A53238541 Chun Hu

1. Introduction

   Kalman filter could be used to estimate the robot's state from control inputs and measurement data. However, most of the motion models and the observation models are nonlinear and Kalman filter could not handle nonlinear system. Therefore, we need to use extended Kalman filter or unscented Kalman filter in nonlinear system. In this project, we need to implement unscented Kalman filter to estimate the robot's orientation.

2. Problem Formulation

   Given a set of IMU data (acceleration$\in R^3$ and angle $\in R^3$), the goal is to calculate the robot's orientation$\in R^3$ and produce panorama image.

3. Technical Approach

   (1) Bias

   Assume the robot will not move in the first N timestamps, the average of first N IMU values of (A_x, A_y, A_z) and (W_x, W_y, W_z) should be (0,0,1) and (0,0,0). Thus, we can calculate the bias using the following formula:

   Bias(A_x) = $\frac{\sum_{i=1}^{N} A_x(i)}{N}$ Bias(A_y) = $\frac{\sum_{i=1}^{N} A_y(i)}{N}$ Bias(A_z) = $\frac{\sum_{i=1}^{N} A_z(i)}{N} - \frac{300*1023}{3300}$

   Bias(W_x) = $\frac{\sum_{i=1}^{N} W_x(i)}{N}$ Bias(W_y) = $\frac{\sum_{i=1}^{N} W_y(i)}{N}$ Bias(W_z) = $\frac{\sum_{i=1}^{N} W_z(i)}{N}$

A_x	A_y	A_z	W_x	W_y	W_z
511	501	513	374	375	370

   (2) Motion model

   $q_{t+1|t}^{i} = q_{t|t} \circ q_w^i \circ \Delta q$

   (3) Measurement model

   $(0, Z_{t+1}^{i}) = \bar{q}_{t+1|t}^{i} \circ [0,0,0,1]^T \circ q_{t+1|t}^{i}$

   (4) Predict step

$\boxed{\text{Predict step}}$ Q : parameter

$E_t^{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ $E_t^{(i)}$ = columns $\left(\pm\sqrt{n(P_{t|t}+Q)}\right)$ (use Cholesky decomposition).

angle velocity

$q_{t+1|t}^{(i)} = q_{t|t} \circ \text{euler2quat}(E^{(i)}) \circ \text{euler2quat}(w\Delta t)$

$q_{t+1}^{mean}, e_{t+1|t}^{(i)} = \text{avgquat}(q_{t+1|t}^{(i)}, \alpha^{(i)}), \quad \alpha^{(0)} = 0, \quad \alpha^{(i)} = \frac{1}{2n} \text{ for } i \neq 0$

$P_{t+1|t} = 2\, e_{t+1|t}^{0}\, e_{t+1|t}^{0\,T} + \frac{1}{2n}\sum_{i}^{2n} e_{t+1|t}^{(i)}\, e_{t+1|t}^{(i)\,T}$

A53238541 Chun Hu

(5) Update step

$\boxed{\text{Update step}}$ P_R : parameter

$$(0, z_{t+1}^{(i)}) = \bar{q}_{t+1|t}^{(i)} \, [0,0,0,1]^T \, q_{t+1|t}^{(i)}$$

$$z_{t+1}^{mean} = 0\, z_{t+1}^{0} + \frac{1}{2n}\sum_{i}^{2n} z_{t+1}^{(i)}$$

$$P_{zz} = 2(z_{t+1}^{0} - z_{t+1}^{mean})(z_{t+1}^{0} - z_{t+1}^{mean})^T + \frac{1}{2n}\sum_{i}^{2n}(z_{t+1}^{(i)} - z_{t+1}^{mean})(z_{t+1}^{(i)} - z_{t+1}^{mean})^T$$

