AA274A: Principles of Robot Autonomy I
Course Notes

Oct 31, 2019

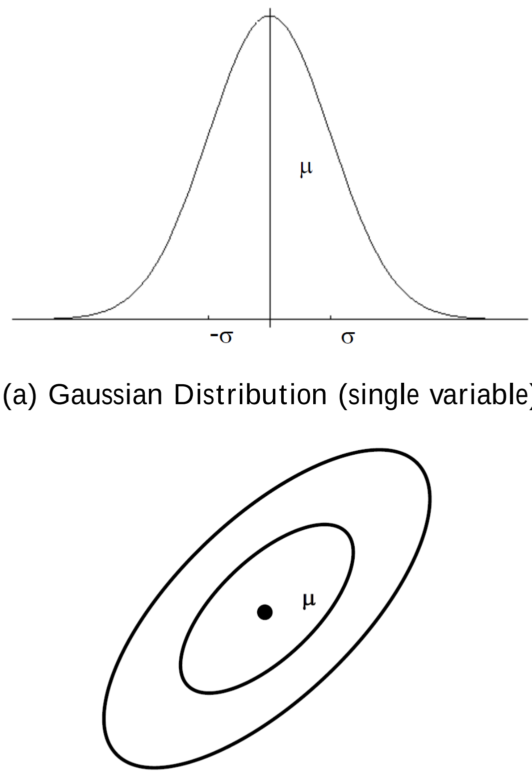
15 Parametric Filters

Last lecture, we discussed the Bayes filter to estimate the state of dynamic systems. However, because the Bayes filter requires iterating over a continuous variable, it is generally impractical due to computational limits. Today, we extend the Bayes filter to a Gaussian dynamical model that forms the basis of robot localization.

Today, we will focus on *parametric* filters that generally begin by restricting the belief of the Bayes filter to a particular distribution. As the name suggests, parameteric filters have a fixed number of *parameters* that describe the robot's belief. The most common and simple distribution to represent robot's belief on state and measurement is a Gaussian distribution. We will discuss why a Gaussian distribution, once again, is a popular choice for robot localization. Next lecture, we will look at *non-parametric* filters, where the robot's belief is represented with discretized "particles".

15.1 Parametric Filters

Parametric filters are a family of state estimators that model the robot's belief state with parametric distributions. As the robot's state evolves, the parameters that fully capture robot's belief distribution are updated based on the latest measurement. Gaussian filters, as the name suggests, use a Gaussian distribution to model the robot's belief state. Why is a Gaussian distribution a parametric distribution? Among many wonderful properties of Gaussian distribution, one is that we only need two parameters, mainly mean μ and variance σ, to describe the robot's entire belief system! As the state evolves, our robot simply needs to update these two parameters (vectors) to retrieve the Gaussian bell curve. There are parametric filters that use other parametric distributions such as the Laplace distribution, but the Gaussian distribution is the most common and has been the most successful in robotics.



(a) Gaussian Distribution (single variable)

(b) Level curves of multivariate Gaussian distribution, $n = 2$.

Figure 1: Visualizations of the multivariate normal (Gaussian) distribution

15.2 Kalman Filter

What is a Kalman Filter? The Kalman filter was invented in the 1950s by Rudolph Emil Kalman as a technique for filtering and prediction in linear systems. The Kalman filter implements belief computation for continuous states.

The key idea behind Kalman filters is that we assume that each belief function (the expected state) follows a multivariate normal distribution (Gaussian distribution). The Kalman filter is simply a result of the Bayesian posterior $P(x_t|y_{1:t}, u_{1:t})$, as we will discuss later. The Kalman filter models beliefs with the moments representation: at time t, the belief is represented by the the mean μ_t and the covariance Σ_t.

Why Gaussian Distribution? While we could in theory choose another distribution, Gaussian distributions are attractive for a number of reasons. First, with Gaussian distributions, the entire belief state of the robot can be represented by two parameters: the mean, μ, and covariance, σ. For a single variable (univariate) Gaussian distribution, the probability distribution can be written as:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{1}$$

We use notation $X \sim \mathcal{N}(x; \mu, \sigma^2)$ to represent that a random variable X comes from a Gaussian probability distribution. Figure 1 shows a Gaussian distribution. Note that once we know two parameters, μ and σ, we can plot the entire distribution.

Robot states and measurements are often more than one-dimensional, i.e., there are more than one unknowns. We can extend the concept of univariate Gaussians to multivariate Gaussians, and its probabilistic distribution is written as:

$$p(x) = |2\pi\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}\right) \tag{2}$$

where $X \sim \mathcal{N}(x; \mu, \Sigma)$. An example Gaussian distribution is shown in Figure 1b: In this simple 2-dimensional case, the mean μ defines the center of the ellipse, and covariance Σ determines length of ellipse's axes. Note, again, how the entire Gaussian distribution can be described just with 2 parameter vectors (μ is $n \times 1$, Σ is $n \times n$).

