應用資料探勘於棒球球種分析

Application of Data Mining Techniques to the Identification Of The Pitch Type

國立中山大學資訊工程學系

106 學年度大學部專題製作競賽

組員：

B033040011 黃柏勳

B033040043 紀儒達

B033040049 曾昱榮

指導教授：張玉盈 教授

目錄

壹、摘要

我們收集網路上的棒球轉播影片，實作出可自動追蹤棒球軌跡的系統，並且分析大量投球的軌跡，依照不同的軌跡特性以及 K-means 演算法分群，藉此辨識出投手投球的球種。

貳、研究動機

棒球是一項非常迷人的運動,它的不可預測性，深深地吸引著廣大的球迷。每一個 play、每一個投打對決都有可能成為影響比賽勝負的關鍵，而在一場棒球比賽中最重要的就是投手，我們除了常常可以看到投手投出一顆血脈噴張的火球來解決打者外，更可以經常看到投手投出一顆顆犀利刁鑽的變化球。而在區分變化球球種上，基本上可以區分為兩種，也就是縱向的變化球（指叉球、變速球等）和橫向的變化球（滑球、伸卡球等）。隨著棒球技巧不斷地發展，許多新興的球種也開始出現，傳統的方法已經沒有辦法準確地分析球種。

因此，我們的目標是設計出一套分辨球種的系統，有鑑於每個投手有不同的出手點、投球技巧，且同一種球投出去的軌跡也會有所不同。我們將收集大量的投球影片，分析所有的可能的軌跡、加速度，再利用資料探勘的技巧和 K-means 演算法歸納出六種常見的球種（直球、滑球、曲球、指叉球、伸卡球、變速球）。最後我們期許未來能夠搭配手機的照相功能，實作出 IOS 與 Android 系統之 APP，做到可以即時分析的效果。未來的使用者只需帶著手機，就能在任何地方磨練自己的投球技巧！

參、球種辨識

3-1 投球資料蒐集

這個部分我們首先搜集許多美國職棒大聯盟投手的電視轉播影片，再將影片中每個 frame 截成圖片，再人工紀錄每球的球速以及對應圖片中棒球的所在座標，而這些球速數據及座標最後將會成為球種的評判標準，示意圖如下(圖 3-1)，此外，在此文我們都以右投手做為研究目標，所有球種的數據資料都是以右投手為主，下文將不再贅述。

