國立東華大學資訊工程系
National Dong Hwa University

109 學年度大學部專題研究報告

109 CSIE Project Report

電子棒球好球帶應用程式開發

指導教授 Advisor： 顏士淨 教授

專題參與人員 Team Member： 陳冠銘
黃議承
洪三峰
江承澔

中 華 民 國 110 年 5 月 21 日

國立東華大學資訊工程學系

專題報告原創性聲明

National Dong Hwa University

Department of Computer Science and Information Engineering

Statement of Originality

本人鄭重聲明：

所呈交的專題報告是在指導老師指導下進行的研究工作及取得的研究成果。除文中已經註明引用的內容外，本報告不包含任何其他個人或集體已經發表或撰寫過的研究成果。對本文的研究做出重要貢獻的個人與集體，均已在文中以明確方式標明。若有違上述聲明，願依校規處分及承擔法律責任。

I hereby affirm that the submitted project report is the result of research under the supervision of my advisor. Except where due references are made, the report contains no material previously published or written by another person or group. All significant facilitators to the project have been mentioned explicitly. Should any part of the statement were breached, I am subject to the punishment enforced by the University and any legal responsibility incurred.

學號 Student No.	學生姓名 Name	親筆簽名 Signature
410725008	陳冠銘	
410725031	黃議承	
410721245	洪三峰	
410721246	江承澔	

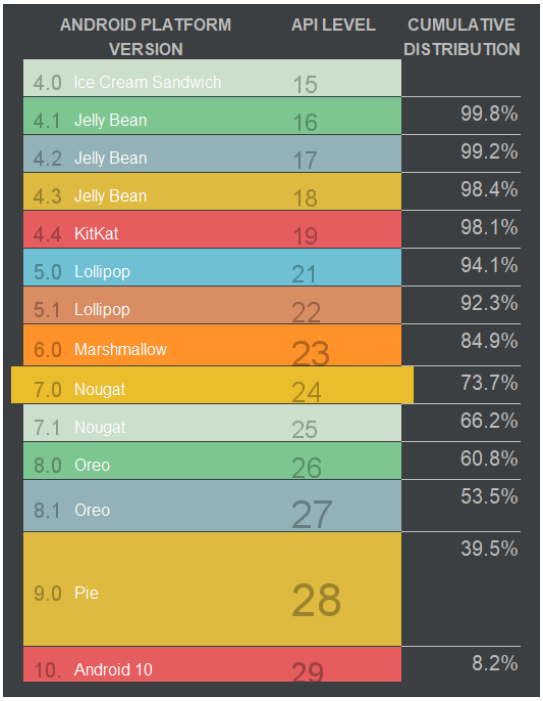
日　　期 Date ：_____

計畫摘要 Abstract

關鍵詞： 棒球、好球帶、Android Python、影像處理(open-cv)、機器學習
Keywords: Baseball, Strike Zone, Android Python, Image Processing, Machine learning

本專題研究為設計並實作出可用於紀錄棒球飛行軌跡、並比對棒球打者之好球帶的行動應用程式，主要提供於學生、上班族、棒球愛好人士…等非職業棒球運動員使用。

應用程式使用 Android 作業系統，最低限制為 Android 7.0 Nougat (Sdk Version 24)，由 Android 官方文件(下圖)可得知，可支援現今 73.7%以上的 Android 手機。



本應用程式除了使用 Android Studio 之 Java 語言編寫之外，還使用了 Chaquopy 套件協助開發。Chaquopy 為英國蘇格蘭地區 Chaquo Ltd(有限公司)所開發之用於 Android 平台的 Python 轉換套件，可供程式開發者在 Java 語言編寫的 Android 應用程式中輕鬆地調用 Python 模組。

本應用程式在開發期間所使用的影片素材、或是專業意見提供，大多數由 國立東華大學棒球社 所提供，在此非常感謝 國立東華大學棒球社 之配合。

目錄 Content

目錄

一、前言 Introduction

「電子棒球好球帶應用程式開發」提供各種棒球練習、業餘棒球賽事進行更精確的好壞球判斷。

隨著國內素人棒球風氣逐漸發展，越來越多非職業的棒球素人想科學化地訓練，但棒球科學分析器材十分昂貴，也需要更多人力。

現今許多球類賽事開始加入輔助裁判系統協助主審裁判進行判決，但仍有許多學生比賽、友誼賽、練習比賽等...小型比賽無法擁有如此昂貴的輔助裁判系統。

『電子棒球好球帶應用程式』 便可在智慧型手機上實現紀錄棒球飛行軌跡並輸出為圖檔。可提供球員練習參考，或是藉由簡單的好壞球判定，提供更多小型比賽擁有更公平的判決。

二、研究動機與研究問題 Motivation and Research Problem

現今許多賽事為了比賽的公正性與公平性，紛紛使用輔助判決系統，例如籃球、足球、羽球…等運動項目。

在多數的大型賽事中均使用鷹眼系統或即時回放系統，作為裁判複審時的依據，當選手質疑裁判的判決或是裁判無法準確判斷時，可經由鷹眼系統所模擬的影像來進行複審並做出最後的判決。

但由於一套鷹眼系統設備的價值動輒要上百萬美元，而後續系統運作以及維護成本也相當昂貴，許多低成本的賽事完全無法負擔如此昂貴的費用。

研究小組成員中，本身喜歡棒球運動，並且參加**國立東華大學棒球社**的陳冠銘、江承澔，以前想要分析自己的投球軌跡，都是一個十分困難的挑戰。

因此我們開發了一款提供紀錄棒球飛行軌跡、簡單地判斷好壞球 之

電子棒球好球帶 應用程式

在現今人手一支的智慧型手機上便可執行，不但極具便利性且不必額外的設備費用與維護成本，讓更多低成本的賽事藉由本應用程式擁有更公平的賽事體驗，也可以讓業餘棒球運動員們有個分析棒球球路的工具。

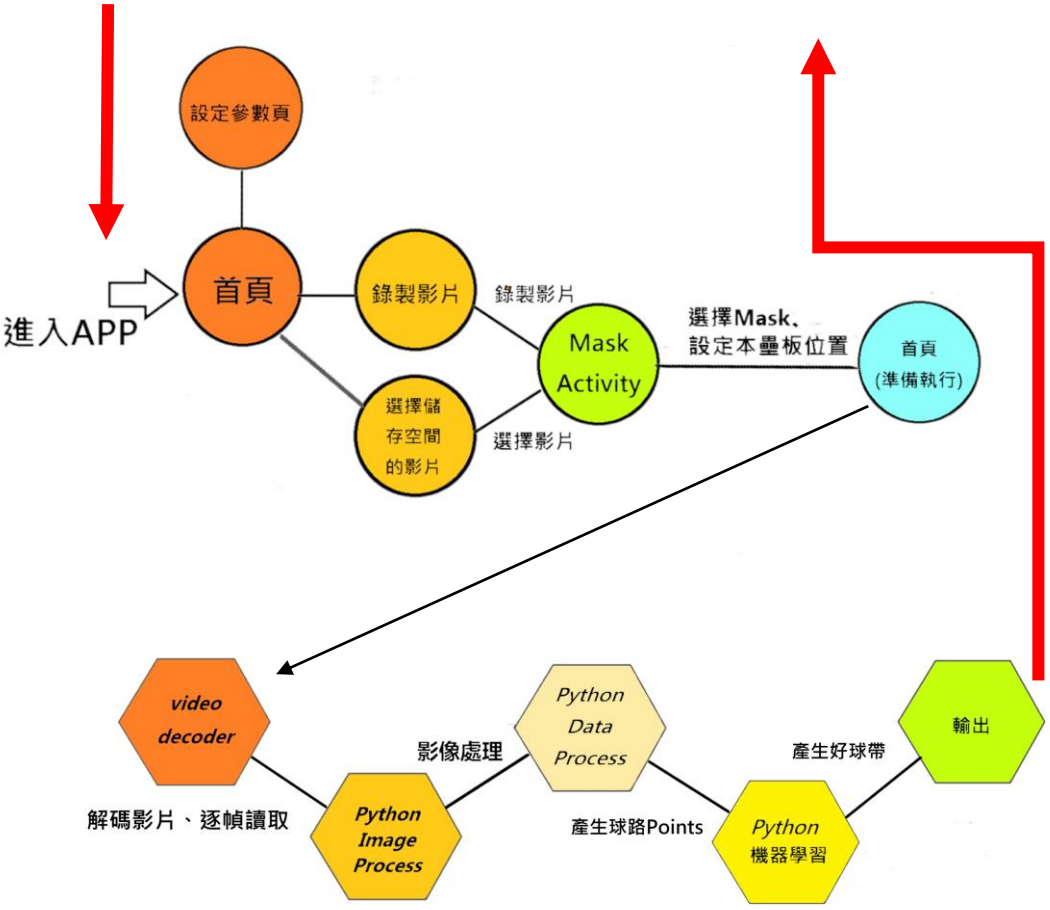
本次的專題研究除了開發應用程式之外，我們還提出了下列問題進行討論：

(一) 機器學習是否能在檢測棒球軌跡上幫忙？

(二) 利用本應用程式得出的軌跡資料，是否能進一步的拓展應用?

