

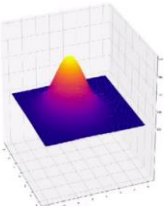
Introduction to Robotics
Professor Balaraman Ravindran
Department of Computer Science
Indian Institute of Technology, Madras
Lecture - 35
Particle Filter

(Refer Slide Time: 00:14)

NPTEL

Non-Parametric Filters

► Gaussian filters assume that the belief distribution is of a specific functional form described by a fixed set of parameters.


$$p(x, \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

Introduction to Robotics Prof. Balaraman Ravindran

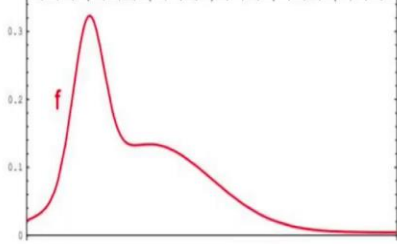
Welcome to Lecture 3 in Week 10. And so, in this lecture, we are going to look at what are called non-parametric filters. So far, we have been looking at Gaussian filters for state estimation. And the Gaussian filters essentially assume that your belief distribution is of a specific functional form. In this case, it was, we assumed that it was a multivariate Gaussian and it is described by a fixed set of parameters in this case, which was like μ and Σ .

So even though we looked at a couple of variations of this, the fundamental underlying assumption is that there is a specific functional form, that describes the belief distribution, and that is given by a fixed set of parameters. So these are called parametric. So now, what we are going to look at are what are called non-parametric filters and there are many, many non-parametric filters. We will look at a specific one.

(Refer Slide Time: 01:10)


Non-Parametric Filters

- ▶ Alternative to Gaussian Filters, non-parametric filters do not assume a fixed functional form over the belief distribution, allowing us to model more complex, non-Gaussian probability distributions.



- ▶ The complexity of the function could vary to fit the distribution, typically estimated by samples drawn from the distribution.

Introduction to Robotics Page 20 of 30 Prof. Balaraman Ravindran



So the idea behind parametric filters is that, idea behind non-parametric filters is that they, they do not assume a fixed functional form. They can look at arbitrary distribution as I can see here. So this is an arbitrary distribution, does not look like a simple Gaussian.

And so, the idea behind non-parametric filters is that the complexity of the function that we want to represent, the density, the belief, belief distribution that we want to represent, the complexity could vary to fit the distribution. So it is not like it is a fixed complexity function like the Gaussian distribution.

It is not of a fixed functional form but the function could vary and typically depending on the kind of samples that we draw from the distribution, we kind of increase or decrease the number of parameters that we use for describing the function itself. So even though we call it non-parametric, does not mean there are no parameters, it just means that there is not a fixed set of parameters which we use for describing the function.

Advantage of this kind of non-parametric filters is that they allow us to look at complex and arbitrary in some cases, arbitrary distributions and we are not restricted to Gaussian distributions. So if you remember, the problem with Gaussian distributions was that there had to be one hypothesis which we thought was most likely and we had small amount of noise around that

hypothesis. So we could not look at a hypothesis that supported us having multiple modes. So it had to be a unimodal belief state and so, the non-parametric filters allow us to get around that.

(Refer Slide Time: 02:54)

The slide is titled "Non-Parametric Filters" and "The Particle Filter". It contains a bullet point: "► Instead of representing the belief distribution by a parametric form, particle filters represent this distribution by a set of samples drawn from the distribution." Below the text is a graph with a red curve labeled 'f' and a set of red dots representing samples. The graph shows a distribution with a peak and a tail. The y-axis ranges from 0 to 0.3. The x-axis is unlabeled. The slide also features the NPTEL logo in the top right corner and a small video inset of a man in the bottom right corner. At the bottom of the slide, it says "Introduction to Robotics", "Page 21 of 28", and "Prof. Balaraman Ravindran".

So what are going to look at in this lecture is specific, non-parametric filter called the Particle Filter. So in the particle filter, instead of representing the belief distribution as a function, I am going to assume that the distribution itself is represented by a set of samples that are drawn from the distribution.

So the curve f describes the distribution that I am trying to represent, and the way I represent the distribution is by just storing a set of samples that were drawn from the distribution. Notice that this distribution is defined on a, on one dimension. So just this x -axis is the dimension, the y -axis is actually the density for the, for the values of the x . So for each value of x , this, the y -axis represents the particular density for the distribution.

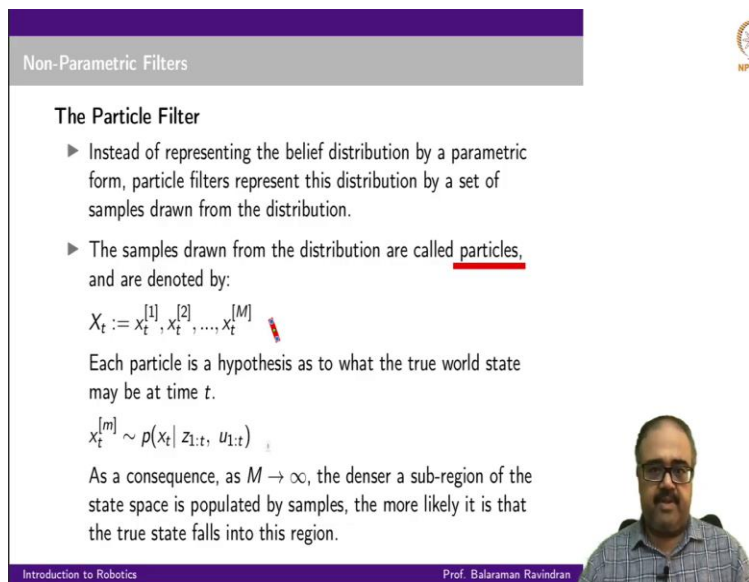
So, essentially, what I am doing here is, I am sampling according to this distribution represented by f and therefore, you can see that wherever the density is high, I get a lot of samples, wherever the density is high I get a lot of samples, and wherever the density is low, I get fewer samples. And you can see intermediate densities, I have intermediate number of samples.

