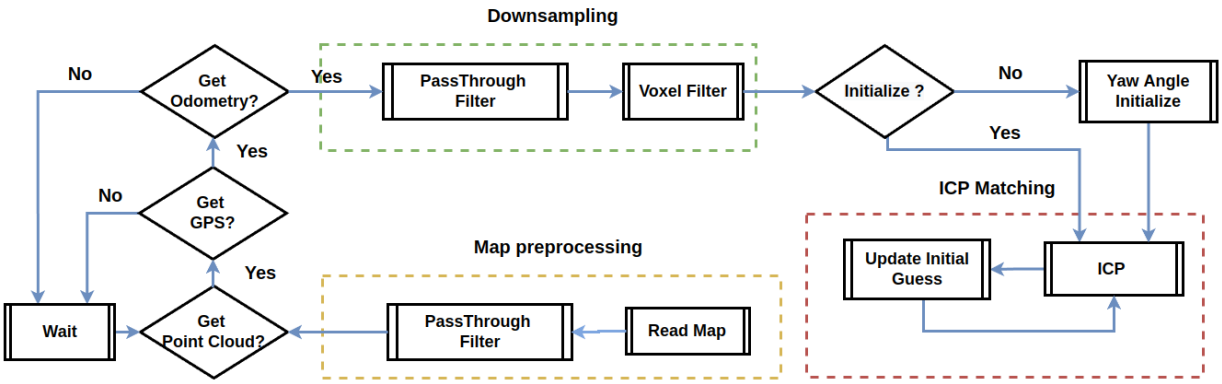
Self-Driving Car Mid Project

310605015 李啓安 機器人學程 碩二

A. Pipeline



Map preprocessing

1. 讀取 .pcd 地圖檔
2. 把重點區域進行切割 (PassThrough Filter)

Downsampling

1. Introduction
   - 嘗試過 passthrough filter 和 voxel filter兩種，task1 只要使用 voxel filter 即可，但是 task 2 和 task 3 需要搭配 passthrough filter 把一些不健康的點雲過濾掉才可以過 baseline
2. PassThrough filter
   - 功能
     - 將 x, y, z 超過threshold的數值拿掉
   - 使用原因
     - 過濾掉不健康點雲(即為妨礙定位的點雲)，因為ICP不會幫我們過濾到壞掉的點，因此我們需要檢查邊界去進行過濾，才可以讓ICP有更好的表現

3. Voxel Filter
    - 功能
        - 將 x, y, z 體積內的點雲取權重後，下取樣
    - 使用原因
        - 把體積內的點雲數量減少，可加快 ICP 運算速度

Get Initial Guess

1. First Time
    a. 參考助教的code, 將yaw角度寫成 while loop 並找出最高的Fitness Score
2. 之後的update
    a. 利用 /wheel_odometry 以及 /imu_data 的動態來讓 initial guess的 translation $\in R^3$ 更加準確

ICP

1. 配對 map 以及 lidar 點雲

Contribution

1. Initial_guess for each iteration

    原版本： 使用上一刻 ICP 的transformation

    我自己的版本： a , b 兩種方法

    a. 上一刻的transformation 加上 加速度與速度

    $\hat{x}_t = x_{t-1} + v_t \cdot dt + \frac{1}{2} \cdot a_t \cdot dt^2$, x_{t-1} 為上一刻的位置，v_t為現在的速度 (可利用 /wheel_odometry 得到) ， a_t 為現在的加速度（可利用 /imu/data 得到）

    - 我們可以把 上一次ICP做完的transformation matrix 中的 translation 替代為 $\hat{x}_t$
    - **Although it does not improve a lot, it does provide a good initial guess for ICP to converge quicker!**

    b. 上一刻的transformation直接取代GPS的位置

- 發現不準確而且會不連續的跳來跳去

2. Filter Map
    a. Introduction:
        i. 透過 Passthrough Filter 直接切割地圖
        ii. 因為直接使用發現ICP要算非常久，而且效果不見得那麼好
    b. 作業中處理：
        - 我利用一開始初始化的座標，以及車子預計的軌跡，把地圖切割我們車子會到的地方出來，這樣做的好處是ICP運算快很多，並且ICP準度也會上升，因為不會受到其他特徵的干擾
    c. 實務上處理：
        - 這個方法如果是real time 定位的狀況，也可以處理，因為我們有GPS，所以可以切割那個時間點的地圖出來做配對（不確定運算速度如何）
3. ICP 調參
    - setMaximumIterations
        - 最大迭代次數，需要在花費時間以及精度做trade off，自己測試會落在 2000 - 3000之間是我的電腦可以接受的範圍
    - .setEuclideanFitnessEpsilon
        - 為歐機李得距離的收斂條件，如果誤差小於此門檻則停止迭代。
        - 可以把 Linear Translation的 誤差降低，但是調整太小可能會讓系統警告收斂條件過於嚴苛
    - .setTransformationEpsilon
        - Rotation Matrix 之間的誤差
        - 可以把 Rotation 的 誤差降低，但是調整太小可能會讓系統警告收斂條件過於嚴苛
    - .setMaxCorrespondenceDistance
        - 對應兩點雲之間的點和點之間最大距離，對結果影響很大
        - 如果範圍擴大，可能會Match一些不好的點雲，但是範圍過小會抓不到特徵

