Statistical Techniques in Robotics (16-831, F10) Lectures #20, #21 (Oct. 28, Nov 2)

Kalman Filtering

Lecturer: Drew Bagnell *Scribe: Zachary Lamb, Jackie Libby*[1]

1 Introduction to Gaussian Filters

Gaussian filters embody the first practical implementations of Bayesian filtering for continuous spaces. Despite a number of deficiencies, they are by far the most popular of techniques implemented to date.

All Gaussians share the same basic idea that beliefs can be represented by multivariate normal distributions. This representation has important consequences. Since Gaussians are unimodal they possess only a single maximum; this can be readily applied to robot localiztion, for example, where the robot posterior is focused around the true state with some amount of uncertainty. These posteriors are of little use in problems that can have multiple hypotheses, such as global estimation problems.

2 The Kalman Filter

2.1 Introduction

The Kalman filter was invented as a technique for filtering and prediction in *linear Gaussian systems*. It implements belief computations (Bayesian filtering) for continuous spaces and is not applicable to discrete/hybrid state spaces.

Kalman filters represent beliefs by the moment parameterization, where the belief at time t is represented by a mean μ_t and covariance Σ_t. All computed posteriors are *Gaussian* if the following properties hold (Markov assumption must hold as well). These three items ensure that all posteriors are always Gaussian for any time t.

1. The state transition probability $p(x_t|u_t, x_{t-1})$ must be linear in their arguments and with added Gaussian noise. This probability therefore assumes linear dynamics and in this form is called *linear Gaussian*. It is represented by the following equation :

$$x_{t+1} = Ax_t + \varepsilon, \qquad \text{where } \varepsilon \sim \mathcal{N}(0, Q) \tag{2.1}$$

2. The measurement probability $p(y_t|x_t)$ must also be linear with its arguments and with added Gaussian noise. This probability is represented by the following probability:

$$y_{t+1} = Cx_t + \delta, \qquad \text{where } \varepsilon \sim \mathcal{N}(0, R) \tag{2.2}$$

3. The initial belief must be normally distributed with some mean μ_0 and covariance Σ_0.

[1]Some content adapted from previous scribs: Hyunggi Cho and Alberto Rodriguez

Combining (2.1) and (2.2) together, we can show how the state and measurment are thought to come from a normal distribution with a mean vector μ and covariance matrix Σ:

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ C\mu_x \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}\right) \tag{2.3}$$

We can further equate each of the terms in the covariance matrix in terms of the top left term Σ_{xx}.

Since $\mu_y = C\mu_x$, then we know,

$$\Sigma_{yy} = R + C\Sigma_{xx}C^T \tag{2.4}$$

Therefore, all that is left is to determine Σ_{xy} since $\Sigma_{yx} = \Sigma_{xy}^T$.

$$\Sigma_{xy} = E\begin{bmatrix} x & y^T \end{bmatrix}$$

Since $y = Cx + \delta$, where δ has a mean of 0 and is independent of all other observations, we have

$$\begin{aligned} \Sigma_{xy} &= E\left[x(Cx+\delta)^T\right] \\ &= E\left[xx^TC^T\right] + E\left[x\delta^T\right] \\ &= E\left[xx^T\right]C^T + E\left[x\right]E\left[\delta^T\right] \\ &= \Sigma_{xx}C^T + 0 \\ \Rightarrow \Sigma_{xy} &= \Sigma_{xx}C^T \end{aligned} \tag{2.5}$$

2.2 The Kalman Filter Algorithm

The Kalman filter algorithm is depicted in the table below. The input to the filter is the belief represented by the mean μ_t and covariance Σ_t at the previous timestep.

