RBE 595 - Advanced Navigation
Lab #2 - Perspective N Point
Jason Patel

Introduction

This lab investigates the hypothesis that a robot can be localized using an observation model based on AprilTags placed on a known map and a calibrated camera. Given the intrinsic parameters of the camera and the known 3D positions of AprilTags, the Perspective-n-Point (PnP) model is employed to estimate the pose of the robot from 2D tag detections in camera images. The estimated trajectory is compared against ground truth data collected via a Vicon motion capture system to assess the performance and accuracy of the observation model.

Problem Description

The system aims to estimate the drone's pose using:

- AprilTag detections from camera images (providing sparse pose measurements)

The dataset includes eight MATLAB files (studentdata0.mat to studentdata7.mat) each containing time-stamped measurements. Ground truth is provided via Vicon motion capture data.

Methods

AprilTags and World Frame Mapping

AprilTags are square fiducial markers that can be uniquely identified and precisely localized in images. Each tag's four corners are defined in a known coordinate system based on a structured grid layout. In this experiment, tags are arranged in a 12x9 grid with non-uniform spacing to reflect physical layout constraints. The method `get_corners_world_frame()` computes the 3D world coordinates for the corners of each AprilTag given its index.

Camera Calibration

The camera used in the experiment is characterized by the following intrinsic parameters:

Camera Matrix:
[[314.1779, 0.0, 199.4848],
 [0.0, 314.2218, 113.7838],
 [0.0, 0.0, 1.0]]

Distortion Coefficients:
[-0.438607, 0.248625, 0.00072, -0.000476, -0.0911]

These parameters were used in OpenCV's `cv2.solvePnP` function to estimate the 6-DOF pose of the robot relative to the world frame.

Perspective-n-Point Model

The PnP problem involves finding the position and orientation of a calibrated camera given a set of 3D points and their 2D projections. In this implementation, 3D points are the AprilTag corners in the world frame, and 2D points are the corresponding tag corners in the image.

- For each frame, the image data and detected tags are processed.
- If no images are detected in the measurement it is skipped
- If a single tag is visible, its corners are used directly; if multiple tags are visible, their corner sets are concatenated.
- `cv2.solvePnP` is used to obtain rotation and translation vectors.
- The transformation from camera to robot frame is applied using known offsets and extrinsic rotation matrices. The translation from camera to robot was modified to be [-.04, 0.0, 0.4] from [-.04, 0.0, -0.3] to fix issues with tracking alignment

Data Processing Pipeline

- MATLAB `.mat` data files are loaded using `scipy.io`.
- Each frame is processed to estimate the camera pose.
- The estimated pose is transformed to the robot's frame.
- Results are stored and later visualized.
- Ground truth from Vicon is time-aligned using linear interpolation.
- Covariance between the estimated and true trajectory is computed to assess consistency.

Results

Trajectory Comparison

A 3D plot of the robot trajectory shows both the Vicon (ground truth) and the estimated paths.

- The blue line represents the actual path.
- The red line indicates the estimated trajectory based on PnP.

Orientation Comparison

Roll, pitch, and yaw values are plotted over time:

- Each subplot displays actual (blue) and estimated (red) values.
- The alignment between these plots indicates the accuracy of orientation estimation.

