# Introduction to Particle-based Filtering for State Estimation in Robotics

Josiah Hanna

## 1 Introduction

In week 3, we introduced the Bayes filter as an optimal method for recursively computing a robot's belief based on the history $z_{1:t}$ and $u_{1:t}$. Bayes filter itself is intractable for large and continuous state-spaces and so, in week 4, we introduced the Kalman filter and the extended Kalman filter which can be applied in continuous state-spaces. The limitation of the Kalman filter class of state estimators is that the state transition and observation function must have Gaussian distributions and be linear (for Kalman filters) or approximately linear (for the EKF). We will now introduce a class of methods that can approximate $\texttt{bel}(x_t)$ in continuous state spaces and for complex, non-linear probability distributions over next states and observations. This is the class of particle-based methods.

The high-level idea of the family of particle-based methods is to represent the robot's $\texttt{bel}(x_t)$ by a finite collection of weighted *particles*. Each particle represents a possible value for the state $x_t$. For example, if the state is defined as the pose of the robot, then each particle corresponds to a possible pose. Let $x_t^i$ represent the $i^{\text{th}}$ particle and $w_i$ represent its weight. Weights will be defined such that the particles for states that should be very likely under $\texttt{bel}(x_t)$ have a large weight and particles for unlikely states have a small weight. At each time-step, $t$, a particle-based algorithm updates the weights (and possibly the set of particles) using the latest control and sensor data, $u_t$ and $z_t$.

## 2 Normalized Importance Sampling

The first particle-based algorithm we will encounter is the normalized importance sampling (NIS) filter. The NIS filter uses a set of $N$ particles. For example, let's consider a robot moving around in a square room where its state is fully described by its pose, $(x, y, \theta)$. Each particle, $x_i$, is then defined as a pose $(x_i, y_i, \theta_i)$ and the $N$ particles are scattered across different possibilities for the robot's pose. Note that there are an infinite number of possible values for the robot's pose (since $x$, $y$, $\theta$ are real-valued) but only a finite number of particles. Each particle is given an initial weight, $w_i$. By default, the weight is $w_i \leftarrow \frac{1}{N}$ so that initially no particle is more likely than the others. Of course, if prior knowledge means that some states are more likely than others then larger weight can be given to matching particles.

At each time-step, $t$, the algorithm receives the robot's control, $u_t$, and new sensor observation, $z_t$. The NIS filter takes the following steps:

1. For each particle, $x_{t-1}^i$, sample $x_t^i \sim p(\cdot|x_{t-1}^i, u_t)$.
2. For each particle, update the weight $w_i \leftarrow w_i \cdot p(z_t|x_t^i)$.
3. Normalize the weights such that $\sum_i w_i = 1$.

Let's consider what the NIS filter is doing. First, the state value associated with each particle evolves stochastically according to the state transition function $p(x_t|x_{t-1}, u_t)$. Second, the weight on each particle is updated based on how well the state value matches the observed $z_t$. If $p(z_t|x_t^i)$ is small relative to other states then the weight (once normalized) will decrease and will conversely increase if $p(z_t|x_t^i)$ is larger.

An important strength of the NIS (and other particle-based) filters is that the state transition model does not have to be known in closed form, provided we have a means to sample from it. This is convenient as it is often easier to generate samples than it is to write out a full probability density function.

# 3 Extracting a State Estimate from the Particles

Particle-based methods represent the robot's belief distribution as a discrete set of particles. However, in robotics applications, we often want to compute an estimate of the most likely state and use this for decision-making. One way to do this would be to just return the particle with maximum weight:

$$\hat{x}_t = x_t^i \text{ with } i \leftarrow \arg\max_i w_i.$$

Doing so will restrict the filter to only returning one of the particle values, even if multiple particles have high weight. A better alternative is to return a weighted average:

$$\hat{x_t} = \sum_i w_i x_t^i.$$

Unlikely particles will receive near-zero weight and won't affect the result while other particles with high weight will affect the average. In the extreme case where only a single particle has high weight ($w_i \approx 1$) then the weighted average is the same as just returning the most likely particle.

A disadvantage of using a weighted average is that the belief could be multi-modal, resulting in the weighted average returning a state value that is very unlikely. For example, in the room example, if the NIS filter has equal weight on two particles representing distinct locations then the weighted average returns a location that is halfway between these location. That location itself may be very unlikely and yet the weighted average returns it because it collapses the multi-modal belief into a single point. A possible solution is to run a clustering algorithm on the particles, compute a weighted average for each cluster, and then return a set of candidate states.

# 4 The Particle Filter

A main drawback of the NIS filter is that the weights will tend toward either 0 or 1. Any particle with $w_i \approx 0$ means that we are keeping track of a very unlikely state possibility. This is wasteful! We should instead try to repurpose such low weight particles so that they correspond to more likely states.

The particle filter accomplishes this goal by resampling the particles according to their weight. The particle filter follows the same steps as the NIS filter but adds two additional steps after normalizing the weights:

1. Define a probability distribution, $q$, over the particles such that particle $i$ has probability $w_i$. Sample $N$ particles with replacement from $q$. Discard the old particles.

2. Set the weight for each particle to $w_i \leftarrow 1/N$.

With these additional steps, unlikely states tend to not be resampled and are instead replaced with particles representing more likely states. Overtime, the set of particles will eventually only contain particles representing more likely states under $p(x_t|z_{1:t}, u_{1:t})$.