Lecture 7: *Kalman and Extended Kalman Filters*

Scribe: *Wesam Alwehaibi, Dhvani Vora, Donghui Cen*

September 18, 2019

Contents

1 Recap **3**

2 Kalman Filter **5**
2.1 Derivation of mean and variance 5
2.2 Algorithm 6
2.3 Understanding and Interpretation 6

3 Extended Kalman Filter **8**
3.1 Algorithm 9
3.2 Example 9

4 Gaussian Sum Filter **12**

1 Recap

In the last lecture, we have studied Bayes Filter and introduced Kalman filter. They both address the state estimation problem in robotics, which is to determine certain quantities from sensor data that are not directly observable. In the real world scenarios, the state estimation of a robot becomes a stochastic/probabilistic problem.

The robot has to estimate its own state from the observed data. A belief reflects the robot's internal knowledge about the state of the environment. As the robot cannot measure the state directly, it tries to estimate the state from the measurement data. There are various ways in which the robot can estimate this state. The most general algorithm for belief calculations is the Bayes' filter algorithm, which we studied in the last lecture. In Bayes Filter Algorithm, we just estimate the belief using conditional probabilities and Markov assumption. The Bayes filter algorithm is as follows:

1. Bayes_Filter $(bel(x_{t-1}), u_t, z_t)$:
2. for all x_t do:
3. $\quad \overline{bel}(x_t) = \Sigma P(x_t|u_t, x_{t-1}) bel(x_{t-1})$
4. $\quad bel(x_t) = \eta P(z_t|x_t) \overline{bel}(x_t)$
5. return $bel(x_t)$

Another way to define recursive state estimators is to assume that the noise is going to have Normal distribution. This family of state estimators, which assume the Gaussian distribution of noise is known as Gaussian filters. There are many different types of Gaussian filter algorithms like Kalman filter, Extended Kalman Filter, Guassian Sum Filter etc. The reason for using Gaussian distribution for estimation is that it is unimodal (they possess a single maxima). This is charactersitic of many tracking problems in robotics, where the distribution is focused around the true state with a small margin of uncertainty.

Let us start understanding this family from Kalman Filter. Kalman Filter is generally used when current state is a linear function of the previous state and the taken action. Following assumptions are made when using Kalman Filter:

1. The state at time t (x_t) is a linear transformation based on the last state (x_{t-1}) and the action taken at time t (u_t).
2. The uncertainty in the state at time t can be modelled as a multivariate Gaussian

distribution($\mathcal{E}$) with 0 mean and R_t covariance matrix.

3. Noise in the state estimation is uncorrelated from the state at time t. Thus, x_t and $\mathcal{E}$ are uncorrelated.

Note that in the general case R_t is dependent on time t, but due to the stationary assumption R is independent of time t and thus will be written as R instead of R_t. From the above assumptions, we can write the equation for state at time t (x_t) as follows:

$$x_t = Ax_{t-1} + Bu_t + \mathcal{E} \tag{1}$$

where $\mathcal{E} \sim N(0, R)$
The observation at time t (z_t) is also assumed to be a linear transformation of state at t (x_t) and the uncertainty in the observations/measurements can also be modelled as a Normal Distribution. Thus,

$$z_t = Cx_t + \delta \tag{2}$$

where $\delta \sim N(0, Q)$

2 Kalman Filter

2.1 Derivation of mean and variance

Let us assume that, expectation of state at time $t-1$ is μ_{t-1}.The expectation of x_t can be derived as follows:

$$E[x_t] = E[Ax_{t-1} + Bu_t + \mathcal{E}]$$

Due to linearity of expectation and u_t is a constant

$$E[x_t] = AE[x_{t-1}] + Bu_t + E[\mathcal{E}]$$

Thus,

$$\mu_t = A\mu_{t-1} + Bu_t + 0 = A\mu_{t-1} + Bu_t \tag{3}$$

Now, the covariance matrix at time t can be calculated as follows:

$\Sigma_t = E[(x_t - \mu_t)(x_t - \mu_t)^T]$

$= E[(Ax_{t-1} + Bu_t + \mathcal{E} - \mu_t)(Ax_{t-1} + Bu_t + \mathcal{E} - \mu_t)^T]$

Now using the equation (3) for μ_t,

$\Sigma_t = E[(Ax_{t-1} + \cancel{Bu_t} + \mathcal{E} - A\mu_{t-1} - \cancel{Bu_t})(Ax_{t-1} + \cancel{Bu_t} + \mathcal{E} - A\mu_{t-1} - \cancel{Bu_t})^T]$

$= E[(Ax_{t-1} + \mathcal{E} - A\mu_{t-1})(Ax_{t-1} + \mathcal{E} - A\mu_{t-1})^T]$

$= E[\{A(x_{t-1} - \mu_{t-1}) + \mathcal{E}\}\{(x_{t-1} - \mu_{t-1})^T A^T + \mathcal{E}^T\}]$

$= E[A(x_{t-1} - \mu_{t-1})(x_{t-1} - \mu_{t-1})^T A^T$
$+ \mathcal{E}(x_{t-1} - \mu_{t-1})^T A^T$
$+ A(x_{t-1} - \mu_{t-1})]\mathcal{E}^T$
$+ \mathcal{E}\mathcal{E}^T]$

Now, in the above equation,
$E[A(x_{t-1} - \mu_{t-1})(x_{t-1} - \mu_{t-1})^T A^T] = AE[(x_{t-1} - \mu_{t-1})(x_{t-1} - \mu_{t-1})^T]A^T$ and
as $E[(x_{t-1} - \mu_{t-1})(x_{t-1} - \mu_{t-1})^T] = \Sigma_{t-1}$,
$E[A(x_{t-1} - \mu_{t-1})(x_{t-1} - \mu_{t-1})^T A^T] = A\Sigma_{t-1}A^T$

In the second and third term as $(x_{t-1} - \mu_{t-1})$ and $\mathcal{E}$ are uncorrelated, both will turn out to be zero. The last term is $E[\mathcal{E}\mathcal{E}^T]$, which is the covariance matrix for the normal

distribution.

Thus,

$$\Sigma_t = A\Sigma_{t-1}A^T + R \tag{4}$$

The equation shows that control inputs (u_t), do not have any effects on covariance, as they are deterministic.

2.2 Algorithm

1. Kalman_Filter ($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2. $\overline{\mu}_t = A_t\mu_{t-1} + B_t u_t$
3. $\overline{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$
4. $K_t = \overline{\Sigma}_t C_t^T (C_t\overline{\Sigma}_t C_t^T + Q_t)^{-1}$
5. $\mu_t = \overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t)$
6. $\Sigma_t = (I - K_t C_t)\overline{\Sigma}_t$
7. return μ_t, Σ_t

2.3 Understanding and Interpretation

As input, Kalman Filter takes the belief at the previous time step $bel(x_{t-1})$ (represented by μ_{t-1} and Σ_{t-1}). To update the belief, the filter also needs the control u_t and the observation z_t at the current time step t. The algorithm outputs the belief at the current time step t (represented by μ_t and Σ_t).

The algorithm first (line 2 & 3) calculates the predicted belief $\overline{bel}(x_t)$ at the current time step t (represented by $\overline{\mu}_t$ and $\overline{\Sigma}_t$) by incorporating only the control u_t. After that, the algorithm (lines 4 to 6) incorporates the measurement z_t to transform predicted belief $\overline{bel}(x_t)$ to desired belief $bel(x_t)$.