$$P_{vv} = P_{zz} + P_R$$

$$P_{xz} = 2\, e_t^{0}(z_{t+1}^{0} - z_{t+1}^{mean}) + \frac{1}{2n}\sum_{i}^{2n} e_t^{(i)}(z_{t+1}^{(i)} - z_{t+1}^{mean})$$

$$K_{t+1} = P_{xz}\cdot P_{vv}^{-1}$$

acc value

$$q_{t+1|t+1} = q_{t+1}^{mean} \circ \mathrm{euler2quat}(K_{t+1}(z - z_t^{mean}))$$

$$P_{t+1|t+1} = P_{t+1|t} - K_{t+1} P_{vv} K_{t+1}^T$$

(6) Quaternion Python class
I define quaternion as a Python class and manipulate the quaternions using the class function.

(7) Quaternion transformation
I use Python function called transform3d to help me transform between Euler angles and quaternions

(8) Quaternion average
Implement the algorithm of slide 5 P15

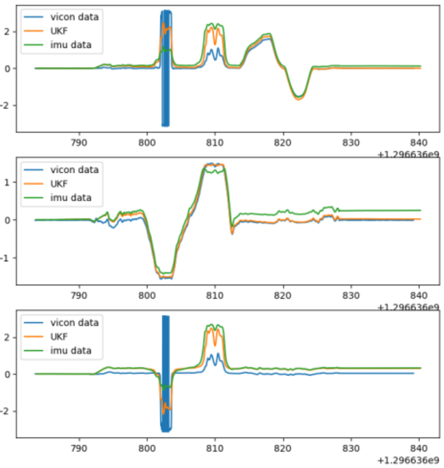
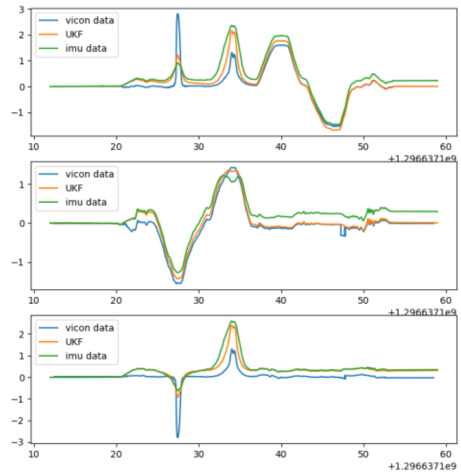
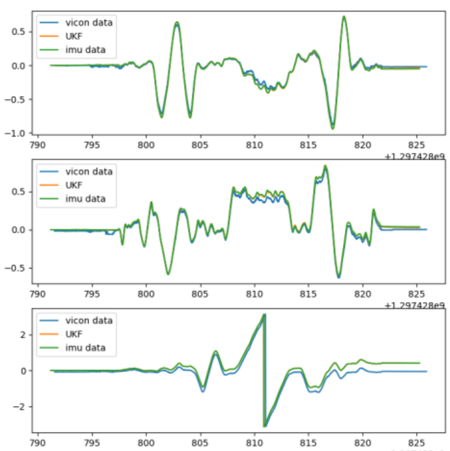
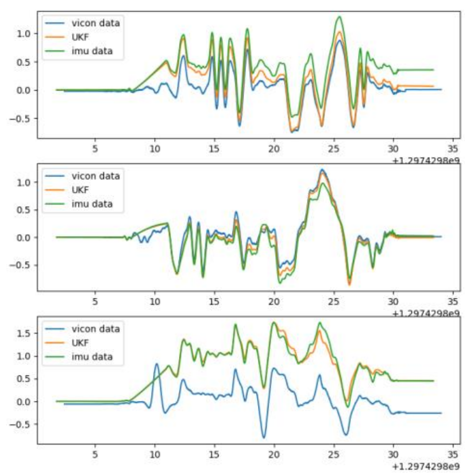
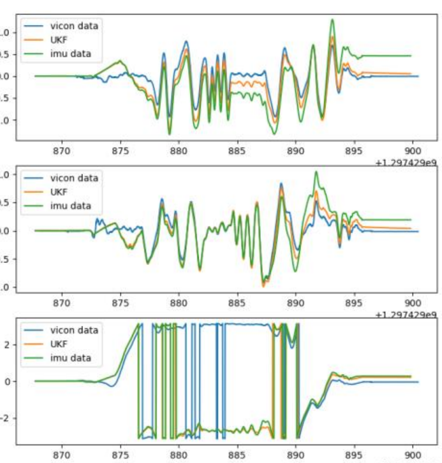
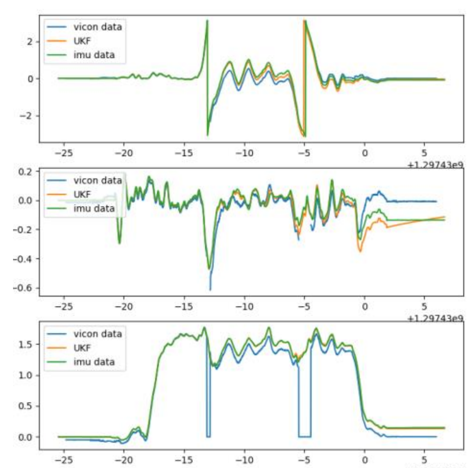
(9) Panorama

