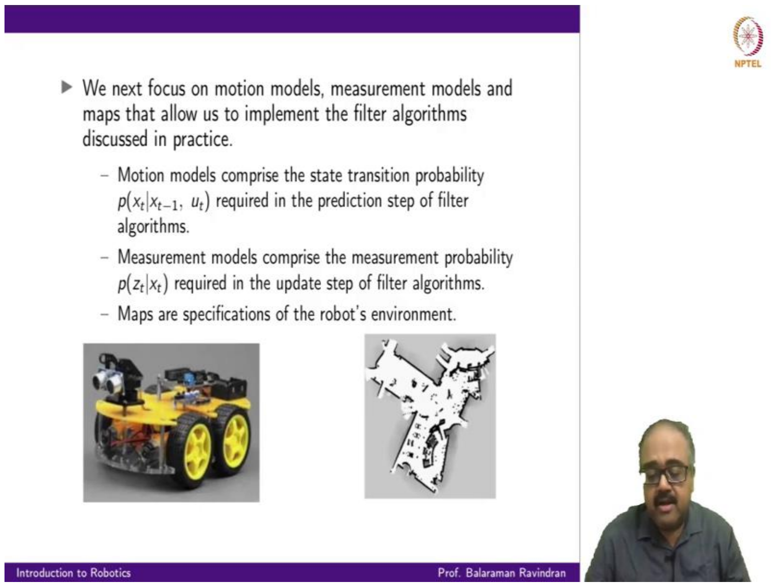
# Introduction of Robotics
**Professor Balaraman Ravindiran**
**Department of computer Science**
**Indian Institute of India, Madras**
**Lecture 37**
**Velocity Motion Model**

Hello everyone and welcome to week 11 of the Introduction to Robotic course and so we are continuing looking at the CS aspect of the course.

(Refer Slide Time: 0:27)



- We next focus on motion models, measurement models and maps that allow us to implement the filter algorithms discussed in practice.
  - Motion models comprise the state transition probability $p(x_t|x_{t-1}, u_t)$ required in the prediction step of filter algorithms.
  - Measurement models comprise the measurement probability $p(z_t|x_t)$ required in the update step of filter algorithms.
  - Maps are specifications of the robot's environment.

So far we have been looking at recursive state estimation and we talked about you know various assumptions we make on the believe model, the motion model and so on for and derived different classes of filters. So this week we will look at a set in other components that we need to allow us to implement these algorithms that we discussion. So we need to really look at how we are going to come up with the motion models. And how we are going to come up with measurements models.

And as well as motion of a map that gives us the specifications of the environment in which we expect the robot to operate in. So the motion models give us if you remember the probability of xt given xt minus 1 and ut, where xt minus 1 is the state at t minus 1 and ut is the action that you perform at time t and xt is the resulting state. So this is the model that we use in the prediction set of the filter algorithms.

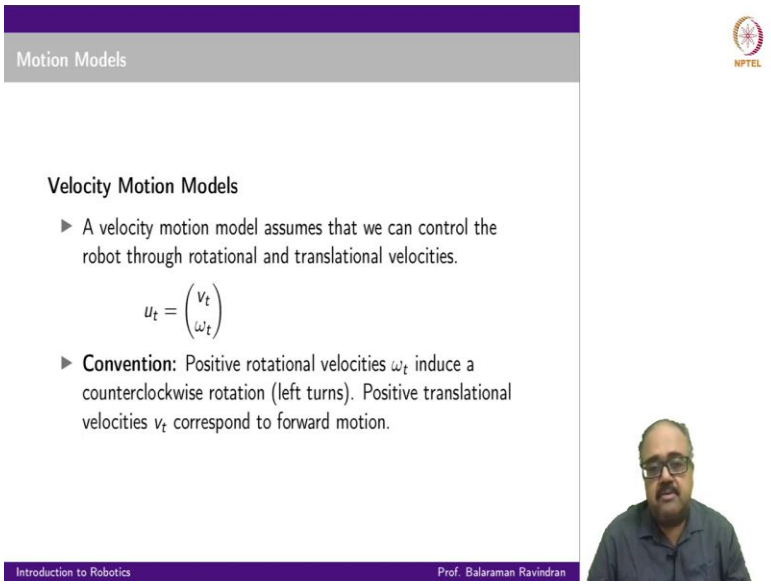
And then the next thing is the measurement model that tells you what is the probability that I will make a specific measurement zt given that I am in state xt. So remember that xt and zt and even ut can be vectors. They are not necessarily scalers even the many of the illustration that we saw where all with the scaler quantities just to make the presentation easier. And again maps like you can see here, a sample here or some kind of representation of the robot's environment and that we will use in reason of the filtering step.

And as we will see later in terms of planning, motion planning and I am trying to design how get from point A to point B in a particular environment. Then we using the notion of a map and most of the maps that we will be looking at are what our 2D maps. When we looking mostly at the 2 dimensional projection of the work space and not necessarily 3D maps and these are usually sufficient for motion planning.

And some forms of localization that we will talk about later. But then we could have more complex maps as well depending on how complicated your work space is. So what I am going to do is first talk to you about certain kinds of motion models. Next we will talk about maps and then we will go on to talking about measurement models and which again as we have done, when the filtering case also.

I will be talking about very specific examples but then depending on what your application is you might want to use more or less complicated model. And that you will have to pick up as you go along.

(Refer Slide Time: 3:22)



Motion Models

**Velocity Motion Models**

- A velocity motion model assumes that we can control the robot through rotational and translational velocities.

$$u_t = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix}$$

- **Convention:** Positive rotational velocities $\omega_t$ induce a counterclockwise rotation (left turns). Positive translational velocities $v_t$ correspond to forward motion.

