Project 3: Non-Linear Kalman Filter
RBE 595: Advanced Robot Navigation

Miheer Diwan
Robotics Engineering
Worcester Polytechnic Institute
msdiwan@wpi.edu

I. Introduction

In this project, we will implement a nonlinear Kalman Filter, specifically the Extended Kalman Filter (EKF), to estimate the pose of a quadcopter drone. Pose estimation is a critical task in robotics, enabling precise control and navigation of aerial vehicles. We will leverage sensor data from AprilTags and motion capture systems to improve the accuracy of pose estimation.

II. Task 1: Pose Estimation

To estimate the pose of the quadcopter drone, we will employ the Perspective-n-Point (PnP) problem-solving approach. This method utilizes the AprilTag corners from the map layout as 3D points in the world and their corresponding 2D projections in the image plane captured by the camera.

The camera calibration matrix and distortion coefficients are given as follows:

$$\text{Camera Matrix} = \begin{bmatrix} 314.1779 & 0 & 199.4848 \\ 0 & 314.2218 & 113.7838 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Distortion Coefficients} = \begin{bmatrix} -0.438607 & 0.248625 & 0.00072 & -0.000476 & -0.0911 \end{bmatrix}$$

The camera model gives the realtion between the homogenous world coordinates and the homogenous image coordinates and is represented as:

$$\mathbf{M} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

After obtaining the orientation data in the form of a rotation vector, we will convert it to a rotation matrix. Using this rotation matrix, we can then extract the three Euler angles using the provided matrix transformation formula:

$$\begin{aligned} \phi &= \text{atan2}(R_{32}, R_{33}) \\ \theta &= \text{atan2}(-R_{31}, \sqrt{R_{32}^2 + R_{33}^2}) \\ \psi &= \text{atan2}(R_{21}, R_{11}) \end{aligned}$$

With this, we can obtain the transformation matrix from the camera frame to the world frame. Additionally, the provided information for the yaw rotated with $\pi/4$ and the roll of π is used for get the transformation from the camera frame to the IMU frame.

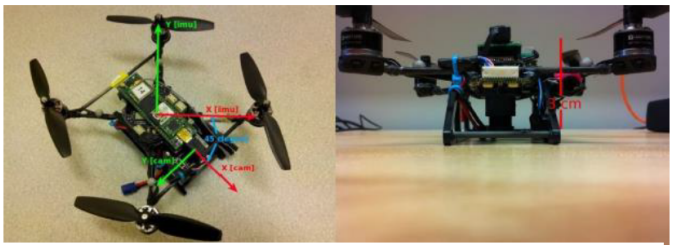


Fig. 1. Drone Configuration

III. Task 2: Visualization

We will visualize the estimated trajectory of the quadcopter by plotting it against ground truth motion capture data. We will visualize the roll, pitch, and yaw angles over time to assess the stability and consistency of the pose estimates. The time stamps have been aligned before plotting as the rate of data collection for the IMU and the Vicon system is different.

