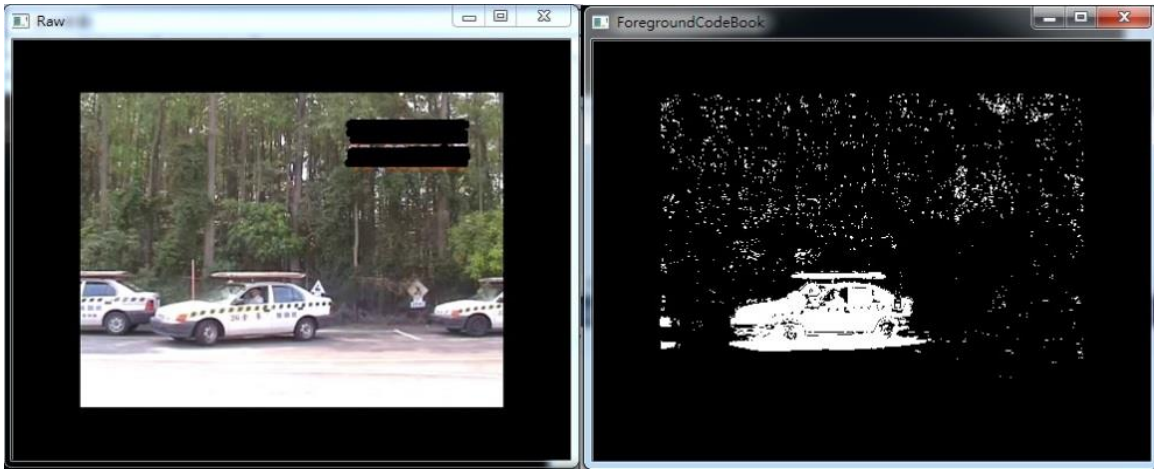
透過攝影機，輸入連續影像。

建立背景：

利用 Codebook 資料結構，彙整出影像中每個 pixel 的 Codebook 後，將其建立為背景，以利後續系統處理。

有無前景進入：

利用 Codebook 資料結構建立的背景結合背景鄉剪髮，去判斷有無任何前景進入，如果有則繼續做下一步驟，沒有則回到連續影像輸入，直到偵測到前景為止。偵測到前景的情況，如下圖所示：

