Extended Kalman Filter SLAM

1 Introduction - Recursive Bayes Filters and Probability Model

Simultaneous Localization and Mapping (SLAM) is the problem of a robot constructing a map of an unknown environment while simultaneously determining its location within that map. At the heart of probabilistic SLAM lies the recursive Bayes filter, which provides a framework for estimating the robot's belief over its state x_t given a sequence of control inputs $u_{1:t}$ and observations $z_{1:t}$.

The belief $bel(x_t)$ at time t is defined as the posterior probability distribution over the state x_t, updated recursively using the Bayes rule:

$$bel(x_t) = \eta\, p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) \int p(x_t \mid x_{t-1}, u_t)\, bel(x_{t-1})\, dx_{t-1}$$

where: - η is a normalization constant, - $p(z_t \mid x_t, z_{1:t-1}, u_{1:t})$ is the measurement model (likelihood of observation z_t given the state), - $p(x_t \mid x_{t-1}, u_t)$ is the motion model (probability of transitioning from state x_{t-1} to x_t given control u_t), - $bel(x_{t-1})$ is the prior belief.

For SLAM, the state x_t includes both the robot's pose and the positions of landmarks in the environment. Assuming Markovian properties (current state depends only on the previous state and current control), the model simplifies to:

$$bel(x_t) = \eta\, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t)\, bel(x_{t-1})\, dx_{t-1}$$

This recursive formulation is the foundation of the Kalman Filter and its extension, the EKF, which handles nonlinearities through linearization.

2 Kalman Filter

The Kalman Filter estimates the state $\boldsymbol{x}_t$ of a linear dynamical system subject to Gaussian noise. The recursive Bayes filter update consists of two steps: prediction and correction (measurement update).

2.1 Linear Gaussian System Model

Motion model:

$$\boldsymbol{x}_t = A_{t-1}\boldsymbol{x}_{t-1} + B_{t-1}\boldsymbol{u}_{t-1} + \boldsymbol{w}_{t-1}, \quad \boldsymbol{w}_{t-1} \sim \mathcal{N}(0, R_{t-1}) \tag{1}$$

Measurement model:

$$\boldsymbol{z}_t = C_t\boldsymbol{x}_t + \boldsymbol{v}_t, \quad \boldsymbol{v}_t \sim \mathcal{N}(0, Q_t) \tag{2}$$

2.2 Prediction Step

The prior (predicted) belief is computed as:

$$\hat{\boldsymbol{x}}_t = A_{t-1}\hat{\boldsymbol{x}}_{t-1} + B_{t-1}\boldsymbol{u}_{t-1} \tag{3}$$
$$P_t^- = A_{t-1}P_{t-1}A_{t-1}^\top + R_{t-1} \tag{4}$$

2.3 Measurement Update (Correction)

When a measurement $\boldsymbol{z}_t$ is received, we compute the Kalman gain:

$$K_t = P_t^- C_t^\top (C_t P_t^- C_t^\top + Q_t)^{-1} \tag{5}$$

Then, the corrected belief is:

$$\hat{\boldsymbol{x}}_t \leftarrow \hat{\boldsymbol{x}}_t^- + K_t(\boldsymbol{z}_t - C_t\hat{\boldsymbol{x}}_t^-) \tag{6}$$
$$P_t \leftarrow (I - K_t C_t)P_t^- \tag{7}$$

3 Extended Kalman Filter (EKF)

When the system is non-linear, we use the Extended Kalman Filter (EKF), which linearizes the models around the current estimate using a first-order Taylor expansion.

3.1 Linearization via Taylor Expansion

For nonlinear functions $g(x)$ (motion model) and $h(x)$ (observation model), EKF uses a first-order Taylor expansion around the current estimate $\hat{x}$:

$$g(x) \approx g(\hat{x}) + G(x - \hat{x}) \tag{8}$$
$$h(x) \approx h(\hat{x}) + H(x - \hat{x}) \tag{9}$$

where $G = \frac{\partial g}{\partial x}\big|_{\hat{x}}$ is the Jacobian of the motion model, and $H = \frac{\partial h}{\partial x}\big|_{\hat{x}}$ is the Jacobian of the observation model.

