# 中華民國第 61 屆中小學科學展覽會

# 作品說明書

---

國中組　生活與應用科學(一)科

## 佳作

## 團隊合作獎

032808

## 智慧影像，「One」視平安~大型車轉彎安全偵測警示系統


學校名稱：高雄市立明華國民中學

| 作者： | 指導老師： |
| --- | --- |
| 國二 顏妤晅 | 陳晏閔 |
| 國二 陳以華 | 王天佑 |
| 國二 曹家瑜 | |



關鍵詞：內輪差、OpenCV、Jetson Nano

# 摘要

本研究主要應用 AI 影像辨識將大型車輛轉彎內輪差及視覺死角危險區域內的行人及車輛定位，立即以語音對駕駛警告危險位置，避免車禍發生。本系統先於 Windows 10 PC 上的 Anaconda3　Spyder 環境以 Python3 使用 OpenCV 函式庫開發，並叫用 YOLOv3 來進行物體的辨識，能正確辨識行人和車輛並正確定位，配合大型車內輪差的計算判定危險性。程式最後佈署於 NVIDIA Jason Nano Ubuntu 系統下並安裝於大型車輛。不同於一般行車視野輔助系統僅以測距的感測器偵測，本系統能明確得知人車種類，明確標定位置，並能立即判定有無危險，更立即以語音警告駕駛，且不干擾駕駛的視覺注意力，同時也擴音警告車外的用路人。研發過程中還設立了車輛影像辨識研究平台，期望後續研究能和自駕車的發展結合造福更多的駕駛及用路人。

## 壹、研究動機

曾經在電視上看到一個很驚人的新聞：一位婦人在過馬路時，因為駕駛聯結車的司機轉彎時沒有看到內輪差視覺死角區域內有人，便意外地把她撞飛了。根據交通部公路總局統計，近三年來，像這樣的案件數有 537 件，撞死 563 人，受傷 260 人，為數不少。當我還在感嘆意外的發生時，突然有人問起了新聞標題中的「內輪差」到底是什麼，卻無人知曉。所以我們想了解內輪差對用路人的危害，以及研究如何避免這樣的案件再次發生。一開始，我們想用一些感測裝置如毫米波雷達來偵測是否有人車進入內輪差及視野死角的範圍，並提醒駕駛，但是查過資料後發現這樣的設備所費不貲，反而近幾年的影像辨識技術越臻完美，成本也愈來愈便宜，於是我們決定要研究這樣子可以救人的影像辨識及警示系統。

## 貳、研究目的

一、設計建置車輛型影像辨識實驗平台

二、探討內輪差視覺死角範圍之計算

三、運用影像辨識偵測內輪差及視覺死角範圍內的物體同時對駕駛及用路人語音警示

## 【背景資訊】

### 一、內輪差與視野死角

**駕駛眼睛本身所看不到的範圍，以及因為車輛構造上或其他道路上的障礙物擋住駕駛人視線的範圍,就是所謂的「視野死角」;在照後鏡的輔助下,仍然看不到的範圍也是屬於「視野死角」。而「內輪差」就是車輛在轉彎時,前後車輪的行經軌跡所造成的差距。這是指當車輛轉彎時,內側的後輪也會向內偏移,其偏移的軌跡與前輪軌跡間的距離叫做「內輪差」**。內輪差是隨車子的前、後輪軸距而變化：車子越長，那麼內輪差之危險區域也愈大。內輪差的大小也隨轉向角度而變化：當看到車子轉彎的角度愈大時，所形成的內輪差之危險區域也愈大，應保持大於車身的三分之一遠之安全距離。例如：一輛長約 12 公尺的公車轉彎時，請保持 12×1/3 =4 公尺以上的安全距離。大型車輛轉彎容易有死角釀成車禍，為提升大型車輛行車安全，順應車輛安全技術演進，交通部於 104 年發布訂定行車視野輔助系統檢測基準，要求車輛應配備可供駕駛人觀察車輛周遭環境狀況的視野輔助工具，明訂大貨車和大客車都要依法必須裝上左右轉和倒車警示器，自 106 年起分階段實施，107 年 1 月 1 日起所有新出廠大型車輛均應設置行車視野輔助系統，並規定自 109 年 1 月 1 日起列為大型車定期檢驗項目。傳統的警示器「嗶嗶聲」不夠明顯，有廠商改用「語音自動警示器」，108 年起全台至少 2000 輛貨車安裝，會播出「車輛轉彎請注意！」、或「車輛倒退請注意！」國語和台語的語音警示，希望用路人都能更注意到車輛要轉彎的警示來減少車禍。然而，有特定的情境是用路人聽到了也注意到了卻受限於行動能力而無法及時逃避。**而此時駕駛「渾然不知」有用路人在行車的路徑上或危險區域內。**

交通部公路總局於 109 年 5 月至 6 月調查大型車輛駕駛人使用行車視野輔助系統情形，，經統計 108 年至 109 年 1 月至 5 月同期，大型車輛左、右轉彎造成機車、自行車、行人之事故死亡人數，有降低趨勢。交通部同時指出，行車視野輔助系統有多種設計樣式，各有其不同特性，其功能係為輔助駕駛人消除行車時之視野死角，倘駕駛因個人內在因素(如：精神不集中)或環境外在因素(如：天候致視線不佳、噪音致聽不清楚)而無法妥善運用安全輔助設備，仍會造成交通事故，因為有些是行動不便的人、或反應遲鈍、走不快的老弱婦孺，駕駛不能期待行人或騎乘機車、腳踏車的用路人聽到警示的聲音就能自行即時逃避危險。

## 二、OpenCV

**OpenCV 的全稱是 Open Source Computer Vision Library，是一個跨平台的電腦視覺庫。**OpenCV 是由英特爾公司發起並參與開發，以 BSD 授權條款授權發行(開源的)，可以在商業和研究領域中免費使用。OpenCV 主要用於開發即時的圖像處理、電腦視覺以及圖型識別程式。

OpenCV 專案最早由英特爾公司於 1999 年啟動，致力於 CPU 密集型的任務，是一個包括如光線追蹤和 3D 顯示的計劃的一部分。早期 OpenCV 的主要目標是：為推進機器視覺的研究提供一套開源且最佳化的基礎庫。提供一個共同的基礎庫，使得開發人員的代碼更容易閱讀和轉讓，促進知識的傳播。透過提供不需要開源或免費的軟體授權，促進商業應用軟體的開發。現在也整合了對 CUDA 的支援。**如今 OpenCV 可應用於解決以下各分項的問題：物體辨識、影像分割、人臉辨識、運動跟蹤、運動分析、動作辨識、手勢辨識、擴增實境、紋理分析、人機互動、機器人、汽車安全駕駛等。**主要功能如下：

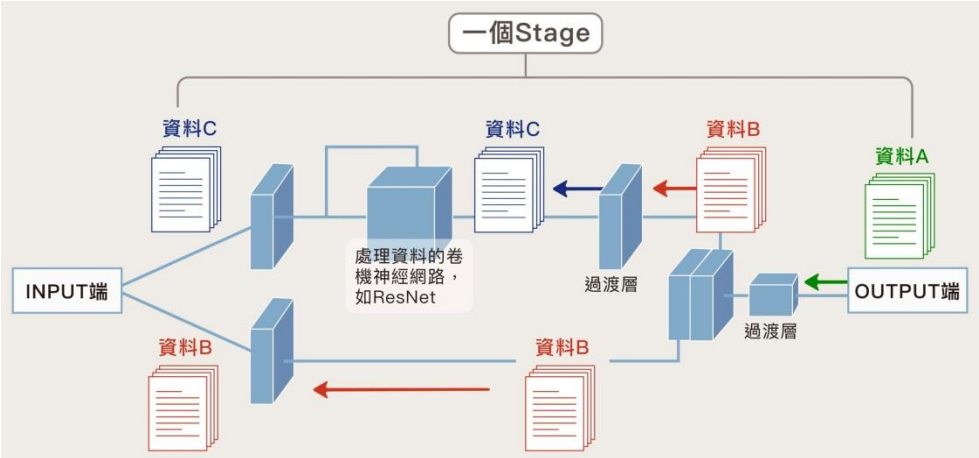
- 影像資料的操作（ 分配、釋放、複製、設定和轉換)。
- 基本的數位影像處理（濾波、邊緣及角點檢測、取樣與差值、色彩轉換形態操作等）。
- 矩陣和向量的操作以及線性代數的演算法（矩陣積、解方程、特徵值以及奇異值等）。
- 視訊的輸入輸出 I/O （檔案與攝像頭的輸入、影像和視訊檔案輸出）。
- 攝像頭定標（發現與跟蹤定標模式、定標、基本矩陣、齊次矩陣估計、立體對應）。
- 各種動態資料結構（列表、佇列、集合、樹、圖等）。
- 結構分析（連線部件、輪廓處理、距離變換、各自距計算、模板匹配、Hough 變換、多邊形逼近、直線擬合、橢圓擬合、Delaunay 三角劃分等）。
- 運動分析（光流、運動分割、跟蹤）。
- 物體辨識（特徵法、隱馬爾可夫模型：HMM）。
- 基本的 GUI （影像與視訊顯示、鍵盤和滑鼠事件處理、滾動條）

OpenCV 可以在 Windows, Linux, Android,OpenBSD, iOS 和 Mac OS 等平台上執行。使用者可以免費在 Github 獲得官方版本，或者從 Git 獲得開發版本的原始碼，用 Cmake 在各平台上進行編譯。在 2012 年，OpenCV 以一個非營利組織來營運，官網為 https://opencv.org。

## 三、YOLO

**「只要讓我看一眼，就知道這是什麼！(You Only Look Once， YOLO）」YOLO，是目前當紅的 AI 物件偵測演算法。中研院資訊科學研究所所長廖弘源及博士後研究員王建堯，與 YOLO 的技術維護者俄羅斯學者博科夫斯基（Alexey Bochkovskiy）共同研發最新的 YOLO v4。**王建堯指出在馬路上運行中的車速度很快，物件辨識必須非常即時，在短時間內就能辨識出車輛，並能持續追蹤，計算車速。這個技術對物件的偵測必須「快、狠、準」。此外，如果把不斷產生資料都上傳雲端運算，不但耗時也會給雲端電腦帶來太大的負擔，因此這個偵測技術的計算量必須夠小，小到可裝在十字路口監視器上的小型計算器，即可完成物件偵測的任務。

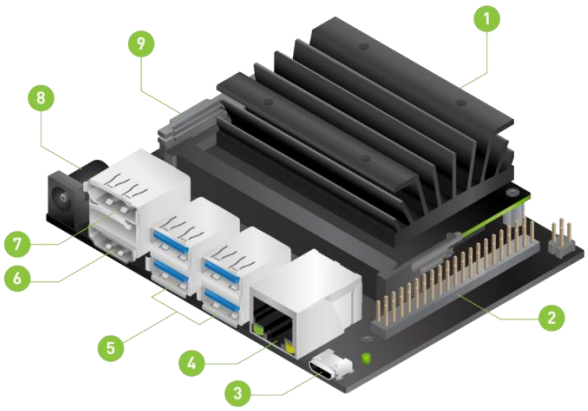
起初，中研院團隊首先研發出局部殘差網路 PRNet（partial residual networks, PRNet），將資訊「分流」，減少無謂的計算量，使運算速度增加兩倍，精確度卻未提升。緊接著研發出跨階段局部網路 CSPNet（cross stage partial network, CSPNet），利用分割－分流－合併的路徑，成功達到了大幅減少計算量、卻能增加學習多元性的目標，因此也得到更好的精確度。發表 CSPNet 之後，吸引博科夫斯基（Alexey Bochkovskiy）的注意，並很快展開合作，將 CSPNet 應用於開發新一代的 YOLO，並於 2020 年 4 月發表了 YOLOv4 ，成為最快、最準的物件偵測技術，引爆全球的 AI 社群。由於 YOLO 的技術是開放的，各式各樣的應用也如雨後春筍般快速出現，舉例來說，YOLOv4 可即時偵測人們的社交距離，或是快速判斷行人有沒有戴口罩。其中的關鍵技術正是 CSPNet，示意圖如下：




資料來源：圖-研之有物｜內容-研究員王建堯

## 四、Jetson Nano


Jetson Nano Developer Kit 是一款體積小巧當功能強大的 AI 開發套件，搭載四核 Cortex-A57 處理器，128 核 Maxwell GPU 以及 4GB LPDDR 內存。支持 NVIDIA JetPack。可以並行運行多個神經網絡對圖像分類，目標檢測，分割等應用。

| | |
|---|---|
|  | 1. SD 卡插槽<br>2. 40pin GPIO<br>3. Micro-USB 5V 電源輸入<br>4. Gigabit 乙太網路埠<br>5. USB 3.0 連接埠<br>6. HDMI 輸出連接埠<br>7. DisplayPort 連接器<br>8. 5V 直流電源輸入<br>9. MIPI CSI 攝影機連接孔 |