So basically, instead of trying to represent this whole thing as a functional form, what I do is, I just represent a set of points, x in this case. I just store a set of points x which in an indirect way represent the distribution. That makes sense?

So in the case of the particle filter, I store a set of samples drawn according to the distribution as opposed to storing like say, something like μ and Σ of the distribution. So instead of storing like a set of parameters for the distribution, I am going to store a set of samples that were drawn from that distribution and that will be the representation I have for my belief distribution.

So I am going to make sure that we are specializing this discussion to beliefs but the notion of particle filters can be used in a variety of other settings as well. Okay so moving on.

(Refer Slide Time: 05:01)



The slide is titled "Non-Parametric Filters" and features a video inset of Prof. Balaraman Ravindran. The main content is as follows:

The Particle Filter

- ▶ Instead of representing the belief distribution by a parametric form, particle filters represent this distribution by a set of samples drawn from the distribution.
- ▶ The samples drawn from the distribution are called particles, and are denoted by:

$$X_t := x_t^{[1]}, x_t^{[2]}, \dots, x_t^{[M]}$$

Each particle is a hypothesis as to what the true world state may be at time t .

$$x_t^{[m]} \sim p(x_t | z_{1:t}, u_{1:t})$$

As a consequence, as $M \rightarrow \infty$, the denser a sub-region of the state space is populated by samples, the more likely it is that the true state falls into this region.

Introduction to Robotics Prof. Balaraman Ravindran

So in the particle filter, like I said, you take a set of samples and these samples are called particles. So these samples are called particles. These samples are called particles and we denote them by x_1 to x_M right, this is basically a set of M particles, capital M particles. And each particle is, in some sense, we can think of as a hypothesis, as to what the true world state may be at time t . It is just a guess if you will, of what we think is a true world state at time t , and the way we generate these particles is by sampling from this distribution.

So a particle at time t is sampled from the distribution which is probability of x_t , given z_1 to t , and u_1 to t . This is basically all the observations I have seen from the beginning of time until

time t and all the actions I have performed beginning of time until time t . So basically what is this? This expression is a belief state. So where do I believe that I am at time t ? So this is essentially what our belief state is and so, the set of particles are sampled from our belief state at time t .

Now, as M tends to infinity, M meaning the number of samples that I draw, the number of particles that I maintain, then this set of particles becomes a very good approximation of the underlying density. So the denser a sub-region of the state space as populated by the samples; the denser the population of samples, the more likely is it that the true state falls into that region.

So it becomes more of a proper representation of the underlying belief distribution. So the particle filter essentially represents the belief as a set of samples drawn from the belief distribution. We do not actually explicitly represent the belief distribution as a functional form here.

(Refer Slide Time: 07:08)


Non-Parametric Filters


▶ As with the previous algorithms, the particle filter algorithm estimates the belief $bel(x_t)$ recursively from the belief $bel(x_{t-1})$.

```

1:  Algorithm Particle.filter( $\mathcal{X}_{t-1}, u_t, z_t$ ):
2:     $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$ 
3:    for  $m = 1$  to  $M$  do
4:      sample  $x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$ 
5:       $w_t^{[m]} = p(z_t | x_t^{[m]})$ 
6:       $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + (x_t^{[m]}, w_t^{[m]})$ 
7:    endfor
8:    for  $m = 1$  to  $M$  do
9:      draw  $i$  with probability  $\propto w_t^{[i]}$ 
10:     add  $x_t^{[i]}$  to  $\mathcal{X}_t$ 
11:    endfor
12:    return  $\mathcal{X}_t$ 

```





Introduction to Robotics
Prof. Balaraman Ravindran

Now, as with the earlier versions of the Bayes filter algorithm, so here is a version of the particle filter algorithm. And again, we update the belief of x_t recursively from belief in x_t minus 1. And as before, the particle filter algorithm takes as input the current action u_t , the current measurement or the current observation z_t , and our representation of $bel x_t$ minus 1.

If you remember, initially in the Bayes filter algorithm, $bel_{x_{t-1}}$ was just supplied as it is and in the Gaussian filters, $bel_{x_{t-1}}$ was supplied by, yeah, μ_{t-1} and Σ_{t-1} . In this case, bel_{t-1} is supplied by the set of particles x_{t-1} . If you remember, that is what we said.

So the, the distribution any point of time is represented by the set of particles x_t . So, so we start off with x_{t-1} as the set of particles that represent belief x_{t-1} . So notice that, so what do I mean by the particles representing the belief? That means that these particles x_{t-1} , there will be M such particles, so these m particles were obtained by somehow sampling from the belief distribution at time $t-1$.

Now, I start off by initializing both my \bar{x}_t and x_t which are two sets of particles, and this you can guess, \bar{x}_t is going to represent $bel_{\bar{x}_t}$, so the $bel_{\bar{x}_t}$ at time t and x_t is going to represent bel_{x_t} ; bel_{x_t} at time t , bel_{x_t} . So \bar{x}_t which is script \bar{x}_t , it is going to represent $bel_{\bar{x}_t}$ of x_t , and script x_t is going to represent bel_{x_t} . So that is what we are going to do.

So I will just run through the steps and then later in the next few slides, you will see how this implements the actual belief update. So what I do is initially, so the lines 3 to 7, so lines 3 to 7 are running through each, each of my particles once. So my particle here is a sample. So each of my particles is going to run through once.

So what it does, the first, so what it does in line 4, so what is happening in line 4 is that, what is happening in line 4 is that I take my previous particle and take one sample, which is the m th sample, I am taking the previous particle from my belief x_{t-1} . And then apply the action u_t which I know. I apply the action u_t and I look at the distribution of the resulting state. I am looking at the distribution of x_t and I am just going to sample an $x_{t,m}$ from this distribution.

