ECE-9253 Robot Localization and Navigation ME-7933 Fundamentals of Robot Mobility Project 2

Kshitij Jindal
New York University
kj1290@nyu.edu

Yining Wen
New York University
yw3997@nyu.edu

Tanay Varshney
New York University
tanay@nyu.edu

Seoho Kang
New York University
shk642@nyu.edu

I. Overview

The project tries to estimate position and orientation of the quadrotor using a vision based 3-D pose estimator with an AprilTags map.

II. Sensor Data

The second project just uses the camera data as the input. However we do have access to IMU and Vicon data to compare our results

III. Approach

A. Vision Based Pose Estimation

1) Environment: The data for this phase was collected using a quadrotor that was either held by hand or flown through a prescribed trajectory over a mat of AprilTags.

Fig 1 shows the layout of the AprilTag mat. The tags are arranged in a 12 x 9 grid. The top left corner of the top left tag should be used as coordinate (0, 0) with the X coordinate going down the mat and the Y coordinate going to the right. Each tag is a 0.152 m square with 0.152 m between tags with the exception of the space between columns 3 and 4, and 6 and 7, which is 0.178 m. Using this information we compute the location of every corner of every tag in the world frame.

2) Calibration: The intrinsic camera calibration matrix and the transformation between the camera and the robot center are given We transform your camera-based pose estimate from the camera frame to the robot frame.

3) Pose Estimation: The data contains the following:

- Time stamp (t) in seconds.
- ID of every AprilTag that is observed in the image (id).
- The center (p0) and four corners of every AprilTag in the image. The corners are given in the order bottom left (p1), bottom right (p2), top right (p3), and top left (p4), see Figure 2
- Rectified image (img).
- IMU data with Euler angles (rpy), angular velocity (omg), and linear acceleration (acc)

Following are the equations needed to compute pose of the quadrotor

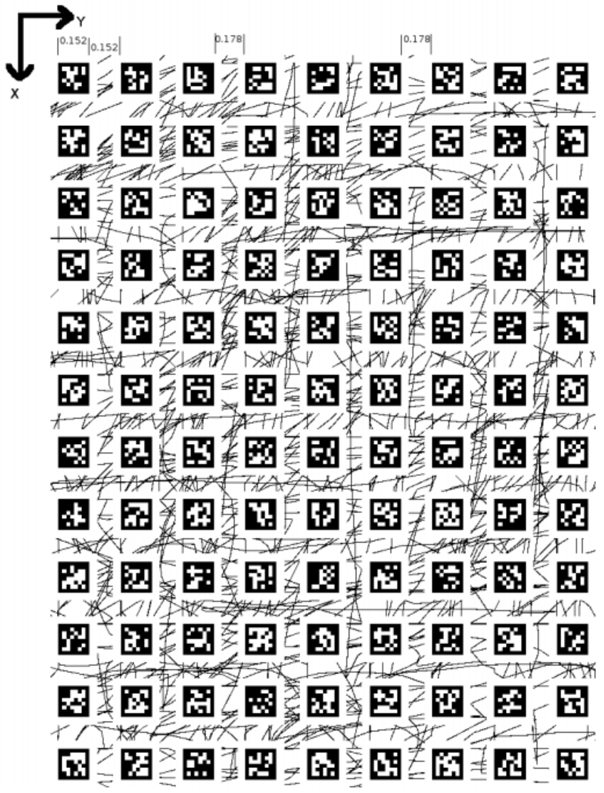

Fig. 1. Map of tags

$$\lambda_i * \begin{bmatrix} \tilde{x_i} \\ \tilde{y_i} \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} * \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -\tilde{x_i}x_i & -\tilde{x_i}y_i & -\tilde{x_i} \\ 0 & 0 & 0 & x_i & y_i & 1 & -\tilde{y_i}x_i & -\tilde{y_i}y_i & -\tilde{y_i} \end{bmatrix} * h = 0$$

$$Ah = 0$$

$$A = USV^T$$

$$h = V_9$$

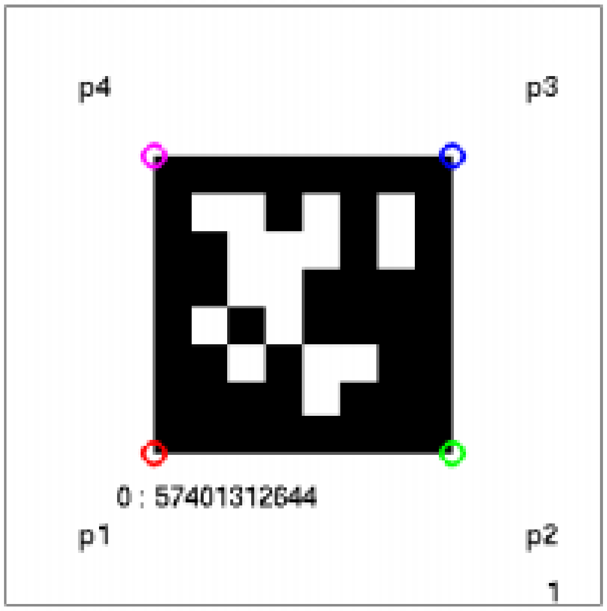

Fig. 2. Tags

$$\begin{bmatrix}\hat{R_1} & \hat{R_2} & \hat{T}\end{bmatrix} = \begin{bmatrix}\hat{r_{11}} & \hat{r_{12}} & \hat{t_1} \\ \hat{r_{21}} & \hat{r_{22}} & \hat{t_2} \\ \hat{r_{31}} & \hat{r_{32}} & \hat{t_3}\end{bmatrix} =$$

$$\begin{bmatrix}f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1\end{bmatrix} \text{^-1*} \begin{bmatrix}h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33}\end{bmatrix}$$

$$\begin{bmatrix}\hat{R_1} & \hat{R_2} & \hat{R_1} \times \hat{R_2}\end{bmatrix} = USV^T$$

$$R = U * \begin{bmatrix}1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(UV^T)\end{bmatrix} * V^T$$

$$T = \hat{T}/||\hat{R_1}||$$

We now convert the matrices to the required frame.