# 參、研究設備及器材

一、 硬體設備及軟體：

| | |
|---|---|
| (一) PC with Windows 10 | (六) NVIDIA Jetson Nano with Ubuntu |
| (二) Anaconda Spyter 開發環境 | (七) Jupytor Notebook 開發環境 |
| (三) OpenCV with Python | (八) Chrome 瀏覽器 |
| (四) YOLO | (九) Jetracer AI Kit |
| (五) Visual Studio 開發環境 | (十) Word |

二、 各式手工具

# 肆、研究流程及結果

## 一、研究平台軟硬體建置：

首先，我們組裝了 Jetracer AI Kit 成為可操控的車子本體，接著加以改裝外型。

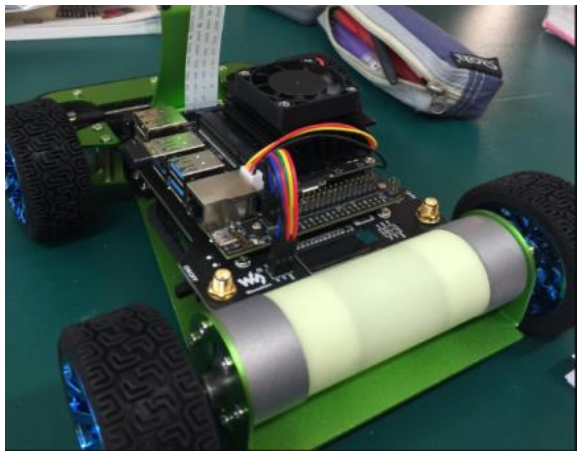
(一) Jetracer 小車組裝：

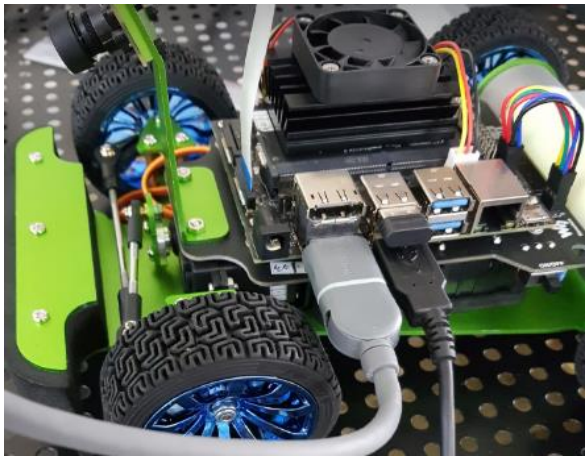
1. 將馬達固定在金屬底盤上，注意這裡用的是 M3*6 螺絲，如果用長螺絲會導致馬達無法正常轉動。
2. 將聯軸器裝入輪胎中，注意這裡需要使用錘子將其擊打進入輪胎， 並用 M4*8 螺絲在另一側固定。
3. 將車輪裝到馬達上，並用黑色螺絲固定，這裡需要用到小扳手，使用長端操作。
4. 將舵機固定支架安裝到金屬底盤上。
5. 將舵機固定放置在支架上，並用螺絲螺母固定，注意舵機方向。
6. 在組裝舵機拉桿的時候，請注意參照這裡示意圖的組裝方向。用兩個 M3 球頭(小)和短的拉桿組成舵盤拉桿，注意這裡兩個球頭最終呈現的效果是相互垂直的。將舵盤用自帶的螺絲固定到舵盤板上，然後用 M2.5*12 螺絲和 M2.5 螺母將拉桿平頭一端鎖在舵盤板上。注意舵盤的卡槽是要朝向外側的。
7. 組裝前輪拉桿，用兩個 M3 球頭(大)和長的拉桿組成前輪拉桿。將大小軸承分別裝入轉向杯的軸承槽中。8. 分別用 M2.5*16 和 M2.5 螺母將舵機拉桿， 前輪拉桿和轉向杯固定在一起，注意舵機拉桿在上面，前輪拉桿在中間，轉向杯的軸承要朝向內側。請注意各個配件的安裝方向。
9. 將車輪用 M4 螺絲和螺母鎖在轉向杯上。注意這裡不要鎖得太鬆，避免小車前輪搖晃，也不能鎖太緊，導致前輪無法正常轉動，鎖完，之後請撥動車輪測試一下是否可以順暢的轉動。
10. 將 M3*22 銅柱和 M3 螺絲固定在前輪位置，以備後面安裝前輪。
11. 將前面組裝好的前輪組合裝入底盤，首先將舵盤對準舵機裝入，然後用 M3*6 螺絲固定。兩邊車輪用 M2*30 螺絲和 M2 自鎖螺母配合三角板固定住。
12. 用 M3*26 銅柱鎖在 JetRacer 擴展板並裝 EVA 墊，用 M3.20 銅柱鎖在防撞條並裝 EVA 墊。

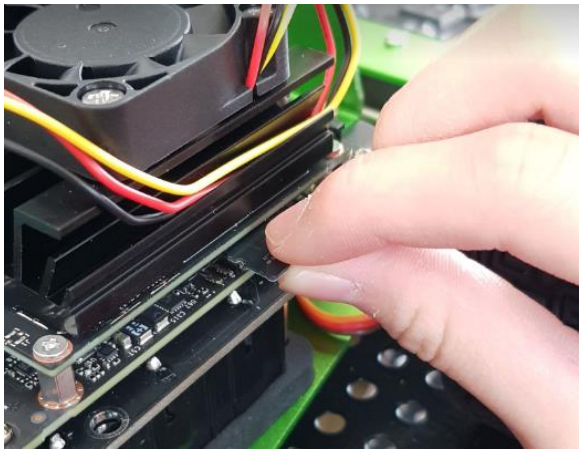
13. 將鏡頭支架和天線固定在 JetRacer 擴展板上。注意安裝方向。
14. 將電池根據正確方向裝入 JetRacer 擴展板，並將舵機和馬達的排線接入擴展板上，馬達左邊對左邊，右邊對右邊。接入舵機線的時候注意對好線序。褐色線對 GND。
15. 調整舵機和馬達的排線，將擴展板用 M3 螺絲固定在底盤上。同時用 M3 螺絲將金屬防撞條固定
16. 用 M2.5 螺絲將 Jetson Nano 固定在擴展板上
17. 拆開 Jetson Nano 核心板，將無線網卡接入到 Jetson Nano，並將天線接好
18. 裝好 Jetson nano, 將鏡頭用尼龍螺絲裝入到鏡頭支架上，注意鏡頭支架和鏡頭之間需要加墊一塊壓克力板避免短路。用 6PIN 排線連接擴展板和 Jetson nano，連接時要根據板子上記號連接 5V 對 5V, 3.3V 對 3.3V
19. 將散熱風扇用螺絲裝入 Jetson nano 注意標籤面朝下。風扇排線插入風扇接口，左對齊對好卡槽插入。

零件開箱：

組裝成果：







作業系統為 Ubuntu ，映像檔可由 Nvidia 官網下載，刷入 SD 記憶卡中並裝於 MicroSD 插槽，接上螢幕鍵盤便可開機使用。另外有內建 Wifi 網路 及 Jupyter Server 開發環境 可遠端網頁登入使用

**(二)外型改裝：**

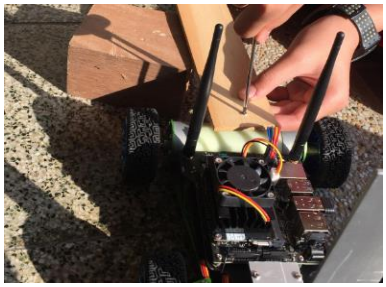
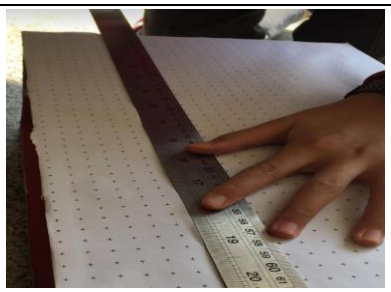
為了研究內輪差對行車安全的影響，我們將車子設計成兩種形式：分為大客車與聯結車。我們改變了 Jetracer AI Kit 後輪位置依照大貨車的比例往後推移，並且縮短後車軸的長度使得前後車輪軸等長，裝上木板延伸固定使其能與車子本體剛性結合。我們以原本馬達在車體上固定的方式，並在木板上改以平頭螺絲與馬達固定，以達最穩固的狀態。

大客車車殼為一體成形，聯結車則由前面的車頭及後面的貨櫃所組成，均以紙箱回收的厚紙板加工製作。長方形紙箱貨櫃依實際比例裁剪，下面加裝長條木板使他能與車頭本體做連結並且還要可以旋轉。又於木板下方裝設輪子使用樂高零件可與木板連接又能達成可動的狀態。使用鐵片將鏡頭固定在後貨櫃的後方，並使細木板延伸讓寬度大於車廂才能拍攝。聯結車車頭則是將紙板設計成接近正方體，並將車頭前面設計有傾斜度以模擬實際車頭造型。輪子處割出圓弧造形，讓前輪能順利轉彎，車頭與後貨櫃的左右兩邊輪子都做同樣造型。

大客車的車殼，將紙板裁剪出符合車子本體比例的車殼，並配合大客車裝設於車廂前方一般後視鏡處的鏡頭，將車殼前面割出 Y 字形，讓鏡頭的連接線能延伸出來。聯結車是設在後面貨櫃上，而且是在貨櫃的後方由後往前拍攝。為了讓大客車鏡頭的 160 度的視角能充分地涵蓋想攝影的範圍，我們把鏡頭撐高，以鐵片固定連接木板並鎖上螺絲並設計有往下的傾斜角度。在鏡頭的位置加上一小塊磨成一定斜度的木板讓鏡頭拍攝的位置往外偏轉，並以額外訂購的加長版(100cm)軟排線連接到控制板。

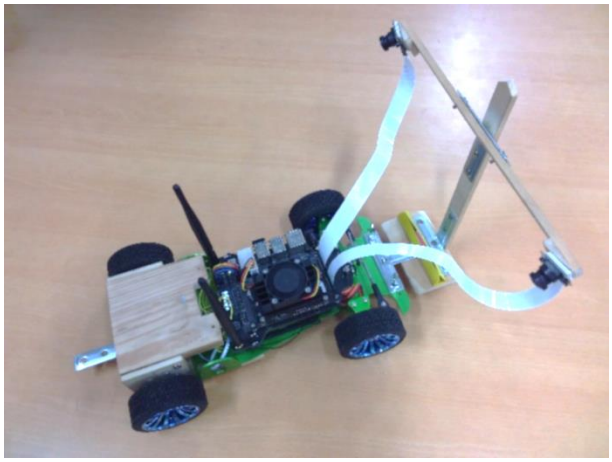
組裝過程：



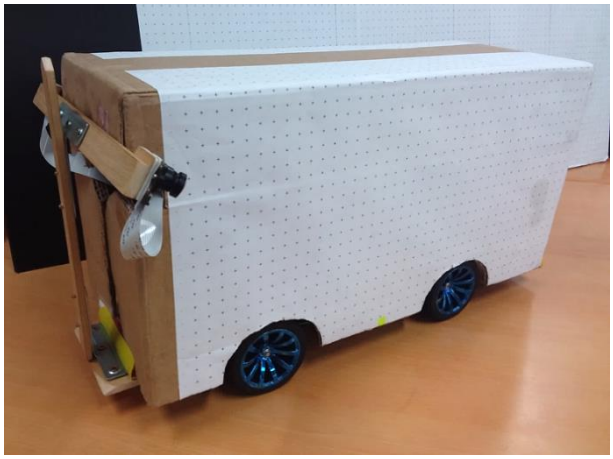



完成圖：

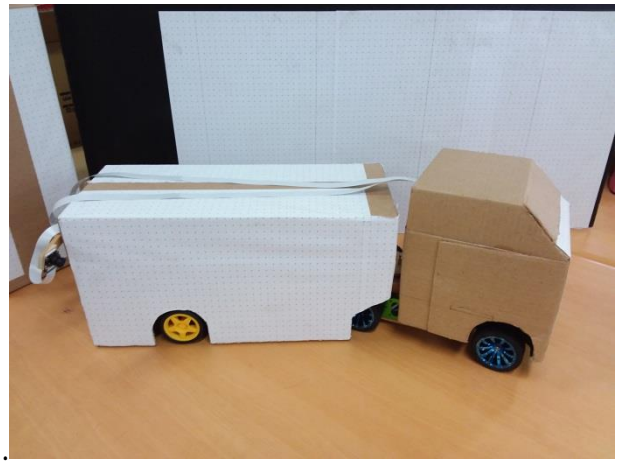
小車車體已安裝後視攝影機



1:12 大客車(乙種)：裝配車殼



1:12 聯結車：裝配車殼



# 二、各項實驗

## 實驗一、使用無線遊戲搖桿經由控制項進行車輛遠端遙控

我們需要建立一個搖桿控制項小部件的實體，並將用它來控制馬達驅動車子。

搖桿控制項小部件用 index 參數作為搖桿對應的值。如果連接了多個搖桿，或者某些遊戲搖桿有多個控制鈕，就會非常有用。要確認控制器的索引值，我們需要做：

1.連結 http://html5gamepad.com

2.按下正在使用的遊戲搖桿上的按鈕

3.請記住按下搖桿按鈕時所回應的 index 值

於是，我們可以使用該索引建立搖桿控制項並在 Jubyter 筆記頁上顯示出來：

```
import ipywidgets.widgets as widgets
controller = widgets.Controller(index=0)  # 將 0 替換成遊戲搖桿所對應的 index 值
display(controller)
```

