專題成果報告
主題：結合多人手指骨架偵測與手勢辨認技術的互動式多媒體應用之研究

組員：石顥澐、林語潔、黃霈昕

指導教授：余執彰

(一) 摘要

傳統展場缺乏技術時，資訊時常只能藉由傳統圖文方式傳遞。雖然近年來有許多互動式資訊設備來輔助讓展場資訊有更便利的應用，但這些設備都有些限制，例如可操作範圍、環境光線等等。本研究旨在解決上述問題，藉由手勢辨識技術實做出一款輕量化、便攜性、可互動之多媒體跑步小遊戲，進而加深觀眾對主題之印象和理解，並結合時事疫情，避免與裝置近距離接觸，以達安全且有趣之體驗。

過往已有成熟的手部辨識技術，然而手部辨識可能會遇到多人手部同時被鏡頭偵測到時，會影響結果的準確性，因此針對上述所提到手部辨識及多個物件偵測之問題，我們也設計了對應之方法。首先取得兩隻手指關節的節點資訊，透過各節點得出每秒步數並估算角色跑步速度；同時判斷互動過程中的不同手勢，對應不同手勢使角色產生不同的動作，最後再加上追蹤演算法，提升多人遊戲精準度，讓使用者能有良好的互動式體驗。

(二)研究動機與研究問題

在科技產品融入生活後，「人機互動」一詞出現，人們所追求的不再只是透過機器解決問題，更希望能與機器有所互動。現今，各大展場以及博物館中，結合主題衍生出的互動設計已成了展覽不可或缺的一環，而手部辨識為其中一種技術。結合手部辨識與多媒體，將已成熟的手部辨識技術加以運用區分手勢並使多媒體介面產生對應變化，開發出人機互動更多的可能性為本研究的最大目標。本研究中，欲探討如何分別運用兩種手勢辨識方法再結合多媒體，主要研究問題如下:

一、 如何運用擷取出的多人手部骨架資訊判斷為 Finger Race 中不同玩家的跑步行為。
二、 如何藉由影像特徵透過 CNN 模型進行手勢辨識改變多媒體介面角色之行為動作。
三、 如何結合判斷出的手勢產生出對應的多媒體介面來達到人機互動。

