中華民國第 62 屆中小學科學展覽會

作品說明書

國中組　生活與應用科學(一)科

探究精神獎

032801

「深」不可「測」——紅蓮燈魚 3D 座標重建與智慧管理系統

學校名稱：宜蘭縣立國華國民中學

作者：	指導老師：
國三　楊雅盈	陳榮政
國三　簡楷真	吳浩誠

關鍵詞：智慧管理、3D 座標、3D 動態模擬

摘要

考量養魚業者未必能隨時隨地監控魚群的活動情形，因此本研究利用 YOLOv5 的視覺辨識模型及 Deep SORT 多目標追蹤演算法進行魚群的座標定位及移動追蹤，開發出一套「智慧魚缸管理系統」，我們分別使用 600、150 張魚隻照片作為辨識的訓練集與測試集，其結果準確度平均高達 99.99%，進一步利用兩鏡頭拍攝的視差以及司乃耳定律所得折射公式重建出魚群中每隻個體的 3D 座標，再根據個體座標的變換計算其活動量，並推論其可能的行為活動，同時以三維動畫模擬出魚群在魚缸內的即時狀態，此外我們選用 OpenCV 的輪廓偵測函式，計算個體的側視面積，由此觀察魚隻的成長情形，最終將上述各數據寫入 MySQL 資料庫作統計分析，當發生特定事件時，將透過 Line Notify 傳送訊息及時通知業者處理。

壹、 研究動機

我們很喜歡看魚在水裡悠游的姿態，然而卻無法時時刻刻關注著魚群的狀況，每一次養魚的經驗都很失敗，後來我們看到一篇報導，提到台灣的養漁業勞動力嚴重老化及短缺，因此我們想利用 AI 的視覺辨識技術打造一套魚類養殖的智慧管理系統，希望藉此降低養魚業者的人力需求，且同時能更迅速、精準地掌握魚群動態，諸如魚群的活動量、覓食行為，甚至死亡都能做到即時的監控與回報，使得養漁業能更有效的管理魚隻。

貳、 研究目的

本研究旨在開發一套智慧管理系統，其能**自動化的監控及記錄魚隻活動情形**，並將數據提供養魚業者作養殖方式的調整參考，而因為魚隻的活動範圍為三維空間，若欲分析魚隻的行為活動，則需先研發即時定位魚隻 3D 座標的技術，方能進一步研究魚類的各種行為辨識。

由於魚種數量繁多，故本篇研究僅以養殖紅蓮燈魚為例，模擬智慧管理系統中的各項技術操作，而實際上利用相同的技術可套用至其餘各式魚種，此即可滿足不同養魚業者的需求，以下是研究架構：

研究一、訓練出最佳魚類辨識模型

研究 1-1：訓練辨識紅蓮燈魚的模型

研究 1-2：訓練辨識多魚種的模型

研究二、建立紅蓮燈魚的 3D 座標

研究 2-1：魚類的軌跡追蹤

研究 2-2：OpenCV 3D 重建

研究 2-3：運用視差進行深度偵測

研究 2-4：紅蓮燈魚的 3D 座標

研究三、系統整合與應用

研究 3-1：3D 動畫軌跡模擬

研究 3-2：運用 3D 座標計算活動量

研究 3-3：計算紅蓮燈魚的側視面積

研究 3-4：辨識紅蓮燈魚的覓食行為

研究 3-5：辨識紅蓮燈魚是否死亡

研究 3-6：MySQL 資料庫數據儲存

參、 研究設備及器材

一、 設備器材

(一) 研究對象

表 3-1：研究對象

中文俗名：紅蓮燈	**學名：***Paracheirodon innesi*
分類地位：脊索動物門/輻鰭魚綱/脂鯉目/脂鯉科	
外型特徵： 本魚體側有一條明顯的鐵藍色條紋從吻部開始，穿過眼睛的上半部，一直延伸至脂鰭。下半身呈鮮紅色，魚體前腹部有一塊銀色；雌魚的體側較高。體長約 2.5 公分。	
中文俗名：小紅豆（紅球）	**學名：***Xiphophorus maculatus*
分類地位：脊索動物門/輻鰭魚綱/鱂形目/花鱂科	
外型特徵： 有許多具有不同體色與名字的雜交種。本研究採用紅色魚種。母球魚的體型較圓大，公魚的體型稍長些；公魚的腹鰭較尖細而長，母魚的腹鰭為整片展開像扇型，面積較大。	

(二) 飼養設備

1. 魚缸過濾器
2. 控溫加熱器(溫度：24℃)
3. LED 燈

(三) 實驗器材

1. 手機 (1289 x 592 個像素) x2
2. 手機架 x2

二、 研究軟體

表 3-2：各式應用軟體

Google Colab	Visual Studio Code	LabelImg	Tracker	LINE

三、 使用程式語言

(一) Python

肆、 研究過程與方法

一、 程式架構圖

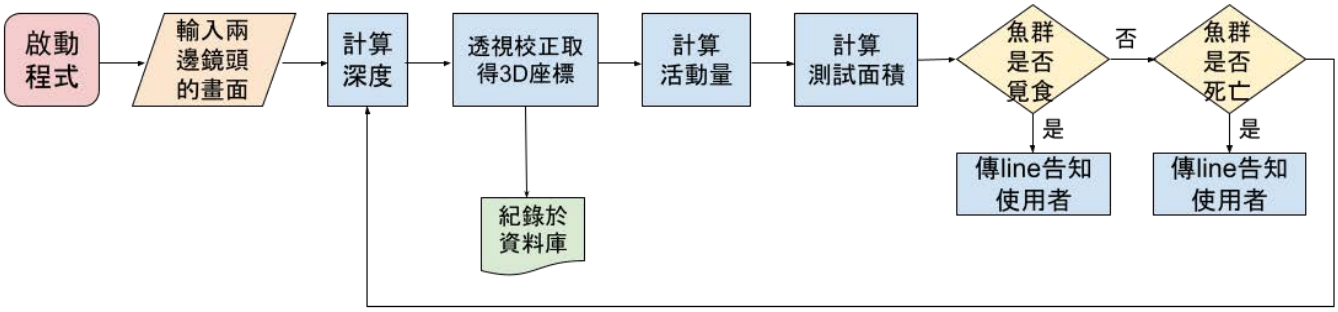


圖 4-1-1：程式架構圖

二、 視覺辨識原理

深度學習的影像辨識原理模仿人類的腦神經科學，效法神經元的多層次學習網路。人類的腦神經是由許多神經元所組成，雖然每個神經元都只能接收簡單的訊號，但神經元會將其接收到的訊號傳遞給其他神經元，而後其他神經元接連被傳遞的訊號觸發。

如圖 4-2-1 影像辨識的原理就在於模仿人類視神經，先透過對邊界的認識強化，再逐步組合出圖像識別的運作方式。套用在深度學習的影像辨識作法就會變成先將圖片分解成許多小像素，做為第一層的輸入資料，接著再經過多層次的演算法處理，從個別像素擷取特徵、組合特徵，再到最後的輸出層結果來完成影像辨識。

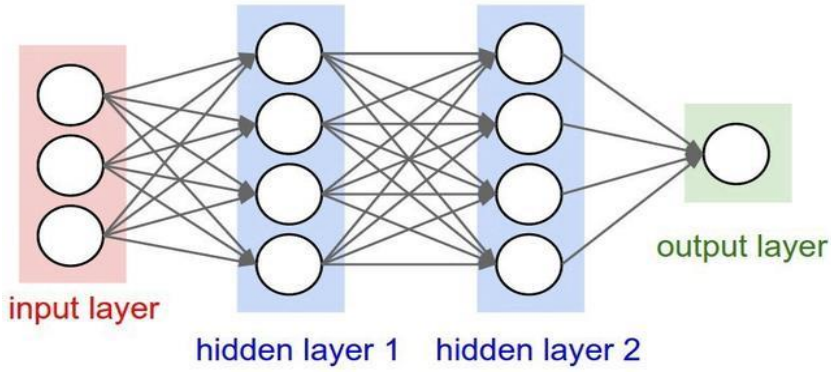


圖 4-2-1：影像辨識原理 取自網路

三、 數學及物理原理

(一) 三角函數：給定一個銳角θ，可以做出一個直角三角形，使得其中的一個內角是θ。設這個三角形中，θ的對邊、鄰邊和斜邊長度分別是a, b, h。

θ的正弦是對邊與斜邊的比值：$\sin\theta = \frac{a}{h}$

θ的正切是對邊與鄰邊的比值：$\tan\theta = \frac{a}{b}$

θ的餘切是鄰邊與對邊的比值：$\cot\theta = \frac{b}{a}$

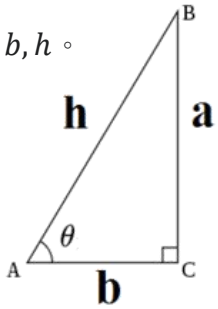


圖 4-3-1：三角函數

(二) 反三角函數

在數學中，反三角函數是三角函數的反函數，如表 4-3-1。

表 4-3-1：反三角函數

名稱	常用符號	定義
反正弦	$y = \sin^{-1} x$	$x = \sin y$
反正切	$y = \tan^{-1} x$	$x = \tan y$
反餘切	$y = \cot^{-1} x$	$x = \cot y$

(三) 司乃耳定律：當光波穿越不同介質時，若兩種介質的折射率不同，會發生折射現象，其入射光和折射光都處於同一平面，並且與界面法線的夾角滿足如下關係：

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{n_2}{n_1}$$

其中，n_1 、n_2分別是兩種介質的折射率，θ_1 、θ_2分別是入射光、折射光與界面法線的夾角，此公式稱為「司乃耳公式」。

四、 實驗步驟

【研究一：訓練出最佳魚類辨識模型】

研究 1-1：訓練辨識紅蓮燈魚的模型

我們選用 YOLOv5 目標檢測系統辨識紅蓮燈魚，然而 YOLOv5 內建的辨識模型辨識的對象不包含紅蓮燈魚，因此我們必須手動標註圖片再訓練模型，得出紅蓮燈魚辨識模型。

(一) 圖像標註

我們上網蒐集有關紅蓮燈魚姿勢的圖片並使用 LabelImg 軟體進行魚位置的標註，每隻魚的標註範圍為以魚的長度(魚頭、魚尾直線距離)及寬度(背鰭至腹鰭直線距離)為寬高的矩形。

圖 4-4-1：圖像標註範圍 取自 Pixabay 圖庫

(二) 模型訓練

使用 600 張照片作為訓練集，150 張照片作為測試集運用 Google Colab 提供的 GPU 圖形處理器進行模型檔的訓練與測試。我們分別對訓練參數 Batch-Size 及 Epoch 進行調整，找出最佳辨識模型。