三、研究方法與步驟 Research Method

(一) 應用程式流程



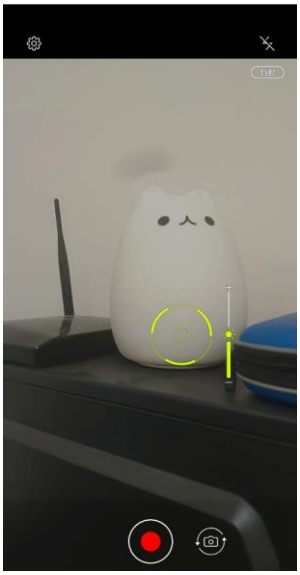
(二) 應用程式開發

1. 錄製影片

本應用程式具有兩種影片來源方式，第一種為錄製影片。

我們透過 Intent Class 中的<*startActivityForResult()*>，

切換至系統內建<*MediaStore.ACTION_VIDEO_CAPTURE*>功能進行錄影，



▲圖 3-2-1A 系統內建之<*MediaStore.ACTION_VIDEO_CAPTURE*>功能

再設定<*onActivityResult()*>接收方才錄製之影片路徑，

因此我們不必親手寫出一個錄影頁面，

即可獲得快速、方便、高效率的錄影方式。

```
public void gotoRecorder(View v){
    Intent takeVideoIntent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
    takeVideoIntent.putExtra(MediaStore.EXTRA_VIDEO_QUALITY, value: 1);
    if (takeVideoIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takeVideoIntent, requestCode: 100);
    }
}
```

▲圖 3-2-1B 錄製影片之程式碼片段

2. 選擇內部儲存空間影片

本應用程式之第二種影片來源方式，即為讀取內部儲存空間之影片。

為了成功訪問手機內部儲存空間，我們必須在<AndroidManifest.xml>中

添加下列兩行獲得權限：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

接下來便可以透過 Intent Class 呼叫<*Intent.ACTION_GET_CONTENT*>

選取手機內部儲存空間之影片，獲得影片路徑。

▲圖 3-2-2A 選取內部儲存空間影片功能

```
public void choose_File_Btn(View v){
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
    intent.setType("video/*");
    intent.addCategory(Intent.CATEGORY_OPENABLE);
    startActivityForResult(intent, requestCode: 100);

}
```

▲圖 3-2-2B 選取內部儲存空間影片之程式碼片段

3. 選擇遮罩(Mask)

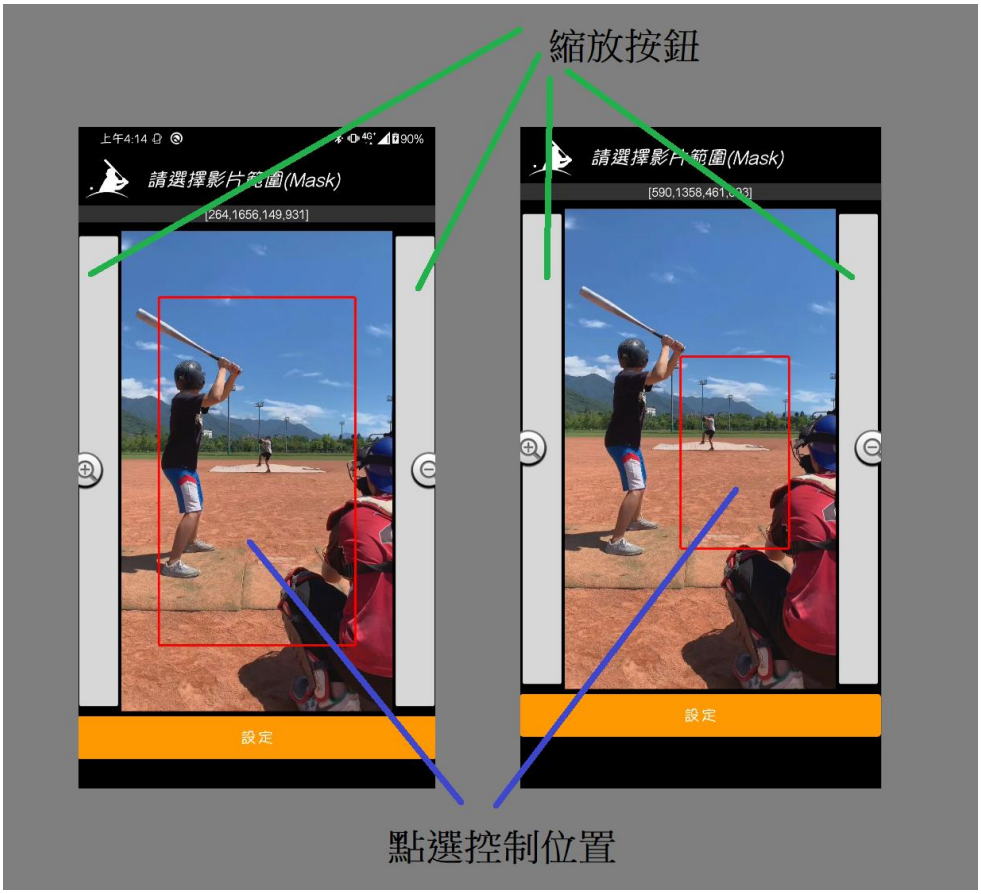
為了縮小檢測範圍和壓縮圖片，達到更加準確及快速的目的，

我們實作了一個頁面讓使用者自定義需要的範圍(Mask)，

使用者只須點選左右兩旁的放大鏡按鈕，即可控制檢測框的大小。

並且可直接點選圖片來控制檢測框的位置，

最後點選『設定』按鈕即可送出 Mask 範圍。



▲圖 3-2-3A 選擇遮罩功能

```
imgview.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        DRAW(MaskLong,(int)(event.getX()),(int)(event.getY()));
        return false;
    }
});
```

▲圖 3-2-3B 選擇遮罩功能之程式碼片段

4. 套件 Chaquopy : Python SDK for Android

▲圖 3-2-4A Chaquopy 官網介紹(https://chaquo.com/chaquopy/)

Chaquopy 是一款可以在 Android 應用程式中，調用 Python 模組的套件。

該套件安裝容易，語法好懂，運行效能也非常優秀，

此外，Chaquopy 支援了許多常用的 Python module，

像是 Numpy, OpenCV-Python, Pandas，

只要在<build.gradle>中加入需要 pip 的套件即可。

但是 Chaquopy 並沒有完美支援 Python module 的所有功能，

使用時需要特加留意，下一小節我們會談到 Chaquopy 跟 OpenCV 之間的相容性問題。

```
Python py = Python.getInstance();
PyObject j2p = py.getModule( s: "imageJAVA2PY");
final PyObject video = j2p.callAttr( key: "java2python");
video.callAttr( key: "set",ignoreFrame,bright,dis_limit,FDI_point,circlePER);
PyObject LINEobj = video.callAttr( key: "getLine");
```

