EKF, UKF

Pieter Abbeel
UC Berkeley EECS

Many slides adapted from Thrun, Burgard and Fox, Probabilistic Robotics

---

**Kalman Filter**

- Kalman Filter = special case of a Bayes’ filter with dynamics model and sensory model being linear Gaussian:

\[
X_0 \sim \mathcal{N}(\mu_0, \Sigma_0) \\
X_{t+1} = A_t X_t + B_t u_t + \epsilon_t \quad \epsilon_t \sim \mathcal{N}(0, Q_t) \\
Z_t = C_t X_t + d_t + \delta_t \quad \delta_t \sim \mathcal{N}(0, R_t)
\]
Kalman Filtering Algorithm

- At time 0: \( X_0 \sim \mathcal{N}(\mu_0|0, \Sigma_0|0) \)
- For \( t = 1, 2, \ldots \)
  - Dynamics update:
    \[
    \begin{align*}
    \mu_{t+1|0:t} &= A_t \mu_{t|0:t} + B_t u_t \\
    \Sigma_{t+1|0:t} &= A_t \Sigma_{t|0:t} A_t^T + Q_t
    \end{align*}
    \]
  - Measurement update:
    \[
    \begin{align*}
    K_{t+1} &= \Sigma_{t+1|0:t} C_{t+1}^T (C_{t+1} \Sigma_{t+1|0:t} C_{t+1}^T + R_{t+1})^{-1} \\
    \mu_{t+1|0:t+1} &= \mu_{t+1|0:t} + K_{t+1} (z_{t+1} - (C_{t+1} \mu_{t+1|0:t} + d)) \\
    \Sigma_{t+1|0:t+1} &= (I - K_{t+1} C_{t+1}) \Sigma_{t+1|0:t}
    \end{align*}
    \]

Nonlinear Dynamical Systems

- Most realistic robotic problems involve nonlinear functions:
  \[
  \begin{align*}
  X_{t+1} &= f_t(X_t, u_t) + \varepsilon_t \quad \varepsilon_t \sim \mathcal{N}(0, Q_t) \\
  Z_t &= h_t(X_t) + \delta_t \quad \delta_t \sim \mathcal{N}(0, R_t)
  \end{align*}
  \]

- Versus linear setting:
  \[
  \begin{align*}
  X_{t+1} &= A_t X_t + B_t u_t + \varepsilon_t \quad \varepsilon_t \sim \mathcal{N}(0, Q_t) \\
  Z_t &= C_t X_t + d_t + \delta_t \quad \delta_t \sim \mathcal{N}(0, R_t)
  \end{align*}
  \]
Linearity Assumption Revisited

$y = x + b$

Non-linear Function

"Gaussian of $p(y)$" has mean and variance of $y$ under $p(y)$
EKF Linearization (1)

- $p(x)$ has high variance relative to region in which linearization is accurate.
EKF Linearization (3)

p(x) has small variance relative to region in which linearization is accurate.

Dynamics model: for \( x_t \) "close to" \( \mu_t \) we have:

\[
f_t(x_t, u_t) \approx f_t(\mu_t, u_t) + \frac{\partial f_t(\mu_t, u_t)}{\partial x_t}(x_t - \mu_t)
= f_t(\mu_t, u_t) + F_t(x_t - \mu_t)
\]

Measurement model: for \( x_t \) "close to" \( \mu_t \) we have:

\[
h_t(x_t) \approx h_t(\mu_t) + \frac{\partial h_t(\mu_t)}{\partial x_t}(x_t - \mu_t)
= h_t(\mu_t) + H_t(x_t - \mu_t)
\]
EKF Linearization: Numerical

\[ f_t(x_t, u_t) \approx f_t(\mu_t, u_t) + \frac{\partial f_t(\mu_t, u_t)}{\partial x_t}(x_t - \mu_t) \]

\[ = f_t(\mu_t, u_t) + F_t(x_t - \mu_t) \]

- Numerically compute \( F_t \) column by column:

\[ \text{for } i = 1, \ldots, n \quad F_t(:, i) = \frac{f_t(\mu_t + \epsilon e_i, u_t) - f_t(\mu_t - \epsilon e_i, u_t)}{2\epsilon} \]

- Here \( e_i \) is the basis vector with all entries equal to zero, except for the \( i \)’th entry, which equals 1.

