Assignment III: Advanced Robot Navigation (RBE 595)

Pradnya Sushil Shinde

April 18, 2024

1 Non-Linear Kalman Filter

The objective of the project is to implement a Nonlinear Kalman Filter to estimate the pose of a quadcopter drone. For the context of the project, the Extended Kalman Filter has been chosen to implement the Filter Model.

In this problem, we will model the pose using a typical fifteen-state model used in inertial navigation. This state vector will consist of the three-axis components of the linear position, orientation, linear velocity, gyroscope bias, and accelerometer bias as shown below:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b}_g \\ \mathbf{b}_a \end{bmatrix} \tag{1}$$

Additionally, we will define the orientation using a Z-X-Y Euler angle parameterization corresponding to rolling, pitching, and yawing $\mathbf{q} = [\phi\theta\psi]$. In rotation matrix form these yield:

$$R(\mathbf{q}) = \begin{bmatrix} \cos\psi\cos\theta - \sin\phi\sin\psi\sin\theta & -\cos\phi\sin\psi & \cos\psi\sin\theta + \cos\theta\sin\phi\sin\psi \\ \cos\theta\sin\psi + \cos\psi\sin\phi\sin\theta & \cos\phi\cos\psi & \sin\psi\sin\theta - \cos\psi\cos\theta\sin\phi \\ -\cos\phi\sin\theta & \sin\phi & \cos\phi\cos\theta \end{bmatrix} \tag{2}$$

2 Introduction

2.1 System Models

Let us define the System MOdels that will help us evaluate the Process and Observations for the filter model.

2.1.1 Process Model

The continuous time process model is defined as:

$$\dot{x} = \begin{bmatrix} \dot{\mathbf{p}} \\ G(\mathbf{q})^{-1}\mathbf{u}_w \\ \mathbf{g} + R(\mathbf{q})\mathbf{u}_a \\ \mathbf{n}_{bg} \\ \mathbf{n}_{ba} \end{bmatrix} \tag{3}$$

where $\mathbf{u}_w$ and $\mathbf{u}_a$ are the commanded angular velocity and linear acceleration, $\mathbf{g}$ is the gravity vector, and $G(\mathbf{q})$ is defined as:

$$G(\mathbf{q}) = \begin{bmatrix} \cos\theta & 0 & -\cos\phi\sin\theta \\ 0 & 1 & \sin\phi \\ \sin\theta & 0 & -\cos\phi\cos\theta \end{bmatrix} \tag{4}$$

2.1.2 Observation Model

Our observation model is relatively simple. We will be using a computer vision-based technique that measures the position and orientation:

$$\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} + \mathbf{v} \tag{5}$$

Our measurement model is dependent on the estimates of the position and orientation that we will obtain in Task I. Essentially z is an array of $[x, y, z, roll, pitch, yaw]$ estimates.

3 Project Implementation

3.1 TASK I: Pose Estimation

We will process the detected tags through our **estimatePose** function. We need to pass Object Coordinates, Image Coordinates, Camera Calibration Matrix, and the Distortion Coefficients to the **solvePnP**() function by **OpenCV**. The shape of our Object Coordinates & Image Coordinates will be $(N, 3, 1)$ and $(N, 2, 1)$ where N will be the number of detected tags per iteration. We will read the Camera Calibration Matrix and the Distortion Coefficients from the *parameters.txt* provided to us. Following is the approach that estimates the pose of the detected tags and returns the position and orientation of the drone in the world frame.

Algorithm 1 Overview: Pose Estimation

Require: Detected Tags, Params

1: Define the Map of Tags as an array: $tagMap$
2: Compute Coordinates of the Tag Map: $tagMap\_coords \leftarrow init\_tagsCoords(tagMap)$
3: Compute Image Coordinates: $img\_coords \leftarrow computeImgCoords(detectedTags)$
4: Compute World Coordinates: $world\_coords \leftarrow computeWorldCoords(detectedTags, tagMap\_coords)$
5: Obtain Camera Matrix: $camMat \leftarrow params["K"]$
6: Obtain Distortion Coefficients: $distCoeffs \leftarrow params["Dst"]$
7: Compute the number of detected tags: $num\_detected\_tags \leftarrow$ length of $world\_coords/4$
8: Solve PnP problem: $success, rvec, tvec \leftarrow solvePnP(world\_coords, img\_coords, camMat, distCoeffs, method)$
9: Define Transformation Parameters: $XYZ, Yaw \leftarrow params["XYZ"], params["Yaw"]$
10: Compute Rotation Matrix: $rotZ \leftarrow$ rotation matrix about Z-axis with angle Yaw
11: Compute Rotation Matrix: $rotX \leftarrow$ rotation matrix about X-axis with angle 180°
12: Compute Total Rotation Matrix: $rot \leftarrow rotX \cdot rotZ$
13: Compute Camera-to-Robot Transformation Matrix: $Hcd \leftarrow$ homogeneous transformation matrix using rot and XYZ
14: Compute Rotation Matrix: $R \leftarrow$ rotation matrix obtained from $rvec$
15: Compute Camera-to-World Transformation Matrix: $Hcw \leftarrow [R, tvec]$
16: Add Bottom Row to Hcw: $Hcw \leftarrow [Hcw; [0, 0, 0, 1]]$
17: Compute Drone-to-World Transformation Matrix: $Hdw \leftarrow Hcw^{-1} \cdot Hcd$
18: Extract Orientation Matrix: $orientation\_mat \leftarrow$ first 3 rows and columns of Hdw
19: Compute Orientation: $orientation \leftarrow computeEulerAngles(orientation\_mat)$
20: Extract Position: $position \leftarrow$ first 3 elements of last column of Hdw
21: **return** Estimated Orientation and Position: $orientation, position$

In the approach described above, a few key steps need to be done to produce an efficient estimate of the pose.

1. It is very important to make sure that **Detected Tags** is not empty. There are instances in the data where the number of detected tags is zero. Therefore it was essential to pass Detected Tags to **estimatePose** only if at least one tag was detected. All the data instances where there were no tags detected were skipped.

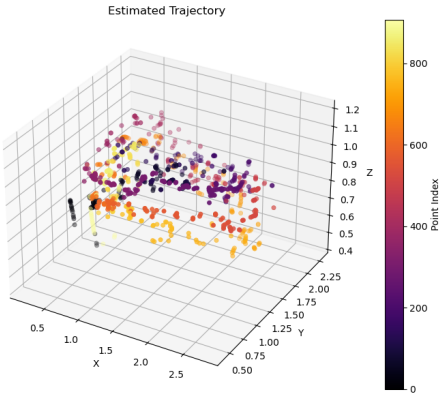
2. When we generate the world coordinates, we need to consider the extra offset that has been given in the tag map. Making sure that this offset gets added correctly is essential.