▲圖 3-2-4B Chaquopy 之程式碼片段

5. 影片解碼 VideoDecorder

上一節我們提到了 Chaquopy 跟 OpenCV 之間存在相容性問題，

像是讀取影片效率非常完善的 cv2.VideoCapture 就無法使用。

因此如果我們想要逐幀讀取影片，還是得從 Java 語言下手。

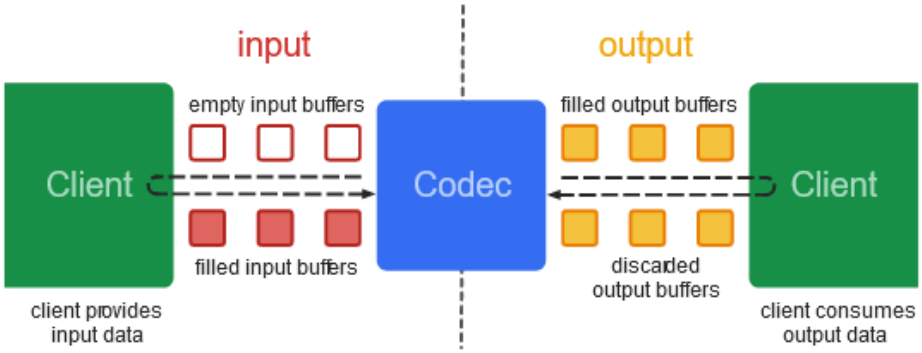
Android API 提供了 MediaExtractor 和 MediaCodec 兩個 classes，

MediaExtractor 可以將影片的視軌、音軌分離，並且獲得一些影片資訊，

MediaCodec 則是透過硬體加速影片的解/編碼，

MediaCodec 在解碼時，會將資訊存在 ByteBuffer 裡，

此時我們便可以讀取 ByteBuffer 裡的資訊，得到影片的每一幀。



▲圖 3-2-5A MediaCodec 官方圖解[1]

```
import android.media.MediaCodec;
import android.media.MediaCodecInfo;
import android.media.MediaExtractor;
import android.media.MediaFormat;
class VideoDecoder {
    public void stop() {
    }
    public void decode(Context c, Uri videoFileURI, DecodeCallback decodeCallback) {
    }
    private void decodeFramesToImage(MediaCodec decoder, MediaExtractor extractor, int width, int height, int rotation,
                                     DecodeCallback decodeCallback) {
    }
    private void getDataFromImage(Image image, int colorFormat, int width, int height) {
    }

    public interface DecodeCallback {
        void onDecode(byte[] yuv, int width, int height, int rotation, int frameCount, long presentationTimeUs);

        void onFinish();

        void onStop();
    }
}
```

▲圖 3-2-5B 我們設計之 VideoDecorder 的基本架構

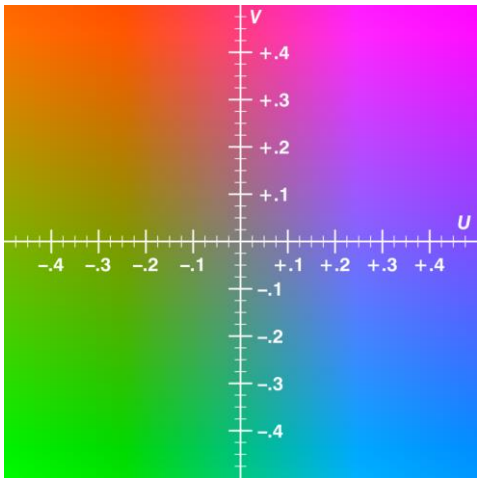
[1] 來源：https://developer.android.com/reference/android/media/MediaCodec

6. 轉換圖片格式(Yuv to Bmp)

在上一個小節，我們得到了影片的每一幀，可以開始影像處理了嗎?

答案是不行！因為我們的圖片格式為 YUV420，不是常用的 RGB 格式。

YUV 是一種顏色編碼的方式，代表著*明亮度(Y)*、*色度(U)*、*濃度(V)*。



▲圖 3-2-6A YUV 的 UV 通道模型

YUV 跟常用的 RGB 是有著轉換公式的：

$$
\begin{aligned}
Y &= 0.299 * R + 0.587 * G + 0.114 * B \\
U &= -0.169 * R - 0.331 * G + 0.5 * B + 128 \\
V &= 0.5 * R - 0.419 * G - 0.081 * B + 128
\end{aligned}
$$

$$
\begin{aligned}
R &= Y + 1.13983 * (V - 128) \\
G &= Y - 0.39465 * (U - 128) - 0.58060 * (V - 128) \\
B &= Y + 2.03211 * (U - 128)
\end{aligned}
$$