Introduction to Robotics — Prof. Balaraman Ravindran

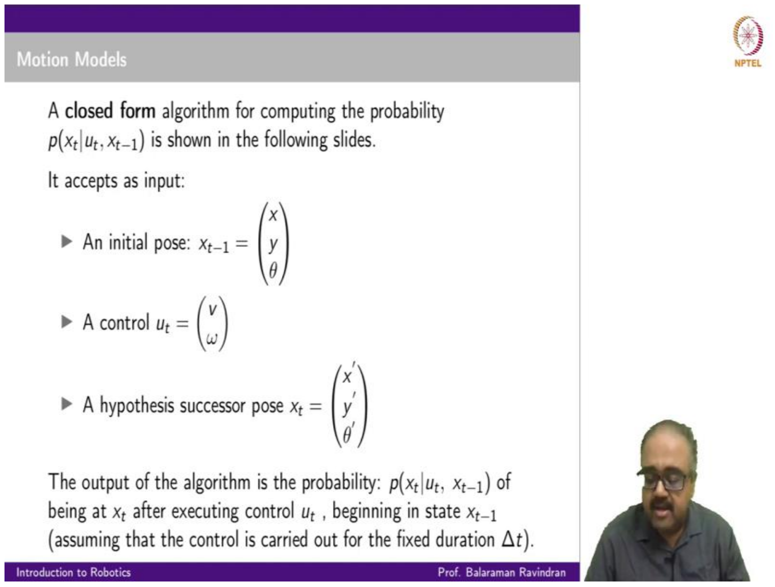
So the first thing I am going to talk about or what are call velocity motion models. A velocity motion model assumes the following. So I am going to assume that by robot state is described by some position it could be x coordinate the y coordinate of the robot and some kind of an orientation theta. So this is a standard 2D robot location and post representation and I am going to assume that the control to this robot is given through rotational and translation velocities.

We could think of control as being specified through a multitude of methods, it could be through torques to motors, it could be through final displacements, or like we see here it could be through the velocities that result from the controls that we apply from the changes to the derives and so on so forth. What is the resulting velocity? So I am going to assume that you have the translational velocity vt and you also have the rotational velocity omega t and this constitute your action at time t.

And the convention we will assume though  we are  not going to involve into it too much. So positive rotational velocities inducer  counter clockwise rotation and negative rotational velocities induce a clockwise rotation. So clockwise would be this counter clockwise would be like that, counter clockwise would be like this. And positive translation velocity vt as is normal correspond to forward motion.

And whichever is the direction orientation that the robot is facing in, it will move forward along that line. So that is what the forward motion get over means.

(Refer Slide Time: 5:11)



## Motion Models

A **closed form** algorithm for computing the probability $p(x_t|u_t, x_{t-1})$ is shown in the following slides.

It accepts as input:

- An initial pose: $x_{t-1} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$
- A control $u_t = \begin{pmatrix} v \\ \omega \end{pmatrix}$
- A hypothesis successor pose $x_t = \begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix}$

The output of the algorithm is the probability: $p(x_t|u_t,\ x_{t-1})$ of being at $x_t$ after executing control $u_t$ , beginning in state $x_{t-1}$ (assuming that the control is carried out for the fixed duration $\Delta t$).

Introduction to Robotics Prof. Balaraman Ravindran

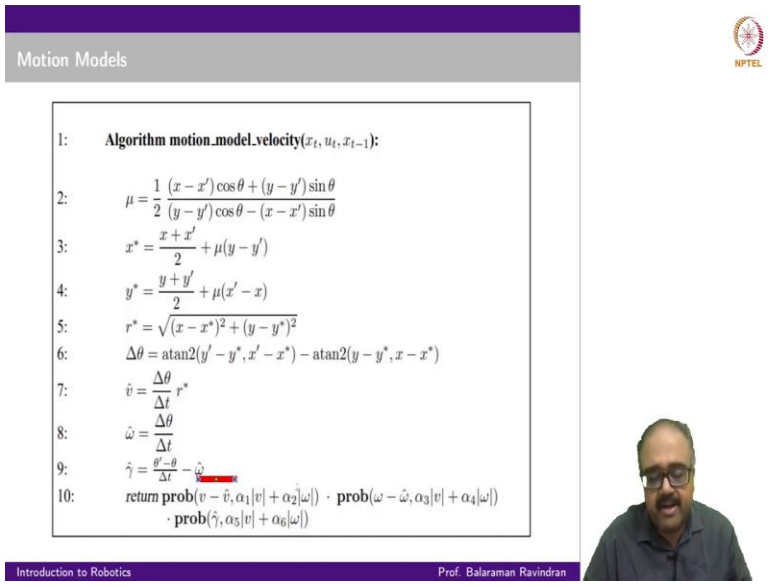
And so what we will do now is look for some kind of a closed form algorithm for computing the probability p of xt given ut and xt minus 1. So this is really what we want this is what the motion model is, so probability of xt given ut and xt minus 1. So this algorithm is going to accept as input an initial pose which is given by x, y and theta under control ut which will denote by v and omega and a potential successor state xt which is given by x prime, y prime and theta prime.

So given all this three this algorithm will return a probability that x prime, y prime, theta prime is resulting state by applying actions v and omega in state x, y and theta. So the resulting state xt is x prime, y prime, theta prime and so what is algorithm is going to compute is, how likely is it that if I start from x, y, and theta and apply v and omega to the robot I will end up with x prime, y prime, and theta prime as mistake what is the probability that will happen.

