國立臺灣科技大學

資訊管理系

實務專題成果報告

學號：[B10809044、B10809016、B10809024、B10809026]

[應用 Hololens 2 於撞球運動輔助系統]

學　　生：[許智翔、陳亮憬、江忠晏、陳定言]

指導教授：楊傳凱 博士

中華民國 111 年 05 月 14 日

目錄

圖目錄

1. 摘要

運動科技在近幾年逐漸有抬頭的趨勢，如今的運動競技不僅是選手之間的較量，更是考驗科技對於選手的輔佐程度，善用科技進行輔佐訓練、測量選手狀態等，不僅能夠提升訓練的品質，也能通過不同於以往的策略來進行教學，研發新的比賽策略等。因此我們組做的專題題目為「應用 AR 的撞球輔助系統」，就是為了解決新手對於撞球的進攻、防守策略不清楚，以及無法確定打擊位置的問題，這些問題都能借由輔助系統進行計算，並將計算成果返回到 AR 的顯示畫面，指引選手找到最佳的打擊路徑。

如今的混合實境趨勢明顯在快速增長中，未來的主流移動設備或許會從現在的手機替換成混合實境的裝置，因此透過這個專題，我們學習到關於混合實境的開發知識，以及延伸探索運動科技領域的其他應用。其中輔助系統的計算除了傳統的演算法之外，也可能需要用到人工智慧，去捕捉到更多的變數。 為此，我們需要完成一款 UWP（Universal Windows Platform）應用的開發，並且將它搭載到 Microsoft 的混合實境裝置 Hololens 2 上，選手在穿戴 Hololens 2 進行撞球運動的時候，前方的感測器會感應到臺面上的情 況，並將圖像進行分析傳送到後臺進行演算，最後再將推薦的最佳打擊路線透過 AR 的方式讓選手直觀的看到並且模仿。

2. 系統分析與設計

2.1 系統設計圖

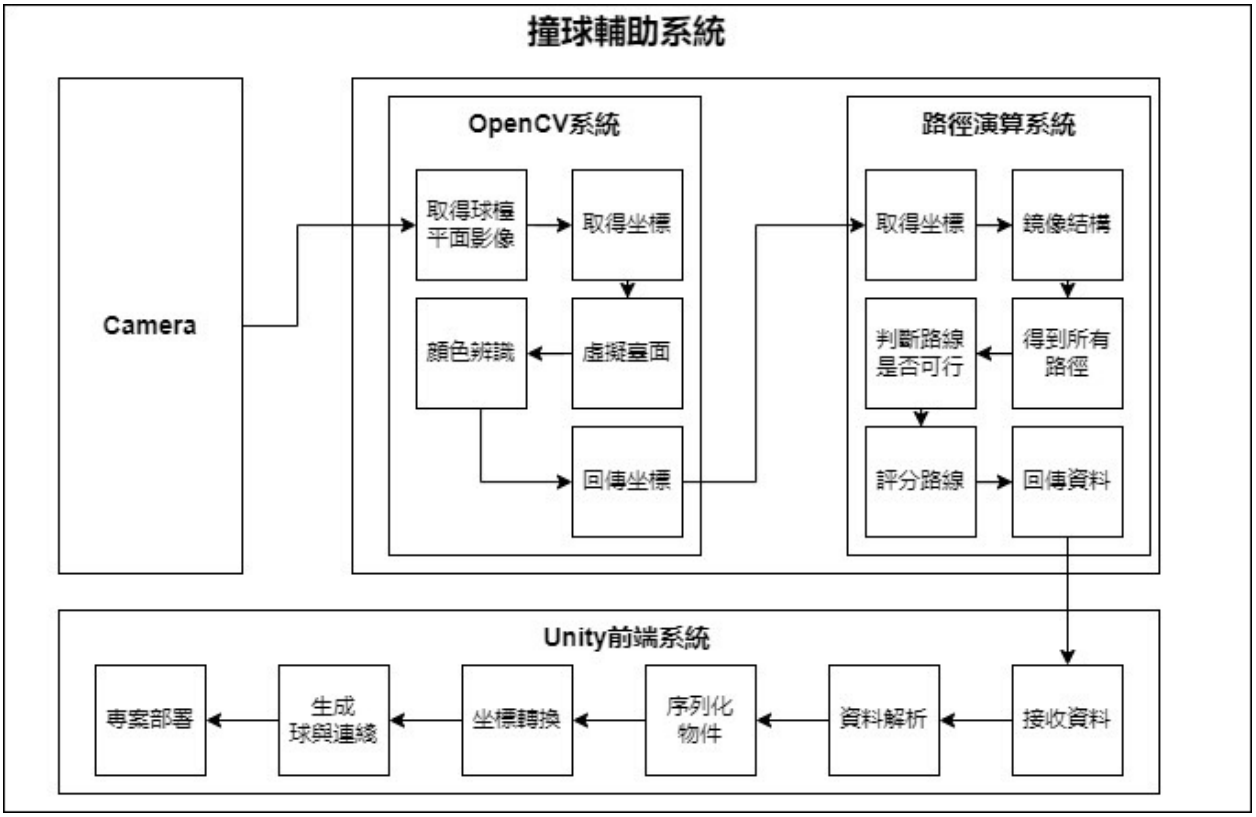

圖 1 系統設計圖

2.2 Hololens2 Unity 前端系統

2.2.1 系統設計圖

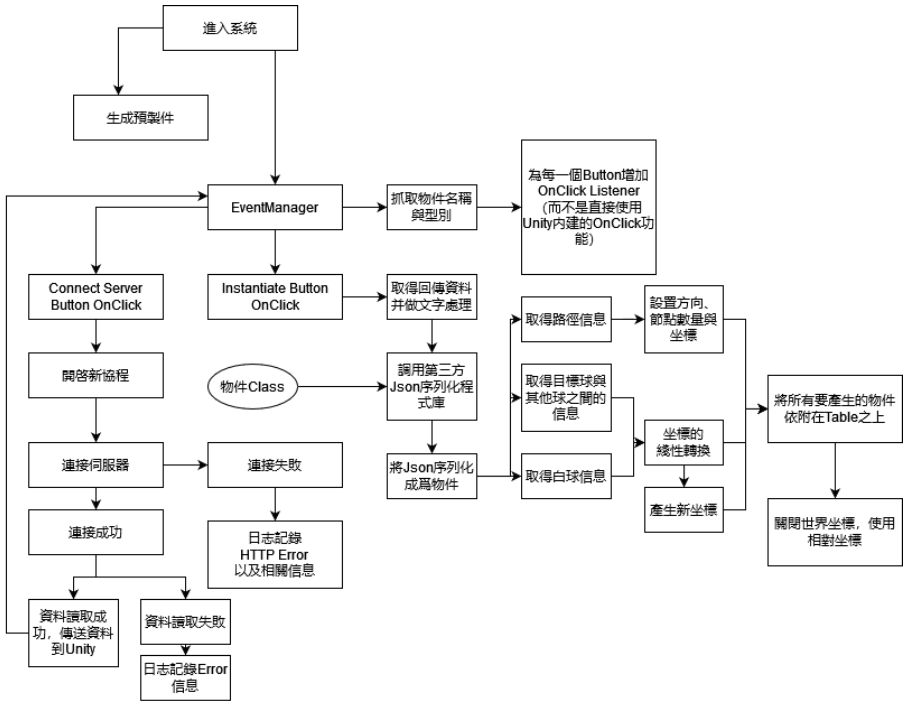

圖 2 Hololens 2 系統設計圖

從系統結構來看，一開始進入系統後，會生成幾個預製件，以及一個空物件（用來當作存放腳本的容器）。在系統啟動的時候，EventManager 會抓取所有畫面上的物件的型別以及名稱，若是形態為“Interactable”之物件（畫面上的 Button 物件，因 Unity 裡面的 Button 型別為 2D 的，若放在系統裡會有 Bug，因此需要用 MRTK 裡面特製的物件組合成一個擁有觸摸回饋的 3D 按鈕），則依據他們的名稱個別附加上 OnClick Listener 。由於 Unity 內建的 OnClick 觸發功能無法直接呼叫開啟協程的方法，因此透過自定義的 OnClick Listener 可以讓使用者按下按鈕時直接進行呼叫。

使用者按下連接伺服器的按鈕後，透過 OnClick Listener 可以觸發方法開啟線程。在使用 Unity 進行開發時，一般不考慮多執行緒（因為在 Unity 中，只能在主執行緒中獲取物體的元件、方法、物件，如果脫離這些，Unity 的很多功能無法實現，那麼多執行緒的存在與否意義就不大了），因此處理一些在主任務之外的需求時，可用 Unity 提供給我們的線程方法。對於線程而言，同一時間只能執行一個線程，它與執行緒最大的不同就在於執行緒是並行的，而線程是序列的，在進行主任務的過程中我們需要一個對資源消耗極大的操作時候，如果在一幀中實現這樣的操作，系統就會變得十分卡頓，這個時候，我們就可以

通過線程，在一定幀內完成該工作的處理，同時不影響主任務的進行。線程是通過迭代器來實現功能的，通過關鍵字 IEnumerator 來定義一個迭代方法，而在迭代器中，最關鍵的是 yield 的使用，這是實現我們線程功能的主要途徑，通過該關鍵方法，可以使得線程的執行暫停、記錄下一次啟動的時間與位置等等，yield 也是指令碼生命週期的一些執行方法，不同的 yield 的方法處於生命週期的不同位置。（可參考下圖）

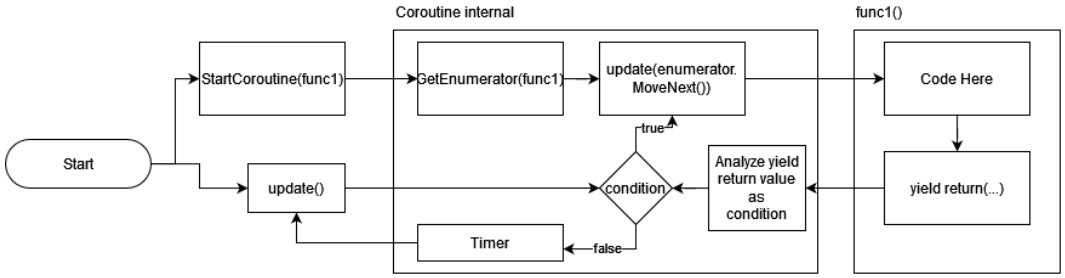

圖 3 線程運作狀態示意圖

線程開啟後與伺服器取得連接，成功取得伺服器回傳的參數后，將資料傳送到 EventManager ，在生成畫面的時候我們就可透過 EventManager 來取得我們需要的資料。

當使用者按下生成的按鈕後，腳本會透過 EventManager 來抓取伺服器回傳的資料（下圖 T Ball 底下的“Cs”一欄）。將資料加以處理後，通過調用第三方函式庫（因 C#原本處理序列化的方式在 Unity 不適用，而 Unity 本身處理序列化的方法有諸多限制，因此決定引用外部的庫）將資料序列化成物件。

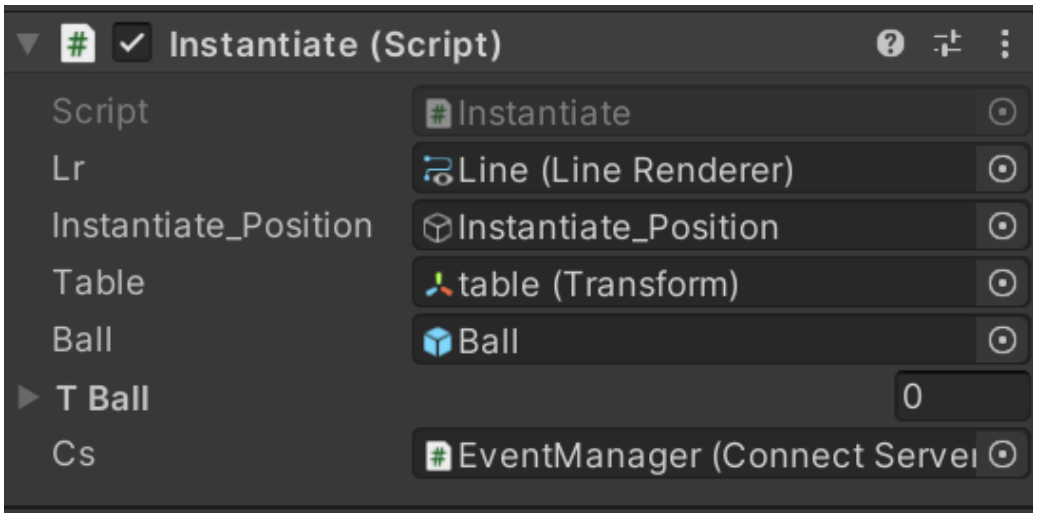

圖 4 不同腳本之間存取變數

透過序列化的物件可得到路徑、子球和母球的信息，通過進行坐標的線性轉換，可將後端傳來的坐標對應到正確的位置並產生新坐標。另外在路徑方面，判斷出路徑有幾個轉折點後，將對應坐標使用 Line Renderer 的方式產生一個帶指向性的方向標來示意擊球點以及擊球后的路徑、碰撞點預測。

因為在 Unity 裡面坐標分為世界坐標以及相對坐標，我們希望產生出來的球以及路線會落在桌面正確的位置，因此要將所有產生的物件依附在桌面上使其成為子物件，并將坐標信息標示為相對坐標，這樣使用者看到的畫面才會是正確的。