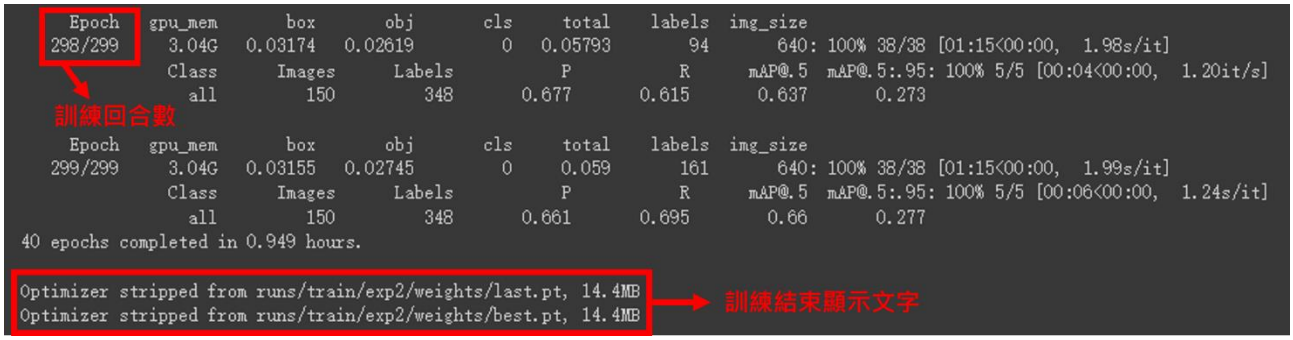


圖 4-4-2：訓練過程

(三) 訓練結果

表 4-4-1：以不同 batch-size、epoch 訓練之模型的辨識結果

Epoch \ Batch-Size	16	32	64
200	辨識率：100% 辨識錯誤：3	辨識率：100% 辨識錯誤：1	辨識率：100% 辨識錯誤：2
300	辨識率：100% 辨識錯誤：2	辨識率：100% 辨識錯誤：3	辨識率：100% 辨識錯誤：4
400	辨識率：100% 辨識錯誤：1	辨識率：100% 辨識錯誤：0	辨識率：100% 辨識錯誤：13

※ Batch-Size：每批資料量的大小。深度學習採用 SGD 訓練，每次訓練在訓練中取 Batch-Size 個樣本訓練。

※ Epoch：訓練回合數，1 個 Epoch 等於使用訓練集中的全部樣本訓練一次。

※ 辨識率 = 辨識到的魚隻數量/總魚隻數量*100%。

※ 註：辨識錯誤指將非魚之物件辨識為魚或將同隻魚辨識為兩隻魚以上。

由表 4-4-1 可得知模型辨識率皆為 100%，代表每隻魚都有被辨識到。但在 Batch-Size 為 16；Epoch 為 200、300、400 以及 batch-size 為 64；Epoch 為 200、400 的模型，有辨識錯誤的情形，並在 Batch-Size 為 32；Epoch 為 200、300 及 batch-size 為 64；Epoch 為 300 的模型，有重複辨識錯誤的情形。我們推測可能是由於 Batch-Size 過小導致過度擬合。

Batch-Size 為 32；Epoch 為 400 的模型辨識效果優於其他，錯誤最低符合能辨識出魚缸中的每隻魚。以我們拍攝的影片進行驗證後，計算出辨**識準確度平均高達 99.99%**。

研究 1-2：訓練辨識多魚種的模型

在縣賽時，評審詢問到若混養兩種或兩種以上不同的魚類，辨識模型是否有辦法區分出是哪一品種的魚。我們想到現實中養魚業者也可能混養魚隻，於是新增此項實驗。

我們選用了方便取得且易飼養的紅球魚(*Xiphophorus maculatus*)，其生性溫和可與其他魚種混養，適合做為第二種魚種進行模型訓練。

(一) 訓練過程

多魚種辨識模型的訓練流程與研究 1-1 紅蓮燈魚辨識模型的訓練過程一致，對訓練參數 Batch-Size 及 Epoch 進行調整。唯一的**差別在於標註時須對兩種魚分別進行標註**。

(二) 測試結果

圖 4-4-3 為使用前述訓練的多魚種辨識模型進行實測的結果，圖中橘色方框標示的為紅蓮燈魚，粉色方框標示的為紅球魚，可以看到此**模型辨識率 100%，能辨識出魚缸中的每隻魚並準確區分出不同魚種。**此版本也作為之後實驗使用之辨識模型。

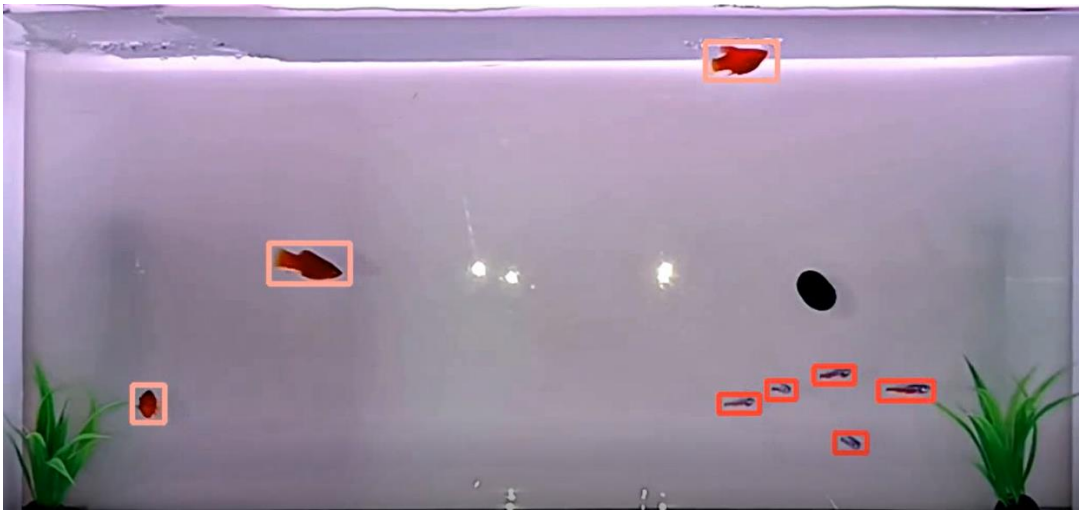


圖 4-4-3：多魚種辨識測試結果

【研究二：紅蓮燈魚的 3D 座標重建】

研究 2-1：魚類的軌跡追蹤

我們想要對魚類行為進行觀察並進行紀錄，但使用一般辨識程式僅能針對一個影格進行個別處理，對於連續性的資料，如魚的軌跡無法蒐集。

(一) 資料查詢

經由上網查詢我們發現目前有 Deep SORT 多目標追蹤算法可以對物件進行追蹤。

(二) 程式測試與改編

目前在 Github 已經有現成的程式供參考，首先我們測試此程式的效果，如圖 4-4-4，其中橘色方框標示的為紅蓮燈魚，粉色方框標示的為紅球魚，然而與一般辨識程式不同的是程式會發給每隻魚一個隨機編號(在方框的上方)，依據此編號就可以得知在程式運行過程中，此隻魚在影片中每個影格的位置，而 **Deep SORT 在辨識魚類與追蹤方面效果顯著，能辨識出魚隻並對其進行追蹤，之後有用到追蹤功能的程式都採用此演算法。**

圖 4-4-4：Deep SORT 物件追蹤測試

研究 2-2：OpenCV 3D 重建

現有的追蹤技術雖然能得到紅蓮燈魚的座標，但是由於透視問題，此座標無法代表實際魚隻所在的座標數值，因此我們想要藉由偵測魚的深度解決透視問題並建立 3D 座標，得到準確的數據，對紅蓮燈魚行為進行分析。

OpenCV 針對 3D 重建提供 StereoBM_create 函式，可以運用兩個鏡頭間的視差對紅蓮燈魚的深度進行判別，取得其 3D 座標。我們想利用其進行深度辨識，以下是測試結果。

圖 4-4-5 運用深淺不一的點代表魚隻的圖片不同位置到鏡頭距離遠近(越遠越深色)。經過測試 OpenCV 的 StereoBM_create 函式對深度辨識效果不佳，對於魚缸無法進行 3D 重建以及計算出深度，推測是由於魚缸本身為透明且相對於影像紅蓮燈魚體積過小導致。因此我們決定採用其他方式辨識魚隻在魚缸內的深度。

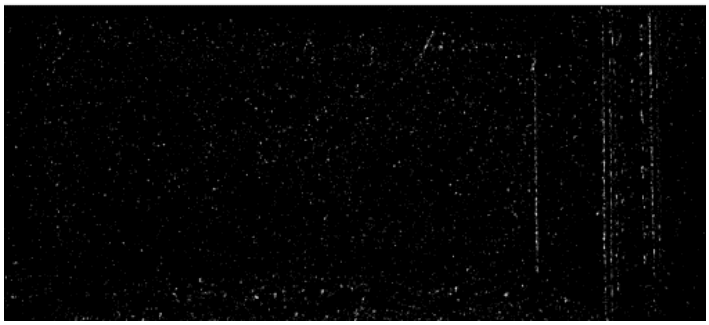


圖 4-4-5：OpenCV 深度辨識測試

研究 2-3：運用視差進行深度偵測

由於使用 OpenCV 提供的函式深度偵測效果不佳，且網路無相關程式資料，所以我們決定利用現有的辨識追蹤技術，藉由魚在兩鏡頭中圖像的座標進行深度計算。

如圖 4-4-6，**本研究使用兩個鏡頭，模擬人類的兩隻眼睛，透過兩隻眼睛所看到的畫面，人才能透過所看到的畫面判斷景物的遠近**，視覺辨識也是相同的道理，透過兩台不同相機所拍攝出來的畫面，進行比對便能得出魚所在座標之的深度。

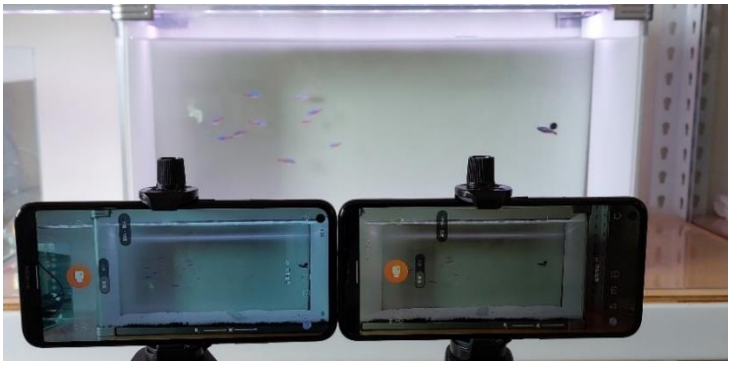


圖 4-4-6：以兩支鏡頭同時進行拍攝以計算深度

(一) 座標定義

為了方便描述，定義魚缸的**寬、高、長分別對應 x、y、z 軸**，其中 z 軸即為我們要求的深度，如圖 4-4-7 所示。

圖 4-4-7：座標定義

(二) 深度計算

如圖 4-4-8，實際物件為點P，左右兩邊的鏡頭分別為點O_L 、O_R，藍色框為拍攝圖像，物件在左右兩鏡頭拍攝之圖像的位置分別為點P_L 、P_R，兩個鏡頭的距離為b，又稱基線(baseline)，我們要求的深度即為 h。