Basically, I am going to perturb this particle by the action. I am going to move it forward assuming that this is the actual state at time $t-1$. I am going to make a prediction as to what would be the state at time t and I am going to sample from that.

So for every particle that I was using to represent x_{t-1} , I am going to sample one particle now to get (myself), get me x_t , get the new x_t . Now, this is a little tricky part here. So what I am

doing now at step 5, so what I am doing at step 5 is essentially trying to account for the observation probabilities or the measurement probabilities.

We know that z_t was the actual measurement that we made and I am assuming that x_t^m is the real state. So I am looking at, okay, hey, if x_t^m was really the correct state, x_t^m was really the correct state what is the probability that I would have made this observation. I am going to assign that as what is called as a weight or the importance weight for the sample x_t^m . So I have w_t^m which is the importance weight for the sample x_t^m .

And then I have, I am just adding to the belief representation, belief x_t representation, the sample x_t^m , so the actual, actual particle as well as the weight. So I am going to add the particle and the weight to the belief x_t , x_t^m , or rather x_t bar. It is not quite, it is not quite belief bar, it is actually something; so the x_t^m incorporates the movement. x_t^m incorporates the movement and the w_t^m actually incorporates the observation. So the x_t bar is not quite a representation of belief bar because it already incorporates the effect of the observation through the weights.

So in fact, I could just stop here. I could just stop here and say that, hey, look now, I started off with set of particles which were x_{t-1} , I had 1 to m particles. Now, I have a new set of particles, x_t^m , and a set of weights that tells me how likely is that the true state to be that particle. Makes sense?

This probability tells me how likely is it that x_t^m , if x_t^m was the true state would have given rise to the observation z_t and now, I have this weight that tells me, okay, how likely is it that this is the real state given that you have observed x_t . So it is, in a way I can still construct, reconstruct my belief, I can approximately reconstruct my belief using these weights and these particles.

Then what is going to happen is that since these particles are now going to contain, suppose I come to a point where a particle has a very low weight. Because it says, hey, look, it is very unlikely that this particle is something that would have given rise to this observation. Or it is just a small probability event of you know, big noise.

So I am trying to move forward move 1 meter but suddenly I move forward 2 meters. I mean, it could very well happen that somebody pushed the robot or something so that it moved forward 2 meters but it could be a very, very low probability event, you know. But then, let us say, that is a

sample I draw. Right now I have gone forward 2 meters but the robot has really not gone forward 2 meters, it has gone forward only 1 meter.

And then I make a measurement I am going to say, huh, this measurement is quite unlikely. But then what is going to happen? I am going to have a location that says the robot is 1 meter away from where it really is with a very low weight. But then when I go back and try to do a resampling of this when I make the next action, I will still keep pushing this particle forward and I am going to give lower and lower and lower weight to it till it becomes vanishingly 0.

So what will happen is because there is so much noise in the system, many of the particles' weight will go to 0. And so, I will be carrying around a lot of particles whose weight is 0 and it is not a great filter then. It is not a great representation of my belief. So if I have, a lot of my particles have weight 0, then I am basically using only one or two samples to represent my distribution and I could be very wrong in that. We will see that in a minute when we go to the actual illustration of this algorithm.

So how do I overcome this? So I know, I told you that already my x_t bar has enough information that incorporates the observations but the set of particles are not really great. So what I do is something called resampling. So this, this phase, lines 8 to 11 of the algorithm, I do what is called resampling of the particles. I have the same, I have M particles now, each having some weight.

Now, what I will do is I will resample M particles, capital M particles again. So new set of particles that I am drawing but the probability that I will draw one of the earlier samples, so remember that I am going to draw samples here with replacement from x_t bar. What I am doing in line 9 is drawing samples with replacement from x_t bar. So people, all of you are familiar with sampling with replacement. So we are going to do sampling with replacement from x_t bar.

So what we do there is we will pick a number i with a probability that is proportional to w_t^i . So what is w_t^i ? It is the weight of the i th particle in my set x_t bar. So I have computed M weights. So this w_t^i is the weight of the i th particle in my x_t bar. So I am going to sample an i proportional, probability proportional to w_t^i .

Basically, that means that I am going to take each w_i divide it by sum of all the w_s , and that gives me the probability that I will sample i . Now, once I have drawn a sample, I am going to add the corresponding particle. The one that is here, x_t^i , I am going to add it to x_t ; script x_t .

So I am going to that M times because M is the size of the particles that I am maintaining. So I will draw M sized particles like this. Notice that these particles I am drawing are exactly the same set of particles here. Just that some particles might be drawn more than once and a few particles could just be left out in this new set M .


Why could particles be left out in the new set M ? Because they have low probability and particles could be repeated in the new set M because the probability of the you know, the probability mass in that particular state in the belief state is very high. And then I do this for M times so that I get a new set of samples M and then I return x_t . So remember, so the last thing that return is bel_{x_t} so this is basically bel_{x_t} .

So this is not quite neatly broken down into the first step where I compute $bel_{\bar{x}}$ and the second phase where I compute bel . So the first phase here is where I actually, you know, do my prediction as well as accounting for my observation. My prediction and the correction computation is already done in the first loop.


So that is my prediction computation and that is my correction computation. I am already done that in the first loop. And what I am doing here in the second loop is essentially resampling the particles so that I actually get a more, more appropriate representation for bel .

(Refer Slide Time: 18:22)

Non-Parametric Filters




- ▶ The set of particles resulting from iterating Step 4 M times is the filter's representation of $\bar{bel}(x_t)$
- ▶ Line 5 calculates for each particle $x_t^{[m]}$ the so-called importance factor, denoted by $w_t^{[m]}$. Importance factors are used to incorporate the measurement z_t into the particle set.




