淡江大學學校財團法人淡江大學
資訊工程學系

# 行人異常行動軌跡偵測

專題成果報告書

**組員**

資工三A 410410319 陳既為
資工三A 410410822 黃才熏
資工三A 410410947 魏劭安
資工三A 410411606 莫子誼
資工三A 410410731 戴浚庭

# 目錄

# 動機與目的

## 研究背景

近期的世界日益動盪，面對不斷升級的地緣政治緊張局勢、社會動亂，以及 COVID-19 大流行的持續後果，安全監視系統的必要性達到了前所未有的高度。地緣政治以及社會動亂的影響深遠，不少人因此受傷甚至喪命；而疫情大流行重塑了公共衛生的動態，改變了全球的運作和互動。這些事件引入了新的焦點，即增強對公共環境的保護：不論是避免隨機攻擊的發生，或是增加社交距離以防止疫情傳播等。

而在當今的社會當中，人們可以輕易地取得攝影機、電腦，以及一些免費但強力的人工智慧(Artificial Intelligence)工具，如 YOLO。透過攝影機，我們可以對其他地方的動態有所掌握；透過電腦，我們可以得到強大的運算能力來執行任務；透過人工智慧工具，我們可以輕易地對取得的畫面進行分割處理。透過這些工具，我們可以建造出一個系統來檢視畫面中誰的移動模式是異常的。通過尖端算法和數據分析，這程式系統能夠識別出偏離既定模式的移動軌跡。無論是移動速度特別快或慢或逗留的個體；還是蛇行移動，無法掌握的個體。

我們希望這樣的軟體應用程式能夠幫助許多地方脫離隨機攻擊事件或是其他行動詭異之人的危險。我們期待能夠促進一個更安全、更有韌性的世界，並致力於在日益不確定的時代中保護公共和平與衛生。

## 研究動機

如上所述，我們非常的重視公共安全及衛生，但是在網上查詢相關專案時，卻發現相關技術的開發並不如預期的多。所以，我們想藉此機會來對此一方向進行比較深入的研究，在回饋社會的同時，也讓我們學習到如何自行開發軟體專案。從流程設計開始，一步一步的推進到資料分析，再

直到實際應用層面。其中，我們也能學習到許多專案開發時常會面臨到的問題，如與教授溝通、與同儕溝通、使用現成的人工智慧模型等。

### 研究目的

本研究論文的目的是開發一結合人工智慧的軟體，以更好的進行畫面中人體的辨識，並且這套軟體應用程式具有實用性，可以利用畫面中每個人的相對資料順利的辨識出哪些人速度過快、過慢、停滯，或是行動模式超出預期範圍。

## 相關研究

### 物件偵測（object detection）

[1][2]背景分割法(Background Subtraction Methods)是一個計算背景及前景併分割兩者的演算法。該演算法能夠捕捉那些不會移動的地方作為背景，並分割出與其不同的地方作為前景。此演算法可以動態的調整背景，也就是說它可以適應不同的變化，包括不同的光線變化、鏡頭角度變化等，以與時俱進的捕捉前景物品。

YOLO(You Only Look Once)[3]是一個熱門的電腦視覺(Computer Vision)演算法。YOLO 是一個卷積神經網路(convolutional neural network)人工智慧的產物，與其他電腦視覺人工智慧演算法的不同在於: YOLO 將電腦視覺視為一個回歸問題。其物件偵測原理是將一張圖片劃分成數個小格，再讓每個格子生成出對於對某件物品的預測的信心指數。值得注意的是，YOLOv3 及其之後的 YOLO 模型結合了殘差神經網絡（Residual Neural Network，簡稱 ResNet）之技術以更好的對更大的人工智慧模型進行訓練。

YOLOv8 [4]是最新的 YOLO 系列模型(在該專題進行時)，由 Ultralytics 製作。其簡單的使用者介面使其非常熱門。使用者也可以再在其之上進行微調。YOLOv8 是基於前幾代 YOLO 之上的成品，其功能包刮物件偵測(object detection)、物界分割(object segmentation)、物件追蹤(object tracking)、物件分類(object classification)，以及人體姿勢預估(pose estimation)[5]。由於 YOLOv8 涵蓋了諸多電腦視覺方面的應用，非常的靈活，YOLOv8 非常的熱門。

## 物件追蹤(object tracking)

ByteTrack[]演算法是基於先檢測後追蹤(tracking-by detection)方式的追蹤方法。他跟以往追蹤算法的最大區別是: ByteTrack 並非單純去掉低分結果，而是利用檢測框和追蹤軌跡之間的相似性。在保留高分檢測的同時，也從低分檢測中去除背景，挖掘出真正的物體，進而降低漏檢並提高軌跡的連貫性。

