Perspective-N-Point Observation Model

Ashwin Disa
Robotics Engineering
Worcester Polytechnic Institute
Email: amdisa@wpi.edu

I. Introduction

Accurate pose estimation is a fundamental requirement in robotics, especially for autonomous aerial systems navigating through known environments. This project focuses on implementing a computer vision-based observation model using the Perspective-n-Point (PnP) method to estimate the pose of a Nano+ quadrotor. The resulting observation model is intended to serve as the measurement model for a subsequent nonlinear filtering project, such as a Particle Filter.

The data used in this project were collected from a downward-facing camera mounted on the drone as it was moved through a trajectory above a mat populated with AprilTags. The AprilTags are arranged in a well-defined 12 × 9 grid with known inter-tag distances, providing a robust reference map in the world frame. This spatial configuration allows us to establish precise 3D-2D correspondences necessary for solving the PnP problem.

The camera calibration parameters, including the intrinsic matrix and distortion coefficients, are provided and are critical for accurate reprojection of world points onto the image plane. Additionally, the relative transformation between the camera and drone body frames must be accounted for to express pose estimates in the robot frame. Ground truth data collected using a motion capture system (Vicon) is available for evaluation but not directly used during estimation.

This project involves several key tasks: (1) developing the pose estimation function based on the PnP method, (2) visualizing and analyzing the estimated drone trajectory compared to ground truth, and (3) quantifying the reliability of the observation model through covariance estimation. The ultimate goal is to create a reliable and reusable module for vision-based localization that integrates seamlessly into future state estimation pipelines.

II. Methodology

This section outlines the implementation details of the Perspective-n-Point (PnP) based observation model used to estimate the pose of a Nano+ quadrotor. The methodology consists of preprocessing camera and tag map data, pose estimation using the PnP algorithm, visualization of the estimated trajectory against ground truth, and estimation of the measurement covariance for use in subsequent nonlinear filtering.

III. Methodology

The camera's intrinsic parameters were extracted from the parameters.txt file. In particular, we obtain the camera matrix

$$\mathbf{K} = \begin{bmatrix} 314.1779 & 0 & 199.4848 \\ 0 & 314.2218 & 113.7838 \\ 0 & 0 & 1 \end{bmatrix}$$

and the distortion coefficients (radial $k_1 = -0.438607,\ k_2 = 0.248625,\ k_3 = -0.0911$, tangential $p_1 = 0.00072,\ p_2 = -0.000476$). AprilTags are arranged in a 12×9 grid on the map, with nominal spacing $0.152\,\text{m}$ except between columns 3–4 and 6–7 where spacing is $0.178\,\text{m}$. The map origin is defined at the top-left corner of the top-left tag, with the x-axis pointing down and the y-axis pointing right in the world frame. The rigid transform from the camera frame to the drone body frame is given by the translation

$$\mathbf{t}_{\text{cam}\to\text{drone}} = \begin{bmatrix} -0.04 \\ 0 \\ -0.03 \end{bmatrix}$$

and the rotation

$$\mathbf{R}_{\text{cam}\to\text{drone}} = \mathbf{R}_{\text{xyz}}(-\pi,\, 0,\, -\tfrac{\pi}{4})\,.$$

To generate the 3D world coordinates of each tag's corners, we first compute the tag's lower-left reference point using the get_tag_position function. Given a tag ID and its layout in a matrix, this function locates the tag's row and column and converts them to world-frame (x, y) coordinates, applying default and exception spacings as needed. Next, get_corners adds the tag size to that reference point to yield the four corner coordinates {top_left, top_right, bottom_right, bottom_left} for each tag.

Detected image-plane corners are similarly organized by the get_pixel_corners function. It takes four $2 \times N$ arrays of raw corner pixel coordinates and a matching list of tag IDs, then constructs a dictionary that maps each tag ID to its four named pixel-corner vectors.

Pose estimation is performed by assembling these 3D–2D correspondences and invoking the get_camera_pose function. Internally, it stacks all tag world-plane corners (with $Z = 0$) into object_points and the detected image corners into image_points, then calls OpenCV's solvePnP with $\mathbf{K}$ and the distortion coefficients to recover the rotation vector $\mathbf{rvec}$ and translation $\mathbf{tvec}$. On success, it returns $(\mathbf{rvec}, \mathbf{tvec})$; otherwise it reports a failure.

The higher-level estimate_pose function integrates this PnP step into the full per-frame pipeline. Given a data record containing the image, tag IDs, and corner arrays $p1, \ldots, p4$, it reshapes and routes these through get_corners and get_pixel_corners, calls get_camera_pose, and then converts $\mathbf{rvec}$ into a 3×3 rotation matrix via Rodrigues. The camera position in the world frame is computed as

$$\mathbf{p}_{\mathrm{cam}} = -\mathbf{R}^{\top}\,\mathbf{tvec}\,,$$

and the drone body-frame position follows by applying the known camera-to-drone translation and rotation offsets:

$$\mathbf{p}_{\mathrm{drone}} = \mathbf{p}_{\mathrm{cam}} - \mathbf{R}^{\top}\,\mathbf{R}_{\mathrm{cam}\to\mathrm{drone}}^{\top}\,\mathbf{t}_{\mathrm{cam}\to\mathrm{drone}}\,.$$

The align_ground_truth function takes as input two time-indexed maps: estimated_dict, which associates each estimation timestamp with its corresponding pose (rotation and translation vectors), and ground_truth_dict, which provides high-frequency ground truth poses indexed by their own timestamps. It begins by sorting both sets of timestamps. For each estimation timestamp t_{est}, it employs a binary search (bisect_left) on the sorted ground truth timestamps to locate the insertion index. If this index is at the boundaries, the earliest or latest ground truth timestamp is chosen; otherwise, it compares the two neighboring timestamps and selects the one closest to t_{est}. The function then builds a new dictionary aligned_ground_truth that maps each t_{est} to the ground truth pose at the chosen closest timestamp. In this way, every estimated pose is paired with the most temporally proximate ground truth sample, facilitating direct error computation and trajectory comparison.