Introduction to Robotics Prof. Balaraman Ravindran

Non-Parametric Filters



- ▶ As with the previous algorithms, the particle filter algorithm estimates the belief $bel(x_t)$ recursively from the belief $bel(x_{t-1})$.

```
1: Algorithm Particle.filter( $\mathcal{X}_{t-1}, u_t, z_t$ ):
2:    $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$ 
3:   for  $m = 1$  to  $M$  do
4:     sample  $x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$ 
5:      $w_t^{[m]} = p(z_t | x_t^{[m]})$ 
6:      $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + (x_t^{[m]}, w_t^{[m]})$ 
7:   endfor
8:   for  $m = 1$  to  $M$  do
9:     draw  $i$  with probability  $\propto w_t^{[i]}$ 
10:    add  $x_t^{[i]}$  to  $\mathcal{X}_t$ 
11:  endfor
12:  return  $\mathcal{X}_t$ 
```




Introduction to Robotics Prof. Balaraman Ravindran

So as we said earlier, so the set of particles that I get from the last step. So is essentially my representation for bel of x_t and so, the one I get in step 4 is kind of, so step 4 is that. So the set of particles is kind of my representation for bel bar.

So that is not yet, that is not yet incorporated in my observation value. So it is a representation for bel bar. And what I finally get from line 9 is my, is my final representation for the belief of x_t .


(Refer Slide Time: 19:04)



Non-Parametric Filters

- ▶ lines 8-11 implement what is known as *resampling* or importance resampling.
 - The algorithm draws with replacement M particles from the temporary set \bar{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 distributed according to $\bar{bel}(x_t)$, after the resampling they are distributed (approximately) according to the posterior $bel(x_t) = \eta p(z_t | x_t^{(m)}) \bar{bel}(x_t)$

Introduction to Robotics Prof. Balaraman Ravindran



So let us look at, let us look at; I will skip this line, I will come back to this or let us go over the slide, it is fine. So like we said, like lines 8 to 11 implement what is known the resampling or the importance resampling process.

And as we said, the algorithm draws M particles with replacement from the set \bar{x}_t . Remember that \bar{x}_t is really not \bar{bel}_t . So actually, the original set of particles that we sampled in line 4 is our \bar{bel} representation, not now.

Now, what we do is essentially doing something like this. So our \bar{bel} of x_t is now sampled, is now kind of equal to the observation probability, the probability of z_t given x_t m times \bar{bel} x_t . So each particle that I am sampling here is going to be, sorry, so we have to edit the last part when we start talking about resampling. So I am going to redo the whole thing for this slide, okay.

So like we, like we looked at, when we were discussing the algorithm, lines 8 to 11 implement what is called resampling, and sometimes it is called importance resampling. And again, I am going to draw M particles with replacement from the set \bar{x}_t . Remember \bar{x}_t is not really the set of particles representing \bar{bel} .

So \bar{bel} is essentially, whatever we computed in line 4 gives you \bar{bel} and \bar{x}_t is the particles with the weights. And like we mentioned, this is already encoding your \bar{bel} x_t in some

way but the particles are not the right set of particles. Therefore we get the better set of particles when sampling with replacement.

And so, before I sample with replacement, they were distributed. The particles were distributed according to $bel(x_t)$ that is what we said. So the x_t s were representation of $bel(x_t)$ already after line 4. But after the resampling, they are distributed according to $bel(x_t)$, more or less. So that is basically the idea here. And so, let us look at how this is going to work.

(Refer Slide Time: 21:42)

Non-Parametric Filters

Illustration of Particle Filter

Introduction to Robotics Prof. Balaraman Ravindran

Non-Parametric Filters

► As with the previous algorithms, the particle filter algorithm estimates the belief $bel(x_t)$ recursively from the belief $bel(x_{t-1})$.

```

1: Algorithm Particle_filter( $\mathcal{X}_{t-1}, u_t, z_t$ ):
2:    $\tilde{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$ 
3:   for  $m = 1$  to  $M$  do
4:     sample  $x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$ 
5:      $w_t^{[m]} = p(z_t | x_t^{[m]})$ 
6:      $\tilde{\mathcal{X}}_t = \tilde{\mathcal{X}}_t + (x_t^{[m]}, w_t^{[m]})$ 
7:   endfor
8:   for  $m = 1$  to  $M$  do
9:     draw  $i$  with probability  $\propto w_t^{[i]}$ 
10:    add  $x_t^{[i]}$  to  $\mathcal{X}_t$ 
11:   endfor
12:   return  $\mathcal{X}_t$ 

```

Introduction to Robotics Prof. Balaraman Ravindran

Here is the simple belief representation. You can see that already this is something that a simple Gaussian filter cannot do. So we are basically having a bimodal distribution. So the state x here is just a single line.

It could be anywhere on this x -axis. The robot could be anywhere on the x -axis. And I am going to assume that there are these two peaks. I am not sure whether the robot is here or whether the robot is here. Of course, I am going to have some baseline probability that could be anywhere on the line because of the noise. But most of my probability mass is now concentrated around these two modes.

And the way I would do my representation; now, here would be a lot of particles will represent, will be present here, and another set of particles will be present here, and there will be very few particles elsewhere. So maybe there are 1 or 2 in other places but most of the particles will be present in these two modes. So this will be my representation of this kind of a belief state using a particle filter.

Now, let us assume that I do the action, move right. I do the action, move right. So what should happen? So what am I going to do is I will pick one particle, so this is what we do. I pick one particle from the set of particles that represent my belief. This is my probability of x_t given x_{t-1} . So that is basically the line that we are trying to look at here. Just let me go back so that you can see what we are talking about.

So what I am doing here is step 4. So I am going to look at the probability of x_t given that I am applying u_t to x_{t-1} . Some particle, some m th particle at time $t-1$; I am going to pick that, I am going to apply action u_t and I am going to sample x_t . So that is basically what we have done here.

So I have taken, I have taken a particle here so you can call this, you can think of this as x_{t-1} , and I have applied u_t which is just this action, go right. I have applied u_t here and then I have sampled from the resulting distribution, assuming that it ends up here. So this will be my, this red dot will be my new x_t .