3. The transformation of the drone to the world should be:

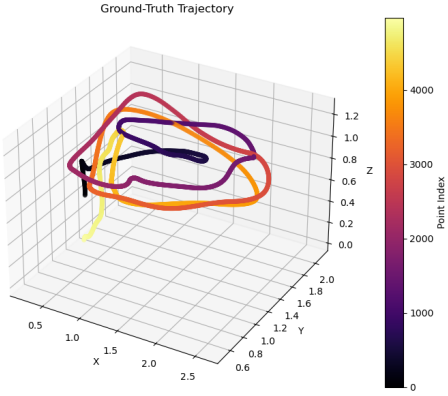
$$T_d^w = inv(T_w^c)T_d^c$$

3.2 TASK II: Visualization

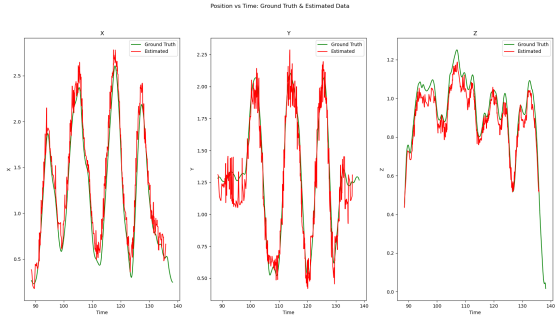
Next, we will visualize our estimates of the drone's pose in the World Frame. We will also compare the ground truth data obtained through Vicon and the pose estimates. The estimated and ground truth positions of each data has been plotted in the form of a trajectory. Refer to the plots below.



