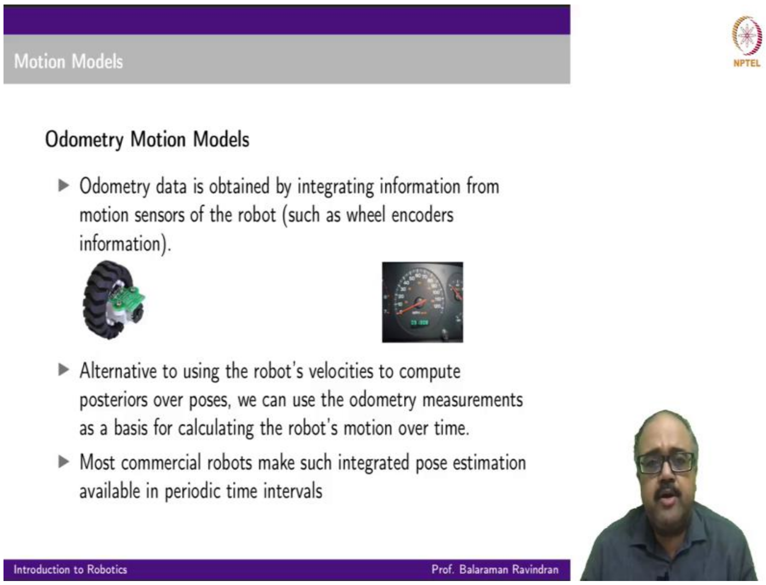
Introduction to Robotics
Professor. Balaraman Ravindiran
Department of Computer Science
Indian Institute Technology, Madras
Lecture No. 38
Odometry Motion Model

(Refer Slide Time: 0:14)



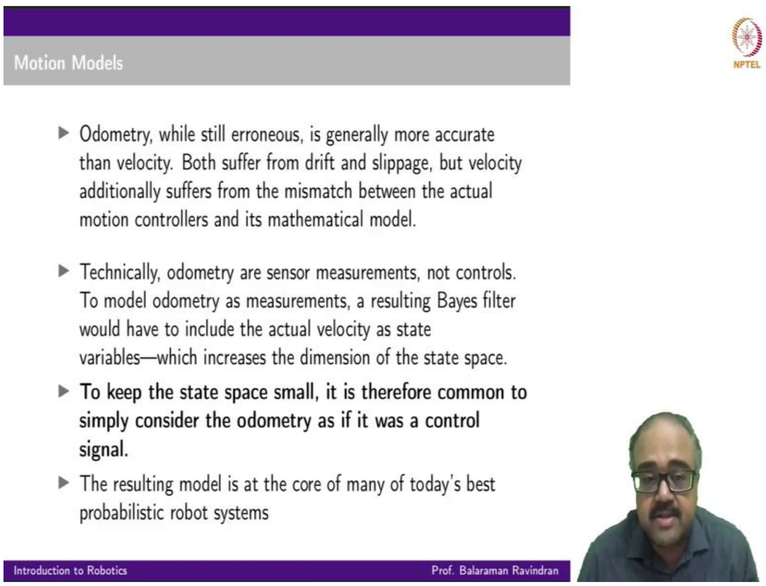
So, in the last lecture we look at the velocity motion model and so today we are going to look at the Odometry motion model. So, the Odometry information is essentially information on the how much the robot has moved, how much the position of the robot has change and the Odometry data is typically obtain from looking at motion sensors that are there on robot it will most popular kind of a motion sensor that we uses the wheel encoder.

So, looking at this kinds of wheel encoder information so and so for, you integrate this information in order to get an estimate on how much you actually moved. So, one way of thinking about it is actually a measurement. It is not really a control it is actually a measurement but another way of thinking about it that you look at this as the effect of the control action. It is effect of control action that was applied to the robot.

And so you use this as a surrogate for the actual control that was given. So, why do we use this so that is what we are mentioning here that is an alternative to using the robot velocities? The main reason that we end up using this is because the quite often the actual velocities that are imparted to the robot at least the controller that we want, that we are setting it to give a certain velocity that we not always get translated into physical movement.

So, there is too much noise in that so often are the more accurate movement estimate or obtain by integrating the motion sensor information integrated from the measurement information rather than from just the control information. And also many commercial platforms make this kinds of accumulated information available to you, to make decision on.

(Refer Slide Time: 2:11)

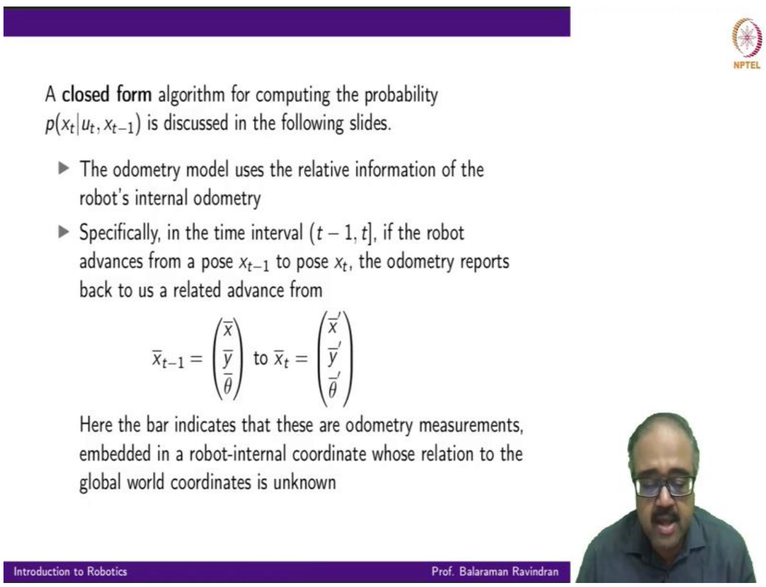


So, while the measurement could still be erroneous, it is actually typically more accurate than the velocity for variety of reasons. So, that could be some kind of drift slippage in the actual operating conditions. And but apart from that velocity model also suffer additional approximation error. So, we have some kind of a mathematical that describe how the velocity maps to the movement, but again there could be mismatches in that as well.

