EL2320 - EKF

John Folkesson

Extended Kalman Filter and Robot Localization
(Chap 7 in Thrun)

Kalman Filter - Hello World

We have a house with two rooms, one thermometer, and a heater in each room.

State: $$\mathbf{x} = \begin{pmatrix} temperature - In - Room - 1 \\ temperature - In - Room - 2 \\ outside - Temperature \end{pmatrix}$$

- The rate of change of the temperature in each room is proportional to the rate of energy (watts) entering or leaving the room.
- The rate of energy passing from room 1 to room 2 is proportional to the temperature difference. And similarly for the outside walls.
- The identical heaters produce heating watts proportional to some control signal plus Gaussian noise.
- The thermometer is in room 1 and is subject to Gaussian noise.

Kalman Filter - Hello World

$$\mathbf{x}_t = \begin{pmatrix} 1-a-b & a & b \\ d & 1-d-c & c \\ 0 & 0 & 1 \end{pmatrix} \mathbf{x}_{t-1} + \begin{pmatrix} g & 0 \\ 0 & g \\ 0 & 0 \end{pmatrix} \mathbf{u}_t + \varepsilon_t$$

$$\mathbf{y}_t = x_t^1 + \delta_t \qquad 1 >> a, b, c, d, g > 0$$

What are the physical meaning of a,b,c,d and g?
How would you check if this system observable?
What if the thermometer was outside?
What happens if I enter the house by opening and closing the door?
What would change if we added a second identical house but no more themometers?

A bonus questions not part of this course:
How would you check controllablity and stability?

Kalman Filter - Hello World

Observable? If we set $\mathbf{u} = 0$ and ignore the noise:

$$\mathbf{x}_t = \begin{pmatrix} 1-a-b & a & b \\ d & 1-d-c & c \\ 0 & 0 & 1 \end{pmatrix} \mathbf{x}_{t-1} = A\mathbf{x}_{t-1}$$

$$\mathbf{y}_t = (1,0,0)\mathbf{x}_t = C\mathbf{x}_t$$

$$\begin{pmatrix} y_{t-3} \\ y_{t-2} \\ y_{t-1} \end{pmatrix} = \begin{pmatrix} C \\ CA \\ CA^2 \end{pmatrix} \mathbf{x}_{t-3} = M\mathbf{x}_{t-3}$$

We can invert M iff determinant of $M \neq 0$

$$\begin{matrix} 1 & 0 & 0 \\ 1-a-b & a & b \\ (1-a-b)^2+ad & d(1-a-b)+(1-d-c)a & (2-a-b)b+ac \end{matrix}$$

Kalman Filter - Hello World

Themometer outside Observable? If we set $\mathbf{u} = 0$ and ignore the noise:

$$\mathbf{x}_t = \begin{pmatrix} 1-a-b & a & b \\ d & 1-d-c & c \\ 0 & 0 & 1 \end{pmatrix} \mathbf{x}_{t-1} = A\mathbf{x}_{t-1}$$

$$\mathbf{y}_t = (0,0,1)\mathbf{x}_t = C\mathbf{x}_t$$

$$\begin{pmatrix} y_{t-3} \\ y_{t-2} \\ y_{t-1} \end{pmatrix} = \begin{pmatrix} C \\ CA \\ CA^2 \end{pmatrix} \mathbf{x}_{t-3} = M\mathbf{x}_{t-3}$$

We can invert M iff determinant of $M \neq 0$

$$\begin{matrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{matrix}$$

Kalman Filter - Hello World

Controllable? Try to use $\mathbf{u}$ to get to any $\mathbf{x}$, again ignore the noise. Imagine $\mathbf{x}_{t-2} = 0 \Rightarrow \mathbf{x}_{t-1} = B\mathbf{u}_{t-1}$:

$$\mathbf{x}_t = \begin{pmatrix} 1-a-b & a & b \\ d & 1-d-c & c \\ 0 & 0 & 1 \end{pmatrix} \mathbf{x}_{t-1} + \begin{pmatrix} g & 0 \\ 0 & g \\ 0 & 0 \end{pmatrix} \mathbf{u}_t$$

