Statistical Techniques in Robotics (16-831, F14) Lecture#5 (Tuesday September 9)

# Good, Bad, and Ugly of Particle Filters

*Lecturer: Drew Bagnell* *Scribes: Greg Seyfarth, Zachary Batts*[1]

This lecture is about the advantages of particle filters, issues that may arise when implementing particle filters, and potential solutions to these issues.

## 1 Review of particle filters

Given the previous set of particles $\chi_{t-1}$, $u_t$, and $z_t$, we perform each of the following steps.

1. Apply the forward motion model to all particles. This means that, for each $i$, we sample $\chi_t^i$ from $p(x_t|x_{t-1}^i, u_t)$.
2. Update the weights for each particle, which means we set $\tilde{w}^i = p(z_t|x_t^i)$ for each i.
3. Normalize the weights. Set $w_i = \frac{\tilde{w^i}}{\Sigma_i \tilde{w^i}}$ for each i.
4. Perform the resampling step. We resample each particle according to the weights $w^i$.
5. Reset each particle's weight to $\frac{1}{n}$ and return the set of particles.

## 2 The Good

Particle filters have a number of advantages which have led to them being very popular in modern robotics.

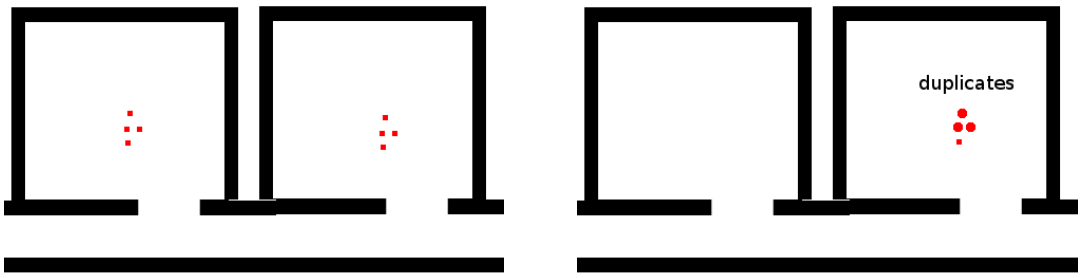
1. Since the particles approximate the posterior belief, we can answer any question we can pose in terms of an expectation.
2. Works for any observation model and any motion model we can design.
3. Particle filters scale well; they are 'embarassingly parallelizable'.
4. They work for high dimensional systems as they are independent of the size of the system.
5. The algorithm is relatively easy to implement.

[1]Some content adapted from previous scribes: David Fouhey

# 3 The Bad

## 3.1 Lack of diversity

Over time, especially with uninformative sensor readings, samples tend to congregate. One metric to benchmark a particle filter is how long it takes for particles to congregate. Take the following example.




(a) Two rooms with equal probability (b) The same filter, after repeated resampling

Figure 1: A problem with assuming the independence of the observations and using naïve sampling. Without gaining any new information, the particle filter converges on one of the rooms.

**Example.** There are two rooms, and the robot is unsure about which room it is in. The sensor readings show equal probability of being in either room. We start out with N particles equally distributed between the two rooms. Over time, the particles will converge to one room, without getting any new sensor data.

**Why does this happen?** Particle filters are non-deterministic. At the resampling step, one room is likely to gain particles. Once this happens, at the next resampling step, that room is more likely to gain particles. Eventually, all the particles converge to one state. Once a state loses particles, it cannot regain them without motion.

**Practical Fixes:**

1. Skip the resampling step if you detect you are not receiving new information.You might detect this by looking at how $\max(\{w_i\})/\min(\{w_i\})$ or entropy of the weights changes.
2. Use low variance resampling (see Fig. 2) (AKA "Stochastic Universal Sampler" in genetic algorithms domain). In one dimension (AKA the "dartboard" example), this method works as follows. A single random number $r$ is drawn, and particles are selected at every $\frac{1}{N}$ units, offset from $r$. This has the desirable property that any particle with normalized weight greater than $\frac{1}{N}$ is guaranteed to be drawn, which eliminates our "Lack of Diversity" problem.
3. Add more particles. This will increase the number of resamples the particles need to congregate to one state. Adding more particles usually helps with most problems.
4. Inject uniform samples. This is an unprincipled hack, and should be used as a last resort.