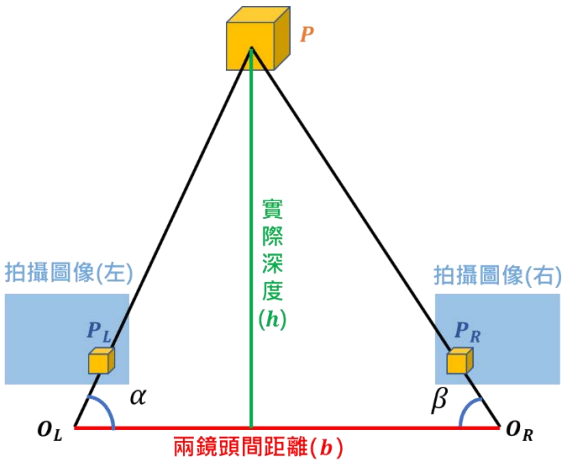


圖 4-4-8：深度計算

可得出以下算式：

$$h = \frac{b}{\cot\alpha + \cot\beta}$$

α及β的數值需要透過相機拍攝範圍角度進行計算，經過測量本研究使用之鏡頭拍攝範圍角度約為 62.8°，圖 4-4-9 是右邊相機拍攝圖像上的物件與相機中心夾角示意圖。

點O_R為右邊鏡頭，$\angle QO_RR$為鏡頭拍攝範圍角度的一半，θ為物件與鏡頭水平基線夾角，我們要求的$\beta = 90° - \theta$。

設$\overline{PR}$為$\overline{QR}$的r倍$(0 \le r \le 1)$可以得出以下算式：

$$\tan 31.4° = \frac{\overline{QR}}{\overline{RO_R}}$$

$$\tan\theta = \frac{\overline{QR}\cdot r}{\overline{RO_R}} = \tan 31.4° \cdot r$$

$$\theta = \tan^{-1}(\tan 31.4° \cdot r)$$

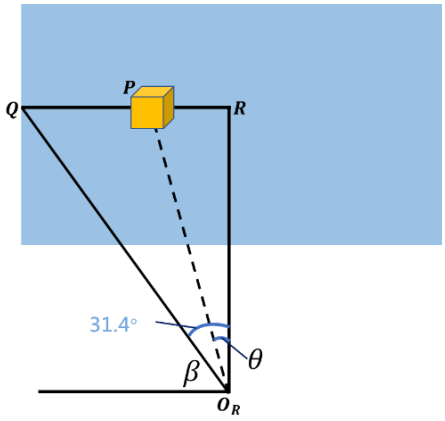


圖 4-4-9： 物件與鏡頭水平基線夾角

然而，**由於魚在水中時牽涉到水及玻璃的折射問題**，計算出的深度會與實際上有所差異，我們利用水的折射率(1.33)、玻璃的折射率(1.55)來算出折射的角度，對深度進行校正。

圖 4-4-10，為方便進行圖片說明，繪製不符合實際比例。魚位置為點P，淡藍色區塊為玻璃，厚度為$u(0.5cm)$，n_1、n_2、n_3分別為水、空氣、玻璃的折射率，實際深度為h，鏡頭至魚缸的距離為l，鏡頭間距離為b。

利用司乃耳公式可得出以下算式：

$$\alpha_2 = \sin^{-1}\left(\frac{\sin\alpha_1 \cdot n_2}{n_3}\right) = \sin^{-1}\left(\frac{\sin\alpha_1}{1.55}\right) \quad 、 \quad \beta_2 = \sin^{-1}\left(\frac{\sin\beta_1 \cdot n_2}{n_3}\right) = \sin^{-1}\left(\frac{\sin\beta_1}{1.55}\right)$$

$$\alpha_3 = \sin^{-1}\left(\frac{\sin\alpha_2 \cdot n_3}{n_1}\right) = \sin^{-1}\left(\frac{\sin\alpha_2 \cdot 1.55}{1.33}\right) \quad 、 \quad \beta_3 = \sin^{-1}\left(\frac{\sin\beta_2 \cdot n_3}{n_1}\right) = \sin^{-1}\left(\frac{\sin\beta_2 \cdot 1.55}{1.33}\right)$$

$$n = b - \tan\alpha_1 \cdot l - \tan\beta_1 \cdot l = b - (\tan\alpha_1 + \tan\beta_1) \cdot l$$

$$m = n - \tan\alpha_2 \cdot u - \tan\beta_2 \cdot u = n - (\tan\alpha_2 + \tan\beta_2) \cdot u$$

$$m = n - \tan\alpha_2 \cdot 0.5 - \tan\beta_2 \cdot 0.5 = n - (\tan\alpha_2 + \tan\beta_2) \cdot 0.5$$

利用前面的深度計算公式可得：

$$h = \frac{m}{\tan\alpha_3 + \tan\beta_3}, \quad m = n - (\tan\alpha_2 + \tan\beta_2) \cdot 0.5$$

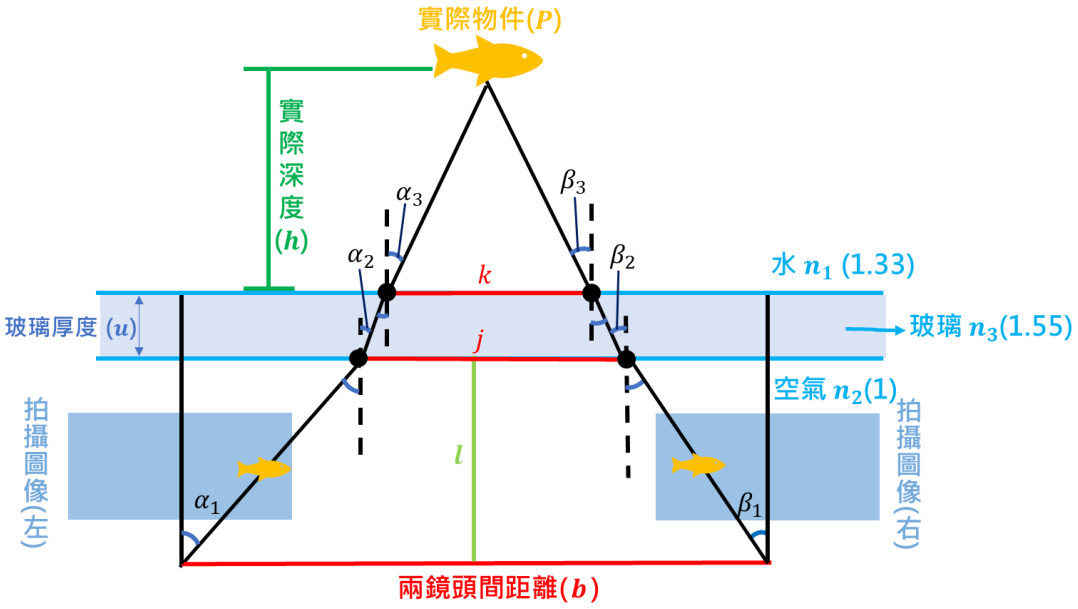


圖 4-4-10：計算玻璃折射率

利用上述公式計算出每隻魚的深度，並進行對應求出其 3D 座標。

(三) 程式流程

為了提升整體程式運行的速度，深度偵測程式採 $\frac{1}{3}$ 秒執行一次，另外，若左右兩邊鏡頭的影像都採用追蹤及辨識，也會耗費程式運行的時間，所以我們分別採用了 YOLOv5 的視覺辨識以及 Deep SORT 的追蹤功能，以左邊鏡頭影像為基準進行追蹤，右邊影像進行辨識。

為了在兩個鏡頭間對應每隻魚的座標，我們**根據每隻魚的 x 座標大小在清單中排序後，在對照每隻魚的 y 座標，將兩個鏡頭所拍攝出 x 座標大小順序相同、y 座標值相同對魚進行匹配。**

如圖 4-4-11 是深度計算程式流程：

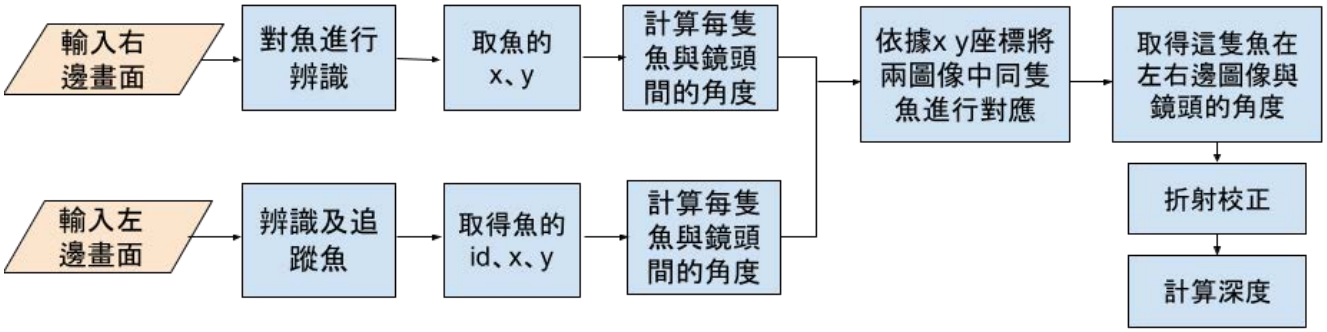


圖 4-4-11：深度計算程式流程

(四) 程式撰寫

為了匹配在左右兩邊圖像的魚隻，我們利用不同魚在兩影像中的 y 座標相同來進行同隻魚第一次的對應，並設立了誤差值。另外，**為了避免兩隻魚在同水平面的情況，以魚 x 座標與魚群最大 x 座標的差值進行第二次的判斷。**魚匹配後，以列表型態儲存其兩邊鏡頭的 x、y 座標。

接下來，使用上述公式以 Python 內建的三角函數計算其深度，在程式撰寫方面考量到**若魚的位置不在兩鏡頭之間，運算方法不同，因此若魚的夾角在另一側便視其為負值。**舉例來說若左邊鏡頭拍攝之圖像中魚的位置在左側，便將其與鏡頭水平基線之夾角設為負值，以利後續運算。以下是深度計算程式碼。

```
for i, ix in zip(fish_l_y, fish_l_x): # 匹配每隻魚的 x、y 座標
    for j, jx in zip(fish_r_y, fish_r_x):
        maxx1 = max(fish_l_x)
        maxx2 = max(fish_r_x)
        deltaxl = maxx1 - ix
        deltaxr = maxx2 - jx
        if abs(i - j) <= 10 and i != 0 and abs(deltaxl - deltaxr) <= 50:
            n = fish_l_y.index(i)
            m = fish_r_y.index(j)
            vid_writer_r.write(im0_r)
```

```
            lsx1x2y[n][0] = fish_l_x[n]
            lsx1x2y[n][1] = fish_r_x[m]
            lsx1x2y[n][2] = i
anglecamera = 31.4 # 相機角度的 1/2
water_refractive_index = 1.333  # 水的折射率
glass_refractive_index = 1.55 # 玻璃的折射率

for i in range(0,15):
    xl = lsx1x2y[i][0]
    if xl != 0:
        xr = lsx1x2y[i][1]
        y = lsx1x2y[i][2]
        l_r = (xl-640)/640
        r_r = (640-xr)/640
        ldeg1 = atan((tan(radians(anglecamera)))*l_r) # 進行角度的計算
        rdeg1 = atan((tan(radians(anglecamera)))*r_r)
        ldeg2 = asin(sin(ldeg1)/glass_refractive_index)
        rdeg2 = asin(sin(rdeg1)/glass_refractive_index)
        ldeg3 = asin(sin(ldeg2)*glass_refractive_index/water_refractive_index)
        rdeg3 = asin(sin(rdeg2)*glass_refractive_index/water_refractive_index)
        disn = b - ((tan(ldeg1)+tan(rdeg1))*distance)
        dism = disn - ((tan(ldeg2)+tan(rdeg2))*5)
        end_deep = dism/(tan(ldeg3)+tan(rdeg3)) # 計算出的深度
```