A. *Estimate Covariance*

The function *estimate_covariance* computes the empirical covariance of the six-dimensional pose residuals between estimated and ground-truth trajectories. It begins by initializing an index counter and an empty list of residual vectors. For each timestamped entry in the estimated dictionary, the routine calls the existing pose-estimation function on the corresponding frame to obtain the rotation matrix R_{est}. Frames for which either the position or rotation estimate is invalid are skipped. Otherwise, the aligned ground-truth array is retrieved; its first three components are the true (x, y, z) position and the next three are the true roll ϕ_{gt}, pitch θ_{gt}, and yaw ψ_{gt}. The estimated rotation R_{est} is converted into Euler angles (ϕ, θ, ψ) in the "xyz" convention, and a wrap-to-zero correction is applied to the roll component offset by π. the residual

$$\mathbf{v}_t \;=\; \mathbf{p}_{\mathrm{gt}}(t) \;-\; \hat{\mathbf{p}}$$

is computed and stored. After processing all valid frames, the residuals form an $n \times 6$ matrix V, where each row is one $\mathbf{v}_t$. If fewer than two samples are available, the function raises an error. Otherwise, it computes the sample covariance matrix

$$R_{\mathrm{cov}} = \frac{1}{n-1}\,(V - \overline{V})^{\top}\,(V - \overline{V})\,,$$

where $\overline{V}$ is the row-wise mean of V. The resulting 6×6 covariance matrix quantifies the empirical uncertainty in both position and orientation and is returned for downstream filtering or analysis. The combined covariance matrix from all 8 trajectories is given below.

$$R = \begin{bmatrix} 5.20 \times 10^{-3} & 4.76 \times 10^{-4} & -1.47 \times 10^{-4} & -3.35 \times 10^{-3} & 2.91 \times 10^{-3} & 1.43 \times 10^{-4} \\ 4.76 \times 10^{-4} & 5.43 \times 10^{-3} & -3.44 \times 10^{-4} & -4.32 \times 10^{-3} & -3.11 \times 10^{-3} & 1.12 \times 10^{-4} \\ -1.47 \times 10^{-4} & -3.44 \times 10^{-4} & 5.89 \times 10^{-4} & 2.18 \times 10^{-4} & 1.74 \times 10^{-4} & -2.68 \times 10^{-5} \\ -3.35 \times 10^{-3} & -4.32 \times 10^{-3} & 2.18 \times 10^{-4} & 6.23 \times 10^{-3} & 5.18 \times 10^{-4} & -2.02 \times 10^{-4} \\ 2.91 \times 10^{-3} & -3.11 \times 10^{-3} & 1.74 \times 10^{-4} & 5.18 \times 10^{-4} & 4.62 \times 10^{-3} & 3.70 \times 10^{-5} \\ 1.43 \times 10^{-4} & 1.12 \times 10^{-4} & -2.68 \times 10^{-5} & -2.02 \times 10^{-4} & 3.70 \times 10^{-5} & 3.40 \times 10^{-5} \end{bmatrix}$$

B. *Error Metrics*

To evaluate the accuracy of our observation model, we compute both positional and angular discrepancies between estimated and ground-truth poses. First, we visualize the estimated and ground-truth trajectories in 3D by plotting the full orientation axes at each timestamp. We then quantify positional accuracy via the instantaneous Euclidean error

$$e_t = \left\|\mathbf{p}_{\mathrm{est}}(t) - \mathbf{p}_{\mathrm{gt}}(t)\right\|_2$$

and the mean positional error

$$\bar{e} = \frac{1}{N} \sum_{t=1}^{N} e_t \, .$$

The MSE is found to be consistently below the 0.2 meter mark. In addition to position, we assess orientation accuracy by extracting roll ϕ, pitch θ, and yaw ψ from both the estimated rotation matrix and the aligned ground-truth. These angle trajectories are plotted over time in three separate subplots—one each for ϕ, θ, and ψ—showing both the estimated and ground-truth curves. This angular comparison reveals biases or drifts in the orientation estimates and complements the positional error analysis to provide a complete quantitative assessment of the model's performance.

IV. Conclusion

In this work, we have presented a PnP-based pose estimation system using AprilTags. The resulting 6×6 residual covariance matrix exhibits diagonal position variances of approximately $5.2 \times 10^{-3}\,\mathrm{m}^2$ in X, $5.4 \times 10^{-3}\,\mathrm{m}^2$ in Y, and $5.9 \times 10^{-4}\,\mathrm{m}^2$ in Z, corresponding to standard deviations of roughly $0.07\,\mathrm{m}$ and $0.02\,\mathrm{m}$, respectively. Orientation variances range from $6.2 \times 10^{-3}\,\mathrm{rad}^2$ ($\sigma \approx 0.08\,\mathrm{rad} \approx 4.6°$) down to $3.4 \times 10^{-5}\,\mathrm{rad}^2$ ($\sigma \approx 0.006\,\mathrm{rad} \approx 0.3°$). Off-diagonal entries on the order of 10^{-4}–10^{-3} indicate moderate coupling between axes. Moreover, the instantaneous Euclidean error and its mean remain below $0.2\,\mathrm{m}$ on average. The next step is to incorporate a nonlinear filter to better estimate the pose of the drone using a process model which will utilize the IMU sensor.