B. Angular and Linear Velocity

We extracted corners in each image using FastFeatures feature detector. We assume that all detected corners will be on the ground plane. For all corners in the image, we compute the sparse optical flow between two consecutive images using the KLT tracker. in the calibrated image frame. We then use the optical flow values and estimate the Linear and Angular Velocities. A RANSAC based outlier rejecter is then used to reject extreme outling features. We then apply a low pass filter to smoothen out the jitter.

C. Velocity Estimation

p = 1 / z A(p) V + B(p) Ω

With known depth the equation reduces to below.

$$\begin{bmatrix}u \\ v\end{bmatrix} = \begin{bmatrix}-1/z & 0 & x/z \\ 0 & -1/z & y/z\end{bmatrix} \begin{bmatrix}v_x \\ v_y \\ v_z\end{bmatrix} +$$

$$\begin{bmatrix}xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x\end{bmatrix} \begin{bmatrix}\Omega_x \\ \Omega_y \\ \Omega_z\end{bmatrix} =$$

$$\begin{bmatrix}-1/z & 0 & x/z & xy & -(1+x^2) & y \\ 0 & -1/z & y/z & 1+y^2 & -xy & -x\end{bmatrix} \begin{bmatrix}v_x \\ v_y \\ v_z \\ \Omega_x \\ \Omega_y \\ \Omega_z\end{bmatrix}$$

$$\text{V^*,}\Omega^* = argmin_{V,\Omega} \sum_{i=1}^{n} ||1/Z_i A(p_i) B(p_i)) \begin{bmatrix}V \\ \Omega\end{bmatrix} - pi'||^2$$

dx = [x - previous_x]'./dt dy = [y - previous_y]'./dt

For the known depth, we can simply use least squares.

- We should find [p0, p1, p2, p3, p4] w.r.t world frame using given id of the data.
- Using the [p0, p1, p2, p3, p4] w.r.t world frame, [p0, p1, p2, p3, p4] w.r.t camera frame and inverse of K value we can derive wRc and wTc.
- We can estimate z w.r.t camera frame which is depth using wRc, wTc, corners, and inverse of K.
- we find optical flow value by equation (x - prev_x)/dt

1) RANSAC-Based Outlier Rejection:

1) RANSAC-Based Outlier Rejection: Ransac is basically outlier detection model. By sampling and fitting minimal error models, we can avoid iteration growing too fast.

Repeat for k iterations

1) Choose a minimal sample set

2) Count the inliers for this set

3) Keep maximum , if it exceeds a desired number of inliers

stop

$$p_success = 1 - (1 - \epsilon^M)^k => k = \frac{log(1 - p_success}{log(1 - \epsilon^M)}$$

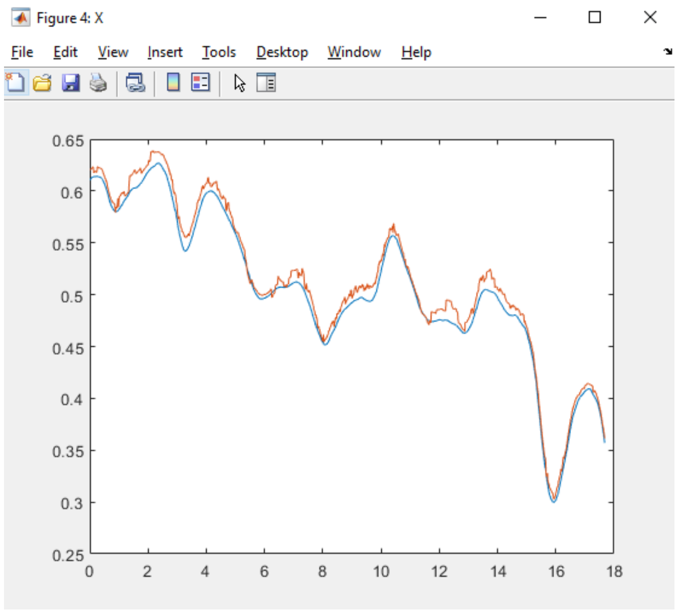

Fig. 3. X

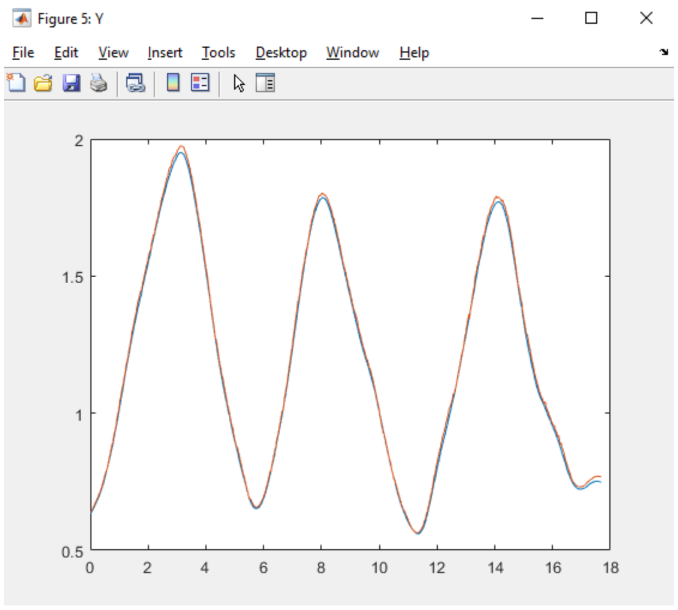

Fig. 4. Y

D. *EKF*

This project, re-uses the Extended Kalman Filter (EKF) developed in project 1 to estimate the position, velocity, and orientation, and sensor biases of the vehicle.

