口罩偵測與社交距離追蹤之防疫系統

指導老師： 連震杰
學生： 林瑋晟、李淑汶

2022/06/07

CONTENTS

1. Motivation

2. Face Mask Detection using YOLOv4

3. Social Distance Tracking using DeepSORT

4. Results of Work

01

Motivation

Motivation

動機

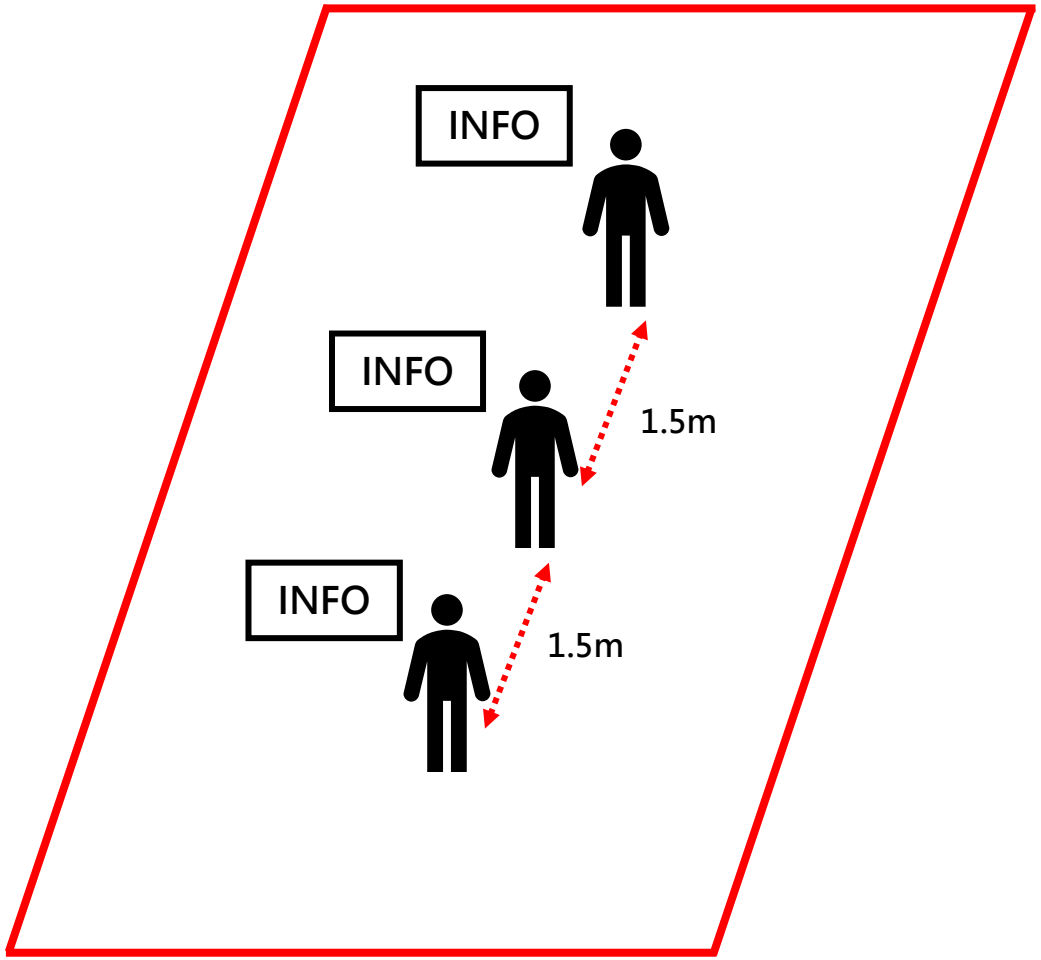
自動辨識佩戴口罩及維持社交距離

在疫情肆虐當下，希望能透過自動化系統來實作偵測及追蹤技術，搭配警告圖示或提示聲，確保商店內顧客皆有依照政府指示配戴口罩，並且保持社交安全距離，同時避免店員與顧客的衝突。

實作功能

1. 辨識是否佩戴口罩
2. 偵測兩人之間的社交距離
3. 顯示顧客資訊 (人流量)

Motivation



物件偵測、追蹤

口罩辨識+人流追蹤系統

使用RGB-D Camera搭配深度學習模型實作物件偵測、追蹤，並在畫面即時顯示相關資訊，例如人流量(編號)、社交距離、是否配戴口罩等等。

使用之技術及器材

1. RGB-D Camera (Realsense D435)
2. Object Detection
3. Multiple Object Tracking
4. Distance Calculate

Face Mask Detection using YOLOv4

YOLOv4

Framework

Input

YOLOv4

Head
(YOLOv3)

