AA 274: Principles of Robotic Autonomy | Winter 2018

Lecture 11: Localization Filters

Scribes: Arthur Binstein, Anqi Fu, Trevor Halsted, Scott Hemley, Alexander Hobbs, Jialong Wang

11.1 Introduction

The first part of this lecture covers filtering techniques including parametric and non-parametric filters. The parametric filters include the Kalman Filter, the Extended Kalman Filter and the Unscented Kalman Filter, while the non-parametric filters covered are the histogram and particle filters. The second part of this lecture is a brief introduction to robot localization.

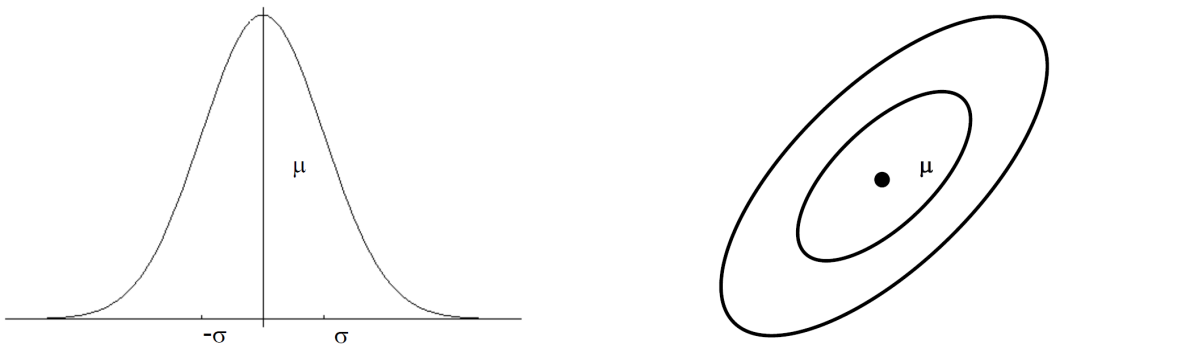
11.2 Parametric Filters

Parametric filters are a family of state estimators that assume the form of a robot's belief distribution and calculate the parameters to fully define that distribution based on the latest information. In particular, Gaussian filters are an important group of parametric filters that assume the belief distribution is a multivariate normal distribution. The Kalman filter and its variants belong to this group of parametric filters.

11.2.1 Kalman Filter

11.2.1.1 Setup

A Kalman filter K_t uses a model of the system dynamics, known control inputs to a system, and noisy measurements to estimate the robot's state. The algorithm calculates weights on the state predicted from



(a) Gaussian Distribution in One Variable (b) Level Curves of Gaussian Distribution in Two Variables

Figure 11.1: Visualizations of the multivariate normal (Gaussian) distribution [1].

the measurements and the state predicted by the model. The key idea behind Kalman filters is that we assume that each belief function (the expected state) follows a multivariate normal distribution (Gaussian distribution), illustrated in Figure 11.1. However, instead of propagating the probability density function itself, we choose to propagate forward the project state (mean) and covariance.

In general, the linearity assumption does not hold (e.g., square root in Euclidean distance calculation) when modeling the robot. But we make several assumptions when using Kalman filters that simplify the problem. First we assume a linear dynamics model of the robot:

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \tag{11.1}$$

where ϵ_t is the independent process noise with variance R_t. Notice here that the probability generative model now is also Gaussian because the Gaussian of a linear equation is still Gaussian. Second we assume that the measurement model is also linear

$$z_t = C_t x_t + \delta_t \tag{11.2}$$

where δ_t is the independent measurement noise with variance Q_t. This ensures the measurement probability is also a Gaussian. The third and last assumption is that the initial belief function is also Gaussian in the form of

$$bel(x_0) = \det(2\pi\Sigma_0)^{-\frac{1}{2}} \exp(-\frac{1}{2}(x_0 - \mu_0)\Sigma_0^{-1}(x_0 - \mu_0)^T) \tag{11.3}$$

such that our posterior *bel* functions will also follow Gaussian distribution.

11.2.1.2 Algorithm

The first step of the algorithm is to predict the state by applying the state transition model to the prior estimate of the state, μ_{t-1}. Similarly, the covariance matrix is updated as well, based on the state transition matrix A_t and the process noise R_t. The intuition for the prediction of the covariance is that the predicted state depends on the previous state according to the matrix A_t. The prior covariance is pre- and post-multiplies by A_t because the covariance is quadratic [2]. Also we can see that the process noise R_t corresponds to the uncertainty in the prediction, which is described by the covariance.

$$\begin{aligned} \bar{\mu}_t &= A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t &= A_t \Sigma_{t-1} A_t^T + R_t \end{aligned} \tag{11.4}$$

After the computation of the prediction, the estimate of the state is corrected according to the current measurement. First, the *Kalman gain* is computed. The Kalman gain describes how much the measurement contributes to the update of the state estimate, and it takes into account the prediction covariance (computed in (11.4)), the measurement model C_t and the measurement noise Q_t. We can observe that a large measurement noise in Q_t corresponds to a smaller K_t. In other words, noisy measurements will not be incorporated into the state estimate as much as measurements that have little noise.

