一、原理

第一部分 YOLOv3:

第二部分 雙目測距:

1.基本介紹

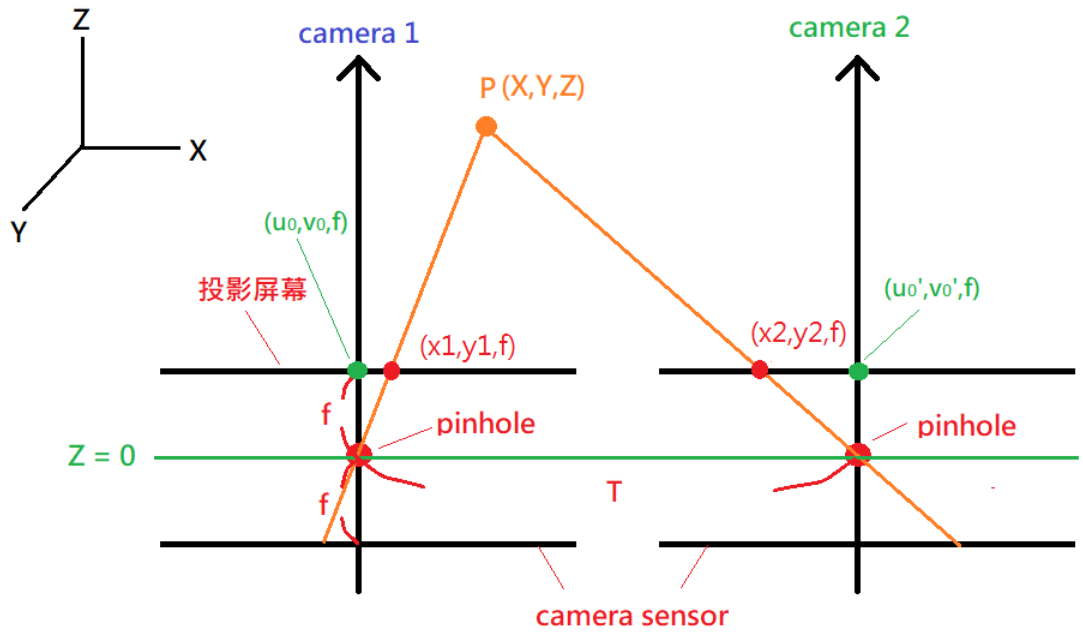
通過對兩幅圖像視差的計算，直接對前方景物（圖像所拍攝到的範圍）進行距離測量，而無需判斷前方出現的是什麼類型的障礙物。所以對於任何類型的障礙物，都能根據距離信息的變化，進行必要的預警或制動。雙目攝像頭的原理與人眼相似。人眼能夠感知物體的遠近，是由於兩隻眼睛對同一個物體呈現的圖像存在差異，也稱“視差”。物體距離越遠，視差越小；反之，視差越大。視差的大小對應着物體與眼睛之間距離的遠近，這也是 3D 電影能夠使人有立體層次感知的原因。

2.優缺點比較

與單目測距相比，單目測距需要不斷更新和維護一個龐大的樣本數據庫，才能保證系統達到較高的識別率，且無法對非標準障礙物進行判斷，其距離並非真正意義上的測量，準確度較低，而若我們採用雙目測距則恰好能解決上述的問題。

在成本上若採用雙目測距則價格會明顯比單目測距來得高，但若是和雷射光測距相比則又相對來得低，因此利用雙鏡頭進行影像測距還是有其一定的研究價值。

3.雙目成像理想模型



其中 P 為物體實際所在位置，T 為兩台攝影機透鏡之間的距離，而 pinhole 為成像穿過透鏡點，並將影像投影在 camera sensor 上，根據以上關係圖我們可以寫出以下公式:

$$\frac{Z}{\mathrm{T}} = \frac{Z-f}{\mathrm{T-x1-x2}}$$

因此可推導出

$$\mathrm{Z} = \frac{f*T}{\mathrm{x1-x2}}$$

其中焦距 f 以及 T 均為固定常數，而 x1 和 x2 均會隨著 P 點位置而有所不同。

當我們計算完 Z 之後 X 以及 Y 點位置便可輕鬆取得
(以下 X 與 Y 為以 camera 1 為中心軸)