- .setRANSACOutlierRejectionThreshold
  - 官方範例有的一個參數，為去除數據中有可能的噪聲
  - 測試過後發現沒有特別改變結果

C. Problem and Solution

1. 車子不前進
   a. 第一題這個問題不會發生，但是到第二題的時候，因為在一個長廊上面所以很容易就會原地不動，原本使用voxel filter ，更改完 pass through filter 就可以成功了。
   b. 推測是 passthrough 把一些干擾的點雲去掉，ICP的match 就更加緊密了
2. ICP被拉走
   a. 這個問題在每個場地都會發生，因為受到一些有問題的點雲，例如太高或是太低點雲，ICP的點雲配對會有一些傾斜的狀況，這個時候一樣去調整 PassThrough Filter 的 limits 就可以改善
3. 運算量大
   a. 每個task 都有可能發生，特別是 task2, task3 的地圖點雲數量又更大了
   b. 可以利用 PassThrough Filter 將地圖切小一點，保留重點區域，這樣運算速度可以塊很多
   c. 可以將ICP Algorithm 每次迭代的上限或是收斂條件調大一點，但是這邊需要跟精度去做trade off，要花一些時間調整一下
   d. 可以使用 voxel filter 對點雲做 downsampling ，可以降低點雲的數目，但是實際測試發現，也有可能造成匹配效果不好

D. Others

- How to run ?
  - 因為進入Yaw 角初始化的時候會有一段時間會收不到 Lidar Message，這樣會遺失一些資料，所以要在初始化的時候將bag 暫停一段時間，等到配對好了以後再繼續播放

- Task1 :

```
$ roslaunch localization itri.launch
$ rosbag play -r 0.1 --pause sdc_localization_1.bag --clock
```

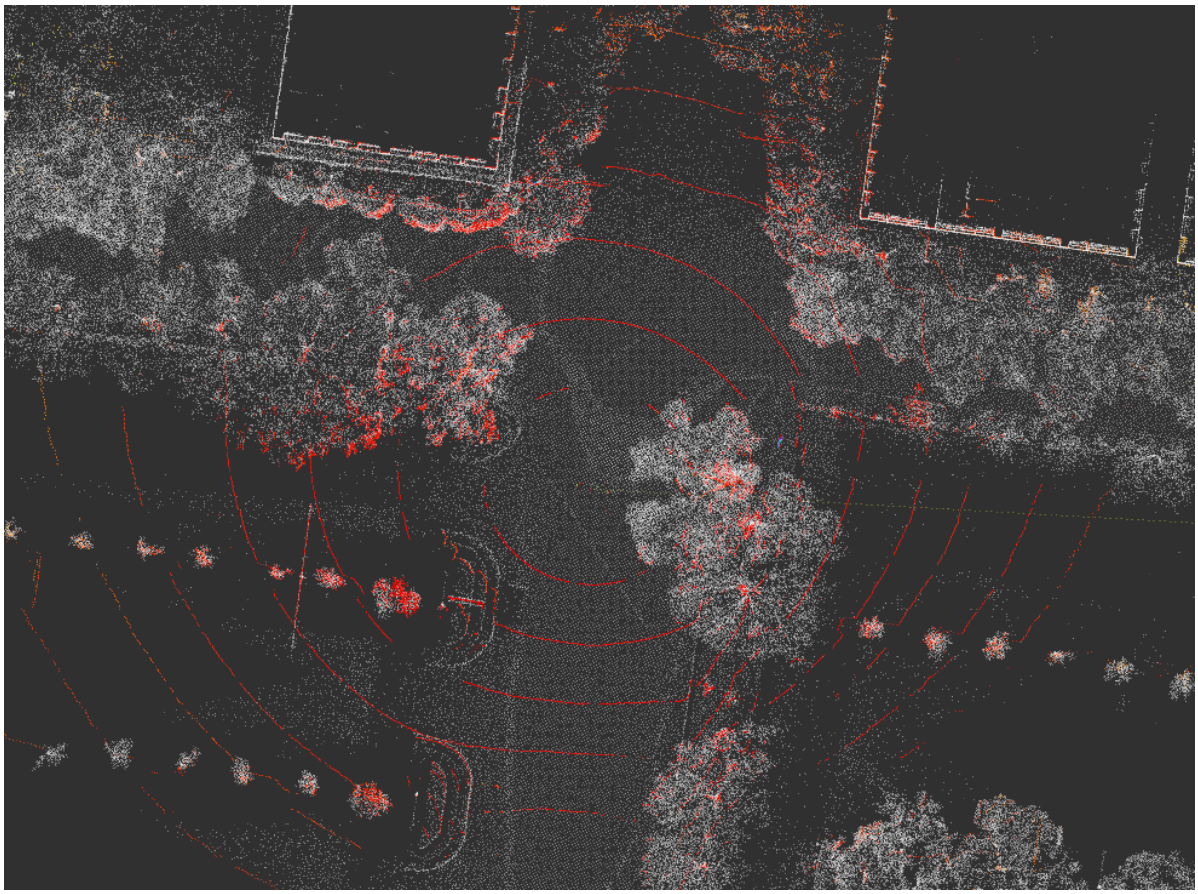
- Task2 :

```
$ roslaunch localization nuscenes2.launch
$ rosbag play -r 0.1 --pause sdc_localization_2.bag --clock
```