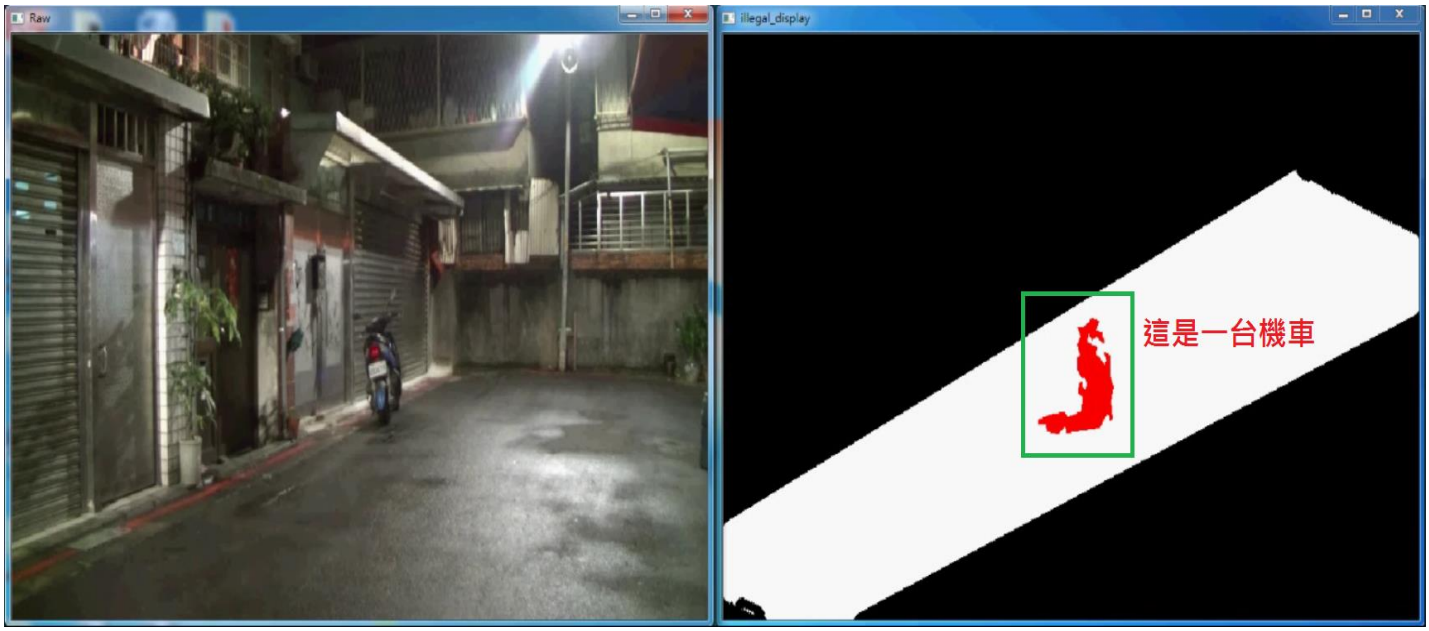


接著利用 MNRL(Maximum negative run-length)時間過濾法，來進一步更新之前所建構的 codebook，藉由計算各個物體 pixel 出現的頻率及存在畫面的時間，來過濾短暫經過的移動物體，如此便可以得到一個明確的背景模型。

是否為車輛：

判斷偵測到的前景，是否為車輛，主要依據是利用 RFID 確認用戶身分，或者使用車牌辨識，利用車牌來判斷是否為車輛，也可使用前景的長寬比例來判斷。如果是則繼續做下一步驟，不是的話則繼續偵測其他前景，直到偵測到車輛之前景為止。偵測到車輛的情況，如圖二十五所示：

