Lecture 07
EKF, UKF, Particle Filters, and SLAM

Instructor: Jingjin Yu
Outline

Kalman filter recap
Extended Kalman filter (EKF)
Unsented Kalman filter (UKF) and particle filters
Simultaneous localization and mapping (SLAM)
Sensing review
Kalman Filter Review

Kalman filter is a type of **Bayesian filters** over a **Hidden Markov model**

- The $x_i$s are **hidden (actual)** system states that are **not directly known**
- We can only observe $x_i$ using sensors to get observations $z_i$
- The (discrete) process is often modeled as a two-step iterative one
  - Noisy state change: $x_k = f(x_{k-1}, u_{k-1}) + \omega_{k-1}$
  - Noisy measurement after state change: $z_k = h(x_k) + \nu_k$
- The sequence of “data” is $u_0, z_1, u_1, z_2, u_2, z_3, ...$
- The goal is to derive an $\hat{x}_k$ as an accurate **estimate** of $x_k$
Kalman Filter Review – Assumptions

Stochastic, discrete-time **linear** system

\[ x_k = Ax_{k-1} + Bu_{k-1} + \omega_{k-1}, \quad \omega_{k-1} \sim N(0, Q) \]  

\[ \Rightarrow x_k, \omega_{k-1} \in \mathbb{R}^n, u_{k-1} \in \mathbb{R}^m, A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m} \]

**Linear** observer (sensor)

\[ z_k = Hx_k + \nu_k, \quad \nu_k \sim N(0, R) \]

\[ \Rightarrow z_k, \nu_k \in \mathbb{R}^\ell, H \in \mathbb{R}^{\ell \times n} \]

Both \( \omega_{k-1} \) and \( \nu_k \) are **zero mean Gaussians** and are **uncorrelated**

\[ \Rightarrow \text{i.e., Cov}(\omega, \nu) = 0 \]
Kalman Filter Review – Formulas

We have the iterative update algorithm

To run the algorithm

⇒ The values of $A$, $B$, and $H$ are known, $u_{k-1}$ and $z_k$ are also known
⇒ The values of $Q$ and $R$ are estimated (system identification or sys ID)
⇒ Initial values $\hat{x}_0$ and $P_0$ are guessed
⇒ Usually $P_k$ and $K_k$ will quickly converge with the right $Q$ and $R$
Extended Kalman Filter (EKF) – Assumptions

Kalman filter requires linearity, i.e.,

\[
x_k = Ax_{k-1} + Bu_{k-1} + \omega_{k-1}, \quad \omega_{k-1} \sim N(0, Q) \quad (1)
\]
\[
z_k = Hx_k + v_k, \quad v_k \sim N(0, R) \quad (2)
\]

These **nice assumptions** often do not hold in practice!

\[\Rightarrow\text{More realistic assumptions are}\]

\[
x_k = f(x_{k-1}, u_{k-1}, \omega_{k-1}), \quad \omega_{k-1} \sim N(0, Q) \quad (1^*)
\]
\[
z_k = h(x_k, v_k), \quad v_k \sim N(0, R) \quad (2^*)
\]

\[\Rightarrow\text{The functions } f \text{ and } h \text{ are non-linear}\]

\[\Rightarrow\text{Here, we know } f, h, u_{k-1}, z_k, \text{ and estimate } Q \text{ and } R\]

\[\Rightarrow\text{But, locally, } f \text{ and } h \text{ may be **linearly approximated**}\]

\[\Rightarrow\text{This is achieved using **Taylor series**}\]

Extended Kalman filter provides an **ad-hoc** extension to Kalman filters based on these assumptions, to compute the estimate, \(\hat{x}_k\).
Update Equations for EKF

We have the iterative update algorithm

Time update

\[
\begin{align*}
\hat{x}_k^- &= f(\hat{x}_{k-1}, u_{k-1}, 0) \\
P_k^- &= A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \\
A_k &= \frac{\partial f}{\partial x} |_{\hat{x}_{k-1}, u_{k-1}, 0} \\
W_k &= \frac{\partial f}{\partial \omega} |_{\hat{x}_{k-1}, u_{k-1}, 0}
\end{align*}
\]

Measurement update

\[
\begin{align*}
\hat{x}_k &= \hat{x}_k^- + K_k (z_k - h(\hat{x}_k^-, 0)) \\
K_k &= \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + V_k R V_k^T} \\
P_k &= (I - K_k H_k) P_k^- \\
H_k &= \frac{\partial h}{\partial x} |_{\hat{x}_k^-, 0} \\
V_k &= \frac{\partial h}{\partial \nu} |_{\hat{x}_k^-, 0}
\end{align*}
\]

