CSE 483: Mobile Robotics

Lecture by: Prof. K. Madhava Krishna **Lecture #**
Scribe: Tanmay Chaudhari, Lincy Pattanaik **Date**:

EKF Localization - Worked out example

1 EKF Algorithm

The EKF can be viewed as two-step process namely, the state prediction and the update step. It is based on feature-based maps which consists of point landmarks based on measurement model. It takes a Gaussian estimate of the current pose as input at time $\mathbf{t}$, with mean $\mu_{\mathbf{t}}$ and covariance $\mathbf{\Sigma_t}$. Further it requires control $\mathbf{u_t}$, map $\mathbf{m}$ and a set of features (measurements) $\mathbf{z_{t+1}} = \{\mathbf{z^1_{t+1}}, \mathbf{z^2_{t+1}}, \dots\}$. Its output is new, revised estimate $\mu_{\mathbf{t+1}}$ and $\mathbf{\Sigma_{t+1}}$. In the following algorithm we use $[\mathbf{x}, \mathbf{y}, \theta]^{\mathbf{T}}$ and $[\mathbf{T}, \phi]^{\mathbf{T}}$ model.

Algorithm 1: EKF Algorithm

Algorithm EKF_localization
Input : $(\mu_t, \Sigma_t, u_{t+1}, z_{t+1}, m)$
Output: $(\mu_{t+1}, \Sigma_{t+1})$

$$\hat{\mu_{t+1}} = \begin{bmatrix} \hat{\mu}_{x,t+1} \\ \hat{\mu}_{y,t+1} \\ \hat{\mu}_{\theta,t+1} \end{bmatrix} = \begin{bmatrix} \mu_{x,t} \\ \mu_{y,t} \\ \mu_{\theta,t} \end{bmatrix} + \begin{bmatrix} Tcos(\mu_{\theta,t}+\phi) \\ Tsin(\mu_{\theta,t}+\phi) \\ \phi \end{bmatrix}$$

$$F = \begin{bmatrix} 1 & 0 & -Tsin(\mu_{\theta,t}+\phi) \\ 0 & 1 & Tcos(\mu_{\theta,t}+\phi) \\ 0 & 0 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} cos(\mu_{\theta,t}+\phi) & -Tsin(\mu_{\theta,t}+\phi) \\ sin(\mu_{\theta,t}+\phi) & Tcos(\mu_{\theta,t}+\phi) \\ 0 & 1 \end{bmatrix}$$

$\hat{\Sigma}_{t+1} = F\Sigma_t F^T + GRG^T$, where R is control covariance

$$Q_{t+1} = \begin{bmatrix} \sigma_r & 0 \\ 0 & \sigma_\psi \end{bmatrix}$$

for all observed features $z^i_{t+1} = [r^i_{t+1}, \psi^i_{t+1}]^T$ do

$$\sigma = \begin{bmatrix} \sigma_x \\ \sigma_y \end{bmatrix} = \begin{bmatrix} m_x - \hat{\mu}_{x,t+1} \\ m_y - \hat{\mu}_{y,t+1} \end{bmatrix}$$

$$q = \sigma^T \sigma$$

$$\hat{z}^i_{t+1} = \begin{bmatrix} \sqrt{q} \\ atan2(\delta_y, \delta_x) - \hat{\mu}_{t+1,\theta} \end{bmatrix}$$

$$H^i_{t+1} = \frac{1}{q} \begin{bmatrix} \sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 \\ \delta_y & \delta_x & -1 \end{bmatrix}$$

$$K^i_{t+1} = \hat{\Sigma}_{t+1} H^{i,T}_{t+1} (H^i_{t+1} \hat{\Sigma}_{t+1} H^{i,T}_{t+1} - Q_{t+1})^{-1}$$

endfor

$\mu_{t+1} = \hat{\mu}_{t+1} + \Sigma_i K^i_{t+1}(z^i_{t+1} - \hat{z}^i_{t+1})$

$\Sigma_{t+1} = (I - \Sigma_i K^i_{t+1} H^i_{t+1})\hat{\Sigma}_{t+1}$

return μ_{t+1}, Σ_{t+1}

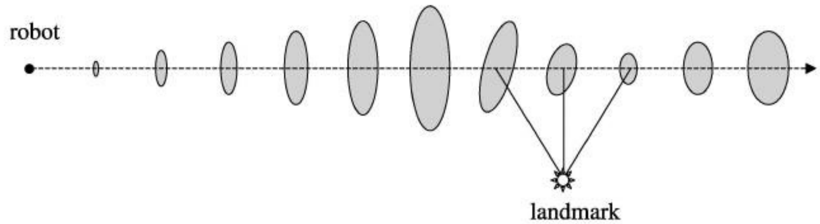

Figure 1: Example of localization using the extended Kalman filter. The robot moves on a straight line. As it progresses, its uncertainty increases gradually, as illustrated by the error ellipse. When it observes a landmark with known positive, the uncertainty is reduced.