A. Dataset 0

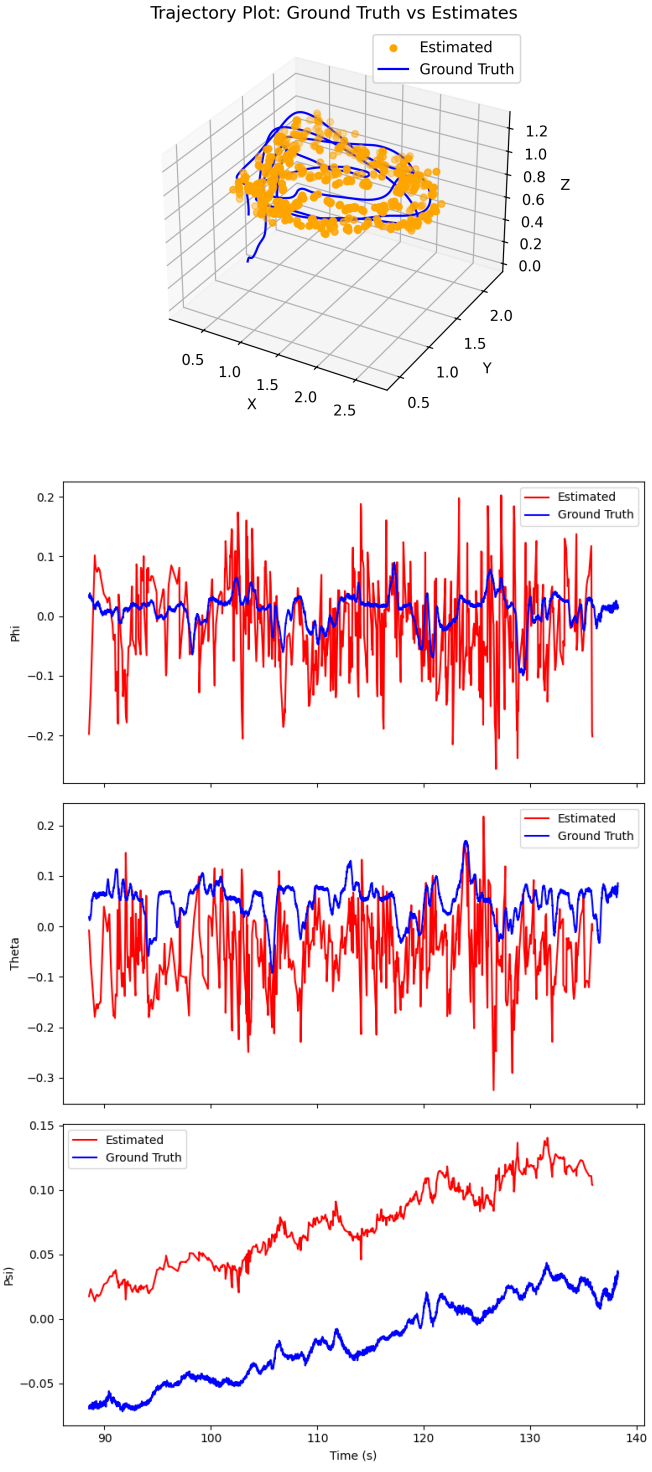


Fig. 2. Plots for Dataset 0

B. Dataset 1

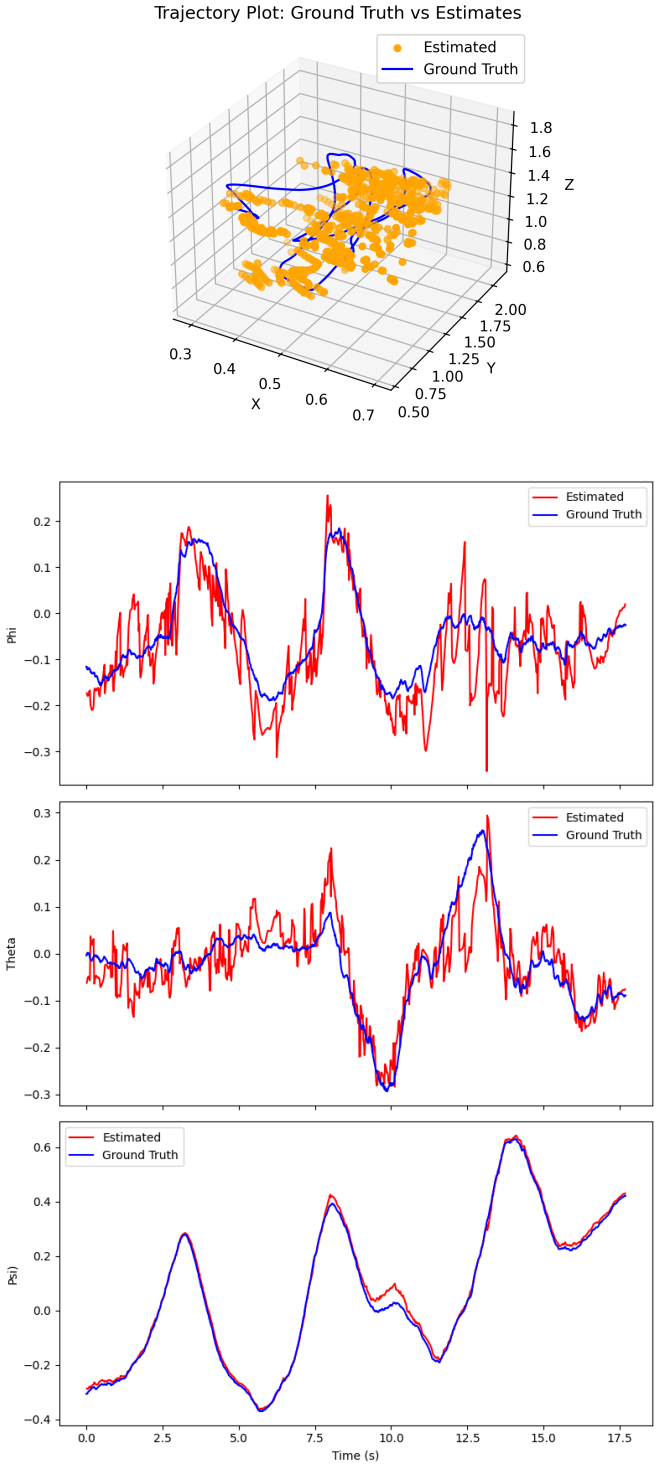


Fig. 3. Plots for Dataset 1

C. Dataset 2

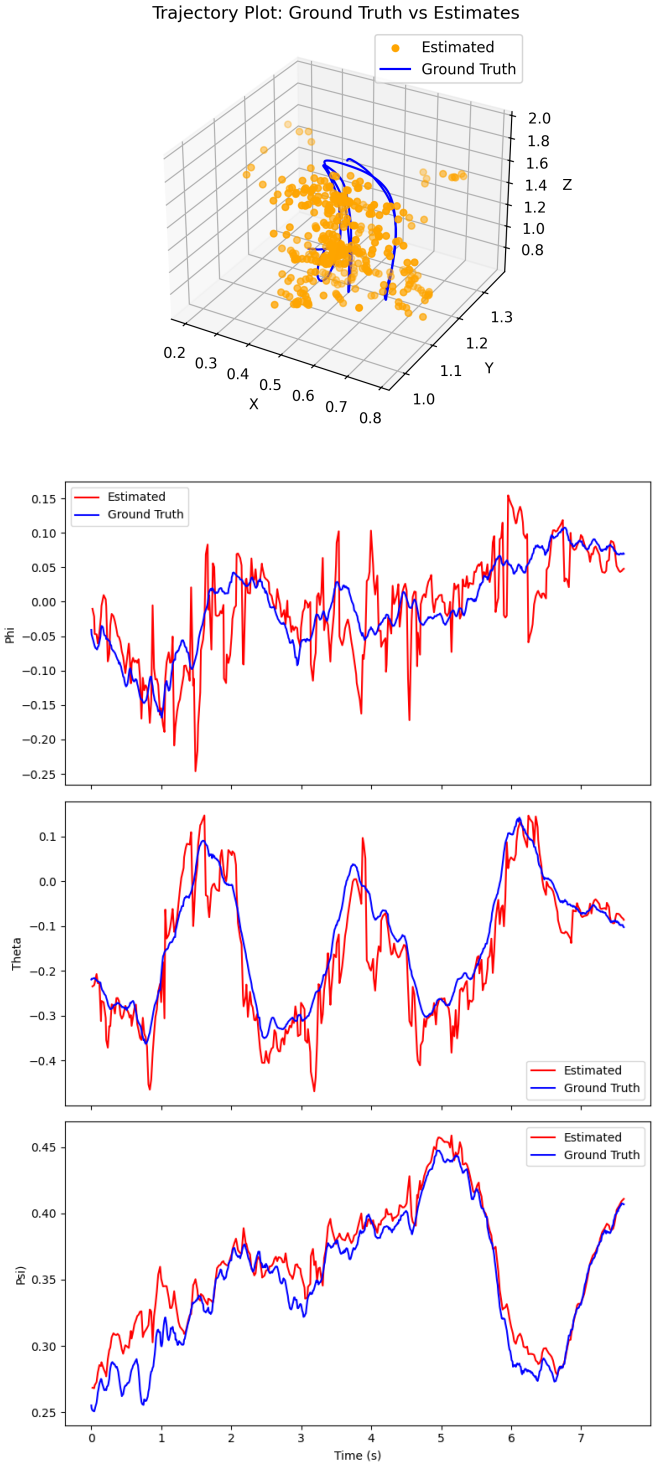


Fig. 4. Plots for Dataset 2

D. Dataset 3

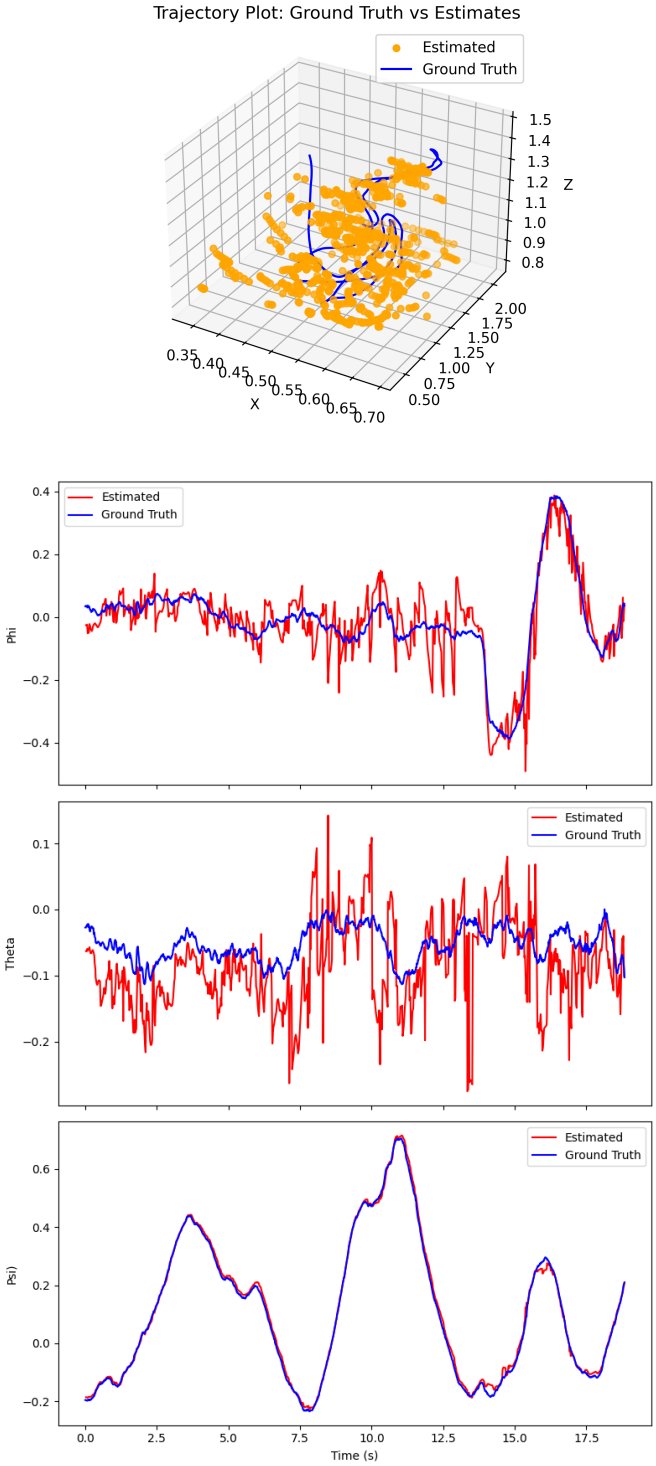