So that is basically what is algorithm is computing. So and the other thing that we are going to assume is the difference between t minus 1 to t is always a fix duration delta t regardless of whether it is 1 to 2 or 15 to 16, whatever is the time step, we will going to assume it is of a fix duration delta t and this allows us to make the model little  simpler. So normally we would have seen this kind of forward, I mean kinematics model in terms of deterministic dynamics given xt minus 1 and ut you know what xt will be and you solve some differently equations and you find that out.

But what we are doing here is trying to make this into problematic model to account for all kinds of non determism that is there in a real system. And this allows us to then directly plugin to the filters that we have seen in the previous weeks.

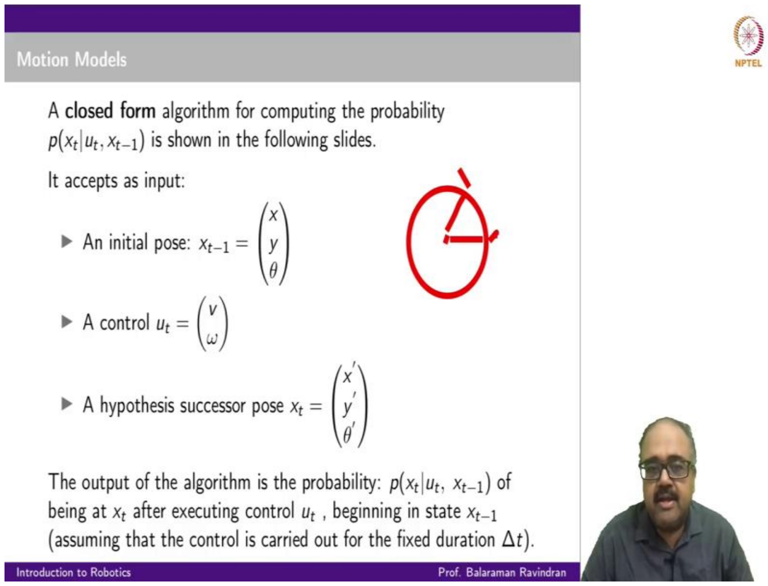
(Refer Slide Time: 7:33)



## Motion Models

```
1:  Algorithm motion_model_velocity($x_t, u_t, x_{t-1}$):
2:      $\mu = \frac{1}{2}\frac{(x-x')\cos\theta + (y-y')\sin\theta}{(y-y')\cos\theta - (x-x')\sin\theta}$
3:      $x^* = \frac{x+x'}{2} + \mu(y-y')$
4:      $y^* = \frac{y+y'}{2} + \mu(x'-x)$
5:      $r^* = \sqrt{(x-x^*)^2 + (y-y^*)^2}$
6:      $\Delta\theta = \text{atan2}(y'-y^*, x'-x^*) - \text{atan2}(y-y^*, x-x^*)$
7:      $\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$
8:      $\hat{\omega} = \frac{\Delta\theta}{\Delta t}$
9:      $\hat{\gamma} = \frac{\theta'-\theta}{\Delta t} - \hat{\omega}$
10:     return $\textbf{prob}(v-\hat{v}, \alpha_1|v| + \alpha_2|\omega|) \cdot \textbf{prob}(\omega-\hat{\omega}, \alpha_3|v| + \alpha_4|\omega|)$
            $\cdot\, \textbf{prob}(\hat{\gamma}, \alpha_5|v| + \alpha_6|\omega|)$
```

Introduction to Robotics Prof. Balaraman Ravindran



## Motion Models

A **closed form** algorithm for computing the probability $p(x_t|u_t, x_{t-1})$ is shown in the following slides.

It accepts as input:

- An initial pose: $x_{t-1} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$
- A control $u_t = \begin{pmatrix} v \\ \omega \end{pmatrix}$
- A hypothesis successor pose $x_t = \begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix}$

The output of the algorithm is the probability: $p(x_t|u_t,\ x_{t-1})$ of being at $x_t$ after executing control $u_t$ , beginning in state $x_{t-1}$ (assuming that the control is carried out for the fixed duration $\Delta t$).

Introduction to Robotics Prof. Balaraman Ravindran

So here is the algorithm, this looks a little complicated, right? So I am going to actually walk you through this slowly. So before that I just wanted to point out one thing so if I have a, I have initial pose x, y, theta and I move in a constant velocity of v and omega throughout the duration delta t. So we are assuming that we start from the location x, y and a specify pose theta. And we are going to move with a constant translation and rotational velocity over a duration delta t.

So what is that going, what is that going to mean? It means that we will be moving in an arc of a circle that is going to be centered at some point. So let us say that you have a so I am going to moving in circle with some center given by some coordinate x star, y star and then I will be marking out and I will be marking out some segment of the circle over which I will move.

So basically I start here then I end here and by orientation here is some theta and by orientation here could be some other theta. So the robot could start here and it will end there and it is starting with some orientation which is theta here and orientation there is theta prime. So because I am moving with this constant velocity I will be prescribe I mean I will be describing some arc of a circle with some center.

So this is essentially what we will be looking at and, so this what this algorithm also captures right now. So what we have to first find out is what is this center of the circle that I am going to take. But notice that this whole idea of moving along this along a circle is fine as long as I have no noise in the vault. If I have some kind of noise and is not going to be exactly the motion that I am going to execute.

So we will have to figure out what will be the actual motion that I will execute as well. So what I am going to do right now is going to assume that. So given that x and x prime where the starting x coordinate and the ending x coordinate of the motion and y and y prime are the starting y coordinate and the ending y coordinate of the motion. And theta was the orientation that I was originally facing, theta was the orientation that I was originally facing.