It uses the filter from the project 1. In place of the vicon data we use the data obtained from camera pose.

IV. Results

The following results are on the file "StudentData1.mat". For all the results, blue lines are Vicon Data and Orange lines are our output.

A. *Pose Estimation*

Following are the results for Pose Estimation

- X see Figure 3
- Y see Figure 4
- Z see Figure 5
- roll see Figure 6
- pitch see Figure 7
- yaw see Figure 8

B. *Angular and Linear Velocity Estimation*

Following are the results for Pose Estimation

- Vx see Figure 9

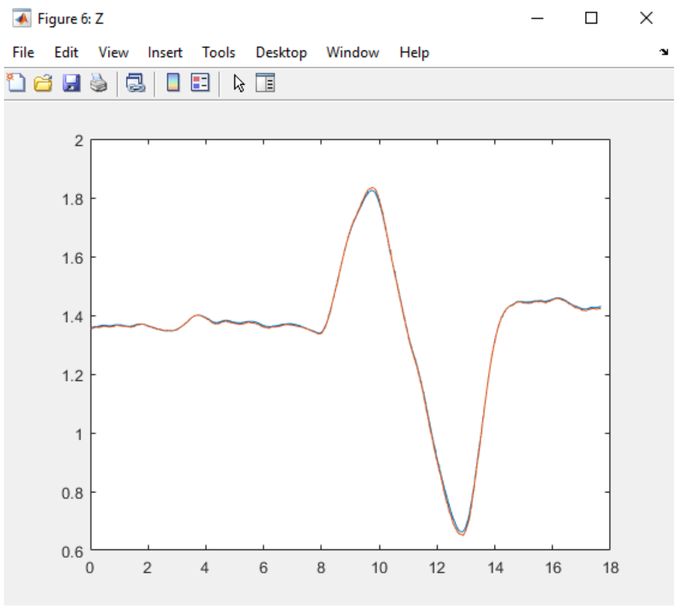

Fig. 5. Z

- Vy see Figure 10
- Vz see Figure 11
- Wx see Figure 12
- Wy see Figure 13
- Wz see Figure 14

C. *EKF filter 1*

We are taking in the both the Pose and the velocities to compute the EKF. Following are the results for Pose