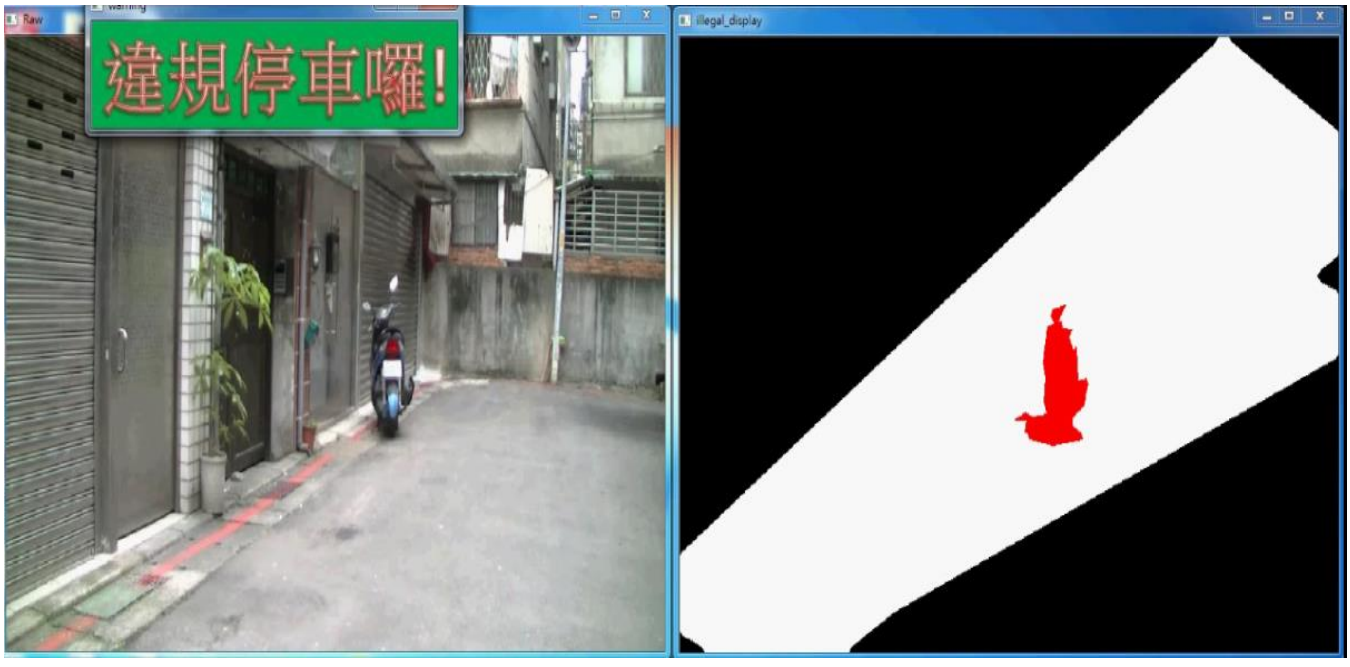
圖二十五：



是否違規停車：

判斷是否違規停車，主要利用前述規劃的違規停車區域，如果有車輛停在違規停車區域中，並持續一段設定的時間(一方面是確定目標車輛之停車行為，另一方面是因為黃色道路標線，只有在停車超過一段時間，才算違規)，則判斷目標為違規停車！ 如圖二十六所示：

圖二十六、此車輛已經停超過設定的時間，而且停在規劃的違規停車區域，將其判斷為違規停車。



（３）APP 推播通知

車主辨識：

透過RFID技術(一種可以遠距離,辨識設定ID之系統,ETC與悠遊卡都是它的應用。)、車牌辨識...等方法，來辨識用戶。

利用 APP 通知用戶：

利用取得的車主辨識結果，查詢用戶，並推播通知其違規停車，請用戶重新停車。(主要使用 PHP、HTML、Java Script...等語言製作此 APP。)

Time delay、通知警方前往取締：

系統會設置一段時間，如果車主在這段時間內，都沒有重新停車的話，將會通知警方前往取締。

APP 的介面與推播功能如下圖所示：

（七）系統特色

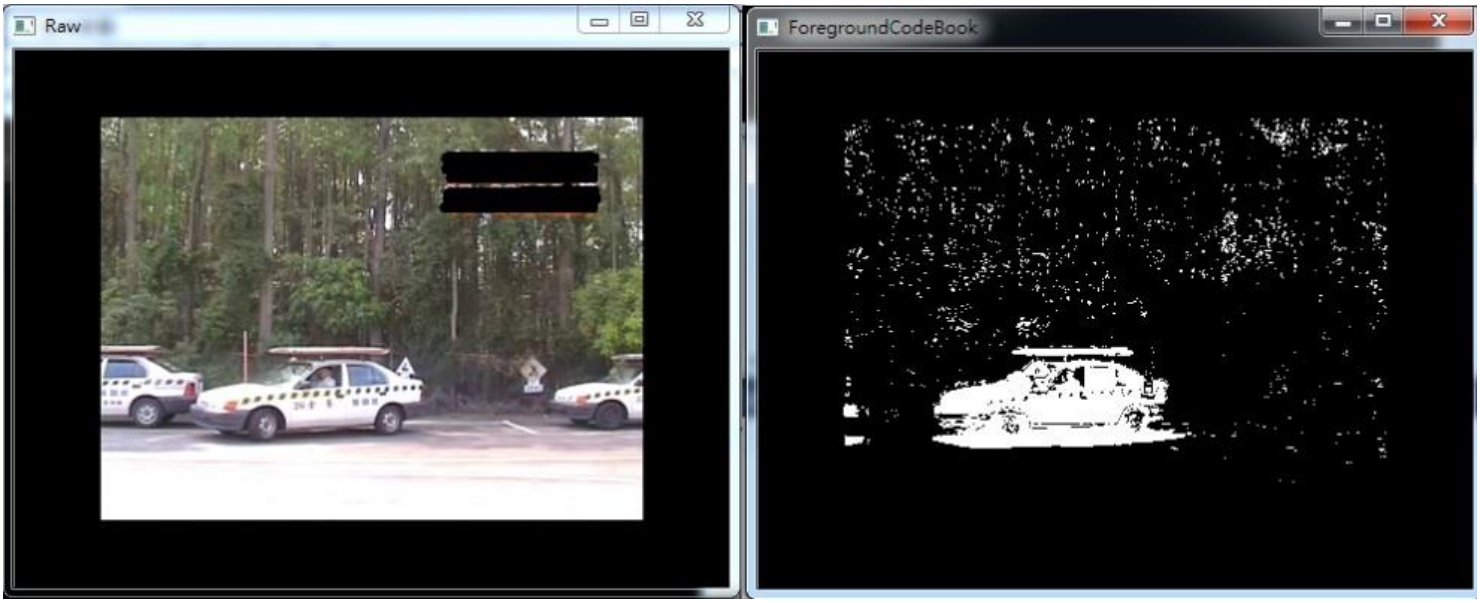
1. 特別的主題：以往有關交通號誌或道路標線的專題或論文，都著重在影像的辨識上，大家比較的都是方法的準確度與效率。本系統不只追求準確度與效率，也同時利用辨識出的結果，做後續的應用。