[ ] BoT-SORT: Robust Associations Multi-Pedestrian Tracking 概述: 多目標追蹤(Multiple Object tracking)旨在影片檢測和估計多個物體的時空軌跡。應用如自動駕駛、影片監控。追蹤主要由兩步驟組成：1. 運動模型和狀態估計，用於預測下一幀中軌跡框的位置。2. 將新的幀檢測與當前軌跡集進行關聯關聯任務的解決方式: a. 物體的定位，主要是預測的軌跡框和檢測框之間的交集-聯合（Intersection over Union, IoU）b. 物體的外觀模型，並解決重新識別（ReID）。本文提出了基於多目標追蹤的基於檢測追蹤方法的三個主要修近和改進。一、Kalman Filter：通過實驗作者發現直接估算邊界框的寬度和高度可以獲得更好的性能。改變 Kalman Filter 的狀態向量。二、Camera Motion Compensation (CMC)：使用 OpenCV 的 Global Motion Compensation 技術來表示背景運動。三、IoU - Re-ID Fusion：利用局於 IoU 和 ReID 的餘弦距離融合方式，以提高檢測和軌跡之間的魯邦性(Robustness)。

## 機器學習（machine learning）

監督式學習(Supervised Learning)，又稱分類法(Classification)，是一種機器學習(machine learning)的方式。此方法係用有標記的(annotated)資料投餵給機器，使機器自行找到所給的資料與備標註之資料的關聯性。監督式學習通常有著較好的成效，但由於實際上大多數的資料皆為未標註的資料，因此實務上還得考慮自己將標註資料的成本。

非監督式學習(Unsupervised Learning)，又稱分群法(Clustering)，也是一種機器學習的方式。此方法主要用來配合未被標註之資料。儘管成效可能不如監督式學習，但由於現存大多數資料皆為未被標註之資料，因此此方法還是備受歡迎。在這次的計畫中，若要使用監督式學習，那麼我們須將自行錄製的影片進行標註，考慮到若要對每一幀進行標註的龐大工程量，我們將採取非監督式學習。

scikit-learn 是一個開源的(open source)機器學習函數庫，內有多種機器學習函數可供選擇，包含回歸(regression)、分類(classification)、分群(clustering)方法等。scikit-learn 的好處在於其簡單的使用方式以及其以整合的統一 API (Application Programming Interface) 。要留意的是，scikit-learn 並不包含深度學習(deep learning)之方法。但考慮到其簡便的 API 得以讓我們嘗試多種不同的超參數(hyper parameters)已獲得最佳成效，我們預計將在這次的專題中使用 scikit-learn 作為機器學習的方法來源。

## 衡量標準(model evaluation metrics)

Accuracy 是一個

Precision

Recall

# 內容描述

## 預期成果

預期中，此次項目應實現以下項目：

1. 順利分辨出每個人體，並追蹤之。
2. 能夠運用大數據資料進行人工智慧模型的訓練，並根據此訓練好的模型來進行異常判斷。
3. 能夠處理預錄好的影像和及時捕捉到的影像。
4. 能夠以易懂、簡潔的方式讓使用者能一眼就明白畫面想表達的意思。

## 著作權聲明

此計畫在執行過程中我們使用了來自 pexels.com 的監視器影片素材。pexels.com 是一個影片樸片素材庫，裡面的素材皆為無著作權。不須特別申明也可使用。但出於對原作者的尊重，我們還是在這邊提及出處。此影片詳細網址為 https://www.pexels.com/video/black-and-white-video-of-people-853889/。此影片的相關截圖也會在本報告書中出現，並以“Black and White Video of People”稱呼。此影片之示意如下圖所示。

## 理論探討/使用技術

問題一：人體辨識與追蹤

首先，兩個問題油然而生：要如何辨別畫面中的物品？以及：算辨識出了物體，要如何知道該物體是否為人？ 針對這個問題，起初的我們並沒有任何概念，於是我們與教授討論後得到了以下方法。

方法一：背景差分法 Background Subtraction（失敗）

計畫的一開始，由於對於這方面的了解太過匱乏，我們並不清楚有哪些可用的軟體可以讓我們直接使用。因此，在經過與教授的討論後，我們著手使用 OpenCV 中的 createBackgroundSubtractorMOG2 來進行作業。createBackgroundSubtractorMOG2 能夠分辨影片中不