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + B\mathbf{u}_t = AB\mathbf{u}_{t-1} + B\mathbf{u}_t$$

$$\mathbf{x}_t = \begin{pmatrix} B & AB \end{pmatrix} \begin{pmatrix} \mathbf{u}_t^T & \mathbf{u}_{t-1}^T \end{pmatrix}^T$$

We can invert for control if this det is not 0

$$\begin{matrix} g & 0 & \dots \\ 0 & g & \dots \\ 0 & 0 & 0 & 0 \end{matrix}$$

Kalman Filter - Algebraic Riccati Equation

If our A, B, C, R, and Q matricies are constant then:
$K_t = \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + R)^{-1}$ (This is known as the Kalman gain)
$\bar{\Sigma}_t = Q + A\Sigma_{t-1}A^T$ $\Sigma_t = \bar{\Sigma}_t - K_t C\bar{\Sigma}_t$
Can be combined to one equation:
$\bar{\Sigma}_t = Q + A[\bar{\Sigma}_{t-1} - \bar{\Sigma}_{t-1}C^T(C\bar{\Sigma}_{t-1}C^T + R)^{-1}C\bar{\Sigma}_{t-1}]A^T$
For the steady state $t \to \infty$:

$$\bar{\Sigma}_\infty = Q_t + A[\bar{\Sigma}_\infty - \bar{\Sigma}_\infty C^T(C\bar{\Sigma}_\infty C^T + R)^{-1}C\bar{\Sigma}_\infty]A^T$$

This is the so called Algebraic Riccati Equation.
Matlab has a solver for this:

$$[K_\infty, \bar{\Sigma}_\infty, \Sigma_{infty}, \Lambda] = dlge(A, I, C, Q, R)$$

where Λ are the eigenvalues of the estimator.
One can usually simplify the implementation to just use K_∞ from the start. We lose the fast convergance of the variable gain which matters most when state uncertianty is much greater than measurement uncertianty.

Kalman Filter - Control vs. Measurement

We separated the measurements data into update measurements $\mathbf{z}_t$ and 'control signals' $\mathbf{u}_t$. This would matter for an estimator that was part of a controller. There we need a real control and we worry about things like the stability and speed (pole placement) of our estimator.

For estimation this does not matter and we can do without any control signal or have a 'control' that is in reality just another sensor measurement.

On the other hand we must include a prediction step with its insertion of Gaussian noise for all parts of the state that evolve in time. This might be just $A = I$ and $Q > 0$ for these rows.

The parts of the state that do not evolve in time can be left out of the predict altogether. The A matrix would be 1 for these rows, while Q and B would be 0. The predict formulas would then be applied as usual.

Extended Kalman Filter

$$\mathbf{x}_t = \mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1}) + \varepsilon_t$$
$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \delta_t$$

$$\mathbf{x}_t \approx G_t(x_{t-1} - \mu_{t-1}) + \mathbf{g}(\mathbf{u}_t, \mu_{t-1}) + \varepsilon_t$$
$$\mathbf{y}_t = \mathbf{z}_t - \mathbf{h}(\bar{\mu}_\mathbf{t}) \approx H_t(\mathbf{x}_t - \bar{\mu}_\mathbf{t}) + \delta_t$$

So that H_t is the Jacobian of $\mathbf{h}$ evaluated at $\bar{\mu}_\mathbf{t}$ and G_t is the Jacobian of $\mathbf{g}$ wrt the $\mathbf{x}_{t-1}$ at $\mu_{\mathbf{t-1}}$.