2 Example of EKF

In the following example motion model $\mu_{\mathbf{t}} = [\mathbf{x_t}, \mathbf{y_t}, \theta_{\mathbf{t}}]^{\mathbf{T}}$ is used in each discrete time step where the robot does the following:.

1. Run the motion model to predict next state $\hat{\mu}_{\mathbf{t+1}}$.
2. Predict state covariance $\hat{\mathbf{\Sigma}}_{\mathbf{t+1}}$ using Jacobians of motion model.
3. Run the update step to get $\mu_{\mathbf{t+1}}$ and $\mathbf{\Sigma}_{\mathbf{t+1}}$.

Since robots do not execute their commands perfectly we model this by incorporating noise in robot's motion.

2.1 Prediction Step

1. Motion model:

   Let the input be current state $\mu_{\mathbf{0}} = [0, 0, 0]^T$, current state covariance $\mathbf{\Sigma_0} = diag(0, 0, 0)$ and next control $\mathbf{u_1} = [1, 0]^T$. To run the motion model we take the next ideal control and corrupt it with noise. Let control covariance $\mathbf{R} = diag(0.01, 0.01)$. Hence $\mathbf{u_{1,actual}} = \mathbf{u_1} + \mathbf{c}$, where control noise $\mathbf{c}$ is sampled from control noise distribution (with $\mathbf{R}$ covariance). One such example is $\mathbf{c} = [0.017, -0.013]^T$. Hence $\mathbf{u_{1,actual}} = [1.017, -0.103]^T$.

   Now using motion model (step 2 in EKF Algorithm) and ideal control $\mathbf{u_1}$, robot's next state $\hat{\mu_{\mathbf{1}}}$ is predicted. This is *where the robot thinks it is.* $\hat{\mu_{\mathbf{1}}}$ (predicted pose) = $[1, 0, 0]^T$.

   To know robot's true pose (*where it actually is*) motion model and actual control needs to be used. So $\mu_{\mathbf{1,actual}}$ (actual pose) = $[1.012, -0.1044, -0.1028]^T$.

2. To linearise the motion model Jacobians $\mathbf{F}$ and $\mathbf{G}$ about the predicted pose are calculated (step 3-4 of EKF algorithm)

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \textit{ and } \mathbf{G} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

Robot's predicted covariance (step 5 of EKF algorithm)

$$\hat{\mathbf{\Sigma}}_{\mathbf{t+1}} = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0.01 \\ 0 & 0.01 & 0.01 \end{bmatrix}.$$

2.2 Update Step

1. Measurement Model:

- Actual measurement ($\mathbf{z_{t+1}}$)

  To update the robot's estimate of the state we make use of measurement (a.k.a observations or sensor readings) i.e. $\mathbf{z_{t+1}}$. We assume that the robot is equipped with a range bearing sensor that can measure all landmarks within sensing range. The measurement uncertainty is characterized by a Gaussian with zero mean and some covariance matrix $\mathbf{Q}$.

  We fire the sensor and get the true sensor measurements, i.e. we get the range and bearing measurements to all landmarks within a stipulated sensor radius. Let there be $\mathbf{N}$ such landmarks, then $\mathbf{z_{t+1}} = [\mathbf{r_1}, \boldsymbol{\psi_1}, \mathbf{r_2}, \boldsymbol{\psi_2}, \ldots, \mathbf{r_N}, \boldsymbol{\psi_N}]^{\mathbf{T}}$ (step 8-11 of EKF).

  These are then corrupted by observation covariance $\mathbf{Q}$. These measurements can be intuitively thought of *what the robot actually sees*. These measurements are obtained using true pose of the robot (*where it actually is*).

  Suppose $\mathbf{N}$ is 4 and $\mathbf{m} = \begin{bmatrix} 5 & 5 \\ -5 & 5 \\ -5 & -5 \\ 5 & -5 \end{bmatrix}$ where each row is tuple $(\mathbf{m_x}, \mathbf{m_y})$. Then

$$\mathbf{z_{t+1}} = \begin{bmatrix} 6.48 \\ 1.01 \\ 7.89 \\ -0.60 \\ 7.75 \\ 0.79 \\ 6.31 \\ -0.78 \end{bmatrix}.$$

Incorporating noise (sampling from Gaussian distribution with zero mean and covariance $\mathbf{Q}$) we get

$$\mathbf{z_{t+1}} = \begin{bmatrix} 6.43 \\ 0.97 \\ 7.91 \\ -0.65 \\ 7.73 \\ 0.77 \\ 6.33 \\ -0.77 \end{bmatrix}.$$

- Expected measurement ($\mathbf{\hat{z}_{t+1}}$)

These measurements are *what the robot expects to see* from *where it thinks it is.* This serves as second estimate of state, which can be used to refine the estimate obtained from motion model.