Neck
(SPP + PAN)

Backbone
(CSPDarknet53)

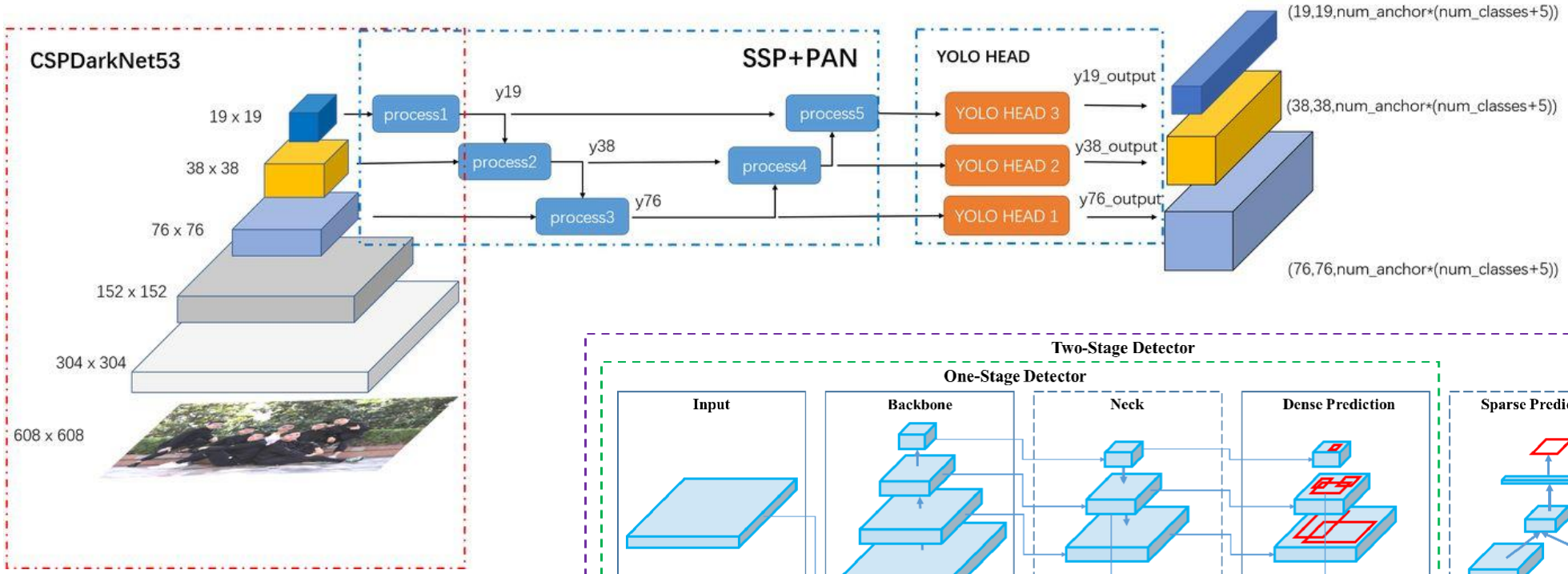
Input

Result

YOLOv4

Model Architecture

- Backbone : CSPDarknet53
- Neck : SPP + PAN
- Head : YOLOv3

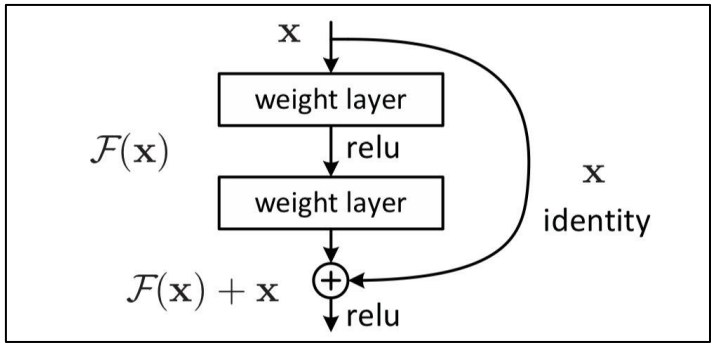


YOLOv4

Backbone：CSPDarknet53

目標: 萃取特徵

- Darknet53
  - 總共 53層 Convolutional (包含 Conv2D、Batch Normalization、Activation)。
  - 借鑒了 ResNet 的 Residual block，使網絡結構更深、尺寸更大。
  - 最後三層: Avgpool、 Connected、 Softmax Layer 僅在 ImageNet 數據集中做訓練使用。
  - 在 Yolov4中使用 Mish做激活函數。