3.2 Non-linear Models

Motion model:

$$\boldsymbol{x}_t = g(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}) + \boldsymbol{w}_{t-1}, \quad \boldsymbol{w}_{t-1} \sim \mathcal{N}(0, R_t) \tag{10}$$

Measurement model:

$$\boldsymbol{z}_t = h(\boldsymbol{x}_t) + \boldsymbol{v}_t, \quad \boldsymbol{v}_t \sim \mathcal{N}(0, Q_t) \tag{11}$$

3.3 State Vector and Covariance

The state vector μ_t includes the robot pose $(\mu_{t,x}, \mu_{t,y}, \mu_{t,\theta})$ and all landmark positions $(\mu_{j,x}, \mu_{j,y})$ for each landmark j:

$$\mu_t = \begin{bmatrix} \mu_{t,x} \\ \mu_{t,y} \\ \mu_{t,\theta} \\ \mu_{j,x} \\ \mu_{j,y} \\ \vdots \end{bmatrix}, \quad \Sigma_t = \mathrm{Cov}[\mu_t]$$

3.4 Motion Model and Jacobian G_t

The motion model $g(\mu_{t-1}, u_t)$ for a differential drive robot with control input $u_t = (v, \omega)$ (translational and rotational velocities) is:

$$g(\mu_{t-1}, u_t) = \begin{bmatrix} \mu_{t-1,x} - \frac{v}{\omega}\sin(\mu_{t-1,\theta}) + \frac{v}{\omega}\sin(\mu_{t-1,\theta} + \omega\Delta t) \\ \mu_{t-1,y} + \frac{v}{\omega}\cos(\mu_{t-1,\theta}) - \frac{v}{\omega}\cos(\mu_{t-1,\theta} + \omega\Delta t) \\ \mu_{t-1,\theta} + \omega\Delta t \end{bmatrix}$$

The Jacobian $G_t = \frac{\partial g}{\partial \mu}$ is derived as:

$$G_t = I + F_x^\top G_t^x F_x$$

where

$$G_t^x = \begin{bmatrix} 1 & 0 & -\frac{v}{\omega}\cos(\mu_{t-1,\theta}) + \frac{v}{\omega}\cos(\mu_{t-1,\theta} + \omega\Delta t) \\ 0 & 1 & -\frac{v}{\omega}\sin(\mu_{t-1,\theta}) + \frac{v}{\omega}\sin(\mu_{t-1,\theta} + \omega\Delta t) \\ 0 & 0 & 1 \end{bmatrix}$$

and F_x is the Jacobian selector matrix.

3.5 Process Noise Covariance R_t

The process noise covariance is:

$$R_t = F_x^\top Q_t F_x$$

where

$$Q_t = \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & \sigma_\theta \end{bmatrix}$$

represents the noise in translational and rotational velocities.

3.6 Measurement Model and Jacobian H_t^i

The expected measurement for landmark j is:

$$h(\mu_t) = \begin{bmatrix} \sqrt{(\mu_{j,x} - \mu_{t,x})^2 + (\mu_{j,y} - \mu_{t,y})^2} \\ \arctan 2(\mu_{j,y} - \mu_{t,y}, \mu_{j,x} - \mu_{t,x}) - \mu_{t,\theta} \end{bmatrix}$$

Define $\delta = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \begin{bmatrix} \mu_{j,x} - \mu_{t,x} \\ \mu_{j,y} - \mu_{t,y} \end{bmatrix}$ and $q = \delta_x^2 + \delta_y^2$. The Jacobian H_t^i is:

$$H_t^i = \frac{1}{\sqrt{q}} \begin{bmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & \sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{bmatrix} F_{x,j}$$

where $F_{x,j}$ is a selector matrix mapping the state into measurement space.