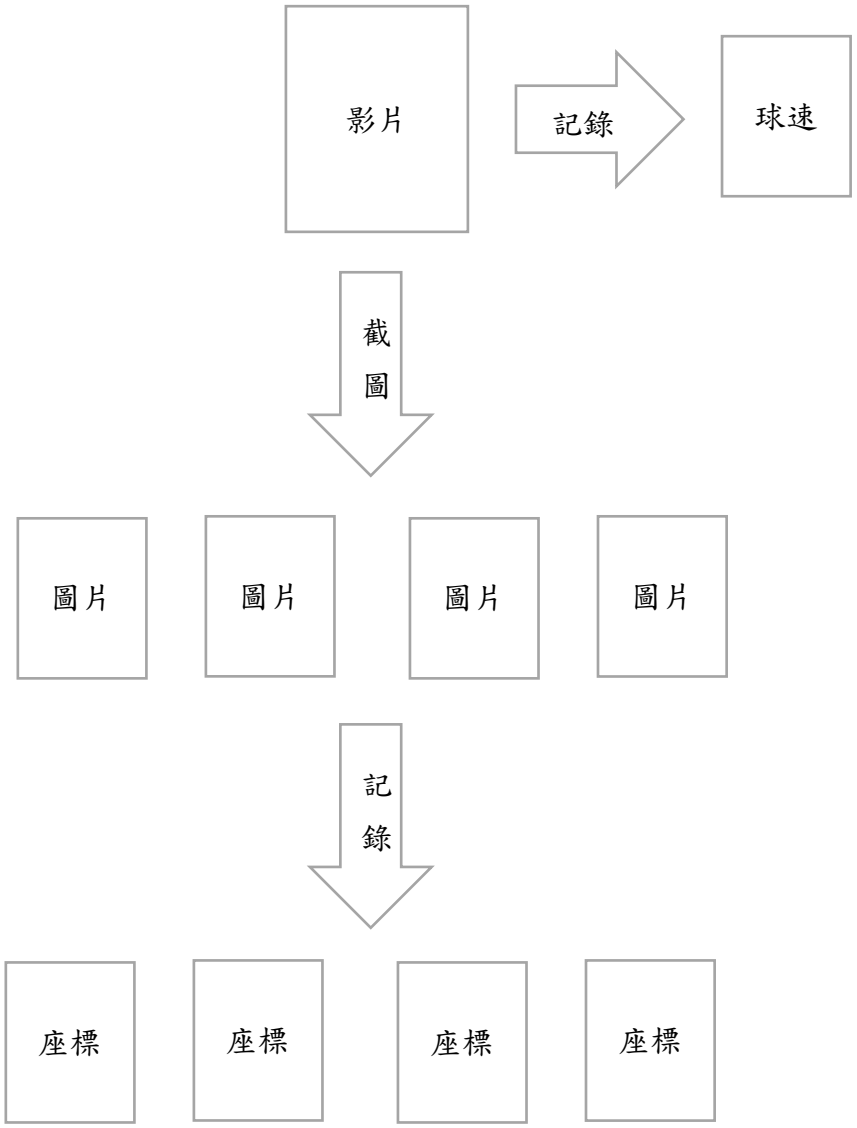

圖 3-1 資料紀錄示意圖

3-2 分析棒球軌跡

為了使每個球種的軌跡可以數據化，我們利用每次投球的座標計算該次投球的縱向與橫向加速度，不同種球種的軌跡特性可以藉由加速度來呈現，示意圖如下(圖 3-2)。

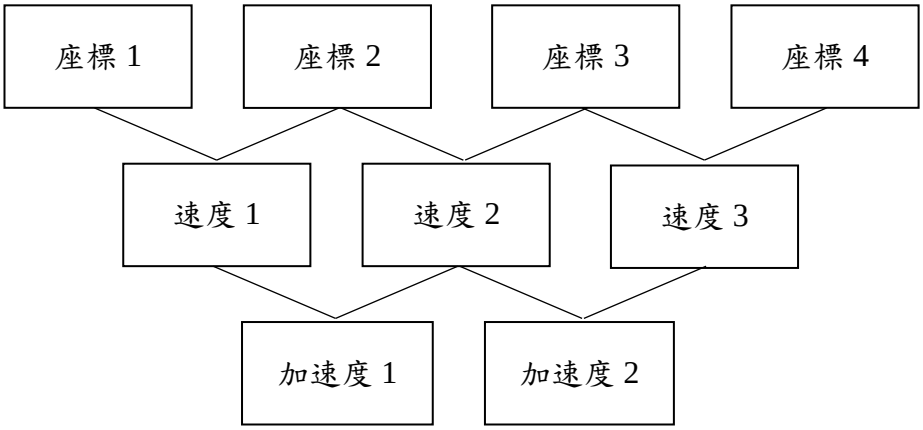

圖 3-2 軌跡加速度示意圖

我們找各球種約若各十顆投球的影片，記錄該座標並且計算這六十顆投球的平均加速度記錄在 XY 座標上，X 軸為橫向平均加速度，Y 軸為縱向平均加速度，軌跡加速度的分布圖如下(圖 3-3)。

此外，根據我們的觀察與試驗，我們嘗試將加速度數據使用 K-means 演算法分成不同數量的群，分群個數過多或過少皆無法適當的將球種歸類，我們發分成三大群最能夠適當的將不同軌跡性質的球種歸類，此三大群分別如下。

Group1─快速球，這群的特性為變化幅度小，快速球為所有球種中變化幅度最微小的球種。Group2─指叉球、變速球、伸卡球，這群的特徵為軌跡會小幅度下墜且稍微向右打者內角彎，Group3─曲球、滑球，此群的特徵是軌跡會向右打者外角彎，下墜幅度範圍從小到大皆有，K-means(K=3)分群示意圖如下(圖 3-4)。