To run the algorithm

- Again, estimate $Q$ and $R$ offline (sys ID)
- Start filter with some initial $\hat{x}_0$ and $P_0$
- The values for $A$, $W$, $H$, and $V$ change in each iteration
- Similar to Kalman filter, $P_k$ and $K_k$ can converge quickly if the model is right
A Note on the Two Update Steps

Both Kalman filter and EKF have time and measurement updates

⇒ Kalman filter

<table>
<thead>
<tr>
<th>Time update</th>
<th>Measurement update</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{x}<em>k^- = A\hat{x}</em>{k-1} + Bu_{k-1}$</td>
<td>$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$</td>
</tr>
<tr>
<td>$P_k^- = AP_{k-1}A^T + Q$</td>
<td>$K_k = P_k^-H^T(HP_k^-H^T + R)^{-1}$, $P_k = (I - K_kH)P_k^-$</td>
</tr>
</tbody>
</table>

⇒ Extended Kalman filter

<table>
<thead>
<tr>
<th>Time update</th>
<th>Measurement update</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{x}<em>k^- = f(\hat{x}</em>{k-1}, u_{k-1}, 0)$</td>
<td>$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0))$</td>
</tr>
<tr>
<td>$P_k^- = A_kP_{k-1}A_k^T + W_kQ_{k-1}W_k^T$</td>
<td>$K_k = P_k^-H_k^T(H_kP_k^-H_k^T + V_kR_{k}V_k^T)^{-1}$</td>
</tr>
<tr>
<td>$A_k = \frac{\partial f}{\partial x}</td>
<td><em>{\hat{x}</em>{k-1}, u_{k-1}, 0}$</td>
</tr>
<tr>
<td>$W_k = \frac{\partial f}{\partial \omega}</td>
<td><em>{\hat{x}</em>{k-1}, u_{k-1}, 0}$</td>
</tr>
</tbody>
</table>

One can mix and match these!

⇒ E.g., one can build a filter with $T_2$ and $M_1$. Or $T_1$ and $M_2$

⇒ This generally applies to two-stage filters including later ones
Issues with the Extended Kalman Filter

There are many issues with EKF

- Can perform poorly with highly non-linear $f$ and $h$
  - i.e., Taylor expansion may not capture $f$ or $h$ well enough
  - This can be particularly true for $f$

- Also, computing $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial \omega}$, and $\frac{\partial h}{\partial v}$ may be difficult or impossible
  - E.g., $f$ may be very complex and may not even have a closed form
Unscented Kalman Filter and Particle Filter

Unscented Kalman filter (UKF) and Particle filter avoid such problems

⇒ For time update
  ⇒ Directly sample \( \hat{x}_{k-1} \) and obtain a certain number of samples \( \tilde{x}_{k-1}^i \) with weights
  ⇒ Directly “push” the samples through \( f \)
  ⇒ Compute \( \hat{x}_k^- \) and \( P_k^- \) from these updated samples
  ⇒ This can be imagined as running many Kalman filters

⇒ Similar steps for measurement update

⇒ Comparison to Kalman filter/EKF

⇒ Difference between UKF and particle filters
  ⇒ UKF uses deterministic samples (so called “unscented” transformation)
  ⇒ Particle filters use Monte Carlo sampling, usually with more samples than UKF
Simultaneous Localization and Mapping

Suppose you arrived a new town (e.g., travel, or playing an RPG)

- How do you explore?
- Move around and look for landmarks
  - Houses, buildings, roads, etc.
- Build map with the landmarks
- Localize yourself on the map

Robots would need to do the same

- Build a map using landmarks
- Localize using the map
- Simultaneous localization and mapping (SLAM)
- This is partially a “chicken-egg” problem
- Very similar to Kalman filter
  - Time update
  - Measurement update
SLAM, More Formally

Problem setup

⇒ A robot moves in an environment with states $x_k$
⇒ Make relative observation of $m = m_1, ..., m_n$ landmark locations
⇒ State history $X_k = \{x_0, ..., x_k\} = X_{k-1} \cup \{x_k\}$
⇒ Control $u_k$ applied at $x_{k-1}$
⇒ $U_k = \{u_1, ..., u_k\} = U_{k-1} \cup \{u_k\}$
⇒ $z_{ik}$: an observation of the $i$-th landmark at time step $k$
⇒ $z_k = (z_{1k}, ..., z_{nk})$ and $Z_k = \{z_1, ..., z_k\} = Z_{k-1} \cup \{z_k\}$
⇒ Motion model: $P(x_k | x_{k-1}, u_k)$
  ⇒ This is similar to $x_k = f(x_{k-1}, u_k, \omega)$
  ⇒ Note the indexing is different from Kalman filter – this is due to convention
⇒ Observation model: $P(z_k | x_k, m)$

