



B11-深度學習蝦隻追蹤 與自動化測量系統

指導教授：翁添雄教授

專題組員：資工四A 吳秉謙
資工四A 吳典祐
資工四A 吳泓著

大綱

- 動機
- 目標
- 研究方法
- 流程圖
- 實作方法

動機

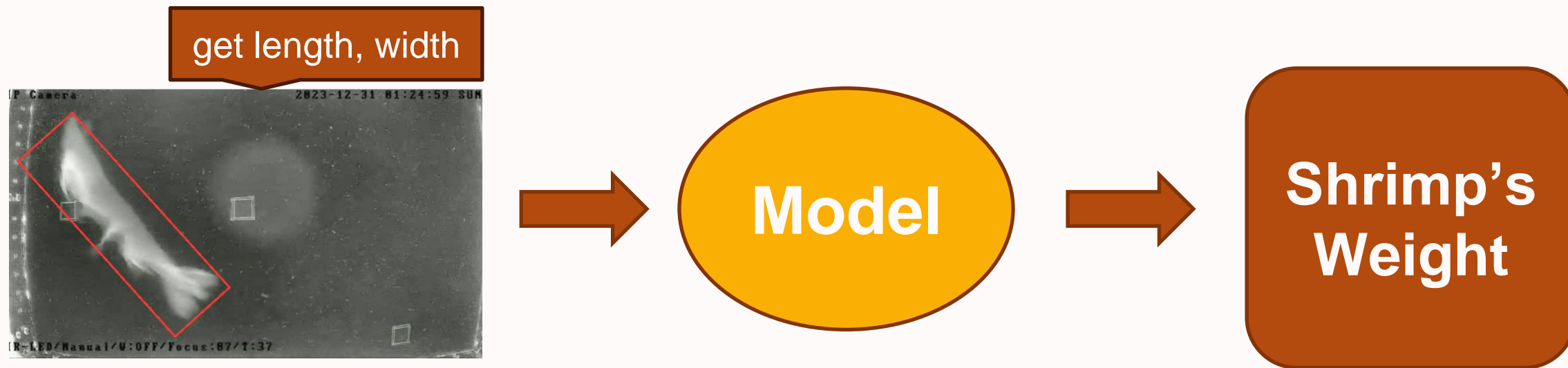
蝦隻的**重量和大小**是決定其市場價值的關鍵。

然而，目前大多數養殖戶仍然依賴**人工測量**來記錄蝦子的**長寬與重量**，這個過程十分**費時費力**。

開發提升蝦類養殖的**生產效率**和**品質管理**的自動化系統

目標

1.



2. 篩選掉模糊的影片，不對其進行檢測，以節省電力消耗。
3. 解決大量影像造成的儲存空間浪費。

研究方法



影像採集

使用攝影機來拍攝蝦子影片；
並且解決此法所衍生的問題。



物件偵測

辨識並框出影像中的蝦隻，
且獲得蝦隻的體長；
並且另外訓練一個模型來
判斷寬度。



預測模型

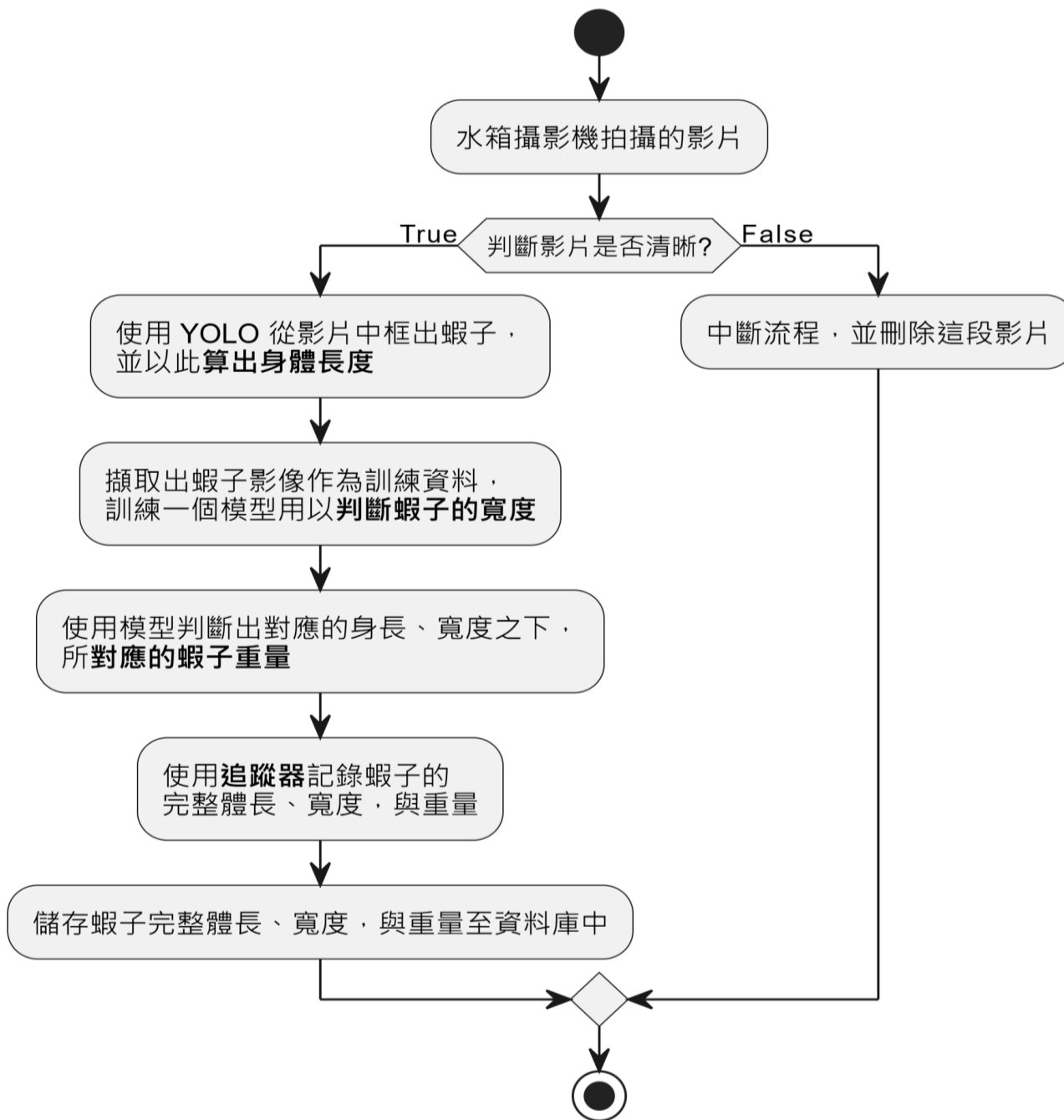
用蝦農提供的數據，訓練
預測模型；將其與先前得
到的體長、寬度對應後，
預測出其重量。