We then treat this as a linear system and do Kalman predict and update.

$$G_t \iff A_t$$
$$\mathbf{g}(\mathbf{u}_t, \mu_{t-1}) \iff A_t\mu_{t-1} + B_t\mathbf{u}_t$$
$$H_t \iff C_t$$

$\mathbf{y}_t$ is defined non-linearly wrt. $\bar{\mu}_t$

Kalman to Extended Kalman Filter

$$\mathbf{x}_t = \mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1}) + \varepsilon_t \qquad \mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \delta_t$$

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + R_t)^{-1} \rightarrow K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + R_t)^{-1}$$

$$\mathbf{y}_t = \mathbf{z}_t - \bar{\mathbf{z}}_t - C_t \bar{\mu}_t \qquad \rightarrow \mathbf{y}_t = \mathbf{z}_t - \mathbf{h}(\bar{\mu}_t)$$

Predict phase:

$$\bar{\Sigma}_t = Q_t + A_t \Sigma_{t-1} A_t^T \qquad \rightarrow \bar{\Sigma}_t = Q_t + G_t \Sigma_{t-1} G_t^T$$

$$\bar{\mu}_{\mathbf{t}} = A_t \mu_{t-1} + B_t \mathbf{u}_t \qquad \rightarrow \bar{\mu}_{\mathbf{t}} = \mathbf{g}(\mathbf{u}_t, \mu_{t-1})$$

and update:

$$\Sigma_t = \bar{\Sigma}_t - K_t C_t \bar{\Sigma}_t \qquad \rightarrow \Sigma_t = \bar{\Sigma}_t - K_t H_t \bar{\Sigma}_t$$

$$\mu_t = \bar{\mu}_{\mathbf{t}} + K_t \mathbf{y}_t \qquad \rightarrow \mu_t = \bar{\mu}_{\mathbf{t}} + K_t \mathbf{y}_t$$

EKF Example - Robot Localization

Robot localization is a good illustration of the EKF. Lets have a robot moving in a plane with its position and orientation, pose, given by:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

So x and y are the cartesian coordinates and θ is the heading relative to the x-axis. Now we might have odometry derived from encoders for the wheel rotations:

$$\mathbf{u} = \begin{pmatrix} ds \\ d\theta \end{pmatrix}$$

EKF Example

Our dynamics then look like:

$$\mathbf{x}_t = \mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1}) + \varepsilon_t$$

$$\bar{\mu}_t = \mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1}) = \begin{pmatrix} x_{t-1} + ds_t \cos\theta_{t-1} \\ y_{t-1} + ds_t \sin\theta_{t-1} \\ \theta_{t-1} + d\theta_t \end{pmatrix};$$

$$\mathbf{x}_t \approx G_t(x_{t-1} - \mu_{t-1}) + \mathbf{g}(\mathbf{u}_t, \mu_{t-1}) + \varepsilon_t$$

$$G_t = \begin{pmatrix} 1 & 0 & -ds_t \sin\theta_{t-1} \\ 0 & 1 & ds_t \cos\theta_{t-1} \\ 0 & 0 & 1 \end{pmatrix}$$

EKF Example

What about noise Q_t. If the noise is due to the control signal we might use:

$$Q_t = \begin{pmatrix} \cos\theta_{t-1} & 0 \\ \sin\theta_{t-1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \sigma_{ds}^2 & 0 \\ 0 & \sigma_{d\theta}^2 \end{pmatrix} \begin{pmatrix} \cos\theta_{t-1} & \sin\theta_{t-1} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

To this we might add a disturbance that might add amounts to the diagonal of Q_t.

$$Q_t = \begin{pmatrix} \sigma_{ds}^2 \cos^2\theta_{t-1} + \sigma_{d,1}^2 & \sigma_{ds}^2 \cos\theta_{t-1}\sin\theta_{t-1} & 0 \\ \sigma_{ds}^2 \cos\theta_{t-1}\sin\theta_{t-1} & \sigma_{ds}^2 \sin^2\theta_{t-1} + \sigma_{d,1}^2 & 0 \\ 0 & 0 & \sigma_{d\theta}^2 + \sigma_{d,2}^2 \end{pmatrix}$$

EKF Example - Robot Localization

Measurements could be direct state measurements like GPS and Compass. Then the update measurement functions would be linear with diagonal $C_t = H_t$. More interesting are point features measured in 2D as a range and angle.

$$\mathbf{h}_t(\mathbf{x}_t) = \begin{pmatrix} \sqrt{(x_t - f_x)^2 + (y_t - f_y)^2} \\ \arctan\left((y_t - f_y), (x_t - f_x)\right) - \theta \end{pmatrix}$$