1. Find longitude (λ) and latitude (ϕ) of each pixel.
Assume the top-left corner pixel's index to be (0,0), the bottom-right corner's pixel's index to be (240, 320).
For pixel index (i, j), it's (latitude, longitude) are $(22.5 - \frac{45}{239} i, -30 + \frac{60}{319} j)$.
2. Convert Spherical $(\phi, \lambda, 1)$ to Cartesian assuming depth 1.
3. Rotate the Cartesian coordinates to the world frame using our UKF result. (Make sure the timestamp is the same from IMU and camera).
4. Convert Cartesian to Cylindrical (ρ, θ, z).
5. Map Cylindrical coordinate to the panorama image.
row index = $(\frac{z+1}{2}) \times$ image. height.
column index = $(\frac{\theta+\pi}{2\pi}) \times$ image width.
6. Copy the original image value to the panorama image.

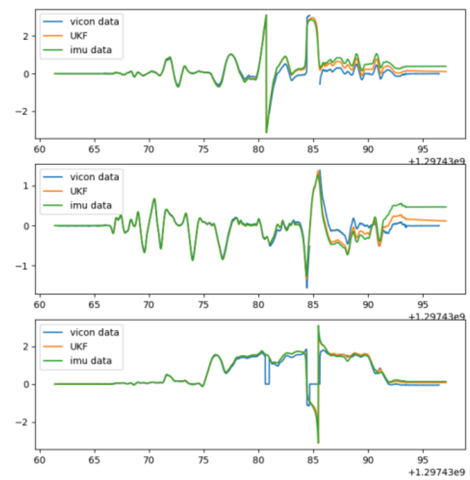
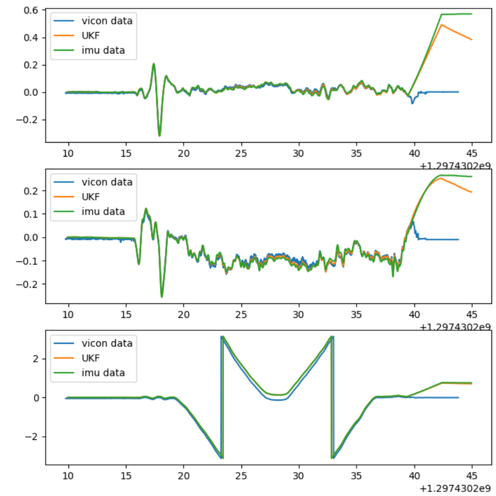
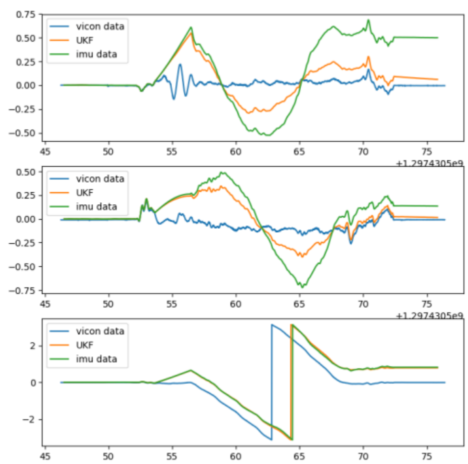
A53238541 Chun Hu