V. Results

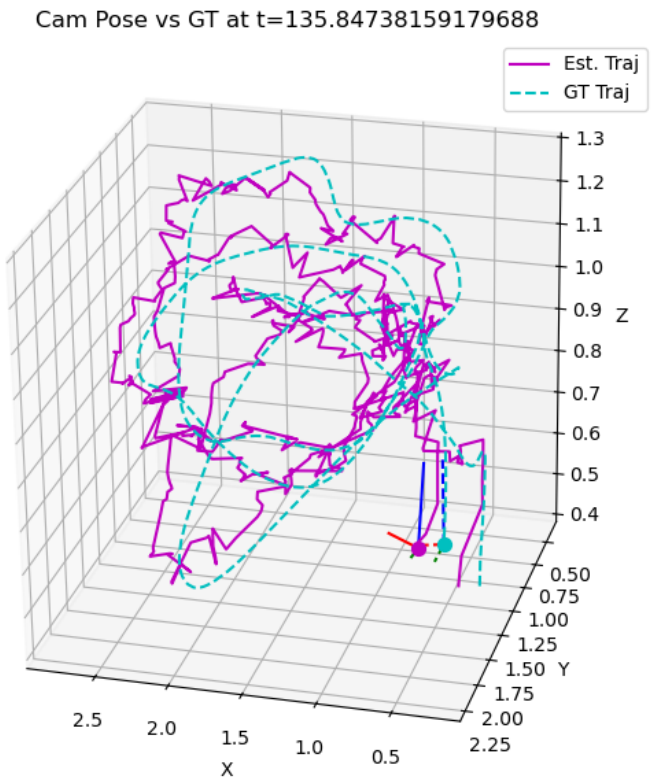

Fig. 1: Trajectory 1.

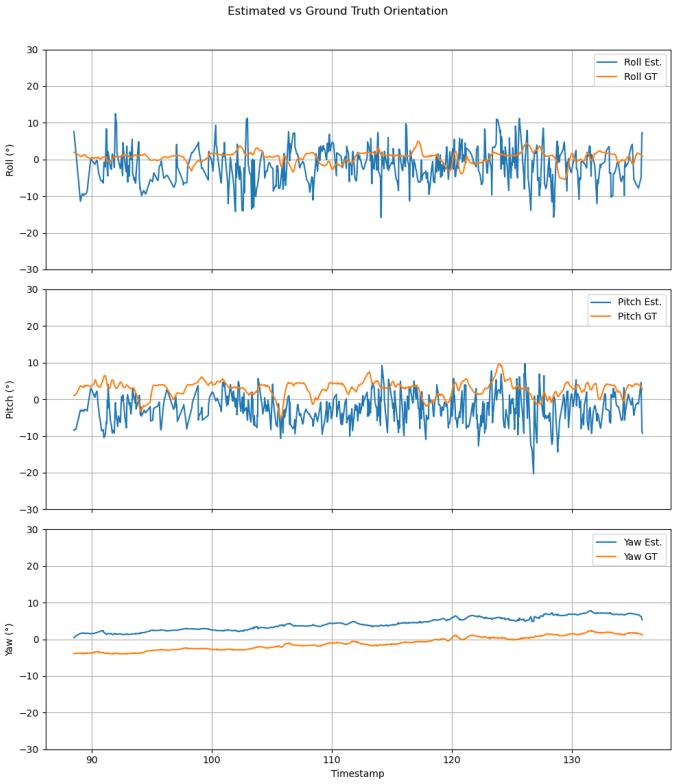

Fig. 2: Ground truth vs Estimated Orientation for trajectory 1.

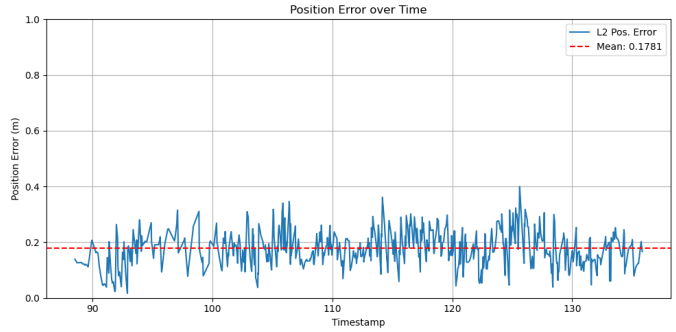

Fig. 3: MSE of Ground truth and estimated position for trajectory 1.

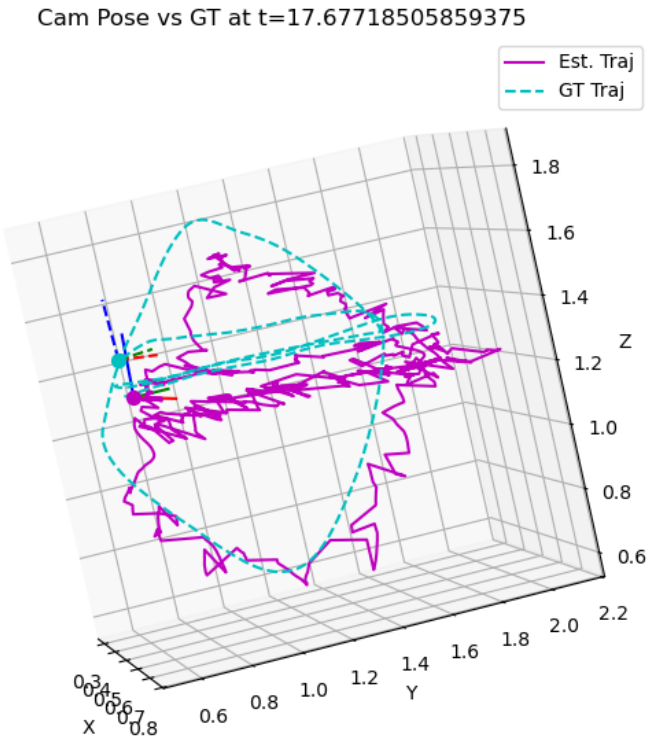

Fig. 4: Trajectory 2.

