元智大學電機工程學系甲組

1082 畢業專題

運用單相機之桌球 3D 軌跡還原

3D Track Recovery for Table Tennis Using Single Camera

學生：蔡永楨、陳煒竣、管謹中、黃裕凱

指導教授：陳敦裕 教授

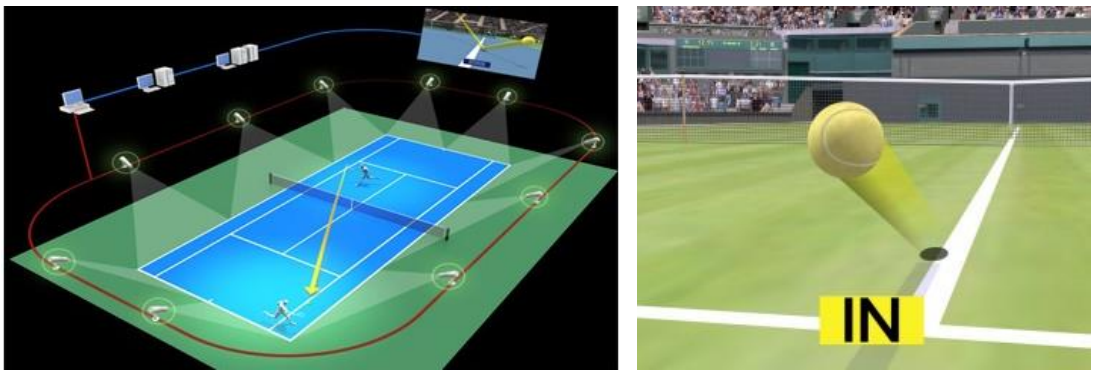
中華民國 109 年 5 月

目錄

一、緒論

1.1 簡介

隨著科技進步，各式的體育競技活動逐漸加入以相機為主的裁判輔助系統，簡單如賽跑項目在終點線錄影可以更客觀的紀錄勝者，複雜如網球的「鷹眼」即時回放系統[6]，如 Figure 1，透過多個高速攝影機從不同角度捕捉網球軌跡，經過電腦運算，最後由螢幕呈現網球的飛行路徑與落點。本文提出以「單台」攝影相機來還原桌球 3D 軌跡的方法，運用傳統影像處理與深度神經網路，偵測出桌球在畫面上的位置，並分割其面積，透過分析球體面積，獲得深度資訊，進而計算出其 3D 資訊。此系統僅需較便宜且易取得的錄影設備錄製影像(本文使用單支手機錄影)，再經由電腦計算便能得出包含 3D 桌球軌跡、落球點與雙方的得分，大幅減少以往需要多相機的不便。



(a)即時回放系統架構
(圖片來源: training with james)

(b)回放畫面
(圖片來源: mediaandtennis.)

Figure 1: 用於網球比賽的鷹眼即時回放系統。

1.2 研究動機與目的

桌球是一種老少皆宜的休閒運動，目前主要是由多個裁判在四周全程關注比賽過程，並將觀察所得加以計分顯示，但是除了正規或是大型賽事，許多時候並沒有裁判能夠主持比賽，且此方式所需人力較多，又易受裁判主觀判斷影響，可能有失公正，而在平常練習時，也經常會有人力不足等因素，導致只能由雙方練

習者自行計分，容易有分神記憶或是遺忘誤算等缺點，因此我們認為輕量、便利且準確的計分與記錄系統似乎可以有效解決上述因為人為計分判定所造成的缺點，而一個完整的自動計分系統通常包含 2D 平面資料與 3D 深度資料，其中獲取 3D 資訊通常需要兩個或以上的多鏡頭系統才能達成，這一情況除了提高空間的限制，也使得設備的成本大幅增加。

由於先前分析的種種情況，我們構思了一個單鏡頭實現桌球計分判斷與 3D 軌跡還原模型的系統。在計分判斷的部分，交由影像辨識技術進行實現，擔當裁判的角色；3D 軌跡還原部分，我們分析每一段來回的擊球，結合落球點等資訊，還原其 3D 軌跡，再結合 GUI 使軌跡能以 3D 模型呈現，而 3D 模型的功用有許多，對於使用此系統的練習者而言，可以經由 3D 軌跡去觀察在發球或回擊時球的路徑與角度，來思考最佳的擊球方式，精進自己的技術。另一方面，對於比賽計分，或許可以利用其 3D 軌跡，對比數有爭議的部分進行驗證。

二、相關研究

2.1 物件偵測

物件偵測在計算機視覺中是一個常見的任務，其目標為定位且分類畫面中的各個物件，傳統利用影像處理演算法來進行物件偵測，如 HOG [7]、SIFT [8]等；近年因為神經網路的崛起，許多用於偵測任務的神經網路也被提出，如 faster RCNN [9]為常見的 two stage 神經網路，faster RCNN 首先挑選出可能的候選區域，再針對候選區域進行分類，進而得出各個物件的位置與類別；而 SSD [10]與 YOLOv3 [11]等 one stage 模型，則是在不犧牲太多準確度的情況下，大幅增加運算速度，使得神經網路能夠實時的預測結果 。

2.2 影像前景分離

由於移動中的桌球屬於前景資訊，若是能將前景與背景有效的分離，便可以

更加突顯桌球的位置， 背景消除(Background Subtraction)是計算機視覺的領域中主要的預處理步驟，運作方式為通過影片中的背景進行對照，利用 OpenCV 內建模型(KNN)實現背景消除，生成 mask 圖片，通過對 mask 二值化圖片分析實現對前景活動物件的區域的提取。

2.3 物件分割

近年常見的物件分割手法包括使用深度神經網路進行物件分割任務(如 Mask-RCNN [5])，其輸入彩色三維影像，先進行物件偵測，取得物件的 bounding box 後，再對 bounding box 內的圖像做像素級的前景與背景分類，以此取得物件的 Mask。

2.4 深度資訊

使用深度相機、光學雷達等能夠直接取得物件的深度資訊，但此類設備通常較為昂貴且不易普遍使用，間接取得深度資訊的方法則可以藉由 2D 影像中已知真實尺寸的物件來推估其他影像中物件的尺寸。

三、研究方法

3.1 系統主要架構

本系統主要可分為三大部分：桌球與落球點偵測、計分系統、3D 資訊還原。Figure 2 為本系統主要架構圖，首先，我們將影片載入並取出第一個 frame 進行球桌八個關鍵點標記，接著對影像進行偵測並記錄球的位置，此位置資訊進行一系列的計算可得落球點，而計分系統可以藉由桌球的位置資訊、落球點資訊以及上述提到的八個關鍵點，透過多重條件的判斷，最終獲取輸入影片的比數。在 3D 還原方面，則以桌球的位置資訊取得面積，結合落球點輔以修正，建立出較接近真實情況的 3D 軌跡圖。

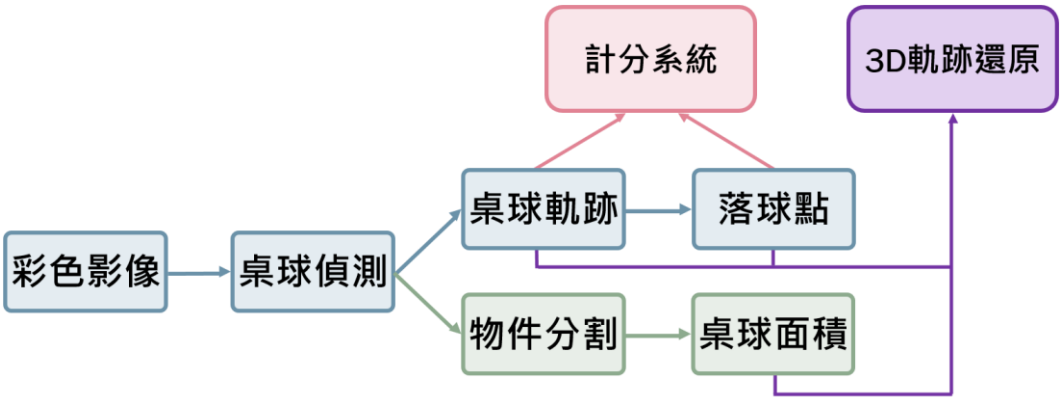


Figure 2: 系統主要架構。

3.2 落球點判斷

3.2.1 detected circle

在第一階段中，我們將影片每一幀讀入，並每隔兩幀進行處理，首先使用 KNN (K-nearest neighbor)法將 frame 的前景與背景分離，加入高斯模糊後再以幀差法突顯出正在移動的物體，最後對圖片進行二值化，進一步過濾圖片中的噪點，得到一張經處理過後的圖片，如 Figure 3。

Figure 3: 經過 KNN 背景分離與幀差結果後的二值化影像。

接下來對 Figure 3 進行霍夫圓偵測，由於霍夫圓會偵測出球與許多非桌球的圓型，所以我們訓練了一個可以判定是否為桌球的神經網路，將霍夫圓偵測出的結果全部輸入神經網路做分類，以此找出正確的球位置，並將球在畫面上標示出來連成軌跡，偵測與軌跡結果如 Figure 4。

Figure 4: 偵測出之球軌跡(黃線為軌跡，紅圈為球體位置)。

由於接下來的落球點判斷的需要，我們記錄連續十三顆球圓心的 x, y 座標作為一組數據，而更新數據的方式為 FIFO，將最先偵測到的圓心座標移出，並將最新的座標放置在最後。

3.2.2 vector method

在落球點偵測部分，我們使用了兩種方法，其中一種為向量角度判定法，Figure 5 為向量角度判定法之流程圖。

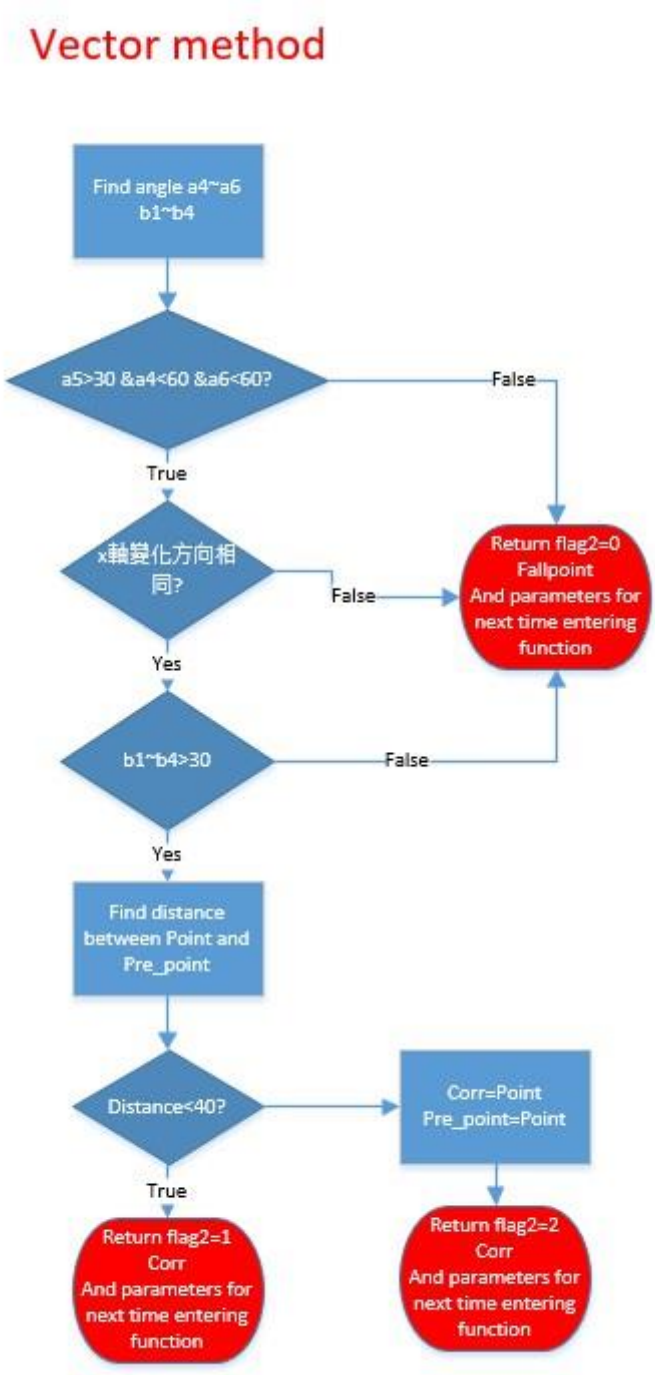


Figure 5: 向量判斷方法流程圖。

第一層判斷條件為取十三顆球中間的五顆(球 4、5、6、7、8)，每兩顆球連成一向量，球 4~6 兩向量之間的角度為 a4，球 5~7 兩向量之間的角度為 a5，球 6~8 兩向量之間的角度為 a6，而落球點與回擊點都有角度的改變，但最大的差別就是 x 軸的方向變化，所以為了解決偵測到回擊點的情況，第