雖然兩者之間有轉換公式，但我們不需要自己埋頭苦幹造出 Function，

可以直接利用 Android API 的 YuvImage 跟 Bitmap 兩個 classes 完成轉換。

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
YuvImage yuvimage = new YuvImage(yuv0, ImageFormat.NV21,Video_width,Video_height, strides: null);
yuvimage.compressToJpeg(new Rect( left: 0,  top: 0,Video_width,Video_height),  quality: 100, baos);
Bitmap bmp = BitmapFactory.decodeByteArray(baos.toByteArray(),  offset: 0, baos.size());
```

▲圖 3-2-6B YUV 轉換至 Bitmap 之程式碼片段

(三) 影像處理

1. 影像前置處理

此節會介紹所使用到的影像處理方法，

與結合遮罩等功能來達成追蹤棒球飛行軌跡。

我們首先使用背景相減法將靜態物件排除，

接著使用型態學的侵蝕和膨脹使各候選物件更完整，

並消除背景相減法後留存的雜訊。

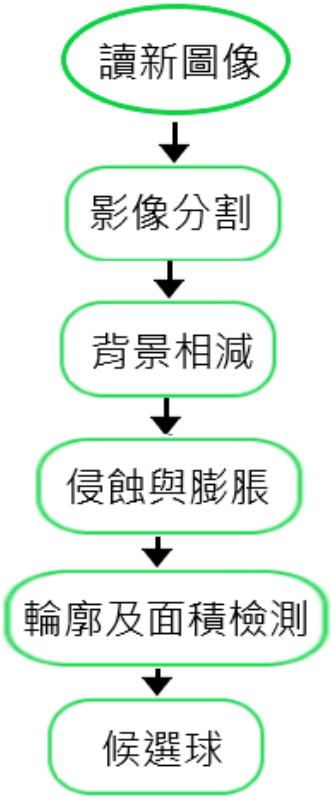
再利用 cv2 進一步找出更加精確的候選球點。

因為在實際影像上有一部分的影像並不會用來判斷球體位置，

使用完整影像處理上述流程會耗費相當的運算時間。

為了優化整體運作時程，

我們建構了遮罩來只對感興趣(正常球路範圍)的局部影像進行處理。



▲圖 3-3-1A 影像前置處理流程圖

2. 連續影格相差法

由於我們選定以擊球員後方拍攝來做為好球帶輔助判定的位置，

基本上是固定的，通常是不會有攝影機移動的影響，

能夠利用連續影格相差法來做投球畫面中移動物件的擷取，

連續影格相差影像是透過比較每兩幀連續影格的像素值差異，

將像素值差異較明顯的像素設為 255，將像素值差異較不明顯的設為 0。

連續影格相差影像 (Frame Difference Image) 定義為：

$$FDI_n(x,y) = \begin{cases} 255, & if\,|I(x,y)_n - I(x,y)_{n-1}| > T \\ 0, & otherwise \end{cases}$$

其中 n 代表影片幀數，T 為像素值差異的閥值，也就是相差的幅度，

$I(x,y)_n$及$I(x,y)_{n-1}$分別代表目前及前一張之影格像素值。

▲圖 3-3-2A 連續影格相差法處理過後之圖像(T=15)

3. 形態學

基本的形態轉換為侵蝕(Erosion)與膨脹(Dilation)能實現多種功能，

如消除雜訊、分割出圖像中連接相鄰的元素。

當影像經過連續影格相差法篩選得到的結果，

仍然存在一些非目標物的近似篩選條件所產生之細小雜訊，

經過侵蝕與膨脹後可將細小雜訊消除與修補輪廓，其效果見圖 3-3-3A。

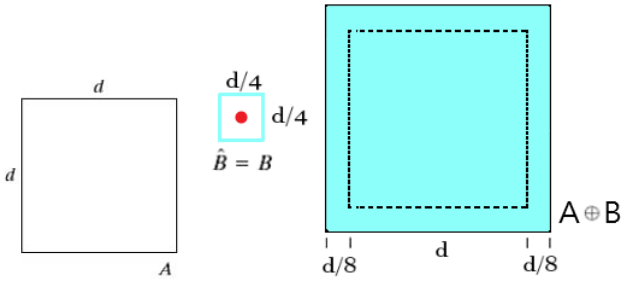
▲圖 3-3-3A 侵蝕與膨脹處理

形態學的運算會先掃描輸入影像的每一格像素，再對其做集合運算，

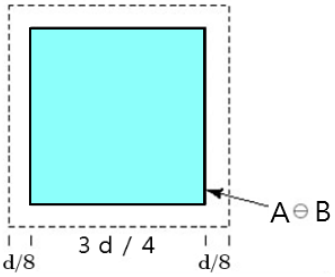
以圖 3-3-3B 為例將依結構元素 B 掃描輸入影像 A，

將 A 之中所有元素以 B 之結構擴展，

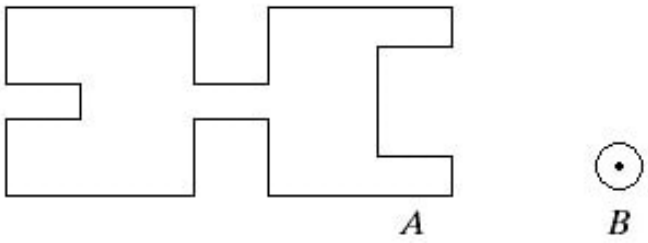
並產生如圖中灰色範圍標示之新影像，此範例為膨脹。



▲圖 3-3-3B 膨脹



▲圖 3-3-3C 侵蝕

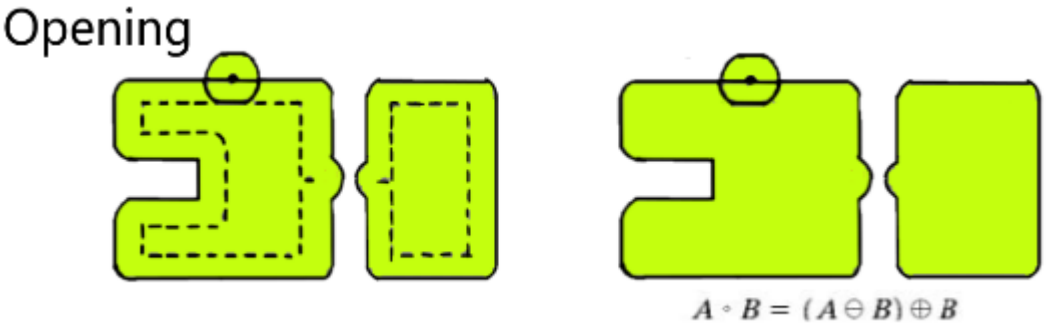


▲圖 3-3-3D　A、B 示意圖

侵蝕與膨脹的順序也會使效果有所不同，

若先侵蝕再膨脹稱為 Opening，可以將圖形凸出的銳角給鈍化，

使目標物輪廓得以平滑、將過於細微連接點消除。

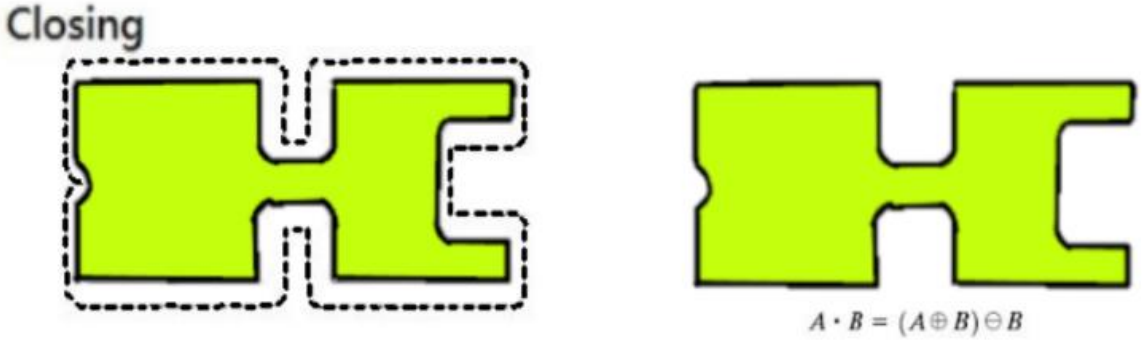


▲圖 3-3-3E Opening

反之則稱為 Closing，