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \tag{11.5}$$

The measurement update step incorporates the Kalman gain (describing how much the measurement contributes to the new state estimate) and the *innovation* (the difference between the actual measurement and the predicted measurement $C_t \bar{\mu}_t$) to correct the state estimate. The amount that the mean of the state distribution is corrected is proportional to the Kalman gain and the innovation. Also, the covariance is adjusted according to the information that is gained by the measurement. For large Kalman gains (i.e., a high degree to which the measurement contributes to the state estimate), the covariance matrix decreases

(i.e., the uncertainty in the belief is reduced). Thus, while the prediction step increases the uncertainty in the robot's belief, the measurement corrects the belief and decreases the uncertainty.

$$\begin{aligned} \mu_t &= \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t) \\ \Sigma_t &= (I - C_tK_t)\bar{\Sigma}_t \end{aligned} \tag{11.6}$$

A few iterations of this algorithm are depicted in Figure 11.2.

Algorithm 1: Kalman Filter

Data: $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
Result: (μ_t, Σ_t)
$\bar{\mu}_t = A_t\mu_{t-1} + B_tu_t$;
$\bar{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$;
$K_t = \bar{\Sigma}_tC_t^T(C_t\bar{\Sigma}_tC_t^T + Q_t)^{-1}$;
$\mu_t = \bar{\mu}_tK_t(z_t - C_t\bar{\mu}_t)$;
$\Sigma_t = (I - C_tK_t)\bar{\Sigma}_t$;
Return: (μ_t, Σ_t)

11.2.1.3 Derivation

The following recursive update equation can be derived by applying Bayes rule to the posterior distribution:

$$p(x_t \mid z_{1:t}, u_{1:t}) = \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t})p(x_t \mid z_{1:t-1}, u_{1:t}). \tag{11.7}$$

Noting that conditional independence gives the simplification

$$p(x_t \mid z_{1:t}, u_{1:t}) = \eta p(z_t \mid x_t)p(x_t \mid z_{1:t-1}, u_{1:t}). \tag{11.8}$$

Given, $\overline{bel}(x_t) = p(x_t \mid z_{1:t-1}, u_{1:t})$, we can assume that the state is complete to obtain the recursive update equation:

$$\overline{bel}(x_t) = \int p(x_t \mid x_{t-1}, u_t)p(x_{t-1} \mid z_{1:t-1}u_{1:t-1})dx_{t-1} \tag{11.9}$$

which can also be expressed as

$$\overline{bel}(x_t) = \int p(x_t \mid x_{t-1}, u_t)bel(x_{t-1})dx_{t-1} \tag{11.10}$$

Since $x_t = A_tx_{t-1} + B_tu_t + \epsilon_t$ with $\epsilon_t \sim \mathcal{N}(\mu_t, R_t)$, we know that $p(x_t \mid x_{t-1}, u_t) \sim \mathcal{N}(A_tx_{t-1} + B_tu_t, R_t)$. Similarly, the Linear-Gaussian assumptions described above guarantee that $bel(x_{t-1}) \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1}$. Then (11.10) can be simplified to

$$\overline{bel}(x_t) = \mathcal{N}(\bar{\mu}_t, \bar{\Sigma}_t) \tag{11.11}$$

where $\bar{\mu}_t = A_t\mu_{t-1} + B_tu_t$ and $\bar{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$ from integrating the product of the Gaussian distributions. Thus, the prediction step can be summarized with (11.11). From (11.8), we have the following relationship between the posterior distribution and the predicted belief:

$$bel(x_t) = \eta p(z_t \mid x_t)\overline{bel}(x_t). \tag{11.12}$$

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

$$bel(x_t) = \mathcal{N}(\mu_t, \Sigma_t), \tag{11.13}$$

for which the equations for μ_t, and Σ_t can be expressed as in (11.6).

