Lecture 8: *The Particle Filter and Motion Modeling*

Scribe: *Kirby Overman, Shivangi Agarwal, Kamal Nimmagadda, Abeeku Bruce-Mensah*

September 23, 2019

Contents

1 Recap

In the previous lecture, we discussed how to estimate current state with observed data using state estimators that assume noise will have a normal distribution. The state estimators include Kalman Filters, Extended Kalman Filters, and Gaussian Sum Filters.

A Kalman Filter is normally used for linear transformations. Kalman Filters require the control u_t and observation z_t at current time t, and the belief at time $t-1$ in order to update the belief in time t. The algorithm calculates the predicted belief $\overline{bel}(x_t)$ at timestep t using only the control u_t. Z_t, the observation, is then used to transform $\overline{bel}(x_t)$ to $bel(x_t)$ which is known as the desired belief.

Extended Kalman Filters are used for non-linear transformations. In order to apply the Kalman Filter to non-linear transformations, the Taylor series expansion must be used. The filter is effective for locally linear functions, as the first expansion of Taylor series can be used for approximation. The mean of Gaussian distribution, u_{t-1}, is used to approximate the function with a most likely value. A note to remember is that belief $bel(x_t)$ found using the Extended Kalman Filter is an approximation.

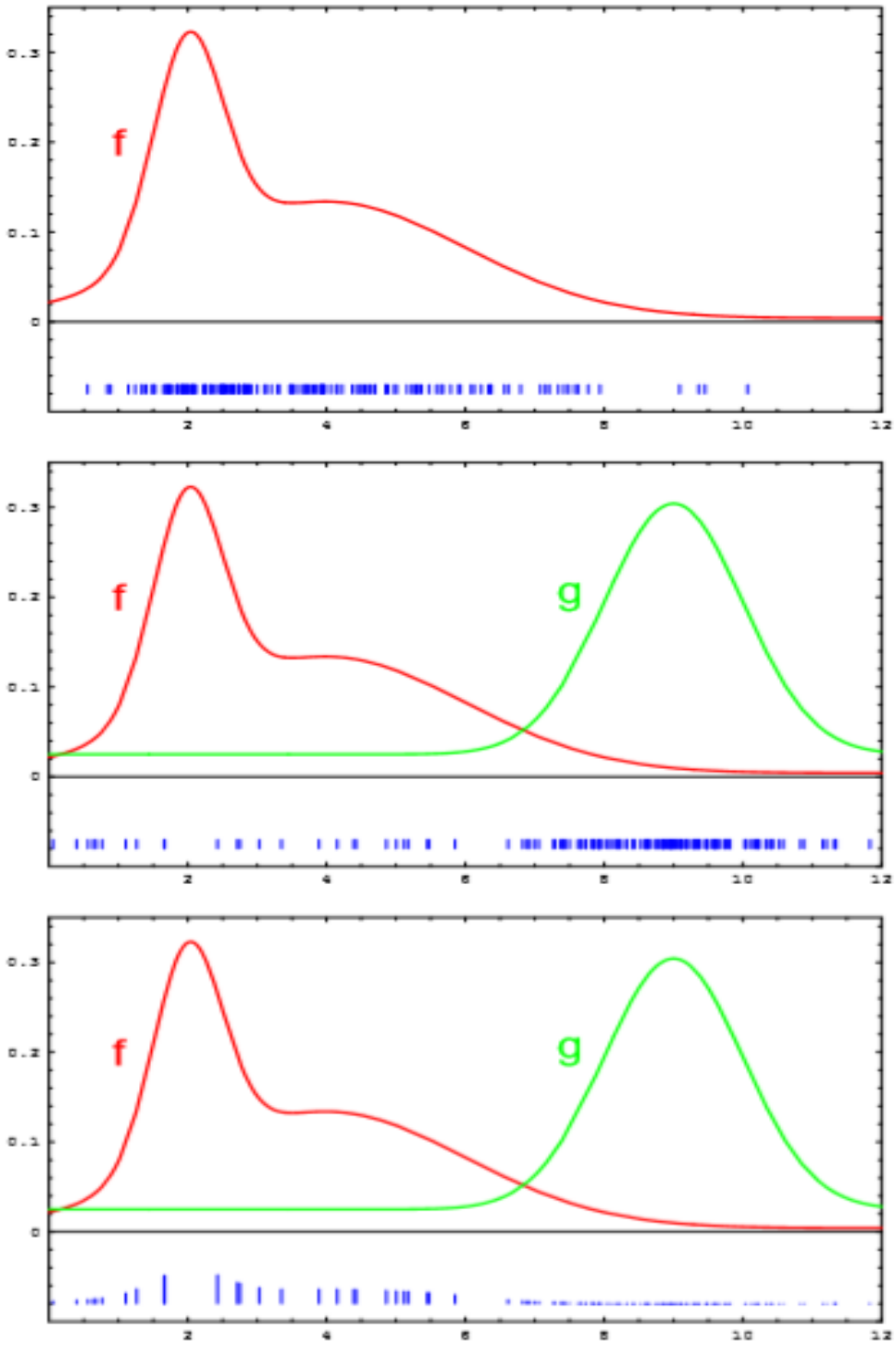
Gaussian Sum Filters should be used when there is more than one peak, or more than one hypotheses exist. The weighted sum of each Gaussian distribution with different means and variances is taken and viewed in a distribution graph. When calculating the probability of being in state x_t, the weight is computed. This weight represents the difference in the actual and expected Gaussians. As the weight is iteratively updated, many weights will diminish and there will be very few of the initial weights left used in the probability calculation.

