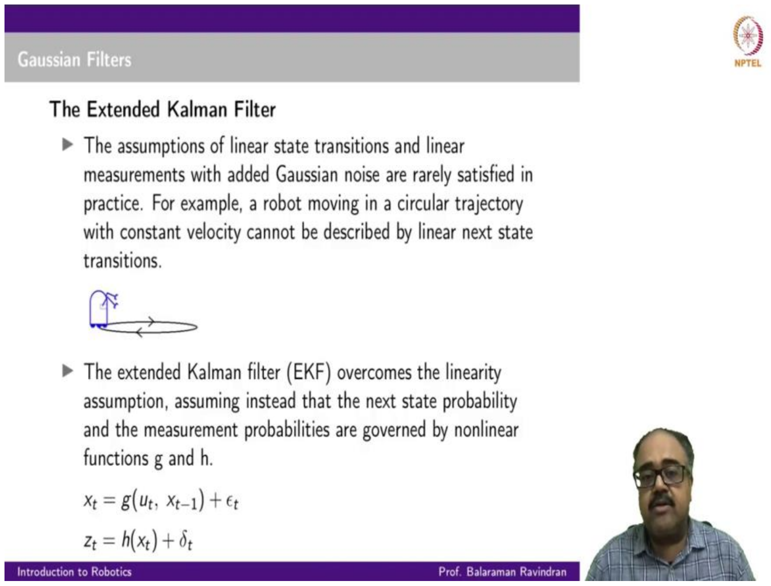
Introduction to Robotics
Professor Balaraman Ravindiran
Department of Computer Science
Indian Institute of Technology, Madras
Lecture - 34
Extended Kalman Filter

(Refer Slide Time: 00:14)



Gaussian Filters

The Extended Kalman Filter

- The assumptions of linear state transitions and linear measurements with added Gaussian noise are rarely satisfied in practice. For example, a robot moving in a circular trajectory with constant velocity cannot be described by linear next state transitions.
- The extended Kalman filter (EKF) overcomes the linearity assumption, assuming instead that the next state probability and the measurement probabilities are governed by nonlinear functions g and h.

$$x_t = g(u_t,\ x_{t-1}) + \epsilon_t$$

$$z_t = h(x_t) + \delta_t$$

Introduction to Robotics — Prof. Balaraman Ravindran

Assumptions that we made for the Kalman Filter, so some things for you to remember is that the Kalman Filter is a base filter algorithm. And the crucial assumptions we made was that the state transitions and the measurement models were both linear. So the state transition was linear in the sense that xt was a linear function of xt minus 1 and a linear function of ut. Likewise, zt was a linear function of xt. So this all, basically, these are the linearity assumptions we made.

And the second assumption we made was that over these linear models, there was an additive Gaussian noise. So we assumed that there was some Gaussian noise added to both the linear transition model as well as the measurement model. And this allowed us to keep the belief distribution as a Gaussian distribution. And that allowed us to localize to a point with some amount of noise around it. So this is assumptions that we made.

And I have mentioned that we use Kalman Filters in practice because these assumptions are often valid and we can get away with it. But sometimes, we need to do something more to get the Kalman Filter to work and so, so, in this lecture, we will look at one such extension to the Kalman Filter.

So here is an example, a very simple example where the linearity assumptions will not hold for us. Suppose you have a robot that is moving in a circle. It could be moving with a constant velocity, does not matter but the dynamics of the movement is not linear. It could be moving at a constant speed but the dynamics cannot be described in a linear fashion.

But one might, you know, think that at any point on the circle, any point on the circle, I might approximate the instantaneous motion of the robot by looking at the tangent to the circle at that point and assuming that the robot is moving on the tangent in a linear fashion. Of course, at every instant, I will have to keep changing the direction of the tangent but for that one particular instant on the circle, I can assume that the dynamics are linear for a very short duration. Is that clear?