Algorithm 1 Basic Kalman Filter Algorithm

for all t **do**

$\mu_{t+1}^- = A\mu_t$

$\Sigma_{t+1}^- = A\Sigma_{t+1}A^T + Q$

$\mu_{t+1} = \mu_{t+1}^- + \Sigma_{t+1}^- C^T(C\Sigma_{t+1}^- C^T + R)^{-1}(y_t - C\mu_t^-)$

$\Sigma_{t+1} = \Sigma_{t+1}^- - \Sigma_{t+1}^- C^T(C\Sigma_{t+1}^- C^T + R)^{-1}C\Sigma_{t+1}^-$

end for

The first two lines of the algorithm calculate predicted beliefs represented by μ^- and Σ^- at the new time $(t+1)$ without incorporating the measurement y_t. The mean is then updated using the deterministic version of the state transition function shown in (2.1) using the variables μ_t^- and Σ_t^-.

In the final two lines, the predicted belief is updated into the desired belief by incorporating the measurement y_t. The quantity $\Sigma_{t+1}^- C^T(C\Sigma_{t+1}^- C^T + R)^{-1}$ is known as the *Kalman Gain*. It details the degree to which each measurement is incorporated into the new state estimate. The third line manipulates the predicted mean in proportion to the Kalman Gain with the quantity $(y_t - C\mu_t^-)$, known as the *innovation*.

This algorithm is computationally quite efficient, with the only complexities arising from the matrix inversions in lines 3 and 4 ($O(d^2)$) and the multiplications in line 4 ($O(n^2)$).

Taking (2.4) and (2.5) from above, we can evaluate the update rules in terms of the mean vector and covariance matrix.

$$\mu_{x|y} = \mu_x + \Sigma_{xx}C^T(R + C\Sigma_{xx}CT)^{-1}(y - C\mu_x) \tag{2.6}$$
$$\Sigma_{x|y} = \Sigma_{xx}C^T(R + C\Sigma_{xx}C^T)^{-1}C\Sigma_{xx} \tag{2.7}$$
$$\Rightarrow \mu_{x|y} = \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - C\mu_x)$$
$$\Rightarrow \Sigma_{x|y} = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}$$

Where $\Sigma_{xy}\Sigma_{yy}^{-1}$ is the *Kalman Gain* and $(y - C\mu_x)$ is the *innovation*.

2.3 Kalman Filter Illustration

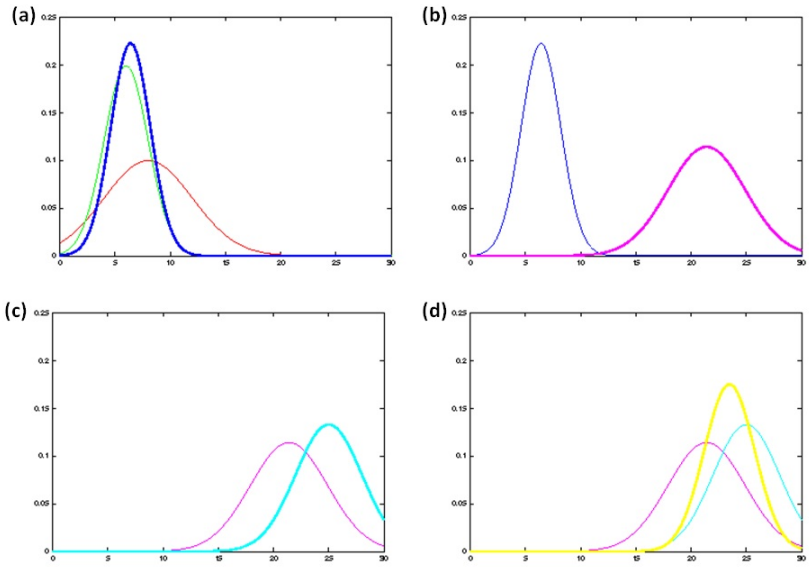

Figure 2.1: Kalman filtering example for 1D localization

Figure 2.1 above illustrates the Kalman filter algorithm for simple 1D localization. Supposing the robot moves in the horizontal direction, the prior of robot's position is shown as the red line in (a). Polling its sensors gives the distribution shown by the green line in (a), and the blue line is a result of the final two lines of the Kalman filter algorithm shown above. In (b) the robot has moved to a new position illustrated by the magenta line. It polls its sensor that report back the position given by the cyan line in (c), and the update from the algorithm gives the posterior given by the yellow line in (d).