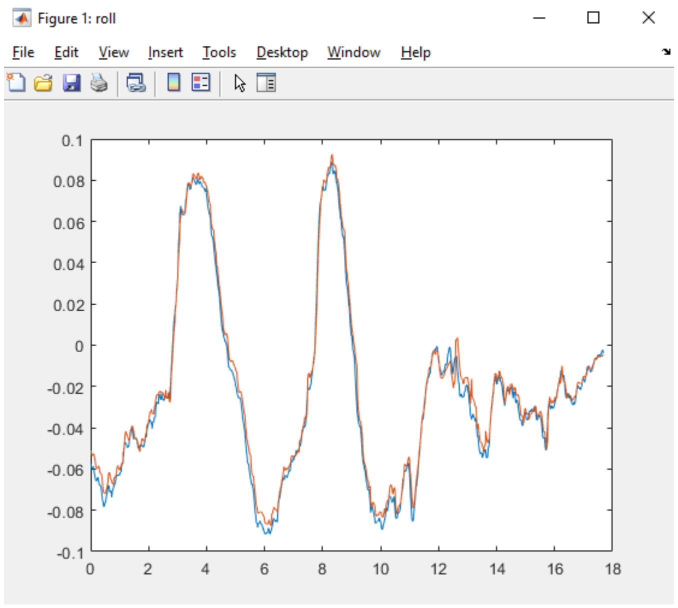

Fig. 6. Roll

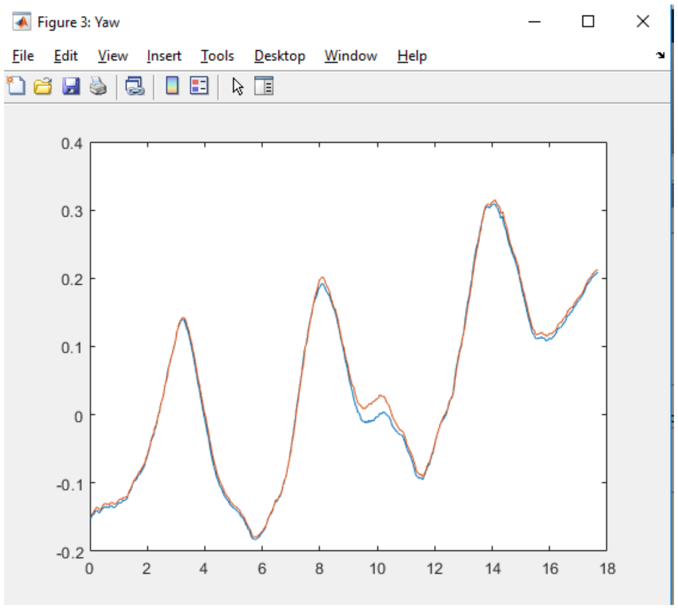

Fig. 8. yaw

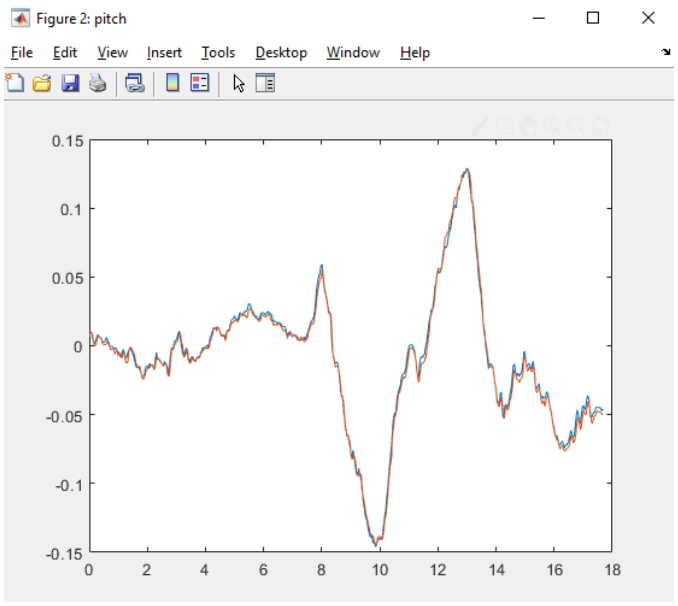

Fig. 7. pitch

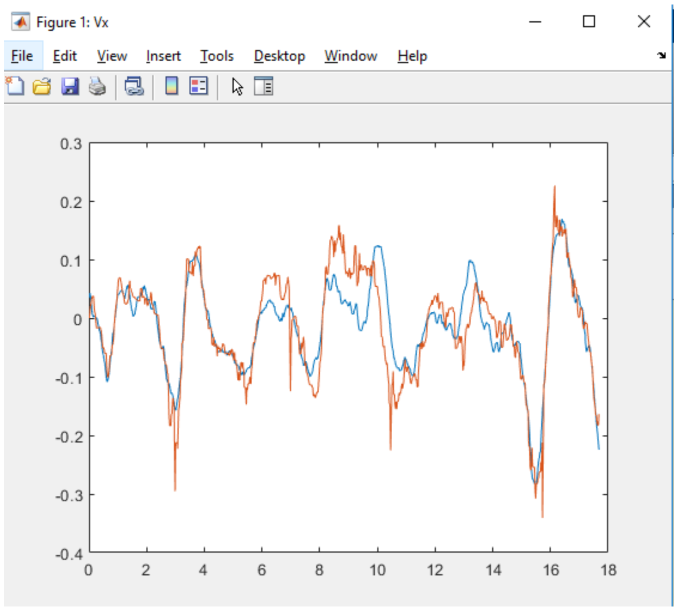

Fig. 9. Vx

Estimation

- EKF X see Figure 15
- EKF Y see Figure 16
- EKF Z see Figure 17
- EKF roll see Figure 18
- EKF pitch see Figure 19
- EKF yaw see Figure 20

D. EKF filter 2

We are taking in just the velocities to compute the EKF.

- EKF X see Figure 21
- EKF Y see Figure 22
- EKF Z see Figure 23
- EKF roll see Figure 24
- EKF pitch see Figure 25
- EKF yaw see Figure 26