To explain in more detail, the predicted mean ($\overline{\mu}_t$) in line 2 is calculated using function (3), the predicted covariance ($\overline{\Sigma}_t$) in line 3 is calculated using function (4). In line 4, K_t is the Kalman gain, which specifies the amount of change to introduce to a state estimate by the measurement. In line 5, desired mean (μ_t) is calculated using the predicted mean ($\overline{\mu}_t$), the measurement (z_t), and the Kalman gain (K_t). In line 6, desired covariance (Σ_t) is

calculated using predicted covariance ($\overline{\Sigma}_t$) and Kalman gain (K_t). Figure 1 illustrates the algorithm.

Insight into Kalman Gain

K_t tries to push the mean of the estimated state towards the observation. If the measurement noise is too high compared to the predicted covariance ($\overline{\Sigma}_t$), Kalman gain K will be very small. This means that the incorporation of the measurement (z_t) into the new state estimate will be very small. Alternatively, if the measurement noise is very small compared to the predicted covariance, the value of the gain (K) will be very large. Kalman Gain will ensure that the data with less noise will contribute more to the desired belief. Also, as you can see in figure 1(f), introducing K_t results in decreasing the uncertainty of the estimated state as μ_t shifts a little in direction of z_t and Σ_t decreases as well.

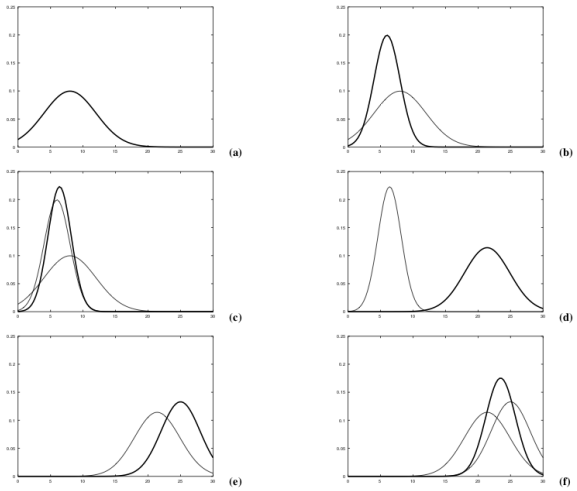


Figure 1: Illustration of Kalman Filter [1]

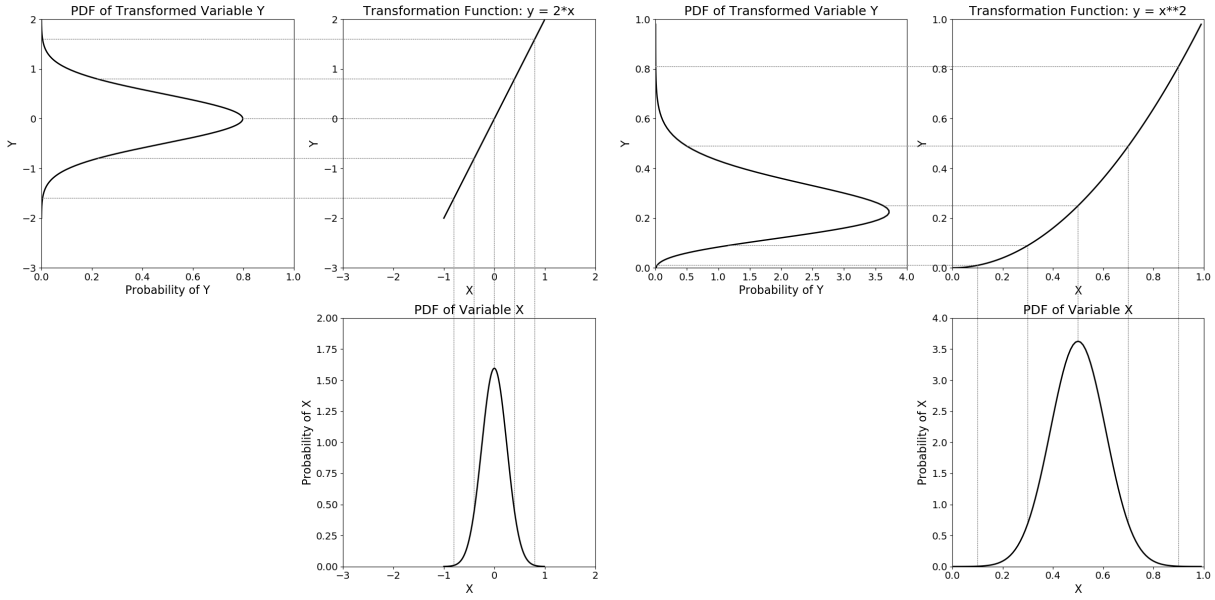
(a) initial belief, (b) a measurement (in bold) with the associated uncertainty, (c) belief after integrating the measurement into the belief using the Kalman filter algorithm, (d) belief after motion to the right (which introduces uncertainty), (e) a new measurement with associated uncertainty, and (f) the resulting belief. Figure and description from Probabilistic Robotics book [1].