The Kalman filter alternates the *sensor update* step (lines 3-4), which incorporates sensor data into

the present belief, with the *motion update* step (lines 1-2), where the belief is predicted according to the action taken by the robot.

3 The Extended Kalman Filter

3.1 Why should we linearize?

The assumptions of linearity for both the measurement and state transition are essential for the correctness of the Kalman filter. The observation that any linear transformation of a Gaussian random variable yields another Gaussian is important in the derivation of the Kalman filter algorithm, so what happens if state transitions and measurements are no longer linear?

Figure 3.1 shows the effect of a nonlinear transformation of a Gaussian random variable. The lower right graph shows the random variable distributed about its mean. It is passed through the nonlinear function represented by the upper right graph. The resulting distribution (upper left) is no longer gaussian in shape, rendering Kalman filtering useless over the domain of the nonlinear function.

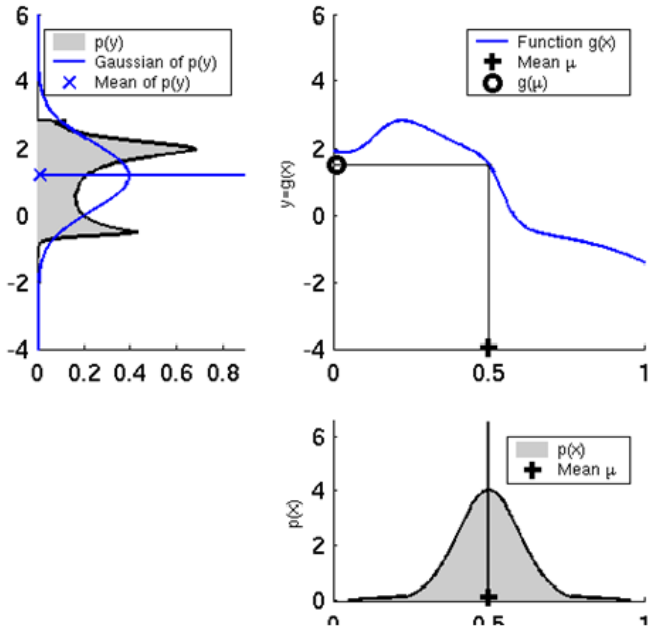

Figure 3.1: Effect of nonlinear transformation of Gaussian random variable.

3.2 The Extended Kalman Filter

Unfortunately, state transitions and measurements are rarely linear in practice. Thus, we would like to be able to model non-linear transformations with our filter. The *Extended Kalman Filter* or *EKF* relaxes the linearity assumption by assuming that the state transition and measurement

probabilities are governed *nonlinear* functions f and g, such that

$$x_{t+1} = f(x_t) + \varepsilon_{t+1}, \qquad \text{where } \varepsilon \sim \mathcal{N}(0, Q) \tag{3.1}$$

$$y_{t+1} = g(x_{t+1}) + \delta_{t+1}, \qquad \text{where } \delta \sim \mathcal{N}(0, R) \tag{3.2}$$

Notice that the function f replaces the matrix A and the function g replaces C. For arbitrary functions f and g, the belief is no longer Gaussian. To combat this, the EKF calculates a Gaussian approximation to the true belief. This is shown in figure 3.2