即使 Controller()中的 index 正確，還是可能看到它顯示出 連接搖桿，然後按任意按鈕 這段文字。這是因為遊戲搖桿在程式中尚未註冊；任意按個按鈕，應該會看到有遊戲搖桿的小部件出現，這也表示我們已經由程式連接了遊戲搖桿。接著我們需要將控制項連接到馬達以操控車子！

```
from jetracer.nvidia_racecar import NvidiaRacecar
import traitlets
car = NvidiaRacecar()
car.throttle_gain = 1
```

NvidiaRacecar 實作了 Racecar 類別，使車子具有 "轉向 steering" 和 "油門 throttle"兩個屬性。我們可以為這兩個屬性設定[-1，1]範圍內的值。執行以下指令，將"航向 steering "設為 0.4，將使車前輪轉向，若車子沒有反應，則它可能還處於手動模式，請切換搖桿上的手動操控開關。

```
car.steering_gain = -0.4
car.steering_offset = 0
car.steering = 0.4
```

NvidiaRacecar 類別 有 "轉向增益值 steering_gain" 和 "轉向修正值 steering_offset" 兩個參數 可以用來校正方向、我們可以藉由執行下列指令，來找出 steering 的預設值。

```
print("steering_gain   =" , car.steering_gain)
print("steering_offset =", car.steering_offset)
```

最後的航向值是用以下方程式計算 $y=ax+b$ 其中,

$a$是轉向增益值 car.steering_gain、$x$是轉向操控值 car.steering 在[-1，1]的範圍內、
$b$是轉向修正值 car.steering_offset、
$y$是 計算所得的航向值，再傳給伺服馬達做方向控制。
我們可以調整這些值來 校正航向。使得操控值為 0 時向前直走。+1 時向右，-1 時向左。
將車子"油門 throttle"設為 0.5，我們可以執行以下的指令
注意：請挪出一些空間讓小車移動，JetRacer 可以跑很快!

```
car.throttle_gain = 0.5
car.throttle = 0.5
```

最後的油門值是用以下方程式計算 $y = ax$ 其中，
$a$是油門增益值 car.throttle_gain、$x$是油門操控值 car.throttle 在[-1，1]的範圍內、
$y$是計算所得的油門值，再傳給左右輪馬達做速度控制。
我們可以藉由執行下列指令，來找出 throttle 預設值。

```
print("throttle_gain   ="  , car.throttle_gain)
print("throttle =", car.throttle)
```

接著我們要加上的第一個控件--馬達控件。我們使用 dlink 函數將 "轉向 steering" 和 "油門 throttle" 兩個計算值 分別連接到搖桿左十字、右十字兩個控制軸。由於控制軸方向和我們對馬達控制直觀的方向相反，我們將配合 dlink 函數用一個 lambda 函數來反轉正負極性。dlink 函數比 link 函數更有彈性，它允許我們在來源和目標之間附加額外的轉換函數。

溫馨提醒：執行下一個單元格時，如果觸動遊戲搖桿控制軸將使車子移動！

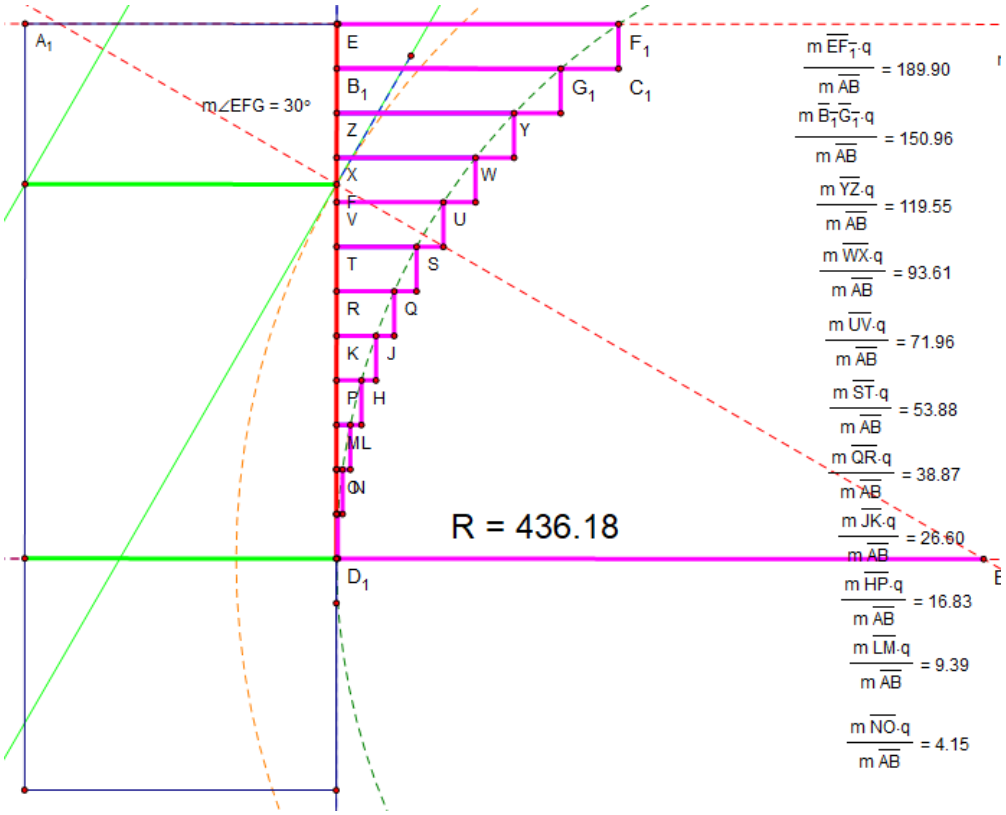
```
left_link  = traitlets.dlink((controller.axes[0], 'value'), (car, 'steering'),
transform=lambda x: -x)
right_link = traitlets.dlink((controller.axes[5], 'value'), (car, 'throttle'),
transform=lambda x: -x)
```

至此，我們可以遙控車子以進行後續的實測。

## 實驗二、遠端遙控車輛實測迴轉半徑及內輪差

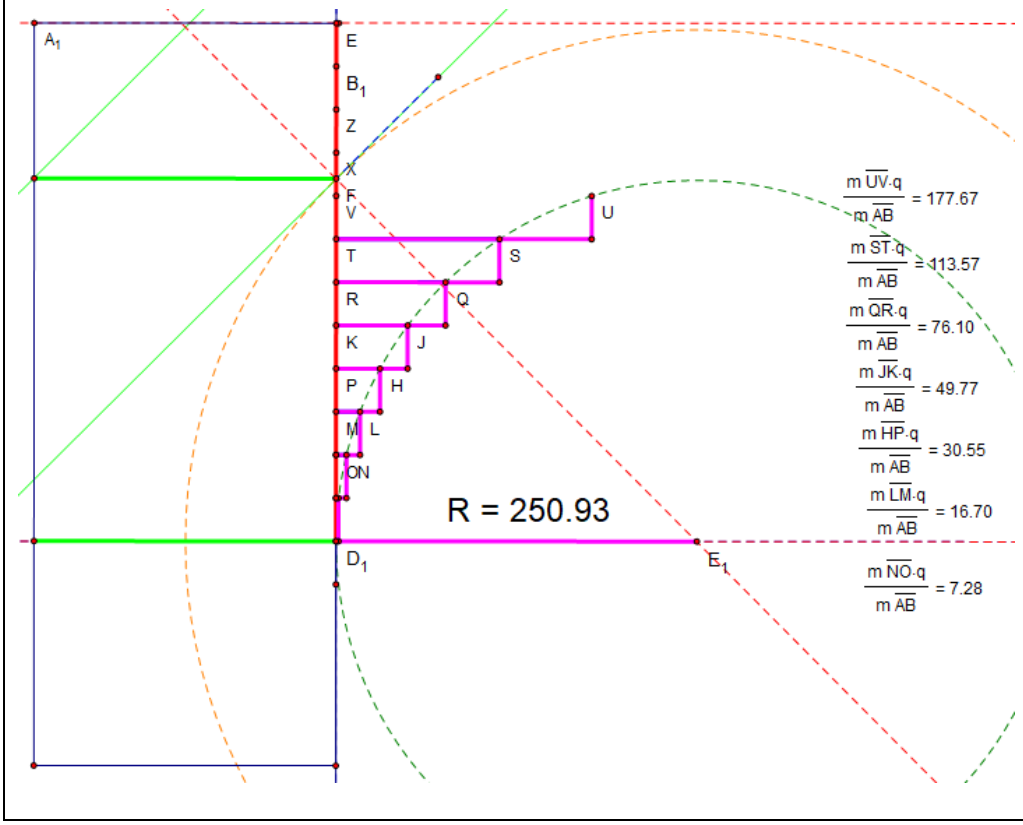
市公車多種的車型中，乙種大客車總車長 516cm，軸距 252cm，轉向角 30 度時，經由理論公式計算由車頭縱向距離，對應到的安全距離如次頁圖表所示。我們可以根據此數值來判定，車輛及人員在哪一個位置是不安全的，其迴轉半徑為 436.40cm。另一個圖表則是轉向角 45 度時的情況，其迴轉半徑為 250.93cm。縱向距離 120 公分以內均落在車輛的行經路跡上，是非常危險的。

總車長 516cm，軸距 252cm，轉向角 30 度，迴轉半徑 436.18cm




| 縱向 | 安全距離 |
|---|---|
| 0 | 189.90 |
| 30 | 150.96 |
| 60 | 119.55 |
| 90 | 93.61 |
| 120 | 71.96 |
| 150 | 53.88 |
| 180 | 38.87 |
| 210 | 26.60 |
| 240 | 16.83 |
| 270 | 9.39 |
| 300 | 4.15 |

總車長 516cm，軸距 252cm，轉向角 45 度，迴轉半徑 250.93cm




| 縱向 | 安全距離 |
|---|---|
| 0 | 無限 |
| 30 | 無限 |
| 60 | 無限 |
| 90 | 無限 |
| 120 | 117.67 |
| 150 | 113.57 |
| 180 | 76.10 |
| 210 | 49.77 |
| 240 | 30.55 |
| 270 | 16.70 |
| 300 | 7.28 |

## 實驗三、辨識並標定危險熱區 4 角，將梯形透視圖影像變換成矩形俯視圖

**程式主要進行兩項影像辨識處裡分別如下：**

**一、透過形狀辨識，標定危險熱區 4 角**

是將影像畫面中於熱區 4 角頂點分別擺放橘色長方形紙板，熱區規劃為前至車前輪、後至車尾、寬 180 公分，透過 OpenCV 數道程序處理自動辨識出 4 個長方形紙板形狀及位置。

程式中先將彩色圖片轉為灰階，再將灰階圖模糊化用以簡化邊框線條訊息。接著用 Canny 這個複合型的邊緣偵測演算法，把濾雜訊的 Gassian 跟邊緣偵測的 Sobel 兩個截然不同的濾波器整合，直接一個函式找出邊緣，接著判定邊數找出指定的長方形。

我們以每個橘色長方形紙板右下角作為目標點，便取得熱區 4 角頂點於所拍攝的透視圖中的畫素座標，接著畫出方框將其 4 角範圍明確地標示出來。4 點座標存檔作為後續影像辨識的邊界依據。

**二、影像透視變換（Perspective Transformation）**

影像透視變換是將影像投影到一個新的視平面，OpenCV 通過以下函式構造矩陣 matrixT，其中 pts1 和 pts2 分別表示變換前後的四個點對應的位置。

```
cv2.getPerspectiveTransform(pts1, pts2)
```