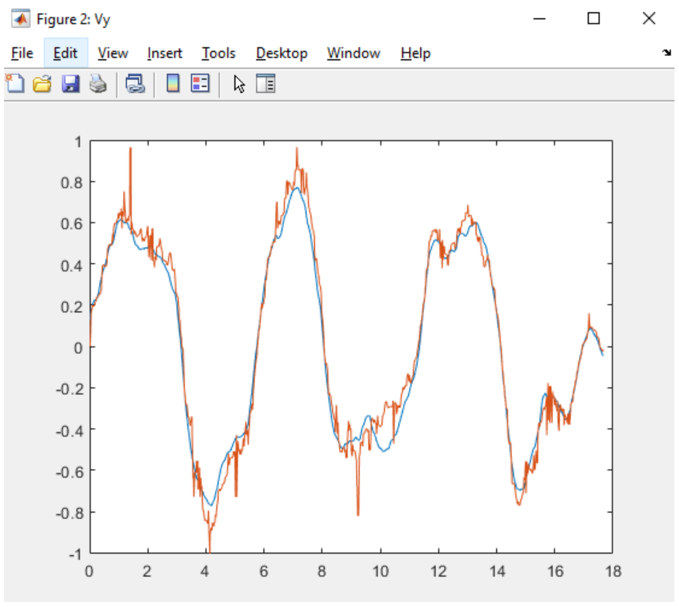

Fig. 10. Vy

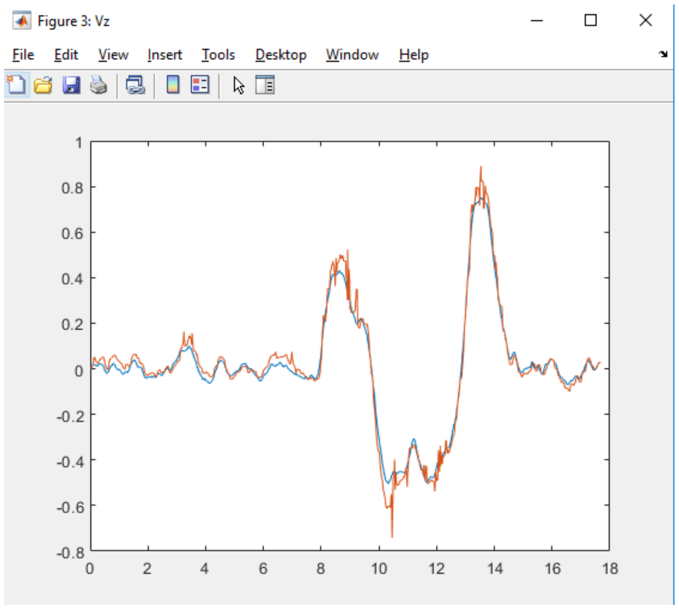

Fig. 11. Vz

V. Conclusion

This project aims to demonstrate the knowledge of pose estimation and velocity estimation by using camera as the sole input. The goal was to effectively use april tags to do the same.

EKF filter was used to estimate the position and velocity with the input coming in from the camera.