- If wanting to approximate \( F_t \) as closely as possible then \( \epsilon \) is chosen to be a small number, but not too small to avoid numerical issues

---

Ordinary Least Squares

- Given: samples \{\((x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\)\}

- Problem: find function of the form \( f(x) = a_0 + a_1 x \) that fits the samples as well as possible in the following sense:

\[ \min_{a_0, a_1} \frac{1}{2} \sum_{i=1}^{m} (a_0 + a_1 x^{(i)} - y^{(i)})^2 \]
Recall our objective: \[ \min_{a_0,a_1} \frac{1}{2} \sum_{i=1}^{m} (a_0 + a_1 x^{(i)} - y^{(i)})^2 \]

Let’s write this in vector notation:
\[ \bar{x}^{(i)} = \begin{bmatrix} 1 \\ x^{(i)} \end{bmatrix} \quad a = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \]
giving:
\[ \min_a \frac{1}{2} \sum_{i=1}^{m} (\bar{x}^{(i)\top} a - y^{(i)})^2 \]

Set gradient equal to zero to find extremum:
\[
0 = \nabla_a(\ldots) = \sum_{i=1}^{m} \bar{x}^{(i)}(\bar{x}^{(i)\top} a - y^{(i)}) \\
= \left( \sum_{i=1}^{m} \bar{x}^{(i)\top} \right) a - \sum_{i=1}^{m} \bar{x}^{(i)} y^{(i)} \\
= \bar{X} \bar{X}^\top a - \bar{X} y \\
a = (\bar{X} \bar{X}^\top)^{-1} \bar{X} y
\]

(See the Matrix Cookbook for matrix identities, including derivatives.)

For our example problem we obtain \( a = [4.75; 2.00] \)
More generally: \( x^{(i)} \in \mathbb{R}^n \)
\[
\min_{a_0, a_1, a_2, \ldots, a_n} \frac{1}{2} \sum_{i=1}^{m} (a_0 + a_1 x_1^{(i)} + a_2 x_2^{(i)} + \ldots + a_n x_n^{(i)} - y^{(i)})^2
\]

In vector notation:
- \( \tilde{x}^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}, \ a = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \) gives:
\[
\min_a \frac{1}{2} \sum_{i=1}^{m} (\tilde{x}^{(i)\top} a - y^{(i)})^2
\]

Set gradient equal to zero to find extremum (exact same derivation as two slides back):

\[
\begin{align*}
a &= (X \tilde{X})^{-1} \tilde{X} y \\
\tilde{X} &= \begin{bmatrix} 1 & 1 & \ldots & 1 \\
x_1^{(1)} & x_1^{(2)} & \ldots & x_1^{(m)} \\
x_2^{(1)} & x_2^{(2)} & \ldots & x_2^{(m)} \\
\vdots & \vdots & \ddots & \vdots \\
x_n^{(1)} & x_n^{(2)} & \ldots & x_n^{(m)} \\
y^{(1)} & y^{(2)} & \ldots & y^{(m)}
\end{bmatrix}
\end{align*}
\]

Vector Valued Ordinary Least Squares Problems

So far have considered approximating a scalar valued function from samples \( \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\} \) with \( x^{(i)} \in \mathbb{R}^n, y^{(i)} \in \mathbb{R} \)

A vector valued function is just many scalar valued functions and we can approximate it the same way by solving an OLS problem multiple times. Concretely, let \( y^{(i)} \in \mathbb{R}^p \) then we have:

Find \( a_0 \in \mathbb{R}^p, A \in \mathbb{R}^{n \times p} \), such that \( \forall i = 1, \ldots, m \ a_0 + A x^{(i)} \approx y^{(i)} \).

