

# Final competition

Team members: 311512064鄧書桓、 311512049 陳緯翰

## Contribution

### 1. Greedy Assignment

```
matched_indices = []
if dist.shape[1] == 0:
    return np.array(matched_indices, np.int32).reshape(-1, 2)
for i in range(dist.shape[0]):
    j = dist[i].argmin()
    if dist[i][j] < 1e16:
        dist[:, j] = 1e18
    matched_indices.append([i, j])
return np.array(matched_indices, np.int32).reshape(-1, 2)
```

貪婪演算法本身為求解局部最佳解，因此最直觀的想法為求取每行的最小值，也就是同一個物體於不同frame下的距離(經過速度誤差修正)。但是，由於這邊的誤差值是對稱的(如下圖):

$$dist = \begin{bmatrix} |I_{d1}' - I_{d1}| & |I_{d1}' - I_{d2}| \\ |I_{d2}' - I_{d1}| & |I_{d2}' - I_{d2}| \\ |I_{d3}' - I_{d1}| & |I_{d3}' - I_{d2}| \end{bmatrix}$$

$I_{di}$  為新 frame 中 物 体 的 位 置 資 訊

$I_{di}'$  為舊 frame 中 物 体 的 位 置 資 訊

由此可知，其實只要求某個對角三角形(右上or左下)的即可，若沒有考慮到此現象，可能會發生同一個物品被追蹤辨識兩次的情況(會導致追蹤結果err太多)。

因此，我只要判斷完一列的最小值，就會將該列最小值的那行都變成 $1e18$ ，以此來防止該行(上個frame來的tracklets)再被計算一次。

其中，這邊的 $1e18$ 與 $1e16$ 是參考內建的匈牙利算法而來的， $1e16$ 為判斷是否追蹤為同一個物品的threshold，我有試著將這個值調低，但是效果變得很差，有些東西會追蹤不到，我認為這個是根據多次try and err而來的，沒有辦法這樣隨意改動。而 $1e18$ 的部分則是為了讓判斷其他物體追蹤時不用考慮到該項目，只要設得比 $1e16$ 還要大即可。

## 2. NUSCENE\_CLS\_VELOCITY\_ERROR

```
NUSCENE_CLS_VELOCITY_ERROR = {  
    'car': 2.5, # 2.5      調低比較好(3)  
    'truck': 4, # 4        調高比較好(4)  
    'bus': 6, # 5.5  
    'trailer': 2,  
    'pedestrian': 1,  
    'motorcycle': 3.5, # 4  
    'bicycle': 2.5,  
    'construction_vehicle': 1, # no  
    'barrier': 1, # no  
    'traffic_cone': 1, # no  
}
```

除了貪婪演算法外，我還有對NUSCENE\_CLS\_VELOCITY\_ERROR做調整，這邊的err我認為是在計算距離誤差時修正會使用到的，像是有些速度較快的東西他的err velocity需要較大，但是在經過多次測試後發現實際上並非這麼直觀，比如還需要考慮到路上的交通壅不壅擠，若擁擠的會，會導致car的速度err不能這麼高。而其他的參數項是以這個想法來做調整的，這邊就不在做過多贅述。

# Problems and Solutions:

## 1. 貪婪演算法

```
'''new'''
if dist.shape[1] == 0:
    return np.array(matched_indices, np.int32).reshape(-1, 2)
for i in range(dist.shape[0]):
    temp = np.argmin(dist, axis=1)
    min_val = 1e19
    for j in range(dist.shape[0]):
        if dist[j][temp[j]] < min_val:
            min_val = dist[j][temp[j]]
            min_dist = [j, temp[j]]
    if min_val < 1e16:
        dist[:, min_dist[1]] = 1e18
        dist[min_dist[0], :] = 1e18
        matched_indices.append(min_dist)

matched_indices.sort(key=lambda x:x[0])
print("matched_indices: ", np.array(matched_indices, np.int32).reshape(-1, 2))
return np.array(matched_indices, np.int32).reshape(-1, 2)

# temp = np.argmin(dist, axis=1)
# print(temp)
# for i in range(len(temp)):
#     matched_indices.append([i, temp[i]])
# matched_indices = np.array(matched_indices)
# print("matched_indices: ", matched_indices)

# # raise NotImplementedError("Greedy algorithm not implemented yet!")
# ### Student implement ##

5e ① 0 △ 4
```