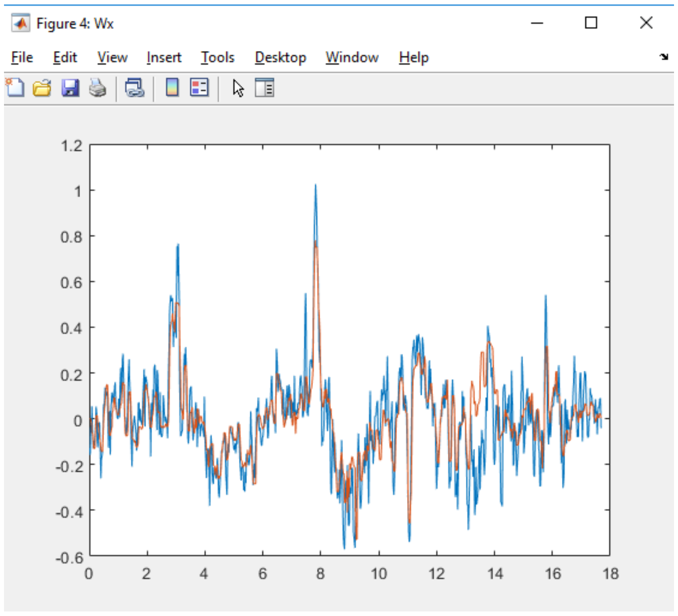

Fig. 12. Wx

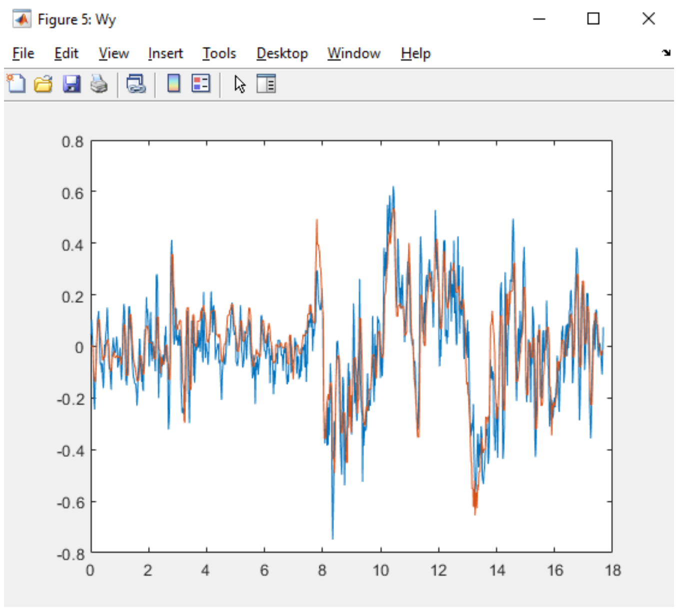

Fig. 13. Wy

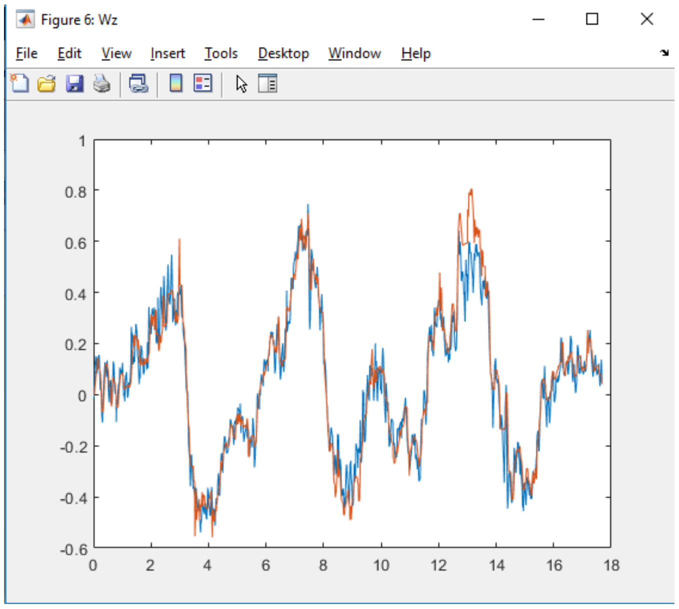

Fig. 14. Wz

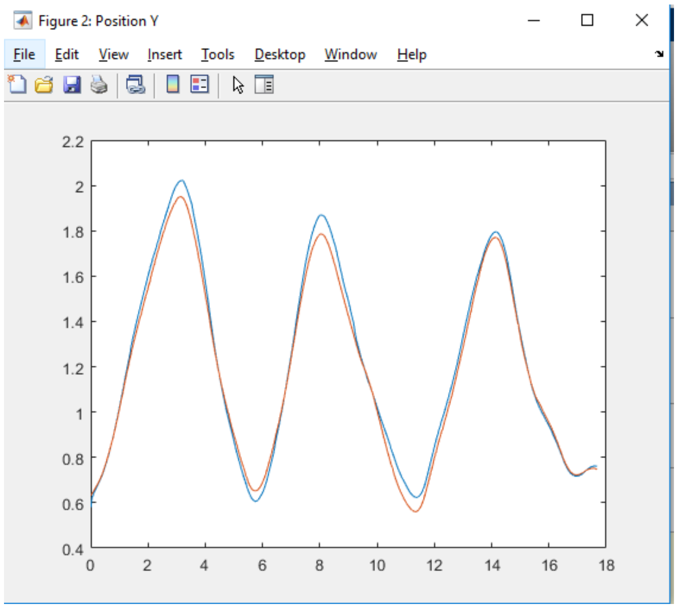

Fig. 16. EKF Y

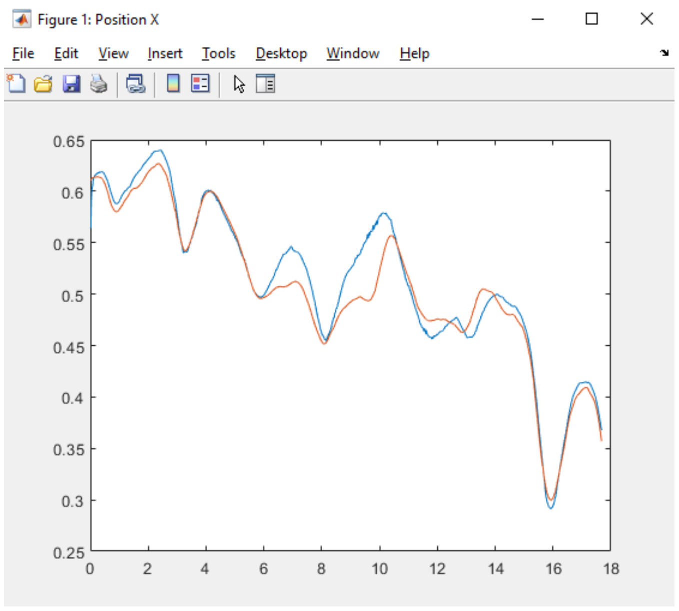

Fig. 15. EKF X