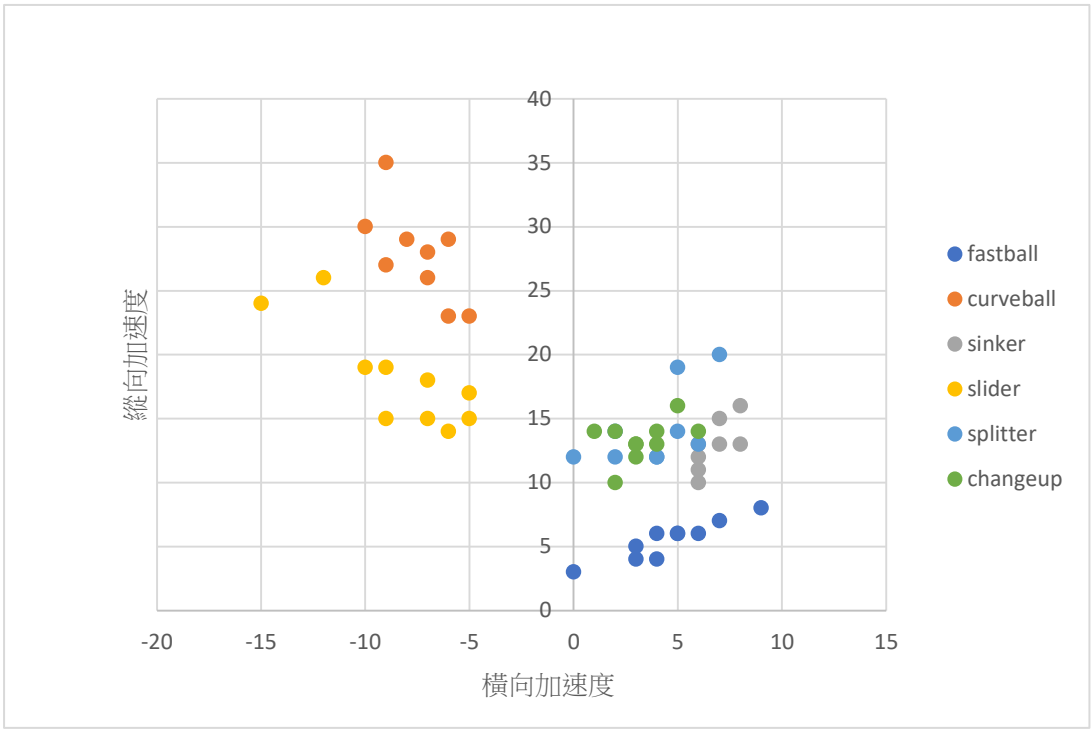

圖 3-3 球種加速度分布圖

3-3 辨識方法

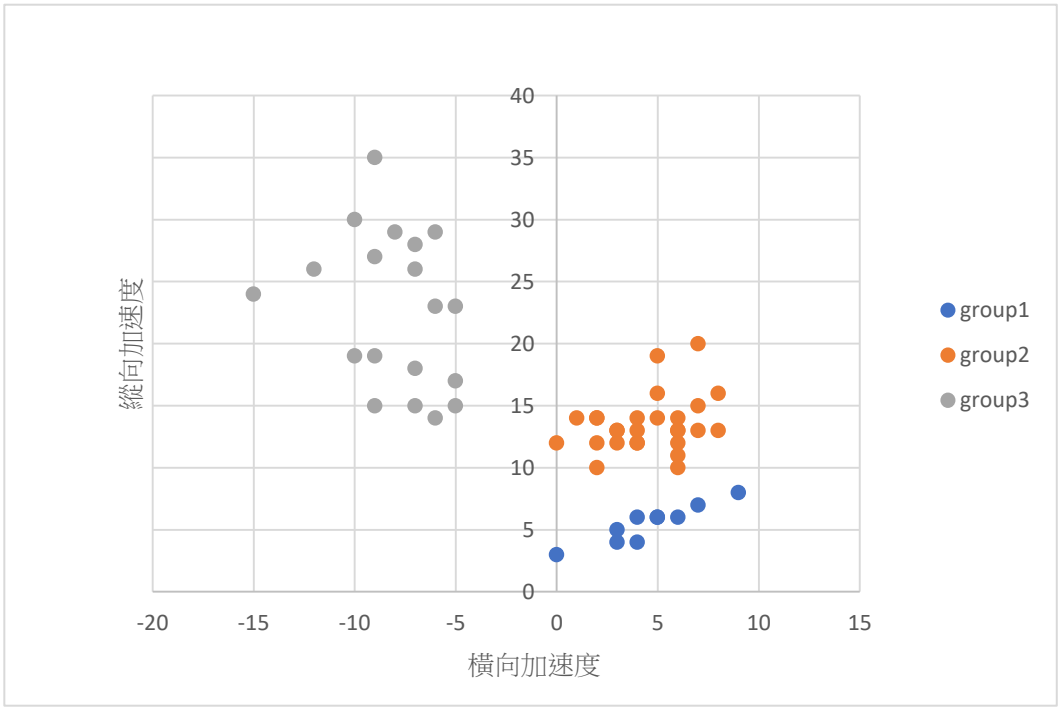

圖 3-4 K-means(K=3)分群示意圖

當我們的系統得到一組連續的軌跡座標及球速，系統首先計算此球的橫向與縱向平均加速度，並且配對到 K-means 演算法分出的三大群中距離最近的群。在三大群中如何再更細分成更小群，如何能更準確的辨識球種，球速將會成為第二層的篩選依據，我們針對不同群有以下的辨識方法。

若該球被分配到 Group1，直接辨識該球為快速球。若被分配到 Group2，則以球速作為判定依據，根據美國職棒大聯盟 2016 年比賽的統計，伸卡球的平均球速為 91.3MPH，指叉球的平均球速為 84.5MPH，變速球的平均球速為 83.6MPH，依據該球的球速最接近何種球速，則判定為該球種。由於 Group3 中的曲球和滑球有較明顯的下墜幅度差異，所以若被分配到 Group3，則會依據 Group3 內的所有點再做一次 K-means(K=2)演算法，將曲球跟滑球明顯分為兩群，該球較靠近何種球種的群中心，則判定為該球種，辨識流程圖如下(圖 3-5)。