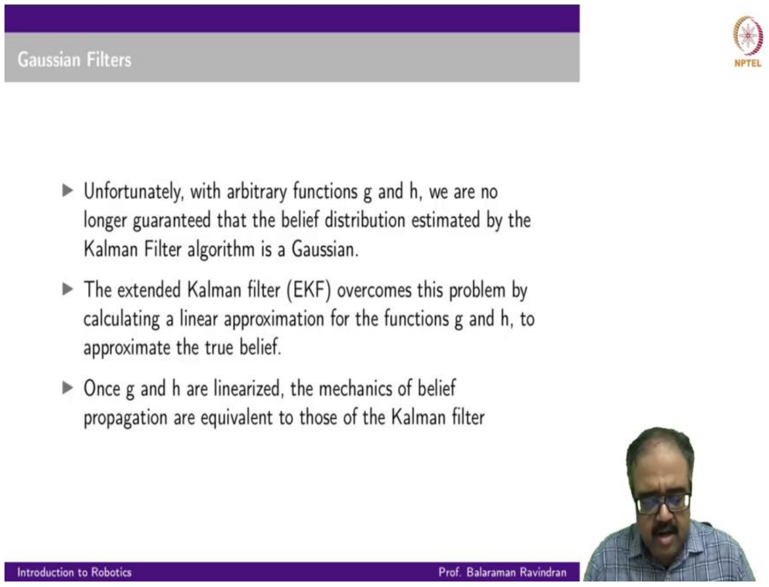
So the whole movement is not linear but at an instant, I can assume that the movement motion is approximated by a linear function. So this is the idea behind what we call the Extended Kalman Filter. So it basically overcomes the linearity assumption by first assuming that, by first assuming that the next state probability and the measurement probabilities are governed by some arbitrary nonlinear function.

So we are going to assume that xt is a function of both ut and xt minus 1, where g is some arbitrary. Again, as before with the Kalman Filter, that is an additive Gaussian noise. So it is still assuming that the noise is Gaussian and it is additive Gaussian noise but we assume that the, the main dynamics instead of being given by a and b, which is like linear multipliers on x and u, I am just going to assume it is an arbitrary nonlinear function g on xt minus 1 and ut.

And similarly, for the measurement case; similarly, for the measurement model, I am going to assume that zt is some arbitrary nonlinear function, h of xt. And again, you have an additive Gaussian noise, which here we denote by delta t, as we did with the Kalman Filter. So is that clear? So we have g, which is an arbitrary function that models the next state probability, and h, this is an arbitrary function that models the measurement probability.

Now, what we are going to do next is just like we did with the circular motion, just like we did with the circular motion, we are going to try and make a linear approximation for these functions at a specific point on the function. So it might not be a great approximation for the whole function but at that point, it is a good approximation of the function.

(Refer Slide Time: 04:28)



Gaussian Filters

- Unfortunately, with arbitrary functions g and h, we are no longer guaranteed that the belief distribution estimated by the Kalman Filter algorithm is a Gaussian.
- The extended Kalman filter (EKF) overcomes this problem by calculating a linear approximation for the functions g and h, to approximate the true belief.
- Once g and h are linearized, the mechanics of belief propagation are equivalent to those of the Kalman filter

Introduction to Robotics Prof. Balaraman Ravindran

So the main reason why we have to do this linearization is as follows. If I have an arbitrary function g and h, so if I am using this, I can still write my update equations. I can still write update equations using g and h, it is just, it does not look very complicated.

But the, but the biggest challenge for me is that the belief distribution, which I start off as a Gaussian distribution, remember, I start off with mu naught and Sigma naught as my belief, the parameters of my belief distribution, and as we keep going around every iteration of the algorithm, I just keep updating the mu and Sigma because my belief distribution stays a Gaussian. So it makes a life a lot easier with the Kalman Filter case.

But in this case, what happens is, we cannot assume that the updated belief distribution would be Gaussian. If it is an arbitrary function, it will most certainly not be a Gaussian. So one way of getting around this is whatever we described right now.

So we create a linear approximation for the function g and h. And now, once I start using the linear approximation in my updates, it just becomes like a Kalman Filter. It is like applying a Kalman Filter at but using different functions, different linear functions, every time I apply it. Where the different linear functions are obtained by approximating these nonlinear equations.

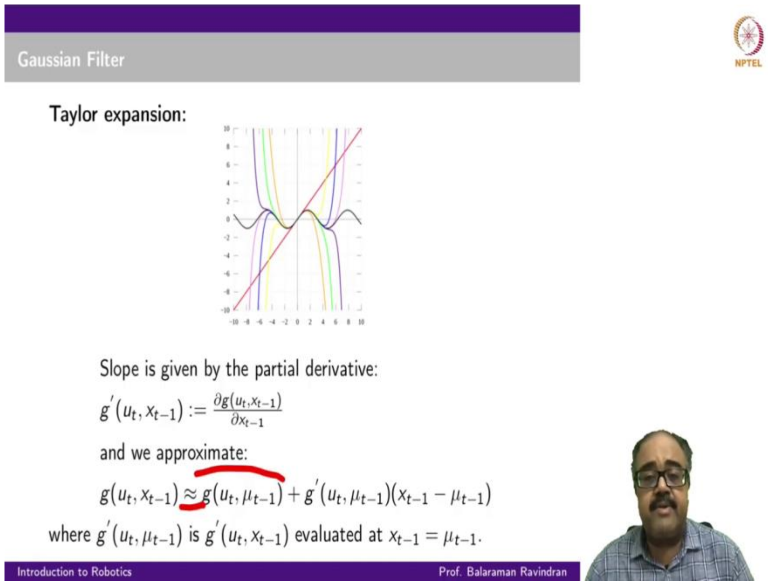
So the actual dynamics of the robot was governed by these nonlinear equations. But when I am doing the belief updates, I just, for a minute think it is linear, so that I can do the same updates or similar updates of what I do with Kalman Filters and leave my belief distribution as a Gaussian distribution. Makes sense?

