ELEC 5660: Introduction to Aerial Robotics
Project 2: Phase 3

Assigned: 04-09-2019 Due: 04-25-2019

1 Project Work

Note: This is the most difficult project, and is required to continue your Project 3. DO NOT think that you can finish it in one day unless you are very familiar with EKF.

In Project 2 Phase 3, you need to implement the extender Kalman filter on your lecture note 9. This is an individual project, which means you must complete it by yourself.

You will be provided with a ROS package named `ekf`, which is the skeleton code for the filter. You need to use it to implement this project fusing the odometry, velocity, height, and acceleration and angular velocity. Actually this project is the most difficult part in this lecture, you are supposed to follow the instructions below step by step.

1.1 Fuse the odometry with the measurement from IMU

For this part, please use the given launch file and the given bag in project 2 phase 3.
The things that you need to do in this part are described as follows.

1. Fuse the odometry obtained from your `tag_detector` with the measurement from IMU.
2. Publishing pose information in the form of `nav_msgs/Odometry`.
3. Plotting position and orientation after filter using **rqt_plot**.
4. Comparing your result with the odometry before filtering.

Some hints are listed as follows

1. The measurement from IMU is in the **body frame**(i.e. IMU's frame), but the pose estimation from your PnP solver is the pose of **world** in the **camera** frame. The important thing is that, this **world** frame's Z axis is pointing down. All frames here are right-hand frame.
2. If you can't understand the above sentences, you only need to know that finally you need calculate $\mathbf{R_{wi}}$ and $\mathbf{T_{wi}}$. In the tag_detector package, what you get is $\mathbf{R_{cw}}$ and $\mathbf{T_{cw}}$, and we provide $\mathbf{R_{ic}}$ and $\mathbf{T_{ic}}$(relative transformation between camera and IMU) in the code. Here $\mathbf{c}$ means camera, $\mathbf{i}$ means IMU and $\mathbf{w}$ means world. You don't need to know the absolute pose of each frame, all you need to care about is only **relative transformations between frames**.
3. Be careful about the discontinuous nature in Euler angles when you obtain the Roll, Pitch and Yaw angles from rotation matrix.

4. IMU data is in high frequency(400 Hz) and image data is in low frequency (20 Hz). The timestamp is already synchronized and you can read them from the repsecting message header.

5. The filter is not very sensitive to your initial value of covariance matrix. But please remember to set a right initial value for your ekf state. And you can try to set a proper value for the gravity(around 9.8 m/s^2 but may not be exact).

In the launch file, we half the play rate of the provided dataset in case your hard disk may stuck when play a big volume of data, you can change the rate in the launch file, the number behind "-r" is the dataset play rate. Figure 1 are some results plotted in rqt_plot, the filter is not fine tuned, but they are correct results.

1.2 Fuse the veclocity with the measurement from IMU and TFMini

For this part, please modify the `ekf` node to be able to fuse the veclocity obtained from your `optical_flow` only. The bag your should use are `simple.bag` and `difficult.bag` in the Project 2 Phase 1.

The things that you need to do in this part are described as follows.

1. Modify your `optical_flow` ros node to publish velocity and height information, and modify your `ekf` ros node to subscribe this topic. We suggest you also use `nav_msgs::Odometry` as the type of the topic.

2. Write a new roslaunch file to connect the specific bag file and your `optical_flow` and `ekf` ros node.

3. Fuse the veclocity you obtained from your `optical_flow` and also the height information with the measurement from IMU.

4. Publishing pose information in the form of `nav_msgs/Odometry`.

5. Plotting odometry and velocity from ground truth and odometry after kalman filter in **rviz**, like what we did in Project 2 Phase 2.

Some hints are listed as follows

1. The $\mathbf{R_{ic}}$ and $\mathbf{T_{ic}}$ need to be modifed, for Part II and Part III.

2. The dimention for the matrix Rt need to be modified to 9×9

3. You should fuse v_x, v_y from `optical_flow` and z directly from height of TFMini.

1.3 Fuse everything together

This part you should use the new bag, which will published later.

Modify your `ekf` node follows the following logic. **This part of code is the only code code that you need to submit for this project.**

1. If there is no height information from TFMini sensor, fuse the odometry obtained from your `tag_detector` with the measurement from IMU, as you implement in Section 1.1

2. If there is no ARMarker detected from images, fuse the velocity obtained from your `optical_flow` with the measurement from IMU, as you implement in Section 1.2

3. If both height information and AR_Marker are available, you should fuse all information together.

You are required to plot odometry and velocity from ground truth and odometry after kalman filter in **rviz**, like what we did in Section 1.2.