(三)研究方法及步驟

本研究成果如圖 1，為加入其他更有趣的互動，實現遊玩體驗的多元性，我們將研究方法再細分成四個部分進行，而簡易流程如圖 2。

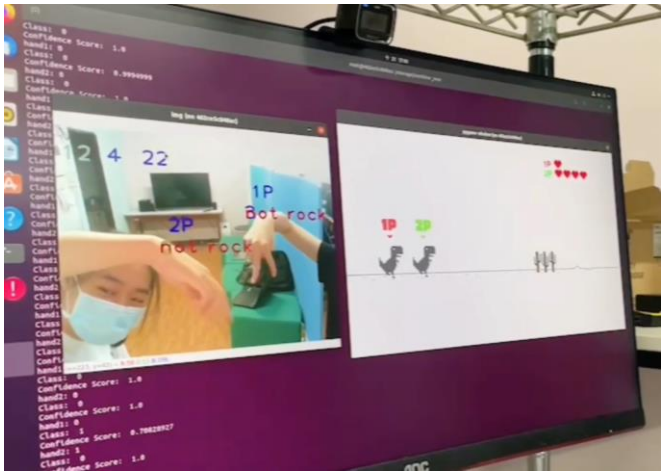

圖 1 Party Game 實拍

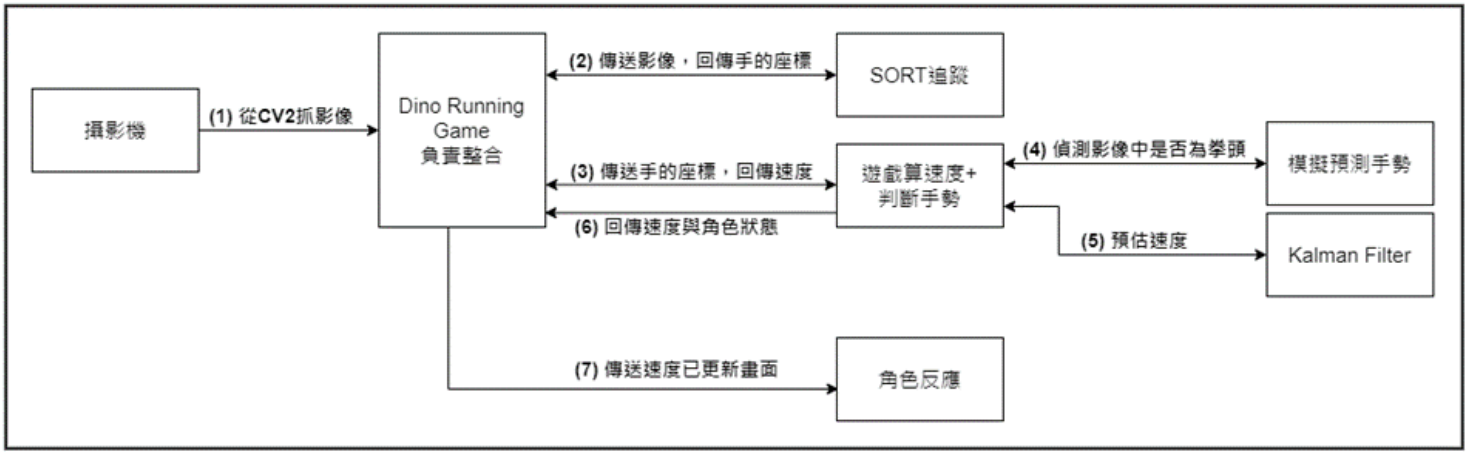

圖 2 研究方法之簡易示意圖

1. 骨架

本研究使用 MediaPipe[1]工具庫所提供的 Hand Landmarks 進行手部偵測與定位，在偵測成功後，此模型會回傳手部的各項資訊，包含手部的 21 個關節點座標(見圖 3)，透過此座標系統，進行初步的手勢辨識，並計算手指的移動速度。

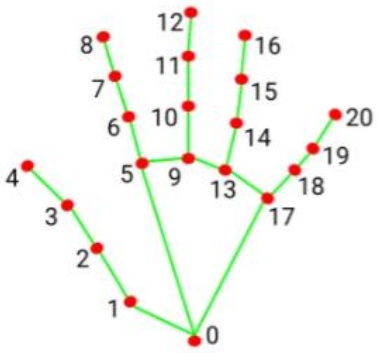

圖 3 MediaPipe Hand Landmarks

(1) 使用骨架辨識手掌朝下之初始狀態

取得手指座標後，就能開始處理跑步狀態辨識的問題。首先，為了能讓手能夠呈現如圖 4 的狀態以開始遊戲，透過獲取食指和中指指尖及其掌骨的座標（參照圖 3 中 5, 8, 9, 12)，來判斷兩隻手指是否為直立朝下的狀態。

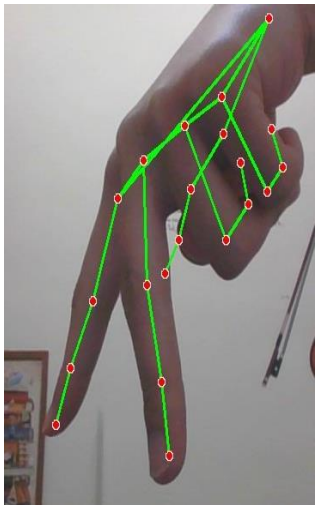

圖 4 手勢初始狀態

(2) 計算步數與指尖移動速度

如果兩隻手呈直立朝下時，接著即可透過計算兩隻手指交錯的次數來計算出一共走了幾步。具體方法我們利用食指指尖的 X 座標減去中指指尖的 X 座標來計算兩隻手指之間的距離（公式 1）此時算出來的值將因每次兩指交錯而出現正負值，此時只要再偵測是否有正負號變化即可計算出步數。公式 1 中變數的定義請參閱圖 3。接著透過每秒紀錄當前累積

之步數除上時間即算出每秒之速度，最後再做正規化就能得出指尖移動速度（公式 2）。

指間距離 = （INDEX_FINGER_TIP - MIDDLE_FINGER_TIP）

(1)

移動速度=當前速度*a+((當前累積步數 - 前一秒累積步數)/time)*b

(2)

其中公式 2 的 a=0.8, b=0.2, time = 1s。

2. 卷積神經網路 (Convolutional Neural Network, CNN) [2]

在幾輪的反覆實驗和測試後發現，透過骨架節點進行手勢辨識時，一些較複雜的動作或角度將沒辦法很好的識別。為解決此問題，本研究訓練用來偵測特定手勢的 CNN 模型，以利於對這些特定動作進行偵測。