另外，我有試過一些調整，像是上方的圖。我先將每列的最小值先列出來，再從中找尋真正的最小值，然後才對該位置的整行調整成 $1e18$ ，這樣的想法為若是依據從上而下的列來考慮最小值的話，可能會導致下方其他真正的該追蹤到的物體被上方的破壞掉。但是，若改成這樣的會，會使的結果變得超級差，因此我後來就沒有用這個方法。

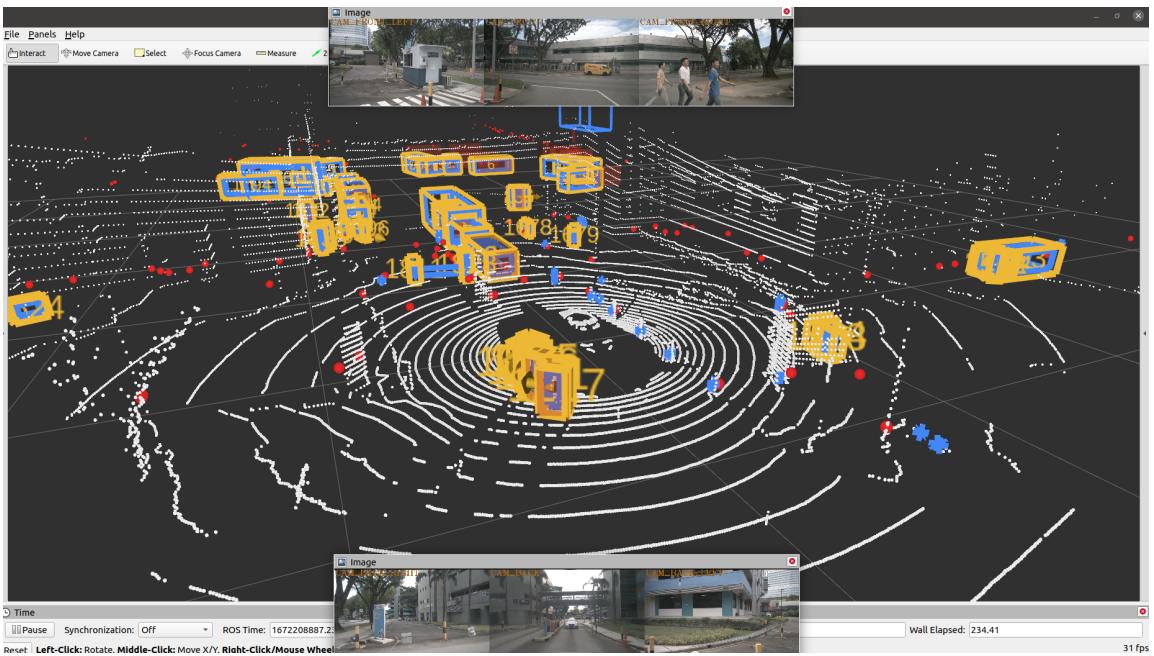