2. 使用 WEB APP：因為智慧型手機日新月異，不同作業系統的手機，如雨後春筍般推陳出新。WEB APP 是一種，可以跨所有作業系統的 APP 型式，而且不須額外學習各種不同的 APP 語言，以致增加撰寫者的負擔，只需要學習一般網頁的語言，例如 PHP、HTML…等等，就可以撰寫 WEB APP。所以本系統可以跨所有智慧型手機平台，不受作業系統的限制！

3. 多元的服務對象：本系統不設限服務的對象，不只服務一般民眾，提醒民眾違規停車；也同時服務警方，協助其取締作業，將系統的價值完全發揮出來！

（八）實作成果

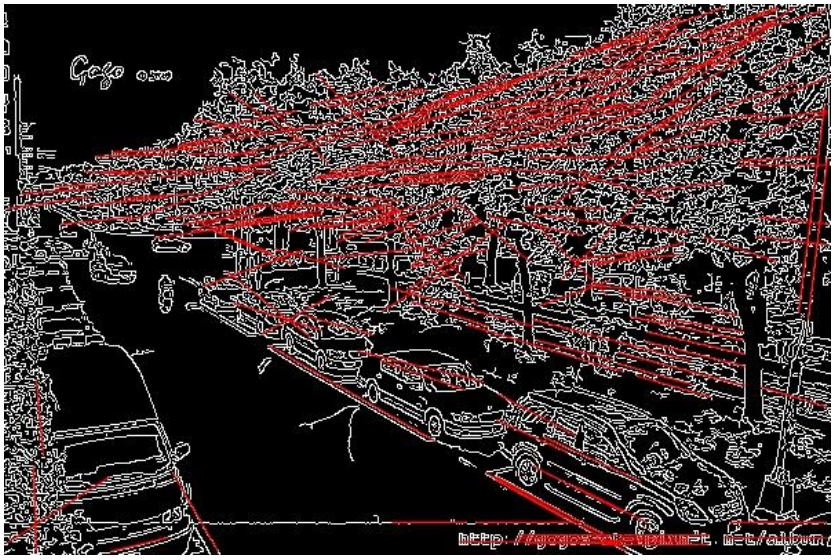
1. 輸入連續影像後，使用 CodeBook 背景相減法偵測前景的進入，以利後續判斷使否為車輛，下圖為實作結果：