二層的判斷條件為 x 軸的變化方向是否為同向，此外，由於軌跡有時會有抖動問題，也容易產生不規則的角度變化，但大多都在短時間內回復正常，因此將球座標以間隔 1~4 幀建立向量並計算角度(b1~b4)作為一判斷條件來解決此問題，最後，不規則落球點(同一個落球點產生兩個轉折點)，當第二個判定點與第一個距離過近，便將它視為同一個落球點，如 Figure 6。

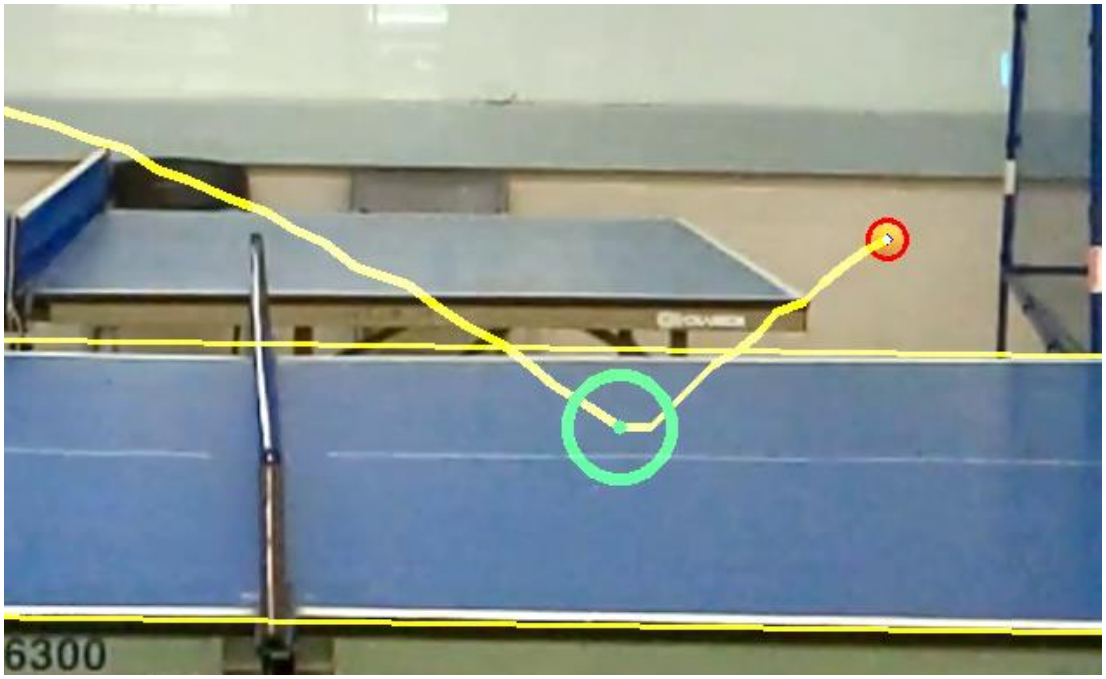


Figure 6: 不規則落球點偵測示意圖。

3.2.3 equation method

前述介紹的向量角度法有一些不足的地方，例如角度過大的落球點以及軌跡平滑的落球點，如 Figure 7，有時會有判斷失敗的情況，為了增加判斷準確度與穩定度，我們新增了第二種偵測落球點的方法，軌跡方程式法，Figure 8 為軌跡方程式法之流程圖。

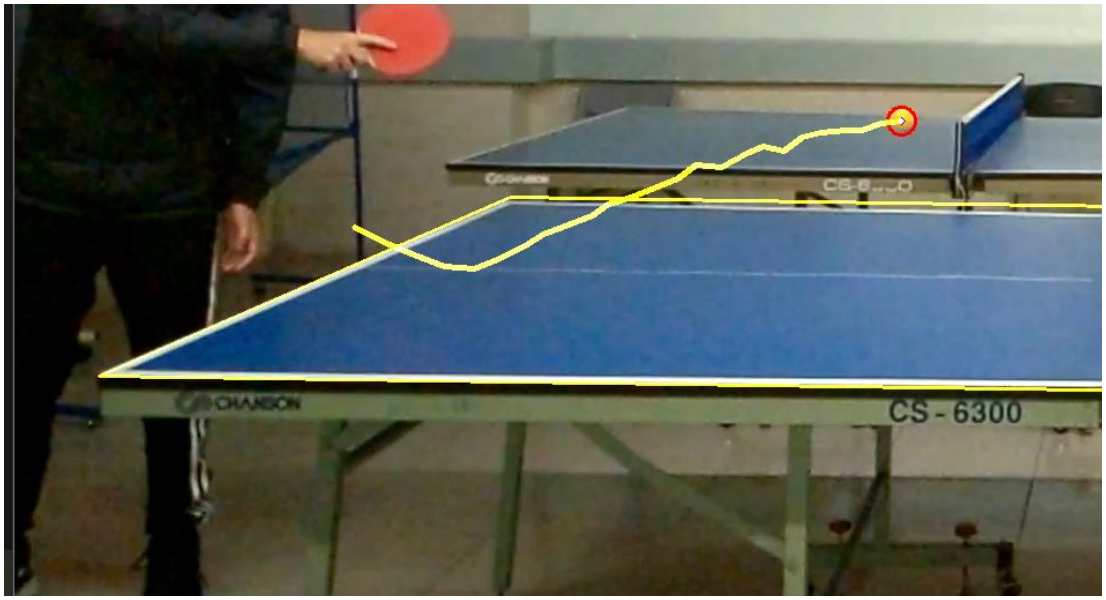


Figure 7: 軌跡平滑落球點未偵測情況。

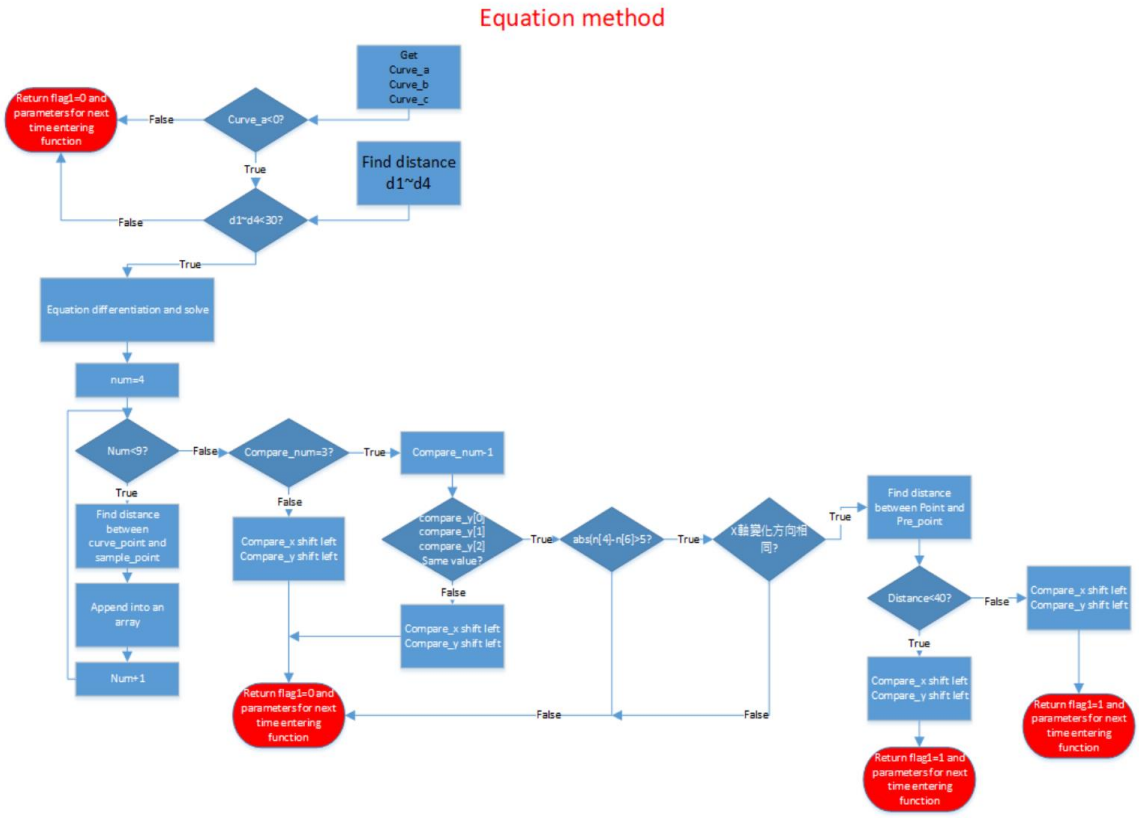


Figure 8: 軌跡方程式判斷方法流程圖。

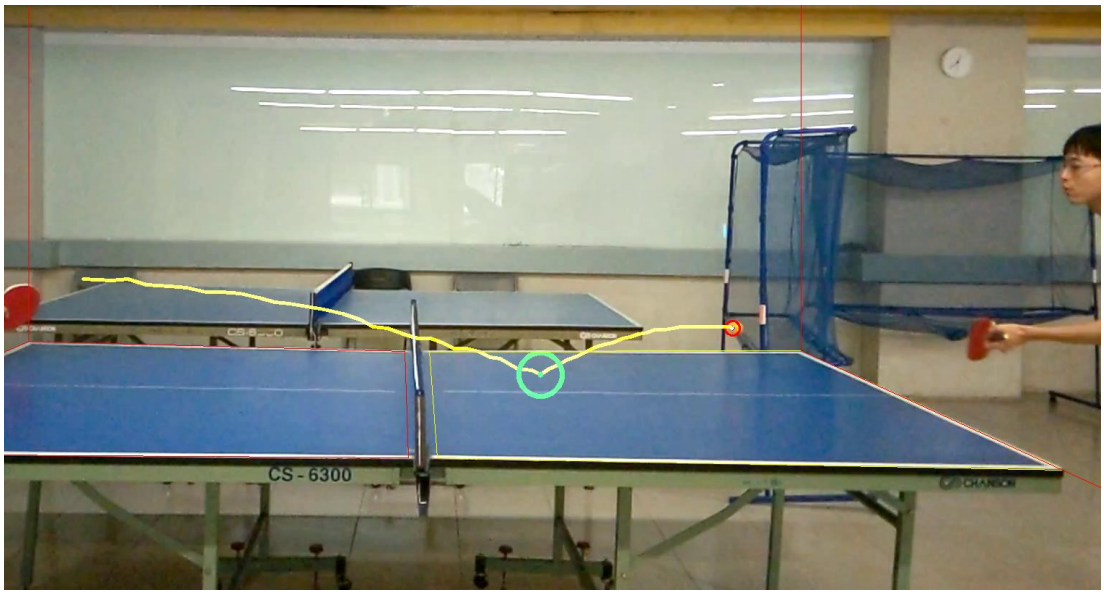
此方法一樣為取中間連續五個點做判斷，它們會產生一擬合曲線，並只取二次項係數小於零的情況，因為落球點附近的擬合曲線必定為凹口向上。當方程式找出之後，算出極值並與所取樣的五個點進行距離的計算，距離最小者為記錄點，重複將五個取樣點更新，移出第一個點並將最新的點放置在最後。為了加強此方法的嚴謹度，採取記錄相鄰三次情況都判定為同一個記錄點，此點才會被視為真正的落球點，同樣的，這邊也進行了回擊點的判斷與不規則落球點的判斷方法。

