Statistical Techniques in Robotics (16-831, F12) Lecture#19 (Wednesday November 7)

Kalman Filtering (part 2)

Lecturer: Drew Bagnell *Scribe: Keheng Zhang* [1]

1 Statistically Linearized Kalman Filters

Recall that last time in the end of the class, we talked about the Extended Kalman Filter and one way to perform its linearization, the Taylor Series Expansion. Assuming we have the general nonlinear system given by the following equations:

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

$$y_{t+1} = g(x_{t+1}) + \delta_{t+1}, \qquad \text{where } \delta \sim \mathcal{N}(0, R) \tag{2}$$

The Taylor Expansion method tries to approximate the transformations using the first order Taylor expansion at the mean of x. We discussed the major flaw with EKF that the Taylor expansion is a poor approximation of most non-linear functions.

In this lecture another approach is given to overcome the limitation of 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. This type of approach also offers the benefit that it does not require continuity or differentiability of the motion and measurement models. Since it is not necessary to compute Jacobian matrices, these methods can offer benefits in terms of computational efficiency as well. However, a downside is that they require the non-linear functions f and g to be provided in closed form.

1.1 Monte-Carlo Kalman Filter

One example of such a filter is the Mante-Carlo Kalman Filter (MCKF). In this method we sample the points around the mean and try to find the best fit (a line, a plane, etc.) over the points and use this approximation to run the Kalman Filter.

This is not 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}. Recall the KF equations for the motion model:

$$\mu^-_{t+1} = A\mu_t \tag{3}$$

$$\Sigma^-_{t+1} = A\Sigma_t A^T + Q \tag{4}$$

[1]Some content adapted from previous scribe by Adam

For now in MCKF we draw samples from the prior distribution x_t^i and pass them through $f(x)$. That way, and with the law of large numbers in hand, we can estimate:

$$\mu_{x_{t+1}}^- = \frac{1}{N}\sum_{i=1}^{N} f(x_t^i) \tag{5}$$

$$\Sigma_{x_{t+1}}^- = \frac{1}{N}\left[\sum \left(f(x_t^i) - \mu_{x_{t+1}}^-\right)\cdot\left(f(x_t^i) - \mu_{x_{t+1}}^-\right)^T\right] + Q \tag{6}$$

NOTE: Q is additive noise, uncorrelated with x_t. If we do not add Q here, we would have to add noise to each sample.

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 in the KF observation model:

$$\mu_{x_{t+1}}^+ = \mu_{x_{t+1}}^- + \Sigma_{XY}\Sigma_{YY}^{-1}(y_t - \mu_y) \tag{7}$$

$$\Sigma_{x_{t+1}}^+ = \Sigma_{t+1}^- - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX} \tag{8}$$

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

$$y_{t+1}^i = g(x_{t+1}^i) \tag{9}$$

$$\mu_y = \frac{1}{N}\sum_{i=1}^{N} y_{t+1}^i \tag{10}$$

$$\Sigma_{YY} = \frac{1}{N}\left[\sum_{i=1}^{N}\left(y_{t+1}^i - \mu_y\right)\cdot\left(y_{t+1}^i - \mu_y\right)^T\right] + R \tag{11}$$

$$\Sigma_{XY} = \frac{1}{N}\left[\sum_{i=1}^{N}\left(x_{t+1}^i - \mu_{x_{t+1}}^-\right)\cdot\left(y_{t+1}^i - \mu_y\right)^T\right] \tag{12}$$

NOTE: In above equations $\mu_y \neq g(\mu_x)$

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 it 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.

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.

1.2 Sigma-Point Filter

Another sampling-based method for dealing with non-linearities is the Sigma-Point Filter. A Sigma-Point Filter has exactly the same formulation as a Monte-Carlo Kalman Filter but it draws samples in a deterministic way from the **mean and simplex**. (With MCKF we usually pick points on principle axis.) Suppose Δ_i are the eigenvectors of the covariance matrix Σ_{x_t}. Then we sample the points as:

$$\mu_{x_t} \pm \lambda_i \Delta_i \qquad i = 1 \dots d$$

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 usually pick the **distance** along the axis as one standard deviation. 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:

$$\mu^-_{x_{t+1}} = w_i \sum_{i=1}^{N} f(x^i_t) \tag{13}$$

$$\Sigma^-_{x_{t+1}} = w_i \left[\sum \left(f(x^i_t) - \mu^-_{x_{t+1}} \right) \cdot \left(f(x^i_t) - \mu^-_{x_{t+1}} \right)^T \right] + Q \tag{14}$$

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, ... Fig. 1 illustrates the linearization applied by UKF.

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.