(五) 深度計算程式的驗證

為了驗證程式計算出的深度與實際的深度是否有誤差，我們**使用紅蓮燈魚的圖片，設計一隻假魚，來模擬紅蓮燈在不同深度及位置時的情況，以程式進行運算並取平均值，以計算誤差。**

假魚擺放位置：魚缸中央

表 4-4-2： 魚缸中央深度計算驗證

實際深度	5 cm	10 cm	15 cm	20 cm
程式辨識深度	5.2 cm	10.27 cm	15.9 cm	18.9 cm
誤差值	0.2 cm	0.27 cm	0.9 cm	1.1 cm

假魚擺放位置：魚缸左側

表 4-4-3： 魚缸左側深度計算驗證

實際深度	5 cm	10 cm	15 cm	20 cm
程式辨識深度	5.04 cm	13.8 cm	16.9 cm	18.38 cm
誤差值	0.04 cm	1.19 cm	1.9 cm	1.62 cm

假魚擺放位置：魚缸右側

表 4-4-4： 魚缸右側深度計算驗證

實際深度	5 cm	10 cm	15 cm	20 cm
程式辨識深度	4.17 cm	9.93 cm	13.4 cm	14.52 cm
誤差值	0.83 cm	0.07 cm	1.6 cm	5.48 cm

上表為將模擬魚擺在不同位置時使用程式辨識偵測出的深度，其中會有誤差值原因推測有 YOLOv5 辨識時位置不精確、儀器架設、鏡頭間距離過小、魚缸兩側角度偏差大，於討論一會詳細說明這些問題。

研究 2-4：紅蓮燈魚的 3D 座標

使用鏡頭拍攝物體時，會發現越後方的物體有往中央集中的趨勢，此即為透視。以本研究隻魚缸為例，假設今有魚在魚缸內其 x、z 座標不變，由 y=2 游到 y=5 的位置，其實際上移動的距離為 3，但由於透視會使測量出的距離遠小於 3。

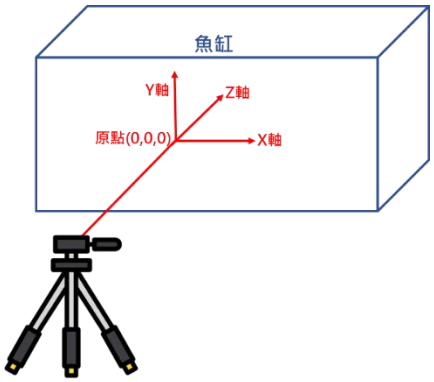


圖 4-4-12： 魚的 3D 座標示意圖

因此要得出紅蓮燈魚的 3D 座標，必須解決由於透視導致在程式中辨識出的 x、y 座標與實際不同的問題，**利用每隻魚與鏡頭水平基準線間的夾角進行運算來得出以鏡頭為中心的 3D 座標**。

(一) 公式推導

如圖 4-4-13，以 y 座標為例，魚的深度為h，鏡頭與魚缸的距離為l，由於水及玻璃的折射，我們需要分別利用θ_1、θ_2、θ_3求得其 y 座標：

$$y = \tan\theta_1 \cdot l + \tan\theta_2 \cdot u + \tan\theta_3 \cdot h = \tan\theta_1 \cdot l + \tan\theta_2 \cdot 0.5 + \tan\theta_3 \cdot h$$

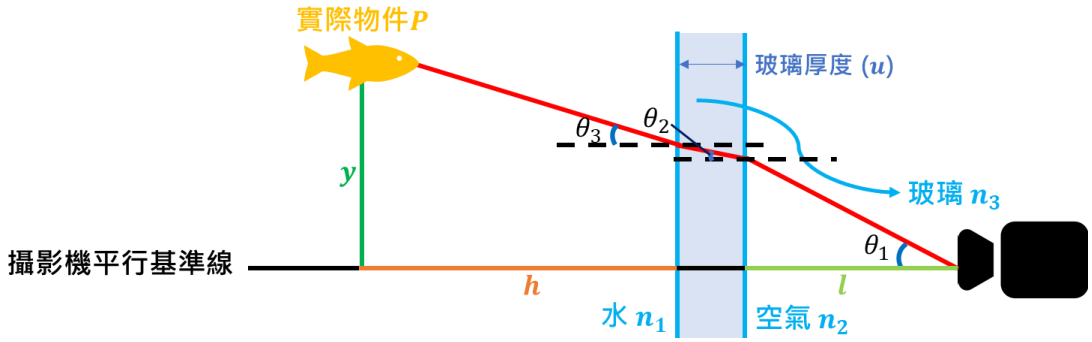


圖 4-4-13：魚的 3D 座標計算

(二) 程式撰寫

利用上述公式計算出以鏡頭水平基線為原點的 3D 座標，x 座標以基線為基準右邊為正，左邊為負，y 座標以基線為基準上方為正，下方為負，z 座標則是以魚至缸壁的距離，即為前面所述的深度。

```
from math import degrees, radians, sin, asin, tan, atan
l_r = (xl-640)/640
r_r = (640-xr)/640
y_r = (y-296)/296
# 運用三角函數計算魚與鏡頭水平基準線間的夾角
ldeg1 = atan((tan(radians(anglecamera)))*l_r)
rdeg1 = atan((tan(radians(anglecamera)))*r_r)
ydeg1 = atan((tan(radians(anglecameray)))*y_r)
ldeg2 = asin(sin(ldeg1)/glass_refractive_index)
rdeg2 = asin(sin(rdeg1)/glass_refractive_index)
ydeg2 = asin(sin(ydeg1)/glass_refractive_index)
ldeg3 = asin(sin(ldeg2)*glass_refractive_index/water_refractive_index)
rdeg3 = asin(sin(rdeg2)*glass_refractive_index/water_refractive_index)
ydeg3 = asin(sin(ydeg2)*glass_refractive_index/water_refractive_index)
disn = b - ((tan(ldeg1)+tan(rdeg1))*distance)
dism = disn - ((tan(ldeg2)+tan(rdeg2))*5)
# 計算深度
end_deep = dism/(tan(ldeg3)+tan(rdeg3))
# 計算 x 座標
coordinatex = tan(ldeg1)*distance + tan(ldeg2)*5 + tan(ldeg3)*end_deep
# 計算 y 座標
coordinatey = tan(ydeg1)*distance + tan(ydeg2)*5 + tan(ydeg3)*end_deep
# z 座標(深度)
coordinatez = end_deep
```

```
# 將資料儲存到列表
coordinatenow[i][0] = coordinatex
coordinatenow[i][1] = coordinatey
coordinatenow[i][2] = coordinatez
```

(六) 實際測試

我們實現了紅蓮燈魚的 3D 座標重建，並進行實際測試，如圖 4-4-14，可以看到藍色文字(x, y, z)分別對應紅蓮燈魚的 x、y、z 座標，單位為 cm，建立此座標能幫助我們進行後續的分析及各種行為的辨識。

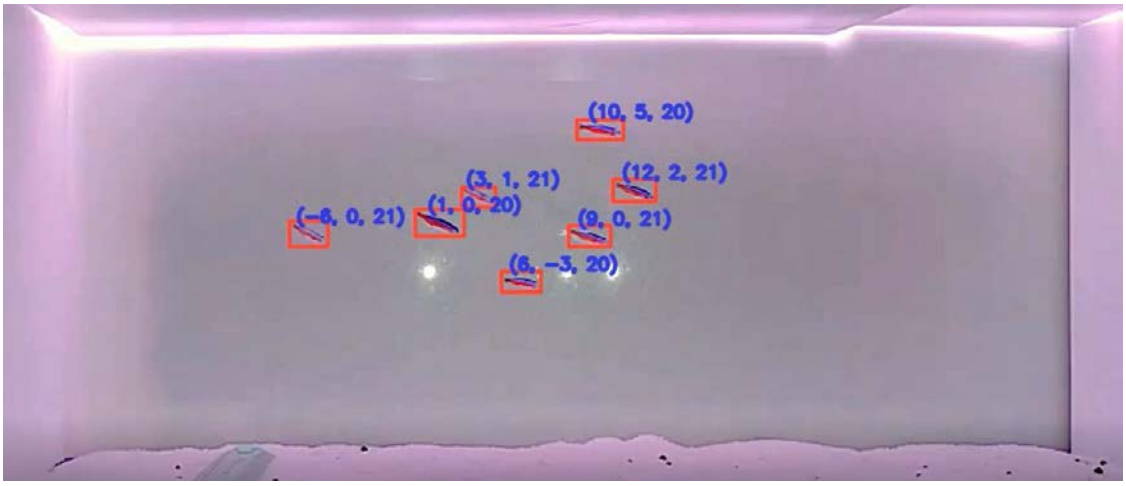


圖 4-4-14：3D 座標重建

【研究三：系統整合與應用】

研究 3-1：3D 動畫軌跡模擬

為了將魚群軌跡可視化並做紀錄，我們想要將魚群游動的路線以動畫模擬，提供飼主可以肉眼觀察的資料。

我們利用 matplotlib 提供之繪製 3D 動畫的函式，利用每隻魚在每個影格中的 3D 座標將魚群的軌跡繪製成動畫，如下頁圖 4-4-15，其中不同的顏色代表不同魚的軌跡，利用 3D 動畫將資料可視化，可以讓養魚業者清楚了解魚隻狀態或其動向。