3.2.4 method conclusion

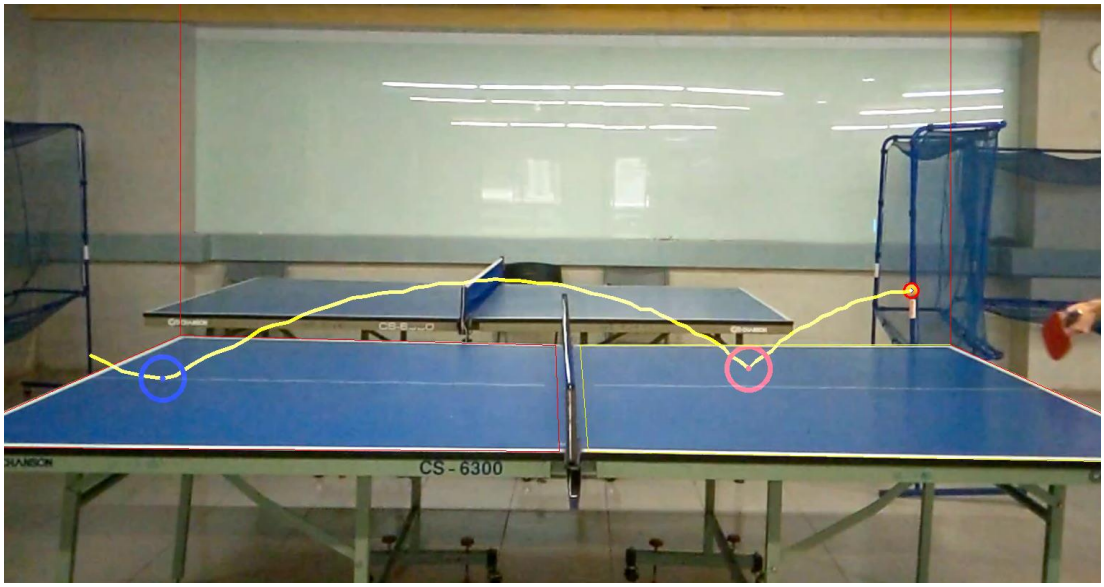
我們將上述兩種方法整合，詳細判斷結果與表示如 Table 1，向量角度判定法所判定情況設為 flag1，值 0 代表沒有判定到落球點，值 1 代表在不規則落球點情況下所判定的落球點，值 2 為正常情況下判定成功的落球點。軌跡方程式判定法設為 flag2，其值代表與向量角度的情況相同，並且記錄前面兩個 frame 的 flag 值，若皆為 0，代表此落球點第一次被偵測出來並標出，避免判定同一個落球點產生衝突。標記顏色部分，向量角度法為綠色，軌跡方程式法為藍色，兩者共同判定為紅色，如 Figure 9。

Table 1: 向量與軌跡方程式結果顏色表。

Flag1	Flag2	落球點標記	顏色
0	0	/	/
0	1	向量	綠
0	2	向量	綠
1	0	軌跡	藍
1	1	軌跡	紅
1	2	軌跡	紅
2	0	軌跡	藍
2	1	軌跡	紅
2	2	軌跡	紅



(a) 綠色圓圈為向量角度判定法偵測。



(b)藍色圓圈為軌跡方程式判定法偵測(左)。紅色圓圈為兩種判定法同時偵測(右)。

Figure 9: 符合不同判斷式以不同顏色呈現。

3.3 計分系統

3.3.1 區域定義

我們將球桌分成桌子兩側的內與外，我們透過球經過邊界的次數來判定球現在位置給予相對應的計分方式。Figure 10 為邊界判定法之流程圖。

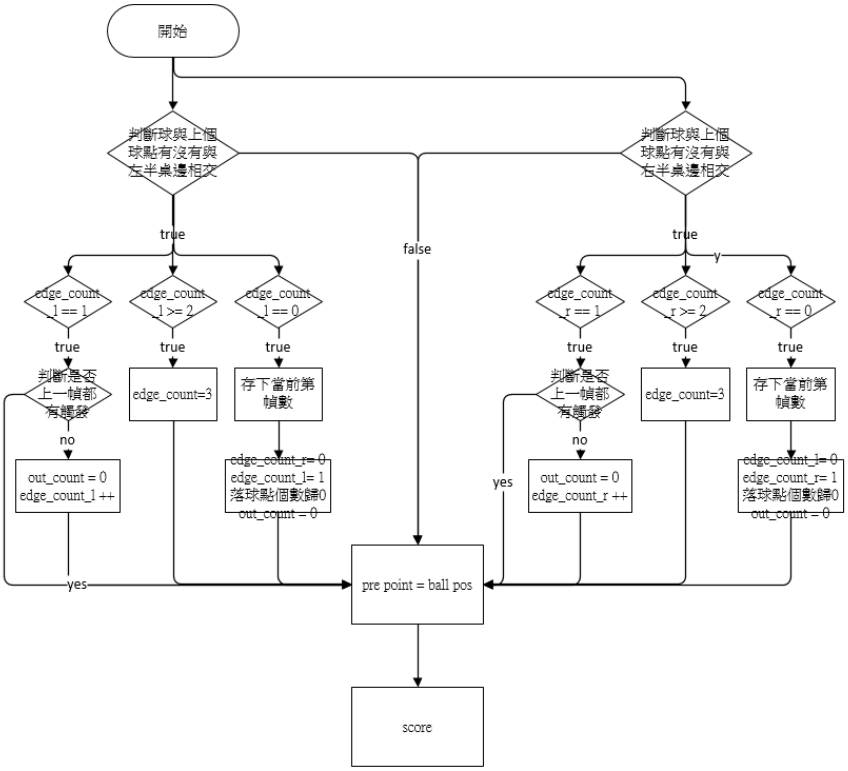


Figure 10: 邊界判定法之流程圖。

3.3.2 計分

我們計分的方式把一球的開始到結束分為三個狀態(起點、途中、死球)，如 Figure 11。死球狀態:我們不斷的刷新球座標，直到球出現在指定的區域，接著判定連續 4 幀球座標是否擁有方向性，如果上述兩者都滿足就進入起點狀態，如 Figure 12。

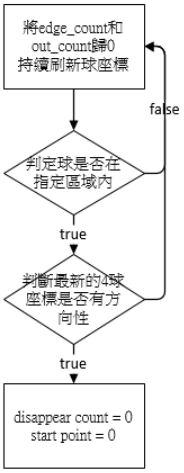
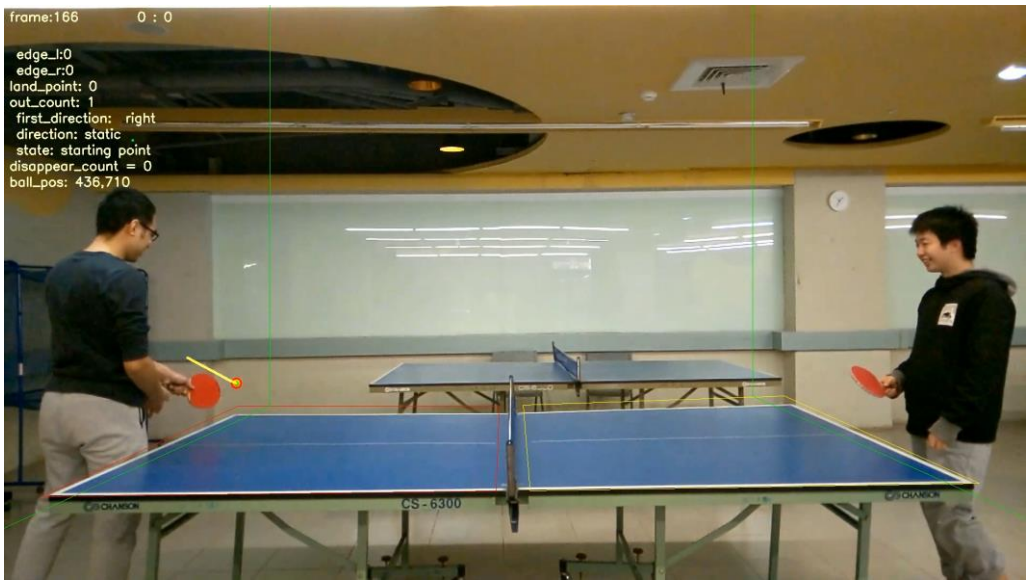


Figure 11: 死球狀態判斷流程圖。

(a) 死球狀態等待球的方向出現



(b) 當球有了方向性後從死球狀態進入起點狀態

Figure 12: 進入起點狀態之過程。

起點狀態: 判斷流程圖如 Figure14，先判定球是否失蹤過久，如果太久就回到死球狀態，反之，透過上面提到的邊界值開始做計分與否的判斷，Table 2 為在起點狀態時會遇到的計分結果(out count 為計數球在該 edge 狀態維持了多少個幀、edge count 為分別經過左右桌邊的次數、land_point 為產生落球點)，如果都沒分數產生，代表這球是一個合理的起點彈跳，那就會進入途中模式。Figure 13 展示起點狀態會經歷的情境。

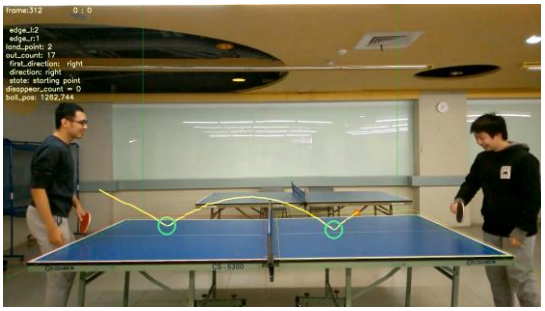
Table 2: 發球得分判定表。

edge_count_l	edge_count_r	from direction	out_count	land_point	get_score	situation
0	0	right	>200	x	left	發球沒進桌子
0	1 or 2	right	>200	x	left	球在自己桌內太久(發球失誤)
0	1 or 2	right	x	>=2	left	球在自己桌內不合理彈跳超過(發球失誤)
1	2	right	>200	>=3	right	進入對方桌子多次彈跳
1	2	right	>200	=1	left	正常得分
0	0	left	>200	x	right	發球沒進桌子
1 or 2	0	left	>200	x	right	球在自己桌內太久(發球失誤)
1 or 2	0	left	x	>=2	right	球在自己桌內不合理彈跳超過(發球失誤)
2	1	left	>200	>=3	left	進入對方桌子多次彈跳
2	1	left	>200	=1	right	正常得分

(a)

(b)

(c)



(d)

(e)

Figure 13: 圖為起點狀態時所會經歷的五個球狀態 (a)球有方向性後進入此狀態但還沒進入球桌 (b)發球進入己方球桌 (c)發球離開己方球桌 (d)發球進入對方球桌 (e)發球期間沒出狀況進入途中狀態。