File 0

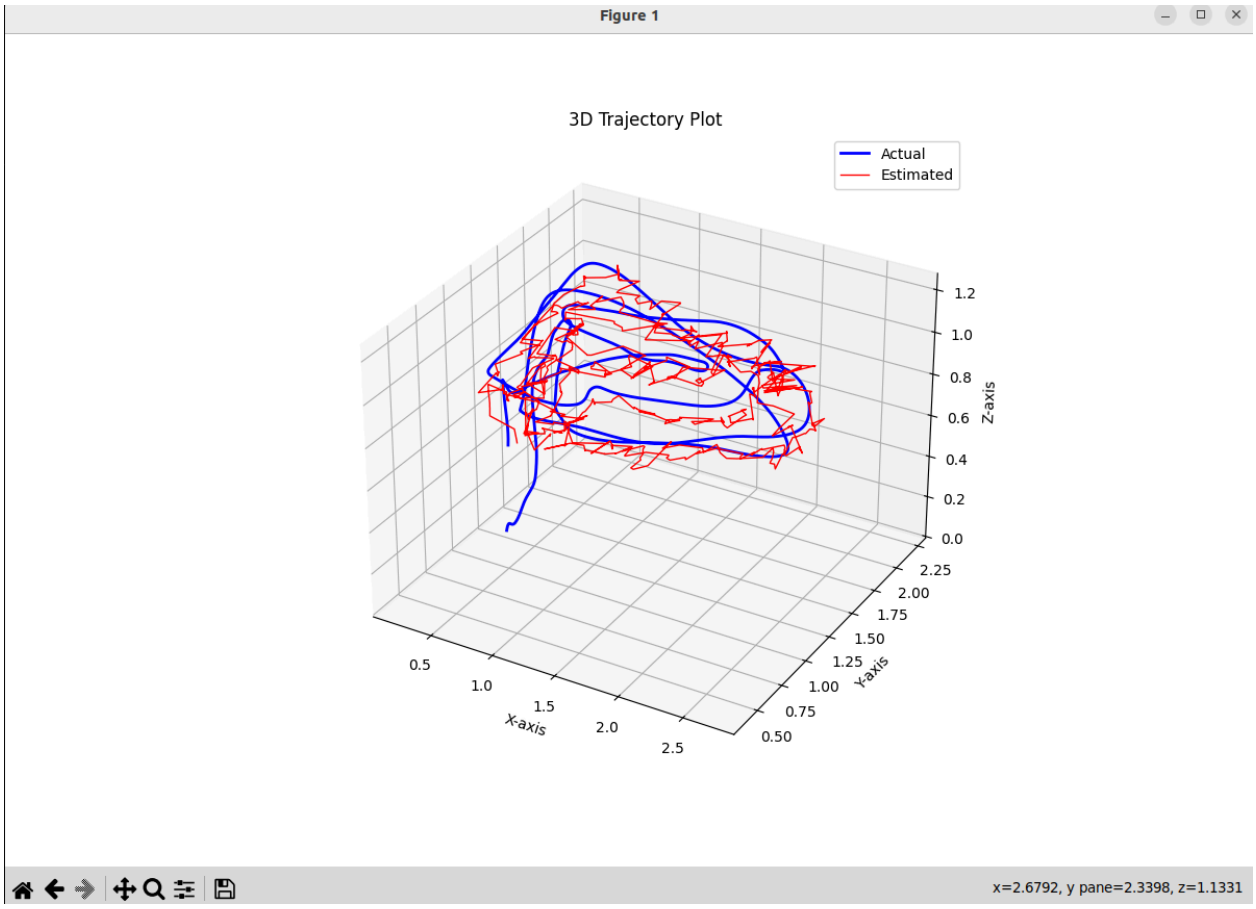



Figure 1
Roll / Pitch / Yaw Plot
Roll Plot
Roll (rad)
Time
Actual
Estimated
Pitch Plot
Pitch (rad)
Time
Actual
Estimated
Yaw Plot
Yaw (rad)
Time
Actual
Estimated

