EKF and PF Localization

Favour Wobidi

May 2025

1 Introduction

The goal of this experiment is to localize a mobile robot in a known environment using noisy control inputs and partial landmark observations. We implement two probabilistic localization methods: the Extended Kalman Filter (EKF) and the Particle Filter (PF). The robot receives odometry inputs and bearing measurements from known landmarks. These measurements are corrupted with noise modeled by parameters (motion noise) and (sensor noise). The challenge is to accurately estimate the robot's trajectory and pose over time using these noisy inputs.

2 Exercise 1: Kalman Gain

Let two Gaussian distributions be defined as:

$$\begin{aligned} f(x) &= \mathcal{N}(x; \mu, \sigma^2), \\ g(x) &= \mathcal{N}(x; z, \sigma^2_{\text{obs}}) \end{aligned}$$

The Kalman Filter correction step is given by:

$$\begin{aligned} K &= \frac{\sigma^2}{\sigma^2 + \sigma^2_{\text{obs}}} \\ \mu' &= \mu + K(z - \mu) \\ \sigma'^2 &= (1 - K)\sigma^2 \end{aligned}$$

We aim to show that multiplying $f(x)$ and $g(x)$ results in a Gaussian distribution with the same μ' and σ'^2.

2.1 Product of Two Gaussians

Ignoring constant scale factors, the product of two Gaussians can be written as:

$$f(x)g(x) \propto \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} - \frac{(x-z)^2}{2\sigma^2_{\text{obs}}}\right)$$

Expanding both terms inside the exponent:

$$= \exp\left(-\frac{1}{2}\left[x^2\left(\frac{1}{\sigma^2}+\frac{1}{\sigma^2_{\text{obs}}}\right) - 2x\left(\frac{\mu}{\sigma^2}+\frac{z}{\sigma^2_{\text{obs}}}\right) + \text{const}\right]\right)$$

This is the exponent of a Gaussian, so the resulting distribution is:

$$\begin{aligned}\sigma'^2 &= \left(\frac{1}{\sigma^2}+\frac{1}{\sigma^2_{\text{obs}}}\right)^{-1} \\ \mu' &= \sigma'^2\left(\frac{\mu}{\sigma^2}+\frac{z}{\sigma^2_{\text{obs}}}\right)\end{aligned}$$

2.2 Comparison with Kalman Filter Equations

We show that these are the same as in the Kalman filter formulation:

Start from:

$$K = \frac{\sigma^2}{\sigma^2+\sigma^2_{\text{obs}}}$$

Then the updated mean:

$$\begin{aligned}\mu' &= \mu + K(z-\mu) \\ &= \mu + \frac{\sigma^2}{\sigma^2+\sigma^2_{\text{obs}}}(z-\mu) \\ &= \frac{\sigma^2_{\text{obs}}\mu + \sigma^2 z}{\sigma^2+\sigma^2_{\text{obs}}}\end{aligned}$$

Which matches:

$$\mu' = \left(\frac{1}{\sigma^2}+\frac{1}{\sigma^2_{\text{obs}}}\right)^{-1}\left(\frac{\mu}{\sigma^2}+\frac{z}{\sigma^2_{\text{obs}}}\right)$$

And for the variance:

$$\sigma'^2 = (1-K)\sigma^2 = \sigma^2 - \frac{\sigma^4}{\sigma^2+\sigma^2_{\text{obs}}} = \frac{\sigma^2\sigma^2_{\text{obs}}}{\sigma^2+\sigma^2_{\text{obs}}}$$

Which matches:

$$\sigma'^2 = \left(\frac{1}{\sigma^2}+\frac{1}{\sigma^2_{\text{obs}}}\right)^{-1}$$

2.3 Interpretation

We have demonstrated that the correction step of the Kalman Filter in 1D is equivalent to multiplying the prior and observation Gaussian densities. The resulting posterior distribution has the same mean and variance as obtained using the Kalman equations. This confirms the probabilistic basis of the Kalman Filter as a Bayesian estimator.

3 Exercise 2: Jacobian Motion Model

3.1 1. Robot State and Control

The robot's state is defined as its 2D position and orientation:

$$s = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

And, the control input is given as:

$$u = \begin{bmatrix} \delta_{\text{rot1}} \\ \delta_{\text{trans}} \\ \delta_{\text{rot2}} \end{bmatrix}$$

This means the robot:

1. Rotates by δ_{rot1},
2. Translates forward by δ_{trans},
3. Rotates again by δ_{rot2}.

3.2 2. Motion Model

The motion model is given by the following equations:

$$\begin{aligned} x' &= x + \delta_{\text{trans}} \cdot \cos(\theta + \delta_{\text{rot1}}) \\ y' &= y + \delta_{\text{trans}} \cdot \sin(\theta + \delta_{\text{rot1}}) \\ \theta' &= \theta + \delta_{\text{rot1}} + \delta_{\text{rot2}} \end{aligned}$$

Let $g(s, u)$ denote the motion model such that $s' = g(s, u)$.

