

第十二屆旺宏科學獎

成果報告書

參賽編號：SA12-093

作品名稱：轉彎處車流及人流的智慧型調控系統

姓名：張雅量

關鍵字：即時車輛辨識、機器學習、交通系統

研究題目：轉彎處車流及人流的智慧型調控系統

摘要

全國警方開始行人路權執法，強力取締不禮讓行人的汽機車。但是在沒有警方控管的路口，禮讓行人的觀念依然薄弱。除了駕駛人觀念的培養、警察的監督外，如果有自動化的調控系統、轉彎號誌來協助，轉彎處的交通會更加容易控管、取締。

本研究乃全國科展得獎作品「機器學習——即時車輛影像辨識」之延伸，使用 OpenCV 作為影像處理之基礎，以其機器學習方法 Haar 特徵檢測，訓練電腦辨識轉彎路口的車輛影像，可達 95% 車輛整體辨出率，84% 平均正確率及 2.4% 誤判率，接近可以應用的標準。而本研究進一步結合 OpenCV 提供的行人辨識，希望建立一套協助轉彎處交通的智慧型調控系統，能取代或協助交通警察，排解轉彎處的人車衝突，指示車輛禮讓行人，並維持車流順暢。礙於無法在路口架設硬體設備(轉彎燈號、調控紅綠燈秒數)，本研究以實地拍攝的轉彎路口影片為基礎，並用電腦模擬路口情況，藉由分類器自動檢測車流量、人流量，再模擬燈號提示車輛或行人通行，使人、車流流暢，停滯時間短。

目前模擬燈號的部分，經過加入時間軸、Y 座標判斷、行人燈納入考量等等改進後，可達約 80% 的正確率，27% 誤判率；違規車輛檢測可達 50% 的正確率。

壹、研究動機

每天早上經過學校旁邊的路口，都會看到教官、義交辛苦的指揮交通。所謂指揮交通，其實只是擋住車輛讓學生通過，或者擋住學生讓車輛通過。這樣不斷重複的工作，無趣又危險，若能以自動化系統協助或取代，或許能改善問題。而為了開發轉彎處車流及人流的智慧型調控系統，第一步必須了解即時的交通狀況，也就是即時的車流狀況。

目前對於即時影像辨識，OpenCV 已訓練出全套的行人辨識分類器，亦有公司開發出車牌辨識的程式，不過尚無可完整辨識車輛的分類器。於是本研究便從即時車輛辨識著手，訓練能辨識出轉彎車輛的分類器，進而研究車輛與行人的交互關係，建構一套轉彎處的交通系統。

貳、研究目的

本研究協助轉彎處人車通行流暢為主要目的，希望藉由訓練電腦辨識轉彎處的車輛，並結合現有的行人辨識，作出一套智慧型交通系統，能適時調控轉彎處的人、車通行順序，使交通順暢，人、車停滯時間短，且能因應交通事故，協助或取代交通指揮的部份工作。



↑圖 1：本研究主要目的，是建立智慧型調控系統解決人車衝突問題，由於無法實地設置設備，本研究直接拍攝實地影片，建立虛擬路口，並以燈號模擬，調控人車流。

參、研究器材及設備

一、個人電腦

(一)硬體

- 1、CPU：Intel(R) Core(TM) i3-2100 CPU @ 3.10GHz
- 2、RAM：8.00GB

(二)軟體

- 1、Python2.7
- 2、Numpy1.6.2
- 3、OpenCV2.0/2.4.3
- 4、Pygame

二、具錄影功能的數位相機

肆、研究過程

一、研究流程

- (一)車輛辨識方法尋找、測試
- (二)車輛辨識分類器訓練
- (三)分類器測試與調整
- (四)智慧型交通調控系統建構
- (五)系統測試與調整

前三點關於車輛分類器的研究已完成，詳見「機器學習——即時車輛影像辨識」及附錄。本研究繼續第四點，主要著墨於智慧型交通調控系統的建構，及系統的測試與調整。

二、機器學習

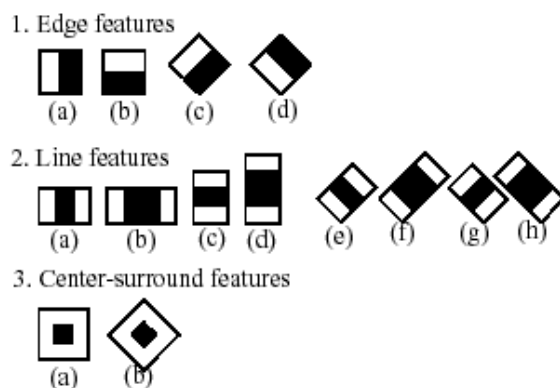
機器學習訓練所得的分類器，是本研究「智慧型調控」的基本依據，所謂「機器學習」主要是設計和分析一些讓計算機可以自動「學習」的算法，並可以從一類從數據中自動分析獲得規律，利用規律對未知數據進行預測。目前已有十分廣泛的應用，如數據挖掘、計算機視覺、自然語言處理、生物特徵識別、搜索引擎等等。

三、OpenCV

本研究用來處理圖片的 OpenCV(Open Source Computer Vision Library) 是一個跨平台的計算機視覺庫。OpenCV 是由英特爾公司發起並參與開發，以 BSD 許可證授權發行，可以在商業和研究領域中免費使用。OpenCV 可用於開發即時的圖像處理、計算機視覺以及模式識別程序。