Fig. 5. Plots for Dataset 3

E. Dataset 4

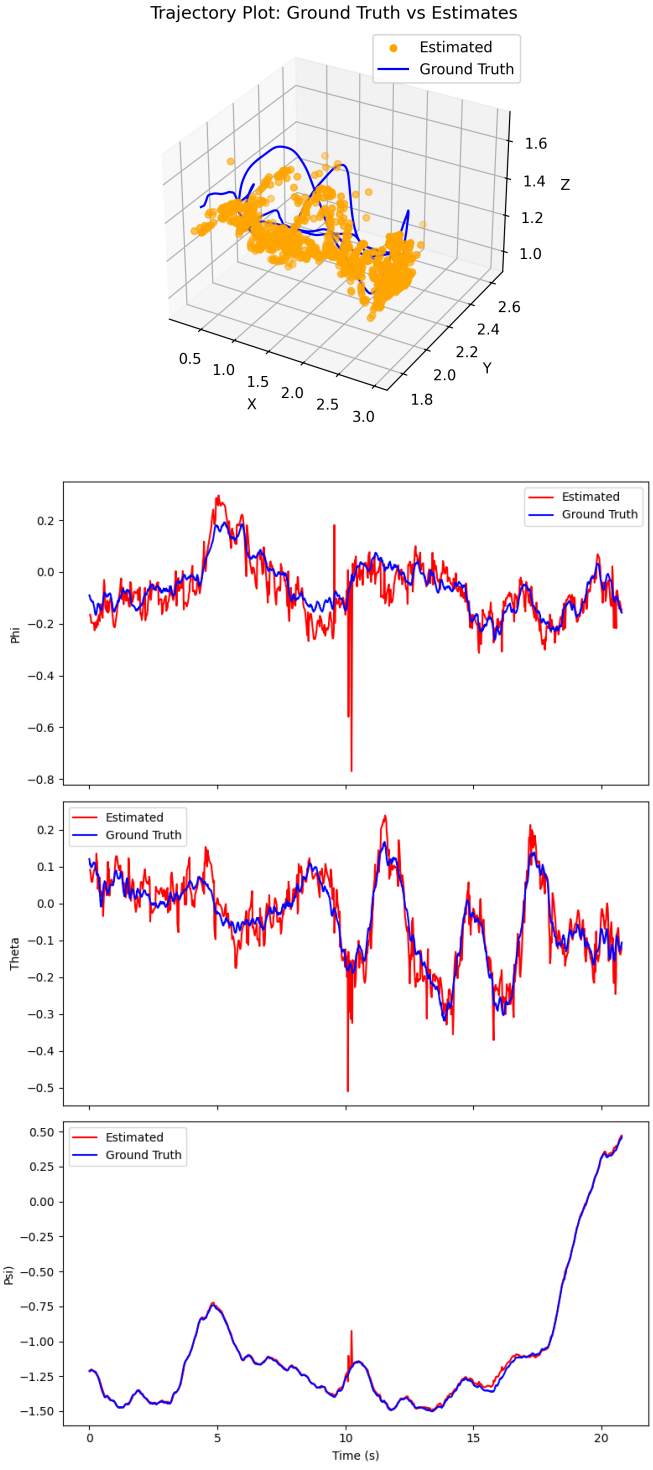


Fig. 6. Plots for Dataset 4

F. Dataset 5

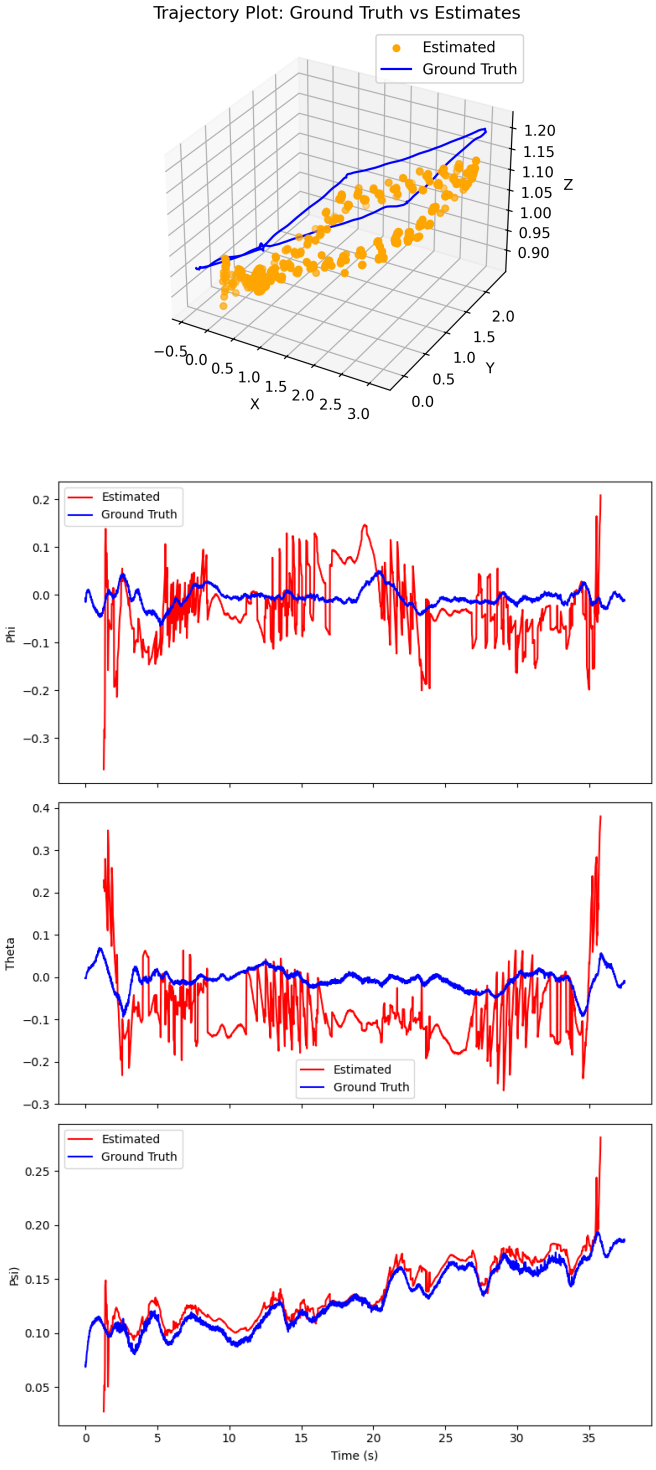


Fig. 7. Plots for Dataset 5

G. Dataset 6

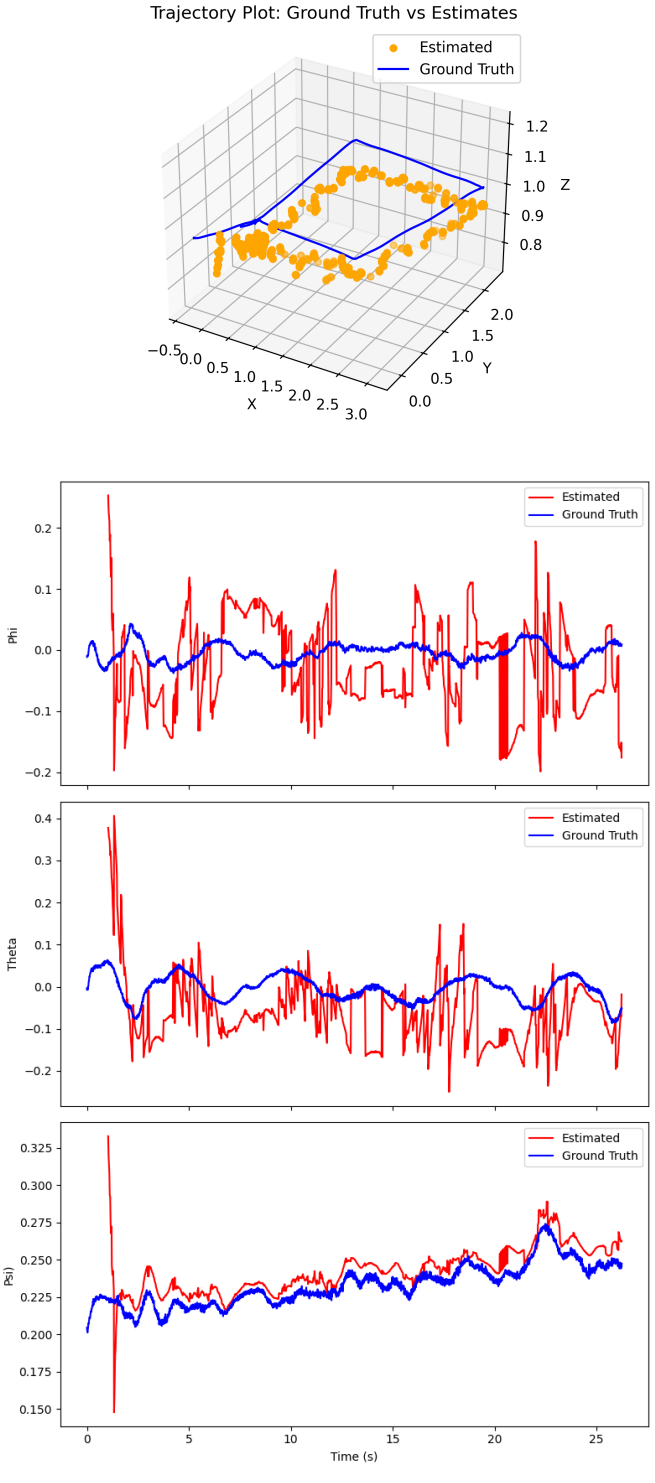


Fig. 8. Plots for Dataset 6