In our vector notation:
\[
\tilde{x}^{(i)\top} = \begin{bmatrix} 1 \\ x^{(i)\top} \end{bmatrix}, \quad \tilde{A} = \begin{bmatrix} a_0 \\ A \end{bmatrix}
\]

Find \( \tilde{A} \) such that \( \forall i = 1, \ldots, m \ \tilde{A} x^{(i)} \approx y^{(i)} \).

This can be solved by solving a separate ordinary least squares problem to find each row of \( \tilde{A} \).
Solving the OLS problem for each row gives us:

\[
(\bar{A}_{j,:})^T = (\bar{X} \bar{X}^T)^{-1} \bar{X} y_j^{(0,\ldots,m)}
\]

\[
y_j^{(0,\ldots,m)} = \begin{bmatrix} y_j^{(0)} & y_j^{(1)} & \cdots & y_j^{(m)} \end{bmatrix}^T
\]

Each OLS problem has the same structure. We have

\[
\bar{A}^T = (\bar{X} \bar{X}^T)^{-1} \bar{X} Y
\]

\[
Y = \begin{bmatrix} y_1^{(0,\ldots,m)} & y_2^{(0,\ldots,m)} & \cdots & y_p^{(0,\ldots,m)} \end{bmatrix}
\]

\[
= \begin{bmatrix} y_1^{(0)} & y_2^{(0)} & \cdots & y_p^{(0)} \\
(y_1^{(1)} & y_2^{(1)} & \cdots & y_p^{(1)} \\
(\cdots & \cdots & \cdots & \cdots \\
(y_1^{(m)} & y_2^{(m)} & \cdots & y_p^{(m)}
\end{bmatrix}
\]

---

**Vector Valued Ordinary Least Squares and EKF Linearization**

- Approximate \( x_{t+1} = f_t(x_t, u_t) \)
  
  with affine function \( a_0 + F_t \cdot x_t \)
  
  by running least squares on samples from the function:
  
  \( \{( x_1^{(i)}, y_1^{(i)}=f(x_1^{(i)},u_i), ( x_2^{(i)}, y_2^{(i)}=f(x_2^{(i)},u_i), \ldots, ( x_p^{(m)}, y_p^{(m)}=f_t(x_p^{(m)},u) \} \)

\[
\begin{bmatrix} a_0 & F_t \end{bmatrix}^T = \bar{A}^T = (\bar{X} \bar{X}^T)^{-1} \bar{X} Y
\]

- Similarly for \( z_{t+1} = h_t(x_t) \)
OLS and EKF Linearization: Sample Point Selection

- OLS vs. traditional (tangent) linearization:

  ![Graph showing OLS vs. traditional linearization](image)

OLS Linearization: choosing samples points

- Perhaps most natural choice:
  - $\mu_t, \mu_t + \Sigma_t^{1/2}, \mu_t - \Sigma_t^{1/2}$

  - reasonable way of trying to cover the region with reasonably high probability mass
Analytical vs. Numerical Linearization

- Numerical (based on least squares or finite differences) could give a more accurate “regional” approximation. Size of region determined by evaluation points.

- Computational efficiency:
  - Analytical derivatives can be cheaper or more expensive than function evaluations

- Development hint:
  - Numerical derivatives tend to be easier to implement
  - If deciding to use analytical derivatives, implementing finite difference derivative and comparing with analytical results can help debugging the analytical derivatives

---

EKF Algorithm

- At time 0: \( X_0 \sim \mathcal{N}(\mu_0|0;0, \Sigma_0|0) \)
- For \( t = 1, 2, \ldots \)
  - Dynamics update: \( f_t(x_t, u_t) \approx a_{0,t} + F_t(x_t - \mu_{t|0:t}) \)
    \[
    (a_{0,t}, F_t) = \text{linearize}(f_t, \mu_{t|0:t}, \Sigma_{t|0:t}, u_t)
    \]
    \[
    \mu_{t+1|0:t} = a_{0,t}
    \]
    \[
    \Sigma_{t+1|0:t} = F_t \Sigma_{t|0:t} F_t^T + Q_t
    \]
  - Measurement update: \( h_{t+1}(x_{t+1}) \approx c_{0,t+1} + H_{t+1}(x_{t+1} - \mu_{t+1|0:t}) \)
    \[
    (c_{0,t+1}, H_{t+1}) = \text{linearize}(h_{t+1}, \mu_{t+1|0:t}, \Sigma_{t+1|0:t})
    \]
    \[
    K_{t+1} = \Sigma_{t+1|0:t} H_{t+1}^T (H_{t+1} \Sigma_{t+1|0:t} H_{t+1}^T + R_{t+1})^{-1}
    \]
    \[
    \mu_{t+1|0:t+1} = \mu_{t+1|0:t} + K_{t+1} (z_{t+1} - c_{0,t+1})
    \]
    \[
    \Sigma_{t+1|0:t+1} = (I - K_{t+1} H_{t+1}) \Sigma_{t+1|0:t}
    \]
EKF Summary

