章節目錄

圖目錄

表格目錄

第一章 緒論

1.1 研究動機

根據世界衛生組織 2019 年出版的《世界視力報告》統計，全球視力損傷或失明人口高達 22 億以上，並且預計在未來的幾十年內，受人口增長與和老齡化等綜合因素影響，患有眼部疾患和視力受損之總人數將顯著上升。

對於視覺障礙者來說自立生活 (independent living)不是簡單的，其中定向(Orientation)及行動 (Mobility)是自立生活中不可或缺的能力。本研究之主要目的旨在提供視障者在戶外環境中所需的感知能力，消除視障者在行走或過馬路時的安全疑慮。雖然台灣在對於視障者的用路安全上設有許多的無障礙設施，例如：導盲磚、過馬路的南北向咘咕聲等，但無障礙環境尚未能全面落實，其中過馬路的南北向咘咕聲設置成本高，且盲人在使用前必須先尋找到按鈕所在之電線杆及按鈕位置。我們這次研究的主題主要會圍繞在透過影像處理的方式對路上障礙物、斑馬線位置、紅綠燈判別等等之分析。

1.2 研究問題

1. 軟體開發及整合 - 影像處理加入深度學習

本研究利用攝影機，也將影像資訊回傳至後端電腦，並利用影像分析的結合 AI 深度學習之功能進行自動分析。研發過程中要不斷克服軟體上的問題，並持續測試其效能是否如同預期。

2. 針對視障者的安全 - 為視障者提供安全、便利的導航

本研究以盲人視覺輔助結合 GPS 導航技術，目標在於視障人士日常生活的獨立與行動自由。儘管過去已有許多盲人導航相關之輔助科技，目前最普遍之輔具仍為導盲白色手杖或是導盲犬。本研究將建置具備視覺與深度學習之實體導引視障者，可在室內外無軌跡或有軌跡之環境進行導航任務。

1.3 預計目標

1. 斑馬線位置尋找

斑馬線，設計讓穿越路口的行人集中由固定的地點通過，也限制汽機車駕駛人在有行人正在橫過或等候橫過馬路的狀況下必須優先禮讓行人。斑馬線的設計，保障了行人穿越路的安全性，但對於視障者來說，確認斑馬線位置與確保自己走在斑馬線上卻是一大挑戰，本系統將透過影像分析引導盲人安全地利用斑馬線穿越路口。

2. 紅綠燈號誌辨別

「紅燈停、綠燈行、黃燈要小心。」簡單的口號，但對視覺障礙者來說卻不簡單。本系統將透過影像分析判別路口紅綠燈號誌，並以語音告知，協助視覺障礙者們和正常的行人一樣平安地通過馬路。

3. 目的地路線規劃

對於視覺障礙者們來說，每到一個新地方都像是前往一個陌生的國家，在一個人生地不熟的地方，一般人可以選擇 GPS 導航來協助自己，但這個選項對於視覺障礙者們說也不全然這麼方便。本系統將結合 GPS 導航與語音控制系統，使他們能通過語音系統的指引，成功抵達目的地，藉此大大提升視覺障礙者使用導航功能的方便性。

4. 語音輸出

對於一個視障者而言，他們並沒辦法像一般人一樣靠著自己的的眼睛去獲得完整的資訊，因此本專題利用耳機連接系統，將我們要傳達的訊息清楚且完善的傳達出來。

第二章 研究方法

2.1 文獻參考

1. 視障者在城市空間中的移動經驗

蘇怡帆、黃國晏、畢恆達(2012)[1]指出：視障者在城市中的移動經驗，受到個人身體特質、科技輔具、社會關係、空間規劃與設計等因素之相互作用而得以實踐。

公共空間環境中的限制主要在於空間訊息以視覺為主，造成視障者取得空間訊息的困境。舉例來說，在城市的公共生活中，明眼人依賴招牌或櫥窗來展示商品、以紅綠燈決定行走的秩序、使用站牌標示公車號碼、以平面地圖確認方位、用觸控式螢幕提款、鈔票上印的墨水數字決定每一張紙鈔的面額大小、建築物的用途名稱也以文字顯示。視障者身處視覺文化霸權的環境中，往往必須與有限的環境資源奮戰。

2.2 理論與方法

2.2.1 深度學習結構建立影像辨識系統[2]

1. 調整圖片大小
   深度學習模型對圖片的大小有很強的要求，必須使用相同大小的數據進行處理，所以實現 yolo 的第一步是在進入神經網絡之前調整全部圖像大小。
2. 對整個圖像進行卷積
   當使用傳統的 DNN（深度學習網絡）進行圖像或圖像識別時，需要對圖像逐一像像素視為一個特徵值進行分析，但特徵值的提取方法是將圖片的二維空間轉換為一維空間，因此分析後往往會打亂原始空間的排列。但是，CNN（卷積神經網絡）具有以下功能，解決了 DNN 的缺點，也為神經網絡帶來了更高的性能。

   a. Convolution layer (卷積層)：的目的就是在保留圖像的空間排列並取得局部圖像作為特徵。

   b. Pooling layer(池化層)：將輸入圖片尺寸縮小以減少每張特徵圖維度並保持重要的特徵，同時帶來以下好處：
   - 減少後續 layer 需要參數，加快系統運作的效率。
   - 具有抗干擾的作用：圖像中某些像素在鄰近區域有微小偏移或差異時，Pooling layer 的輸出影響不大，結果仍是不變的。
   - 減少過度擬合 over-fitting 的情況。

3. 使用非最大值抑制算法求得目標區域 NMS(非最大值抑制算法)目的在於消除多餘的邊界框，找尋最佳的物體檢測位置，其運算法的核心為以下特點

   a. 信度最高的一個 boundingbox(bbox)作為目標，然後對比剩下 bbox 與目標 bbox 之間的交叉區域

   b. 如果交叉區域大於設定的閾值，那麼在剩下的 bbox 中去除該 bbox(即使該 bbox 的置信度與目標 bbox 的可信度一樣)

   c. 將第二置信度高的 bbox 作為目標，重複 1、2

2.2.2 以 YOLOv4 製作特點之實現[3]

YOLOv4 是目前對於物件偵測被最廣泛應用的一項技術，在保證速度的同時，大幅提高模型的檢測精度，並降低硬體使用的要求。由下圖可以看見 YOLOv4 在 MS COCO 數據集上獲得了 43.5% 的 AP 值 (65.7% AP50)。在與 EfficientDet 同等性能的情況下，速度是 EfficientDet 的二倍；而與 YOLOv3 相比，AP 和 FPS 分別提高了 10% 和 12%。