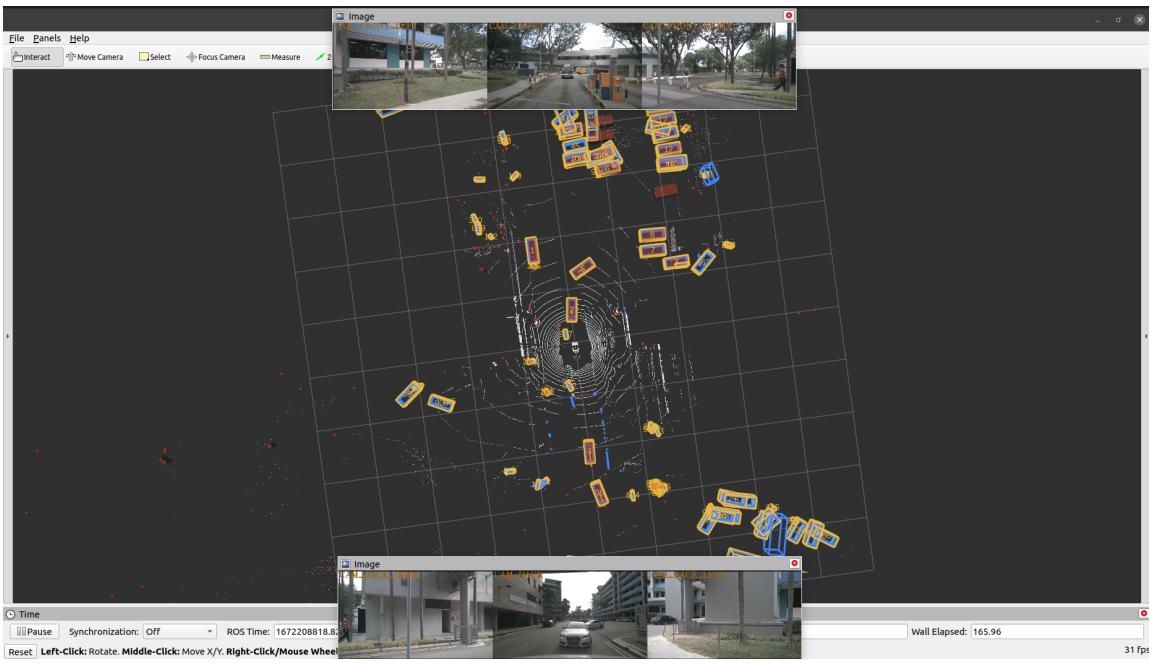
我認為導致上方想法不行的原因為之後再計算unmatch det 與 trk時是根據前幾個id 與unmatch的id來判別的(同個物體的id會在之後幾個frame中不斷+1，以此來做判斷)，所以這樣打亂順序續做最小值判斷雖然理論上沒有問題，但是因為與後方的判斷方式不同，所以無法成功。