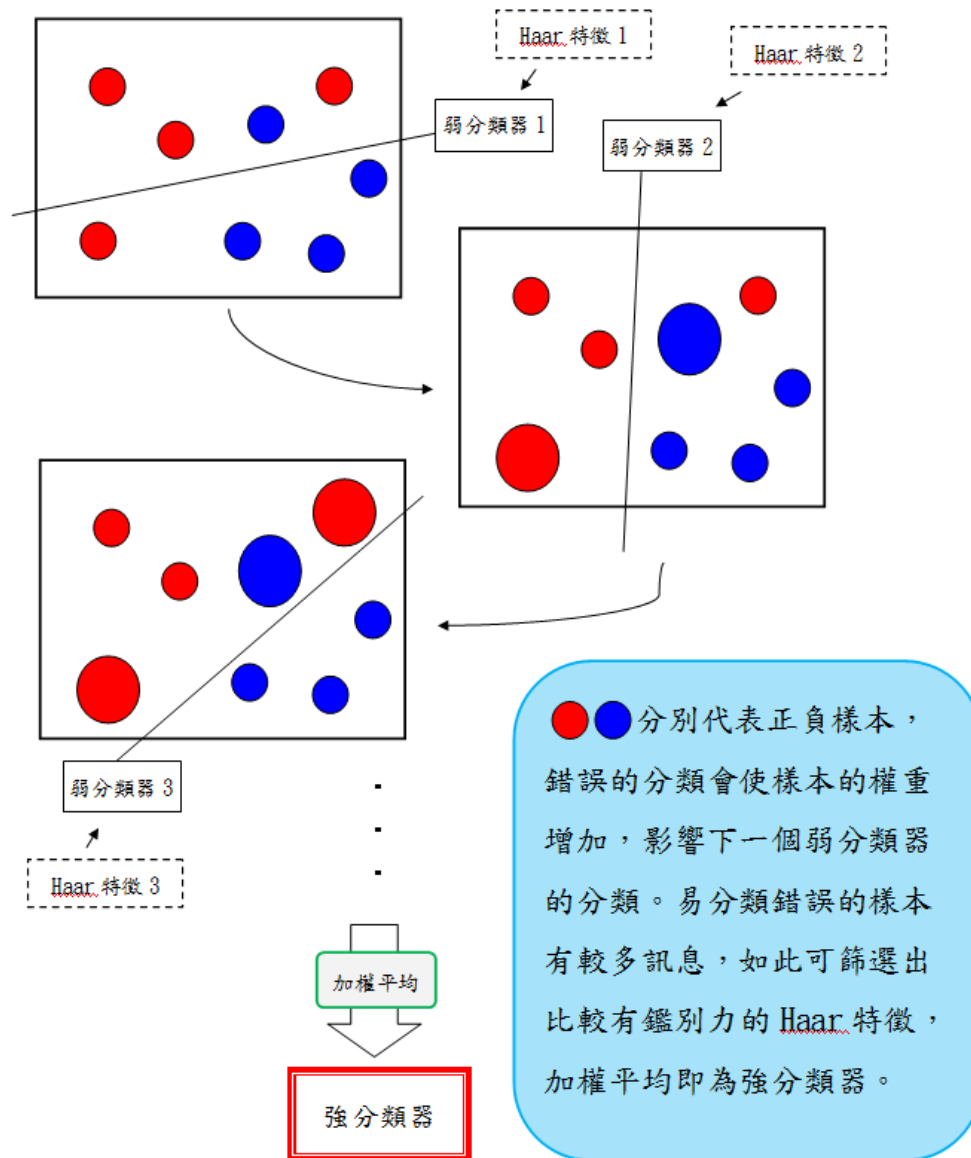
四、Haar 特徵檢測

Haar 特徵檢測是 OpenCV 開發中的程式，其方法大致為，以機器學習的 AdaBoost 方法找出正負樣本中 Haar 特徵的規律，用以辨識未知的圖片。



↑圖 2：常見的 Haar 特徵

圖 3：AdaBoost 訓練示意圖



利用 AdaBoost 方法，以正負樣本訓練，生成的分類器，可即時地偵測影像裡的目標。

五、Haar 特徵檢測之車輛辨識

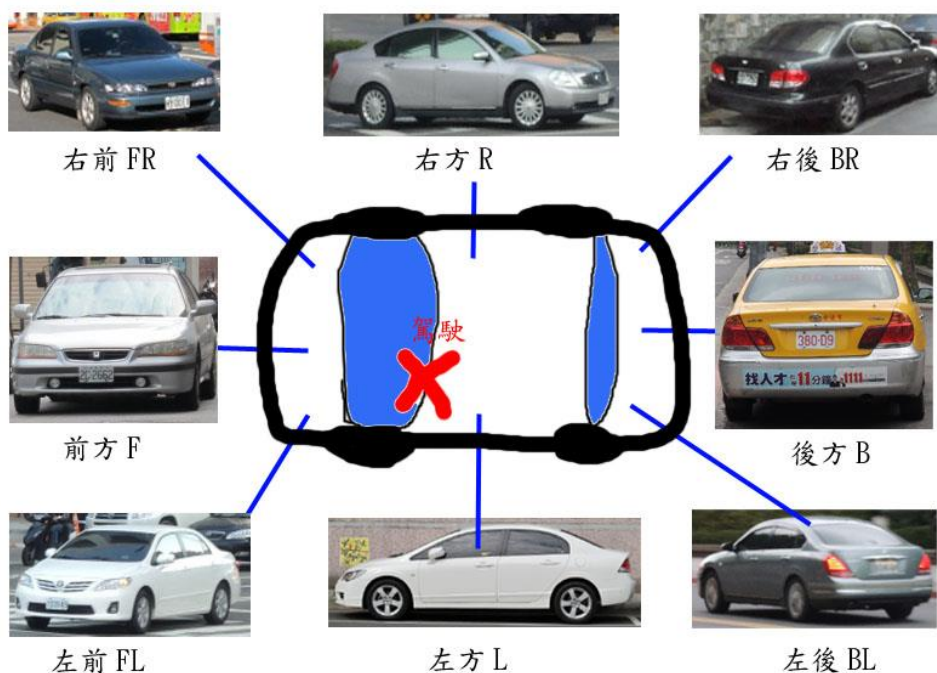
(一)車輛辨識方法尋找、測試

最後選擇以 OpenCV 的 Haar 特徵檢測方法訓練電腦辨識車輛，詳見「機器學習——即時車輛影像辨識」。

(二)車輛方向分割

隨後開始車輛圖片的蒐集，考量 Haar 特徵檢測的特性(左右顛倒的物體特徵完全不同)，本研究將水平角度看到的車輛以 45 度為單位，分

為 8 個方向，見圖 4 所示。



↑圖 4：分割方向示意圖及各方向正樣本範例

(二)正負樣本製作

詳見「機器學習——即時車輛影像辨識」。

(三)各分類器訓練資料

詳見附錄。

(四)訓練器測試方式與用詞說明

1、計算方法

詳見附錄。

2、判斷標準及特殊狀況說明

詳見附錄。

3、數據計算公式說明

$$\text{整體辨出率} = \text{車輛成功辨識數} / \text{總車輛數} * 100\%$$

整體辨出率的意義即是否成功辨識每輛車，一輛車只計算一次。

$$\text{平均正確率} = \text{正樣本成功辨識數} / \text{總測試正樣本數} * 100\%$$

平均正確率則代表分類器的穩定度，是否在可以辨識出每個角度的車輛，重複出現的車輛重複計算。

$$\text{誤判率} = \text{誤判數} / (\text{誤判數} + \text{正樣本成功辨識數}) * 100\%$$

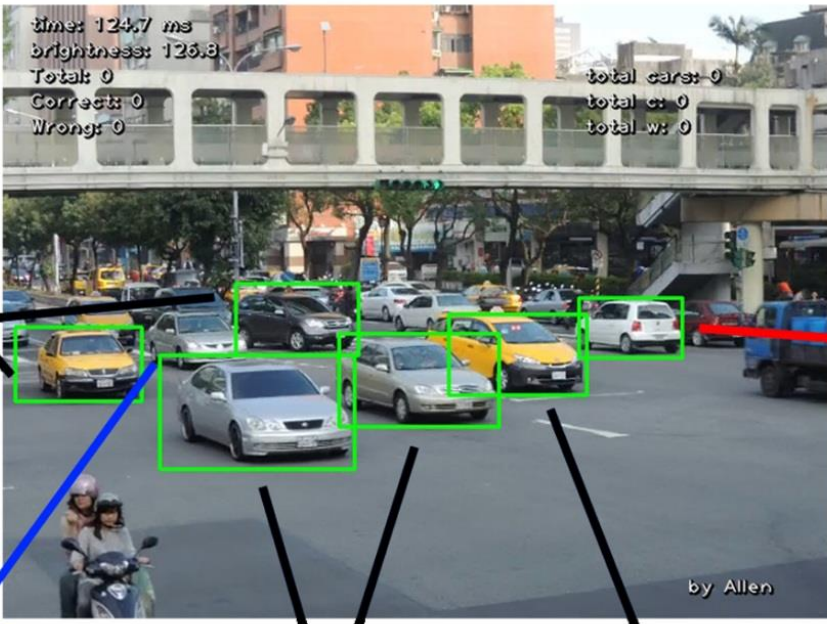
誤判率意謂分類器的可信度，表示辨識出的影像中有多少比例不是車輛。

	車輛成功辨識數 / 總車輛數 = 整體辨出率	正樣本成功辨識數 / 總正樣本數 = 平均正確率	誤判數 / (誤判數 + 總正樣本數) = 誤判率
	0/1 = 0.000%	0/1 = 0.000%	無意義
	1/1 = 100.000%	1/2 = 50.000%	0/1 = 0.000%
	1/1 = 100.000%	2/3 = 66.667%	1/3 = 33.333%
	1/1 = 100.000%	3/4 = 75.000%	1/4 = 25.000%
	1/1 = 100.000%	3/5 = 60.000%	1/4 = 25.000%

↑圖 5：車輛分類器測試數據計算說明

4、範例

若為 FR 方向分類器測試，則總正樣本數 = 6，成功辨識 = 5，誤判 = 1。
 對此圖片正確率 = $5/6 = 83.333\%$ ，誤判率 = $1/(5+1) = 16.667\%$ 。



time: 124.7 ms
 brightness: 125.3
 Total: 0
 Correct: 0
 Wrong: 0

total cars: 0
 total c: 0
 total w: 0

略偏左方向與偏右方向，仍納入計算，成功辨識 +2，總正樣本數 +2

RL 方向，誤判 +1

總正樣本數 +1
 未辨識出，降低正確率

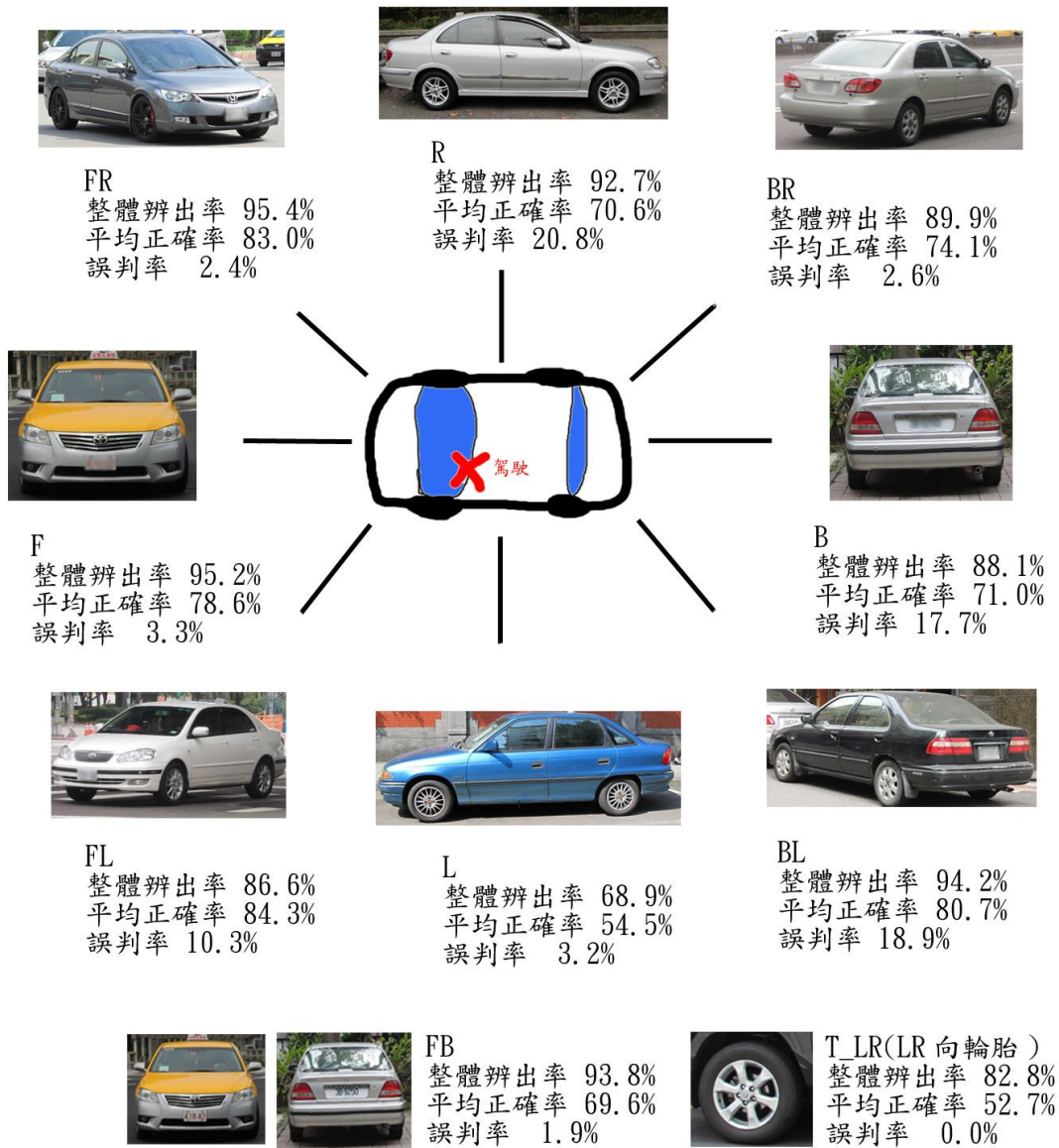
成功辨識 +2
 總正樣本數 +2

受遮蔽，但方向正確仍納入計算。
 成功辨識 +1，總正樣本數 +1

by Allen

↑圖 6：車輛分類器測試數據計算範例

(五)現階段各方向分類器測試結果



↑圖 7：車輛分割方向示意圖及結果測試

詳細結果參見附錄。

六、智慧型交通調控系統說明

(一)硬體設置

理想的系統處理以影像為主，道路攝影機為必要設備，用以取得即時交通資訊，另可設置雷達、重量感測器等作為輔助。設置燈號、發音器或柵欄，用以警示或阻擋人車非法通行。所有系統線路連接至控制箱，由晶片及遠端電腦控制。

本研究的智慧型交通調控系統並未架構硬體設施，是以實地拍攝(與路口攝影機同高，見圖 8)的路口影片，在電腦建立虛擬交通路口，標示人車位置，並以模擬燈號排解人車衝突，並調控人、車流。



↑圖 8：實地拍攝錄影機架設高度示意圖

(二)系統運作機制及功能

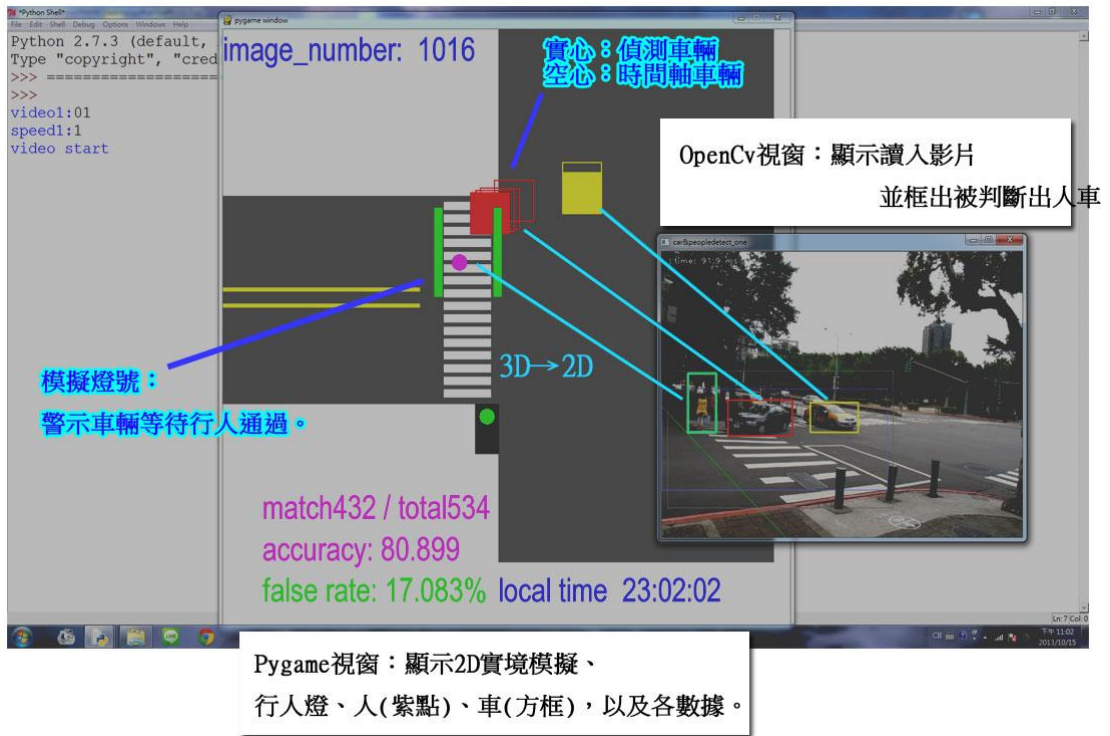
藉由車輛辨識，可以得知車輛數與車輛位置，亦可推斷車流量及車速，掌握即時的車流情況。同理可得知即時的人流資訊，從而評估交通情況，加以調控。

- 1、人、車通行順序指示：依人、車流量，指示人或車通行，並以燈號、哨音或柵欄，警示、阻擋等待的一方。
- 2、調控紅綠燈秒數：開發演算法，找到最有利人車順暢的交通控制，車流量改變時調整紅綠燈秒數。
- 3、違規檢測與記錄：偵測違規人、車，進行記錄，方便取締。
- 4、意外檢測與因應：安全也是系統實用的必要條件，需設置意外檢測、因應機制，面臨交通意外時，能適時通知、傳送現場影像給相關單位，並避免意外擴大。