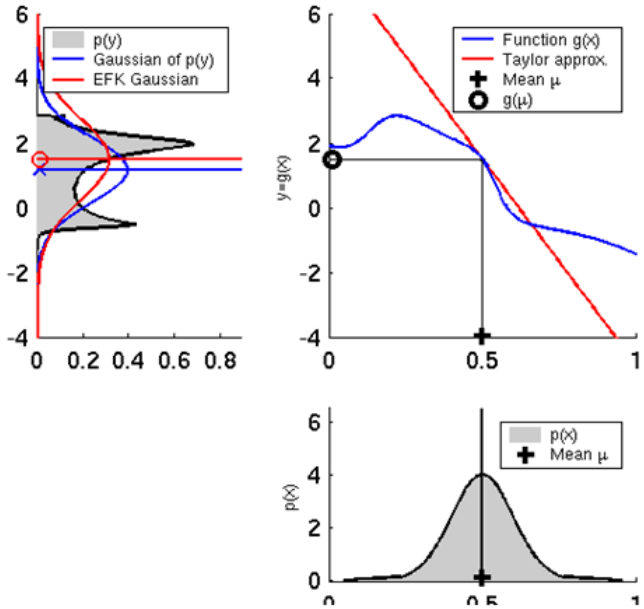

Figure 3.2: Illustration of EKF

The dashed line on the upper left graph represents the Gaussian approximation (*Linearization*) to the density of the random variable y. Because it represents belief at some time t by some mean μ_t and covariance Σ_t, it still keeps from the original Kalman filter the gaussian belief distribution, but this belief is now an approximation. The goal of the EKF is therefore to efficiently estimate the mean and covariance of a posterior.

Linearization approximates the nonlinear function f and g at the mean of the Gaussian. Projecting it through the linear function shown in the upper right graph of 3.2 a linear approximation is shown in red of the upper left graph.

3.3 Linearization via the Taylor Series Expansion

One way to perform this linearization is to use the Taylor expansion. In this case we approximate the transformations using the first order Taylor expansion evaluated at the mean estimate of x at

time t, μ_t:

$$x_{t+1} \approx f(\mu_t) + \frac{\partial f}{\partial x}(\mu_t)(x_t - \mu_t) + \varepsilon \tag{3.3}$$

$$y_{t+1} \approx g(\mu^-_{t+1}) + \frac{\partial g}{\partial x}(\mu^-_{t+1})(x_{t+1} - \mu^-_{t+1}) + \delta \tag{3.4}$$

The first order derivative terms $\frac{\partial f}{\partial x}(\mu_t)$ and $\frac{\partial g}{\partial x}(\mu^-_{t+1})$, also referred to as the Jacobian, is used here as the linear transformation, while the $f(\mu_t)$ term just serves to shift the mean of the transformation.

3.4 New Update Rules

Using the non-linear transformation above, we get the following new update equations for our mean and variance estimates for the motion model:

$$\mu^-_{t+1} = f(\mu^+_t) \tag{3.5}$$

$$\Sigma^-_{t+1} = J_f \Sigma^+_t J_f^T + Q \tag{3.6}$$

And for the sensor model:

$$\mu_{t+1} = \mu^-_{t+1} + \Sigma^-_{t+1} J_g^T (J_g \Sigma^-_{t+1} J_g^T + R)^{-1} [y_{t+1} - g(\mu^-_{t+1})] \tag{3.7}$$

$$\Sigma_{t+1} = \Sigma^-_{t+1} - \Sigma^-_{t+1} J_g^T (J_g \Sigma^-_{t+1} J_g^T + R)^{-1} J_g \Sigma^-_{t+1} \tag{3.8}$$

where $J_f = \frac{\partial f}{\partial x}(\mu^+_t)$ and $J_g = \frac{\partial g}{\partial x}(\mu^-_{t+1})$. These lines replace lines 3 and 4 in the Kalman filter algorithm presented above.

3.5 Problems with Extended Kalman Filtering

- Taylor expansion is a poor approximation of most non-linear functions. The goodness of the linearization depends on two factors: the degree of uncertainty and the amount of local nonlinearity in the functions being approximated.
- Differentiation of the functions f and g can be slightly bothersome, both for you, personally, because you might have forgotten calculus, and because the function might not be continuous across the range in which the linearization is used. An example of a discontinuous function is the hinge loss function for SVM's. Furthermore, for SVM's, you will get driven towards this discontinuity. (Why? Someone figure this out and explain it to me.)
- If the linearization is a poor approximation, the the confidence ellipse will not encapsulate the correct answer unless you make it very large. For this reason, the ellipses are often overconfident. An overconfident filter can cause divergence, because it starts rejecting the measurements it is given. It prances off into a state of egotistical oblivion, giving no respect to the poor measurements that are just trying to help it out.
- EKFs are notoriously bad at handling multiple distinct hypotheses.