引數意義：pts1 表示透視變換前的 4 個點對應位置， pts2 表示變換後的 4 個點對應位置

得到 matrixT 後，再通過以下函式進行透視變換。

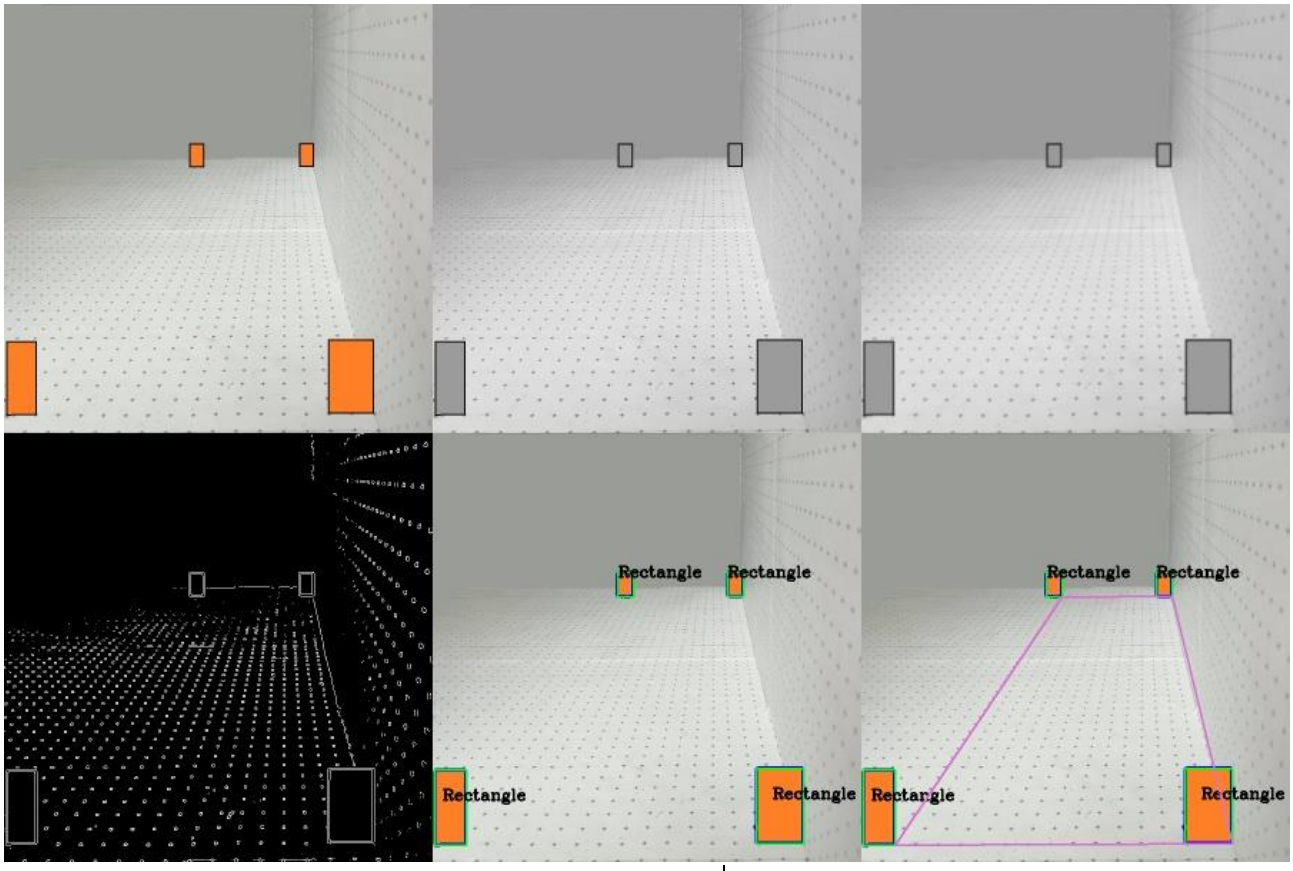
```
cv2.warpPerspective(source,T, (h_width,h_height))
```

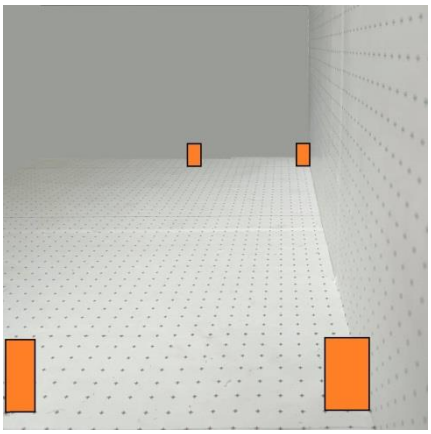
引數意義： source 表示原始影像； matrixT 表示透視變換矩陣； (h_width,h_height)表示變換後的影像大小； h_width 表示寬度(pixls)，h_height 表示高度(pixls)

另外，程式一開始便要求輸入並建立車輛的尺寸參數，作為後續真實距離的換算依據。

程式輸出：

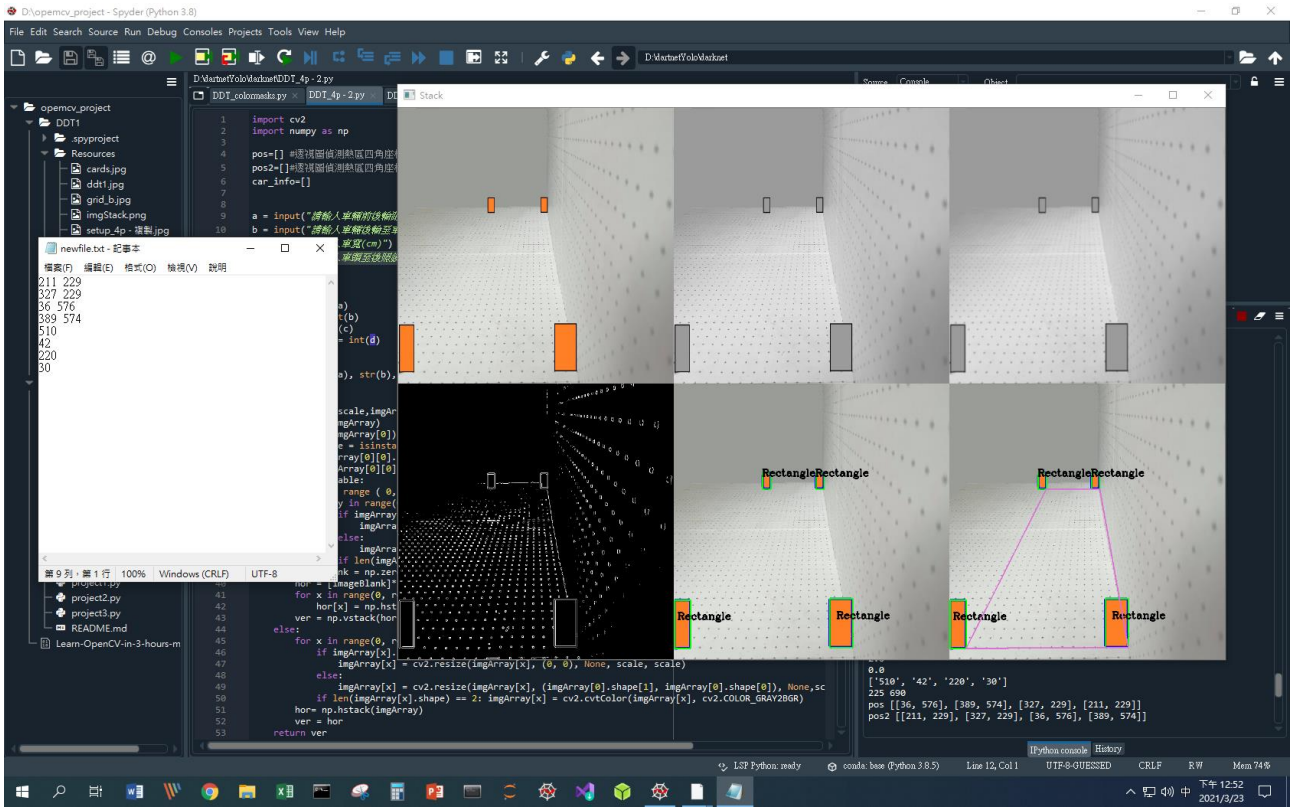
```
[80, 100, 520, 220]       #車輛尺寸
225 775                   #長方形熱區依比例小尺寸
4                          #辨識出 4 個長方形區域（面積太小的全部排除）
4
5
5
pos1 [[36, 576], [389, 574], [327, 229], [211, 229]]  #透視變換前的 4 個點對應位置
pos2 [[211, 229], [327, 229], [36, 576], [389, 574]]  #透視變換後的 4 個點對應位置
```