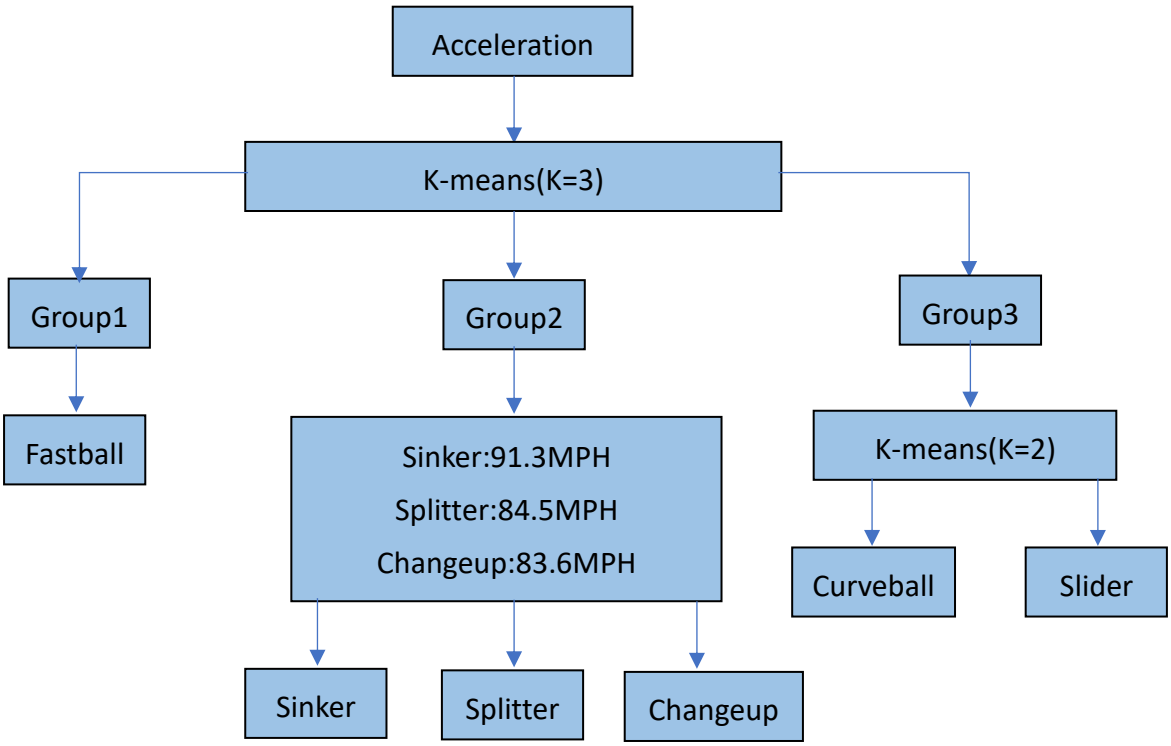

圖 3-5 辨識流程圖

3-4 成果

下圖綠色代表球的座標位置，F 代表快速球，S 代表伸卡球，P 代表指叉球，H 代表變速球，L 代表滑球，C 代表曲球，紅色代表群中心。

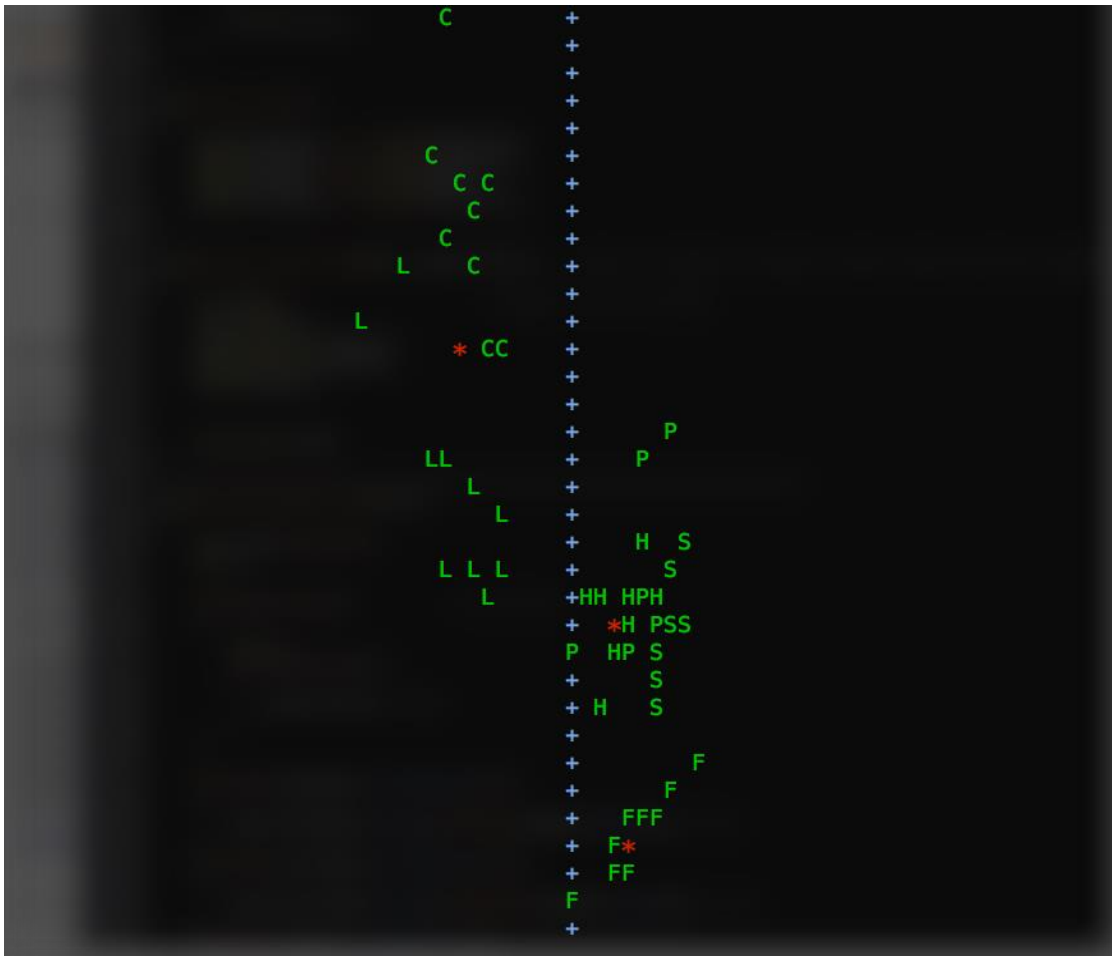

圖 3-6 K-means(K=3)分群及球種資料分布圖

下圖為滑球與曲球第二次執行 K-means 分群結果圖，紅點代表群中心。

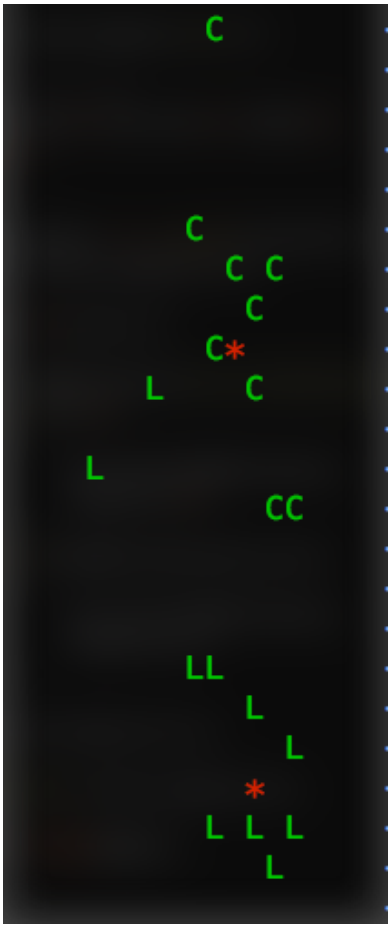

圖 3-7 K-means(K=2)分群與滑球、曲球分布圖

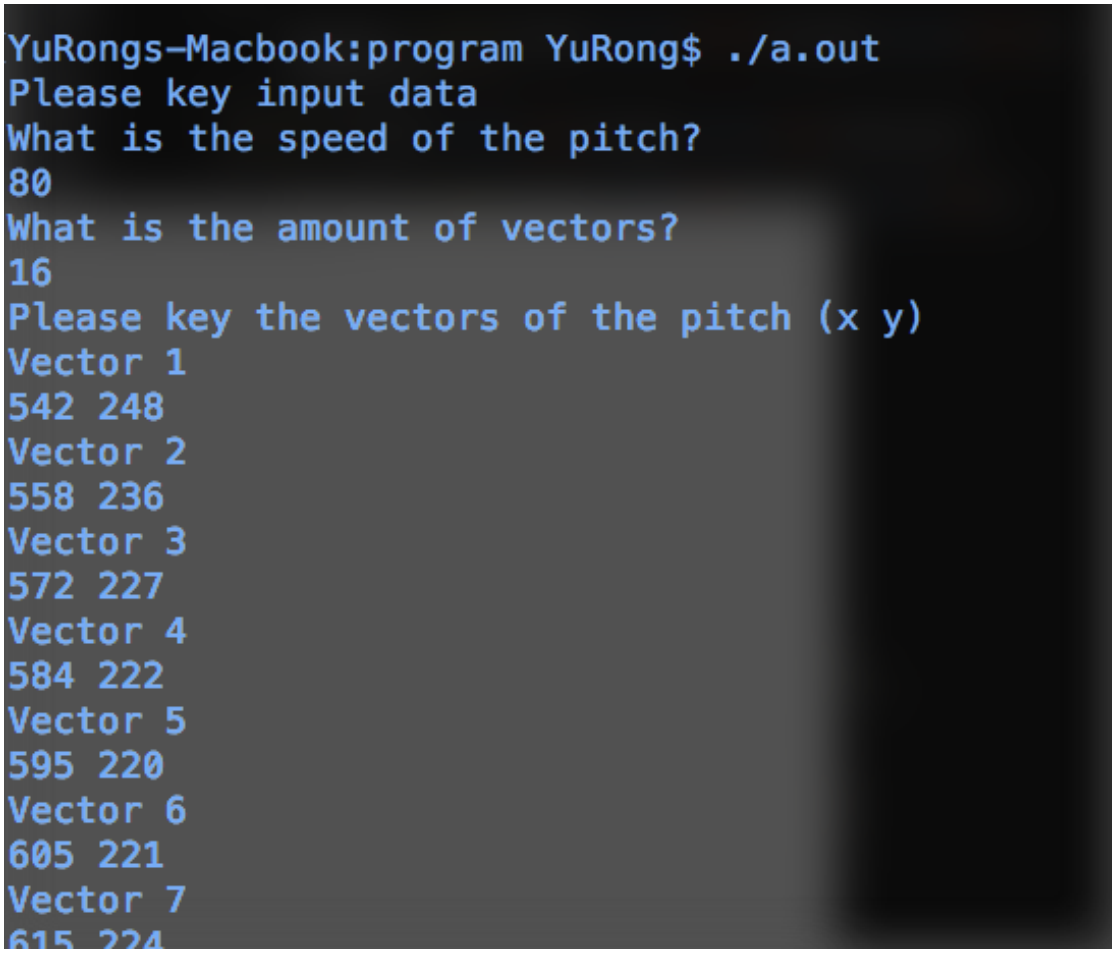

圖 3-8 執行範例

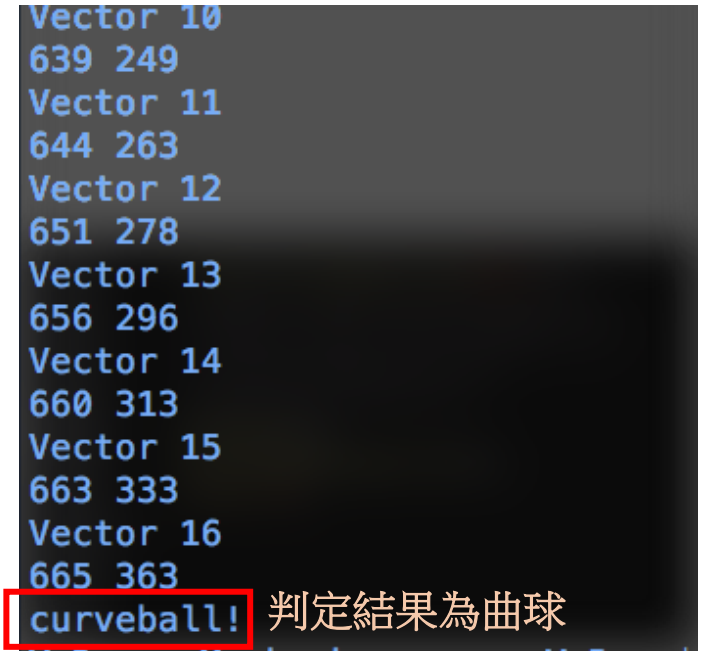

圖 3-9 執行範例

3-5 實驗結果

從網路上準備十顆隨機的球種投球數據，先以人工辨別這十顆球的球種，再輸入這十顆球的數據給球種辨別程式執行，最後計算程式執行結果與人工辨別結果相同的比例。

Pitch 1		Speed	84
人工辨別	Changeup	程式辨別	Changeup

Pitch 2		Speed	86
人工辨別	Splitter	程式辨別	Fastball

Pitch 3		Speed	92
人工辨別	Sinker	程式辨別	Sinker