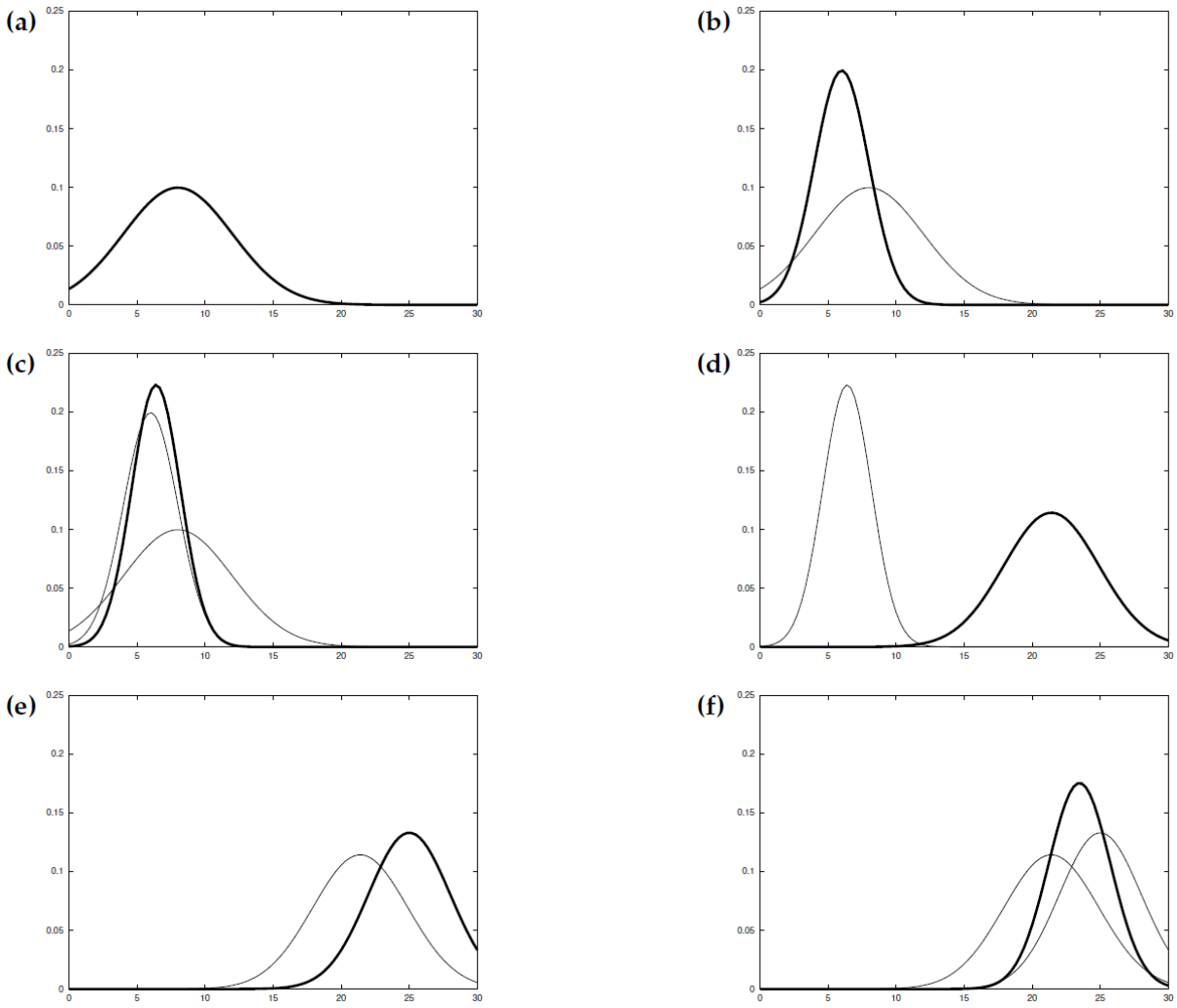


Figure 11.2: Visualization of the Kalman Filter Algorithm from [2]. Current step is in bold and previous data are in gray: (a) the initial belief distribution, (b) the measurement with uncertainty in bold and initial belief in gray, (c) updated belief in bold with previous belief and measurement in gray, (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.

11.2.2 Extended Kalman Filter (EKF)

The Extended Kalman Filter (EKF) is a Kalman Filter that relaxes the linearity assumption of the models. Instead of trying to find the exact posterior solution, it seeks to compute a Gaussian approximation. The belief distribution is passed through the filter and results in a non-Gaussian distribution. Then we find the best Gaussian fit to this resulting distribution.

The assumption in EKF is that we can 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

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + J_g(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1}) \tag{11.14}$$

where J_g is the Jacobian matrix of the function g. 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

$$p(x_t \mid u_t, x_{t-1}) = \det(2\pi R_t)^{-1/2} \exp\left(-\frac{1}{2}\left[x_t - g(u_t, \mu_{t-1}) - G_t \cdot (x_{t-1} - \mu_{t-1})\right]^T R_t^{-1} \left[x_t - g(u_t, \mu_{t-1}) - G_t \cdot (x_{t-1} - \mu_{t-1})\right]\right) \tag{11.15}$$

And for the measurement model $z_t = h(x_t)$, where

$$h(x_t) = h(\bar{\mu}_t) + J_h(\bar{\mu}_t)(x_t - \bar{\mu}_t) \tag{11.16}$$

The Jacobian of the measurement function is $J_h(\bar{\mu}_t)$, for which we use the notation H_t. Then the probability distribution of the measurement given the state at time t is

$$p(z_t \mid x_t) = \det(2\pi Q_t)^{-1/2} \exp\left(-\frac{1}{2}\left[z_t - h(\bar{\mu}_t) - H_t \cdot (x_t - \bar{\mu}_t)\right]^T Q_t^{-1} \left[z_t - h(\bar{\mu}_t) - H_t \cdot (x_t - \bar{\mu}_t)\right]\right) \tag{11.17}$$

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 11.3.