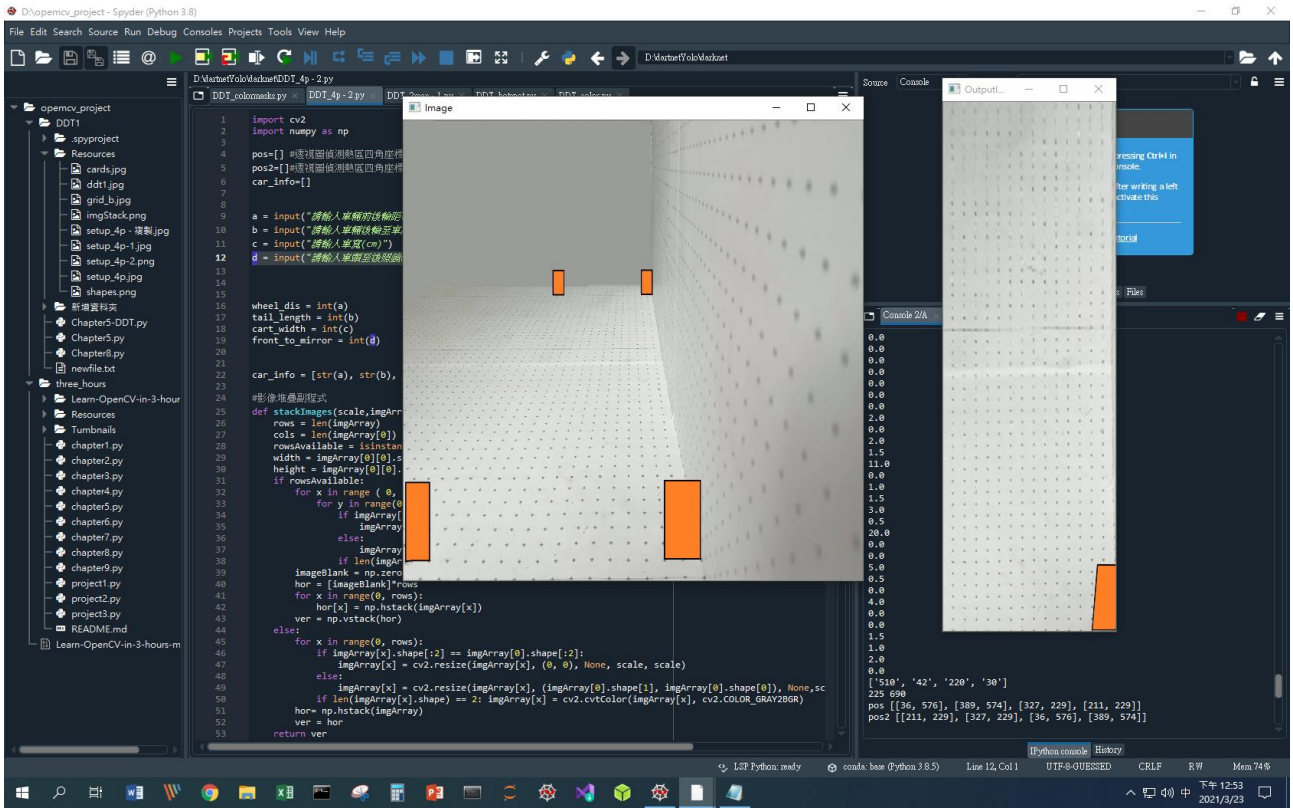


轉換前之透視圖

轉換後之俯視圖




程式碼：

```
import cv2
import numpy as np


#影像堆疊副程式
def stackImages(scale,imgArray):
    rows = len(imgArray)
    cols = len(imgArray[0])
```

```
    rowsAvailable = isinstance(imgArray[0], list)
    width = imgArray[0][0].shape[1]
    height = imgArray[0][0].shape[0]
    if rowsAvailable:
        for x in range ( 0, rows):
            for y in range(0, cols):
                if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale,
scale)
                else:
                    imgArray[x][y] = cv2.resize(imgArray[x][y],
(imgArray[0][0].shape[1], imgArray[0][0].shape[0]), None, scale, scale)
                if len(imgArray[x][y].shape) == 2: imgArray[x][y]=
cv2.cvtColor( imgArray[x][y], cv2.COLOR_GRAY2BGR)
        imageBlank = np.zeros((height, width, 3), np.uint8)
        hor = [imageBlank]*rows
        for x in range(0, rows):
            hor[x] = np.hstack(imgArray[x])
        ver = np.vstack(hor)
    else:
        for x in range(0, rows):
            if imgArray[x].shape[:2] == imgArray[0].shape[:2]:
                imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)
            else:
                imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1],
imgArray[0].shape[0]), None,scale, scale)
            if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x],
cv2.COLOR_GRAY2BGR)
        hor= np.hstack(imgArray)
        ver = hor
    return ver



#輪廓偵測副程式
def getContours(img):
    contours,hierarchy =
cv2.findContours(img,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
    for cnt in contours:
        area = cv2.contourArea(cnt)
        # - print(area)
#-----------------------------------------------------------------
        if area>500:
            cv2.drawContours(imgContour, cnt, -1, (255, 0, 0), 3)
            peri = cv2.arcLength(cnt,True)
            approx = cv2.approxPolyDP(cnt,0.02*peri,True)
            print(len(approx))
#-----------------------------------------------------------------
            objCor = len(approx)
            x, y, w, h = cv2.boundingRect(approx)
            if objCor ==3: objectType ="Tri"
            elif objCor == 4:
                aspRatio = w/float(h)
                if aspRatio >0.95 and aspRatio <1.05: objectType= "Square"
                else:objectType="Rectangle"
            elif objCor>4: objectType= "Rectangle "
            else:objectType="None"
            cv2.rectangle(imgContour,(x,y),(x+w,y+h),(0,255,0),2)
            cv2.putText(imgContour,objectType,
                        (x+(w//2)-10,y+(h//2)-10),cv2.FONT_HERSHEY_COMPLEX,0.7,
                        (0,0,0),2)
            pos.append([x+w,y+h])
```

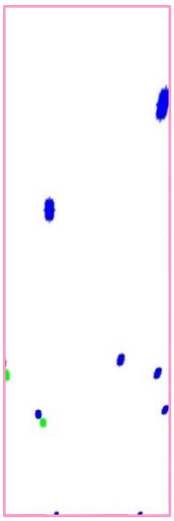
```
#-------------------------------------------------------------------------------
#  main()
#

# a = input("請輸入車頭至前輪的距離(cm)")
# b = input("請輸入車尾至後輪的距離(cm)")
# c = input("請輸入車輛前後輪距(cm)")
# d = input("請輸入車寬(cm)")
# front_len = int(a)
# back_len  = int(b)
# core_len = int(c)
# width = int(d)
#車頭, 車尾, 軸距, 車寬
front_len, back_len, core_len, width = 80, 100, 520, 220
car_info = [front_len, back_len, core_len, width]
print(car_info)

sr = 12
h_width = int(180* (1/12)* sr)
h_height = int((core_len+ back_len)* (1/12) *sr)   #設定視窗大小
print(h_width, h_height)

pos=[] #透視圖偵測熱區四角座標
pos2=[] #透視圖偵測熱區四角座標(依左上,右上,左下,右下排序)

#處理圖片
path = 'resources/photo/setup_4p-1.jpg'
img = cv2.imread(path)
img = cv2.resize(img , (600,600))  #轉換圖形尺寸
imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
imgBlur = cv2.GaussianBlur(imgGray,(7,7),1)
imgCanny = cv2.Canny(imgBlur,50,50)
imgContour = img.copy()
getContours(imgCanny)

imgHotspot = imgContour.copy()  #複製標記出長方形的圖片

sum_xy=[]
sum_xy_sorted=[]
sum_xy=[pos[0][0]+pos[0][1],pos[1][0]+pos[1][1],pos[2][0]+pos[2][1],pos[3][0]+po
s[3][1]]
sum_xy_sorted = sorted(sum_xy)
for i in range(4):
    for j in range(4):
        if sum_xy[j] == sum_xy_sorted[i]:
            pos2.append(pos[j])

#畫出梯形
cv2.line(imgHotspot, (pos2[0][0], pos2[0][1]), (pos2[1][0], pos2[1][1]), (214, 112,
218), 2)
cv2.line(imgHotspot, (pos2[0][0], pos2[0][1]), (pos2[2][0], pos2[2][1]), (214, 112,
218), 2)
cv2.line(imgHotspot, (pos2[1][0], pos2[1][1]), (pos2[3][0], pos2[3][1]), (214, 112,
218), 2)
cv2.line(imgHotspot, (pos2[2][0], pos2[2][1]), (pos2[3][0], pos2[3][1]), (214, 112,
218), 2)

#從透視圖轉為俯視圖
pts1 = np.float32([pos2[0],pos2[2,pos2[1],pos2[3]])#熱區四角座標已排序
pts2 = np.float32([[0,0],[h_width,0],[0,h_height],[h_width,h_height]])#俯視圖四角座
標
```

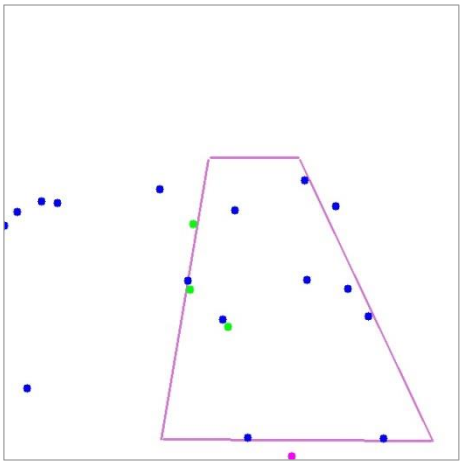
```
matrixT = cv2.getPerspectiveTransform(pts1,pts2)#轉換矩陣
imgOutput = cv2.warpPerspective(img, matrixT,(h_width,h_height))

#堆疊影像
imgBlank = np.zeros_like(img)
imgStack = stackImages(0.5,([img     ,imgGray   ,imgBlur   ],
                        [imgCanny,imgContour,imgHotspot]))

#影像後製及顯示
cv2.imwrite("resources/photo/imgRectangle.jpg",imgOutput)
cv2.imwrite("resources/photo/imgStack.jpg", imgStack)
#cv2.imshow("Stack", imgStack)
print('pos',pos)

img1 = cv2.imread("resources/setup_4p-1.png")

#寫入資料
file = open("newCarSetting.txt", "w")
for x in pos2:
    file.write(str(x[0])+' '+str(x[1]))
    file.write("\n")
for x in car_info:
    file.write(str(x))
    file.write("\n")
file.close()
print('pos2',pos2)
```

## 實驗四、辨識人物車輛並標示出立足點

### 程式主要 用 YOLOv3 模型辨識大型車側人車並標示出立足點

將實驗三所取得的熱區 4 角座標畫出 4 邊形，把運用 YOLOv3 模型所辨識出來的人車點在透視圖上標示畫上立足點，再經由透視變換轉化成長方形的俯視圖範圍，可精確將人物車立足輛點定位於俯視圖。所得的定位圖如下：(模擬由後視鏡觀察到的畫面)





程式碼：

```
import cv2
import numpy as np

net = cv2.dnn.readNetFromDarknet("cfg/yolov3.cfg","yolov3.weights")  #讀取模型
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
classes = [line.strip() for line in open("data/coco.names")]  #分類標籤
colors = [(255,0,0), (0,255,0), (51,153,255), (255,0,255),(0,0,255)] #框選顏色
          ##blue      ##green      ##orange     ##purple
          ##人        ##腳踏車      ##           ##摩托車

img = cv2.imread("resources/photo/truck.jpg")  #讀取圖片
canvas = np.ones((600 ,600 ,3) , dtype = "uint8" )#建立新畫布
canvas[:] = (255 ,255 ,255)#設定畫布色彩

scale_percent = 60             #圖片大小比例設定
print(img.shape[1],img.shape[0])#印出圖片資料

#轉換圖片尺寸
width =600
height =600
dim = (width, height)
resized = cv2.resize(img, dim,  interpolation = cv2.INTER_AREA)
img = resized
blob = cv2.dnn.blobFromImage(img, 1/255.0, (416, 416), (0, 0, 0), True, crop=False)
#圖形預處理以符合輸入圖片規格
net.setInput(blob)  #圖片輸入模型
outs = net.forward(output_layers)  #偵測結果
class_ids=[]#存偵測到的標籤索引
confidences=[]#存信心指數
boxes=[]#存矩形座標
pos=[]#存物體右下角座標

for out in outs:
    for detection in out:
        tx, ty, tw, th, confidence = detection[0:5]  #取得坐標比值及信心資料
        scores = detection[5:]
        class_id = np.argmax(scores)  #取得標籤索引
        if confidence > 0.3:  #信心指數大於 0.3 才算
            center_x = int(tx * width)
            center_y = int(ty * height)
            #換算方框尺寸
            w = int(tw * width)
            h = int(th * height)
            #取得左上角座標
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)
            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)
            #存座標、信心指數、標籤索引加入對應清單

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.3, 0.4)  #消除重疊框選
font = cv2.FONT_HERSHEY_PLAIN#設定字型
pos_4p=[]#儲存讀到的 4 角座標
```

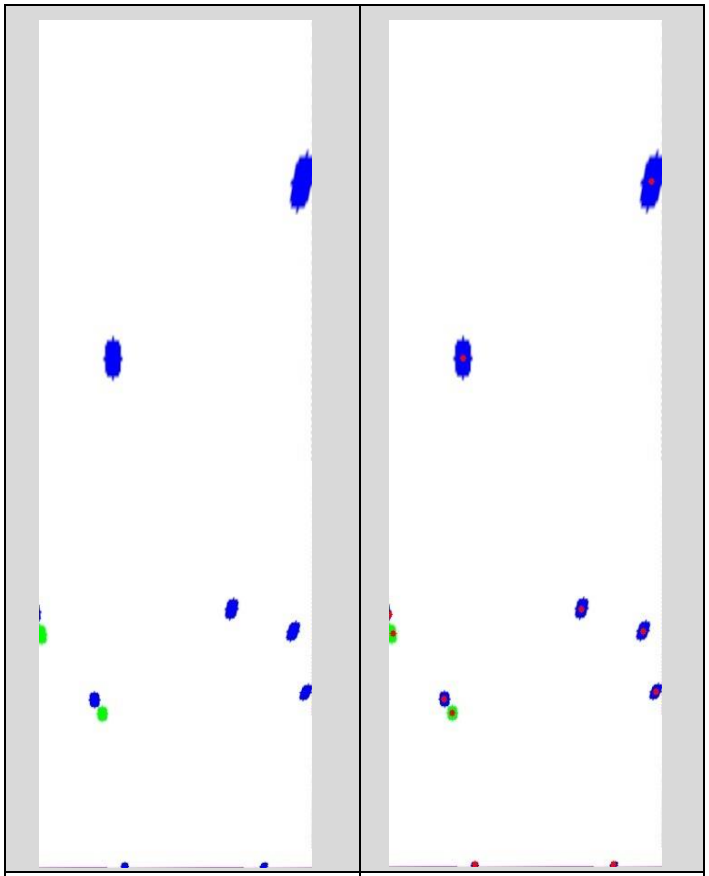
```
fp = open("info.txt", "r")#開啟資料
lines = fp.readlines()

#將資料轉為清單
for l in range(4):
    cor_xy = lines[l].split(' ')
    print(cor_xy)
    pos_4p.append([int(cor_xy[0]),int(cor_xy[1])])

print(pos_4p)
fp.close()

cv2.line(canvas, (pos_4p[0][0], pos_4p[0][1]-3), (pos_4p[1][0], pos_4p[1][1]-3),
(214, 112, 218), 2)#畫出梯形上底的線
cv2.line(canvas, (pos_4p[0][0]-2, pos_4p[0][1]), (pos_4p[2][0]-2, pos_4p[2][1]),
(214, 112, 218), 2)#畫出梯形左斜邊的線
cv2.line(canvas, (pos_4p[1][0]+3, pos_4p[1][1]), (pos_4p[3][0]+3, pos_4p[3][1]),
(214, 112, 218), 2)#畫出梯形右斜邊的線
cv2.line(canvas, (pos_4p[2][0], pos_4p[2][1]), (pos_4p[3][0], pos_4p[3][1]), (214,
112, 218), 2)#畫出梯形下底的線

#畫出物體以右下角座標延伸出的兩條線
for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        a = confidences[i]
        color = colors[class_ids[i]%5]
        cv2.line(img, ( x + w, y),(x + w, y + h), color, 3)#右邊垂直線
        cv2.line(img, ( x, y + h),(x + w, y + h), color, 3)#下方水平線
        cv2.circle(canvas, (x + w, y + h), 5, color, -1)
        cv2.putText(img, label, (x, y - 5), font, 1, color, 2)
        cv2.putText(img, str(a), (x + 15, y - 15), font, 1, color, 2)
        pos.append([x+w,y+h])


#將相機觀察視角轉為俯視圖
d_width, d_height = 300, 800
pts1 = np.float32([pos_4p[0],pos_4p[1],pos_4p[2],pos_4p[3]])
pts2 = np.float32([[0,0],[d_width,0],[0,d_height],[d_width,d_height]])
matrix = cv2.getPerspectiveTransform(pts1,pts2)
imgSpot = cv2.warpPerspective(canvas, matrix,(d_width,d_height))

#輸出螢幕
#cv2.imshow('resized', resized)
#cv2.imshow('canvas', canvas)
#cv2.imshow('imgSpot', imgSpot)

#輸出檔案
cv2.imwrite("data/photo/resized.jpg", resized)
cv2.imwrite("data/photo/canvas.jpg", canvas)
cv2.imwrite("data/photo/spot_person.png", imgSpot)

#偵測並轉換出正確位置
```

