NoisyTruth

Unscented Kalman Filter (UKF) — From Scratch Implementation

1 Introduction

Real-world systems are never perfectly observable. Sensors are noisy, models are imperfect, and uncertainty is unavoidable. This repository implements the **Unscented Kalman Filter (UKF)** from scratch to address the problem of reliable state estimation in nonlinear systems under noise.

2 Why Noise Exists in Sensor Models

In practice, no sensor measures the true state directly. Every measurement is corrupted by noise due to:

- Electronic noise (thermal, quantization)
- Environmental disturbances
- Sensor resolution limits
- Calibration errors
- Time delays and synchronization issues

A generic sensor measurement model is:

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k \tag{1}$$

where:

- $\mathbf{x}_k$ is the true system state
- $h(\cdot)$ is the (possibly nonlinear) measurement function
- $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R})$ is the measurement noise

Similarly, system dynamics are modeled as:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1} \tag{2}$$

where $\mathbf{w}_{k-1} \sim \mathcal{N}(0, \mathbf{Q})$ is process noise.

2.1 Why Sensor Readings Alone Are Not Enough

- Sensors provide instantaneous but noisy data
- Models provide smooth but imperfect predictions
- Neither alone is reliable

A Kalman filter provides a principled way to optimally combine both.

3 Kalman Filter Overview

The **Kalman Filter** is an optimal recursive estimator for linear systems with Gaussian noise.

3.1 Linear System Model

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{3}$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{4}$$

4 Classical Kalman Filter Algorithm

4.1 Prediction Step

State prediction

$$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1} \tag{5}$$

Covariance prediction

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q} \tag{6}$$

4.2 Update Step

Kalman Gain

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T\left(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R}\right)^{-1} \tag{7}$$

State update

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-) \tag{8}$$

Covariance update

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^- \tag{9}$$

5 Limitations of the Classical Kalman Filter

Most real-world systems are nonlinear:

- Robot motion models
- IMU and inertial sensors
- Radar and LiDAR measurements

Linear Kalman filters fail under strong nonlinearities, leading to biased estimates or divergence.

6 Extended Kalman Filter (EKF)

The **Extended Kalman Filter** addresses nonlinearity by linearizing the system using first-order Taylor expansion.

6.1 Jacobian Matrices

$$\mathbf{A}_k = \frac{\partial f}{\partial \mathbf{x}}, \quad \mathbf{H}_k = \frac{\partial h}{\partial \mathbf{x}} \tag{10}$$

6.2 Limitations of EKF

- Jacobians are difficult to derive
- Linearization errors accumulate
- Poor performance for highly nonlinear systems
- Can diverge easily

7 Unscented Kalman Filter (UKF)

The **Unscented Kalman Filter** avoids linearization entirely.

7.1 Key Idea: Unscented Transform

Instead of approximating nonlinear functions, UKF approximates the probability distribution by propagating carefully chosen sample points (sigma points).

8 Sigma Points

For a state dimension n, UKF generates $2n + 1$ sigma points.

$$\chi_0 = \mathbf{x} \tag{11}$$

$$\chi_i = \mathbf{x} + \sqrt{(n+\lambda)\mathbf{P}}_i \tag{12}$$

$$\chi_{i+n} = \mathbf{x} - \sqrt{(n+\lambda)\mathbf{P}}_i \tag{13}$$

where:

$$\lambda = \alpha^2(n+\kappa) - n \tag{14}$$

9 UKF Prediction Step

1. Generate sigma points
2. Propagate through system model:
$$\chi_i^- = f(\chi_i) \tag{15}$$
3. Compute predicted mean:
$$\hat{\mathbf{x}}^- = \sum_i W_i^m \chi_i^- \tag{16}$$
4. Compute predicted covariance:
$$\mathbf{P}^- = \sum_i W_i^c (\chi_i^- - \hat{\mathbf{x}}^-)(\chi_i^- - \hat{\mathbf{x}}^-)^T + \mathbf{Q} \tag{17}$$

10 UKF Measurement Update

1. Transform sigma points into measurement space:
$$\mathbf{z}_i = h(\chi_i^-) \tag{18}$$
2. Predicted measurement mean:
$$\hat{\mathbf{z}} = \sum_i W_i^m \mathbf{z}_i \tag{19}$$
3. Measurement covariance:
$$\mathbf{S} = \sum_i W_i^c (\mathbf{z}_i - \hat{\mathbf{z}})(\mathbf{z}_i - \hat{\mathbf{z}})^T + \mathbf{R} \tag{20}$$
4. Cross covariance:
$$\mathbf{P}_{xz} = \sum_i W_i^c (\chi_i^- - \hat{\mathbf{x}}^-)(\mathbf{z}_i - \hat{\mathbf{z}})^T \tag{21}$$
5. Kalman Gain:
$$\mathbf{K} = \mathbf{P}_{xz}\mathbf{S}^{-1} \tag{22}$$
6. State and covariance update:
$$\hat{\mathbf{x}} = \hat{\mathbf{x}}^- + \mathbf{K}(\mathbf{z} - \hat{\mathbf{z}}) \tag{23}$$
$$\mathbf{P} = \mathbf{P}^- - \mathbf{K}\mathbf{S}\mathbf{K}^T \tag{24}$$