File 1

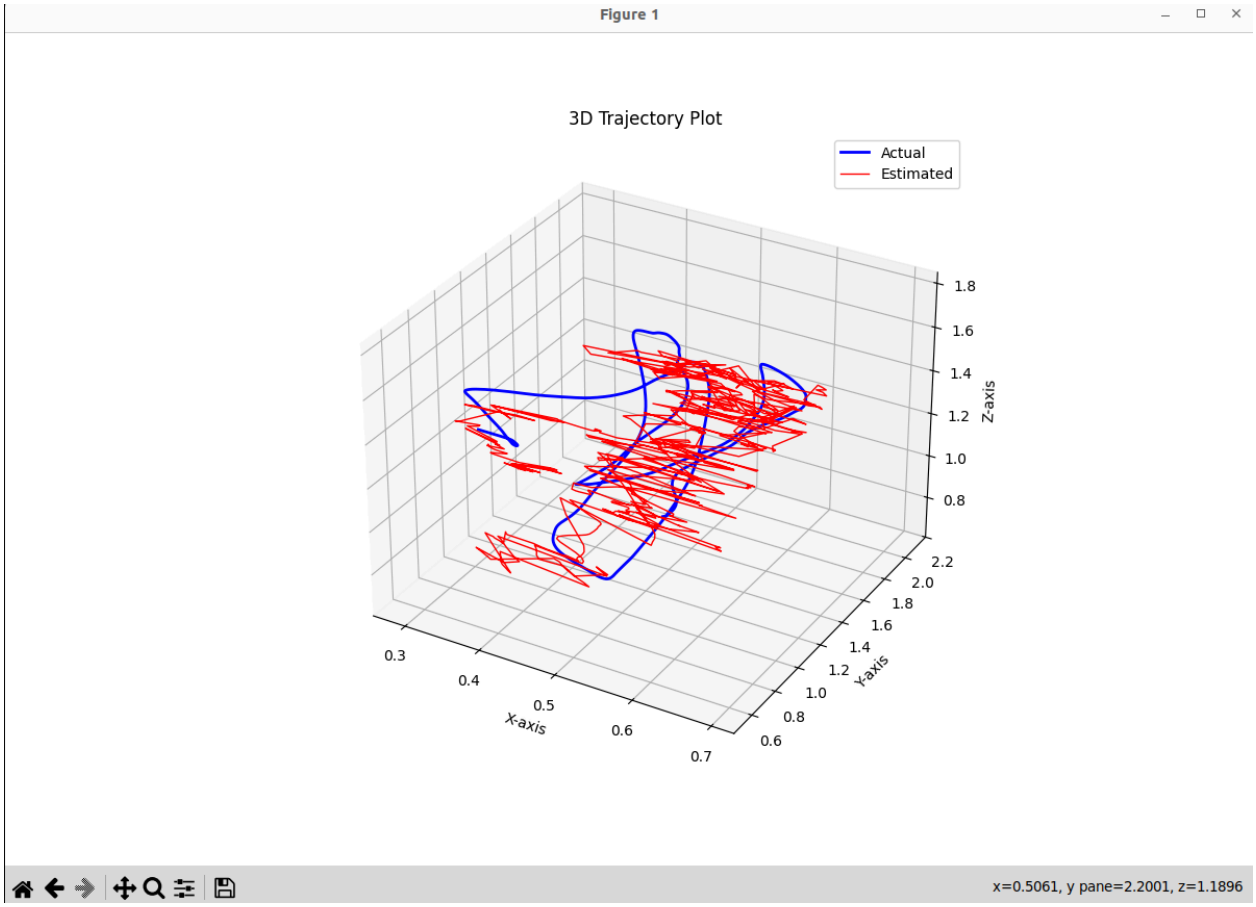


Figure 1
3D Trajectory Plot
Actual
Estimated
X-axis
Y-axis
Z-axis
x=0.5061, y pane=2.2001, z=1.1896

Figure 1


Roll / Pitch / Yaw Plot
Roll Plot
Roll (rad)
Time
Actual
Estimated
Pitch Plot
Pitch (rad)
Time
Actual
Estimated
Yaw Plot
Yaw (rad)
Time
Actual
Estimated

File 2

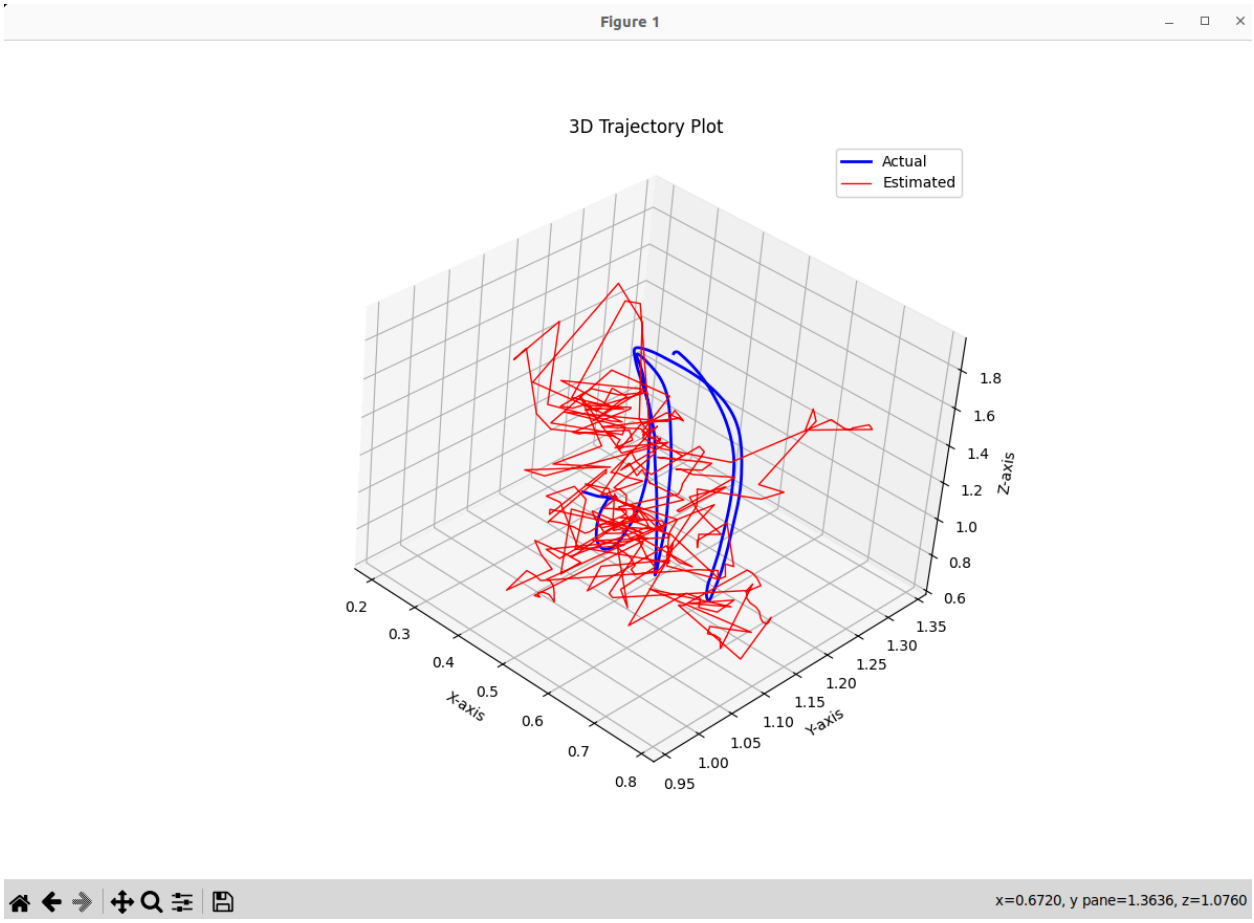


Figure 1
3D Trajectory Plot
Actual
Estimated
X-axis
Y-axis
Z-axis
x=0.6720, y pane=1.3636, z=1.0760


Roll / Pitch / Yaw Plot
Roll Plot
Actual
Estimated
Roll (rad)
Time
Pitch Plot
Pitch (rad)
Yaw Plot
Yaw (rad)