物件追蹤

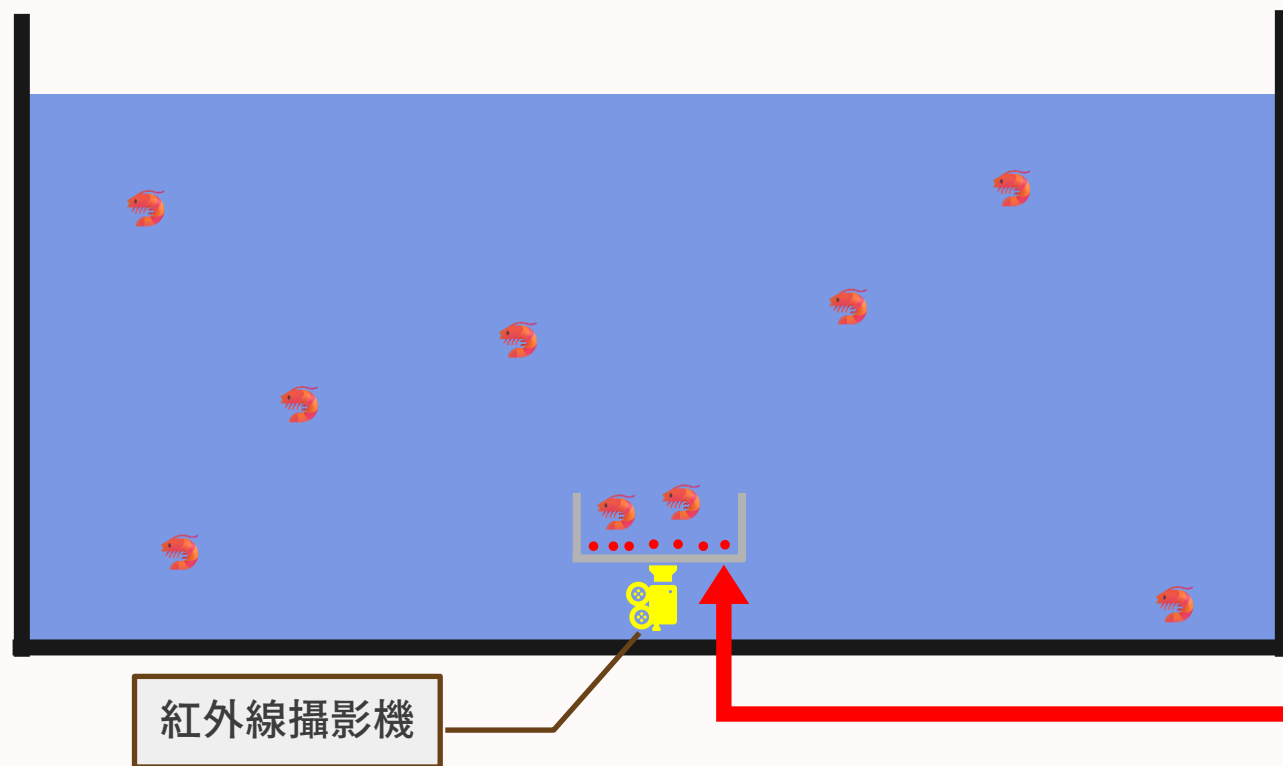
記錄並確保影像中，
蝦隻的完整資料。

流程圖



實作方法1 - 影像採集

實作方法1 - 影像採集示意圖



水箱攝影機

實作方法1 - 拍攝結果及影像標註示意圖



實作方法1 - 影像採集的**衍生問題**



儲存空間浪費

- 每部影片時間長
- 不是每個frame都會出現蝦子
- 水質模糊



電力浪費

- GPU對模糊影片的不必要使用

實作方法1 - 解決方法



Image Classification with Logistic Regression

區分哪些影像水質能夠進行蝦隻辨識

無法辨識就不需要消耗電力進行辨識

- 水質清楚 -> 進行蝦隻辨識
- 水質模糊 -> 中斷執行並刪除影像

一部影片的耗電量差異(以GPU為準)

完整跑完模糊的影片所需的耗電量

```
GPU 總耗電量: 24173.10 J (6.7147 Wh)  
平均功耗: 159.32 W  
執行時間: 00:00:31
```

篩掉模糊的影片所需的耗電量

```
Predicted Class: blur  
GPU 總耗電量: 133.18 J (0.0370 Wh)
```

一部影片可節省約 **6.6777 Wh**

(如果是模糊的影片)

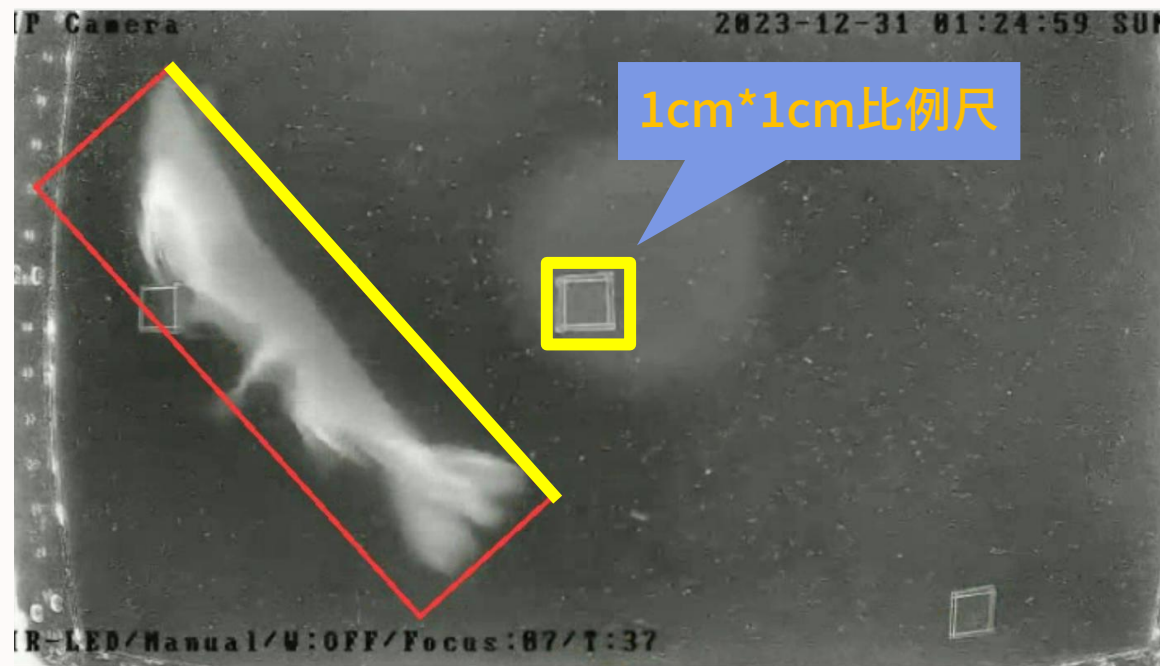
	完整跑完	判斷是否模糊
耗電量	6.7147 Wh	0.0370 Wh
耗時	151秒	1秒

實作方法2 - 物件偵測

實作方法2 - 物件偵測

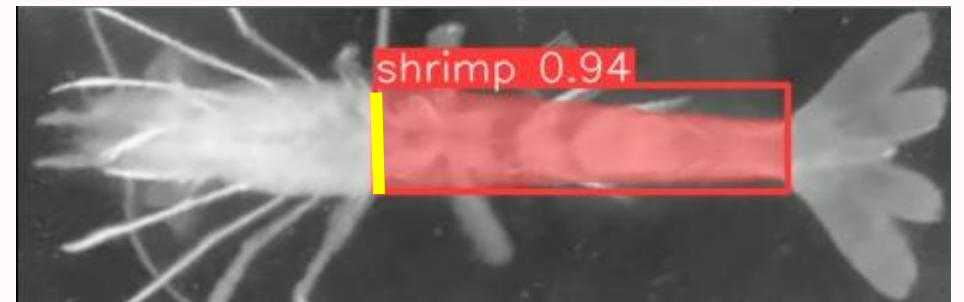
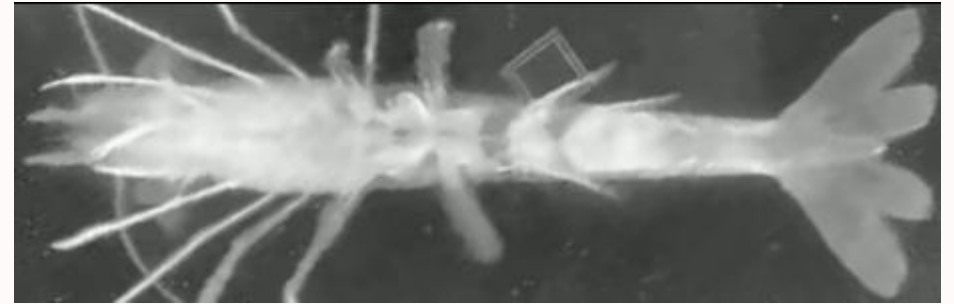
- 本次實驗中，我們使用了YOLOv8-OBB來進行檢測。
- 檢測並框出蝦子後，**透過比例尺來推算出蝦隻的體長**