So, well the Odometry information is essentially the motion that actually happens and that kind of allows us to ignore the other errors that are come from this modelling problems. So, technically like I said earlier or Odometry or actually sensor measurement that are not control measurements and so if I actually want to think of the Odometry as measurements itself I might have to add more to the state variables. And therefore, instead of expanding my state dimension include things like velocity and so on so for not just the x, y, and theta like we saw earlier.

So, to avoid including the velocity and the other things just part of the state information because of looking at the sensor measurements. So, we typically take this Odometry as a control signal. We will see in the next couple of slides how this is done. But the Odometry is treated like a control signal and then that allows us to actually define a new kind of motion model that is based on the actual Odometry measurements as suppose to the real control that was given. And so this is quite often in many of the current day systems, this kind of Odometry motion models.

(Refer Slide Time: 4:15)



So, the idea here is that we are going to look at the Odometry signal as the action. So, the ut that we talk about would be the odometry measurements that we make as we see in the next slide. So, as before our goal here when we are trying to build a motion model is to come up with a probability distribution of xt given the previous state xt minus 1 and the control action ut. So, so if you think of the actual dynamics of the system as taking the robot from a pose xt minus 1 to pose xt, I can think of the Odometry as reporting and advanced from x bar t minus 1 to x bar t where the bars, x bar t minus 1 x bar t are actually the measured values of the pose at time t minus 1 under time t.

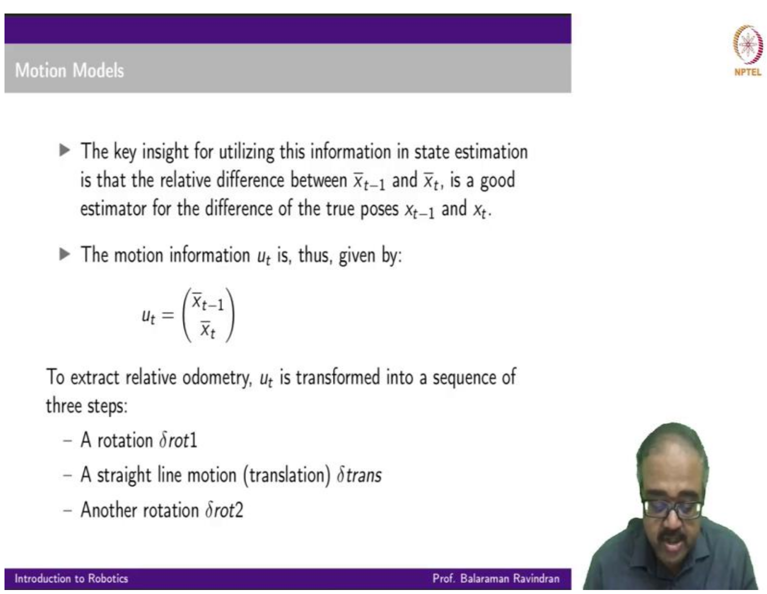
While xt minus 1 and xt or the actual pose at time t minus 1 and time t. x bar t minus 1 and x bar t or the measured pose as measured by the odometry readings at time t minus 1 and at time t. And we denotes these as we did before so where xt minus 1 we denoted as x, y, and theta. And xt we denoted as x bar, y bar, and theta bar likewise, what will do is for xt x bar t minus 1 will denote these readings as x bar, y bar, and theta bar which is basically the x coordinate measured or estimated, the x coordinate measured or estimated from the Odometry readings at time t minus 1 this is a y coordinate estimated at time t minus 1 and this is the orientation estimated at time t minus 1.

And likewise, this is x bar prime is the x coordinate estimated at time t, y coordinate estimated at time t and the orientation estimated at time t. So, this is basically this will be your rotation. So, while the robot actually move from xt minus 1 at time t minus 1 to xt at time the, the Odometry tells us it moves from x bar t minus 1 to x bar t in that time interval.

So, what is the use of this? So, one of the main reasons why this Odometry information is useful even if I am not able to do a correct correspondence between the x bar, y bar, theta bar to x, y, theta, this change that I observe in the estimate from x bar I mean x bar t minus 1 to x bar t. It is similar of very close to the change that actually happened between xt minus 1 to xt. Get that? Even if the x bar t minus 1 and x bar t are erroneous estimate of the poses at t minus 1 and the, I could still say that the change from t minus 1 to t is similar for both the estimated quantities and the actual quantities.

So, from the estimated quantity if I can get the change, then I can apply it to the actual xt minus 1 that I have and also find out what xt is. So, this is essentially this is what allow us to build the probabilistic model for the actual state based on the estimated measurement form the Odometry.

