智慧感知與決策

311512064鄧書桓

**Task1:**

1. **Code:**
2. Tranning segmentation model

Floor1:

Model of **other scene**:

DATASET:

  imgMaxSize: 1000

  imgSizes: (300, 375, 450, 525, 600)

  list\_train: ./data/others\_training.odgt

  list\_val: ./data/data\_for\_reconstruct.odgt

  num\_class: 101

  padding\_constant: 8

  random\_flip: True

  root\_dataset: ""

  segm\_downsampling\_rate: 8

DIR: ckpt/others

MODEL:

  arch\_decoder: ppm\_deepsup

  arch\_encoder: resnet50dilated

  fc\_dim: 2048

  weights\_decoder:

  weights\_encoder:

TEST:

  batch\_size: 1

  checkpoint: epoch\_20.pth

  result: ./

TRAIN:

  batch\_size\_per\_gpu: 2

  beta1: 0.9

  deep\_sup\_scale: 0.4

  disp\_iter: 20

  epoch\_iters: 650

  fix\_bn: False

  lr\_decoder: 0.02

  lr\_encoder: 0.02

  lr\_pow: 0.9

  num\_epoch: 20

  optim: SGD

  seed: 304

  start\_epoch: 0

  weight\_decay: 0.0001

  workers: 16

VAL:

  batch\_size: 1

  checkpoint: epoch\_20.pth

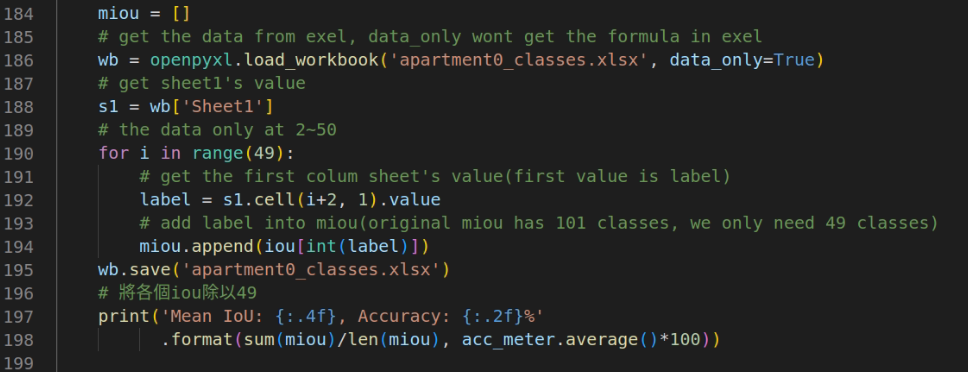
  visualize: True

我這邊使用ade20k-resnet50dilated-ppm\_deepsup 作為other scenes訓練的model，dataset部分使用1300張照片，其中平均分配在其餘房間(總共10個個130張)，這邊嘗試過使用較少的房間，但得出來的結果很差。另外，visualize需要更改為True才能將照片存下。

使用的參數如上，主要調整的是batch\_size為2， epoch\_iter為650，這邊是根據相乘為dataset大小而選定的，其餘的超參數越調效果越差，故最後用一開始給定的。

MIOU:

這邊的設計為先讀取excel文件內的特定資料，並根據給定的49個calss(存在iou list中)去做資料讀取，並將讀取到的資料寫入miou，最後再將miou除以lenth(miou)已得到想要的答案。詳細的做法在下面的code中表示:



Validation result:

一張含有 文字 的圖片

自動產生的描述

Model of **apartment0**:

DATASET:

  imgMaxSize: 1000

  imgSizes: (300, 375, 450, 525, 600)

  list\_train: ./data/custom\_training.odgt

  list\_val: ./data/data\_for\_reconstruct.odgt

  num\_class: 101

  padding\_constant: 8

  random\_flip: True

  root\_dataset: ""

  segm\_downsampling\_rate: 8

DIR: ckpt/apartment0

MODEL:

  arch\_decoder: ppm\_deepsup

  arch\_encoder: resnet50dilated

  fc\_dim: 2048

  weights\_decoder:

  weights\_encoder:

TEST:

  batch\_size: 1

  checkpoint: epoch\_20.pth

  result: ./

TRAIN:

  batch\_size\_per\_gpu: 2

  beta1: 0.9

  deep\_sup\_scale: 0.4

  disp\_iter: 20

  epoch\_iters: 650

  fix\_bn: False

  lr\_decoder: 0.02

  lr\_encoder: 0.02

  lr\_pow: 0.9

  num\_epoch: 20

  optim: SGD

  seed: 304

  start\_epoch: 0

  weight\_decay: 0.0001

  workers: 16

VAL:

  batch\_size: 1

  checkpoint: epoch\_20.pth

  visualize: True

我這邊使用ade20k-resnet50dilated-ppm\_deepsup 作為apartment0訓練的model，dataset部分使用1300張照片，目的為和other scene數據集的數量相同，這邊嘗試過使用較少的房間，但得出來的結果很差。

使用的參數如上，主要調整的是batch\_size為2， epoch\_iter為650，這邊是根據相乘為dataset大小而選定的，其餘的超參數越調效果越差，故最後用一開始給定的。

MIOU:

此處的MIOU與上述的相同，就不再多做贅述。

Validation result:

一張含有 文字 的圖片

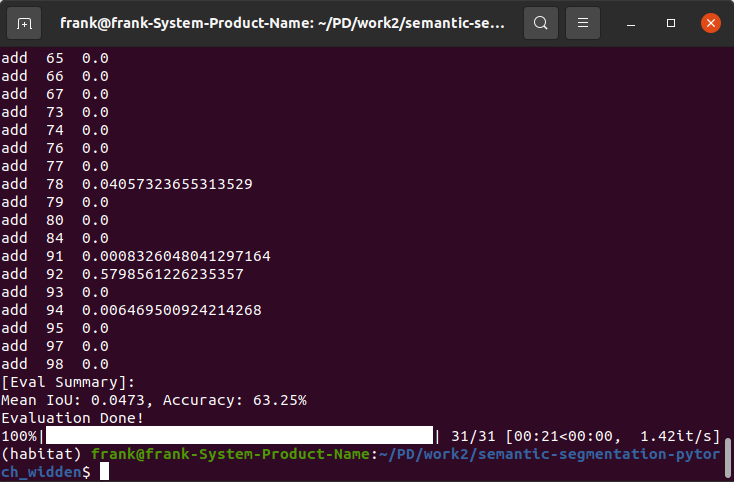
自動產生的描述

這邊得到的Accuracy與others scene相比，有著較高的準確率，我認為是因為這邊的dataset本就是由apartment0收集的，故使用相同環境下的照片經過model得到的語意分割圖的結果較佳。

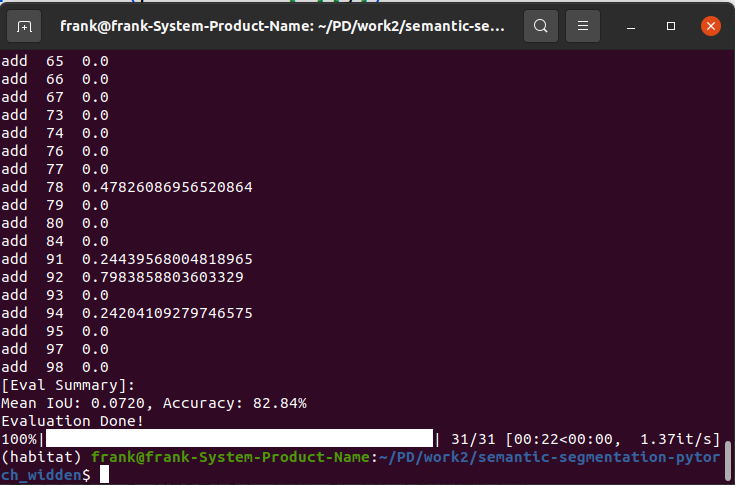
Floor2:

這邊的model與IMOU的設計都與floor1相同，這邊就不做更多的贅述。此處僅表示IMOU在floor2的效果。

IMOU of others:



IMOU of apartment0:



這邊得到的Accuracy與others scene相比，有著較高的準確率，我認為是因為這邊的dataset本就是由apartment0收集的，故使用相同環境下的照片經過model得到的語意分割圖的結果較佳。

2.3D semantic map reconstruction:

這邊設計為只要將路徑更改至所要重建的語意照片資料夾，即可自動生成點雲、自動重建與使用custom voxel down 將最後的重建點雲圖降維。

而custom voxel down的部分是我與實驗室的朋友一起討論想出來的，所以會有點相似。一開始是使用open3D內的get\_axis\_aligned\_bounding\_box()函式來獲取重建後地圖的bounding box與其參數(如質心位置與長寬高)。接著利用利用質心加減長高一半的長度來獲得bounding box位於x, y, z 軸上的起始點與終點。

接著根據voxel的大小分別在x,y,z軸上取質，想法如下:

1. 先slice出x軸上的第一個voxel，因此會得到x軸上第一個voxel且y, z軸都沒被slice的部分。
2. 再對y軸slice出第一個voxel，會得到x、y軸第一個voxel且整個z軸的部分。
3. 再對z軸slice出第一個voxel，這樣將會得到x、y、z軸上第一個voxel的正方形box。
4. 接著根據上面所得的部分去slice出該位置的color，可以得到x、y、z軸上第一個voxel中的color。
5. 之後計算此voxel中哪個color最多，並將此顏色變為這個voxel的顏色。
6. 將此顯示於此voxel的正中央(根據voxel的長寬高與質心來實現)。
7. 最後將x、y、z軸上各個voxel都執行上述步驟(使用3個for迴圈達成)就能得到custom\_voxel\_down想要的結果。

詳細的做法在下面的code中表示:

# create bounding box for custom voxel down

def custom\_voxel\_down(final\_pcd, voxel):

    # initialize box and other list

    custom\_point = []

    custom\_color = []

    num = 1

    box = final\_pcd.get\_axis\_aligned\_bounding\_box()

    box.color = (1, 1, 1)

    # get array (num\_of\_point \* 3)

    point\_all = np.asarray(final\_pcd.points)

    color\_all = np.asarray(final\_pcd.colors)

    # get center, extent of box (box is bounding final pcd)

    box\_center = box.get\_center()

    box\_margin = box.get\_extent()

    # get the start, end of x, y, z

    x\_start = round(box\_center[0] - box\_margin[0]/2, 5)

    x\_end = round(box\_center[0] + box\_margin[0]/2, 5)

    y\_start = round(box\_center[1] - box\_margin[1]/2, 5)

    y\_end = round(box\_center[1] + box\_margin[1]/2, 5)

    z\_start = round(box\_center[2] - box\_margin[2]/2, 5)

    z\_end = round(box\_center[2] + box\_margin[2]/2, 5)

    voxel\_down\_pcd = o3d.geometry.PointCloud()

    total\_iteration = int((x\_end-x\_start)/voxel) \* \

        int((y\_end-y\_start)/voxel) \* int((z\_end-z\_start)/voxel)

    for i in range(int(x\_start \* (10\*\*5)), int(x\_end \* (10\*\*5)), int(voxel \* (10\*\*5))):

        for j in range(int(y\_start \* (10\*\*5)), int(y\_end \* (10\*\*5)), int(voxel \* (10\*\*5))):

            for k in range(int(z\_start \* (10\*\*5)), int(z\_end \* (10\*\*5)), int(voxel \* (10\*\*5))):

                print("iteration for {} ........{}%:".format(

                    str(num), round(num/total\_iteration \* 100, 2)))

                num = num + 1

                # slice the box in x direction

                slice\_x = point\_all[np.where((point\_all[:, 0] >= (i/(10\*\*5))) &

                                             (point\_all[:, 0] <= (i/(10\*\*5)+voxel)))]

                # slice the remaining box in y direction

                slice\_y = slice\_x[np.where(

                    (slice\_x[:, 1] >= (j/(10\*\*5))) & (slice\_x[:, 1] <= (j/(10\*\*5)+voxel)))]

                # slice the remaining box in z direction and get the corresponding color

                slice\_z = slice\_y[np.where(

                    (slice\_y[:, 2] >= (k/(10\*\*5))) & (slice\_y[:, 2] <= (k/(10\*\*5)+voxel)))]

                # get the color with corresponding slices

                # get the color in x slice

                slice\_color = color\_all[np.where(

                    (point\_all[:, 0] >= (i/(10\*\*5))) & (point\_all[:, 0] <= (i/(10\*\*5)+voxel)))]

