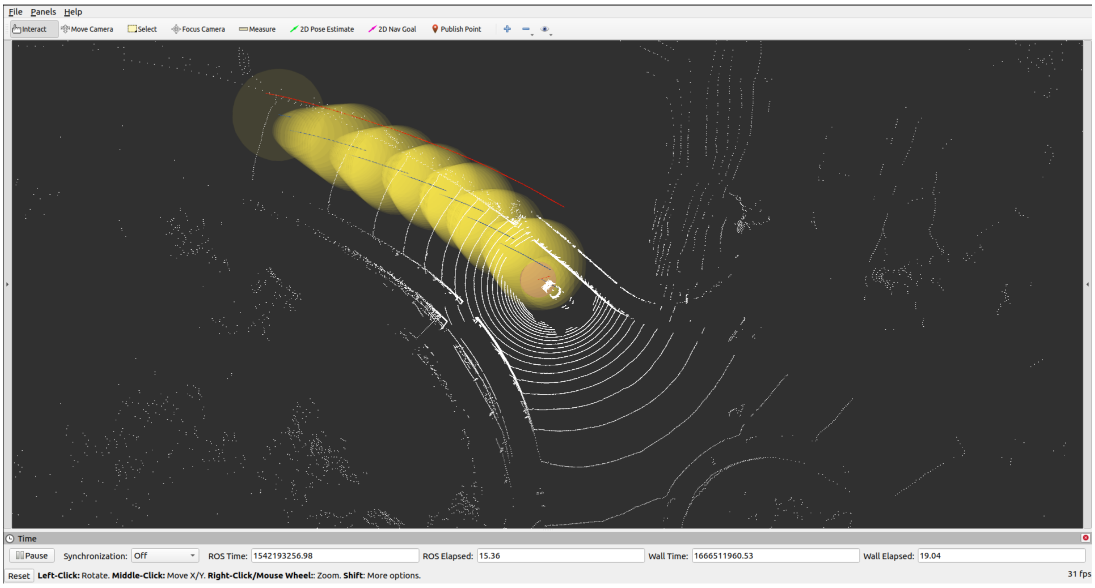
Report of HW4

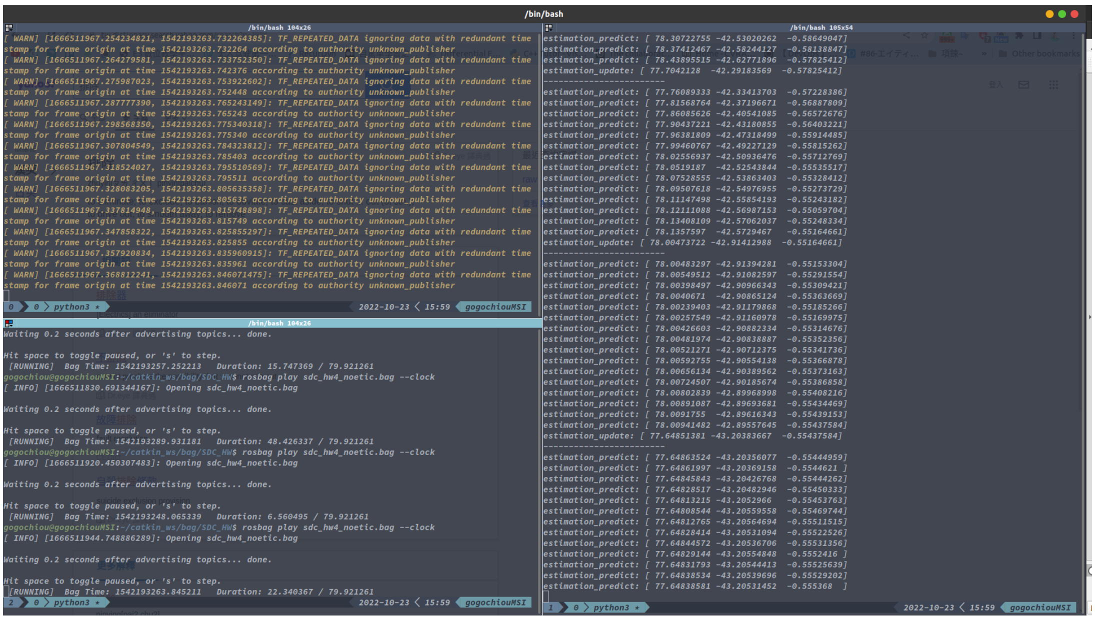
> Student ID : A111137
>
> Name : 邱柏鎧 (gogochiou)

Result

1. state with [x, y, yaw]

   In 3 state condition, I try two method, one is scale the cov with adding np.identity(3), and the other one is assuming the covariance become covariance of current radar odometry divide covariance of last radar odometry. Two methods have similar result, and successfully make the covariance become larger. However, there is still one problem for the latter, that is calculation limitation for 0 divide 0.