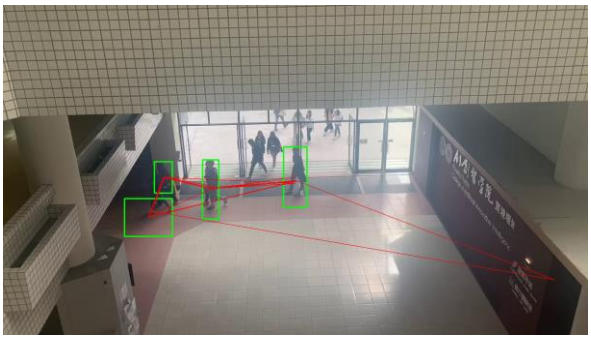
對照組一：實際畫面

對照組一：前後分差後之畫面

會動的部分作為背景，以分割出那些移動的前景。透過前後景的不同差來判斷人體（背景差分法）[1]。此方法有幾項缺點：第一，沒有移動的物體會被認為是背景而不是前景，因此無法被分割出來，簡單來說，此方法只能對在移動的物體有所反應。第二，由於只是簡單的對畫面進行相減以裁取出在移動的物體，該方法無法分辨該物體是人還是物，我們後來想到以物體的長寬做限制(因為正常人站立時高會大於寬)，但此方法對於新環境的適應力很差，因為只要攝影機角度一改或是遠近一變，所有參數皆會失去原本的意義。第三，此方法只能夠分割出在移動的物體，無法追蹤之。追蹤方法需額外

撰寫，原本計畫是根據個人的走路向量、以及每個人在每幀的存在理論上都應該要有重疊等因子進行是否是同一人的判斷。並也有進行到這步驟。但後期由於工程過於浩大，加上後來發現有較新的其他方法，因而沒有完成此步驟。

兩圖比較後可以發現，再對照組一的實際畫面中的四個綠色框中，只有三個人是被正確的辨識出來的，另一個是影子。而其他人，如左下兩人則因為在畫面中所占的面積太小，未到達門檻值而不被認為是人類。



對照組二: 實際畫面

對照組二 : 前後分差後之畫面

在第二對照組中可以發現的第四個問題是: 此方法對於鏡頭的晃動非常的敏感。鏡頭只要一晃動，使大量面積產生前後景不相等的地方，就會導致機器誤判。此缺點為原本沒預料到的，且此缺點非常致命，因為只要架設鏡頭的地基有些許震動，如大量人潮經過；或是在開放空間風比較大的地方，此人體辨別功能將直接作廢。

方法二: YOLO 物件辨識、追蹤（成功）

後來發現的新方法便是使用 YOLO，使用 YOLO 有幾項優點: 第一，只要看起來是人型，或是說只要電腦可以看出該物體是人型的，那麼這物體就可以被辨識出來，無須擔心是否有人是靜止不動的而無法被辨識出來。這點尤其重要，因為我們在考慮異常行為時也需考慮到那些停滯的人。第二，由於只有人形物體會被標註，因此不必再額外設條件，如常寬比等條件等的去判斷被標註之物體是否為人類。第三，YOLO 有已經設計好的 API (Application Programming

Interface) 可以直接帶入追蹤的函數(function)使用，非常容易理解與操作。

從下圖可以看到，使用 YOLO 進行人體偵測有著很高的成功率，並且可以很簡單地進行物件追蹤，也可以很簡單的取得這些人的移動紀錄座標，以標示劃出移動軌跡(綠色線)。


圖一: YOLO 人體偵測及追蹤成效

問題二: 移動軌跡歸類(正常與不正常)

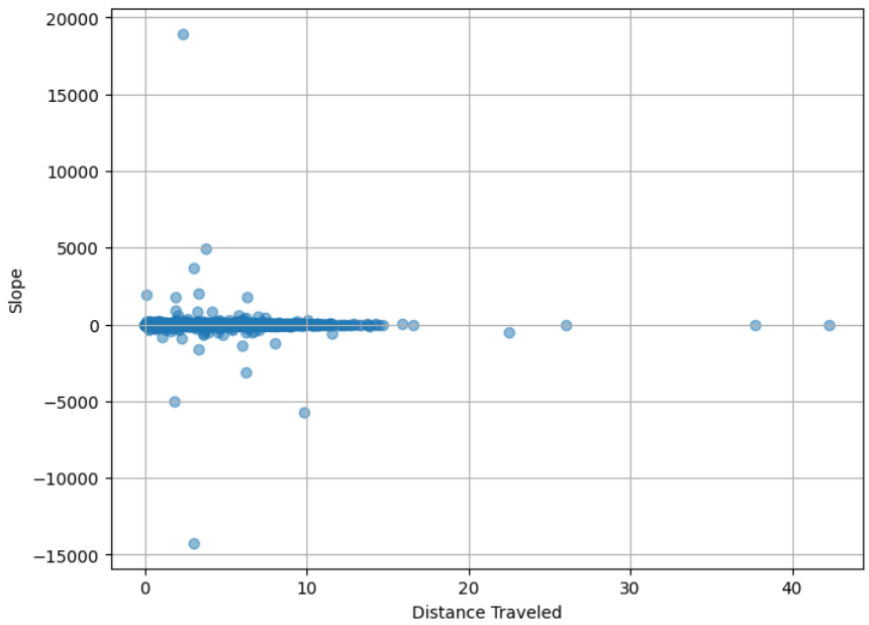
透過 YOLO 的物件偵測，我們成功的取得了畫面中人們的移動歷史紀錄(座標)。下一步的動作便是開始分析這些人的移動座標並判斷誰應該被判為正常，又誰應該被判為異常。我們所想到的特徵(features)有: 速度以及方向。速度可判斷出過快(FAST)、過慢(SLOW)、停滯(STOP)；而方向(DIRECTION)則可以判斷哪些人的移動軌跡無法預測、脫離正常理解。值得注意的是，這些判斷的基準皆沒有一個統一標準，皆為我們用肉眼以及常識去判斷出哪些為異常哪些為正常。