Yolov4 與 Yolov3 相比，其實整體架構是相同的，但多了 CSP 結構與 PAN 結構。下列簡單介紹 Yolov4 主要創新之處。

1. 輸入端：這裡指的創新主要是訓練時對輸入端的改進，主要包括 Mosaic 數據增強、cmBN、SAT 自對抗訓練。

2. BackBone 主幹網絡：將各種新的方式結合起來，包括：CSPDarknet53、Mish 激活函數、Dropblock。

3. Neck：目標檢測網絡在 BackBone 和最後的輸出層之間往往會插入一些層，比如 Yolov4 中的 SPP 模塊、FPN+PAN 結構。

4. Prediction：輸出層的錨框機制和 Yolov3 相同，主要改進的是訓練時的損失函數 CIOU_Loss，以及預測框篩選的 nms 變為 DIOU_nms

2.3 研究步驟

我們將整個研究過程分為以下四個階段，並根據每個階段的情況規劃進度表。

2.3.1 第一階段：訓練機器學習方法正確辨識不同物件

1. 使用攝影機紀錄斑馬線、紅綠燈之影像。
2. 以 Python 整合 OpenCV、Keras、TensorFlow 等系統，用收集好的影像訓練機器學習方式辨別各種不同類型的物件（斑馬線、紅綠燈、行人、汽機車等）。
3. 使用攝影機將新影像輸入至訓練好的機器學習方法，並得到正確的辨識結果。