More importantly, however, modelling the robot's belief with Gaussians is computationally critical as Gaussian distributions have a closed form expression for the maximum-likelihood estimates. This means when we find the most likely solutions given new data, the solution can also be expressed with a Gaussian (and therefore require only two parameter vectors to describe the entire distribution). This is a unique property of Gaussian distributions which is not true for most distributions. In the same vein, one can show

1. If $X \sim \mathcal{N}(\mu, \Sigma)$, then
$$Y = AX + b \sim \mathcal{N}(A\mu + b, A\Sigma A^T).$$

2. If $X_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ for $i = 1, 2$, and X_1, X_2 are independent, then
$$Y = X_1 + X_2 \sim \mathcal{N}(\mu_1 + \mu_2, \Sigma_1 + \Sigma_2).$$

3. The product of Gaussian pdf's is also Gaussian.

Because of the computational advantage and simplicity of Gaussian distributions, Kalman filters model the robot's belief with Gaussian distributions even though most systems are not truly linear Gaussian.

Necessary Assumptions for the Kalman Filter. The Kalman filter is a result of modelling the robot's behavior with Gaussians. Derivation of the Kalman filter requires three assumptions.

1. We model the robot's state with linear Gaussian systems. If the state variables are continuous and the conditional distributions are linear Gaussian, then the model is called a linear dynamical system. At time t, the state x_t is given by

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t, \tag{3}$$

   where x_{t-1} is the previous state, u_t is the most recent control input, and ϵ_t is the independent *process noise* with distribution $V \sim \mathcal{N}(0, R_t)$.

   Consequence: Given a particular x_{t-1}, u_t and assuming that ϵ_t is Gaussian (a very standard assumption for noise), x_t is also Gaussian. Specifically, using the properties of Gaussians given above,

$$x_t \sim \mathcal{N}(A_t x_{t-1} + B_t u_t, R_t).$$

2. As a part of the linear dynamical system, the measurement model is also linear Gaussian. Our measurement z_t at time t given true state x_t is

$$z_t = C_t x_t + \delta_t,$$

   where δ_t is the independent measurement noise with distribution $\mathcal{N}(0, Q_t)$

   Consequence: This ensures the measurement probability is also a Gaussian. Specifically,

$$z_t \sim \mathcal{N}(C_t x_t, Q_t).$$

3. In addition, we model our initial belief function to be Gaussian

$$bel(x_0) = |2\pi\Sigma_0|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x_0 - \mu_0)\Sigma_0^{-1}(x_0 - \mu_0)^T\right).$$

   Consequence: This assumption, in conjunction with the prior two, ensures that all posterior *bel* functions will also be Gaussian distributions. As mentioned before, this assumption is sometimes quite poor; however, it drastically reduces the computational complexity of the problem.

Earlier, we mentioned that Gaussian distributions are closed, i.e., all subsequent belief states derived by Bayes' Rule retain their Gaussian characteristic. Since Gaussian distributions are uniquely determined by the first two moments (that is, the mean and covariance), this further means that our estimator propagates only two vectors (or, in the multivariate case, the covariance matrix) through time rather than a full expression for the pdf.