So remember that my forward movement would be along the direction theta, so the center of this circle that I will be following is given by x star and y star as this equation. So you can actually look at the book to figure out how we arrive at this expression but this is essentially like saying that okay I am starting at x, y and I am ending at x prime, y prime. So a perpendicular drawn at the midpoint of these two points.

A perpendicular line drawn at the midpoint of these two point should go through the center of the circle. So that is essentially the idea here, so that is where we come up with so x star and y star essentially trying to figure out what should be the midpoint of the circle assuming that x and that x, y is the starting point and x prime, y prime is the ending point. And I am having a angle theta to start with. So this is essentially what I have.

And solving for this equation I get mu is it given by this expression and so I compute mu first and then plugged that in here to find what my x star and y star is. And the distance I am from x star and y star when I start basically or basically when I end when you could use either one of these should give me the radius of the circle that I am traversing. And likewise the angle that is subtended by x, y and x prime y prime at x star, y star gives me the total angle that I have traversed from my beginning.

Now, given that so this is the total angle I have traversed I can compute what my v hat and omega hat. So what are v hat and omega hat? These are the actual velocities that I travel with. So what is v and omega that is given as part of ut, v and omega that is given as part of ut are the actual velocities that I applied to the robot. Due to variety of noise parameters in the world the robot actually travels at v hat and omega hat.

Now how do I get this v hat and omega hat? This is basically the rate at which my angle changes times the radius. So this basically gives me the distance I have travel and delta t so r star into delta theta gives me the distance I travel and delta t is the time I took. So this basically actual velocity of travel and this delta theta is the angle that I have traverse according to this and delta t is the time I took.

Therefore, omega hat is the actual angle of velocity that I have. So intuitively I will hope it makes sense, so this x star, y star is the center of the circle that I am traversing when I am moving with velocity vt and omega t will be an omega. That is the assumption here and assuming that this will be the circle that I am traversing assuming that I have constant velocity for the time delta t constant angle of velocity and translation velocity.

So this is the center of the circle this is the radius of the circle, so this is the total angle I have traversed. And this is the effective translation velocity and effective angle of velocity. Now what is going to happen is because I am assuming that. So I am traveling with this constant translational and angular velocity. And my entire computation has basically assume that I am going to end up but x prime, y prime I never looked at theta prime, I never looked at theta prime.

Therefore, this model up till line 8 does not really ensure that I end up at theta prime, it ensures that I end up at x prime and y prime assuming I am ending up x prime and y prime is what it is doing the computation, it never assumes how did I end up with theta prime. So for that what we do is we look at the rate at which so we look at how much angle, angular velocity we have already exclude.

And then, we compute a correction which we call gamma prime. So this correction gamma prime is the correction that we make to the, the effective angular velocity. So that I end up with theta prime and not theta plus delta theta, is that make sense? So theta plus delta theta is what I am right now modeling here. So I am saying that delta theta is the angle that I would have moved if I start off at x, y with angle theta end apart x prime, y prime.

I am assuming I would have acting my pose should have change by delta theta but because that is noise in the angular rotation also I end up at some theta prime.  Now theta prime did not necessarily be delta plus, I am sorry did not necessarily be theta plus delta theta. It is independent value that we know that is given to you as part of xt. So account for the noise in the angular rotation, so we take the actual angular distance that is traversed.

Find out actual angular velocity should be subtract the number that we already computed which is omega hat and call this correction to the angular velocity has gamma hat. So my actual model now is my real translation velocity is v hat. My real angular velocity is omega hat plus gamma hat but action that I have applied is v and omega.

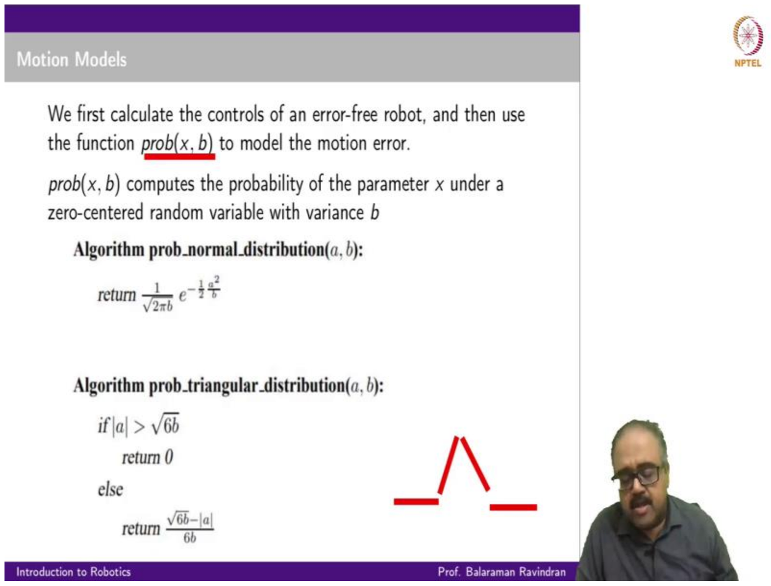
So now the question that we have to really ask ourselves is, hey if I apply v and omega what is the probability that the true velocity that I see will be v hat and omega hat plus gamma hat? Is that clear? So now the question the noise question the probability question becomes given that I applied v and omega given that I applied v and omega as the actual actions at time t.

What is the probability that the effective actions that where applied to the robot or v hat and omega hat plus gamma hat. So I really, I would like v hat equal to v and omega hat to be equal to omega. And I am sorry and omega hat  to be equal to omega and gamma hat to be 0 I do not have to apply additional correction at all. I really like my v hat to be v and omega hat to be omega and my gamma hat to be 0, so that case is the ideal noise free case.

