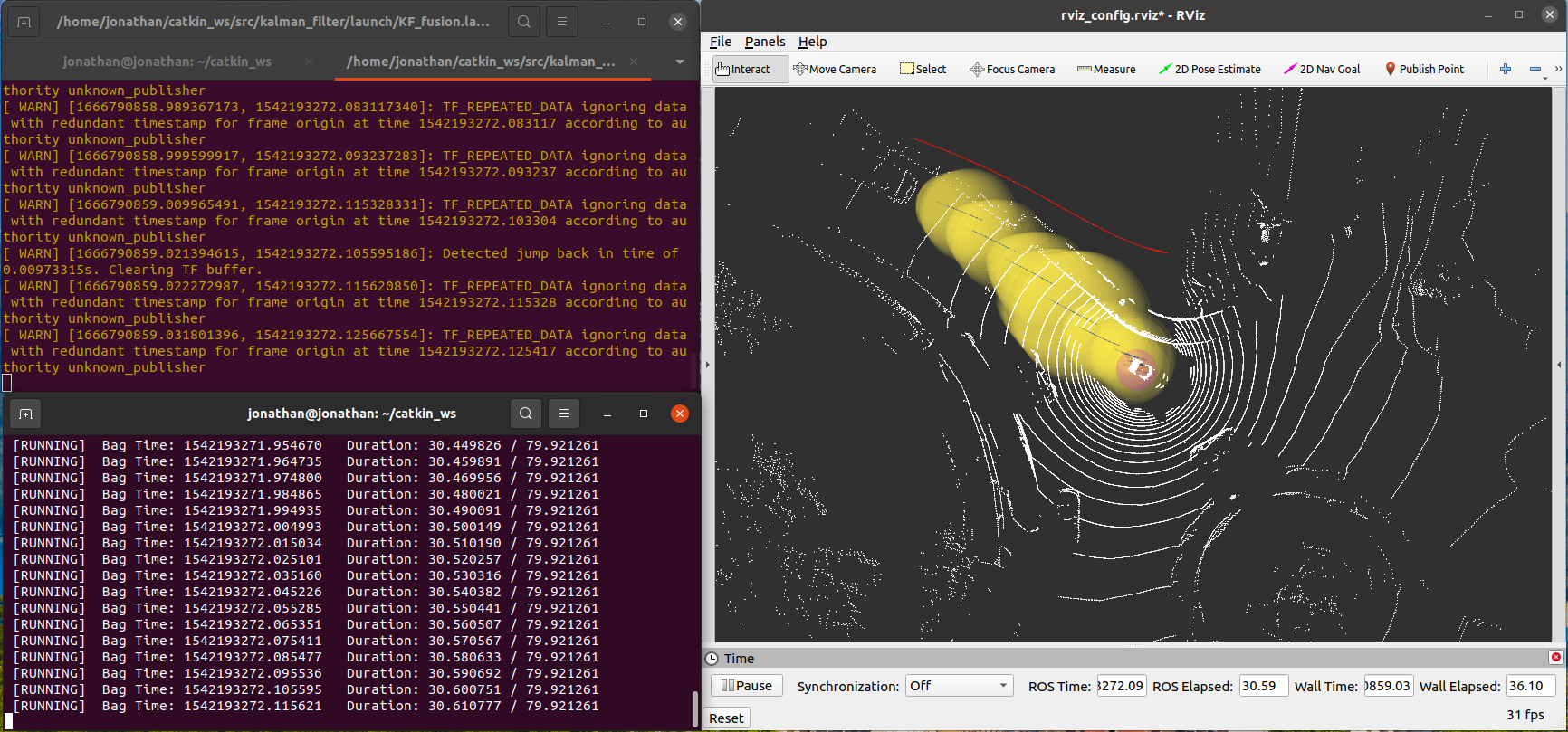
**Homework #4 Report  
[EECN30168] Self-Driving Cars 2022**