七、智慧型交通調控系統建構

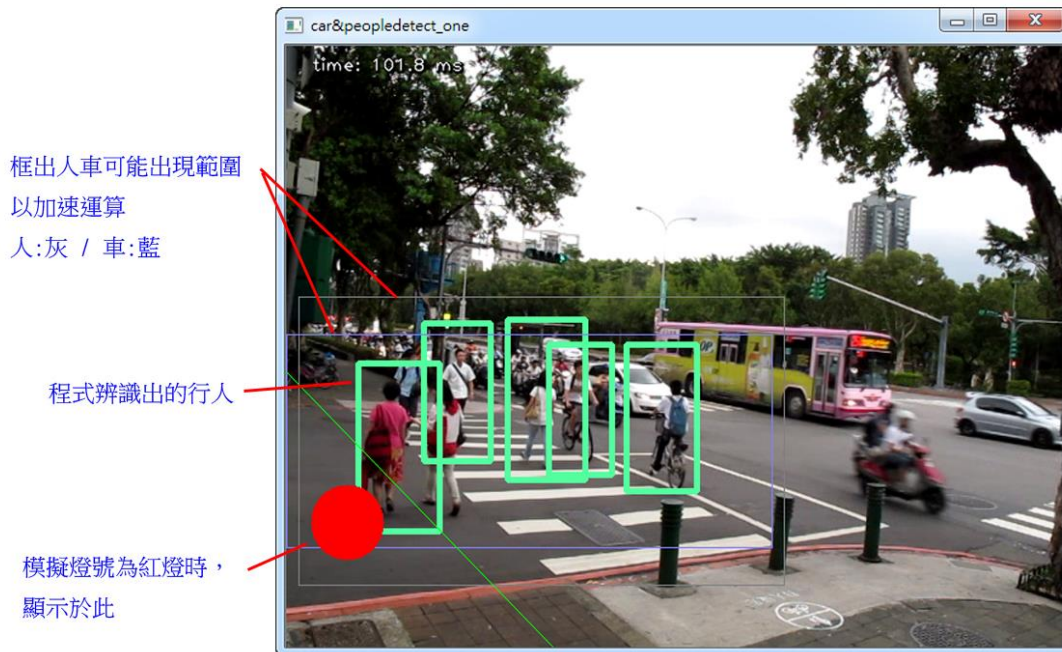
(一) 程式架構說明

為方便呈現結果及利於研究進行，將 OpenCv 視窗的實境拍攝影片資訊轉換成 Pygame 視窗的 2D 平面模擬，相當於該 T 字路口鳥瞰圖。檢測時，以 OpenCv 視窗辨識出之矩形座標為依據，將其顯示在 Pygame 視窗上，見圖 9。



↑ 圖 9：程式示意圖，左方為路口的平面模擬圖，右方為真實的路口影片。

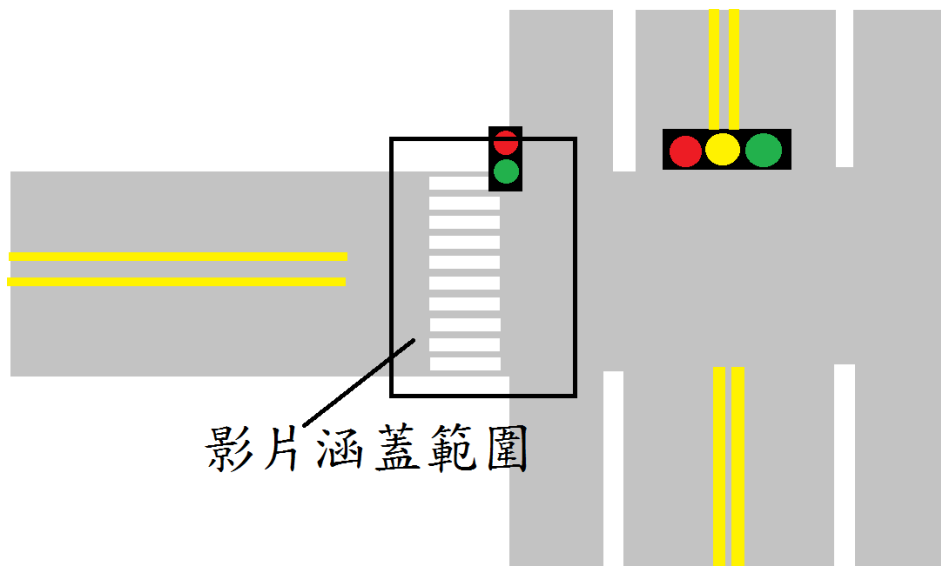
由於檢測整張圖片耗費過多運算時間，於 OpenCv 視窗畫出兩個長方形範圍，分別表示人及車可能出現之範圍，程式僅在此範圍運算，見圖 10。



↑圖 10：限制運算範圍示意圖

(二) Python 程式運行流程

- 1、輸入及讀取影片檔，於螢幕播放。影片為 14 部不同時段同一路口、角度的實地錄影，內容為 T 字路口(金華新生路口)，人車頻繁通過之時段，依取得順序編號 1~14，共約 2 小時。以固定的比率取樣其中圖片，讓程式運算並計算其數據。



↑圖 11：影片涵蓋範圍示意圖

- 2、置入分類器，於影片視窗顯示辨出車輛與行人。

車輛分類器：FR03、F03、FL03、LR03

行人分類器：由 OpenCV 提供的 hogcascade_pedestrians.xml

辨識範圍：僅單方向斑馬線（上圖影片涵蓋範圍）。

- 3、模擬路口實境，使用 Pygame 繪出路口平面圖。
- 4、分類器運算，影片視窗(OpenCv 視窗)框出辨識出的車輛及行人，並同時依據座標在 Pygame 視窗顯示。
- 5、根據程式辨識出的人車數、位置，計算人車關係。目前以行人優先，只要行人出現在斑馬線上，便使模擬燈號亮紅燈，阻擋車輛通過。

八、系統測試與改進：模擬燈號

(一)用詞說明

主要測試模擬燈號的調控是否恰當，原則上有行人在斑馬線左半部時，模擬燈號必須亮紅燈以阻擋轉彎車輛通行。

Total：人工判定模擬燈號應該亮紅燈的總圖片張數。

Match：實際測試在應該亮紅燈時亮紅燈的總圖片張數。

Wrong：在不該亮燈時亮紅燈的總圖片張數。**應該亮紅燈時沒亮並非誤判，而是燈號正確率降低。**

燈號正確率(Accracy) = Match / Total * 100%

燈號正確率，簡稱正確率用來表示，模擬燈號是否在應該阻擋車輛時正常運作(亮紅燈)。

燈號誤判率(False Rate) = Wrong / (Wrong + Match) * 100%

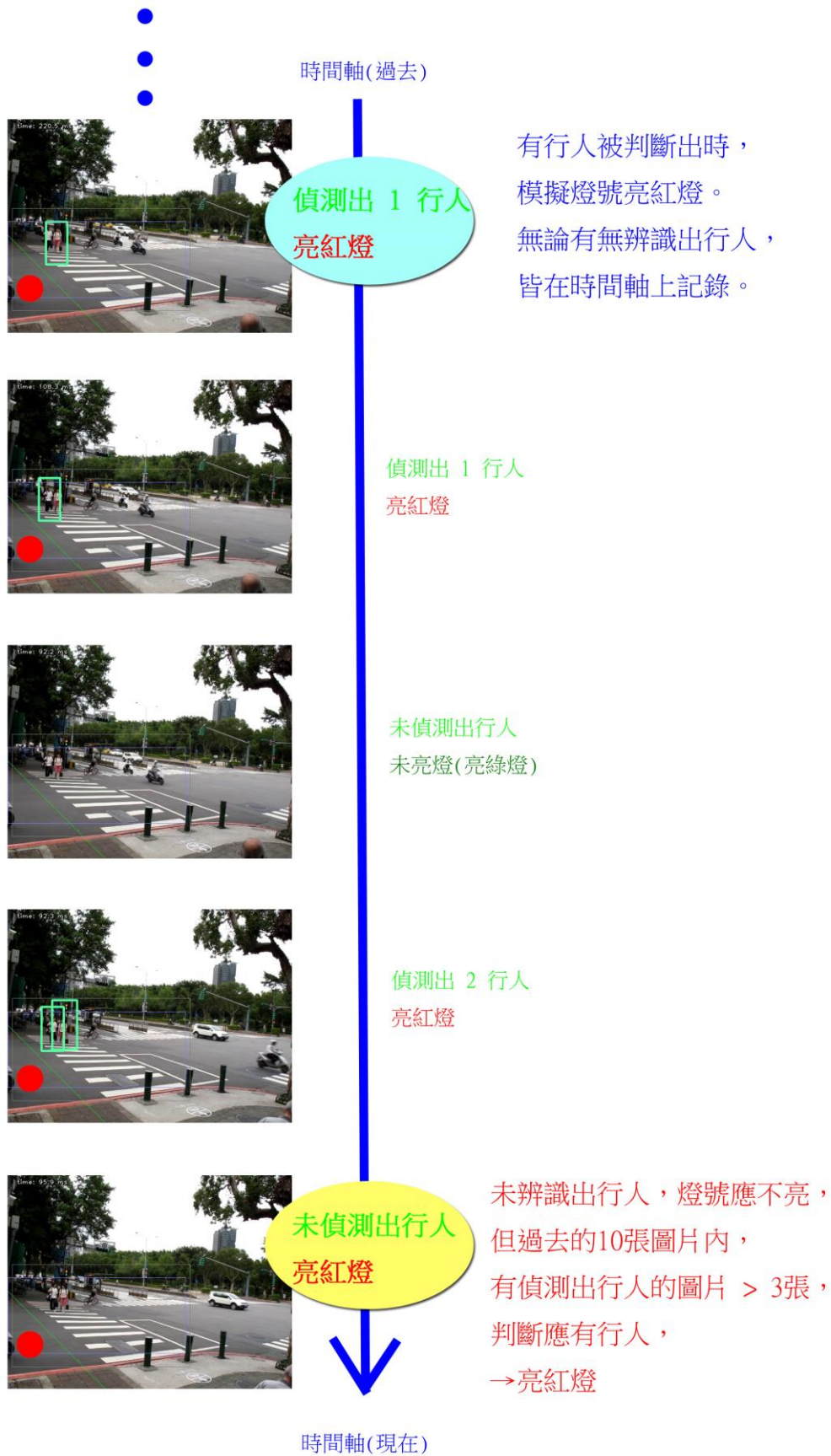
燈號誤判率，簡稱誤判率用來表示，模擬燈號亮紅燈的時段中，有多少比率是錯誤的(沒有人卻亮了紅燈)。

(二)首次測試

首次測試結果不甚理想，正確率不到 70%。推測是因為只在行人被辨識出時才正確判斷，而行人辨識分類器正確率偏低，且位置不精確所致。

(三)加入時間軸(殘影)

為改進正確率問題，在程式內加上「時間軸」之概念，即「殘影」，見圖 12。行人分類器的誤判率並不高，只是欠缺穩定度。只要加入曾經辨識出的行人，作為判斷依據，當辨出行人時將其座標紀錄於時間軸(陣列)，並以時間軸上座標之個數、位置差來判斷是否有應辨出之行人，便可提高正確率。