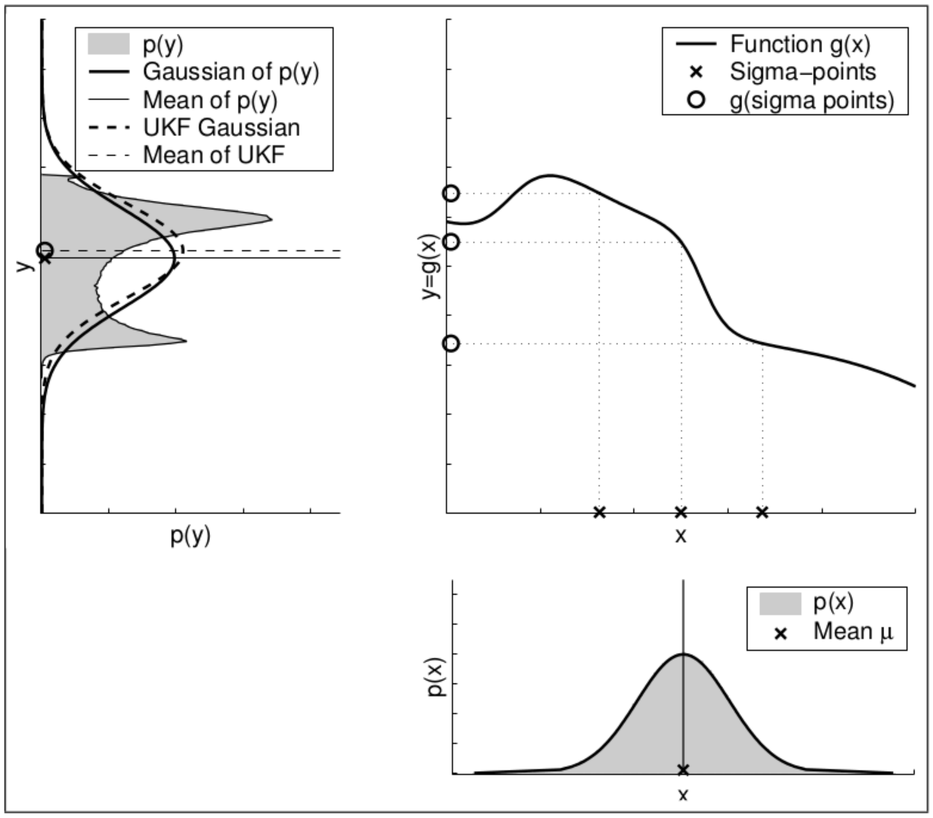

Figure 1: Linearization applied by UKF.

Summary of Sigma-Point Filter

- Compute sigma points and weights
- Transform sigma points through system dynamics
- Reconstruct random variable statistics using weighted sample mean and covariance
- Perform Kalman measurement update

The Unscented Kalman Filter (UKF) is a type of Sigma-Point Filter, for specific choices of λ_i's and w_i's. Fig. 2 gives the comparision between the actual transformation, and the approximations given by the linear approximation and the unscented transformation:

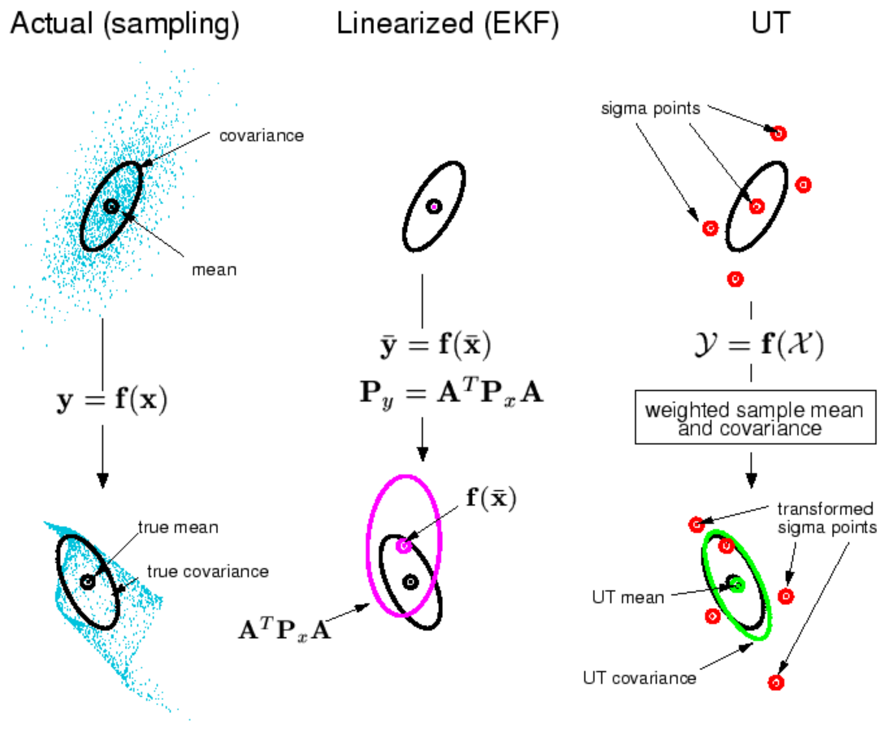
Actual (sampling)
Linearized (EKF)
UT
covariance
mean
$\mathbf{y} = \mathbf{f}(\mathbf{x})$
true mean
true covariance
$\bar{\mathbf{y}} = \mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{P}_y = \mathbf{A}^T\mathbf{P}_x\mathbf{A}$
$\mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{A}^T\mathbf{P}_x\mathbf{A}$
sigma points
$\mathcal{Y} = \mathbf{f}(\mathcal{X})$
weighted sample mean and covariance
transformed sigma points
UT mean
UT covariance

Figure 2: 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)