Pitch 4		Speed	82
人工辨別	Curveball	程式辨別	Curveball

Pitch 5		Speed	95
人工辨別	Sinker	程式辨別	Sinker

Pitch 6		Speed	95
人工辨別	Fastball	程式辨別	Fastball

Pitch 7		Speed	86
人工辨別	Splitter	程式辨別	Splitter

Pitch 8		Speed	84
人工辨別	Changeup	程式辨別	Changeup

Pitch 9		Speed	82
人工辨別	Slider	程式辨別	Slider

Pitch 10		Speed	63
人工辨別	Curveball	程式辨別	Slider

Hit	8	Miss	2
Hit rate	80%		

肆、軌跡偵測

為了達成能夠及時辨識球種的理想，我們勢必先實作出能夠自動偵測球座標的系統，此功能還正在開發的階段，尚未能夠理想達程目標。實作流程大致分為下列幾階段，擷取視訊影像、圖片二值化、圖形辨識，而這些功能我們選擇使用 OpenCV 的內建函式庫實作。

4-1 擷取視訊影像

我們使用 OpenCV 的內建函式庫直接將影片中所有 frame 擷取出來，如下圖 (圖 4-1)。

圖 4-1 擷取視訊影像

4-2 圖片切割

由於球的軌跡主要是在中心區域內，因此將圖片中心區域的部分切割出來，以減少後續影像處理的成本，大幅降低執行的時間。