H. Dataset 7

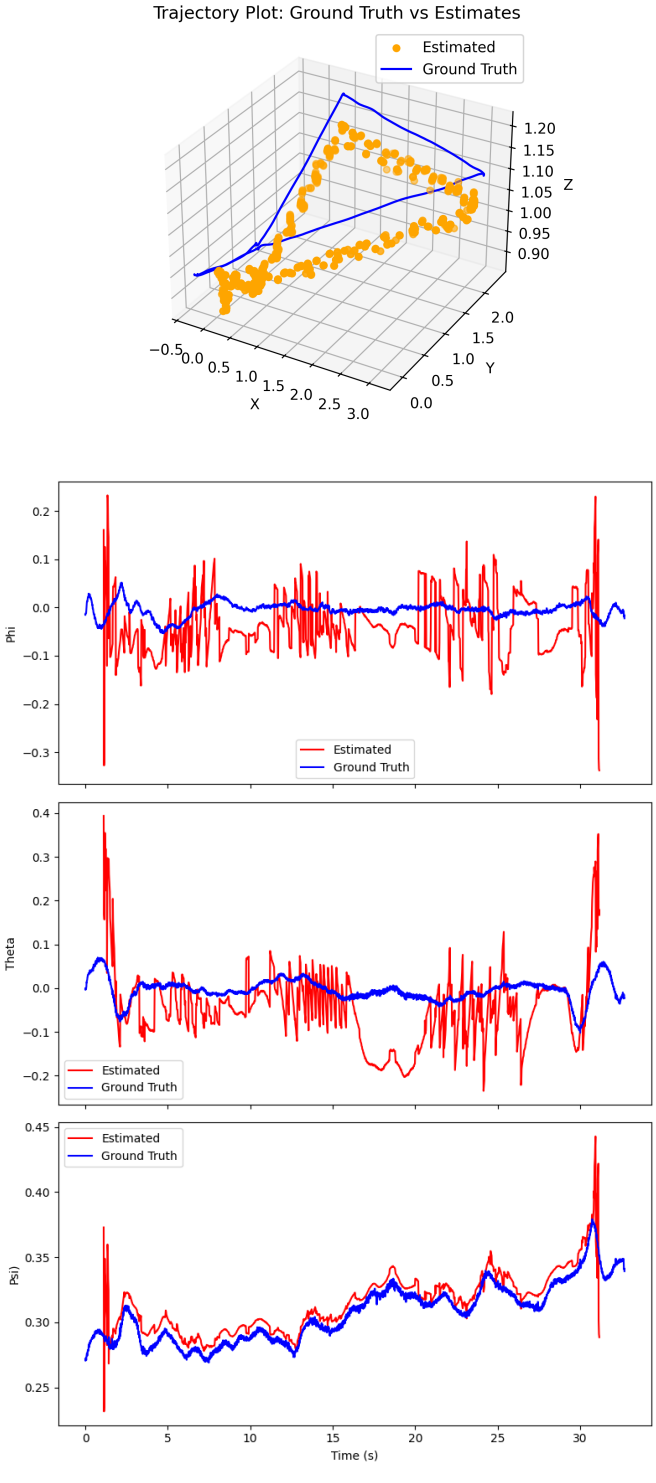


Fig. 9. Plots for Dataset 7

IV. Task 3: Covariance Estimation

To characterize the uncertainty in sensor measurements, we will compute the covariance matrix for the observation model. For this task, the goal is to estimate the covariance matrix in the observation model: $\mathcal{N}(0, \mathbf{R}) \rightarrow \mathbf{R}$. We will assume that the noise is zero-mean. Next, we can use the formula for sample covariance:

$$\mathbf{R} = \frac{1}{n-1} \sum_{t=1}^{n} \boldsymbol{\nu}_t \boldsymbol{\nu}_t^T$$

To get the value of $\boldsymbol{\nu}_t$ at time t, rearrange the observation model to solve for $\boldsymbol{\nu}_t$. For the state, we use the ground truth data.

$$\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} + \boldsymbol{\nu}$$

The average covariance R for our model is given in the README file.

