ESE650 Learning in Robotics, 2018 Spring
Project 2: Orientation Estimation
Wudao Ling

Introduction

In this project, raw IMU data including gyroscope and accelerator data is given to estimate quadrotor's orientation. Vicon data is also provided for evaluation. Then estimated orientation together with camera data were used to generate a panorama.
I started with cleaning sensor data according to IMU device reference. Then I derived the function to compute orientation only based on one sensor. Later a Quaternion-based Unscented Kalman Filter was implemented to fuse gyroscope and accelerator, and its estimates were compared with Vicon ground truth. With a robust orientation estimation, I projected camera images to a sphere surrounding the global camera to generate a panorama.

1 Sensor Calibration

From IMU reference, the sample calculation that convert from the raw A/D values to physical units are:

$$scale\_factor = Vref/(1023 * sensitivity) \tag{1}$$
$$value = (raw - bias) * scale\_factor \tag{2}$$

From the document of ArduIMU+ V2, we know that Vref is 3300 mV, acc sensitivity is 330 mV/g and gyro sensitivity is 3.33 mV/degree.

Accelerometer
I took the mean of first 10 samples of acc data as bias because quadrotor stays still at the beginning. And it is necessary to correct the z-axis bias by substracting gravity $1/scale\_factor$.

Gyroscope
Similarly, the mean of first 10 samples of gyro data is used as bias. The gyro data is converted from degree to radian.

2 Orientation Estimated by Gyroscope

The rotation in this project is represented as Quaternion. Quaternion multiplication between q_a and q_b equals to the rotation q_b followed by q_a.
Given 3-axis angular velocity and timestamps from gyroscope plus an initial estimate, we can accumulate and estimate the orientation of quadrotor as follows:

$$angle: \quad \Delta\alpha = ||\vec{\omega}|| * \Delta t \tag{3}$$

$$axis: \quad \vec{e} = \frac{\vec{\omega}}{||\vec{\omega}||} \tag{4}$$

$$\Delta q = [cos(\Delta\alpha/2) \quad \vec{e}sin(\Delta\alpha/2)] \tag{5}$$

$$q_{k+1} = q_k \Delta q \tag{6}$$

Where q_k is the previous state, Δq is the rotation quaternion, and q_{k+1} is the transformed state.

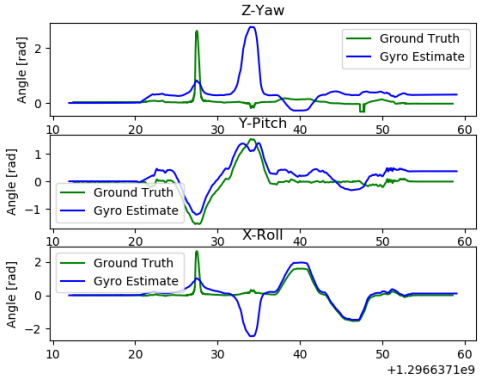

Figure 1: Gyro estimate on dataset 2

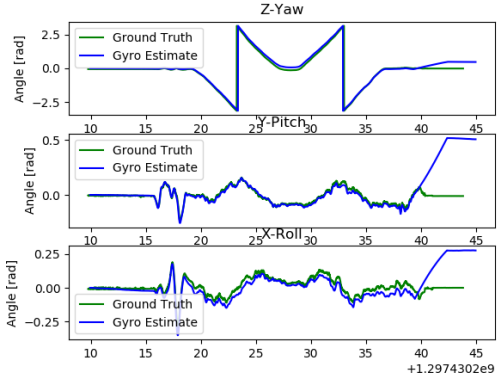

Figure 2: Gyro estimate on dataset 8

3 Measurement by Accelerometer

Given a orientation estimate from Gyroscope, we can apply quaternion rotation on gravity vector to estimate Accelerometer's reading. From the diagram we can know true acc reading is actually more noisy than estimated measurement.