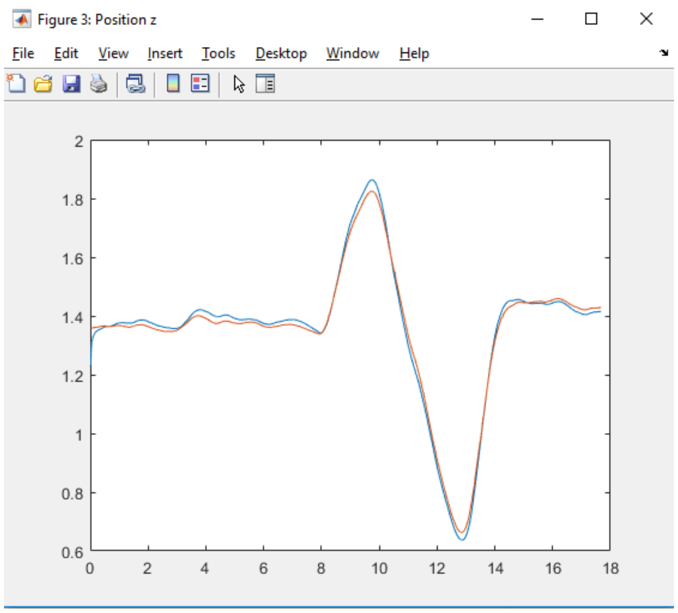

Fig. 17. EKF Z

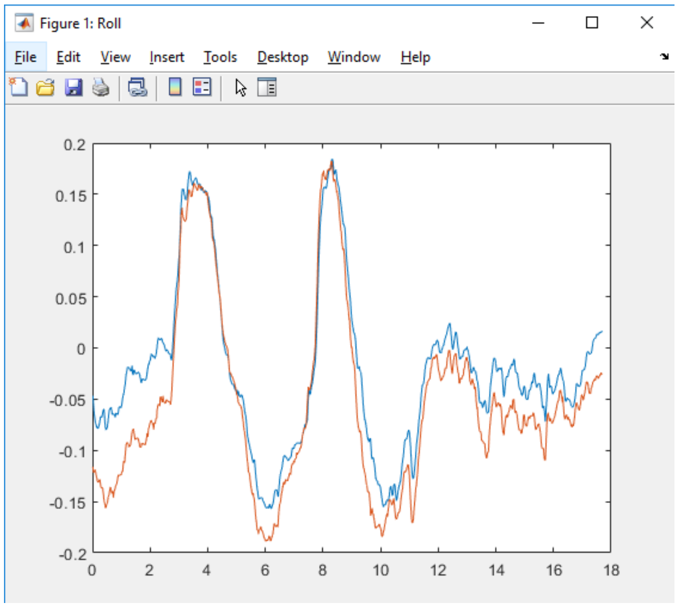

Fig. 18. EKF Roll

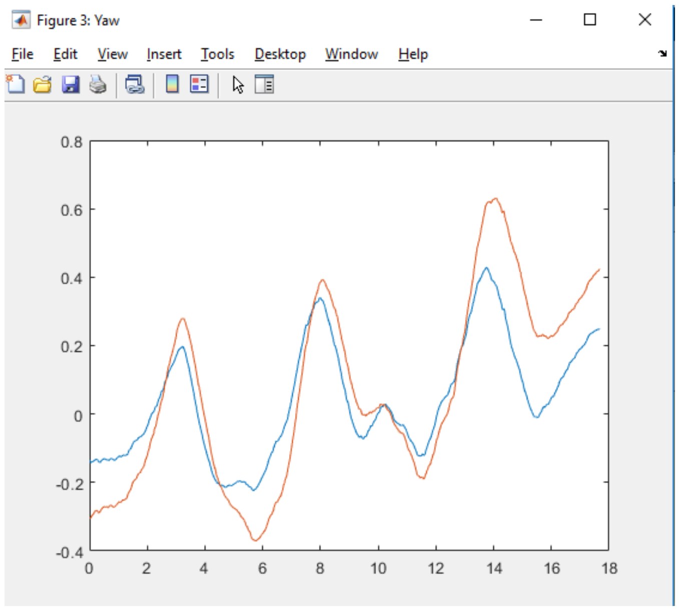

Fig. 20. EKF yaw

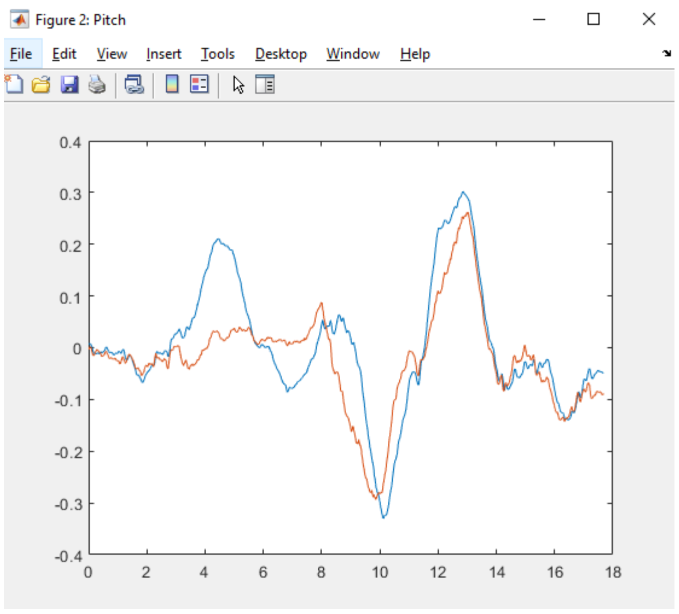

Fig. 19. EKF pitch

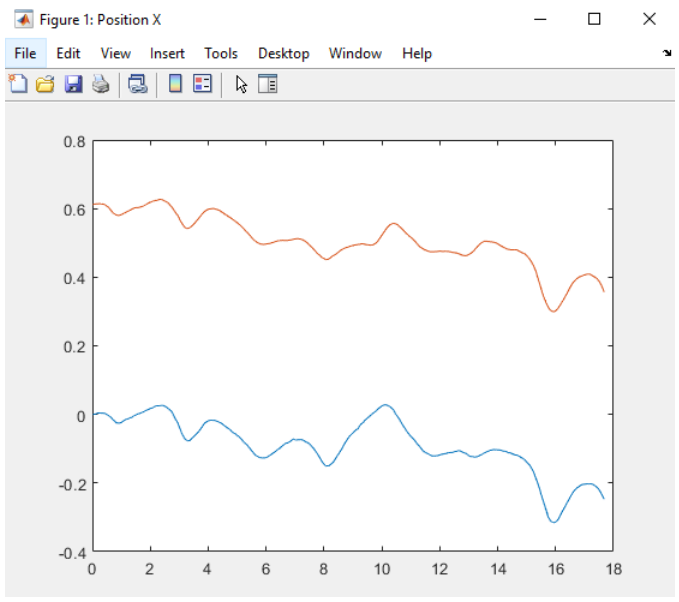

Fig. 21. EKF X