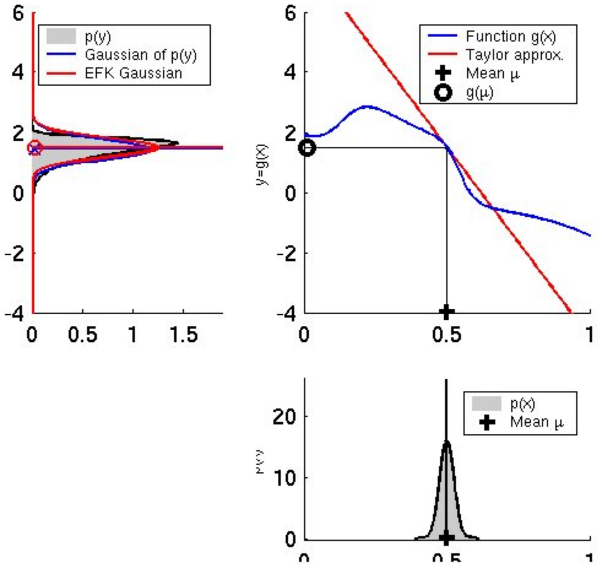
Here (f_x, f_y) is the coordinate of the point feature. To be useful the robot will need a map of many of these point features for the environment it is operating in.

$$R_t =$$
$$\begin{pmatrix} \frac{\partial h_1}{\partial f_x} & \frac{\partial h_1}{\partial f_y} \\ \frac{\partial h_2}{\partial f_x} & \frac{\partial h_2}{\partial f_y} \end{pmatrix} \begin{pmatrix} \sigma^2_{fx} & 0 \\ 0 & \sigma^2_{fy} \end{pmatrix} \begin{pmatrix} \frac{\partial h_1}{\partial f_x} & \frac{\partial h_2}{\partial f_x} \\ \frac{\partial h_1}{\partial f_y} & \frac{\partial h_2}{\partial f_y} \end{pmatrix} + \begin{pmatrix} \sigma^2_{range} & 0 \\ 0 & \sigma^2_{bear} \end{pmatrix}$$

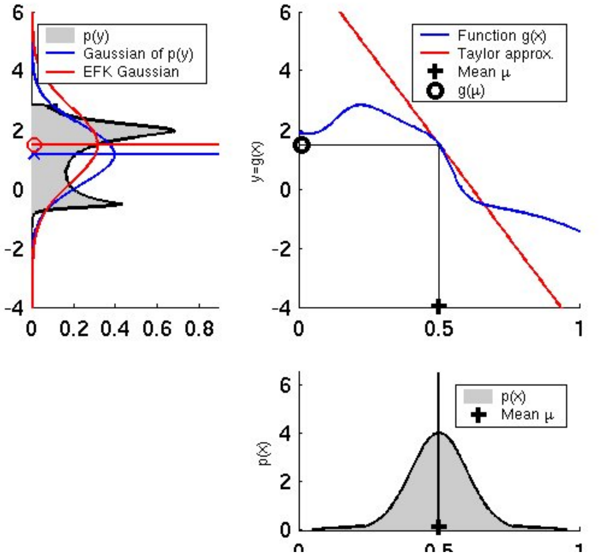
EKF Example - Robot Localization

The R_t can be written as the sensor uncertianty in range and bearing plus a term for the uncertianty of the feature location left and right multiplied by the Jacobian of $\mathbf{h}$ wrt (f_x, f_y) and its transpose.
For Simultaneous Localization and mapping SLAM we include the feature coordinates as part of the state. They become static state variables.

