Sequential Monte Carlo estimations in Robotics

Nagesh Adluru

1 Sequential Monte Carlo estimations or in other words PARTICLE FILTERS

BASIC IDEA: Instead of heuristically assuming features about distributions and tracking them simulate distributions using randomly drawn samples (**particles**) from the distributions.

The drawn particles can be used to for a functional estimate $(E(f(x)))$ as:

$$E(f(x)) = \int_x f(x)p(x|y)dx \tag{1}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) \text{ where } x \text{ i.i.d from } p(x|y) \tag{2}$$

As $N \to \infty$ the simulation converges to true estimate.

ADVANTAGE: Gives power to filter non-linear, non-Gaussian without working out complicated analytical math governing the processes and gives us computational feasibility.

BIGGEST HURDLE: Drawing samples from a distribution and trying to estimate it is a chicken-egg problem! So we constantly *have* to iterate between updating the distribution and drawing samples and this can go wrong in several problem instances. The idea can be engineered based on an application.

BAYESIAN UPDATES: Almost always the update procedure in filtering techniques rests on Bayes rule.

2 Robot pose learning (localization)

Pose of a robot is it's position and it's heading direction (x, y, θ). From now on pose is represented by using only x.

The online learning of the pose is posed as the following filtering problem:

$$p(x_t|z_{1:t}, u_{1:t}, m) \tag{3}$$

where

- $u_{1:t}$ is the sequence of odometry measurements.
- $z_{1:t}$ is the sequence of range measurements.
- m is the map of the environment the robot is in.

Using Bayes rule and assumptions (1st order Markov process, observational independence) to figure out the updates:

$$\begin{aligned} p(x_t|z_{1:t}, u_{1:t}, m) &= p(x_t|z_{1:t-1}, u_{1:t}, m, z_t) \\ &= \frac{p(z_t|x_t, z_{1:t-1}, u_{1:t}, m)p(x_t|z_{1:t-1}, u_{1:t}, m)}{p(z_t|z_{1:t-1}, u_{1:t}, m)} \\ &= \eta p(z_t|x_t, m)\int_{x_{t-1}} p(x_t|x_{t-1}, u_t, m)p(x_{t-1}|z_{1:t-1}, u_{1:t-1}, m)dx_{t-1} \end{aligned} \tag{4}$$

NOW where comes the MC estimation? Thinking as an engineer only suffices :) As you can see we need to estimate the integral. We have the integral approximated by previous stage MC-estimation of $p(x_{t-1}|z_{1:t-1}, u_{1:t-1}, m)$ and hence can approximate the integral with the weighted kernel estimate as:

$$\int_{x_{t-1}} p(x_t|x_{t-1}, u_t, m)p(x_{t-1}|z_{1:t-1}, u_{1:t-1}, m)dx_{t-1} \approx \sum_{i=1}^{N} p(x_t|x_{t-1}^{(i)}, u_t, m)w_{t-1}^{(i)} \tag{5}$$

OK! We have samples for $p(x_{t-1}|z_{1:t-1}, u_{1:t-1}, m)$ but what about samples for $p(x_t|z_{1:t}, u_{1:t}, m)$. Since we cannot sample from $p(x_t|z_{1:t}, u_{1:t}, m)$ directly (if we could then there's no need for this "predict-update" problem) let's use $\int_{x_{t-1}} p(x_t|x_{t-1}, u_t, m)p(x_{t-1}|z_{1:t-1}, u_{1:t-1}, m)dx_{t-1}$ as the "proposal" and weight the particles as

$$\hat{w}_t(x_t) = \frac{p(x_t|z_{1:t}, u_{1:t}, m)}{\int_{x_{t-1}} p(x_t|x_{t-1}, u_t, m)p(x_{t-1}|z_{1:t-1}, u_{1:t-1}, m)dx_{t-1}} \tag{6}$$

$$= \frac{\eta p(z_t|x_t, m) \cancel{\int_{x_{t-1}} p(x_t|x_{t-1}, u_t, m)p(x_{t-1}|z_{1:t-1}, u_{1:t-1}, m)dx_{t-1}}}{\cancel{\int_{x_{t-1}} p(x_t|x_{t-1}, u_t, m)p(x_{t-1}|z_{1:t-1}, u_{1:t-1}, m)dx_{t-1}}} \tag{7}$$

Guess what, we just saw an example of *Sequential* MC estimation! It is iterative but we get *new* observations in every stage that let's us improve our estimate. Contrast this with Metropolis-Hastings (on your own) if interested.