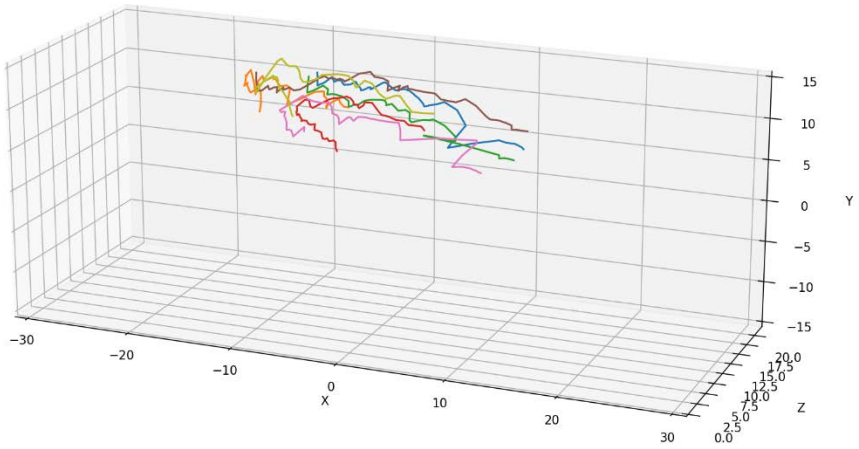


圖 4-4-15: 繪製魚群 3D 軌跡模擬動畫

研究 3-2：運用 3D 座標計算活動量

(一) 實驗觀察

為了讓使用者在管理方面能了解魚隻的狀況，針對個別魚隻進行管理，甚至對於整體的魚群健康狀況了解更詳細，我們想要利用 3D 座標，來計算魚的活動量。

(二) 程式撰寫

利用前述程式計算的 3D 座標，定位每隻魚在不同時間點的位置。**藉由追蹤功能將前後時間點的座標以畢氏定理算出個別魚隻移動的路徑，得出個別魚隻的活動量，並將數個距離相加計算出總路徑長，**以路徑長的長短來判斷魚群在相同時間段的活動量，對紅蓮燈魚的健康狀況進行管理。

```
for i in range(0,15)：
    fishx1 = lsx1x2ybefore[i][0] # 魚上次的 x 座標
    fishy1 = lsx1x2ybefore[i][1] # 魚上次的 y 座標
    fishz1 = lsx1x2ybefore[i][2] # 魚上次的 z 座標
    fishx2 = lsx1x2ynow[i][0] # 此次的 x 座標
    fishy2 = lsx1x2ynow[i][1] # 此次的 y 座標
    fishz2 = lsx1x2ynow[i][2] # 此次的 z 座標
    # 以畢氏定理進行計算個別魚的活動量
    shf=((((fishx1-fishx2)**2+(fishy1-fishy2)**2)**0.5)**2+(fishz1-fishz2)**2)**0.5
    shift_value += shf # 魚的位移累加
text = "shift：" + str(shift_value)
color = (0, 255, 255)
cv2.putText(im0_l, text, (10, 10),cv2.FONT_HERSHEY_SIMPLEX, 1, color,1,cv2.LINE_AA)
```

(三) 實際測試

如圖 4-4-16，左上角的藍色文字顯示的是魚群整體的活動量(單位為 cm)，此數值可幫助養魚業者在飼養期間了解魚群的狀況與其習性。

圖 4-4-16：活動量計算測試

研究 3-3：計算紅球魚的側視面積

(一) 實驗觀察

為了讓養魚業者能清楚了解魚隻成長情形，我們設計了此實驗。一般在描述魚的成長情形與進行大小衡量時，大多是利用魚的長度進行描述，然而對於在相同體長下體型差異大的魚種，此描述方法不夠精確，因此我們決定用另一種描述方式——魚的側視面積。

(二) 側視面積計算流程

一開始採取抓取魚隻的輪廓並計算輪廓圍成封閉區域的面積之作法，然而在拍攝過程中魚的姿勢不一定能完全側身對著攝影鏡頭，於是改採計一段時間內個別魚隻在畫面中面積的最大值作為其側視面積，以避免在計算面積時魚沒有完全側身對著鏡頭的問題。

我們利用 OpenCV 的 findContours 輪廓偵測函式得出其輪廓，圖 4-4-17 為紅球魚的輪廓偵測結果，再利用 OpenCV 的 contourArea 函式即可得出魚的面積，然而計算出的面積單位為像素，為方便比較決定換算為平方公分。我們利用深度偵測方法進行換算，以下是換算方式。

圖 4-4-17：紅球魚輪廓偵測

如圖，點 A、點 B 分別為魚的左側點及右側點，首先，利用前述的 3D 座標計算方式求出此兩點的實際x座標$\boldsymbol{x_i}$、$\boldsymbol{x_j}$，**利用此兩座標的差值與 A、B 兩點間像素差的比值，即可換算出實際面積。**

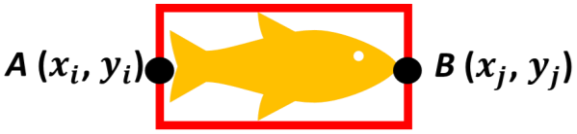


圖 4-4-18：魚的左側點及右側點示意圖

$$r = \frac{x_j - x_i}{A、B兩點像素差}$$

$$實際面積 = 以像素為單位的面積 \cdot r^2$$

(三) 程式撰寫

首先，我們利用前述 3D 座標計算方式求出魚兩側的 x 座標，以求出左側點及右側點 x 座標的差值與兩點之間的像素數量的比值。接著擷取此魚隻標註範圍再往外延伸 10 像素的矩形影像，將此影像二值化並對其進行 not 運算，最後對此擷取影像進行輪廓偵側，然而偵測時可能會偵測到不只單隻魚的輪廓，也會偵測到其餘物件或魚隻的輪廓，因此我們以所有偵測到的輪廓圍出之封閉圖形面積的最大值為此魚隻的輪廓。

得出輪廓後，利用 contourArea 函式計算出輪廓圍成的區域之面積，最後利用前述比值計算出魚的實際側視面積。

```
# 求出左側點及右側點 x 座標的差值與兩點之間的像素數量的比值
ratio = (width_l_x - width_r_x)/(redball_l_weightheight[i][0]*2)
# 擷取魚隻影像
crop_img = im0_l[int(redballlsx1x2y[i][2]-redball_l_weightheight[i][1]-
5):int(redballlsx1x2y[i][2]+redball_l_weightheight[i][1]+5),
int(redballlsx1x2y[i][0]-redball_l_weightheight[i][0]-
5):int(redballlsx1x2y[i][0]+redball_l_weightheight[i][0]+5)]
crop_img = np.uint8(crop_img)
gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY) # 轉灰階
ret,binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY) # 二值化
binary = cv2.bitwise_not(binary) # not 操作
# 取出輪廓
contours, hierarchy=cv2.findContours(binary, cv2.RETR_CCOMP, cv2.CHAIN_APPROX_NONE)
area = []
# 找出面積最大者
for i in range(len(contours)):
    a = cv2.contourArea(contours[i])
    area.append(abs(a))
max_area = max(area)
actual_area = max_area*(ratio**2) # 計算出側視面積
```

研究 3-4：辨識紅蓮燈魚的覓食行為

(一) 實驗觀察

若飼主在投餵飼料後，在短時間沒有被吃完，會造成水質髒亂甚至導致過濾系統崩解或藻類過度孳生，因此觀察魚是否有覓食行為可以幫助飼主更了解魚的狀況。

因為飼料浮在水上，且紅蓮燈魚屬於生活在中下層的魚種，其在覓食時會往上游進行覓食，因此只要指定時間**判斷魚的 y 座標與水面距離是否小於 5 cm**，即可知道其覓食狀況。

(二) 程式撰寫

利用追蹤功能，我們可以得出同隻魚的路徑，來辨識其是否有覓食行為。另外使用python 提供的 datetime 函式來判斷時間是否屬於餵食時段。

```
nowtime = datetime.datetime.today() # 取出現在時間
settime = "17：41" # 設定餵食時間(時:分)
loc_dt_format = nowtime.strftime("%H：%M") # 將現在時間格式化
if loc_dt_format == settime： # 如果現在時間為設定的餵食時間
    if coordinatey > 6： # 如果魚的 y 座標與水面距離小於 5 cm
       feedtime += 1
```

(三) 實際測試

我們討論出辨識魚隻是否在覓食的機制，圖 4-4-19 中顯示紅蓮燈魚的覓食情形，得知此數據可以維護水質且反映魚隻的健康。

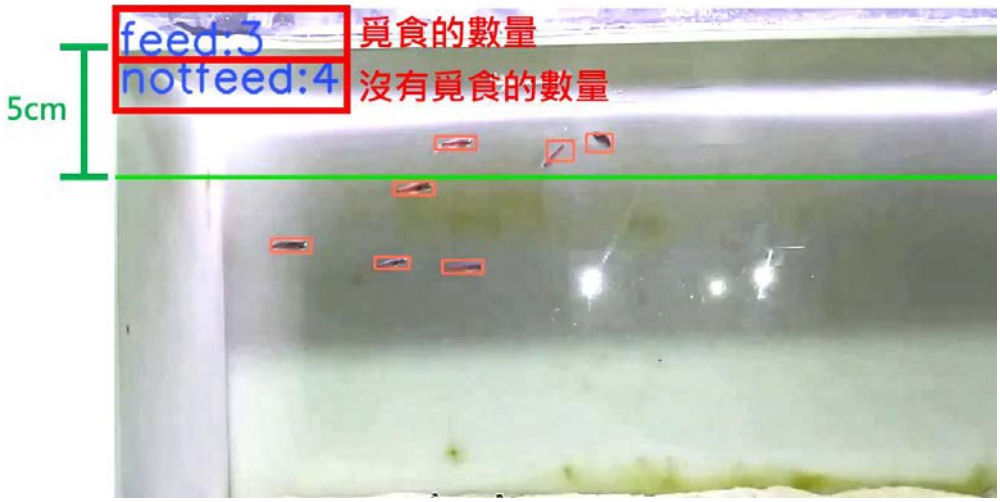


圖 4-4-19：覓食偵測示意圖

研究 3-5：辨識紅蓮燈魚是否死亡

(一) 實驗觀察

我們觀察到魚類在死亡時的活動量趨近為零，並且死亡以後，會因失去浮力而沉落水中，必須在**魚隻死亡後盡快進行處理，否則會汙染水質或影響整體魚群**。

(二) 程式撰寫

利用前述的活動量計算程式判斷魚是否死亡，以活動量小於 5cm 且 y 座標小於 -13cm (魚沉在水底)代表魚死亡。這裡為了方便測試，這裡的程式以**持續 5 秒活動量小於 5cm 判定這隻魚為死亡**，參數可調整。

```
shiftcount = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] # 判定魚死亡計時列表(1 代表 1/3 秒)
if shift < 5 and coordinatey < -13 and shiftcount[i+1] != None：
    shiftcount[i+1] += 1 # 將死亡計時列表加 1
    if shiftcount[i+1] >= 15： # 記數大於一定值，也就是魚持續處於死亡狀態，則判定此魚死亡
        dietime += 1 # 死亡隻數加 1
        shiftcount[i+1] = None
    else：
        shiftcount[i+1] = 0
```

(三) 運用 LINE 傳送通知

魚類死亡之後，若放置太久易對水質造成影響導致後續處理上的困難，因此我們想利用 LINE Notify 的主動推播功能，在紅蓮燈魚死亡後能即時告知飼主魚缸的狀況。

```
import requests
import cv2
def lineNotifyMessage(token, msg, img=None)： # 傳送信息的函式
    headers = {"Authorization"： "Bearer " + token}
    payload = {'message'： msg}
    files = {'imageFile'： open(img, 'rb')} if img else None
    r = requests.post(
        "https：//notify-api.line.me/api/notify", headers=headers, params=payload,
files=files)
    if files：
        files['imageFile'].close()
    return r.status_code
token = "此處放置自己的金鑰" # 金鑰
```

```
cv2.imwrite("img/img.jpg", im0_1) # 傳送的圖片
img = "img/img.jpg"
message = "有" + str(dietime) + "隻魚死亡" # 傳送的信息
lineNotifyMessage(token, message, img)
```

(四) 實際測試

如圖 4-4-20，LINE Notify 是 LINE 提供的一項服務，可以利用自動推播功能及時告訴養魚業者，魚隻死亡情況，並傳送截圖及文字，對於魚缸的情形也可更加了解。