4-3 圖片二值化

所謂的二值化，意思就是將圖片的灰階度設為最大或最小，一般都是 255 或 0，也就是將圖片轉為明顯的黑白圖片，可以大量地減少圖片的運算量。要二

值化之前會先設定一個門檻值，灰階高於這個門檻就設為 255，反之設為 0。在此我們直接運用 OpenCV 內建的函式實作。

在這步驟，二值化目的為將棒球在圖片中變全白色，其餘不重要的背景變為黑色，可以方便系統偵測球的位置。在此時，門檻值的設定相當重要，若是門檻值設的太低會導致圖片中背景充斥著白色，有可能會影響系統偵測球，若是門檻值設得太高，由於圖片中棒球不夠白的部分會低於門檻，而導致球的形狀變得不完整，甚至是整顆球都無法高於門檻，而在二值化圖片裡看不到球，示意圖如下(圖 4-2)。

圖 4-2 圖片二值化

4-4 多筆樣本比對

為了讓系統能自動偵測球的位置，我們應用 OpenCV 中的函式 matchTemplate(InputArray image, InputArray templ, OutputArray result, int method)。我們在多張二值化圖片中擷取某些球形較為明顯、清晰的區塊作為樣本，如下圖(圖 4-3)。透過 matchTemplate 函式可以藉由樣本照片與輸入的圖片做比對，找出與樣本圖片相似度最大的所在位置，並且使用矩形灰框標示出來。然而，樣本比對的結果未必是正確的，因此增加樣本數目，以提高正確率。比對結果如下(圖 4-4 至圖 4-6)。