File 3

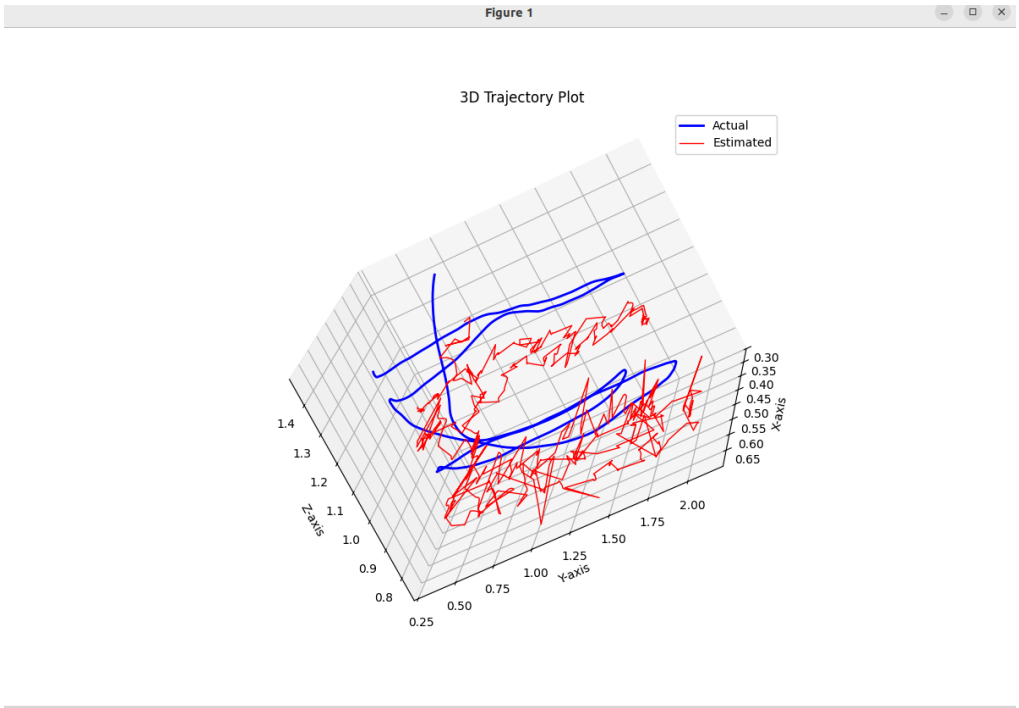
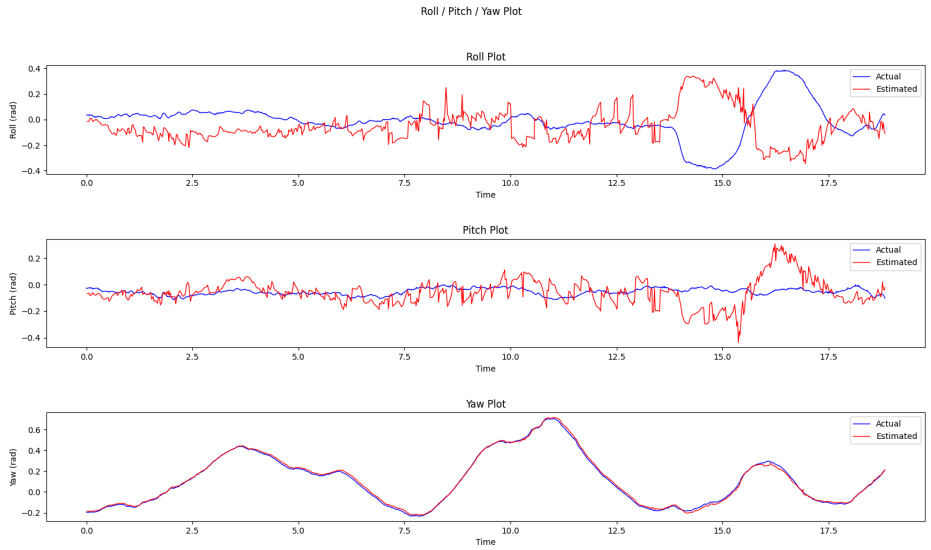
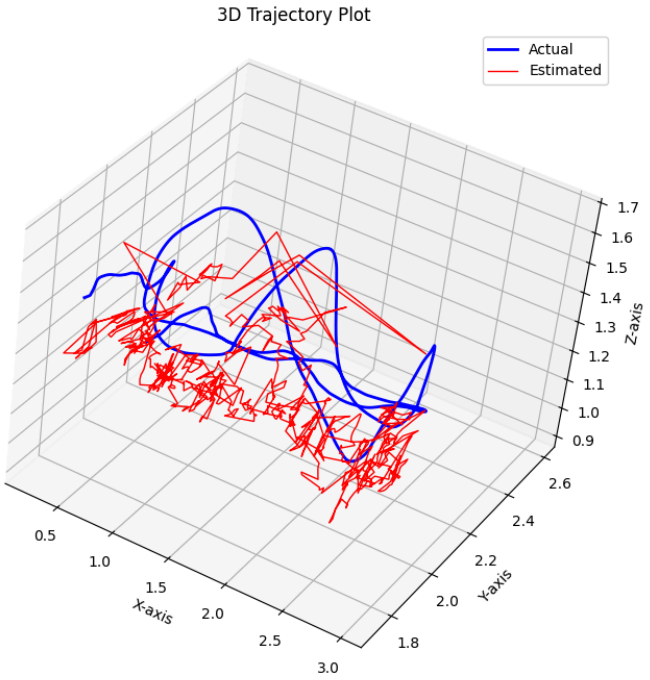