只保留有檢測到蝦子的影格，
捨棄沒出現蝦子的影格



實作方法2 - 判斷寬度

- 將蝦子的影像部分擷取出來，作為後續的訓練資料
- 訓練一個segmentation model，來判斷出寬度



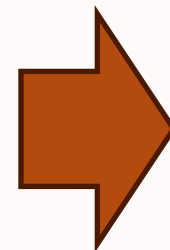
實作方法2 - 總結 - 可節省空間

原始一個月總影片是 90G

透過程式碼

只保留出現蝦子的影格，
也不保留水質不佳的影像。

最後保留下的結果為 42G



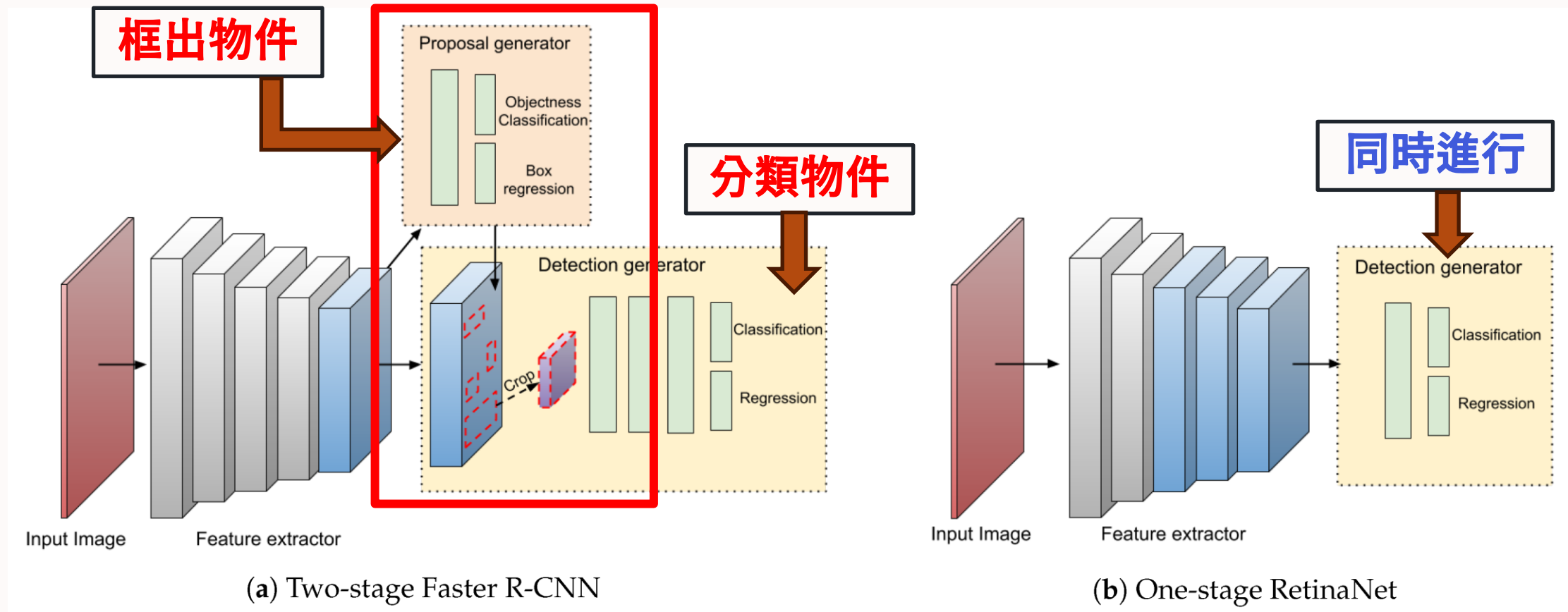
節省了53%

```
[(base) providence@ubun22:~$ du -sh clipped_vid/  
42G      clipped_vid/
```

實作方法2 - 節省空間DEMO



實作方法2 - 物件偵測 - 模型架構差異



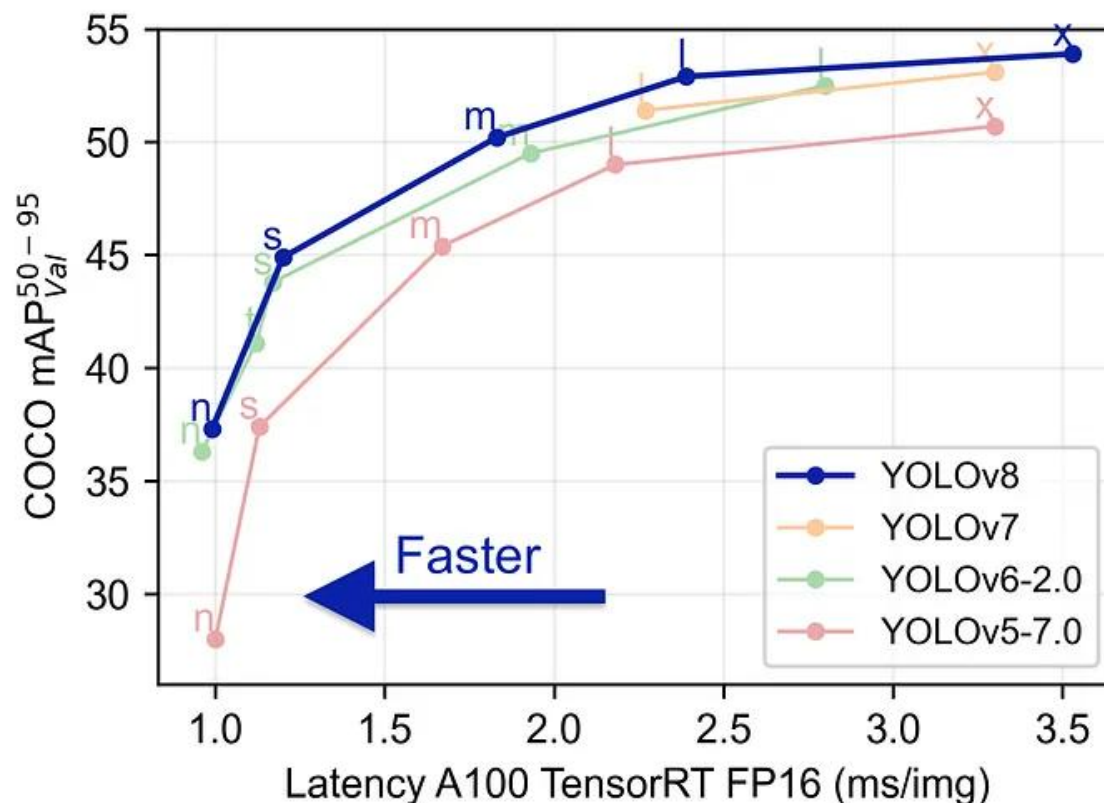
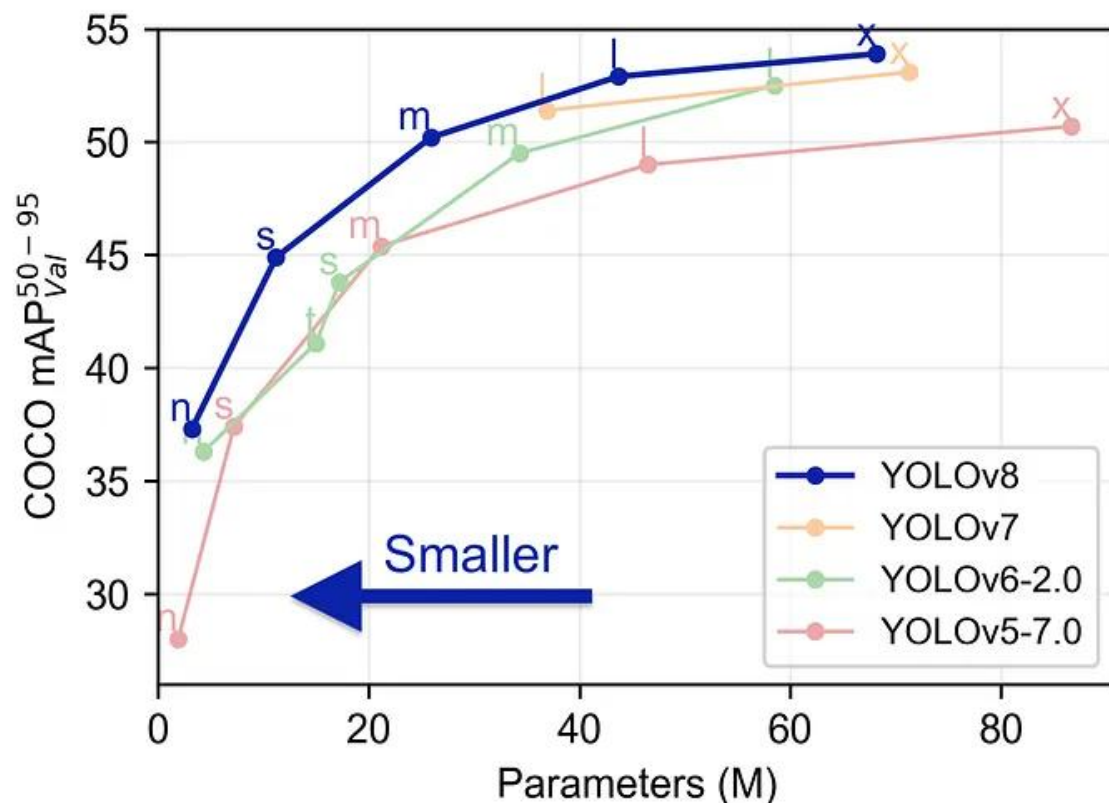
實作方法2 - 物件偵測 - 模型效能差異

