AA 274: Principles of Robotic Autonomy Winter 2018

Lecture 11: Localization Filters

11.1 Introduction

Last lecture, we discussed the Bayes filter to continuously update our belief on something, for example, a robot's location. However, because the Bayes filter requires iterating over a continuous variable, it is generally impracticable due to computational limits.

The first part of this lecture covers filtering techniques which are based on the Bayes filter but which make certain approximations which render these filters practically implementable, which the Bayes filter is not. We discuss two main classes of filter: parametric filters, and non-parametric filters. *Parametric* filters generally begin by restricting the belief of the Bayes filter to a particular distribution, such as a Gaussian, and proceeding to estimate the *parameters* determining this specific distribution, such as the mean and variance in the case of Gaussian. In *non-parametric* filters, on the other hand, we make no restrictive assumption as to the distribution of the belief; rather, we discretize the belief into "particles," thus replacing the continuous variable in the Bayes filter iteration with a discrete variable and thus making the iteration practicable. Thus while both parametric and non-parametric filters are based on the Bayes filter, they make completely orthogonal simplifying assumptions, and thus we should expect the error source for each to be different as well: for parametric filters, errors will arise from our initial restriction on the possible belief, for non-parametric filters, errors will arise due to the discretization.

The second part of this lecture is a brief introduction to robot localization.

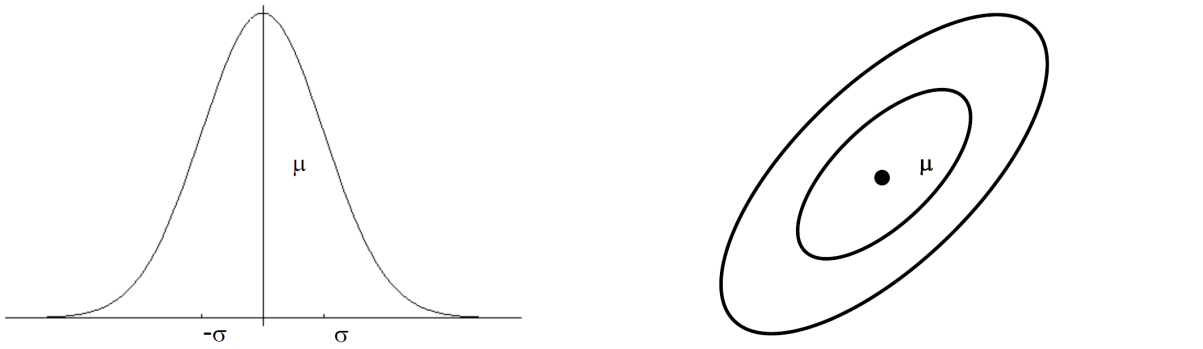
11.2 Parametric Filters

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

The Kalman filter 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 a belief on the state predicted from the model and then updates this belief to be in closer accord with the actual observations / measurements. Thus the belief output by a Kalman filter is informed both by our prior belief (past observations / assumptions) and our observations (current noisy measurements).



(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 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. While we could in theory choose another distribution, Gaussians are convenient for two reasons: (1) assuming the Gaussian is computationally convenient as it produces a closed form solution for updating the Kalman filter, and (2) the Law of Large Numbers suggests that assuming a Gaussian distribution is probably reasonable in most cases. There are instances, however, where this assumption is poor and leads to complications: for example, when a robot is first turned on and has no idea where it is, the uni-modal Gaussian assumption is clearly poor..

Before proceeding into the details of the Kalman filter, let us briefly discuss the assumptions we will make to simplify our analysis. The first, as previously mentioned, is that our belief state is Gaussian. We will use three properties of Gaussian random variables.

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 a Gaussian.

In addition to our Gaussian assumption, we will make three further assumptions on our system itself.

1. We assume that our robot's state evolves via a linear dynamical system. That is, at time t, the state x_t is given by
$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t,$$
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 $\mathcal{N}(0, R_t)$.

   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. We assume that the measurement model is also linear. That is, if we attempt to measure the state of our robot at time t and true state x_t, our measurement z_t is given by

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

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. We assume that the initial belief function is also Gaussian in the form of

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

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

Note that, by the union of these three assumption, all belief states throughout time will follow a Gaussian distribution. Since Gaussian distributions are uniquely determined by the first two moments (that is, the mean and the variance), this further means that we need only propagate the mean and the variance (or, in the multivariate case, the covariance matrix) through time rather than a full expression for the pdf.

11.2.1.2 Algorithm

We will now consider the actual algorithm which implements the Kalman filter. As with the more general Bayes filter, the Kalman filter can be broken down into two steps: a *prediction* step based on the prior belief and a *correction* step based on the most recent measurement (See Fig 11.2).

Note that, by assumptions 1 and 3 given above, we our previous belief state $bel(x_{t-1})$ is known to be Gaussian, and as our process noise $\epsilon(t)$ is also assumed to be Gaussian, $bel(x_t)$ will be Gaussian. Consequently, we need only be provided with $(\mu_{t-1}, \Sigma_{t-1})$ instead of the complete $bel(x_{t-1})$, and we need only estimate (μ_t, Σ_t) instead of the complete $bel(x_t)$.