(1) 資料處理

訓練資料主要分成兩種類，一為握拳之手勢影像(見圖 5)，另一則為非握拳外之各種手勢影像(見圖 6)，總計共約 15000 張影像。在訓練之前，因遊戲進行中，傳進模型的圖片尺寸不固定，所以在資料蒐集階段為確保模型再縮放圖片後也能正確辨識，蒐集的影像也採不固定尺寸。

圖 5 拳頭資料集

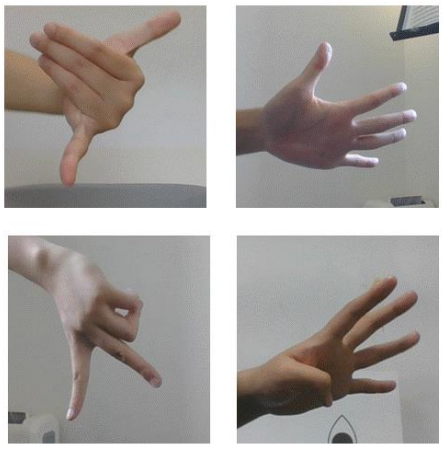

圖 6 非拳頭資料集

另外，我們也使用了 Data Augmentation[3]中的水平翻轉(Horizontal Flip)的手法來增加資料集(見圖 7)。

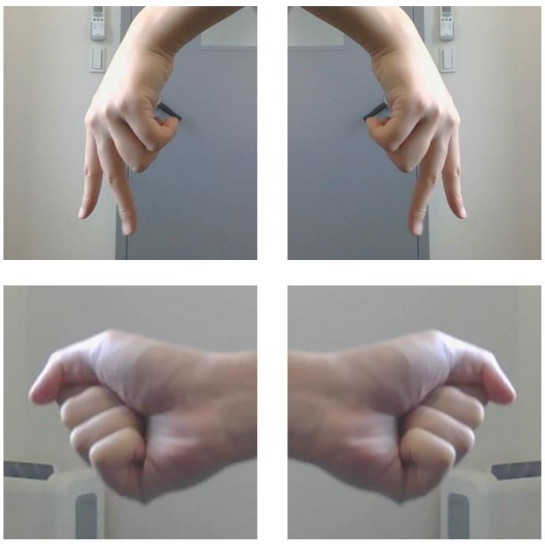

圖 7 採用 Horizontal Flip 後的資料集

(2) 模型訓練

本研究使用 Transfer Learning[4]之技術訓練模型，並選擇以 MobileNetV2[5]作為 Pretrained Model[6]，除體積小以確保遊玩過程能 real time 外，更是測試比較各種模型的訓練狀況後，一致認為其綜合表現最佳而選擇。模型訓練分為兩階段，第一階段選擇以 Adam[7]作為 optimizer 以訓練分類器，並設有 Early Stopping[8]的機制防止訓練過度造成 over fitting 的狀況。第二階段則以 0.0001 學習率的 Stochastic Gradient Descent (SGD) [9]進行 Fine Tune[10]，以訓練出更準確的模型。

(3) 結合 Party Game

CNN 模型訓練完成後，再次放入手勢圖片，遊戲結果將使角色做出相應動作。

3. Tracking

為了能準確地讓多位玩家同時體驗，我們結合了 Simple Online And Realtime Tracking(SORT)[11]與骨架相似度比對的技術來處理互動過程中

玩家的手因交疊而消失在畫面中的問題，並能使非玩家的手出現於鏡頭時不會干擾遊戲的進行。用先前的 MediaPipe hand 所提供之手部資訊中的 bbox(bounding box)參數，對所偵測到的手運用 Kalman Filter(卡爾曼濾波)[12]演算法進行預測，推測出那隻手於下一個畫面可能出現的位置(見圖 8)。並使用匈牙利演算法[13]將手可能出現的位置與下一個畫面所偵測到的手進行比對，再運用 MediaPipe hand 所提供之手部骨架資訊對可能為同一隻手之骨架進行 cosine 相似度比對，來判斷舊畫面的手與新出現的手是否為同一隻，期望手若消失後再出現仍能判斷為同一隻手，且當畫面中有非玩家的手出現時不會被誤判為玩家，以達到追蹤的目的。