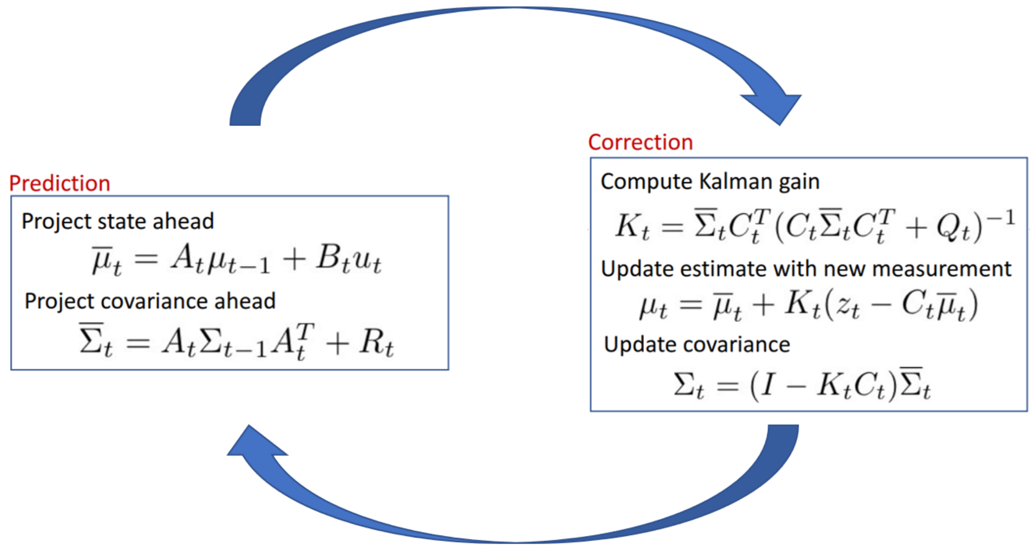


Figure 2: Prediction / correction loop in the Kalman filter

Algorithm. The Kalman filter is a recursive Bayes filter applied to time-varying linear Gaussian systems. Therefore, the Kalman filter can be broken down into two steps: 1) a *prediction* step to propagate the state through time ("recursive"); 2) an *update* step to correct our belief based on the new measurement ("Bayes Filter"). The Kalman filter repeats the prediction and correction steps through time as shown in Fig 2.

Prediction step. In this step, we simulate the ideal system dynamic based on the prior, i.e., estimation of the system dynamic with zero noise. We do this simply by propagating our prior through the equation for Gaussian linear systems.

$$\bar{\mu}_t = A_t\mu_{t\ 1} + B_t u_t \tag{4}$$
$$\bar{\Sigma}_t = A_t\Sigma_{t\ 1}A_t^T + R_t \tag{5}$$

where $\mu_{t\ 1}$ and $\Sigma_{t\ 1}$ are our prior, and $\bar{\mu}_t$ and $\bar{\Sigma}_t$ are the new prior; u_t is our last control input.

Update step. We make inferences to our state based on the newest measurement using Bayes' Filter. It can be shown that Bayes Filter applied to linear Gaussian systems results in another linear Gaussian system, specifically:

$$\mu_t = \bar{\mu}_t + K_t(z_t \quad C_t\bar{\mu}_t) \tag{6}$$
$$\Sigma_t = (I \quad C_tK_t)\bar{\Sigma}_t \tag{7}$$

where K_t, called *Kalman gain*, is $K_t = \bar{\Sigma}_t C_t^T(C_t\bar{\Sigma}_t C_t^T + Q_t)^{\ 1}$. The fact that the update step performed on a Gaussian distribution is also Gaussian reveals the beauty of modeling our belief with Gaussians. The prediction and update steps simply reduce to propagating μ_t and Σ_t for each time step.

K_t is called the Kalman gain as it could be interpreted as analagous to the proportional gain in classical PID controllers. Larger K_t indicates more uncertainty in our prediction than our measurements, and therefore we should apply a larger correction to our belief. Smaller K_t, on the other hand, indicates more uncertainty in our measurement than our prediction, and therefore our correction term is small. The uncertainty decreases after our measurement update, hence the decrease in covariance. Details of Kalman filter implementation is outlined in Algorithm 3.

Data: $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
Result: (μ_t, Σ_t)
$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$ ▷ Predict mean
$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$ ▷ Predict covariance
$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$ ▷ Calculate Kalman gain
$\mu_t = \bar{\mu}_t K_t (z_t - C_t \bar{\mu}_t)$ ▷ Correct mean
$\Sigma_t = (I - C_t K_t) \bar{\Sigma}_t$ ▷ Correct covariance
Return: (μ_t, Σ_t)

Algorithm 1: Kalman Filter

Derivation. We will not go over the full derivation of the Kalman filter; however, it's beneficial to understand the mathematical background behind the Kalman filter.

Prediction step. Recursive Bayes Filter can be written as

$$p(x_t \mid z_{1:t}, u_{1:t}) = \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}). \tag{8}$$

where $\eta = \frac{1}{\int_{x_t} p(z_t|x_t)p(x_t|z_{1:t-1})dx_t}$, which is simply a constant. If we make the Markov Process assumption, our measurement z_t at each step depends only on the previous state x_t, not $z_{1:t-1}$ or $u_{1:t}$. From this conditional independence, we can simplify (8) as

$$p(x_t \mid z_{1:t}, u_{1:t}) = \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}). \tag{9}$$

We rearrange 9 to derive

$$\overline{bel}(x_t) = p(x_t \mid z_{1:t-1}, u_{1:t}) = \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1} u_{1:t-1}) dx_{t-1} \tag{10}$$

which can also be expressed as

$$\overline{bel}(x_t) = \int p(x_t \mid x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1} \tag{11}$$