First we make a prediction of the state, projecting mean and covariance ahead based on our previous belief state $bel(x_{t-1})$ and our most recent control input u_t.

$$\begin{aligned} \bar{\mu}_t &= A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t &= A_t \Sigma_{t-1} A_t^T + R_t \end{aligned} \tag{11.1}$$

Then we need to correct the prediction with our measurement – that is, we adjust our predicted belief to be more in accord with our most recent observation / measurement z_t. First we compute the Kalman gain,

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}. \tag{11.2}$$

At a high level, the Kalman gain can be thought of as the ratio of how much uncertainty we have in our prediction over how much uncertainty we have in our measurement. Having computed the Kalman gain, we update our belief function via a linear update.

$$\begin{aligned} \mu_t &= \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t) \\ \Sigma_t &= (I - C_t K_t)\bar{\Sigma}_t \end{aligned} \tag{11.3}$$

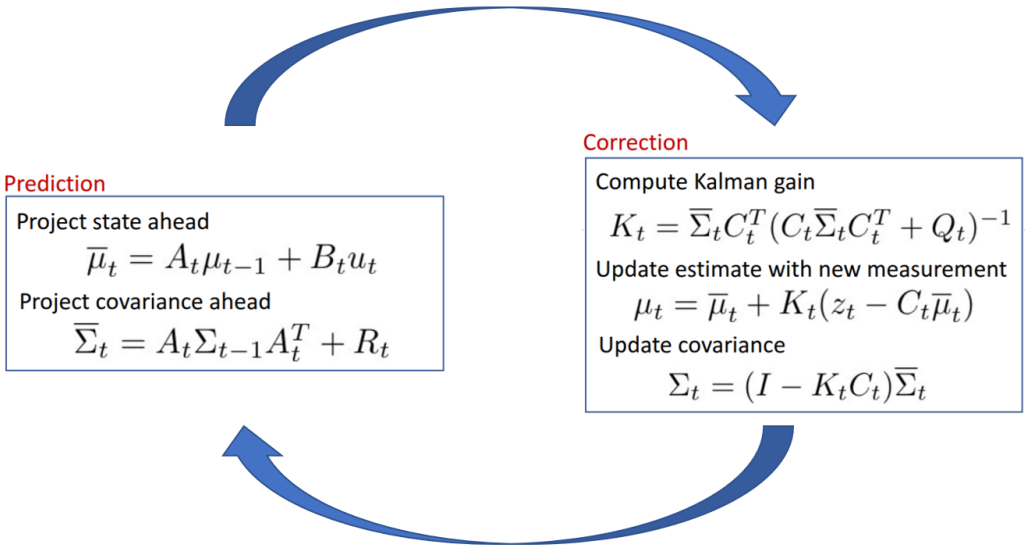


Figure 11.2: Prediction / Correction loop in Kalman Filter

That is, we correct our belief by considering the difference between what we actually observed (z_t) and what we would have observed had our predicted belief been correct ($C_t\bar{\mu}_t$). As intuition would suggest, if K_t is large, meaning that we have much more uncertainty in our prediction than in our measurement, our correction term is large; if K_t small, meaning that we have much more uncertainty in our measurement than in our prediction, then our correction term is small.

The correction to the covariance matrix is less obvious, but follows from similar logic.

We then repeat this process. A few iterations of this algorithm are depicted in Figure 11.3.

Algorithm 1 Kalman Filter

1: **Data:** $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
2: **Result:** (μ_t, Σ_t)
3: $\bar{\mu}_t = A_t\mu_{t-1} + B_t u_t$ ▷ Predict mean
4: $\bar{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$ ▷ Predict covar
5: $K_t = \bar{\Sigma}_t C_t^T (C_t\bar{\Sigma}_t C_t^T + Q_t)^{-1}$ ▷ Kalman gain
6: $\mu_t = \bar{\mu}_t K_t(z_t - C_t\bar{\mu}_t)$ ▷ Correct mean
7: $\Sigma_t = (I - C_t K_t)\bar{\Sigma}_t$ ▷ Correct covar
8: **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:

$$p(x_t \mid z_{1:t}, u_{1:t}) = \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}). \tag{11.4}$$