↑ 圖 12：時間軸示意圖

圖13：有無殘影對燈號**正確率**之影響

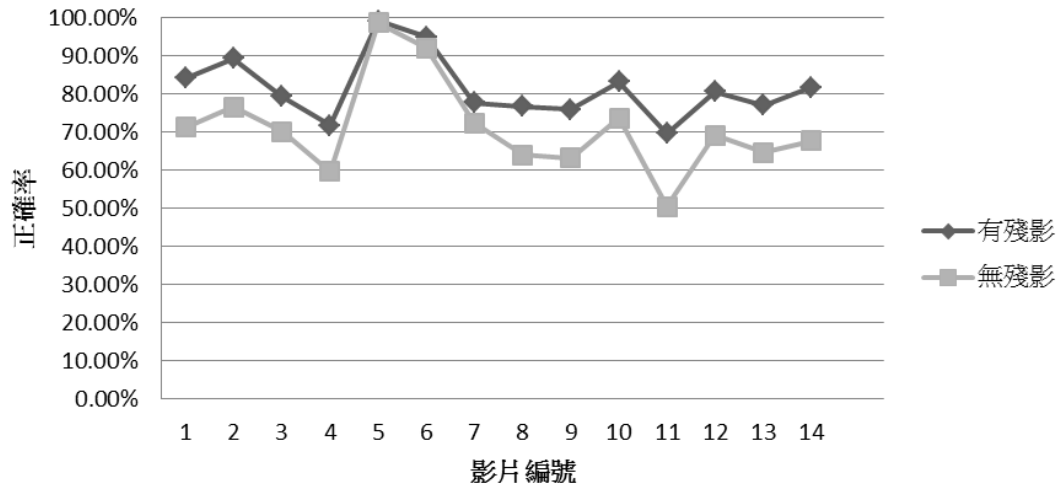


圖14：有無殘影對燈號**誤判率**之影響

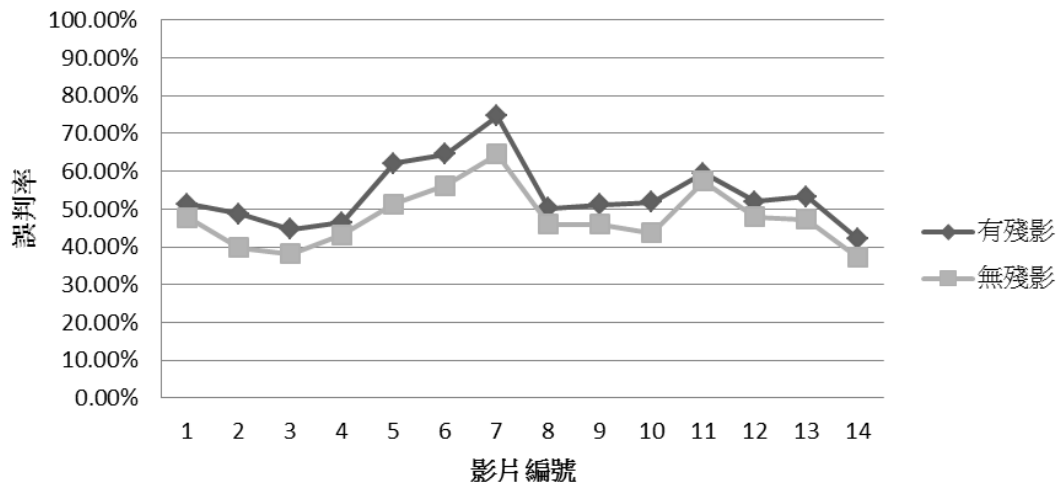
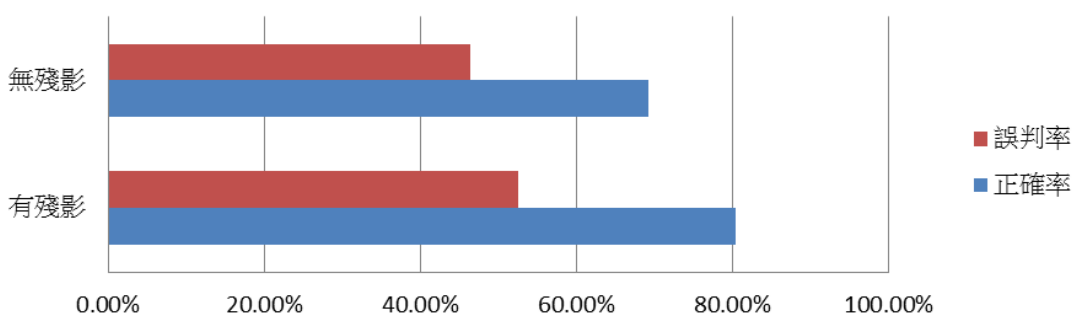


圖15：有無殘影對燈號**正確/誤判率**之影響



↑ 測試結果正確率上升至 80%，但同時誤判率亦從 46% 上升至 52%。

(四) 圖片取樣比率

由於影片是以 30 張圖/秒紀錄，若以程式 30 張圖都處理，運算時間會大於一秒(一張圖片約需 0.06~0.11 秒處理)，無法及時處理資訊。

為改善此問題，進行每秒分別取 30、10、6、3、2 張圖處理的測試，找出適當的圖片取樣比率，以便程式順利執行。

圖16：每秒取樣比例對燈號**正確率**之影響

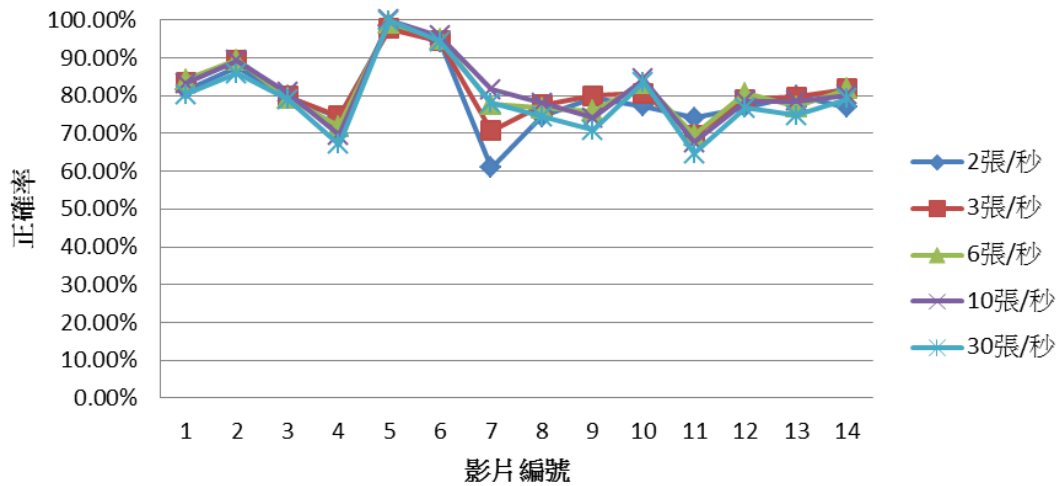


圖17：每秒取樣比例對燈號**誤判率**之影響

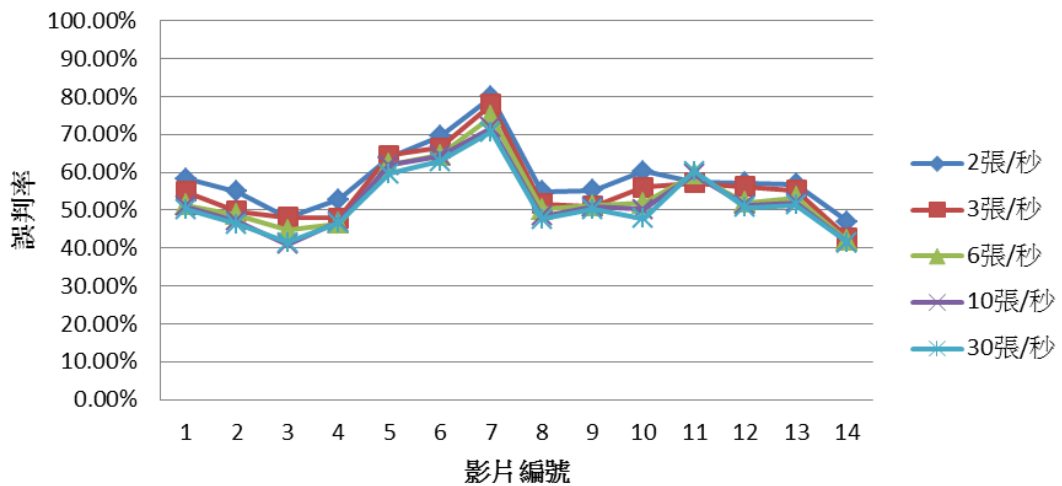
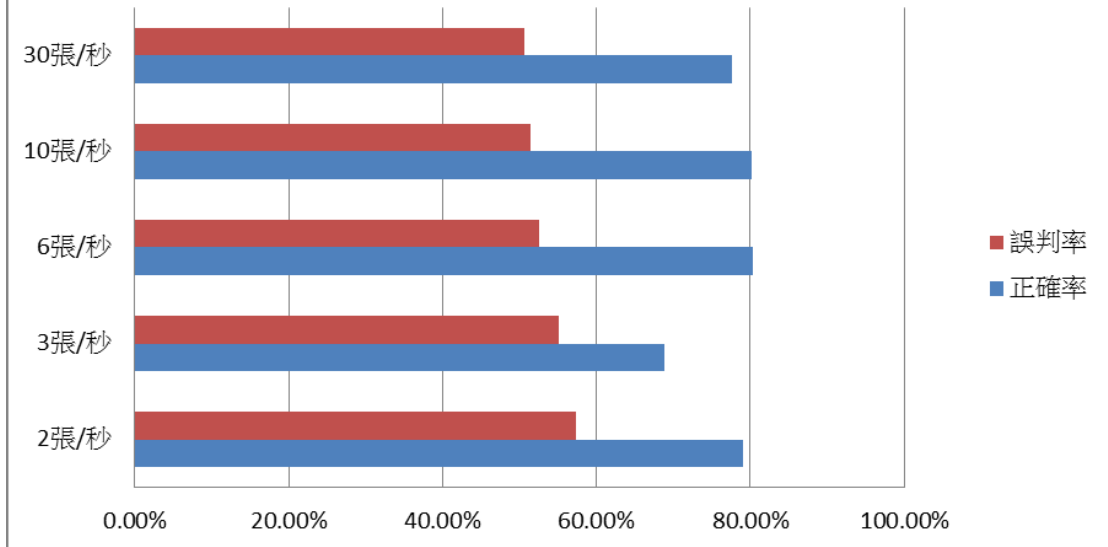


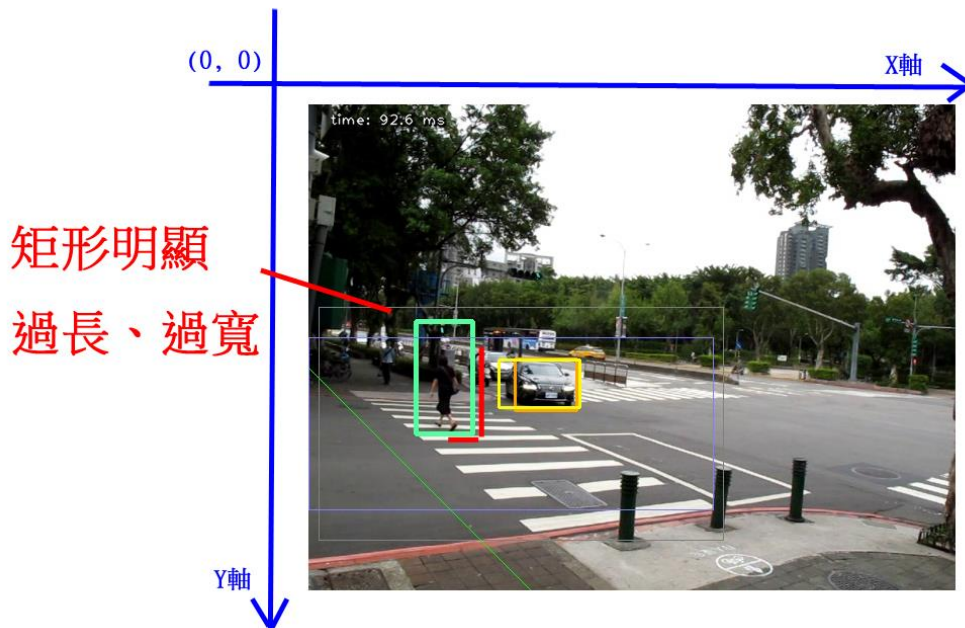
圖18：每秒取樣比例對燈號**正確/誤判**率之影響



↑由圖可見結果無太大差異，但仍是以前略過越少張圖片的正確率較高、誤判率較低。權衡資訊量及圖片的取樣比例影響，取每秒取 6 張圖做判斷(每 5 張判斷 1 次)。

(五)Y 軸座標判斷

原判斷時，只取被判斷出之矩形的 X 軸座標判斷位置，但人流辨識時，有時會因為所框出的範圍不精確(寬度太大)而導致誤判(見圖 19)。於是加入 Y 軸座標作為判斷標準，使判斷更精準。