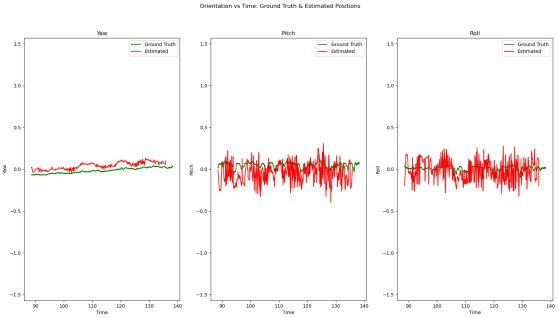
(a) Estimated Pose



(b) Ground Truth Pose

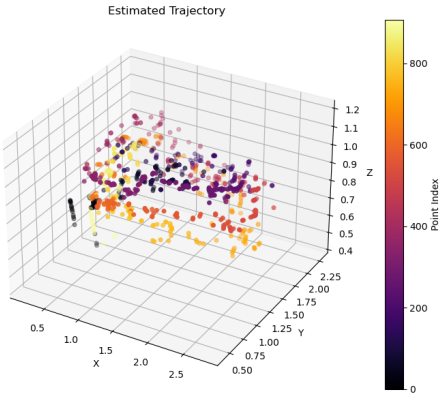


(c) Positions Comparison

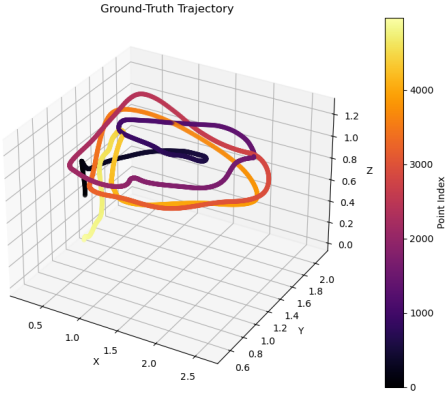


(d) Orientation Comparison

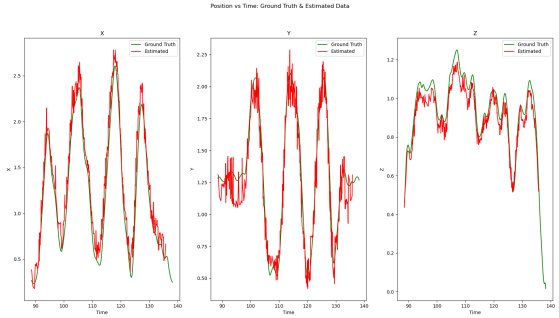
Figure 1: **Student Data 0**



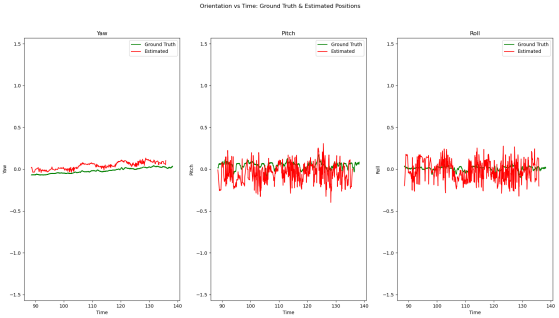
(a) Estimated Pose



(b) Ground Truth Pose

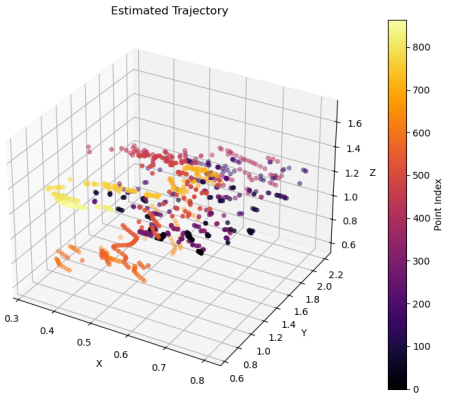


(c) Positions Comparison

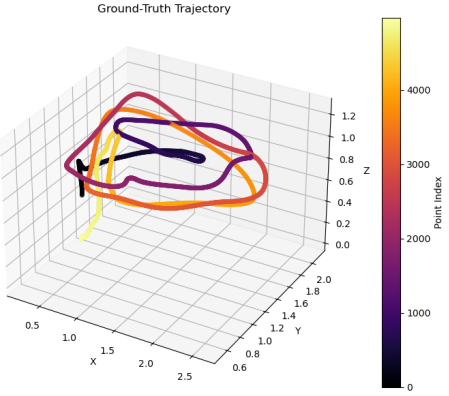


(d) Orientation Comparison

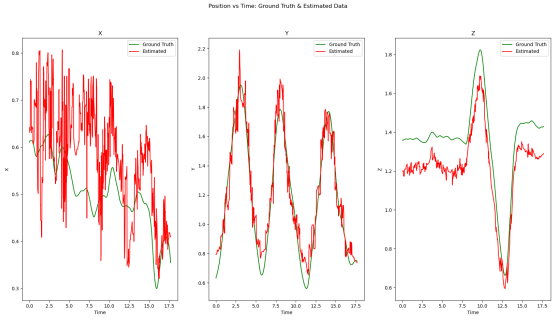
Figure 2: **Student Data 0**



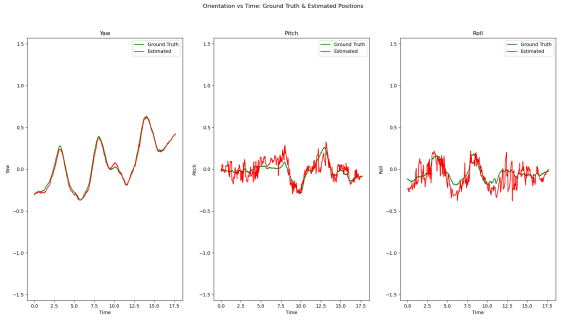
(a) Estimated Pose



(b) Ground Truth Pose

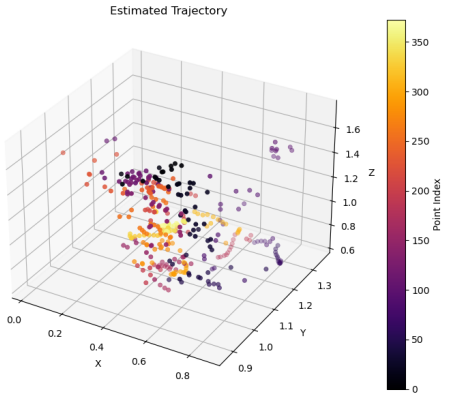


(c) Positions Comparison

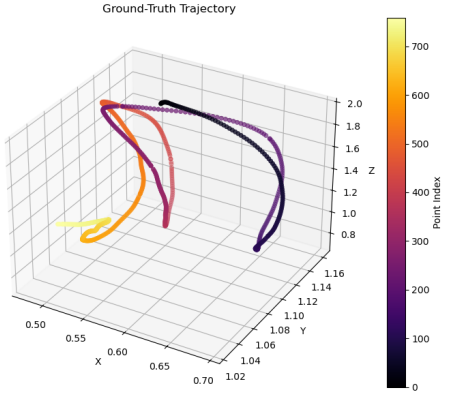


(d) Orientation Comparison

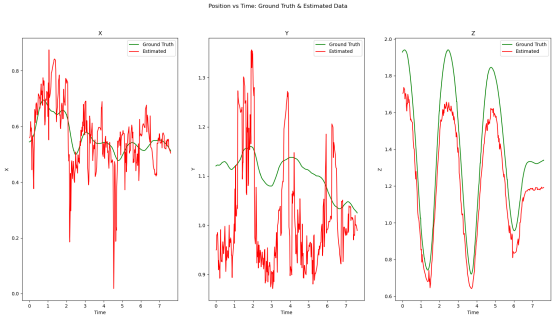
Figure 3: **Student Data 1**



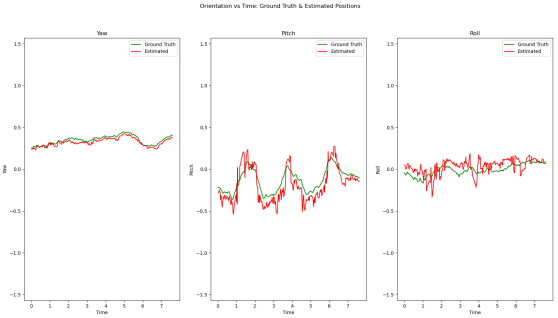
(a) Estimated Pose



(b) Ground Truth Pose

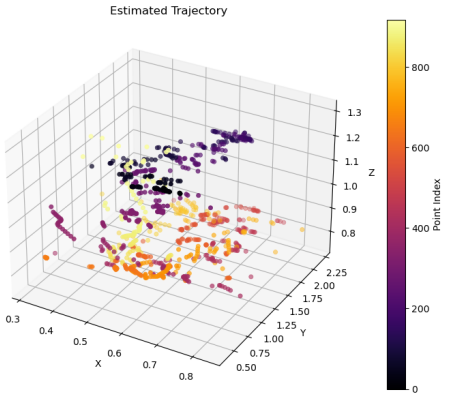


(c) Positions Comparison

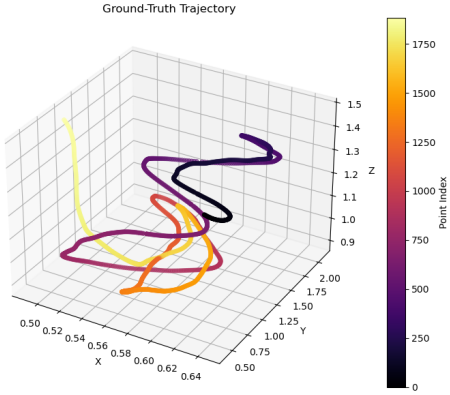


(d) Orientation Comparison

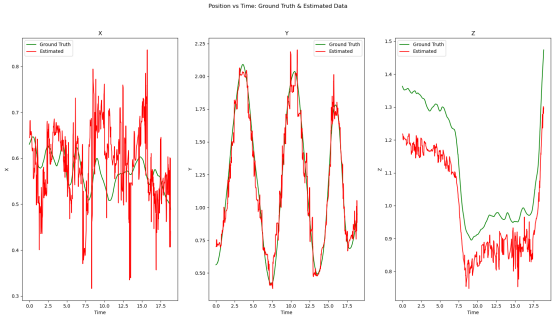
Figure 4: **Student Data 2**



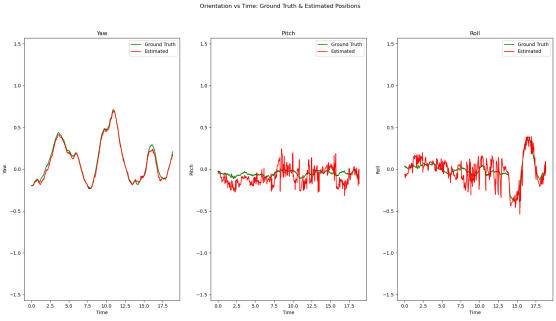
(a) Estimated Pose



(b) Ground Truth Pose

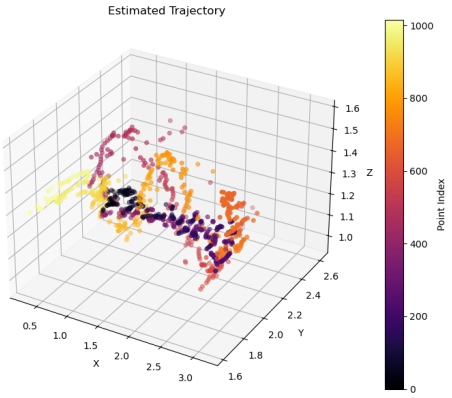


(c) Positions Comparison

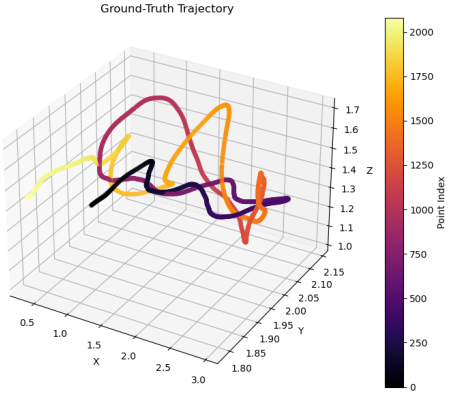


(d) Orientation Comparison

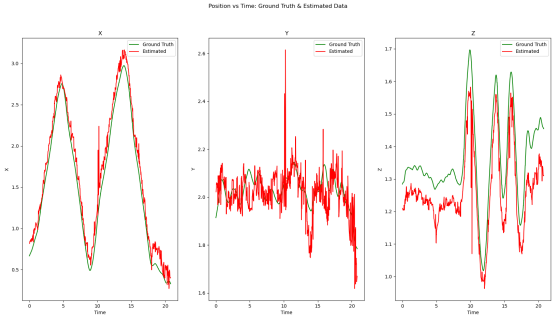
Figure 5: **Student Data 3**



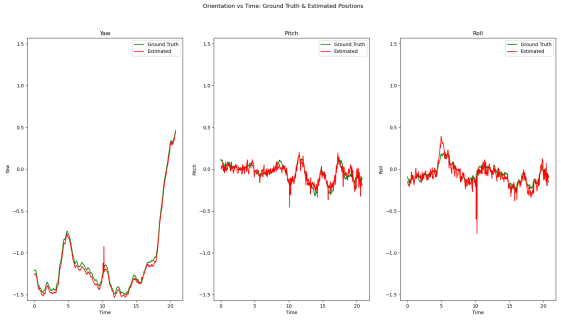
(a) Estimated Pose



(b) Ground Truth Pose

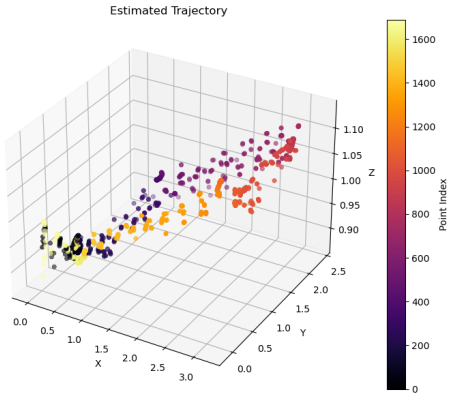


(c) Positions Comparison

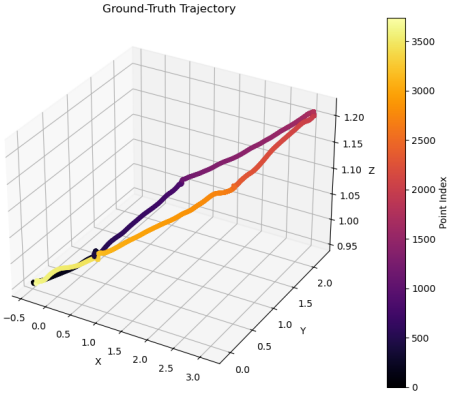


(d) Orientation Comparison

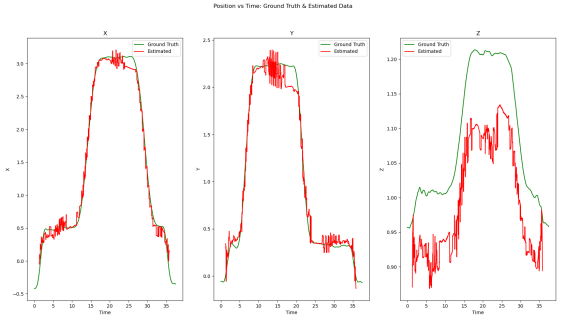
Figure 6: **Student Data 4**



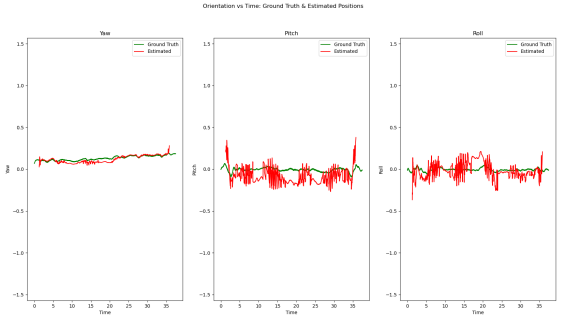
(a) Estimated Pose



(b) Ground Truth Pose

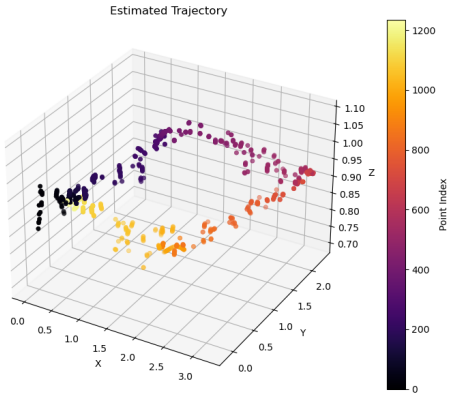


(c) Positions Comparison

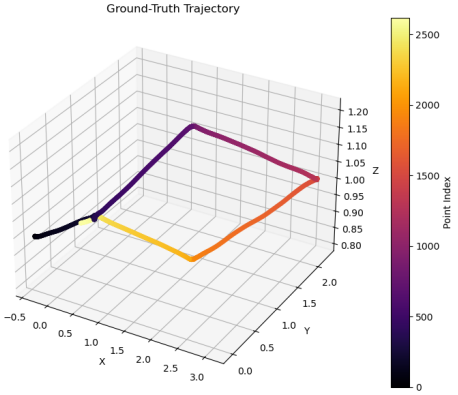


(d) Orientation Comparison

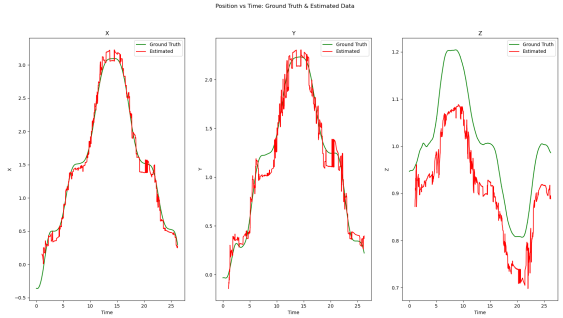
Figure 7: **Student Data 5**



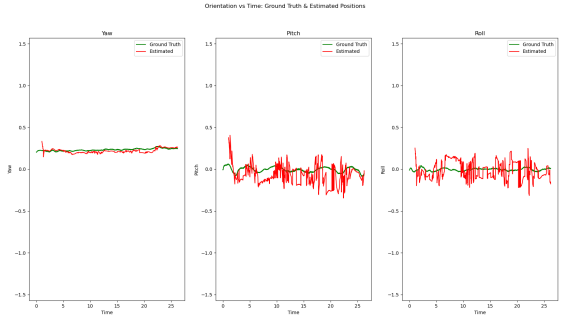
(a) Estimated Pose