3 Extended Kalman Filter

Kalman filter is used mainly for linear transforms. To use the algorithm for non-linear transform some changes needs to be made. Extended Kalman Filter is a non-linear version of Kalman Filter. Let's assume we have a system as below:

$$x_t = g(u_t, x_{t-1}) + \mathcal{E}_t, \quad \mathcal{E} \sim \mathcal{N}(0, Q)$$
$$z_t = h(x_t) + \delta_t, \quad \delta \sim \mathcal{N}(0, R)$$

In Kalman Filter, state transition function g and observation model function h are linear. The Gaussian assumption won't be broken for linear transformations as shown in figure 2(a). However, Gaussian distribution applied with non-linear transformation is not a Gaussian distribution as shown in figure 2(b).



(a) Linear Transformation: $y = 2 * x$

(b) Nonlinear Transformation: $y = x^2$

Figure 2: Transformation on Gaussian Distribution

In order to extend the applications of Kalman Filter to non-linear transform, first order Taylor series expansion is used. For any function $f(x)$, first-order Taylor series expansion is:

$$f(x) = f(\bar{x}) + f'(x)(x - \bar{x}).$$

Thus, Extended Kalman filter works very well when the function is locally linear (so that it can be approximated using only first order Taylor expansion). The Extended Kalman Filter would not give good results when functions g or h are not locally linear.

Now, in case of non-linear approximation of state x_t, $g(u_t, x_{t-1})$ can be approximated as follows:

$$g(u_t, x_{t-1}) = g(u_t, \mu_{t-1}) + g'(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1})$$

where $g'(u_t, \mu_{t-1}) = G_t$ is the Jacobian

As we would like to approximate the function g with most likely value, the mean of Gaussian distribution (μ_{t-1}) is used.

3.1 Algorithm

1. Extended_Kalman_Filter $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:
2. $\overline{\mu}_t = g(u_t, \mu_{t-1})$
3. $\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
4. $K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}$
5. $\mu_t = \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t))$
6. $\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$
7. return μ_t, Σ_t

The algorithm is very similar to the Kalman filter. The difference is that the linear predictions are replaced with their non-linear counterparts (state transition becomes non-linear function g), and that the linear system matrices are replaced with Jacobians (A_t becomes G_t; C_t becomes H_t). The resulting belief $(bel(x_t))$ also differs from Kalman's resulting belief in the sense that Kalman's belief is exact, while Extended Kalman's belief is an approximation.

3.2 Example

This is an example of using Extended Kalman Filter on system having non-linear differentiable observation transition function. Let us assume to have a radar system to observe the position of a plane flying at height h with a constant velocity. We have x to be the horizontal distance; h to be the vertical distance of plane; z to be the straight line distance

as shown in the figure 3 below.