↑圖 19：影片座標及矩形過寬示意圖

圖20：加入Y座標判斷對燈號**正確率**之影響

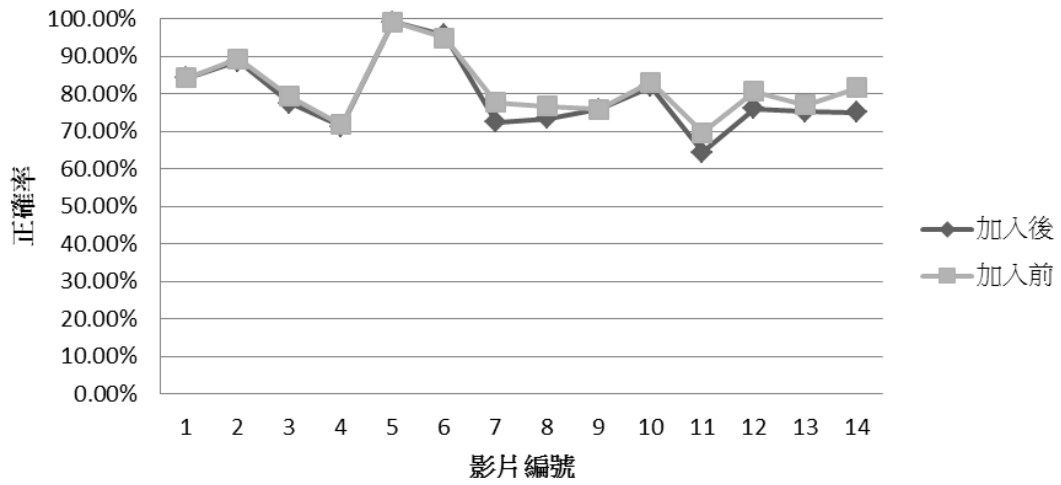


圖21：加入Y座標判斷對燈號**誤判率**之影響

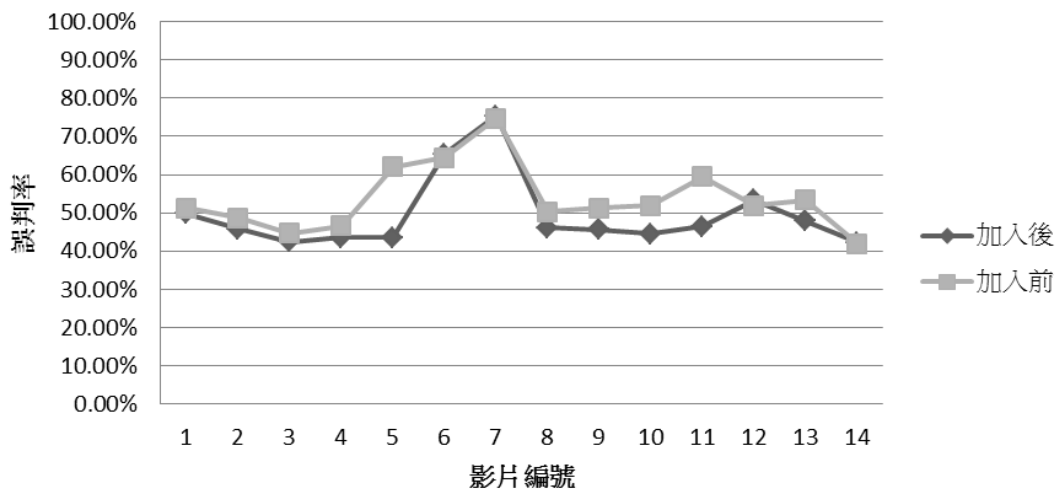
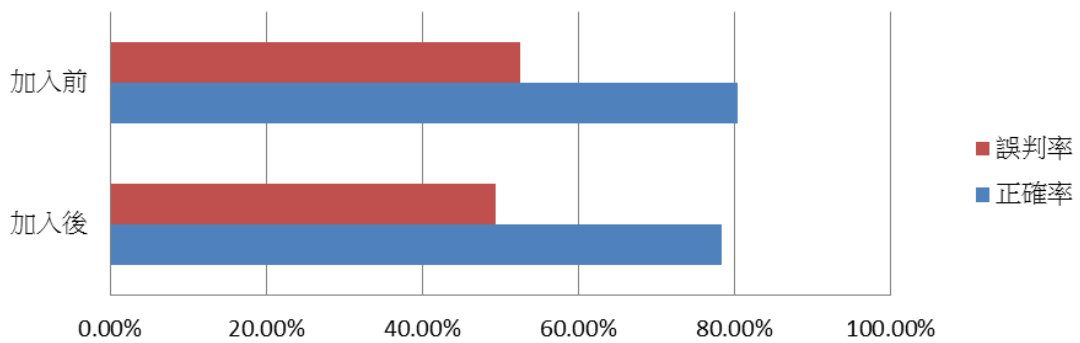


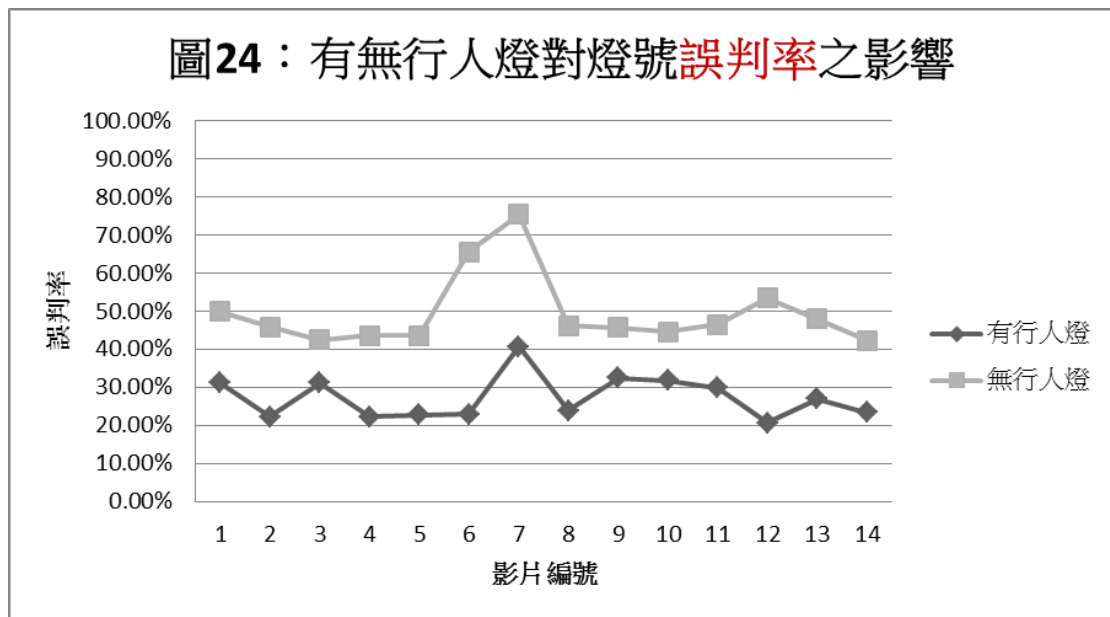
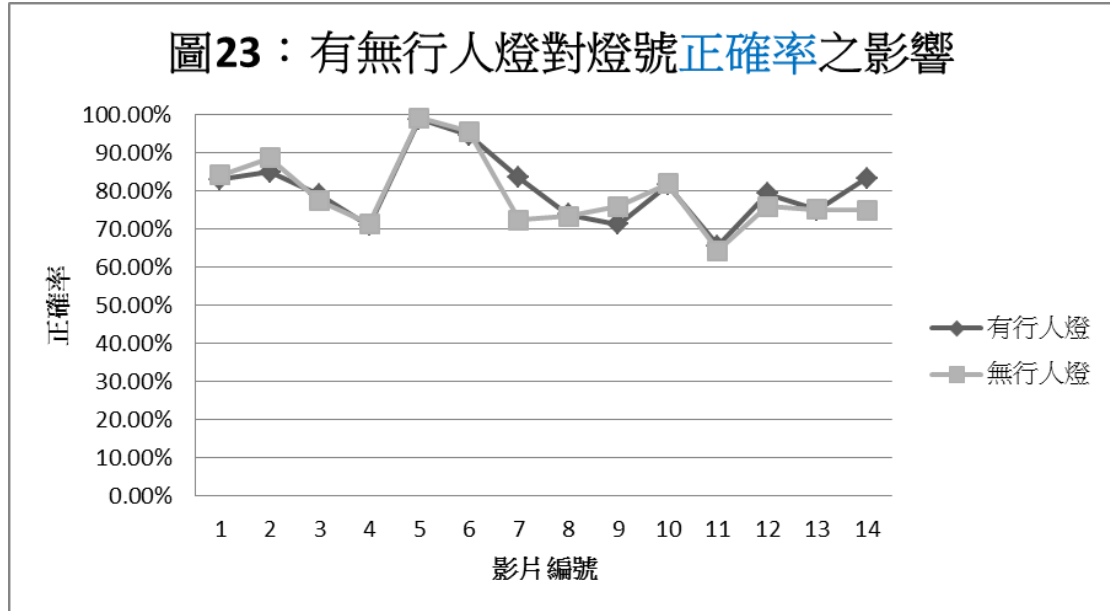
圖22：加入Y座標判斷對燈號**正確/誤判率**之影響

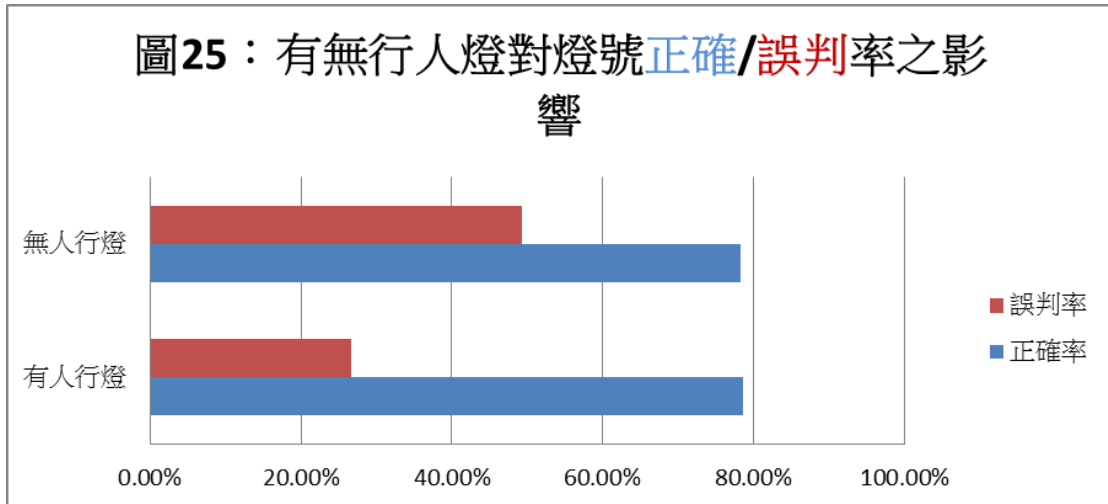


↑ 正確率無太大差別，但誤判率明顯降低。

(六)行人燈

由於行人正確通過時間皆在行人燈(有小綠人指示行人通行的燈號)為綠燈時,故於 Pygame 加入行人燈(與影片中行人燈亮同號),使行人燈為紅燈時,避免因行人過於靠近斑馬線所造成的誤判。

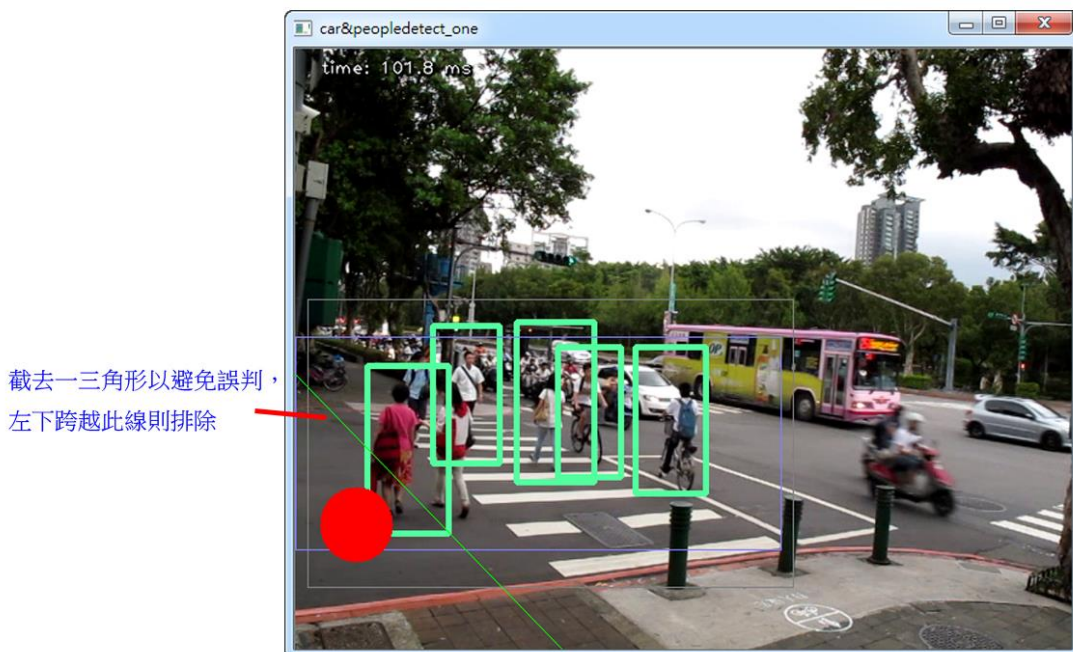




↑減少運算量，並同時使誤判率下降。

(七)OpenCv 辨識範圍修正

測試時，常有行人等待紅綠燈時站在從影片播放角度重疊斑馬線處(影片視窗左下方)，造成誤判率提升，故修改 OpenCv 視窗之辨視範圍，將該矩形之左下截去一三角形，見圖 26。



