RBE 595 - Advanced Navigation
Lab #3 - Nonlinear Filters
Jason Patel

Introduction

State estimation plays a critical role in mobile robotics and autonomous systems, especially for localization and navigation in uncertain environments. This report presents a comparative study of two widely used nonlinear state estimation methods: the Unscented Kalman Filter (UKF) and the Particle Filter (PF). The focus is on estimating the full 6-DOF pose (position and orientation) of a drone using visual and inertial sensor data.

In this lab we experimented with 2 types of filters to fuse inertial data from the IMU with measurement data from a PersepectiveNPoint model in order to perform localization of a Quadcopter.

Problem Description

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

- AprilTag detections from camera images (providing sparse pose measurements)
- IMU data (accelerometer and gyroscope readings)

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.

State Space

A 15-dimensional state vector: Position (3), orientation (3), velocity (3), gyro bias (3), accel bias (3), was used to represent the state space.

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

Process Model

The process model is based on the acceleration output from the IMU. We account for the higher order dynamics of the drone (pitching and rolling) and account for the change in the IMU output with respect for the gravity vector.
The gyro and accelerometers are modeled with noise as follows:

$$\hat{\boldsymbol{\omega}} = \boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g$$

$$\hat{\boldsymbol{a}} = R(\mathbf{q})^T(\mathbf{a} - \mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a$$

Where Uw= $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z] = \mathbf{u}_\omega$

And Ua= $\boldsymbol{a} = [\ddot{p}_x, \ddot{p}_y, \ddot{p}_z] = \mathbf{u}_a$

Continuous time process model:

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

, where Uw and Ua are the commanded angular velocity and linear acceleration, g is the gravity vector, and G(g) and R(q) are defined as follows:

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

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

Measurement Model

The measurement model is a simple vector plus measurement noise (which is omitted from the UKF filter).

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

The h(x) function maps the measurement to the state space using a simple 6x15 matrix. A simple example as show here:

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

Methods

Both filters are implemented in Python. Key classes include:

- `MeasurementData`: Loads and processes AprilTag and IMU data
- `Localization`: Runs either UKF or PF over datasets
- `UkfFilter` and `ParticleFilter`: Contain specific implementations

Each filter estimates the state over time and compares against ground truth using:

- Trajectory plots
- Orientation plots (roll, pitch, yaw)
- Covariance analysis
- RMSE calculations for position

Unscented Kalman Filter

Used the 15-dimensional state representation. Sigma points are generated using the Julier method from the original UKF paper. The measurement model is a linear mapping from the visual sensor model (April Tag locations in the global frame) using the H matrix. This method uses deterministic sampling of the sigma points to propagate uncertainty. The propagation model, or state transition model, does not include added noise directly, and is non-linear. Because the UKF algorithm does not require direct linearization of the state transition model, the f(x) of the propagation was left as a non-linear model.
The filterpy library from the github repo of the particle filter book, and was used to perform the sigma point propagation. The covariance of the measurement model was originally taken from a calculated covariance matrix performed on matlab file 5, but was later replaced with a custom measurement covariance matrix.

Particle Filter

As the particle filter is a monte carlo based approach without a covariance matrix to introduce noise, the noise was added to the Ua and Uw vectors directly. The particle filter was evaluated with 250, 500, 750, 1000, 2000, 3000 particle filters in the experiment.

RMSE

RMSE, root-mean-squared error, was used to compare measurement model's estimate (April tags) to the ground truth vicon data, and then further to compare the filters estimates to the vicon data, and then a simple comparison difference between the 2 to determine if the filtered results yielded additional accuracy, which I call RMSE Advantage.

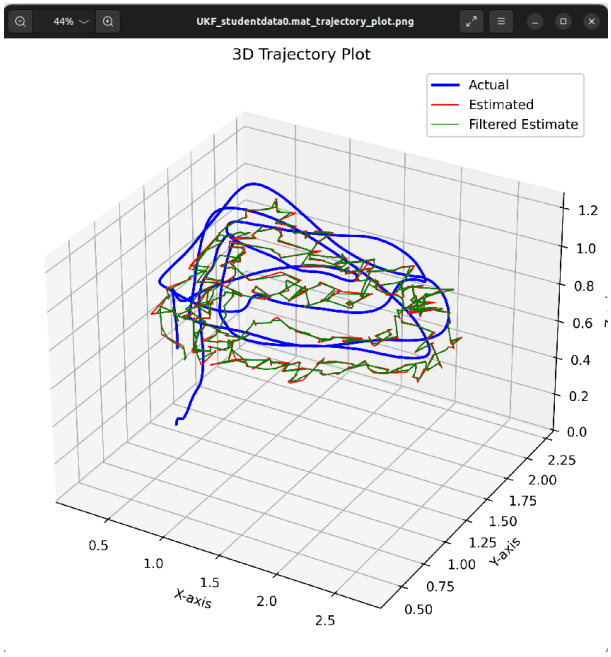
$$RMSE = \sqrt{\frac{1}{n}\sum_{i}^{n}(x_i - \bar{x}_i)}\,)$$