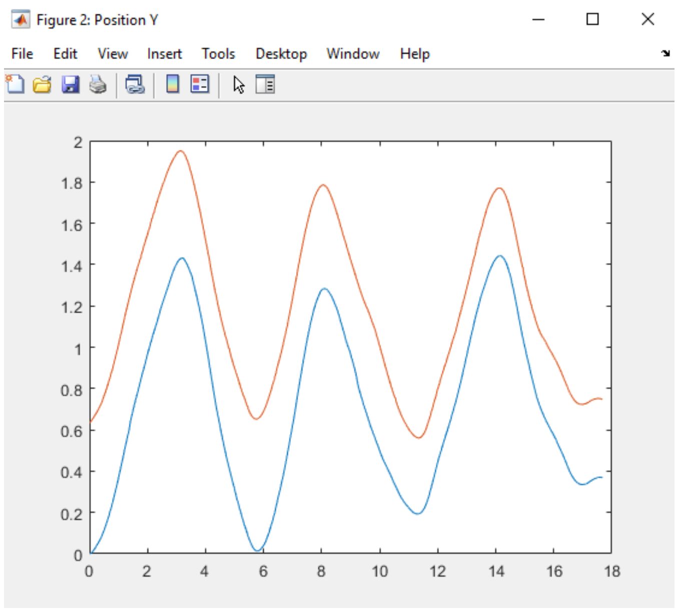

Fig. 22. EKF Y

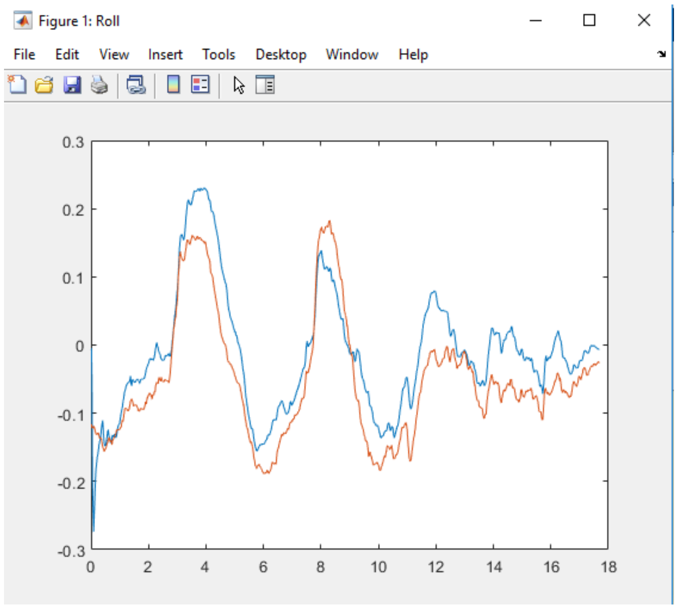

Fig. 24. EKF Roll

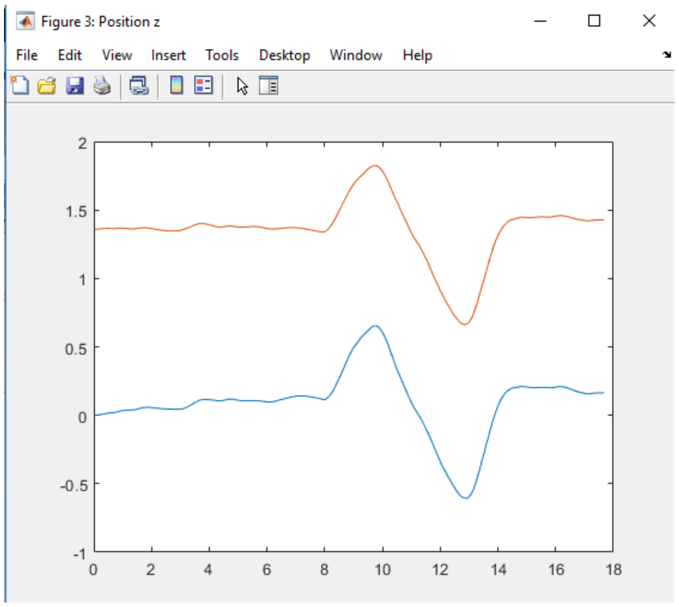

Fig. 23. EKF Z

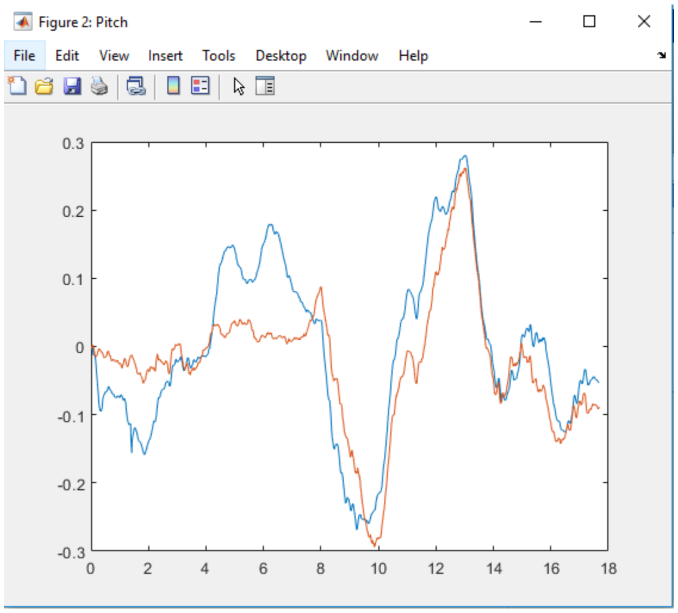

Fig. 25. EKF pitch

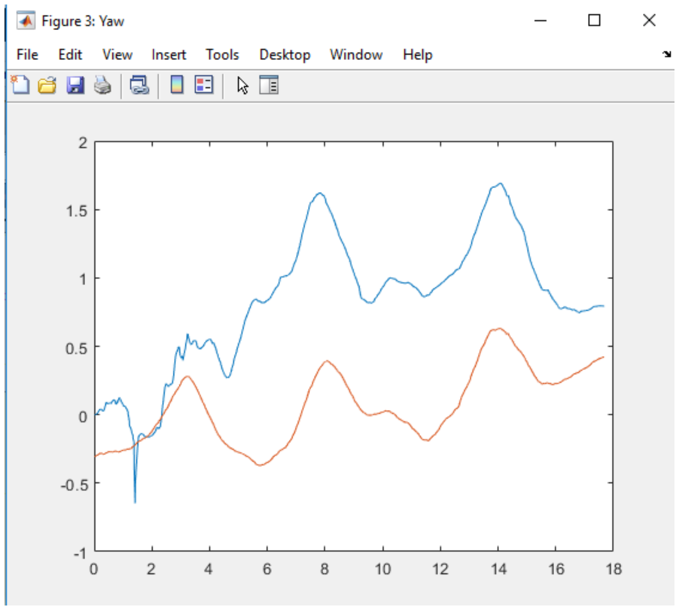

Fig. 26. EKF yaw