11.2.2.1 EKF Algorithm

Unlike ordinary Kalman Filters, the EKF algorithm uses Jacobian matrices of the linearized model g and h instead of the linear system matrices. In the prediction step we propagate forward the expected value of the state assuming no noise. We also replace the covariance matrix A_t with Jacobian G_t from the linearization. While in the correction step we replace C_t in the Kalman gain with Jacobian H_t. The linearized dynamics equations in (11.4) are replaced by the following:

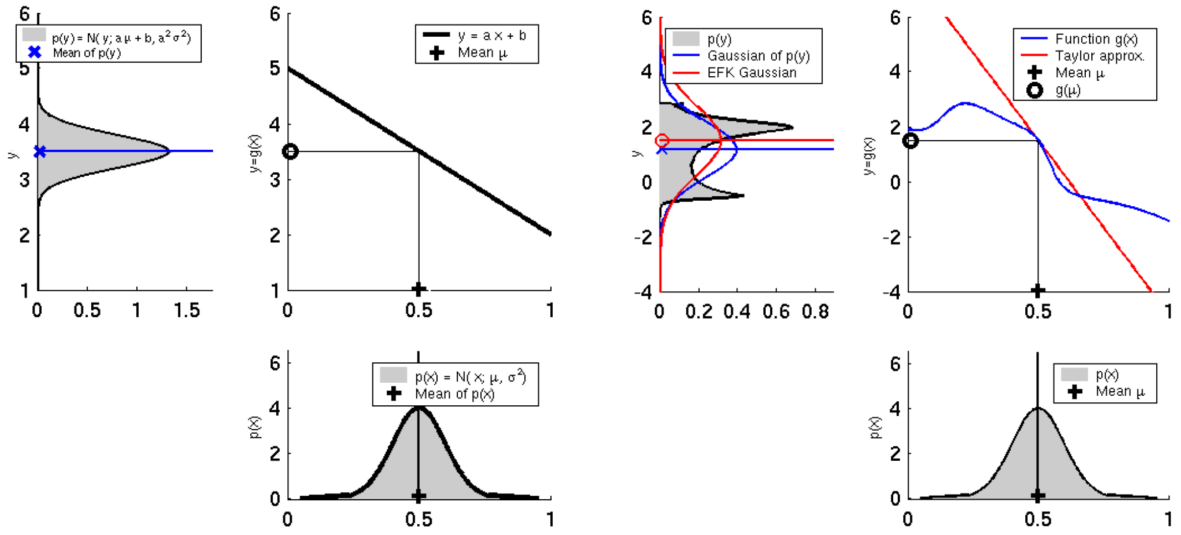
$$\begin{aligned} \bar{\mu}_t &= g(u_t, \mu_{t-1}) \\ \bar{\Sigma}_t &= G_t \Sigma_{t-1} G_t^T + R_t \end{aligned} \tag{11.18}$$

Then we need to correct the prediction with our measurement. First we compute the Kalman gain,

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \tag{11.19}$$

Then we update our belief function

$$\begin{aligned} \mu_t &= \bar{\mu}_t + K_t(z_t - h(\mu_t)) \\ \Sigma_t &= (I - K_t H_t)\bar{\Sigma}_t \end{aligned} \tag{11.20}$$



(a) Kalman Filter Linearity Assumption

(b) EKF Linearization

Figure 11.3: Comparison of (a) the linearity assumption of the standard Kalman Filter [1] and (b) its extension to nonlinear functions through linearization [1].

Repeat this process.

Algorithm 2: Extended Kalman Filter

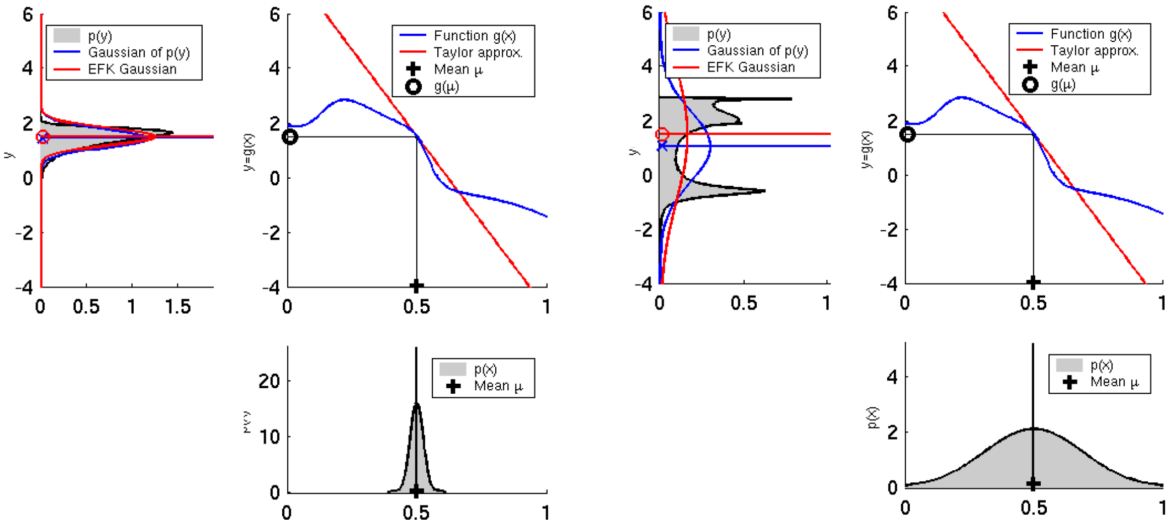
Data: $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
Result: (μ_t, Σ_t)
$\bar{\mu}_t = g(u_t, \mu_{t-1})$;
$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$;
$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$;
$\mu_t = \bar{\mu}_t + K_t(z_t - h(\mu_t))$;
$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$;
Return: (μ_t, Σ_t)

11.2.2.2 EKF Results

The accuracy of the linearity assumption in the EKF is strongly dependent 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 11.4a, the EKF provides an accurate estimate of the true state. If there is high variance in the prior distribution, as in Figure 11.4b, the EKF will propagate forward a poor estimate of a poor estimate, and can become very ineffective.