Extended Kalman Filters are only as good as their approximation about the mean of the estimate. It is important to keep the uncertainty of the state estimate small when applying EKFs to avoid negative effects from the nonlinearities.

3.6 Example: Non-Linear Regression

In this example we want to use an EKF for a non-linear regression problem. In this case our state vector is a set of weights w and we wish to estimate the mean and variance μ_w, Σ_w with our EKF, which we will just refer to as μ and Σ from here on.

Table 1: Comparing a general Kalman Filter with a Bayes Linear Regression Example

KF	BLR	Difference
x	w	
A	I	No process model for how weights are changing over time
C_t	x^T	Different linearlization at every time step.
Q	0	Growing uncertainty with time.
Y	y	$y = w^T x + \epsilon$
Σ_0	Σ_w	prior
μ_0	μ_w	prior

For our observation model we want to use the non-linear sigmoid function $\sigma(x) = \frac{1}{1+\exp(x)}$:

$$y_t = \sigma(w^T x_t) + \varepsilon, \qquad \text{where } \varepsilon \sim \mathcal{N}(0, \gamma^2) \tag{3.9}$$

Note that our output in this case is a single value, so our Jacobian will be a vector. We can find the Jacobian J_σ using the chain rule and the fact that $\frac{\partial \sigma}{\partial x} = \sigma(x)(1-\sigma(x))$. Here we actually want to find the derivative of $g(w) = \sigma(w^T x)$, so we use the chain rule with $z = w^T x$.

$$\frac{\partial g}{\partial w} = \frac{\partial g}{\partial z}\frac{\partial z}{\partial w} \tag{3.10}$$
$$= \sigma(w^T x)(1 - \sigma(w^T x))x^T \tag{3.11}$$
$$J_\sigma = \frac{\partial g}{\partial w}(\mu) \tag{3.12}$$
$$= \sigma(\mu^T x)(1 - \sigma(\mu^T x))x^T \tag{3.13}$$

This gives update equations:

$$\mu_{t+1} = \mu_t^+ + \Sigma_t^+ J_\sigma^T (J_\sigma \Sigma_t^+ J_\sigma^T + \gamma^2)^{-1}[y_{t+1} - \sigma(\mu_t^{+T} x)] \tag{3.14}$$
$$\Sigma_{t+1} = \Sigma_t^+ - \Sigma_t^+ J_\sigma^T (J_\sigma \Sigma_t^+ J_\sigma^T + \gamma^2)^{-1} J_\sigma \Sigma_t^+ \tag{3.15}$$

For this non-linear regression example, there is no motion step, so the update step just takes the output of the update step from the last round. In other words, $\mu_{t+1}^- = \mu_t^+$ and $\Sigma_{t+1}^- = \Sigma_t^+$. Alternatively, we could add in a pseudo motion model to represent growing uncertainty with each time step:

$$\mu_{t+1}^- = \mu_t \tag{3.16}$$
$$\Sigma_{t+1}^- = \Sigma_t + \lambda I \tag{3.17}$$

where λ is forgetting factor and I is an identity matrix.

Also, since we use the sigmoid function for this example, we use γ^2 instead of σ^2 to denote the variance, to avoid confusion. More generally, this would be the covariance, R, but in this example the sigmoid function outputs a scalar.

4 Other techniques to deal with nonlinear systems

Assuming we have the general nonlinear system given by equations (3.1) and (3.2), we can use other approaches besides the EKF. A *Statistically Linearized Kalman Filter* tries to overcome that limitation by approximating the Jacobian matrix of the system in a broader region centered at the state of the system.

4.1 Montecarlo Kalman Filter

One example of such a filter is the Mantecarlo Kalman Filter (MCKF). This isn't actually a real filter used in practice, but it is presented here for the sake of demonstration, as a precursor to the discussion of the Unscented Kalman Filter next.

