國立清華大學資訊工程系 106 學年度下學期專題報告

專題名稱	家用型服務機器人的視覺辨識應用				
參加競賽或計畫	□ 參加對外競賽	□ 參與其他計畫	■無參加對外競賽或任何計畫		
學號	104021227	104070040	104062122	104081011	104062137
姓名	賴宥儒	林聖亞	鄭又維	葉韋辰	黃彥儒

摘要

隨著科技進步，機器有能力可以蒐集、知道、處理空間資訊，在 2D 平面的情況下，我們知道機器已經可以將細節捕捉的很好，分辨出差異並找到規律，但若是現在傳入的 Input 不再是 2D 的平面而是 3D 的空間呢？這又會是個不同的方向，我們有別於傳統而改採用具有深度的資訊，嘗試在加入深度的狀況下也能讓機器有個好的學習結果。

我們將深入了解 FaceNet 及 PointNet 的運作原理，知道為何 FaceNet 及 PointNet 在自身的領域能有非常傑出的表現，嘗試結合並妥善運用這兩個神經網路的優點，利用 FaceNet 對於臉部的精準辨識，以及 PointNet 對於 3D 空間資訊的整合分析能力，讓我們能透過這深度學習的技術，藉由機器人的雙眼----具有深度的攝像機，去對人類在室內空間中的活動有所了解。在危險即將發生或人類出現不預期的行動時，可以主動即時發出警告，減少遺憾的發生，亦或者是可以在被動的情況下，依照人類的指示來達到要求，未來更可以由得到的資料來學習、建立模型來提供進一步的服務。

中華民國 107 年 5 月

目錄

一、專題研究動機與目的

此專題研究目的在於模擬機器人視覺，結合空間偵測與人臉辨識技術，透過具有深度的攝像機動態取得的資訊，對房間中的人做辨識，並且知道人在環境中的位置。透過深度學習的技術，實作出一個能同時判斷人物所在位置的人臉辨識系統。

此系統可能應用的方向：

(一) 辨識技術完備後與動機系老師合作，結合機器手臂做機器人自動取物系統。

(二) 爲有幼兒的家庭做小孩靠近危險物品的警示。

二、現有相關研究概況及比較

近年來，電腦視覺蓬勃發展，其中一個重要的分水嶺為深度學習的應用。2010 年，ImageNet 舉辦的大規模視覺辨識競賽之中，表現最好的團隊只有 72% 的準確率。到了 2012 年，來自加拿大多倫多大學的團隊採用深度學習法，把準確率提升到 84%，是一個重大的突破，從此扭轉電腦視覺領域的研究方向。到了 2015 年，前幾名參賽團隊都採用了此方法，也成功將準確率有效的提高到 94%以上。另一方面，如果要將此技術應用在機器人身上，就必須要克服龐大的運算量，以及有效提升運算速度，才能達到 Real Time 的可能性，進而為滿足人類生活所需。相關應用包括了自動車對於周遭環境的勘查、影音資料的搜尋與分析、人臉辨識系統等等。

三、設計原理

(一)PointNet

1.原理

相對於 CNN 以 pixel 的 rgb 當作 input，PointNet 以多個點在 3D 空間中的 xyz 座標當作 input，也就是 point cloud。Point cloud 可以用 LiDAR 或有深度的感測器照射實際的物體或場景來產生，有實現 real-time 辨識的潛力。或是可以透過建模再轉換成 point cloud 的格式。

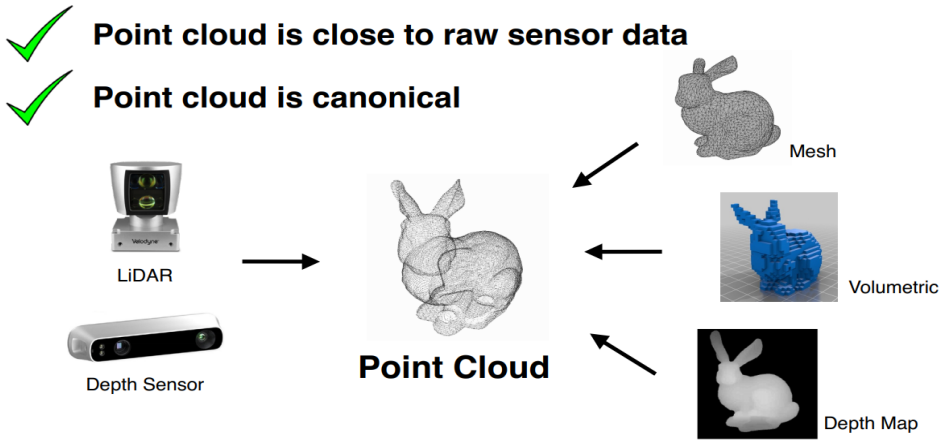

圖 1. 產生 Point Cloud 的方法

PointNet 本身有三種應用，第一種是給一個物體的 point cloud 來辨別他是甚麼物體；第二種是給一個物體的 point cloud，對它作 segmentaion(也就是辨別每個 point cloud 中的點分別是甚麼)來將這個物體切成更小的部件；第三種是給一整個房間的 point cloud，對它作 segmentaion 來將房間中的各個物體辨識出來。

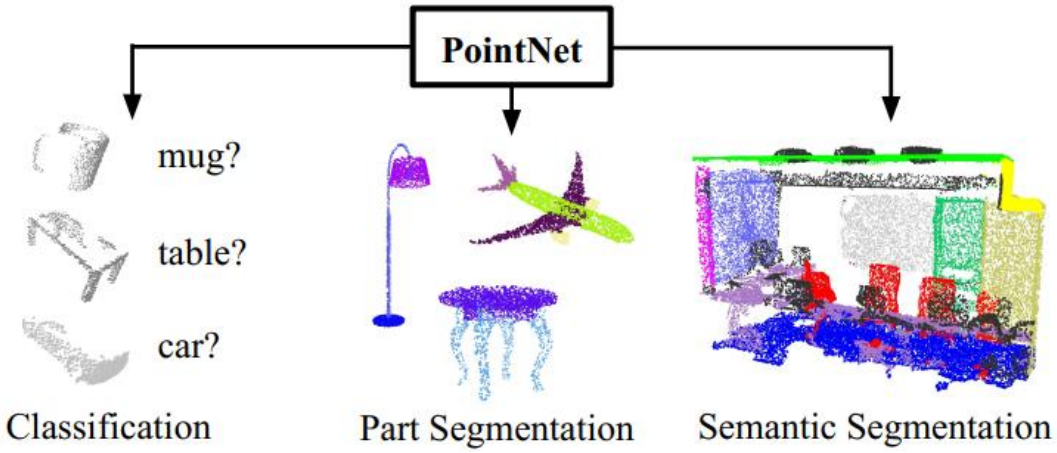

圖 2. PointNet 基本的三種應用

Joint Alignment Network：
同一種物體的 point cloud 可以是各種不同角度和長寬，所以 train 出一個 3*3 的矩陣，叫 T-Net，並將 input 乘上這個矩陣，就可以達到對一些 geometric transformation 的不變性，也就是同一種物體的不同角度和長寬的 point cloud 乘上這個矩陣以後達到相似的值，這部分叫 input transform。而經過 mlp(multi-layer perceptron，也就是多層 fully-connected layer)以後的 feature 也可以用同樣的方法作 feature transformation。