SLAM is to compute the following probability distribution

$$P(x_k, m | Z_k, U_k, x_0)$$

SLAM also has time and observation (measurement) updates
SLAM Time Update

The time update makes predictions based on $x_{k-1}$ and $u_k$, i.e.,

$$P( x_k, \mathbf{m} | Z_{k-1}, U_k, x_0 ) = \int P( x_k, x_{k-1}, \mathbf{m} | Z_{k-1}, U_k, x_0 ) dx_{k-1} \quad \text{(marginalization)}$$

$$= \int \frac{P(x_k,x_{k-1}, \mathbf{m}, Z_{k-1}, U_k, x_0)}{P(Z_{k-1}, U_k, x_0)} dx_{k-1}$$

$$= \int \frac{P(x_k,x_{k-1}, \mathbf{m}, Z_{k-1}, U_k, x_0)}{P(x_{k-1}, \mathbf{m}, Z_{k-1}, U_k, x_0)} \frac{P(x_{k-1}, \mathbf{m}, Z_{k-1}, U_k, x_0)}{P(Z_{k-1}, U_k, x_0)} dx_{k-1}$$

$$= \int P( x_k | x_{k-1}, \mathbf{m}, Z_{k-1}, U_k, x_0 ) P(x_{k-1}, \mathbf{m} | Z_{k-1}, U_k, x_0) dx_{k-1}$$

$$= \int P( x_k | x_{k-1}, u_k ) P(x_{k-1}, \mathbf{m} | Z_{k-1}, U_{k-1}, x_0) dx_{k-1}$$

⇒ The last step applies two conditional independences

⇒ $P(x_k | x_{k-1}, u_k) = P(x_k | x_{k-1}, \mathbf{m}, Z_{k-1}, U_k, x_0)$

⇒ $P(x_{k-1}, \mathbf{m} | Z_{k-1}, U_{k-1}, x_0) = P(x_{k-1}, \mathbf{m} | Z_{k-1}, U_k, x_0)$

⇒ The term $P( x_k | x_{k-1}, u_k )$ is provided

⇒ E.g., $x_k = Ax_{k-1} + Bu_k + \omega_{k-1}$ in a Kalman filter

⇒ The term $P(x_{k-1}, \mathbf{m} | Z_{k-1}, U_{k-1}, x_0)$ is from previous iteration or $x_0$

⇒ So this step is basically the same as the time update of a Kalman filter
SLAM Observation Update

The observation update estimate $x_k, m$ based on time update and $z_k$

$$
P ( x_k, m \mid Z_k, U_k, x_0 )
= P ( x_k, m \mid Z_k, Z_{k-1}, U_k, x_0 )
= \frac{P(x_k, m, z_k, Z_{k-1}, U_k, x_0)}{P(z_k, Z_{k-1}, U_k, x_0)}
= \frac{P(x_k, m, z_k, Z_{k-1}, U_k, x_0) P(x_k, m, Z_{k-1}, U_k, x_0)}{P(x_k, m, Z_{k-1}, U_k, x_0) P(Z_{k-1}, U_k, x_0)} \frac{P(Z_{k-1}, U_k, x_0)}{P(z_k, Z_{k-1}, U_k, x_0)}
= \frac{P(z_k \mid x_k, m) P(x_k, m \mid Z_{k-1}, U_k, x_0)}{P(z_k \mid Z_{k-1}, U_k, x_0)}
\propto P(z_k \mid x_k, m) P(x_k, m \mid Z_{k-1}, U_k, x_0)
$$

$\Rightarrow$ The term $P(z_k \mid Z_{k-1}, U_k, x_0)$ can be normalized and does not matter

$\Rightarrow$ The term $P(z_k \mid x_k, m)$ is based on observation

$\Rightarrow$ In extended Kalman filter, this is just $z_k = h(x_k, v_k)$

$\Rightarrow$ In SLAM this is the challenging step

$\Rightarrow$ Uses many techniques, e.g., iterative closest point fitting (ICP)

$\Rightarrow$ Not part of the focus of this course – mostly computer vision techniques

$\Rightarrow$ The term $P(x_k, m \mid Z_{k-1}, U_k, x_0)$ is from time update
Sensing Review

Sensor mechanisms

- Triangulation
- Trilateration

Localization techniques

- Triangulation
- Trilateration

Bayesian filters

- Kalman filter, EKF
- UKF/particle filters
- Simultaneous localization and mapping (SLAM)