V. Task 4: EKF Implementation

A. EKF Algorithm

- Prediction:

$$\hat{\mu}_t = \mu_{t-1} + \delta_t f(\mu_{t-1}, u_t, 0)$$
$$\hat{\Sigma}_t = F_t \Sigma_{t-1} F_t^T + V_t Q_t V_t^T$$

- Measurement update:

$$K = \hat{\Sigma}_t G_t^T (G_t \hat{\Sigma}_t G_t^T + W_t R_t W_t^T)^{-1}$$
$$\mu_t = \hat{\mu}_t + K_t(z_t - g(\hat{\mu}_t, 0))$$
$$\Sigma_t = \hat{\Sigma}_t - K_t G_t \hat{\Sigma}_t$$

B. State-Space representation

We use a 15-state model for this project. The system state is represented by the vector $\mathbf{x}$ and is given by:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b_g} \\ \mathbf{b_a} \end{bmatrix}$$

where:

- $\mathbf{p}$ is the 3x1 position vector.
- $\mathbf{q}$ is the 3x1 orientation vector.
- $\dot{\mathbf{p}}$ is the 3x1 velocity vector (time derivative of position).
- $\mathbf{b_g}$ is the 3x1 gyroscope bias.
- $\mathbf{b_a}$ is the 3x1 accelerometer bias.

The input vector to the system is given by:

$$\mathbf{u} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ \ddot{p}_x \\ \ddot{p}_y \\ \ddot{p}_z \end{bmatrix}$$

The continuous-time state-space model is given by:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \mathbf{G}(\mathbf{q})^{-1} \mathbf{u}_\omega \\ \mathbf{g} + \mathbf{R}(\mathbf{q}) \mathbf{u}_a \\ \mathbf{n}_b^g \\ \mathbf{n}_b^a \end{bmatrix}$$

C. Process Model

The Jacobian of the system is given by:

$$\mathbf{A} = \begin{bmatrix} \frac{\partial \dot{p}}{\partial p} & \frac{\partial \dot{p}}{\partial q} & \frac{\partial \dot{p}}{\partial \dot{p}} & \frac{\partial \dot{p}}{\partial b_g} & \frac{\partial \dot{p}}{\partial b_a} \\ \frac{\partial \dot{q}}{\partial p} & \frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial \dot{p}} & \frac{\partial \dot{q}}{\partial b_g} & \frac{\partial \dot{q}}{\partial b_a} \\ \frac{\partial \ddot{p}}{\partial p} & \frac{\partial \ddot{p}}{\partial q} & \frac{\partial \ddot{p}}{\partial \dot{p}} & \frac{\partial \ddot{p}}{\partial b_g} & \frac{\partial \ddot{p}}{\partial b_a} \\ \frac{\partial \dot{b}_g}{\partial p} & \frac{\partial \dot{b}_g}{\partial q} & \frac{\partial \dot{b}_g}{\partial \dot{p}} & \frac{\partial \dot{b}_g}{\partial b_g} & \frac{\partial \dot{b}_g}{\partial b_a} \\ \frac{\partial \dot{b}_a}{\partial p} & \frac{\partial \dot{b}_a}{\partial q} & \frac{\partial \dot{b}_a}{\partial \dot{p}} & \frac{\partial \dot{b}_a}{\partial b_g} & \frac{\partial \dot{b}_a}{\partial b_a} \end{bmatrix}$$

We can discretize the continuous-time model using the following formulas:

$$\mathbf{F} = (I + \mathbf{A}\Delta t)$$

The $\mathbf{Q}$ covariance matrix is defined as a diagonal matrix with a mean value of 0.00001

D. Observation Model

$$z = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b_g} \\ \mathbf{b_a} \end{bmatrix} + \mathcal{N}(0, \mathbf{R})$$

where $\mathbf{H}$ is the observation matrix defined as:

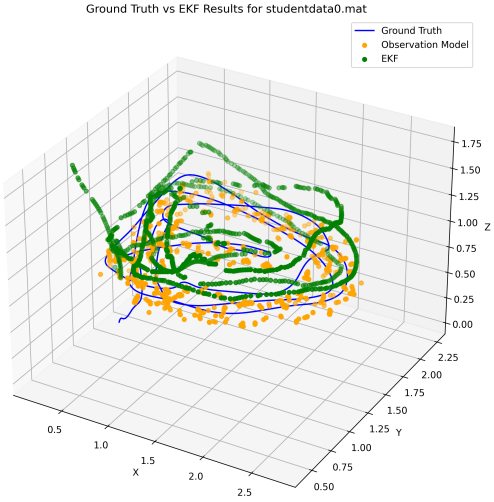
$$\mathbf{H} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \end{bmatrix}$$

and $\mathbf{R}$ is obtained from Task 3 as the average covariance.

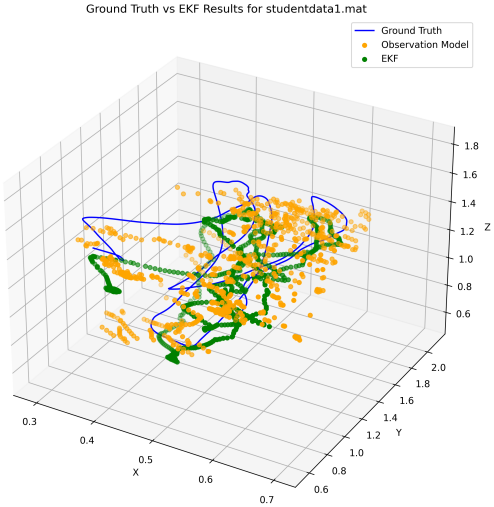
VI. Conclusion

In summary, the Extended Kalman Filter (EKF) provides an effective method for state estimation in nonlinear systems by incorporating sensor measurements and system dynamics. Through this report, we have explored the fundamental principles of the EKF, including system modeling, prediction, and measurement update steps. We visualise the results obtained against the Vicon groundtruth. It should be noted that while the EKF is very good, it is computationally very heavy and takes a lot of time because of the Jacobian calculation involved.