4. Results
   (1) UKF

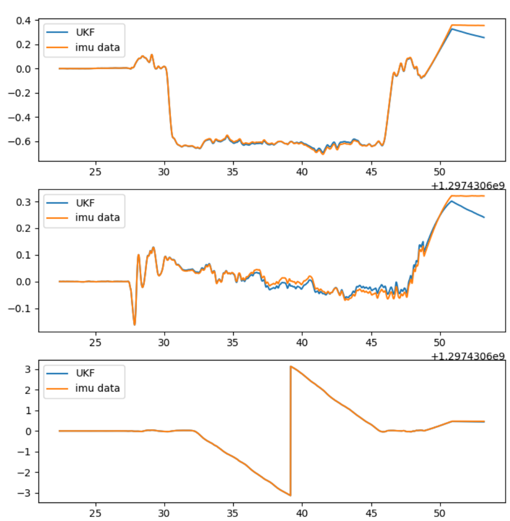
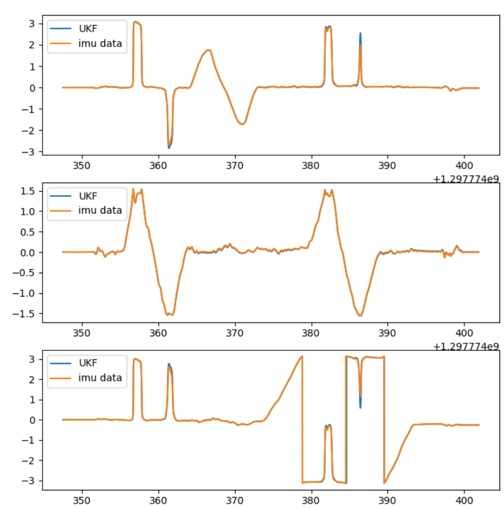
   Train data

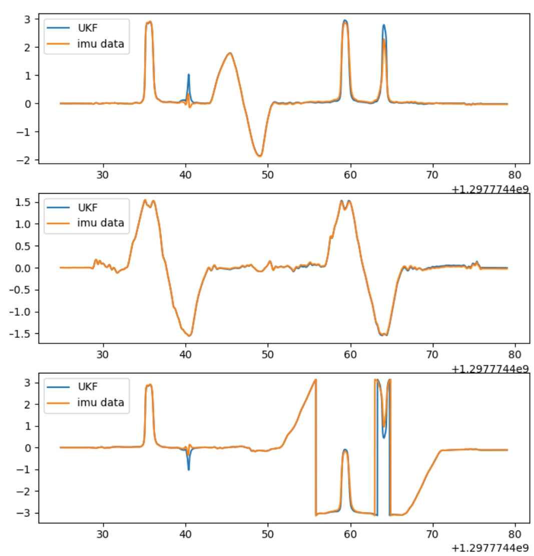
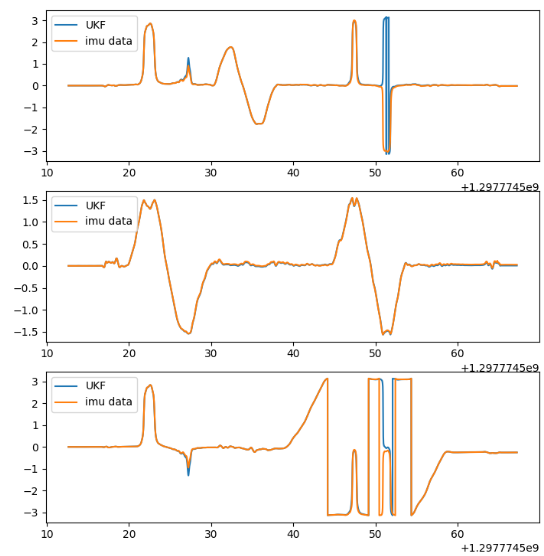
1	2
	