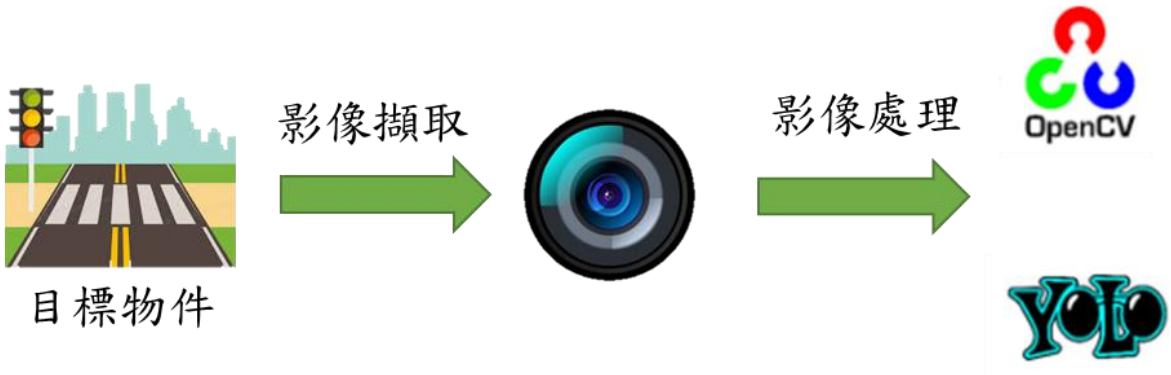

圖 2.3-1 物件樣本訓練架構

2.3.2 第二階段：打造與使用者間溝通的橋樑

1. 設計一個方便使用者配戴的裝置
2. 安裝語音輸出所需軟體及硬體
3. 模擬各種判斷的結果，傳輸給耳機進行輸出

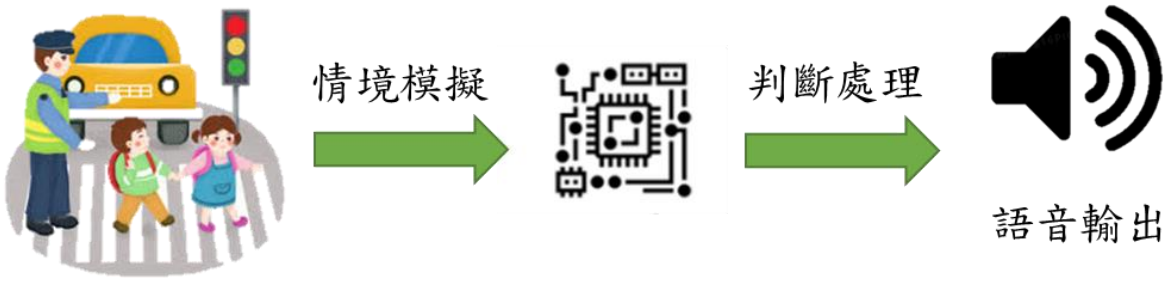

圖 2.3-2 使用者操作模擬

2.3.3 第三階段：Google API 導航及系統整合

1. 安裝 GPS 模組以獲取 GPS 訊號

2. 利用 GPS 訊號結合 Google API 進行導航

圖 2.3-3 導航流程

3. 將上述提到的所有功能進行整合

2.3.4 專案甘特圖

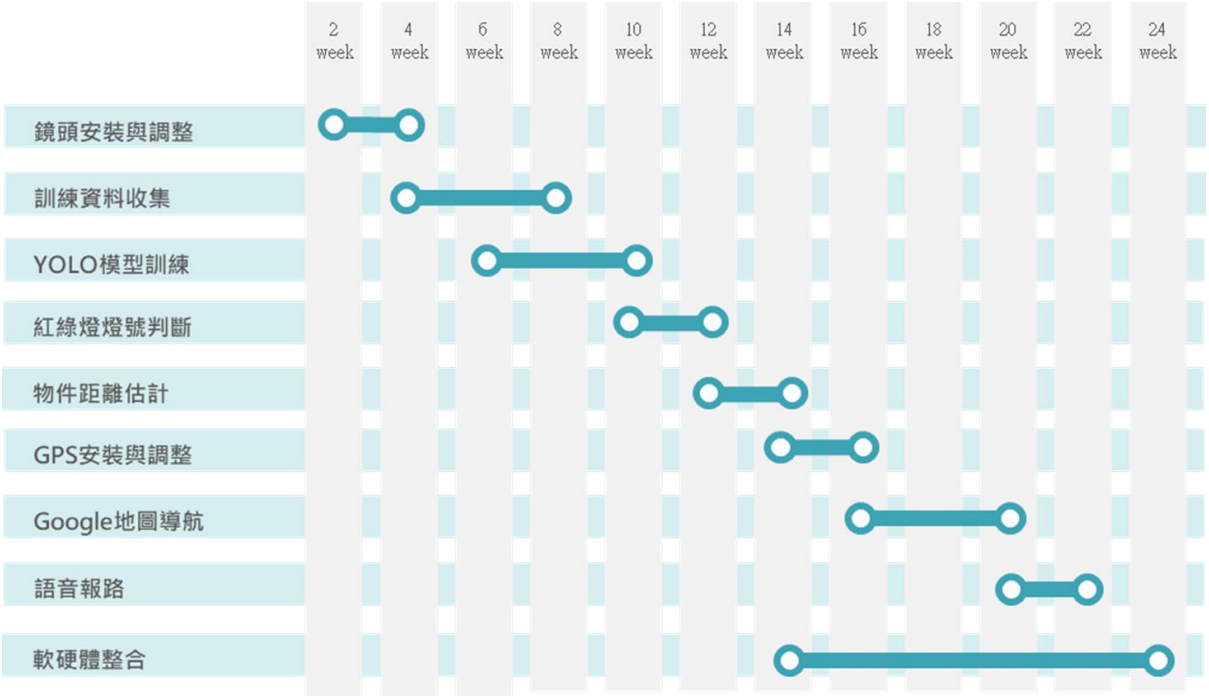

圖 2.3-4 專案甘特圖

2.4 影像資料增補(Data Augmentation)

在進行深度學習訓練時，我們經常需要大量的數據來保證訓練過程中不會產生過擬合(Overfitting)，但資料是稀缺的資源，大部份有價值的資料都掌握在資金雄厚的公司或在相關領域深耕許久的企業手裡，個人開發者或普通公司很難擁有或者搜集所需的資料。因此，我們採取影像資料增補來解決這個問題。

影像資料增補是將單一圖片經過旋轉、調整大小、縮放或亮度、色溫、翻轉等改變後，我們的人眼仍然可以將其識別為同一張照片，但對於機器來說卻是完全不同的新圖像，所以，資料增補是對已有的圖片進行修改和變形，以創建更多的圖片供機器學習使用，彌補數據上的不足。

第三章研究設備與工具

3.1 硬體架構

1. 影像擷取：樹莓派(Raspberry Pi) 攝影機模組[4]

   為了達到 360°保護視障者的目標，我們特別選用兩顆 225 度超廣角魚眼攝影機模組，分別設置於一前一後，全方面接收視障者所在的環境資料。此攝影機模組能錄製每秒 30 張的 1080p 高畫質影片，並且配有一對紅外線夜視補光燈，能偵測環境光暗，自動調整，讓攝影機能獲得更真實的環境資料。

2. 樹莓派 Raspberry Pi[5]

   樹莓派，是一款基於 Linux 系統的單板機電腦，體積只有一張信用卡的大小，是一張平價且易取得的開發版，只要搭配簡單的配件即可組合出不同的應用。它由英國的樹莓派基金會所開發，目的是以低價硬體及自由軟體刺激在學校的基本電腦科學教育。由於樹莓派設計有對外的輸出/入介面(GPIO)，可讓程式師控制外部的自組電路，因此樹莓派也常被使用於機電控制方面的領域。

3. Jetson Xavier NX[6]

   Jetson Xavier NX 是 NVIDIA 推出的小尺寸模組系統 (SOM)。雖然體積僅 70 mm x 45 mm，但卻可提供高達 21 兆次的運算能力，以及超過每秒 59.7 GB 的記憶體頻寬、影片編碼與解碼功能。讓使用者能以平行方式執行多個現代類神經網路，及處理多個高解析度感應器的資料，十分適合在嵌入式和邊緣系統中作為主要運算中心。

3.2 軟體架構

1. 影像處理： OpenCV[7]

   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，可以在商業和研究領域中免費使用。OpenCV 可用於開發實時的圖像處理、電腦視覺以及模式識別程式。該程式庫也可以使用英特爾公司的 IPP 進行加速處理。而 OpenCV 被廣泛使用於解決如人機互動，物體識別，圖像分割，人臉識別，動作識別，運動跟蹤，機器人操作。

2. 開發語言： Python

   Python 的創始人為 Guido van Rossum，為一種物件導向的高階程式語言，強調程式碼的簡潔、明確，因此相比其他高階語言(C++、Java)能夠用更少的程式碼完成功能，方便使用並可在大多數的系統中運行，以減少開發及維護成本的觀念進行發展。

   除此之外，Python 也具有豐富和強大的函式庫，足以支持絕大多數日常應用，像是 Web 程式、作業系統、GUI 開發、遊戲、網頁爬蟲以及機器學習等.....各個方面，Python 都經常被拿來應用。而我們將透過 Python 來整合 OpenCV、Keras、TensorFlow 這些影像分析及深度學習相關的功能。

3. 深度學習：YOLOV4

   YOLO 系列 (You only look once， Yolo)是物件偵測 (object detection) 的類神經網路演算法，最大的特色是直接 end-to-end 做物件偵測，利用整張圖片作為神經網路的輸入，直接預測物品坐標位置、預測信心值(confidence) 和物體所屬的類別。

   其輕量、依賴少、演算法高效率的特性，能夠達到即時(real-time)偵測 的速度需求，在工業應用領域很有價值，例如：行人偵測、工業影像偵測等等。缺點是對位置的預測較不精確，且對小物體偵測的效果不佳。

4. Google Map API (Application Programming Interface)

   Google map api 是由 Google 所提供的服務，允許使用者使用自己的內容和圖像自定義地圖，以便在網頁和移動設備上顯示。 Maps JavaScript API 具有四種基本地圖類型（路線圖、衛星、混合和地形），使用者可以使用圖層和 樣式、控件和事件以及各種服務和庫對其進行修改，以建立強大的地圖應用程式。

5. Pubnub[8]

PubNub 是一家位於加利福尼亞州舊金山的實時通信平台和實時基礎設施即服務 (IaaS) 公司。該公司為軟件和硬件開發人員提供產品，以構建實時 Web、移動和物聯網 (IoT) 應用程序。