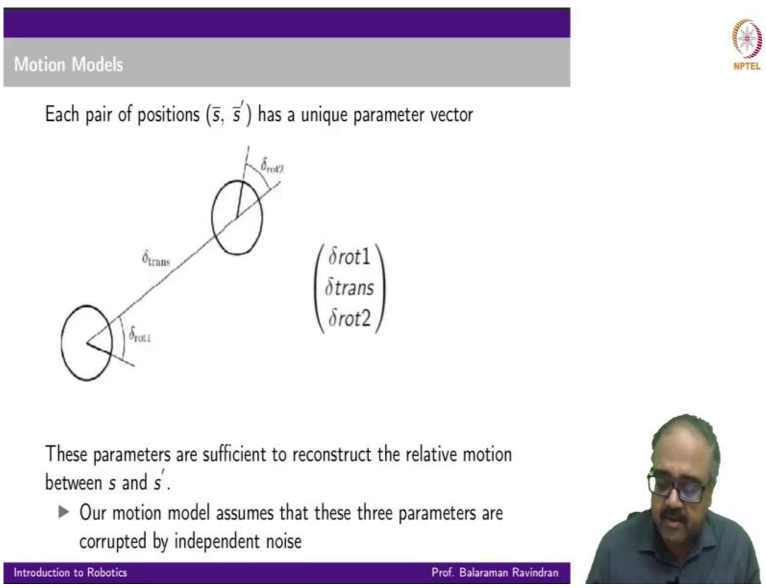
(Refer Slide Time: 7:57)



So, that is basically what we saying here and so the difference is useful. So, now the control itself I am going to now represented by both the x bar t minus 1 and the x bar t basically it is what I really need from the control ut is to figure out what the change was, so instead of actually computing the change and giving that as the ut which is the motion information. So, I am going to say ut basically consist of 2 successive estimated poses which is x bar t minus 1 x bar t that is what my ut is.

So, what I do now is I transform this the relative Odometry information into the sequence of 3 steps which we will call us delta rotation 1, delta translation and delta rotation 2.

(Refer Slide Time: 8:53)

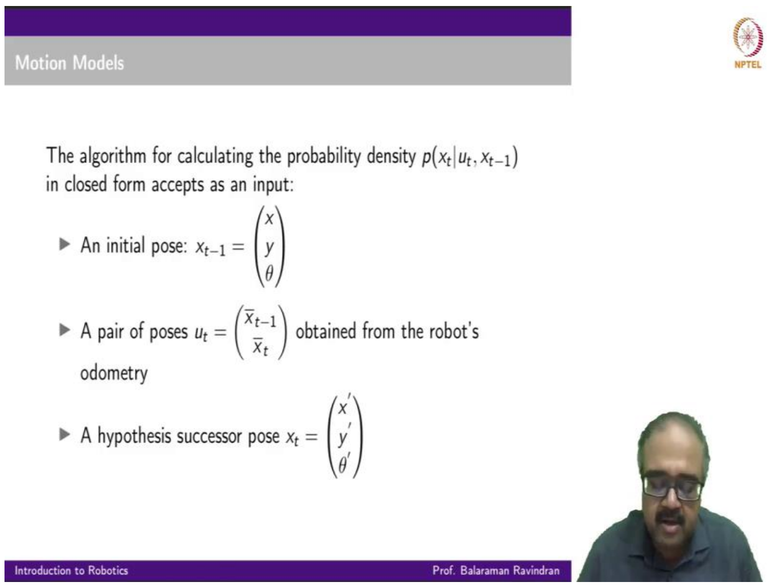


So, what are these? So, delta rotation 1 is essentially taking the original orientation of the robot and moving it so that it faces in the direction of the translation. So, I know this so basically this is essentially my x bar t minus 1 and that is my x bar the, so what I do is this delta rotation 1 is rotating the robot the pose that theta bar so that it now points in the direction of the translation where the direction I know from when a x bar, y bar and x bar prime and y bar prime.

So, now it point in the direction of the translation. Then I do the delta translation so that I move the robot to the destination location and I still have a residual angle that I need to model. So, delta rotation 2 basically rotates the robot to the final orientation which is theta prime this is theta bar prime is the final orientation of the robot from the direction of motion. So, I will take the direction of motion and change it to the final orientation. So, I have decompose my total motion into delta rotation 1 delta translation and then another delta rotation 2.

So, this is essentially how I do it. So, given any pair of positions s bar and s bar prime which are any 2 poses, I can basically reduce it into a unique delta rotation 1, delta trans and delta rotation 2. And for every pairs of state I can basically recover this. And then what I do is just I said did in the velocity motion model, I am going to assume that the errors independent sources of noise for all the 3 and my overall probability is basically given by the product of this 3 noises.

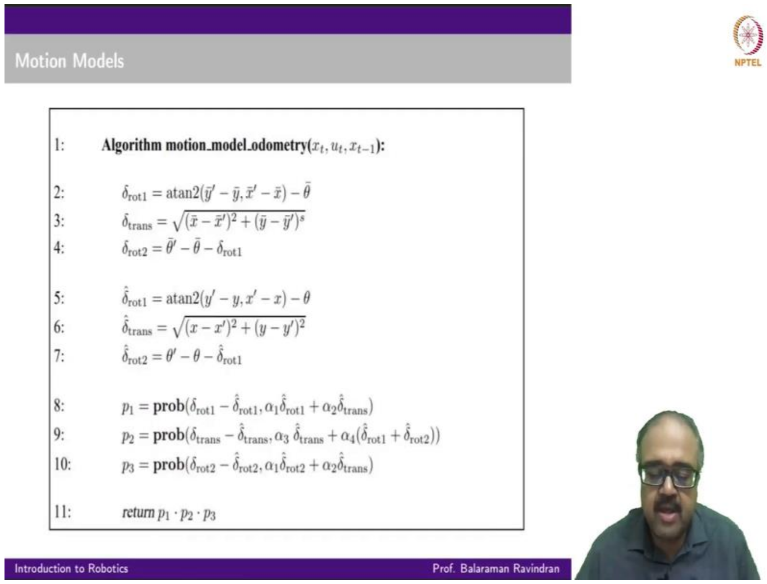
(Refer Slide Time: 10:46)



So, again while we are constructing this algorithm that will output the probability, suppose I give you an initial pose xt minus 1. Control which is ut this case which is 2 sets of measurements xt minus 1 bar and xt bar and a final pose xt, given this, this algorithm is going to return the probability of xt happening when I apply ut to xt minus 1. So, what is applying ut mean here that means that I have done the translation to my state as given by these 2 vectors.