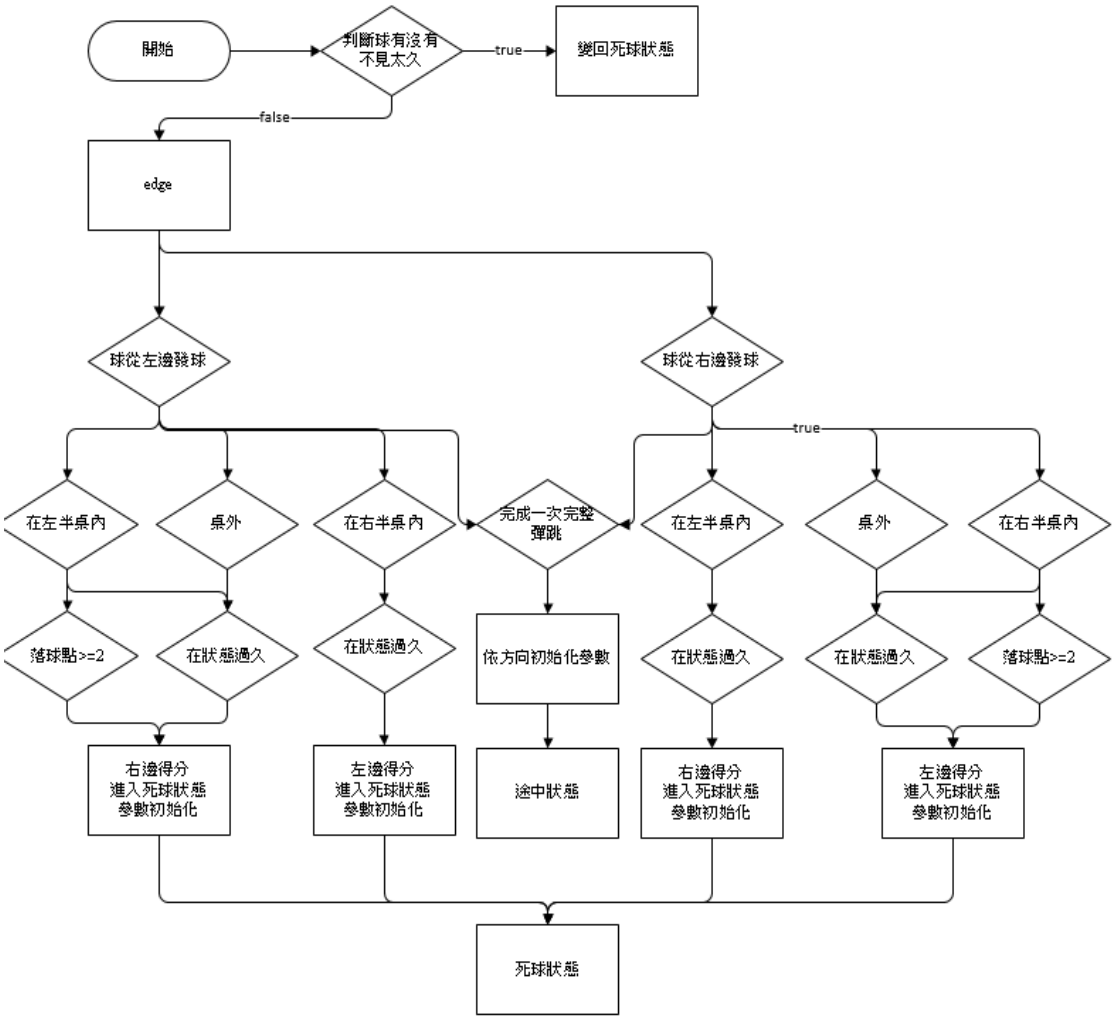
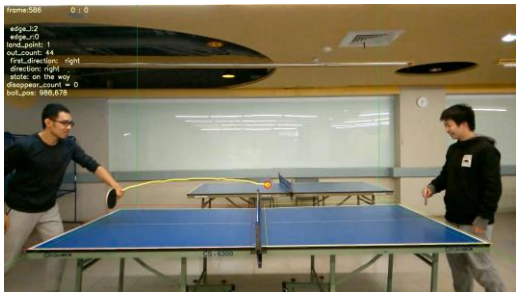


Figure 14: 發球狀態判斷流程圖。

途中狀態: 得分判斷流程如 Figure 16，途中狀態不斷的刷新 edge count 和 land point，直到產生分數為止，接著初始化所有值，進入死球狀態。Table 3 為途中狀態會產生的計分結果。Figure 15 展示途中狀態會經歷的情境。



(a)

(b)

(c)

Figure 15: 圖為途中狀態時所會經歷的三個球狀態 (a)球在被回擊的路上(參數還沒更新) (b)球進入對方球桌(參數更新) (c)球離開對方球桌。

Table 3: 途中得分判定表。

edge_count_l	edge_count_r	out_count	land_point	get_score	situation
1	0	>200	0	left	出界(剛好路徑削到)
1	0	>200	=1	right	正常得分
1	0	x	>=2	right	正常得分
0	1	>200	0	right	出界(剛好路徑削到)
0	1	>200	>=1	left	正常得分
0	1	x	>=2	left	正常得分
2 or 3	0	>200	0	right	出界或打到自己桌內
2 or 3	0	>200	=1	left	正常得分
2 or 3	0	x	>=2	right	打到自己桌內
0	2 or 3	>200	0	left	出界或打到自己桌內
0	2 or 3	>200	=1	right	正常得分
0	2 or 3	x	>=2	left	打到自己桌內

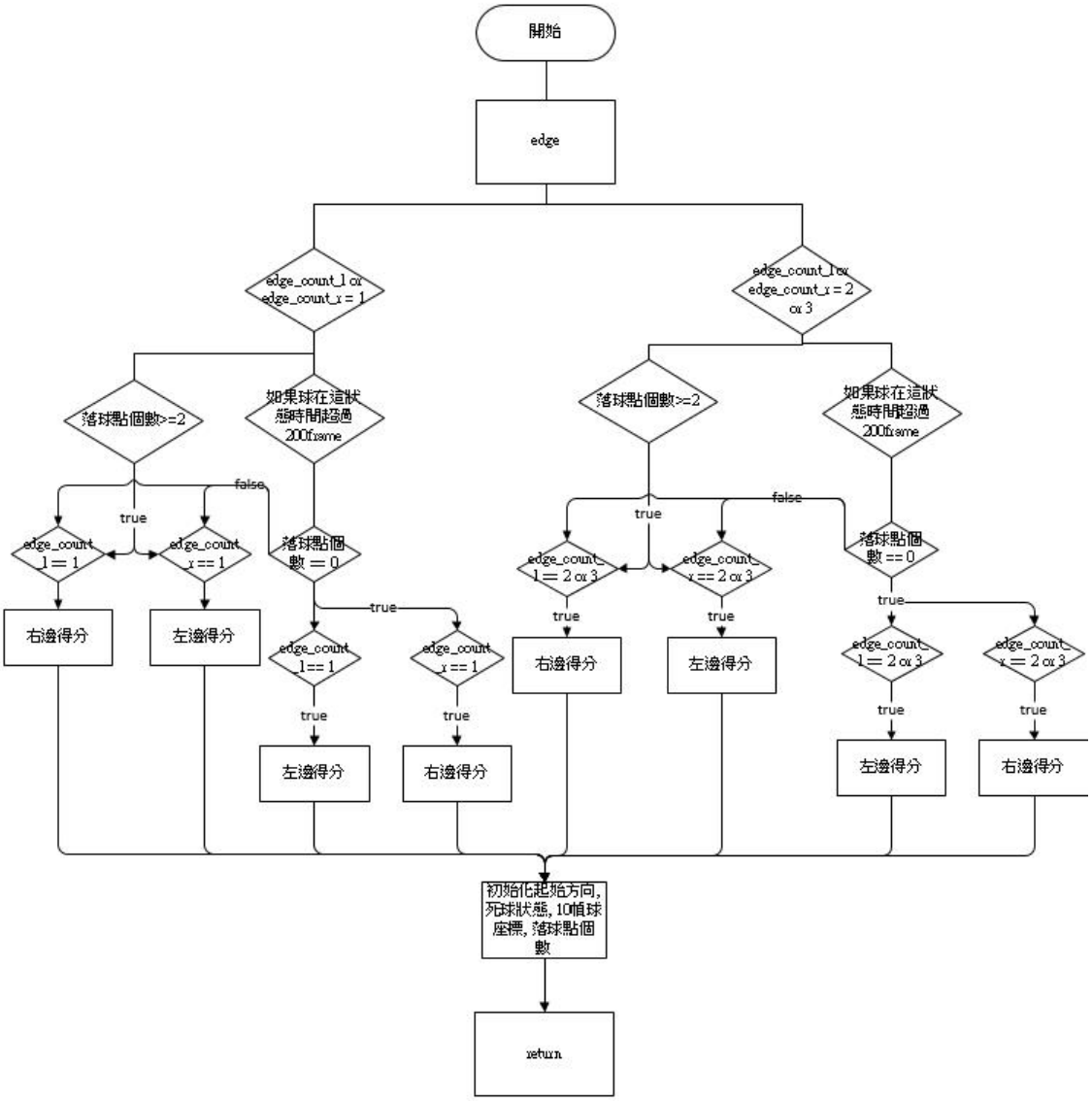


Figure 16: 得分判斷流程圖。

3.4 深度資訊(Y 座標)

為了獲得球的深度資訊(也就是球距離相機的遠近)，我們分析球在影像上的面積來獲得深度資訊。示意圖如 Figure 17，由於我們已知標準的桌球桌的尺寸(長 274 公分，寬 152.5 公分)與球桌在畫面中的長度(像素)，也已知標準桌球的直徑(4.5 公分)，接著定義球桌的左上角為原點，便能藉由球在影像中的面積來推算出球的深度資訊(Y 座標)。

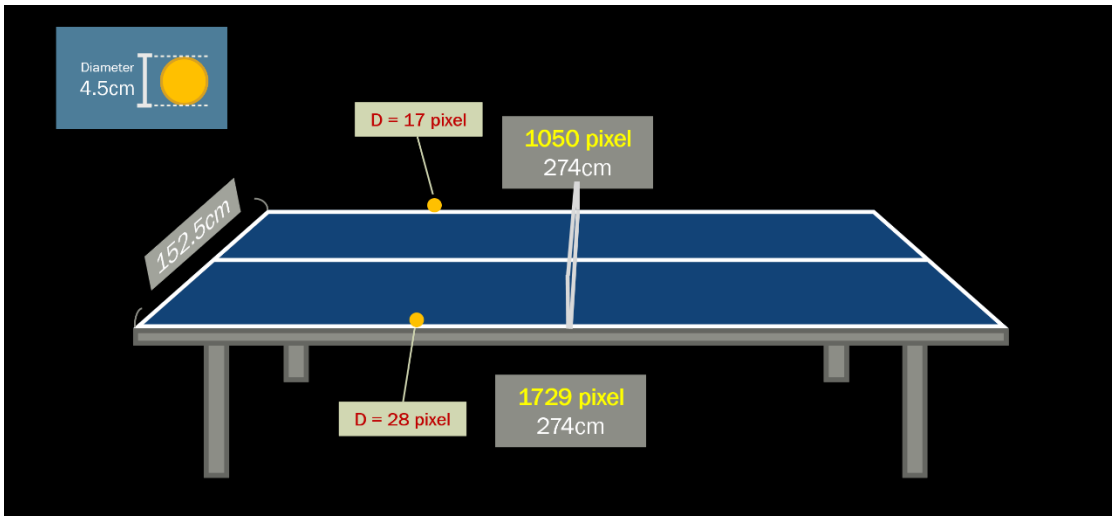


Figure 17: 桌球與球桌尺寸示意圖，已知球桌長寬與球的直徑，便可推算出球在不同位置時的圖像中直徑，也可以藉由圖像中直徑反推出真實深度(Y 座標)。

3.5 球面積

取得影像中的球面積為一個分割任務，而分割任務可以由許多方法來完成，包括傳統的色域分析或是利用神經網路預測。我們比較了兩種方法：使用 HSV 色域進行分割與使用卷積神經網路(CNN)進行分割。HSV 的部分，色域選擇了橘色桌球的色域來進行分割；而神經網路部份，架構如 Figure 18，我們選用了 MobileNet 作為 backbone 來取出特徵，再加入上採樣層，使結果能夠還原成與原始影像相同的大小，最後預測出球在影像中的 Mask。結果顯示，CNN 在各方面都優於傳統的 HSV 色域分析，因此我們採用 CNN 來進行分割任務。