2.2.2Unity 前端系統

對於前端的部分，想要取得後端計算出來的參數需要和後端伺服器進行溝通，通過發送 HTTP Request（Create a Web Request object），

向後端發送請求，若是成功後端會發送一個 JSON 格式的 raw 文件，再通過 Download Handler 將取得的 JSON 儲存下來，並序列化成為物件以更方便前端系統利用。

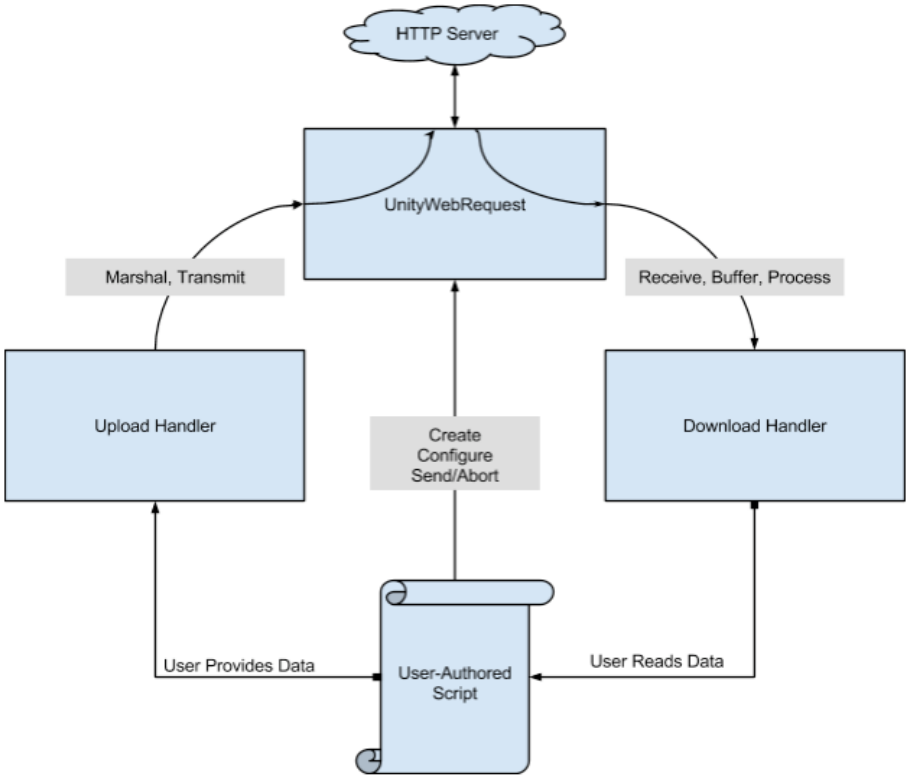

圖 5 Unity 內建 HTTP Request 工作模式

2.2.3開發系統環境

軟體：Unity version-2020.3.19f1

1. Target Device: HoloLens
2. Architecture: ARM64
3. Build Type: D3D Project
4. Target SDK Version: 10.0.19041.0
5. Visual Studio Version: Visual Studio 2019
6. Build configuration: Release
7. Windows 組建支援 (IL2CPP)
8. Mixed Reality OpenXR

軟體：Visual Studio2019

1. .NET 桌面開發

2. 使用 C++ 的傳統型開發
3. 通用 Windows 平台 (UWP) 開發
4. Windows 10 SDK 10.0.19041.0

軟體：HoloLens 2 Emulator (Windows Holographic)

軟體：Mixed Reality 功能工具

1. .NET 5.0
2. Mixed Reality Toolkit Foundation
3. Mixed Reality OpenXR 外掛程式

2.2.4運行系統環境

Microsoft HoloLens 2 是未連線的全像攝影電腦。 它精簡 HoloLens (第 1 代) 開始的全像攝影運算旅程，以提供更熟悉且沉浸式的體驗，並搭配更多選項在混合實境中共同作業。 HoloLens 2 會在 Windows 全像攝影作業系統上執行，其以 Windows 10 的「類別」為基礎，為使用者提供、系統管理員和開發人員提供健全、高效能且安全的平臺。

系統：Windows Holographic Operating System

2.2.5系統功能與介面說明

2.2.5.1 流程

圖 6 系統流程

此系統是撞球的 AR 輔助系統，以使用者來說，最可能使用的場景就是在玩撞球的過程中使用此系統，因此，過於頻繁和複雜的操作除了會打斷玩家的思考，還可能因此打斷了玩家剛進入游戲的狀態。因此，此系統的操作應該盡可能地直覺、簡單。

此系統的可供使用者操作的步驟僅有二步：①取得推薦的路線規劃②顯示推薦的路線，在使用者一開始進入此系統時，需要先連接上我們的伺服器，因此只需按一下畫面中連接按鈕即可取得由我們的後端伺服器算出來的參數，再點擊生成按鈕，即可將路徑顯示。

接下來，使用者可以參考系統推算出來的路徑與擊球點來進行擊球。在這整個過程中首先系統會通過攝像頭取得目前球桌的

畫面，在經過 OpenCV 對桌面的畫面處理後，取得每顆子球的位置、洞口的位置，以及母球的位置，并將參數傳送到路線演算系統，路線演算系統處理後，最終會將最好的路徑規劃傳送至 Hololens2 主機上顯示出來。

2.2.5.2 功能

取得參數-通過點擊連接伺服器按鈕，取得伺服器回傳之參數生成畫面-取得參數過後，將所得到的資訊通過 AR 的方式呈現出來

2.2.5.3 操作與介面

進入系統後先點擊“連接伺服器”，再點擊“生成”按鈕，在遊戲過程中每打一局就需要重複一次，產生的畫面可作為參考畫面，在打擊球的時候建議將頭盔掀起，以避免虛擬畫面和真實畫面重疊，以及頭盔鏡面折射產生的影響。

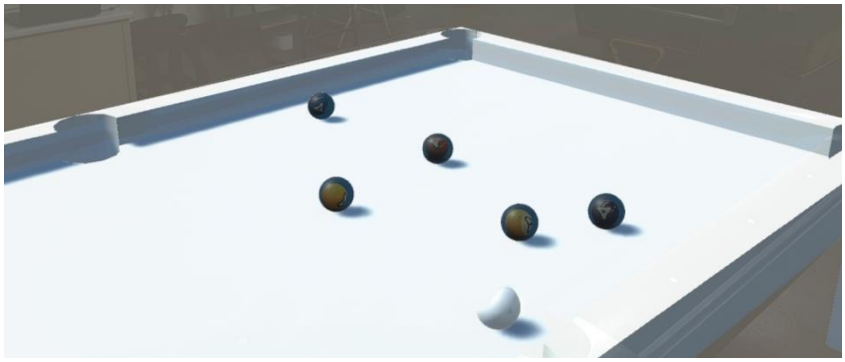

圖 7 虛擬球檯與母子球示意圖

2.3 球桌辨識演算法

2.3.1系統流程圖

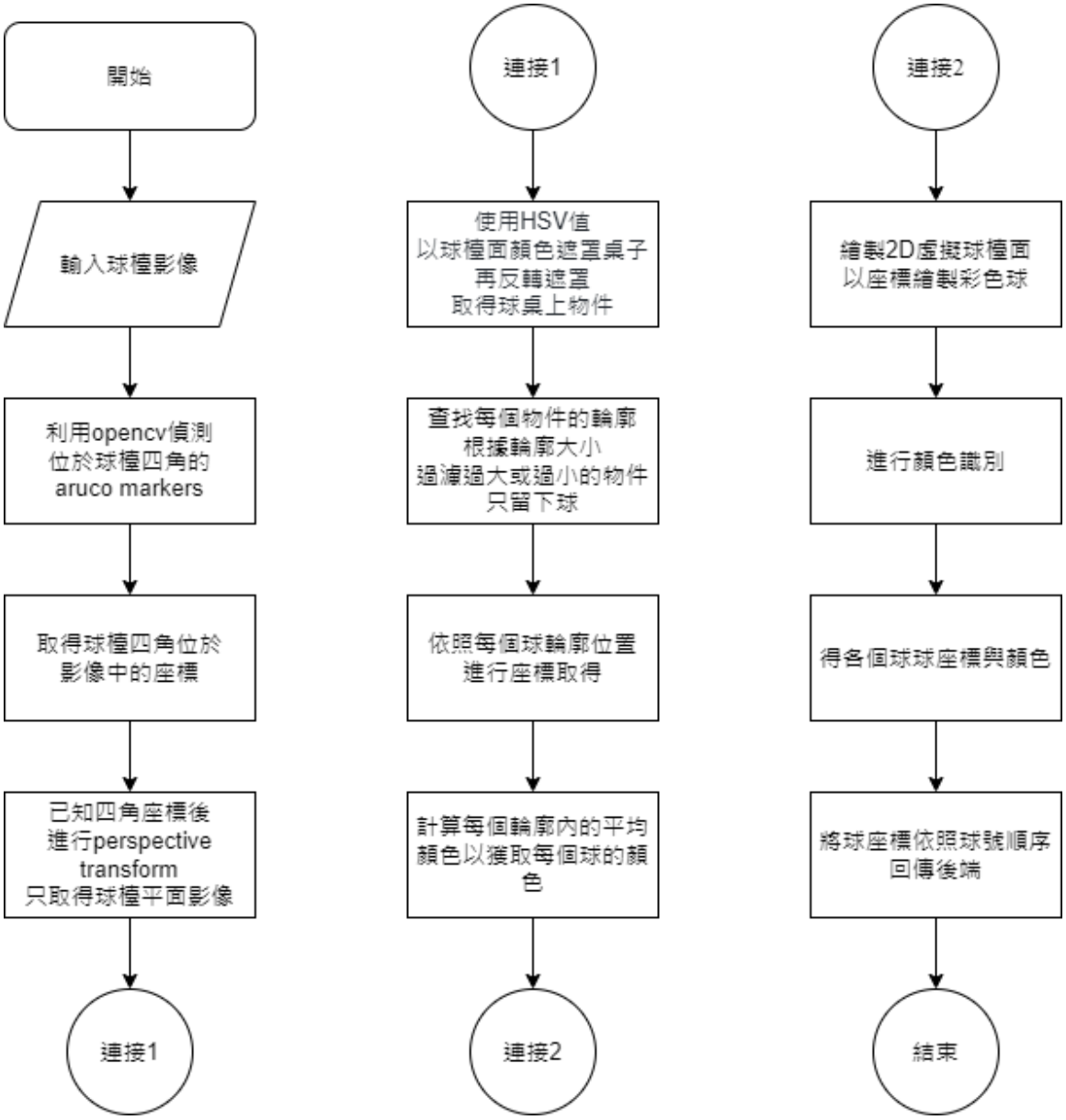

圖 8 系統流程圖

2.3.2球檯處理

由於我們只需要球檯上的畫面，幾號球分別在哪一座標，其他背景我們不需要。所以我們在球檯的四個角放上 Aruco marker，以利我們做球檯定位，知道球檯的四個角在我們影像中的四個座標。

Aruco marker 是由黑色邊框和內部二進制矩陣組成的方形標記，內部的矩陣決定了它們的 id。黑色的邊界有利於快速檢測到圖像，二進制編碼可以驗證 id。Aruco 模塊提供了 Dictionary 類來描述 marker 的字典，字典越小，內部距離就越大。例如:DICT_6X6_250 是一個預定義的字典，它包含 6x6 位的 marker，總共有 250 個 marker。本專題採用 4x4 位的 marker 進行。

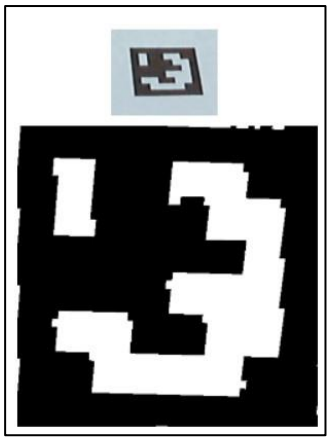

圖 9 Aruco marker 圖像偵測

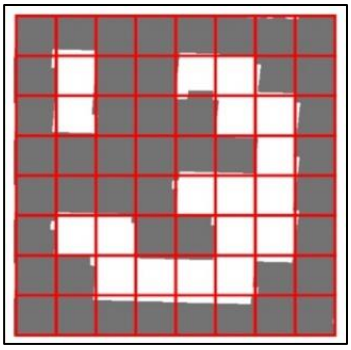

圖 10 Aruco marker 分割

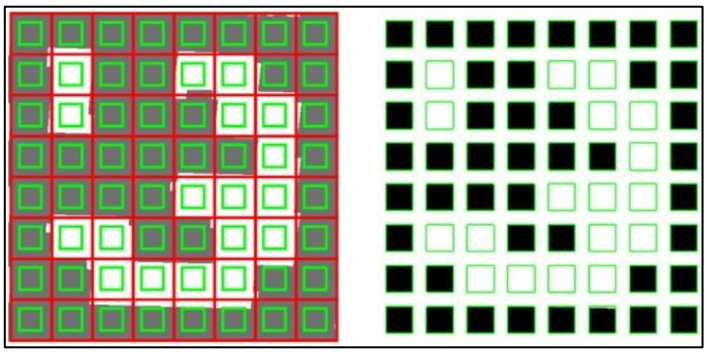

圖 11 Ossu 閾值化以分離白色和黑色位

接下來檢測 Marker，給定一個 Aruco marker 的圖像進行分析，以找到哪些形狀可以被識別為 Markers。

首先是利用自適應性閾值來分割 marker，然後從閾值化的圖像中提取外形輪廓，並且捨棄那些非凸多邊形的，以及那些不是方形的。

檢測完 marker 之後，我們要分析它的內部編碼來確定它們是否確實是 marker。首先提取每個標記的標記位。然後對規範的圖像用 Ossu 閾值化以分離白色和黑色位。根據 marker 大小和邊界大小被分為不同格子，我們統計落在每個格子中的黑白像素數目來決定這是黑色還是白色的位。最終，我們分析這些位數來決定這個 marker 是屬於哪個特定字典的。