2 Importance Sampling

Importance sampling is a methodology in which when we have a distribution f(x) that is too difficult to draw samples. It, at times, becomes easier to use function g(x) to create an importance weight

$$weight = w = (\frac{f(x)}{g(x)})$$

approximate the sampling you are looking for and then refer that sampling. Samples are taken from the function g(x) and multiplied by the importance weight in order to generate a new sample set with which to work. In the figure below you can see the distribution f with a more complex sampling, the distribution g overlayed, and the new sampling in the image.


f
f
g
f
g

3 Particle Filter

Most of the conventional filter become hard to sample when the state-space distribution is of non-linear dynamics and has a non-Gaussian distribution. To solve this problem of unusable sampling, we use particle filters. Particle filters are non-parametric filters that do not rely on the fixed functional form of the posterior. They approximate the posterior by a finite number of parameters. The key idea in particle filters is to represent $bel(x_t)$ by a set of random space samples drawn from this posterior. An alternative is to propose a distribution where drawing samples is significantly simple and assign weights while sampling such that $weight(w)$ complements the change in *target distribution* and *proposed distribution*

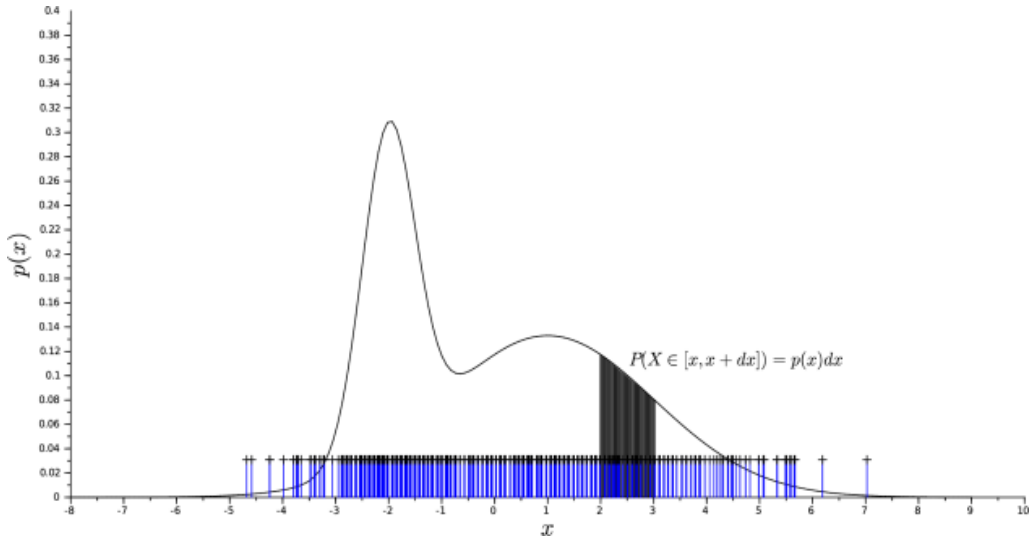


Figure 1: Sample illustration of a Particle Filter

Figure 1 illustrates an instance of samples being drawn from the distribution. Particle filters are highly efficient at time as they allow for representation of a much broader space distribution than Gaussian. Another advantage of the sample based representation is the ability to model non-linear transformations of random variables.