11 EKF vs UKF Comparison

Feature	EKF	UKF
Linearization	Jacobians	None
Accuracy	First-order	Higher-order
Stability	Can diverge	More robust
Implementation	Complex	Cleaner

12 Introduction to Localization

Localization is the process of estimating the state of a robot using noisy sensor measurements and a mathematical model of the robot's motion. The state typically includes the robot's position, orientation, and velocity. For mobile robots, accurate localization is essential for navigation, control, and autonomy.

Mathematically, localization aims to compute the posterior probability distribution:

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}, \mathbf{u}_{1:k}) \tag{25}$$

where:

- $\mathbf{x}_k$ is the robot state at time k,
- $\mathbf{z}_{1:k}$ are all sensor measurements up to time k,
- $\mathbf{u}_{1:k}$ are all control inputs or inertial inputs up to time k.

13 Importance of Localization

Localization is a fundamental requirement for robotic systems because:

- Navigation requires knowing the current position.
- Control algorithms depend on accurate state estimates.
- Mapping and perception rely on consistent pose estimates.
- Autonomous decision-making is impossible without localization.

For underwater robots, localization is particularly challenging because GPS signals do not propagate underwater, making sensor fusion techniques essential.

14 State-Space Model for Localization

Robot localization is commonly formulated as a nonlinear state-space model:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \tag{26}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k \tag{27}$$

where:

- $f(\cdot)$ is the process (motion) model,
- $h(\cdot)$ is the measurement model,
- $\mathbf{w}_k \sim \mathcal{N}(0, Q)$ is process noise,
- $\mathbf{v}_k \sim \mathcal{N}(0, R)$ is measurement noise.

15 Offline Localization using a Constant Velocity Model

The localization code `track1.py` implements an offline localization algorithm based on the Unscented Kalman Filter (UKF). The objective of this implementation is to estimate the three-dimensional position of a robot using logged sensor data. This implementation is intentionally simplified and is primarily designed for educational and experimental purposes.

15.1 State Definition

The system state is defined as:

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & z_k & v_{x,k} & v_{y,k} & v_{z,k} \end{bmatrix}^T \tag{28}$$

where:

- (x_k, y_k, z_k) represent the position of the robot in a global reference frame,
- $(v_{x,k}, v_{y,k}, v_{z,k})$ represent the linear velocity components in the same frame.

This state formulation assumes that the robot's orientation does not influence its motion, which is a strong simplification. As a result, this model is not suitable for vehicles whose motion is tightly coupled with orientation, such as underwater vehicles or aerial robots.

15.2 Process Model

The process model assumes that the robot moves with constant velocity between successive time steps. Given a time increment Δt, the state evolution is defined as:

$$x_{k+1} = x_k + v_{x,k}\Delta t \tag{29}$$
$$y_{k+1} = y_k + v_{y,k}\Delta t \tag{30}$$
$$z_{k+1} = z_k + v_{z,k}\Delta t \tag{31}$$
$$v_{x,k+1} = v_{x,k} \tag{32}$$
$$v_{y,k+1} = v_{y,k} \tag{33}$$
$$v_{z,k+1} = v_{z,k} \tag{34}$$

This model captures translational motion but neglects acceleration and rotational dynamics. In the implementation, this logic is encapsulated in the function `constant_velocity_model()`, which is passed to the UKF as the nonlinear state transition function.

15.3 Measurement Models

Two types of measurements are considered:

- **GPS measurements**: provide direct observations of the horizontal position (x, y).
- **Depth measurements**: provide observations of the vertical position z.

The corresponding measurement equations are:

$$\mathbf{z}_{\text{GPS}} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \mathbf{v}_{\text{GPS}} \tag{35}$$
$$z_{\text{depth}} = z_k + v_{\text{depth}} \tag{36}$$

where $\mathbf{v}_{\text{GPS}}$ and v_{depth} represent measurement noise. In the code, these measurements are fused sequentially using UKF update steps whenever the corresponding sensor data is available.