## 實驗五、辨識俯視圖中物體與車輛橫向及縱向距離

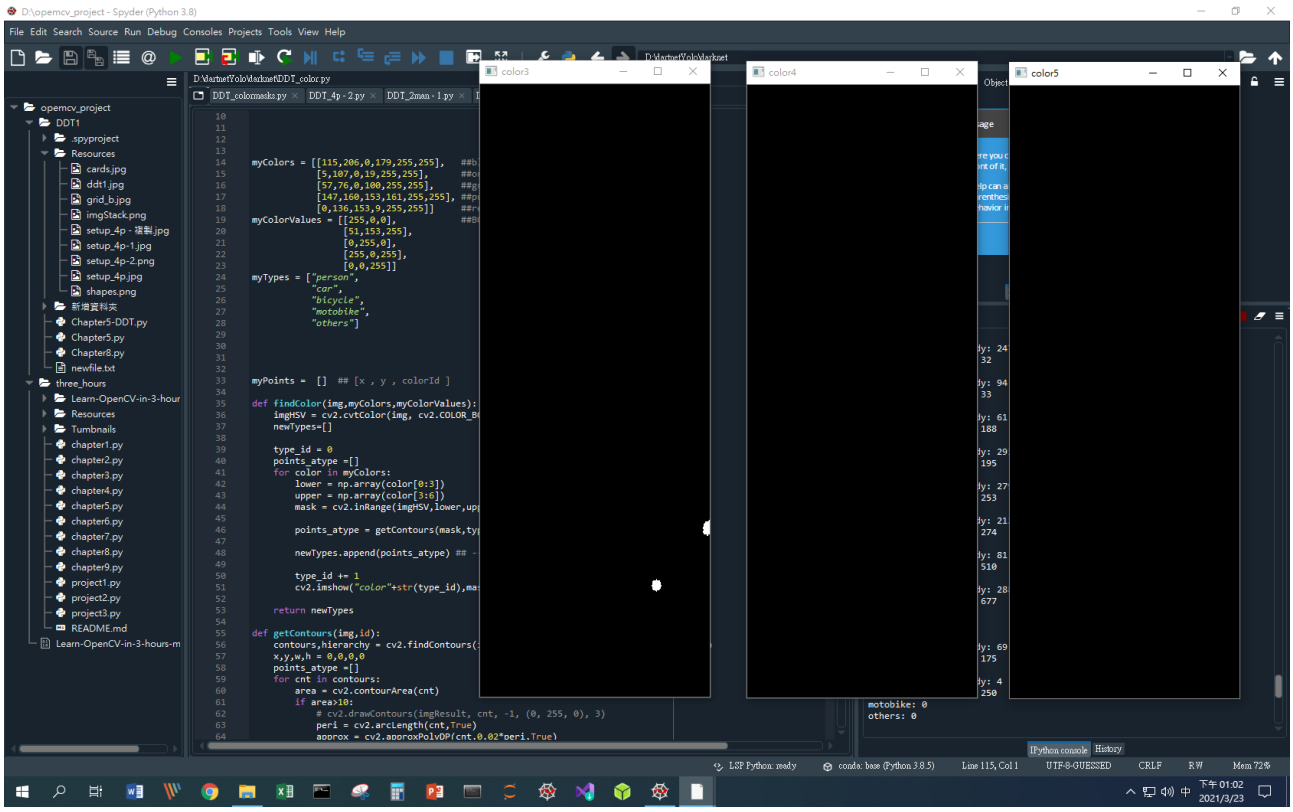
### 主程式：

輸出結果：

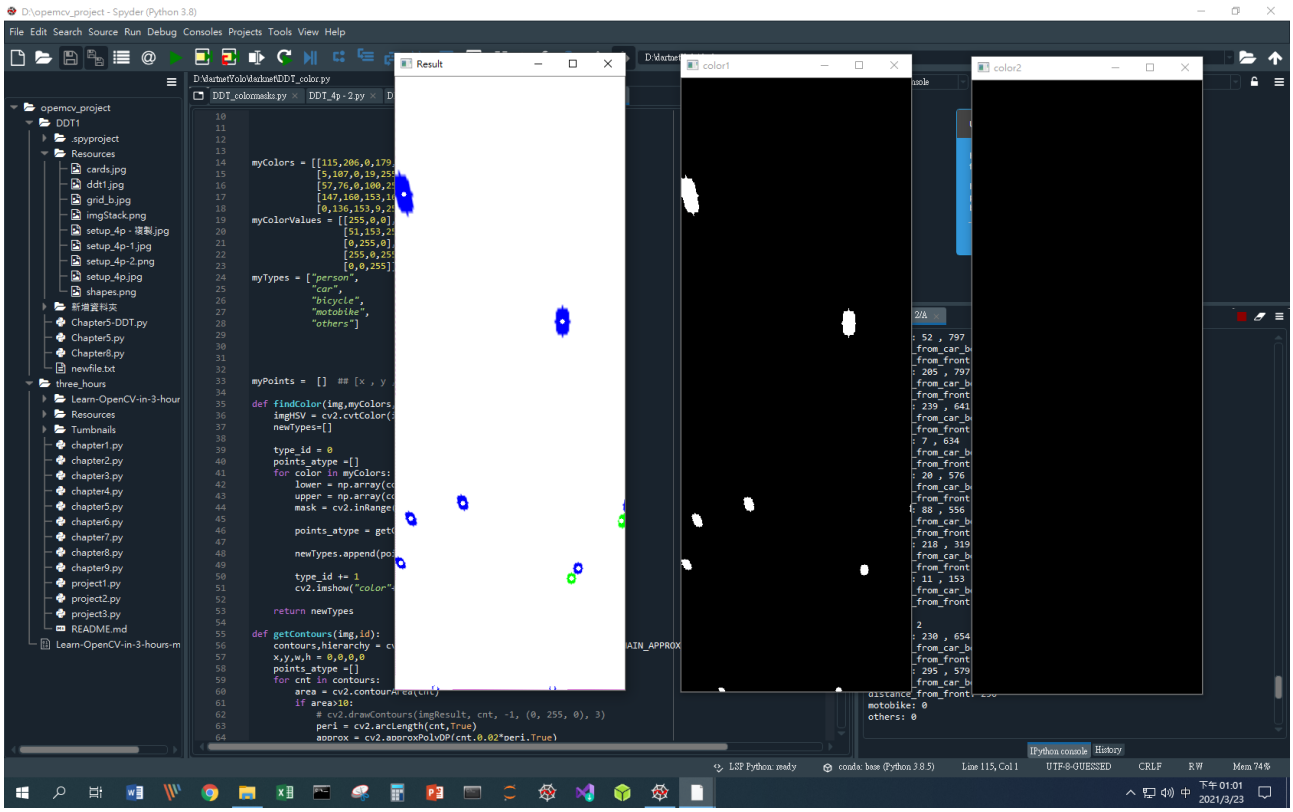
```
80
人: 9
    P: 205 , 797      位置[橫向  56 cm,縱向  81 cm]
    P:  53 , 797      位置[橫向 148 cm,縱向  81 cm]
    P: 239 , 641      位置[橫向  36 cm,縱向 175 cm]
    P:  6 , 634       位置[橫向 176 cm,縱向 179 cm]
    P:  20 , 577      位置[橫向 167 cm,縱向 213 cm]
    P: 299 , 561      位置[橫向   0 cm,縱向 223 cm]
    P:  88 , 556      位置[橫向 127 cm,縱向 226 cm]
    P: 218 , 319      位置[橫向  48 cm,縱向 368 cm]
    P: 11 , 152       位置[橫向 173 cm,縱向 468 cm]
汽車: 0
自行車: 2
    P: 230 , 654      位置[橫向 41 cm,縱向 167 cm]
    P: 295 , 579      位置[橫向  2 cm,縱向 212 cm]
摩托車: 0
其他: 0
熱區圖像[寬,長][ 300 , 800 ]
```



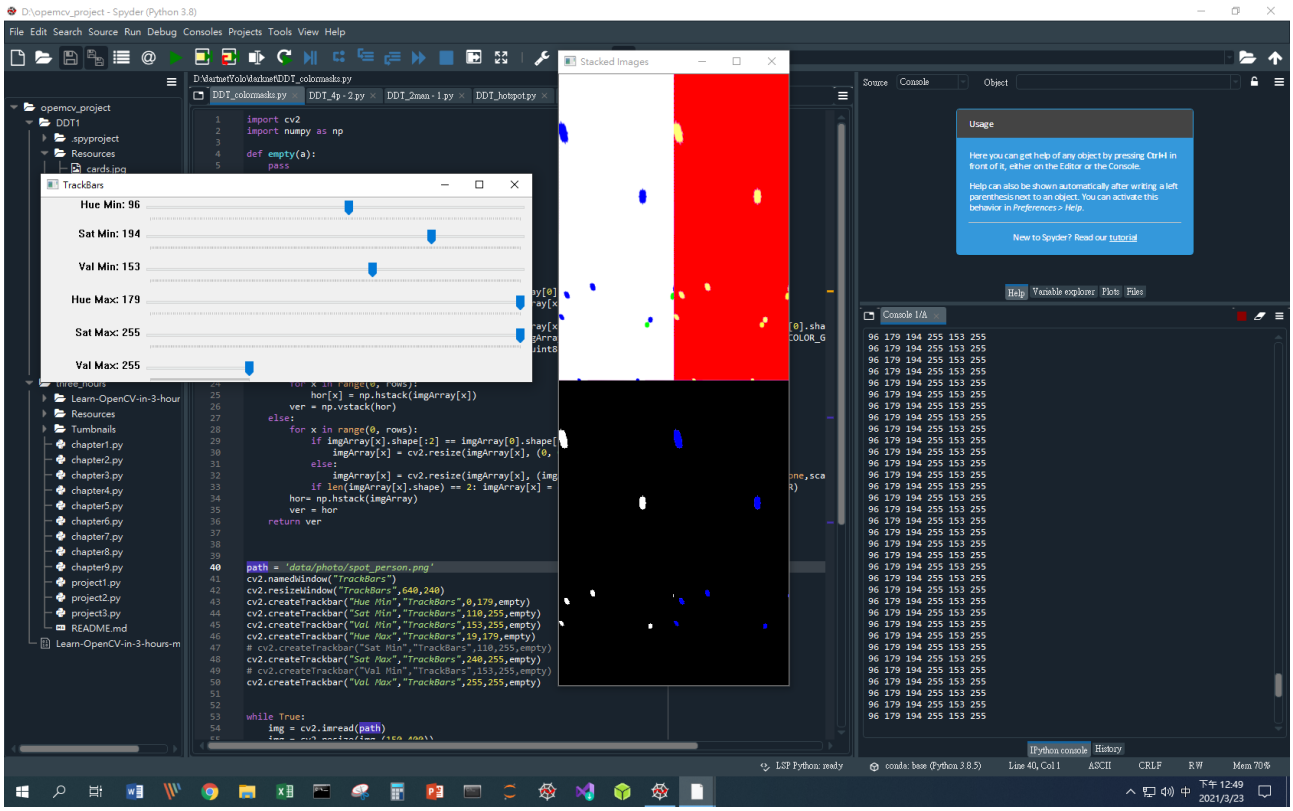
| 未標定 | 已辨識出並標定(紅點) |
|---|---|




綠色遮罩為　自行車位置




藍色遮罩為　行人位置




找尋個顏色 HSV 三值範圍的上限和下限

程式碼：

```
import cv2
import numpy as np

# 初始化清單
myColors = [[115,206,0,179,255,255],   ## HSV blue 藍
            [5,107,0,19,255,255],      #  HSV orange 橘
            [57,76,0,100,255,255],     #  HSV green 綠
            [147,160,153,161,255,255], #  HSV purple 紫
            [0,136,153,9,255,255]]     ## HSV red 紅

myColorValues = [[255,0,0],            ## BGR blue 藍
                 [51,153,255],         #  BGR orange 橘
                 [0,255,0],            #  BGR green 綠
                 [255,0,255],          #  BGR purple 紫
                 [0,0,255]]            ## BGR red 紅

myTypes = ["人","汽車","自行車","摩托車","其他"]

#顏色辨識篩選副程式
def findColor(img, myColors, myColorValues):
    imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    newTypes = []
    t_id = 0
    for color in myColors:
        lower = np.array(color[0:3])
        upper = np.array(color[3:6])
        mask = cv2.inRange(imgHSV,lower,upper)
```