Symmetry Function for Unordered Input：
point cloud 沒有順序性，因此想要達到對 n 個點的 n!排列的不變性，也就是不管 point cloud 各點怎麼排列都能達到同樣的結果。所以在 classification network 後半段對 n*1024 做 max pool，也就是每個點有 1024 維度的 feature，而在每個維度都對 n 點取最大的值，如此就能達到對 n!排列的不變性，得到 1024 維度的 global feature，再經過 mlp 為 k 維度的 output scores，其中最大的值的 index 就是 prediction，如此就能對 k 種物體做 classification。

Local and Global Information Aggregation：
要達到 segmentation，需要這個點的 local feature 和整體的 global feature，因此將原本 n*64 矩陣的每一點後面 concatenate 1024 維度的 global feature，再經過 mlp 後達到 n*m，以對每一點辨識 m 種物體或部件。(在 part segmentation 中是辨識部件，在 segmemtic segmentation 中是辨識完整物體) 這邊的 segmentation 不像 mask-rcnn 因為 224*224 pixel 數量較多，計算量太大，需要 downsampling 再 upsampling；PointNet 中的 n 個點是 uniformly sample 1024~4096 個點而已。

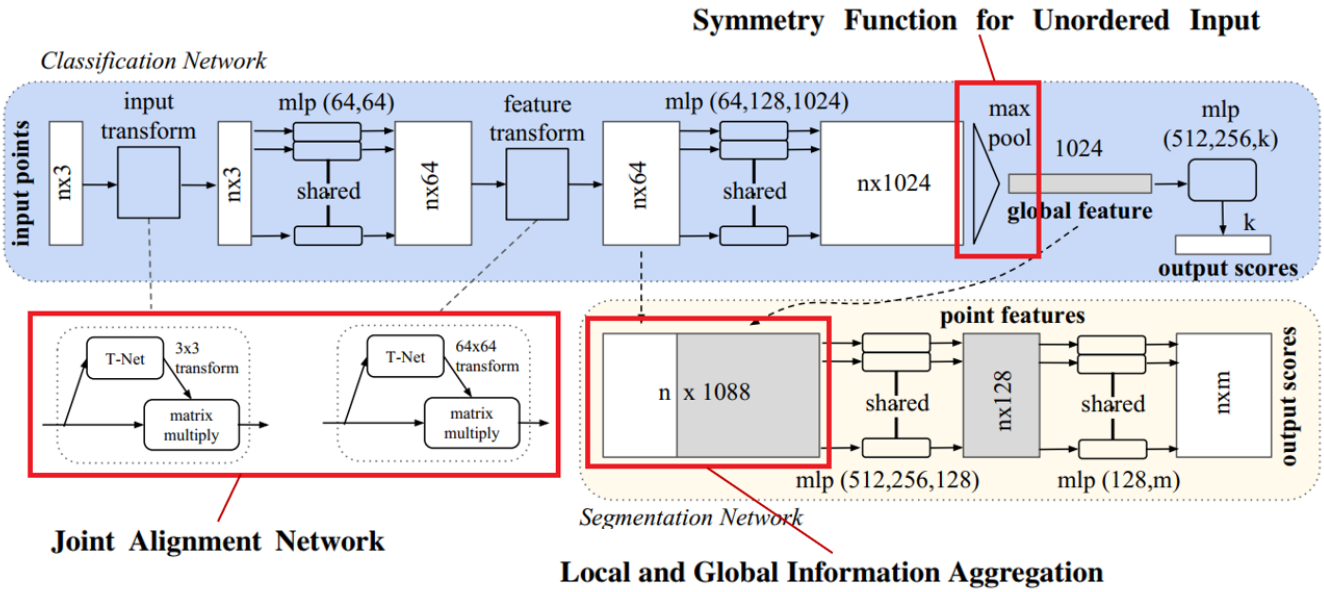

圖 3. PointNet 的架構

Robustness test：

在 robustness test 方面，可以看到相對於 VoxNet，PointNet 表現較好，在少於 7 成資料以後，準確度才開始有顯著下降。VoxNet 是將 point cloud 切成 32*32*32 的 voxel，並用 occupancy grids 算出物體在一個 voxel 中的佔有率，再經過 CNN。它 robustness 較低的原因主要可能是因為物體 rotate 以後的點的分布完全不一樣，點較少時就難以辨認；而 PointNet 則可以透過 rotate 物體，符合大致輪廓就可以辨識出來。左圖可以看到 PointNet 只要最少有 Critical Point Sets 的形狀，最多點的 Upper-bound Shapes 之間，都能成功辨識出來。

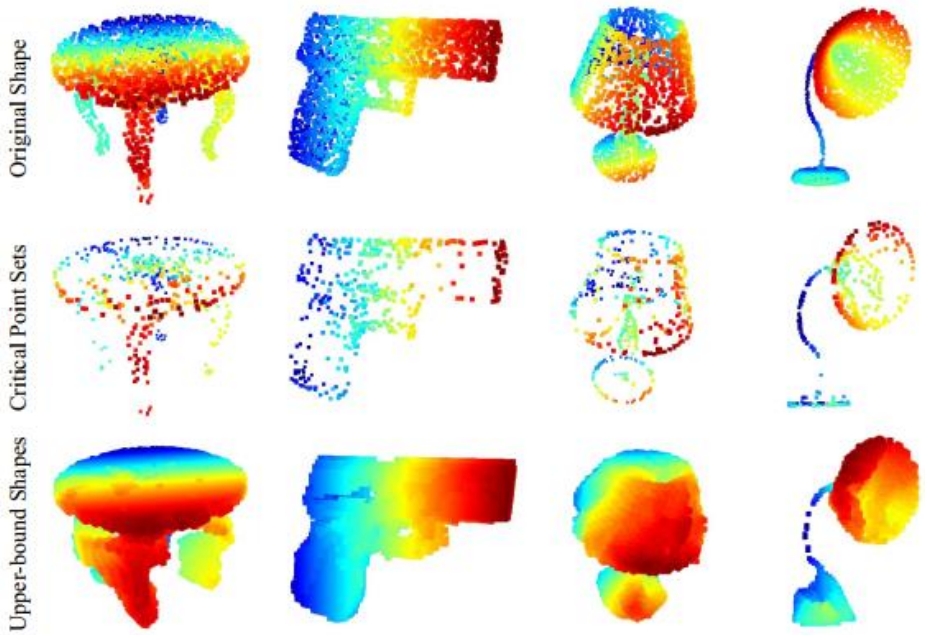

圖 4. 第一行是原物體的 Point Cloud，第二行是最少需要的點(Critical Point Sets)，第三行是最多可辨別的點(Upper-bound Shapes)。