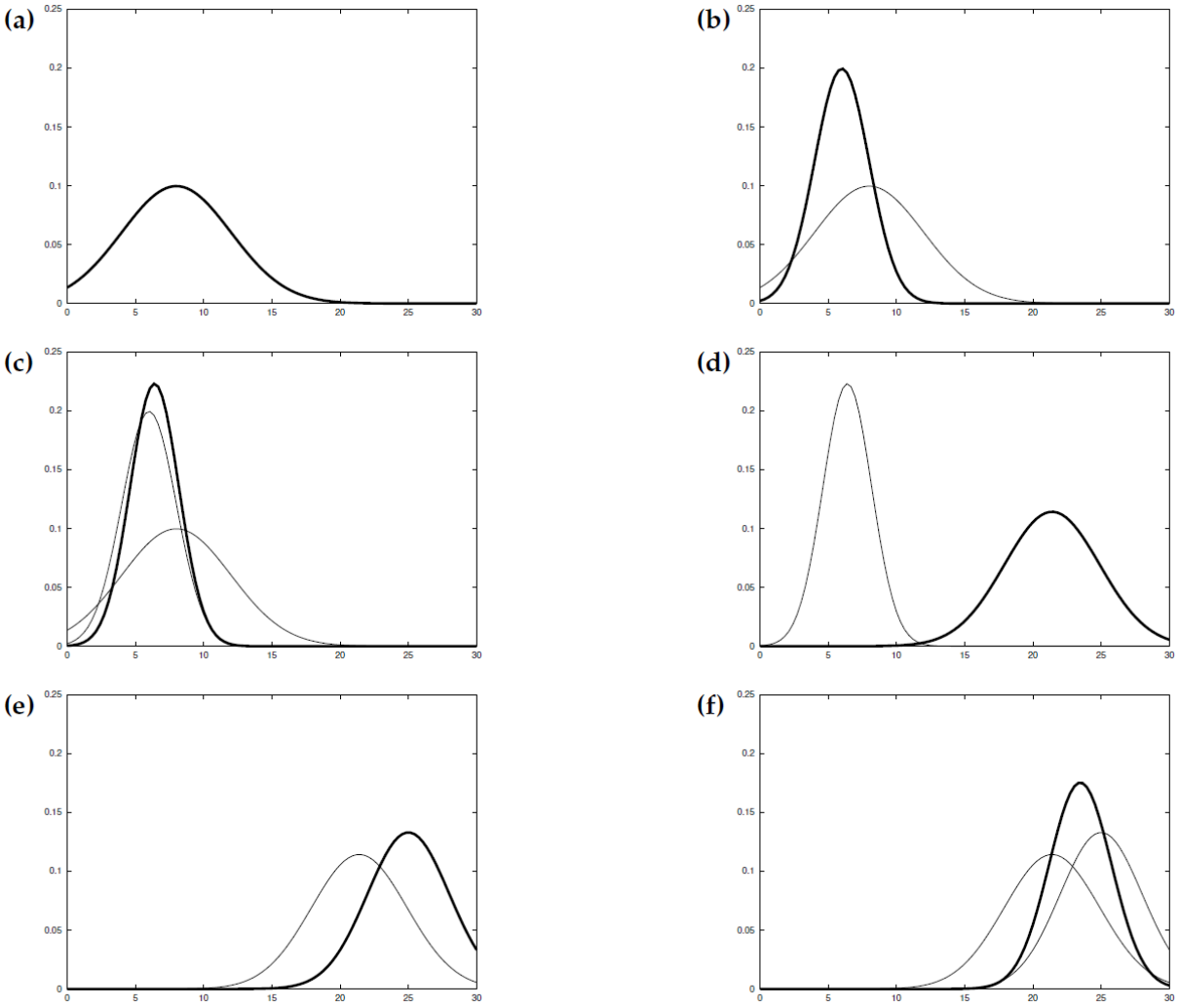


Figure 11.3: Visualization of the Kalman Filter Algorithm from [2]. Current step is in bold and previous data are in gray: (a) the initial belief distribution, (b) the measurement with uncertainty in bold and initial belief in gray, (c) updated belief in bold with previous belief and measurement in gray, (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.

Note that conditional independence gives the simplification

$$p(x_t \mid z_{1:t}, u_{1:t}) = \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}). \tag{11.5}$$

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:

$$\overline{bel}(x_t) = \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1} u_{1:t-1}) dx_{t-1} \tag{11.6}$$

which can also be expressed as

$$\overline{bel}(x_t) = \int p(x_t \mid x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1} \tag{11.7}$$

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 (11.7) can be simplified to

$$\overline{bel}(x_t) = \mathcal{N}(\bar{\mu}_t, \bar{\Sigma}_t) \tag{11.8}$$

where $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$ and $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$ from integrating the product of the Gaussian distributions. Thus, the prediction step can be summarized with (11.8). From (11.5), we have the following relationship between the posterior distribution and the predicted belief:

$$bel(x_t) = \eta p(z_t \mid x_t) \overline{bel}(x_t). \tag{11.9}$$

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

$$bel(x_t) = \mathcal{N}(\mu_t, \Sigma_t), \tag{11.10}$$

for which the equations for μ_t, and Σ_t can be expressed as in (11.3).

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.