方法一: 機器學習(失敗)

計畫的一開始，我們的構想是使用 scikit-learn 所提供的分群(clustering)工具來整理、歸納數據。會選擇使用分群法而非分類法(classification)的原因是因為我們並沒有辦法找到相關的資料集是有包含被標記(annotated)的行人之資料，而我們也沒有時間與

足夠的資料去做出此資料集，更無法保證此訓練資料集是足夠應付所有場景的。

在選定要使用分群法之後，我們便著手進行數據前處理，即透過所得的座標計算出移動距離以及斜率。其中，移動距離可以推斷出速度，而斜率代表移動方向。下圖為一短影片[ ]之示意圖與該影片中人們移動的特徵座標圖。縱軸為斜率，橫軸為移動距離，每一點皆為某人在某一幀之資料。要注意的是，由於目前先取得一整個影片的資料點再進行影片編輯，因此此方法無法配合即時影像，只能給一個完整的預錄影像。



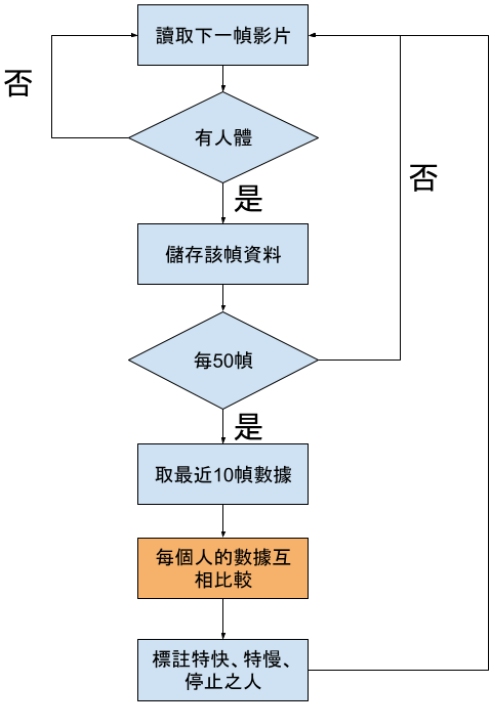
從座標圖可以明顯看到有幾個顯眼的離群值，理論上他們應該被標示為異常。然而實際上，這些異常值僅為數據上的異常，由於這些資料點皆僅為某一幀所記錄到的數據，因此實際上以一整個影片來看的話，那些數據上異常的人可能為正常。舉例來說，以縱軸斜率來看，某人因一個瞬間的晃動(可能是偵測誤差或是轉彎時)而導致此人的移動斜率以垂直或接近垂直的角度移動，造成此人的斜率接近無限大，造成明顯離群，但我們不能就因此判斷此人是異常的。意即，我們不能只透過某一幀來判斷整體，我們必須想辦法把資料點串起來，而非一幀一幀判斷。然而這將引入下一個問題: 要怎麼把每幀的資料串成一序列? 而很不幸的是，我們並沒有找到此演算法。於是我們嘗試下一個方法，即從原本一次餵入一整個影片改成每十幀都餵入一次這十幀的平均值。此方法可以處理即時影像，但下一個問題馬上反映了出來: 此方法每次投餵都會有不同的人被標為異常。由於每一幀都餵入一次資料，每一次都或多或少有不同的人被視為異常，導致整個影片非常混亂，沒有一個統一性。到這裡，我們只剩兩個月可以繼續做此專題，於是我們隨即嘗試下一辦法。

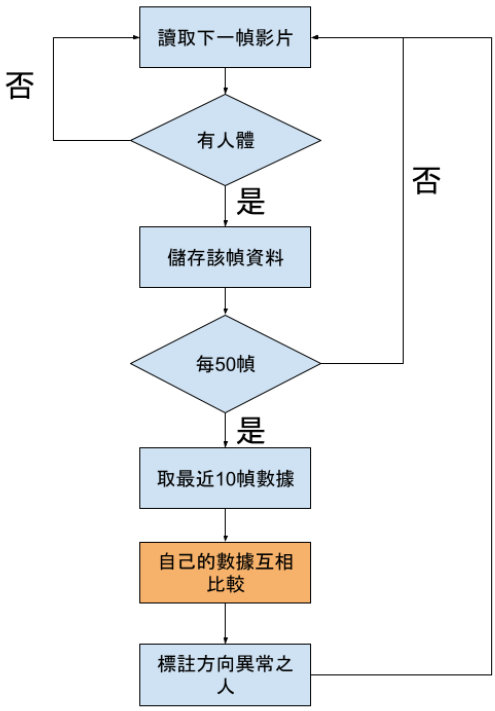
方法二: 平均值與標準差(成功)