But then we would not be here, if everything was ideal and noise free and therefore we are looking at this noisy case. And we are going to model this probability that you apply v what is the chance that the actual velocity is v hat. So I am going to look at v minus, I am going to look at the term v minus v hat and I am going to define a probability distribution for v minus v hat. Remember what would I like it to be ideally I would like it to be 0.

So what this function probability is it models some probability distribution that has a 0 mean which is most likely thing, so 0 mean I would like to have it, I like it have 0 mean and some variants given by this term.

(Refer Slide Time: 18:44)



## Motion Models

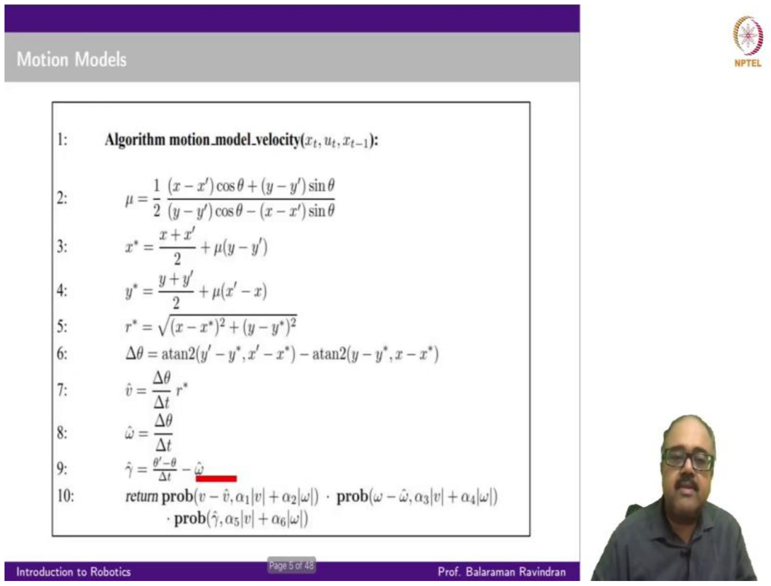
We first calculate the controls of an error-free robot, and then use the function *prob*$(x, b)$ to model the motion error.

*prob*$(x, b)$ computes the probability of the parameter $x$ under a zero-centered random variable with variance $b$

```
Algorithm prob_normal_distribution($a, b$):
    return $\frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2}\frac{a^2}{b}}$

Algorithm prob_triangular_distribution($a, b$):
    if $|a| > \sqrt{6b}$
        return 0
    else
        return $\frac{\sqrt{6b}-|a|}{6b}$
```

Introduction to Robotics    Prof. Balaraman Ravindran



## Motion Models

```
1:  Algorithm motion_model_velocity($x_t, u_t, x_{t-1}$):
2:      $\mu = \frac{1}{2}\frac{(x-x')\cos\theta + (y-y')\sin\theta}{(y-y')\cos\theta - (x-x')\sin\theta}$
3:      $x^* = \frac{x+x'}{2} + \mu(y-y')$
4:      $y^* = \frac{y+y'}{2} + \mu(x'-x)$
5:      $r^* = \sqrt{(x-x^*)^2 + (y-y^*)^2}$
6:      $\Delta\theta = \text{atan2}(y'-y^*, x'-x^*) - \text{atan2}(y-y^*, x-x^*)$
7:      $\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$
8:      $\hat{\omega} = \frac{\Delta\theta}{\Delta t}$
9:      $\hat{\gamma} = \frac{\theta'-\theta}{\Delta t} - \hat{\omega}$
10:     return prob$(v-\hat{v}, \alpha_1|v| + \alpha_2|\omega|)$ · prob$(\omega-\hat{\omega}, \alpha_3|v| + \alpha_4|\omega|)$
            · prob$(\hat{\gamma}, \alpha_5|v| + \alpha_6|\omega|)$
```

Introduction to Robotics    Page 5 of 48    Prof. Balaraman Ravindran

We will see in the next slide, so the probability function so this is what we had probability x, v is going to tell you, what is the probability that you observe e x when the actual distribution of e x is model with a variable that has 0 mean and variants of b. So that is what we are looking at. So if you think about the various terms that we have modeled here. Look at the various terms we have here we have v minus v hat, so remember ideally v hat should be v, therefore v minus v hat should be 0.

So we would like this should be this probability function to have the maximum probability at 0 but the v minus v hat will be some small number. Therefore, if it is close to 0 we would like to have this to have a high probability if it is very far from 0 will act this travel low probability. And how we control the width of this decay width of this probability is given by this expression which is alpha 1 v plus alpha 2 omega.

But alpha 1 and alpha 2 or actually noise parameters that tell you how much the variation in the velocity depends on the actual magnitude of velocity both in the translational case and in the rotational case. So I would be expect naturally if you are moving very fast you have a higher error and whether it is in angular direction or in the translational direction.

So whether so whichever component of the velocity it is, the faster you are the more the error would be and so therefore alpha 1 and alpha 2 in fact all the alphas tend to have a positive small, positive quantities. So they model the influence of the velocities on the errors. So the first term gives you, what is the probability of v minus v hat occurring. So we are already computed v hat we know what v is given to you and so what is the probability that you will that difference.

The second term likewise thus set for omega and here alpha 3 and alpha 4 again or parameters that look at how the magnitude of the velocities effect the error in the angular position. And likewise  the second component of the angular error which is gamma hat and again is given by 0 mean and variants b distribution where again b is a factor that depends both on magnitude of the translation velocity magnitude of the rotational velocity.