3	**4**
	
5	**6**
	

A53238541 Chun Hu

7	8
	
9	
	

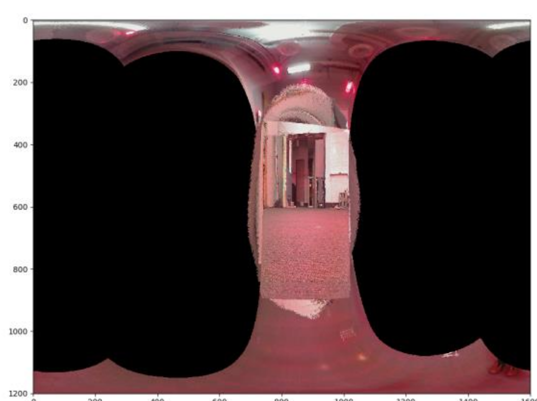
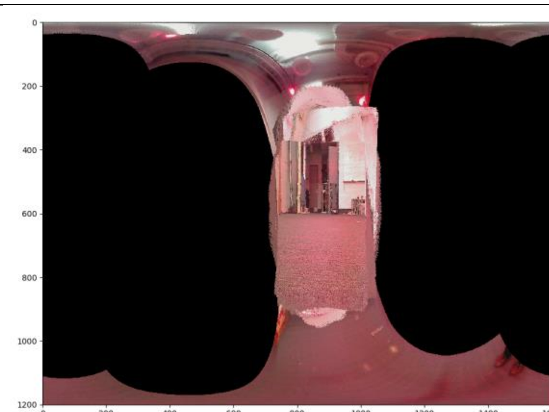
Test data

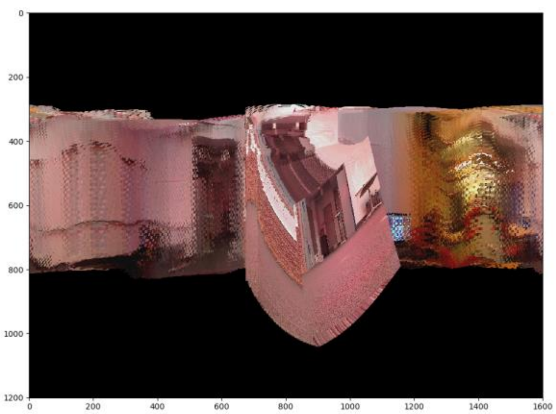
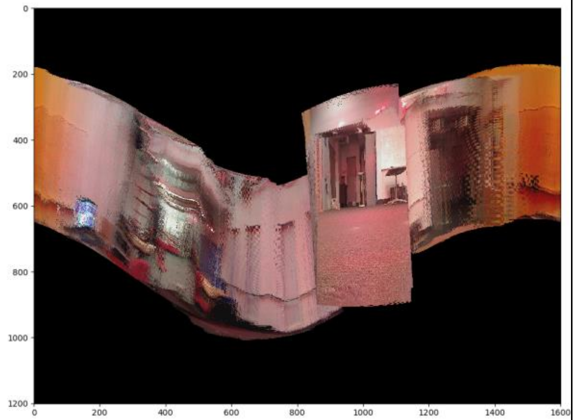
10	11
	