(b) Ground Truth Pose

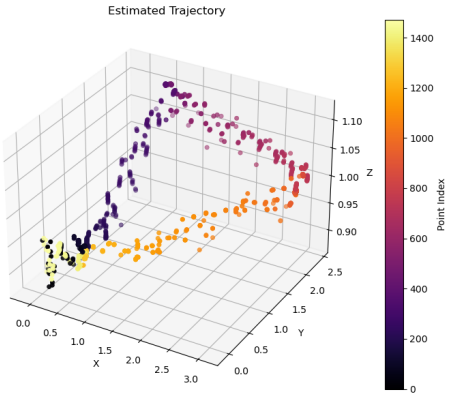


(c) Positions Comparison

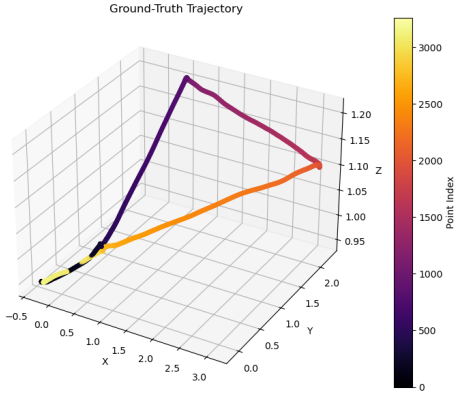


(d) Orientation Comparison

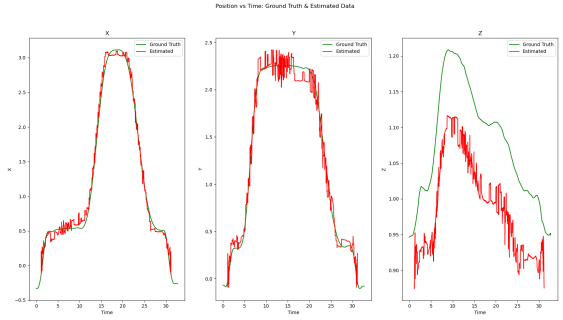
Figure 8: **Student Data 6**



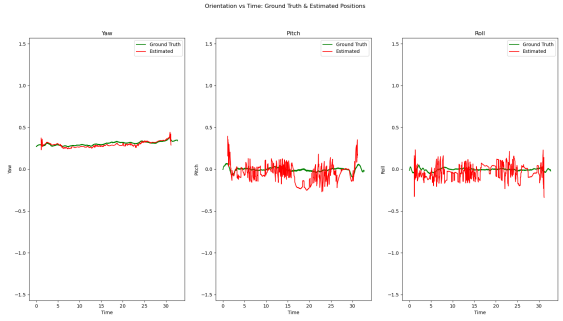
(a) Estimated Pose



(b) Ground Truth Pose



(c) Positions Comparison



(d) Orientation Comparison

Figure 9: **Student Data 7**

3.3 Task III: Covariance Estimation

We define the covariance using the IMU Error state which is defined as the residuals between the ground truth data and estimated data at a given timestamp. To calculate covariance, we use the formula for sample covariance:

$$R = \frac{1}{n-1}\sum_{t=1}^{n} \mathbf{v_t}\mathbf{v_t}^T \tag{6}$$

To get the value of $\mathbf{v_t}$ at $\mathbf{t}$ rearrange the observation model to solve for $\mathbf{v_t}$. For the state, we will use the ground truth data. $\mathbf{v_t}$ is our error state that defines the residuals between the ground truth and the estimates that we get from our observation model. The function *estimateCovariances* given below, generates an estimated state using the observation model at a given timestamp and then moves forward to match the ground truth data related to the data estimates at that timestamp. This way we generate two states: ground truth and estimated. We then use the equation defined above to sample covariance.

Let us now evaluate each dataset based on the estimated covaraiances.

Covariance Estimates for dataset studentdata0:

$$\begin{bmatrix} 0.02894873 & 0.00035015 & -0.00624655 & -0.01029323 & -0.0045547 & 0.00860519 \\ 0.00035015 & 0.01396505 & 0.00137181 & -0.01067139 & 0.01136407 & -0.00151906 \\ -0.00624655 & 0.00137181 & 0.00312318 & 0.00144324 & 0.00474604 & -0.00289554 \\ -0.01029323 & -0.01067139 & 0.00144324 & 0.01358077 & -0.00508598 & -0.00267434 \\ -0.0045547 & 0.01136407 & 0.00474604 & -0.00508598 & 0.01999573 & -0.00524485 \\ 0.00860519 & -0.00151906 & -0.00289554 & -0.00267434 & -0.00524485 & 0.00424479 \end{bmatrix}$$

Covariance Estimates for dataset studentdata1:

$$\begin{bmatrix} 0.01284599 & 0.00675027 & -0.01081179 & -0.0059248 & 0.00682737 & -0.00113557 \\ 0.00675027 & 0.01196752 & -0.00546916 & -0.00995215 & 0.00505807 & -0.0001446 \\ -0.01081179 & -0.00546916 & 0.02044729 & 0.00557909 & -0.00264406 & 0.00215943 \\ -0.0059248 & -0.00995215 & 0.00557909 & 0.00924547 & -0.00411386 & 0.00034004 \\ 0.00682737 & 0.00505807 & -0.00264406 & -0.00411386 & 0.00584187 & -0.00084823 \\ -0.00113557 & -0.0001446 & 0.00215943 & 0.00034004 & -0.00008482 & 0.0007547 \end{bmatrix}$$

Covariance Estimates for dataset studentdata2:

$$\begin{bmatrix} 0.0081628 & 0.00359794 & 0.00220744 & -0.0013195 & 0.00729266 & 0.00055579 \\ 0.00359794 & 0.01899539 & 0.01815314 & -0.0116297 & 0.01273038 & 0.00224676 \\ 0.00220744 & 0.01815314 & 0.03114883 & -0.00875887 & 0.01276606 & 0.0044744 \\ -0.0013195 & -0.0116297 & -0.00875887 & 0.0082751 & -0.00711079 & -0.0009408 \\ 0.00729266 & 0.01273038 & 0.01276606 & -0.00711079 & 0.01317591 & 0.00187667 \\ 0.00055579 & 0.00224676 & 0.0044744 & -0.0009408 & 0.00187667 & 0.00083141 \end{bmatrix}$$

Covariance Estimates for dataset studentdata3:

$$\begin{bmatrix} 0.00669198 & 0.00181932 & -0.00127696 & -0.002676 & 0.004455 & -0.00018413 \\ 0.00181932 & 0.00872282 & 0.00060016 & -0.00782755 & 0.0045595 & 0.00079854 \\ -0.00127696 & 0.00060016 & 0.01490303 & 0.00002053 & 0.00512637 & 0.00216952 \\ -0.002676 & -0.00782755 & 0.00002053 & 0.00774332 & -0.00449094 & -0.0005493 \\ 0.004455 & 0.0045595 & 0.00512637 & -0.00449094 & 0.00835971 & 0.0013105 \\ -0.00018413 & 0.00079854 & 0.00216952 & -0.0005493 & 0.0013105 & 0.00081961 \end{bmatrix}$$

Covariance Estimates for dataset studentdata4:

$$\begin{bmatrix} 0.02509984 & -0.00009114 & -0.01420279 & -0.00577039 & 0.00031091 & -0.00482437 \\ -0.00009114 & 0.00586676 & 0.00077105 & -0.00212836 & -0.0022801 & 0.00042899 \\ -0.01420279 & 0.00077105 & 0.0100301 & 0.00200813 & 0.00039953 & 0.00331021 \\ -0.00577039 & -0.00212836 & 0.00200813 & 0.00381115 & -0.00003221 & 0.00014021 \\ 0.00031091 & -0.0022801 & 0.00039953 & -0.00003221 & 0.00249519 & 0.00028618 \\ -0.00482437 & 0.00042899 & 0.00331021 & 0.00014021 & 0.00028618 & 0.00169381 \end{bmatrix}$$