圖 4-4-20： LINE Notify 自動推播功能

研究 3-6：MySQL 資料庫數據儲存

為了讓使用者方便處理魚群的各項資料，並且日後能夠快速地對資料進行操作，如修改、刪除，我們使用了 MySQL 資料庫儲存魚類各項數據。

(一) 儲存在 MySQL 資料庫數據的數據

我們選用了魚類的 x、y、z 座標作為資料庫儲存的內容，並且每隻魚的數據以程式分別儲存在不同的資料表。

(二) 程式撰寫

```
db_settings = {"這裡是一些連接資料庫所需的資料"}
conn = pymysql.connect(**db_settings) # 連接資料庫
with conn.cursor() as cursor:
    tablename = "table"+ str(i+1) # 設定要儲存的資料表
    # 插入資料指令
    command = f"INSERT INTO {tablename}(videotime, x, y, z)VALUES(%s, %s, %s, %s)"
    # 要插入的內容如時間、x、y、z 座標
    cursor.execute(command, (round(frame_idx/30, 2), coordinatenow[i][0],
coordinatenow[i][1], coordinatenow[i][2]))
    # 執行指令
    conn.commit()
```

(三) 實際測試

圖 4-4-21 為編號 3 魚的數據儲存，可以看到資料表中紀錄了魚在不同時間下的 x、y、z 座標，這些數據可以幫助使用者了解魚群狀況並對軌跡進行分析。

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

videotime	x	y	z
13	2.1	-2.1	0.4
13.1	2.4	-2.8	28.3
13.2	1.8	-2.8	26.3
13.3	1.6	-2.7	25.1
13.4	0.9	-2.1	-2.4
14	2.5	-2.7	4.2
14.1	1.5	-2.6	2.8
14.2	0.2	-2.6	3.5
14.3	-1.2	-2.6	4.8
14.4	-2.2	-2.6	5
14.5	-3.2	-2.7	6.5
14.6	-8.3	-5.5	106.2
14.7	-4.7	-2.7	7.6
14.8	-5.1	-2.7	6
14.9	-5.7	-2.7	7.4
15	-6.1	-2.8	8
15.1	-6.4	-2.8	8

table3 1 ×

圖 4-4-21： MySQL 資料庫數據儲存

伍、 討論

討論一：深度偵測誤差值

在研究 2-3 中計算深度時會有誤差，並隨著魚往魚缸兩側移動，誤差值會增大，推測可能是由於下列原因：

(一) YOLOv5 辨識時位置不精確：YOLOv5 辨識時，在標註魚隻時會有些誤差，而這些細微的誤差會造成在偵測深度時的誤差，未來我們想開發另一種辨識技術，標註魚的位置時能更準確，增加深度偵測的準確率。

(二) 儀器架設：鏡頭拍攝方向與魚缸沒有完全垂直。

(三) 鏡頭間距離過小：實驗測試時鏡頭間的距離只有 16 cm，導致若偵測魚的位置時出現稍微偏差，就會影響深度偵測結果。

(四) 魚缸兩側角度偏差大：在偵測深度時，魚缸兩側位置的魚到鏡頭中心的角度較大，導致在計算深度時的誤差會較魚缸中央的魚誤差更大。

討論二：活動量偵測之準確性

活動量測量以每一影格魚隻 3D 座標，用畢氏定理算出個別魚隻移動的路徑，因此根據所採取之時間間隔不同，偵測之數據也會有所不同。假如採取之時間間隔越小，時間點越密集，則準確性提高，因為更貼近動態魚群的移動；反之採取時間點間段越大，準確性則越低，無法捕捉到時間點之間之魚群動向。然後時間點抓取越密集、需要運算之座標數量增多，則硬體運算量便會增大。本研究為取得準確度與硬體設備能負荷的運算量範圍兩者間的平衡，以每 $\frac{1}{3}$ 秒魚群的座標進行運算。

討論三：本技術與傳統 3D 重建方式進行比較

表 5-2-1 可得知，本研究技術相較於傳統方法有速度快、深度偵測準確率高的優勢，並建立 3D 座標增加了追蹤、折射校正功能，最後將數據傳輸到資料庫。

表 5-2-1：與傳統 3D 重建方式進行比較

	Tracker3D	OpenCV 3D 重建	使用本研究技術 進行深度偵測
方式 簡述	手動標註或追蹤物件的位置來計算軌跡	利用兩個鏡頭所拍攝出的畫面進行比對計算深度	運用兩個鏡頭間的視差對魚類的深度進行判別，取得其 3D 座標
深度偵測 準確率	佳	極差	佳
速度	以手動標註時間為準	3 秒/幀	5 秒/幀 (推測是硬體效能不夠致)
建立 3D 座標	✓		✓
折射 校正			✓
追蹤 功能			✓
優點	能確保每隻魚的匹配	不需計算參數、鏡頭角度等數據	運用數學計算，即時辨識與追蹤魚的位置
缺點	過於花費時間	準確度不足	需要先訓練辨識模型

陸、 結論與未來展望

一、 結論

本篇研究利用 AI 相關技術開發出一套完整的智慧管理系統，其技術及功能如下：

(一) 研發技術

1. **路徑追蹤**：利用 YOLOv5 及 Deep SORT 辨識及追蹤魚隻。
2. **深度偵測**：利用兩鏡頭視差推導出深度計算公式 $h = \frac{b}{\cot\alpha + \cot\beta}$。
3. **折射校正**：由於光線通過不同介質時的折射率不相同，故透過司乃耳定律推導出深度座標的校正公式 $h = \frac{x}{\cot(90°-A_1)+\cot(90°-B_1)}$，其中 $x = \frac{b(h'-l)}{h'}$。
4. **建立 3D 座標**：整合前述 1 至 3 點重建出魚隻個別的 3D 座標並利用 matplotlib 的函式將魚群的移動軌跡投射至虛擬的空間座標圖上。

(二) 功能應用

1. **活動量測量**：由 3D 座標計算魚隻移動的路徑長作為活動量數據。
2. **成長情形紀錄**：透過計算魚隻側視面積作為成長情形的量化數據。
3. **覓食行為辨識**：設定魚隻游動的高度基準以辨識其覓食行為次數。
4. **死亡辨識及 LINE 自動推播**：運用 LINE Notify 及時回報魚隻死亡情形。
5. **資料庫**：將上述所有數據作自動化紀錄，便於後續作調用分析。

本研究有別於一般在單一介質中進行型物件辨識與 3D 座標定位之技術，更以拍攝魚缸會橫跨空氣、玻璃、水三種介質之情境，研發出在多重介質中進行即時物件辨識與物件追蹤之技術，並進行 3D 座標定位及資料儲存與分析。

我們的系統除了自動、速度快、準確率高，更有資料儲存、LINE 自動推播的優勢，改善傳統手動標註耗時費力的缺點，並結合辨識與追蹤功能，建立軌跡對魚的情況有效管理。除了提供養魚業者使用，亦可**作為生物研究工具或其他三維數據處理，應用範圍廣。**

二、 未來展望

(一) 希望未來能新增養殖魚場實測，實現系統能在實際場域被運用。

(二) 本研究開發之系統速度雖然相對於以往已經大幅提升，但對於及時的影像處理仍稍顯不足，因此未來我們想要調整程式邏輯或使用多執行緒等方法進行改善。

(三) 目前此技術還在測試階段，之後可設計圖形使用者介面，方便使用者進行操作。

柒、 參考資料

一、Mikel Broström (2020). *Real-time multi-object tracker using YOLOv5 and deep sort.* GitHub. https://github.com/mikel-brostrom/YOLODeepv5_Deep SORT_Pytorch

二、Doxygen (2022, March 11). *Depth Map from Stereo Images.*OpenCV. https://docs.opencv.org/3.4/dd/d53/tutorial_py_depthmap.html

【評語】032801

1. 本作品利用 AI 影像辨識，判定魚的種類、座標位置、深度，並追蹤其活動軌跡，判定其活動力、進食狀況、成長狀況、或死亡，並適時發出簡訊通知，可套用在養殖魚場的水產監控，具有實際應用價值。

2. 多種魚種的辨識只有兩種，所以正確率為 100%，但若魚種增多或魚種間相似都會影響辨識率。

3. 在重建 3D 模型中，用兩個攝影機來建構誤差大，為何不在垂直方向再架設一攝影機就可以簡化問題，減少深度估算誤差？上網蒐集有關紅蓮燈魚姿勢的圖片並使用 Labeling 軟體進行魚位置的標註，為何不使用真實自己拍攝的照片？

4. 動態追蹤魚的活動的軌跡如何評估精準度？追蹤時還要避免魚彼此重疊的問題所產生的誤判。

5. 判定只要在指定時間內判斷魚的 y 座標與水面距離是否小於 5 cm，即可知道其覓食狀況。程式以持續 5 秒活動量小於 5cm 判定這隻魚為死亡。五公分是如何定義出的？魚類死亡的數值 5 秒鐘與活動量小於 5 cm 有沒有統計佐證，正確率與 false alarm 會多少？

作品簡報

「深」不可「測」

—紅蓮燈魚3D座標重建與智慧管理系統

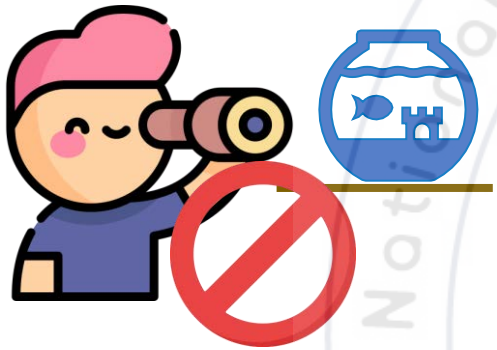
科　別：生活與應用科學科(一)

組　別：國中組

研究動機

個人經驗

無法時刻關注魚群狀況



社會議題

養殖漁業人口短缺

昔日養殖王國　逐漸衰退

2009.12.23 03:36

地區: 臺灣
分類: 社會關懷
標籤: 中正e報 東北亞

地與人的關係，導正漁民追求產量而非質量的
度利用的概念。而漁村人力的老化以及外流，
人力和技術無法進入舊漁村和高齡漁民，也是

智慧魚類管理系統

研究目的

原因

- 魚群健康狀況
- 避免水質髒亂
- 即時死亡通知
- 成長情形追蹤

功能

- 活動量計算
- 覓食行為辨識
- 死亡狀態辨識
- 側視面積測量

技術

- 追蹤魚隻動向、軌跡
- 研發3D座標建立技術

訓練辨識紅蓮燈魚的模型

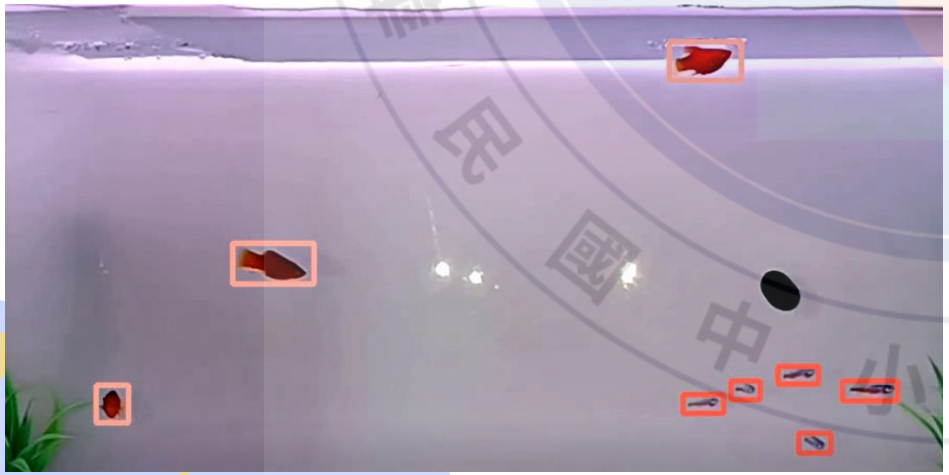
Epoch \ Batch-Size		16	32	64
200	辨識率	100%	100%	100%
	誤判(個)	3	1	2
300	辨識率	100%	100%	100%
	誤判(個)	2	3	4
400	辨識率	100%	100%	100%
	誤判(個)	1	0	13