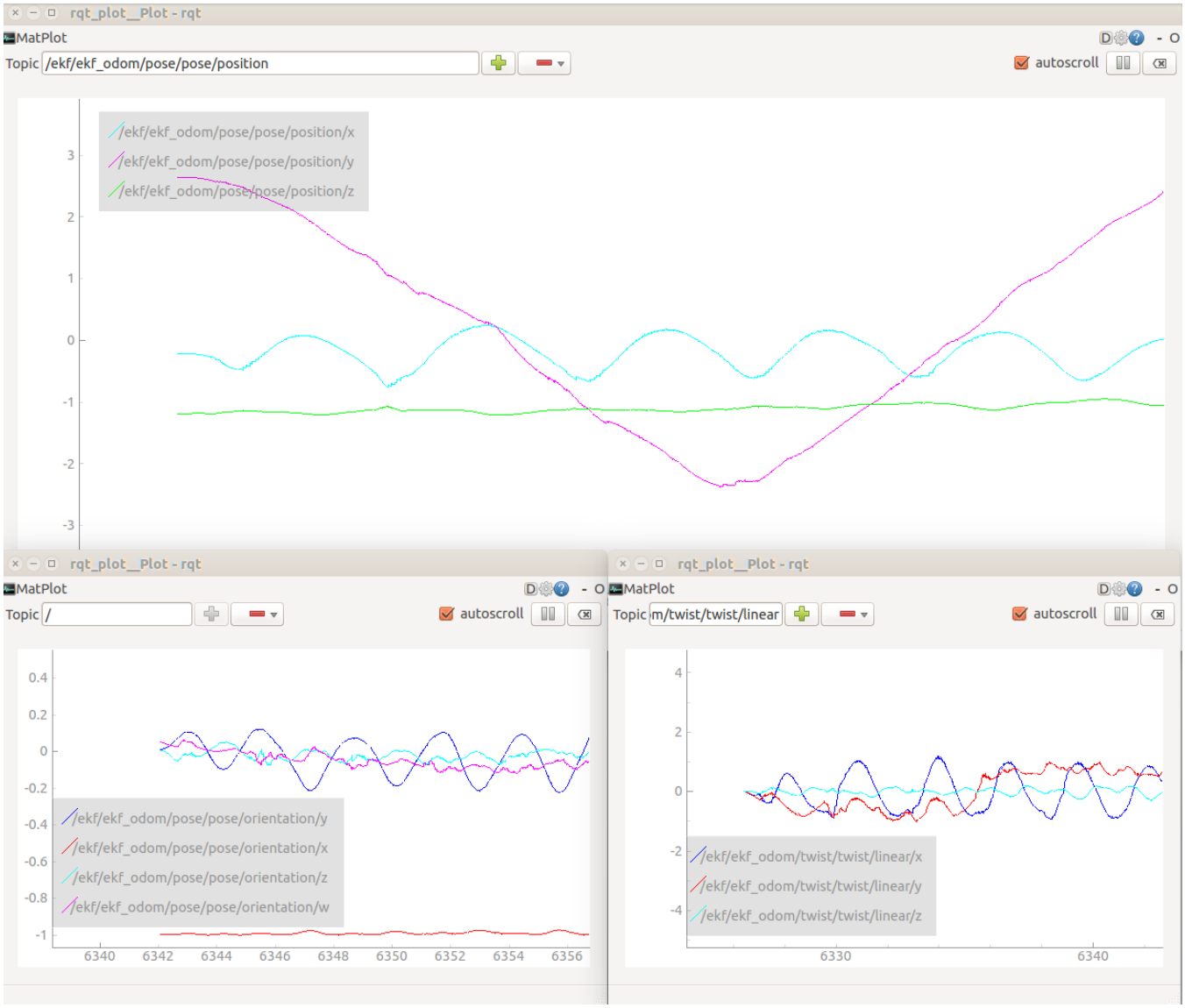

Figure 1: Position(above), orientation(left, down) and velocity(right, down)

2 Submission

When you complete the tasks you could submit your code and documents to canvas before **Apr. 25th, 2019 23:59:59**. The title of your submission should be "`proj2phase3_YOUR-NAMES`".

Your submission should contain:

1. A **maximum 2-page** document including:
   (a) Figures plotted by **rqt_plot** and **rviz**.
   (b) Descriptions about your implementation.
   (c) Any other things we should be aware of.
2. All files you need to run your code.