I have my initial pose which is x, y, theta and then I have the translation that delta root 1 and delta trance and delta root 2 as specified by this pair of vector in ut that is my action and then I finally have the successive pose given by x prime, y prime, theta prime. What is the probability that this is the successive pose given that I started with that state and then applied the translation or the applied the action as specified by these two measurement? So, that is basically the problem here.

(Refer Slide Time: 11:58)



So right now this how we are going to do this. So, lines 2, 3 and 4, lines 2, 3, and 4 actually compute the rotation and the translation differences as per the measured pose which is ut. So, this is basically telling me how much has the pose change from x bar t minus 1 to x bar t that is what 2, 3, 4 tells me. 5, 6, 7 tells me how much the position has change from xt minus 1 to xt that is what the cap is.

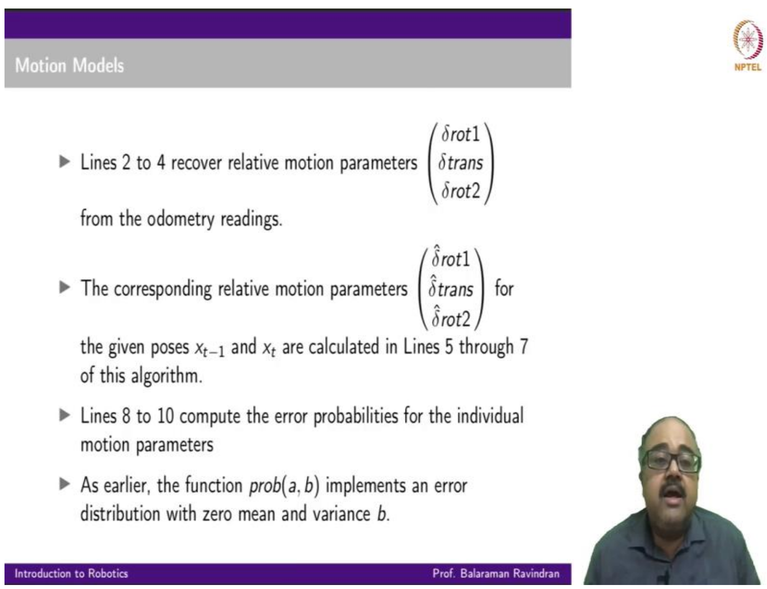
Delta cap root 1 is basically delta root 1 the same expression is delta root 1 but computed on the x, y and x bar, y bar, here it is computed on x x prime, y prime here it is computed on x bar, y bar, and x prime bar and y prime bar and theta bar while 5, 6, 7 I actually use the measurements with the state given in xt and xt minus 1. Now once I have done this basically what lines 8, 9, and 10 do is exactly the same noise model that we had earlier.

So, now this is a 0 mean probability distribution with variances as this. So, I really want my rotation 1 and my rotation 2, rotation 1 delta had to be the same value. So, I basically want my Odometer measurements to be accurate. So, if I assuming the Odometer measurements are accurate, then delta root 1 minus delta cap root 1 should be 0. So, I will model this noise in the first rotation difference by the factor p 1 it basically gives me sample from a 0 mean probability distribution with variants given by alpha 1 root 1 hat plus alpha 2 delta trance cap.

So, this mean how much I rotate and how much I translate and both of this it going to contribute to the noise in the first rotation. Similarly, the both the rotation and the translation errors changes or going to contribute to the error in the translation and finally the translation and the second rotation, the magnitude of those is going to contribute to the error in the second rotation that we have. So, each we are assuming as independent of sources of error.

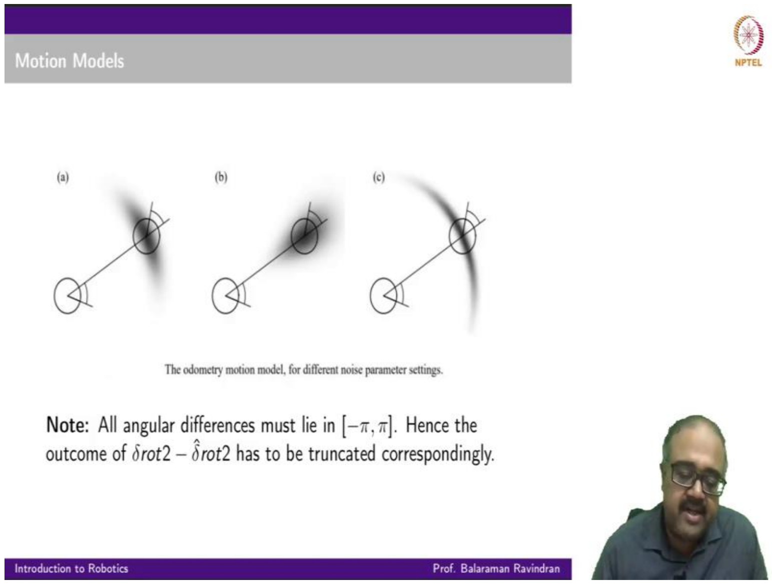
So, I have p1 times p2 times p3 very similar to what we had earlier so it is not very complicated except that the control is given by slightly different way of specifying it which is the odometry information. And we also get slightly different noise model, there it was earlier it depended on the magnitude of the velocities, their noise that we use was depended on the magnitude of the translation and the rotational velocities, but now it is going to be at based on the magnitude of the actual rotation and the actual magnitude of the translation itself.