3.1 Mathematical Derivation of Particle Filters

Let us consider a state space distribution with non-linear dynamics and non-Gaussian distribution. The belief of a state xt is given as

$$bel(x_{0:t}) = p(x_{0:t}/u_{1:t}, z_{1:t}) = \eta p(z_t/x_{0:t}, z_{1:t-1}, u_{1:t}) p(x_{0:t}/z_{1:t-1}, u_{1:t})$$

By applying Markov's assumption, the equation condenses to

$$bel(x_{0:t}) = \eta p(z_t/x_t)p(x_t/x_{0:t-1}, z_{1:t-1}, u_{1:t-1})p(x_{0:t-1}/z_{1:t-1}, u_{1:t})$$

As the probability of any state is only dependent on the previous state and input given, the probability $p(x_t/x_{0:t-1}, z_{1:t-1}, u_{1:t-1})$ becomes $p(x_t/x_{t-1}, u_t)$. Therefore, the belief becomes

$$bel(x_{0:t}) = \eta p(z_t/x_t)p(x_t/x_{t-1}, u_t)p(x_{0:t-1}/z_{1:t-1}, u_{1:t-1})$$

In the above equation, the probability $p(x_{0:t-1}/z_{1:t-1}, u_{1:t-1})$ is simply the belief of state x_{t-1} (i.e $bel(x_{t-1})$).

$$bel(x_{0:t}) = \eta p(z_t/x_t)p(x_t/x_{t-1}, u_t)bel(x_{0:t-1})$$

Now, we assign weight w^m for each sample x^m for the proposed distribution.

$$w_t^{[m]} = \frac{targetdistribution}{proposeddistribution}$$

Now, let us assume the proposed distribution's first particle is sampled by prior $p(x_0)$. Assume that the particle set at time $t-1$, has a belief of $bel(x_{0:t-1})$. For the m^{th} particle in the set $x_{0:t-1}^{[m]}$ the sample generated $x_t^{[m]}$ is $p(x_t/x_{t-1}, u_t)bel(x_{0:t-1})$. Thus, weight assignment becomes

$$w_t^{[m]} = \frac{\eta p(z_t/x_t)p(x_t/x_{t-1}, u_t)bel(x_{0:t-1})}{p(x_t/x_{t-1}, u_t)bel(x_{0:t-1})}$$

$$w_t^{[m]} = \eta p(z_t/x_t)$$

The constant η does not have any significance as the probabilities are taken proportional to the weights. The summation of each sampling with respect to their weights gives a proper sampling that we are supposed to sample.

The intuition behind particle filters is to approximate the belief $bel(x_t)$ 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 $bel(x_t)$:

$$x_t^{[m]} = p(x_t/z_{1:t}, u_{1:t})$$

Particle filter algorithm first constructs a temporary particle set X_t containing $x_t^1, x_t^2, ..., x_t^m$. This collection of particles represent the $\bar{bel}(x_t)$. It does this by taking each particle in the input set and further approximating them to the posterior distribution $bel(x_t)$.

The algorithm then generates a hypothetical state $x_t^{[}m]$ for time t based on particle $x_{t-1}^{[m]}$ and given input. This is followed by assigning weight that corresponds to the $bel(x_t)$.

This is followed by what is known as re-sampling or importance sampling.The algorithm draws with replacement M particles from the temporary set $\bar{X}_t$. The probability

Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):
$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
for $m = 1$ to M do
  sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
  $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
endfor
for $m = 1$ to M do
  draw i with probability $\propto w_t^{[i]}$
  add $x_t^{[i]}$ to $\mathcal{X}_t$
endfor
return $\mathcal{X}_t$

of drawing each particle is given by its importance weight. Re-sampling transforms a particle set of M particles into another particle set of the same size.

4 Motion Modeling

In this section we concern ourselves with the state prediction step of the Bayesian filter. More specifically, during an implementation of a Bayesian filter, what is a good model for the state transition probability $p(x_t|x_{t-1}, u_t)$? Let's start by looking at an example.

Consider a robot operating in a three dimension state space, such that the robot pose[1] is defined by the vector

$$\mathbf{x} = [x, y, \theta]^T.$$

Let x, y represent the robots coordinates with respect to the world at time $t-1$, and θ represent its orientation[2] at time $t-1$. In other words, let