ALGORITHMIC PERSPECTIVE:

- **Initializing:** For $i = 1, \dots, N$ draw $x_1^{(i)}$ from $p(x_1)$.
- **Sampling/Predicting:** For $i = 1, \dots, N$ sample from the proposal $x_t^{(i)} \sim \sum_{j=1}^N p(x_t|x_{t-1}^{(j)}, u_t, m)w_{t-1}^{(j)}$.
- **Weighting/Updating:** For $i = 1, \dots, N$ evaluate the importance weights $\hat{w}_t^{(i)} = \eta p(z_t|x_t^{(i)}, m)$. Then normalize the importance weights $w_t^{(i)} = \frac{\hat{w}_t^{(i)}}{\sum_{j=1}^N \hat{w}_t^{(j)}}$.

3 Robot map learning (mapping)

Here the state to be learnt is the map of the environment the robot is in. For an online learning the filter would be

$$p(m_t|x_{1:t}, z_{1:t}) \tag{8}$$

Guess what we have analytical solution for this filter!!! Occupancy grids is one example. We don't need MC-estimation. It would not be wise to use MC anyways since m_t is usually very high dimensional (equal to number of features in a map) and particle filter also suffers *curse of dimensionality.* Which says in a high dimensional space we need lot of data reason something strong because of the exponential increase in volume of the hypercube.

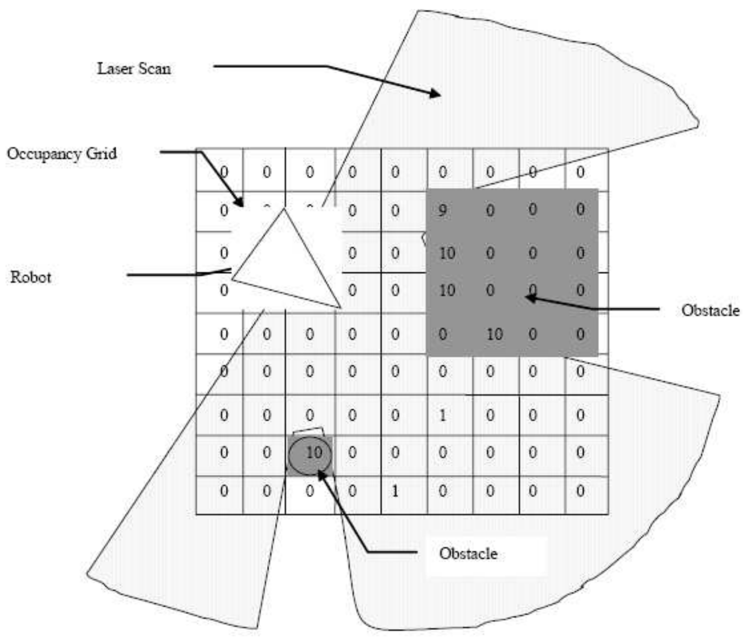

Figure 4 Occupancy grid after 10 scans

4 Simultaneous Localization and Mapping

The filter here is:

$$p(x_{1:t}, m_t | z_{1:t}, u_{1:t}) \tag{9}$$

For this there is no known analytic expression so we use MC estimation. So usually it's hard to filter more than ~ 50 features in m_t. But fortunately Rao-Blackwellization with state-decomposition theorem which roughly stating leads that if a state can be decomposed into analytical sub-parts then it's indeed *better* than direct estimation!

Since we know $p(m_t | x_{1:t}, z_{1:t})$ is analytically computable let's decompose the SLAM posterior as:

$$\begin{aligned} p(x_{1:t}, m_t | z_{1:t}, u_{1:t}) &= p(x_{1:t} | z_{1:t}, u_{1:t}) p(m_t | z_{1:t}, u_{1:t}, x_{1:t}) \\ &= p(x_{1:t} | z_{1:t}, u_{1:t}) p(m_t | z_{1:t}, x_{1:t}) \ (\because \ m_t \text{ is independent of } u_{1:t} \text{ given } z_{1:t}, x_{1:t}) \end{aligned} \tag{10}$$