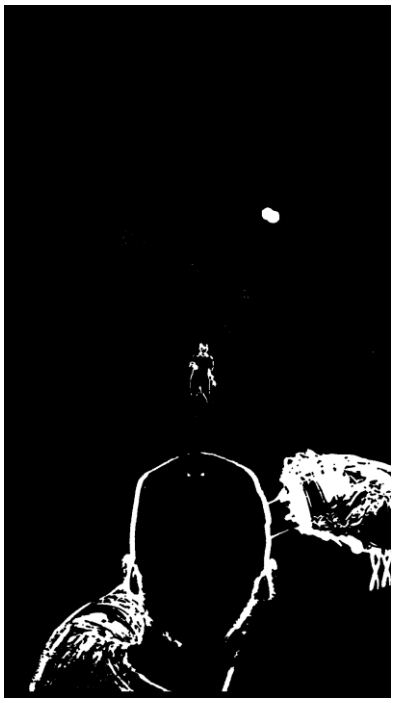
將圖形內陷的銳角給鈍化，也會使目標物因連接過於細小，

而相似於斷點的情況增強連接，因此能夠處理裂縫、與填補缺洞。



▲圖 3-3-3F Closing

4. 邊緣檢測



▲圖 3-3-4A 圖像經由上述影像處理之結果

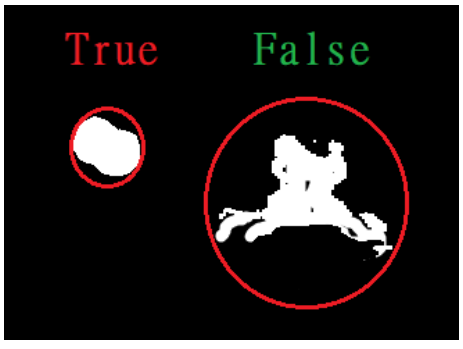
經過上述兩種影像處理之後，可以發現圖像上的資料幾乎都為連續的。

因此我們再使用邊緣檢測，計算出每個資料分區的面積。

接下來過濾尺寸，並設計一個圓形濾波器來檢測該分區為圓形的可能性。

圓形濾波器：先找出該資料區塊的最小外接圓，

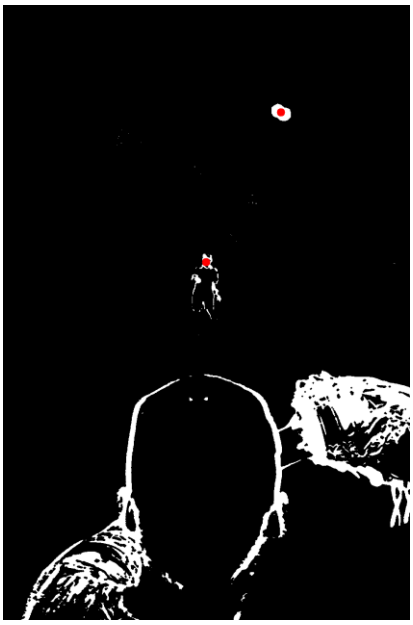
再計算資料面積是否佔足圓面積比例。



▲圖 3-3-4B 圓形濾波器示意圖

(四) 資料處理

1. 資料前置處理

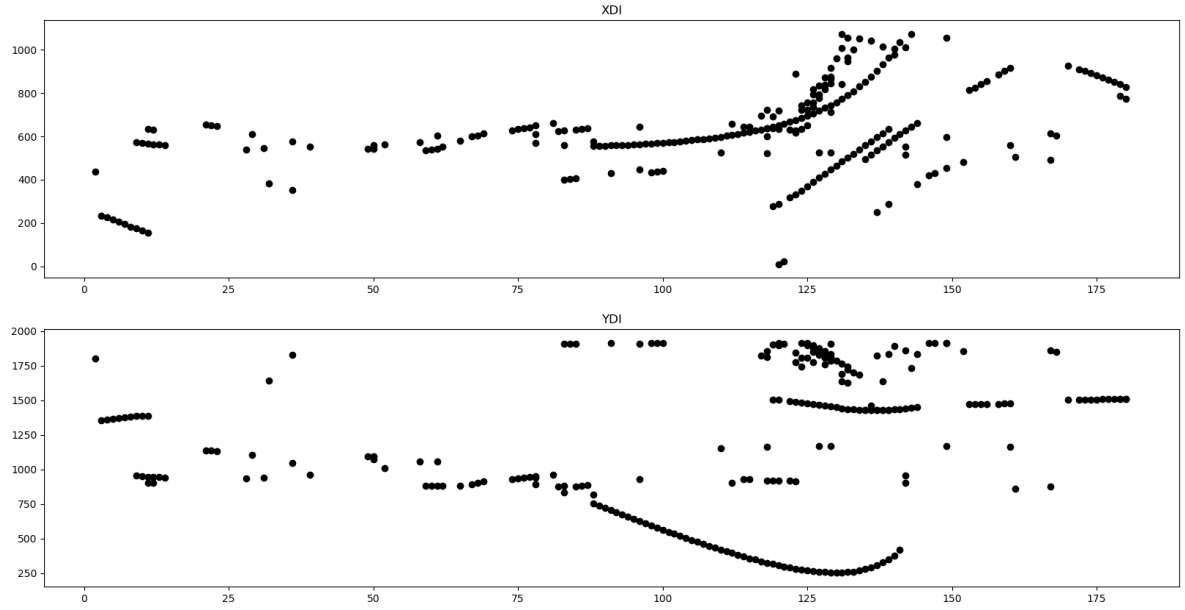


◀圖 3-4-1A 圓形濾波器輸出結果

將圖片以圓形濾波器過濾之後，會產生如圖 3-4-1A 那樣的座標點。

如果我們將整段投球影片都進行影像前置處理後，

以 X 座標/影片幀數、Y 座標/影片幀數 為座標，即可產生兩張圖表，



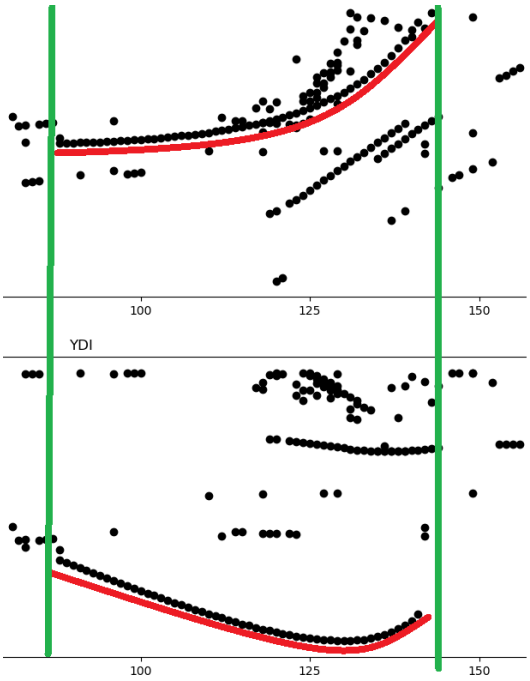
▲圖 3-4-1B XDI 圖(上) YDI 圖(下)

2. XYDI 軌跡連線尋找法

藉由觀察 圖 3-4-1B ，我們不難發現圖表上各有一條較長的曲線，

這兩條曲線有著連續性的變化，並且前面也經過圓形濾波器的檢查，

這時候，我們幾乎可以篤定這條線就是我們所需要的球路了！

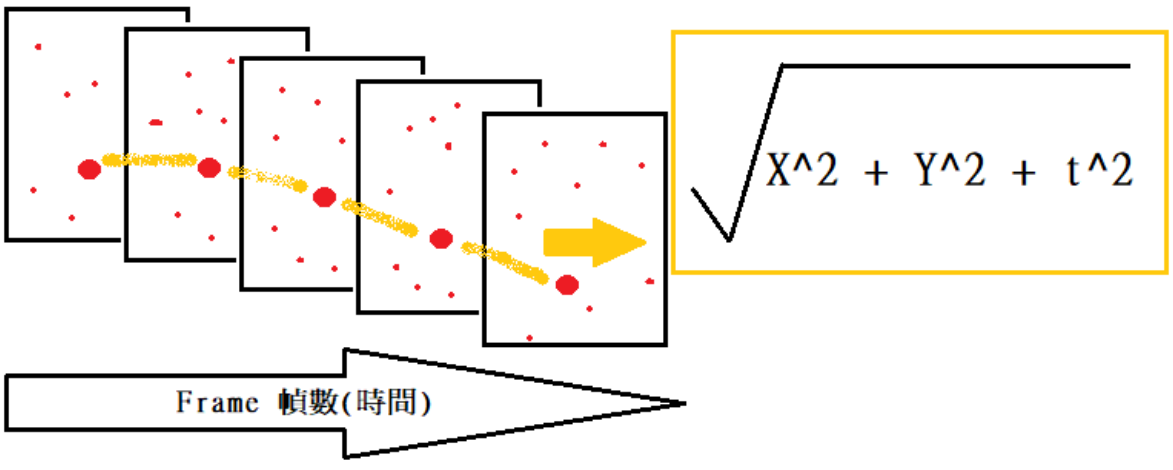


▲圖 3-4-2A 圖片觀察結果　　▼圖 3-4-2B 演算法概念

演算法設計：

我們能用三維的距離公式來判斷兩幀之間的軌跡變化量，

所以只要判斷相鄰兩幀的兩點軌跡變化量，**<u>很快就能找出目標曲線。</u>**



(五) 擊球員影像深度學習

做完了軌跡的判別，接下來我們要找出好球帶位置，

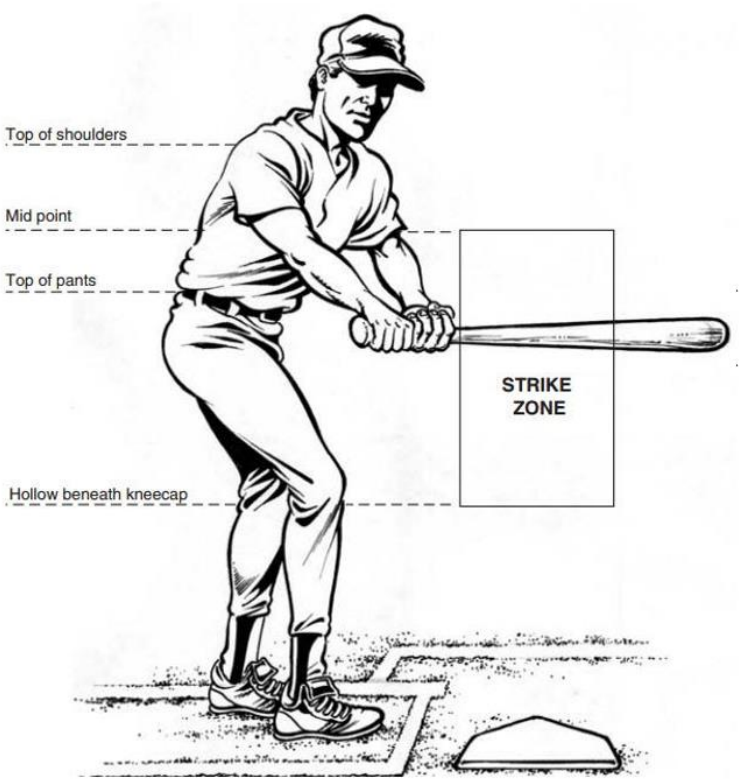
因為好球帶位置會隨著擊球員的身高、打擊姿勢有所差異，

根據 MLB 規則[2]

" the area over home plate from the midpoint between a batter's shoulders and the top of the uniform pants -- when the batter is in his stance and prepared to swing at a pitched ball -- and a point just below the kneecap."

也就是說從肩膀和制服褲子的頂部之間的中點，

到準備打擊時於膝蓋骨下方的一點是好球帶的高度，如下圖：



▲圖 3-5A MLB 好球帶規定

根據資料，我們選出打者與其肩膀、膝蓋還有鞋子來做為深度學習的目標，

我們使用 Yolov5s 的網路模型來做為訓練工具，

Yolov5 是基於 pytorch 實現的，架構與 YOLOv3、YOLOv4 有一些不一樣，

輸入端方面，YOLOv5 和 YOLOv4 同樣是採用 Mosaic 數據增強。

Mosaic 數據增強

Mosaic 是參考 CutMix 數據增強的方式，

以兩張圖片進行拼接做為訓練資料—改良而成的，

用 4 張圖片隨機縮放、剪裁、排放的方式進行拼接，

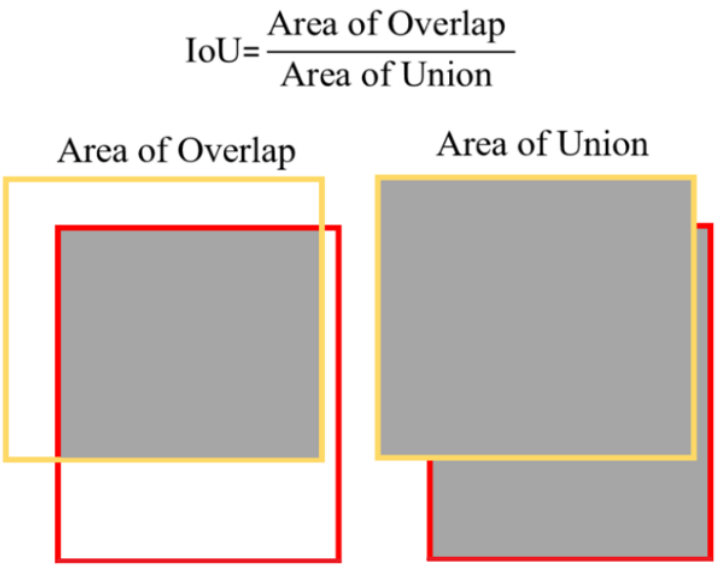
能夠更加豐富數據集，而縮放時增加的小目標也能讓網路模型性能更好。

▲圖 3-5B 本專題訓練過程中的 Mosaic 數據增強資料

IOU & GIOU_Loss

Prediction 方面 YOLOv5 採用 GIOU_Loss 做 Bounding box 的損失函數，

在此先介紹 IOU，簡單來說就是兩物件的重疊(overlap)的比例，



▲圖 3-5C IOU 示意圖

IOU_Loss：

如圖可看出 IOU 的 loss 其實主要是交集/聯集，但也存在一些問題：

狀態 1 的情形下，預測和目標框不交集，IOU=0，