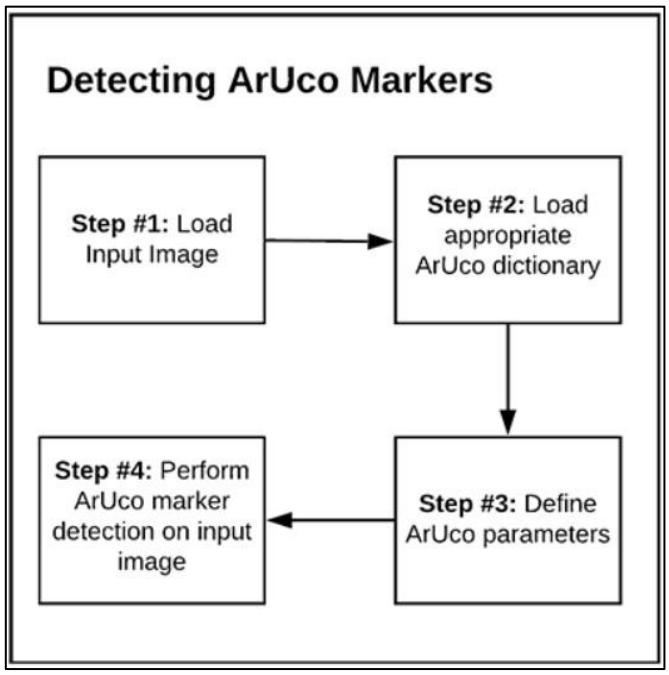

圖 12 Aruco 偵測步驟

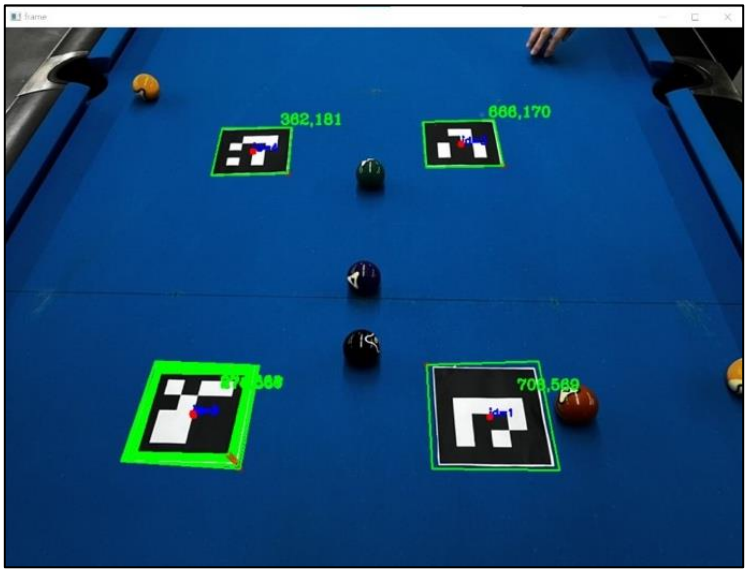

圖 13 偵測測試

再來，由於當我們利用 webcam、手機或者其他影像輸入裝置拍攝球桌時，拍攝的角度不一定是垂直向下，所以我們利用 perspective transform 取得平面的球檯畫面圖。

透視變換(Perspective Transformation)是將圖片投影到一個新的視平面(Viewing Plane)，也稱作投影映射(Projective Mapping)。通用的變換公式為：

$$[x', y', w'] = [u, v, w]\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

（u，v）為原始圖像像素坐標，（x=x'/w'，y=y'/w'）為變換之後的圖像像素坐標。

透視變換矩陣:Transform=$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}=\begin{bmatrix} T_1 & T_2 \\ T_3 & a_{33} \end{bmatrix}$

$T_1 = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$表示圖像線性變換。

$T_2 = [a_{13}\ a_{23}]^T$產生圖像透視變換。

$T_3 = [a_{31}\ a_{32}]$用於圖像平移。

經過透視變換之後的圖片通常不是平行四邊形，重寫之前的變換公式可以得到：

$$x = \frac{x'}{w'} = \frac{a_{11}u + a_{21}v + a_{31}}{a_{13}u + a_{23}v + a_{33}}$$

$$y = \frac{y'}{w'} = \frac{a_{12}u + a_{22}v + a_{32}}{a_{13}u + a_{23}v + a_{33}}$$

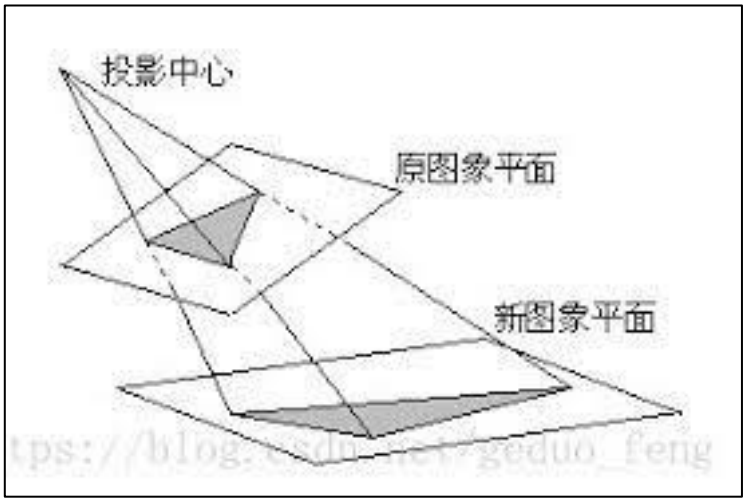

圖 14 透視變換示意圖

所以，給定透視變換對應的四對像素點坐標，即可求得透視變換矩陣；反之，給定透視變換矩陣，即可對圖像或像素點坐標完成透視變換。

圖 15 透視變換前

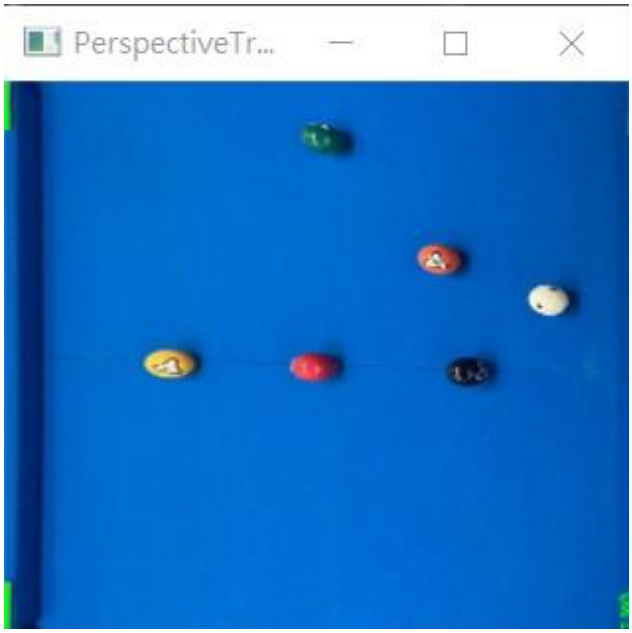

圖 16 透視變換後

2.3.3檯面處理

接下來我們為了要取得各個球的座標以及顏色，我們利用 HSV 色彩空間值把檯面的顏色給遮罩過濾掉(掩模 mask)，再反轉遮罩，這樣就能知道其他球或是其他物件的位子。

所謂的 HSV 即色相、飽和度、明度（Hue, Saturation, Value），色相（H）是色彩的基本屬性，就是平常所說的顏色名稱，如紅色、黃色等。飽和度（S）是指色彩的純度，越高色彩越純，低則逐漸變灰，取 0-100%的數值。明度（V），亮度（L），取 0-100%、越高越亮。所以各個顏色都有其 HSV 範圍值。

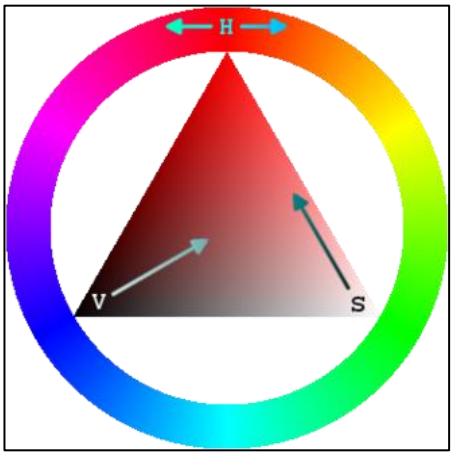

圖 17 HSV 平面圖

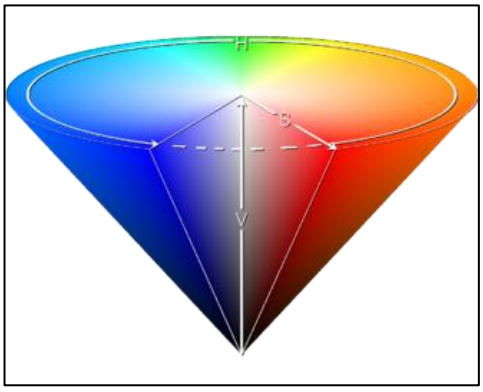

圖 18 HSV 立體圖

※例如藍色閥值:lower:([97,100,117]) upper:([117,255,255])。

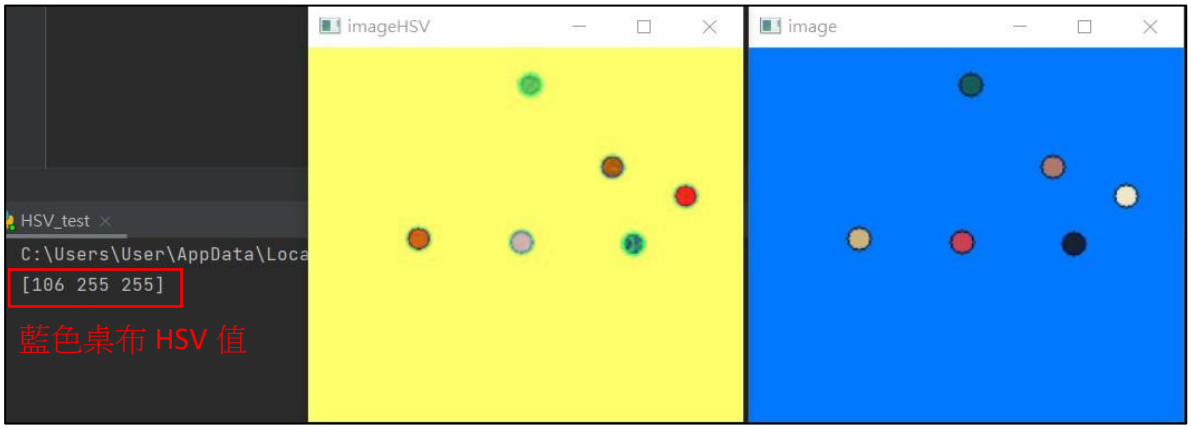

圖 19 檯面藍色桌布 HSV 值測試

我們現在要實現追蹤專題球檯的藍色：首先取得影像模糊化後將顏色空間轉換到 HSV，設定藍色閾值，製作掩膜(mask)，最後對圖像進行掩膜(mask)操作，並反轉遮罩，即可將藍色部分的區域給蓋掉。

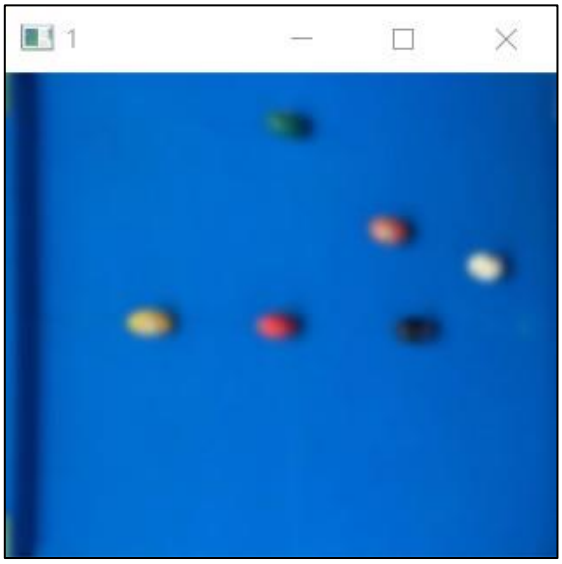

圖 20 高斯模糊

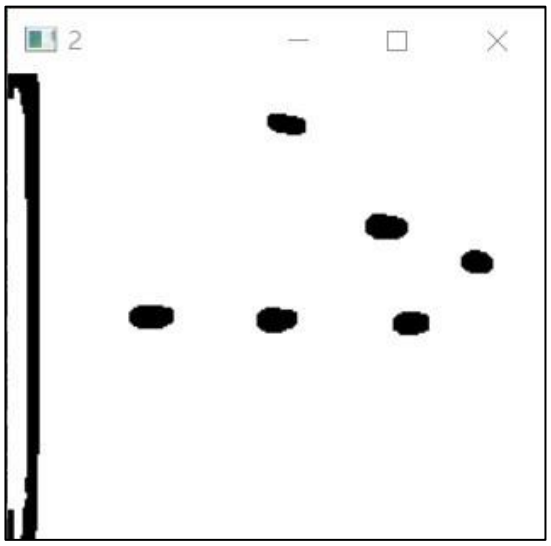

圖 21 檯面遮罩

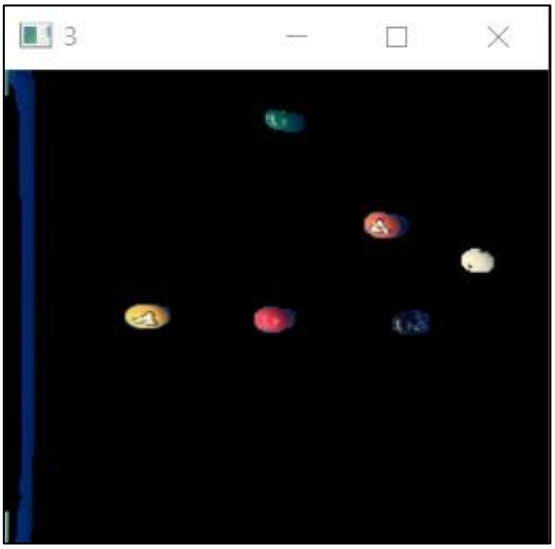

圖 22 反轉遮罩

再來，根據反轉遮罩後把所有物件框起來，我們利用 opencv 的輪廓特徵查找每個物件的輪廓，以及裡用最小外接矩形 cv2.minAreaRect 來框出物件，並把過大過小明顯不是球的大小的矩形給排除掉。