So we need to track only $p(x_{1:t} | z_{1:t}, u_{1:t})$ using SMC estimation. Let's follow the approach we took for pose learning. Let's first get the update equation using Bayes rule:

$$p(x_{1:t}|z_{1:t}, u_{1:t}) = \eta p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t}) p(x_t|x_{1:t-1}, u_{1:t}) p(x_{1:t-1}|z_{1:t-1}, u_{1:t-1}) \tag{11}$$

The derivation follows:

$$p(x_{1:t}|z_{1:t}, u_{1:t}) = \frac{p(x_{1:t}|u_{1:t}) p(z_{1:t}|x_{1:t}, u_{1:t})}{p(z_{1:t}|u_{1:t})}$$

$$\left(\text{using Baye's rule } p(A|B,C) = \frac{p(A|B)p(C|A,B)}{p(C|B)}\right)$$

where $A = x_{1:t}, B = u_{1:t}, C = z_{1:t}$

$$= \frac{p(x_t|x_{1:t-1}, u_{1:t})\, p(x_{1:t-1}|u_{1:t-1})\, p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t})\, p(z_{1:t-1}|x_{1:t-1}, u_{1:t-1})}{p(z_t|z_{1:t-1}, u_{1:t})\, p(z_{1:t-1}|u_{1:t-1})}$$

$$= \frac{p(x_t|x_{1:t-1}, u_{1:t}) p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t})\, p(x_{1:t-1}|z_{1:t-1}, u_{1:t-1})}{p(z_t|z_{1:t-1}, u_{1:t})}$$

using Bayes rule with $A = x_{1:t-1}, B = u_{1:t-1}, C = z_{1:t-1}$

$$= \eta p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t}) p(x_t|x_{1:t-1}, u_{1:t}) p(x_{1:t-1}|z_{1:t-1}, u_{1:t-1}) \tag{12}$$

Contrast this with pose learning (eq. (4)) where we have an additional integral.

Since sampling directly from $p(x_{1:t}|z_{1:t}, u_{1:t})$ is hard we sample from a proposal π that is constructed *sequentially* as:

$$\pi(x_{1:t}|z_{1:t}, u_{1:t}) = \pi(x_t|x_{1:t-1}, z_{1:t}, u_{1:t}) \pi(x_{1:t-1}|z_{1:t-1}, u_{1:t-1}) \tag{13}$$

which implies that $x_{1:t} \sim \pi(x_{1:t}|z_{1:t}, u_{1:t})$ actually means $x_t \sim \pi(x_t|x_{1:t-1}, z_{1:t}, u_{1:t})$ and $x_{1:t} =< x_t, x_{1:t-1} >$.

The important weights are computed as:

$$\begin{aligned}
\hat{w}_t(x_{1:t}) &= \frac{p(x_{1:t}|z_{1:t}, u_{1:t})}{\pi(x_{1:t}|z_{1:t}, u_{1:t})} \\
&= \frac{\eta p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t}) p(x_t|x_{1:t-1}, u_{1:t})\, p(x_{1:t-1}|z_{1:t-1}, u_{1:t-1})}{\pi(x_t|x_{1:t-1}, z_{1:t}, u_{1:t})\, \pi(x_{1:t-1}|z_{1:t-1}, u_{1:t-1})} \\
&= \hat{w}_{t-1}(x_{1:t-1}) \frac{\eta p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t}) p(x_t|x_{1:t-1}, u_{1:t})}{\pi(x_t|x_{1:t-1}, z_{1:t}, u_{1:t})}
\end{aligned} \tag{14}$$

ALGORITHMIC PERSPECTIVE:

- **Initializing:** For $i = 1, \ldots, N$, draw $x_1^{(i)}$ from $p(x_1)$.
- **Sampling/Predicting:** For $i = 1, \ldots, N$, sample from the proposal $x_t^{(i)} \sim \pi(x_t | x_{1:t-1}^{(i)}, z_{1:t}, u_{1:t})$. And $x_{1:t}^{(i)} \equiv < x_t^{(i)}, x_{1:t-1}^{(i)} >$.
- **Weighting/Updating:** For $i = 1, \ldots, N$, evaluate the importance weights
  $\hat{w}_t^{(i)} = \hat{w}_{t-1}^{(i)} \frac{\eta p(z_t | z_{1:t-1}, x_{1:t}^{(i)}, u_{1:t}) p(x_t^{(i)} | x_{1:t-1}^{(i)}, u_{1:t})}{\pi(x_t^{(i)} | x_{1:t-1}^{(i)}, z_{1:t}, u_{1:t})}$ Then normalize the importance weights $w_t^{(i)} = \frac{\hat{w}_t^{(i)}}{\sum_{j=1}^{N} \hat{w}_t^{(j)}}$.