- **Highly efficient**: Polynomial in measurement dimensionality \( k \) and state dimensionality \( n \):
  \[ O(k^{2.376} + n^2) \]

- **Not optimal**!
- **Can diverge** if nonlinearities are large!
- Works surprisingly well even when all assumptions are violated!
Assume we know the distribution over $X$ and it has a mean $\bar{x}$.

$Y = f(X)$

EKF approximates $f$ by first order and ignores higher-order terms.

UKF uses $f$ exactly, but approximates $p(x)$.

---

UKF intuition why it can perform better

- EKF approximates $f$ by first order and ignores higher-order terms.
- UKF uses $f$ exactly, but approximates $p(x)$.

\[ [\text{Julier and Uhlmann, 1997}] \]
When would the UKF significantly outperform the EKF?

Analytical derivatives, finite-difference derivatives, and least squares will all end up with a horizontal linearization → they’d predict zero variance in \( Y = f(X) \)

A crude preliminary investigation of whether we can get EKF to match UKF by particular choice of points used in the least squares fitting

Let’s assume the mean is zero, and ignore it in this consideration. Not sure yet if assuming mean would fundamentally change anything or not.

For the EKF we want to find \( A \) such that

\[
AX = f(X)
\]

The least squares solution for \( A \) is given by:

\[
A = \left( f(X) X^\top X \right)^{-1} f(X)
\]

We choose \( X = \mathcal{D}(x_1-x_0) \) and get:

\[
A = \left( f(x_1) f(x_1)^\top \right) \left( \mathcal{D}(x_1-x_0) \mathcal{D}(x_1-x_0)^\top \right)^{-1}
\]

\[
= \left( f(x_1) f(x_1)^\top \right) \mathcal{D}(x_1-x_0) \mathcal{D}(x_1-x_0)^\top
\]

This gives for the covariance matrix after propagation:

\[
\mathbf{A} \mathbf{A}^\top = \frac{1}{2} \left[ f(x_1) f(x_1)^\top \right] \mathbf{D}(x_1-x_0) \mathbf{D}(x_1-x_0)^\top
\]

\[
= \frac{1}{2} \left[ f(x_1) f(x_1)^\top \right] \mathbf{D}(x_1-x_0) \mathbf{D}(x_1-x_0)^\top
\]

\[
= \frac{1}{2} \left[ f(x_1) f(x_1)^\top \right] \mathbf{D}(x_1-x_0) \mathbf{D}(x_1-x_0)^\top
\]

\[
= \frac{1}{2} \left[ f(x_1) f(x_1)^\top \right] \mathbf{D}(x_1-x_0) \mathbf{D}(x_1-x_0)^\top
\]

So we have that if \( f(x) = f(x) \), then \( \mathbf{A} \mathbf{A}^\top = 0 \), which is a terrible estimate!

**Fact:** Every \( f \) can be written as the sum of symmetric and anti-symmetric, i.e.,

\[
3f, f = f^s + f^as, \text{ with } f^s(x) = f(x) \text{ and } f^as(x) = -f(x).
\]

Indeed, this can be done by choosing \( f^s(x) = \frac{f(x) + f(-x)}{2} \) and \( f^as(x) = \frac{f(x) - f(-x)}{2} \).