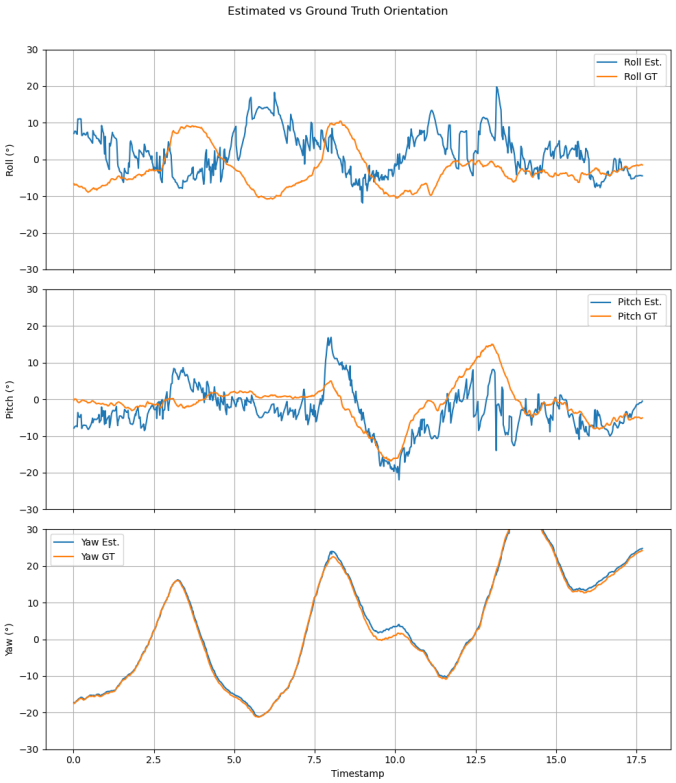

Fig. 5: Ground truth vs Estimated Orientation for trajectory 2.

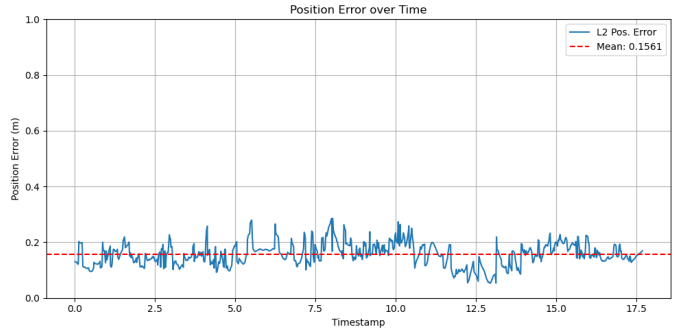

Fig. 6: MSE of Ground truth and estimated position for trajectory 2.

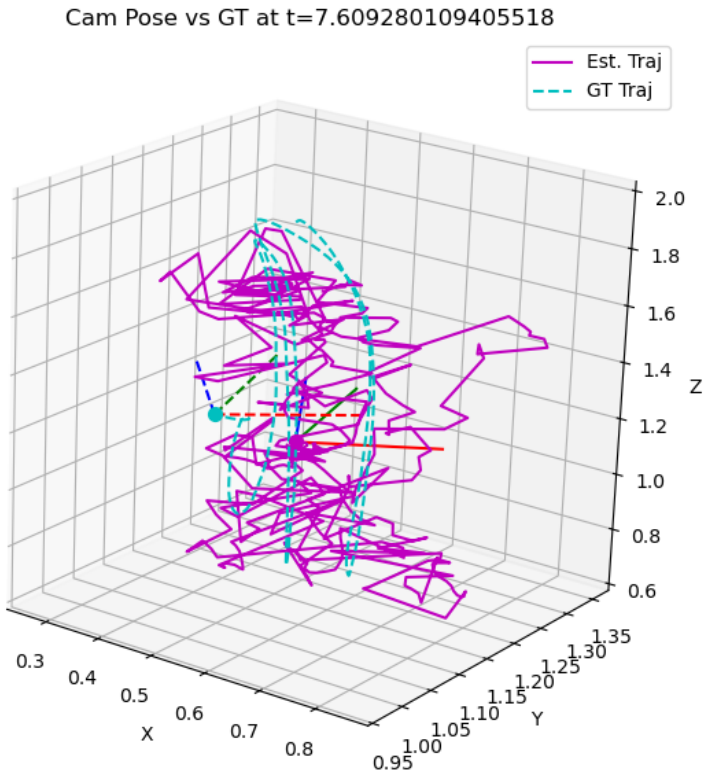

Fig. 7: Trajectory 3.

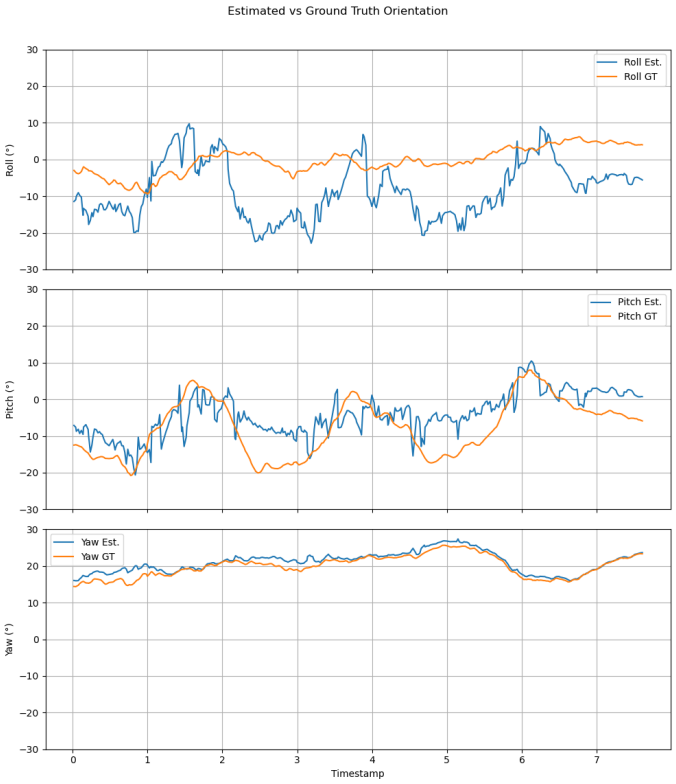

Fig. 8: Ground truth vs Estimated Orientation for trajectory 3.