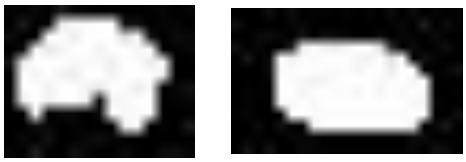

圖 4-3 樣本

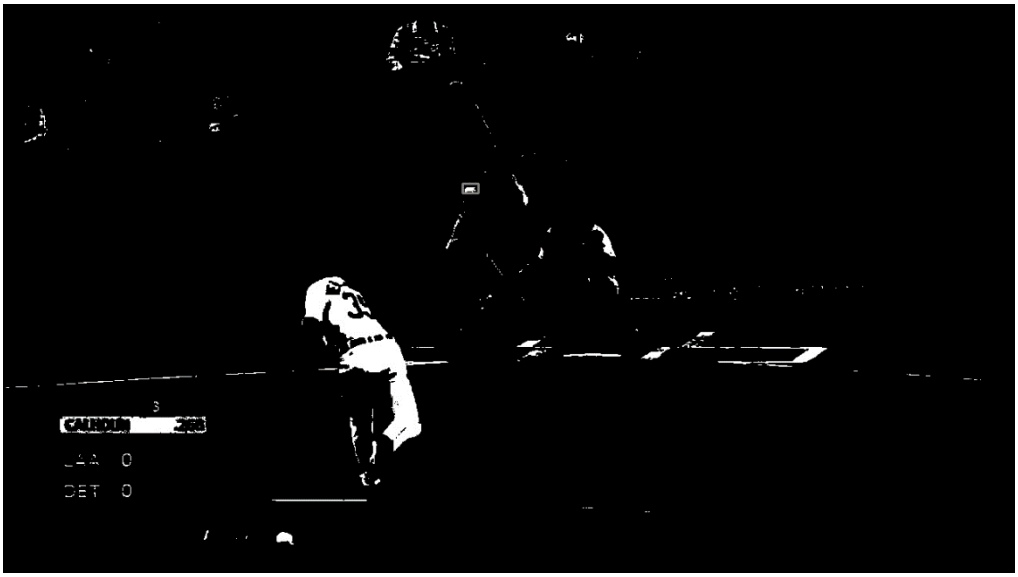

圖 4-4 比對結果

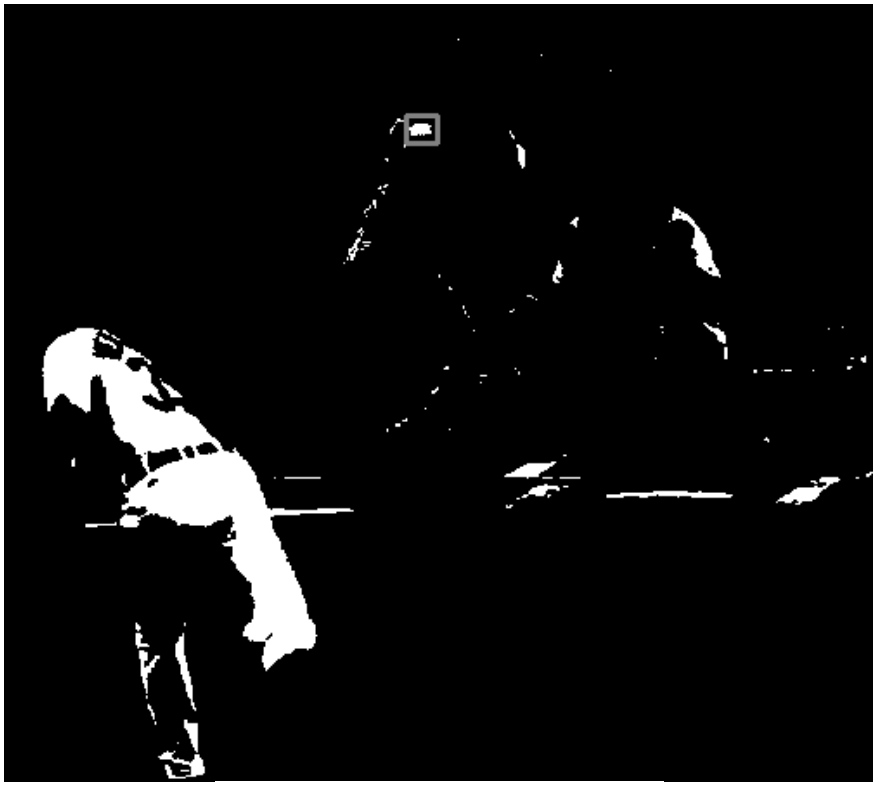

圖 4-5 成功比對結果

圖 4-6 失敗比對結果

4-5 分析樣本結果

因為有多筆樣本結果,所以需要設計程式讓電腦自動判斷何者樣本的結果為正確，主要分為下列兩部分:

(1) 判斷第一張圖片之棒球座標:

累計與其他樣本位置相似之數目有多少,選出數目最多者之位置做為第一組座標。

(2) 判斷剩餘圖片之棒球座標:

與前一張圖之棒球座標做比較，由於每張圖之間的間隔時間小，因此棒球座標位移不應相距太遠，若相距太遠，表示該樣本結果有誤。接著將通過上述條件篩選之樣本加入目前棒球之飛行軌跡，判斷是否符合物理現象(飛行軌跡是否連續)。最後再計算符合條件之樣本與其他樣本相似位置個數，選出數目最多者做為該時間點之軌跡座標。

(3) 軌跡座標完整化

經過(1)、(2)步驟的處理後，由於影片中許多環境因素影響，該次投球的軌跡截圖中，仍有可能無法比對成功，所以並不是每一次的投球都能夠完整的分析出軌跡座標。在一次完整的投球中，若有少數圖片無法判別座標，我們將根據前後圖片座標的軌跡推測該遺失的座標。