Figure 1
3D Trajectory Plot
Actual
Estimated
Z-axis
Y-axis
X-axis
x pane=0.2997, y=1.5681, z=0.8159


Roll / Pitch / Yaw Plot
Roll Plot
Roll (rad)
Pitch Plot
Pitch (rad)
Yaw Plot
Yaw (rad)
Time
Actual
Estimated

File 4



x=2.3471, y pane=2.6573, z=1.0367

File 5

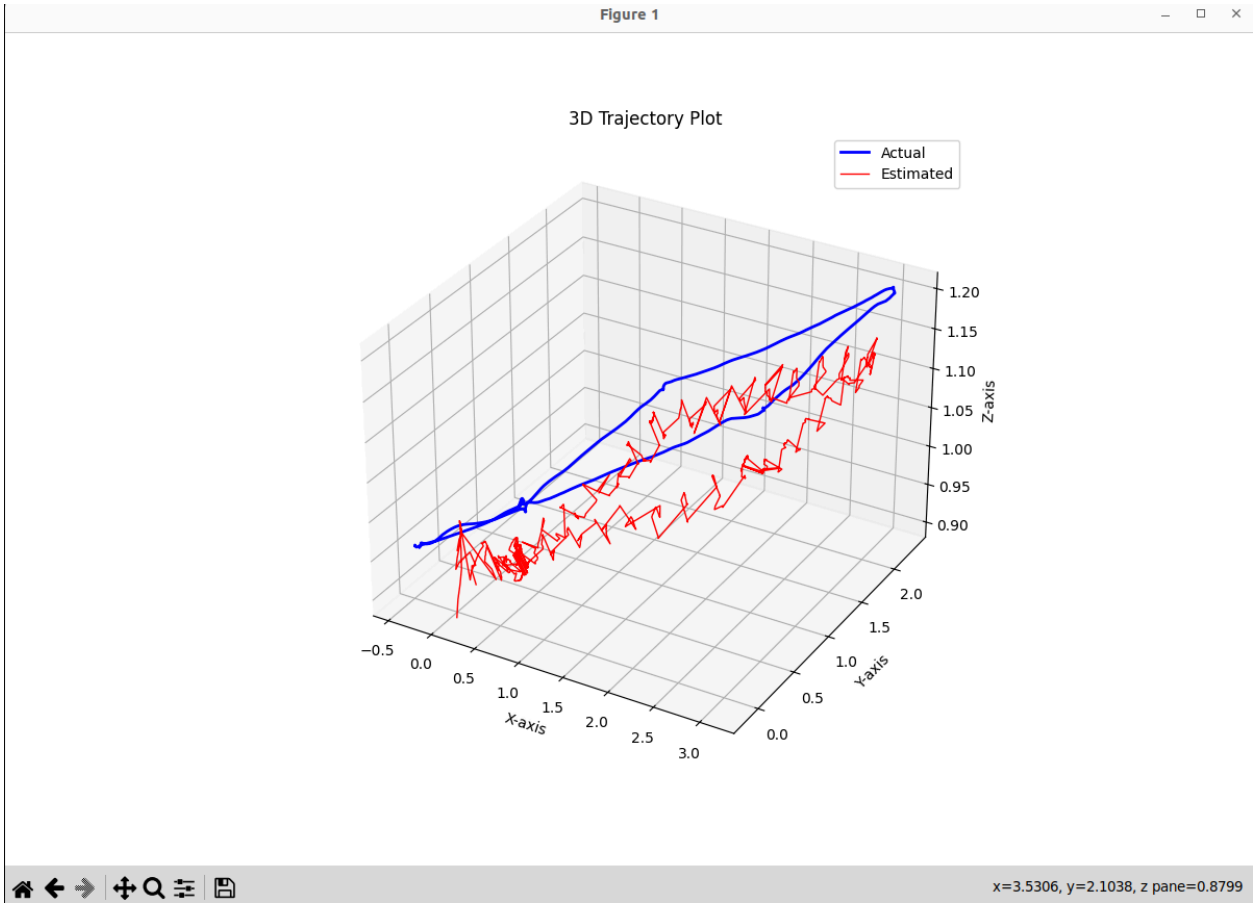
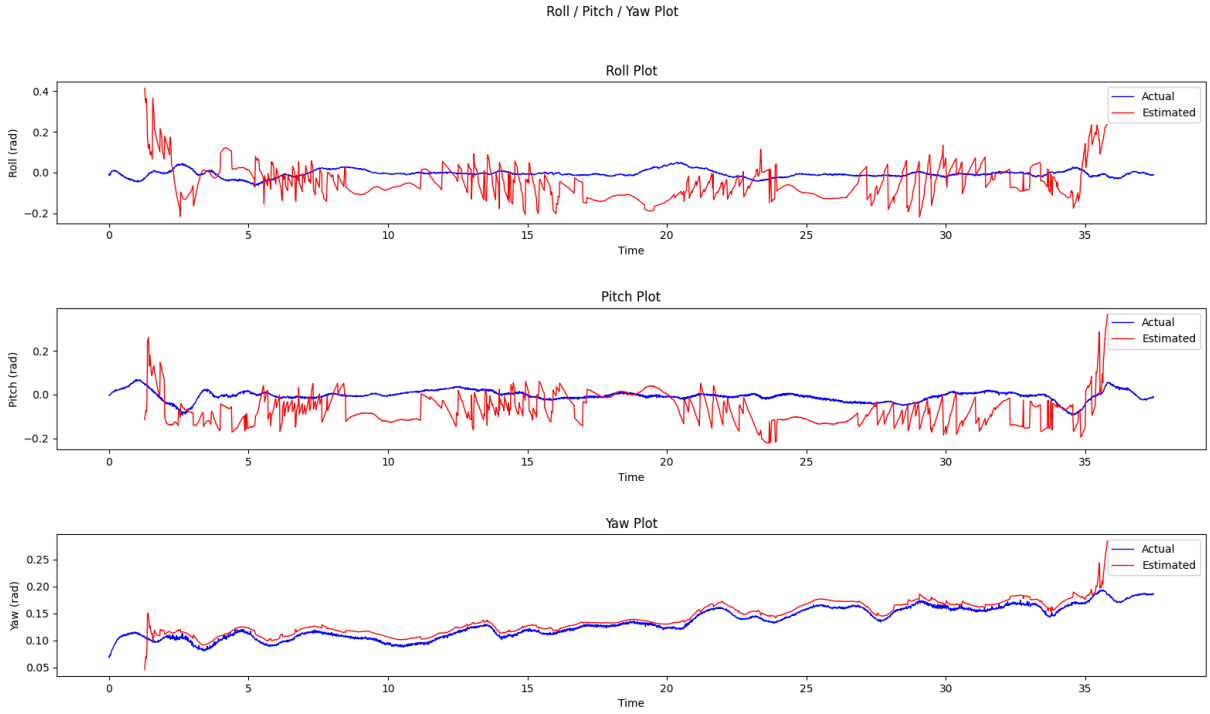