並無法反映出兩者間的距離，亦無法改善兩個框不相交的情形。

狀態 2 和 3，當兩預測框大小相同，IOU 也相同時，

IOU_Loss 無法分辨其相交狀況的不同。

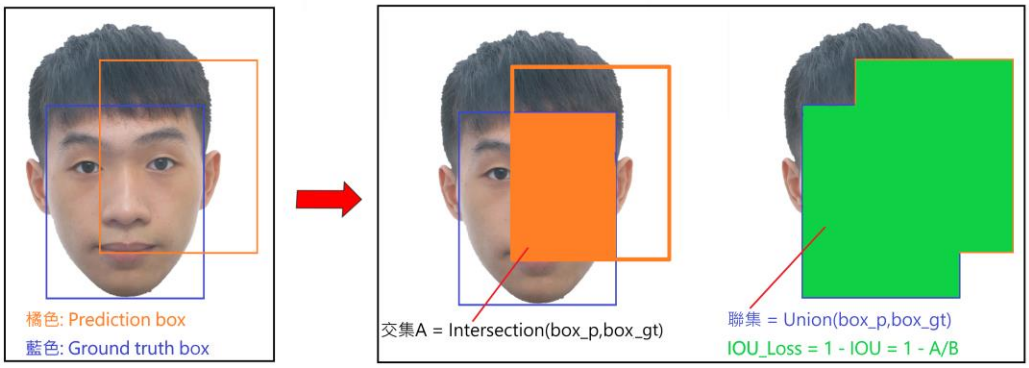
GIOU_Loss：

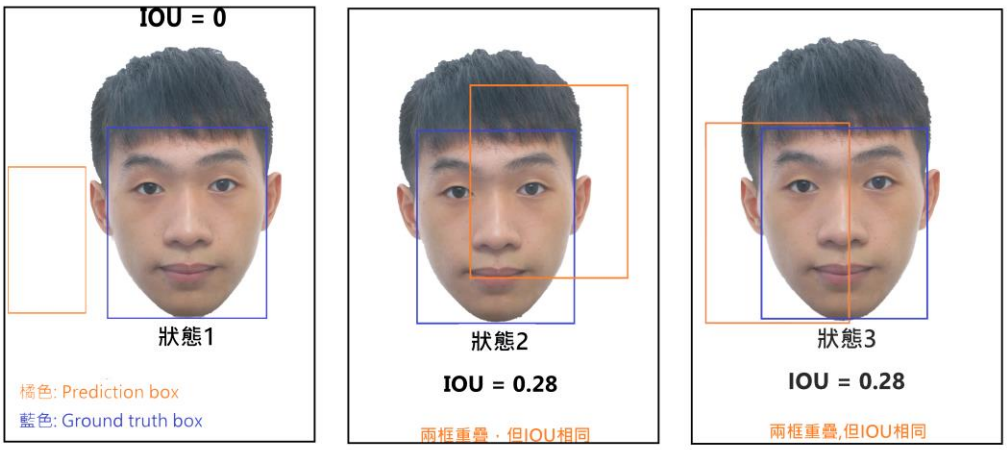
可從圖中看出增加了相交範圍的衡量方法，

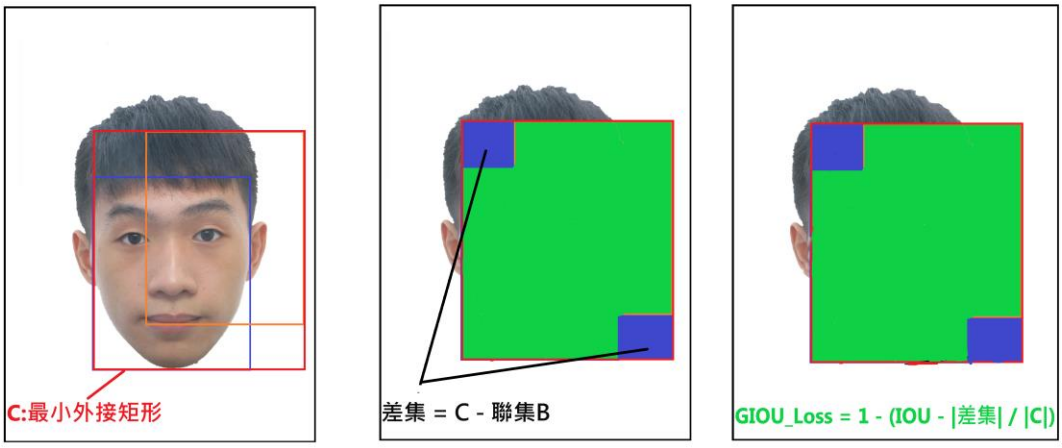
解決部分 IOU_Loss 情況的缺失。

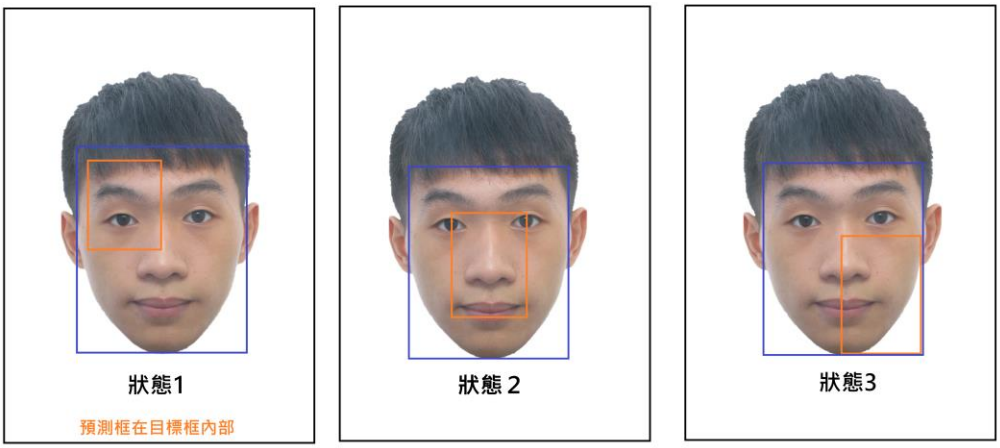
但當預測框在目標框中且大小相同時，GIOU 值是相同的，

而此時又出現了 IOU 無法區分相對位置關係的情形。


橘色: Prediction box
藍色: Ground truth box
交集A = Intersection(box_p,box_gt)
聯集 = Union(box_p,box_gt)
IOU_Loss = 1 - IOU = 1 - A/B


IOU = 0
狀態1
橘色: Prediction box
藍色: Ground truth box
狀態2
IOU = 0.28
兩框重疊，但IOU相同
狀態3
IOU = 0.28
兩框重疊,但IOU相同


C:最小外接矩形
差集 = C - 聯集B
GIOU_Loss = 1 - (IOU - |差集| / |C|)


狀態1
預測框在目標框內部
狀態 2
狀態3

模型品質分析

介紹完 YOLOv5 在建構過程所使用的技術後，

接下來讓我們來講辛苦 label 加上花了大把時間訓練出來的模型，

該怎麼進行評估，不過在那之前先解釋一下幾個重要指標的意義。

Precision & Recall：

假如以檢測貓咪為例：

		真實情況	
		貓	非貓
模型	貓	是貓且判斷為貓 (TP)	非貓但判斷為貓 (FP)
	非貓	是貓但判斷為非貓 (FN)	非貓且判斷為非貓 (TN)

▲圖 3-5D

Precision = 「所有被檢測為貓」中「被分類為貓且是貓」的比例。

= TP / (TP + FP)

Recall =「所有貓」中「被分類為貓且是貓」的比例。

= TP / (TP + FN)

也就是說單純 Precision 高並沒有意義，

假若有 50 個真樣本和 50 個負樣本，

而模型將 49 個真樣本和 50 個負樣本判斷為負，

剩下的一真樣本判斷為真，那麼此一模型的 precision 依然為 100%。

但這個模型很明顯地不符合預期，

此時就須依靠 Recall 值來衡量其找出所有真樣本的能力，

以上例: 1/50 = 0.02，就能明顯得出此模型的訓練結果並不好。

mAP@0.5 & mAP@0.5:0.95

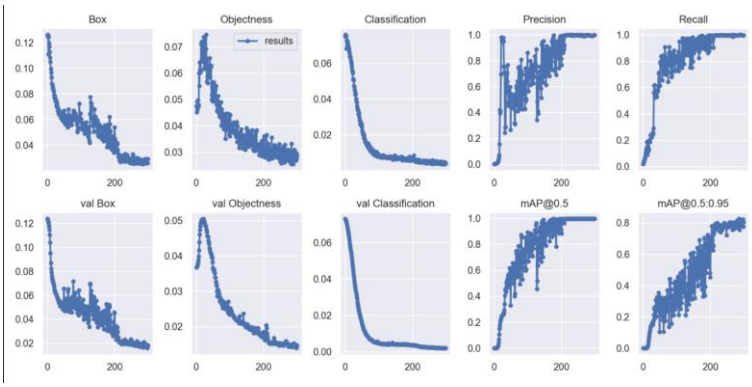
就是 mAP 是用 Precision 和 Recall 作為兩軸作圖後圍成的面積，

m 表示各分類之平均，@後面的數表示判定 iou 為正負樣本的閾值，

@0.5:0.95 表示閾值取 0.5:0.05:0.95 後取均值。

實際判別範例：

一般來說 precision 和 Recall 的波動幅度不大，會有較好的訓練結果。



▲圖 3-5E 第一個模型

圖 3-5E 為本專題研究中第一個訓練出來的模型，

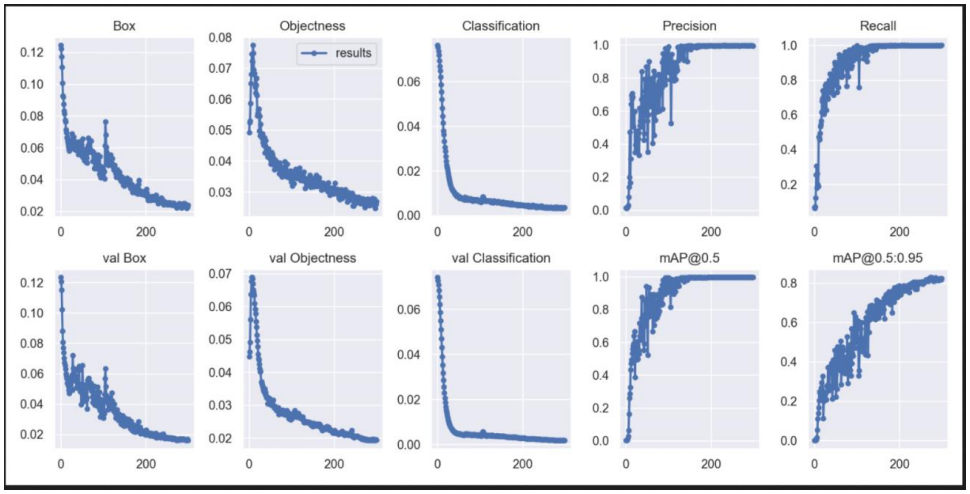
訓練用的資料只有 50 幾張，此時的 label 並沒有任何標記上的技巧，

而 P 和 R 的波動幅度也非常大，

GIOU 損失函數均值、目標檢測遺失值也當然很大，

最後實際檢測的結果數值當然也都不高。

◀圖 3-5F 模型一之檢測結果



▲圖 3-5G 增加資料量的模型 8

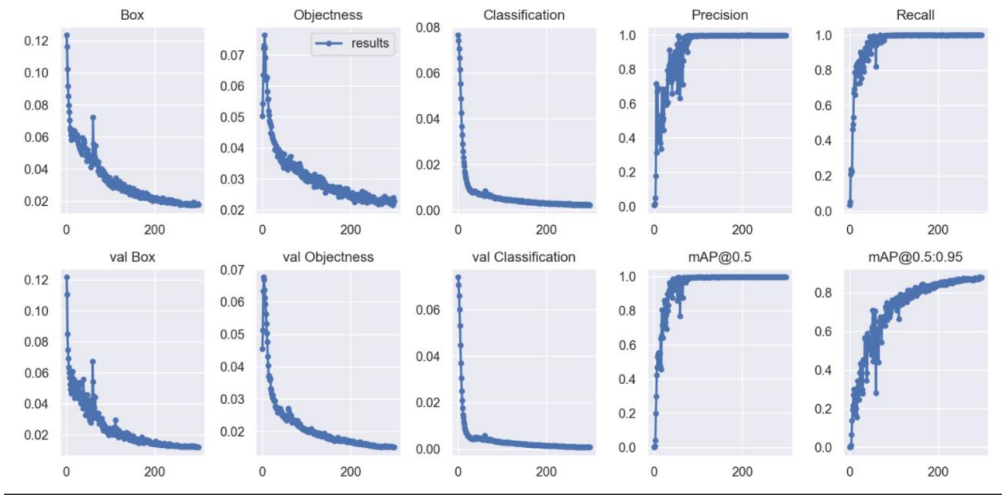
圖 3-5G 為將資料數量大量增加後的模型，P 和 R 的波動有所改善，

不算十分良好，但遺失均值也有所許改善，

不過此模型在實測上依然有一些問題，

標記範圍過大以及特定角度無法正確辨識。

▲圖 3-5H 模型 8 檢測結果



▲圖 3-5I 增加資料量與改善 label 的模型 12