Since $x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$ with $\epsilon_t \sim \mathcal{N}(\mu_t, R_t)$, we know that $p(x_t \mid x_{t-1}, u_t) \sim \mathcal{N}(A_t x_{t-1} + B_t u_t, R_t)$. Similarly, the linear Gaussian assumptions described above guarantee that $bel(x_{t-1}) \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$. Then (11) can be simplified to

$$\overline{bel}(x_t) = \mathcal{N}(\bar{\mu}_t, \bar{\Sigma}_t) \tag{12}$$

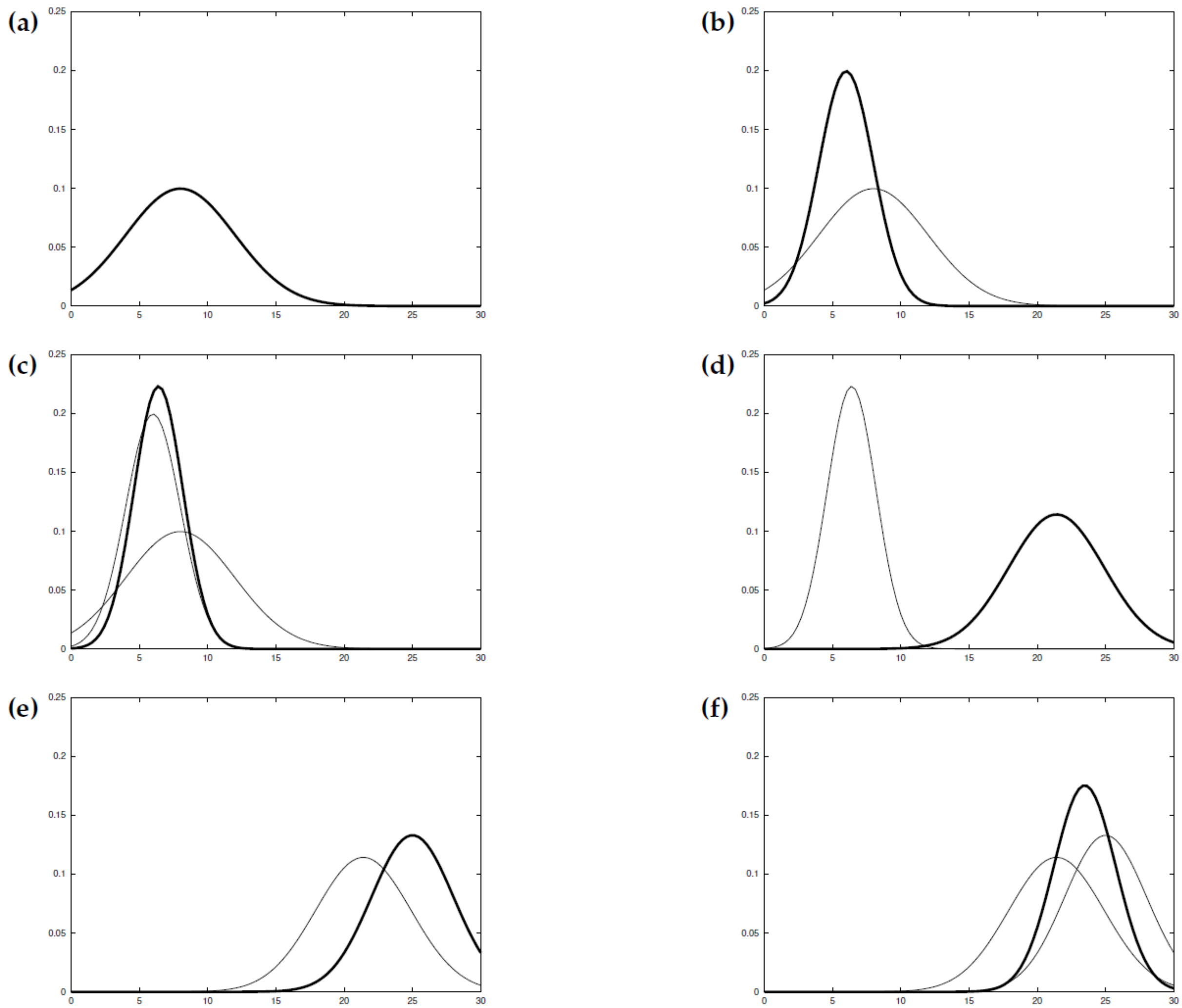


Figure 3: Visualization of the Kalman Filter Algorithm from [TBF05]. Current step is in bold and previous data are in gray: (a) the initial belief distribution, (b) the measurement with uncertainty in bold and initial belief in gray, (c) updated belief in bold with previous belief and measurement in gray, (d) belief after motion to the right with added uncertainty in bold and previous belief in gray, (e) new measurement with uncertainty in bold and latest belief in gray, and (f) the updated belief in bold with previous belief and measurement in gray.