3.7 Kalman Gain K_t^i

The Kalman gain for observation i is:

$$K_t^i = \Sigma_t (H_t^i)^\top \left[(H_t^i \Sigma_t (H_t^i)^\top + Q_t) \right]^{-1}$$

4 EKF SLAM Algorithm

Algorithm EKF_SLAM(known_correspondences($z_{1:t}, z_{1:t}, u_{1:t}, z_{1:t}$)):

1. $F_x = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \end{bmatrix}$
2. $\mu_t = \mu_{t-1} + F_x^\top \begin{bmatrix} -\frac{v}{\omega}\sin(\mu_{t-1,\theta}) + \frac{v}{\omega}\sin(\mu_{t-1,\theta} + \omega\Delta t) \\ \frac{v}{\omega}\cos(\mu_{t-1,\theta}) - \frac{v}{\omega}\cos(\mu_{t-1,\theta} + \omega\Delta t) \\ \omega\Delta t \end{bmatrix}$
3. $\mu_t = \mu_{t-1} + F_x^\top \left(\begin{bmatrix} -\frac{v}{\omega}\sin(\mu_{t-1,\theta}) + \frac{v}{\omega}\sin(\mu_{t-1,\theta} + \omega\Delta t) \\ \frac{v}{\omega}\cos(\mu_{t-1,\theta}) - \frac{v}{\omega}\cos(\mu_{t-1,\theta} + \omega\Delta t) \\ \omega\Delta t \end{bmatrix} \right)$
4. $G_t = I + F_x^\top \begin{bmatrix} 0 & 0 & \frac{v}{\omega}\cos(\mu_{t-1,\theta}) - \frac{v}{\omega}\cos(\mu_{t-1,\theta} + \omega\Delta t) \\ 0 & 0 & \frac{v}{\omega}\sin(\mu_{t-1,\theta}) - \frac{v}{\omega}\sin(\mu_{t-1,\theta} + \omega\Delta t) \\ 0 & 0 & 0 \end{bmatrix} F_x$
5. $\Sigma_t = G_t \Sigma_{t-1} G_t^\top + F_x^\top \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & \sigma_\theta \end{bmatrix} F_x$
6. **for all observed features** $z = (r^i, \phi^i, j)^\top$ **do**
7. **if landmark** j **never seen before**

8. $\begin{bmatrix}\mu_{j,x}\\ \mu_{j,y}\end{bmatrix} = \begin{bmatrix}\mu_{t,x}\\ \mu_{t,y}\end{bmatrix} + \begin{bmatrix}r^i \cos(\phi^i + \mu_{t,\theta})\\ r^i \sin(\phi^i + \mu_{t,\theta})\end{bmatrix}$
9. **endif**
10. $\delta_x = \mu_{j,x} - \mu_{t,x}, \quad \delta_y = \mu_{j,y} - \mu_{t,y}$
11. $q = \delta_x^2 + \delta_y^2$
12. $\hat{z}^i = \begin{bmatrix}\sqrt{q}\\ \arctan 2(\delta_y, \delta_x) - \mu_{t,\theta}\end{bmatrix}$
13. $H_t^i = \frac{1}{\sqrt{q}} \begin{bmatrix}-\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & \sqrt{q}\delta_x & \sqrt{q}\delta_y\\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x\end{bmatrix} F_{x,j}$
14. $K_t^i = \Sigma_t (H_t^i)^\top \left[H_t^i \Sigma_t (H_t^i)^\top + \begin{bmatrix}\sigma_r & 0\\ 0 & \sigma_\phi\end{bmatrix}\right]^{-1}$
15. $\mu_t = \mu_t + K_t^i (z^i - \hat{z}^i)$
16. $\Sigma_t = (I - K_t^i H_t^i)\Sigma_t$
17. **endfor**
18. **return** μ_t, Σ_t

5 Conclusion

The Extended Kalman Filter (EKF) offers significant advantages for SLAM applications. Its ability to handle nonlinear motion and observation models through linearization makes it computationally efficient, enabling real-time estimation of robot pose and landmark positions. EKF provides a structured approach to fuse noisy sensor data with motion predictions, reducing uncertainty over time and maintaining a consistent map. Additionally, its recursive nature allows for incremental updates, making it suitable for resource-constrained robotic systems.

EKF is widely used in various SLAM scenarios, including indoor mobile robot navigation, where it integrates data from odometry and range sensors (e.g., sonar or laser) to build 2D maps. It is also applied in autonomous vehicle localization, combining GPS and inertial measurement units, and in underwater robotics for mapping with sonar. These applications highlight EKF's versatility in environments with moderate nonlinearity and Gaussian noise, though its performance depends on accurate linearization and initial conditions.