Residual block

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

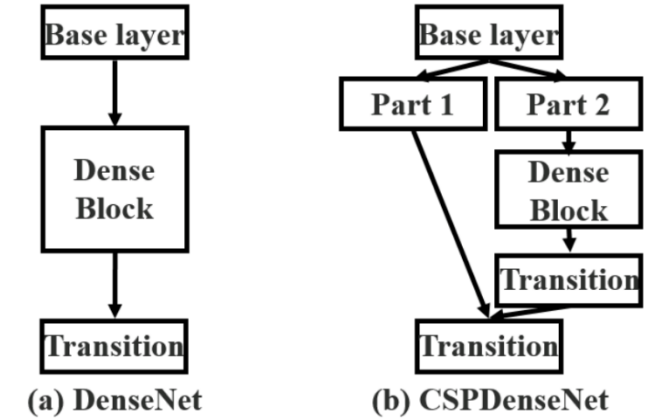
Darknet-53

YOLOv4

Backbone：CSPDarknet53

目標: 萃取特徵

- CSPNet(Cross Stage Partial Network)
  - 保持甚至提高模型準確率的同時，降低 10%~ 20%的計算量。
  - 將 Base layer按比例分成兩部分後， 其中一份原封不動，另外一份經過 Resblock body 以及 transition 後，concate 兩部分，再經過一次 transition。
  - Base layer：ResBlock 前的 Convolutional輸出的 feature map。
  - Transition layer：1x1 Conv. Layer及 Pooling Layer。



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Resblock body

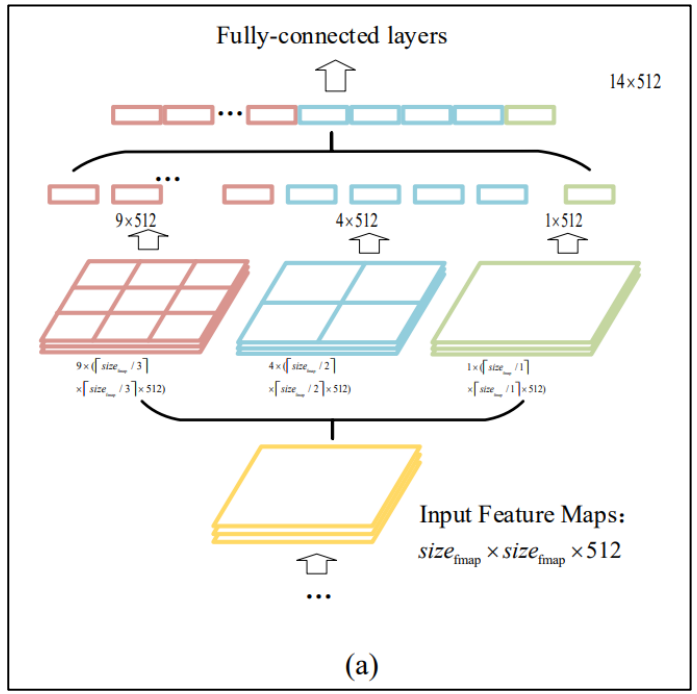
Resblock

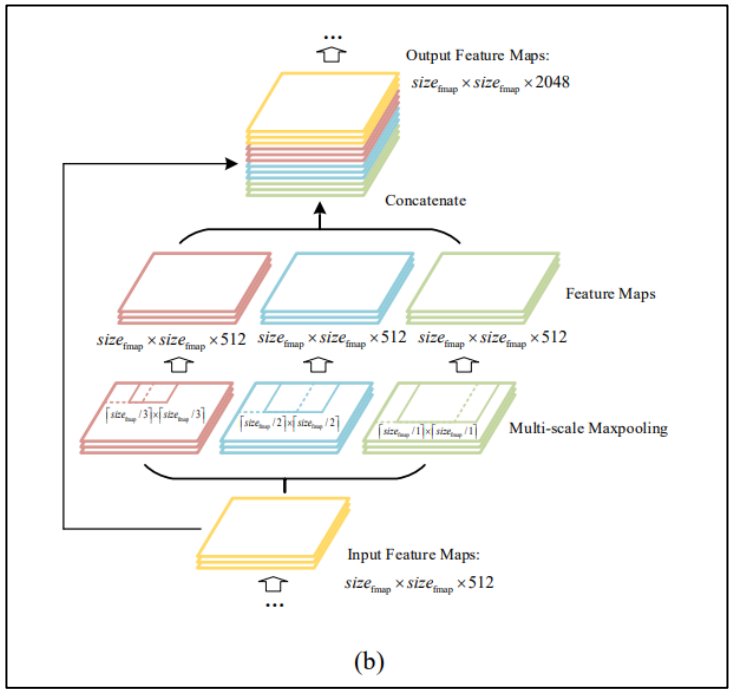
YOLOv4

Neck：SPP + PAN

目標: 更好的融合特徵

- SPP(Spatial Pyramid Pooling)
  - 使任意大小的 feature map輸出成固定大小
  - 最後一層從 fully connect改成 concatenation，後面得以繼續添加 CNN module