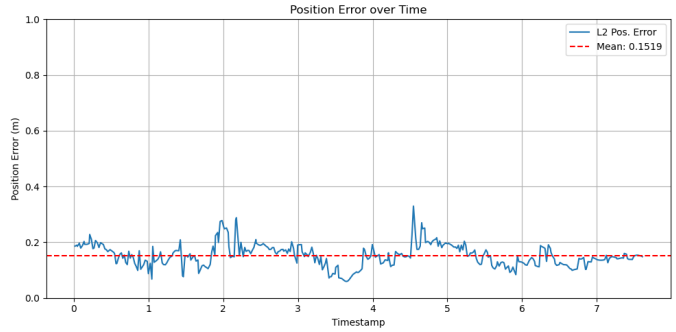

Fig. 9: MSE of Ground truth and estimated position for trajectory 3.

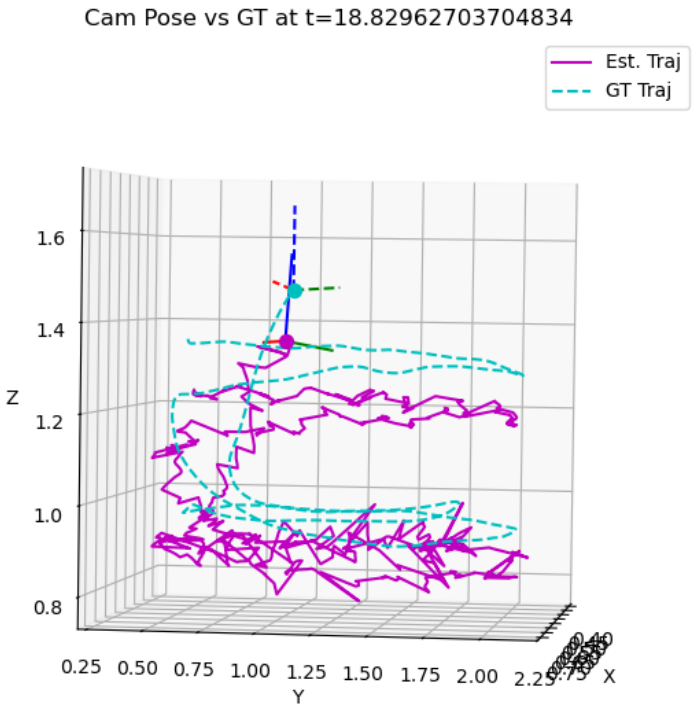

Fig. 10: Trajectory 4.

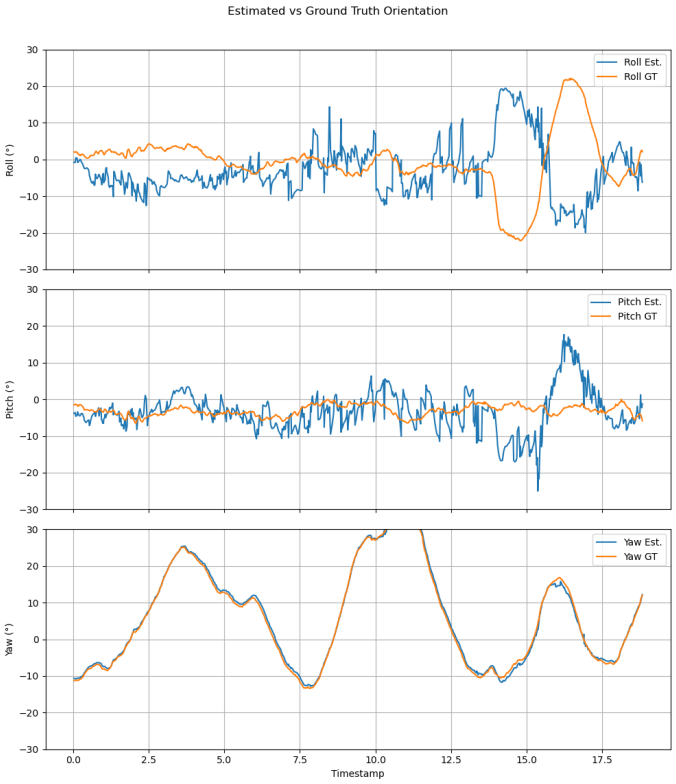

Fig. 11: Ground truth vs Estimated Orientation for trajectory 4.

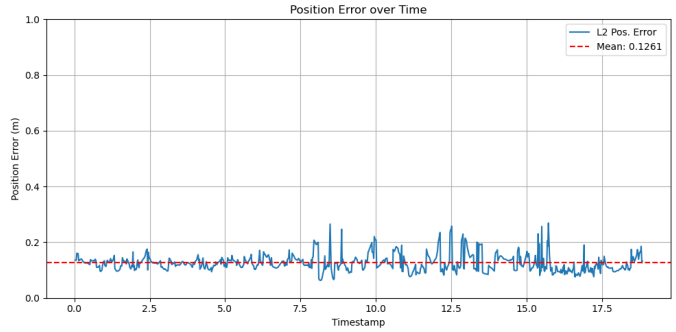

Fig. 12: MSE of Ground truth and estimated position for trajectory 4.