Table 4. The Indicators of Models in Identifying Difficult Samples

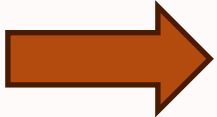
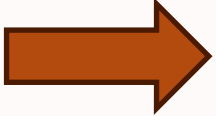
Algorithm	MAP/%	FPS	Model Size
YOLO v3	78.52	69	89M
Faster R-CNN	79.63	3	426M
SSD	78.69	41	149M

實作方法2 - 物件偵測 - YOLO版本差異

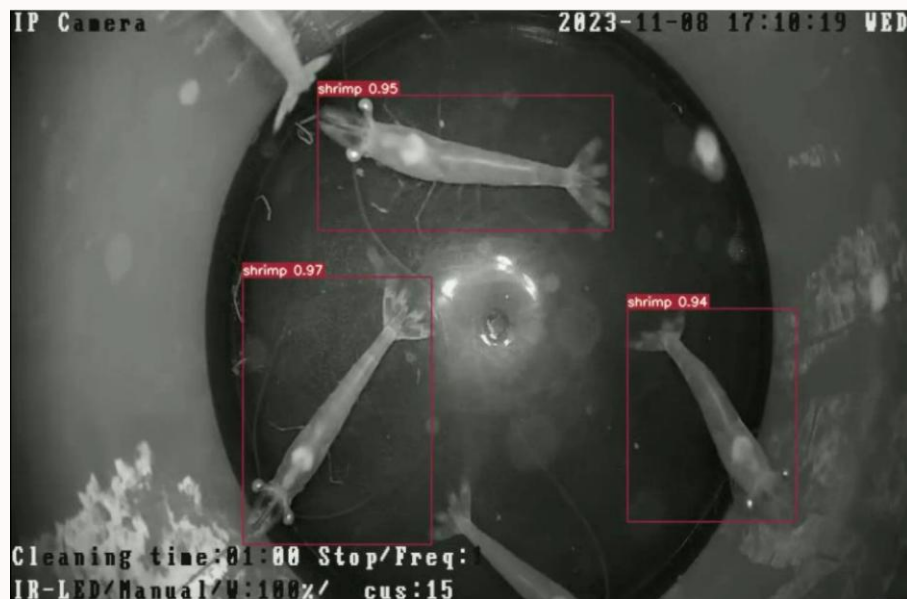
- 在今天的實驗中，我們使用的YOLO版本是v8，因為更快更準確。



實作方法2 - 物件偵測 - YOLOv8-OBB

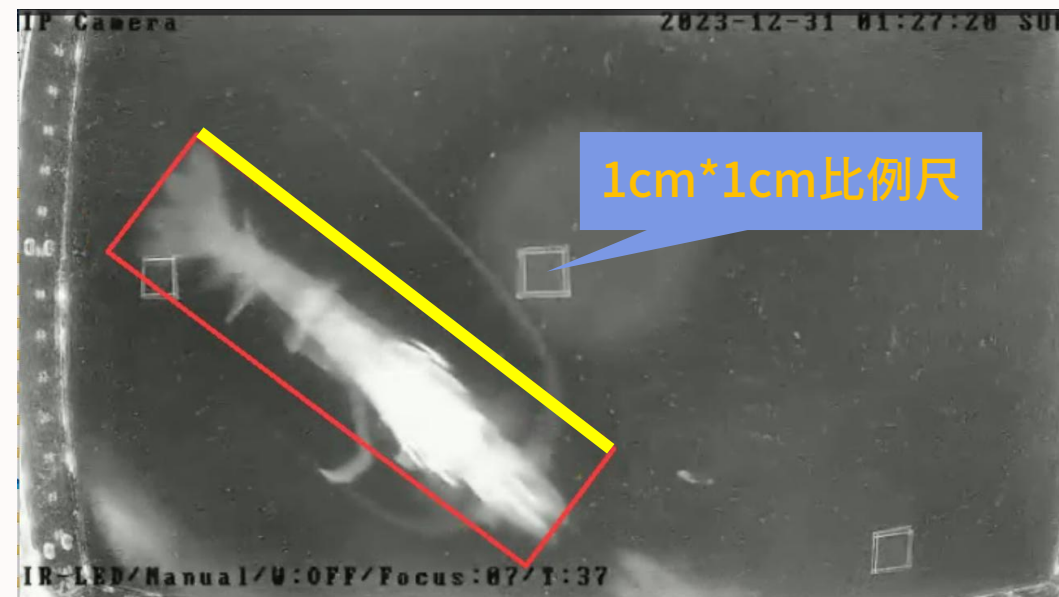
- YOLOv8-**OBB** (**O**riented **B**ounding **B**ox)
- 與普通v8的差異  Bounding Box 可隨目標旋轉
- 為何本實驗要使用  可以更加貼和目標，使測量變得容易

實作方法2 - 物件偵測 - 使用YOLOv8-OBB原因



Axis-Aligned Bounding Box (AABB)

僅能辨識並框出物件所在**位置**



Oriented Bounding Box (OBB)

可計算矩形框的最長邊獲得**蝦子身長**

實作方法2 - 物件偵測 - 使用YOLOv8-OBB原因

- 總結:

1. YOLO模型參數較少較輕量。
2. 辨識速度較快(即時)。
3. 準確度與其他模型差異不大。
4. 爲了使用它的OBB版本來獲取蝦隻的體長。

實作方法3 - 預測模型

實作方法3 – 為何用模型預測體重？

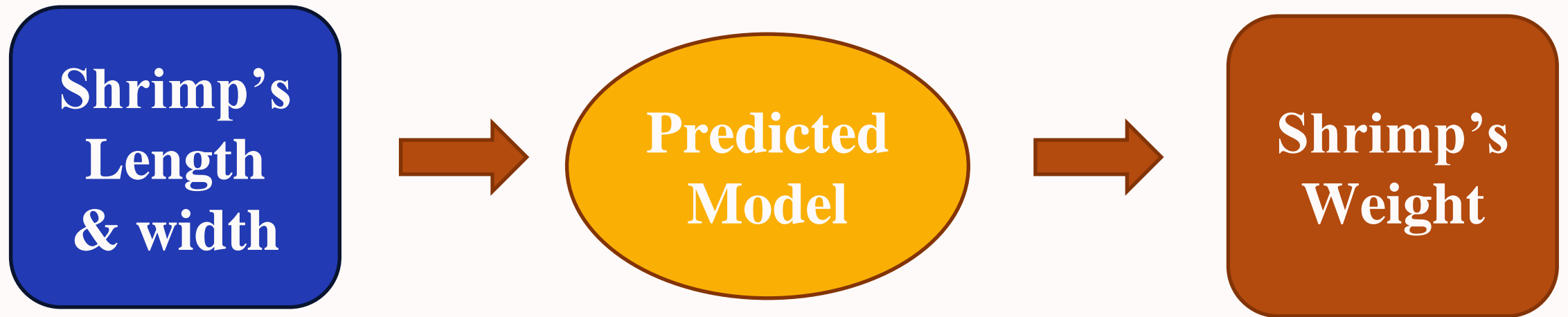
池號	12E-2池	家族編號	NCKU#10f2 (P0295_20230914)		
測量日期	序號	體重 (g)	全長 (mm)	頭胸甲長 (mm)	寬度 (腹部第1節處) (mm)
2024/6/26	1	48	166.6	70.3	15
	2	51	199.1	80.8	20.4
	3	16	136.3	57.4	12.2
	4	23	150.4	63.2	14.5
	5	40	177.3	73.6	17.7
	6	32	161.7	71.2	17.1
	7	13	120.6	50.8	10.2
	8	22	147.2	62.8	14.4
	9	23	149.7	61.8	13.7
	10	20	145	63	13.5
	11	15	135.7	61.2	13.8
	12	19	157.6	67	16
	13	10	117	49.8	11.6
	14	9	120.4	52.7	11
	15	4	111.1	47.5	8.7
	16	4	101.7	43.5	8.2
	17	10	113.8	50.3	9.3
	18	6	117	49.3	12.3
	19	2	90.8	38.9	7.2
	20	5	92.8	39.5	8.1
	21	4	88.1	38	8.4



