16 Nonparametric Filters

Last lecture, we discussed how to perform state estimation using parametric filters. Parametric filters refer to filters that derive posterior based on probabilistic distribution described by parameters. The primary example of parametric filter was Kalman Filter that uses Gaussian distribution. Today, we discuss nonparametric filters that do not make any assumptions on the robot’s belief distribution. In a way, nonparametric filters are extension of Unscented Kalman Filter (UKF): robot’s belief distribution is approximately with a finite number of samples. The accuracy of the representation depends on the number of values used, which results in a trade-off between expressiveness and computational burden. Nonparametric filters are more robust to nonlinearities and discontinuities in the inference space, but also tend to be more complicated and time-consuming to implement.

16.1 Histogram Filter

Histogram filters is one of the most intuitive nonparametric filters. We describe our belief with a discrete approximation of a continuous distribution of beliefs. An example of such belief is \( X = \{ \text{head, tail} \} \) or \( X = \{ \text{red, yellow, blue} \} \). (For a contrast, recall Kalman filter and Gaussian distribution works only on continuous random variables.) Formerly, our continuous belief \( X \) is decomposed to finitely many bins,

\[
\text{dom}(X_t) = x_{1,t} \cup x_{2,t} \cup ... x_{k,t}.
\]

Each region \( x_{k,t} \) is assigned a probability \( p_{k,t} \), which is approximated in a piece-wise manner, assuming that all events that belong to the same bin are equally likely. We can express probability using uniform distribution:

\[
p(x_t) = \frac{p_{k,t}}{|x_{k,t}|}, \quad x_t \in x_{k,t}.
\]
Data: \( \{p_{k,t-1}\}, u_t, z_t \)
Result: \( \{p_{k,t}\} \)

\begin{algorithm}
\begin{algorithmic}
\State \textbf{foreach} \( k \) \textbf{do}
\State \( \bar{p}_{k,t} = \sum_i p(X_t = x_k | u_t, X_{t-1} = x_i) p_{i,t-1}; \)
\State \( p_{k,t} = \eta p(z_t | X_t = x_k) \bar{p}_{k,t}; \)
\EndFor
\State \textbf{Return} \( \{p_{k,t}\} \)
\end{algorithmic}
\end{algorithm}

Figure 1: Histogram representation of belief from initial sensor measurement from [TBF05]

Our system dynamic \( p(x_t | u_t, x_{t-1}) \) and measurement models \( p(z_t | x_t) \) are also discretized. Probability distribution of each bin can be represented with the mean state \( \hat{x}_{k,t} \):

\[
\hat{x}_{k,t} = \left| x_{k,t} \right|^{-1} \int_{x_{k,t}} x_t dx_t.
\] (3)

Figure 1 describes robot’s discretized belief. Given a state \( x_{k,t} \), the conditional probability \( p(z_t | x_{k,t}) \) is the probability of that measurement conditional on the representative state in that region, \( p(z_t | \hat{x}_{k,t}) \). A similar process is used to approximate the state transition probabilities by applying Bayes’ law,

\[
p(x_{k,t} | u_t, x_{i,t-1}) = \eta |x_{k,t}| p(\hat{x}_{k,t} | u_t, \hat{x}_{i,t-1})
\] (4)

Finally, we execute a discrete Bayes’ filter on the discretized probabilities to estimate the full belief distribution. Implementation details of histogram filter is shown in ??.
16.2 Particle Filter

The particle filter is an alternative nonparametric implementation of the Bayes filter. Just like histogram filters, particle filters approximate the posterior by a finite number of parameters. However, they differ in the way these parameters are generated, and in which they populate the state space. The key idea of the particle filter is to represent the posterior $\text{bel}(x_t)$ by a set of random state samples drawn from this posterior. Instead of representing the distribution by a parametric form (the exponential function that defines the density of a normal distribution), particle filters represent a distribution by a set of samples drawn from this distribution. Such a representation is approximate, but it is nonparametric, and therefore can represent a much broader space of distributions than, for example, Gaussians.

In particle filters, the samples of a posterior distribution are called particles and are denoted $X_t = x_t^1, x_t^2, ..., x_t^M$ (5).

Each particle $x_t^m$ (with $1 \leq m \leq M$) is a concrete instantiation of the state at time $t$, that is, a hypothesis as to what the true world state may be at time $t$. Here $M$ denotes the number of particles in the particle set $X_t$. In practice, the number of particles $M$ is often a large number, e.g., $M = 1000$. In some implementations $M$ is a function of $t$ or of other quantities related to the belief $\text{bel}(x_t)$.