Using step 8-11 of EKF algorithm we get

$$\mathbf{\hat{z}_{t+1}} = \begin{bmatrix} 7.81 \\ -0.69 \\ 7.81 \\ 0.69 \\ 6.40 \\ -0.89 \\ 6.40 \\ -0.89 \end{bmatrix}.$$

- Observation Jacobian ($\mathbf{H}$) and Kalman Gain ($\mathbf{K}$)

There are two primary ways in which update can be, viz. Incremental and Batch modes. In batch update, a joint update is computed for all landmarks (here 4) in a particular time step whereas in incremental mode, update is performed on a per-landmark basis. Hence the dimensions of $\mathbf{H}$, $\mathbf{S}$ and $\mathbf{K}$ differ in both modes.

For each observable landmark $\mathbf{m^i}$, following $\mathbf{H^i}$ are computed (from step 11 of EKF algorithm).

$$\mathbf{H^1} = \begin{bmatrix} 0.94 & -0.78 & 0 \\ 0.12 & 0.15 & -1 \end{bmatrix}$$

$$\mathbf{H^2} = \begin{bmatrix} 0.94 & 0.78 & 0 \\ -0.12 & 0.15 & -1 \end{bmatrix}$$

$$\mathbf{H^3} = \begin{bmatrix} -0.62 & 0.78 & 0 \\ -0.12 & -0.098 & -1 \end{bmatrix}$$

$$\mathbf{H^4} = \begin{bmatrix} -0.62 & 0.78 & 0 \\ -0.12 & -0.098 & -1 \end{bmatrix}.$$

In incremental mode, next state $\mu_{\mathbf{t+1}}$ and associated covariance $\mathbf{\Sigma_{t+1}}$ is updated 4 times with each of the 4 landmarks.

Using 1^{st} landmark $\mathbf{m^1}$, $\mathbf{K^1} = \begin{bmatrix} -0.17 & 0.04 \\ -0.08 & -0.11 \\ -0.08 & -0.11 \end{bmatrix}$ for which the state updates to $\mu^{\mathbf{1}}_{\mathbf{t+1}} = \begin{bmatrix} 2.6813 \\ 1.2049 \\ 1.2049 \end{bmatrix}$ and $\mathbf{\Sigma^1_{t+1}} = \begin{bmatrix} 0.00029 & 1.04852 * 10^{-6} & 1.04852 * 10^{-6} \\ 1.04852 * 10^{-6} & 9.90978 * 10^{-5} & 9.90978 * 10^{-5} \\ 1.04852 * 10^{-6} & 9.90978 * 10^{-5} & 9.90978 * 10^{-5} \end{bmatrix}$.

Similarly after 4 updates, we get
$\mu^{\mathbf{4}}_{\mathbf{t+1}} = \begin{bmatrix} 1.75 \\ 0.75 \\ 0.75 \end{bmatrix}$ and $\mathbf{\Sigma^4_{t+1}} = \begin{bmatrix} 0.00032 & 1.3953 * 10^{-5} & 1.3953 * 10^{-5} \\ 1.3953 * 10^{-5} & 0.000122 & 0.000122 \\ 1.3953 * 10^{-5} & 0.000122 & 0.000122 \end{bmatrix}$.

In batch mode, update is done only once using all 4 landmarks at a time. To obtain Kalman gain $\mathbf{K}$, $\mathbf{H_{8x3}}$ is made by concatenating observation Jacobians $\mathbf{H^1}$ to $\mathbf{H^4}$. Corresponding to batch mode, following $\mathbf{K_{3x8}}$ is computed.

$$\mathbf{K} = \begin{bmatrix} 0.29 & 0.03 & 0.31 & -0.05 & -0.19 & -0.05 & -0.19 & -0.05 \\ -0.08 & -0.10 & 0.10 & -0.10 & 0.09 & -0.14 & 0.09 & -0.14 \\ -0.08 & -0.10 & 0.10 & -0.10 & 0.09 & -0.14 & 0.09 & -0.14 \end{bmatrix}.$$

Finally using steps 14 and 15, we get

$$\mu_{\mathbf{t+1}} = \begin{bmatrix} 1.75 \\ 0.75 \\ 0.75 \end{bmatrix} \text{ and } \mathbf{\Sigma_{t+1}} = \begin{bmatrix} 0.00032 & 1.3953 * 10^{-5} & 1.3953 * 10^{-5} \\ 1.3953 * 10^{-5} & 0.000122 & 0.000122 \\ 1.3953 * 10^{-5} & 0.000122 & 0.000122 \end{bmatrix}.$$