$$\mathrm{X} = \frac{Z}{\mathrm{f}} * x1 = \frac{T}{\mathrm{x1-x2}} * x1$$

$$\mathrm{Y} = \frac{Z}{\mathrm{f}} * y1 = \frac{T}{\mathrm{x1-x2}} * y1$$

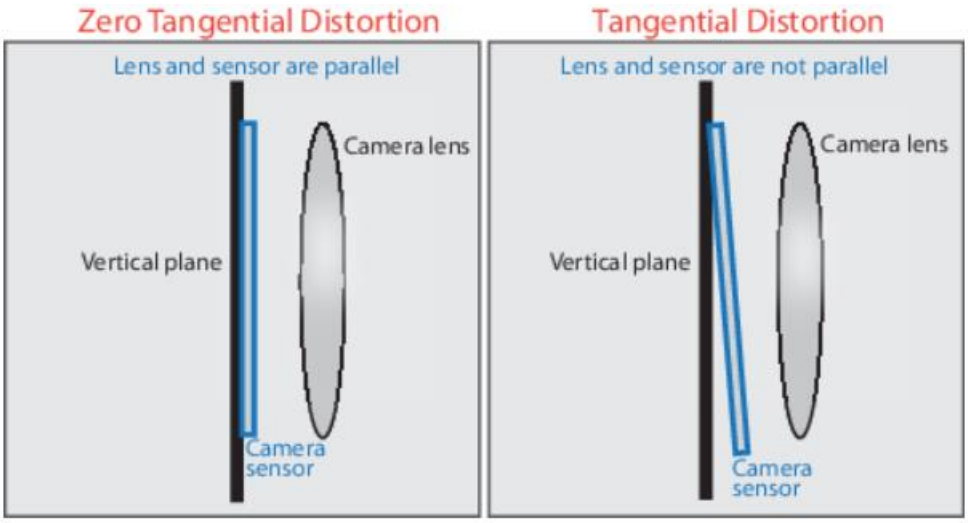
然而若想透過以上公式在現實中進行距離上的計算，還得必須將參數 f 及 T 給量測出來才可以進行實際上的應用。

4.相機畸變問題

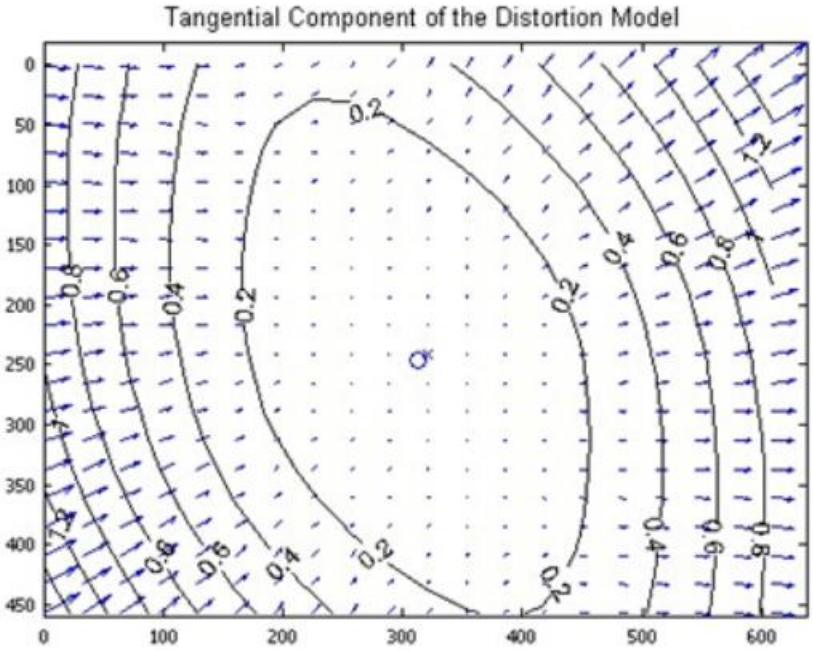
在理想上，我們會希望兩台攝影機的成像能夠完整的被映射在同一個平面上，但實際上並非如此理想，因此我們必須透過數學的方式來矯正這些問題。

(a) Tangential Distortion

切向畸變是由於透鏡本身與相機傳感器平面（成像平面）或圖像平面不平行而產生的，這種情況多是由於透鏡被粘貼到鏡頭模組上的安裝偏差導致如圖所示:

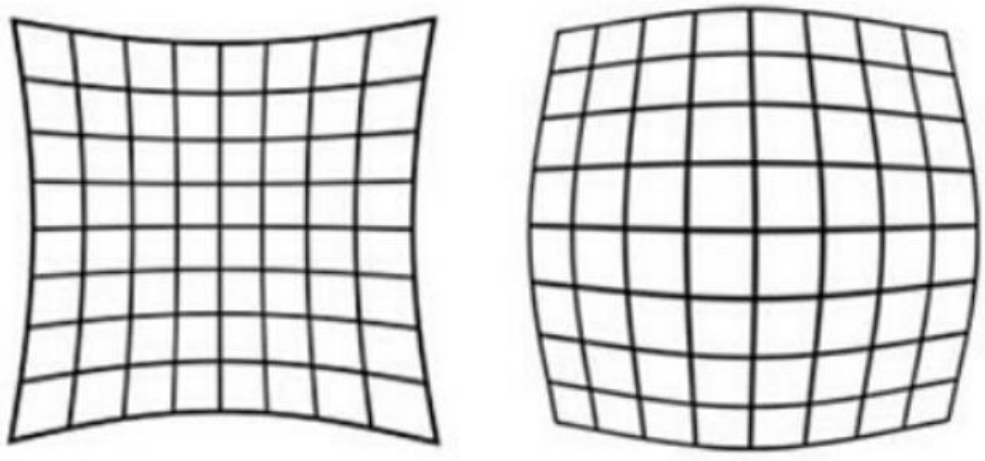


下圖顯示某個透鏡的切向畸變示意圖，大體上畸變位移相對於左下到右上角的連線是對稱的，說明該鏡頭在垂直於該方向上有一個旋轉角度。



(b) Radial Distortion

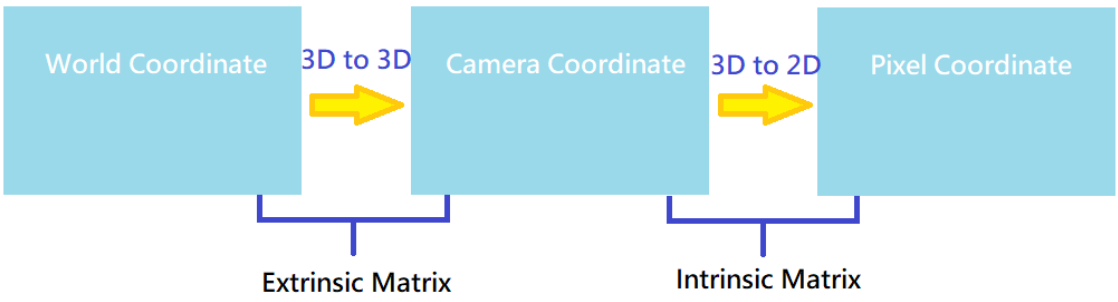
Radial Distortion 發生的原因在於當光線經過透鏡的邊緣時，其彎曲的程度比在透鏡的中心為大，這種畸變在普通廉價的鏡頭中表現更加明顯，徑向畸變主要包括桶形畸變和枕形畸變兩種。以下分別是枕形(左)和桶形(右)畸變示意圖：



5.相機畸變及旋轉的修正

由於雙鏡頭測距是從鏡頭中的成像以及焦距 f 還有兩個鏡頭之間的距離 T 去推算現實世界中鏡頭中心與測量目標的距離，且為了使得現實中的成象能夠更精準的投影在鏡頭的平面上，我們必須把剛才所介紹的畸變問題考慮進來並且進行修正，而最好的工具便是矩陣代數中的座標轉換，透過矩陣的座標轉換以及矯正畸變的數學公式能夠使得三維世界的座標更完美的被映射在二維的平面座標上。

在開始之前先介紹座標轉換的流程，也就是世界座標(3D)經過相機外部參數矩(Extrinsic Matrix)的作用轉換成相機座標(3D)，而相機座標(3D) 再經過相機內部參數矩陣(Intrinsic Matrix)的投影作用轉換成影像座標(2D)，如下圖所示:



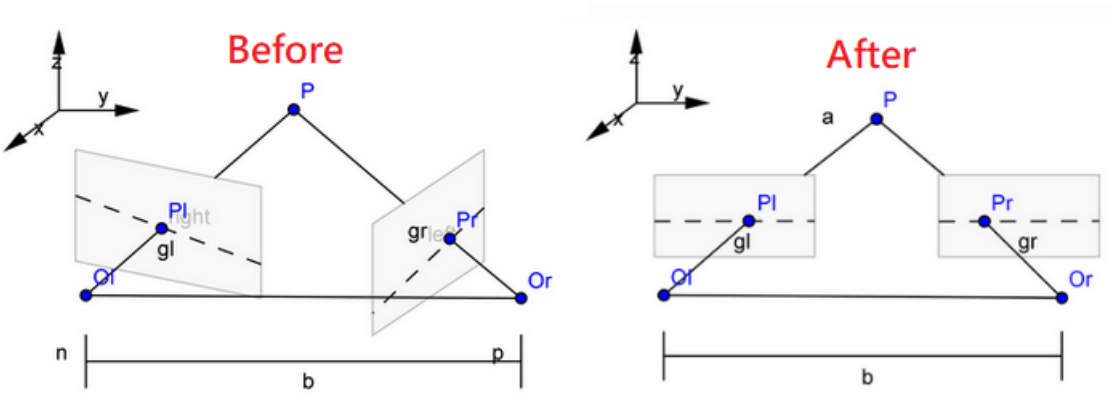
$$Zc\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Camera coordinate　Intrinsic Matrix　Extrinsic Matrix　World Coordinate

對於整體流程有了些概念之後以下將開始一一介紹各個矩陣所代表的涵義

(a) Rotation Matrix

由於我們的鏡頭是由兩台攝影機所合成的，在合成的過程中必定會有些許的誤差產生，因此勢必要透過 3D 旋轉矩陣來修正此問題，如圖下所示:



(b) Extrinsic Matrix

Extrinsic Matrix (外部參數矩陣) 是在拍攝物體固定的情況用來描述相機的旋轉及位移；或是相反的，在相機固定的情況下，用來描述拍攝物體的旋轉及位移，透過 Extrinsic Matrix 可將 World Coordinate 轉換至 Camera Coordinate，為 3D-3D 的轉換，矩陣如下所示:

$$[R\,|\,t] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right]$$

旋轉矩陣 (Rotation Matrix)　位移向量

(c) Intrinsic Matrix

Intrinsic Matrix (內部參數矩陣) 用來表示相機的內在屬性，通過它可以將三維相機坐標轉換為二維的象素坐標，如下所示:

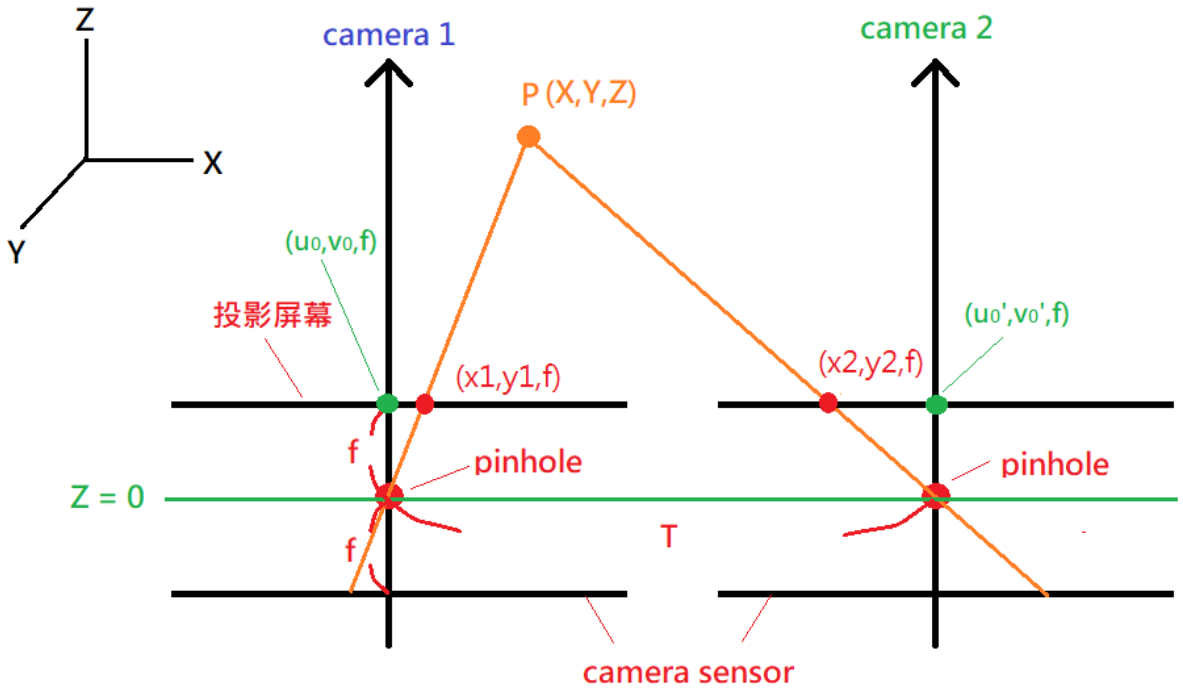
$$\begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

其中此矩陣之參數 fx、fy 為在焦距為 f 之下投影屏幕上 X、Y 根據光軸所對應之象素，Cx、Cy 則為光軸座標，且以上參數均為考慮 Radial distortion 及 Tangential distortion 後進行修改所得到的參數。

最後將 Intrinsic Matrix 與 Extrinsic Matrix 進行矩陣相乘便可以得到三維世界到二維相機投影的座標轉換，**此外在這裡三維世界座標向量是指投影屏幕上的座標，因此 Z 便為焦距 f。**

6.將相機二維座標回推至現實座標

透過前面所提到的方法將鏡頭畸變進行修正過後，我們便可開始利用在 **3.雙目成像理想模型**所提到的數學公式回推待測物體實際所在的世界座標，在下圖中 (U_0,V_0) 以及 (U_0',V_0') 分別表示 camera1 及 camera2 在各自的二維向素坐標下的束線中心坐標



根據先前所推導的以下三個公式，我們可以寫出矩陣數學式如下 (我們稱 Q 矩陣) :

$$\mathrm{Z} = \frac{f * T}{\mathrm{x1} - \mathrm{x2}}$$

$$\mathrm{X} = \frac{Z}{\mathrm{f}} * x1 = \frac{T}{\mathrm{x1} - \mathrm{x2}} * x1$$

$$\mathrm{Y} = \frac{Z}{\mathrm{f}} * y1 = \frac{T}{\mathrm{x1} - \mathrm{x2}} * y1$$