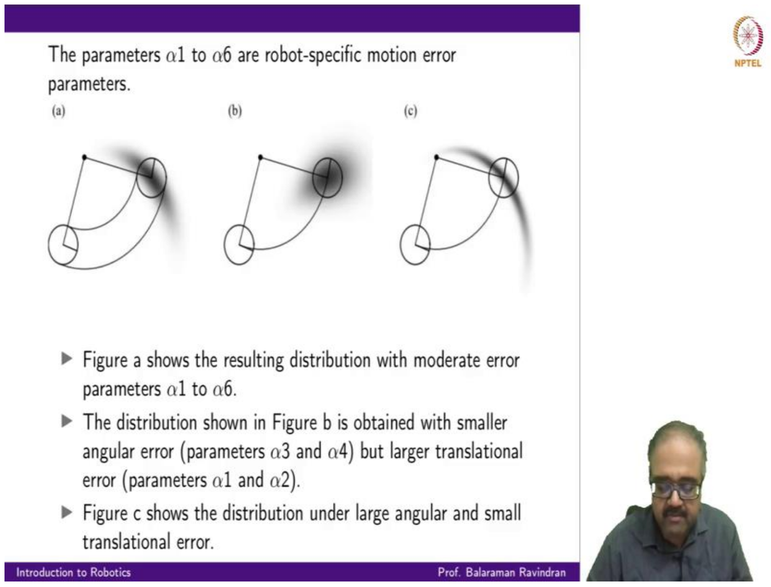
And you have alpha 5 and alpha 6 so all the alphas are positive quantities and each of this pairs of alphas determine the effect of the magnitude of the velocity on the respective errors. And so this probability function where I am looking at the probability of x and given variants b. So the mean of this distribution is always going to be 0. Therefore, I could model this as a Gaussian random variable.

So that gives me this expression gives me the Gaussian as you can see its 0 mean the mu is not, mu does not figure in to be expression at all or I could think of it as a triangular distribution where the profile looks like a equilateral triangle. The profile looks like a equilateral triangle something like this and so outside of this triangles based the probabilities would be 0 and within this it will basically follow this kind linear decay.

So this is basically this would be the 0 mean part, that would be the 0 part where you have the highest probability and then you have the linear decay of the both side and outside of a certain range the probability would be 0 as indicated here. This is call the triangular distribution so you could use the normal distribution which is our friend which we seen it multiple times in the past or we can look at the triangular distribution, here makes sense.

So it is a little complicated, so lots of things were packed into this so basically here we assume that there is initial noise free, there is initial noise free translation. But the actual velocity and that is apply to it is v hat and omega hat plus gamma hat. And then to see that this is the actual velocity components that are applied. So for this to be actual velocity components that were applied, now what is the probability that will happen assuming that the true velocity is that we used where v and omega, so that is basically what this function returns.

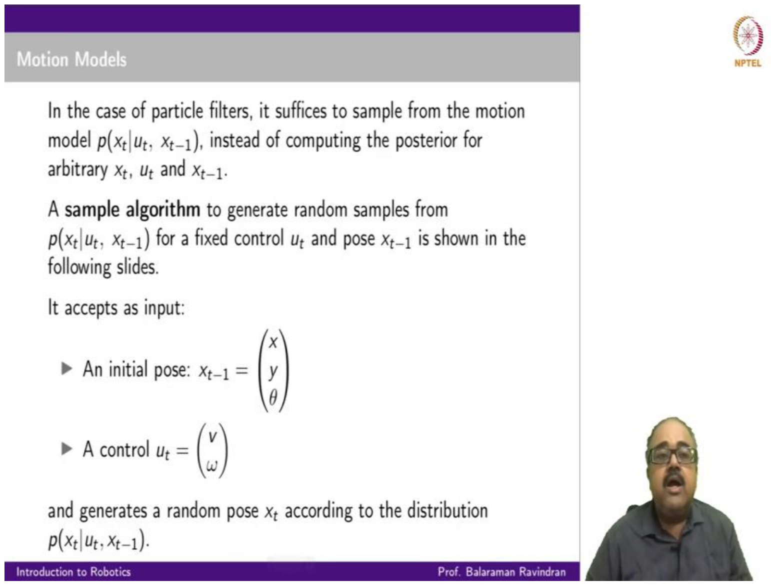
(Refer Slide Time: 24:17)



The parameters $\alpha1$ to $\alpha6$ are robot-specific motion error parameters.

- Figure a shows the resulting distribution with moderate error parameters $\alpha1$ to $\alpha6$.
- The distribution shown in Figure b is obtained with smaller angular error (parameters $\alpha3$ and $\alpha4$) but larger translational error (parameters $\alpha1$ and $\alpha2$).
- Figure c shows the distribution under large angular and small translational error.

Introduction to Robotics Prof. Balaraman Ravindran



## Motion Models

In the case of particle filters, it suffices to sample from the motion model $p(x_t|u_t,\ x_{t-1})$, instead of computing the posterior for arbitrary $x_t$, $u_t$ and $x_{t-1}$.

A **sample algorithm** to generate random samples from $p(x_t|u_t,\ x_{t-1})$ for a fixed control $u_t$ and pose $x_{t-1}$ is shown in the following slides.

It accepts as input:

- An initial pose: $x_{t-1} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$
- A control $u_t = \begin{pmatrix} v \\ \omega \end{pmatrix}$

and generates a random pose $x_t$ according to the distribution $p(x_t|u_t, x_{t-1})$.