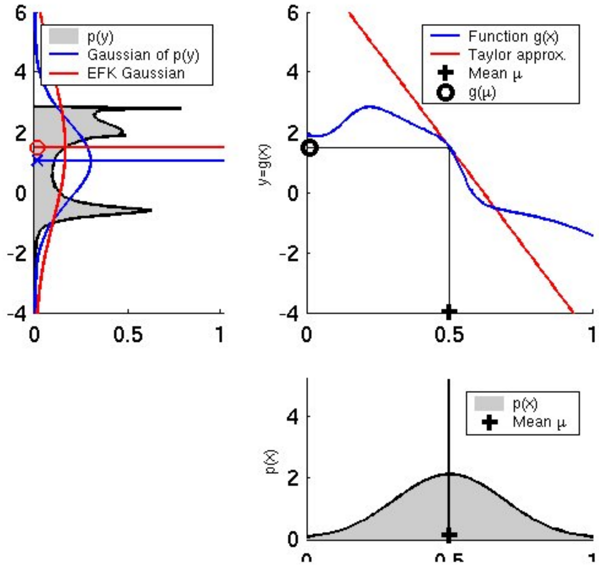
Extended Kalman Filter - Linearizations Narrow



Extended Kalman Filter - Linearizations Medium



Extended Kalman Filter - Linearizations Wide



Extended Kalman Filter - Linearizations

Solutions to this generally try to trace more points than just the mean through the non-linearity or find a better point point to linearize around.
Examples are:
Iterated Extended Kalman Filter IEKF
Particle Filters,
Graphical Methods that allow re-linearization.
Linear Regressive Kalman Filter LRKF which is a class that contains:

- Unscented Kalman Filter (UKF)
- Central Difference Filter (CDF)
- 1^{st} Order Divided Difference Filter (DD1)

Suggested reading for those wanting to go deeper:
Kalman Filters for non-linear systems: a comparison of performance, Lefebvre et. al. (2004)
Consistency of the EKF-SLAM Algorithm, Bailey et. al. (2006)

Extended Kalman Filter - Linearizations

One issue is the distortion of Gaussians by the non-linear transform. Another is that the original noise might not be Gaussian to begin with.
Either of these can cause a real EKF to converge to the wrong value, diverge, or just become inconsistent.
An estimator is said to be inconsistent if the a posteriori distribution is incorrectly estimated. So the mean and or the variance for Gaussians.

A too small Covariance is said to be optimistic while too large is pessimistic.
Optimistic estimates are a problem since they prevent the mean from moving to a better estimate.
Pessimistic estimates are a problem in that they make data association ambiguous.

Extended Kalman Filter - Linearizations

We can test the innovation process

$\delta_t = \mathbf{z}_t - \mathbf{h}(\bar{\mu}_\mathbf{t})$.

by comparing

$V = \sum \delta_t [H_t \bar{\Sigma}_t H_t^T + R_t]^{-1} \delta_t$

to a χ_n^2 where n is the sum of the dimensions of the $\mathbf{y}_t$.

More on this in a few slides from now.

Extended Kalman Filter - Incorrect Update

A practical issue is incorrect measurement data feeding the update. Causes are:

- spurious sensor reading,
- errors in processing the sensor data, e.g poor feature detection
- illusions e.g. reflections, shadows,...
- wrong coorespondance e.g poor feature recognition

The main weapon used is to examine the probability of the measurement and discard improbable ones.

Extended Kalman Filter - Coorespondance

Coorespondance is the matching of detected features in the sensor data to previously seen features.
To decide the coorespondance we to look at the match likelihood:

$$L_i = p(\mathbf{y}_{t,i}|\bar{\mu}_t)p(\bar{\mu}_t|y_1 \dots y_{t-1})$$

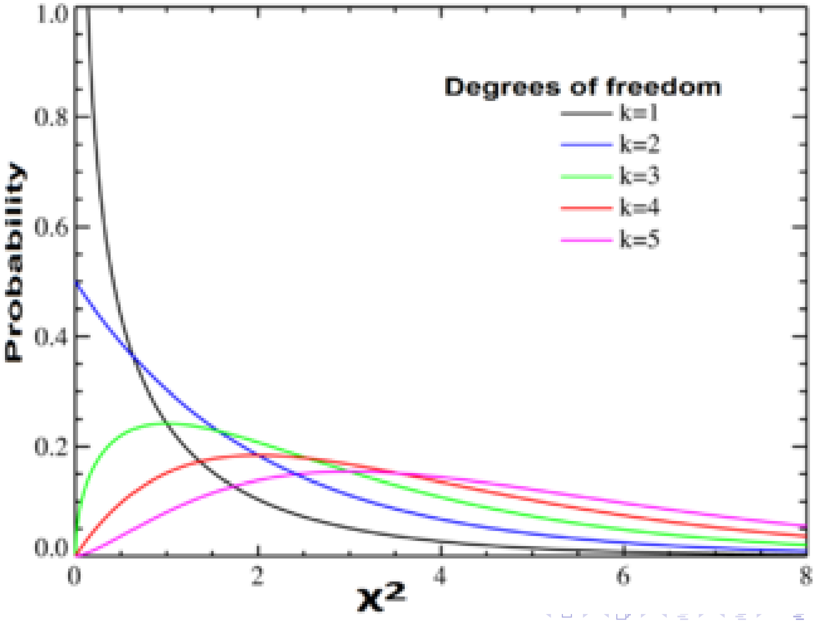
where $y_{t,i} = \mathbf{z}_t - \mathbf{h}_i(\bar{\mu}_\mathbf{t})$, is the innovation of the measurement when associated with feature i.
For Gaussian systems the probablity above evaluates to:

$$L_i = G(\mathbf{y}_{t,i}, 0, S_{t,i}).$$

$$S_{t,i} = H_{t,i}\bar{\Sigma}_t H_{t,i}^T + R_t$$

We choose the i that maximizes L_i subject to $L_i >$ threshold. If none are above the threshold then this is an unknown feature. This is known as maximum likelihood data association.

chi square


1.0
0.8
0.6
0.4
0.2
0.0
Probability
0
2
4
6
8
X^2
Degrees of freedom
k=1
k=2
k=3
k=4
k=5

Extended Kalman Filter - Mahalonobis Test

$L_i = G(\mathbf{y}_{t,i}, 0, S_{t,i}).$ $\quad S_{t,i} = H_{t,i}\bar{\Sigma}_t H_{t,i}^T + R_t$

m_i is the sum of the squares of normal random variables:
$m_i = \mathbf{y}_{t,i}^T S_{t,i}^{-1} \mathbf{y}_{t,i}$

is a χ_n^2 random variable with $n =$dimension of $\mathbf{y}_{t,i}$.

$\sqrt{m_i}$ is called the mahalonobis distance.

The Mahalanobis matching algorithm compares m_i rather than the more complicated L_i. The threshold can be gotten from tables of the χ_n^2 distribution at the desired confidence
level.
This does not give the same answer as MLE (for none
non-Gaussian likelihoods or if the different features have different S)but it is not clear which is better. MLE suffer from depending on 'scaling' which makes the Mahalanobis test 'feel' better.

Extended Kalman Filter - Matching

$$L_i = G(\mathbf{y}_{t,i}, 0, S_{t,i}), \quad S_{t,i} = H_{t,i}\bar{\Sigma}_t H_{t,i}^T + R_t, \quad m_i = \mathbf{y}_{t,i}^T S_{t,i}^{-1} \mathbf{y}_{t,i}$$

We have a robot in 2D, (x, y, θ), with a range sensor. The predict step produced this posteriori: $\overline{bel}(\mathbf{x}) = N(\bar{\mu}, \bar{\Sigma})$ where
$\bar{\mu} = (0, 0, 0)^T$ and

$$\bar{\Sigma} = \begin{pmatrix} .25 & 0 & 0 \\ 0 & .25 & 0 \\ 0 & 0 & .01 \end{pmatrix}$$

There are known features at $xy = (1, 0)$ and $(2, 0)$
The range sensor gives us a value $r = 1.5$ with Gaussian noise with $R = .5$.
What is our measurement function $\mathbf{h}(\mathbf{x}|feature_i) =$?
What is the jacobian $H(\mathbf{x}|i) =$?
What are the mahalanobis distances?
What is the 'likelihood' of the range being from each feature?
$p(r|\mathbf{x} = \bar{\mu}, feature_i)$