11.2.3 Unscented Kalman Filter (UKF)

Where in EKF, a Gaussian prediction is propagated through a simplified dynamic model, in UKF an approximated belief is propagated through the correct nonlinear model. This is accomplished by choosing points near the mean, called sigma points, transforming these points using the nonlinear model, and com-



(a) EKF With Low Variance

(b) EKF With High Variance

Figure 11.4: EKF performance for different levels of uncertainty [1].

puting a posterior Gaussian distribution using these transformed sigma points. The differences between the treatment on nonlinearities in the EKF and UKF are illustrated in Figure 11.5.

11.2.3.1 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. 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.

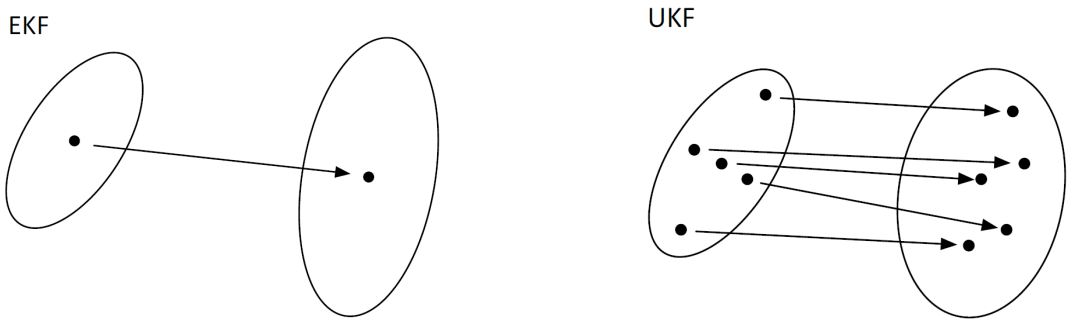
$$\begin{aligned} \mathcal{X}^{[0]} &= \mu \\ \mathcal{X}^{[i]} &= \mu + \left(\sqrt{(n+\lambda)\Sigma}\right)_i \qquad \text{for } i = 1, \ldots, n \\ \mathcal{X}^{[i]} &= \mu - \left(\sqrt{(n+\lambda)\Sigma}\right)_{i-n} \qquad \text{for } i = n+1, \ldots, 2n, \end{aligned} \tag{11.21}$$

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.

$$\begin{aligned} w_m^{[0]} &= \frac{\lambda}{n+\lambda}, \qquad w_c^{[0]} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \\ w_m^{[i]} &= w_c^{[i]} = \frac{1}{2(n+\lambda)} \qquad \text{for } i = 1, \ldots, 2n. \end{aligned} \tag{11.22}$$

A value of $\beta = 2$ is typically used [2]. After computing the sigma points, we pass them through the nonlinear dynamics function,

$$\mathcal{Y}^{[i]} = g(\mathcal{X}^{[i]}) \tag{11.23}$$



(a) EKF Transform (b) UKF Transform

Figure 11.5: Transforms from prior to posterior for (a) EKF [1] and (b) UKF [1]. 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.

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

$$\begin{aligned} \mu' &= \sum_{i=0}^{2n} w_m^{[i]} \mathcal{Y}^{[i]} \\ \Sigma' &= \sum_{i=0}^{2n} w_c^{[i]} (\mathcal{Y}^{[i]} - \mu')(\mathcal{Y}^{[i]} - \mu')^T \end{aligned} \tag{11.24}$$

11.2.3.2 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:

$$\bar{\mathcal{X}}_t^* = g(u_t, \mathcal{X}_{t-1}) \tag{11.25}$$

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 (11.22):

$$\begin{aligned} \bar{\mu}_t &= \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_t^{*[i]} \\ \bar{\Sigma}_t &= \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t \right) \left(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t \right)^T + R_t \end{aligned} \tag{11.26}$$

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 (11.21). Propagating the sigma points through the nonlinear measurement function gives

$$\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t), \tag{11.27}$$

from which the predicted observation can be computed:

$$\begin{aligned}\hat{z}_t &= \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]} \\ S_t &= \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)\left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^T + Q_t \\ \bar{\Sigma}_t^{x,z} &= \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t\right)\left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^T\end{aligned} \tag{11.28}$$

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:

$$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1} \tag{11.29}$$

Then we can update the belief function:

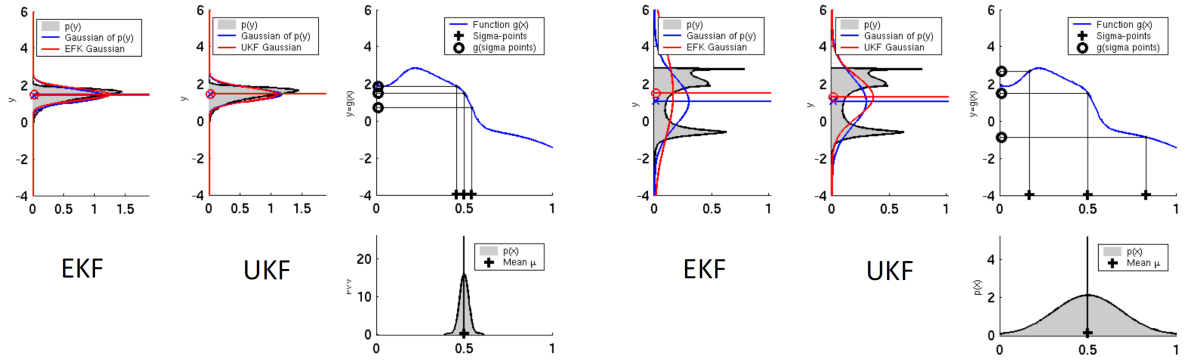
$$\begin{aligned}\mu_t &= \bar{\mu}_t + K_t(z_t - \hat{z}_t) \\ \Sigma_t &= \bar{\Sigma}_t - K_t S_t K_t^T\end{aligned} \tag{11.30}$$

Algorithm 3: Unscented Kalman Filter

Data: $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
Result: (μ_t, Σ_t)
$\mathcal{X}_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \gamma\sqrt{\Sigma_{t-1}} \quad \mu_{t-1} - \gamma\sqrt{\Sigma_{t-1}})$;
$\bar{\mathcal{X}}_t^* = g(u_t, \mathcal{X}_{t-1})$;
$\bar{\mu}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_t^{*[i]}$;
$\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t\right)\left(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t\right)^T + R_t$;
$\bar{\mathcal{X}}_t = \left(\bar{\mu}_t \quad \bar{\mu}_t + \gamma\sqrt{\bar{\Sigma}_t} \quad \bar{\mu}_t - \gamma\sqrt{\bar{\Sigma}_t}\right)$;
$\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$;
$\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$;
$S_t = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)\left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^T + Q_t$;
$\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t\right)\left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^T$;
$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$;
$\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$;
$\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$;
Return: (μ_t, Σ_t)

11.2.3.3 UKF Results

While the asymptotic complexities for the UKF and EKF algorithms are equivalent, the EKF is typically marginally faster than the UKF in practice. However, the UKF has the advantage if not requiring computation of Jacobians, which can sometimes pose an analytic challenge. The UKF is much less sensitive to variance than the EKF because it propagates forward the exact nonlinear motion and measurement processes. As Figure 11.6 shows, the UKF and EKF perform very similarly with low variance, but the UKF outperforms the EKF with high variance.



(a) UKF with Low Variance

(b) UKF with High Variance

Figure 11.6: The performance of the UKF compared to the EKF for different levels of uncertainty [1].

11.3 Nonparametric Filters

Nonparametric filters do not make any assumptions on the robot's belief distribution. Instead, they approximate the distribution discretely with a finite number of values. The accuracy of the representation depends on the number of values used, which results in a trade-off between expressiveness and computational burden. They are more robust to nonlinearities and discontinuities in the inference space, but also tend to be more complicated and time-consuming to implement.

11.3.1 Histogram Filter

Histogram filters are one way of finding a discrete approximation of a continuous distribution of beliefs. They first decompose a continuous space into finitely many bins,

$$\text{dom}(X_t) = x_{1,t} \cup x_{2,t} \cup ...x_{k,t} \tag{11.31}$$

then each region $x_{k,t}$ is assigned a probability $p_{k,t}$, which is approximated in a piecewise scheme based on the assumption of uniform density in each bin.

$$p(x_t) = \frac{p_{k,t}}{|x_{k,t}|}, \quad x_t \in x_{k,t} \tag{11.32}$$

To calculate the probability distribution, the motion and measurement models are discretized, and the central mass in each region is chosen as its representative state $\hat{x}_{k,t}$,

$$\hat{x}_{k,t} = |x_{k,t}|^{-1} \int_{x_{k,t}} x_t dx_t \tag{11.33}$$

Given a state $x_{k,t}$, the conditional probability $p(z_t \mid x_{k,t})$ is set equal to the probability of that measurement conditional on the representative state in that region, $p(z_t \mid \hat{x}_{k,t})$. A similar process is used to approximate the state transition probabilities by applying Bayes' law,

$$p(x_{k,t} \mid u_t, x_{i,t-1}) = \eta |x_{k,t}| p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1}) \tag{11.34}$$

Finally, we execute a discrete Bayes' filter on the discretized probabilities to estimate the full belief distribution. A histogram representation of a robot's belief is shown in Figure 11.7.