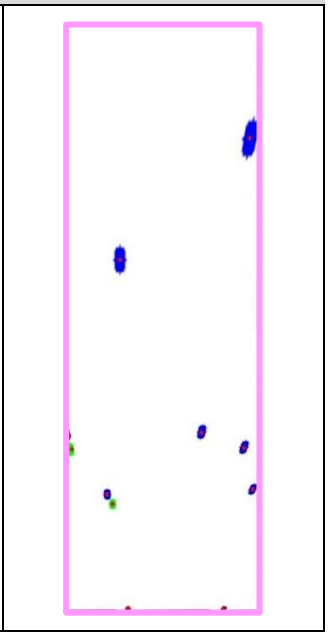
```
        newTypes.append(getContours(mask,t_id))
        # cv2.imshow("color"+str(t_id),mask)
        t_id += 1
    return newTypes

#偵測區塊輪廓副程式
def getContours(img,tid):
    contours,hierarchy =
cv2.findContours(img,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
    #print(hierarchy)                        #印出偵測分類樹

    x,y,w,h = 0,0,0,0
    points_atype = []
    for cnt in contours:
        area = cv2.contourArea(cnt)         #取得輪廓面積
        if area>0:
            #cv2.drawContours(imgResult, cnt, -1, (0, 0, 0), 2)   #畫出輪廓
            peri = cv2.arcLength(cnt,True)                       #取得輪廓周長
            approx = cv2.approxPolyDP(cnt,0.02*peri,True)
            x, y, w, h = cv2.boundingRect(approx)                #取得矩形邊界
            points_atype.append([x, y, w, h])
            #劃出中心點
            cv2.circle(imgResult,(int(x+w/2),int(y+h/2)),3,(0,0,255),cv2.FILLED)
    return points_atype

#--------------------------------------------------------------------------
# main()
#
img = cv2.imread("resources/photo/spot_person.jpg")                  #讀取圖片
imgResult = img.copy()

myPoints = findColor(img, myColors, myColorValues)                   #依顏色分類找點

with open("newCarSetting.txt",'r') as x:                             #取得車輛尺寸參數
    line = x.readlines()
    front_len = int(line[4]) #車頭長
    rare_len = int(line[5])
    print(front_len)

if len(myPoints)!=0:
    # drawOnCanvas(myPoints,myColorValues)
    # print("總清單",myPoints)
    tid = 0
    for aTypes in myPoints:
        print (myTypes[tid]+":",len(aTypes))                         #印出分類名稱:數量
        tid += 1
        for point in aTypes:
            x, y, w, h = point[0],point[1],point[2],point[3]
            print("    P:",int(x+w/2),",",int(y+h/2))                #印出分類清單
            print("    位置[橫向",str(int((img.shape[1]-(x+w/2))/300*180)),"cm,縱向
",str(int(front_len + (img.shape[0]-(y+h/2))/300*180)),"cm]")

print("熱區圖像[寬,長][",img.shape[1],",",img.shape[0],"]")
```

## 實驗六、整合辨識並作危險警示

在實驗三系統軟體初始化完成後，便可整合實驗四、實驗五之程式碼一次執行並停用中間檢查點的各種輸出，便可以一次性偵測出所有車子周遭的人物及車輛。並且在前輪轉彎正負 10 度時，依據實驗二所計算內輪差的範圍一經落入範圍內便啟動語音警告。(合併的程式碼不再重複說明)。最後可依實驗二的數據判定是否安全，若再轉向角為 30 度的情況，輸出結果如下，其中共有五個點是危險的：

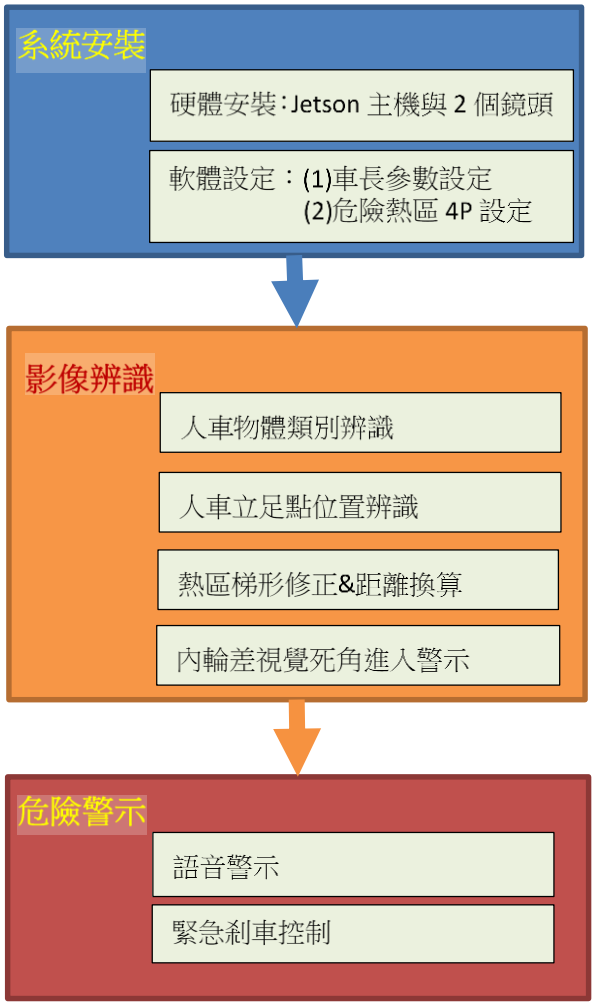
輸出結果：

```
80
人: 9
    P: 205 , 797      位置[橫向  56 cm,縱向  81 cm]    危險
    P:  53 , 797      位置[橫向 148 cm,縱向  81 cm]
    P: 239 , 641      位置[橫向  36 cm,縱向 175 cm]    危險
    P:  6 , 634       位置[橫向 176 cm,縱向 179 cm]
    P:  20 , 577      位置[橫向 167 cm,縱向 213 cm]
    P: 299 , 561      位置[橫向   0 cm,縱向 223 cm]    危險
    P:  88 , 556      位置[橫向 127 cm,縱向 226 cm]
    P: 218 , 319      位置[橫向  48 cm,縱向 368 cm]
    P: 11 , 152       位置[橫向 173 cm,縱向 468 cm]
汽車: 0
自行車: 2
    P: 230 , 654      位置[橫向 41 cm,縱向 167 cm]    危險
    P: 295 , 579      位置[橫向  2 cm,縱向 212 cm]    危險
摩托車: 0
其他: 0
熱區圖像[寬,長][ 300 , 800 ]
```



# 伍、結論

一、本研究能運用影像辨識偵測內輪差及視覺死角範圍內的物體並予以警示，避免大型車輛轉彎時內輪差及視覺死角造成的傷害。科技日新月異，影像辨識的技術也不斷翻新，期待能隨時整合更新的技術造福人群遠離傷害。




大型車轉彎安全警示系統圖

## 二、與其他系統比較：

市售行車輔助系統相關產品中功能與我們最接近的為一款環景影像輔助系統作比較，以下稱為產品 A。

1. 產品 A：透過 4 顆魚眼鏡頭取得路人及路面的影像，欲讓駕駛掌握周遭的情境。其將車子兩側魚眼鏡頭的畫面轉貼在鳥瞰圖上，將球面影像範圍轉換成長方形以構成 360 度無死角畫面。原先的球面影像又再一次扭曲，其結果難以辨識出任何的人物或道路兩側物

體的影像，易使人感到混亂。我們的系統使用 160 度視角的鏡頭，直接準確偵測車子四周範圍內物體並報讀其出現的位置，訊息清晰明確。

2. 產品 A：由於是錄下車外影像提供駕駛觀看，但駕駛可能會因為過度專注於車內的影像而無法專注開車。我們的系統提供語音報讀確指出人事物位置，不干擾駕駛注意力。
3. 產品 A 大車盲區警示系統(較適用於大客車)：在轉彎時，車內螢幕會顯示轉彎向的危險距離，粗略分三種部份：最外側(綠色)、中間(黃色)、最內側(紅色)，當用路人在轉彎時進入大客車的紅色區域時車內會有”叮叮”的警示聲音，提醒駕駛，視覺提醒警示燈則裝在左右 A 柱上，燈泡太小容易錯失提醒。影像提醒容易造成駕駛分心發生意外。我們的系統精確計算人事物位置，及內輪差危險區域，精確判斷。
4. 產品 A 價格 2 萬 1 千元，我們的成本僅為其 3 分之一 優惠實用。

# 陸、未來展望

1. 大型車輛車尾較長時，會有反方向的擺動亦可加入本模組的考量當中以增加安全性。
2. 希望有機會能將此轉彎安全警示配合自駕車的其他模組一起實驗，提高自駕車的安全性，真正造福更多人避免意外傷害。
3. 未來希望能應用更快的提升 OpenCV 的版本以叫用 YOLOv4 進行更快速正確的辨識。

# 柒、參考資料


一、 Python 機器學習超進化：AI 影像辨識跨界應用實戰 鄧文淵,文淵閣工作室 碁峰出版社

二、 危險的距離－死神的區域內輪差，中華民國第 54 屆中小學科學展覽會作品說明書國中組 數學科 桃園縣立楊梅國民中學 張毓文，黃紫涵，呂明軒

三、 後甲國中-利用 Arduino 感應器解決內輪差及視線死角問題
http://activity.tn.edu.tw/study/work/107/利用 Arduino 感應器解決內輪差及視線死角問題.pdf

四、 研之有物：一眼揪出你有沒有超速！世界第一物件偵測技術：YOLOv4
https://research.sinica.edu.tw/yolov4-hong-yuan-mark-liao/

# 【評語】032808

1. 本作品應用 AI 影像辨識系統偵測車輛轉彎因內輪差造成駕駛視線死角區的行人，即時發出音訊警告，這個主題切合生活實際場合。

2. 由於以模型車進行測試，實物卡車與實驗製作卡車有一定比例上差距，且未能在實際車輛與道路上測試，是否能克服實際的複雜路況，推持正確的穩定度，如何校正實際角度位置與距離等，需進一步釐清，才能推論其優於現行市面商品。

3. 來源影像受限於車載取像裝置之影像拍攝視角，並使用 OPEN CV、YOLOv4 開源軟體，在影像擷取完整性以及計算的延遲方面的影響，仍可深入探討。

4. 總體而言，本作品有很好的創意，且作者發揮團隊合作精神，值得嘉獎。

# 作品簡報

作品編號：032808　　國中組 生活與應用科學(一)科

# 智慧影像，「One 視」平安~

## 大型車轉彎安全偵測警示系統

# 研究動機

現階段政府規定大型車須裝設駕駛輔助系統，其中最常見的形式為以語音警告「**車輛要轉彎，請注意安全。**」


圖片來源：

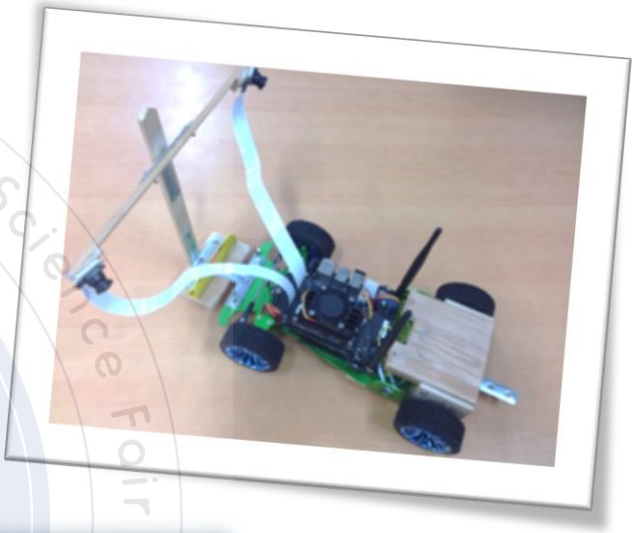
然而，用路人若來不及離開危險區域，或因行動不便無能力避開危險，傷害可能就此發生。