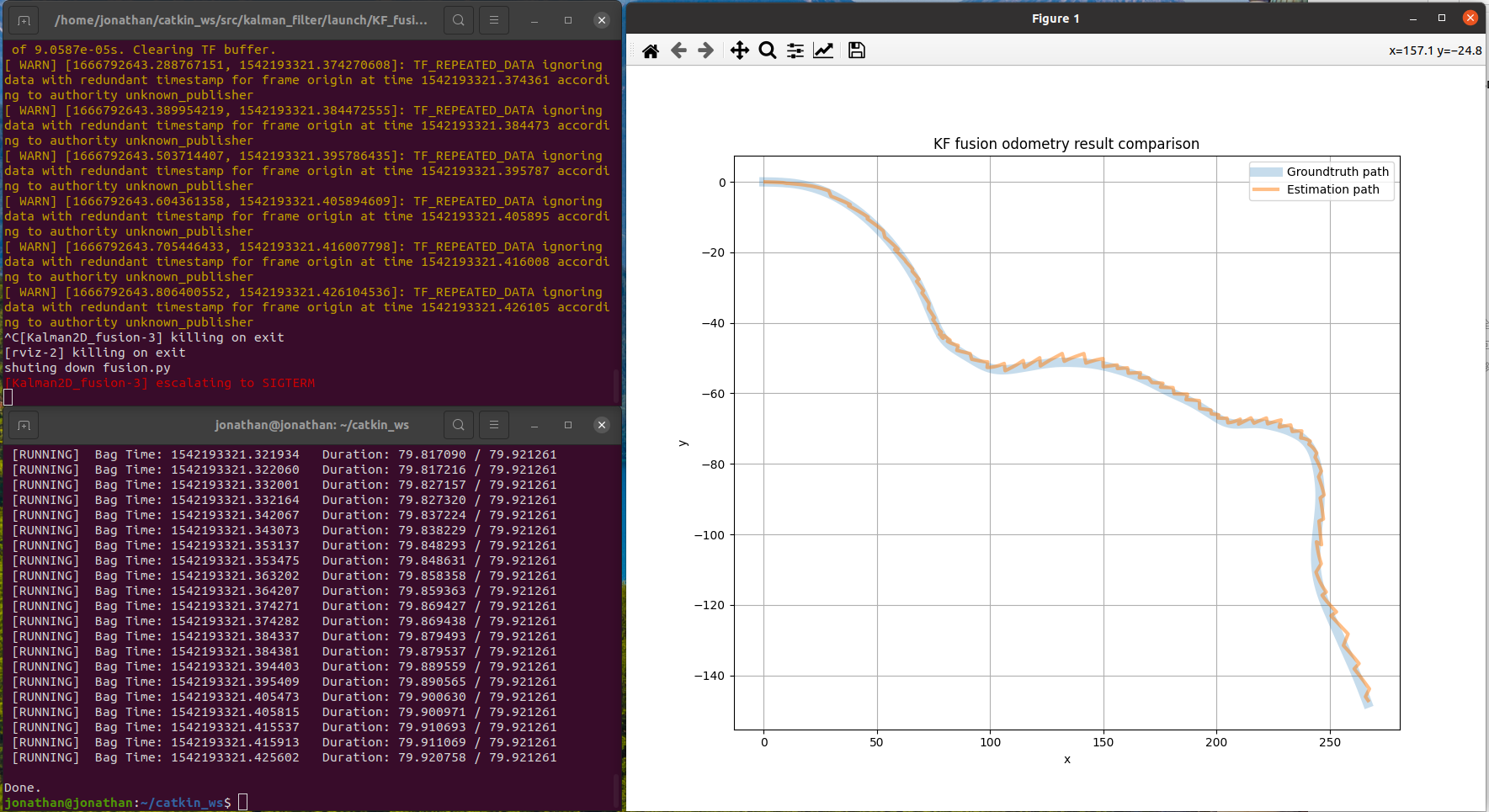
**Student ID:** 311605004 **Name:** 劉子齊 **Date: 2022.10.26**

1. **Introduction**

In this homework, our goal is to apply the Kalman filter we implemented in the previous homework on a real set of self-driving car data. Furthermore, we need to visualize the result through “rviz”, and make further adjustments to our program through the observations to the output result, as shown in the following session.

1. **Result**

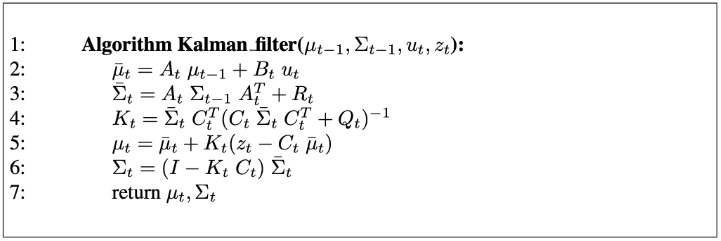
****



1. **Discussions**

**3-1. Design of the Kalman filter and the Parameters**

For the implementation of the Kalman filter, I followed the algorithm in the following figure.



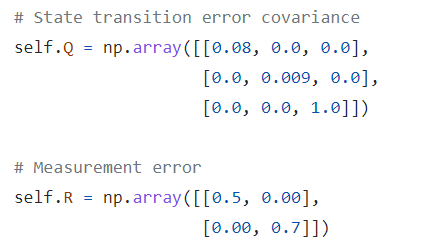
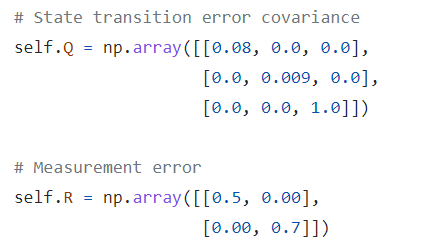
For the prediction function, I first calculated the predicted state estimate x. I obtain the expected state estimate x by adding the dot product of the transition matrix A and the state estimate x, and the dot product of the transition matrix B and the control vector u. Then I calculate the error matrix P by adding up the covariance of the state transition error Q with the dot product of the transition matrix A, the error matrix P, and the transform of the transition matrix A.

For the update function, I first calculate the innovation covariance S, which can be found in the parentheses in the equation of the optimal Kalman gain K in the algorithm above. For the innovation covariance S, I added up the measurement error R with the dot product of the observation matrix H, the error matrix P, and the transform of the observation matrix H. Then I obtained the optimal Kalman gain K by calculating the dot product of the error matrix P, the transform of the observation matrix H, and the inverse of the innovation covariance S matrix.

Then I’ll calculate the measurement post-fit residual y, which is the subtraction of the dot product of the observation matrix H and the state estimate x to the matrix z. With the measurement post-fit residual y, we can get the final state estimate x by adding the state estimate x itself with the dot product of the optimal Kalman gain K and the measurement post-fit residual y.

Lastly, for the error matrix P, we first initialize a 3x3 matrix I with np.eye(), which with ones on the diagonal and zeros elsewhere. Then we calculate the error matrix P by calculating the dot product of the subtraction of the dot product of the optimal Kalman gain K and the observation matrix H from the 3x3 matrix we just created and the error matrix P itself. Finally, we get the final error matrix P, and the following figure is the plotting result of the Kalman filter I implemented.

For the parameters of the Kalman filter, I made the adjustments like I mentioned in the previous project, which I adjust the value to the corresponding axis which offset the most, and make further adjustment through the result obtained. As a result, the following figure is how I eventually set the parameters of the Kalman filter.



**3-2. Covariance Matrix of GPS & Radar Odometry**