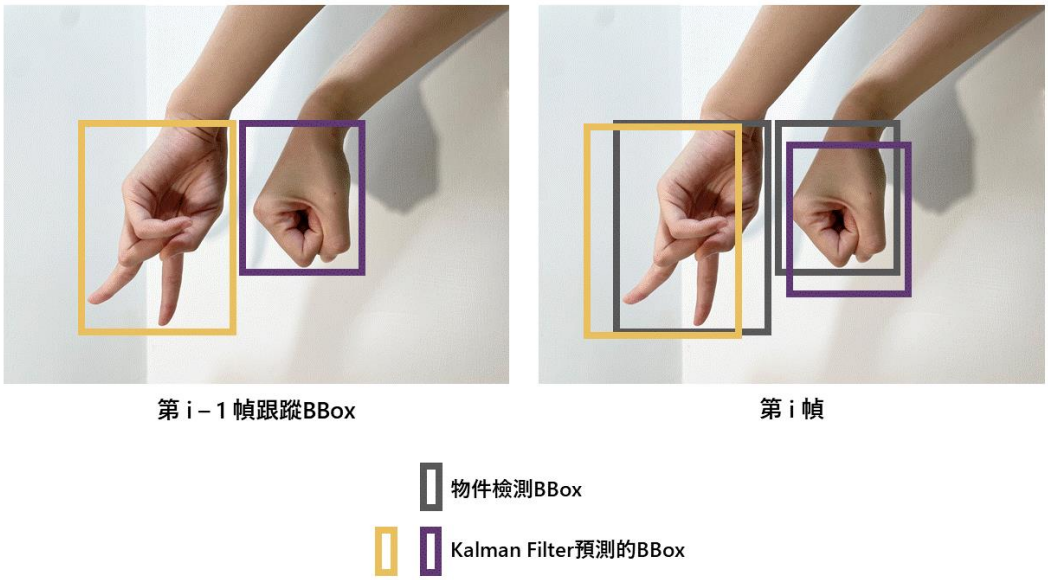

圖 8 Kalman Filter 追蹤示意圖

4. 多媒體畫面

為了提升視覺效果，針對遊戲角色的移動速度，分別使用了 Kalman Filter 演算法以及加入錨點計算的概念來處理多媒體畫面。

(1) Kalman Filter

Kalman Filter 處理透過 MediaPipe 估算角色跑步速度時雜訊引起的不確定性，使用先前根據食指與中指指間距離計算出之速度作為測量值，依末速度公式，經過卡爾曼濾波器演算法得出推測之速度，使角色速度更加穩定。

$$\bar{x}_t = F_t x_{t-1} + B_t u_t \quad (3)$$

$$P_t = F P_{t-1} F^T + Q \quad (4)$$

$$x_t = \bar{x}_t + K_t(z_t - H\bar{x}_t) \quad (5)$$

$$K_t = P_t H^T (H\bar{P}_t H^T + K)^{-1} \quad (6)$$

$$Z_t = \mathrm{H}\, x_t + \text{雜訊} \quad (7)$$

而原公式是根據物理公式(公式 8, 9)得到公式 10, 11, 12

$$x_t = x_{t-1} + V_{t-1} * \Delta t + u_t * \frac{\Delta t^2}{2} \quad (8)$$

$$V_t = V_{t-1} + u_t * \Delta t \quad (9)$$

$$\begin{bmatrix} x_t \\ V_t \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ V_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} u_t \quad (10)$$

$$F_t = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \quad (11)$$

$$B_t = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} \quad (12)$$

但由於我們要求的是速度的預測而不是位移的預測，因此我們利用末速度公式 V = V0 + at，將其改寫為公式 13, 12, 15。

$$\begin{bmatrix} V_t \\ a_t \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_{t-1} \\ a_{t-1} \end{bmatrix} \quad (13)$$

$$F_t = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \quad (14)$$

$$B_t = 0 \quad (15)$$

(2) 2D 橫向卷軸運鏡[14]

為了讓角色在不超出畫面邊界的同時，移動速度更加自然，因此我們作了以下處理。首先設定畫面中 X 座標=780 的位置為邊界(見圖 9)，而移動速度的計算方式就分為尚未到達邊界以及到達邊界，前者由於角色的 X 座標會右移，因此角色實際移動的距離為背景左移速度和角色右移速度的差，後者則是當速度維持時，角色因到達邊界使得錨點須固定在邊界上，由於角色的 X 座標不會再向右移，因此背景左移的速度直接就是