Covariance Estimates for dataset studentdata5:

$$\begin{bmatrix} 0.00966551 & 0.00088821 & 0.00312471 & 0.00025683 & 0.00829163 & -0.00030109 \\ 0.00088821 & 0.01113058 & 0.00199249 & -0.00891953 & 0.00600691 & 0.0017527 \\ 0.00312471 & 0.00199249 & 0.00943707 & 0.00047644 & 0.00773127 & 0.00119448 \\ 0.00025683 & -0.00891953 & 0.00047644 & 0.00787112 & -0.00260679 & -0.00105222 \\ 0.00829163 & 0.00600691 & 0.00773127 & -0.00260679 & 0.01369761 & 0.00156968 \\ -0.00030109 & 0.0017527 & 0.00119448 & -0.00105222 & 0.00156968 & 0.00063982 \end{bmatrix}$$

Covariance Estimates for dataset studentdata6:

$$\begin{bmatrix} 0.00948482 & 0.00454839 & 0.00103349 & -0.00320743 & 0.00946928 & 0.00004417 \\ 0.00454839 & 0.01373006 & 0.00295133 & -0.01176913 & 0.00974742 & 0.00129343 \\ 0.00103349 & 0.00295133 & 0.00945069 & 0.00030763 & 0.00630168 & 0.00153037 \\ -0.00320743 & -0.01176913 & 0.00030763 & 0.01167644 & -0.00578416 & -0.00046069 \\ 0.00946928 & 0.00974742 & 0.00630168 & -0.00578416 & 0.01611117 & 0.00169115 \\ 0.00004417 & 0.00129343 & 0.00153037 & -0.00046069 & 0.00169115 & 0.00064961 \end{bmatrix}$$

Covariance Estimates for dataset studentdata7:

$$\begin{bmatrix} 0.00822665 & 0.00342751 & -0.00182822 & -0.00226246 & 0.00545776 & -0.00041663 \\ 0.00342751 & 0.01076723 & 0.00010564 & -0.00818434 & 0.00746801 & 0.00069698 \\ -0.00182822 & 0.00010564 & 0.00977171 & 0.00294697 & 0.00453422 & 0.00121002 \\ -0.00226246 & -0.00818434 & 0.00294697 & 0.00784792 & -0.00288566 & 0.00008461 \\ 0.00545776 & 0.00746801 & 0.00453422 & -0.00288566 & 0.01115503 & 0.00126034 \\ -0.00041663 & 0.00069698 & 0.00121002 & 0.00008461 & 0.00126034 & 0.00045602 \end{bmatrix}$$

The cumulative Covariance Estimate is:

$$\begin{bmatrix} 0.01364079 & 0.00266133 & -0.00350008 & -0.00389962 & 0.00469374 & 0.00029292 \\ 0.00266133 & 0.01189318 & 0.00255956 & -0.00888527 & 0.00683178 & 0.00069422 \\ -0.00350008 & 0.00255956 & 0.01353899 & 0.0005029 & 0.00487014 & 0.00164411 \\ -0.00389962 & -0.00888527 & 0.0005029 & 0.00875641 & -0.0040138 & -0.00063906 \\ 0.00469374 & 0.00683178 & 0.00487014 & -0.0040138 & 0.01135403 & 0.0003331 \\ 0.00029292 & 0.00069422 & 0.00164411 & -0.00063906 & 0.0003331 & 0.00126122 \end{bmatrix}$$

We have used the **Cumulative Covariance Matrix** as the **Measurement Noise Covariance R** in our implementation which is the average of all the covariance matrices of all datasets. A generalised Covariance Matrix is defined below:

$$C = \begin{bmatrix} C_{xx} & C_{xy} & C_{xz} & C_{x\text{roll}} & C_{x\text{pitch}} & C_{x\text{yaw}} \\ C_{xy} & C_{yy} & C_{yz} & C_{y\text{roll}} & C_{y\text{pitch}} & C_{y\text{yaw}} \\ C_{xz} & C_{yz} & C_{zz} & C_{z\text{roll}} & C_{z\text{pitch}} & C_{z\text{yaw}} \\ C_{\text{roll}x} & C_{\text{roll}y} & C_{\text{roll}z} & C_{\text{rollroll}} & C_{\text{rollpitch}} & C_{\text{rollyaw}} \\ C_{\text{pitch}x} & C_{\text{pitch}y} & C_{\text{pitch}z} & C_{\text{pitchroll}} & C_{\text{pitchpitch}} & C_{\text{pitchyaw}} \\ C_{\text{yaw}x} & C_{\text{yaw}y} & C_{\text{yaw}z} & C_{\text{yawroll}} & C_{\text{yawpitch}} & C_{\text{yawyaw}} \end{bmatrix}$$

3.3.1 Evaluation

- To evaluate the performance of each estimated state variable w.r.t its corresponding ground truth state variable (For eg. estimate position x and ground truth position x), we need to observe the values of diagonal covariances.
- The off-diagonal covariance values on the other hand tell us about the performance of each of the estimated state variables w.r.t other ground truth state variables (For eg. estimate position x and ground truth orientation $roll$).
- The covariance for each dataset seems relatively small. But the plots are describing a different picture altogether. We can see a lot of noise in the comparison plots for position and orientation. The noise indicates failure to compensate for the sensor noise that may have collected inaccurate tag pose.
- To overcome the noise in the estimates, we can use robust PnP techniques such as RANSAC.

3.4 Task IV: Extended Kalman Filter

Extended Kalman Filter has been implemented as described below:

1. Initialize current state, **x**

$$\hat{x} = [positionEstimates, orientationEstimates, velocityData, accelerometerbias, gyroscopebias] \tag{7}$$

2. Predict state based on current state **x**, control inputs **u** and estimate covariance matrix **P**
3. Define Process Model as described in section **2.2.1** and compute jacobian **A** of the state. The Jacobian computation is based on the **jacobian** function by **sympy**.
4. The state transition matrix, F is calculated using the Jacobian matrix A, calculated as:

$$F = I + A \times dt \tag{8}$$

5. The new state x is calculated as:

$$x = x + x_{\text{dot}} \times dt \tag{9}$$

6. Finally, the new covariance matrix P is updated as:

$$P = F \times P \tag{10}$$

7. Update state using the Measurement Model which is based on the IMU readings.
8. The Kalman gain is computed as:

$$K = P \times H^T \times (H \times P \times H^T + R)^{-1} \tag{11}$$

9. The updated state estimate x is given by:

$$x = x + K \times y \tag{12}$$

10. The updated covariance matric P is given by:

$$P = (I - K \times H) \times P \tag{13}$$

11. Here H matrix is defined as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 10: EKF 0

The 1s defined here are concerned with position and orientation measurements (estimates).