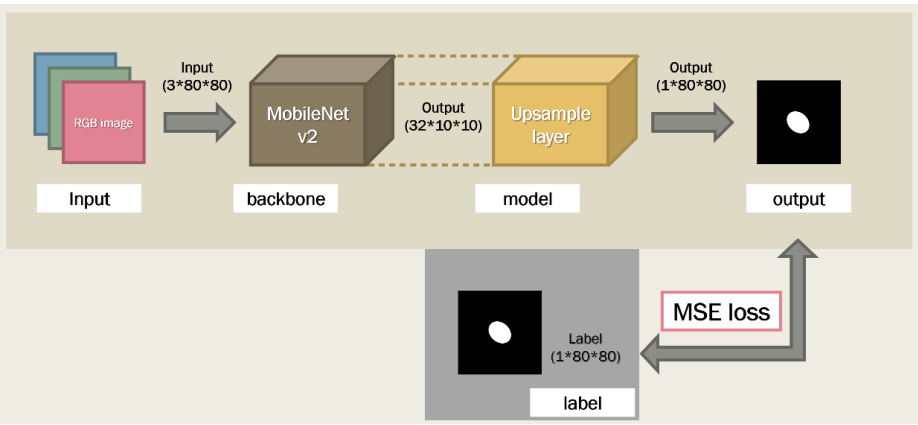


Figure 18: 分割任務神經網路架構圖。

由於球在錄影的影像中會因為快速移動而產生殘影，直接計算球體面積，會造成錯誤的結果，因此我們先計算神經網路預測出的 Mask 的最大內接圓，以最大內接圓的直徑做為計算深度資訊的依據，如 Figure 19。

(a) (b) (c)

Figure 19: (a)移動快速而產生殘影的原始圖像，(b)經過神經網路預測出的 Mask(白色區域)，(c)Mask 的最大內接圓(紅圈)。

3.6 3D 資訊

首先，我們定義真實世界得物體座標為三維的(X, Y, Z)，經過拍攝後，可以在影像上形成物件，其座標為二維的(u, v)，如 Figure 20。

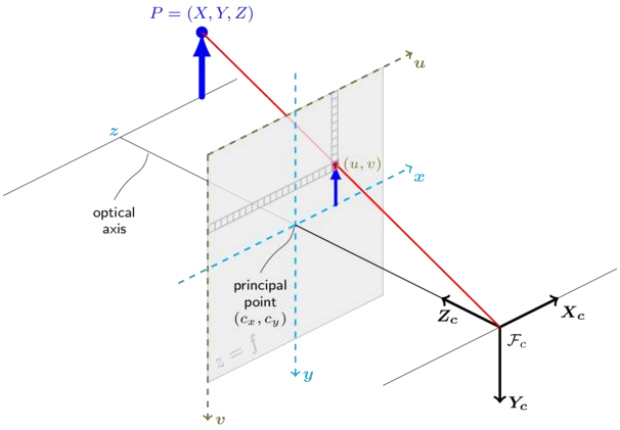


Figure 20: 針孔相機模型投影(圖片來源: OpenCV document)。

而影像上的二維座標與真實世界的三維座標可以藉由「針孔相機模型」公式來互相進行轉換，如公式(1)。

$$s\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}f_x & 0 & c_x\\0 & f_y & c_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_1\\r_{21} & r_{22} & r_{23} & t_2\\r_{31} & r_{32} & r_{33} & t_3\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix} \quad (1)$$

其中相機內部參數與外部參數可藉由事先測量而得到，將內部參數與外部參數計算後，可以得到(2)：

$$\begin{bmatrix}su\\sv\\s\end{bmatrix}=\begin{bmatrix}P_{11} & P_{12} & P_{13} & P_{14}\\P_{21} & P_{22} & P_{23} & P_{24}\\P_{31} & P_{32} & P_{33} & P_{34}\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix} \quad (2)$$

而矩陣運算可以得到(3)：

$$\begin{aligned}su &= PX + P_{12}Y + P_{13}Z + P_{14}\\sv &= P_{21}X + P_{22}Y + P_{23}Z + P_{24}\\s &= P_{31}X + P_{32}Y + P_{33}Z + P_{34}\end{aligned} \quad (3)$$

將 s 帶入其中，可得(4)：

$$\begin{aligned}&\mathrm{X}(P_{11} - \mathrm{u}P_{31}) + \mathrm{Y}(P_{12} - \mathrm{u}P_{32}) + \mathrm{Z}(P_{13} - uP_{33}) = uP_{34} - P_{14}\\&\mathrm{X}(P_{21} - \mathrm{v}P_{31}) + \mathrm{Y}(P_{22} - \mathrm{v}P_{32}) + \mathrm{Z}(P_{23} - vP_{33}) = vP_{34} - P_{24}\end{aligned} \quad (4)$$

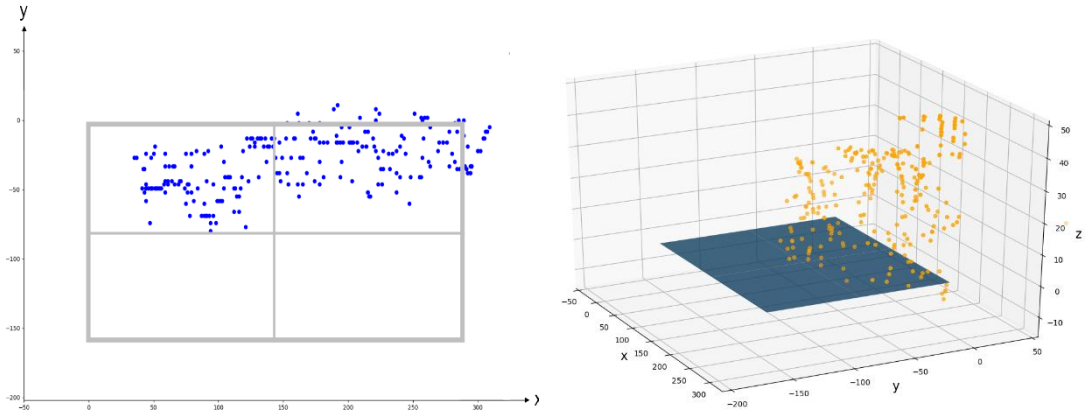
其中 Y 座標可以藉由前面所提的方法計算出來，將 Y 項與已知參數移至方程式等號右邊，可以得到(5)：

$$\begin{aligned}&\mathrm{X}(P_{11} - \mathrm{u}P_{31}) + \mathrm{Z}(P_{13} - uP_{33}) = uP_{34} - P_{14} - \mathrm{Y}(P_{12} - \mathrm{u}P_{32})\\&\mathrm{X}(P_{21} - \mathrm{v}P_{31}) + \mathrm{Z}(P_{23} - vP_{33}) = vP_{34} - P_{24} - \mathrm{Y}(P_{22} - \mathrm{v}P_{32})\end{aligned} \quad (5)$$

因此之後只要先利用前述方法計算出 Y 座標，再將影像中座標(u, v)帶入方程式中，便可得出完整的真實世界座標(X, Y, Z)。

3.7 資料處理(平滑化與擬合)

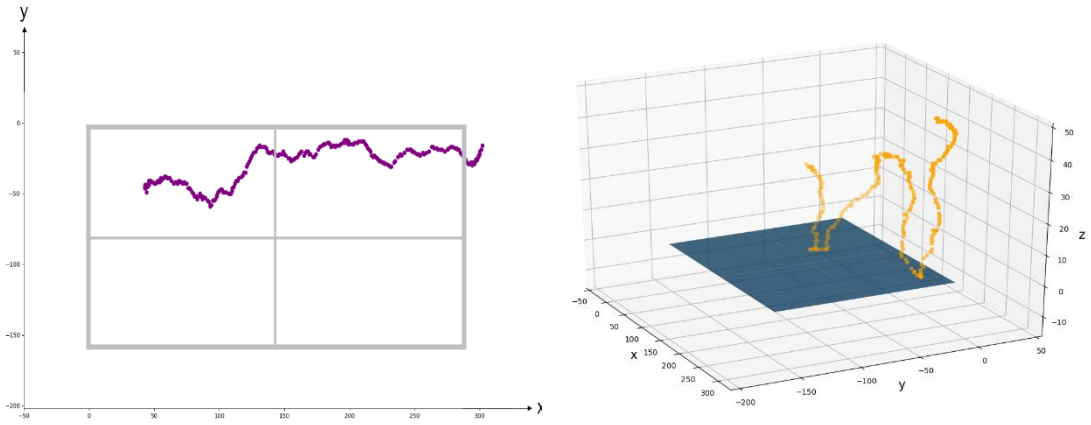
經由計算後可以得知，桌球在離相機較近的桌邊時，影像中直徑應為 28 pixel，在離相機較遠的桌邊時，直徑應為 17 pixel，而在真實尺寸中，球桌的寬度為 152.5 公分。由此可知，深度資訊的量化誤差可能非常的大，分割模型所計算出的球直徑每 1 pixel 的誤差會造成現實座標高達 13 公分的誤差。Figure 21 為經過前述手法獲得的深度資訊散佈圖，可見其誤差十分明顯。



(a)俯視圖　　(b)3D 側視圖

Figure 21: 經計算後得到未經任何處理的原始資料。

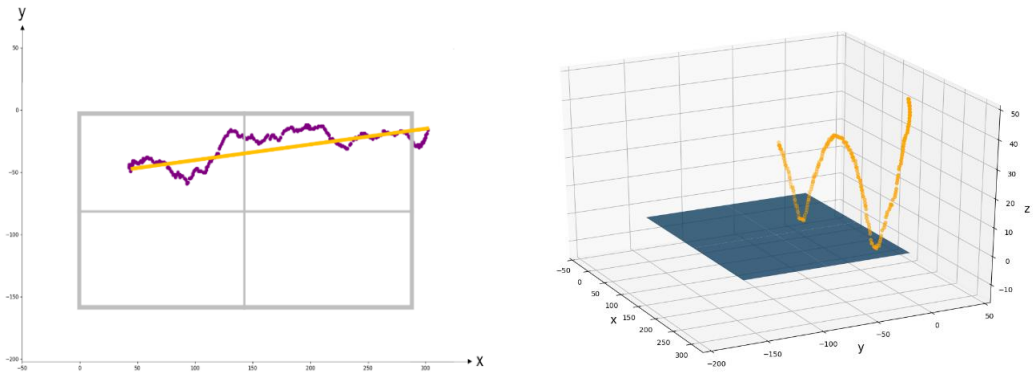
為了修正深度資訊，我們參考連續數個點的資訊，將資料平滑化，平滑化方式為參考連續 10 個點的位置，去除最大值、次大值、最小值與次小值，再進行平均，結果如 Figure 22，經過平滑化後的資料在深度資訊上的表現相對正確許多，但實際上桌球行進的路徑在俯視圖來看，應要為一條直線(不考慮曲球的情況下)，因此結果仍然不太合理。