角色移動速度，此時為達到與前者移動速度相符，背景捲動速度就會增加，以達到此目的。

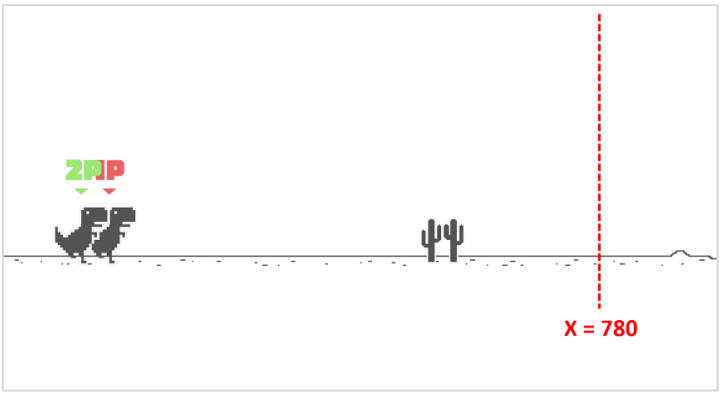

圖 9 X 座標=780 之示意圖

(四)實驗結果與討論

1. 系統介紹

(1) 系統名稱

恐龍手指跑步賽(RDG)

(2) 系統非功能需求

非功能需求編號	非功能需求描述
RDG-NF-001	紀錄當前玩家的手部辨識資訊，用以防止遊戲玩家之外的手，因進入鏡頭偵測範圍而干擾遊戲。
RDG -NF-002	針對遊戲畫面幀數，訂定閾值，若低於閾值則會進行相對應處理，以避免因幀數過低而導致畫面不順暢。
RDG -NF-003	為了使遊戲角色的移動速度更加流暢，另外使用了 Kalman Filter 演算法來做額外的數據處理，以維持遊戲畫面的穩定性。
RDG -NF-004	針對現成套件 MediaPipe 在高速手指移動時，未能夠精確地貼合手指骨架，系統以 CNN 模型來補足現成套件的有限性，提高手勢辨識的精準度。

(3)系統功能需求

功能需求編號	功能需求描述
RDG -F-001	使用者在鏡頭前做出特定手勢後，系統能夠即時辨識是否為特定手勢，並使遊戲角色做出相對應的動作。
RDG -F-002	系統會計算各玩家的手指移動速度，並依照速度轉換成步數，並顯示在遊戲畫面上。
RDG -F-003	系統提供遊戲角色是否觸碰到障礙物的偵測，並在遊戲畫面中呈現相對的反應。
RDG -F-004	系統能夠支援多人同時競賽的模式。

2. 結果

(1) 使用 CNN 進行手勢辨識

最終本研究選擇以辨識結果較佳的卷積神經網路模型來完成手勢辨識，手勢總共分為拳頭及非拳頭兩個類別，角色會根據不同手勢做出相對應的動作。當模型辨識為非拳頭時，角色會呈現跑步狀態，辨識為拳頭時，則會從跑步狀態切換為跳躍狀態(見圖 10)。

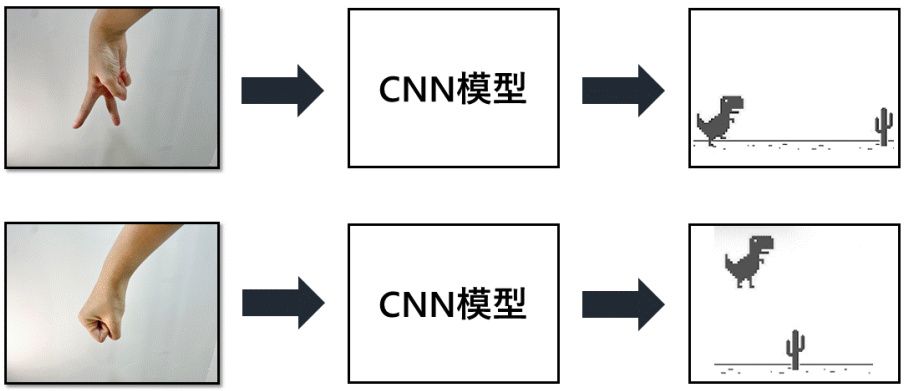

圖 10 訓練完成的 CNN 模型會得到之相應結果

(2) 使用 Tracking

使用 SORT 演算法以及骨架相似度比對以後，成功解決了非預期的手因被鏡頭偵測到而誤判玩家的問題，也做到當玩家的手消失後再出現仍能夠判斷為同一隻手，使得遊戲在上述情況仍能順利進行。圖 11 可見 SORT 演算法及骨架相似度比對會記錄每隻手的識別 ID，藉由預測及 ID 的比對可成功防止獲取到非預期的手部資訊。