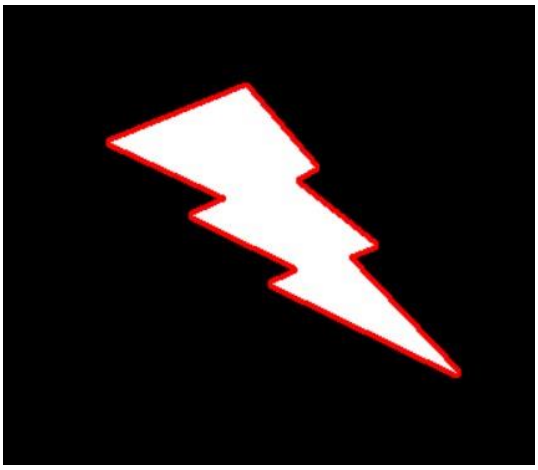

圖 23 特徵矩-圖形輪廓

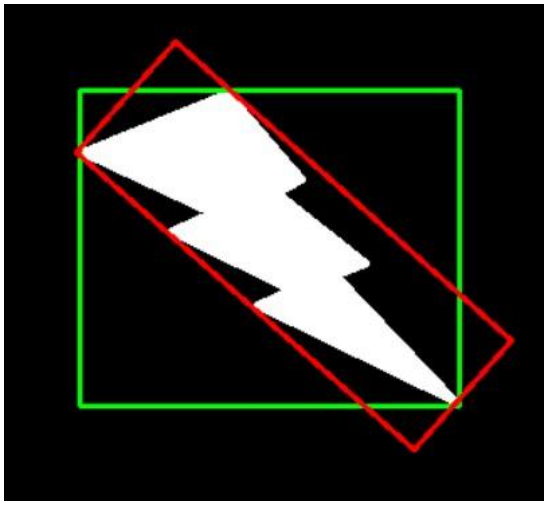

圖 24 外接正矩形與最小外接矩形

綠框代表的是外接正矩形，紅框代表的是最小外接矩形

最後在剩下的矩形中，再次利用輪廓特徵矩計算後得到他們的質心，我們用來當他們的座標 x,y。數學上，「矩」的概念是用來度量一組具有一定形態特點的點陣。如果這些點代表質量，那麼：零階矩表示所有點的質量；一階矩表示質心。而特徵矩可以幫助我們計算一些影像的特徵，例如物體的質心，物體的面積等。

質心:$C_x = \frac{M_{10}}{M_{00}}$，$C_y = \frac{M_{01}}{M_{00}}$。　　零階矩：M00。　一階矩：M10,M01。

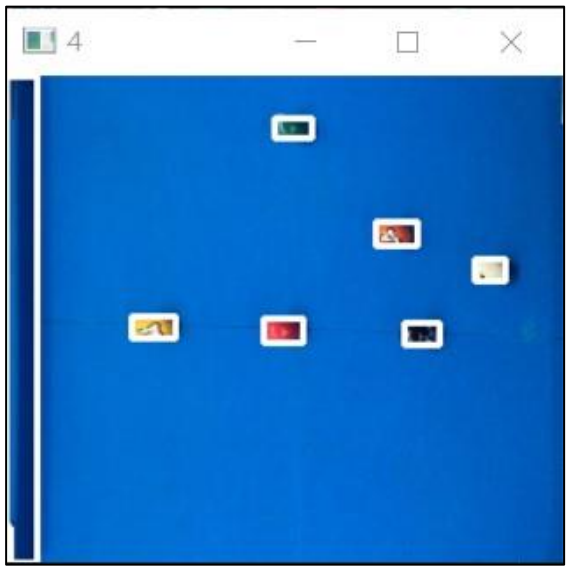

圖 25 查找每個物件的輪廓

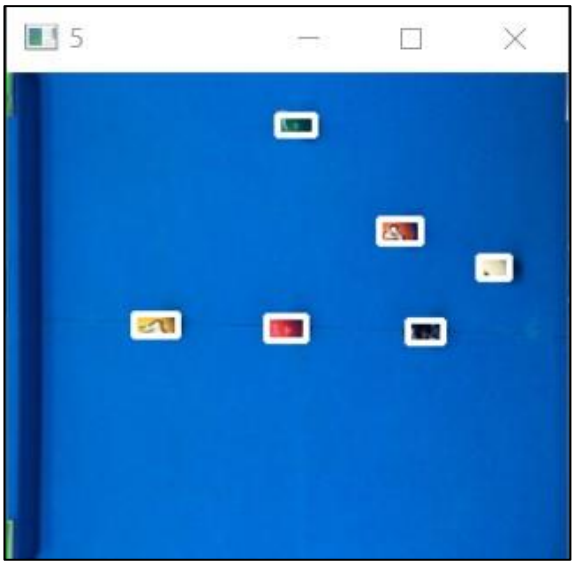

圖 26 過濾輪廓

接著我們利用輪廓內的平均顏色並依照各個球的座標，繪製彩球以及檯面。

圖 27 計算每個輪廓內的平均顏色

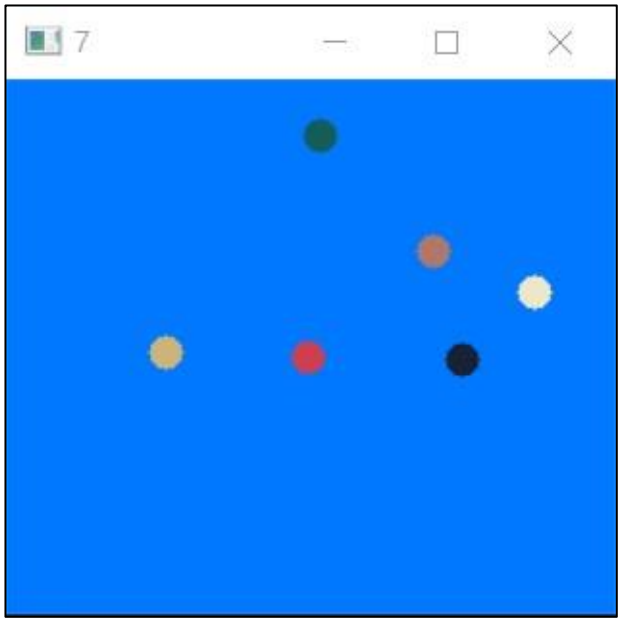

圖 28 繪製 2D 檯面

最後我們利用我們繪製出來的檯面圖，根據 HSV 的閥值判斷球的顏色或是是否存在於檯面上。將我們所得到的球的座標依球號順序傳給後端。

```
白球 [219, 88]
黃球 [66, 113]
紅球 [125, 115]
橘球 [177, 71]
綠球 [130, 23]
黑球 [189, 116]
測試球 [null, null]
{'0': [[219], [88]], '1': [[66], [113]], '3': [[125], [115]], '5': [[177], [71]], '6': [[130], [23]], '8': [[189], [116]]}
```

圖 29 座標結果輸出

2.4 算術撞球演算法

2.4.1 系統流程圖

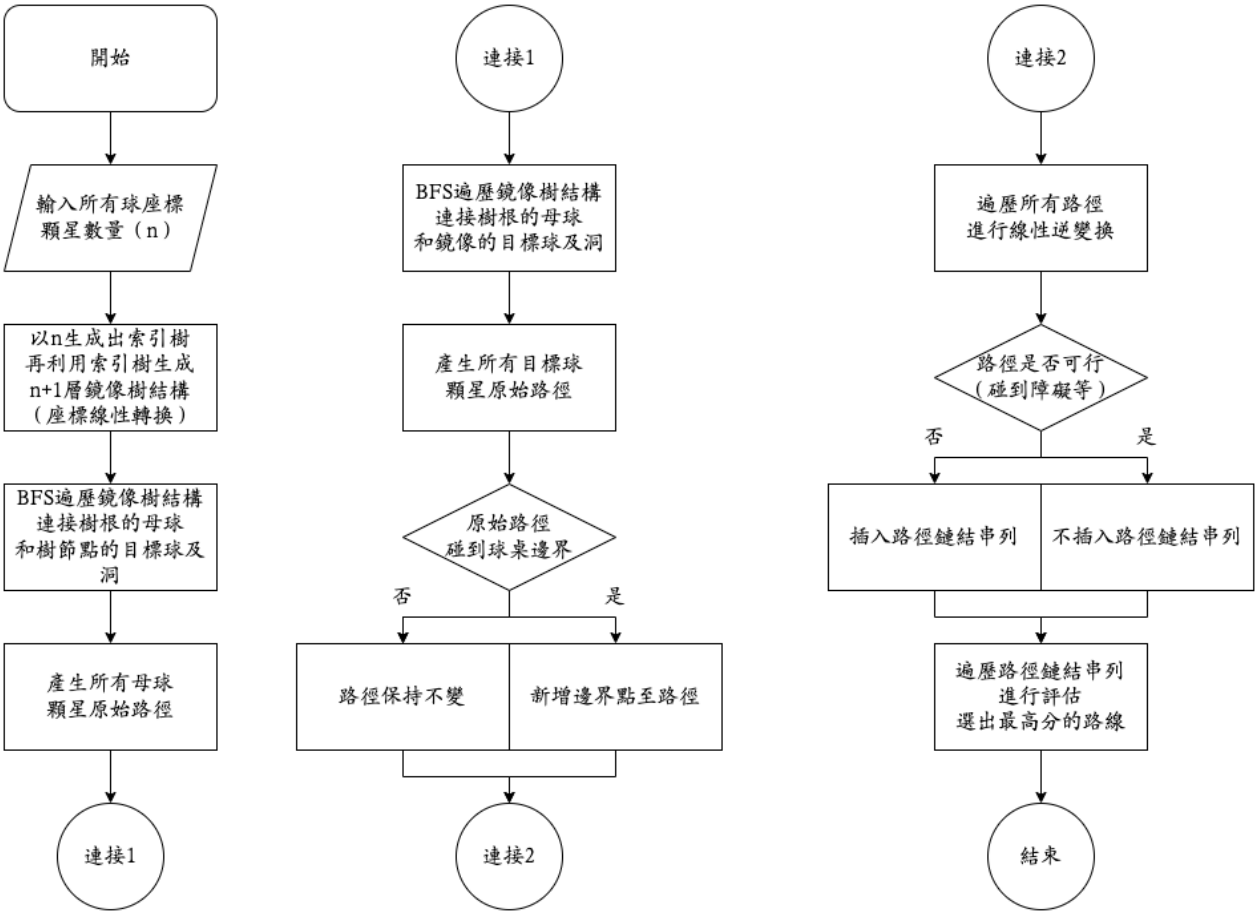

圖 30 系統流程圖

2.4.2 資料結構

在撞球路徑演算法中，本專題設計了一資料結構，用以存取撞球路徑（Path）及移動路徑（Moving List）。首先，由於在演算撞球路徑時，會產生出許多有效及無效路徑，因此在後續判斷是否為有效路徑時，需要進行多次的刪除。由於雙向鏈結串列（Doubly Linked List）新增刪除的時間複雜度為 O(1)，低於陣列（Array）的 O(N)，故採用雙向鏈結串列作為存取撞球路徑的資料結構。

而在雙向鏈結串列中，每個節點會儲存移動路徑的所有點，包含了起始點、瞄準點、目標球心、顆星邊界點（取決於幾顆星）及洞口點，而這些資料大多只需讀取而無需刪改，故採用陣列作為移動路徑的資料結構。