where $\bar{\mu}_t = A_t\mu_{t-1} + B_t u_t$ and $\bar{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$ come from integrating the product of the Gaussian distributions. Thus, the prediction step can be summarized with (12). From (9), we have the following relationship between the posterior distribution and the predicted belief:

$$bel(x_t) = \eta p(z_t \mid x_t)\overline{bel}(x_t). \tag{13}$$

The assumptions of a linear Gaussian system provide that $p(z_t \mid x_t) \sim \mathcal{N}(Cx_t, Q_t)$ and that $\overline{bel}(x_t) \sim \mathcal{N}(\bar{\mu}_t, \bar{\Sigma}_t)$. Noting that the product of two Gaussians is Gaussian, we have

$$bel(x_t) = \mathcal{N}(\mu_t, \Sigma_t), \tag{14}$$

for which the equations for μ_t, and Σ_t can be expressed as in (7).

15.3 Extended Kalman Filter (EKF)

While the Kalman filter works beautifully on Gaussian linear systems, the reality is that most real life systems are non-linear. The Extended Kalman Filter (EKF) is a Kalman filter that relaxes the linearity. EKF simply adapts a linearized Gaussian model using 1st order Taylor approximations. The system dynamics are also linearized. After linearization, the rest of belief propagation is essentially identical to the Kalman filter.

Consider a nonlinear system dynamics and measurement model

$$x_t = g(u_t, x_{t-1}) + \epsilon_t \tag{15}$$

$$z_t = h(x_t) + \delta_t, \tag{16}$$

where $g(u_t, x_{t-1})$ and $h(x_t)$ are nonlinear functions.

We use the first-order Taylor expansion to linearize g and h around the most likely state (e.g. the mean of the state), then pass beliefs through linearized models. For the dynamics equation, we propagate the state as $x_t = g(u_t, x_{t-1})$, with $g(u_t, x_{t-1})$ in the form of

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + J_g(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1}) \tag{17}$$

where J_g is the Jacobian matrix of g, $J_g = \frac{\partial g(x_t, u_t)}{\partial x_t}$. Using the notation $G_t = J_g(u_t, \mu_{t-1})$, the probability distribution of the state at time t given the state at time $t-1$ and the control action taken at time t is expressed as

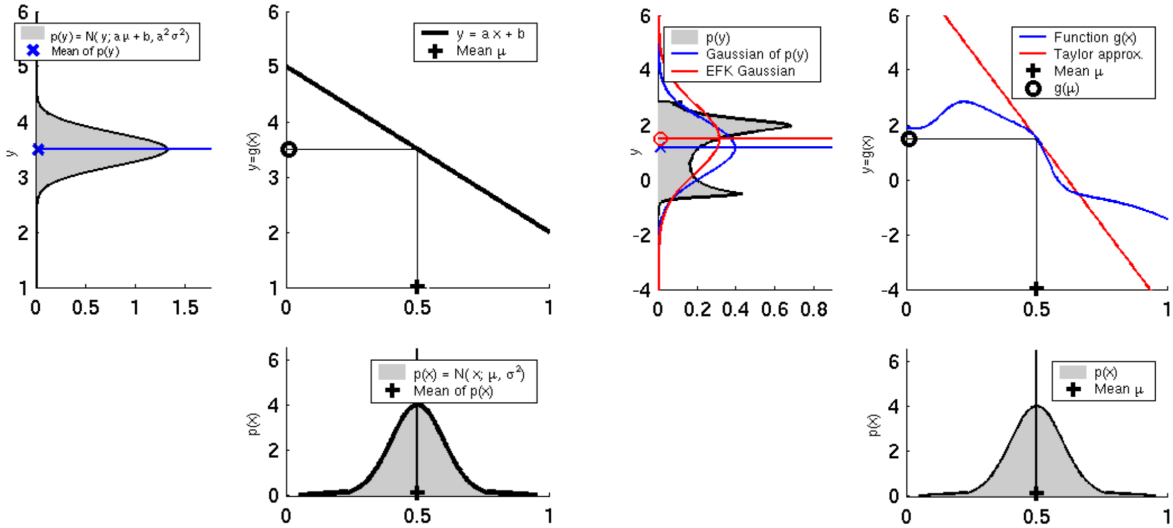
$$p(x_t \mid u_t, x_{t-1}) = \det(2\pi R_t)^{-1/2} \exp\left(-\frac{1}{2}M_t^T R_t^{-1} M_t\right) \tag{18}$$

where $M_t = [x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})]$. Similarly, the measurement model $z_t = h(x_t)$ can be expressed as

$$h(x_t) = h(\bar{\mu}_t) + J_h(\bar{\mu}_t)(x_t - \bar{\mu}_t) \tag{19}$$

The Jacobian of the measurement function is $J_h(\bar{\mu}_t) = \frac{\partial g(x_t, u_t)}{x_t}$. For simplicity, we use $J_h(\bar{\mu}_t) = H_t$. Given this notation, the probability distribution of the measurement given the state at time t is

$$p(z_t \mid x_t) = \det(2\pi Q_t)^{-1/2} \exp\left(-\frac{1}{2}N_t^T Q_t^{-1} N_t\right) \tag{20}$$