↑圖 26：影片辨識範圍修正示意圖

圖27：辨識範圍修正對燈號**正確率**之影響

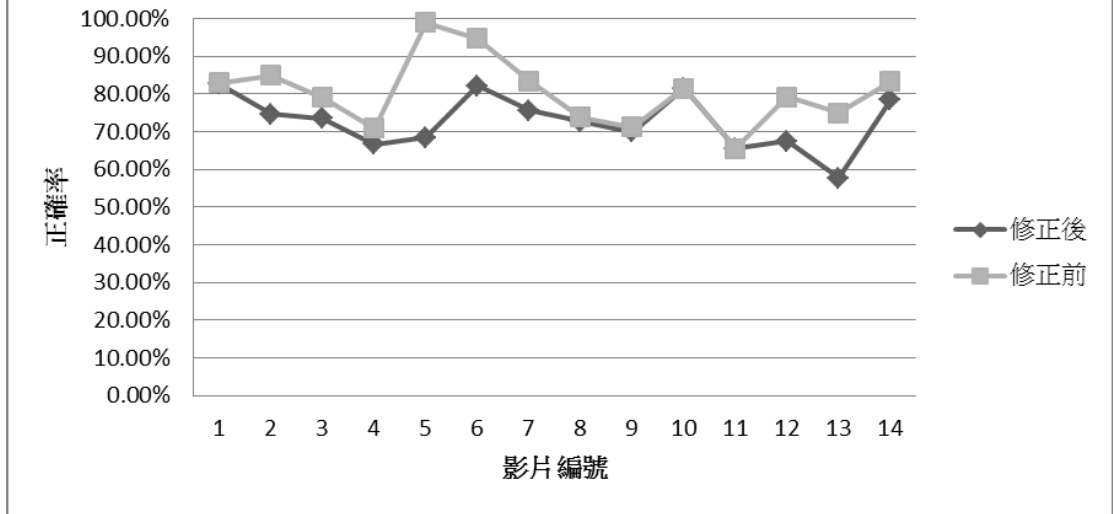


圖28：辨識範圍修正對燈號**誤判率**之影響

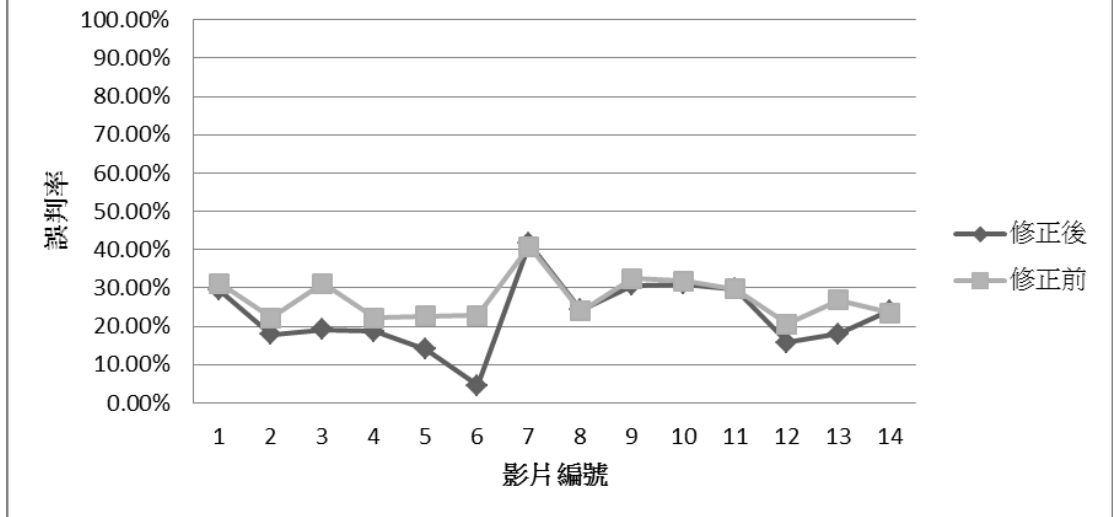
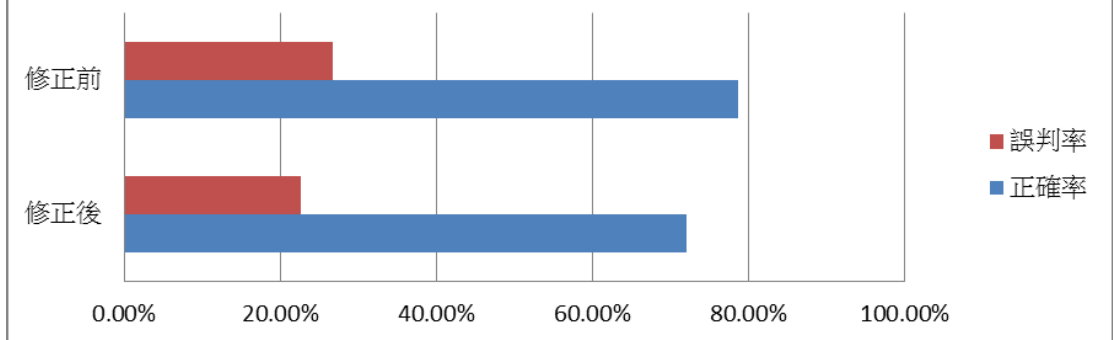
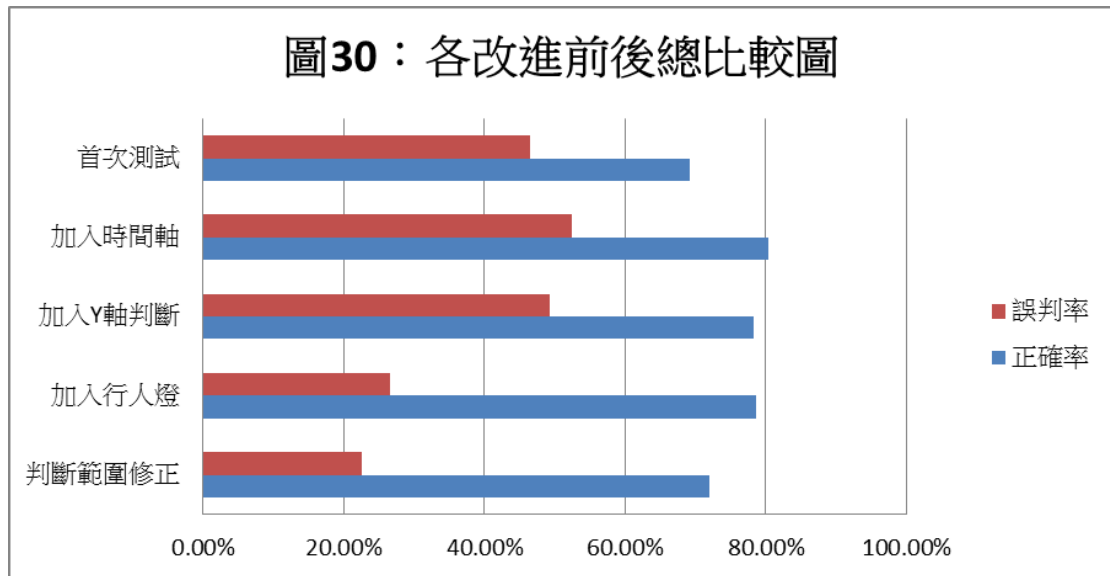


圖29：辨識範圍修正對燈號**正確/誤判率**之影響



↑ 由於位置正確的人也常因判斷位置不精確而被誤判，修正判斷範圍後正確率與

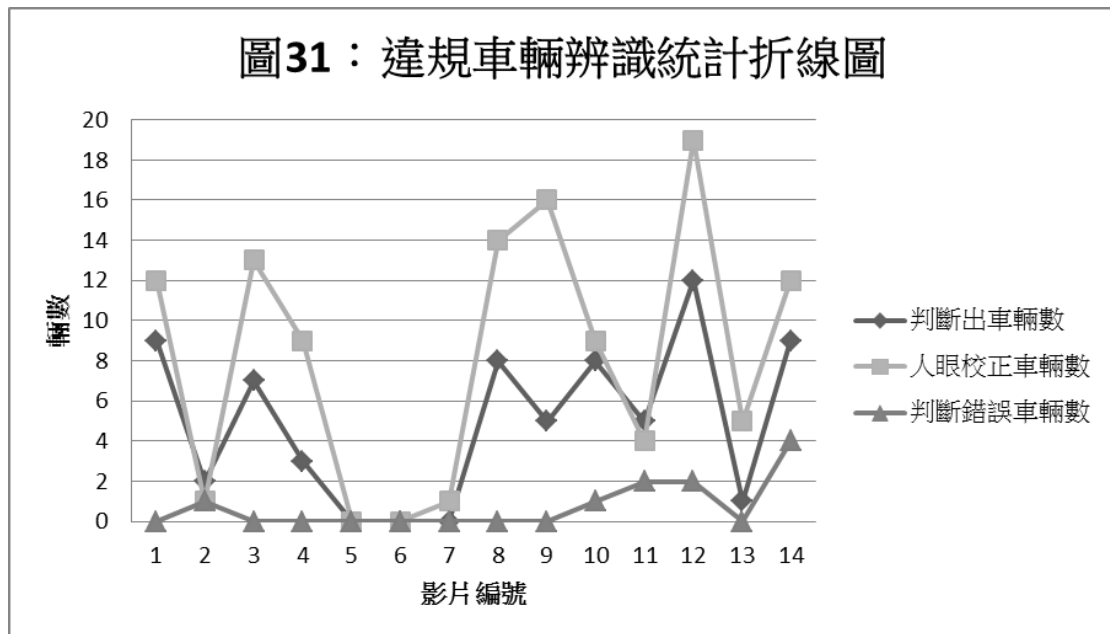
錯誤率同時下降，不甚理想。

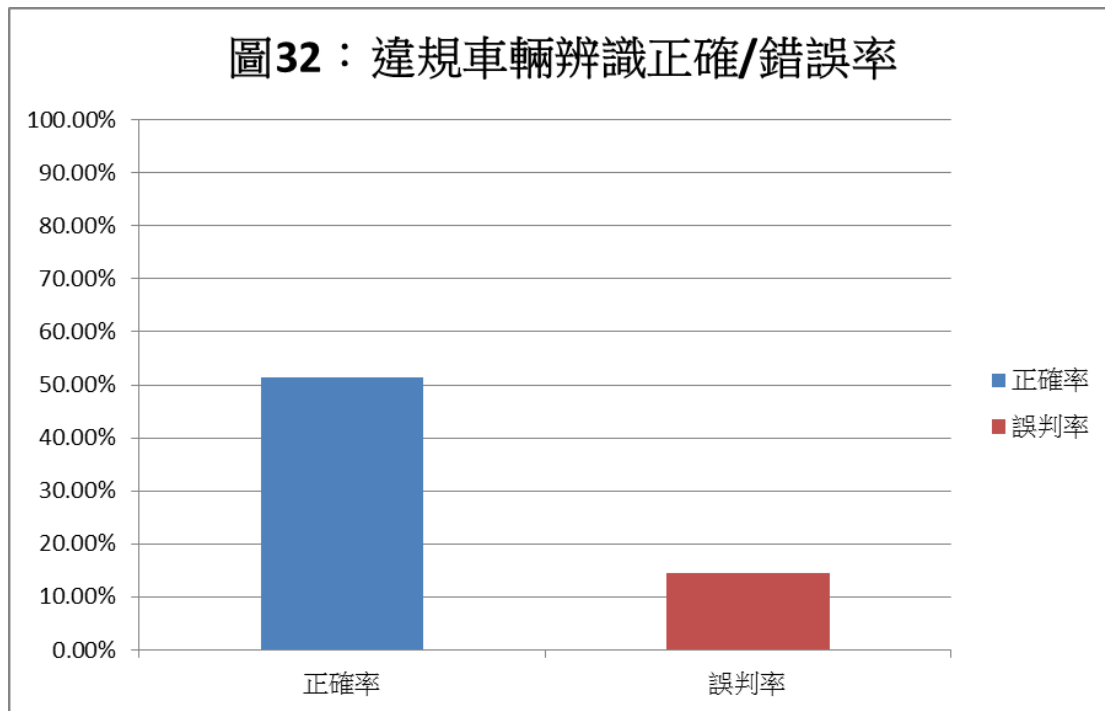


九、系統測試與改進：違規車輛檢測

完成假想燈號後，結合車輛辨識的分類器，便可檢測無視紅燈轉彎的違規車輛。以人工計算違規的車輛數，再進行測試。

程式檢測出F方向的車輛，代表車輛在等候行人通過；檢測出FL或L方向的車輛，代表車輛正通過斑馬線，若此時假想燈號是紅燈，該車輛及違規。亦使用時間軸的概念，增加正確率。實際應用時亦可在地上設置重量檢測，判斷是否有車輛經過斑馬線，可穩定判斷。