由於時間的緊迫以及機器學習的成效不盡如人意，我們採取比較原始的方法: 計算平均值與標準差。由於這方法是我們做後採取的方法，我會較為深入的解釋其中的原理。

開始之前先介紹整個程式的架構: 此程式以幀為時間單位記錄每一幀的行人移動資料，之後每 50 幀就處理一次資料，計算出每個人的移動距離等必要數據，爾後將每個人的移動距離互相對比，判斷出誰是太快(“FAST”)、太慢(“SLOW”)，或是停滯(“STOP”)。雖然是每 50 幀處理一次資料，但我們只會取其中的最後 10 幀。因為如果將這 50 幀的資料全部納入考量，那麼此資料會受到較久之前的數據的影響。而如果幀數太少則是會導致前面所描述的問題，使異常判斷與我們肉眼所觀察到的事實不吻合。

異常方向(“DIRECTION”)的判斷方法則稍微不太一樣。不同之處在於異常方向判斷是跟自己比較，而非跟其他人交叉比對。此程式會預測一個人的移動方向，並對超出預期移動方向的人標註“DIRECTION”。



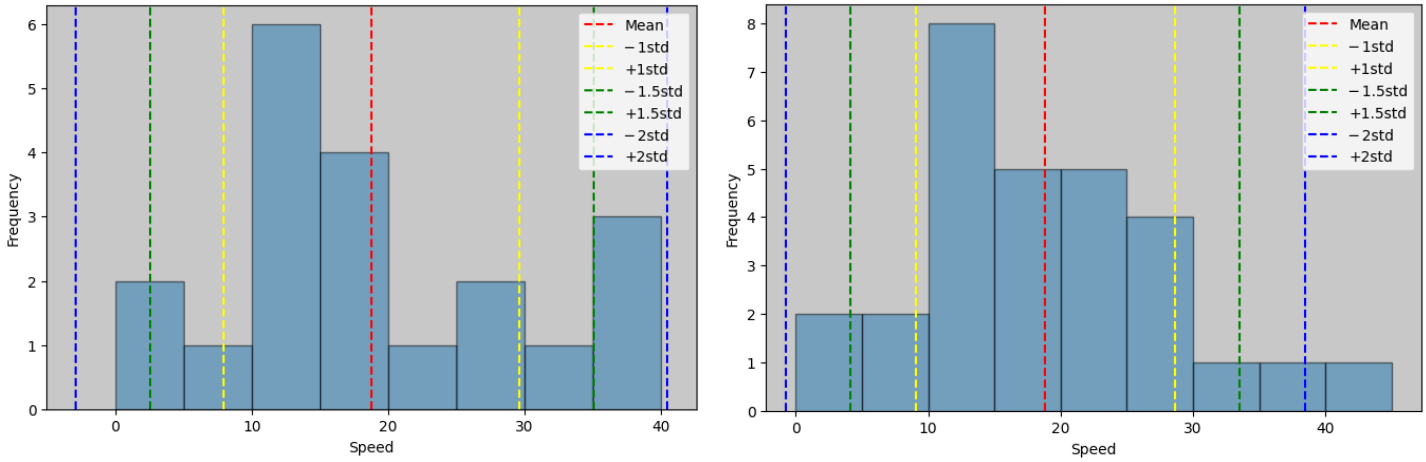




左圖為判斷哪些人速度為異常的流程圖，包含過快、過慢、停滯。右圖為判斷哪些人方向為異常的流程圖。橘色部分為邏輯不同的地方，速度異常的判斷邏輯是每個人的數據交叉互相比對，而方向異常的判斷邏輯是單個人的數據與自己比較。

接下來我們將逐個解釋我們如何判斷過快、過慢、停滯以及方向異常的。

快慢判斷:

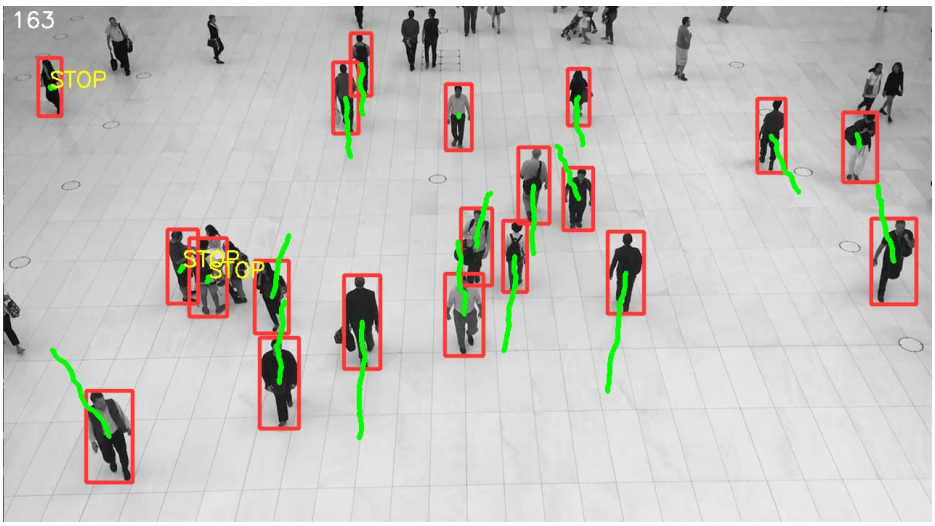
此方法非常的純粹，即計算每同一時間段(10 幀)中每個人的平均移動距離，再根據此每個人的平均值再算出平均值及標準差。之後，平均移動距離大於平均值加 1.5 個標準差的人將被視為太快並標示“FAST”；而速度小於平均值減 1.5 個標準差的人將被視為太慢並標示 “SLOW”。至於為什麼我們是以 1.5 個標準差為基準的原因，則是經過我們將資料繪製出來後的判斷以及多次測試後的結果。將每個人的平均移動距離繪製出來的直方圖如下。




這兩張圖分別為[]影片中地 41 至 50 幀及第 91 至 100 幀中每個人之移動平均速度。圖中，藍色直方為每個人移動移動距離的平均值分布；紅色虛線為每個人移動移動距離的平均值；綠色虛線為平均值加減 1.5 個標準差的值；藍色虛線為平均值加減 2 個標準差的值。從兩張圖可以看到當取 2 個標準差時，其範圍將超過大多數的值，甚至在左邊的圖中更是沒有任何的人在兩個標準差的範圍之外。然而這與我們肉眼直接觀看影片時所觀察到的現象背道而馳。意即，2 個標準差的要求太過於嚴格了，導致該被視為太快/太慢的人沒有辦法被偵測出來。而平均值加減 1 個標準差的範圍(綠虛線)又太過狹窄，導致過多的人被標“FAST”或“SLOW”。綜合評估下來，我們認為平均值加減 1.5 個標準差是最為合適的。

停滯判斷:

在做停滯判斷的過程中我們有注意到一個現象: 那些我們認知中是停滯的人實際上其實也是有輕微的晃動，例如身體擺動。因此，我們不能夠把沒移動的基準設為完全沒有移動，而是要允許一個較小的範圍容忍這些晃動或是誤差。下一個問題接踵而至: 這個誤差值要是多少呢? 很顯然的這不能是一個定值，否則鏡頭遠近一變或者角度一改就會使這值失去意義。於是我們決定使用即時獲得的資料來定義此數值。首先一樣的，先取得每個人平均移動距離的平均值，這個值讓我們知道了這畫面中大多數的人的移動移動距離，再將這數值除以 45，即得到這個可容忍晃動範圍的數值。至於 45 這個數值也是經過我們多次實驗後的選擇。更大的數值導致該被標示“STOP” 的人沒被標示；更小的數值則導致移動較慢的人被標記“STOP”。



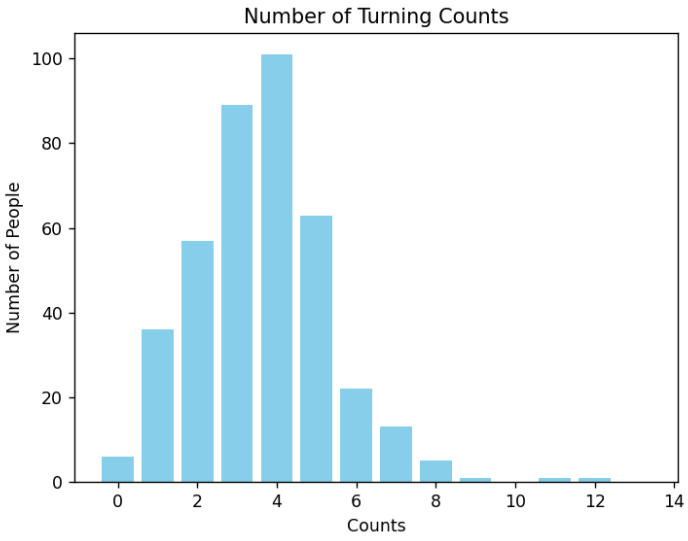

綠色線條為每個人的移動軌跡，我們可以注意到左上角被標記“STOP”的人的綠色線近乎為一點，因此此人被標記“STOP”。值得注意的是雖然此綠色線近乎為一點，但也並非真的為一點，所以如果把判斷基準設為完全靜止不動的話則無法偵測到此人是停滯的。