In relaxing the linearity assumption to use the EKF, we now assume that the dynamics of the system are given by

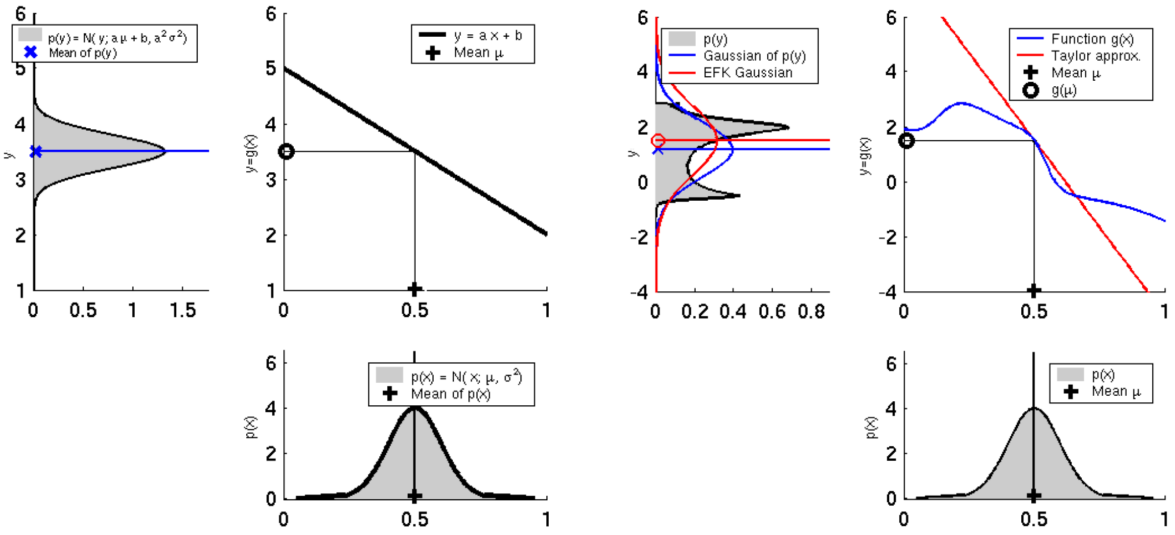
$$x_t = g(u_t, x_{t-1}) + \epsilon_t. \tag{11.11}$$

Furthermore, the measurement model is now given by

$$z_t = h(x_t) + \delta_t. \tag{11.12}$$

A key idea 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

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + J_g(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1}) \tag{11.13}$$



(a) Kalman Filter Linearity Assumption

(b) EKF Linearization

Figure 11.4: Comparison of (a) the linearity assumption of the standard Kalman Filter [1] and (b) its extension to nonlinear functions through linearization [1].

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

$$p(x_t \mid u_t, x_{t-1}) = \det(2\pi R_t)^{-1/2} \exp\left(-\frac{1}{2}\left[x_t - g(u_t, \mu_{t-1}) - G_t \cdot (x_{t-1} - \mu_{t-1})\right]^T R_t^{-1} \left[x_t - g(u_t, \mu_{t-1}) - G_t \cdot (x_{t-1} - \mu_{t-1})\right]\right) \tag{11.14}$$

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

$$h(x_t) = h(\bar{\mu}_t) + J_h(\bar{\mu}_t)(x_t - \bar{\mu}_t) \tag{11.15}$$

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

$$p(z_t \mid x_t) = \det(2\pi Q_t)^{-1/2} \exp\left(-\frac{1}{2}\left[z_t - h(\bar{\mu}_t) - H_t \cdot (x_t - \bar{\mu}_t)\right]^T Q_t^{-1} \left[z_t - h(\bar{\mu}_t) - H_t \cdot (x_t - \bar{\mu}_t)\right]\right) \tag{11.16}$$

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 11.4.

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 (11.1) are replaced by the following:

$$\begin{aligned} \bar{\mu}_t &= g(u_t, \mu_{t-1}) \\ \bar{\Sigma}_t &= G_t \Sigma_{t-1} G_t^T + R_t \end{aligned} \tag{11.17}$$

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

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \tag{11.18}$$

Then we update our belief function

$$\begin{aligned} \mu_t &= \bar{\mu}_t + K_t(z_t - h(\mu_t)) \\ \Sigma_t &= (I - K_t H_t)\bar{\Sigma}_t \end{aligned} \tag{11.19}$$

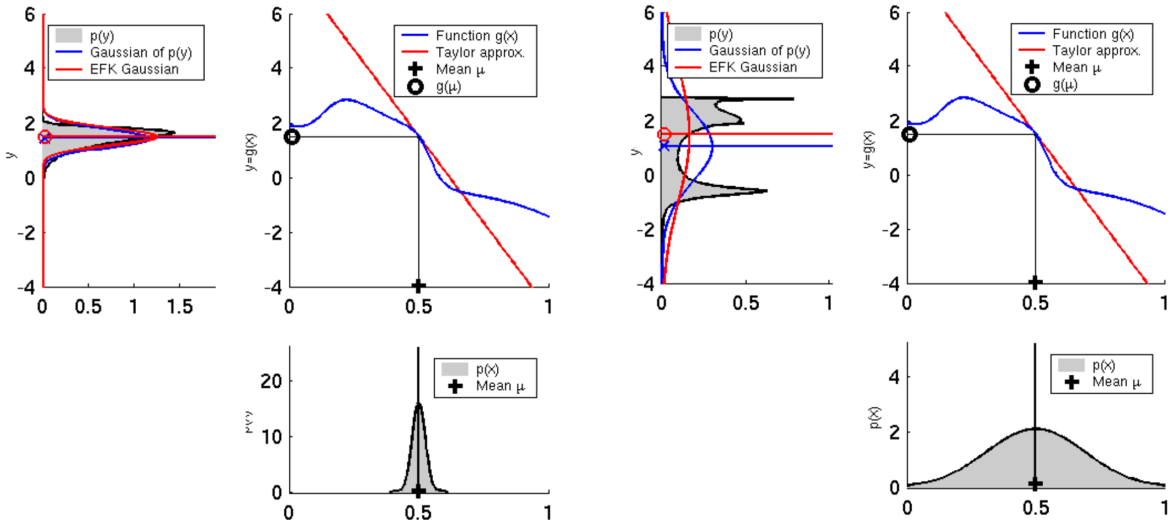
We then repeat this process.