此模型為更多資料的訓練，不過稍微有所不同，

我們將之前所有 label 做重新調整，將每個標記範圍做統一，

確保每個 label 的位置有正確框住目標特徵，

此舉也有部分是因為資料量依然太少(只有數百張，還不到千張)，

而這麼做也確實目標的偵測有明顯的改善也更加穩定，

P 和 R 的波動幅度也平緩了許多，

不過可惜的是手肘部分的辨別依然是個問題，

也許在有更多不同集球姿勢的資料後能夠更精確其偵測。

▲圖 3-5J 模型 12 之檢測結果

(六) 失敗經驗

1. 使用 Kivy 來開發 Android 應用程式：

Kivy(https://kivy.org/)是一款 Python module，主要功能為開發桌面版應用程式，實際編寫方式類似 Google 的 Flutter。

一開始打算採用 Kivy 來開發的原因是，Kivy 是一款強大的跨平台應用程式套件，只要編寫一次代碼，便可轉換成 Windows、Linux、Android、IOS 等各種系統的圖形化介面。並且 Kivy 開發語言是用比較快速上手的 Python。

但是，Kivy 跟 Python 終究不是 Android 系統原生就支援的，在許多方面的功能就會受到限制。

(1) 相機元件相容性問題： Kivy 內建的相機元件並不支援 Android 系統，且 Python 常用的 OpenCV 也無法解決此問題，因此對於本應用程式需要有錄影功能是一大傷害。

(2) Kivy 打包成 Android 應用程式的編譯問題： Kivy 將桌面應用程式轉換至 Android 平台時，會將 Python 程式編譯成 Cython，進而再轉換成類似 Java 的套件。由於以上步驟，許多 Python module 會出現編譯上的問題，例如本應用程式最重要的 OpenCV-Python。

2. 使用 cv2.HoughCircles()來尋找軌跡：

我們目前尋找軌跡的方法為：連續影格相差>形態學>邊緣檢測>XYDI，

本來規畫是利用 HoughCircles，整體方法為：

連續影格相差>形態學>HoughCircles。

但是 HoughCircles 再找完美的圓上比較拿手，像是經過形態學變化過後的棒球就十分難辨認，需要把 HoughCircles 的條件都調至寬鬆才有機會尋找到正確球路位置，但這樣畫面上會出現許多不相干的錯誤點，導致拖累整體準確度及速度，算出來又會有許多點，有點畫蛇添足。

最後找到了這篇論文

A Trajectory-Based Ball Tracking Framework with Visual Enrichment for Broadcast Baseball Video

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.109.180&rep=rep1&type=pdf

並參考該篇論文中的 XDI/YDI 物理模型想法，

最後開發出 **XYDI 軌跡連線尋找法**作為程式尋找軌跡使用。

四、結果及討論 Research Results an Conclusions

以下為實做出來的 APP 頁面

1. 應用程式 Logo

◀圖 4-1A 應用程式 Logo

2. 應用程式進入畫面(主畫面)

◀圖 4-2A 應用程式進入畫面(主畫面)

開啟本應用程式之後，會看到如此的畫面。

左上角的 icon 是可以點選的，可以叫出設定選單。

右上角為影片來源，左側為錄製影片，右側為選擇影片。

第一次開啟本應用程式時，會跳出要求授予應用程式權限的對話框，要求使用者提供足夠運行本程式之權限。

每當應用程式重啟時，也會檢查是否缺少權限。

3. 設定選單

◀圖 4-3A 應用程式設定選單

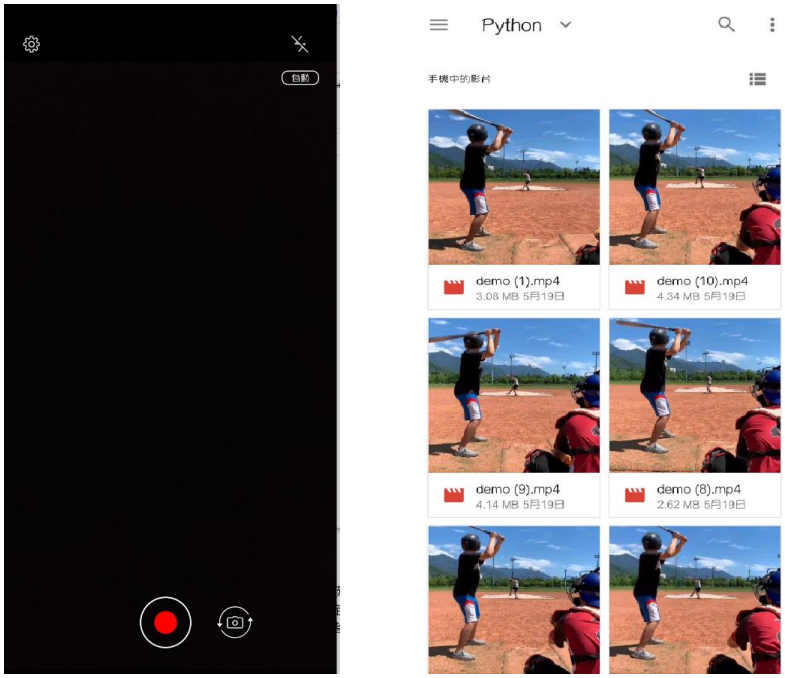
在點選主畫面左上角的 icon 選擇設定之後，會進入設定選單。

此設定選單是繼承了 PreferenceFragmentCompat class，

因此設定會儲存在手機內部儲存空間之中，關閉應用程式會保留所有設定。

當設定完成之後，再次點選左上角的圖示即可保存設定並退回主頁面。

4. 選擇影片來源



◀圖 4-4A 影片來源

在主頁面中點選右上角影片來源按鈕即會跳到本畫面。

5. 調整影片擷取範圍及設定本壘板位置