2. state with [x, y, yaw, dx, dy, dyaw]

The design of it will be explain after, but the result is similar th 3 state condition, because it is hard to find the covariance since we only have radar odometry, which merely provides three states' covariances. In my design method, there is some problem that [dx, dy, dyaw] does not improve anything.

Discussion

1. How do you design the Kalman filter and the parameters?

   - Predict

     ```
     def predict(self, u):
         self.x = np.matmul(self.A,self.x) + np.matmul(self.B,u)
         self.P = 
     np.matmul(np.matmul(self.A,self.P),np.transpose(self.A)) + self.R
     ```

   - Update

     My strategy is making the state always [x, y, yaw], even we don't need the state of yaw. I make z become 3*1 and H become 3*3 matrix. Other setting is same as the pseudo code of KF algorithm. This setting is feasible for 6 state condition, cause we don't update [dx, dy, dyaw]

     ```
     def update(self, z):
         ## Make z become 3*1 and H become 3*3 matrix
         im_z = np.transpose(np.append(z, 0))
         im_H = np.vstack([self.H, np.array([0, 0, 0])])
         ## Kalman Filter update part
         temp_sigma = np.matmul(np.matmul(im_H, self.P), 
     np.transpose(im_H)) + self.Q
         Kt = np.matmul(np.matmul(self.P, np.transpose(im_H)), 
     np.linalg.inv(temp_sigma))
         self.x = self.x + np.matmul(Kt,(im_z - np.matmul(im_H, 
     self.x)))
         self.P = np.matmul((np.identity(3) - np.matmul(Kt, im_H)), 
     self.P)

         if np.isnan(np.sum(self.x)) == True :
             raise ValueError

         return self.x, self.P
     ```

   - 3 state condition

     $$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

   - 6 state condition - X=[x, y, yaw, dx, dy, dyaw]

$$X_t = AX_{t-1} + Bu_t$$

In my opinion, state of difference should be $(x,y,yaw)*{t} - (x,y,yaw)*{t-1}$. So the setting become :

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

2. What is the **covariance** matrix of **GPS, radar odometry** and what does it mean?

   - 3 state condition

     Because the covariance of radar odometry is too small (~10^-7), so the state will converge to radar odometry. So, my design is similar to scale the value from radar odometry.

```
odometry_covariance = np.identity(3)
odometry_covariance += np.array(data.pose.covariance).reshape(6,
-1)[:3,:3]
```

$$R = \begin{bmatrix} 1 + cov_{x_radar} & 0 & 0 \\ 0 & 1 + cov_{y_radar} & 0 \\ 0 & 0 & 1 + cov_{yaw_radar} \end{bmatrix}$$

     And for the GPS, I just use the covariance of GPS topic. Because in my calculation is 3*3, so i make covariance of yaw still have a constant value.

$$Q = \begin{bmatrix} cov_{x_GPS} & 0 & 0 \\ 0 & cov_{y_GPS} & 0 \\ 0 & 0 & constant \end{bmatrix}$$

   - 6 state condition

     I try the method similar to 3 state condition scaling, but the result was same. Then i design the covariance as below :

$$R = \begin{bmatrix} \frac{cov_{x,t}}{cov_{x,t-1}} & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{cov_{y,t}}{cov_{y,t-1}} & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{cov_{yaw,t}}{cov_{yaw,t-1}} & 0 & 0 & 1 \\ 1 & 0 & 0 & cov_{x,t} & 0 & 0 \\ 0 & 1 & 0 & 0 & cov_{y,t} & 0 \\ 0 & 0 & 1 & 0 & 0 & cov_{yaw,t} \end{bmatrix}$$

But after testing, it work similar to 3 state condition. Also, another problem is fraction method will make sometimes calculation error, that is value of **nan** will occur while staying still.

GPS is setting as 3 state condition.