補齊遺失座標的做法，我們假設該次投球之軌跡呈等加速度運動。根據前後座標之速度與位移量，計算遺失座標期間的加速度，以補足遺失座標。假設遺失座標為第 n 筆座標，首先計算第 n-2 筆到第 n-1 筆的座標速度 v，再計算第 n-1 筆座標至下一筆有效座標第 k 筆座標之位移量 ΔX，時間間距為$t = k - (n - 1)$。代入等加速度運動之公式$\Delta X = vt + \frac{at^2}{2}$，計算出加速度 a。再根據加速度 a 及第 n-2 筆至 n-1 筆座標之速度推算第 n 筆開始之遺失座標。

4-6 軌跡偵測流程圖

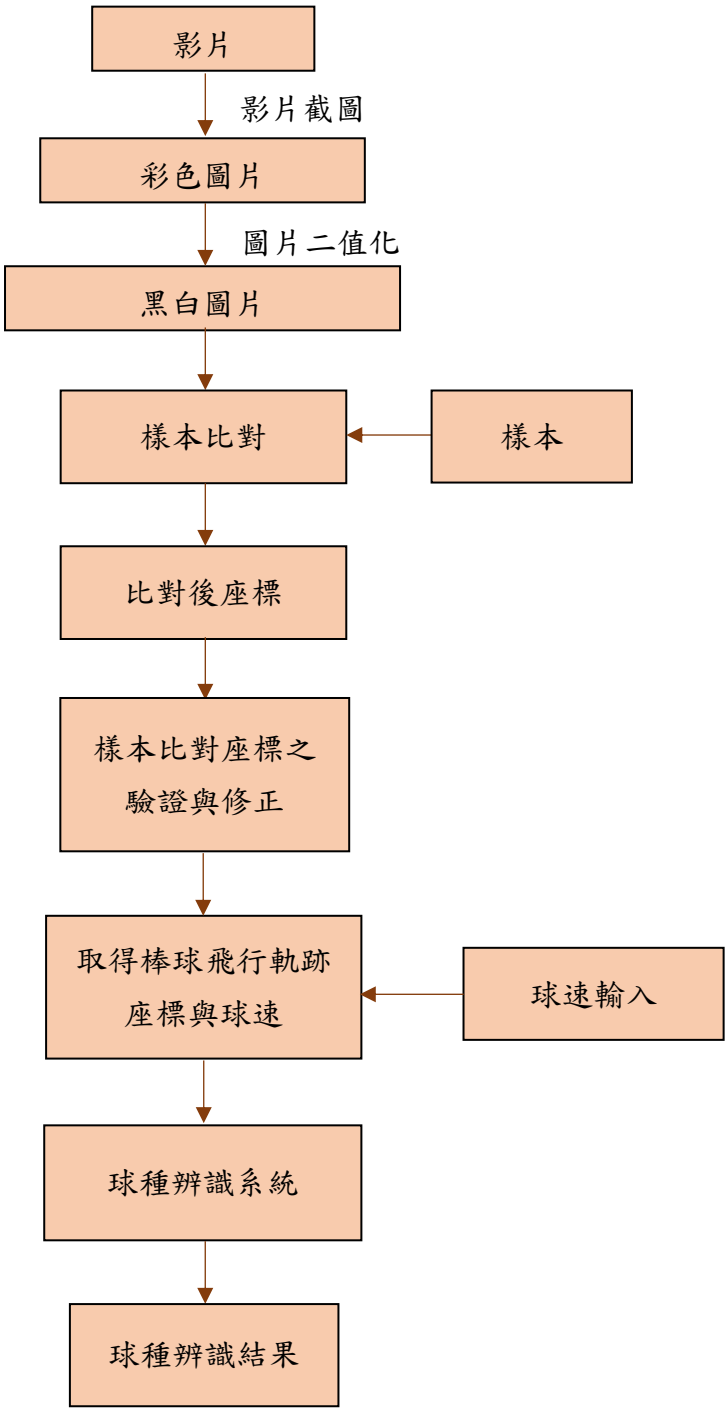

圖 4-7 軌跡偵測流程圖

陸、結論

單論球種辨識的部分，應用在大聯盟電視轉播上，系統的準確度有 80%，整體表現還算優異，其中兩顆誤判的球中，有一球為 63MPH 的曲球，根據大聯盟比賽中的統計，大聯盟投手的曲球球速普遍分布在 70MPH 至 80MPH 之間，顯然的，我們的系統對於球速較為極端的變化球辨識力不足，也因此延伸出一個問題，若是對於較低層級的比賽中，投手普遍球速不如大聯盟投手，這套系統是否還能有優異的辨識能力是待觀察的重點。

未來希望能夠再將軌跡偵測的部分做得更加完善，使的這個系統更加完整，甚至朝向及時辨識球種的目標努力。

柒、參考文獻

[1] K-means 演算法筆記

https://dotblogs.com.tw/dragon229/2013/02/04/89919

[2] MLB 2016 平均球速統計

http://www.fangraphs.com/leaders.aspx?pos=all&stats=pit&lg=all&qual=0&type=10&season=2016&month=0&season1=2016&ind=0&team=0,ss&rost=0&age=0&filter=&players=0

[3] 擷取視訊影像

https://cg2010studio.com/2012/05/18/opencv-%E6%93%B7%E5%8F%96%E8%A6%96%E8%A8%8A%E5%BD%B1%E5%83%8F-extract-video-frames/

[4] OpenCV API Reference

https://docs.opencv.org/2.4/modules/refman.html