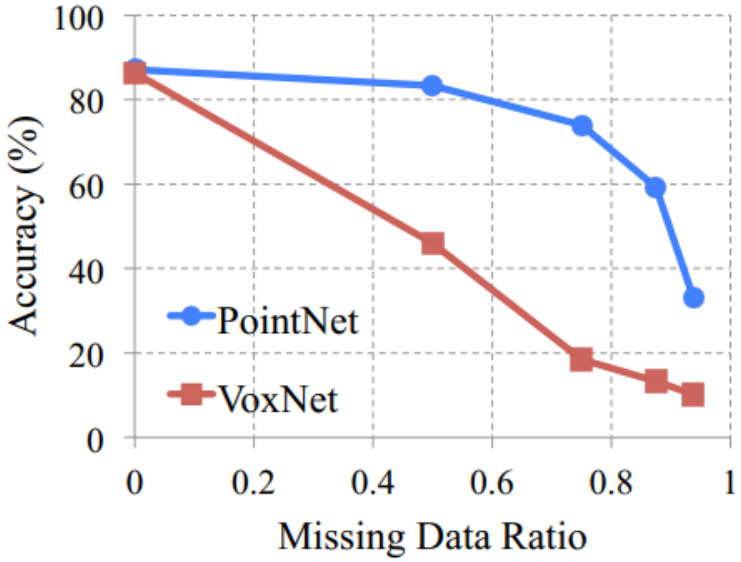

圖 5. PointNet 與 VoxNet 比較，Point Cloud 資料有缺失時的準確度表現。

2.classification

Input 讀取：

利用 farthest point sampling 在表面取樣，圖 6 中共取樣了 2048 個點數。

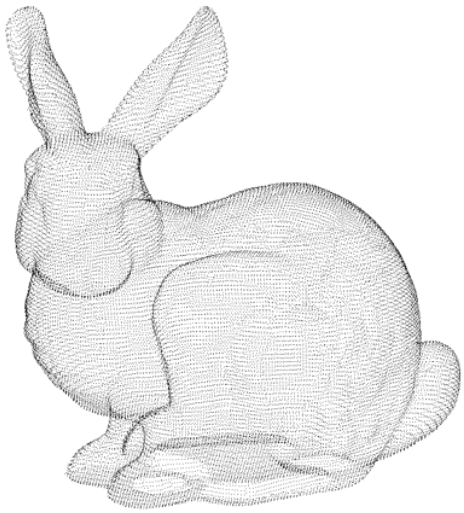

圖 6

Point Cloud models：

目前用來訓練神經網路的模型是由 ModelNet 所提供。在這個 dataset 之中，共有 40 種不同 class 的物件，也就是說，在神經網路訓練完畢之後，將具備分辨這四十種物件的能力。

Preparing my own data set：

- Step1：透過具有深度資訊的硬體設備捕捉物件，得到一個 3D modle。
- Step2：將此 3D modle 轉換為 PLY（立體空間中的點座標）的格式。
- Step3：選取固定數量的點座標（ex:2048），並將其寫到 HDF5 file 中。

Visualization：

將這些點座標透過軟體轉換為可以用肉眼辨識的圖案，可以協助我們在 training 的過程中找到問題可能的原因，範例如圖 7、圖 8。

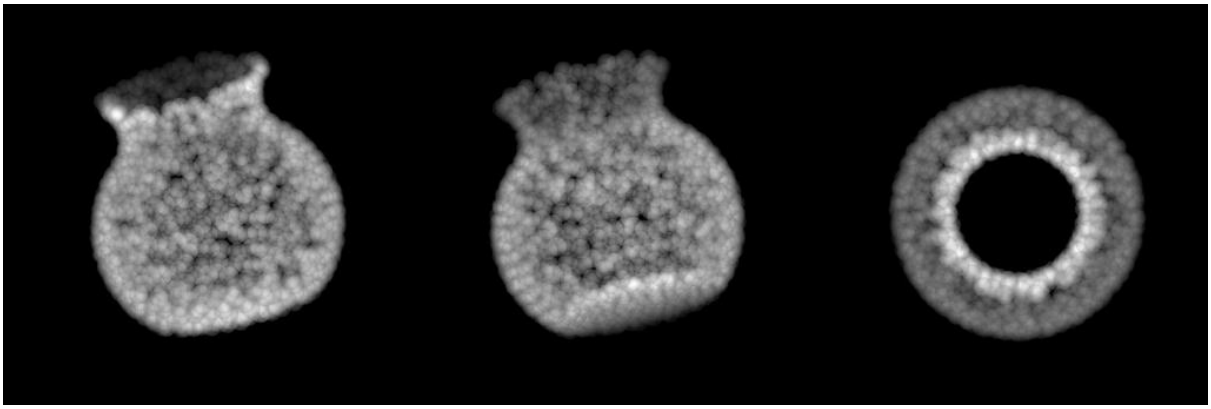

圖 7. 花瓶

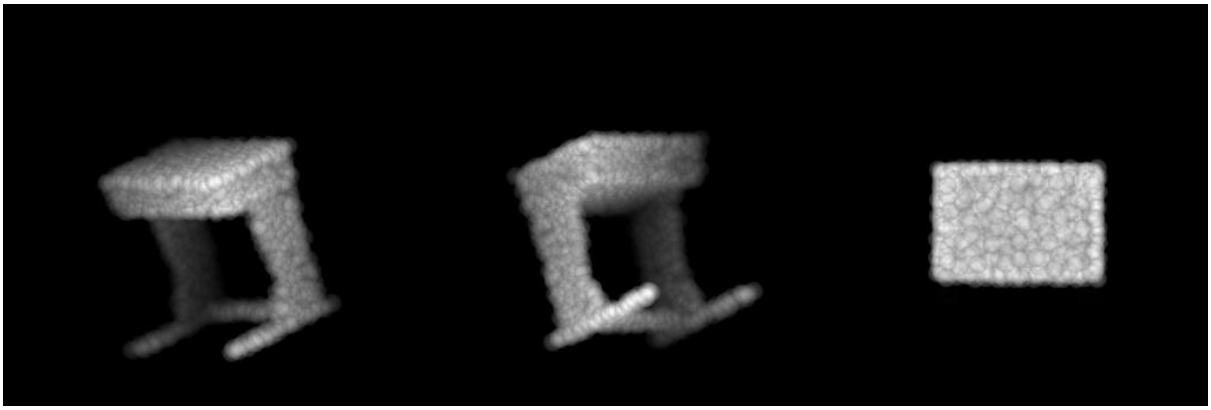

圖 8. 課桌椅

Evaluation：

在訓練完畢整個神經網路之後，針對以上 40 個項目做測試，分析其準確性，如圖 9。

其中以紅色框框標記的花盆項目比較特別，將上述項目經由 visualization 後會發現，其實辨識錯誤的原因，可能是當初 label 為花盆的圖片定義不明確，導致網路在學習的過程中有誤解產生。圖 10、圖 11 是辨識錯誤為其他項目的花盆。