Algorithm 2 Extended Kalman Filter

1: **Data:** $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
2: **Result:** (μ_t, Σ_t)
3: $\bar{\mu}_t = g(u_t, \mu_{t-1})$ ▷ Predict mean
4: $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$ ▷ Predict covar
5: $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$ ▷ Kalman gain
6: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\mu_t))$ ▷ Correct mean
7: $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$ ▷ Correct covar
8: **Result:** (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 11.5a, the EKF provides an accurate estimate of the true state. If there is high variance in the prior distribution, as in Figure 11.5b, the 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 11.5: EKF performance for different levels of uncertainty [1].

11.2.2.3 More on EKF

These following resources provide more insights into this topic.

Robot Localization and Kalman Filters Rudy Negenborn.

Implementation of extended Kalman filter-based simultaneous localization and mapping: a point feature approach Manigandan Nagarajan SanthanakrishnanE, John Bosco Balaguru RayappanRamkumar Kannan.

Locating a two-wheel robot using extended kalman filter by Javad Zolghadr and Yuanli Cai. (This can be found at Google)

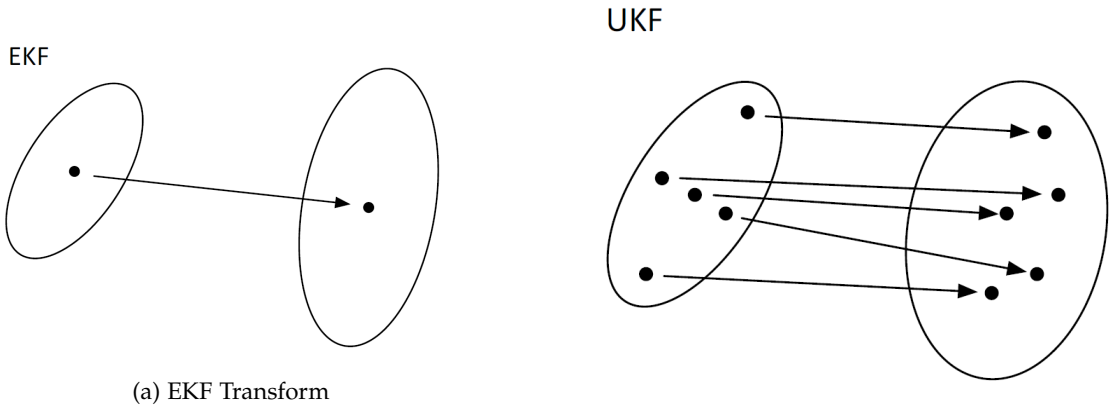
Extended Kalman Filter Tutorial by Gabriel A. Terejanu, University at Buffalo

Using the Kalman filter Extended Kalman filter by Doz. G. Bleser and Prof. Stricker

(Summary of extended kalman filter can be found in this one)

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 computing 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 11.6.



(a) EKF Transform

(b) UKF Transform

Figure 11.6: 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.

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.

$$\begin{aligned} \mathcal{X}^{[0]} &= \mu \\ \mathcal{X}^{[i]} &= \mu + \left(\sqrt{(n+\lambda)\Sigma}\right)_i \qquad \text{for } i = 1, \ldots, n \\ \mathcal{X}^{[i]} &= \mu - \left(\sqrt{(n+\lambda)\Sigma}\right)_{i-n} \qquad \text{for } i = n+1, \ldots, 2n, \end{aligned} \tag{11.20}$$

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.

$$\begin{aligned} w_m^{[0]} &= \frac{\lambda}{n+\lambda}, \qquad w_c^{[0]} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \\ w_m^{[i]} &= w_c^{[i]} = \frac{1}{2(n+\lambda)} \qquad \text{for } i = 1, \ldots, 2n. \end{aligned} \tag{11.21}$$

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

$$\mathcal{Y}^{[i]} = g(\mathcal{X}^{[i]}) \tag{11.22}$$

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

$$\begin{aligned}\mu' &= \sum_{i=0}^{2n} w_m^{[i]} \mathcal{Y}^{[i]} \\ \Sigma' &= \sum_{i=0}^{2n} w_c^{[i]} (\mathcal{Y}^{[i]} - \mu')(\mathcal{Y}^{[i]} - \mu')^T\end{aligned} \tag{11.23}$$