↑ 因行人通過時常擋住後方車輛，正確率未達理想，若能增加反方向的攝影機，必能大幅提升正確率。

伍、結論

一、車輛辨識

本研究將車輛圖片分為八個方向，分別進行訓練，並直接以影片測試。可達到 95% 的整體辨出率，84% 的平均正確率，2.4% 的誤判率，說明本研究所訓練的分類器，已可以穩定的辨識出車輛影像。

二、智慧型交通調控系統：模擬燈號

(一) 初步建構

本研究使用 Python 建立能輸入影片並進行檢測的程式，利用 OpenCV 公開之行人分類器，辨識出行人時調控模擬燈號，阻擋轉彎車輛，但初步測試正確率不到 70%，且誤判率過高。

(二) 系統改進

經過加入時間軸、Y 座標判斷、行人燈納入考量等等改進後，可達約 80% 的正確率，27% 誤判率，若再自行訓練行人辨識器，或同時設置不同方向的攝影機，應有更好的成果。

三、智慧型交通調控系統：違規車輛檢測

結合車輛辨識及模擬燈號系統，本研究可即時模擬路口實境並檢測違規車輛。測試結果顯示，系統目前能檢測出約 50% 的違規車輛，仍有車輛被遮蔽等問題待改進。

陸、討論及應用

一、優缺點

車輛辨識器擁有高辨出率、正確率；行人辨識結合 Python 系統後亦有不錯的辨出率，且兩者皆能及時辨識，並透過 Pygame 視窗將路況完整、清楚的呈現；而且僅需一台普通畫質的攝影機，此為系統顯著的優點。但車輛辨識時因角度問題或被行人遮擋導致兩者無法配合，是未來改進的方向。

二、未來展望與應用

本研究之系統可自動偵測行人車，並以燈號警示車輛，排解人車衝突。但是目前一律是行人優先，如果有塞車的情況，可能要切換成車輛優先等其他模式，才能做到完全的智慧型調控。未來可嘗試實際架設燈號，協助或取代義交的工作；抑或參照人車數量調控紅綠燈秒數，使交通更順暢。

除上述目標外，人、車辨識的系統亦可應用於：

1、車流、人流量統計

若能配合車牌、人臉辨識避免重覆計算，即可紀錄道路之人車流量，或追蹤特定人車。

2、輔助自動駕駛

及時辨識周遭人車有助於自動化駕駛。

3、配合定位系統指示救護車最佳行駛路線。

4、都市車流控管

若都市各大路口都架設本系統，即可做更高階的車流調控，藉由調整紅綠燈秒數，及車輛分流，適當紓緩塞車情形，並徹底檢測違規車輛。

另外，亦可自行訓練合適的路口行人辨識器，或架設兩台攝影機，以不同方向檢測，並結合其他的辨識方法，以提高燈號的正確率，優化系統運作。

柒、參考資料

一、機器學習相關網站

- (一) LibSVM 官方網站
- (二) 維基百科：機器學習條目、SVM 條目
- (三) Online courser：Machine Learning 相關課程








二、OpenCV 與 Haar/LBP 檢測相關網站


- (一) 維基百科：OpenCV 條目、Local binary patterns 條目
- (二) OpenCV 官方網站
- (三) 周明才，OpenCV 之 HaarTraining 算法剖析，2008.10.08 第二版
- (四) Naotoshi Seo：Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features
- (五) Robotics@Cyborg：How to make your own haar trained ".xml" files
- (六) Computer Vision Software：FAQ: OpenCV Haartraining
- (七) Step by Step：OpenCV haartraining 的 xml 檔製作
- (八) Stackoverflow.com：所有關於 OpenCV 之 HaarTraining 的問題、部分 OpenCV 語法問題(繁多不及備載)

捌、附錄

一、車輛辨識

(一)各分類器訓練資料

方向	分類器 編號	正樣本 數(張)	負樣本 數(張)	負樣本加入 其他方向車 輛	長寬 (像 素)	訓練約 略耗費 時間
FL 	FL01	168	102	無	34*20	1 天
	FL02	417	532	R、BR、B 各 100	32*20	7 天
	FL03	628	404	R、B 各 100	38*20	1.5 天
F 	F01	127	80	無	24*24	8 小時
	F02	397	232	無	24*24	2 天
	F03	611	500	R、BR、L 各 100	24*20	1 天
FR 	FR01	197	102	無	28*20	1 天
	FR02	401	232	無	34*20	9 天
	FR03	664	323	無	34*20	3.5 天
R 	R01	112	80	無	40*20	1 小時
	R02	311	483	F、BL 各 100	46*20	5.5 天
BR 	BR01	158	107	無	32*20	1 小時
	BR02	330	533	F、FL、L 各 100	34*20	4 天
B 	B01	142	80	無	22*20	1 小時
	B02	389	400	R、L 各 100	24*20	15 小時
BL 	BL01	134	100	無	32*20	1 小時
	BL02	323	300	F 100	38*20	1 天
L 	L01	135	80	無	40*20	6 小時
	L02	354	549	FR、R、BR 各 100	40*20	6 天
FB  	FB04	1095	604	R150、L150、 BR50、BL50)	24*20	8 小時

T_LR		T_LR02	600	200	無	20*20	2.5 天
------	---	--------	-----	-----	---	-------	-------

(二)各數據測試方法

測試影片格式為 640*480，fps=30 的.MOV 檔案。分類器在讀入影片後，會針對每張圖片進行辨識，並框出已辨識出的車輛(亦即每秒程式會辨識 30 張圖片)，為計算分類器辨識的各項數據，每秒只取出 5 張圖片，再手動計算各項數據。

(三)判斷標準及特殊狀況說明

對於每個分類器，僅有該方向的車輛列入計算。不符合訓練時正樣本格式的車輛，如模糊、受遮蔽等等，不列入圖片總車輛數、總測試正樣本數；但若程式有偵測出，則等同於符合正樣本格式，列入各項數據。

對於影片中取出的每張圖片，會先計算出現的車輛數、成功辨識的車輛數(程式框出)及誤判的車輛數(程式框出不是車輛的東西，沒有被框出來的車輛不算誤判)，最後再進行加總。

總車輛數：影片中出現的車輛總數，一輛車子只算一次。

車輛成功辨識數：出現的車輛中，被程式辨識出的數量。一輛車子只算一次。

總測試正樣本數：從影片中取出的每張圖片，出現的車輛數加總。一輛車子可能在不同的時間點出現，重複出現會重複計算。

正樣本成功辨識數：從影片中取出的每張圖片，成功辨識出的車輛數加總。一輛車子可能在不同的時間點被辨識出，將重複計算。

誤判：程式框出不是車輛的東西(如馬路、天空)，沒有被框出來的車輛不算誤判。

誤判數：從影片中取出的每張圖片，誤判的數量加總。

(四)數據計算公式說明

整體辨出率 = 車輛成功辨識數 / 總車輛數 * 100%

整體辨出率的意義即是否成功辨識每輛車，一輛車只計算一次。

平均正確率 = 正樣本成功辨識數 / 總測試正樣本數 * 100%

平均正確率則代表分類器的穩定度，是否在可以辨識出每個角度的車輛，重複出現的車輛重複計算。

誤判率 = 誤判數 / (誤判數 + 正樣本成功辨識數) * 100%

誤判率意謂分類器的可信度，表示辨識出的影像中有多少比例不是車輛。

(二)各分類器訓練結果

方向	分類器編號	正樣本數 (張)	負樣本數 (張)	整體辨出 率	平均正確 率	誤判率
FL	FL01 (*註 1)	168	102	缺 (*註 2)	78.743%	12.741%
	FL02	417	532	缺	77.204%	5.837%
	FL03	628	404	86.614%	84.257%	10.272%
F	F01	127	80	缺	67.189%	6.140%
	F02	397	232	缺	73.792%	6.320%
	F03	611	500	95.238%	78.609%	3.250%
FR	FR01	197	102	82.067%	74.087%	9.924%
	FR02	401	232	94.048%	81.292%	3.326%
	FR03	664	323	95.367%	83.034%	2.429%
R	R01	112	80	缺	46.454%	14.400%
	R02	311	483	92.727%	70.648%	20.798%
BR	BR01	158	107	缺	46.518%	10.215%
	BR02	330	533	89.855%	74.138%	2.642%
B	B01	142	80	缺	24.573%	6.504%
	B02	389	400	88.136%	71.037%	17.687%
BL	BL01	134	100	缺	38.378%	12.346%
	BL02	323	300	94.231%	80.723%	18.886%
L	L01	135	80	缺	33.015%	4.420%
	L02	354	549	68.919% (*註 3)	54.480%	3.185%
FB (*註 4)	FB04	1095	604	93.750%	69.642%	1.947%
T_LR (*註 5)	T_LR02	600	200	82.822%	52.698%	0.000%

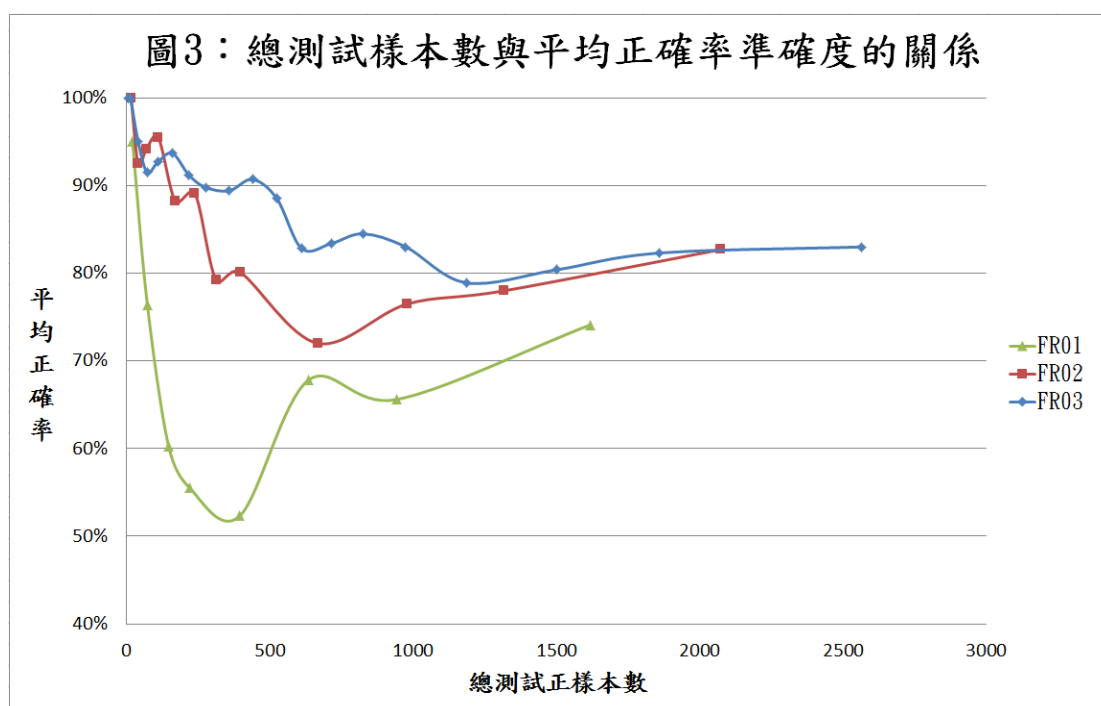
*註 1：分類器編號代表訓練的階段，原則上蒐集到的正樣本數達 100 張時，會進行第一階段訓練，第二階段 300 張，第三階段 600 張，第四階段 1000 張。此階段純歸因於應樣本蒐集速度與分類器訓練速度制定，無特殊意義。