我們可以發現，因為當初在訓練花盆的圖片中有出現植物，導致神經網路有所誤會，這也是我們在之後的實作要避免的問題。

```
eval mean loss: 0.596086
eval accuracy: 0.881280
eval avg class acc: 0.857355
  airplane:     0.990
   bathtub:     0.860
       bed:     0.980
     bench:     0.700
 bookshelf:     0.900
    bottle:     0.950
      bowl:     0.950
       car:     0.990
     chair:     0.970
      cone:     0.950
       cup:     0.700
   curtain:     0.850
      desk:     0.814
      door:     0.950
   dresser:     0.698
flower_pot:     0.200
 glass_box:     0.960
    guitar:     1.000
  keyboard:     1.000
      lamp:     0.950
    laptop:     1.000
    mantel:     0.950
   monitor:     0.940
night_stand:    0.733
    person:     0.900
     piano:     0.850
     plant:     0.770
     radio:     0.750
range_hood:     0.900
      sink:     0.700
      sofa:     0.970
    stairs:     0.900
     stool:     0.900
     table:     0.810
      tent:     0.950
    toilet:     0.980
  tv_stand:     0.770
      vase:     0.760
  wardrobe:     0.550
      xbox:     0.850
```

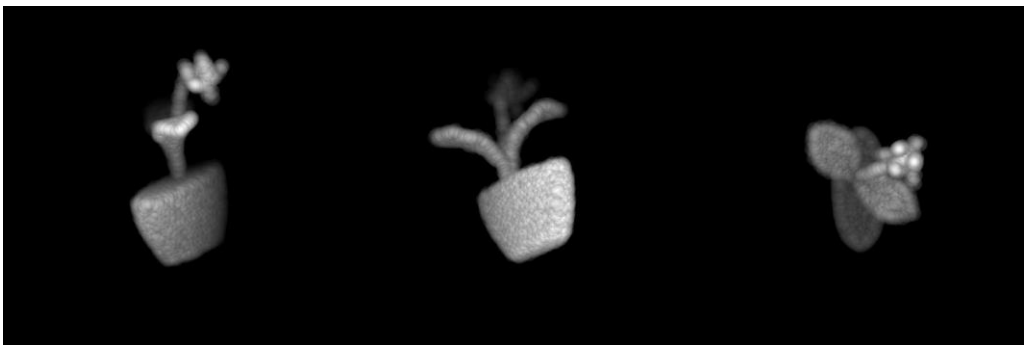

圖 10. 上面 label 為花盆的圖片被誤認為植物。

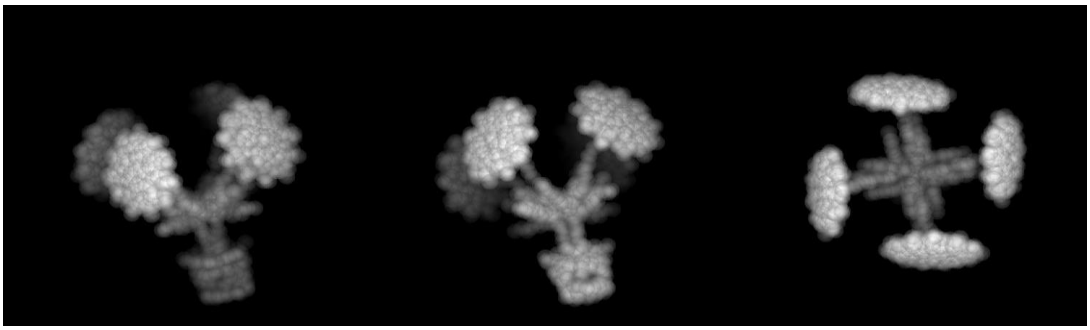

圖 11. 上面 label 為花盆的圖片被誤認為植物。

圖 9

3.Part segmentation

What is Part Segmentation？

給定一個物體 3D 掃描後的模型，part segmentation 的任務是指出每個點是該物體的某個部分，如：桌腳、杯子的把手等。

Input ：

對一個物體取樣 2048 個點，包含 x,y,z 的座標，故一個物體需 2048x3 這麼多資訊。

Accuracy：

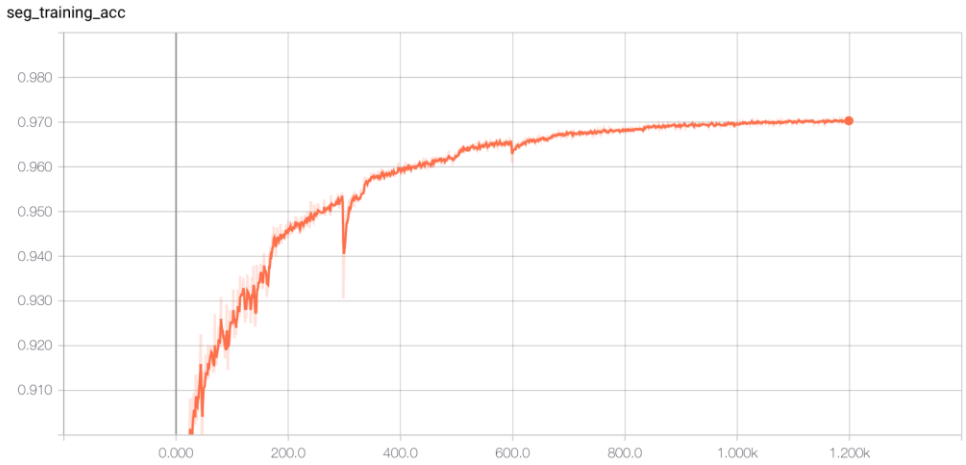

圖 12. 訓練模型時的準確度變化，橫軸單位為步數(step)，縱軸為精準度(%)。

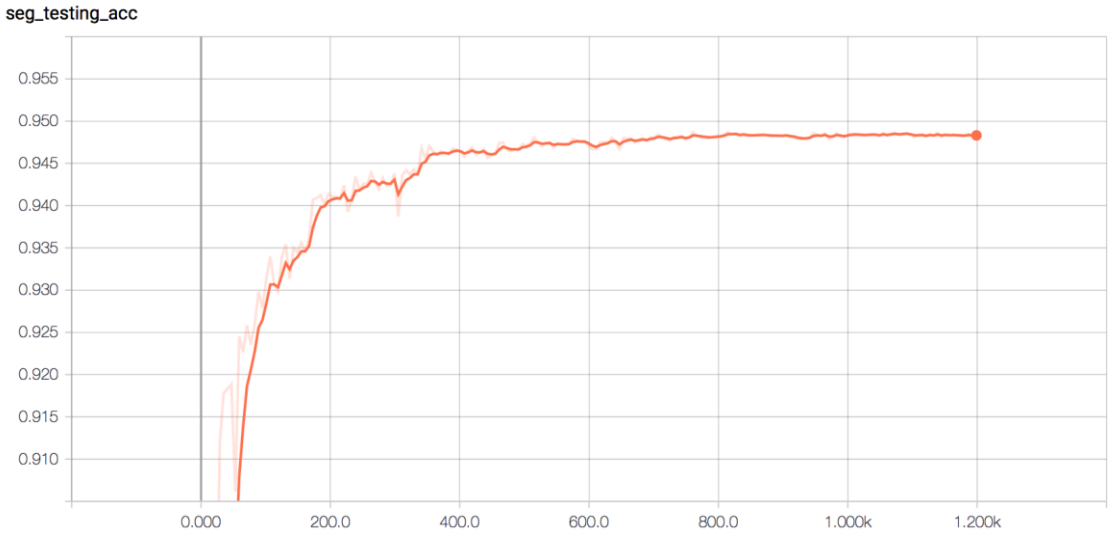

圖 13. 測試時的準確度變化，橫軸單位為步數(step)，縱軸為精準度(%)。

表 1

	mean	aero	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate board	table