費時費力

人工測量出共202筆蝦子長寬與體重的數據

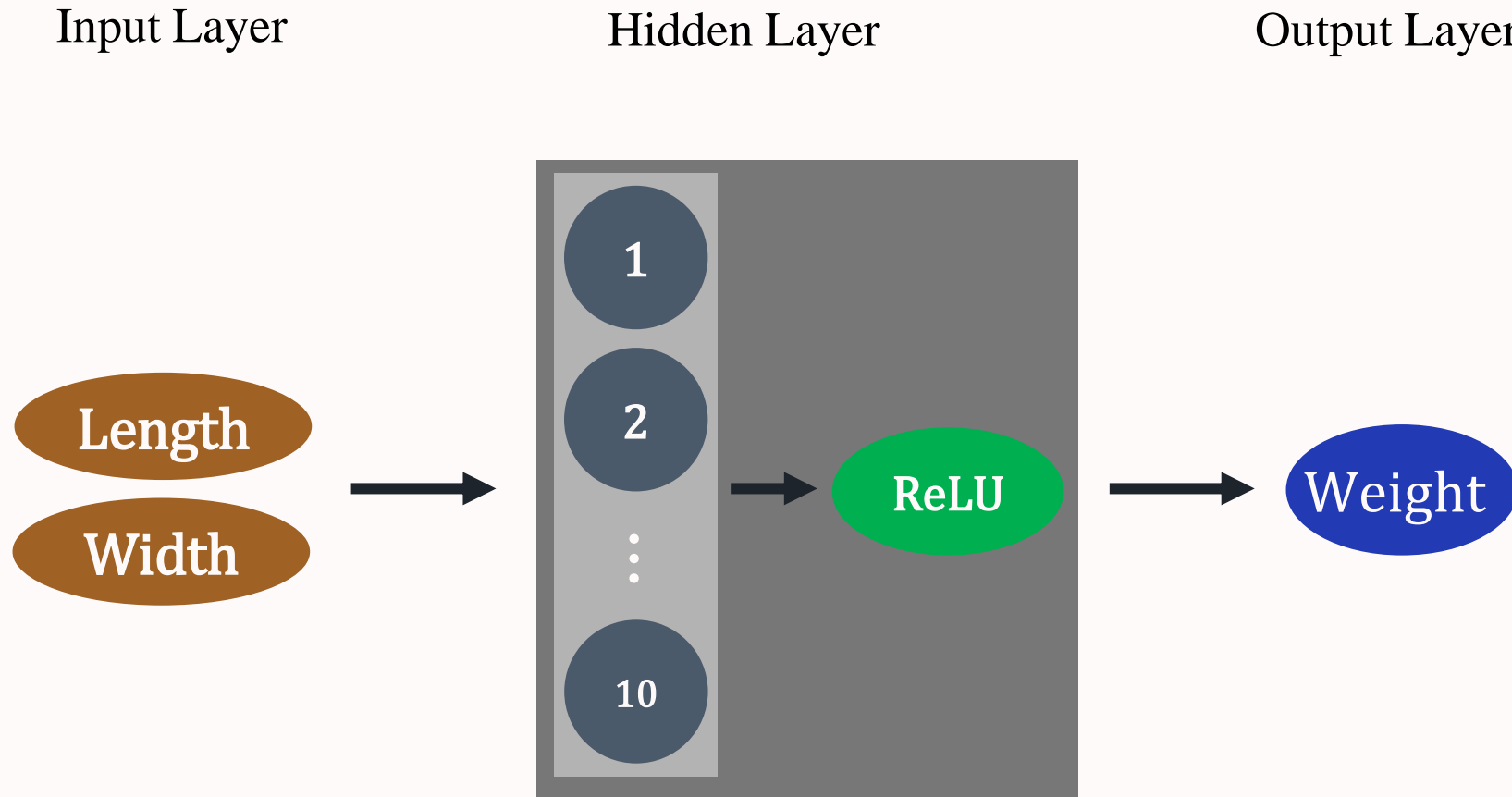
實作方法3 - 深度學習預測蝦隻體重



實作方法3 - 資料預處理

- 蝦子長寬與重量進行Standardization
- 資料集：80%訓練集，20%驗證集

實作方法3 - 模型架構 - 單層前饋神經網路



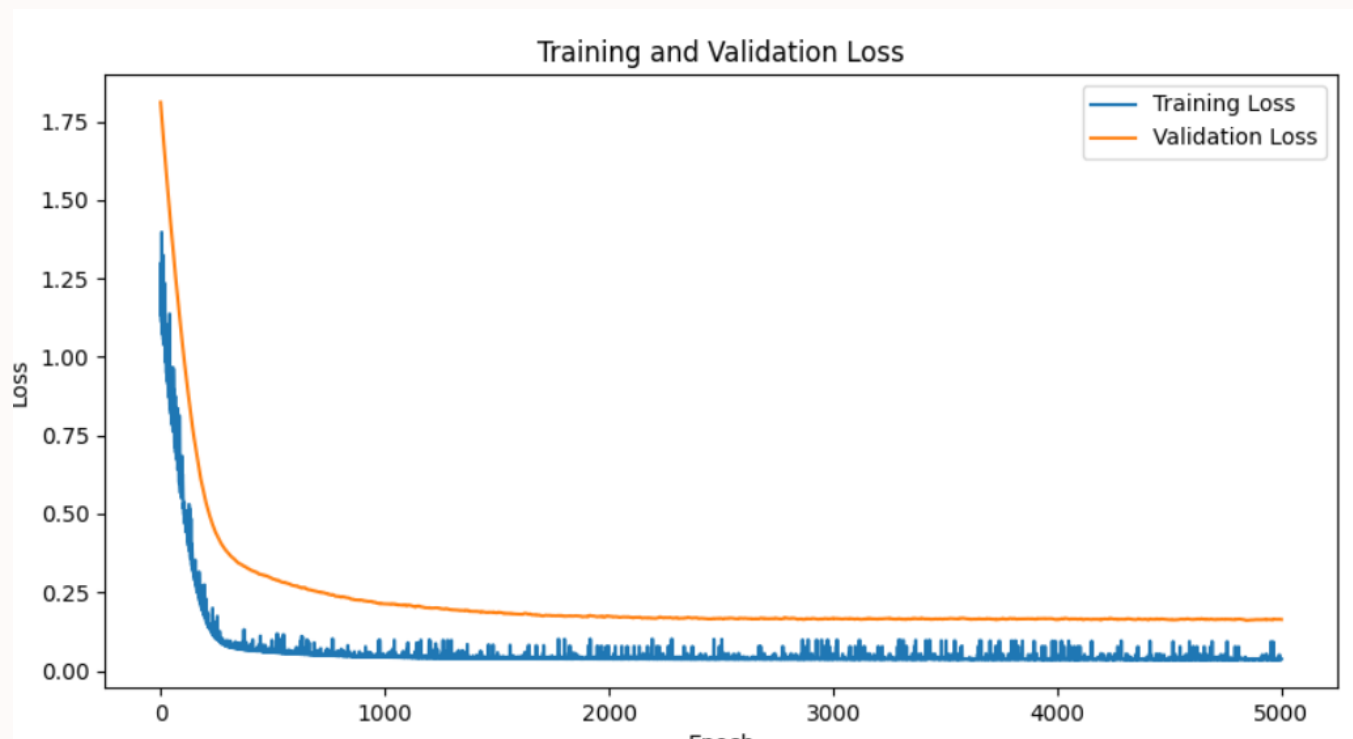
實作方法3 - 模型訓練 - 超參數選擇

Parameters:

- Epochs = 5000
- batch_size = 8
- learning_rate = 1e-04
- Optimizer = Adam
- 均方誤差 (MSE) : 5.3394
- 平均絕對誤差 (MAE) : 3.0888

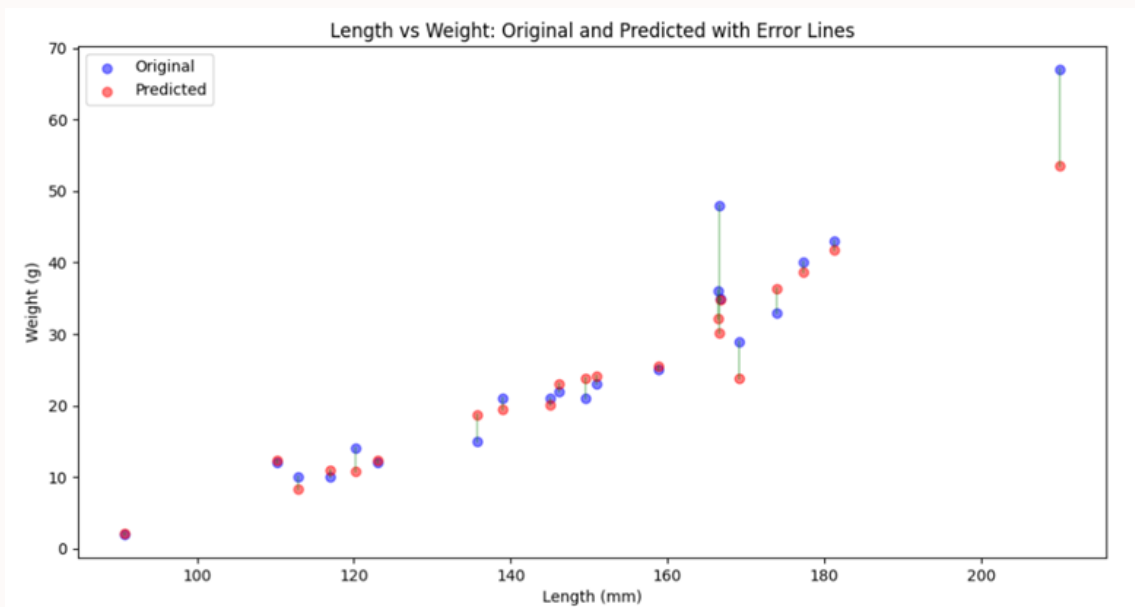
實作方法3 - 模型訓練結果

訓練與驗證損失圖

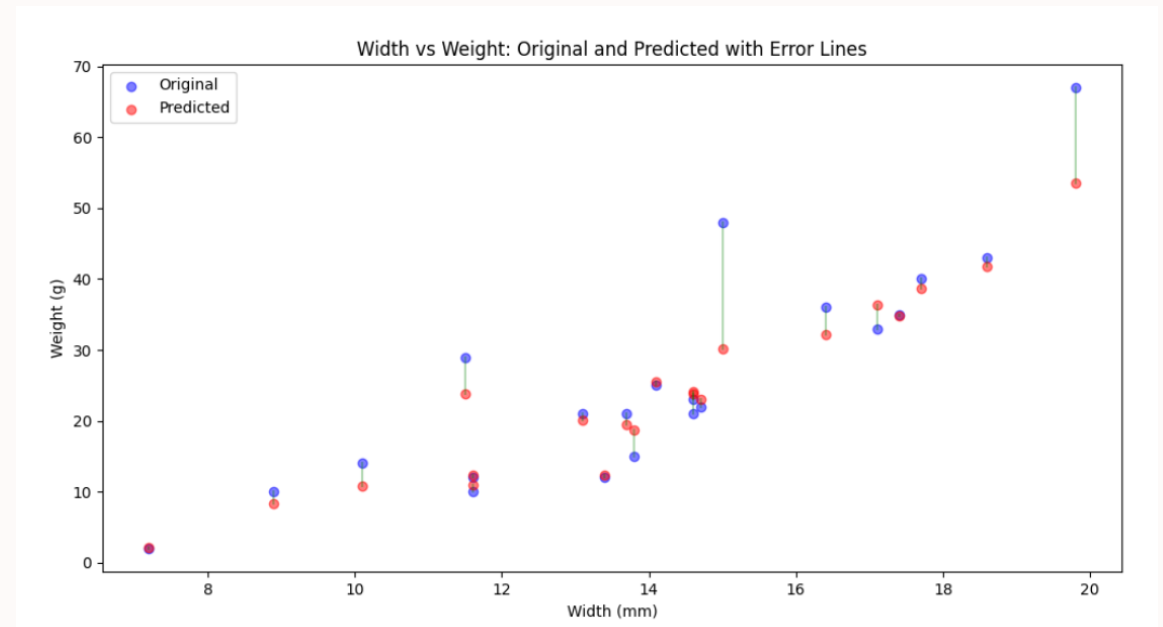


實作方法3 - 模型預測結果

- 實際長度與重量與模型預測重量的比較圖



- 實際寬度與重量與模型預測的比較圖



實作方法3 – 體重預測DEMO



實作方法4 - 物件追蹤

實作方法4 - 物件追蹤



Why object tracking?