6. Espeak

是一款適用於 Linux、Windows 和其他平台的開源軟體語音合成器。其輕量且支援多國語言的特性，正好適合為本研究的語音通知系統提供支援。

第四章研究結果與分析

4.1 系統架構

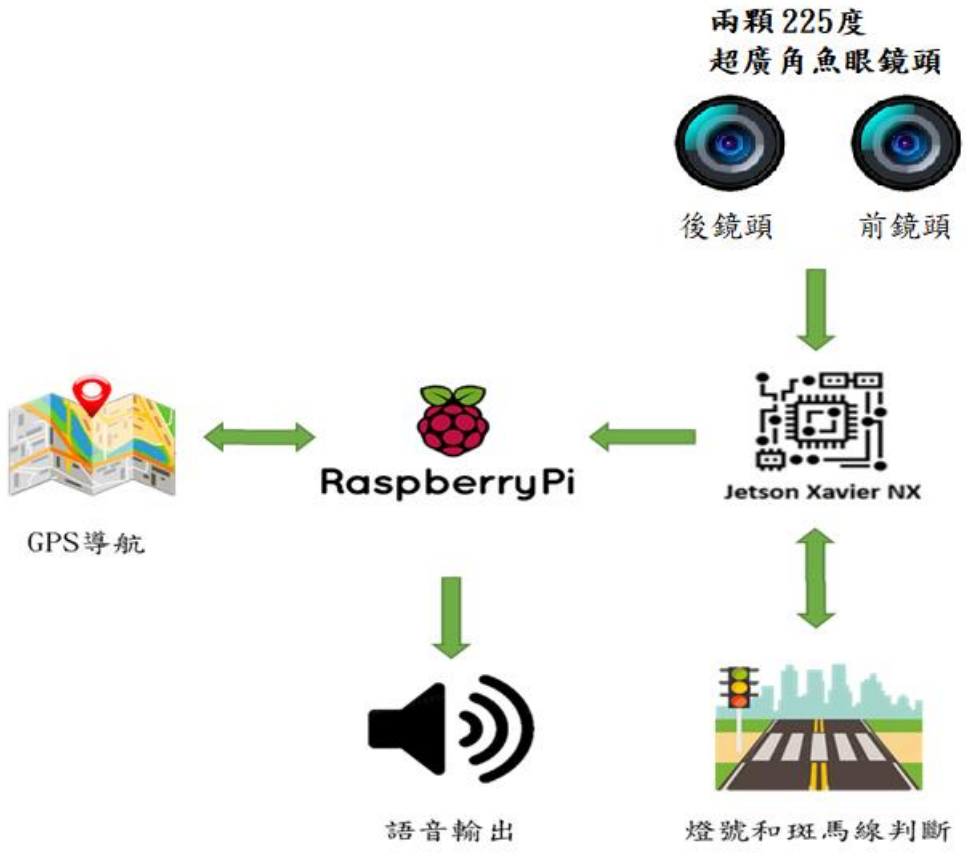

圖 4.1-1 系統架構圖

這次專題所達成的功能，是依靠著 Jetson Xavier NX 以及樹莓派協作來完成的。Jetson Xavier NX 擁有較好的運算能力，所以負責較複雜的運算，例如影像及深度學習的處理，並且將最終運算結果傳給樹莓派。樹莓派在獲得結果的同時會進行導航，並將資料整合後透過耳機輸出。

4.1.1 Jetson Xavier NX 執行重點

1. 影像擷取

   將鏡頭裝設在 Jetson Xavier NX 上面，以利於進行即時影像處理。

2. 影像處理：魚眼矯正

   由於我們的魚眼鏡頭所獲取到的影像是變形的，我們會將其逕行矯正以利於 YOLO 的深度學習

3. 深度學習：YOLOV4

   因為 Jetson Xavier NX 有相較於樹莓派更佳的運算能力，所以負責執行 YOLO 物體識別 (Object detection)的運算，在判別斑馬線、紅綠燈等等的位置後，再將結果傳至樹莓派。

   Jetson Xavier NX 會同時處理前後鏡頭的影像輸入，前鏡頭影像資料主要用於定位斑馬線位置，以及判別紅綠燈燈號；後鏡頭影像資料則主要偵測行人、腳踏車與車輛的運行，以確保視障者的安全。