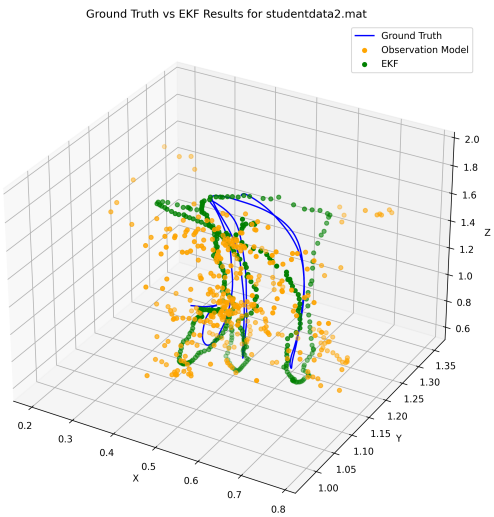
A. Dataset 0



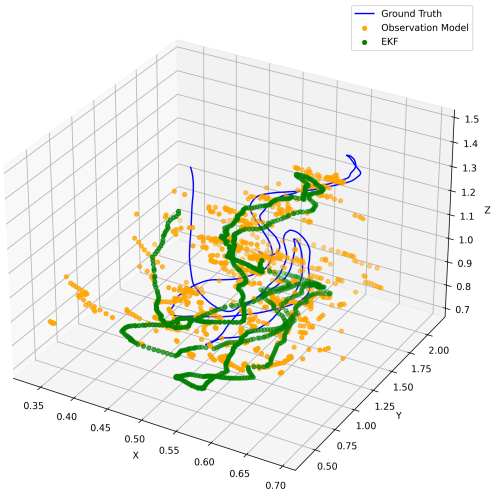
B. Dataset 1



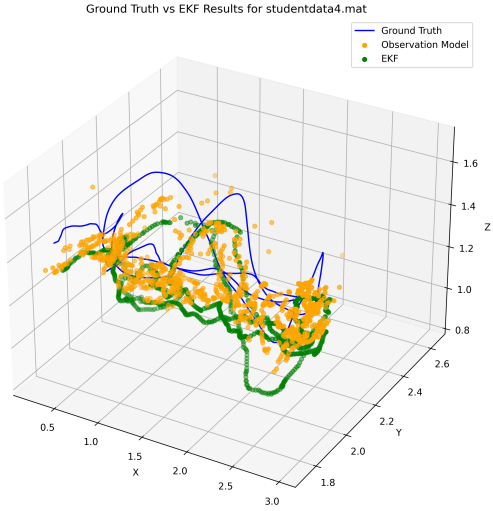
C. Dataset 2



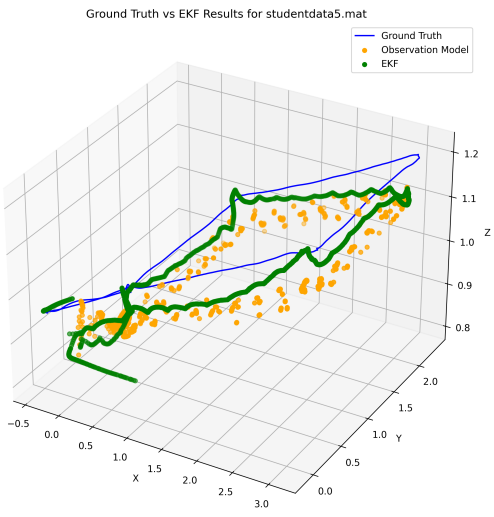
D. Dataset 3



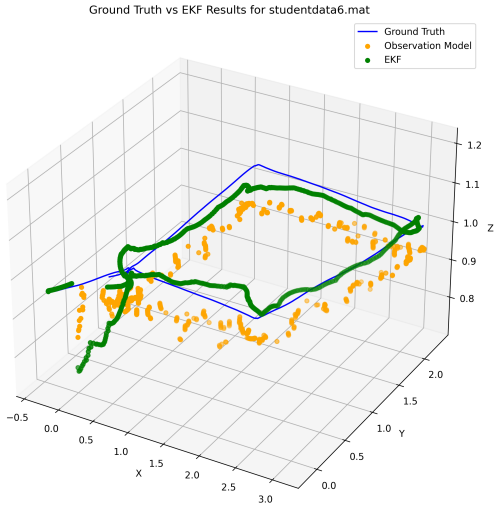
E. Dataset 4



F. Dataset 5



G. Dataset 6



H. Dataset 7