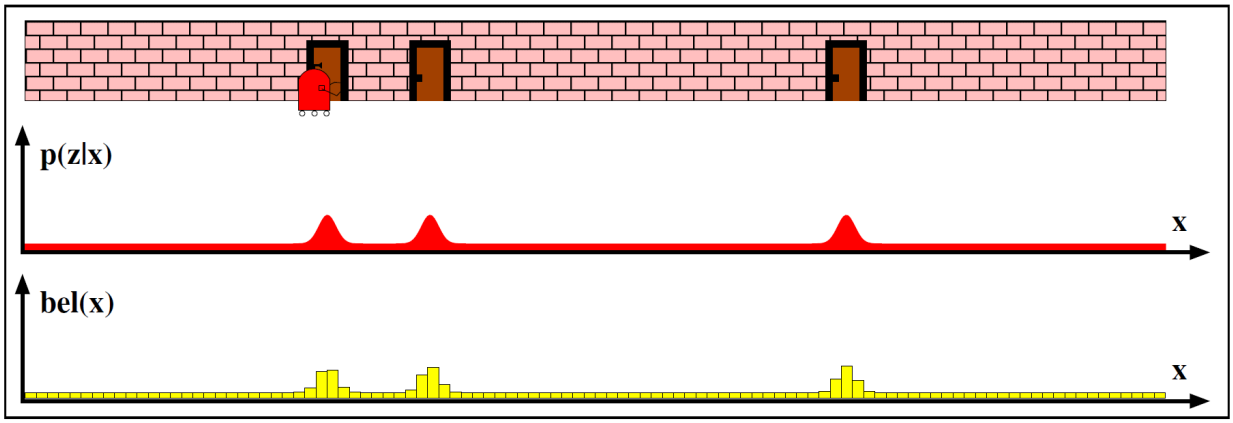


Figure 11.7: Histogram representation of belief from initial sensor measurement from [2].

11.3.2 Particle Filter

A particle filter approximates the posterior distribution with a finite number of particles, denoted as

$$\mathcal{X}_t = x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]} \tag{11.35}$$

with each $x_t^{[k]}$ representing a hypothesis of what the true world state would be at time t. The denser the particles, the higher the probability of landing in a region. The algorithm operates iteratively: given $\mathcal{X}_{t-1}$, we construct the next particle set $\mathcal{X}_t$ by resampling $\mathcal{X}_{t-1}$ to correct the predicted belief function, incorporating our measurements into the calculation. This is done by computing the probability we observe a particular measurement conditional on the state, then drawing particles with replacement using these probabilities.

After resampling, particles are distributed as

$$bel(X_t) = \eta P(z_t \mid x_t^{[m]}) \overline{bel}(X_t) \tag{11.36}$$

for $M \to \infty$. That is, in the limit, we recover the distribution of the true nonlinear filter. ($M \approx 1000$ achieves acceptable accuracy in practice). A few iterations of the particle filter algorithm for robot localization are shown in Figure 11.8.

Algorithm 4: Particle Filter

Data: $\mathcal{X}_{t-1}, u_t, z_t$
Result: $\mathcal{X}_t$
$\bar{\mathcal{X}}_t = X_t = \emptyset$;
for $m = 1$ **to** M **do**
  sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$;
  $w_t^{[m]} = p(z_t \mid x_t^{[m]})$;
  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t \cup \left(x_t^{[m]}, w_t^{[m]}\right)$;
end
for $i = 1$ **to** M **do**
  draw i with probability $\propto w_t^{[i]}$;
  add $x_t^{[i]}$ to $\mathcal{X}_t$;
end
Return: $\mathcal{X}_t$

11.4 Robot Localization

Localization is the process of determining the correspondence between a robot's placement and the environment's map coordinates. Like before, a control input u_t gives rise to a robot state x_t, which produces measurements z_t. The distinction is that this time, the map m influences our measurements and possibly control parameters as well. Different situations thus may call for different filters. This influence is shown in the updated Bayes network (Hidden Markov Model) for robot localization in Figure 11.9. Similarly, the influence of the map (door locations) can be seen in the measurement model $p(z \mid x)$ in Figures 11.7, 11.8b, and 11.8d.

In position tracking, the initial pose of the robot is known with certainty. EKF works well for this problem because the belief distribution is concentrated (indeed, a point mass) at the mean. On the other hand, global localization assumes the initial pose is unknown, so nonparametric filters like histogram or particle filters are preferable. Finally, a case of theoretical interest arises if the initial pose is unknown and the robot can be "kidnapped" during its operation, i.e., a discontinuity may exist in its state $\mathbf{x}(t)$. This models the situation when a filter diverges to a significant enough degree that it must be reset. Hence, our localization algorithm should be capable of restarting from a different point/orientation in space. The kidnapped robot localization problem is best addressed with nonparametric, multi-hypothesis filters.

Environmental changes are another important consideration in mobile robot localization. A static environment assumes that the robot is the only object that moves. In contrast, a dynamic environmental model allows other objects, such as people, to move as well. This problem is usually addressed by augmenting the state vector or rejecting outliers.

Passive localization assumes that a single module takes measurements, and the robot's motion is unrelated to its localization process. A more sophisticated approach is active localization, where the robot's movement is aimed (at least partly) toward improving its understanding of the environment. For example, a robot in a corner will tend to reorient itself to face the rest of the room, so it can collect environmental information as it moves along the wall. In the same situation, a passively localized robot may simply slide along with its camera facing the wall.

Finally, a topic of current research is multi-robot localization: how can a team of robots cooperatively localize themselves within an environment?