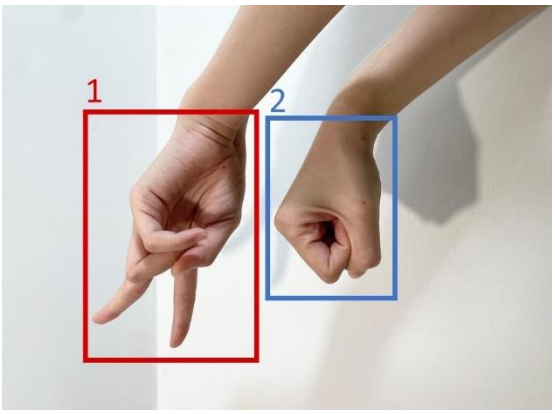

圖 11 Tracking 結果示意圖

(3) 使用 Kalman Filter 增加速度穩定性

在使用 Kalman Filter 演算法後，角色速度的穩定性大幅提升，圖 12 藍線為透過 MediaPipe 估算之角色跑步速度，呈現多處鋸齒狀的線條，紅線為經 Kalman Filter 處理後之角色跑步速度，可看出經 Kalman Filter 處理後之角色速度更加穩定。

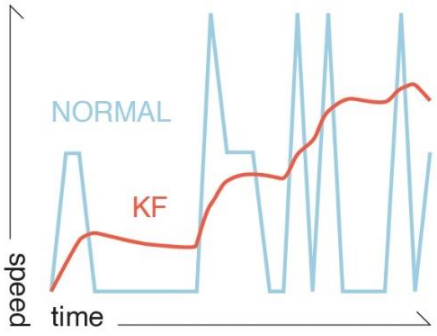

圖 12 使用 KF 前後之曲線圖

(五)未來展望

為解決手部辨識相關產品於空間及裝置上的限制，本研究使用 MediaPipe 工具庫及卷積網路模型相互配合辨識手勢，同時運用 Simple Online And Realtime Tracking(SORT)避免可預期之干擾，並使用 Kalman Filter 優化多媒體之呈現。以期開發一款不需操作裝置且無空間限制即能多人互動之媒體，不再受於傳統載體侷限，而是能輕鬆地在不同場景與身邊朋友即時互動，期望能應用於人機互動相關發展，也可為手部辨識相關應用提供參考。最終更期望能應用在如沉浸式互動體驗相關之展覽或是博物館等場所，令資訊傳遞方式更有趣，同時也加強大眾對資訊的印象和吸收力。

(六) 參考文獻

[1] “MediaPipe,” [線上].Available: https://google.github.io/mediapipe/

[2] “CNN,” [線上].Available: https://www.tensorflow.org/tutorials/images/cnn

[3] “Data augmentation,” [線上].Available: https://www.tensorflow.org/tutorials/images/data_augmentation

[4] “遷移學習,” [線上].Available: https://blogs.nvidia.com.tw/2019/02/07/what-is-transfer-learning/

[5] “MobileNetV2,” [線上].Available: https://keras.io/api/applications/mobilenet/

[6] “Pretrained Model,” [線上].Available: https://keras.io/api/applications/

[7] “Adam,” [線上].Available: https://keras.io/api/optimizers/adam/

[8] “EarlyStopping,” [線上].Available: https://keras.io/api/callbacks/early_stopping/

[9] “SGD,” [線上].Available: https://keras.io/api/optimizers/sgd/

[10] “Transfer learning and fine-tuning,” [線上].Available: https://www.tensorflow.org/tutorials/images/transfer_learning

[11] “Simple Online and Realtime Tracking,” [線上]. Available: https://github.com/abewley/sort

[12] R. Faragher, “Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation”, IEEE Signal Processing Magazine, Sept. 2012.

[13] Harold W. Kuhn, "The Hungarian Method for the assignment problem", Naval Research Logistics Quarterly, 2: 83–97, 1955. Kuhn's original publication.

[14] “Scroll Back - 2D 捲軸遊戲的攝影機理論與實務,” [線上]. Available: https://igdshare.org/content/gdc2015-2d-scrolling-itay-keren