◀圖 4-5A 選擇影片擷取範圍

當選擇完影片來源之後，就會進入到此頁面，

並同時 Post 起始影像至 AI Lab 架設的伺服器(http://ecg-dais-tz.apulse.ai:8080/services/)

本頁面一共就三個按鈕，分別為左右兩側的縮放按鈕，及下方的設定按鈕。

中間的預覽視窗是可以點選的，所以調整好你要的方框大小，

即可點選預覽窗來設定位置，調整完擷取範圍之大小位置後，

即可按下下方的『設定』按鈕前往下一畫面設定本壘板位置。

設定本壘板位置的目的在於，如果我們想要定位好球帶，

就必須知道本壘板位置，有了本壘板位置即可更加精確地顯示好球帶範圍。

6. 執行應用程式功能

◀圖 4-6A 應用程式就緒

當設定完影片範圍跟本壘板位置之後，就會回到我們最熟悉的主頁面了！

畫面中會顯示使用者剛才設定的範圍，可供使用者進行確認。

若您的手速不是非常之快，抑或是您的的網路速度十分正常，

這時候剛剛 Post 的資料應該已經回傳完畢，可透過設定內的『顯示 Post log』

來查看當前 Post 狀態，若顯示 Post ERROR，請重新選擇影片來源。

當一切就緒之後，便可點選下方的『GO』開始執行，

此時的執行速度取決於使用者的手機效能等級、與影片解析度和品質。

7. 應用程式執行完畢

◀圖 4-7A 應用程式執行完畢

若應用程式執行順利，你應該會看到如圖所示之畫面，

如果沒有產生好球帶，請去檢查設定內的『打者開關』是否關上，

或是打開『顯示 Post log』來確認 Post 狀態。

如果沒有正常顯示球路，請調整影片擷取範圍，及有機會成功輸出。

本應用程式會自動將圖片輸出至「/storage/emulated/0/eMSZ/」資料夾下。

8. 應用程式執行測試

我們總共使用了 15 組影片做測試，
(檔案位置：https://github.com/410725008/eMobileStrikeZone/tree/main/Python)

其中 10 組是有打者(擊球員)的測資，另外 5 組是只有投手之測試影片。

有打者的 10 組影片，有 5 組是直接可以辨識出球路，有 3 組則是手動選取影片範圍後，即可辨識出球路。

只有投手的 5 組影片，在進入『設定－關閉打者』後，皆可辨識出正常球路。

經過測試，只要場地背景乾淨，不要有過多干擾，打者(擊球員)與捕手沒有過於激烈動作，有 80%的信心可以辨識出 80%以上的球路。

以下為問題討論

(一)機器學習是否能在檢測棒球軌跡上幫忙？

我們認為是一定可以的，不管是在物件偵測，

還是資料模型的建立之上，機器學習都能發揮他的長才。

像是預測接下來的球路，或是分類球路軌跡等。

(二) 利用本應用程式得出的軌跡資料，是否能進一步的拓展應用?

本來我們是打算在最後加上一個訓練球路模型，

進而建置一套只需要其中幾個斷點，就能產生完整球路。

對於擁有完整數據的球路來說，

可由此模型來判斷該球路之球速、球質、變化量、變化角度等…

各種資訊，並且判斷該球路是否真正通過好球帶之平面。

往後的空閒時間，

我們將繼續完成此專題研究未完成的遺憾，

為自己所熱愛的運動盡一份心力！！！

五、參考文獻 References

[1] A Trajectory-Based Ball Tracking Framework with Visual Enrichment for Broadcast Baseball Videos.
[2] https://www.mlb.com/glossary/rules/strike-zone
[3] 機器學習\統計方法: 模型評估-驗證指標(validation index)

[4] 深度學習系列-什麼是 ap-map

[5] Kivy \ https://github.com/kivy/kivy

[6] Chaquopy \ https://chaquo.com/chaquopy/doc/current/

[7] MediaExtractor\ https://developer.android.com/reference/android/media/MediaExtractor

[8] MediaCodec\ https://developer.android.com/reference/android/media/MediaCodec