So this is x_t minus 1 m, this is x_t m. So it is basically the step. So every, for every particle here, so I will pick that, I will apply the transition, whatever is the action and I will sample from the resulting distribution and I will get a state. Great, and now I am moving on.

So what is the next thing I have to do? Once I have done that, basically, I end up with this as my belief distribution. You can see that. So from these two being my modes, now I have become, these two have become my modes. And I have basically moved. So basically, what does it mean?

The set of particles that were here have moved here. The set of particles that were there have moved here, and the set of particles that were here have moved here. That make sense? And you can see that it has flattened out a little bit. It is not as peaked as it was here. It has flattened out a little bit. Why is that? Because there is some noise in the movement.

So when you try to move one particle from here forward, it does not always move the same distance forward. There will be some kind of a region of uncertainty around the end. So it might end up here, it might end up a little bit earlier or it may end up a little bit later. So the motion need not be deterministic and there will be some kind of a smearing out of the particles. And that is essentially what you see here and why this is, why this looks a little bit more flat.

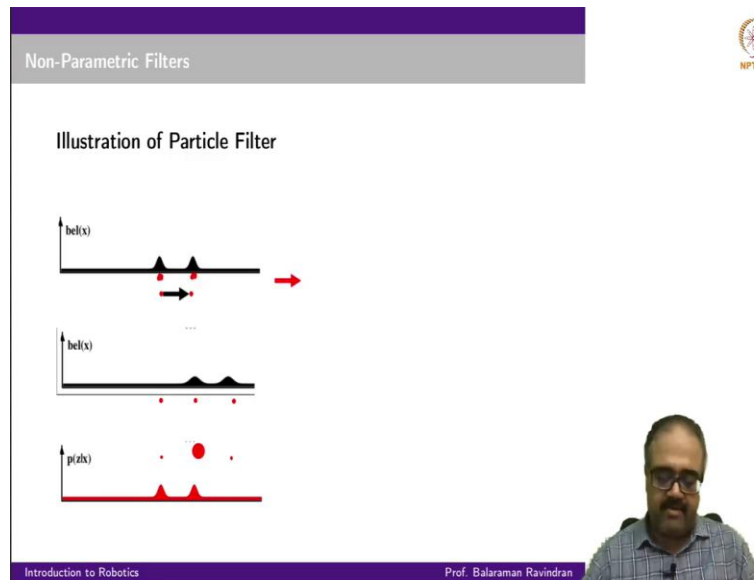
Next what I have to do is account for the, I have to account for the observations. So now, this is what my observations tell me. What does my observation tell me here? So okay, so whatever observation I am seeing now, so maybe I am seeing a patch of red on the wall, or I am seeing a door on the wall and things like that. So that door or patch of red could only occur in these two places.

I think there is a door here on the wall and there is a door here. There are two doors here, there is door here and there is another door here. And so, now, even though I have moved to the right, I have moved to the right and I actually still see a door. Let us say, my observation says that, hey, there is a door in front of you.

Now, we know from the way the observation works that the door can only be seen in one of these two places. We know that the door can be seen only in one of these two places. So what does that mean? The robot has to be either here, right in front of this door or it has to be in front of this door.

But we started off with the belief that had robot in front of one of the two doors and now, I moved right, correct and again I see a door. So I know from my observation probabilities that it has to be one of these two places but I know from my motion model it has to be one of these two places likely, therefore, I am most probably here right and most probably here. So how does that work in this particular case?

(Refer Slide Time: 27:05)



So now let us pick a few particles now. I am going to tell you how this whole importance resampling part works. Now, we already saw how we projected one particle and got this, so we got this particle here and how we projected, now we could have projected this forward and let us say, we got another particle here, I am just picking 3 particles as samples.

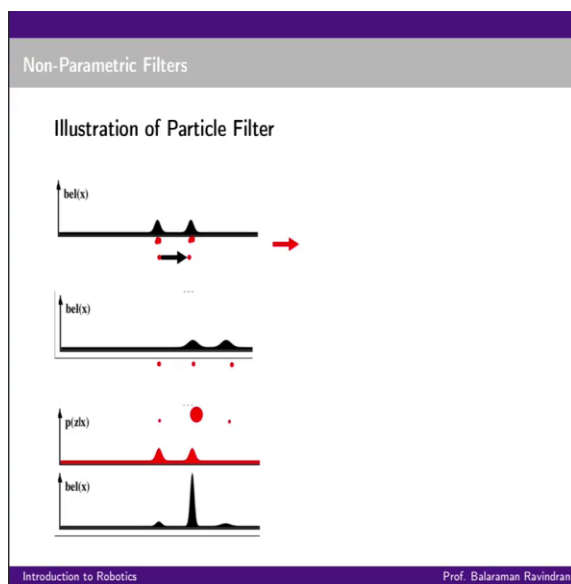
And then, this particle could have come from either the motion models telling me that a particle has not moved at all, there is some probability of x_t being the same as x_{t-1} . Maybe that is a chance of a failure of the action, or it could be that there were some stray particles here or there to accommodate for background noise, and one of those particles kind of happened to be here.

Just whatever is the reason, I am going to pick these 3 particles as samples to tell you how the importance resampling is going to work. So when I start off, when I do the prediction update, when I am actually moving these particles forward based on the action, all of these start out with an equal weight.

So all of these would have notionally equal weight, I have not assigned weights yet, But now, I am going to assign weights. So how are my weights going to be? They are going to be, looking at the probability, I am going to add the probability that the observation happens here. So literally, so the weights will now, there will be a lot of weight for this guy because I have my, so this is my posterior.

I have updated this, so this is my prediction bell bar and this is my, it is going to be the overall weight assigned to this particle. So these now, sizes have become proportional to the weight and these you can see that, have actually gone down from their default value. These weights have become very, very small. Now, what I am going to do is I am going to resample these particles according to these weights.