$$\mathbf{x}_{t-1} = [x, y, \theta]^T.$$

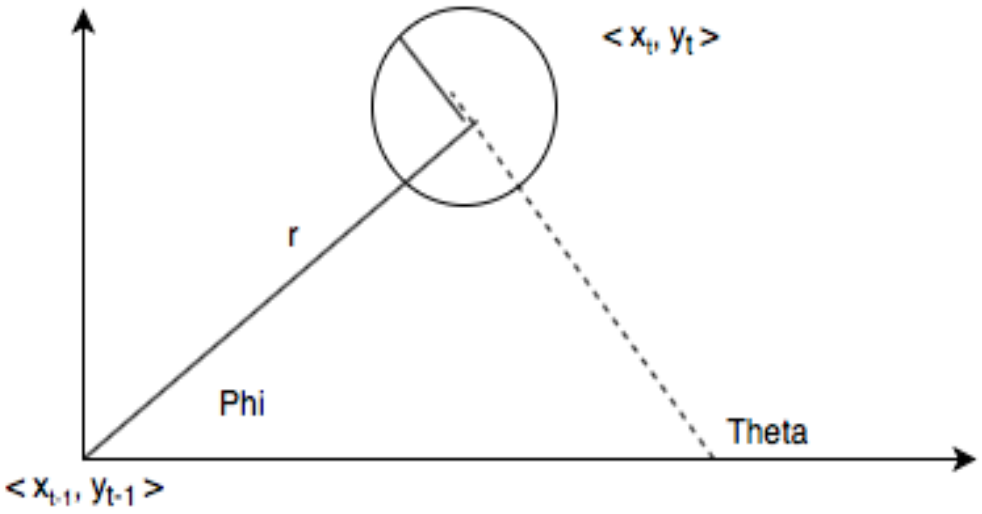
There are multiple state transition model types. Choosing the appropriate model type requires a strong understanding of the robot's capabilities and dynamics. For our example, we will assume the commonly used Velocity Model.

Let the control vector, u_t be defined as

$$u_t = [v, w]^T$$

such that v, w are the translational velocity and angular velocity respectively.

[1] !!!! Add pose definition
[2] Typically, this value is referred to by yaw, bearing, or heading.



As the diagram above illustrates, for a given orientation at time $t-1$ we can readily compute the new orientation θ'. We use the relationship between velocity and position to determine

$$\theta' = \theta + w\Delta t$$

.

For our x', y' coordinates, we must first model their dependence to the angular displacement of the robot.

First we need to solve for ϕ. Considering the sum of the angles of the triangle made by the line of θ and r must equal π, we can produce the following:

$$\phi + \pi - \theta + \frac{\pi}{2} = \pi.$$

Solving for ϕ gives

$$\phi = \theta - \frac{\pi}{2}.$$

For our x, y approximations, excluding the affect of the change in θ, we have

$$\bar{x}_t = x_{t-1} + rcos(\phi) = x_{t-1} + rcos(\theta - \frac{\pi}{2})$$

and

$$\bar{y}_t = y_{t-1} + rsin(\phi) = t_{t-1} + rsin(\theta - \frac{\pi}{2}).$$

We can further simpifiy by reducing out the $\frac{\pi}{2}$ term.

$$\bar{x}_t = x_{t-1} - rsin(\theta)$$

and

$$\bar{y}_t = y_{t-1} + rcos(\theta).$$

Now taking the change in orientation into account,

$$x_t = x_{t-1} - rsin(\theta) + rsin(\theta + w\Delta t)$$

and

$$y_t = y_{t-1} + rcos(\theta) - rcos(\theta + w\Delta t).$$

Next we can introduce the control into our example construction. r is the control applied to the robot at time t to cause any changes between $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$.

For this example we assume the control dynamic $r = \frac{v}{w}$. In an perfect environment, the velocity model, or any other motion model, will have zero error in the controls. Realistically, such perfection does not exist. With this in mind, we assume some error distributions ϵ_v and ϵ_w each of which are Gaussian distributions with zero mean and σ^2 variance. This gives us the equations

$$\hat{v} = v + \epsilon_v$$

and

$$\hat{w} = w + \epsilon_w.$$

By substitution of these error-prone controls, we finalize our example construction using the velocity model.

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} - \frac{\hat{v}}{\hat{w}} sin(\theta) + \frac{\hat{v}}{\hat{w}} sin(\theta + w\Delta t) \\ y_{t-1} + \frac{\hat{v}}{\hat{w}} cos(\theta) - \frac{\hat{v}}{\hat{w}} cos(\theta + w\Delta t) \\ \theta_{t-1} + \hat{w}\Delta t \end{bmatrix}.$$

This model also leads us to an algorithm produced here.

Algorithm 1 SAMPLE-MOTION-VELOCITY($u_t, \mathbf{x}_{t-1}$)

1: $\hat{v} = v +$ SAMPLE(ϵ_v)
2: $\hat{w} = w +$ SAMPLE(ϵ_w)
3: $x_t = x_{t-1} - \frac{\hat{v}}{\hat{w}} sin(\theta) + \frac{\hat{v}}{\hat{w}} sin(\theta + w\Delta t)$
4: $y_t = y_{t-1} + \frac{\hat{v}}{\hat{w}} cos(\theta) - \frac{\hat{v}}{\hat{w}} cos(\theta + w\Delta t)$
5: $\theta_t = \theta_{t-1} + \hat{w}\Delta t$

5 References

1. Sebastian Thrun,, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. MIT press, 2005. (TBF)