4.1.2 樹莓派執行重點

1. GPS 訊號獲取

   結合 GPS 模組來獲得 GPS 訊號

2. 導航

   將獲得的 GPS 訊號結合 Google MAP API 進行導航

3. 語音輸出

   從上面導航運算結果加上前面 Jetson Xavier NX 運算完所回傳過來的結果，進行判斷然後透過耳機輸出語音

4.2 系統流程圖

4.2.1 Jetson Xavier NX 系統流程圖

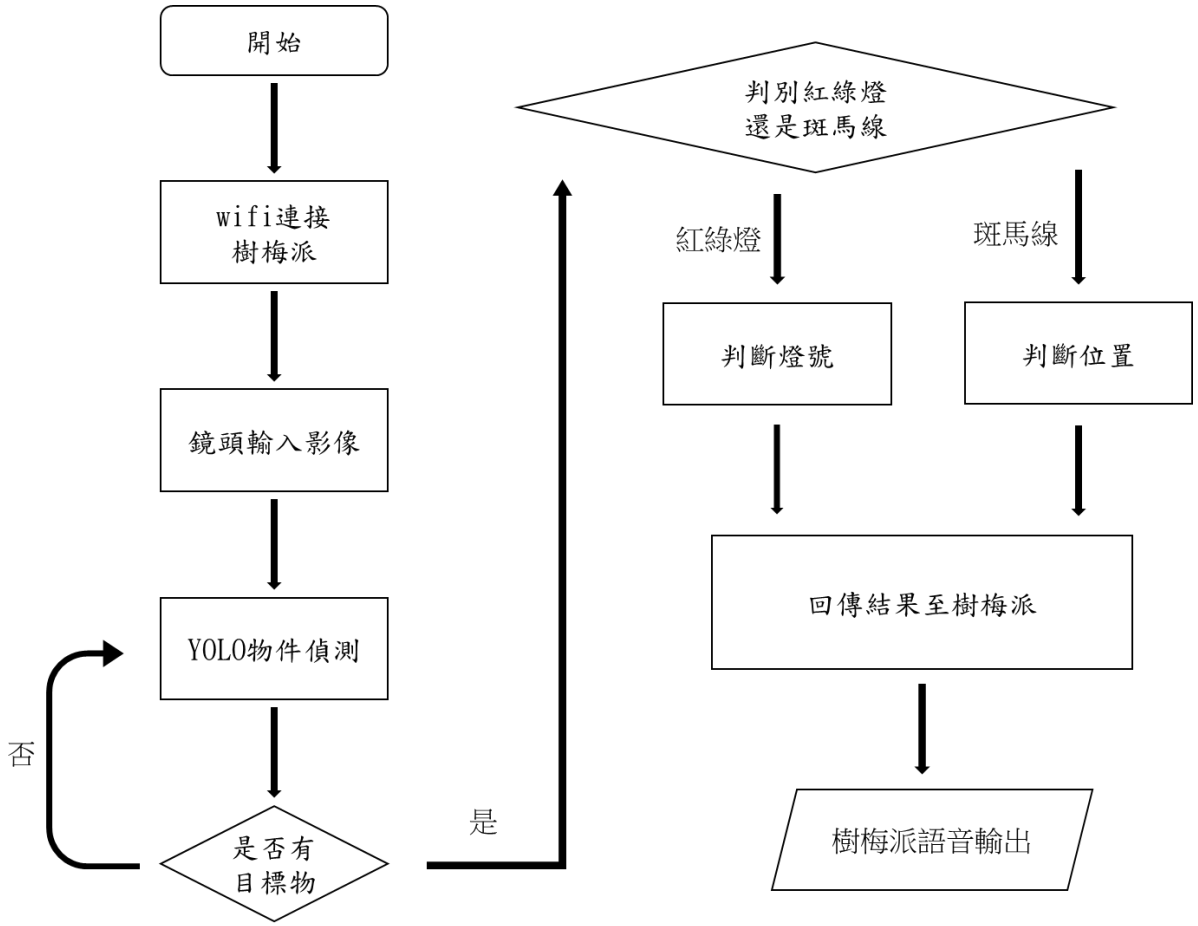

圖 4.2-1Jetson Xavier NX 系統流程圖

4.2.2 樹莓派系統流程圖

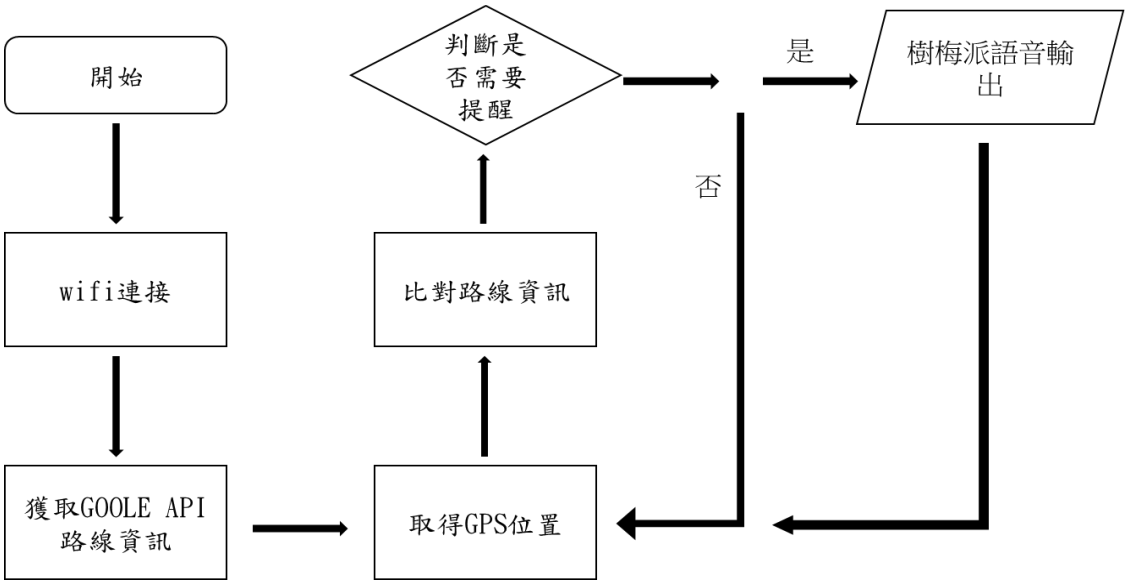

圖 4.2-2 樹莓派系統流程圖 (圖 7)

4.3 深度學習模型製作

4.3.1 YOLO 訓練[9]

我們使用 YOLO 深度學習模型架構建立辨識模型。
以下，我們使用綠燈、紅燈、斑馬線三種不同的物件來建立模型。

首先，準備大量的圖片作為訓練資料，之後將圖片中的物件以定界框（Bounding box）做標記（Annotation）。標記後的圖片會產生.cfg 檔，其中會包含圖片編碼，以及圖片中各個物件的位置資訊。

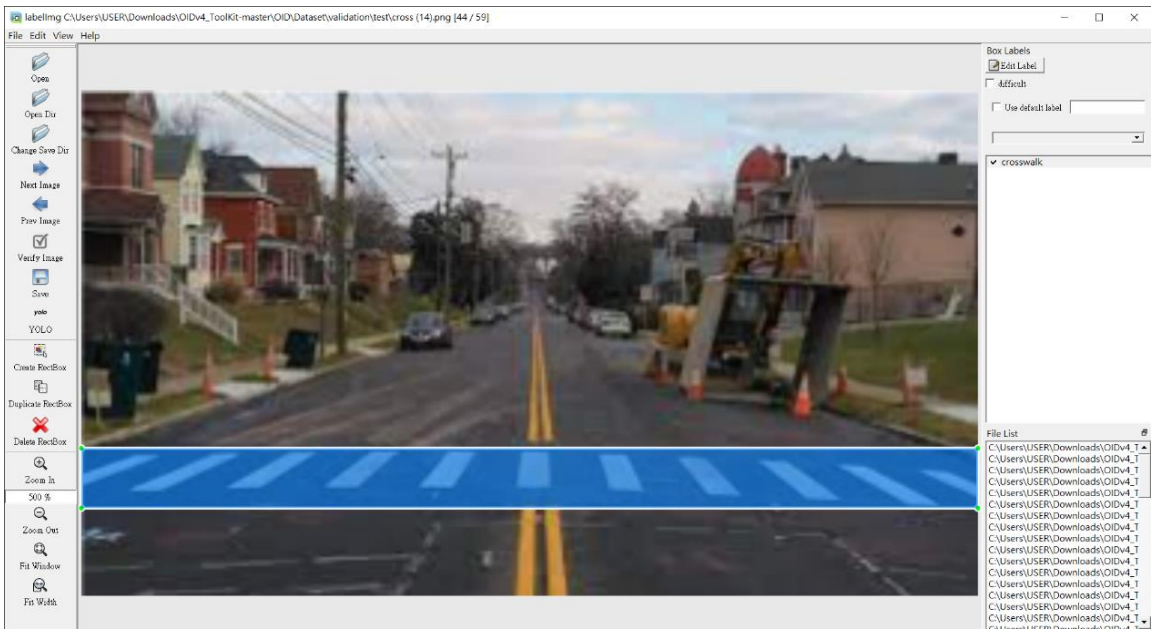

圖 4.3-1 影像標記 (圖 8)