*註 2：研究一開始並無整體辨出率之概念，故較早訓練之分類器資料從缺。

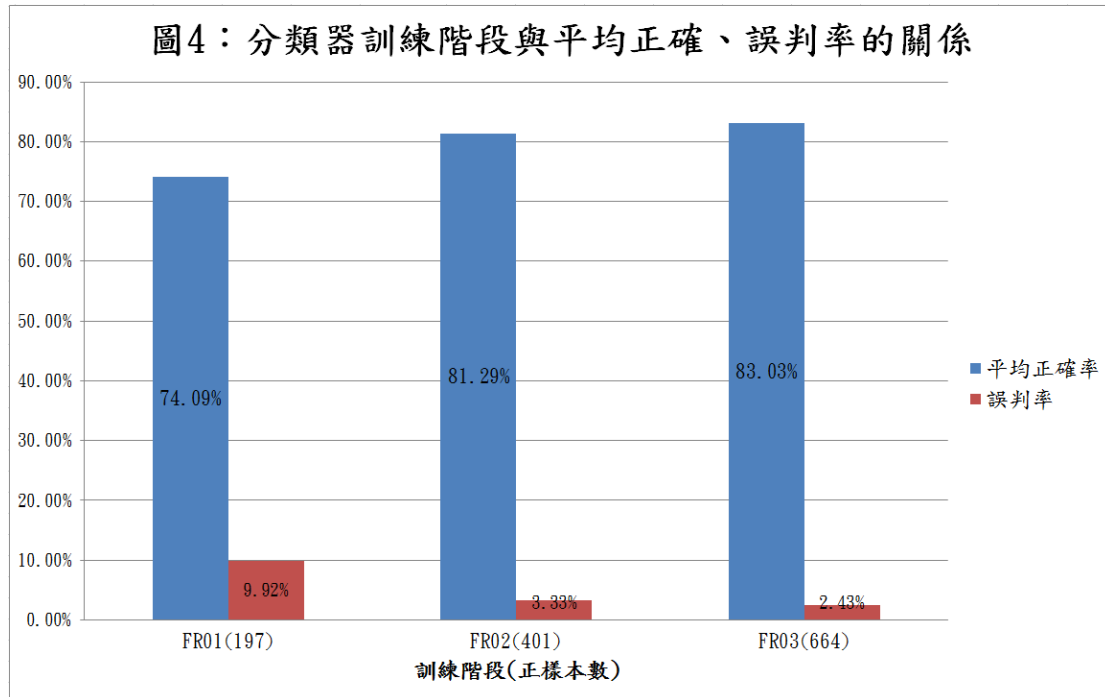
*註 3：L 方向測試結果較 R 方向差，可能歸因於長寬比之設定不當，又或 L 方向樣本多取自路旁 靜止車輛，與行進車輛有異。

*註 4：FB 方向即車輛正面(F)加上背面(B)。

*註 5：T_LR 方向為側面觀看車輪辨識，作為車輛辨識輔助之用。



↑總測試正樣本數增加，所測得的平均正確率趨穩定。



↑ 訓練所用的正樣本數與分類器的測試結果呈正相關。

二、系統測試：模擬燈號詳細數據

(一)無殘影(首次測試)

Video	影片長	Python圖	有人總數	正確判讀	誤判	正確率	誤判率	備註
01	11:40	6	759	541	496	71.316%	47.830%	
02	10:01	6	627	480	317	76.579%	39.774%	
03	9:30	6	992	696	429	70.190%	38.133%	
04	12:00	6	1050	627	477	59.748%	43.207%	
05	3:58	6	274	271	286	98.761%	51.346%	
06	8:10	6	417	384	492	92.086%	56.164%	
07	10:07	6	227	164	299	72.374%	64.579%	
08	10:00	6	851	545	463	64.057%	45.933%	
09	10:00	6	751	474	404	63.149%	46.014%	
10	10:00	6	777	573	444	73.764%	43.658%	
11	6:52	6	307	155	208	50.423%	57.300%	
12	12:33	6	1175	813	748	69.191%	47.918%	
13	10:00	6	738	477	427	64.599%	47.235%	
14	7:31	6	585	396	234	67.669%	37.143%	
Total	2:12:22	6	9530	6596	5724	69.213%	46.461%	

(二)取樣比率(已加入殘影)

Video	影片長	Python 讀圖	有人 總數	正確 判讀	誤判	正確率	誤判率	備 註
01	11:40	30	3793	3046	3072	80.306%	50.212%	
02	10:01	30	3134	2692	2317	85.897%	46.257%	
03	9:30	30	4958	3916	2778	78.983%	41.500%	
04	12:00	30	5247	3523	3084	67.143%	46.678%	
05	3:58	30	1372	1369	2021	99.781%	59.617%	
06	8:10	30	2085	1966	3331	94.293%	62.885%	
07	10:07	30	1133	883	2124	77.935%	70.635%	
08	10:00	30	4254	3166	2878	74.424%	47.617%	
09	10:00	30	3753	2661	2676	70.903%	50.141%	
10	10:00	30	3884	3244	2971	83.522%	47.804%	
11	6:52	30	1537	995	1510	64.736%	60.279%	
12	12:33	30	5875	4507	4627	76.715%	50.657%	
13	10:00	30	3692	2762	2914	74.810%	51.339%	
14	7:31	30	2926	2314	1640	79.084%	41.477%	
Total	2:12:22	30	47643	36994	37943	77.648%	50.633%	

Video	影片長	Python 讀圖	有人 總數	正確 判讀	誤判	正確率	誤判率	備 註
01	11:40	10	1264	1052	1094	83.206%	50.979%	
02	10:01	10	1045	933	833	89.311%	47.169%	
03	9:30	10	1653	1335	928	80.779%	41.008%	
04	12:00	10	1749	1216	1074	69.525%	46.900%	
05	3:58	10	458	457	741	99.927%	61.853%	
06	8:10	10	695	665	1202	95.683%	64.381%	
07	10:07	10	378	308	784	81.553%	71.795%	
08	10:00	10	1418	1106	1040	77.997%	48.462%	
09	10:00	10	1251	928	958	74.181%	50.795%	
10	10:00	10	1295	1092	1101	84.346%	50.205%	
11	6:52	10	512	346	509	67.534%	59.532%	
12	12:33	10	1958	1536	1623	78.434%	51.377%	
13	10:00	10	1231	966	1046	78.494%	51.988%	
14	7:31	10	975	781	555	80.075%	41.542%	
Total	2:12:22	10	1588 2	12721	13488	80.097%	51.463%	

Video	影片長	Python 讀圖	有人 總數	正確 判讀	誤判	正確率	誤判率	備 註
01	11:40	6	759	639	674	84.234%	51.333%	
02	10:01	6	627	560	533	89.343%	48.765%	
03	9:30	6	992	786	636	79.266%	44.726%	
04	12:00	6	1050	753	655	71.755%	46.520%	
05	3:58	6	274	272	445	99.125%	62.064%	
06	8:10	6	417	396	720	94.964%	64.516%	
07	10:07	6	227	176	517	77.670%	74.676%	
08	10:00	6	851	653	659	76.751%	50.229%	
09	10:00	6	751	570	598	75.939%	51.199%	
10	10:00	6	777	646	695	83.162%	51.827%	
11	6:52	6	307	214	313	69.616%	59.393%	
12	12:33	6	1175	947	1025	80.596%	51.978%	
13	10:00	6	738	569	650	77.059%	53.322%	
14	7:31	6	585	478	346	81.681%	41.990%	
Total	2:12:22	6	9530	7659	8466	80.367%	52.502%	

Video	影片長	Python 讀圖	有人 總數	正確 判讀	誤判	正確率	誤判率	備 註
01	11:40	3	379	316	386	83.311%	54.986%	
02	10:01	3	313	280	277	89.343%	49.731%	
03	9:30	3	496	396	366	79.871%	48.031%	
04	12:00	3	525	391	360	74.519%	47.936%	
05	3:58	3	137	134	244	97.668%	64.550%	
06	8:10	3	209	197	390	94.484%	66.440%	
07	10:07	3	113	80	283	70.609%	77.961%	
08	10:00	3	425	330	353	77.574%	51.684%	
09	10:00	3	375	300	312	79.936%	50.980%	
10	10:00	3	388	313	399	80.587%	56.039%	
11	6:52	3	153	107	143	69.616%	57.200%	
12	12:33	3	588	464	595	78.979%	56.185%	
13	10:00	3	369	294	363	79.632%	55.251%	
14	7:31	3	293	239	178	81.681%	42.686%	
Total	2:12:22	3	4709	3245	3986	68.911%	55.124%	

Video	影片長	Python 讀圖	有人總數	正確判讀	誤判	正確率	誤判率	備註
01	11:40	2	253	206	289	81.466%	58.384%	
02	10:01	2	209	183	223	87.588%	54.926%	
03	9:30	2	331	264	243	79.871%	47.929%	
04	12:00	2	350	260	291	74.328%	52.813%	
05	3:58	2	91	90	159	98.397%	63.855%	
06	8:10	2	139	131	298	94.245%	69.464%	
07	10:07	2	76	46	183	60.900%	79.913%	
08	10:00	2	284	211	257	74.401%	54.915%	
09	10:00	2	250	199	245	79.536%	55.180%	
10	10:00	2	259	200	304	77.240%	60.317%	
11	6:52	2	102	76	103	74.170%	57.542%	
12	12:33	2	392	301	402	76.851%	57.183%	
13	10:00	2	246	197	260	80.038%	56.893%	
14	7:31	2	195	150	133	76.897%	46.996%	
Total	2:12:22	2	3177	2514	3390	79.131%	57.419%	

(三)加入 Y 座標判斷

Video	影片長	Python 讀圖	有人總數	正確判讀	誤判	正確率	誤判率	備註
01	11:40	6	759	639	635	84.234%	49.843%	
02	10:01	6	627	556	470	88.705%	45.809%	
03	9:30	6	992	768	566	77.451%	42.429%	
04	12:00	6	1050	749	578	71.374%	43.557%	
05	3:58	6	274	272	210	99.125%	43.568%	
06	8:10	6	417	399	757	95.683%	65.484%	
07	10:07	6	227	164	502	72.374%	75.375%	
08	10:00	6	851	625	535	73.460%	46.121%	
09	10:00	6	751	570	478	75.939%	45.611%	
10	10:00	6	777	636	510	81.874%	44.503%	
11	6:52	6	307	198	172	64.411%	46.486%	
12	12:33	6	1175	893	1022	76.000%	53.368%	
13	10:00	6	738	555	510	75.163%	47.887%	
14	7:31	6	585	439	321	75.017%	42.237%	
Total	2:12:22	6	9530	7463	7266	78.311%	49.331%	

(四)加入行人燈

Video	影片長	Python 讀圖	有人 總數	正確 判讀	誤判	正確率	誤判率	備 註
01	11:40	6	656	544	246	82.977%	31.139%	
02	10:01	6	532	452	129	84.994%	22.203%	
03	9:30	6	943	745	336	79.037%	31.082%	
04	12:00	6	1008	715	204	70.904%	22.198%	
05	3:58	6	223	221	65	98.926%	22.727%	
06	8:10	6	414	392	116	94.686%	22.835%	
07	10:07	6	138	115	79	83.576%	40.722%	
08	10:00	6	775	572	180	73.826%	23.936%	
09	10:00	6	724	516	248	71.310%	32.461%	
10	10:00	6	661	539	252	81.568%	31.858%	
11	6:52	6	280	184	78	65.621%	29.771%	
12	12:33	6	1121	889	230	79.304%	20.554%	
13	10:00	6	725	544	200	74.993%	26.882%	
14	7:31	6	549	458	140	83.394%	23.411%	
Total	2:12:22	6	8749	6886	2503	78.706%	26.659%	

(五)判斷範圍修正

Video	影片長	Python 讀圖	有人 總數	正確 判讀	誤判	正確率	誤判率	備 註
01	11:40	6	656	542	225	82.672%	29.335%	
02	10:01	6	532	397	86	74.652%	17.805%	
03	9:30	6	943	693	164	73.520%	19.137%	
04	12:00	6	1008	673	154	66.739%	18.622%	
05	3:58	6	223	153	25	68.487%	14.045%	
06	8:10	6	414	340	16	82.126%	4.494%	
07	10:07	6	138	104	74	75.581%	41.573%	
08	10:00	6	775	564	180	72.793%	24.194%	
09	10:00	6	724	506	224	69.928%	30.685%	
10	10:00	6	661	539	242	81.568%	30.986%	
11	6:52	6	280	184	78	65.621%	29.771%	
12	12:33	6	1121	757	142	67.529%	15.795%	
13	10:00	6	725	419	92	57.761%	18.004%	
14	7:31	6	549	431	137	78.478%	24.120%	
Total	2:12:22	6	8749	6302	1839	72.031%	22.589%	

二、系統測試：違規車輛檢測詳細數據

影片	判斷出圖片張數	判斷出車輛數	人眼校正車輛數	錯誤 ※ = 誤判 1 輛	正確率	錯誤率
01	18	9	12		75.000%	
02	2	2	1	※	100.000%	
03	17	7	13		53.846%	
04	6	3	9		33.333%	
05	0	0	0		X	
06	0	0	0		X	
07	0	0	1		0.000%	
08	18	8	14		57.143%	
09	15	5	16		31.250%	
10	13	8	9	※	77.778%	
11	12	5	4	※※	75.000%	
12	27	12	19	※※	52.632%	
13	2	1	5		20.000%	
14	15	9	12	※※※※	41.667%	
Total	145	69	115	10	51.304%	14.493%