5 Optimality, resampling schedule and proposal

OPTIMALITY: Optimality of filtering via simulation can be measured using variance of the importance weights of the particles. In a perfect simulation scenario, weights of all particles must be same meaning the samples are *randomly* drawn from the posterior which is being simulated. This can also be noted in the weight equations above where the importance weights would be 1 if $\pi = p$ and the normalized weights would be $1/N$. In such a case the variance of weights will be 0.

But it's been shown that the variance increases over time for a finite N. This problem is called *weight degeneracy* problem.

RESAMPLING: Resampling helps in reducing the variance by replacing low weight particles with duplicates of higher weight particles. But naive resampling might lead to having only one particle duplicated which is called *particle depletion* and if you notice this also means the particles are not random in fact they are too deterministic. One important note is that after resampling, the weights of all particles are reset to $1/N$.

Adaptive resampling is an easy engineering fix that let's us schedule resampling according to a flag that tries to maintain a balance between reducing variance and avoiding bias.

The flag variable is $N_{eff} = \frac{1}{\sum_{i=1}^{N} w_t^{(i)}}$. If $N_{eff} < N/2$ resampling is *not* done otherwise we resample.

There are more complex fixes available.

PROPOSAL DISTRIBUTION: Though resampling is a temporary relief it is not a *solution*. Selection and design of proposal distribution can be helpful in spreading around particles into low variance and high likelihood regions. But in general there is no restriction from the particle filter point of view to choose a proposal distribution. We will look at two of them viz.

(a) $p(x_t|x_{1:t-1}, u_{1:t})$

(b) $p(x_t|x_{1:t-1}, z_{1:t}, u_{1:t})$

If we choose the first option then besides drawing samples the main difference is in the weight recursion eq. (14).

- **Sampling:** Drawing $x_t \sim p(x_t|x_{1:t-1}, u_{1:t})$ is simple because we are *given* a closed form which is usually Gaussian of the form $\mathcal{N}(\mu, \Sigma)$ based on the odometry motion model of the robot.

- **Weight update:** By plugging in the proposal eq. (14) is simplified as:

$$w_t(x_{1:t}) = \eta w_{t-1}(x_{1:t-1}) \frac{\cancel{p(x_t|x_{1:t-1}, u_{1:t})} p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t})}{\cancel{p(x_t|x_{1:t-1}, u_{1:t})}}$$
$$= \eta w_{t-1}(x_{1:t-1}) p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t})$$
$$= \eta w_{t-1}(x_{1:t-1}) p(z_t|m_{t-1}, x_t) \because \ z_t \text{ is independent of } u_{1:t} \tag{15}$$

and conditioning on $x_{1:t-1}, z_{1:t-1}$ is equivalent to conditioning on m_{t-1}.

If we choose the second option then sampling is also non-trivial because there is no easy closed form available. So we use MC-estimation to simulate the proposal distribution!

So let's get the update equation for the optimal proposal.

$$p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t}) = p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1}, z_t)$$
$$= \frac{p(z_t|x_t, x_{1:t-1}, u_{1:t}, z_{1:t-1}) p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1})}{p(z_t|x_{1:t-1}, u_{1:t}, z_{1:t-1})}$$
$$\left(\text{using Bayes rule } p(A|B,C,D,E) = \frac{p(E|A,B,C,D)p(A|B,C,D)}{p(E|B,C,D)}\right)$$

where $A = x_t, B = x_{1:t-1}, C = u_{1:t}, D = z_{1:t-1}, E = z_t$

$$= \frac{p(z_t|x_t, x_{1:t-1}, u_{1:t}, z_{1:t-1}) p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1})}{\int_{x_t} p(z_t|x_t, x_{1:t-1}, u_{1:t}, z_{1:t-1}) dx_t} \tag{16}$$