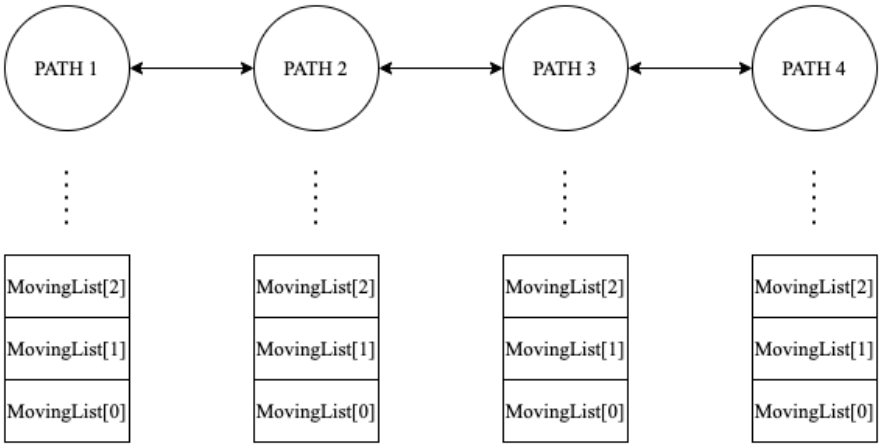

圖 31 撞球路徑資料結構

2.4.3演算法設計

2.4.3.1 索引樹、球桌鏡像樹

基於物理學中的反射定律，擊球點於正中心時，入射角等於反射角，若是要找到一入射角使得母球撞到目標球，讓目標球進洞；或是母球撞到目標球，找到另一入射角使得目標球進洞，並不能透過暴力法（Brute-Force）找尋到正確角度，因為效率十分低。

母球只要打擊鏡像的目標球至鏡像的洞，或是打擊目標球至鏡像的洞，便能經由顆星進洞。只要將這段包含鏡像的路徑經由線性逆變換，便可得知顆星移動路徑如圖 34，亦可計算出入射角。

在產生索引樹時，需要避免生成重複的樹節點，此舉得以減少記憶體的浪費，像是$N_{r,r}$會生成$N_{r+1,r}$、$N_{r-1,r}$、$N_{r,r+1}$及$N_{r,r-1}$四個子節點，隨後在$N_{r+1,r}$生成子節點時需避免生成$N_{r,r}$，以此類推，如圖 35。

因此，索引樹是球桌鏡像樹結構的對照，遍歷索引樹節點$N_{i,j}$，便能透過線性變換生成出對應的球桌鏡像樹節點$T_{i,j}$，而球桌鏡像樹則是此演算法之基石。

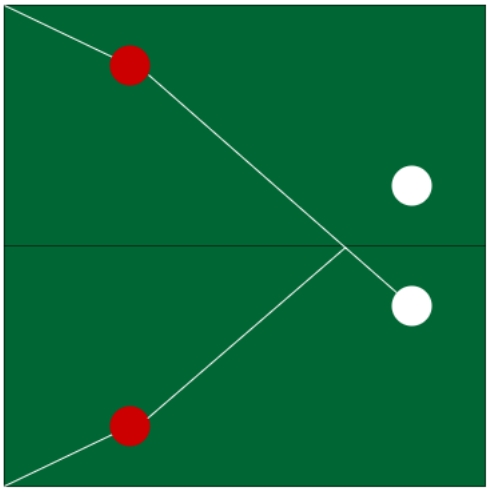

圖 32 顆星移動路徑

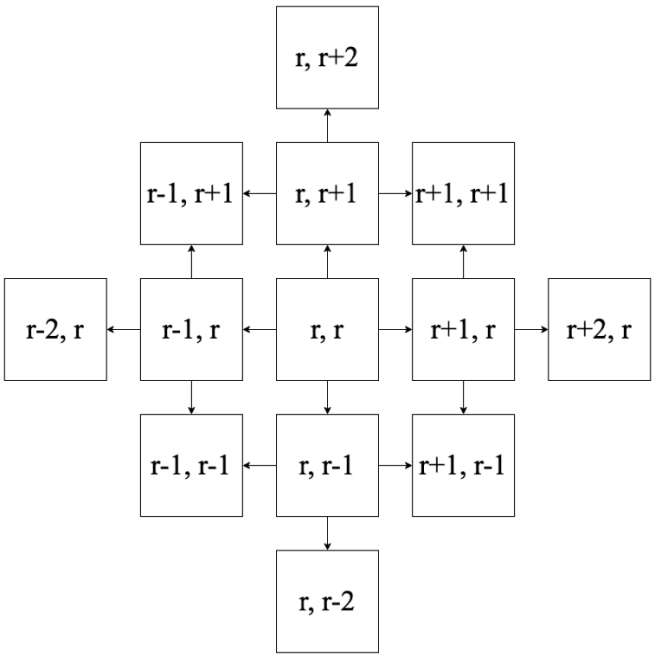

圖 33 索引樹

2.4.3.2 線性變換及逆變換

我們透過線性代數及索引樹和球桌鏡像樹的關聯，推導出線性變換及逆變換的通式。在生成索引樹後，我們得知根節點索引為$N_{r,r}$，其子節點索引為$N_{r+1,r}$、$N_{r-1,r}$、$N_{r,r+1}$及$N_{r,r-1}$，分別代表右側鏡像、左側鏡像、上側鏡像及下側鏡像的索引值。

首先，X 軸反射矩陣為$M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$、Y 軸反射矩陣為$M_y = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$，若是要將$T_{r,r}$轉換成$T_{i,j}$時，要先計算索引差值$D_{(2*1)} = N_{i,j} - N_{r,r}$，才能得知經過了幾次水平反射、垂直反射，而每兩次反射就會轉換為本身，因此找出最少變換次數$C_x = \left|D_{(0,1)} mod 2\right|$、$C_y = \left|D_{(0,0)} mod 2\right|$，以減少矩陣的運算量。

如果$C_x \neq 0$，代表著球桌為 Y 軸鏡像，則$D = D + \begin{bmatrix} 0 \\ 1 \end{bmatrix}$；而如果$C_y \neq 0$，代表著球桌為 X 軸鏡像，則$D = D + \begin{bmatrix} 1 \\ 0 \end{bmatrix}$，$D$作為後續偏差平移的計算。

假設撞球桌的參數矩陣$S = \begin{bmatrix} l & 0 \\ 0 & w \end{bmatrix}, l = 長, w = 寬$，可以計算出偏移量為$S * T$，最後亦可推導出線性變換的公式為（2-1），指的是經過C_x次的 Y 軸反射、C_y次的 X 軸反射再加上偏移量。

$$T_{i,j} = M_{\mathrm{x}}^{C_{\mathrm{x}}} * M_{\mathrm{y}}^{C_{\mathrm{y}}} * T_{r,r} + S^T * D \qquad (2\text{-}1)$$

此外，經由線性代數的推導，我們也可以得知線性逆變換的公式為（2-2），演算的過程如同線性變換，不過唯一不同的是$N_{i,j}$，我們必須藉由座標以計算其處在的索引位置，得出$N_{i,j} = \begin{bmatrix} \left\lfloor \frac{x}{l} \right\rfloor \\ \left\lfloor \frac{y}{w} \right\rfloor \end{bmatrix}$，以利後續將移動路徑轉回至根節點（原球桌的座標系）。

$$T_{r,r} = M_{\mathrm{y}}^{-C_{\mathrm{y}}} * M_{\mathrm{x}}^{-C_{\mathrm{x}}} * \left(T_{i,j} - S^T * D\right) \qquad (2\text{-}2)$$

有了以上兩個公式及索引樹結構，我們可以紀錄根節點的索引$N_{r,r}$，並使用廣度優先搜尋任意的索引$N_{i,j}$，透過上述的線性變換將$T_{r,r}$轉換至$T_{i,j}$，生成球桌鏡像樹的結構。

2.4.3.3 瞄準點

只要連接目標球與洞，從目標球的球心延伸兩倍球的半徑，便是瞄準點。以圖 36 舉例，已知洞的座標為$f = (x_f, y_f)$，目標球的座標為$s = (x_s, y_s)$，進而推導出座標差為$d = f - s$、角度$\theta = tan^{-1}\left(\frac{d_{(1,0)}}{d_{(0,0)}}\right)$，不過當$d_{(0,0)} = 0$時，代表這條線是垂直的，故$\theta = \frac{\pi}{2}$。

此外，必須考慮目標球分別在洞的左上、右上、左下及右下的情況，因此經由分別探討後得出瞄準點的公式為（2-3），當$d_{(0,0)} > 0 \leftrightarrow c_x = -1, d_{(1,0)} > 0 \leftrightarrow c_y = -1$，反之$c_x, c_y = 1$。

$$s + \begin{bmatrix} c_x * 2 * r * cos(\theta) \\ c_y * 2 * r * sin(\theta) \end{bmatrix} \qquad (2\text{-}3)$$

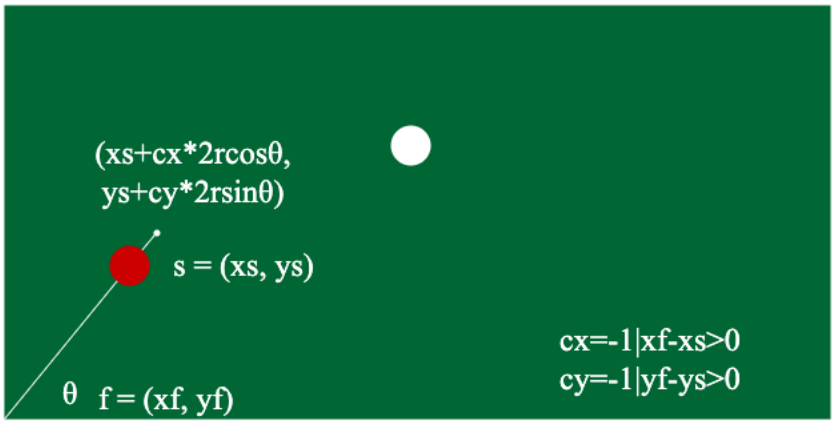

圖 34 撞球瞄準點

2.4.3.4 探討不同情況下的顆星路徑

根據反射定律，只要母球瞄準球桌鏡像中的目標球至球桌鏡像中的洞，就會是母球顆星，如圖 37。反之，若母球瞄準同球桌上的目標球至球桌鏡像中的洞，就會是子球顆星，如圖 38。當理

解上述的顆星概念後，便能夠將此演算法推廣至 n 顆星的情形，只要經由鏡像 n 次，故能得到 n 顆星的路徑。

因此，只需要遍歷球桌鏡像樹的結構，分別連接根節點（原球桌）的母球至子節點（球桌鏡像）中目標球的瞄準點（球桌鏡像中的洞），或是至根節點中目標球的瞄準點（球桌鏡像中的洞），亦可分別演算出子球、母球顆星的路徑。

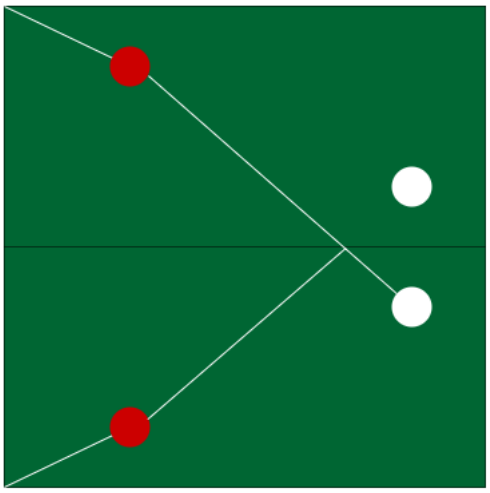

圖 35 母球顆星

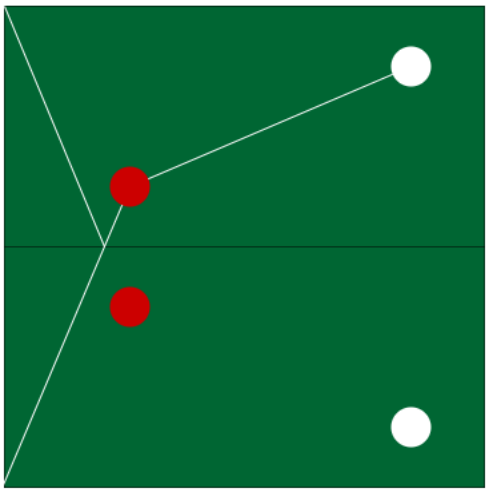

圖 36 子球顆星

2.4.3.5 路徑是否碰到邊界

由於演算出來的原始路徑為直線，但實際的顆星路徑是非連續性的直線，因此需計算原始路徑是否有碰到球桌邊界作為斷點，後續俾能藉由此斷點連接非連續性的直線。

假設球桌的長為 200、寬為 100，直線的開始點$P_s = (150,30)$、結束點$P_e = (550,230)$，線段$\overline{ES}$經過邊界點分別為$(200,55)$、$(290,100)$、$(400,155)$、$(490,200)$，這些斷點將成為顆星路徑的連接點，如圖 39。

這些連接點是開始點與結束點之間的點，而這個點恰好符合長度的倍數，好比150與550之間符合200的倍數為200、400，再將這些點帶進直線方程式找到對應的x、y即可。