我們認為若能**及時以影像偵測危險狀態、主動告知駕駛**，協助司機**積極地執行安全駕駛**，可使憾事不再發生。

# 研究架構

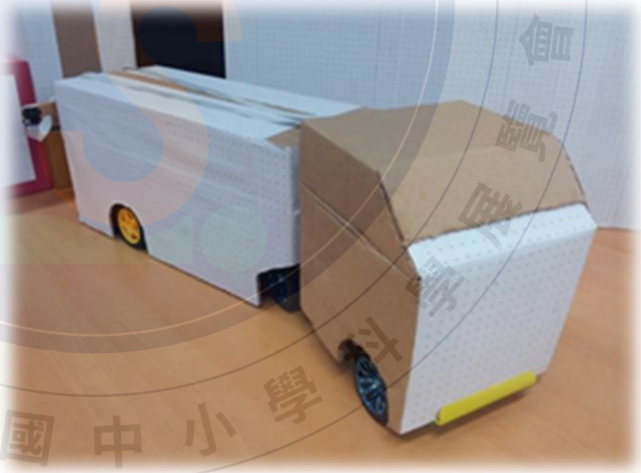

大型車轉彎安全偵測警示系統研發
主系統
1設定車輛長寬參數
2標定危險熱區
3將梯形的透視圖轉為長方形的俯視圖
4辨識車側人物車輛並標示立足點
5計算物體與車輛之相對位置
6語音警示危險
研究平台
1建置輔助駕駛研究平台
2製作大客車及聯結車模型
3遠端遙控實驗取得內輪差數據
4計算內輪差安全距離

# 建置研究平台



1:12大客車

1:12聯結車



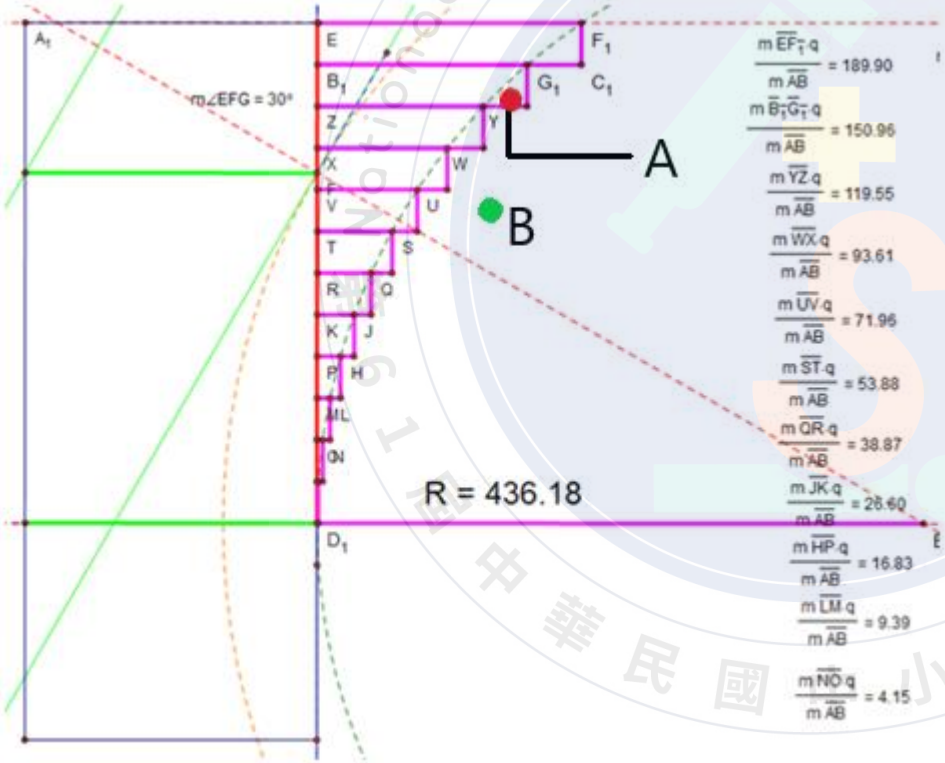
# 實驗及結果~

## 一、使用無線搖桿進行車輛遠端遙控

使用WiFi網路、無線搖桿操控模型車收集內輪差的數據

# 實驗及結果~

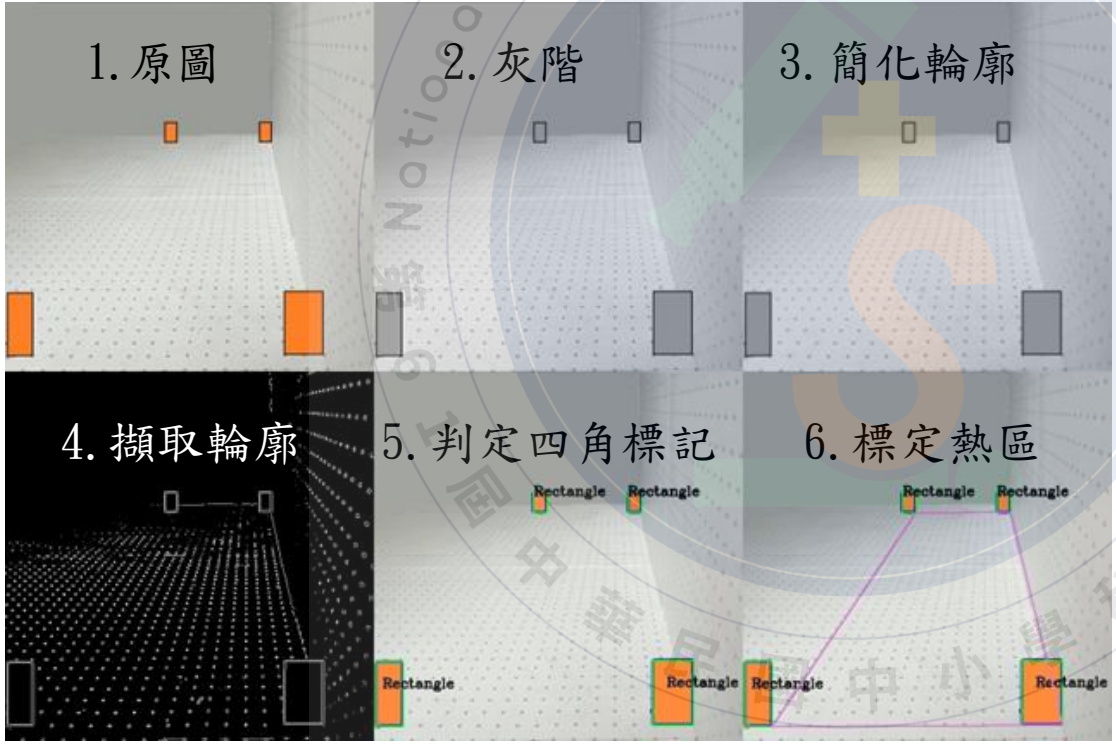
## 二、計算內輪差的安全距離




| 縱向位置(cm) | 安全距離(cm) |
|---|---|
| 0 | 189.90 |
| 30 | 150.96 |
| 60 | 119.55 |
| 90 | 93.61 |
| 120 | 71.96 |
| 150 | 53.88 |
| 180 | 38.87 |
| 210 | 26.60 |
| 240 | 16.83 |
| 270 | 9.39 |
| 300 | 4.15 |

# 實驗及結果~

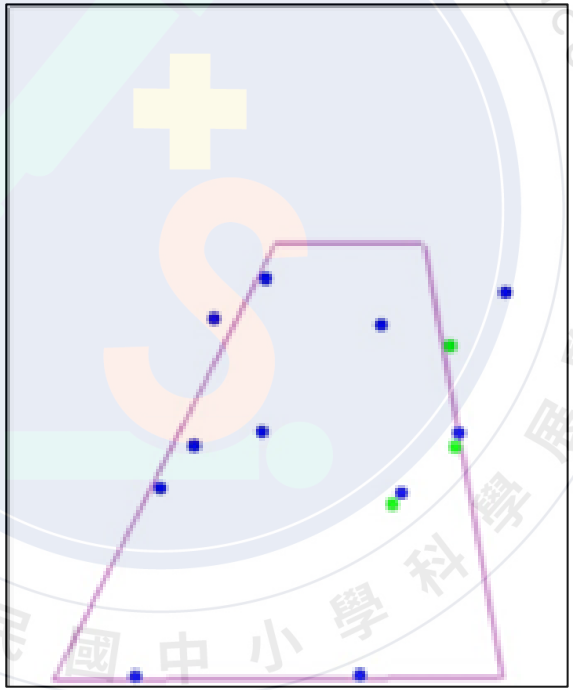
## 三、輸入參數&標定熱區&影像透視變換

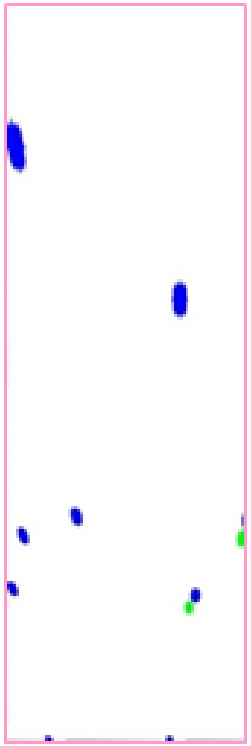



長方形俯視圖

# 實驗及結果~

## 四、辨識車側人物車輛並標示出立足點

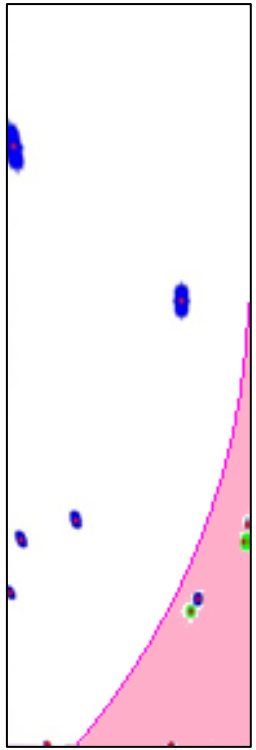


# 實驗及結果~

## 五、辨識物體與車輛橫向及縱向距離

輸出結果：

```
人: 9
    P: 205 , 797      位置[橫向  56 cm,縱向  81 cm]  危險
    P:  53 , 797      位置[橫向 148 cm,縱向  81 cm]
    P: 239 , 641      位置[橫向  36 cm,縱向 175 cm]  危險
    P:   6 , 634      位置[橫向 176 cm,縱向 179 cm]
    P:  20 , 577      位置[橫向 167 cm,縱向 213 cm]
    P: 299 , 561      位置[橫向   0 cm,縱向 223 cm]  危險
    P:  88 , 556      位置[橫向 127 cm,縱向 226 cm]
    P: 218 , 319      位置[橫向  48 cm,縱向 368 cm]
    P: 11 , 152       位置[橫向 173 cm,縱向 468 cm]
汽車: 0
自行車: 2
    P: 230 , 654      位置[橫向 41 cm,縱向 167 cm]  危險
    P: 295 , 579      位置[橫向  2 cm,縱向 212 cm]  危險
摩托車: 0
其他: 0
熱區圖像[寬,長][ 300 , 800 ]
```



# 實驗及結果~

## 六、透過語音模組，警示駕駛

橫向56，縱向81公分處
有3人，危險！

# 結論

1. 本研究建置了開發平台並建立車輛轉彎時內輪差數據，以影像辨識技術偵測危險範圍內的物體並警示，能及時避免造成傷害。
2. 本研究透過影像辨識建立危險熱區，並標出立足點、判斷是否落入危險區域。如在危險範圍內立刻主動用語音提醒駕駛，輔助駕駛能及時反應並且擔起更多交通安全的責任。

# 未來展望

1. 希望和客運及貨運公司合作安裝本系統，以維護及提升國人的交通安全。
2. 希望能將此轉彎安全警示系統配合自駕車的模組，以提高自駕車的安全性，造福更多用路人避免意外傷害。
3. 希望能應用更快的OpenCV版本以叫用YOLOv4，進行更快速更正確的辨識。

參考文獻:危險的距離－死神的區域內輪差，中華民國第 54 屆中小學科學展覽會作品說明書國中組 數學科
桃園縣立楊梅國民中學 張毓文，黃紫涵，呂明軒