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:

$$\bar{\mathcal{X}}_t^* = g(u_t, \mathcal{X}_{t-1}) \tag{11.24}$$

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 (11.21):

$$\begin{aligned}\bar{\mu}_t &= \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_t^{*[i]} \\ \bar{\Sigma}_t &= \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t\right) \left(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t\right)^T + R_t\end{aligned} \tag{11.25}$$

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

$$\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t), \tag{11.26}$$

from which the predicted observation can be computed:

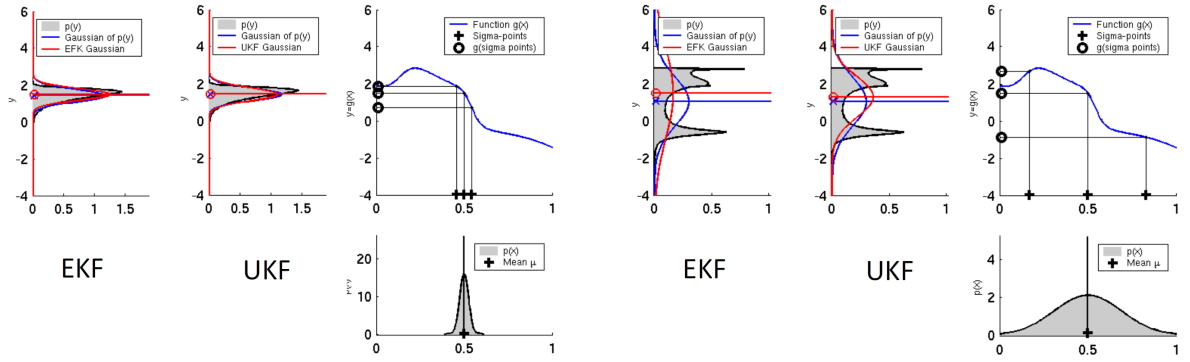
$$\begin{aligned}\hat{z}_t &= \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]} \\ S_t &= \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right) \left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^T + Q_t \\ \bar{\Sigma}_t^{x,z} &= \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t\right) \left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^T\end{aligned} \tag{11.27}$$

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:

$$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1} \tag{11.28}$$

Then we can update the belief function:

$$\begin{aligned}\mu_t &= \bar{\mu}_t + K_t(z_t - \hat{z}_t) \\ \Sigma_t &= \bar{\Sigma}_t - K_t S_t K_t^T\end{aligned} \tag{11.29}$$



(a) UKF with Low Variance

(b) UKF with High Variance

Figure 11.7: The performance of the UKF compared to the EKF for different levels of uncertainty [1].

Algorithm 3 Unscented Kalman Filter

Data: $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
Result: (μ_t, Σ_t)
$\mathcal{X}_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \gamma\sqrt{\Sigma_{t-1}} \quad \mu_{t-1} - \gamma\sqrt{\Sigma_{t-1}})$ ▷ Form sigma points (11.20)
$\bar{\mathcal{X}}_t^* = g(u_t, \mathcal{X}_{t-1})$ ▷ 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 covar
$\bar{\mathcal{X}}_t = \left(\bar{\mu}_t \quad \bar{\mu}_t + \gamma\sqrt{\bar{\Sigma}_t} \quad \bar{\mu}_t - \gamma\sqrt{\bar{\Sigma}_t}\right)$ ▷ 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-covar
$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$ ▷ 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 covar
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 11.7 shows, the UKF and EKF perform very similarly with low variance, but the UKF outperforms the EKF with high variance.

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,

$$\text{dom}(X_t) = x_{1,t} \cup x_{2,t} \cup ...x_{k,t} \tag{11.30}$$

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.

$$p(x_t) = \frac{p_{k,t}}{|x_{k,t}|}, \quad x_t \in x_{k,t} \tag{11.31}$$

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 $\hat{x}_{k,t}$,

$$\hat{x}_{k,t} = |x_{k,t}|^{-1} \int_{x_{k,t}} x_t dx_t \tag{11.32}$$

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,

$$p(x_{k,t} \mid u_t, x_{i,t-1}) = \eta |x_{k,t}| p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1}) \tag{11.33}$$

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 11.8.

11.3.2 Particle Filter

11.3.2.1 Approximating with Particles

Particle filters differ from Kalman filters or any parametric filters in the sense that it does not assume any type of belief distribution. Instead, particle filters can approximate any PDF using a large numbers of "particles". To approximate a PDF with particles, particles are generated based on the relative likelihood of these samples directly. For example, given a random variable X with PDF $Pr[X = 0] = 0.1$ and $Pr[X = 1] = 0.9$, 10% of the particles will have value 0 and 90% of the particles will have value 1.