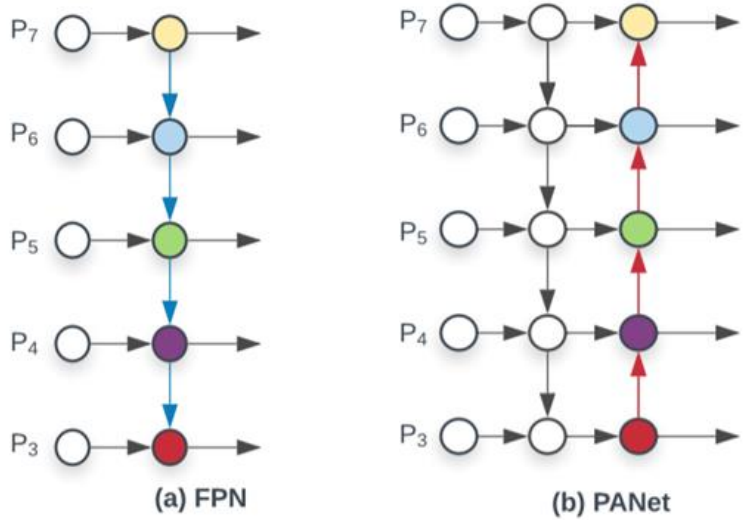


YOLOv4

Neck：SPP + PAN

目標: 更好的融合特徵

- PANet(Path Aggregation Network)
  - 基於 FPN(Feature Pyramid Network)，新增 bottom-up path augmentation
  - 將原先 PAN利用相加獲得新的 feature map的方法改成 concatenation



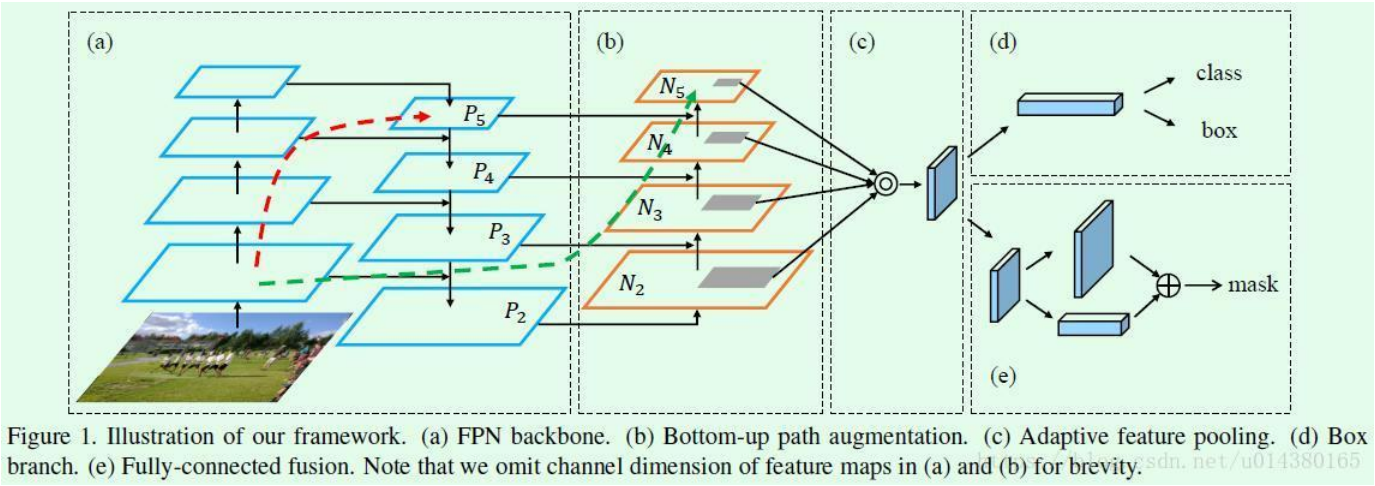


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.