(a)俯視圖　　(b)3D 側視圖

Figure 22: 經過平滑化後的軌跡。

為了符合桌球應有的正常軌跡(也就是俯視圖來看，應要為一條直線)，我們接著將此平滑化後的資料擬合至一條直線上，如 Figure23，觀察其 3D 側視圖，可見其軌跡變得合理許多，符合正常桌球行進的軌跡。

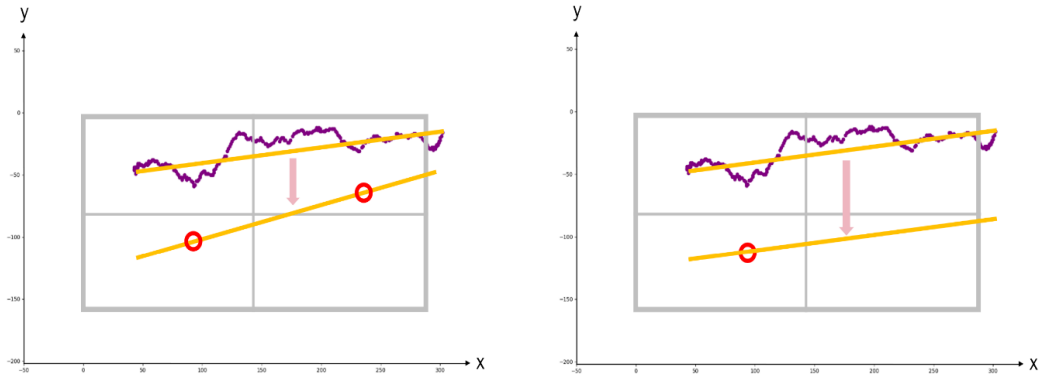


(a)俯視圖　　(b)3D 側視圖

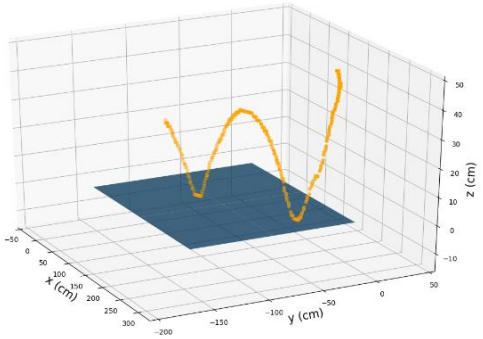
Figure 23: 經過曲線擬合後的軌跡。

最後參考落球點來調整預測路徑。在未失分的情況下，發球時會產生兩個落球點，而一般回擊會產生一個落球點，因此當偵測到兩個落球點時，將擬合的直線平移並改變其斜率，使其經過兩個落球點，如 Figure 24 (a)；而僅有一個落球

點時，將擬合的直線平移至經過僅有的落球點，如 Figure 24 (b)，如果沒有落球點或落球點大於兩個時則不進行平移或斜率調整。



(a)俯視圖-兩個落球點　　(b)俯視圖-單一落球點



(c)3D 側視圖

Figure 24: 經過落球點校正後的軌跡。

四、實驗與相關統計數據:

本文使用的拍攝設備為 HTC U12+ 手機，拍攝畫質為 1920*1080 pixel，使用慢動作攝影進行拍攝，每秒紀錄 240 張影像。

4.1 桌球偵測

桌球偵測部分，測試影片總共有 3 部，每部影片分為距離誤差 10 像素和 20 像素。

計算準確度部分，定義

correct 為 true positive (情況為球，偵測為球)

miss1 為 false negative (情況為球，沒有偵測為球)

miss2 為 false positive (情況不為球，偵測為球)

計算公式：

$$accuracy = correct / (correct + miss1 + miss\ 2)$$

$$precision = correct / (correct + miss2)$$

$$recall = correct / (correct + miss1)$$

Table 4: 三部影片桌球偵測準確度 (誤差距離 10、20)。

0331_2(1000)			
	accuracy	recall	precision
距離 10	37.35%	52.63%	56.27%
距離 20	89.30%	91.29%	97.61%
0109_1			
	accuracy	recall	precision
距離 10	95.24%	96.58%	98.56%
距離 20	95.40%	96.66%	98.65%
0109_2			
	accuracy	recall	precision
距離 10	94.50%	95.34%	99.06%
距離 20	94.78%	95.49%	99.21%

由 Table 4 可知，編號 0109 的兩部影片在距離 10 像素情況下就可以有接近

100%的準確度，而在 0331_2 影片中，誤差相較於其他測試資料大，推測為此影片中球速大於另外兩部影片所造成，但若進一步將閥值放寬至距離為 20 像素，其準確度也能達到 98%左右，可見這部影片偵測的誤差大部分落在距離 10~20 像素左右。

4.2 落球點偵測

在落球點偵測部分，Table 5 是測試四部影片的情況，每部影片分為距離誤差 10 像素與 30 像素，而 test1~test6 分別代表：

Method 1：a1~a9，no b1~b4，無不規則落球點判定，無 equation method

Method 2：a4~a6，no b1~b4，無不規則落球點判定，無 equation method

Method 3：a4~a6，no b1~b4，有不規則落球點判定，無 equation method

Method 4：only vector method but no equation method

Method 5：only equation method but no vector method

Method 6：both vector method and equation method

Perfect：除去觸網後所產生的大量不規則落球點判定

計算準確度部分，定義

correct 為 true positive (情況為落球點，偵測為落球點)

miss1 為 false negative (情況為落球點，沒有偵測為落球點)

miss2 為 false positive (情況不為落球點，偵測為落球點)

影片資訊：

20200109_1_trim_1 落球點個數：53 個

20200109_1_trim_2 落球點個數：30 個

20200331_2 落球點個數：243 個

20200331_3 落球點個數：254 個

Table 5: 四部影片落球點偵測準確度 (距離誤差 10、30)。

(a)

	20200109_1_Trim_1 accuracy(10)						
	Method1	Method2	Method3	Method4	Method5	Method6	perfect
accuracy	29.09%	36.84%	47.82%	55.38%	47.36%	50.00%	79.41%
miss1	37	25	20	17	26	14	3
miss2	2	23	16	12	4	25	4
precision	88.88%	54.90%	67.34%	75.00%	87.09%	60.93%	87.09%
recall	30.18%	52.83%	62.26%	67.92%	50.94%	73.58%	90.00%

(b)

	20200109_1_Trim_1 accuracy(30)						
	Method1	Method2	Method3	Method4	Method5	Method6	perfect
accuracy	31.48%	42.24%	52.23%	62.90%	58.49%	67.14%	96.77%
miss1	36	22	18	14	22	6	0
miss2	1	20	14	9	0	17	1
precision	94.44%	60.78%	71.42%	81.25%	100.00%	73.43%	96.77%
recall	32.07%	58.49%	66.03%	73.58%	58.49%	88.67%	100.00%

(c)

	20200109_1_Trim_2 accuracy(10)					
	Method1	Method2	Method3	Method4	Method5	Method6
accuracy	45.16%	51.61%	81.81%	90.62%	58.06%	100.00%
miss1	16	14	3	1	12	0
miss2	1	1	3	2	1	0
precision	93.33%	94.11%	90.00%	93.54%	94.73%	100.00%
recall	46.66%	53.33%	90.00%	96.66%	60.00%	100.00%

(d)

	20200109_1_Trim_2 accuracy(30)					
	Method1	Method2	Method3	Method4	Method5	Method6
accuracy	50.00%	56.66%	87.50%	96.77%	63.33%	100.00%
miss1	15	13	2	0	11	0
miss2	0	0	2	1	0	0
precision	100.00%	100.00%	93.33%	96.77%	100.00%	100.00%
recall	50.00%	56.66%	93.33%	100.00%	63.33%	100.00%

(e)

	20200331_2 accuracy(10)						
	Method1	Method2	Method3	Method4	Method5	Method6	perfect
accuracy	34.76%	41.19%	53.33%	60.27%	45.08%	54.92%	68.24%
miss1	154	126	75	67	125	70	37
miss2	13	41	72	49	19	72	37
precision	87.25%	74.05%	70.00%	78.22%	86.13%	70.61%	81.12%
recall	36.62%	48.14%	69.13%	72.42%	48.55%	71.19%	81.12%

(f)

	20200331_2 accuracy(30)						
	Method1	Method2	Method3	Method4	Method5	Method6	perfect
accuracy	39.11%	47.97%	67.12%	75.93%	54.47%	78.75%	96.00%
miss1	146	113	49	41	109	28	4
miss2	5	28	46	23	3	30	4
precision	95.09%	82.27%	80.83%	89.77%	97.81%	87.75%	97.95%
recall	39.91%	53.49%	79.83%	83.12%	55.14%	88.47%	97.95%

(g)

	20200331_3 accuracy(10)						
	Method1	Method2	Method3	Method4	Method5	Method6	perfect
accuracy	29.58%	35.48%	37.55%	47.11%	53.28%	41.62%	52.69%
miss1	175	144	87	107	108	105	61
miss2	13	56	196	58	20	104	53
precision	85.86%	66.26%	46.30%	71.70%	87.95%	58.89%	70.55%
recall	31.10%	43.30%	66.53%	57.87%	57.48%	58.66%	67.55%

(h)

	Method1	Method2	Method3	Method4	Method5	Method6	perfect
accuracy	35.15%	43.34%	59.53%	68.13%	64.06%	67.88%	84.00%
miss1	164	127	23	68	90	49	20
miss2	2	39	134	19	2	48	12
precision	97.82%	76.50%	63.28%	90.73%	98.79%	81.02%	93.33%
recall	35.43%	50.00%	90.94%	73.22%	64.56%	80.70%	89.36%

Table 5 的 Method1 部分為最初版本的落球點偵測準確度，我們最初的想法為盡可能多考慮落球點前後偵測到的桌球作為判斷依據，於是取了前後各六顆球，總計十三顆，計算出 a1~a9 共九個角度，此測試由於門檻設置太嚴苛，導致 recall 偏低，代表許多落球點沒有被偵測出。

為了改善 Method 1 門檻過高的問題，Method 2 則減少判斷球數，只取以落球點為中心前後各兩顆球，得出 a4~a6 的角度，從數據可看出 recall 雖有上升但變化不大，且經過觀察，我們發現除了取樣球數的問題，對於不規則落球點的判定效果也不好。

考慮到需要解決不規則落球點的問題，我們將角度閥值進一步放寬，讓不規則落球點的轉折點都會被偵測出，觀察得知，通常首次偵測到的點為正確落球點，所以只取該點作為正解，由數據得知新增不規則落球點判定可使 recall 大幅提升，但是因為閥值放寬造成系統過度敏感，輕微的軌跡抖動都會誤判為落球點，使得 precision 下降許多。

為了增加系統穩定度，新設立了四個角度 b1~b4，分別為以落球點為中心前後向外考慮一個點至四個點的情況，可以避免軌跡小幅震動所造成的誤判，數據上可看出相較於 Method 3，Method 4 的 precision 上升許多，但是 recall 變化不同影片有不同情況，比較各影片看出若是球速越快，越容易造成大角度或無明顯尖點之落球點，這是目前偵測效果不足的部分，在 0331_3 這部影片尤為明顯，導致 Table 5 (g)、(h)中 Method 4 的 recall 不升反降。

由於向量角度法判定落球點有其極限，因此加入了軌跡方程式判定法來突破向量偵測瓶頸，Method 5 為單純使用軌跡方程式法判定落球點，雖然準確度不如向量角度法，但此方法對於偵測大角度或無明顯尖點之落球點有很好的效果。