11.3.2.2 Particle Filter Algorithm

This intuition can be formally shown as the following: let a particle filter approximates the posterior distribution with a finite number of particles, denoted as

$$\mathcal{X}_t = x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]} \tag{11.34}$$

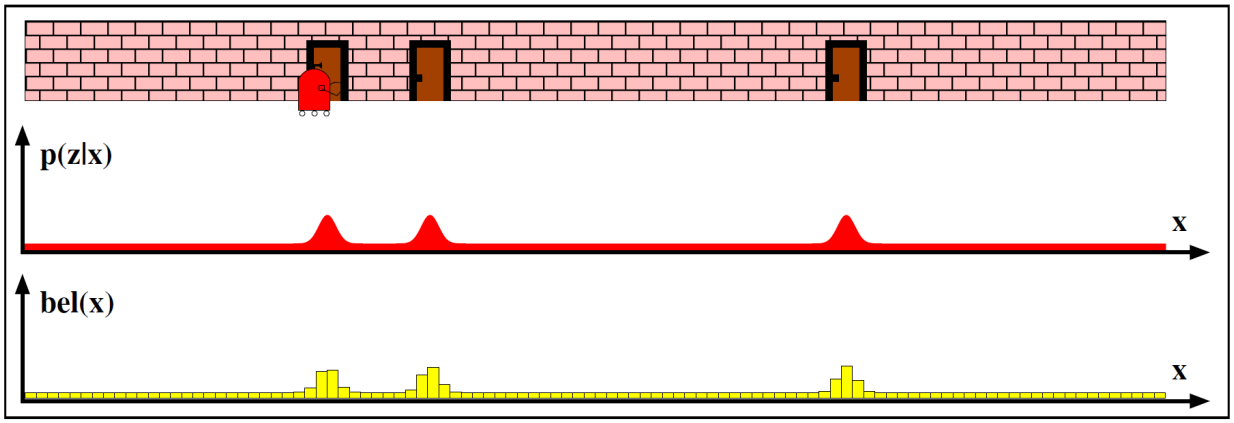


Figure 11.8: Histogram representation of belief from initial sensor measurement from [2].

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: assuming we have M particles, given $\mathcal{X}_{t-1}$, we construct the next particle set $\mathcal{X}_t$ by first construct a predicted particle set, then use the measurement result to correct the belief state. First, a predicted particle set is constructed by drawing samples from the current belief state $p(x_t|u_t, x_{t-1}^{[m]})$. Meanwhile, the probability we observe a particular measurement conditional on the state, $w_t^{[m]}$, is also calculated. Next, the correction step, often referred to as resampling with replacement, incorporates our measurements into the calculation. A corrected set of particles are generated by drawing a particle i from the predicted particle set with probability proportional to $w_t^{[i]}$.

After resampling, particles are distributed as

$$bel(X_t) = \eta P(z_t \mid x_t^{[m]}) \overline{bel}(X_t) \tag{11.35}$$

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 11.9.

Note that after several resampling steps, particles will often collapsed into very few states. This situation is called particle deprivation. To correct this, we can inject n random particles into the corrected particle set after each iteration.

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 11.10. Similarly, the influence of the map (door locations) can be seen in the measurement model $p(z \mid x)$ in Figures 11.8, 11.9b, and 11.9d.

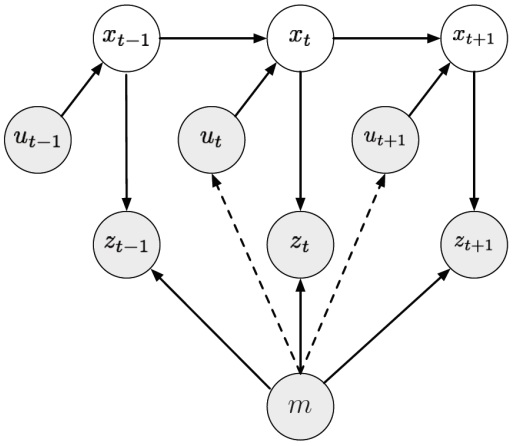


Figure 11.10: Hidden Markov Model for robot localization, where a map informs sensor measurements and control actions [1].

11.4.1 A Taxonomy of Localization Problems

11.4.1.1 Local vs Global Localization

The type of localization algorithm used is heavily depending on the information available at initialization and during a robot's operation.

In *position tracking*, the initial pose of the robot is known with certainty. This approach assumes that as the robot moves in its environment the pose error remains relatively small and concentrated near the robot's true pose. The EKF works well for this problem because it approximates the error using a unimodal (i.e. Gaussian) distribution.

On the other hand, *global localization* assumes the initial pose is unknown. In this scenario, the belief distribution is inherently multi-modal as the robot attempts to localize itself. In this case, non-parametric filters like histogram or particle filters are preferable.

A final case is that of the *kidnapped robot*. In this case 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 non-parametric, multi-hypothesis filters.