2. Hough Transform：

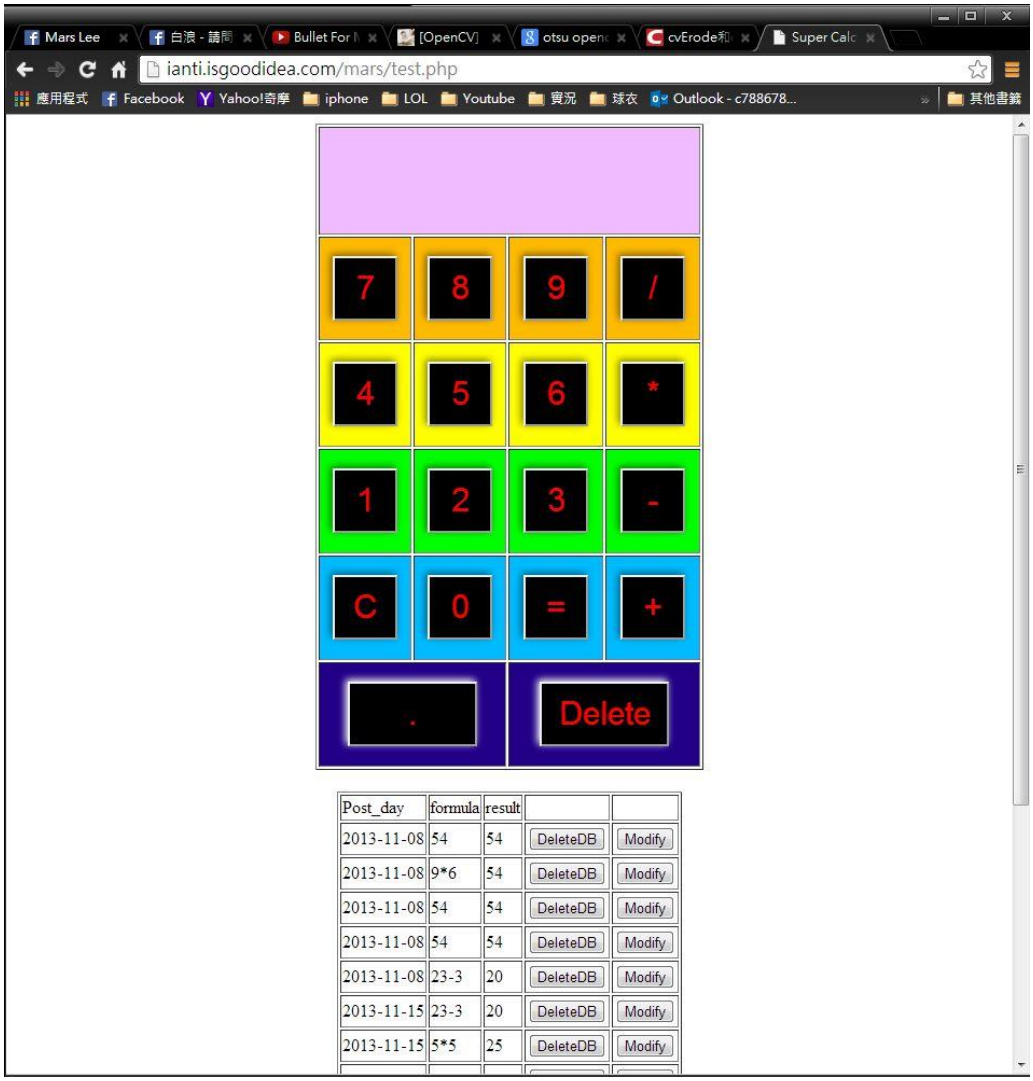
處理前：

處理後：



3. 撰寫 WEB APP 的前置練習：

<	2014	>	<	1	>	
Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

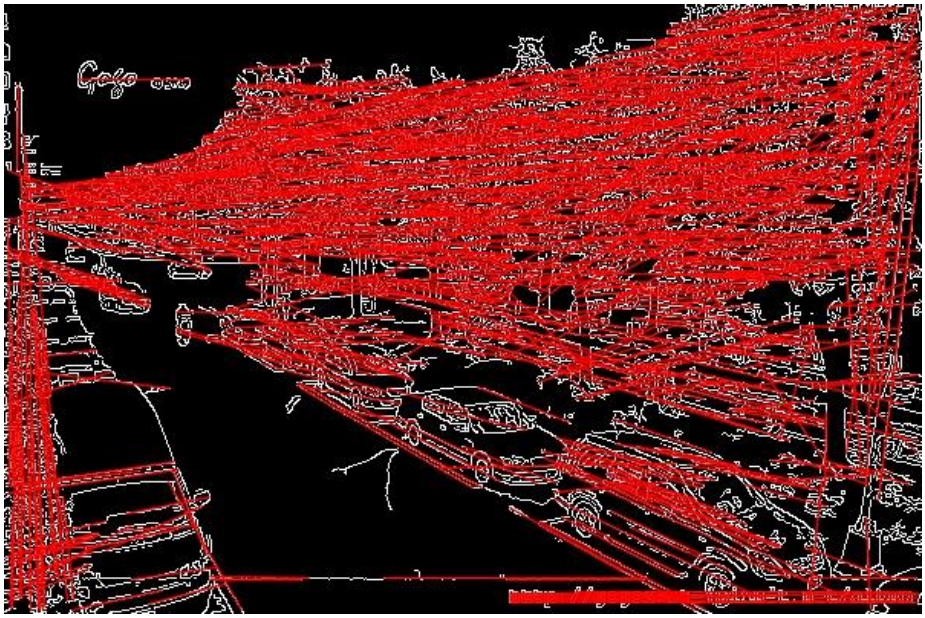


第一個是網頁版的萬年曆，主要是練習 HTML 中的 table 標籤運用與 php 時間函數的使用。

第二個為一個網頁版的計算機，主要由 PHP、HTML、MySQL 及 Java Script 等語言完成，可以計算一連串的算式，例如：1+2-3*6，並將結果存入資料庫（下面的部分）。