Using this decomposition we get (after some algebraic simplifications by canceling out terms)

\[
\mathbf{A} \mathbf{A}^\top = f^s(x) f^a(x^\top)
\]

This suggests that even with what seems the most reasonable way to linearize to capture the nonlinear function \( f \), we only end up capturing its symmetric part (perhaps naturally so, as linearization only gives us an symmetric comp.)
Original unscented transform

- Picks a minimal set of sample points that match 1st, 2nd and 3rd moments of a Gaussian:

\[
\begin{align*}
X_0 &= \bar{x} \\
X_i &= \bar{x} + \left( \sqrt{(n + \kappa)P_{xx}} \right)_i \quad W_i = \kappa / (n + \kappa) \\
X_i+n &= \bar{x} - \left( \sqrt{(n + \kappa)P_{xx}} \right)_i \quad W_{i+n} = 1/2(n + \kappa)
\end{align*}
\]

- \(\bar{x}\) = mean, \(P_{xx}\) = covariance, \(i\rightarrow\) i'th column, \(x \in \mathbb{R}^n\)
- \(\kappa\) : extra degree of freedom to fine-tune the higher order moments of the approximation; when \(x\) is Gaussian, \(n+\kappa = 3\) is a suggested heuristic
- \(L = \sqrt{P_{xx}}\) can be chosen to be any matrix satisfying:
  - \(L L^T = P_{xx}\)

[Julier and Uhlmann, 1997]

Unscented Kalman filter

- Dynamics update:
  - Can simply use unscented transform and estimate the mean and variance at the next time from the sample points
- Observation update:
  - Use sigma-points from unscented transform to compute the covariance matrix between \(X_t\) and \(Z_t\). Then can do the standard update.
Algorithm Unscented_Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
1. $X_{t-1} = (\mu_{t-1}, \mu_{t-1} + \gamma \sqrt{\Sigma_{t-1}}, \mu_{t-1} - \gamma \sqrt{\Sigma_{t-1}})$
2. $\tilde{X}_t = g(\mu_t, X_{t-1})$
3. $\tilde{\mu}_t = \sum_{i=0}^{2n} w_i[i] X_t^i[i]$
4. $\tilde{\Sigma}_t = \sum_{i=0}^{2n} w_i[i] (\tilde{X}_t^i[i] - \tilde{\mu}_t) (\tilde{X}_t^i[i] - \tilde{\mu}_t)^T + R_t$
5. $\tilde{X}_t = (\tilde{\mu}_t, \tilde{\mu}_t + \gamma \sqrt{\tilde{\Sigma}_t}, \tilde{\mu}_t - \gamma \sqrt{\tilde{\Sigma}_t})$
6. $\tilde{z}_t = h(\tilde{X}_t)$
7. $\tilde{z}_t = \sum_{i=0}^{2n} w_i[i] \tilde{z}_t^i$
8. $S_t = \sum_{i=0}^{2n} w_i[i] (\tilde{Z}_t^i[i] - \tilde{z}_t) (\tilde{Z}_t^i[i] - \tilde{z}_t)^T + Q_t$
9. $S_{t}^{x,x} = \sum_{i=0}^{2n} w_i[i] (\tilde{X}_t^i[i] - \tilde{\mu}_t) (\tilde{Z}_t^i[i] - \tilde{z}_t)^T$
10. $K_t = S_{t}^{x,x} S_t^{-1}$
11. $\mu_t = \tilde{\mu}_t + K_t (z_t - \tilde{z}_t)$
12. $\Sigma_t = \Sigma_{t} - K_t S_t K_t^T$
13. return $\mu_t, \Sigma_t$

Here $L = \sqrt{\Sigma}$ can be chosen to be any $n \times n$ matrix satisfying: $L L^T = \Sigma$

[Table 3.4 in Probabilistic Robotics]

---

**UKF Summary**

- **Highly efficient:** Same complexity as EKF, with a constant factor slower in typical practical applications
- **Better linearization than EKF:** Accurate in first two terms of Taylor expansion (EKF only first term) + capturing more aspects of the higher order terms
- **Derivative-free:** No Jacobians needed
- **Still not optimal!**