Motion Model

Given the equation of the motion model $x_{t+1} = f(x_t) + \epsilon$, μ_t and Σ_t we need to estimate μ^-_{t+1} and Σ^-_{t+1}. For that we draw samples from the prior distribution x^i_t and pass them through $f(x)$. That way, and with the law of large numbers in hand, we can estimate:

$$\mu^-_{x_{t+1}} = \frac{1}{N}\sum_{i=1}^{N} f(x^i_t) \tag{4.1}$$

$$\Sigma^-_{x_{t+1}} = \frac{1}{N}\left[\sum \left(f(x^i_t) - \mu^-_{x_{t+1}}\right) \cdot \left(f(x^i_t) - \mu^-_{x_{t+1}}\right)^T\right] + Q \tag{4.2}$$

NOTE: Q is additive noise, uncorrelated with x_t.

Observation Model

Given the equation of the observation model $y_{t+1} = g(x_{t+1}) + \delta$ the update rules are the same as before:

$$\mu^+_{x_{t+1}} = \mu^-_{x_{t+1}} + \Sigma_{XY}\Sigma^{-1}_{YY}\left(y_t - \mu_y\right) \tag{4.3}$$

$$\Sigma^+_{x_{t+1}} = \Sigma^-_{t+1} - \Sigma_{XY}\Sigma^{-1}_{YY}\Sigma_{YX} \tag{4.4}$$

But now we approximate μ_y, Σ_{YY} and Σ_{XY} as:

$$\mu_y = \frac{1}{N}\sum_{i=1}^{N} g(x_t^i) \tag{4.5}$$

$$\Sigma_{YY} = \frac{1}{N}\left[\sum_{i=1}^{N}\left(g(x_t^i)-\mu_y\right)\cdot\left(g(x_t^i)-\mu_y\right)^T\right] + R \tag{4.6}$$

$$\Sigma_{XY} = \frac{1}{N}\left[\sum_{i=1}^{N}\left(x_t^i-\mu_{x_t}\right)\cdot\left(g(x_t^i)-\mu_y\right)^T\right] \tag{4.7}$$

In the above equations, we can choose to replace $\mu^-_{x_{t+1}}$ with $f(\mu_{x_t})$ and μ_y with $g(\mu^-_{x_{t+1}})$. In other words, we take the mean of our samples and pass them through the nonlinear functions f and g, instead of passing every sample through the functions. If the samples were perfectly gaussian, these two methods would produce the same result.

Note that last year's lectures used x^i_{t+1} instead of x^i_t in these update equations. I believe it is more correct to use x^i_t. There can be two cases: case 1 is that we're doing an update step on the same timestep as a motion step, (which is what the overall notation is suggesting with the +'s and -'s). In this case, we would not have x^i_{t+1}'s to sample from, because we are still at time step t. Case 2 is that we're doing an update step at the next timestep, and that we are treating the motion and update steps as asynchronous events that happen independently from each other. In this case, we would look at the update step separately from the motion step, and we could leave the time subscript out altogether because an update step happens at one moment in time.

Comparing the Montecarlo Kalman Filter (MCKF) with the standard Particle Filter (PF):

- Good things about the MCKF
  - The MCKF forces some smoothing on the uncertainty that simplifies the process to get a solution.
  - The MCKF doesn't need as many samples as the PF. The number of samples is on the order of $O(d^2)$. To estimate n parameters, we need $O(n)$ particles. Here we have d^2 parameters.
- Bad things about the MCKF
  - The MCKF will always be unimodal, while a the PF can perfectly maintain several modes in the estimation of the distribution.

4.2 Sigma-Point Filter

A Sigma-Point Filter has exactly the same formulation as a Montecarlo Kalman Filter but it draws samples in a deterministic way from interesting locations. Suppose Δ_i are the eigenvectors of the covariance matrix Σ_{x_t}. Then we sample the points as:

$$\mu_{x_t} \pm \lambda_i \Delta_i \quad i = 1 \ldots d$$

where d is the dimension of x_t and λ_i is proportional to the eigenvalue corresponding to eigenvector Δ_i. In other words, we pick one point along each eigenvector direction. We also sometimes pick the last point as the mean itself, so the number of points is $2d + 1$. Then, the update rule for the motion model becomes:

$$\mu^-_{x_{t+1}} = w_i \sum_{i=1}^{N} f(x_t^i) \tag{4.8}$$

$$\Sigma^-_{x_{t+1}} = w_i \left[\sum \left(f(x_t^i) - \mu^-_{x_{t+1}} \right) \cdot \left(f(x_t^i) - \mu^-_{x_{t+1}} \right)^T \right] + Q \tag{4.9}$$