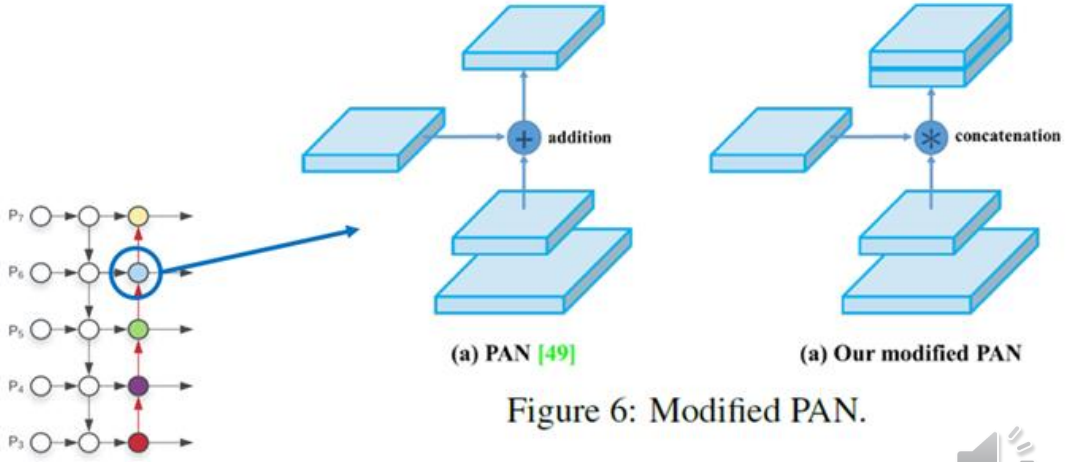
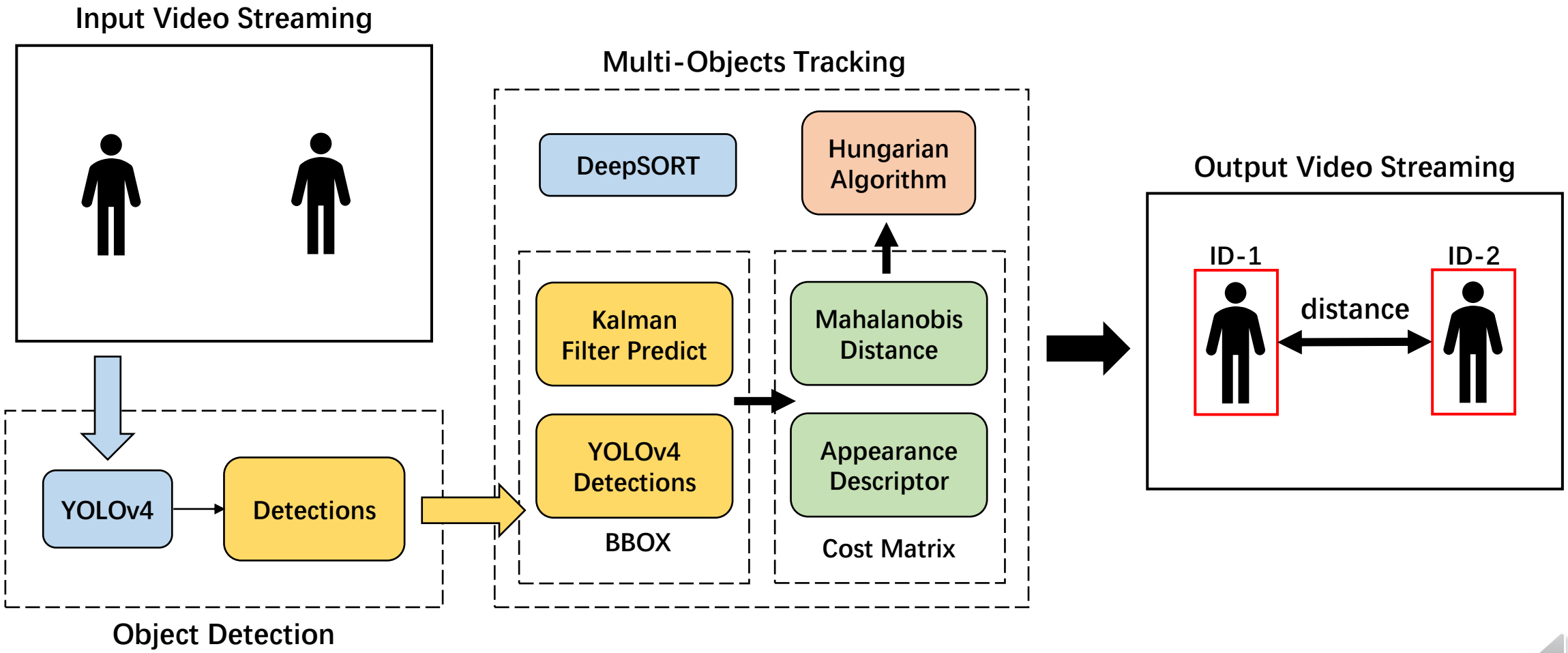


Figure 6: Modified PAN.