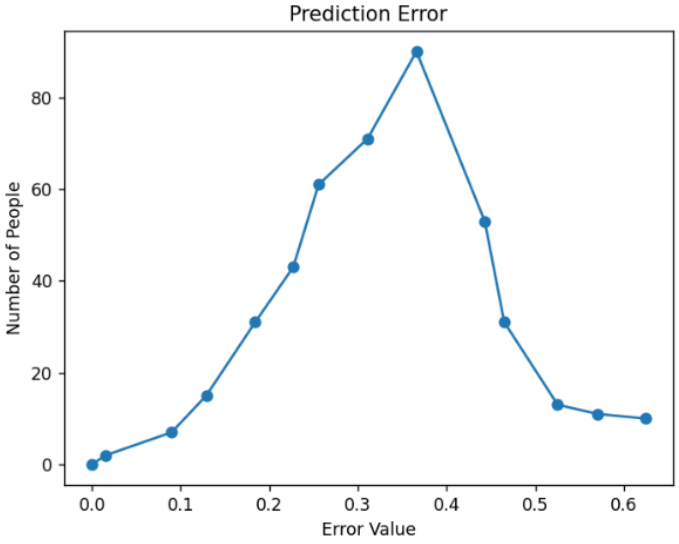
異常方向判斷:

在討論何為方向異常的時候，我們有討論了我們認為異常的目標，我們希望抓出的是蛇行或者旋轉亂繞，這種行為在路徑上都有一個

共同點-不斷的轉彎，我們給每個人 5 次的轉換方向機會，5 次是由多部影片實際跑出來後決定，大多數人離開畫面前為 3 至 4 次。

對於方向異常的判斷方式，我們利用 10 幀的路徑點，由最舊的 1 幀跟 5 幀形成一個向量，再加在第 6 幀上形成預測的第 10 幀的位置，最後再跟實際的第 10 幀去做比較。我們通過多部影片去計算了大部分行人跟預測落點的落差距離，大約都落在 0.37 個向量長度，於是我們決定將我們接受的實際誤差設定為 0.4 個向量長度，程式對人的走向預測誤差對人數分布圖如下面左圖所示。只要超過 0.4 個向量長度，就將轉向次數加 1，超過 5 次則在畫面上將此人標記 “DIRECTION”以示此人方向異常，轉彎次數對人數分布圖如下面右圖所示。並且考慮到 STOP 的情況目標可能會在原地打轉的情況，所以只要判定為 STOP 就不會輸出方向問題，並且將轉向次數歸 0，避免目標開始移動因為轉向次數超過 5 而直接顯示方向異常。



# 結論

在這次的計畫中，我們學習到了很多程式設計時會遇到的實務面問題，例如如何進行版本控制、與教授與組員溝通、自己嘗試進行資料處理。而成品方面，我們做出了一個可以判斷預錄好的影像，或是即時攝影機捕捉到的畫面中哪些人行動軌跡是異常的程式。其中的異常狀態有 4 種，包括過快、太快(“FAST”)、太慢(“SLOW”)、停滯(“STOP”)，以及方向異常(“DIRECTION”)。其準確率(Accuracy)分別為 0.917、0.941、0.784，以及 0.829；詳細描述如下所示。

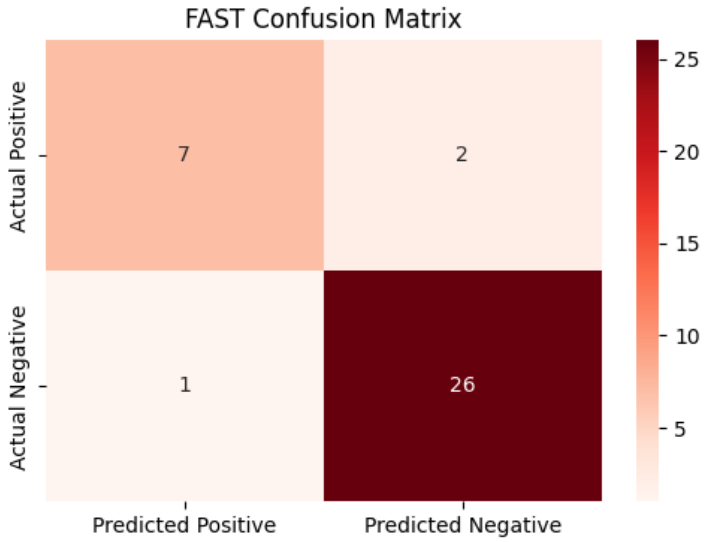
## 過快(FAST)