(Refer Slide Time: 15:06)



So, that this is basically explaining what is happening their lines 2 to 4 I get the relative motion parameters from the odometer readings and lines 5, 6, 7 I get the actual the relative motion parameters from the actual poses that were given and 8 to 10 8, 9 and 10 compute the error probabilities for the individual motions and then finally I multiple all of them to give back the overall probability of and seeing that actual translation.

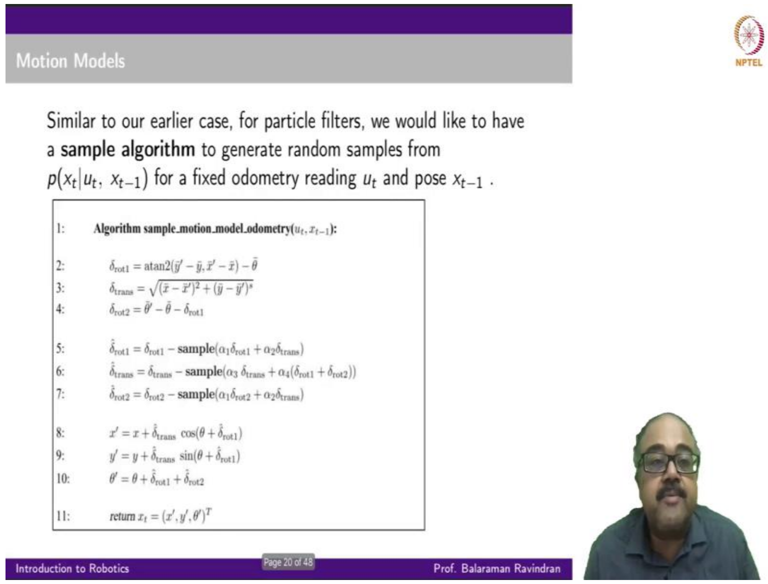
(Refer Slide Time: 15:37)



And you can see the same setting here like we did with the a different alpha settings, so this is again for a normal alpha setting and you can see that both the angular and the translation spread because all the alphas are normally spaced. And here is more translational error and lesser angular error and in c I have more angular error and less translation error. Exactly the same kinds of parameters settings as we had earlier.

So, that actual alpha 1 to alpha 6 values might be different, but the settings are similar. Let notice that the final angular distances differences might lay between minus pi and plus pi. Therefore, we have to make sure that we are truncating everything appropriately.

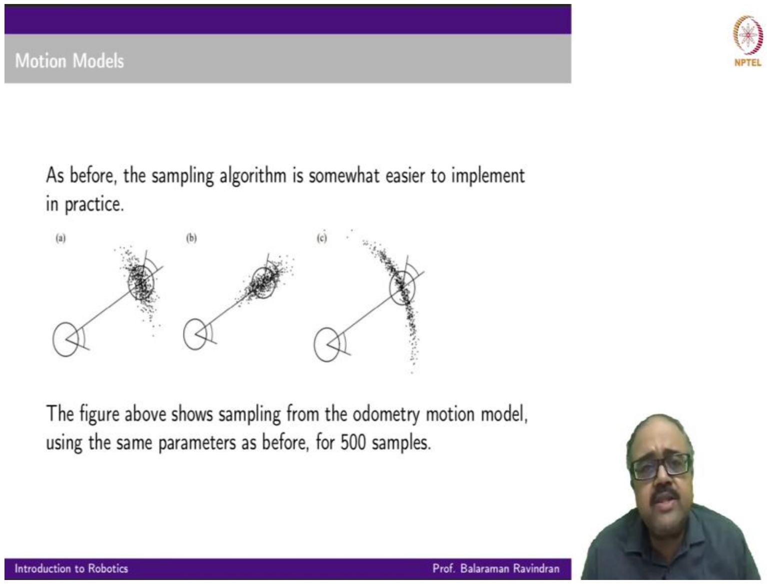
(Refer Slide Time: 16:26)



And just like we did earlier with the particle filter version of the motion model, we can do a particle filter version of the motion model where we first compute delta root 1, delta trans delta root 2. And then we sample a delta prime. Remember when I am using the particle filter version I do not get an xt as an input. So, I cannot compute delta root 1 because I do not have the xt as input but what I do is I randomly sample values for the delta cap, all the 3 delta cap values I sample randomly.

And once I have a sample then I do a deterministic computation of what x prime, y prime and theta prime should be based on the cap values I have computed. And then that gives me the final sample of the state and if I repeatedly call this function I will get many many different samples.