Since we cannot directly sample from the optimal proposal let's sample from $p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1})$ and assign weights (CAUTION: these are called first-stage weights and not to be confused with the actual weights for the particles representing the state $x_{1:t}^{(i)}$) to be:

$$
\begin{aligned}
\tilde{w}(x_t) &= \frac{p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t})}{p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1})} \\
&= \frac{\frac{p(z_t|x_t,x_{1:t-1},u_{1:t},z_{1:t-1})\cancel{p(x_t|x_{1:t-1},u_{1:t},z_{1:t-1})}}{\int_{x_t} p(z_t|x_t,x_{1:t-1},u_{1:t},z_{1:t-1})dx_t}}{\cancel{p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1})}} \\
&= \frac{p(z_t|x_t, x_{1:t-1}, u_{1:t}, z_{1:t-1})}{\int_{x_t} p(z_t|x_t, x_{1:t-1}, u_{1:t}, z_{1:t-1})dx_t} \\
&\approx \frac{p(z_t|x_t, x_{1:t-1}, z_{1:t-1}, u_{1:t})}{\sum_{j=1}^{K} p(z_t|\tilde{x}_t^{(j)}, x_{1:t-1}, z_{1:t-1}, u_{1:t})} \\
&= \frac{p(z_t|x_t, m_{t-1})}{\sum_{j=1}^{K} p(z_t|\tilde{x}_t^{(j)}, m_{t-1})} \qquad (17)
\end{aligned}
$$

where $\{\tilde{x}_t^{(j)}\}_{j=1}^{K}$ are the K samples drawn from $p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1})$. And conditioning $x_{1:t-1}, z_{1:t-1}$ is equivalent to conditioning on m_{t-1}. And z_t is independent of $u_{1:t}$. Now let's look at the **weight update**.

$$
\begin{aligned}
w_t(x_{1:t}) &= w_{t-1}(x_{1:t-1})\frac{\eta p(x_t|x_{1:t-1}, u_{1:t})p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t})}{p(x_t|x_{1:t-1}, z_{1:t}, u_{1:t})} \\
&= w_{t-1}(x_{1:t-1})\frac{\eta p(x_t|x_{1:t-1}, u_{1:t})p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t})}{p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1}, z_t)} \\
&= w_{t-1}(x_{1:t-1})\frac{\eta p(x_t|x_{1:t-1}, u_{1:t})p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t})}{\frac{p(z_t|x_t,x_{1:t-1},u_{1:t},z_{1:t-1})p(x_t|x_{1:t-1},u_{1:t},z_{1:t-1})}{p(z_t|x_{1:t-1},u_{1:t},z_{1:t-1})}} \quad \text{using Bayes rule} \\
&= w_{t-1}(x_{1:t-1})\frac{\eta p(x_t|x_{1:t-1}, u_{1:t})\cancel{p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t})}}{\frac{\cancel{p(z_t|x_t,x_{1:t-1},u_{1:t},z_{1:t-1})}p(x_t|x_{1:t-1},u_{1:t},z_{1:t-1})}{p(z_t|x_{1:t-1},u_{1:t},z_{1:t-1})}} \\
&= w_{t-1}(x_{1:t-1})\frac{\eta p(x_t|x_{1:t-1}, u_{1:t})}{p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1})}\, p(z_t|x_{1:t-1}, u_{1:t}, z_{1:t-1}) \\
&= w_{t-1}(x_{1:t-1})\frac{p(x_t|x_{1:t-1}, u_{1:t})}{p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1})}\, \int_{x_t} p(z_t|x_t, x_{1:t-1}, u_{1:t}, z_{1:t-1})dx_t \\
&\approx w_{t-1}(x_{1:t-1})\frac{p(x_t|x_{1:t-1}, u_{1:t})}{p(x_t|x_{1:t-1}, u_{1:t}, z_{1:t-1})}\, \sum_{j=1}^{K} p(z_t|\tilde{x}_t^{(j)}, x_{1:t-1}, z_{1:t-1}, u_{1:t}) \qquad (18)
\end{aligned}
$$

6 Caveats to be aware of

Filtering SLAM posterior is significantly more involved compared to Localization. Most important catch is that we sample in an *increasing* space of dimension with time and it is *bound* to diverge!!! There's already work done to fix this to some extent using Marginal Particle Filters which filters in the fixed dimension only. The key contributions by those guys was reducing the SLAM posterior to similar to that of Localization *and* also clever computational speedups.