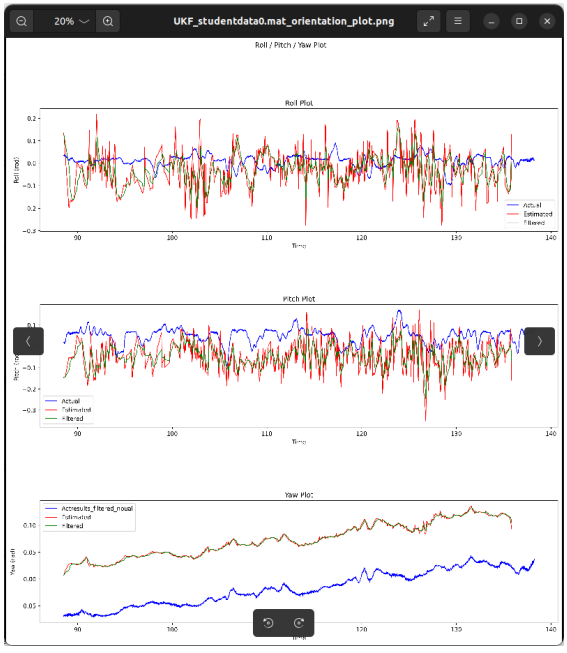
Results

Trajectory and orientation plots are saved in `./plots/`

UKF Results

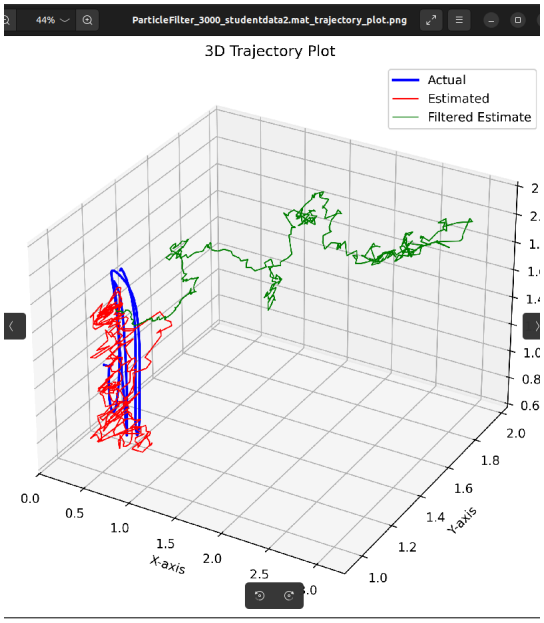
- Run over all 8 datasets
- Produces stable, however jagged, trajectory estimates
- RMSE advantage (average difference between measurement model RMSE and UKF estimates across datasets): 0.10

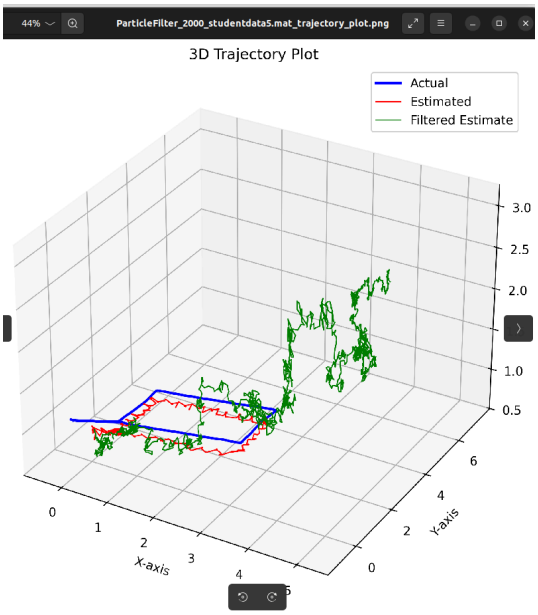




Particle Filter Results

- Run over all 8 datasets
- Produces unstable and jagged trajectory estimates
- RMSE advantage (average difference between measurement model RMSE and UKF estimates across datasets): 1.08 when particle count is 2000





Summary of Results

Metric	UKF	PF
Model Assumption	Gaussian	Nonparametric
Computation	Efficient	Scales with particles
Accuracy	Good for mild nonlinearity	Better for high nonlinearity
RMSE Stability	High	Varies with particle count

UKF is efficient and suitable when the noise is approximately Gaussian and system nonlinearity is moderate. PF offers better flexibility and accuracy under high noise and nonlinearity but at a computational cost.

Conclusion

Both the UKF and PF can accurately estimate 6-DOF pose using camera and IMU data. In this experiment the Particle Filter did not provide the correct output as expected. The UKF provides a reliable and efficient approach for real-time applications, while the PF should deliver higher accuracy with enough computational resources. The choice depends on the application constraints and noise characteristics.