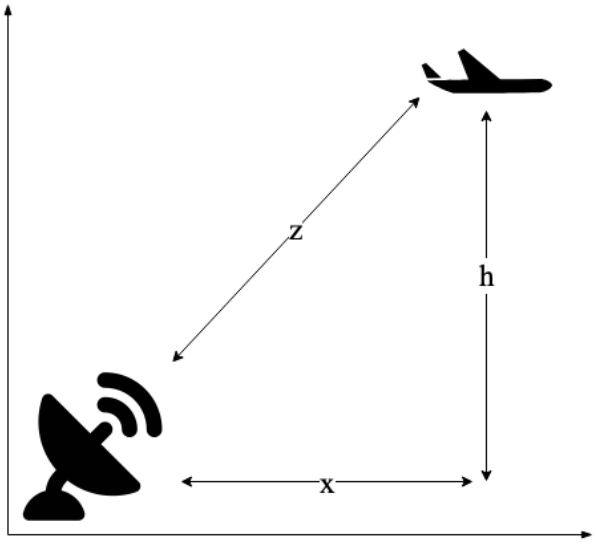


Figure 3: Radar tracking plane

The state of the plane is

$$x_t = [x, v, h]^T$$

where x is the horizontal distance, v is the velocity, and h is height. Then we have the state transition

$$x_t = g(x_{t-1})$$

$$\begin{bmatrix} x_{t,1} \\ x_{t,2} \\ x_{t,3} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1,1} \\ x_{t-1,2} \\ x_{t-1,3} \end{bmatrix}$$

The observation function h of radar observed value z is

$$z_t = h(x_t) = \sqrt{x_{t,3}^2 + x_{t,1}^2}$$

Therefore the observation transition matrix is the Jacobian of h

$$\begin{aligned} H_t &= \begin{bmatrix} \frac{\partial h}{\partial x_{t,1}} & \frac{\partial h}{\partial x_{t,2}} & \frac{\partial h}{\partial x_{t,3}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{2x_{t,1}}{2\sqrt{x_{t,3}^2 + x_{t,1}^2}} & 0 & \frac{2x_{t,3}}{2\sqrt{x_{t,3}^2 + x_{t,1}^2}} \end{bmatrix} \end{aligned}$$

After using the Extended Kalman Filter algorithm for state estimation, the 4, 5 shows the graphs for actual measured range and estimation errors for x,v and h at different time steps.

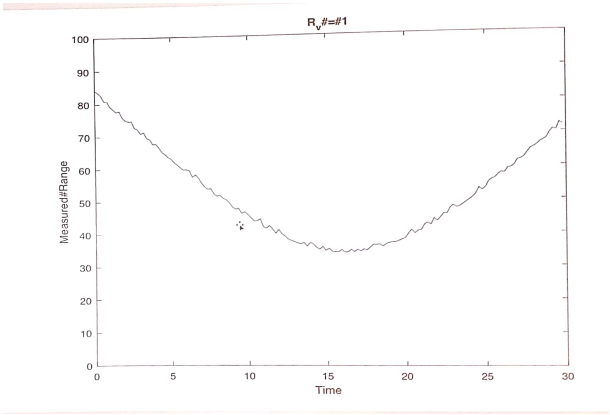
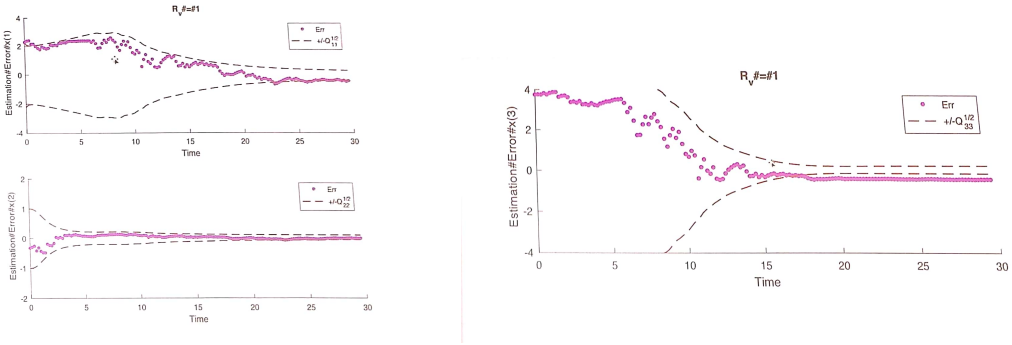