12	13
	

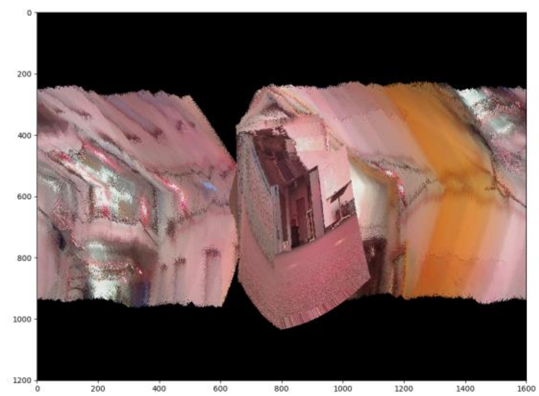
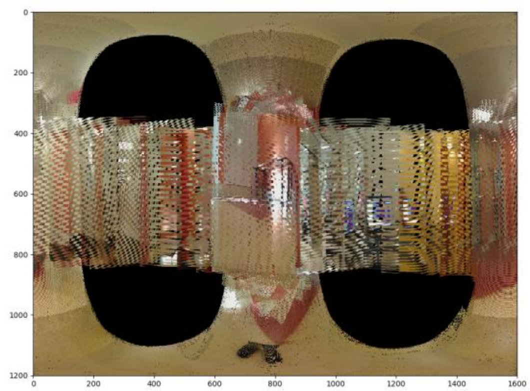
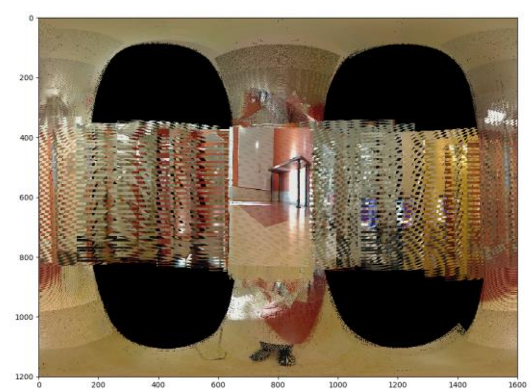
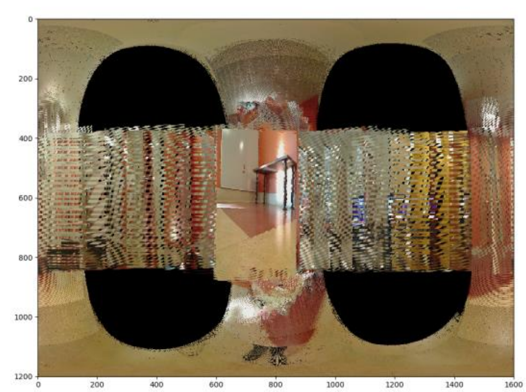
1. Parameter tuning
   On the piazza website, professor suggests us to set the value of our state vector, process and observation noise covariance matrix to 0.0001 * identity matrix. However, after trying several times on the training data, I decide to set the value of state vector covariance matrix to 0.01 * identity matrix, process noise covariance matrix to 0.000001 * identity matrix and observation noise covariance matrix to identity matrix. It is because our angular velocity integration is really close to the Vicon data, the process noise should be small. On the other hand, the observation noise should be large to make the filter trusts the angle data more.
2. UKF
   Since the UKF uses the angle data and the measurement data at the same time, it performs better than only using the angle data. We can the proof in the train data set, the UKF (orange line) is closer to the Vicon than the IMU data (green line).

(2) Panorama

1	2
	
8	9

Test data

10	11
	
12	13
	

1. Timestamp
   To get the camera's orientation, we need to know the camera's timestamp and get the orientation value which is closest to the timestamp. To achieve the goal, I simply use a foor loop to find the timestamp -closest orientation.
2. Image rotate in robot's x axis
   I think there is a problem in my result because the image rotates along robot's x axis in every panorama image. However, I am not sure the problem comes from the UKF's inaccuracy or the panorama function.