$$z = q^{-1}gq \qquad g = [0, 0, 0, 1]^T \tag{7}$$

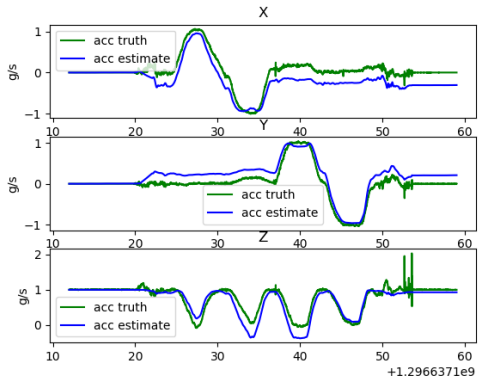

Figure 3: Acc measurement on dataset 2

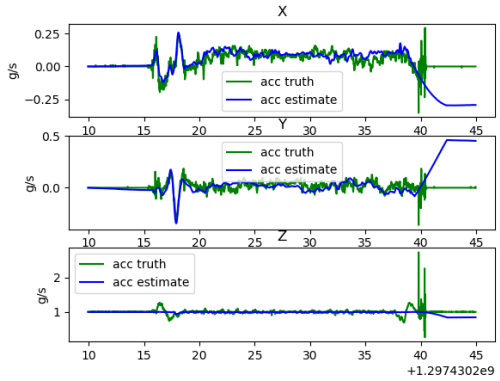

Figure 4: Acc measurement on dataset 8

4 Unscented Kalman Filter

Kalman Filter has numerous applications in technology, including sensor fusion. The basic Kalman Filter is limited to linear assumption, while complex system like this project is nonlinear with both process and measurement model.
Extend Kalman Filter (EKF) resolves this by linearizing the nonlinear transformation around mean. But the computation of jacobian could be expensive.
Unscented Kalman Filter (UKF) approximates the Gaussian distribution by sampling sigma points. Nonlinear transformation are applied on sigma points and forms a new distribution.
In this project, I implemented a UKF to gain robust and fast performance. I applied gyroscope data in process model as a control input and applied accelerometer data in measurement model. The state vector is orientation in quaternion.

Initialization

$$QuaternionMean: \quad q_0 = [1,0,0,0]^T \tag{8}$$
$$VectorCovariance: \quad P_0 \tag{9}$$
$$ProcessNoiseCovariance: \quad Q \tag{10}$$
$$MeasurementNoiseCovariance: \quad R \tag{11}$$

Compute Quaternion Sigma Points

$$S = \sqrt{P_k + Q} \tag{12}$$
$$W = \pm\sqrt{2n} * S \tag{13}$$
$$X = q_k q_W \tag{14}$$

Process Model

$$q\Delta = vec2quat(gyro\Delta t) \tag{15}$$
$$Y = q_X q\Delta \tag{16}$$
$$q_{priori} = mean(Y) \quad P_{priori} = cov(Y) \tag{17}$$

Measurement Model

$$Z = q_Y^{-1} g q_Y \tag{18}$$
$$z_k = mean(Z) \quad P_{zz} = cov(Z) \tag{19}$$
$$v_k = acc - z_k \tag{20}$$
$$P_{vv} = P_{zz} + R \tag{21}$$
$$P_{xz} = cov(Y - q_{priori}, Z - z_k) \tag{22}$$

Update

$$K_k = P_{xz} P_{vv}^{-1} \tag{23}$$
$$q_{posterior} = vec2quat(K_k v_k) q_{priori} \tag{24}$$
$$P_{posterior} = P_{priori} - K_k P_{vv} K_k^T \tag{25}$$