(Refer Slide Time: 29:06)



- As with the previous algorithms, the particle filter algorithm estimates the belief $bel(x_t)$ recursively from the belief $bel(x_{t-1})$.

1: **Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):**

```

2:    $\bar{X}_t = \mathcal{X}_t = \emptyset$ 
3:   for  $m = 1$  to  $M$  do
4:     sample  $x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$ 
5:      $w_t^{[m]} = p(z_t | x_t^{[m]})$ 
6:      $\bar{X}_t = \bar{X}_t + (x_t^{[m]}, w_t^{[m]})$ 
7:   endfor
8:   for  $m = 1$  to  $M$  do
9:     draw  $i$  with probability  $\propto w_t^{[i]}$ 
10:    add  $x_t^{[i]}$  to  $\mathcal{X}_t$ 
11:  endfor
12:  return  $\mathcal{X}_t$ 

```

When I resample the particles according to these weights, then what will happen is I will, more and more particles will end up in this region as opposed to the other regions and I can see that even though this was my belief update, after I do the reweighting, after I assign the weights and I do the resampling this becomes my belief.

So this is, this was my original belief. This is belief of x_{t-1} , this is belief of x_t , and this is belief update of x_t . Make sense? So this is exactly how the particle filter will operate. So I will have a set of particles, to begin with. I will have a set of particles, to begin with, I will pick each particle from here, I will apply my action, and I will sample from the resulting location.

So I will have a distribution over where this particle could end up, so I am going to sample from that. So I am going to basically take this set of particles and I am going to move them and so, I am going to get something like this. So there will be a slightly you know, spread out distribution that looks like that.

And then, I am going to accommodate my observation probabilities. I am going to accommodate observation probabilities and observation here, as you know, was that I saw a door, therefore, I am going to accommodate that. And wherever the particles correspond to you know, the motion being reliable and the observation matching what you see, those are going to get high weights.

Now, I am going to resample these particles. These particles should have now moved out toward here. I am going to resample them according to the weights. So probability proportional to the

weights and I will end up with something that looks like this. There will be lot, lot of particles in this space. So many of these particles will end up coming here and fewer and fewer particles would go here.

Basically, I will be resampling the same particle multiple times because as you can see, that if my density here is going to be much higher than what I had as densities here, so more fraction of my M particles, my capital M particles, a larger fraction of those should end up in, should end up in this region. A larger fraction of those particles should end up in this region. So how is that going to work out?

So a lot of particles that are here will receive a small weight and when I resample, many of them will not even get sampled. Remember I am sampling with replacement so if these particles have a small weight, they will not get sampled. But the particles that are here, and because they have a large weight now, all the particles here, they have a large weight, will get sampled again and again.

Remember, I am sampling with replacement. Even If I sample high weight particle once, it goes back into the pool so I might actually sample it again. So many of the particles that correspond to this part of the state will keep getting sampled again and again, so I will be repeating those.

So I am basically going to make my, the particles, set of particles get concentrated around this region, maybe there will be one or two occasional particles that are spread out but most of my particles are going to be here.

So the next time I am going to do a forward or a backward action or whatever motion here, that activity is going to basically be applied to this part of the state space and less likely to the other parts of the state space. So mostly, the particles will get concentrated in one region of the state space. So I hope that makes sense.

So what I am going to do now is just go back and look at the algorithm once more, so that it makes, it is now people can see it more clearly. So what we do here, we start off with a set of particles that represent belief at time $t - 1$. I have my action at time t and I have my observation at time t . We saw all of these.

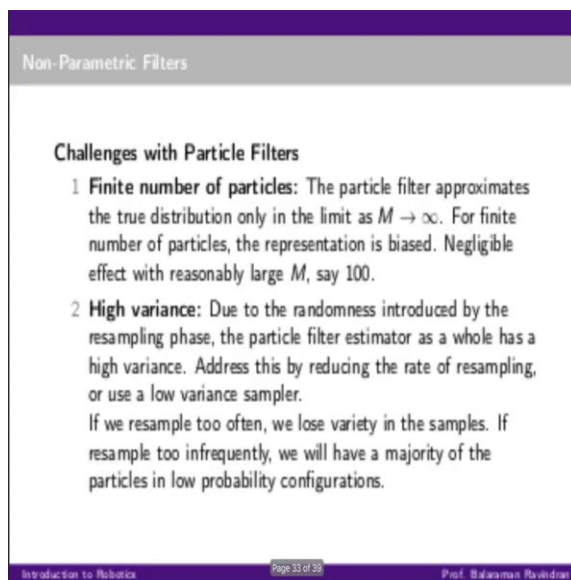
The action was to go right, observation was see a door. And now, for every particle that I have in x , script x_{t-1} , I am going to take each particle one at a time, that is, I am taking a particle, I am applying action u_t , I am going to apply action u_t and then pick a new particle x_t . So that is a particle corresponding to $t-1$.

And this particle essentially, you can think of it like, hey, I am moving my state from x_{t-1} to x_t because I did action u_t . So that is basically it. So at this point, at step 4, we saw that the second belief that we had, that the belief bar is formed by doing this.

The next thing I do is assign weights depending on how likely it is that I am going to see an observation given that this is the particle that I have drawn. And it turns out that, so the places where we can see a door is the one that we are likely to pick and so, the weights for those are very high.

And then I am going to assign the, I am going to add the weights to my particle and create x_t bar and I am going to resample from that. And when I resample from that, I am more likely to sample in particles from regions which correspond both to the motion prediction as well as to the observation probabilities and that is essentially what you saw here. So this is what happens when I do the resampling. And so, that basically what the particle filter does.