# shapes		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
Wu [27]	-	63.2	-	-	-	73.5	-	-	-	74.4	-	-	-	-	-	-	74.8
Yi [29]	81.4	81.0	78.4	77.7	**75.7**	87.6	61.9	**92.0**	85.4	**82.5**	**95.7**	**70.6**	91.9	**85.9**	53.1	69.8	75.3
3DCNN	79.4	75.1	72.8	73.3	70.0	87.2	63.5	88.4	79.6	74.4	93.9	58.7	91.8	76.4	51.2	65.3	77.1
Ours	**83.7**	**83.4**	**78.7**	**82.5**	74.9	**89.6**	**73.0**	91.5	**85.9**	80.8	95.3	65.2	**93.0**	81.2	**57.9**	**72.8**	**80.6**

使用 shapeNet part dataset 測試的結果，量測方式為 mIoU(%)，3DCNN 為論文作者自己提供。

關於表一中 Wu[27]使用的方法大致如下：電腦將原始圖過度分割成數個小塊，用戶需要對那些小塊進行標記，而標記的結果會傳播給未標記塊，最後得到人與電腦共同標記的分割結果，人類可以一直對小塊進行標記直到滿意。
關於表一中 Yi[29]使用的方法大致如下：給定一個輸入照片，使用他們設計的介面來輸入一些標記，將這些標記自動傳播給其他未標記的形狀，然後詢問人類最符合結果的標記，由此得知預測的準確度並進行改善。

IoU(intersection over union)：

為預測結果與實際結果的交集面積除以聯集面積。若聯集面積為 A，交集面積為 B，則 IoU = A/B x100%，A、B 如圖 14、圖 15 所示：

A：　　B：

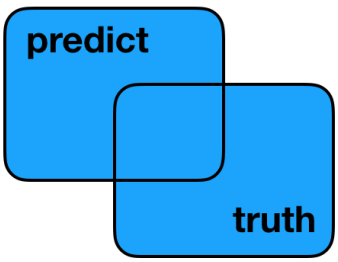

圖 14

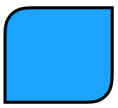

圖 15

mIoU(mean interscetion over union)：

為將所有種類的 IoU 結果相加，除以種類的個數。

一個預測結果與真實結果範例：

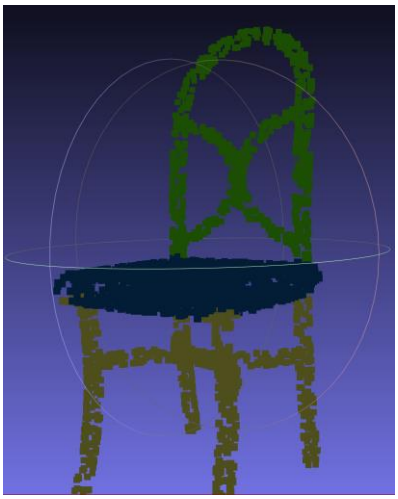

圖 16

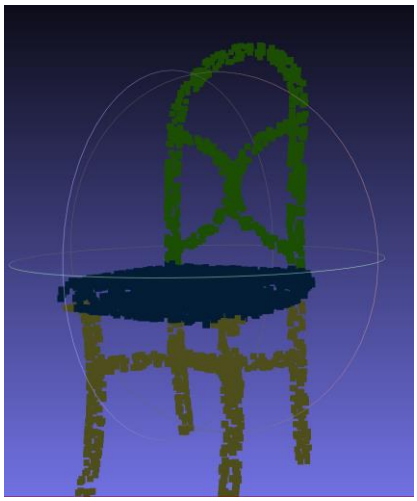

圖 17

圖 16 為真實結果，圖 17 為預測結果。

4.semantic segmentation 描述

輸入格式和 preprocess：

- 對整個房間做 segmentation，給予 point cloud 中每一點一個 label。
- 將整個房間切成多個 1 公尺立方體。

- 一次將一個立方體丟進 PointNet 中，每一個立方體中 uniformly sample 4096 個點。
- 每一點的格式為 xyzrgbXYZ。(batch, 4096, 9)。
- xyz 座標都是正數，z 為高度，單位是公尺。
- rgb normalize 為 0 到 1。
- XYZ 是將 xyz 分別 normalize 為 0 到 1。這個 XYZ 的作用主要是辨別屋頂，地板等需要知道在房間中特定位置的物體。

Evaluate time & visualization：
處理一個立方體大約要 100ms，而處理完一個 30 立方公尺的房間要 **3 秒**左右，所以將來放進機器人做 real-time 處理時，降低計算量和壓縮 PointNet 是一大挑戰。

