An extended Kalman filter for a mobile robot

© Chris Atkeson 2004

A mobile robot (base)

| \( v_R \) | Right wheel velocity |
| \( v_L \) | Left wheel velocity |
| \( \theta \) | Angle to X axis |
| \( l \) | Distance between wheels |

Sensors: Encoder, Gyro, Vision

Zooming In On Previous Slide

Kinematics

\[
\begin{align*}
x_{k+1} &= x_k - 0.5(v_R + v_L)dt \sin \theta_k \\
y_{k+1} &= y_k + 0.5(v_R + v_L)dt \cos \theta_k \\
\theta_{k+1} &= \theta_k + \frac{v_R - v_L}{l}dt \\
\dot{\theta} &= \frac{v_R - v_L}{l}, \quad v_{tot} = \frac{v_R + v_L}{2}
\end{align*}
\]

Extended Kalman Filter (Kinematic)

\[
x_{k+1} = Ax_k + Bu_k + w_k
\]

\[
c_{ek} = e_{x_k} - e_{\dot{x}_k} \quad c_{ek} = e_{\dot{x}_k} - e_{\dot{x}_k}
\]

\[
u = \begin{bmatrix} e_x & e_\dot{x} \end{bmatrix}^T
\]

\[
A = \begin{bmatrix} 1 & 0 & 0 & -0.5dt \sin \theta_k & -0.5dt \sin \theta_k \\
0 & 1 & 0 & 0.5dt \cos \theta_k & 0.5dt \cos \theta_k \\
0 & 0 & 1 & dt/l & -dt/l \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \end{bmatrix}
\]
Process Noise and Initial Variance

\[ Q = \mathbb{E}\{\mathbf{w}\mathbf{w}^T\} = \begin{bmatrix} Q_{11} & 0 & \cdots & 0 \\ 0 & Q_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_{mm} \end{bmatrix} \]

\[ P_0 = \begin{bmatrix} \varepsilon & 0 & \cdots & 0 \\ 0 & \varepsilon & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \varepsilon \end{bmatrix} \]

Prediction Equations

\[ x^-_k = A x^-_{k-1} + B u_{k-1} \]

\[ P^-_k = A P^-_{k-1} A^T + Q \]

Encoder, Gyro update

\[ z_k = H x_k + v_k \]

\[ z = [e_R, e_L, g]^T \]

\[ H = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/l & -1/l \end{bmatrix} \]

Measurement Noise

\[ R = \mathbb{E}\{\mathbf{v}\mathbf{v}^T\} = \begin{bmatrix} R_{11} & 0 & \cdots & 0 \\ 0 & R_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{nn} \end{bmatrix} \]

Measurement Update

\[ K = P^-_k H^T (H P^-_k H^T + R)^{-1} \]

\[ x^-_k = x^-_k + K (z_k - H x^-_k) \]

\[ P^-_k = (I - KH) P^-_k \]

Vision Update (velocity)

\[ z = g(x) \]

\[ z = \begin{bmatrix} v_r \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0.5(v_r + v_L) \\ (v_r - v_L)/l \end{bmatrix} \]

\[ H = \frac{\partial g}{\partial x} = \begin{bmatrix} 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 1/l & -1/l \end{bmatrix} \]
**Vision Update (landmark at \((x_L, y_L)\))**

\[
Z = g(x)
\]

\[
z = \begin{bmatrix}
x_v \\
y_v
\end{bmatrix} = \begin{bmatrix}
\cos(\theta) & \sin(\theta) \\
-\sin(\theta) & \cos(\theta)
\end{bmatrix} \begin{bmatrix}
x_L - x \\
y_L - y
\end{bmatrix}
\]

\[
H = \frac{\partial g}{\partial x}
\]

\[
H = \begin{bmatrix}
-x \cos(\theta) & -x \sin(\theta) & -(x_L - x) \sin(\theta) + (y_L - y) \cos(\theta) & 0 & 0 \\
-x \sin(\theta) & -x \cos(\theta) & -(x_L - x) \cos(\theta) - (y_L - y) \sin(\theta) & 0 & 0
\end{bmatrix}
\]

---

**SLAM**

- SLAM (Simultaneous Localization and Mapping) puts landmark locations as part of state to be estimated in the EKF.
- Prediction step is trivial (landmark doesn’t move)
- Measurement example below.
- Many landmarks means you have a very large state vector.
- Current research is addressing how to handle this well.

\[
x = \begin{bmatrix}
x, y, \theta, x_L, y_L
\end{bmatrix}^T
\]

---

**Notes**

- “Extended” KF because of angle in A matrix and full state in predicting visual observations
- A, B, H, Q, and R are sparse or diagonal, so should use special purpose coding for efficiency
- Dimensionality of inversion depends on number of sensors \((HP, H^T + R)^{-1}\)
- Different sampling rates can be handled with a variable length prediction and different Hs
- Need to measure gyro bias when stopped
- Need to handle slipping, vision glitches

---

**What I would like to see**

- Combine particle system and Kalman filter, so each particle maintains a simple distribution, instead of just a point estimate.
- More accurate modeling of belief state?
- More efficient?

---

**Particle Filtering with EKF Particles**

- Each particle is EKF, with weight.
- As particles overlap, merge them and add weights.
- As particles become infeasible, kill them.
- As particles become too certain, confuse them.
- Add new particles in empty spaces according to some prior.

---

- UWash demos
What If You Took Into Account the Mobile Robot Dynamics?

- Need to change input $u$ to be motor torques.
- This changes prediction step only.
- How do $v_R$ and $v_L$ depend on motor torques?

A Kalman Filter for a Rocket (1D)

Very Simple Dynamics

Dynamics

$$F = m \ddot{x}$$

So

$$\dot{x}_{k+1} = \dot{x}_k + \frac{F dt}{m}$$

Kalman Filter (Dynamic)

$$x_{k+1} = Ax_k + Bu_k + w_k$$

$$x = [x, \dot{x}]^T$$

$$u = [F]^T$$

$$A = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ dt / m \end{bmatrix}$$