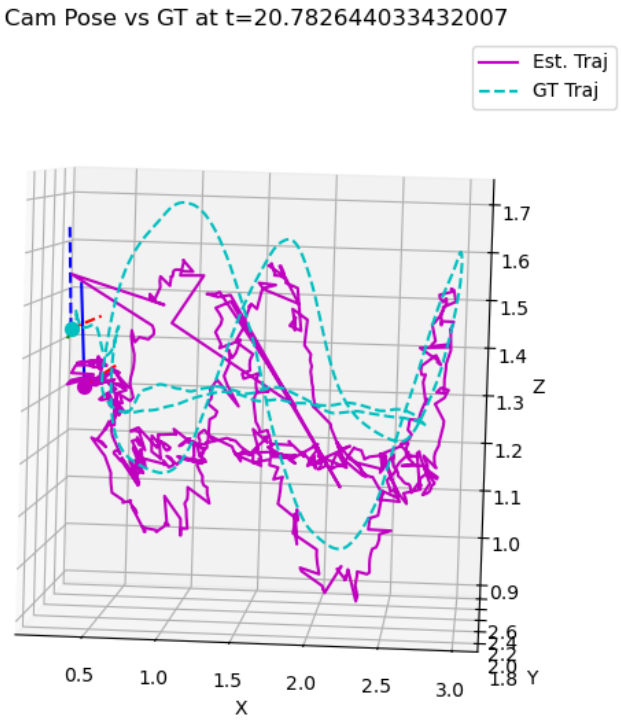

Fig. 13: Trajectory 5.

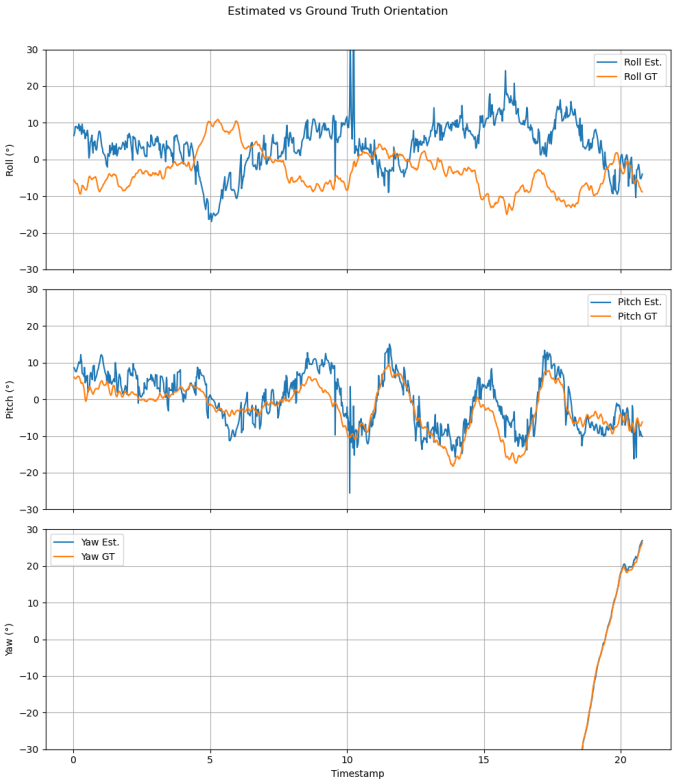

Fig. 14: Ground truth vs Estimated Orientation for trajectory 5.

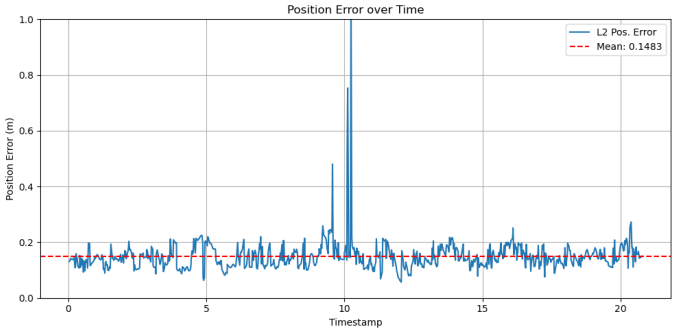

Fig. 15: MSE of Ground truth and estimated position for trajectory 5.

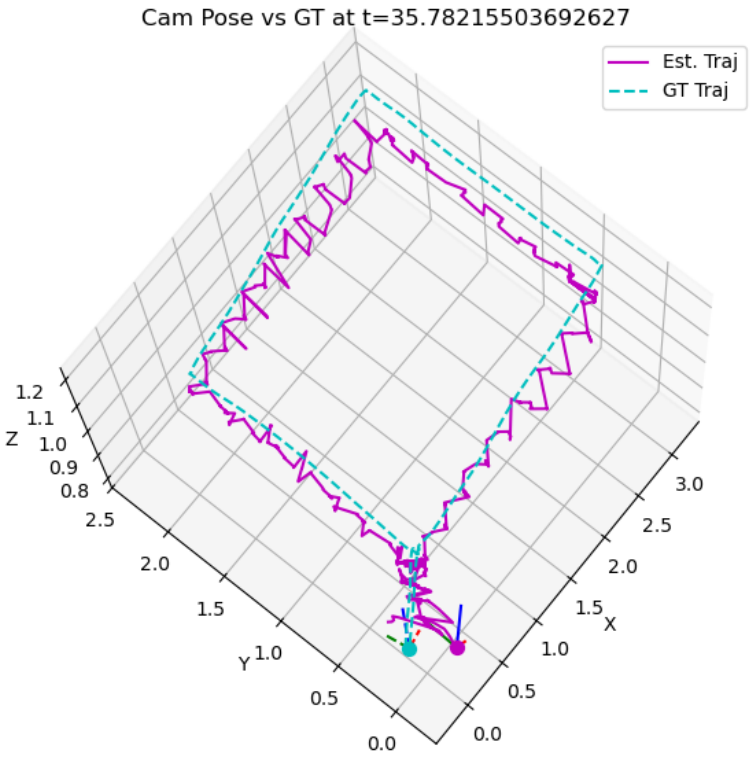

Fig. 16: Trajectory 6.