由於少偵測到落球點對計分系統的影響較大，所以我們在 Method 6 中將向量角度判定法與軌跡方程式判定法結合，提升了落球點判定的敏感度，雖導致數據中的 precision 較先前降低，但卻可以明顯減低對於計分系統的影響，另外，通常觸網後會造成大量不規則的軌跡，影響落球點的判斷，於是嘗試剔除觸網後的情況(perfect)來判斷落球點偵測是否有符合預期的效果，對比於 Method 6，剔除觸網後的 precision 跟 recall 都獲得顯著上升。

由於少偵測到落球點對計分系統的影響較大，所以我們結合軌跡方程式提升了落球點判定的敏感度(test6)，雖導致數據中的 miss2 較先前增加，但卻可以明顯降低對於計分系統的影響，另外，通常觸網後會造成大量不規則的軌跡，影響落球點的判斷，於是嘗試剔除觸網後的情況(perfect)來判斷落球點偵測是否有符合預期的效果。

4.3 計分系統

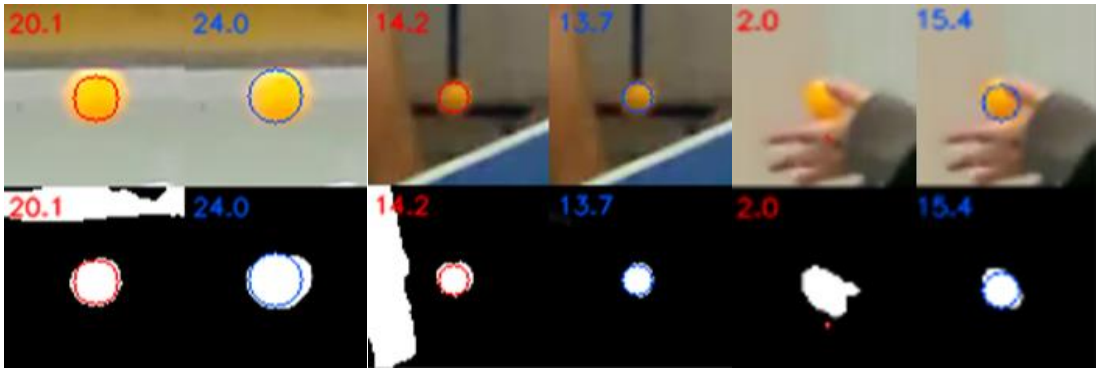
Disappear point 主要是處理球在得分後在桌外被抓到，Specific area 主要處理當球在完成得分後，可能會在桌內繼續做有方向性的運動，以上兩種狀況可能導致再次進入計分狀態，會使邊界系統與計分演算產生問題。Table 6 為使用上述兩種方法的結果比較。

Table 6: 得分系統準確度(normal 代表使用 disappear point 和 specific area，no disappear point 和 no specific area 分別將一個功能關閉)

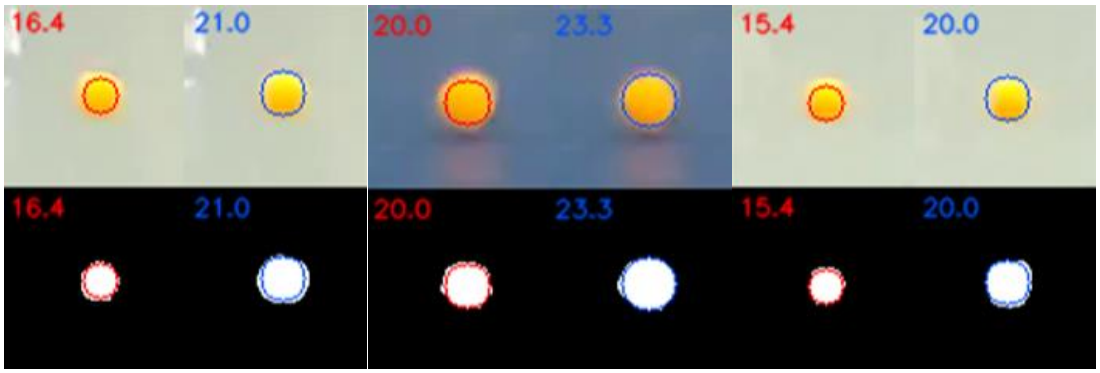
	tp	tn	fp	fn	accuracy	precision	recall
normal	63	0	2	2	0.940299	0.969231	0.969231
no disappear point	49	0	7	2	0.844828	0.875	0.960784
no specific area	34	0	7	3	0.772727	0.829268	0.918919

4.4 圖像分割方法(HSV 色域與 CNN 模型)

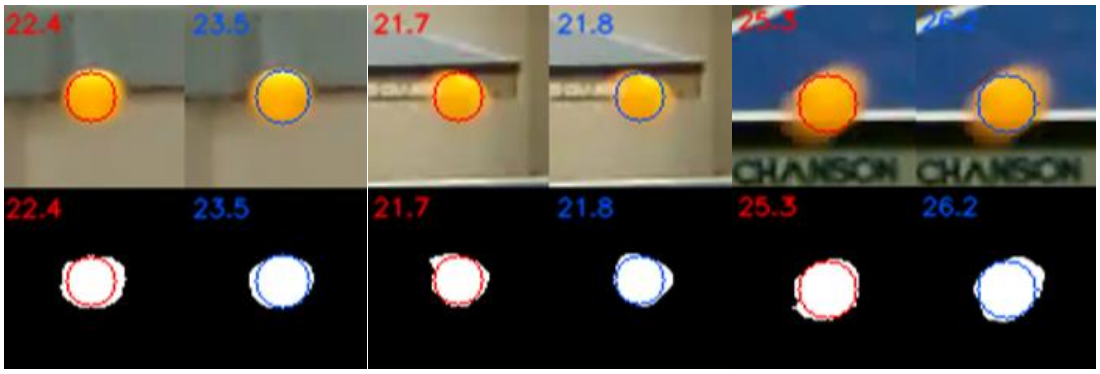
Figure 25 為色域分析(HSV)與 CNN 分割效果的比較，可以發現，在各個情況下，CNN 模型都能有比起 HSV 分割一樣或更好的效果。



(a) 結果雜訊比較



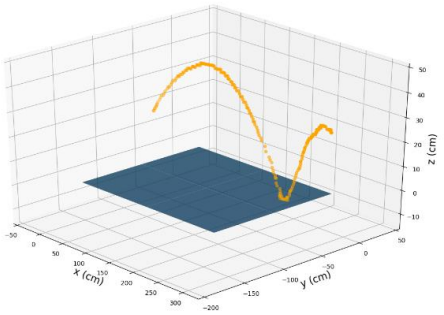
(b) 結果 Mask(內接圓直徑)比較



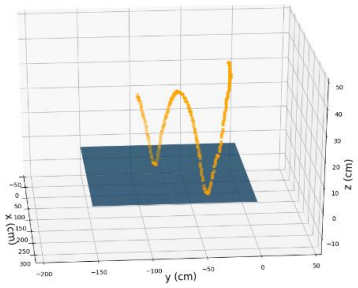
(c) 結果比較

Figure 25: HSV 與 CNN 比較，圖片中左半部(紅色)為 HSV 分析得到的結果，右半部(藍色)為 CNN 模型預測出來的分割結果，每張圖左上角的數字為最大內接圓的直徑(單位:Pixel) (a)再背景顏色相似等情況下，CNN 模型能取得較好的效果且能減少結果的雜訊 (b)CNN 模型能夠更好的預測出球的 Mask (c)HSV 原本能夠處理好的 CNN 模型也能夠處理好。

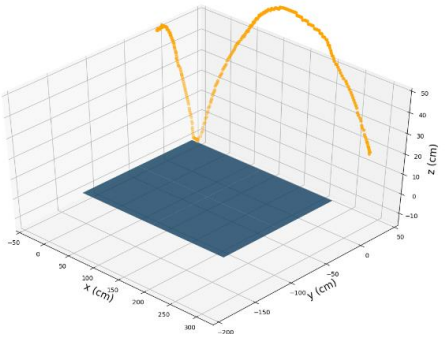
4.5 3D 軌跡還原



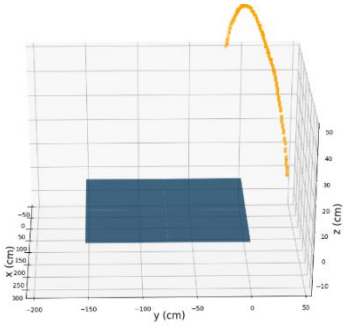
(a)



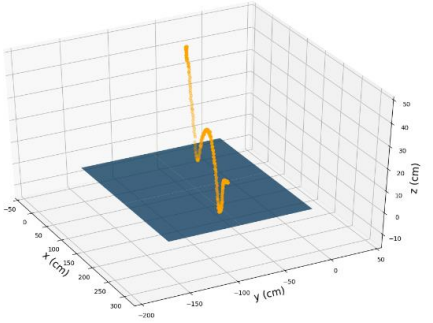
(b)



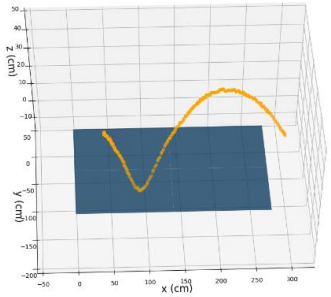
(c)



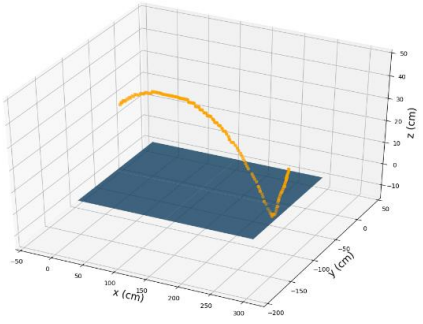
(d)



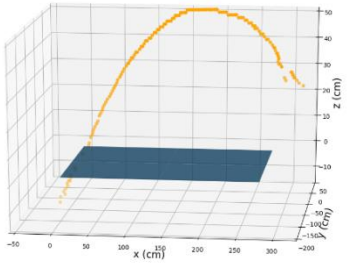
(e)



(f)



(g)



(h)

Figure 26: (a)~(h)為不同軌跡經過分析後得到的 3D 軌跡預測圖，大部分結果視覺上都呈現的不錯，但在沒有落球點可以參考的情況下，軌跡有時會不太正確，如(d)與(h)，雖然看似正常，但其實軌跡資訊不太正確。

五、GUI 圖形使用者介面

為了能夠完整的呈現此作品，我們設計了圖形化使用者介面(GUI)，此 GUI 能夠使用包含所有前述的功能，包括軌跡、落球點、計分系統與 3D 軌跡還原。GUI 分為分析模式與回放模式。分析模式如 Figure 27，點選「Video」能夠選擇影片，再點下「Start」就會開始分析影片，並能實時觀看處理進度，視窗會顯示影片與各種訊息，包括球的軌跡、落球點、雙方得分等，而點選「Replay」可以進入回放模式。