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -u_0 \\ 0 & 1 & 0 & -v_0 \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{-1}{T_x} & \frac{u_0 - u_0'}{T_x} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ d \\ 1 \end{bmatrix}$$

其中定義比較不一樣的是:

$$\mathrm{X} = \frac{\mathrm{X_w}}{W} \quad \mathrm{Y} = \frac{\mathrm{Y_w}}{\mathrm{W}} \quad \mathrm{Z} = \frac{\mathrm{Z_w}}{W}$$

$$\mathrm{d} = \mathrm{x1} - \mathrm{x2}$$

$$\mathrm{T_x} = -\mathrm{T}$$

以上矩陣所算出來的 X Y Z 是以 camera1 的束線為原點之坐標，因此若要將兩台攝影機之終點設為原點只要將 X+T/2 即可，最後攝影機中心點距離待測物體 P 點的距離即為:

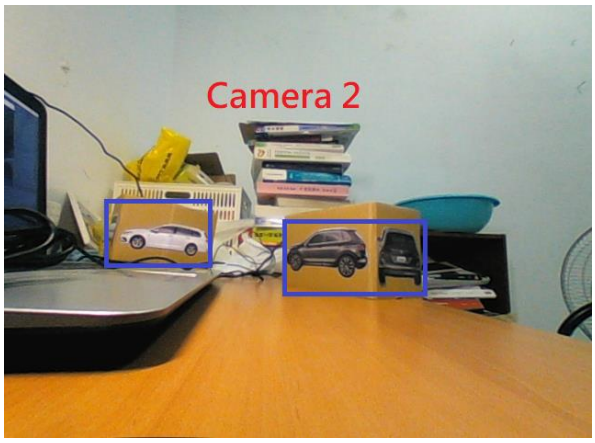
$$\sqrt{(\mathrm{X} + \mathrm{T}/2)^2 + \mathrm{Y}^2 + \mathrm{Z}^2}$$

第三部分 利用 Yolov3 達成雙鏡頭匹配:

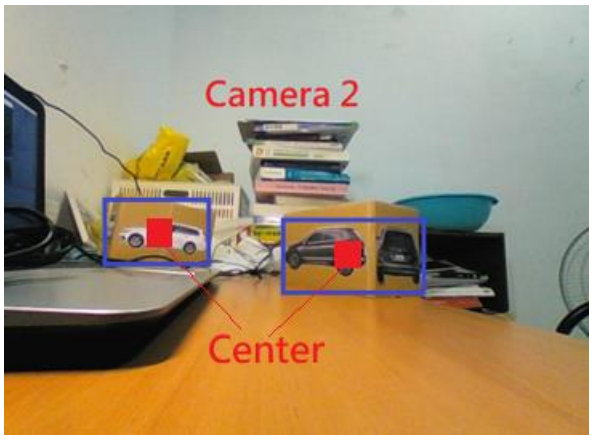
由第二部分我們可以知道，若我們想要透過 Q 矩陣知道待測物體 P 點所在的三維世界坐標位置的話，我們必須要先知道 P 點分別在 camera1 與 camera2 中所在位置的座標位置 (x1,y1)、(x2,y2)，否則一切皆為空談，而本專題便是透過 yolov3 來達成雙目匹配，而 yolov3 不只運算速度較 R-CNN 等神經網路快，且再雙鏡頭測距中也相較於傳統利用視差深度來進行雙鏡頭匹配的方法更具有一定的準確性，以下將開始進行解說。

1.Yolov3 找出物體所在位置

首先將 camera1 以及 camera2 通過畸變矯正後所得出的影像分別丟入 Yolov3 神經網路，而理論上會得出如下結果:



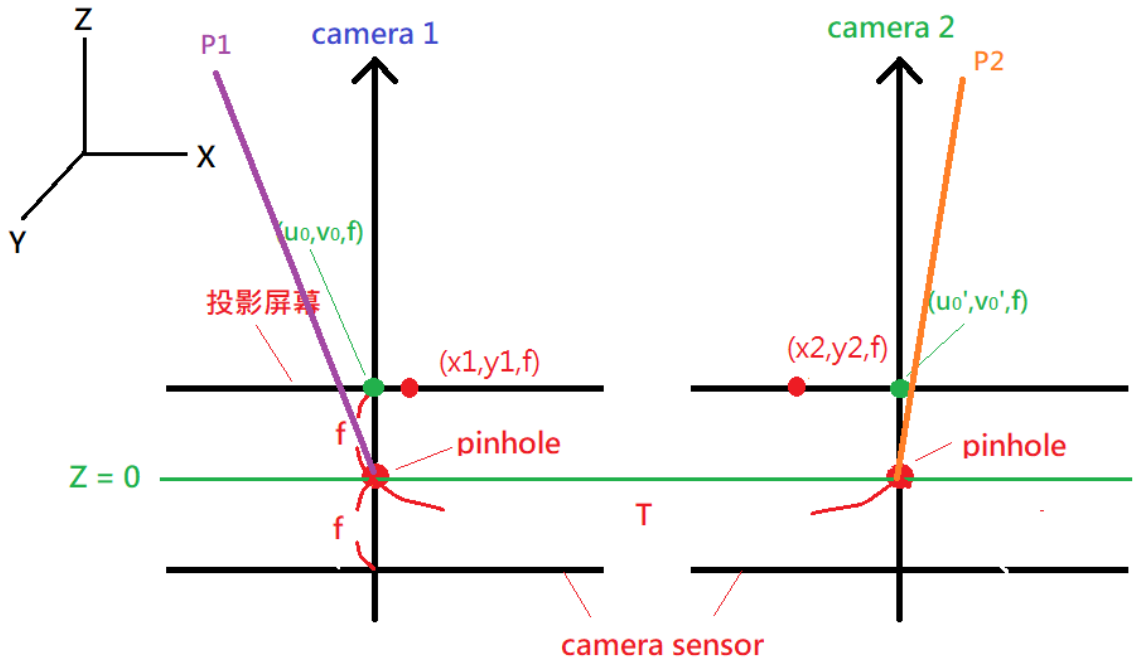
然而我們必須要知道白車與黑車所分別對應在 camera1 與 camera2 中的點事情才能進行下去，而透過 Yolov3 演算法我們可以知道白車與黑車分別的 x,y 坐標以及寬度跟高度，因此在專題中我們透過此資訊找出各個物體所分別對應的中點，如下所示:



找出中點後，我們將 camera1 與 camera2 中紅色方塊區域的每個相素進行相減後平方總和(即為計算誤差平方)，將 camera1 與 camera2 中誤差為最小的兩個物件視為同點，而最後也成功達成了匹配。

2.如何進一步提升準確性?

透過上一個方法，我們可以在正常的情況下判別不同的物體，但若是在燈光較不理想，或者是車子為兩台顏色一樣的車時進行像素誤差平方的方法可能就會有出錯，因此我們可以再針對匹配的算法進行修正，在本專題中若判定物件在 camera1 與 camera2 中象素的距離大於 T (camera1 與 camera2 束線距離)，則不會將此視為同一物件，如下圖所示:



由圖中可知，P1 及 P2 明顯不可能為同一點，因此這樣做必定為合理的假設，在實際執行中也得到了非常好的效果如下圖所示，其中 Car2 及 Car3 均為紅車。

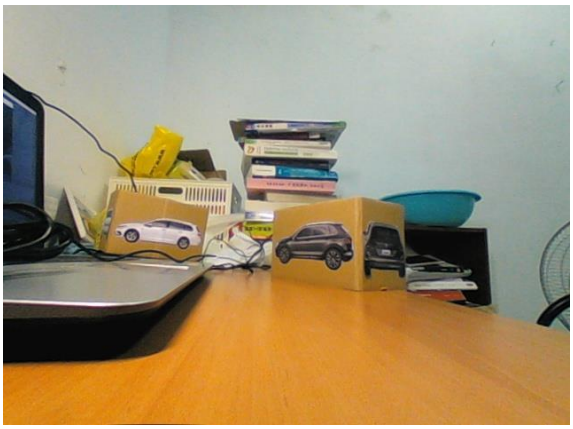
二、專題實現方法

由於許多理論部分已經在**原理**中帶過，因此在**專題實現方法**中不再做過多冗長的敘述。

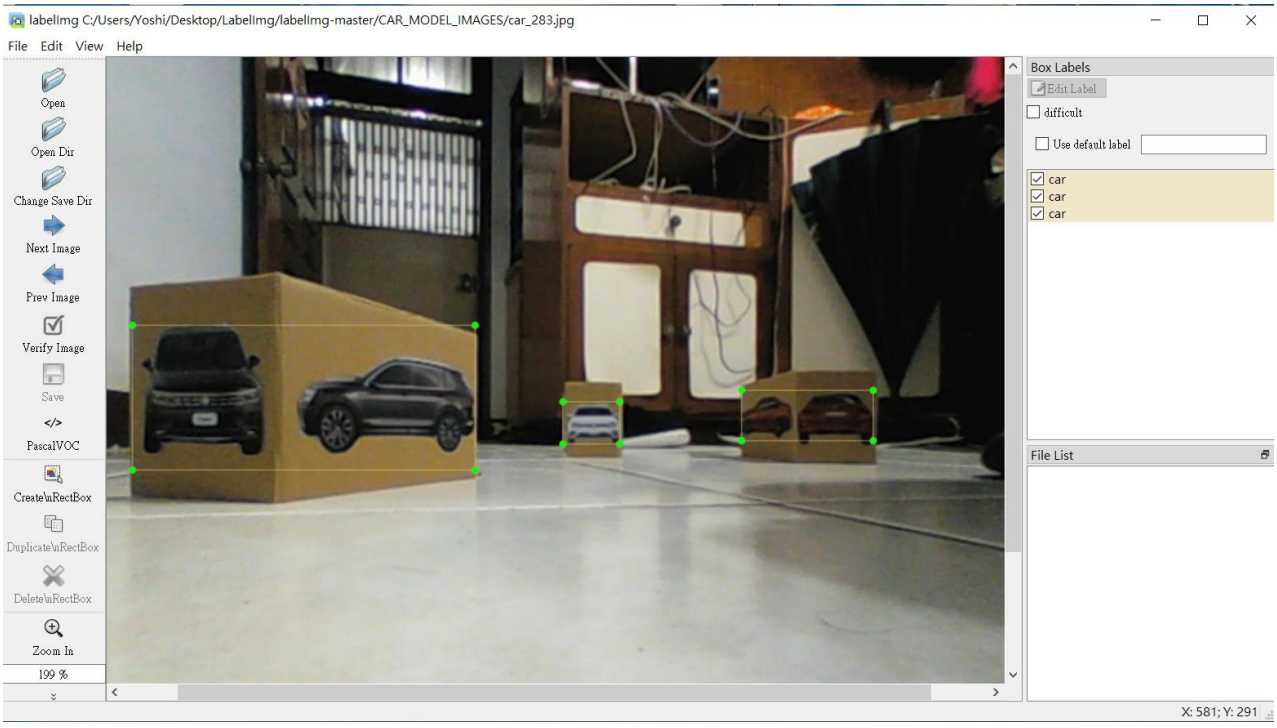
第一部分 YOLOv3 模型訓練:

1.蒐集車輛樣本，並針對各種不同角度、場景、距離進行拍攝

在各種不同的場景、燈光、距離下進行取樣。



2.利用 labelImg 對蒐集好的物件進行標記



3.利用 Yolov3 開發者所提供的套件"darknet"將標記過後的物件進行訓練。

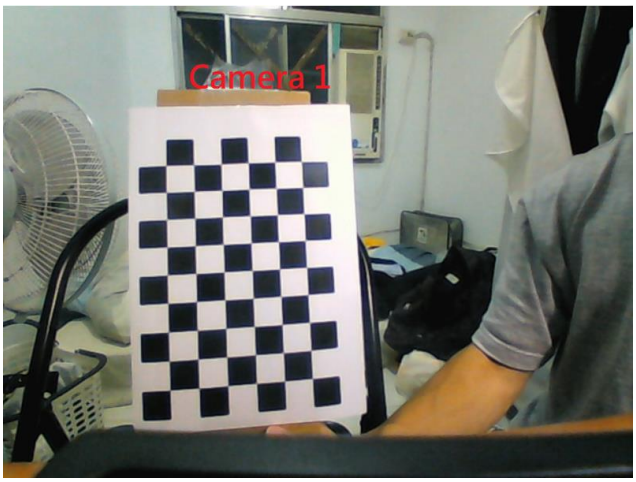
4.將訓練完的模型結合自己的程式碼做修改。

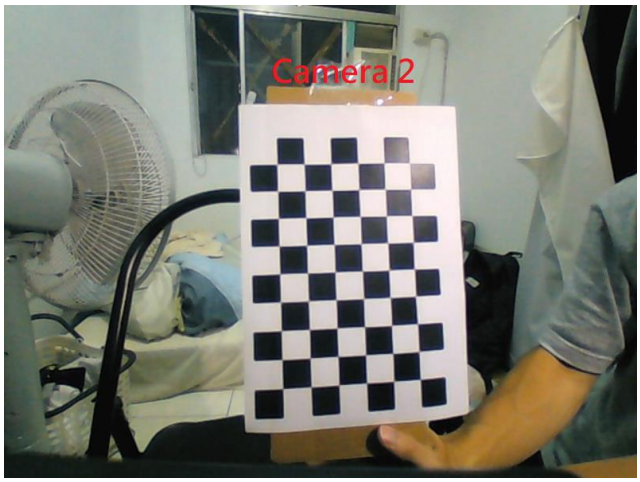
第二部分 攝影機參數以及畸變矯正:

1.將兩台單鏡頭攝影機利用木板及膠帶組合成雙鏡頭攝影機

2.利用 Matlab 將兩台攝影機進行參數測量

先準備黑白棋盤圖，並同時透過 camera1 及 camera2 同時拍下約 40 組照片如下所示:

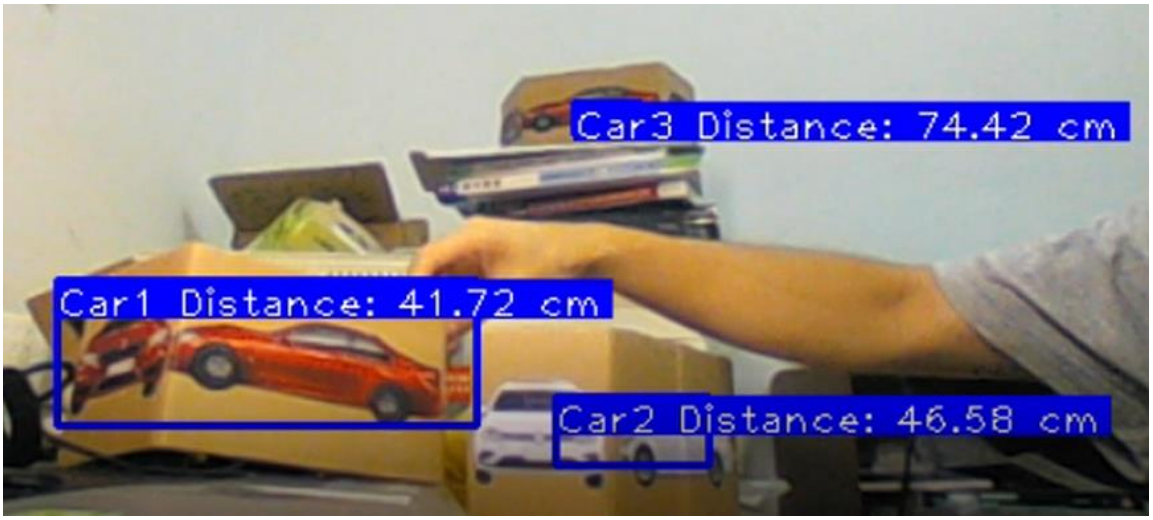




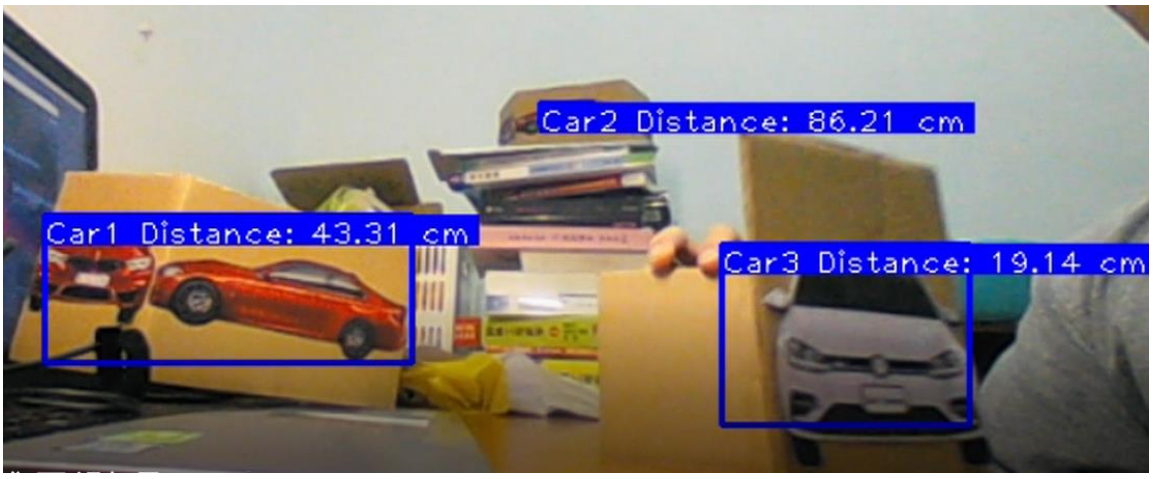
拍攝完後將 40 組照片丟進 Matlab 的 **stereo camera calibration** 之後便可以分別得出 Camera1 與 Camera2 的 **Intrinsic Matrix** (內部參數矩陣)、**FocalLength** (焦距)、**Radial Distortion** (徑向畸變)、**TangentialDistortion** (切向畸變) 以及兩者之間的參數 **TranslationOfCamera2** (兩鏡頭的束線距離)、**RotationOfCamera2** (兩鏡頭間的旋轉矩陣)。

3.利用 python 中現有的 cv2 套件結合 Matlab 中所得到的參數矩陣進行數學處理

三、成果展示


Car3 Distance: 74.42 cm
Car1 Distance: 41.72 cm
Car2 Distance: 46.58 cm


Car1 Distance: 34.92 cm
Car2 Distance: 44.80 cm


Car2 Distance: 86.21 cm
Car1 Distance: 43.31 cm
Car3 Distance: 19.14 cm

四、參考資料

https://www.cnblogs.com/zyly/p/9373991.html

http://silverwind1982.pixnet.net/blog/post/153218861

https://blog.csdn.net/dcrmg/article/details/52950141

http://zhixinliu.com/2016/11/15/2016-11-15-camera-intrinsic/