Figure 4: Measured Range (z)



(a) Estimation errors for x and v

(b) Estimation error for h

Figure 5: Estimation Errors

As you can see from the figures after the measured range reaches the minimum at time t=15 (Figure 4), the estimation errors for x,v and h (Figure 5) decreases and remains near zero throughout after that, which is one of the EKF features.

4 Gaussian Sum Filter

Previously discussed filters are unimodal distribution, i.e. they fail in situations where more than one peak exist. An example of such a situation is when a robot is at a crossing and can turn left or right, two hypothesis exist and a unimodal Gaussian fails. In such situations, a Gaussian sum filter is used.

In a Gaussian sum filter, a collection of Gaussian distributions in which each Gaussian has a different mean and variance is used such that each Gaussian in the collection is updated and then the weighted sum of these Gaussian distributions is taken (each Gaussian's degree of contribution to the state's probability distribution is determined by its weight). The collection of Gaussian distributions and the resulting distribution can be viewed in one dimension as given in figure 6 below.

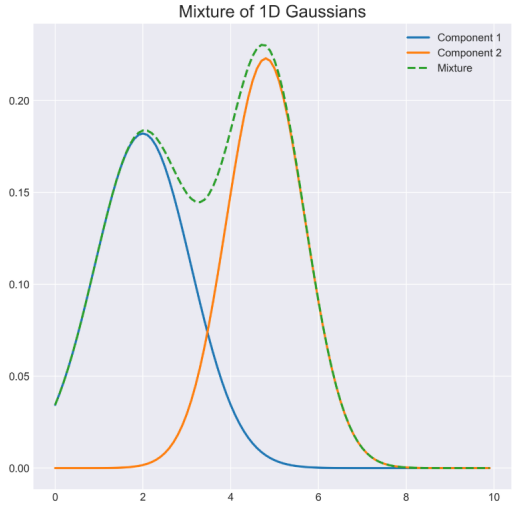


Figure 6: Gaussian Sum Distribution (1D) [2]

The probability of being in state x_t is represented as:

$$p(x_t) = \Sigma_{i=1}^{NG} W_t^i N(\overline{x}_t^i, Q^i)$$

for each weight, b is computed. b represents how far away the actual measurement is from the expected measurement for a given Gaussian

$$b_t^i = N(z_t - h(\overline{\mu}_t^i, \mathcal{E}^i)$$

After that the weight is updated iteratively. Initially, there will be many weights, however, after few iterations of weight updates we end up with a much smaller number

of weights (other weights will become smaller and eventually vanish). The weight is updated at each time step as follows:

$$W_{t+1}^{i} = \frac{W_t^i . b_t^i}{\Sigma_{j=1}^{NG} W_t^j . b_t^j}$$

Additional Material

For additional material, Please refer to:

- Stefan Schaal's Lecture on Kalman Filtering: `https://www.youtube.com/watch?v=XzZt-2D5zS4`
- Cyrill Stachniss's lecture and slides on EKF: `http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam03-ekf.pdf` ***and*** `https://www.youtube.com/watch?v=DE6Jn2cB4J4&t=2270s`
- Kalman and Extended Kalman Filters: Concept, Derivation and Properties, at: `http://users.isr.ist.utl.pt/~mir/pub/kalman.pdf`

References

[1] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[2] Angus Turner. Gaussian mixture models in pytorch. `https://angusturner.github.io/generative_models/2017/11/03/pytorch-gaussian-mixture-model.html`. Accessed: 2019-09-22.