The intuition behind particle filters is to approximate the belief $\text{bel}(x_t)$ by the set of particles $X_t$. Ideally, the likelihood for a state hypothesis $x_t$ to be included in the particle set $X_t$ shall be proportional to its Bayes filter posterior $\text{bel}(x_t)$:

$$x_t^m \sim p(x_t \mid z_{1:t}, u_{1:t})$$ (6)

As a consequence of (6), the denser a subregion of the state space is populated by samples, the more likely it is that the true state falls into this region. As we will discuss below, the property (6) holds only asymptotically for $M \to \infty$ for the standard particle filter algorithm. For finite $M$, particles are drawn from a slightly different distribution. In practice, this difference is negligible as long as the number of particles is not too small (e.g., $M \geq 100$).

Just like all other Bayes filter algorithms discussed thus far, the particle filter algorithm constructs the belief $\text{bel}(x_t)$ recursively from the belief $\text{bel}(x_{t-1})$ one time step earlier. Since beliefs are represented by sets of particles, this means that particle filters construct the particle set $X_t$ recursively from the set $X_{t-1}$. The most basic variant of the particle filter algorithm is stated in Figure 2. The input of this algorithm is the particle set $X_{t-1}$, along with the most recent control $u_t$ and the most recent measurement $z_t$. The algorithm then first constructs a temporary particle set $\bar{X}$ which is reminiscent but not equivalent) to the belief $\bar{\text{bel}}(x_t)$. It does this by systematically processing each particle $x_{t-1}^m$ in the input particle set $X_{t-1}$ as follows.

---

1Most of this section is a direct excerpt from [TBF05].
1. Line 4 generates a hypothetical state \( x_t \) for time \( t \) based on the particle \( x_{t-1} \) and the control \( u_t \). The resulting sample is indexed by \( m \), indicating that it is generated from the \( m \)-th particle in \( X_{t-1} \). This step involves sampling from the next state distribution \( p(x_t | u_t, x_{t-1}) \). To implement this step, one needs to be able to sample from \( p(x_t | u_t, x_{t-1}) \). The ability to sample from the state transition probability is not given for arbitrary distributions \( p(x_t | u_t, x_{t-1}) \). The set of particles resulting from iterating Step 4 \( M \) times is the filter’s representation of \( \tilde{\text{bel}}(x_t) \).

2. Line 5 calculates for each particle \( x_t[m] \) the so-called importance factor, denoted \( w_t[m] \). Importance factors are used to incorporate the measurement \( z_t \) into the particle set. The importance, thus, is the probability of the measurement \( z_t \) under the particle \( x_t[m] \), that is, \( w_t[m] = p(z_t | x_t[m]) \). If we interpret \( w_t[m] \) as the weight of a particle, the set of weighted particles represents (in approximation) the Bayes filter posterior \( \tilde{\text{bel}}(x_t) \).

3. The real “trick” of the particle filter algorithm occurs in Lines 8 through 11 in Figure 2. These lines implemented what is known as resampling or importance resampling. The algorithm draws with replacement \( M \) particles from the temporary set \( \tilde{X}_t \). The probability of drawing each particle is given by its importance weight. Resampling transforms a particle set of \( M \) particles into another particle set of the same size. By incorporating the importance weights into the resampling process, the distribution of the particles change: whereas before the resampling step, they were distribution according to \( \tilde{\text{bel}}(x_t) \), after the resampling they are distributed (approximately) according to the posterior \( \text{bel}(x_t) = \eta p(z_t | x_t[m]) \tilde{\text{bel}}(x_t) \). In fact, the resulting sample set usually possesses many duplicates, since particles are drawn with replacement. More important are the particles that are not contained in \( X_t \): those tend to be the particles with lower importance weights.

A few iterations of the particle filter for robot localization are shown in Figure 3.
Figure 3: Particle filter used for robot localization from [TBF05]: (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.
The resampling step has the important function to force particles back to the posterior \( \text{bel}(x_t) \). In fact, an alternative (and usually inferior) version of the particle filter would never resample, but instead would maintain for each particle an importance weight that is initialized by 1 and updated multiplicatively:

\[
    w_t^{[m]} = p(z_t | x_t^{[m]})w_{t-1}^{[m]}
\]

Such a particle filter algorithm would still approximate the posterior, but many of its particles would end up in regions of low posterior probability. As a result, it would require many more particles; how many depends on the shape of the posterior. The resampling step is a probabilistic implementation of the Darwinian idea of survival of the fittest: It refocuses the particle set to regions in state space with high posterior probability. By doing so, it focuses the computational resources of the filter algorithm to regions in the state space where they matter the most.

**References**