3. Jacobian with Respect to the State ($G = \partial g / \partial s$)

We compute the Jacobian matrix G as the partial derivative of g with respect to the state s:

$$G = \begin{bmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} & \frac{\partial x'}{\partial \theta} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} & \frac{\partial y'}{\partial \theta} \\ \frac{\partial \theta'}{\partial x} & \frac{\partial \theta'}{\partial y} & \frac{\partial \theta'}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\delta_{\text{trans}} \cdot \sin(\theta + \delta_{\text{rot1}}) \\ 0 & 1 & \delta_{\text{trans}} \cdot \cos(\theta + \delta_{\text{rot1}}) \\ 0 & 0 & 1 \end{bmatrix}$$

3.3 4. Jacobian with Respect to the Control ($V = \partial g / \partial u$)

We compute the Jacobian matrix V as the partial derivative of g with respect to the control u:

$$V = \begin{bmatrix} \frac{\partial x'}{\partial \delta_{\text{rot1}}} & \frac{\partial x'}{\partial \delta_{\text{trans}}} & \frac{\partial x'}{\partial \delta_{\text{rot2}}} \\ \frac{\partial y'}{\partial \delta_{\text{rot1}}} & \frac{\partial y'}{\partial \delta_{\text{trans}}} & \frac{\partial y'}{\partial \delta_{\text{rot2}}} \\ \frac{\partial \theta'}{\partial \delta_{\text{rot1}}} & \frac{\partial \theta'}{\partial \delta_{\text{trans}}} & \frac{\partial \theta'}{\partial \delta_{\text{rot2}}} \end{bmatrix} = \begin{bmatrix} -\delta_{\text{trans}} \cdot \sin(\theta + \delta_{\text{rot1}}) & \cos(\theta + \delta_{\text{rot1}}) & 0 \\ \delta_{\text{trans}} \cdot \cos(\theta + \delta_{\text{rot1}}) & \sin(\theta + \delta_{\text{rot1}}) & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

3.4 5. Interpretation

- G captures how changes in the robot's current state affect its predicted future state.
- V captures how changes in the control commands affect the predicted state.

This model is critical for applying both the Extended Kalman Filter (EKF) and the Particle Filter (PF) in probabilistic robot localization.

4 Exercise 3: Extended Kalman Filter (EKF)

The EKF is implemented in `ekf.py` and relies on three key Jacobians: G (state w.r.t. motion), V (state w.r.t. control), and H (observation w.r.t. state). The update function combines a prediction step using the motion model and a correction step using landmark observations.

4.1 Visual Results

With default parameters, the EKF tracks the robot's trajectory with high accuracy, particularly during linear movements. Minor deviations appear during sharp turns due to the filter's linearization.

4.2 Effect of Varying α and β

We scaled the noise parameters with factors $r = [1/64, 1/16, 1/4, 4, 16, 64]$.

- **Low r values:** The filter is overconfident and diverges.
- **High r values:** Conservative but prone to instability.
- **Best performance:** Observed around $r = 1$ to 4.

4.3 ANEES Analysis

- **ANEES ≈ 1:** Filter is well-calibrated.
- **ANEES $\gg 1$:** Filter underestimates uncertainty.
- **ANEES $\ll 1$:** Filter overestimates uncertainty.

5 Exercise 4: Particle Filter (PF)

Each particle represents a hypothesis about the robot's position. The update step applies motion, computes weights based on bearing observations, and resamples using low-variance sampling.

PF estimates are noisier but track the true trajectory well, especially in areas with good landmark visibility. Recovery from localization loss is possible due to particle diversity.

5.1 Effect of Varying α and β

- **Robust to noise:** Better than EKF under high uncertainty.
- **Too few particles:** Leads to degeneracy.
- **Moderate noise:** PF performs well with stable updates.

5.2 ANEES and Particle Count Analysis

- **20 particles:** High error and instability.
- **50 particles:** Acceptable in many cases.
- **500 particles:** Low error and best ANEES values.

6 Conclusion

The EKF provides smoother and faster localization under mild noise but is vulnerable to divergence under model mismatches. PF, while computationally heavier, is more robust and adaptive. ANEES analysis confirms that accurate uncertainty modeling is key to reliable localization.

7 References

- **Probabilistic Robotics, Thrun, Burgard, and Fox**
- **CSE571 Lecture Notes, Winter 2019**
- **Assignment Handout and Starter Code**