Figure 1
3D Trajectory Plot
Actual
Estimated
1.20
1.15
1.10
1.05
1.00
0.95
0.90
Z-axis
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
X-axis
0.0
0.5
1.0
1.5
2.0
Y-axis
x=3.5306, y=2.1038, z pane=0.8799

Figure 1


Roll / Pitch / Yaw Plot
Roll Plot
Roll (rad)
Time
Actual
Estimated
Pitch Plot
Pitch (rad)
Time
Actual
Estimated
Yaw Plot
Yaw (rad)
Time
Actual
Estimated

File 6

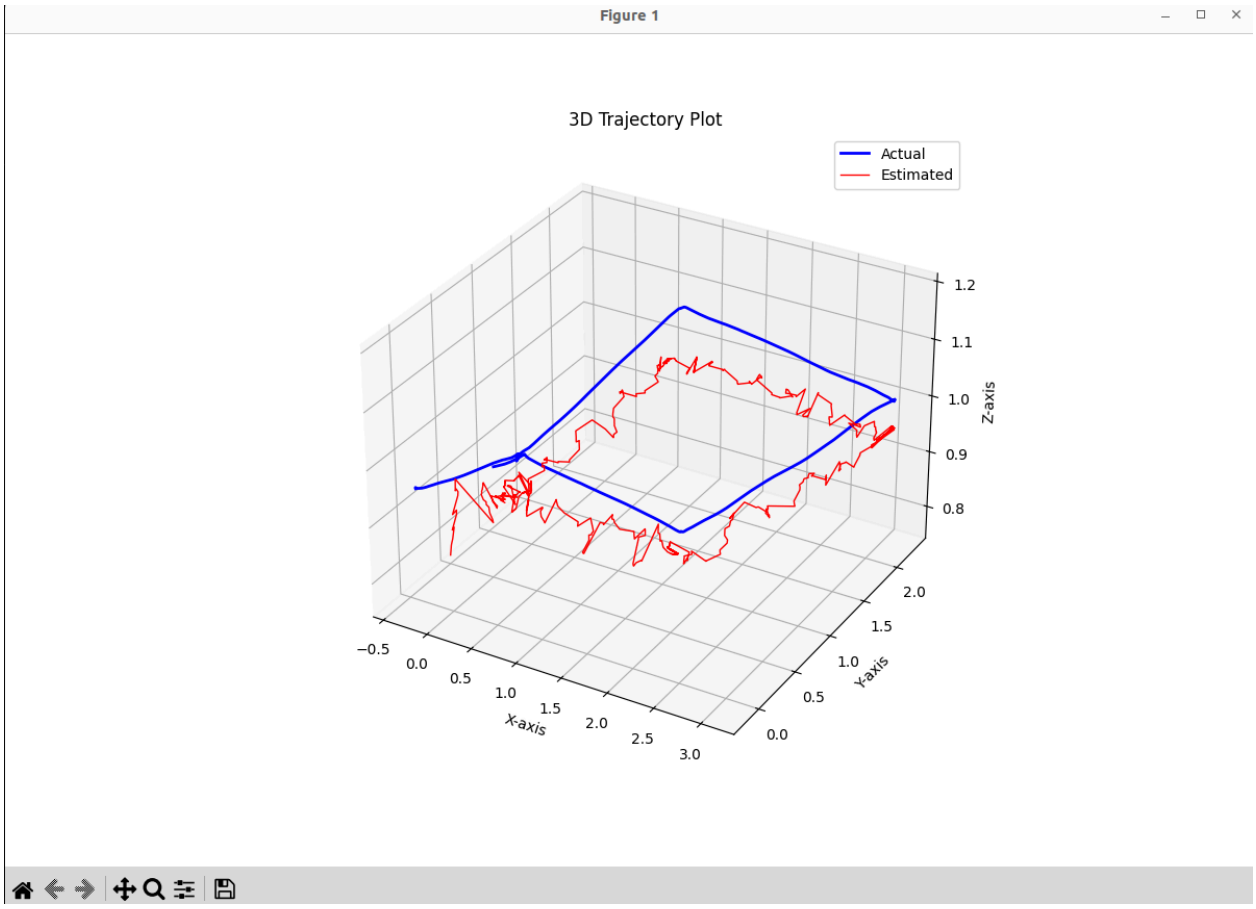


Figure 1
3D Trajectory Plot
Actual
Estimated
X-axis
Y-axis
Z-axis


Roll / Pitch / Yaw Plot
Roll Plot
Roll (rad)
Pitch Plot
Pitch (rad)
Yaw Plot
Yaw (rad)
Time
Actual
Estimated

File 7

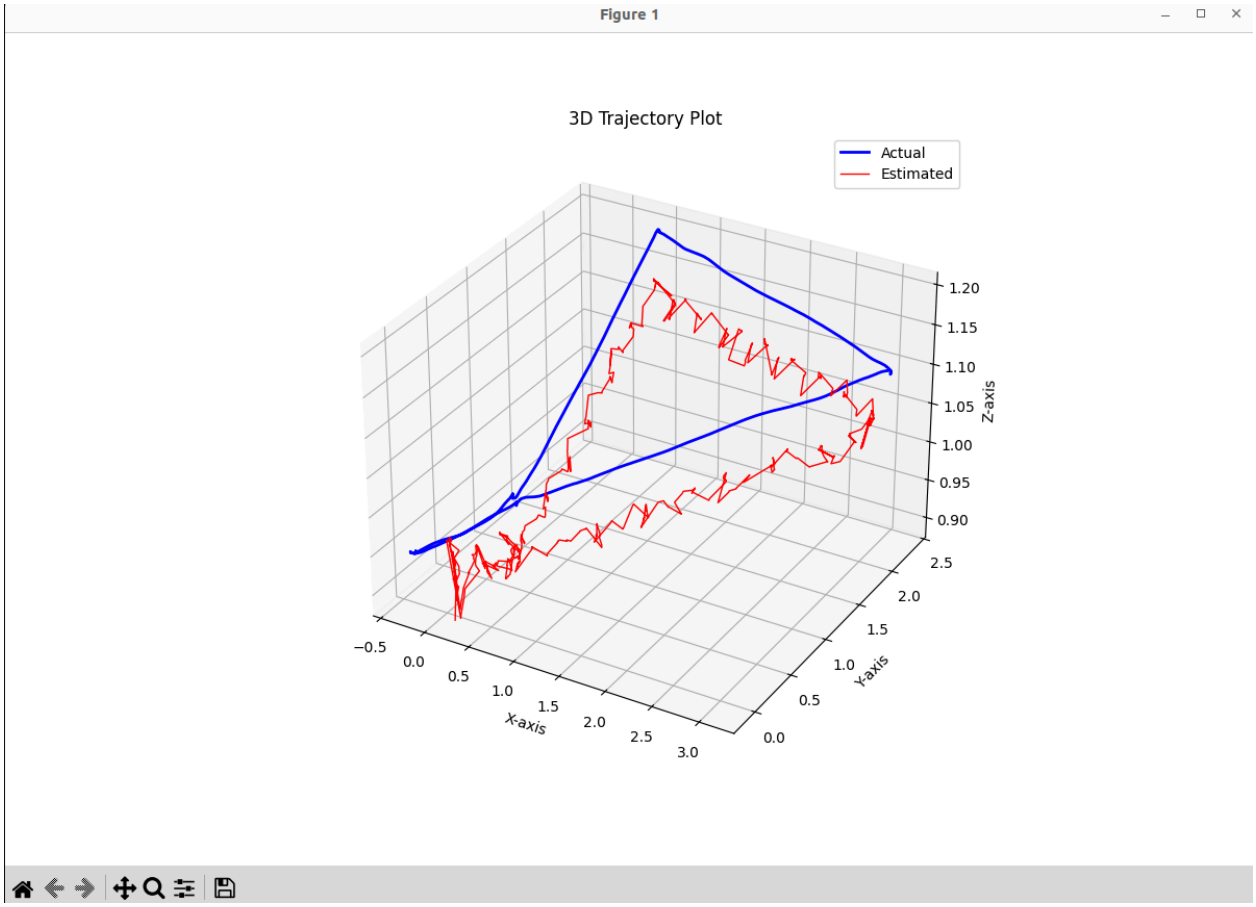
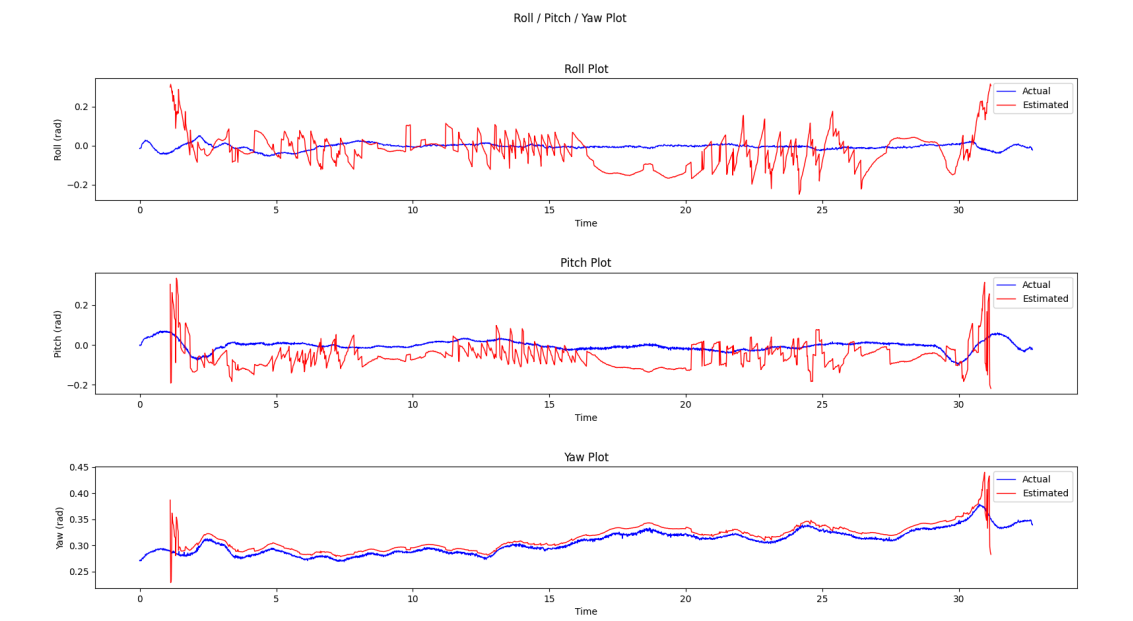