(Refer Slide Time: 17:19)

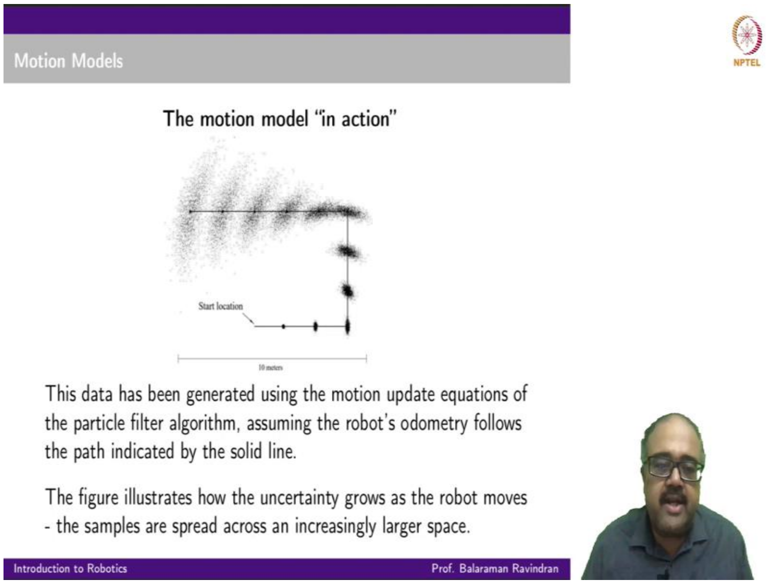


And just like last time you can see the again for 500 samples. This look like a very very similar figure to what we saw the motion model. So, the the challenge with using the Odometry model.is that the Odometry information is available to you only after the motion is completed not before the motion is happen in the filtering use case that we have seen so far.

So, for filtering it is fine because we typically use the motion model only after the actual motion as been completed. But when you are trying to do planning this will see later, we really need to make predictions before the motion is completed. In such cases we have to fall back on using the something like the velocity motion model.

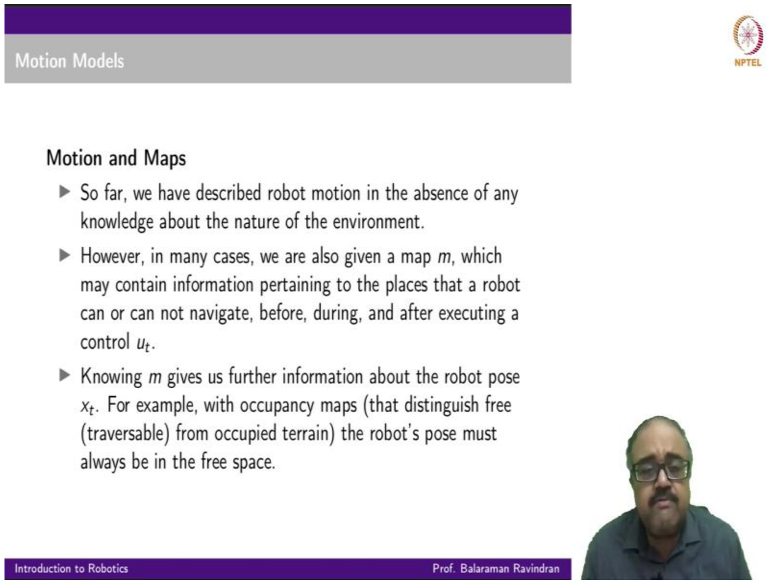
So, the Odometry motion model can be use the in cases where the motion has been completed. And we are just using the particle filter for doing the state estimation or the base filters for the using state estimation in such cases we can use the Odometry model because we really needed to complete.

(Refer Slide Time: 18:23)



So, typically this is something what you would say here I am I have applied particle filter based Odometry model here. And you can see that as a keep moving, so my estimate become increasingly uncertain. This is only based on the motion model, I am not in cooperated any observations here. And once you start in cooperating observations, these things will again start relocalizing but right now just to give you a feel of what happens when I use the motion model alone to make the predictions as I keep moving, the probability distribution keeps spreading out. So, basically started here and I have moved around like that and you can see that the noise always keeps increasing then until I make a measurement so that I can collapse this uncertainty.

(Refer Slide Time: 19:15)

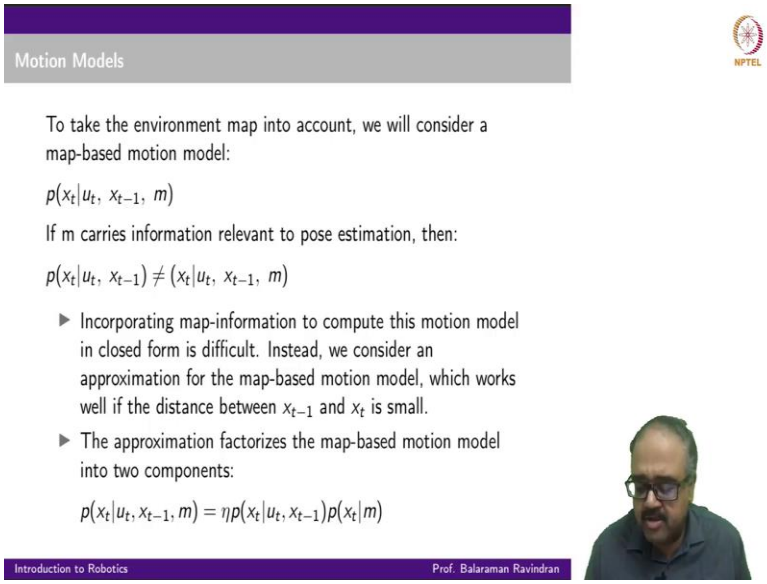


So, just a small note here we will come back to maps in greater detail later on but just to tell you the important of in cooperating information when I am doing this kind of motion models same thing with the measurement model. So, we have described all the motion so far assuming that we have no knowledge about the environment and anything that needs to be captured is somehow captured in the motion model itself but the motion models at we looked are fairly straight forward and simple.