(Refer Slide Time: 34:57)



Non-Parametric Filters

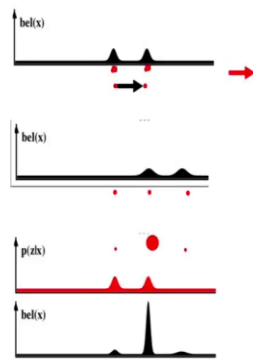
Challenges with Particle Filters

- 1 **Finite number of particles:** The particle filter approximates the true distribution only in the limit as $M \rightarrow \infty$. For finite number of particles, the representation is biased. Negligible effect with reasonably large M , say 100.
- 2 **High variance:** Due to the randomness introduced by the resampling phase, the particle filter estimator as a whole has a high variance. Address this by reducing the rate of resampling, or use a low variance sampler.
If we resample too often, we lose variety in the samples. If resample too infrequently, we will have a majority of the particles in low probability configurations.

Introduction to Robotics Page 33 of 58 Prof. Balaraman Ravindran



Illustration of Particle Filter



So particle filters are great but there are some challenges with using particle filters. So the first challenge is we use a finite number of particles. So you know that we said that when M tends to infinity, when capital M tends to infinity, the particle filter approximates the true distribution.

But for a finite number of particles, there are all kinds of problems that come in and we typically say that the representation is biased because of the small number of particles that you are using. And if there is more nuanced, more complex distribution, you might not have enough particles to represent the whole region.

But then what we typically find in practice for most applications that we are interested in, in the robotics domain is as long as M is reasonably large, and say even 100 or maybe in the 100s if you will, this bias effect is negligible except in very, very rare cases.

And therefore, for all practical purposes, you can still use the particle filter but you have to use a large number of particles. You cannot just say that I am going to use 3 particles, 4 particles; you will have to use a large number of particles to make sure that you do not get into the biases associated with the finite number of particles.

So the second problem is there is a high variance that comes in when you are using a particle filter because of the randomness that we are having in the resampling phase. So what happens is, if you remember what we saw here in this particular case, a lot of these samples that were

retained in your belief representation were essentially the same sample being repeated. So, and this is something that will always happen.

Suppose I had like 10 samples here and I had 10 samples here, when I do the resampling, I am going to get, say, 18 samples here and 2 samples here. And to, just to keep the numbers small; so I had 10 here and 10 here, now I do the resampling, I am going to get 18 here and I am going to get 2 here.

So that 2 is not a problem but if the 18 is going to have a lot of repeated copies of the same 10. So basically, what is happening is I am kind of reducing the variety in the, in the particles. Now, that is fine. You will think that reducing the variety in the particles will reduce the variance. It does but that is for a specific run of the particle filter.

If I give you the same sequence of observations and same sequence of actions and observations and ask you to rerun the particle filter, just because there is a randomness here in which particles I choose to resample, I might end up with a different set of particles being repeated.

Again, the variance could be low for that particular run but across runs which particles are retained and which particles are used for continuing to update my thing could be very different because I am going to concentrate on a small number which are kept repeated, which are repeated. I am going to concentrate on small number of particles that are repeated.

So this could lead to a huge variance in the estimate. So one way of getting around that is to say that, hey, this I am getting this only because I am resampling so often. So every time I resample I kind of reduce variety in the particles. So why do not I resample you know, maybe at a lesser frequency. So I can reduce the rate of resampling.

Instead of resampling after every movement, maybe I will resample after every 10 movements or every 20 movements or every 30 actions, instead of resampling after every action. So that is one way of doing it. Then there are other things called the low variance samplers for particle filters which we could use and I am not getting into that. I mean it is discussed in the book if you are interested in looking at it.

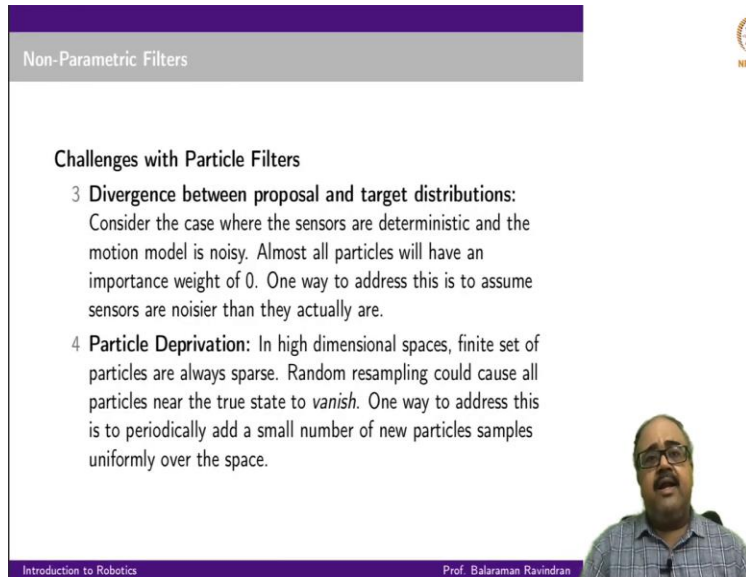
But then, when I am going to reduce the rate of resampling I have to be very wary of some things. So we already saw that if we resample too often, we lose the variety in the samples. But on the other hand, if we resample too infrequently, if I say I am going to resample every 100 actions or every 200 actions or something like that, what will happen is the problem that I pointed out earlier, you remember? I said, what happens if we do not do resampling?

I can still reconstruct my belief just by using the particles I get for my belief and the weights, but what will happen is as I keep doing these updates again and again and again, the particles are more likely to move into low probability configurations and eventually, the probability is to go to 0 of the particle being there and that is going to create a problem because my set of particles will become not very representative. I will only have a very small set of particles, not the M that I thought I have.

So if I resample too infrequently, we will have majority of the particles in low probability configurations. So we have to figure out what is the right rate of resampling so that we can maintain this and it comes from you know, it is lot of factors that go into it.

It depends on the complexity of the motion model, how noisy your sensors are, how noisy your actuators are, and so, typically, it takes you a few you know, iterations before you can get this, this whole thing correct. And it comes, it becomes easier with more experience using these. And particle filters again are very, very popularly used in many of these state estimation problems.