預計使用 HTML 建構整個 WEB APP 的主架構，接著輔以 CSS 做一些美化與特效，另外搭配 PHP，除了連結 MySQL 資料庫之外，也將用來撰寫大部分的功能，而 Java Script 則負責 AJAX 技術的部分，AJAX 技術可以有效的讓整個系統，使用上更為順暢。我將持續精進上述提到的 4 種語言，以利後續 WEB APP 的撰寫。

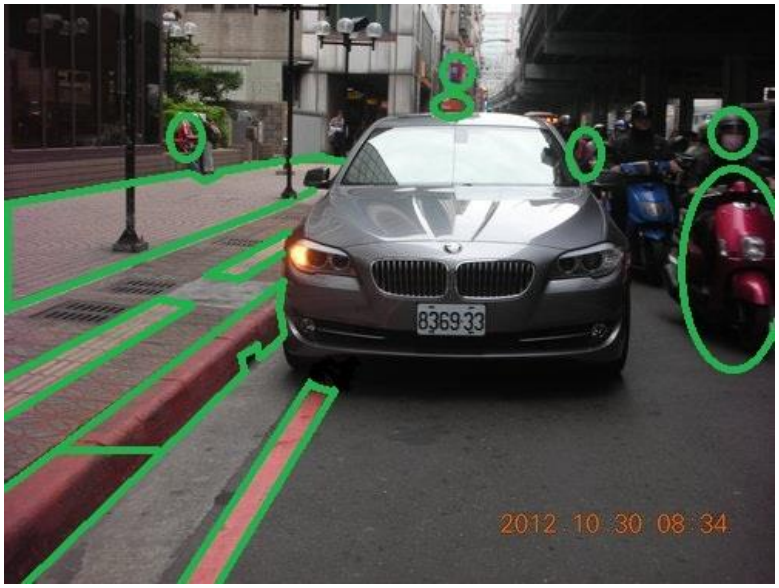
（九）研究時遇到的問題

1.進行 Hough Transform 時，篩選到過多的線段：



如上圖所示，因為條件的設置會遇到很多困難，影像上的雜訊又無法有效地去除。就會產生這種情況，導致後續的處理速度變得非常緩慢，而且也非常難以分析出正確的道路標線及其位置。