03. Social Distance Tracking using DeepSORT

DeepSORT

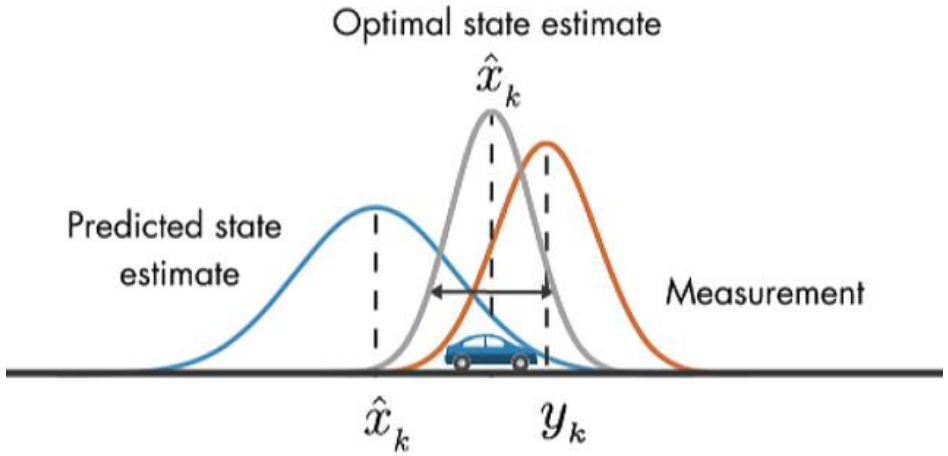
Framework



DeepSORT

基於 SORT 演算法上再改進

- SORT 演算法簡介:
  - 利用 Kalman Filter 預測目標位置
    - 預測當前位置以提高準確度
    - 藉由 $\text{bbox}_{\text{prediction}}$ 及 $\text{bbox}_{\text{dectection}}$ 計算出 $\text{bbox}_{\text{optimal}}$
    - $\mathbf{X} = [u, v, r, h, \dot{x}, \dot{y}, \dot{r}, \dot{h}]^T$ (DeepSORT 版本)
    - u, v 代表 bbox 的 x, y 座標、 r 代表長寬比、 h 代表高度，以及其對應的速度 $\dot{x}, \dot{y}, \dot{r}, \dot{h}$
  - 通過 Hungarian algorithm 關聯檢測框及目標
    - 解決detections及tracks的匹配問題
  - 以 IoU 作爲前後幀間目標關係度量指標
    - 以 IoU 作為 Cost Matrix 基準值
    - 設置 min IoU threshold



Kalman Filter

DeepSORT

Cost matrix (Hungarian algorithm)

目標: 找到更好的匹配成本

- Motion information
  - Mahalanobis distance $d^{(1)}$ (Kalman 預測框與檢測框)
  - $d^{(1)}(i,j) = (d_j - y_i)^T S_i^{-1} (d_j - y_i)$

  ※ detection d_j, track y_i, d 和 y 的斜方差矩陣 S_i^{-1}
- Appearance information
  - Re-Identification (ReID)
  - cosine distance $d^{(2)}$ between track and detection
  - $d^{(2)}(i,j) = min\{ (1 - r_j^T r_k^{(i)}) \mid r_k^{(i)} \in R_i \}$