※ 辨識率 $= \frac{\text{辨識到的魚隻數量}}{\text{總魚隻數量}} \times 100\%$。

***Batch − Size*為32；*Epoch*為400的模型效果優於其他**，能辨識出魚缸中的每隻魚。以我們拍攝的影片 進行驗證，計算出**準確度平均高達99.99%**。

準確率高達
99.99%

訓練辨識多魚種的模型



圖中紅色方框標示的為紅蓮燈魚，粉色為紅球魚，此**模型辨識率100%，能辨識出魚缸中的每隻魚並準確區分出不同魚種。**

追蹤魚類軌跡

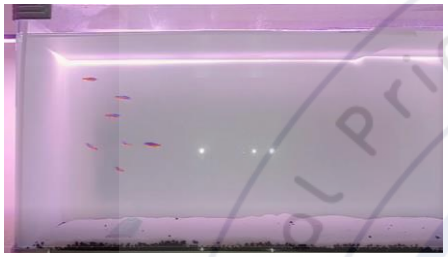
Yolov5 物件偵測：針對影片中各影格依序進行辨識。

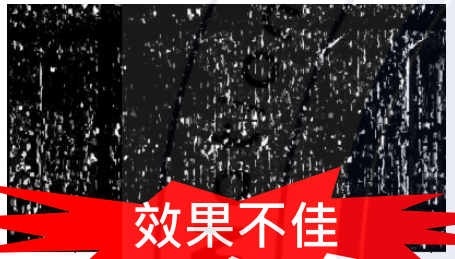
Yolov5 物件偵測 + Deep SORT多目標追蹤算法：將前後兩幀影格辨識出的魚進行比對，得出每隻魚的動向與軌跡。

實際測試

建立紅蓮燈魚的3D座標

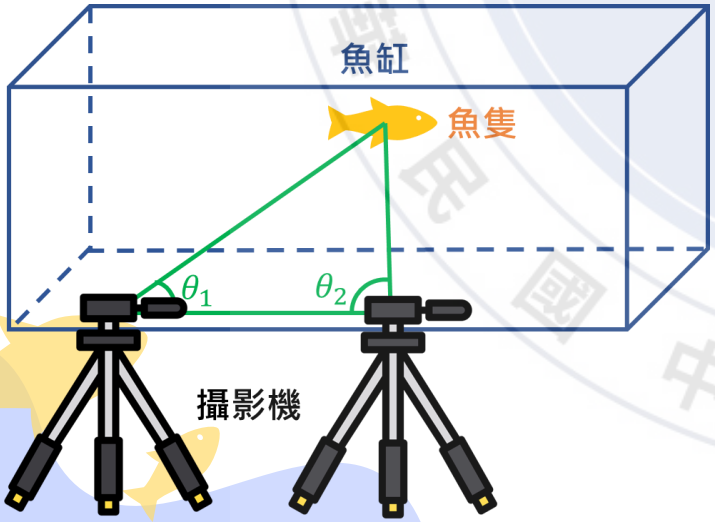
OpenCV 3D重建



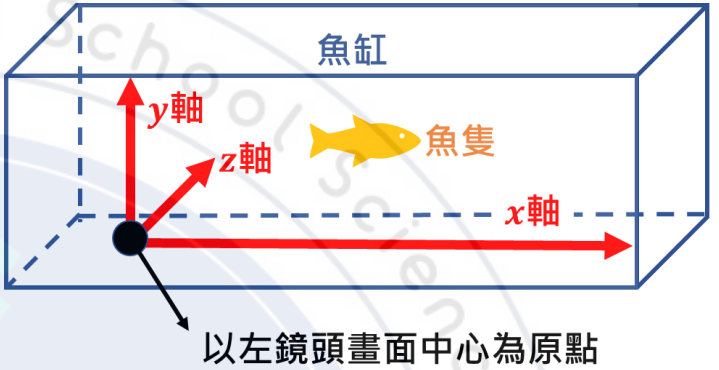


效果不佳

以雙鏡頭拍攝進行深度偵測

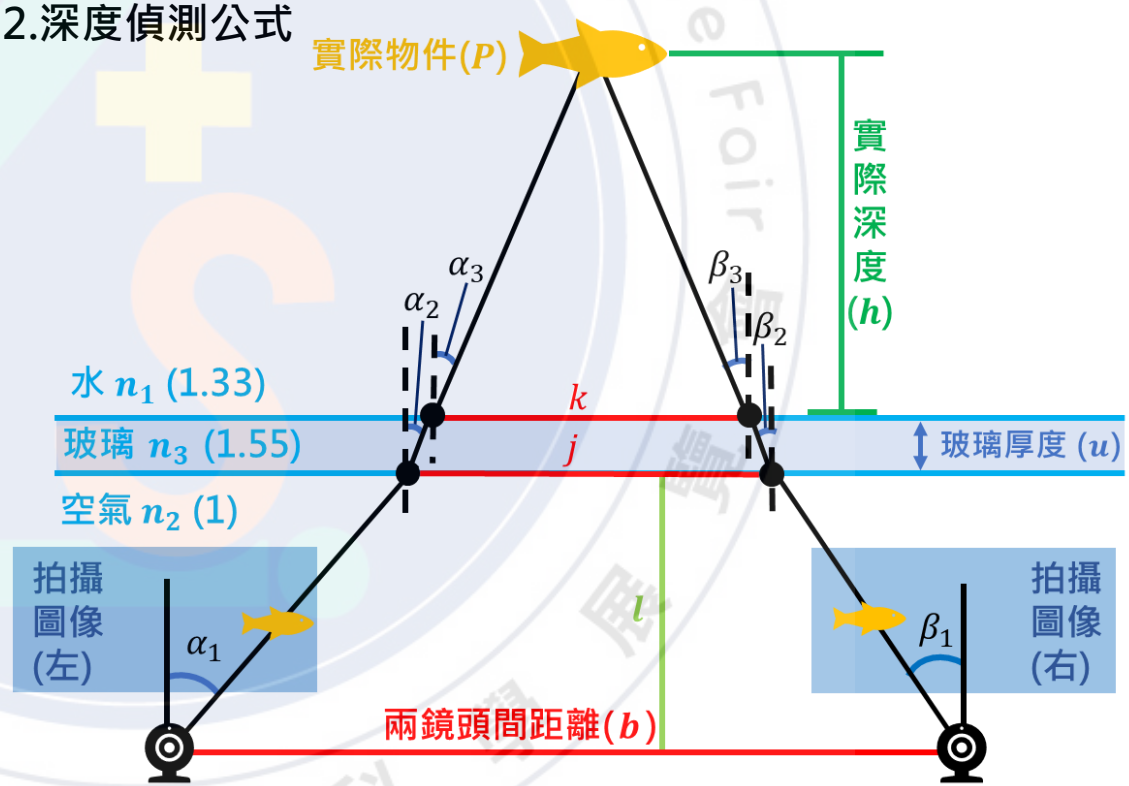


1. 自定義3D座標系統



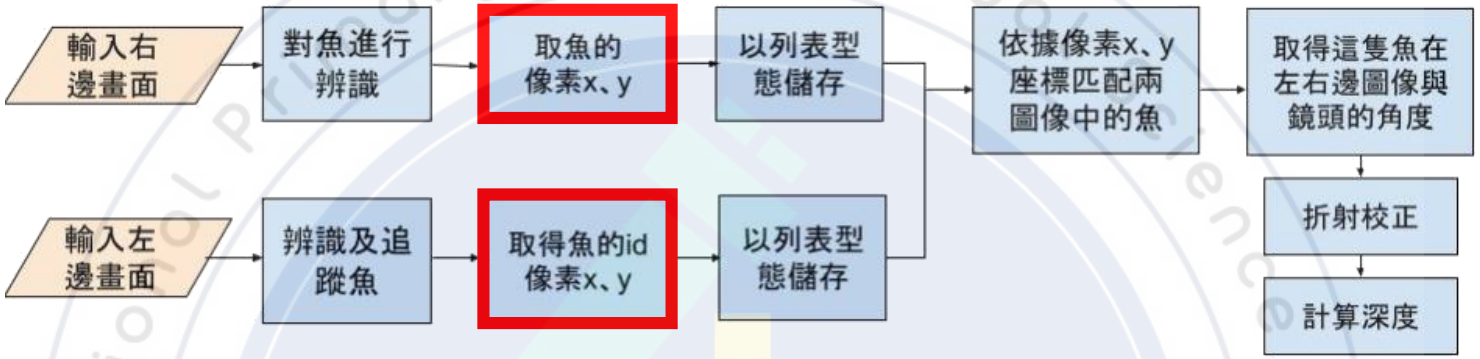
以左鏡頭畫面中心為原點

2.深度偵測公式



$$h = \frac{b - (\tan\alpha_1 + \tan\beta_1)\cdot l - (\tan\alpha_2 + \tan\beta_2)\cdot u}{\tan\alpha_3 + \tan\beta_3}$$

深度偵測程式流程圖



深度偵測驗證

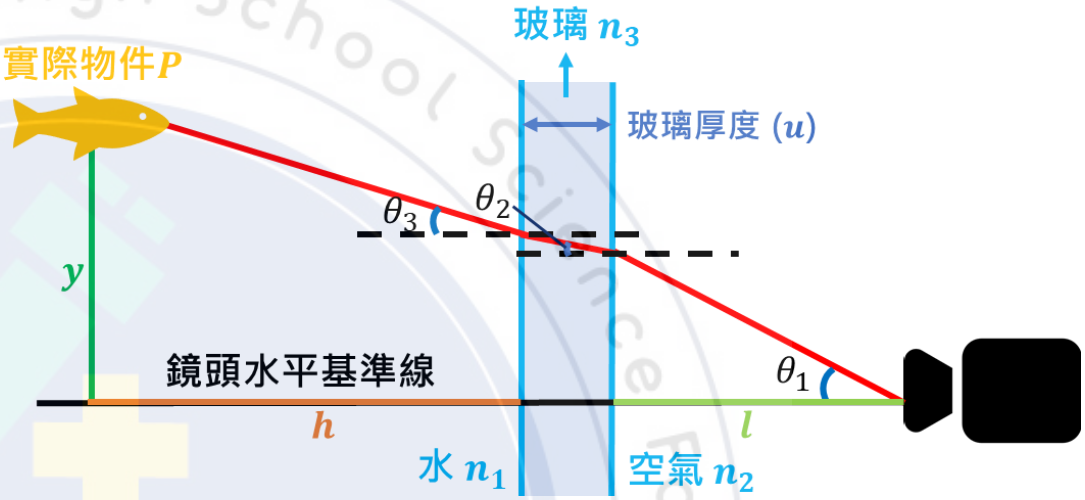
單位：公分

模擬魚擺放位置 \ 實際深度		5	10	15	20
魚缸中央	程式辨識深度	5.2	10.27	15.9	18.9
	誤差值	0.2	0.27	0.9	1.1
魚缸左側	程式辨識深度	5.04	8.71	16.9	18.38
	誤差值	0.04	1.19	1.9	1.62
魚缸右側	程式辨識深度	4.17	9.93	13.4	14.52
	誤差值	0.83	0.07	1.6	5.48