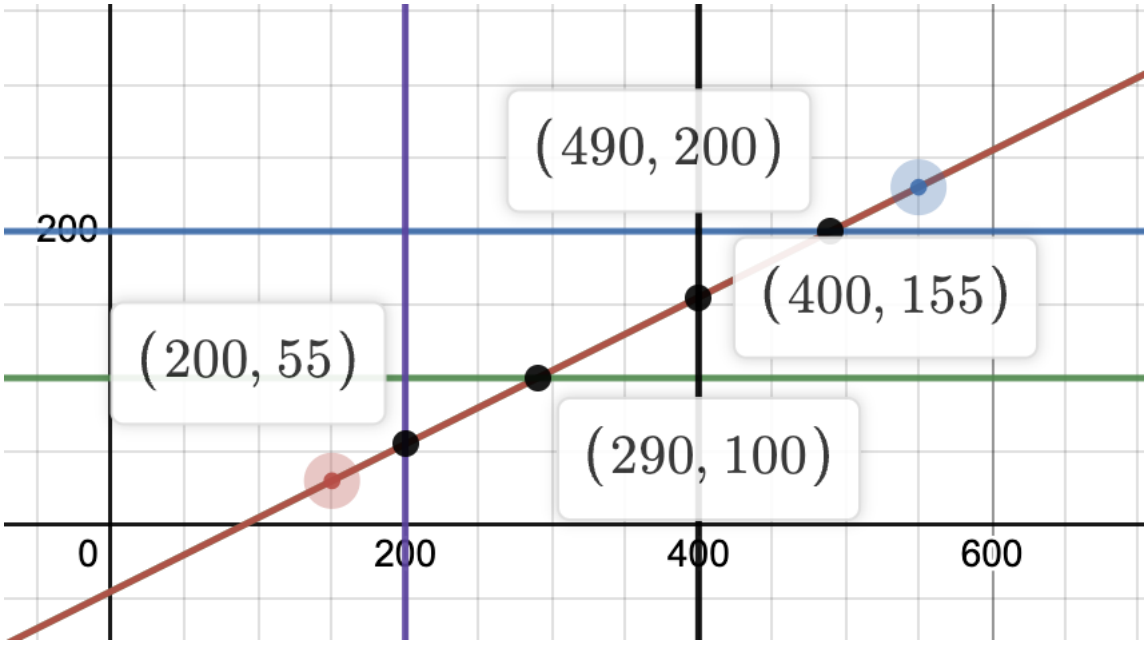

圖 37 路徑是否碰到邊界

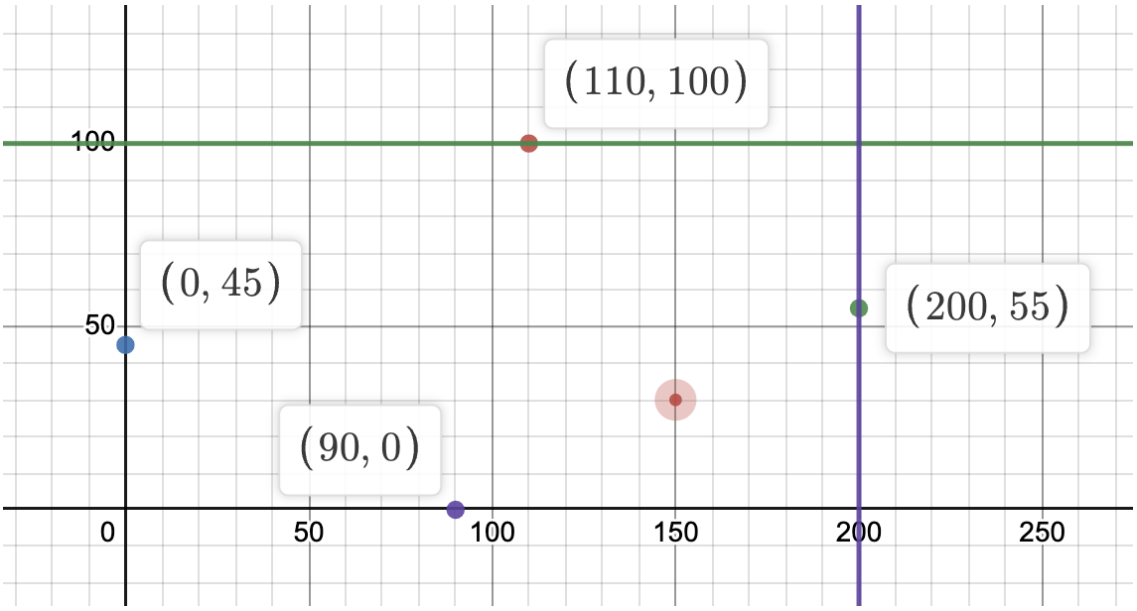

圖 38 經由逆變換後的斷點

2.4.3.6 路徑是否可行

路徑的可行標準主要分作為兩個部分，第一個部分為角度的判斷，第二部分為路徑是否碰到障礙球。

首先，如圖 41 所示，母球會在碰到瞄準點前先碰觸到目標球，以至於目標球無法依照路徑移動，因此我們需要是計算θ為多少，而根據內積公式$\overrightarrow{AB} \cdot \overrightarrow{CA} = \left\|\overrightarrow{AB}\right\|\left\|\overrightarrow{CA}\right\| cos(\theta)$，我們可以推導出$\theta = arccos\frac{\overrightarrow{AB}\cdot\overrightarrow{CA}}{\left\|\overrightarrow{AB}\right\|\left\|\overrightarrow{CA}\right\|}$，若$\theta \geq \frac{\pi}{2}$時，代表此路徑不可行。

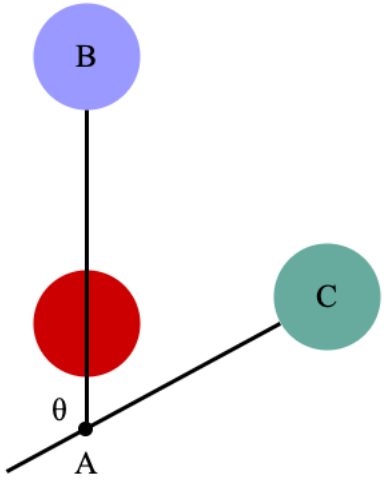

圖 39 路徑角度判斷

再來，我們需要判斷這條路徑是否有碰到其他球，如圖 42，如果有碰到其他球的話，代表母球或目標球無法依照路徑移動，不過判斷路徑是否有碰到其他球，會遠比判斷其他球是否有碰到路徑更加麻煩，因此在此我們會介紹如何判斷其他球是否有碰到路徑。

我們知道可以透過計算點到直線的距離是否小於兩倍球半徑，以判斷其他球是否有碰到此路徑。然而路徑是多條線段組成，若是考慮點到直線的距離，有些不在線段範圍內的球會因為直線的特性（沒有端點、無限長度）而被判斷成小於兩倍球半徑。

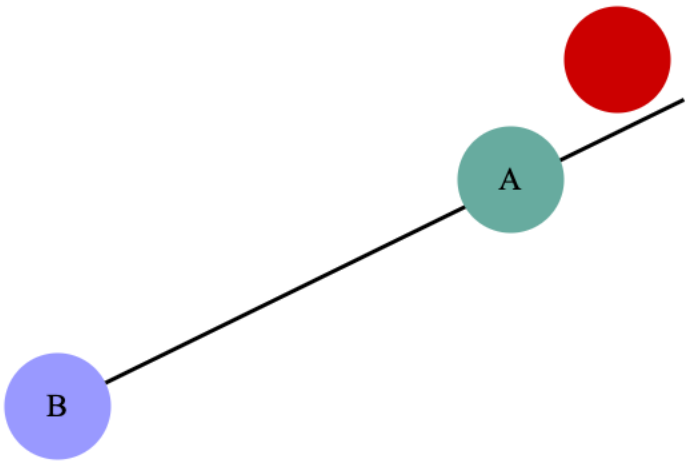

圖 40 路徑是否碰到其他球

因此，我們需要探討如何計算點到線段的距離，圖上的f、g及h分別三種情形為C'、C及C''到線段的長度。首先，計算$\overrightarrow{AC}$於$\overrightarrow{AB}$上的投影向量長，其公式為$\left\|\overrightarrow{AC}\right\| * cos(\theta)$，再計算投影向量長與$\overrightarrow{AB}$長度比例$r = \frac{\left\|\overrightarrow{AC}\right\| * cos(\theta)}{\left\|\overrightarrow{AB}\right\|}$，最後根據正負號以及大小可以計算出點到線段的距離，如 2-4 及圖 43。

$$d = \begin{cases} \left\|\overrightarrow{AC'}\right\|, \text{if } r < 0 \\ \left\|\overrightarrow{AC''}\right\|, \text{if } r > 1 \\ \left\|\overrightarrow{AC}\right\|, \text{otherwise} \end{cases} \quad (2\text{-}4)$$

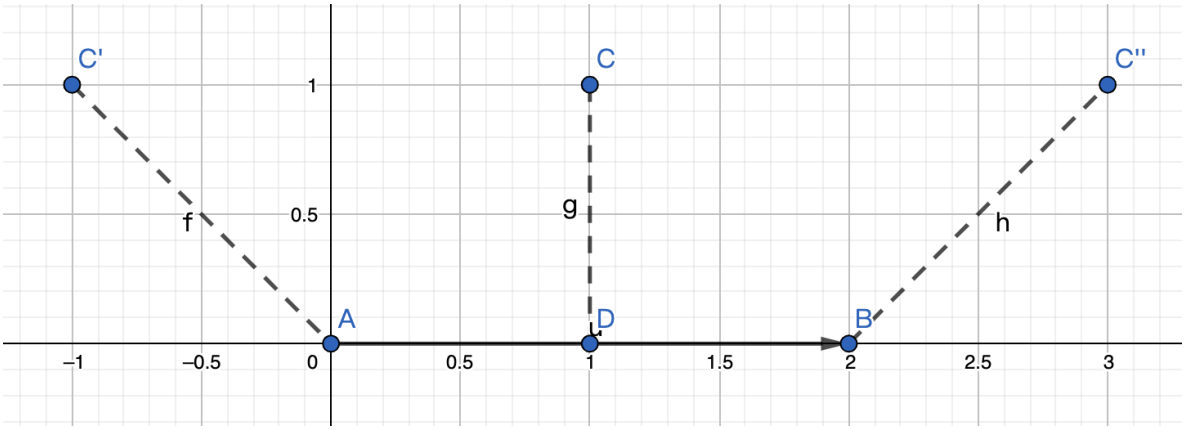

圖 41 點到線段的距離

2.4.3.7 評估演算法

依照上述演算法的流程後，演算出所有的路徑、線性逆變換回原座標系的路徑，最後排除掉不可行的路徑後，剩下的路徑就會是可行的，我們必須在這些路徑中選出一條最好的，給予使用者建議。由於在這個演算法的前提假設是基於物理反射定律，擊球點於正中間，因此變因主要有擊球角度、路徑長度及顆星次數。

首先，擊球角度的評估可以根據動量守恆定律$m_1\overrightarrow{v_1} + m_2\overrightarrow{v_2} = m_1\overrightarrow{v_1'} + m_2\overrightarrow{v_2'}$，如圖 44，找到目標球的動量、母球初動量與角度的關係$\frac{\|\overrightarrow{v_2'}\|}{\|\overrightarrow{v1}\|}/\theta$，這代表著每變動θ會有多少的動量從母球分給目標球。又因為$\overrightarrow{v_1} = \overrightarrow{v_1'} + \overrightarrow{v_2'} \leftrightarrow m_2, \overrightarrow{v_2} = 0 \wedge m_1 = m_2$，因此可以得出$\frac{\|\overrightarrow{v_2'}\|}{\|\overrightarrow{v1}\|} = cos(\theta)$的結論。

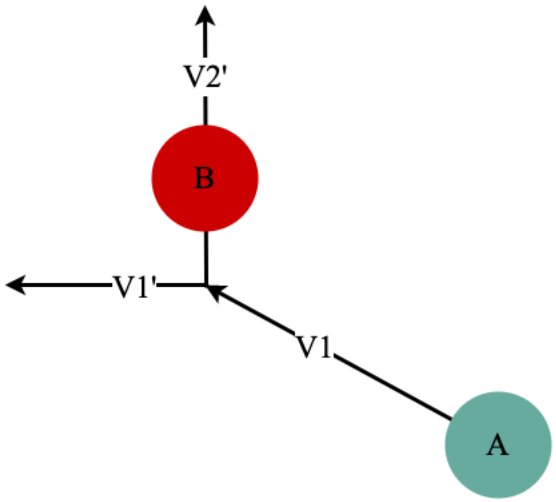

圖 42 動量守恆

再來，路徑長度的評估可以根據速度與距離的關係$\left\|\vec{V_f}^2\right\| = \left\|\vec{V_i}^2\right\| - 2aS$，找到末速度、初速度與距離的關係$\frac{\|\vec{V_f}\|}{\|\vec{V_i}\|}/S$，這代表著每變動$S$會有多少速度比例的變化，$\frac{\|\vec{V_f}\|}{\|\vec{V_i}\|} = \frac{\sqrt{\|\vec{V_i}\|^2 - 2aS}}{\|\vec{V_i}\|} = \sqrt{1 - \frac{2aS}{\|\vec{V_i}\|^2}}$，由於加速度$a$是由球桌摩擦力所決定、$\|\vec{V_i}\|$是由出杆力度所決定，這些變因不好量測，在此認定為常數，因此此時可以推出$\sqrt{1-S}$關係式，但又因為這樣的關係式S會受限於≤ 1，因此必須擴展至n倍球桌對角線長度，最終得出$\frac{\sqrt{n\sqrt{l^2+w^2}-S}}{\sqrt{n\sqrt{l^2+w^2}}}$的關係式，足以表示速度的變化與移動的距離的關聯。

我們推測顆星次數對難度有指數上升的關聯，因此在此假設每經過一次顆星便會有α倍的乘數，其公式為α^n。最後，再將這三個評估係數相乘，便會是最後的評估值，越低代表這條路線越難打，反之則越好打。

2.5 撞球的基因演算法

基因演算法為一種強化學習（Reinforcement Learning）的運用，也是一種演化式計算（Evolutionary Computing），顧名思義此種演算法即是參考達爾文的「進化論」，透過「物競天擇，適者生存」