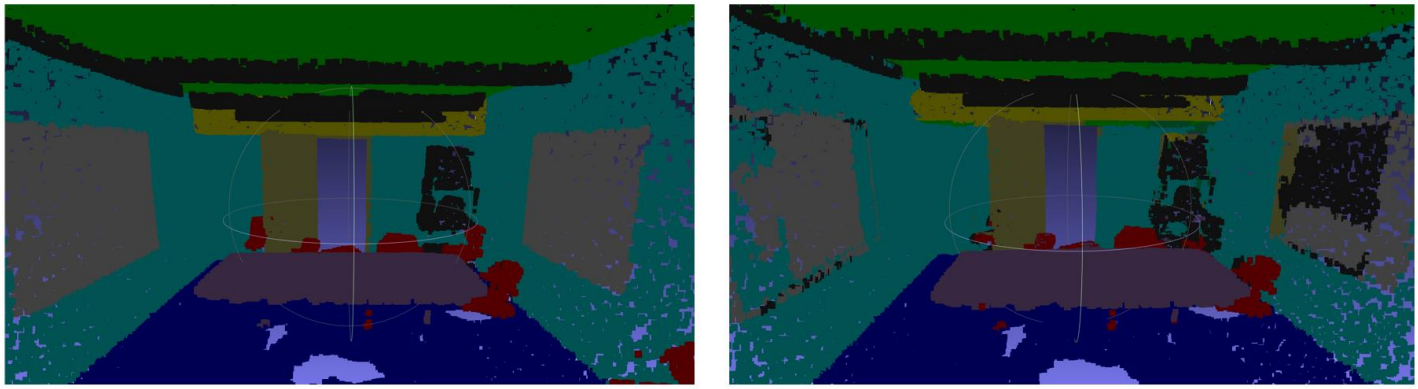

圖 18. sementic segmentation 的 visualization：一個 30 立方公尺的會議室的 ground truth 和 PointNet 做出的 prediction。prediction 的右側窗戶有較明顯的分辨錯誤，誤認為黑色的雜物和深黃色的門。

Object Detection：
利用 semantic segmentation 的結果，可以用 BFS 將鄰近的相同 label 的點框出整個物體。太少的點會被忽略，桌子和椅子等物體的 bounding box 則會被延伸至地板。
為了解決有連成一排的椅子無法個別被區分開來，對這種物體 train 出 binary classification，並用 sliding shape 在 3D 空間中舉出多個可能的框，再用該 binary classification 判斷哪個框中較可能是一個椅子，重複面積過高的框則用 non-maximum suppression(對 IOU 超過一定值的幾個框合併為一個框)去除。

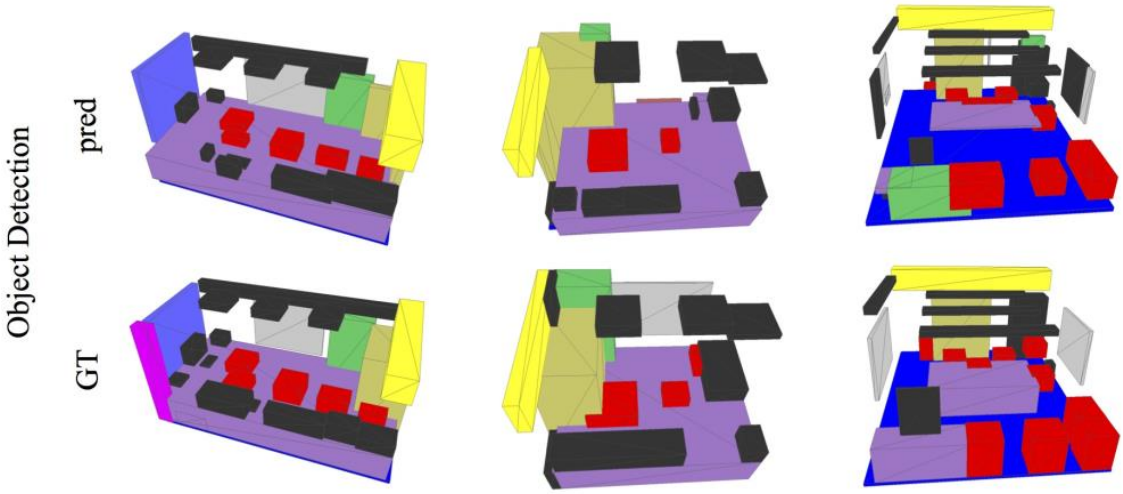

圖 19. semantic segmentation 的延伸：Object Detection 在 3D 空間中框出各個單一物體

Accuracy：

這邊做為比較的 baseline 是用類似 VoxNet 的 3D CNN 的做法，將 point cloud 切成 32*32*32 的 voxel，並用 occupancy grids 算出物體在一個 voxel 中的佔有率，再經過 CNN。下圖可見 PointNet 大多都有較好的 accuracy。

	mean IoU	overall accuracy
Ours baseline	20.12	53.19
Ours PointNet	**47.71**	**78.62**

Table 3. **Results on semantic segmentation in scenes.** Metric is average IoU over 13 classes (structural and furniture elements plus clutter) and classification accuracy calculated on points.

	table	chair	sofa	board	mean
# instance	455	1363	55	137	
Armeni et al. [1]	46.02	16.15	**6.78**	3.91	18.22
Ours	**46.67**	**33.80**	4.76	**11.72**	**24.24**

Table 4. **Results on 3D object detection in scenes.** Metric is average precision with threshold IoU 0.5 computed in 3D volumes.

圖 20. semantic segmentation 和 Object Detection 的準確率

(二)FaceNet

1.簡介

FaceNet 是一個人臉辨識深度學習網路架構，將人臉特徵映射至 128 維空間去比較座標之間的距離，藉由我們所設定的 threshold，可以將與參考點間小於 threshold 的距離的特徵圖座標視為同一個人達到辨認的效果，根據這樣的去做到 face verification、face recognition 與 clustering 的功能。

圖 21. 我們先將 deep architecture 視為一個黑盒子，在後面去另行做介紹，batch 批次為 input 丟入 deep architecture，而這個 deep architecture 是沒有用 softmax loss，經過 L2 標準化，embedding 到 128 維空間做特徵表示，計算 triplet loss。

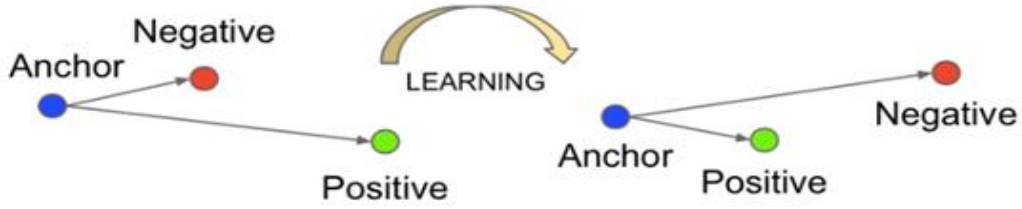

圖 22. triplet 是一個三元組，從訓練樣本集中隨機取出一個樣本定為 anchor，然後從訓練樣本中取出和 anchor 同類者為 positive，不同類者為 negative，以此構成三元組 (anchor, positive, negative)。我們希望透過機器學習，在 128 維的空間內，盡可能讓每個 positive 和 anchor 間的距離小於每個 anchor 和 negative 間的距離。