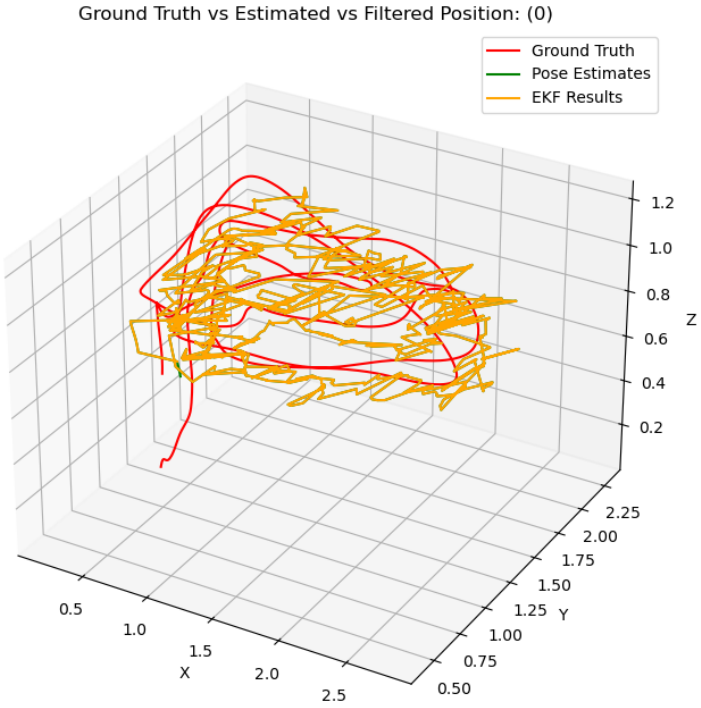


Figure 11: EKF 0

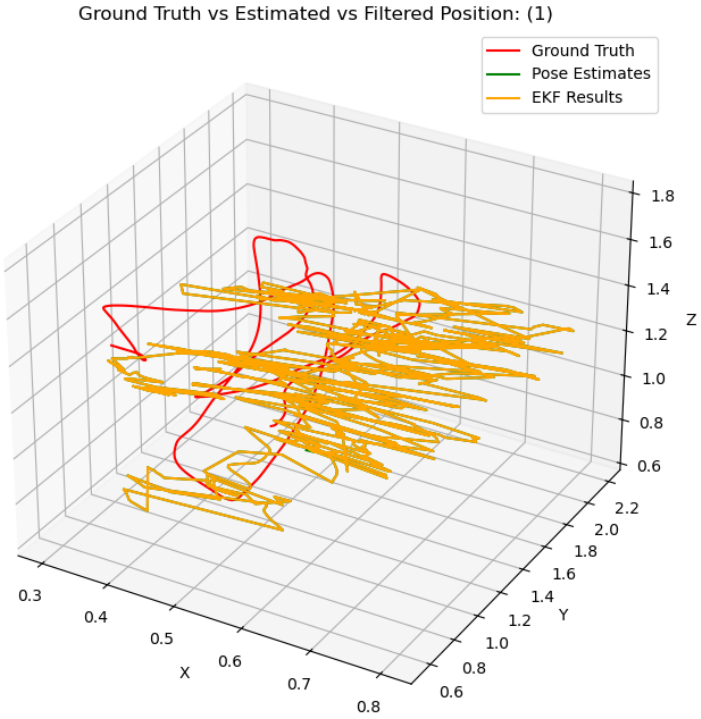


Figure 12: EKF 1

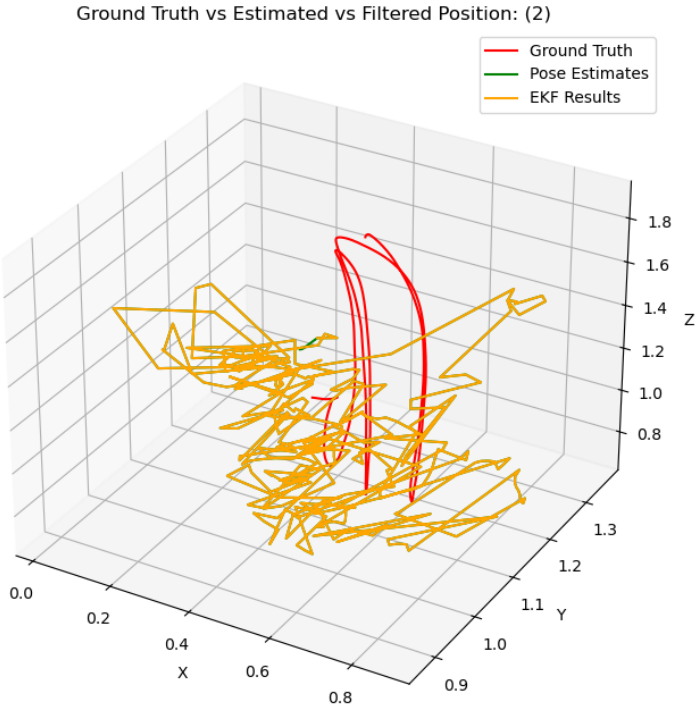


Figure 13: EKF 2

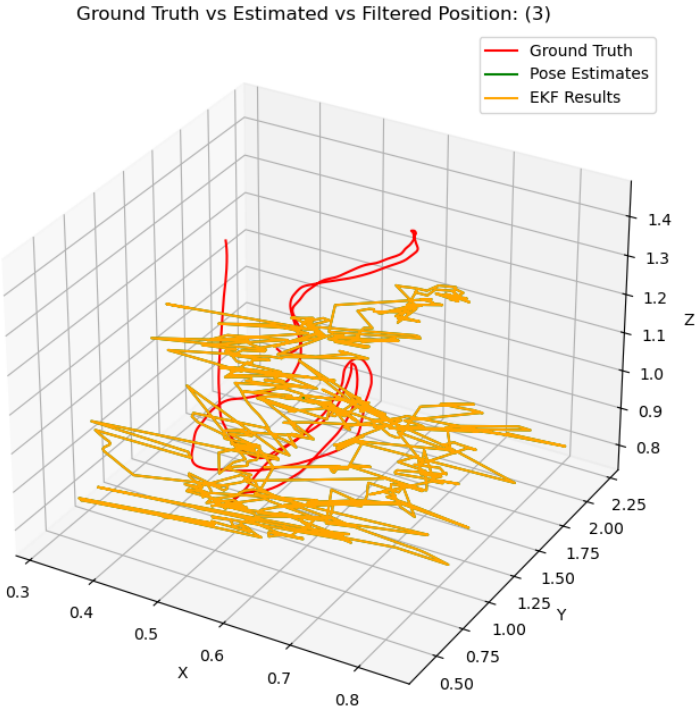


Figure 14: EKF 3

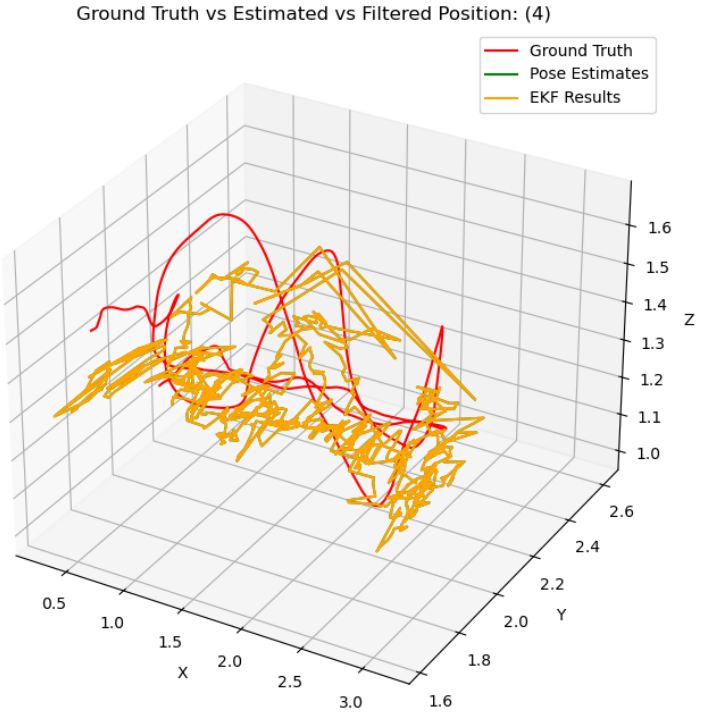


Figure 15: EKF 4

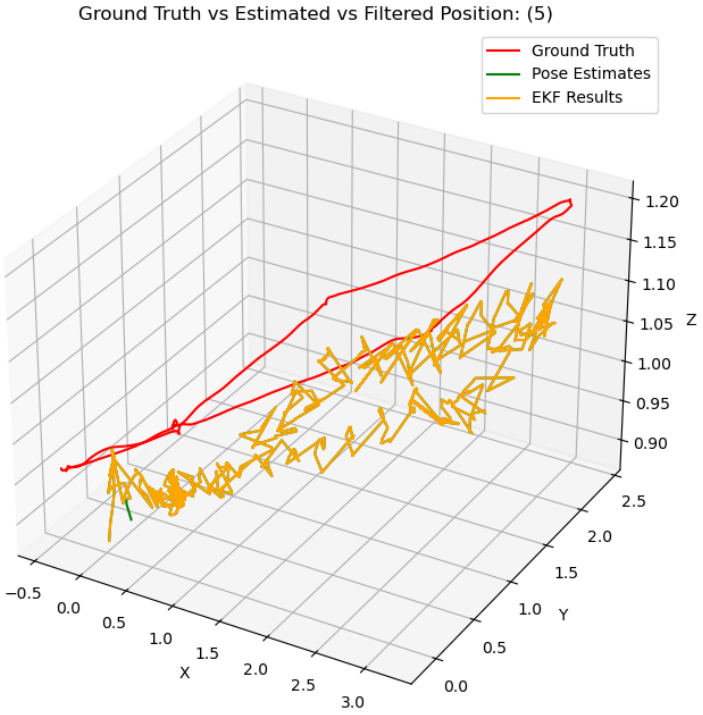


Figure 16: EKF 5

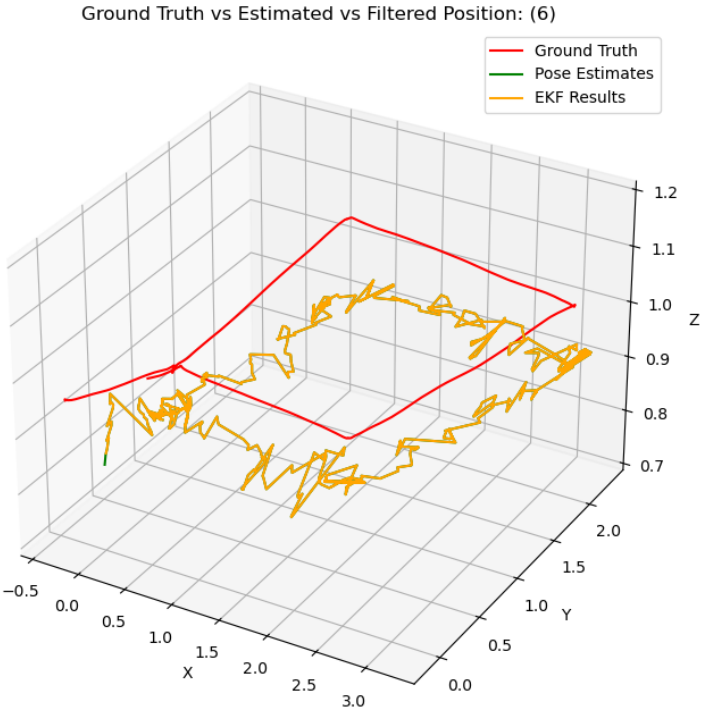


Figure 17: EKF 6

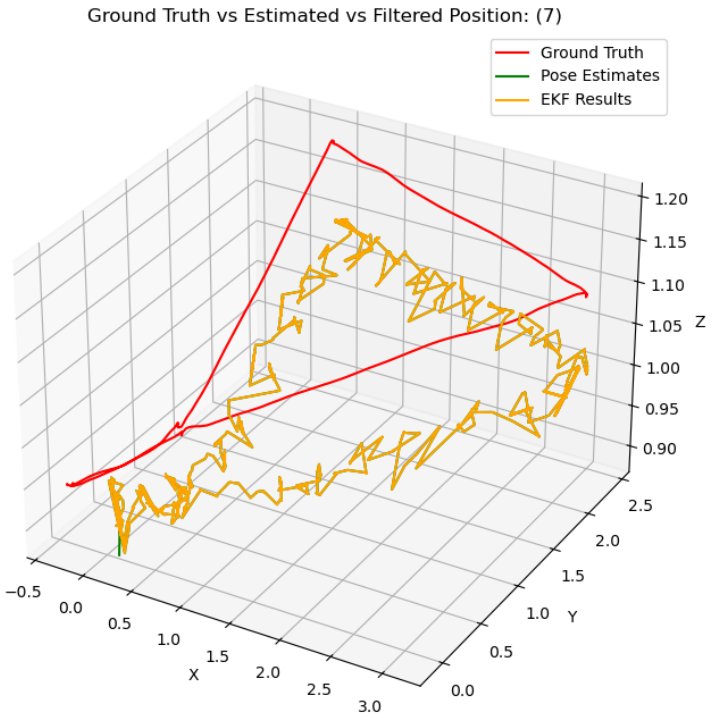


Figure 18: EKF 7

3.4.1 Evaluation

- EKF results are noisy and require fine-tuning. This can be accomplished by tuning the Estimate Covariance Matrix and Process Noise Covariance Matrix.
- Specifically our estimates are noisy, therefore, tuning the Estimate Covariance Matrix is essential.

4 Analysis

Following is the analysis drawn based on the results obtained in previous sections:

1. The pose estimates are noisy. This noise is highly observant in the orientation estimates. This could be due to error in the calculation of transformations or the computation of Euler angles.
2. The Non-Linear Kalman FIlter should deal with this noise however we observe that it fails to do so (estimates and filtered plots are coinciding to some extent).

From the analysis above, to improve filter results, we can prefer to implement UKF over EKF. This holds true for the following reasons:

1. UKF handles non-linearity better. Since the dynamics of the drone are highly non-linear, UKF is probably a better choice because it uses a deterministic sampling technique to propagate the mean and covariance through the nonlinear functions, whereas EKF linearizes these functions using a first-order Taylor expansion, which might not be accurate for highly nonlinear systems.

2. UKF is generally more robust to model inaccuracies and measurement noise compared to EKF because it directly captures the statistical properties of the state estimation without relying heavily on linearization. Since it is stated that the IMU data is highly noisy in our case, the filter model is likely to showcase noisy outputs.

3. UKF tends to maintain consistency (i.e., the estimated covariance matrix remains positive definite) even for highly nonlinear systems, whereas EKF mightbp suffer from inconsistency issues due to linearization errors.

The reasons for choosing EKF were:

1. EKF seemed to be computationally more efficient compared to UKF.

2. The simplicity and familiarity of EKF made it more preferrable.