## 2. 場景分析

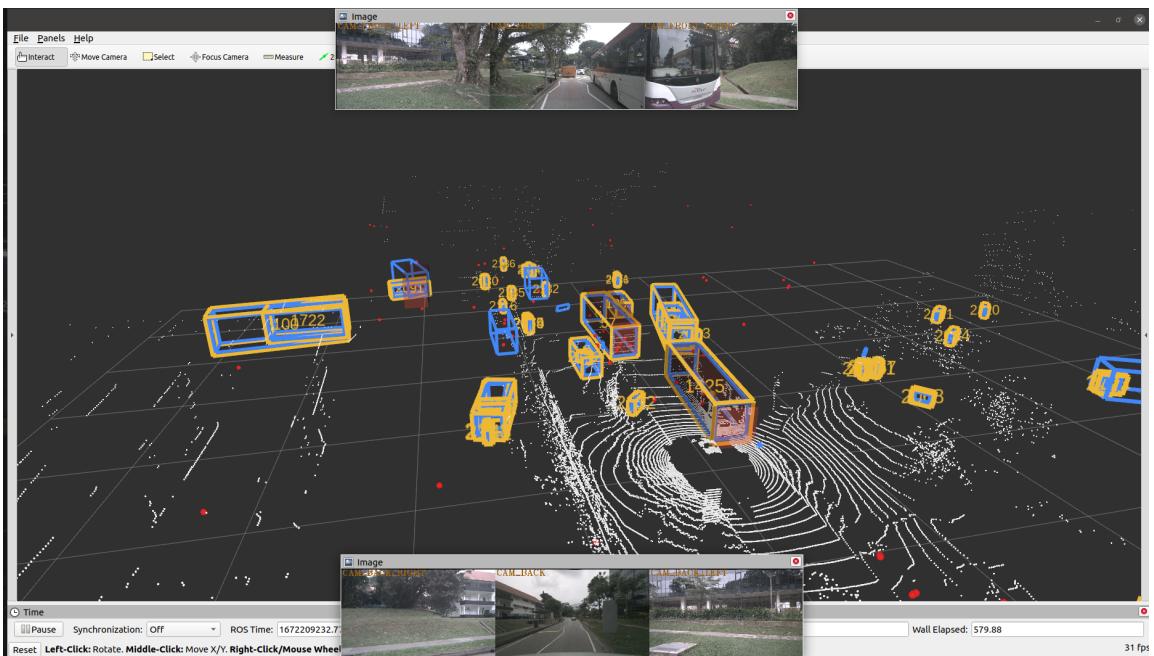
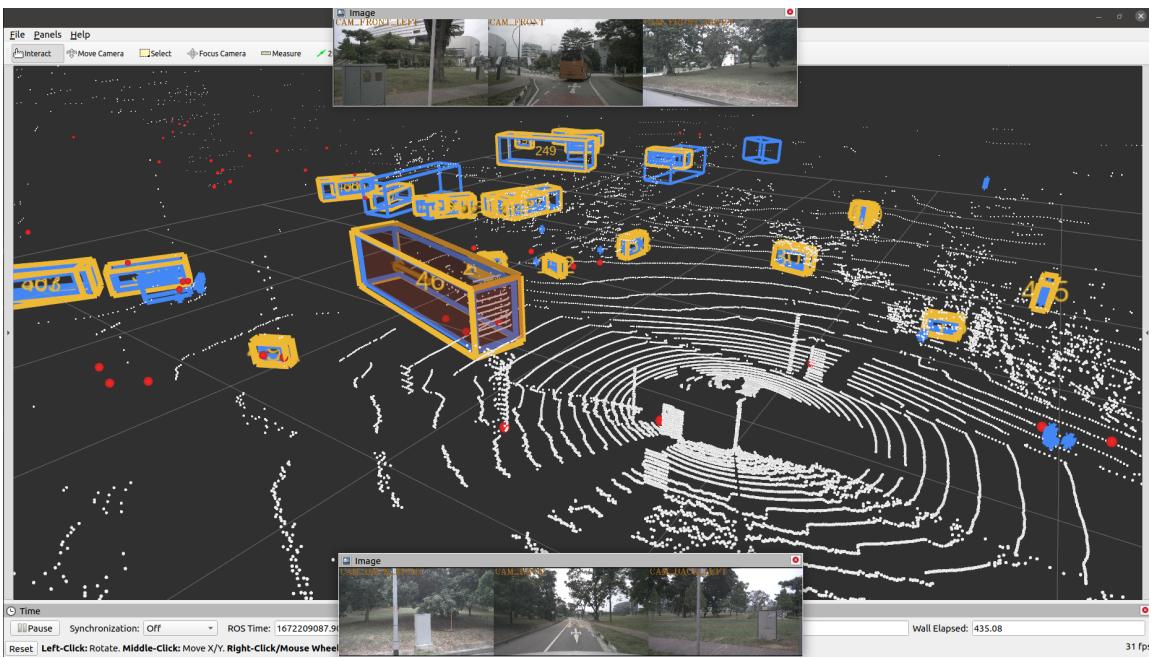
黃：預測方框