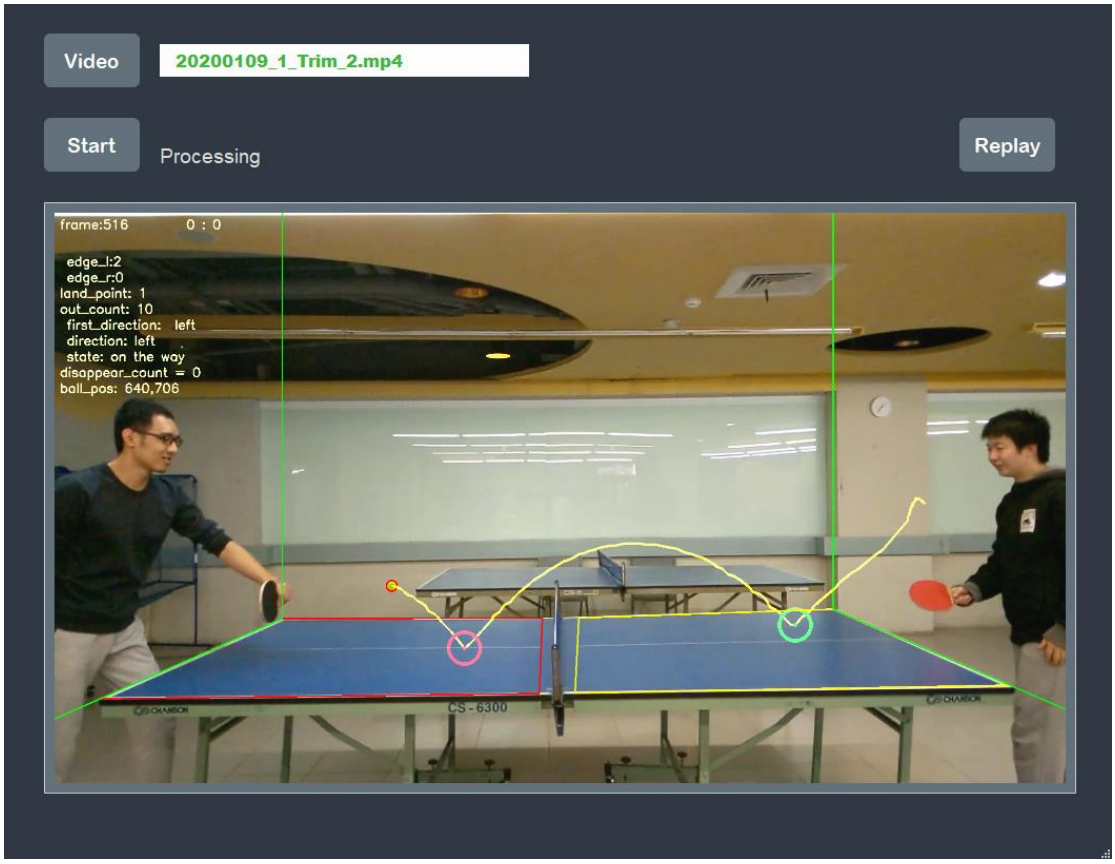


Figure 27: GUI，分析模式，能夠實時觀看處理進度。

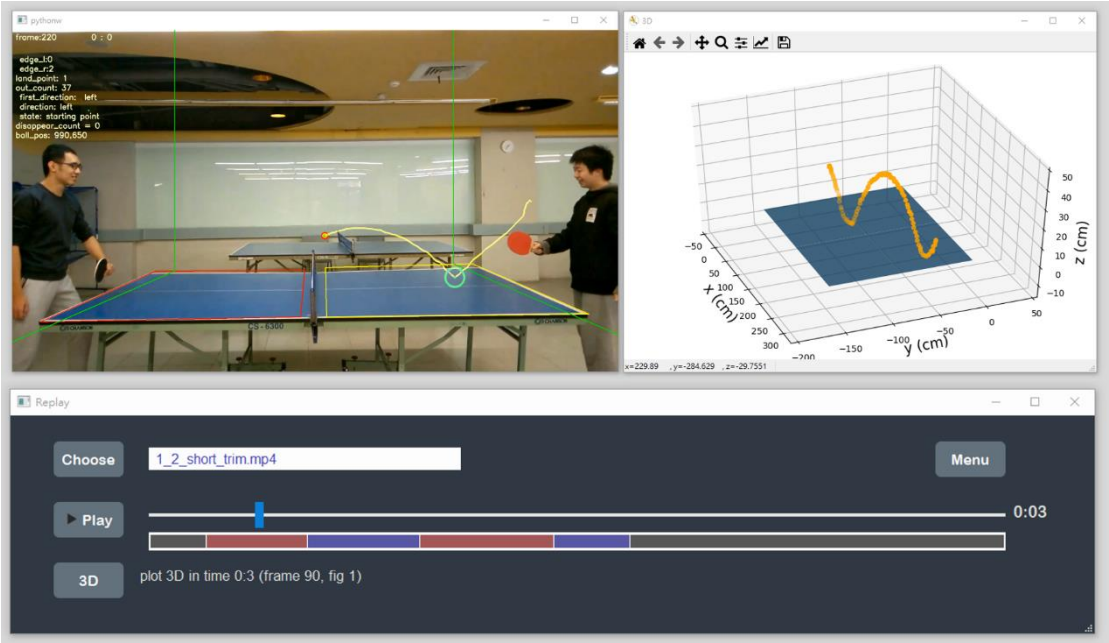


Figure 28: GUI，回放模式，能夠根據時間軸快速選擇要觀看的片段，並展示其 3D 軌跡。

回放模式如 Figure 28，點選「Choose」可以選擇要回放的影片，按下「Play」後系統會自動讀取資訊，播放已經畫上軌跡、落球點等資訊的影片，並加入時間軸，時間軸以不同顏色表示不同狀態，灰色代表此區間沒有球，紅色表示右方擊球(也就是球的行進方向為左)，藍色表示左方擊球(球的行進方向為右)，此時間軸能夠方便使用者快速找到想要回放的區間，接著將滑桿移至想要回放的區間後，點下「3D」便能夠顯示此區間的 3D 軌跡，此 3D 視窗能夠拉動，從不同視角觀看 3D 軌跡。

六、問題討論與未來工作

5.1 偵測不到與零星偵測造成分數誤判

偵測不到: 現有的系統對於觸網後的不規則軌跡偵測能力較差與在球拍上時會有短暫的連續幾幀抓不到球，容易造成計分系統的誤判，目前我們在處理觸

網的部分有兩種方式，第一是多增加了計時器用來計算球在某狀態下持續了多少時間，如果時間超過了就會直接判定此球為觸網，第二是判斷落球點的數量，當落球點個數過多時就會直接判定球已經產生不合法的彈跳了，直接判定此球觸網，剩餘的直接擦網直接導致抓不到的部分只會有 5.6 幀，並不會導致計分的問題。

零星偵測:因偵測系統會抓到整個畫面的球，導致死球狀態時會抓到一些桌外零星的球，這些都會影響到起點的判定，起初我們只使用單幀在指定位置就當作起點，導致計分會出錯，後面我們新增了使用連續幀的方向來判定此球是否適合當作一個起點的開始，效果的確有提升，不過有時候偵測效果過好導致桌外亂彈的球也能產生一個方向性，計分系統也就跟著出錯，最終我們加入了球失蹤計數器，當球還沒進入桌內時，只要球在畫面中消失過久，就會再回到死球狀態，直到球有方向性的進入到桌內，不過這方法會壓低方向的取幀數量，在球拍上時球會抓不太到球，而回到死球模式，等下次抓到時已經開到桌面，所以我們能使用的幀數會被限制。

5.2 速度性能提升

目前的處理速度大約為 20fps 左右(使用的設備為 CPU: AMD R7 3700X，GPU: Nvidia RTX 2080)，由於二值化後的圖片結束後仍存有大量待偵測物件，為了確保桌球會被偵測，霍夫圓偵測閥值範圍無法定義太嚴謹，在較寬鬆的條件下，越容易偵測到更多的圓形，為了辨別是否為桌球，每一個偵測出的圓都必須輸入進去分類器模型內，造成速度大幅下降。目前思考可嘗試解決的方法為 connected component，由於前景部分可分離出正在對打的兩人與移動中的球，所以我們認為利用此方法加上侵蝕或膨脹的調整，可將偵測後物件數量降低，在理想狀況下能夠減少至三個，再輸入至分類器模型內，達到提速的效果。

5.3 2D 落球點限制

因為我們使用的畫面是 2D 的，所以有些落球點的部分會產生瑕疵，像是 Figure 29，Figure 29 (a)在球出桌面後，以 2D 的角度來看很快的球又重新進入了桌面，導致此時又會開始偵測落球點，所以在桌外這個轉折點就被當作了一個落球點，Figure 29 (b)的部分球剛好在很靠近桌邊的地方被擊球者擊飛，在 2D 的狀況下看起來就很像撞擊到桌面而彈起來，所以也產生了落球點，在未來我們可以把我們的 3D 資訊合進來的話，我們就有高度資訊能得知球與桌面的距離，這樣就能更好的決定何時要抓落球點了。

(a)

(b)

Figure 29: 落球點偵測失誤。

5.4 3D 資訊之討論

5.4.1 真實深度資訊

由於真實的 3D 位置必須依靠多組相機或深度相機、光學雷達等儀器才能取得其準確的位置，我們沒有相關的設備，因此目前 3D 軌跡還原的結果，完全依靠肉眼判斷結果的好壞，沒有數據方面的資料可以比較，未來可以加入能夠獲取精確深度資訊的設備，使結果方面的數據能得到驗證。

5.4.2 無落球點的軌跡修正

3D 軌跡在最後作修正時，需要依靠正確的落球點來進行修正，但有許多狀況會造成落球點不為一次或兩次(像是界外球不會產生落球點)，此時資料無法得到修正，3D 軌跡就會產生相比於真實軌跡一定的偏移，而可行的修正方式是使用更高解析度的相機(本文使用的相機解析度為 1920*1080 pixel)。

七、結論

在此專題中，我們成功利用影像辨識技術設計出一個桌球自動計分系統，結合神經網路的使用，從單相機拍攝的影片中還原出深度資訊，並呈現出桌球的 3D 軌跡圖，最終整合在 GUI 中，讓使用者方便操作，在回放影片時可以任意移動影片時間軸查看各片段的資訊與 3D 軌跡圖。以往的技術是由感測器配合微處理器實現，而本專題則全由軟體運行，從一個不同的角度切入，雖說未能超越利用硬體設備得出的成果，但能最大程度降低硬體成本，實際測試可獲得不錯的效果，誤差也在合理的範圍內，之後希望可以透過修正與改進上述所提到的問題，改善整個系統，增加運算速度，並降低效能占用。

八、參考資料

[1] Virendra Kumar Yadav, Saumya Batham, Amit Kumar Mallik, Sep 2013, False Circle Detection Algorithm based on Minimum Support Percentage and Euclidean Distance.

[2] Poorna Banerjee Dasgupta, Dec 2015, An Analytical Evaluation of Matricizing Least-Square-Errors Curve Fitting to Support High Performance Computation on Large Datasets.

[3] Luca Bertinetto, Jack Valmadre, Jo˜ao F. Henriques, Andrea Vedaldi Philip H. S. Torr, September 14 2016, Fully-Convolutional Siamese Networks for Object Tracking.

[4] Mark Sandler Andrew Howard Menglong Zhu Andrey Zhmoginov Liang-Chieh Chen Google Inc., March 21 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, Mar 20 2017, Mask R-CNN.

[6] Hawk Eye Innovations ,https://www.hawkeyeinnovations.com/sports/tennis.

[7] N Dalal, B Triggs, July 25 2005, Histograms of Oriented Gradients for Human Detection.

[8] H Bay, T Tuytelaars, L Van Gool, October 31 2006, Surf: Speeded up robust features.

[9] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, Jun 4 2015, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.

[10] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, Dec 8 2015, SSD: Single Shot MultiBox Detector

[11] Joseph Redmon, Ali Farhadi, Apr 8 2018, YOLOv3: An Incremental Improvement.

九、分工表

	蔡永楨	陳煒竣	管謹中	黃裕凱
桌球偵測演算法	✓			✓
落球點偵測演算法	✓			✓
計分系統演算法		✓		
3D 資訊還原演算法			✓	
GUI 設計	✓	✓	✓	✓
書面報告撰寫	✓	✓	✓	✓
海報製作	✓	✓	✓	✓
貢獻比例	**25%**	**25%**	**25%**	**25%**