Introduction to Robotics Prof. Balaraman Ravindran

So this is one of looking at it where I am actually returning the probability to you and so here is an example. So remember I have told you alpha 1 alpha 6 or the once that control how much noise is there in the model. And so here are 3 different predictions made by this motion model based on different values of alphas. So the first figure has a moderate value for alpha 1 through alpha 6.

It is a moderate value for alpha 1 to alpha 6 so this is the starting position that is the starting position and that is the ending position. So this is x, y and that is theta and this is x prime, y prime and that is theta prime. So now if I look at it so this is basically my probability distribution so you can see that it is spread out both in the translational sense and also in the angular sense both in the translation sense and the angular sense this distribution is spread out.

Now, when I look at the second case here, the distribution is spread out only in the translational sense. And the angular distribution is small, so that means that my alpha 3, alpha 4, alpha 5, alpha 6 or small. And my alpha 1 and alpha 2 have large translation fact alpha 1 and alpha 2 have a larger error in figure b then they doing for that a. And in figure c you can see that I have my alpha 1, alpha 2 or small therefore I have a smaller translational error but I have a larger angular error than I had in a.

So basically my alpha 3, alpha 4, 5 and 6 would be larger in figure c and alpha 1, alpha 2 would be smaller than in figure a. So basically we kind of prediction that you would see for the probability distribution or for various  values of alpha 1 alpha 6. So we look at the proper fully parameter x model in the previous example. A slightly simpler version could have been to use a particle filter.

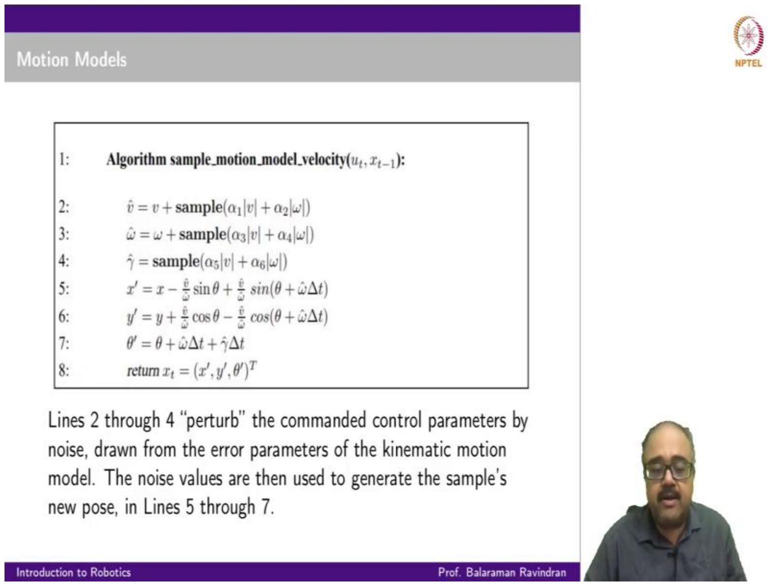
So if you are trying to use a particle filter in particle filter basically if you remember I do not really computed in closed form all I do is draw samples according to the motion model. So as long in I really need to only draw sample according to the motion model instead I have computing a full form distribution like I was doing earlier for every point I have query I have return a value all I need to do is, hey can you return 10 samples from this forward model according to this distribution?

So basically I need to able to draw samples according to probability of xt given ut, xt minus 1 if you look remember when we are talking about the particle filter we never said anything about the motion model. All we really need is a way to draw samples from this distribution. So what this is going to accept. Now what this motion model is going to accept, is the initial pose like an xt minus 1 given by x, y, and theta and the control v and omega.

And it is going to generate the a random pose xt according to this distribution. Notice  that we are going to make the same assumption that we had for the previous model, what are the assumption that we had for the previous model? We are assume that the velocity is actually v prime and v prime is given by v hat and v hat is given by v you know some plus some noise that is proportional to both the linear magnitude of the velocity and the velocity of the angular velocity, magnitude of translational and magnitude of angular velocity.

So v might be the actual angle that actual action we applied but some other v hat and the omega hat are the true velocities with which the robot is moving and that true velocity distribution is what we have to be looking at when we are sampling from for the particles at xt.

(Refer Slide Time: 28:31)



## Motion Models

```
1: Algorithm sample_motion_model_velocity($u_t, x_{t-1}$):

2:     $\hat{v} = v + \textbf{sample}(\alpha_1|v| + \alpha_2|\omega|)$
3:     $\hat{\omega} = \omega + \textbf{sample}(\alpha_3|v| + \alpha_4|\omega|)$
4:     $\hat{\gamma} = \textbf{sample}(\alpha_5|v| + \alpha_6|\omega|)$
5:     $x' = x - \frac{\hat{v}}{\hat{\omega}} \sin\theta + \frac{\hat{v}}{\hat{\omega}} sin(\theta + \hat{\omega}\Delta t)$
6:     $y' = y + \frac{\hat{v}}{\hat{\omega}} \cos\theta - \frac{\hat{v}}{\hat{\omega}} cos(\theta + \hat{\omega}\Delta t)$
7:     $\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$
8:     return $x_t = (x', y', \theta')^T$
```

Lines 2 through 4 "perturb" the commanded control parameters by noise, drawn from the error parameters of the kinematic motion model. The noise values are then used to generate the sample's new pose, in Lines 5 through 7.

Introduction to Robotics Prof. Balaraman Ravindran

So looking at the figure again so you can again see what is, what we are going to do. So we have original v I am going to draw a noise because I remember at 0 mean. So sample basically takes a 0 mean distribution with variants of alpha 1 v plus alpha 2 omega it going to take this as the input this is basically the variants of the distribution I am sampling from. So I draw a sample from that distribution and that gives me my v hat likewise I draw a sample from alpha 3 v plus alpha 4 omega with as a variants.