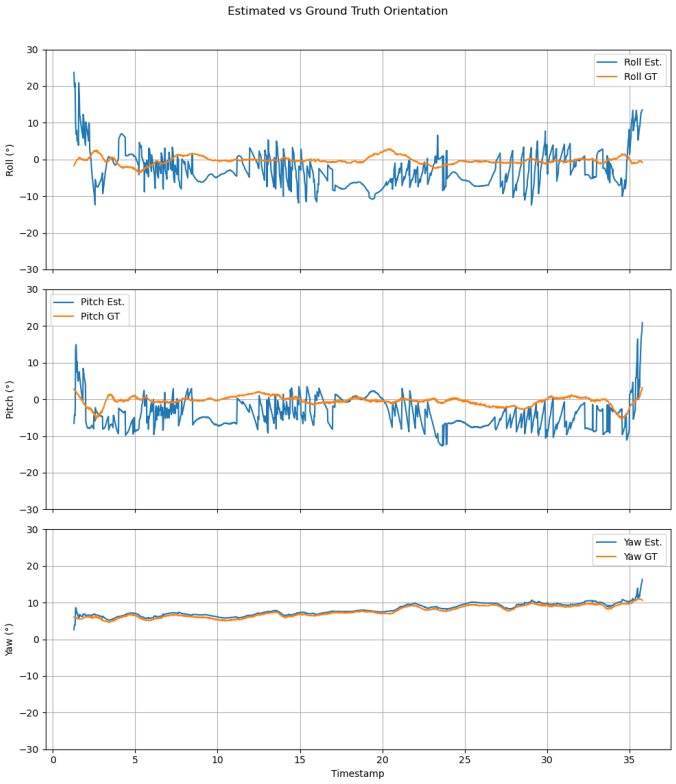

Fig. 17: Ground truth vs Estimated Orientation for trajectory 6.

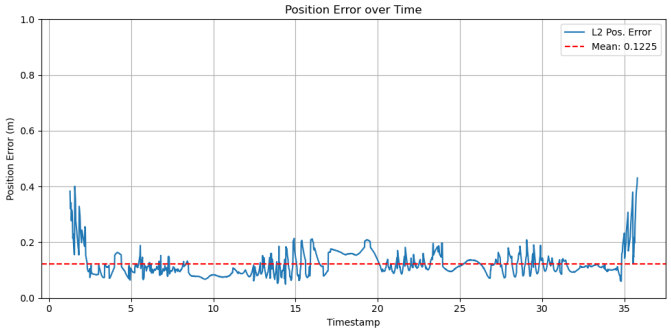

Fig. 18: MSE of Ground truth and estimated position for trajectory 1.

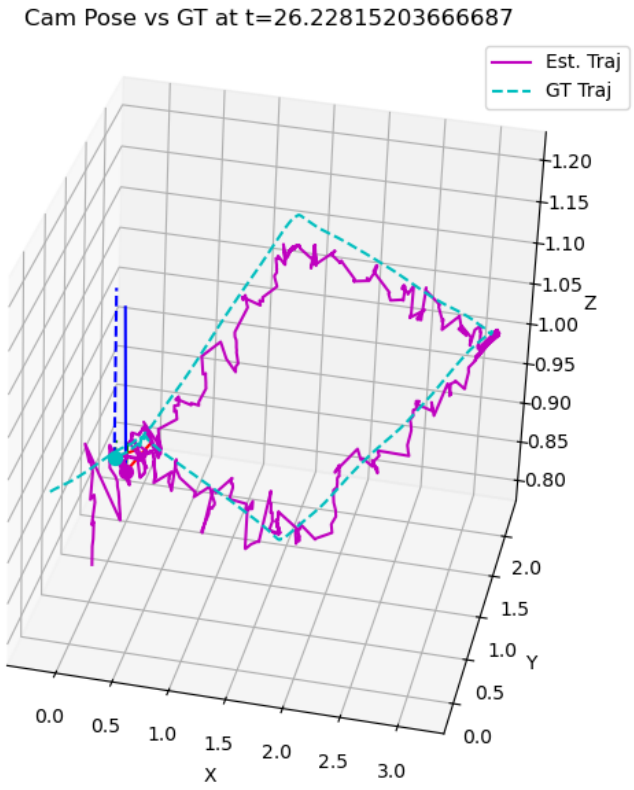

Fig. 19: Trajectory 7.

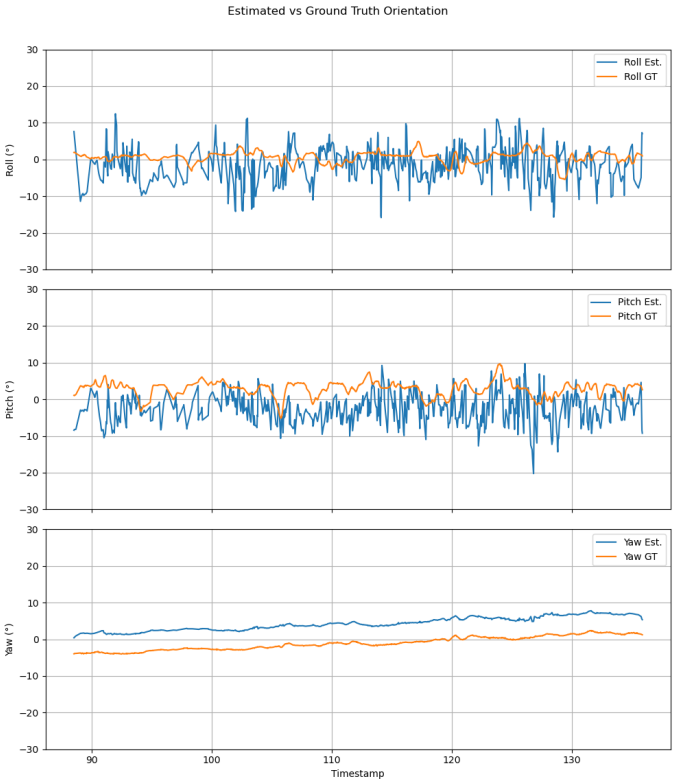

Fig. 20: Ground truth vs Estimated Orientation for trajectory 7.