藍：Ground Truth

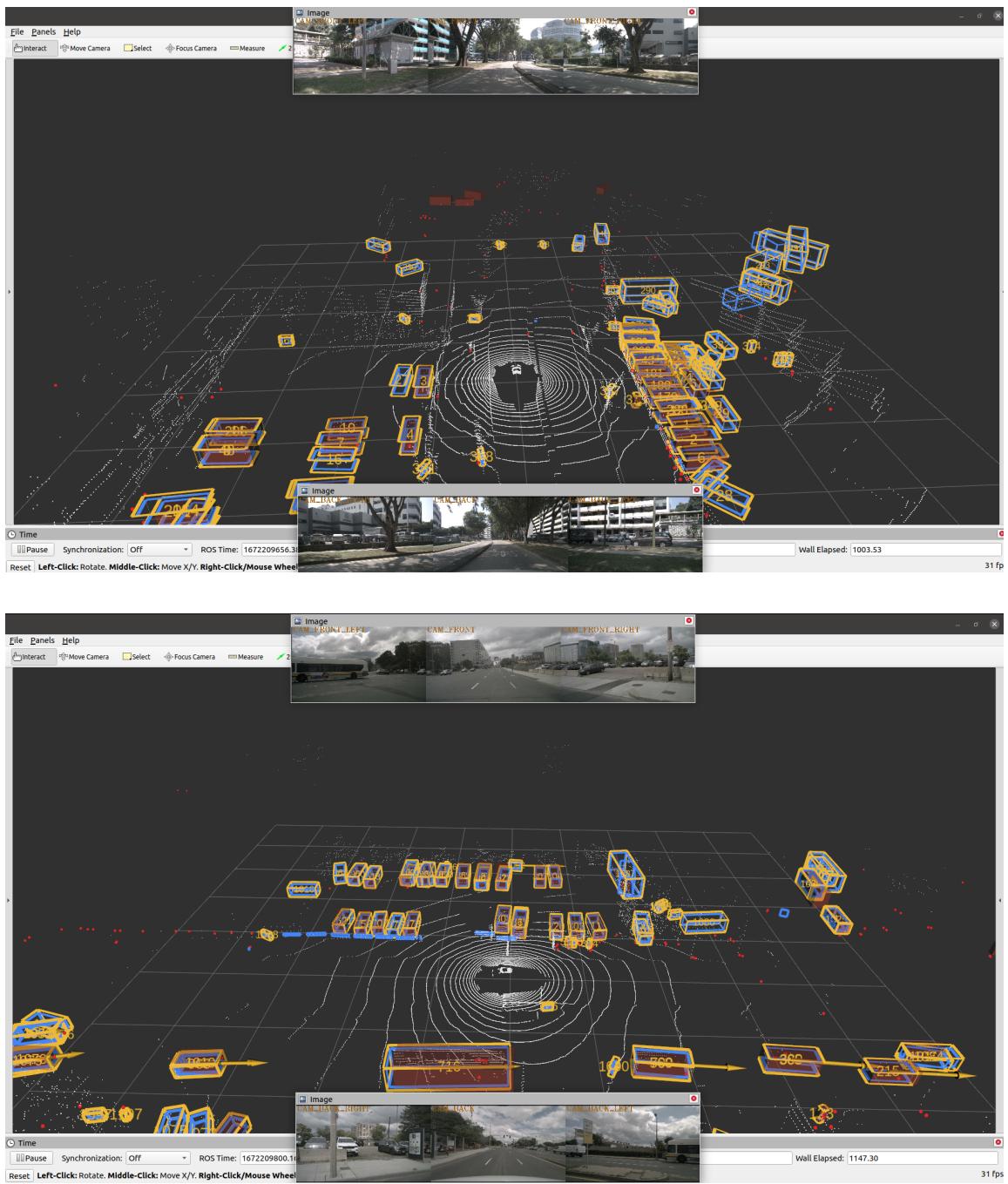
- 越遠的物體因為硬體限制的原因，可能造成預測較不準確。



- 在某些場景中，同一物體的index可能會因為車子的交會，又或者是車輛太密集而造成index的錯誤預測。



- 停車因為車子都沒有移動的情況下，預測的非常準確。



## **Work division:**

陳緯翰(50%): 可視化與分析結果，討論如何實現貪婪演算法與參數調整

鄧書桓(50%): 分析track 與tracker 的程式碼與實現貪婪演算法與參數調整