解決辦法：



如上圖所示，可以一開始就使用紅、黃、白三種 RGB 值，去篩選出道路標線可能之位置。雖然有時候會有雜訊出現，例如背景有紅色的招牌，但這些雜訊在進行 Hough Transform

之後，都會被過濾掉，這樣就可以有效的減少 Hough Transform 找到的線段。進而提高整個程式的效能，與降低後續分析的難度。

2.如何讓系統與 WEB APP 結合

利用 Java Script 或 PHP 去做影像處理的速度還有待商確，所以必須想辦法結合 C 語言與 WEB APP，或者去查詢一些特殊的方法，可以讓 C 語言與 WEB APP 有效的溝通。

解決辦法：

這個問題牽扯到的技術層面很廣，後來得知要使用 Linux 系統中的 shell 來透過 C 語言傳送 Url(網址)，並在網址中帶值，傳輸到網路頁面，藉此就可以完善的讓 C 語言與 WEB APP 有效的溝通！

3.Codebook 背景相減演算法中 K-means 的分群群數

這個議題很重要，因為我必須確切知道分幾群，才能將 codebook 的功能完整發揮出來。

解決辦法：

後來我在修習 Data Mining 這堂課時找到解答，只要使用 Greedy Method 即可解決這個問題，一開始先找所有要分群的點之質心點，接著尋找與質心點最遠的點，將其當作第一個分群的質心點，然後再找到與第一個分群質心點最遠的點，當作第二分群的質心點，直到找到下一質心點的距離，低於我設定的距離時，則表示已經確定要分幾群了。

（十）後續研究／實作方向

1. 在這學期修習完 Data Mining 的課程後，額外學到很多 Mining 與分析的方法，後續將會利用這些方法，針對顏色與線段做更精密的分析，提升整個系統的效率。

2. 持續精進 PHP、HTML、Java Script 等網頁語言的撰寫能力，改善 WEB APP 的功能，並嘗試上架 Google play 與 IOS 的 app store。

3. 與警察局或市政府洽談，將本系統推廣出去。

4. 結合 RFID 的技術，用其克服多台車輛同時進入影像時，會干擾違規停車判斷的問題。

（十一）參考文獻

[1] 李建興、張凱倫、林應樸，”即時動態車牌辨識”， 技術學刊 第二十五卷 第二期，2010。

[2] 張 瑀、江衍漢、蔡瑋哲，”車牌辨識系統”，逢甲大學資訊工程學系專題研究報告，2010。

[3] 林郁佐，”車牌辨識系統之實作”，南華大學資訊工程學系。

[4] 田智婷，“智慧停車場的車牌辨識系統”，臺灣師範大學資訊工程學系專題研究報告，2008。

[5] 巫明侃，“基於二階二維 Haar 離散小波轉換的車牌偵測方法及其 Google Android 嵌入式系統實作”，2009。

[6] 黃麟鈞、劉良謙、李偉嘉，“自動籃球計分暨追蹤系統”，2011。

[7] 張傑閔、張厥煒，運動視訊場景中動態物件搜尋與追蹤方法(A Dynamic Object Searching and Tracking APProach in Sports Video Scenes) 臺北科技大學學報第四十之一期,2006,pp.63-66.

[8] 林珏汝、羅苑瑜、許芝郡、鍾旻樺，實作codebook背景模型辨別連續影像前背景，東海大學資訊工程系。