(Refer Slide Time: 40:44)



The slide is titled "Non-Parametric Filters" and features the NPTEL logo in the top right corner. The main content is under the heading "Challenges with Particle Filters". It lists two challenges: 3. Divergence between proposal and target distributions: Consider the case where the sensors are deterministic and the motion model is noisy. Almost all particles will have an importance weight of 0. One way to address this is to assume sensors are noisier than they actually are. 4. Particle Deprivation: In high dimensional spaces, finite set of particles are always sparse. Random resampling could cause all particles near the true state to vanish. One way to address this is to periodically add a small number of new particles samples uniformly over the space. A video inset of Prof. Balaraman Ravindran is visible in the bottom right corner of the slide.

Non-Parametric Filters

Challenges with Particle Filters

3 **Divergence between proposal and target distributions:**
Consider the case where the sensors are deterministic and the motion model is noisy. Almost all particles will have an importance weight of 0. One way to address this is to assume sensors are noisier than they actually are.

4 **Particle Deprivation:** In high dimensional spaces, finite set of particles are always sparse. Random resampling could cause all particles near the true state to *vanish*. One way to address this is to periodically add a small number of new particles samples uniformly over the space.

Introduction to Robotics Prof. Balaraman Ravindran

And there are couple of other challenges. So one problem is that, so the belief and the belief distributions can become very, very different when you know when we have sensors and actuators that have different levels of noise.

Here is an extreme example. Consider the case where the sensors are almost deterministic and the motion has some noise. So now, what happens is I move forward. I move forward and then my, my motion model is going to smear my particles out over a range. But let us say that my sensor is very accurate. It can tell me exactly what the x and y , or the x -coordinate is for the robot.

Now, since my motion model is noisy and it has smeared out the particles, most of the particles are going to have a weight of 0 because they are not at the right x , y coordinates. In fact, given the way this whole thing is going to operate, almost all the particles will have a weight of 0 because maybe 1 or 2 of them will actually end up at the exact right coordinates, most of the others would be slightly off.

And since my sensor is deterministic and gives me the exact x -coordinate, all the others, the probability of z which is at specific x -coordinate given the location of the particle would be 0. So what is going to happen is my weights almost all fade away to 0. And now, sampling with

replacement is just not going to make any sense. Maybe there are one or two particles or, which will have a non-zero weight. I will basically be repeating those particles again and again.

Now, this will be very, very different from what is actual belief that I wanted to represent. And in fact, what happens is, maybe in one step you get that but as you keep repeating these updates multiple time, you have all your weights go to 0 in which case you will have no particles to resample.

So one of the ways that, one of the simple ways that people address this, it typically it comes from having sensors that are too focused and, but you are not able to predict the motion correctly. So what people do is they assume that the sensors are noisier. Basically, you look at your probability of z given x and you add a little bit of noise to it.

You do not use the actual probability of z given x , you add more noise to it than is actually there in the sensor. And this allows you to get around. This is a very cheap and simple way of making sure this divergence problem does not happen.

And the last problem, last challenge, I should not say last problem, last challenge in implementing particle filters is something called the particle deprivation problem, or particle deprivation challenge.

So typically, I mean we have been talking about one dimensions and other things because easy to draw but typically, you are operating in a very high dimensional space. So a lot of different state variables that you have to take care of while representing the robot. And any finite, any small finite set of particles, even I am talking of 100, 200, or something like that it will always be sparse in a high dimensional space.

So when I am randomly resampling things, when I am especially, when I am initially, when I am doing my random resampling, so it could, I could very easily come to a point where particles that are close to the true state vanish. And so, instead of being at the exact correct state, instead of being close to the true state. I might actually sample things that are slightly farther away, and given the space is high dimension, so everything is sparse.

There will be only a few particles, to begin with, that were very close to the true state, and now, they will vanish because of the random resampling. And I could start sampling slightly farther away from the true state but then, once I start projecting those particles into the future, they will move further and further away from where the actual state of the robot is and I might come to a point where I might not have any particles close to the true state for me to make any reasonable belief updates. I might actually be completely deluded into thinking somewhere else.

So one way to address this is to not resample M particles every time, resample something less than M and then add a small number of new particles, sampled uniformly over the space. And just randomly you know, reset some of the particles instead of where I was projecting it and going, just randomly reset the particles to somewhere over the space and then continue that.

So your importance resampling, instead of sampling M states, M particles will sample numbers M minus k where k is the number of particles you add to refresh the particle set. And you do not have to do this every time step. If you do this every time step then you are actually losing the whole effect of the particle filter. You do this, say, every few time steps, maybe every 10 time steps, or every 100 time steps.

Or you could do something more cleverer. You could look at the variance of the particles. If you find that the particles are all you know, kind of clustering around one state, then you could say, hey, I am going to do a resampling. Then I am going to do this refresh of the particles. So you could say, okay, the variance of the particle space is getting low. I am going to just stop and resample or add a new set of particles; not resample, it is refresh.

I am going to just refresh the particles and so, I am going to take k new particles and then add them randomly over the space, and then I will start updating. Hopefully, it will cover up for any, any problems that could be there.

So particle filters are very powerful. They allow us to model arbitrary distribution, they are simple to implement. And then, in many cases, people have very, very efficient implementation of particle filters, so that you can do this at run-time so the robot movement is not impeded by all the particle computations that you have to do.

And other thing is I did not say anything about what the motion model and what the observation model should be. I mean they could be anything, I do not have to assume any specific form for it as long as it is easy to compute so that I can plug that into my lines 4 and 5 of my algorithm and I can do this. As long as it is in some form where it is easy for me to sample from.

And there are some caveats, there are some things that you have to be careful about but given advantages that you get of using particle filters, so it makes sense to look at these challenges and make sure that you are addressing them. And like I said, there are easy ways to address these challenges and there are more robust ways to do this as well. And so, we just go ahead and use particle filters. I will stop here.