的演化淘汰觀念所發展的一種演算法。更簡單的來說，每個問題都可以都運用此種演算法找到合適的解，且此解會透過演算法進化，結果一次比一次更佳。

每個存活的生物都擁有細胞，每個細胞擁有相同組合的染色體，染色體則是由 DNA 與蛋白質所組成，而基因（Genes）則特指在 DNA 序列上是能夠表現出特徵的蛋白質，例如：人的黑髮、藍眼睛等。基因演算法所模擬進化論中的「物競天擇」，是指當一個物種本身可以適應當前的環境，便可以順利的存活並將自身的基因繁衍至下個世代(Generations)，意思是該物種的後代（Offspring）能夠保有母體原有好的基因，並表現出其基因的特徵，有利於後代的生存。

而回到撞球遊戲當中，行動空間（Action Space）是離散的，意思是包含母球、子球在球桌上的位置，以及球與球的撞擊點有無限多個組合，所有撞球的可能性幾乎是無法被窮舉的。因此我們除了以人撞球的思維來做撞球演算法設計，也期望運用強化學習不斷嘗試錯誤（Trial-And-Error）的學習方式，讓智慧型代理（Intelligent Agent）透過模擬的撞球模型以及上述的演化式訓練，替每一桿擊球找出不同於傳統的最佳解。

2.5.1基因演算法的架構

2.5.1.1 基因演算法的基礎

遺傳演算法為了將欲求解的問題變數或參數模擬類似染色體的資料結構（Chromosome-Like Data），便以有限長的陣列編碼（Encoding）。每個陣列及代表著不同染色體，而染色體及是我們欲求解問題的解答，而陣列當中的變數則是每個染色體當中的基因，基因代表著不同個體所表現出不同的特徵。在撞球應用當中染色體代表的解答及是如何對母球出杆，而基因則代表每一杆不同的角度、力道等特徵。在編碼完成後，接著應用一些遺傳運算元（Operators）如轉型（Crossover）、突變（Mutation）對大量的染色體做運算，運算後產生的子代除了母體中具優勢的特徵外，也有可能因為基因的交換與突變而比母體表現更佳。

2.5.1.2 搜尋空間（Search）

當我們欲解一個問題，會試著從多個可行解（Feasible Solutions）當中找出一項最好的，而存放所有可行解的空間，及稱

為搜尋空間，在基因演算法當中，則將搜尋空間令稱為群體（Population）。每個搜尋空間的可行解皆可以合適度（Fitness）做標記，並根據合適度找出問題的最佳解。

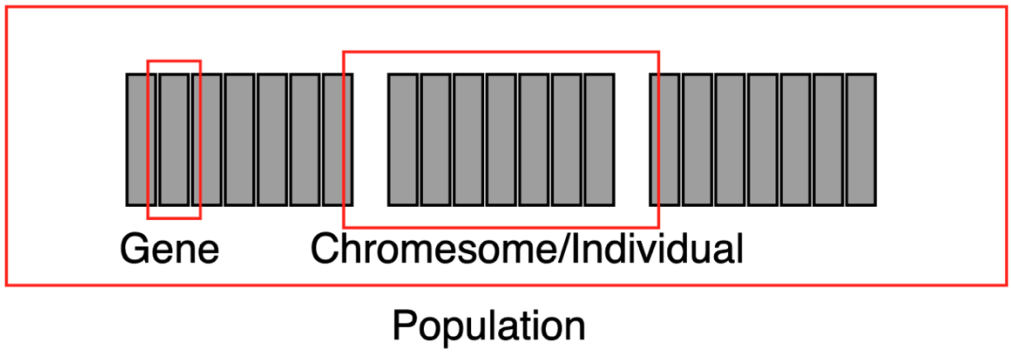

圖 43 搜尋空間

2.5.1.3 合適度評分（Fitness Score）

合適度評分顯示出每一個染色體在環境中所擁有的競爭力，當染色體獲得相對高的合適度評分後，將會從中選擇為母體並進行繁殖，直到新的群體重新被產生，並將繁殖的後代進入下個世代競爭。因此，每個新的世代，將會擁有比前個世代相對更好的染色體，因為每個世代的染色體皆為前個世代所擁有高合適度評分所繁衍的後代，此現象代表著問題的解透過評分，每次都更優於前一次。若當新的世代與前個世代沒有太多差別，代表這個群體已經收斂（Converged），收斂後此解及為問題的最佳解。

2.5.1.4 遺傳運算元（Operators of Genetic Algorithms）

當一個世代的群體被成功產生且合適度評分完成，將會用以下的遺傳運算元來進化群體：

2.5.1.4.1 轉型運算元

在做完合適度評分後，將會從中選擇兩個染色體進行交配，其中的基因將會進行交換，並產生一個新的後代（個體），此過程稱為轉型。

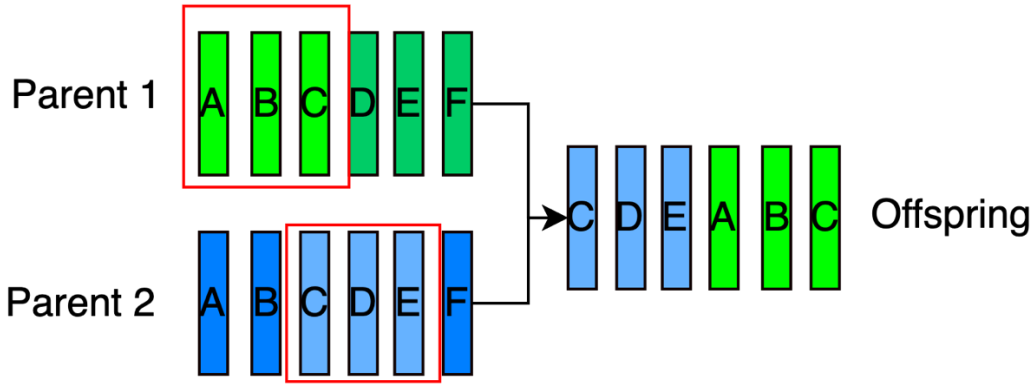

圖 44 轉型運算元

2.5.1.4.2 突變運算元

在產生新的後代後，將隨機的基因插入至染色體，用意是為了維持群體的多樣性，以防止群體過早的收斂。

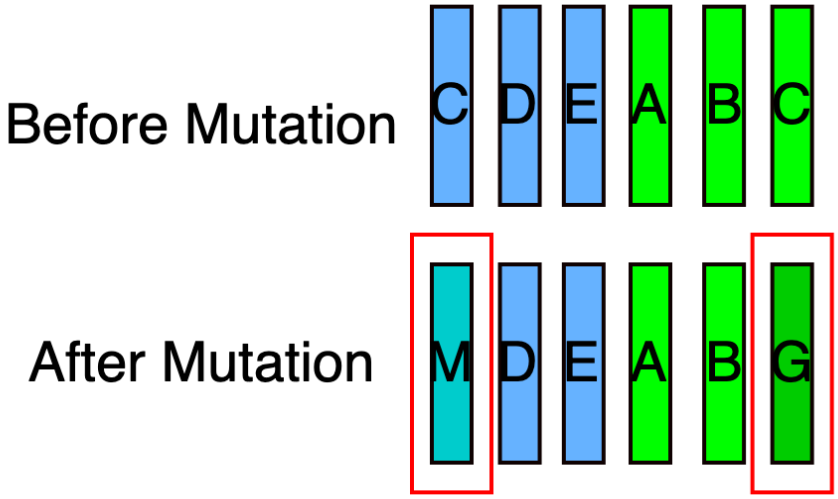

圖 45 突變運算元

2.5.1.5 基因演算法的執行順序

基於以上基因演算法的執行順序可以總結為：

a. 隨機產生新的群體 p

b. 計算出群體中每個染色體的合適度
c. 重複直到收斂：
    i. 根據合適度從群體中選擇母染色體
    ii. 將兩個母染色體轉型直到產生出新的群體
    iii. 對新群體中的後代進行突變
    iv. 計算出新群體中每個染色體的合適度

2.5.2基因演算法在撞球遊戲中的設計

此演算法以物件導向設計，分別將染色體、玩家、群體分別建立為 Genotype, Player, Population 物件。Genotype 為玩家打撞球行為的編碼型態，也是基因演算法中的基因（Gene）；而 Player 及為實際撞球的行為，並且針對每個玩家皆有設置評分；Population 則存放玩家的陣列，並在所有玩家完成擊球後依評分從中挑選進行遺傳運算元的運算。

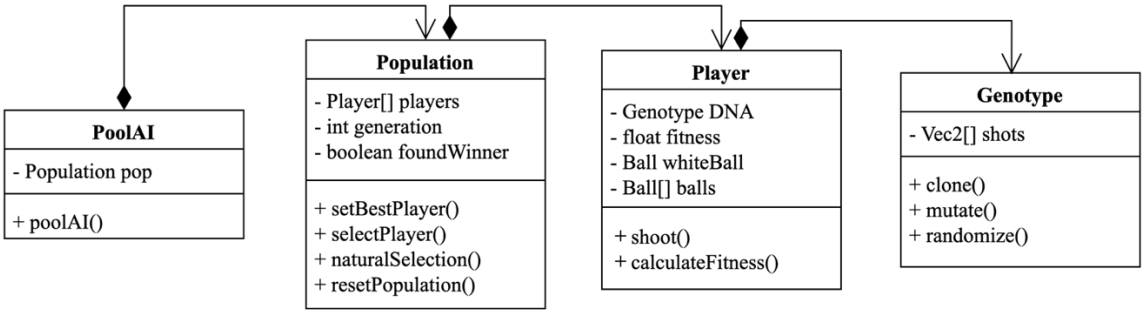

圖 46 基因演算法設計

2.6 系統開發時程

任務名稱	7月	8月	9月	10月	11月	12月	1月	2月	3月	4月	5月
預期											
資料蒐集											
預期											
架構擬定											
預期											
雛形整合											
預期											
程式碼撰寫											
預期											
整合測試											
預期											
文件及報告撰寫											