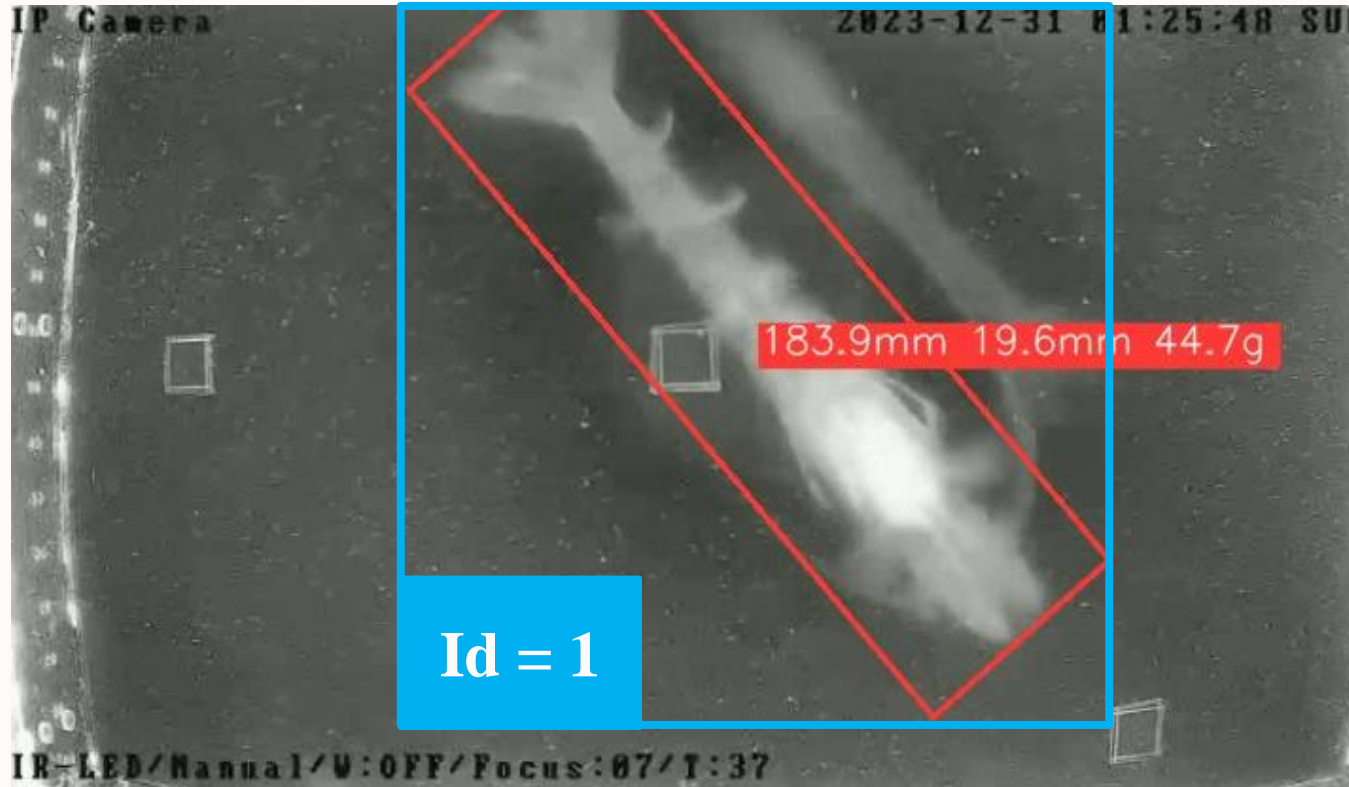
長度受到哪些因素影響?

- 蝦隻的傾斜角度
- 蝦隻在影像邊緣出現

Result

- 抓到的蝦隻不是完整長度
- 儲存太多不完整或重複的資料

實作方法4 - 物件追蹤



EX:在影片中的00:00:00~00:00:02
追蹤到一隻蝦子(1 sec = 15 frames)