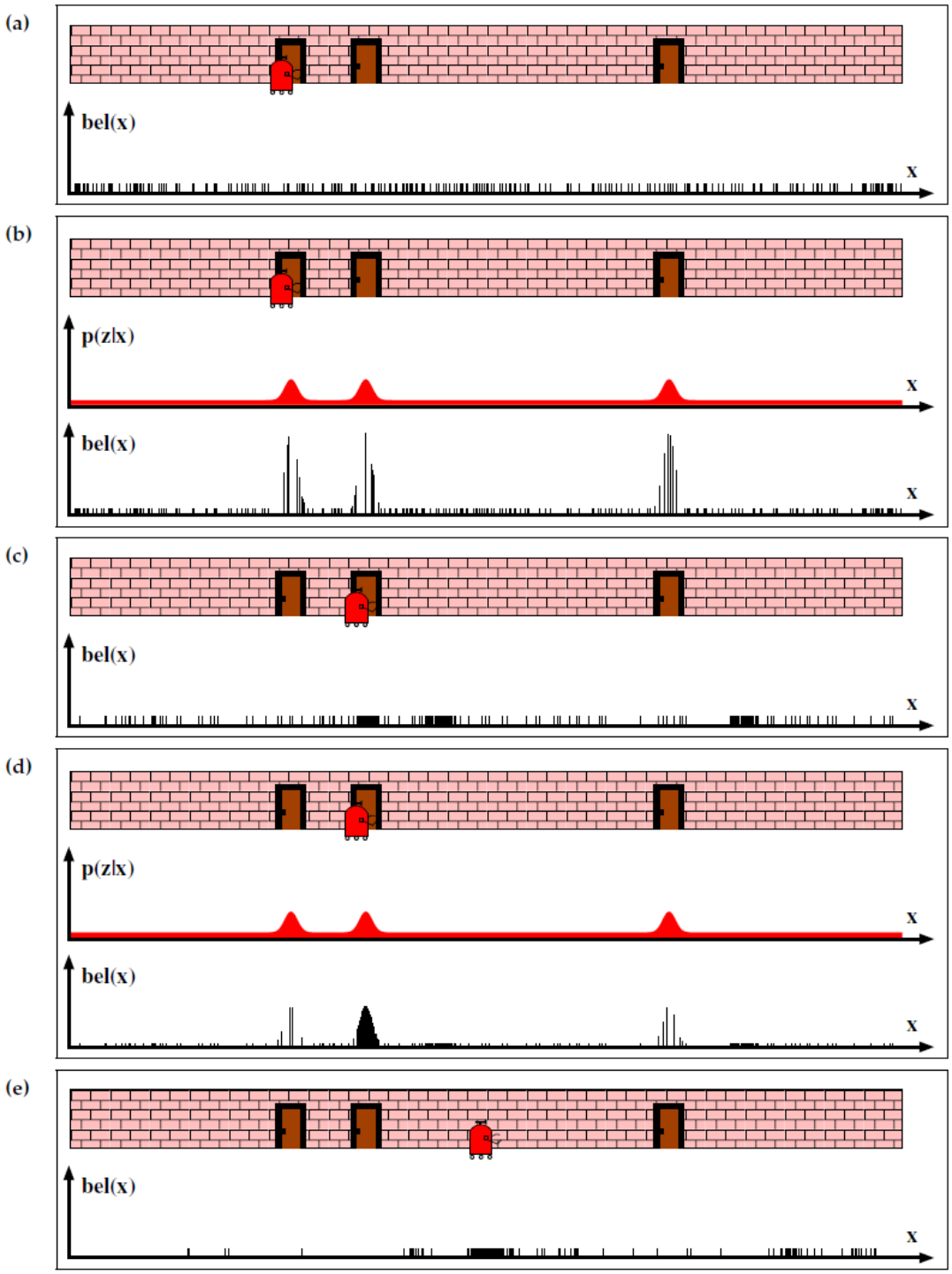


Figure 11.8: Particle filter used for robot localization from [2]: (a) Initial particles sampled uniformly over entire state space, (b) same set of particles from (a) after importance weighting with initial sensor measurement, (c) resampled particles from weighted distribution after motion, (d) importance weighting of new particle set with new sensor measurement, and (e) resampled particle set after further motion.

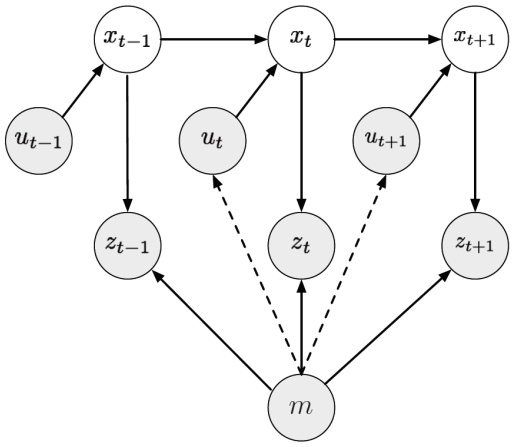


Figure 11.9: Hidden Markov Model for robot localization, where a map informs sensor measurements and control actions [1].

References

[1] Marco Pavone. *AA 274 Principles of Robotic AutonomyLocalization II: parametric and non-parametric filters*. Online. Presentation. Feb. 2018.

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