So, they basically look at small linear motion, this is small rotation followed by a translation. It really does not talk about anything that is there in the environment itself. So, in many cases we typically have some kind of a map. The map tells us whatever information that we have about the environment in which the robot is currently moving. And so we look at later there are something called occupancy map so for example that tells us whether the particular location or particular pose is free.

Free meaning that it the robot can actually move over that space or is it occupied and occupied meaning it could be an obstacle it could be a table, chair could be some kind of walls or whatever are the robot is able to go over that space. So, this kinds of occupancy maps will tell us whether this space is free or not and so the robot suppose must always be in the free space. So, knowing the map allow us to further refine our motion model.

(Refer Slide Time: 20:50)

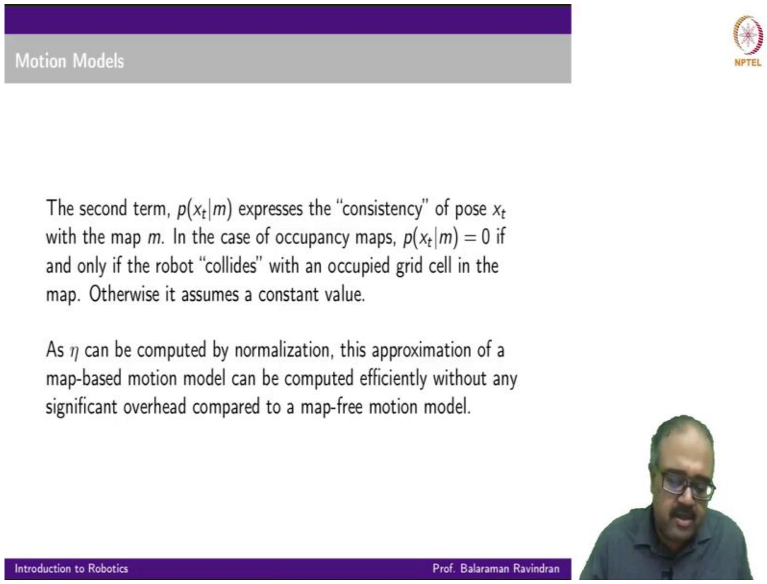


So, now motion model should start looking like this what is the probability of xt given that I have started in xt minus 1, I did action ut and I am operating in a map m. So, I will have to start conditioning by motion models on the map m and if m carries information relevant to the post estimation. So, if m is something that is going to actually effect what I am estimating, then typically ignoring the pose information is going to give me wrong information, wrong estimate of the pose.

So, it can become arbitrarily complex we can arbitrarily complex, so what typically we do is whenever the motion is very small whenever I am going to make a very small translation from xt minus 1 xt. So, what do I mean by that, that means I should be making these predictions very frequently. I should make the prediction very frequently between so every pose I should not wait for the large motion to be completed before it make the prediction.

In such cases I can approximate this probability by 2 things. So, I can look at the original motion model like I had earlier which is probability of xt given ut and xt minus 1 and some kind of a validity model. So, what is the probability that xt is a valid pose given that I am operating in map m? So, I move from xt minus 1 to xt that is xt a valid pose so given that I am operating in map m. And eta as usual some kind of a normalizing factor so this only checking for the validity of final pose and as long as changes of small. So, they should be they should be fine.

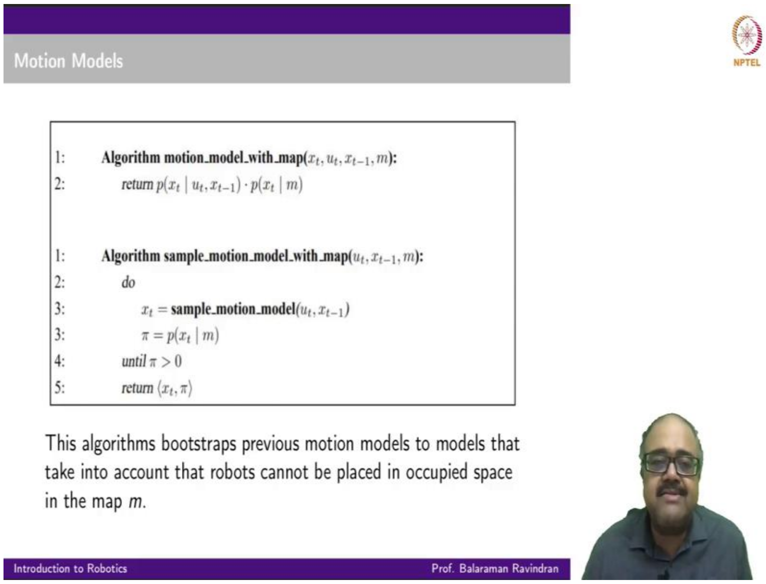
(Refer Slide Time: 22:43)



So, the second term sometimes you can also think of it as a consistency of the pose with respect to the map m for example in the case of occupancy map. So, the probability of xt given m is 0. If the robot collides with an occupied grind otherwise will give it some small constant value depending on what the normalization factor is. This can become your probability estimate so becomes very easy case now.

So, xt given m is if xt is actually occupied grid cell in m, then the probability 0 xt is not an occupied grid if it further map says that xt is free in, then it will have some constant value. So, it becomes easy so the computation does not the in complex. So, basically the original motion model it will be multiplied by 0 or by a constant and the constant could protesbly observed into eta. So, it can either take it as being multiplied by 0 or multiplied by 1.