4.3.1 資料增補（Data augmentation）

用影像資料增補技術將單張影像透過旋轉、位移等影像處理方法變為多張影像。

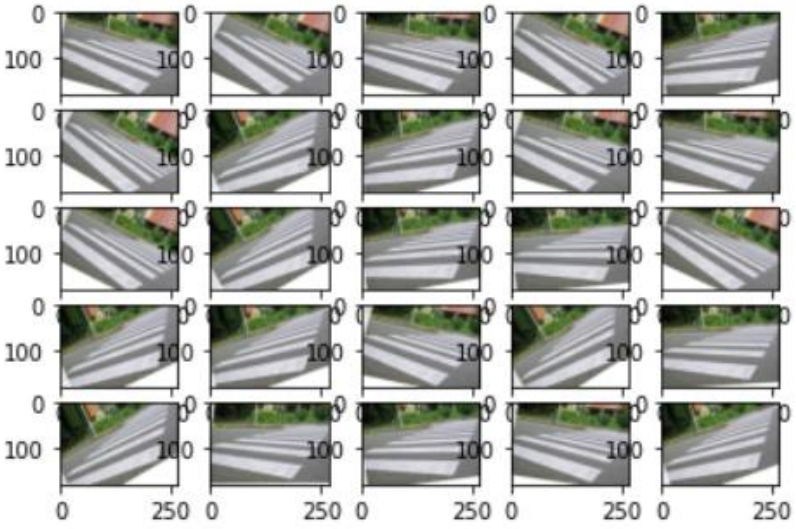

圖 4.3-2 資料增補 (圖 9)

4.4 物件辨識率測試

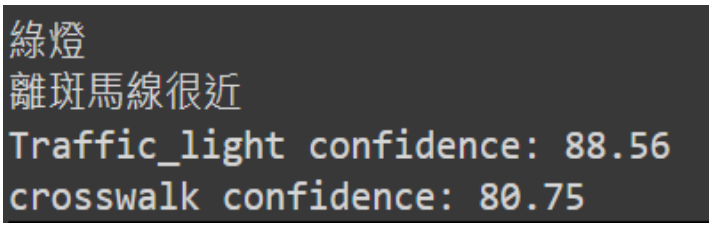

圖 4. 4-1 測試結果_1

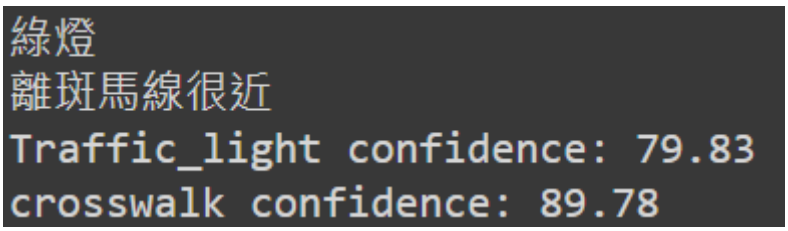

圖 4.4-3 測試結果_3

圖 4.4-4 測試結果_4

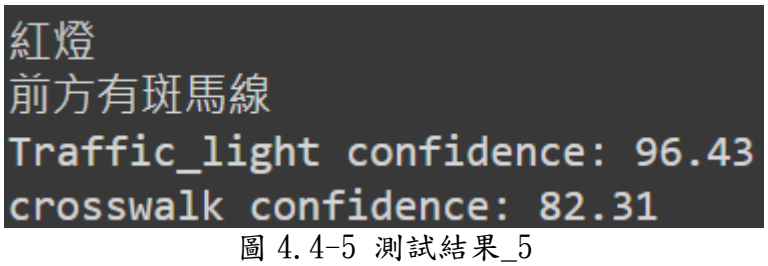

圖 4.4-5 測試結果_5

圖 4.4-6 測試結果_6

4.5 紅綠燈燈號判斷及誤判修正

1. 首先抓取紅綠燈的 bounding box

2. 並將它轉成 HSV 色彩空間，因為 HSV 色彩空間對於光罩的影響較小

3. 接著以紅色與綠色的遮罩套在影像上，並比較兩者的面積，藉此來判斷是紅燈抑或是綠燈燈號。

圖 4.5-1 偵測紅綠燈燈號

4. 影像有時間序列相互關聯的特性，所以我們將最近五張影像的燈號數據儲存在陣列中，做投票表決機制，若其中有少數的誤判，其他正確的燈號數據就會將其抑制

4.6 GOOGLE MAP API 導航[10]

1. 申請 Google Maps API Key，使用 Maps Javascript API，於網頁中嵌入地圖

2. 結合北市政府公開資料庫，找到人行道並給予更好的路線安排

   台北市政府公開資料平台裡的數據為 TWD97，而國際上常用的經緯度座標單位為 WGS84，並解要將地圖數據導入 GOOGLE MAP API 必須使用 WGS84，所以必須經過單位換算。

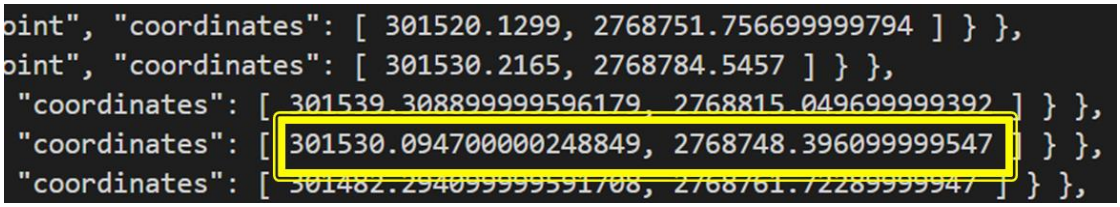

圖 4.6-1 TWD97

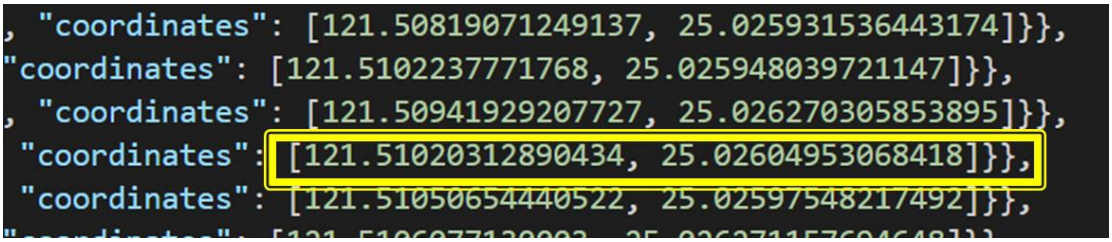

圖 4.6-2 WGS84

3. 規劃路徑，並將規劃出的路線於地圖上繪製出來，並將路徑細節回傳

```
// 繪製路線
directionsService.route(request, function (result, status) {
    if (status == 'OK') {
        // 回傳路線上每個步驟的細節
        console.log(result.routes[0].legs[0].steps);
        directionsDisplay.setDirections(result);
        path = result.routes[0].legs[0].steps;
        pubnub.publish({
            channel: "path",
            message: path
        });

    } else {
        console.log(status);
    }
});
```