(a) Kalman Filter Linearity Assumption

(b) EKF Linearization

Figure 4: Comparison of (a) the linearity assumption of the standard Kalman Filter and (b) its extension to nonlinear functions through linearization.

where $N_t = [z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t)]$. A comparison of the linearity requirement of the standard Kalman Filter and the performance of the EKF on a nonlinear function that violates this requirement is shown in Figure 4.

EKF Algorithm As mentioned earlier, EKF is essentially identical to KF except for the linearized g and h using Jacobian.

Prediction step. We propagate our prior through idealized (noise-free) system dynamics through time. The only difference, however, is that instead of simply using a linear Gaussian model, we use the linearized model.

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) \tag{21}$$
$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t \tag{22}$$

Update step. We make corrections to our prior based on the newest measurement using Bayes' Law. Similarly as with the Kalman filter, our belief is updated using

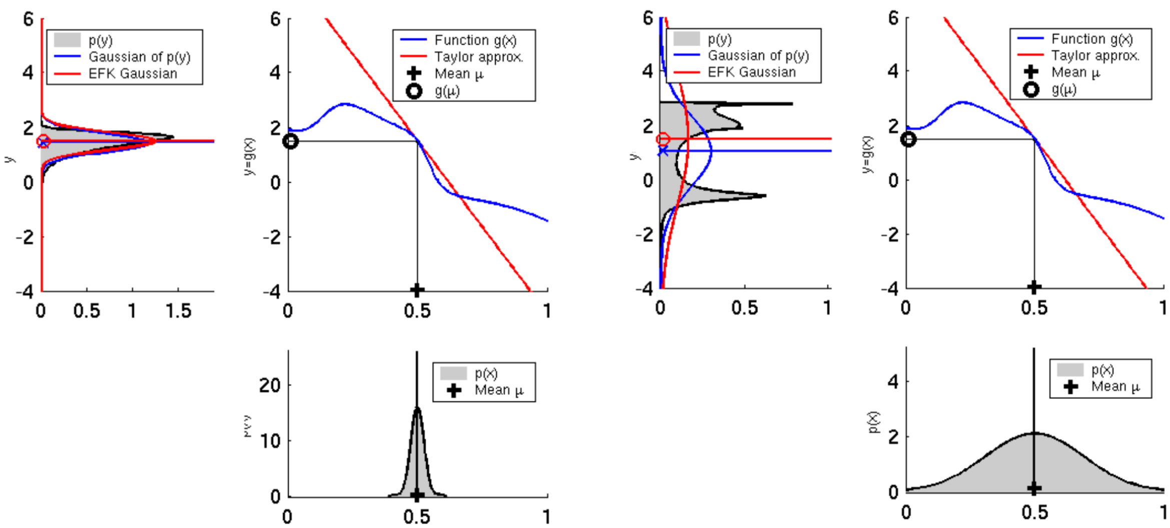
$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\mu_t)) \tag{23}$$
$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t \tag{24}$$

where Kalman gain K_t is again $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$. The prediction and update steps are repeated throughout the simulation. Below, Algorithm 2 outlines the implementation details of EKF.

Data: $(\mu_{t\ \ 1}, \Sigma_{t\ \ 1}), u_t, z_t$
Result: (μ_t, Σ_t)
$\bar{\mu}_t = g(u_t, \mu_{t\ \ 1})$ ▷ Predict mean
$\bar{\Sigma}_t = G_t \Sigma_{t\ \ 1} G_t^T + R_t$ ▷ Predict covar
$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{\ \ 1}$ ▷ Kalman gain
$\mu_t = \bar{\mu}_t + K_t(z_t \quad h(\mu_t))$ ▷ Correct mean
$\Sigma_t = (I \quad K_t H_t)\bar{\Sigma}_t$ ▷ Correct covar
Result: (μ_t, Σ_t)

Algorithm 2: Extended Kalman Filter

EKF Results While EKF models nonlinear systems more accurately than KF, linearizing the system is still a simplification for which accuracny can vary. The accuracy of the linearity assumption in EKF depends heavily on the variance. Linearizing around the mean is only valid for points that are near the mean. So if the variance is low, as in Figure 5a, EKF provides an accurate estimate of the true state. If there is high variance in the prior distribution, as in Figure 5b, EKF will propagate forward a poor estimate of a poor estimate, and can become very ineffective. Advantages of EKF over other nonlinear variants of the Kalman filter include its relative computational efficiency.