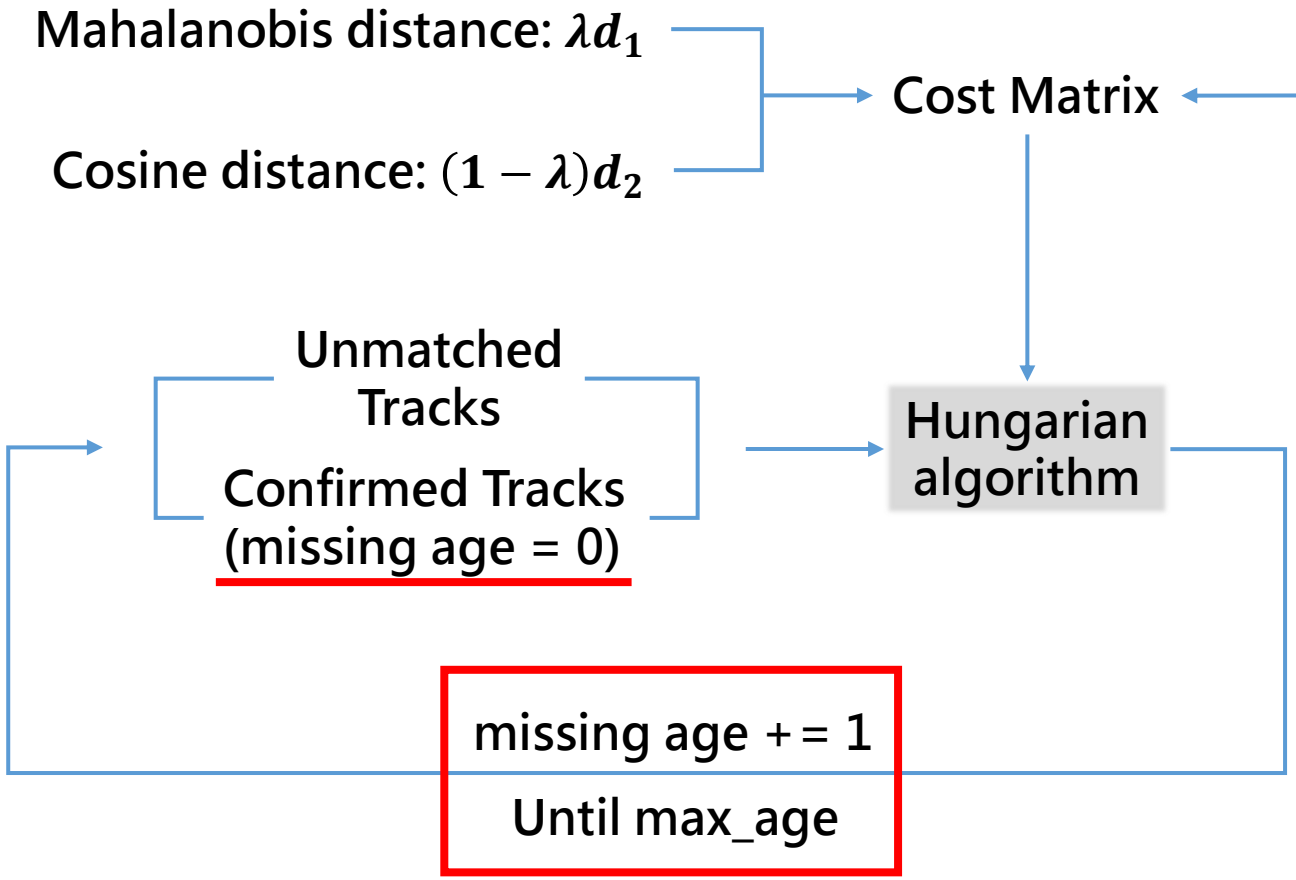
    ※ appearance descriptor r
- $Cost(i,j) = \lambda d^{(1)}(i,j) + (1-\lambda) d^{(2)}(i,j)$
  - 根據使用環境不同，調整使用不同λ，但原則上以d_2為主 (Appearance Information)

Name	Patch Size/Stride	Output Size
Conv 1	$3 \times 3/1$	$32 \times 128 \times 64$
Conv 2	$3 \times 3/1$	$32 \times 128 \times 64$
Max Pool 3	$3 \times 3/2$	$32 \times 64 \times 32$
Residual 4	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 5	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 6	$3 \times 3/2$	$64 \times 32 \times 16$
Residual 7	$3 \times 3/1$	$64 \times 32 \times 16$
Residual 8	$3 \times 3/2$	$128 \times 16 \times 8$
Residual 9	$3 \times 3/1$	$128 \times 16 \times 8$
Dense 10		128
Batch and ℓ_2 normalization		128

ReID CNN architecture

DeepSORT

Matching Cascade



Listing 1 Matching Cascade

Input: Track indices $\mathcal{T} = \{1, \ldots, N\}$, Detection indices $\mathcal{D} = \{1, \ldots, M\}$, Maximum age $A_{\max}$

Compute cost matrix $\boldsymbol{C} = [c_{i,j}]$ using Eq. 5
Compute gate matrix $\boldsymbol{B} = [b_{i,j}]$ using Eq. 6
Initialize set of matches $\mathcal{M} \leftarrow \emptyset$
Initialize set of unmatched detections $\mathcal{U} \leftarrow \mathcal{D}$
for $n \in \{1, \ldots, A_{\max}\}$ **do**
  Select tracks by age $\mathcal{T}_n \leftarrow \{i \in \mathcal{T} \mid a_i = n\}$
  $[x_{i,j}] \leftarrow \text{min\_cost\_matching}(\boldsymbol{C}, \mathcal{T}_n, \mathcal{U})$
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{(i,j) \mid b_{i,j} \cdot x_{i,j} > 0\}$
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{j \mid \sum_i b_{i,j} \cdot x_{i,j} > 0\}$
end for
return $\mathcal{M}, \mathcal{U}$