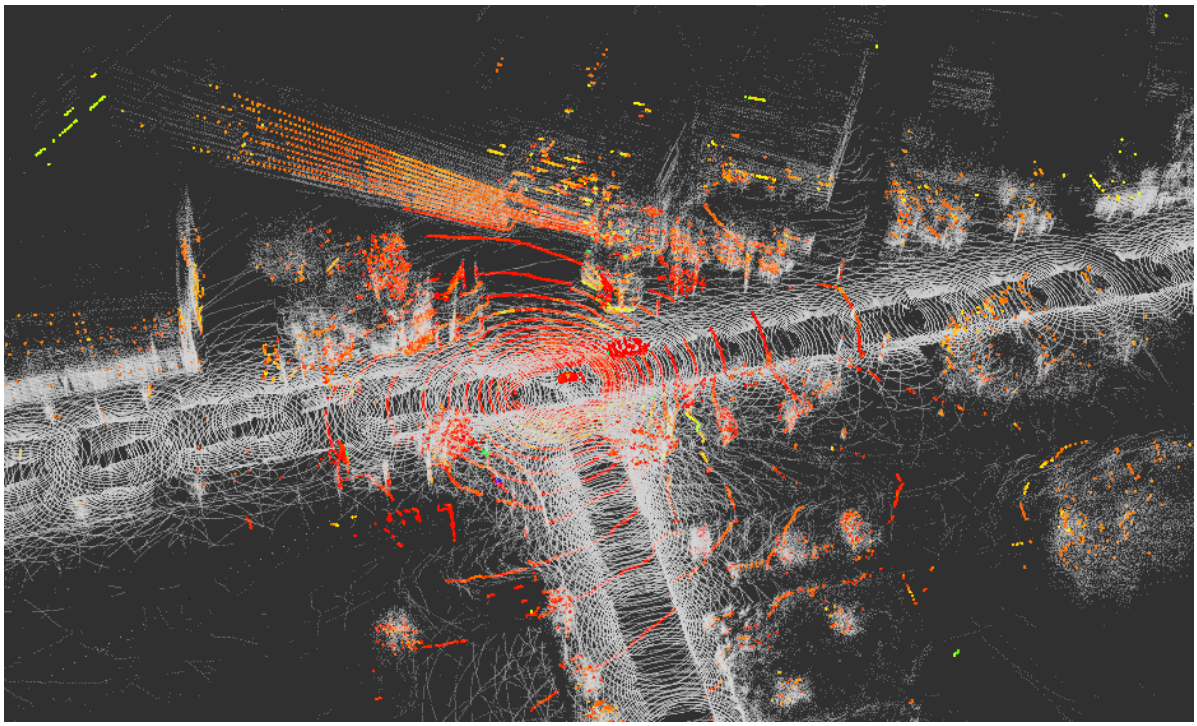
- Task3 :

```
$ roslaunch localization nuscenes3.launch
$ rosbag play -r 0.1 --pause sdc_localization_3.bag --clock
```

- 每個任務的最後位置
  - Task1



- Task2



- Task3