加上追蹤器

- .. 追蹤蝦子知道這隻蝦子出現的時間
- .. 無法紀錄一筆最完整的長度、體重
- .. 能導致儲存多資料錯誤但資料記錄
- .. 確保資料併儲時問題會翻倍

Frame
length
weight

1	2	3	30	N
190	190	195	200	215	...	205
55	55	57	59	65	...	60

實作方法4 - 物件追蹤 - 追蹤器的條件

1) Market-1501

Id:1



Id:2



行人重新識別資料集

追蹤器的條件

- 根據運動軌跡追蹤
- 能夠進行重新識別

Re-identification的難處

- 製作Reid資料集需要
針對連續影像進行編號
- 十分耗時且費力

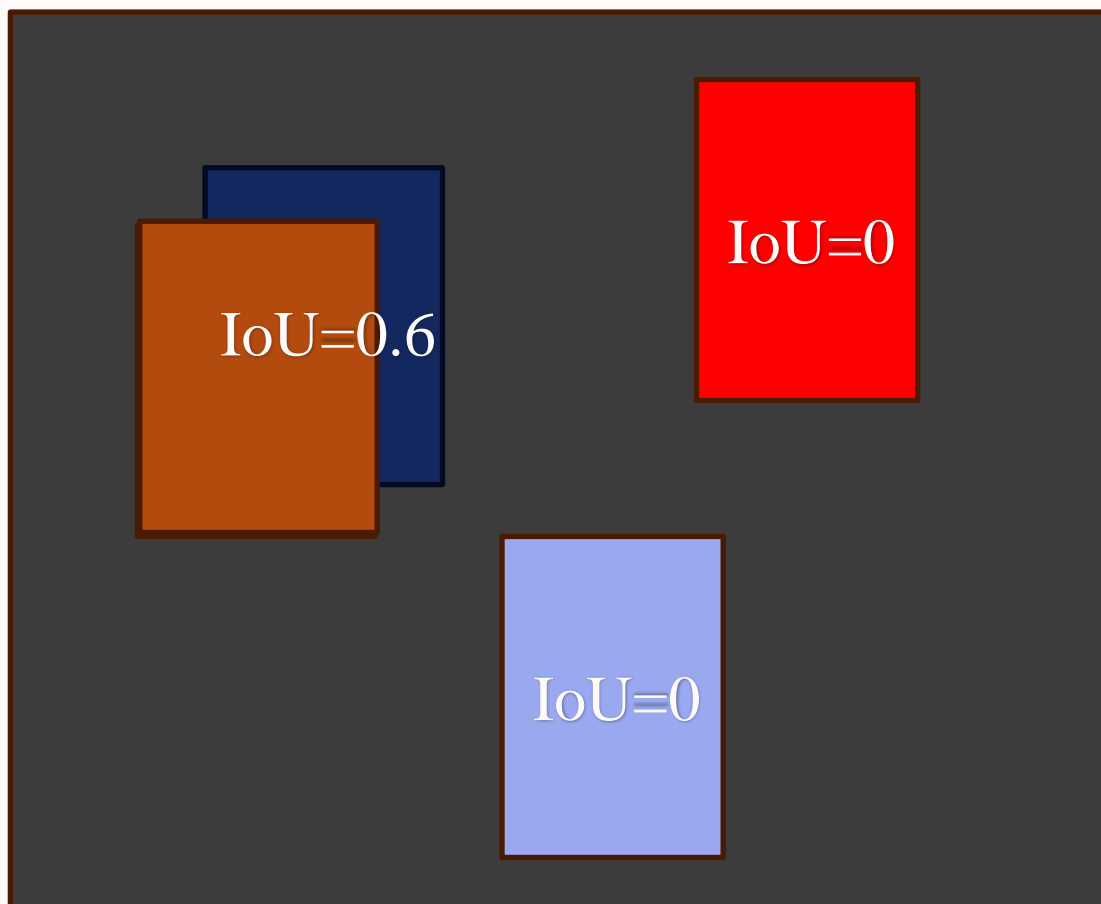
實作方法4 - 物件追蹤 - 選擇追蹤器



選擇Norfair的原因

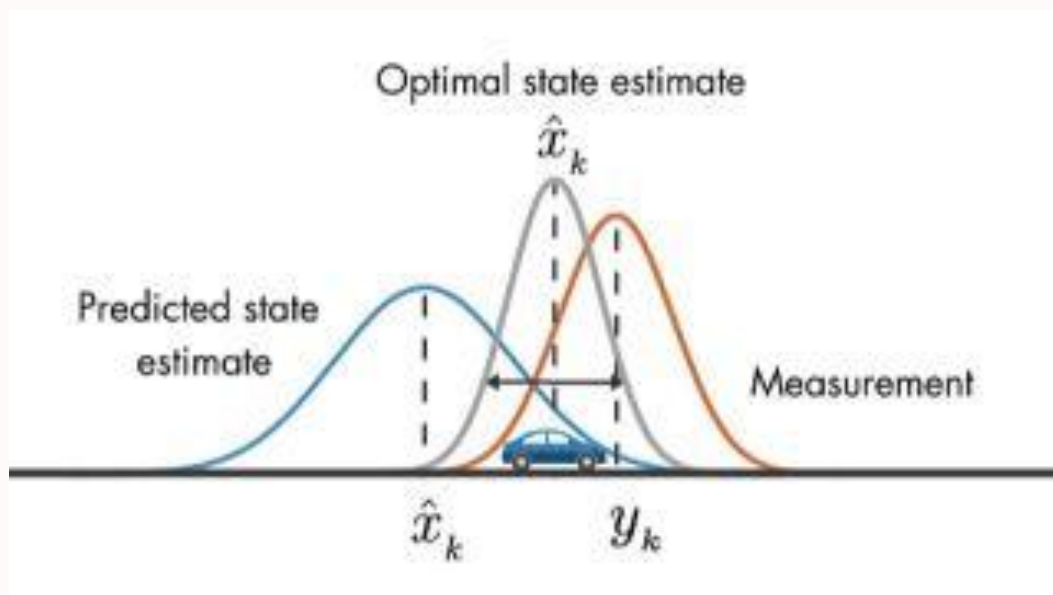
- Norfair可以通過自定義外觀或特徵的距離函數來達成Reid的功能。
- 可避免花費大量時間標註與製作資料集

實作方法4 – 追蹤器的追蹤流程



- 透過計算 IoU (Intersection over Union) 的距離度量方法，前一幀的物體與未被追蹤的新物體之間兩個檢測框重疊程度。
- 選擇 IoU 值越高的物件作為匹配對象，並將其 ID 分配給物體。

實作方法4 - Norfair的追蹤流程 - 卡爾曼濾波



EX:以車子移動來舉例

- 卡爾曼濾波用於狀態估計的技術，主要用於**預測下一個狀態**和**更新當前狀態**。
- 可以想成透過卡爾曼濾波我們可以**預測和校正物體的追蹤框位置**。

實作方法4 - 追蹤流程 - Re-identification



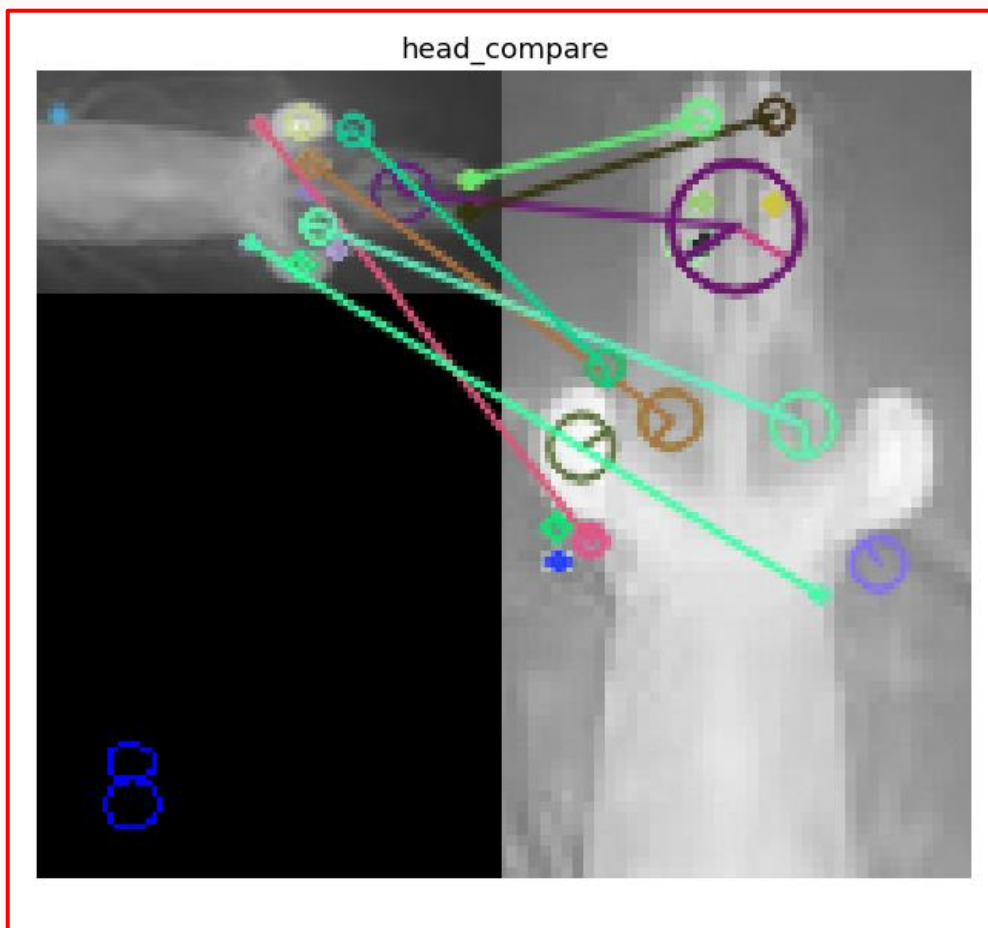
蝦子移動時會出現的問題

- 會被其他蝦子遮擋
- 可能會游出鏡頭範圍外

如何實現 **Reid** ?

- 將辨識到的蝦隻裁切出來

實作方法4 - 追蹤流程 - Reid



- 使用SIFT(Scale-invariant feature transform)特徵點檢測算法
- 將新影像的物件與消失的追蹤對象比較特徵點去計算匹配率
- 最後設定閾值來判定是否為相同物件達成物體的重新識別

$$\text{match_rate} = \frac{2 \times \text{len(matches)}}{\text{len(des2)} + \text{len(des1)}}$$

實作方法4 - 物件追蹤 - 資料紀錄

	Start time	End time	Max len	Weight
1	00:00_10	00:01_26	178.6	41.5
2	00:04_65	00:05_83	198.9	55.2
3	00:05_81	00:06_94	167	33.7



Shrimp
VideoName
StartTime
EndTime
MaxLength
Weight

實作方法4 - 追蹤DEMO



實作方法4 - 物件追蹤 - 物件追蹤DEMO



非常感謝評審們的耐心聆聽！