圖 4.6-3 路徑細節回傳

4. 網頁顯示**現在路徑規劃**

圖 4.6-4 路線顯示

4.7 魚眼矯正[11]

1. 使用鏡頭拍攝棋盤格，以獲取魚眼矯正係數（DIM， K， D)
2. 然後就可以利用矯正係數進行相片矯正

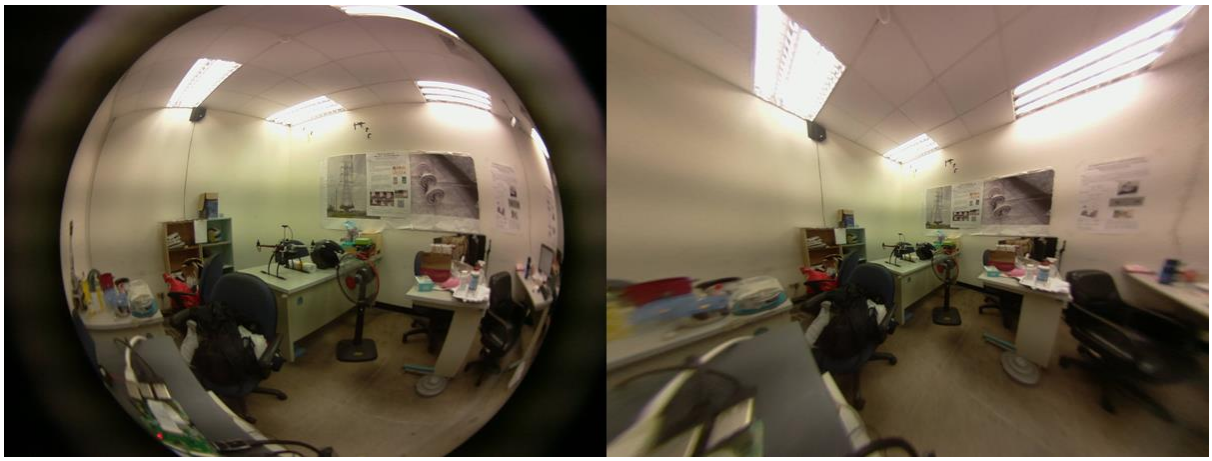

圖 4.7-1 相片矯正結果

4.8 語音輸出

在語音輸出的部份，我們使用開源的語音合成工具 ESPEAK，它能以共振峰合成的方法將文本轉為語音檔，並且支援多種語言。我們利用此工具將提醒與指令製成語音檔輸出給使用者。

第五章結論與未來方向

5.1 結論

根據本專題研究成果，我們能成功識別斑馬線位置及紅綠燈燈號，並能透過耳機告知視障者斑馬線的位置和距離以及是否適合通行。同時也具備 GPS 導航功能，能引導視障者到達欲前往的目的地。

對一般人而言，在外安全地行走是理所當然的事，但對視障者來說卻是艱鉅的任務，他們在行動上會面臨許多困難，道路及環境上的種種都可能形成障礙，很多視障者甚至會為此選擇足不出戶。對於一個視障者而言，他們很難享有出門走路、散散心的樂趣。

綜合以上，希望本專題的研究成果能多少帶給他們一些幫助。而往也期望能結合更多功能，替視障者打造出一個更友善、更平等的生活環境，使他們能願意踏出家門，探索不一樣的世界。

5.2 未來展望

(1)結合公共運輸工具

目前在台灣的公共運輸工具雖有提供無障礙乘車，但對視障者而言招車反而才是一大難題，本專題希望能結合大眾運輸系統，替視障者創造一個更美好便利的無障礙環境，在視障者需要搭乘大眾運輸，例如：公車、火車、捷運時，可以發送訊號通知司機或站務人員，使他們能有時間預做準備，以此提升視障者使用公共運輸的體驗感受。

(2)結合叫車系統

希望能與計程車、Uber、Line Taxi 等提供乘車服務的車隊結合，視障者需要乘車時，能傳送位置訊號給司機，使他們能準確地到達視障者所在的位置，給予他們更便利的乘車服務，使視障者出門在外也不用擔心沒有交通工具可以搭乘。

附錄(一) 斑馬線判斷

順序	判斷物	判斷結果	confidence
1	斑馬線	斑馬線	76.46951
2	斑馬線	斑馬線	84.82155
3	斑馬線	斑馬線	75.27881
4	斑馬線	斑馬線	76.85141
5	斑馬線	無	
6	斑馬線	斑馬線	77.78645
7	斑馬線	斑馬線	76.27221
8	斑馬線	斑馬線	82.95529
9	斑馬線	斑馬線	79.3733
10	斑馬線	斑馬線	84.80406
11	斑馬線	斑馬線	84.68918
12	斑馬線	斑馬線	80.116
13	斑馬線	斑馬線	75.54749
14	斑馬線	斑馬線	77.553
15	斑馬線	斑馬線	84.57353
16	斑馬線	斑馬線	81.69201
17	斑馬線	無	
18	斑馬線	斑馬線	81.53577
19	斑馬線	斑馬線	83.09509
20	斑馬線	斑馬線	81.09497
21	斑馬線	斑馬線	79.37453
22	斑馬線	斑馬線	82.02005
23	斑馬線	斑馬線	75.7313
24	斑馬線	斑馬線	81.05142
25	斑馬線	斑馬線	84.57709
26	斑馬線	斑馬線	76.39211
27	斑馬線	斑馬線	82.75648
28	斑馬線	斑馬線	83.31156
29	斑馬線	斑馬線	75.72139
30	斑馬線	斑馬線	84.49459
31	斑馬線	斑馬線	84.59793
32	斑馬線	斑馬線	83.85949
33	斑馬線	無	
34	斑馬線	斑馬線	76.04439
35	斑馬線	斑馬線	78.72296
36	斑馬線	斑馬線	78.96087
37	斑馬線	斑馬線	81.83162
38	斑馬線	斑馬線	75.56537
39	斑馬線	斑馬線	76.95493
40	斑馬線	斑馬線	81.20185
41	斑馬線	斑馬線	75.24483
42	斑馬線	斑馬線	82.94504