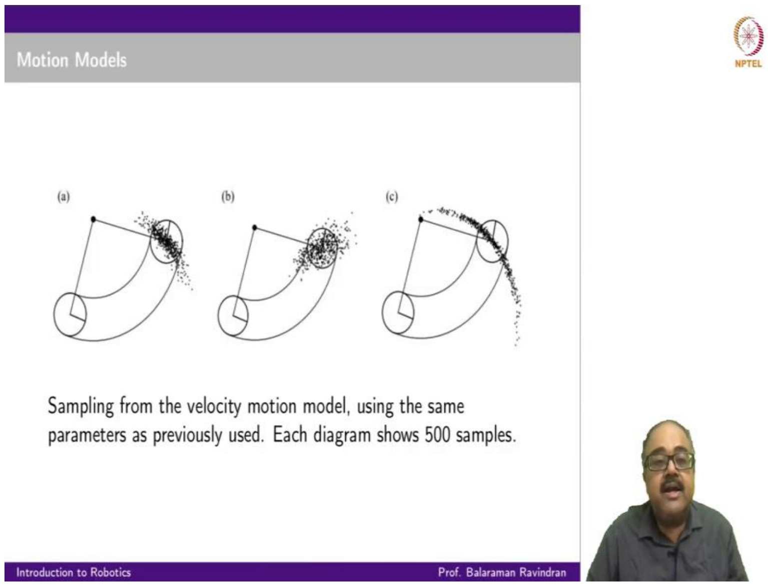
And that added to omega gives me omega hat likewise I have sample a gamma hat as well. Now my actual angle is theta prime is given by theta plus omega hat delta t plus gamma hat delta t.

And x prime, and my y prime are given by these expressions this is essentially the forward model. I start from x, y facing an angle theta and I move with v hat omega hat as my velocities therefore I end up with x prime and y prime as my final positions.

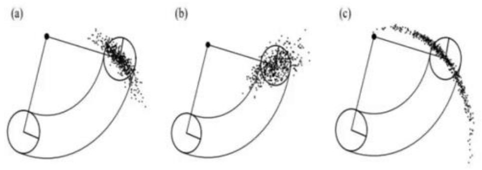
So you can think of lines 2 to 4 as actually doing the stochastic part of it which given a v give me a v hat given a omega give me a omega hat and give me gamma hat. So this is basically the stochastic part of it. So perturb the actual v and omega to get my v hat, omega hat, and gamma hat and once I get that act them actual prediction of x prime, y prime and theta prime a betterment stick equations just the using the kinematic motion model to get me the actual locations and if I call this function repeatedly I going to get different samples depending on what values I sample for v hat, omega hat, and gamma hat.

So this is basically the particle filter version of it. So this is slightly easier to implement and you goes with the particle filter of that we use for the motion tracking earlier.

(Refer Slide Time: 30:40)



## Motion Models



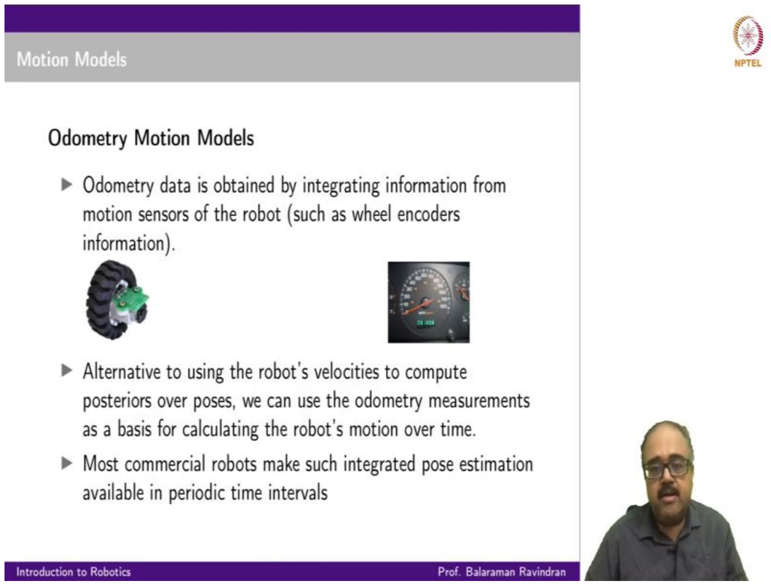
Sampling from the velocity motion model, using the same parameters as previously used. Each diagram shows 500 samples.

Introduction to Robotics Prof. Balaraman Ravindran

And just like what we saw earlier, so if I do this and if I draw day 500 particles so this a, b, and c correspond to the same alpha setting that we had in the previous figure a, b, and c. and you can see here this is for so for moderate error so you see a moderate error values of alphas. You see moderate spread in both the angle and the velocities in this case the velocity error is higher but angular error is smaller.

And in this case the velocity error is smaller and the angular is higher and you can see the particles getting spread out exactly the way we saw the distribution is getting modeled in the previous case. So this is essentially our discussion on the velocity model for the motion model.

(Refer Slide Time: 31:24)



Motion Models

Odometry Motion Models

- Odometry data is obtained by integrating information from motion sensors of the robot (such as wheel encoders information).
- Alternative to using the robot's velocities to compute posteriors over poses, we can use the odometry measurements as a basis for calculating the robot's motion over time.
- Most commercial robots make such integrated pose estimation available in periodic time intervals

Introduction to Robotics Prof. Balaraman Ravindran

So, next one that we are going to look at is the Odometry motion model.