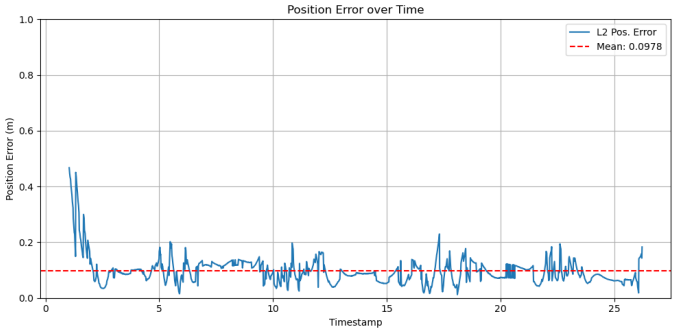

Fig. 21: MSE of Ground truth and estimated position for trajectory 7.

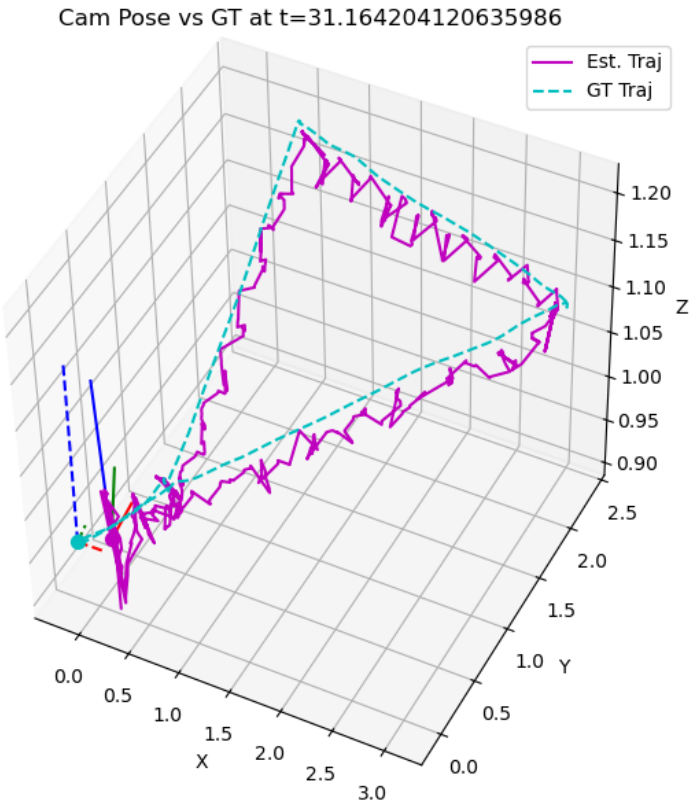

Fig. 22: Trajectory 8.

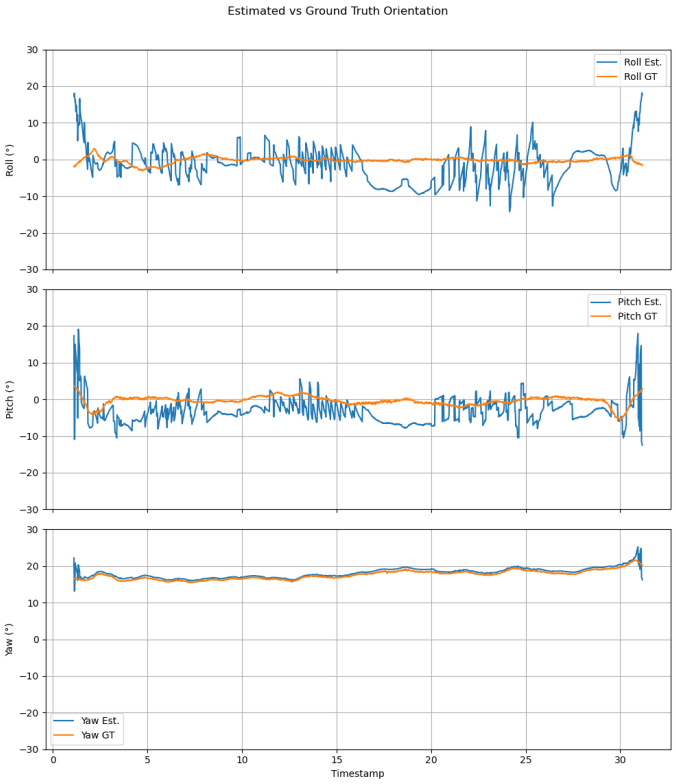

Fig. 23: Ground truth vs Estimated Orientation for trajectory 8.

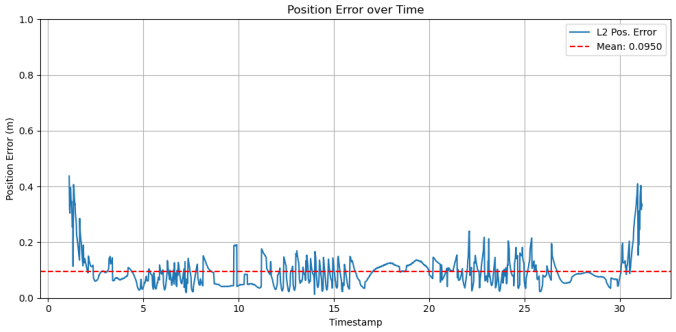

Fig. 24: MSE of Ground truth and estimated position for trajectory 8.