                # get the color in xy slice

                slice\_color = slice\_color[np.where(

                    (slice\_x[:, 1] >= (j/(10\*\*5))) & (slice\_x[:, 1] <= (j/(10\*\*5)+voxel)))]

                # get the color in xyz slice

                slice\_color = slice\_color[np.where(

                    (slice\_y[:, 2] >= (k/(10\*\*5))) & (slice\_y[:, 2] <= (k/(10\*\*5)+voxel)))]

                #計算出現次數最多的顏色,並新增到point, color

                # 若沒做四捨五入,會有太多種顏色

                slice\_color = np.around(slice\_color, 2)

                # return the numbers of each different data

                unique, counts = np.unique(

                    slice\_color, axis=0, return\_counts=True)

                if counts != []:

                    # find the max number of color and use index to get it's value

                    major\_color = unique[counts.tolist().index(max(counts))]

                    # get the center of that voxel

                    custom\_point.append(

                        [(2\*(i/(10\*\*5))+voxel)/2, (2\*(j/(10\*\*5))+voxel)/2, (2\*(k/(10\*\*5))+voxel)/2])

                    custom\_color.append(major\_color)

    voxel\_down\_pcd.points = o3d.utility.Vector3dVector(custom\_point)

    voxel\_down\_pcd.colors = o3d.utility.Vector3dVector(custom\_color)

    return voxel\_down\_pcd, box

1. **Result and Discussion**

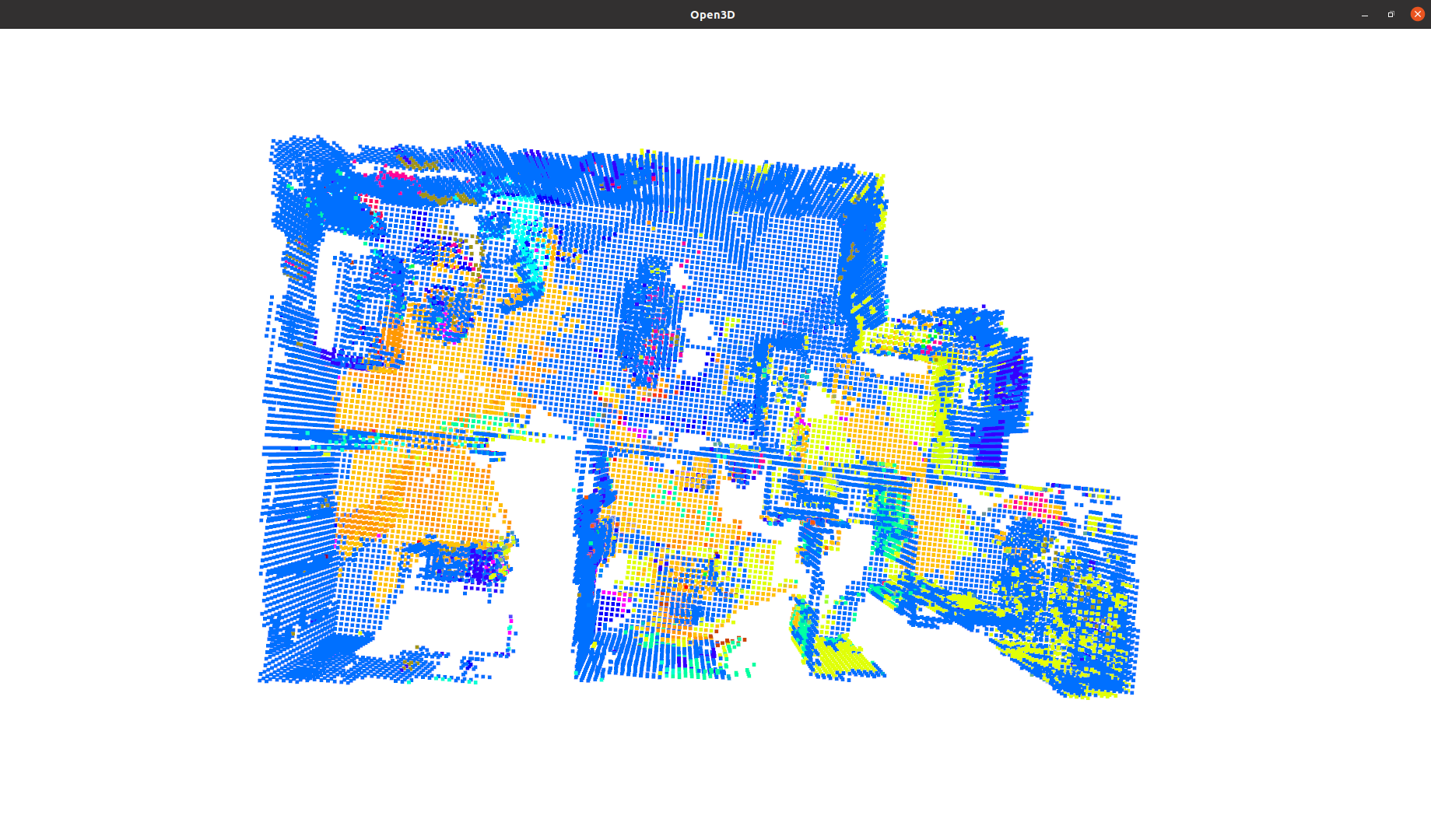
Floor1:

Train on other scene:

一張含有 文字 的圖片

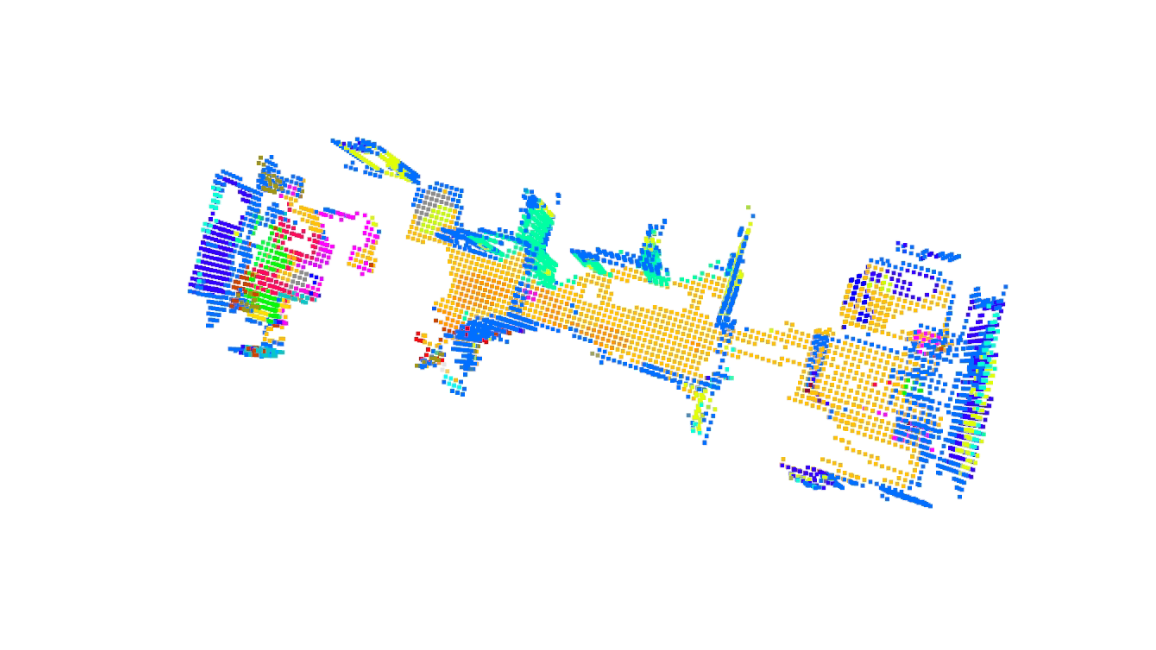
自動產生的描述

Train on apartment0:

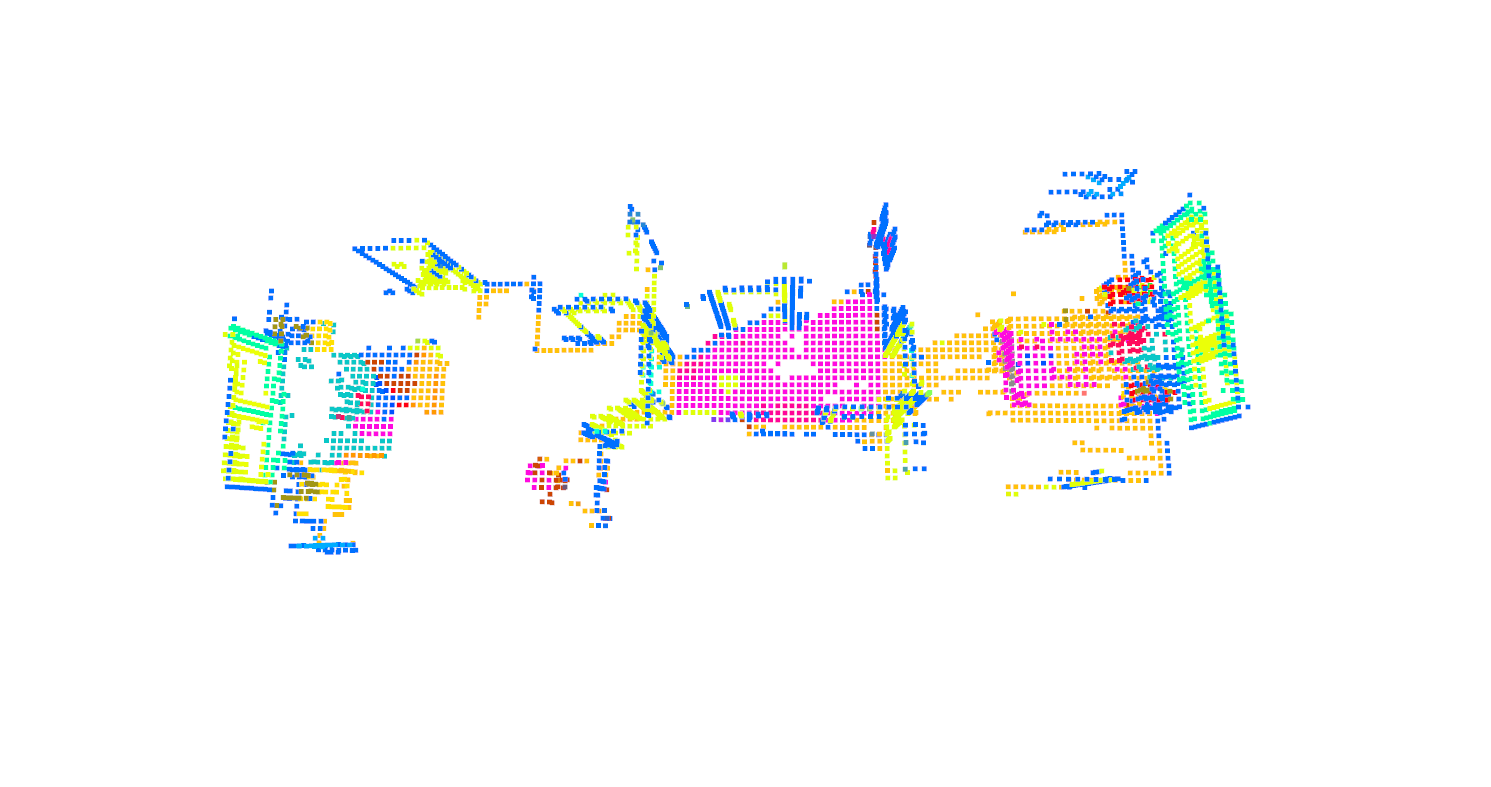


Floor2:

Train on other scene:



Train on apartment0:



雖然已經調整過許多超參數與試過多種model，但是上完色重建的效果都很差，所以只好推選這組重建效果較好的放在此報告上，我認為是因為本身的顯卡較破爛，訓練出來的成果很差且超級耗時，才會導致這樣。

另外，因為電腦的主機板年代較久遠，因此無法承受較高的custum\_voxel\_down，也就是無法使用較小的voxel去做降維，我的電腦能接受最低的voxel是0.1，也因此成果圖的顆粒感會看起來這麼重。

除此之外，由於floor2的場地較複雜，若是使用太多照片去做重建，我的電腦會不堪負荷而自動關機，因此floor2的重建效果較差，這邊僅用32張圖去做重建，希望助教見諒。