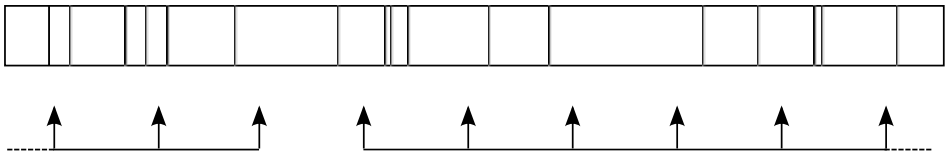


Figure 2: Low Variance Resampling; in practice, this approach is preferred over the naïve approach.

### 3.2 Precise Observation Models may make the PF fail

Consider a very accurate sensor with an observation model similar to that in Fig. 3. If we sample uniformly along $x_t$ (i.e after we push particles through a motion model), it is unlikely that any of the sames will fall under the slim peak of the model. In fact, it may be more likely for a particle to fall wider under the second, wider, peak.

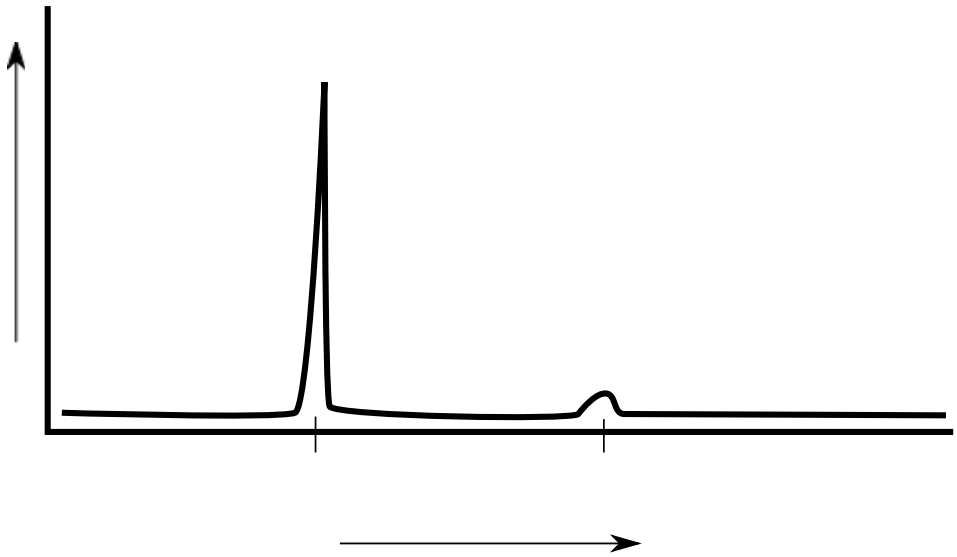


Figure 3: A very good observation model. Counterintuitively, this reliabiilty may pose problems for a vanilla implementation of a particle filter.

**Fixes:** There are a number of fixes for this problem, including more principled approaches and unprincipled and less effective hacks.

1. **Sample from a better model than the motion model:** the underlying problem is that the the proposal distribution (motion model) does not match the distribution that we want to sample from (observation model). The best, and most principled solution is to change the distribution. For instance, if one can sample directly from the observation model, one should do so. This is generally known as a dual particle filter.

2. **Rao-Blackwellization:** Solve as much of the problem analytically to reduce the dimensionality of your resampling step.

3. **Squash the observation model:** take all of the probabilities to some power $1/p$ less than 1. When this is done, $p$ observations count as 1 observation.

4. **More particles:** Sample more particles to ensure that that the peak is sampled (although this is unsatisfactory, since it doesn't guarantee a particle will land under the desired peak).

5. **Pretend your observation model is worse:** convolve the observation model with a Gaussian, and pretend that the resulting much less confident observation model is the actual model. This is a last resort, as it throws away a lot of valuable data.

### 3.3 Measuring Particle Filter Performance is Difficult

There is no convenient way of relating accuracy to number of particles. Similarly, particle filters offer no measure of confidence in their readings.

### 3.4 Particle Filters are Expensive Computationally

Despite being scalable (parallelizable), a good particle filter still requires a LOT of particles.

**Fix:** If your distribution is unimodal, it is a good idea to use a Kalman filter instead.

### 3.5 Particle Filters are Non-Deterministic

For the same input, a particle filter can produce different outputs, which makes them difficult to predict and debug.

**Fix:** Use a fixed random seed, which will produce consistent results.

## 4 The Ugly (Common Misconceptions)

1. **Particle Filter = "PF's are sample from motion model, weight by observation model."** is wrong. You don't need to sample from the motion model, and in practice you often don't. Sometimes, for example, you might sample directly from the observation model.

2. **PFs are for localization only** is wrong. You can apply particle filters to any state estimation problem.

3. **PFs are anything that samples** is wrong. NIS uses samples, but does not resample. Another counter-example is a Gibbs filter.