This assumes the weights sum to 1. There are different versions of Sigma-Point filters and they all differ on how weights w_i are selected. All versions choose weights so that the method behaves perfectly for a gaussian model (linear dynamics) and then optimize the weights for different criteria. Different versions include Unscented Kalman Filter, Central Difference Kalman Filter, ...

The computational cost of Sigma-Point type filters is $O(d^3)$ for finding the eigenvectors (usually implemented by the SVD decomposition) plus $2d + 1$ evaluations of the motion model $f(x)$.

Evaluating the Sigma-Point Filter:

- Good things
  - It needs less particles to run, and hence reduces the number of motion model evaluations, which can be costly.
  - It only needs to be implemented once because the algorithm is generic, for any choice of your model, (f, g).
  - You don't have pretend like you know calculus, because there are no Jacobians.
- Bad things
  - It can perform really bad if facing an adversarial problem, because it samples the space in a deterministic way.
  - As the dimension goes up, the center weight can become negative. (One fix is to leave the center value out for the variance calculations.)
  - The eigenvalue decomposition need to be done at every time step, which can be costly. Also, SVD is not always deterministic, so you count bound the computation time. This makes it difficult to run online.

The Unscented Kalman Filter (UKF) is a type of Sigma-Point Filter, for specific choices of λ_i's and w_i's. Fig. 4.1 gives a pictorial representation of the UKF:

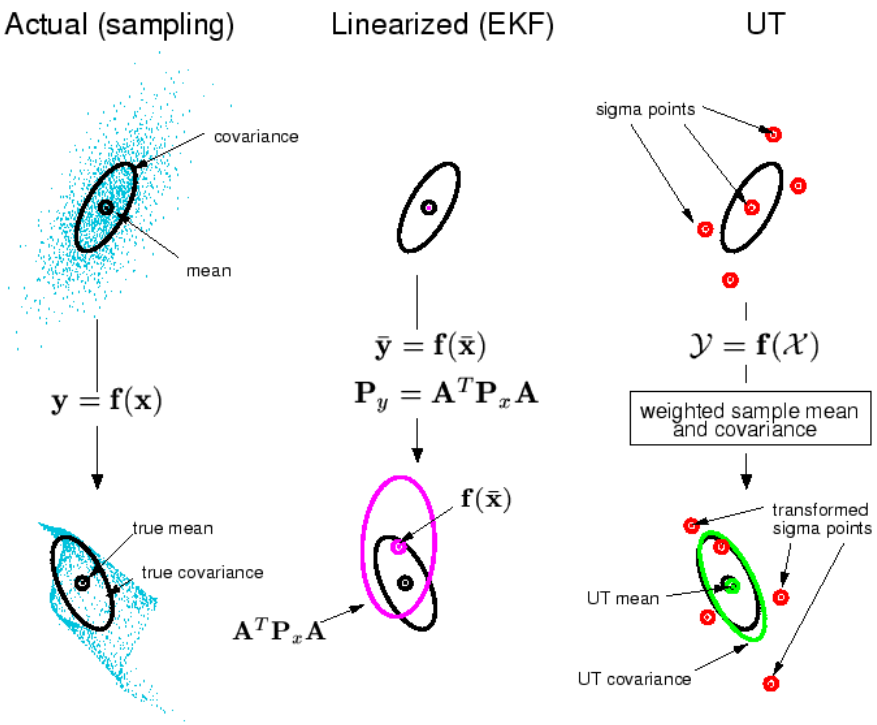

Figure 4.1: A comparison of the actual transformation, and the approximations given by the linear approximation and the unscented transformation.
(Source: http://cslu.cse.ogi.edu/nsel/ukf/node6.html)