圖 47 甘特圖

3. 系統實作成果

在進行每一桿擊球前，將先針對球桌狀態進行俯視圖的拍攝，並上傳伺服器。

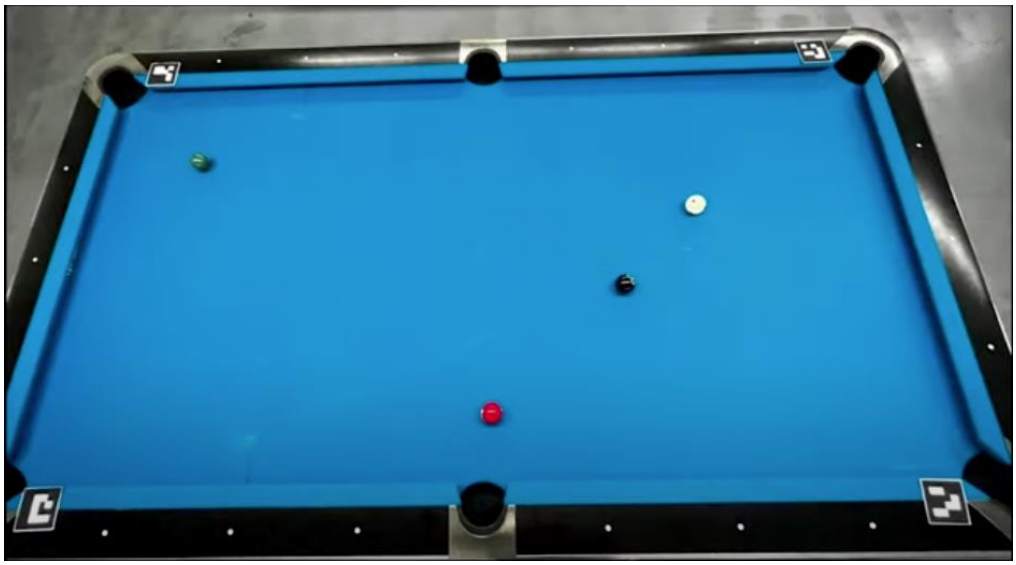

圖 48 球桌俯視圖

拍攝完畢後，回到 Hololens 2 裝置並按下“Create Ball”，將會進行球桌的辨識以及處理。

圖 49 Hololens 操作

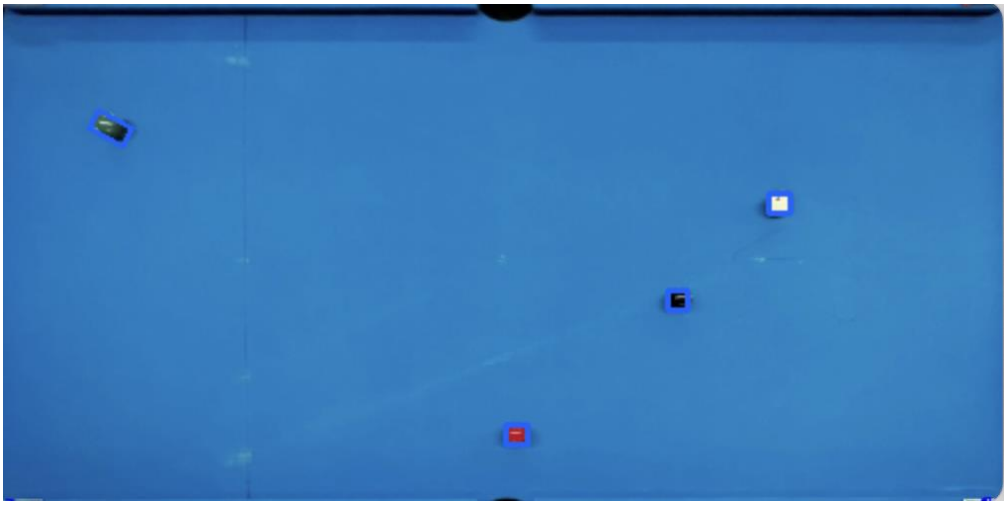

圖 50 球桌辨識及處理 1

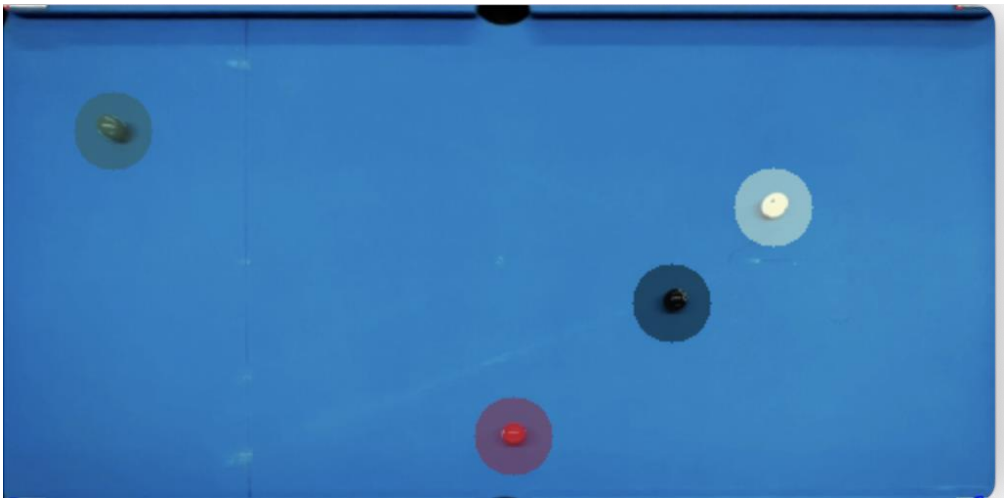

圖 51 球桌辨識及處理 2

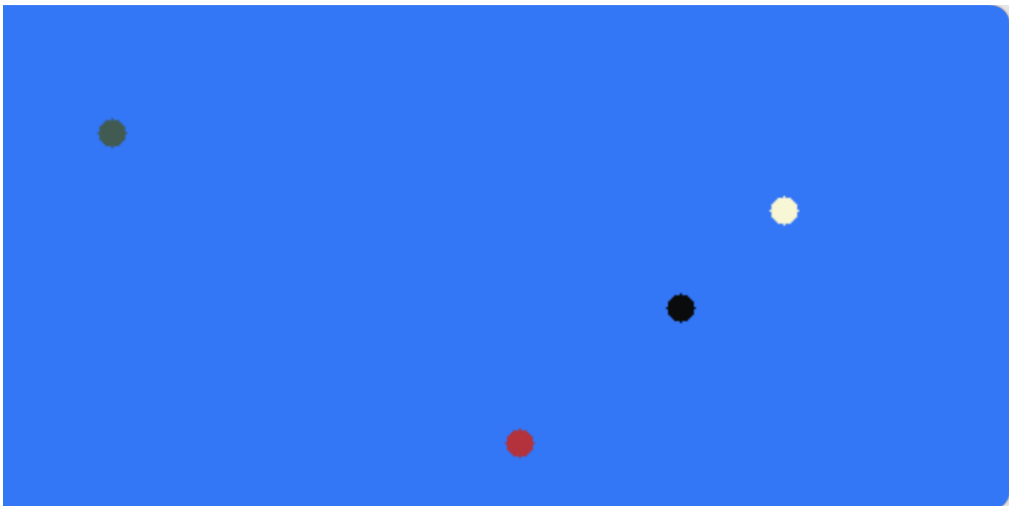

圖 52 球桌辨識及處理 3

做完球台辨識以及處理後，便進行撞球路徑的演算，求出所有可能解，並從中挑出評分最高的路徑。

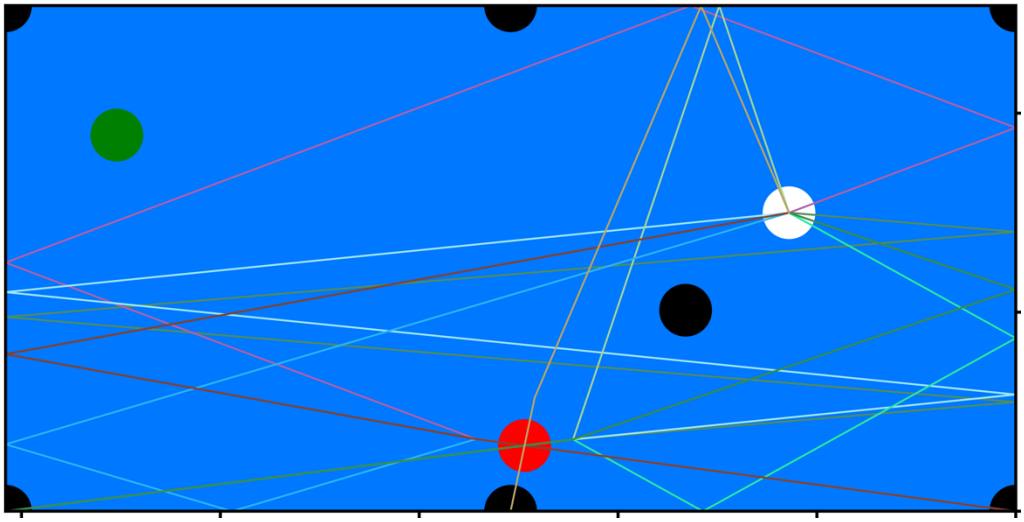

圖 53 所有擊球路徑

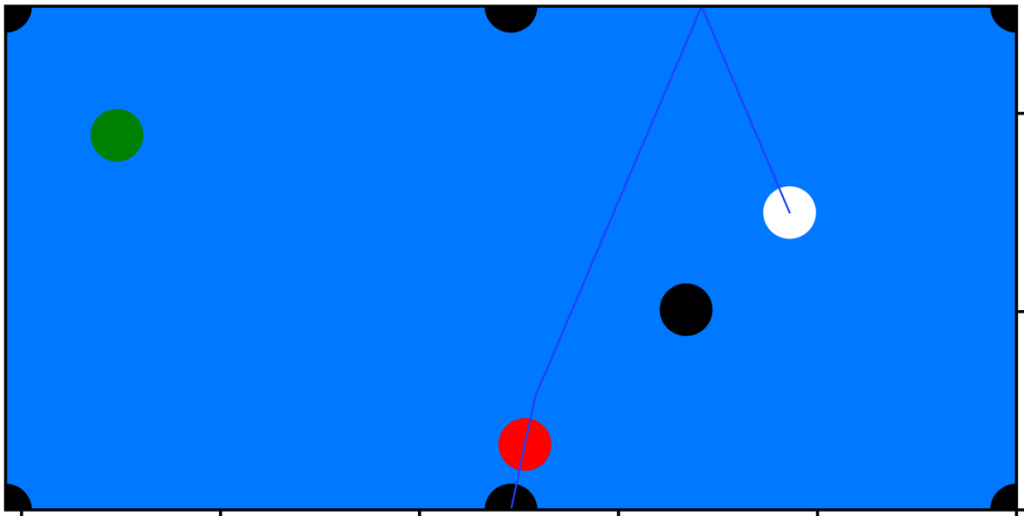

圖 54 最佳擊球路徑

有了最佳路徑後， Hololens 2 便可以向伺服器請求路徑，並動態生成球、路線在使用者介面上，使用者即可參照畫面的建議路徑擊球。

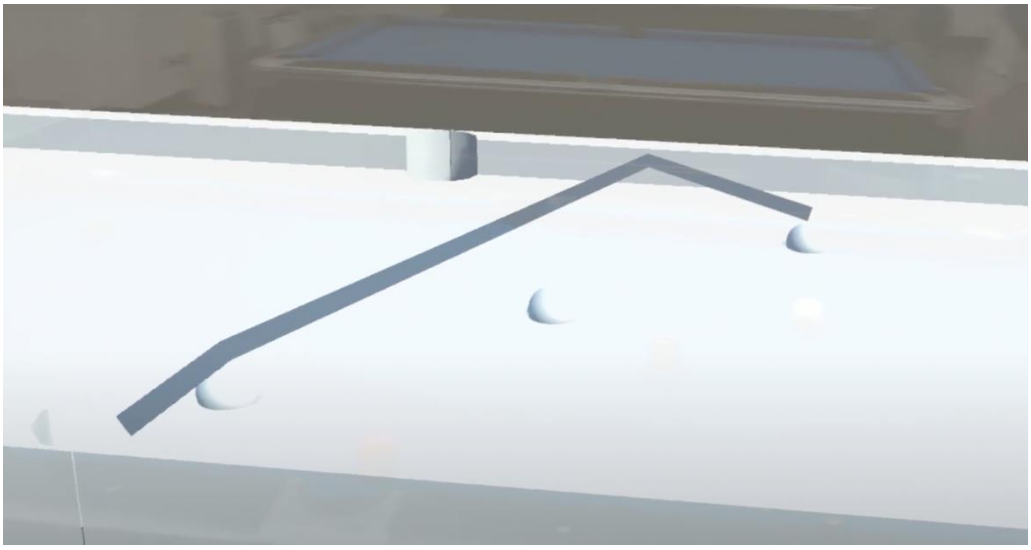

圖 55 Hololens 2 顯示最佳路徑

4. 結論與未來展望

在專題製作中，我們專題目前已經完成了 Hololens 2 與路徑演算法的結合，透過 Hololens 2 投影出一虛擬球桌及演算法的路線，以達到初步輔助使用者的目的，而我們將透過各個面向在下方探討其未來展望。

4.1 Hololens 2

由於 Hololens 2 的開發社群上還不是很完善，很多技術依然靠我們自行嘗試與摸索。在未來若能夠先優化球桌與球的模型建置，將可以很好的輔助使用者在擊球時的判斷。

4.2 球桌辨識

由於球色在撞球規則當中，是很重要的規則之一，因此在球桌辨識上，未來若能使用機器學習針對 CV 在撞球顏色的閥值做訓練，將能很大的避免球桌辨識上的顏色誤差。以及若也能針對球位置的座標做校正訓練，也能很好的讓後端路徑演算法計算出更符合現實的擊球建議。

4.3 路徑演算法

關於路徑演算法，我們認為傳統的演算法設計很難考量到球局中的所有可能，其中包含：做球、進攻、防守策略等。所以為了設計考量所有球局狀態及規則的演算法，勢必需要使用機器學習來輔

佐，並將不同的撞球規則及球局狀態納入演算的考量來訓練，如此提供的路徑建議才會足夠符合賽局需求。

另外，為了要讓機器學習能夠在正確的環境下訓練，撞球實體模型的建置將會很重要，要讓實體模型足夠類似現實當中的物理狀態，才能避免機器學習訓練與現實撞球擁有擊球的誤差。

參考文獻

Microsoft 混合實境官方文檔：
https://docs.microsoft.com/zh-tw/windows/mixed-reality/

Unity 官方文檔：
https://docs.unity.com/

阿新，「Unity 中四種座標系之間的關聯」，程式人生，2019-02-15
https://www.796t.com/content/1550211504.html

Walterlv，「Unity 引用 DLL 或安裝 NuGet 包」，ITW01，2020-04-28
https://itw01.com/84LLYE3.html

如何在 .NET 中序列化和還原序列化 (封送處理和 unmarshal) JSON：
https://docs.microsoft.com/zh-tw/dotnet/standard/serialization/system-text-json-how-to?pivots=dotnet-6-0

头号理想，「不同脚本之間調用方法」，CSDN，2020-03-18
https://blog.csdn.net/weixin_44302602/article/details/104938982

Github 範本：
https://github.com/MicrosoftDocs/mixed-reality.zh-TW/tree/live/mixed-reality-docs

Dinesh Punni，「Unity HoloLens Tutorial 2019 - Spatial Mapping」，Youtube，Dec 1, 2019
https://www.youtube.com/watch?v=nAG2wJv12xE

Julia Schwarz「How to know which button was clicked ? 」，StackOverflow，Sep 24, 2019

https://stackoverflow.com/questions/58063110/how-to-know-which-button-was-clicked-unity-hololens2-mrtk

Ken，「Unity JsonUtility 的局限性」，騰訊游戲學堂，2018-12-18
https://gameinstitute.qq.com/community/detail/128931

编程宝库，「使用 LineRenderer 动态划线」
http://www.codebaoku.com/it-csharp/it-csharp-209428.html

Baeldung，「Design a Genetic Algorithm in Java」，May 23, 2022
https://www.baeldung.com/java-genetic-algorithm

Marek Obitko，Hochschule für Technik und Wirtschaft Dresden (FH) (University of Applied Sciences)，「Introduction to Genetic Algorithms」，September 1998
https://www.obitko.com/tutorials/genetic-algorithms/selection.php

國立交通大學，「附錄A：遺傳演算法」
https://ir.nctu.edu.tw/bitstream/11536/48945/11/654911.pdf

linuxtut，「To measure the position of the object on the desk」
https://www.linuxtut.com/en/c6e468da7007734c897f/

阿新，「使用opencv識別影象紅色區域,並輸出紅色區域中心點座標」，程式人生，2020-06-03
https://www.796t.com/article.php?id=22296

Antonella Perucca，Professor for Mathematics and its Didactics at the University of Luxembourg,「Arithmetic billiards」，University of Cambridge，24 April, 2018
https://plus.maths.org/content/arithmetic-billiards-0