Accuracy = 0.9166666666666666

Precision = 0.875

Recall = 0.7777777777777778

F1_Score = 0.823529411764706



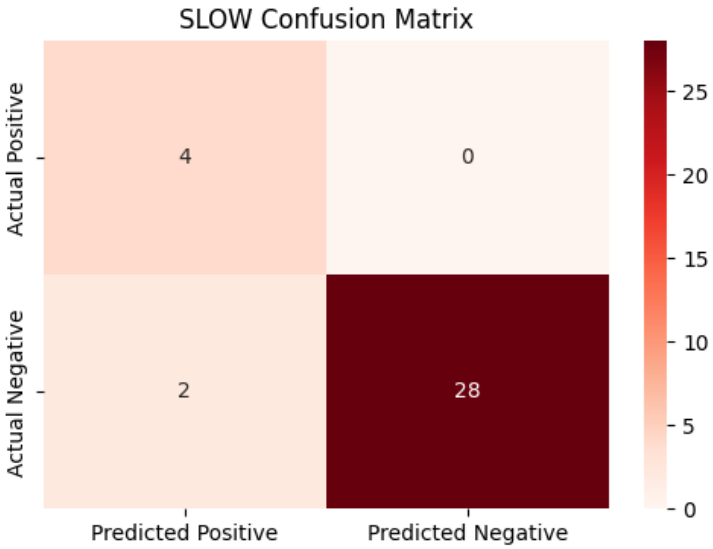

## 過慢(SLOW)

Accuracy = 0.9411764705882353

Precision = 0.6666666666666666

Recall = 1.0

F1_Score = 0.8



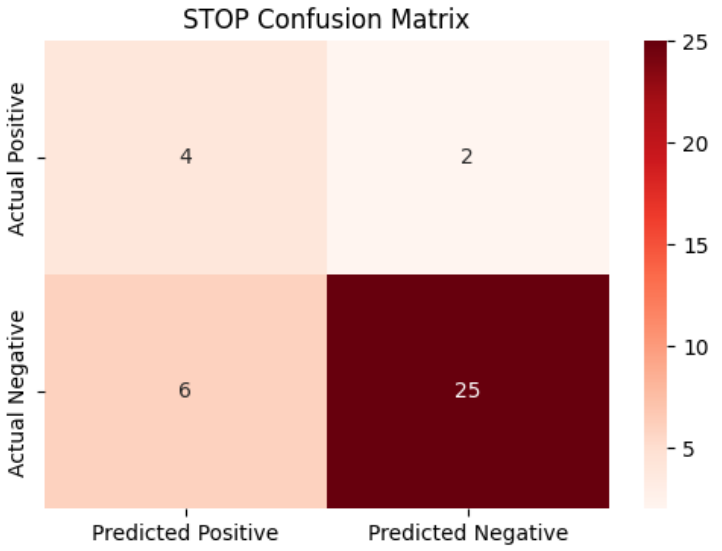
## 停滯 (STOP)

Accuracy = 0.7837837837837838

Precision = 0.4

Recall = 0.6666666666666666

F1_Score = 0.5



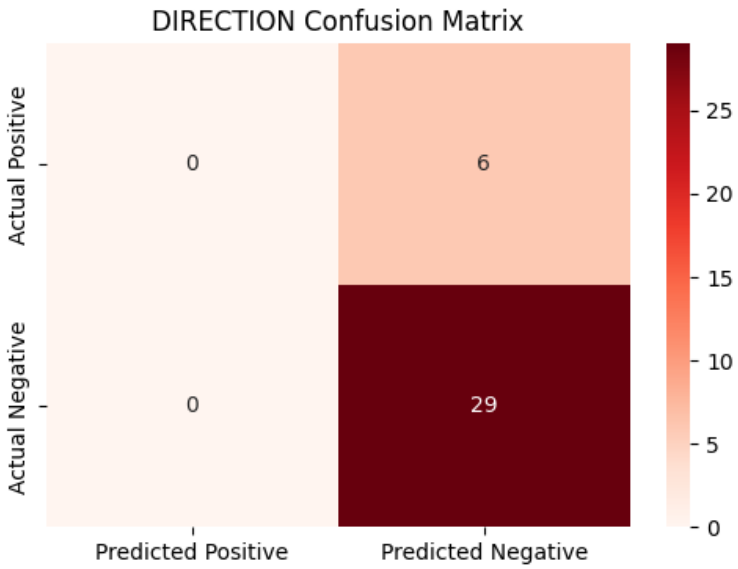

## 方向異常(DIRECTION)

Accuracy = 0.8285714285714286

Precision = NA (divided by zero)

Recall = 0.0

F1_Score = NA (Precision not defined)




從圖表中可以看到我們的程式對過快(FAST)以及過慢(SLOW)有著較高的準確度，其次是停滯(STOP)。而方向異常的部分則因為

# 參考資料

[1] OpenCV. (n.d.). Background subtraction. Retrieved from https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html

[] https://medium.com/@muhammadsabih56/background-subtraction-in-computer-vision-402ddc79cb1b
[]
[]