2.triplet loss

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 , \tag{1}$$

$$\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \mathcal{T} . \tag{2}$$

我們希望(anchor, positive, negative)之間具有上述的關係，Alpha 為我們定義的值，代表負樣本對和正樣本對之間要有的最小距離。T 代表在訓練集中可能存在的三元組。

$$\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$

我們的損失函數可以用這個數學式去表示，若[ ]中值大於 0 則取其值為損失，若是[ ]中值小於 0 則損失為 0。

3.triplet selection

在做 triplet 的選擇時，我們會希望三元組能對訓練有幫助又能快速收斂，但符合(1)式的 triplet 其實很多，其中蠻多 triplet 對訓練效果有限且會減緩收斂時間，因此我們要選擇最容易違反(1)式的 triplet。

$$\text{argmax}_{x_i^p} \|f(x_i^a) - f(x_i^p)\|_2^2$$

$$\text{argmin}_{x_i^n} \|f(x_i^a) - f(x_i^n)\|_2^2.$$

理想化的狀況，我們希望找出 hard-positive 和 hard-negative，即有最大距離的正樣本對和最小距離的負樣本對，但從整個訓練集中計算 argmax 和 argmin 是不可行的。

FaceNet 採用在線生成的方式，從 mini-batch 中找出 hard positive/negative。但在訓練中取用 hard negative 卻會導致很容易找到 local minimum，因此對(1)式做修改得到

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2$$

負樣本對的區間在修正後就變比較大，會有更低機率陷入 local minimum。

4.deep architecture network

可以採用不同的結構，如 ZF-NET(NN1 如表 2)、Inception 模型(NN2 如表 3、NNS1、NNS2)等。根據實際情況會有不同的 architecture，比如在手機上運行，因為有記憶體與運算量的限制，準確率高但計算量大的 NN1、NN2 可能就沒那麼適用，NNS1、NNS2 亦或是最近新的 mobileNet 這種降低準確率換取對記憶體和運算量更低需求的 architecture 可能就更適合。

表 2

layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

NN1，ZF-NET 為主體架構，參數量多，FLOPS 也大。

表 3

type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L_2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L_2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L_2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L_2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L_2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L_2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

NN2，以 Google 推出的 inception 模型為主體架構，inception 為模塊化設計，以此衍生出多種變型(NNS1、NNS2)，參數量降低，但負擔還是很重。

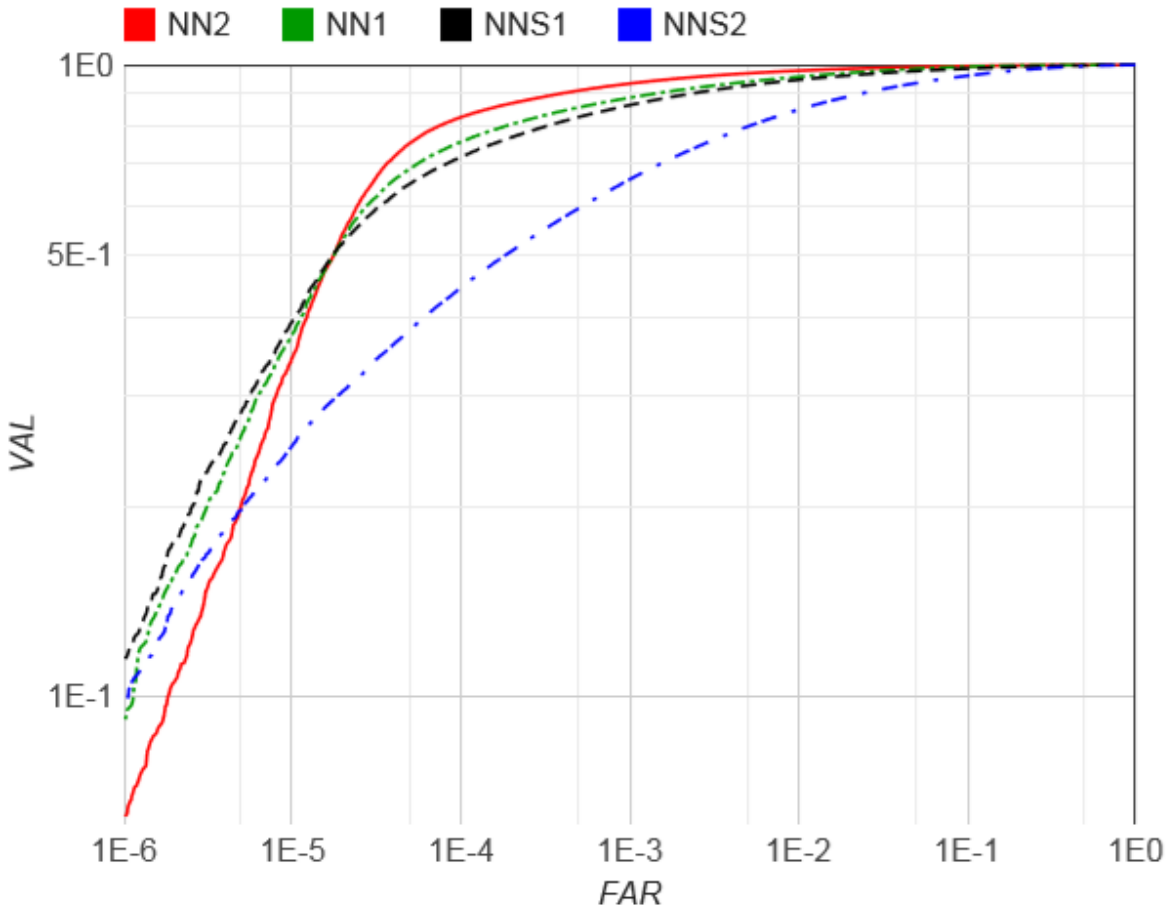

圖 23. NN1、NN2、NNS1、NNS2 準確率比較

四、預估研究方法與步驟(應用)

(一)機器人記錄各個房間的資訊

首先，Pointnet 的 sementic segmentation 可以使機器人掌握這個房間裡各種物體和設施的位置。機器人可以把走過的房間的資訊記錄下來，並定期用深度攝影機更新物體位置。遇到新的房間或情況時，可以先花時間把整個房間的資訊記錄下來，或是只根據目前看到的部分進行 segmentation。

(二)機器人將物體在 3D 空間中框出來並辨識 - Frustum PointNet

Frustum PointNet 會需要一般攝影機的 rgb 資訊和深度攝影機的 rgbd 資訊，因為一般攝影機的 rgb 資訊解析度較高，bounding box 會比較準確。先用 rcnn 的方法將平面的物體的 bounding box 框出來，再用 projection matrix 得到沿著這個方向的 bouding box 在 3D 空間中 near 和 far 之間的整個 frustum。接著對這個 frustum 裡面的點做 segmentation，標出目標物體的那幾個點，並算出那些點的中心。但因為攝影機照到的點都是該物體靠近攝影機那一面的點，實際物體的中心和攝影機照到的點的中心差距很大，因此用 T-Net 找到實際物體的中心，再對 3D box 進行預測。