(Refer Slide Time: 23:47)

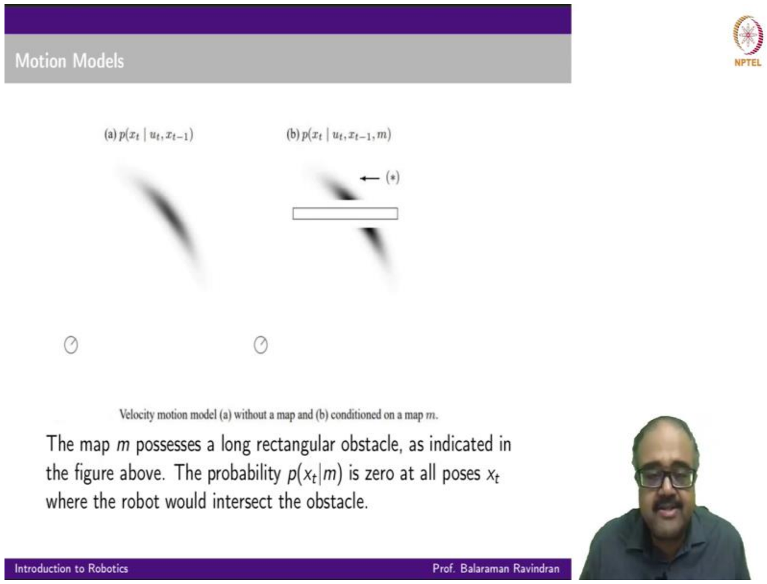


So, here is the simple example and so I take an xt which the original sample of the motion model and then I look at the probability of xt being valid given m. If the probability is greater than 0, then this is a valid pose so I will return that xt, pi to my motion model. If my pie is 0 that means that wherever I moved the sample that have generated for xt as an outer valid pose this is then keep sampling until I get a valid pose final pose xt.

So, this technique is call rejection sampling. So, I have a complicated distribution to sample from but the complication can be broken into 2 part they have a simple distribution which is the motion model that the simple motion model at they can sample from but then I have certain exclusion. So, whenever I figure whenever I hit one of those excluded samples under the model. I just ignore it, I rejected and I keep drawing more samples.

So, the effective sampling distribution would not be the one given by the motion model itself that we had earlier but where the probabilities corresponding to these xt's which have 0 probability and this distribution have been set 0. So, this called rejection sampling it is a very easy way of accounting for this kind of corner rejection cases without making the base sampling distribution more complicated.

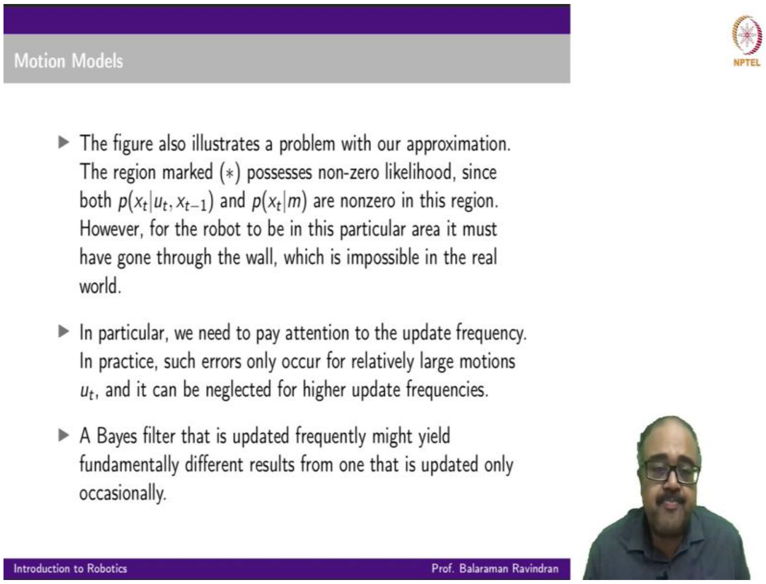
(Refer Slide Time: 25:21)



So, this is about something like. So, my original motion model let us is this gives me the probability distribution as indicated here in this figure. And let us say that there is actually obstacle here that is wall or a partition here that I cannot occupy. So, there will be some small region around the partition because my robot has a volume. So, there is a small region around the partition which the robot cannot occupied all those states has to become 0 probability.

So now, instead of sampling from this distribution for my next state I will sampling from this distribution note that this parts are become actually denser. So, this are become darker therefore the probability mass here is higher than the probability mass here. And so everywhere else the probability would be 0. So, this still there is a problem look at this region which is I have denoted by a star, this region still there is a problem because even thou according to the map this is a valid region for the robot that is standing here, for the robot that is standing here to reach this region it had to goes through the obstacle. So, that is the challenging part.

(Refer Slide Time: 26:43)



So, this what we basically saying here so in the real world that is not possible. So, if our updates of small. If I am basically I am updating very frequently then this can kind of situations are going to very rarely occur in practice because as soon as I move little bit I would not find I can never find myself on the other side of the obstacle. I will find myself in a state where the where I am actually hitting the obstacle therefore I will, I will not move I have to look at different way of a completing the motion.

It is only because I am looking at a larger translation here. I am actually these kinds of infusible cases and even in other cases we have sometimes we might have to actually check for the entire path if it is collision free. So, sometime you have to do a very expensive check to make sure that we are looking at valid translations, the valid motions. So, we have to be very careful about choosing our update frequency for the filter as well.