Figure 1
3D Trajectory Plot
Actual
Estimated
X-axis
Y-axis
Z-axis


Roll / Pitch / Yaw Plot
Roll Plot
Roll (rad)
Pitch Plot
Pitch (rad)
Yaw Plot
Yaw (rad)
Time
Actual
Estimated

Covariance Determination

A covariance matrix is computed between the estimated and true positions and orientation. To align the actual data to the vicon data at different time steps from the sensor data linear interpolation is used on the vicon data for the covariance calculation. This covariance matrix will be used in the non-linear lab as the measurement model's covariance matrix for the UKF or EKF filters. To test that the calculation of the matrix is correct a test to determine if it is symmetric and positive definite was implemented in numpy.

- The matrix is tested to be symmetric and positive definite, indicating a correct covariance matrix
    - ```
      # Check if symmetric matrix
      if np.allclose(self.cov_matrix, self.cov_matrix.T):
          print("Covariance matrix is symmetric.")
      else:
          print("Covariance matrix is not symmetric.")

      # Check if positive definite
      eigenvalues = np.linalg.eigvals(self.cov_matrix)
      if np.all(eigenvalues > 0):
          print("Covariance matrix is positive definite.")
      else:
          print("Covariance matrix is not positive definite.")
      ```
- This serves as a quantitative measure of the model's noise.

```
Covariance Matrix:
[[ 0.008  0.001  0.005  0.006  0.005 -0.000]
 [ 0.001  0.004 -0.000  0.004 -0.002 -0.000]
 [ 0.005 -0.000  0.007  0.002  0.004 -0.001]
 [ 0.006  0.004  0.002  0.008  0.001 -0.000]
 [ 0.005 -0.002  0.004  0.001  0.006 -0.000]
 [-0.000 -0.000 -0.001 -0.000 -0.000  0.000]]
Covariance matrix is symmetric.
Covariance matrix is positive definite.
```

Conclusion

The experiment validates the hypothesis that with known AprilTag locations and calibrated camera intrinsics, an effective observation model can be implemented using the Perspective-n-Point algorithm. The estimated trajectory closely matches the Vicon ground truth. This model can serve as a reliable localization method in structured environments where fiducial markers are available.
```