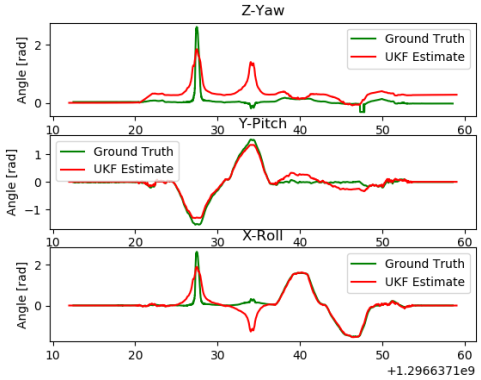

Figure 5: UKF estimate on dataset 2

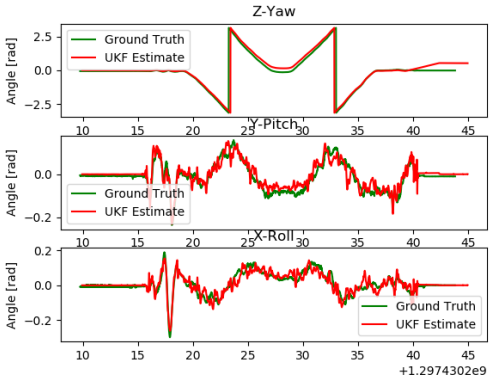

Figure 6: UKF estimate on dataset 8

5 Image Stitching

I made several assumptions before making panorama. The camera is pointing at +x axis with center at (1,0,0). The field of view is $\pi/3$ horizontally and $\pi/4$ vertically. Images are projected to a unit sphere surrounding camera. The panorama is designed to be 1920*960, because I decide to cover $[-\pi, \pi]$ about Z axis and $[-\pi/2, \pi/2]$ about Y axis and also the image size is 320*240 ($\pi/3$*$\pi/4$). The image stitching is achieved by following steps: First generate meshgrid for FOV in body frame in spherical coordinate and transform to cartesian. Second, find the closest orientation estimate by timestamps and rotate coordinates to global frame. Then, project back to spherical coordinates, use altitude/azimuth as y/x on a plane and scale up to pixels. Last, paint camera data to these pixels. The panorama video can be access at `https://www.youtube.com/watch?v=t8ejbLpqYuA` and `https://www.youtube.com/watch?v=4kOsHbubh1A`.

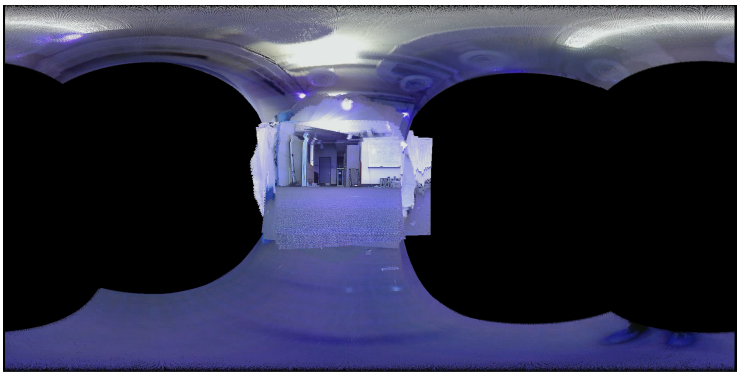

Figure 7: Panorama on dataset 2

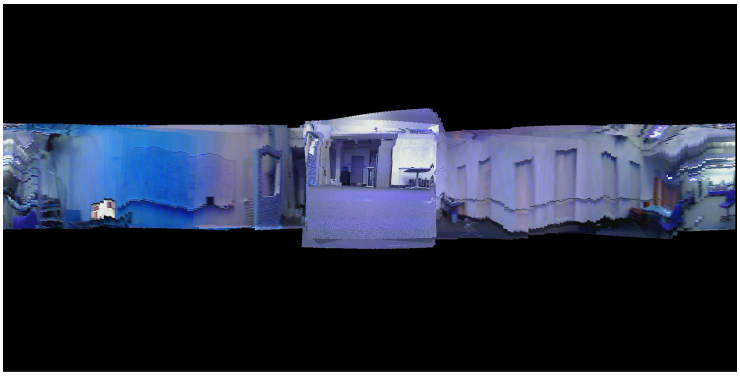

Figure 8: Panorama on dataset 8

6 Test Results

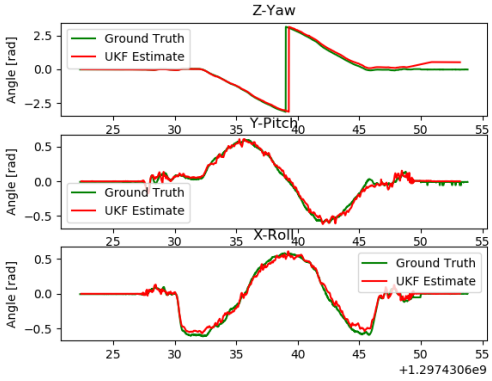

Figure 9: UKF estimate on dataset 10

Figure 10: Panorama on dataset 10

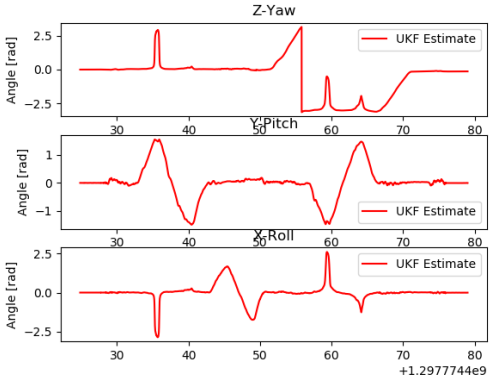

Figure 11: UKF estimate on dataset 10

Figure 12: Panorama on dataset 10