15.4 Interpretation

This localization approach demonstrates the fundamental idea of state estimation: a prediction step based on a motion model is continuously corrected using noisy sensor measurements. While the model is overly simplistic, it serves as a useful introduction to UKF-based localization and provides insight into the role of process and measurement models.

16 Generic Unscented Kalman Filter Implementation

The file `track2.py` contains a fully generic implementation of the Unscented Kalman Filter. This code does not perform localization by itself; instead, it provides the mathematical framework required to estimate the state of any nonlinear dynamical system.

16.1 Role of the UKF

The UKF is designed to estimate the state of a system described by the nonlinear equations:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \Delta t, \mathbf{u}_k) + \mathbf{w}_k \tag{37}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k \tag{38}$$

where:

- $f(\cdot)$ is the process model,
- $h(\cdot)$ is the measurement model,
- $\mathbf{w}_k$ and $\mathbf{v}_k$ represent process and measurement noise.

The UKF approximates the probability distribution of the state using a deterministic set of sigma points rather than linearization.

16.2 Key Components of the Implementation

The UKF implementation includes the following essential components:

- **Sigma point generation**: Sigma points are generated using Cholesky decomposition of the covariance matrix to ensure numerical stability.
- **Prediction step**: Each sigma point is propagated through the nonlinear process model.
- **Measurement update**: Sigma points are transformed into measurement space and used to compute the Kalman gain.
- **Covariance propagation**: The covariance matrix is updated in a symmetric and numerically stable manner.

This generic design allows the same UKF implementation to be reused across different localization problems simply by changing the process and measurement models.

17 Real-Time ROS2 Localization for an AUV

The final localization implementation is a real-time ROS2 node designed for an autonomous underwater vehicle (AUV). Unlike the offline implementation, this system must operate continuously and handle asynchronous sensor measurements.

17.1 State Definition

The state vector is defined as:

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & z_k & \phi_k & \theta_k & \psi_k & u_k & v_k & w_k \end{bmatrix}^T \tag{39}$$

where:

- (x_k, y_k, z_k) represent the vehicle position in a world-fixed frame,
- $(\phi_k, \theta_k, \psi_k)$ represent roll, pitch, and yaw,
- (u_k, v_k, w_k) represent body-frame velocities.

This formulation captures the full six-degree-of-freedom motion of the AUV.

17.2 Process Model

The process model integrates inertial measurements from the IMU. Angular velocities (p, q, r) are used to propagate orientation:

$$\phi_{k+1} = \phi_k + p\Delta t \tag{40}$$

$$\theta_{k+1} = \theta_k + q\Delta t \tag{41}$$

$$\psi_{k+1} = \psi_k + r\Delta t \tag{42}$$

Linear accelerations are used to propagate body-frame velocity while compensating for gravity:

$$\mathbf{v}_{k+1} = \mathbf{v}_k + (\mathbf{a}_{\text{body}} - \mathbf{g}_{\text{body}})\Delta t \tag{43}$$

The position update is obtained by rotating body-frame velocity into the world frame:

$$\mathbf{p}_{k+1} = \mathbf{p}_k + R(\phi, \theta, \psi)\mathbf{v}_{\text{body}}\Delta t \tag{44}$$

17.3 Measurement Models

Two primary sensors are used to correct the predicted state:

- **Doppler Velocity Log (DVL)**: provides direct measurements of body-frame velocities (u, v, w).
- **Pressure sensor**: provides measurements of depth, corresponding to the z component of position.

These measurements are incorporated using UKF update steps whenever sensor data becomes available.

17.4 Interpretation

This ROS2 localization node performs full six-degree-of-freedom localization in real time. It represents a realistic and deployable localization system suitable for underwater robotic applications, where accurate state estimation is critical for navigation and control.

18 Applications

- Robotics and autonomous vehicles
- Sensor fusion
- Tracking and navigation
- Control systems
- Research and Development projects

19 Conclusion

This implementation provides a clear and faithful realization of the Unscented Kalman Filter by directly translating its mathematical formulation into readable Python code. The filter maintains a probabilistic state using a mean and covariance, generates sigma points via the Unscented Transform, propagates them through an arbitrary nonlinear system model, and reconstructs the predicted state without any linearization or Jacobians. The prediction and update steps strictly follow UKF theory, with explicit computation of weighted means, covariances, cross-covariances, and Kalman gain. Important features of the code include support for nonlinear dynamics through user-defined models, partial state measurements, configurable UKF tuning parameters, and thread-safe execution using locks. Overall, the design prioritizes correctness, transparency, and educational value, making the code suitable both for practical use and for developing a deep understanding of nonlinear state estimation.