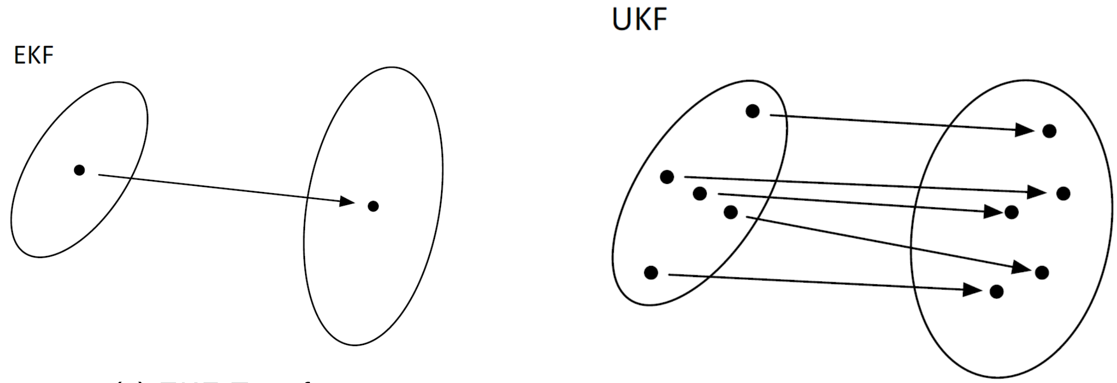


(a) EKF With Low Variance (b) EKF With High Variance

Figure 5: EKF performance for different levels of uncertainty.

15.4 Unscented Kalman Filter

Unscented Kalman Filter (UKF) was first introduced in 1995 to overcome a main drawback of EKF that linearization errors could lead to divergence in filter algorithms. UKF attempts to implement the Kalman filter without explicit linearization or using Jacobians to model



(a) EKF Transform

(b) UKF Transform

Figure 6: Transforms from prior to posterior for (a) EKF and (b) UKF. The EKF handles nonlinearities by linearizing at a single point. The UKF instead computes values of the nonlinear function at many sigma points and fits a Gaussian to the resulting points.

nonlinear dynamics. Instead, UKF represents the Gaussian distribution using *sigma points*. Sigma points are the points around the mean to represent a Gaussian belief distribution. The benefit of using sigma points is that the system dynamics do not have to be linear. The differences between the treatment on nonlinearities in the EKF and UKF are illustrated in Figure 6.

At a high level, UKF finds the sigma points from the mean and covariance through the *unscented transform* at each time step. Instead of propagating the mean and covariance though linearized system dynamics, sigma points are now directly propagated through time with nonlinear system dynamics. Afterwards, the new mean and covariance are found by taking inverse unscented transform over the new sigma points.

Unscented transform. Here, we assume an n-dimensional Gaussian $\mathcal{N}(\mu, \Sigma)$. Instead of linearizing about the mean, the unscented transform propagates "sigma points"—samples chosen to represent the probability distribution—through the nonlinear function $g(x_t, u_t)$. There are a total of $2n+1$ sigma points (numbered $i = 0, \ldots, 2n$): one point at the mean and two points symmetrically distributed according to the covariance in each of the n dimensions.

$$
\begin{aligned}
\mathcal{X}^{[0]} &= \mu \\
\mathcal{X}^{[i]} &= \mu + \left(\sqrt{(n+\lambda)\Sigma}\right)_i \qquad \text{for } i = 1, \ldots, n \\
\mathcal{X}^{[i]} &= \mu - \left(\sqrt{(n+\lambda)\Sigma}\right)_{i-n} \qquad \text{for } i = n+1, \ldots, 2n,
\end{aligned}
\tag{25}
$$

where $\lambda = \alpha^2(n+\kappa) - n$, and α and κ parameterize the distance from the off-mean sigma points $\mathcal{X}^{[i]}$ from the mean $\mathcal{X}^{[0]}$. In calculating the sigma points (for instance in Algorithm 3), we use the notation $\gamma = \sqrt{n+\lambda}$. Each sigma point is associated with two weights: $w_m^{[i]}$, used to compute the mean, and $w_c^{[i]}$, used to compute the covariance.

$$
\begin{aligned}
w_m^{[0]} &= \frac{\lambda}{n+\lambda}, \qquad w_c^{[0]} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \\
w_m^{[i]} &= w_c^{[i]} = \frac{1}{2(n+\lambda)} \qquad \text{for } i = 1, \ldots, 2n.
\end{aligned} \tag{26}
$$

A value of $\beta = 2$ is typically used [TBF05].