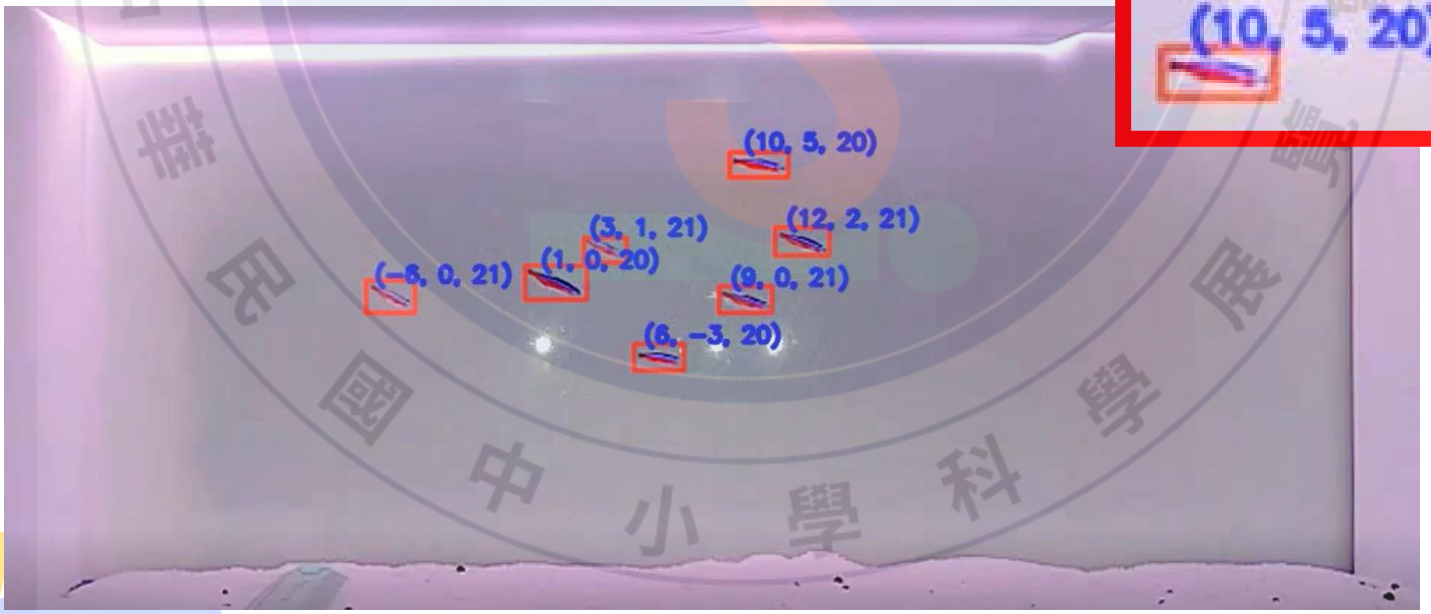
魚的3D座標

- **z座標**：即為魚的深度
- **x座標**：利用其像素x座標與z座標進行換算
- **y座標**：利用其像素y座標與z座標進行換算

$$y = \tan\theta_1 \cdot l + \tan\theta_2 \cdot u + \tan\theta_3 \cdot h$$



實際測試



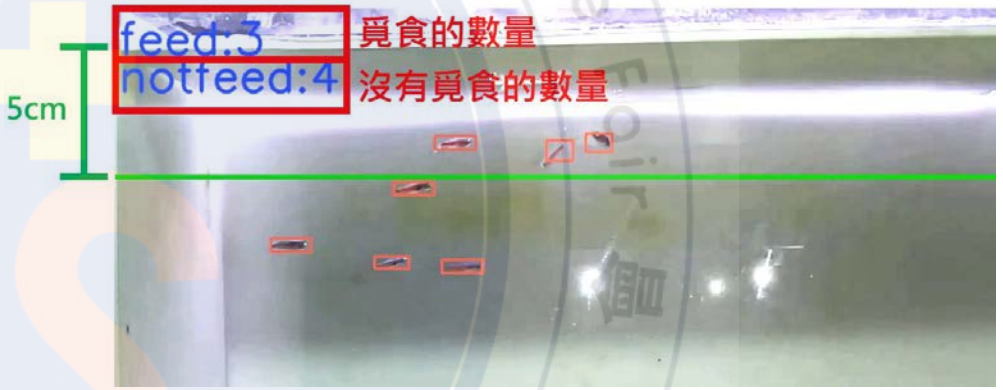
(x, y, z)

1. 計算紅蓮燈魚的活動量

設定時間間隔抓取每隻魚隻的$3D$座標，利用畢氏定理算出個別魚隻移動的路徑長，並相加求出魚群總活動量。

2. 辨識紅蓮燈魚的覓食行為

設定投餵飼料的時間，判斷此時魚的y座標與水面距離是否小於5 cm。



3. 判斷紅蓮燈魚是否死亡

持續一段時間魚的活動量小於$5cm$且y座標小於 $-13cm$ (魚沉在水底)，代表魚死亡。

利用$LINE\ Notify$的主動推播功能，即時告知養魚者魚缸情況。

4.計算魚的側視面積

Step1：抓取魚的輪廓

Step2：取一段時間內輪廓圍成的區塊中的像素數最大值

Step3：得出此隻魚的側視像素數

Step4：算出實際面積

計算魚左側點及右側點的3D座標(x_i、x_j)

※ x_i、x_j的單位為公分

計算x_i、x_j差值與A、B兩點像素差的比值，利用此比值算出實際面積，單位為平方公分。

$$r = \frac{x_j - x_i}{A、B\text{兩點像素差}}$$

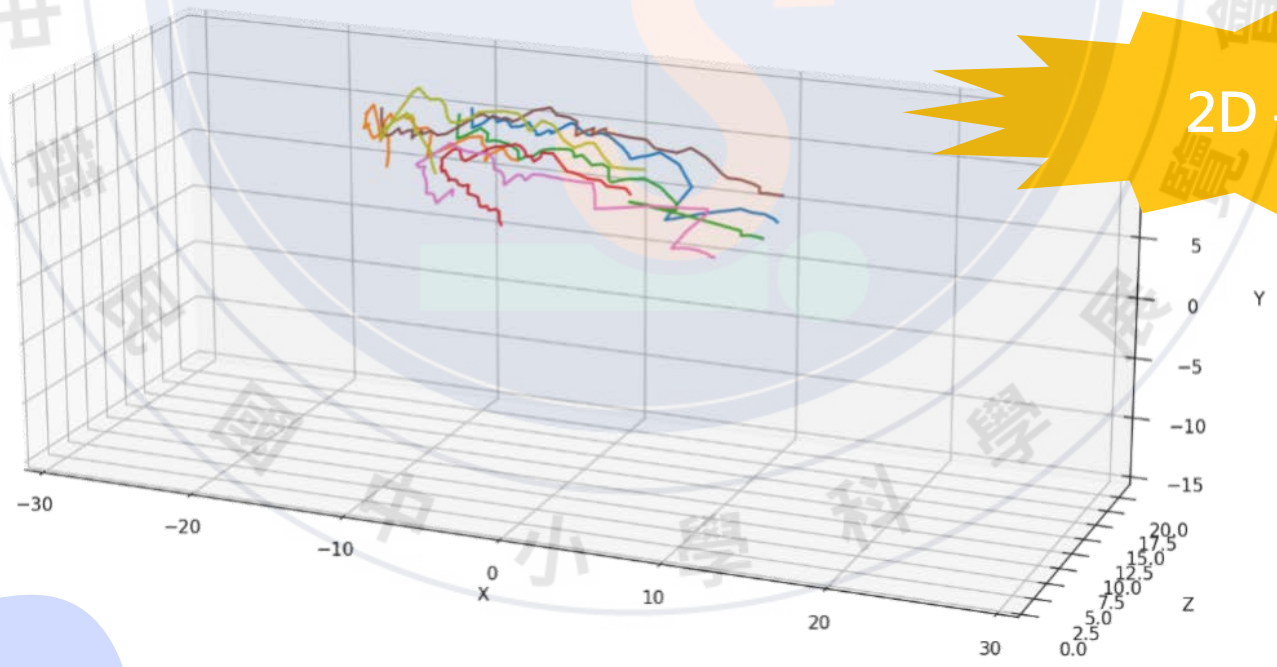
實際面積 = 魚的側視像素數 $\cdot r^2$

5.MySQL資料庫

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

videotime	x	y	z
13	2.1	-2.1	0.4
13.1	2.4	-2.8	28.3
13.2	1.8	-2.8	26.3
13.3	1.6	-2.7	25.1
13.4	0.9	-2.1	-2.4
14	2.5	-2.7	4.2
14.1	1.5	-2.6	2.8
14.2	0.2	-2.6	3.5

6. 3D動態軌跡模擬 ── 結合Deep SORT與魚類3D座標



討論

討論一：深度偵測誤差值

實驗3中計算深度會有誤差，推測是由於下列原因：

YOLOv5辨識時位置不精確

儀器架設

鏡頭間距離過小

討論二：活動量偵測之準確性

活動量測量以每一影格魚隻3D座標，用畢氏定理算出個別魚隻移動的路徑。

程式抓取時間間隔越小

準確度越高

運算量越大

本研究為取得準確度與硬體設備能負荷的運算量範圍兩者間的平衡，以每 $\frac{1}{3}$ 秒魚群的座標進行運算。

討論三：本技術與傳統3D重建方式進行比較

	Tracker3D	OpenCV 3D重建	本研究深度偵測技術
深度偵測準確度	高	差	高
速度	手動標註時間為準	3秒/幀	5秒/幀
折射校正	X	X	O
3D座標	X	X	O
追蹤功能	X	X	O

結論

研發技術

- 1 路徑追蹤
- 2 3D座標重建
  - a 深度偵測
  - b 折射校正

系統功能

- 1 活動量統計
- 2 成長情形紀錄
- 3 覓食行為辨識
- 4 死亡辨識
- 5 LINE自動推播
- 5 MySQL後端資料庫

未來展望

未來目標

- 進行場域實測
- 圖形使用者介面
- 增加多執行緒

實際應用

- 應用於養魚業
減少勞力成本
- 運用於其他
生物研究

參考資料

一、Mikel Broström (2020). *Real-time multi-object tracker using YOLOv5 and deep sort.* GitHub. https://github.com/mikel-brostrom/YOLODeepv5_Deep SORT_Pytorch

二、Doxygen (2022, March 11). *Depth Map from Stereo Images.*OpenCV. https://docs.opencv.org/3.4/dd/d53/tutorial_py_depthmap.html