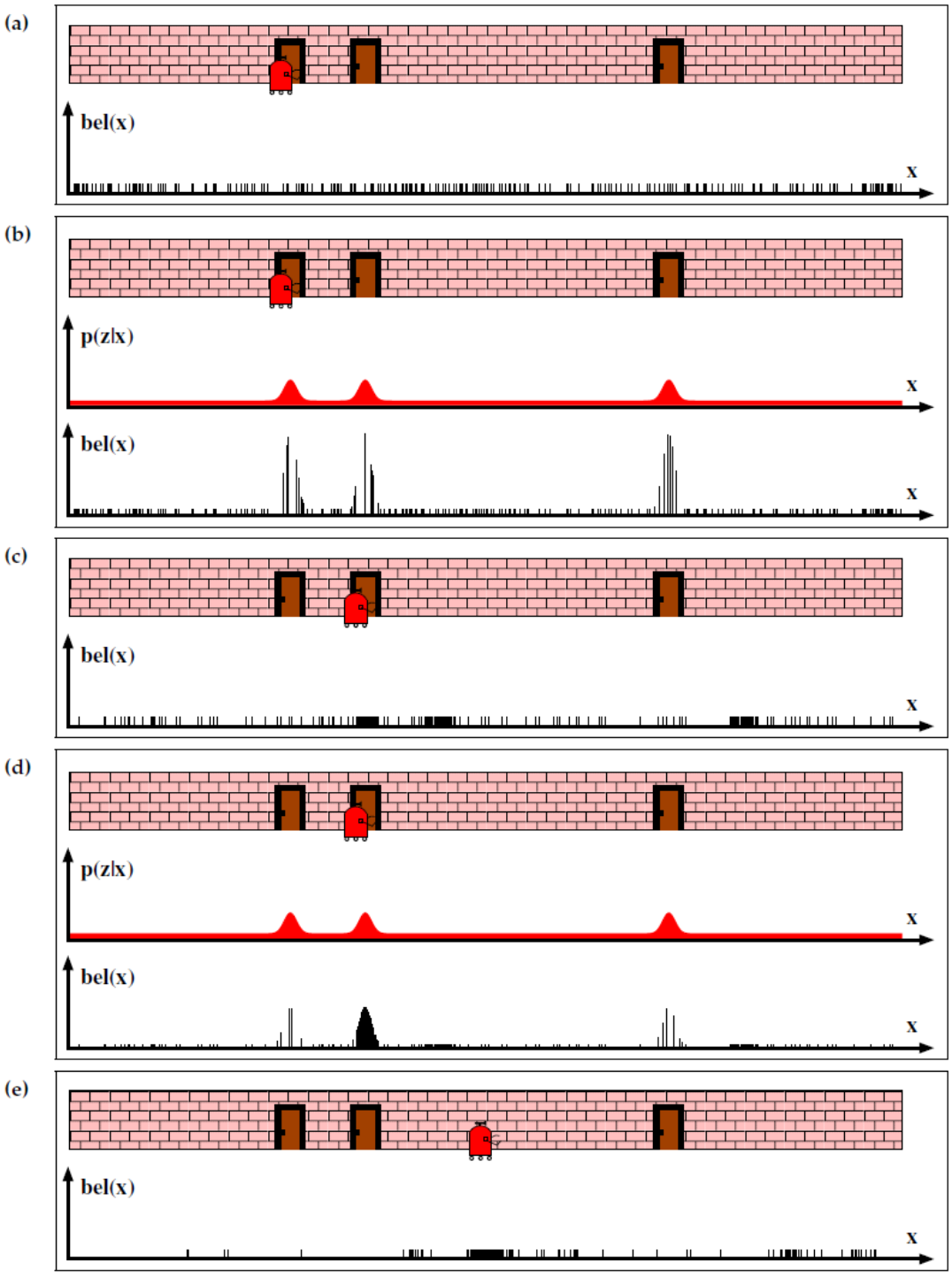


Figure 11.9: Particle filter used for robot localization from [2]: (a) Initial particles sampled uniformly over entire state space, (b) same set of particles from (a) after importance weighting with initial sensor measurement, (c) resampled particles from weighted distribution after motion, (d) importance weighting of new particle set with new sensor measurement, and (e) resampled particle set after further motion.

11.4.1.2 Static vs Dynamic Environments

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 to include the movement of dynamic entities, or by filtering the sensor data to remove the effects of environment dynamics.

11.4.1.3 Passive vs Active Approaches

Localization problems can be further split into passive and active depending on whether the localization algorithm can control the motion of the robot. *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. However, since active approaches require control over the robot, they tend to be insufficient in practice. They are often combined with passive techniques that operate when the robot is performing tasks other than localization.

11.4.1.4 Single Robot vs Multi-Robot

A final consideration in the localization problem is the number of robots involved. *Single-robot localization* is the most commonly studied and utilized approach. Only a single robot is used in this scheme, and this approach offers the advantage of having all data collected in a single platform.

Multi-robot localization occurs when a team of robots share information in such a way that one robot's belief can be used to influence another robot's belief if the relative location between robots is known.

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.

Contributors

Winter 2019: Spencer Diehl, Jennifer Lin, Georgia Murray, Akshay Rajagopal, Mimi Su, JQ Zhang
Winter 2018: Arthur Binstein, Anqi Fu, Trevor Halsted, Scott Hemley, Alexander Hobbs, Jialong Wang