設置消失時間上限，當物件一定時間內
都沒被追蹤到，就會從Tracks中被移除

Distance Calculate

RGD-D Camera

Distance Calculate

- Intrinsic Camera Parameters
- Distortion Models
- Depth Image
- 將圖像座標轉換成3D的點座標並計算距離
  - $x = (u - c_x) \times d \div f_x$
  - $y = (v - c_y) \times d \div f_y$
  - $z = d$
  - $distance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$

※ Pixel Coordinates (u, v), Depth d

※ Point Coordinates (x, y, z), Principal Point (c_x, c_y), Focal Length (f_x, f_y)

04

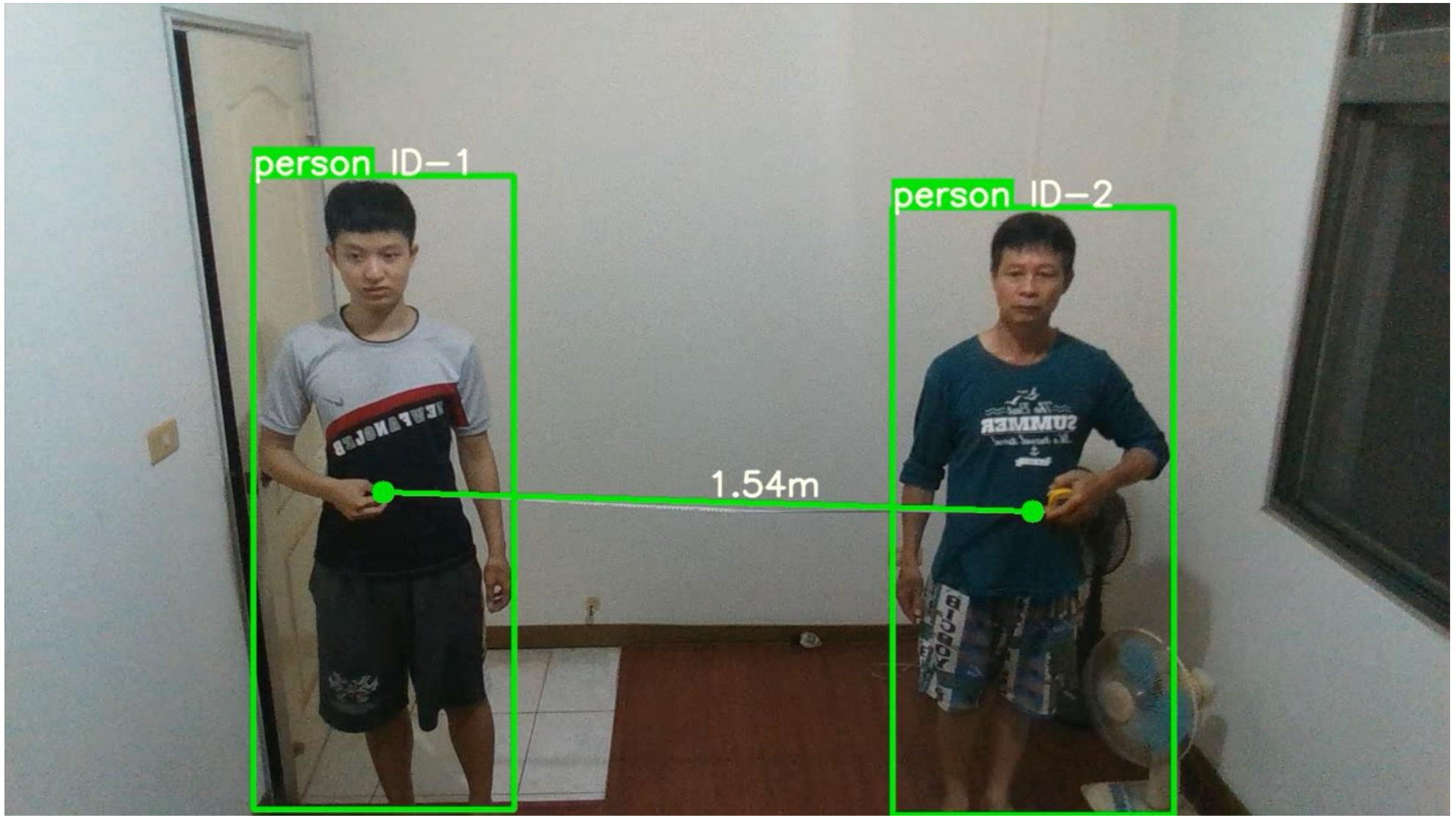
Results of Work

Results of Work

Face Mask Detection using YOLOv4

Results of Work

Social Distance Tracking using DeepSORT



THANKS

感 謝 您 的 觀 看

2022