43	斑馬線	斑馬線	82.05474
44	斑馬線	斑馬線	84.35785
45	斑馬線	斑馬線	75.717
46	斑馬線	斑馬線	82.71851
47	斑馬線	斑馬線	77.00048
48	斑馬線	斑馬線	81.14054
49	斑馬線	斑馬線	81.76237
50	斑馬線	無	

附錄(二) 綠燈判斷

順序	判斷物	判斷結果	confidence
1	綠燈	綠燈	91.83437
2	綠燈	綠燈	82.78144
3	綠燈	綠燈	92.87401
4	綠燈	綠燈	89.53417
5	綠燈	綠燈	85.9151
6	綠燈	綠燈	93.84071
7	綠燈	綠燈	82.58158
8	綠燈	綠燈	80.62801
9	綠燈	綠燈	93.12891
10	綠燈	綠燈	93.59477
11	綠燈	綠燈	93.83557
12	綠燈	綠燈	83.41147
13	綠燈	綠燈	92.24269
14	綠燈	紅燈	67.5274
15	綠燈	綠燈	87.50121
16	綠燈	綠燈	80.62622
17	綠燈	綠燈	87.23888
18	綠燈	綠燈	93.34615
19	綠燈	綠燈	84.03301
20	綠燈	綠燈	86.36849
21	綠燈	綠燈	88.31363
22	綠燈	綠燈	89.27429
23	綠燈	綠燈	81.08853
24	綠燈	綠燈	82.71325
25	綠燈	綠燈	85.12964
26	綠燈	綠燈	83.79742
27	綠燈	綠燈	84.27235
28	綠燈	綠燈	94.78947
29	綠燈	綠燈	88.1379
30	綠燈	紅燈	76.2783
31	綠燈	綠燈	81.20291
32	綠燈	綠燈	83.8619
33	綠燈	綠燈	92.96297
34	綠燈	綠燈	93.17276
35	綠燈	綠燈	83.74083
36	綠燈	綠燈	82.50946
37	綠燈	綠燈	93.18049
38	綠燈	綠燈	80.28804
39	綠燈	綠燈	88.09501
40	綠燈	綠燈	88.25213
41	綠燈	綠燈	83.78579
42	綠燈	綠燈	83.96828

43	綠燈	綠燈	82.73578
44	綠燈	紅燈	56.5247
45	綠燈	綠燈	85.67287
46	綠燈	綠燈	85.16915
47	綠燈	綠燈	82.5635
48	綠燈	綠燈	94.18403
49	綠燈	綠燈	88.03294
50	綠燈	綠燈	82.90344

附錄(三) 紅燈判斷

順序	判斷物	判斷結果	confidence
1	紅燈	紅燈	92.33967
2	紅燈	紅燈	86.16254
3	紅燈	紅燈	84.22662
4	紅燈	紅燈	85.13027
5	紅燈	紅燈	89.43478
6	紅燈	紅燈	81.51519
7	紅燈	紅燈	84.35831
8	紅燈	紅燈	89.29727
9	紅燈	紅燈	89.84367
10	紅燈	紅燈	90.98659
11	紅燈	紅燈	87.13147
12	紅燈	紅燈	94.26629
13	紅燈	紅燈	81.24853
14	紅燈	紅燈	89.02311
15	紅燈	紅燈	81.36868
16	紅燈	紅燈	93.99107
17	紅燈	綠燈	67.5664
18	紅燈	紅燈	89.79409
19	紅燈	紅燈	90.54154
20	紅燈	紅燈	86.06272
21	紅燈	紅燈	92.94942
22	紅燈	紅燈	93.17588
23	紅燈	紅燈	89.82924
24	紅燈	紅燈	83.28457
25	紅燈	紅燈	87.80061
26	紅燈	紅燈	86.37111
27	紅燈	紅燈	86.1277
28	紅燈	紅燈	89.65066
29	紅燈	紅燈	87.442
30	紅燈	紅燈	86.68804
31	紅燈	綠燈	67.7498
32	紅燈	紅燈	93.04639
33	紅燈	紅燈	81.91875
34	紅燈	紅燈	86.08502
35	紅燈	紅燈	90.54552
36	紅燈	紅燈	83.46323
37	紅燈	紅燈	85.04403
38	紅燈	紅燈	88.18678
39	紅燈	紅燈	87.8722
40	紅燈	紅燈	80.66965
41	紅燈	紅燈	90.97755
42	紅燈	紅燈	91.95355

43	紅燈	紅燈	91.34444
44	紅燈	紅燈	94.08251
45	紅燈	紅燈	89.54949
46	紅燈	紅燈	93.17291
47	紅燈	紅燈	87.11269
48	紅燈	紅燈	83.55252
49	紅燈	綠燈	51.6723
50	紅燈	紅燈	90.57093

參考文獻

[1] 蘇怡帆， 黃國晏， & 畢恆達. (2012). 視障者在臺北市空間中的移動經驗. 特殊教育學報， *36*， 93–114. https：//bihspace.com/download/學術出版/視障者在臺北市空間中的移動經驗.pdf

[2] 周秉誼. (n.d.). *淺談 Deep Learning 原理及應用*. https：//www.cc.ntu.edu.tw/chinese/epaper/0038/20160920_3805.html

[3] Redmon， J. (n.d.). *YOLO： Real-Time Object Detection*. https：//pjreddie.com/darknet/yolo/

[4] *Getting Started with the Camera Module*. (n.d.). Raspberrypi.Org. https：//projects.raspberrypi.org/en/projects/getting-started-with-picamera

[5] *Rasberry Pi Foundation*. (n.d.). Raspberrypi.Org. https：//www.raspberrypi.org/

[6] *JETSON XAVIER NX*. https：//www.nvidia.com/zh-tw/autonomous-machines/embedded-systems/jetson-xavier-nx/

[7] *OpenCV*. (n.d.). Opencv.Org. https：//opencv.org/

[8] *PubNub*. (n.d.). Pubnub.Com. https：//www.pubnub.com/

[9] 李 謦伊. (2020). *Yolov4 筆記*. https：//medium.com/ching-i/tagged/yolov4

[10] *Maps JavaScript API*. (n.d.). Google Maps Platform. https：//developers.google.com/maps/documentation/javascript/overview

[11] Donkey_1993. (2020， January). *鱼眼摄像头的畸变矫正方法-Python+opencv*. https：//blog.csdn.net/donkey_1993/article/details/103909811