(三)機器人辨識有深度的人臉 - 結合 PointNet 與 FaceNet

若是要辨識有深度的人臉，則可以用這個 Frustum PointNet 的 bounding box 框出人臉，接著得到預測的 3D box 後，進行 PointNet 原本的 classification，若確定是人臉，使用 FaceNet 的方法，出來的 feature 轉成 128 維度以後，放置在一個 128 維度的歐幾里得空間中，使用 Triplet Loss 作為其 loss function，判斷空間中距離較近的為同一人的可能性較高，較遠則可能性較低，並定一個 threshold 作為判斷依據，於距離小於 threshold 則判斷為同一人，最後使用 SVC 做 classification。

$$L = \arg\min \sum_i \left(\|x_i^a - x_i^p\|_2^2 - \|x_i^a - x_i^n\|_2^2 + threshold \right)_+$$

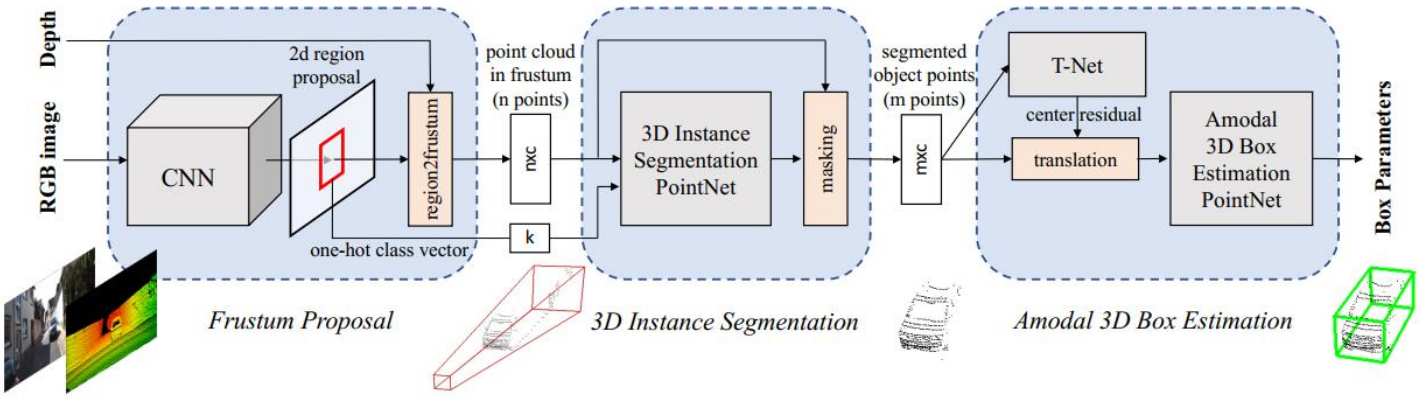

圖 23. Frustum PointNet 的架構

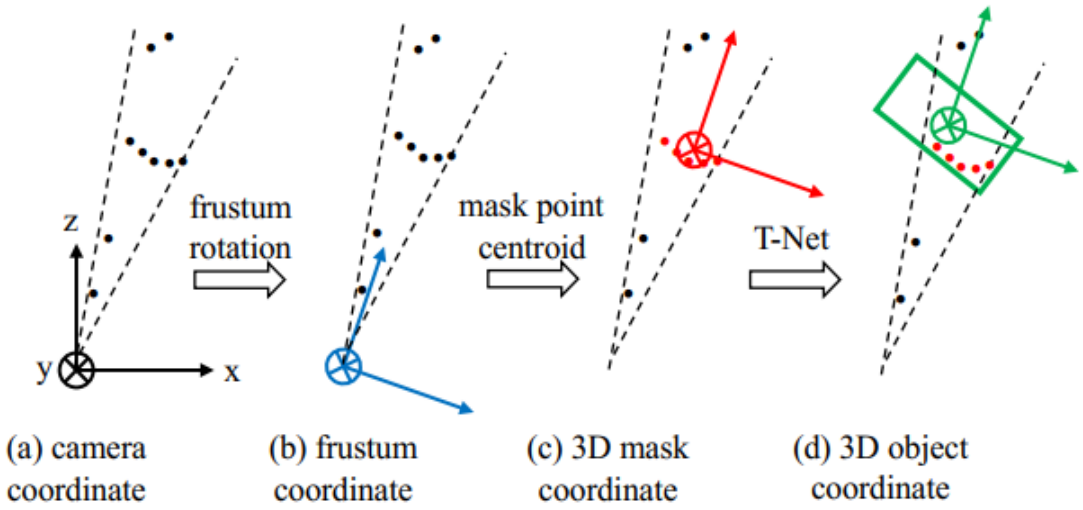

圖 24. 找到物體中心並框出物體的過程

五、團隊合作方式

我們的團隊分成兩個小組，分別對 PointNet 與 FaceNet 進行研究，了解其架構後，整合兩個模型的特點，對架構進行修改，以達到我們的研究目標。

六、初步成果(與接下來的計畫)

利用 PointNet 我們可以把場景分析、對物體分門別類及切出物體的輪廓，想要的話可以更進一步，取得每個物體的部位，例如：椅子的腳、坐墊與靠背。

利用 FaceNet 我們可以在給定兩張人臉的狀況下，去判斷兩個人是否相同。

未來的研究方向：

(一)將兩個 network 整合：期望能在 PointNet 分辨出是人這個類別之後，進一步使用 FaceNet 來分辨出是家中的哪個成員。

目前 PointNet 僅能分辨出物體，比如：菜刀，不過它並不知道菜刀可以切水果，經由我們的的裝置，希望可以在進一步提升解析度之後（解析度足夠的情況下，應該可以分辨出人們的各個關節、部位的活動情況），在匿名的情況下，取得人們如何使用各種物體的資訊，構造一個 network 使得它可以學會使用人們早就已經習以為常的各種物品，往更聰明的機器人邁進。

(二)model compression：目前無論是 PointNet 或是 Facenet，要在 embedded system 上運行都有其難度，也無法達到真正 real time 進行辨識，因此我們將對於如何降低 model 的 computing power 以及增加運算速度進行研究。

七、參考文獻

1.gitHub 程式碼與說明：
https://github.com/charlesq34/pointnet
2. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation
http://stanford.edu/~rqi/pointnet/
3. PointNet: Presentation video
https://www.youtube.com/watch?v=Cge-hot0Oc0
4. PointNet: Slide
http://stanford.edu/~rqi/pointnet/docs/cvpr17_pointnet_slides.pdf
5. ModelNet
http://modelnet.cs.princeton.edu/
6. PointNet 論文參考
https://arxiv.org/pdf/1612.00593.pdf
7. Frustum PointNets for 3D Object Detection from RGB-D Data
https://arxiv.org/abs/1711.08488
8. Florian Schroff, Dmitry Kalenichenko and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. on *CVP*R 2015.6
9. github 程式碼參考：https://github.com/davidsandberg/facenet