Inverse unscented transform. After computing the sigma points, we pass them through the nonlinear dynamics function,

$$
\mathcal{Y}^{[i]} = g(\mathcal{X}^{[i]}) \tag{27}
$$

Then given the weights we have chosen, we are able to recover the mean and covariance of the distribution,

$$
\begin{aligned}
\mu' &= \sum_{i=0}^{2n} w_m^{[i]} \mathcal{Y}^{[i]} \\
\Sigma' &= \sum_{i=0}^{2n} w_c^{[i]} (\mathcal{Y}^{[i]} - \mu')(\mathcal{Y}^{[i]} - \mu')^T
\end{aligned} \tag{28}
$$

UKF Algorithm The UKF algorithm takes the same inputs as the other variants of the Kalman filter: the probability distribution of the state at the previous timestep, as well as the current control input and measurement. At each time step, we compute the sigma points and then propagate them through the dynamics equation:

$$
\bar{\mathcal{X}}_t^* = g(u_t, \mathcal{X}_{t-1}) \tag{29}
$$

Note that $\bar{\mathcal{X}}_t^*$ do not necessarily correspond to a Gaussian distribution (i.e., maintain their symmetry) after this nonlinear transformation. Then, from $\bar{\mathcal{X}}_t^*$, we can compute the *predicted belief* $(\bar{\mu}_t, \bar{\Sigma}_t)$ using the weighting as shown in (26):

$$
\begin{aligned}
\bar{\mu}_t &= \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_t^{*[i]} \\
\bar{\Sigma}_t &= \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t\right)\left(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t\right)^T + R_t
\end{aligned} \tag{30}
$$

We compute the predicted sigma-points $\bar{\mathcal{X}}_t$ from the predicted belief $(\bar{\mu}_t, \bar{\Sigma}_t)$ as follows, using the same rules as in (25). Propagating the sigma points through the nonlinear measurement function gives

$$
\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t), \tag{31}
$$

from which the predicted observation can be computed:

$$
\begin{aligned}
\hat{z}_t &= \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]} \\
S_t &= \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)\left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^T + Q_t \\
\bar{\Sigma}_t^{x,z} &= \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t\right)\left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^T
\end{aligned}
\tag{32}
$$

The quantity S_t represents the uncertainty as the $(H_t\bar{\Sigma}_t H_t^T + Q_t)$ term in the EKF algorithm, while $\bar{\Sigma}_t^{x,z}$ describes the cross-covariance between the state and the measurement. Together, these two terms contribute to the Kalman gain:

$$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1} \tag{33}$$

Then we can update the belief function:

$$
\begin{aligned}
\mu_t &= \bar{\mu}_t + K_t(z_t - \hat{z}_t) \\
\Sigma_t &= \bar{\Sigma}_t - K_t S_t K_t^T
\end{aligned}
\tag{34}
$$

Implementation details of UKF are found in 3.

Data: $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
Result: (μ_t, Σ_t)
$\mathcal{X}_{t-1} = \left(\mu_{t-1} \quad \mu_{t-1} + \gamma\sqrt{\Sigma_{t-1}} \quad \mu_{t-1} - \gamma\sqrt{\Sigma_{t-1}}\right)$ ▷ Form sigma points (25)
$\bar{\mathcal{X}}_t^* = g(u_t, \mathcal{X}_{t-1})$ ▷ Propagate through dynamics
$\bar{\mu}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_t^{*[i]}$ ▷ Predict mean
$\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t\right)\left(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t\right)^T + R_t$ ▷ Predict covariance
$\bar{\mathcal{X}}_t = \left(\bar{\mu}_t \quad \bar{\mu}_t + \gamma\sqrt{\bar{\Sigma}_t} \quad \bar{\mu}_t - \gamma\sqrt{\bar{\Sigma}_t}\right)$ ▷ Predict sigma points
$\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$ ▷ Propagate through nonlinear measurement
$\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$ ▷ Predicted observation
$S_t = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)\left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^T + Q_t$ ▷ Uncertainty
$\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t\right)\left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^T$ ▷ Cross-covariance
$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$ ▷ Kalman gain
$\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$ ▷ Correct mean
$\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$ ▷ Correct covariance
Return: (μ_t, Σ_t)

Algorithm 3: Unscented Kalman Filter