So the reason we are linearizing this here, for doing the update. The reason we went to nonlinear case because linear assumption was too restrictive for modeling realistic dynamics. And now, we are going back to linearization so that I can get my update to stay Gaussian, updated beliefs to stay Gaussian. So in some sense, I am trying to get the best of both worlds. You see, what is called an Extended Kalman Filter.

So there are like numerous extensions to Kalman Filter, like hundreds of extension to Kalman Filter. People have different kinds of flavors of Kalman Filters and so, each one of these is going to cater to very, very specific assumptions about the robot, the motion model, the world that you are operating in, and the kind of belief distributions you need, and so on, so forth. And almost all of them have this idea that you are looking at additive Gaussian noise. That is what makes it a family of Kalman Filters.

And so, but then the various other parameters that you would need to model would change. So in this course, we are going to look at only the Extended Kalman Filter because it is very useful, and also simple to understand extension. And you are free to look at other material as you go along. So let us look at how we are going to do this linearization.

(Refer Slide Time: 07:39)



Gaussian Filter

Taylor expansion:

Slope is given by the partial derivative:

$g'(u_t, x_{t-1}) := \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}}$

and we approximate:

$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + g'(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1})$

where $g'(u_t, \mu_{t-1})$ is $g'(u_t, x_{t-1})$ evaluated at $x_{t-1} = \mu_{t-1}$.

Introduction to Robotics Prof. Balaraman Ravindran

So I assume some of you are familiar with what is known as the Taylor Series Expansion of a function. Suppose I have a function g of ut, xt minus 1, I can expand it around a point. In this case, I am expanding it at mu t minus 1. I can do that for evaluating the function at mu t minus 1 and keep adding higher powers of the difference. Here, I have just approximated it with just the first term.

So if people remember the Taylor Series, so this would be like g prime ut, mu t minus 1 into xt minus 1 minus mu t minus 1 plus g double prime ut mu t minus 1 into xt minus 1 minus mu t minus 1, the whole squared, and so on, so forth. So that would be the Taylor Expansion. But if you just take the first term, so you can see here, here is a picture of how the Taylor Expansion work like. So that is trigonometric curve, it is a sin, so that is sine theta.

And then, the red line is essentially the Taylor Expansion of sin theta around 0, with just one term, just the expansion that we showed here. So this expansion for the theta equal to 0. And you can see that it is a straight line, but then we have linearized it and it is a pretty decent approximation of the function just around 0.

It is not, not quite that and that is a small deviation, but exactly at 0, it is equal. And so, that is essentially what. We want we have a linear approximation that is good enough very, very close to the point at which we are doing the approximation.

So the higher curves, so you can see that there is a yellow curve and the orange curve, and so on, so forth, these are as you keep adding more and more terms to the Taylor Series Expansion, it becomes closer and closer to the full function. But we are not interested in that. So all we are really interested in is making sure that we have a linear approximation at the point where we want to compute this.

So coming back, now I am going to use g prime and we already, I mentioned this, this is a fairly standard notation. So g prime ut comma xt minus 1 is essentially the partial derivative of g with respect to x; not with respect to u. So it is basically so it is doh g of ut xt minus 1 divided by doh xt minus 1. So it is basically the derivative with respect to x and not with respect to u. So that is what g prime is. And so, we are going to do this approximation.

So now, remember I said, we try to approximate around a certain point. We try to approximate around a certain point. In this in this picture here, we have approximating sin x or sin theta around the equal to 0. So now, I would really like to approximate my g around a point for x. So what is the point that I am going to approximate this around?

So I really need the approximation of the dynamics at t minus 1, so then I can predict what will be the stated time t, correct? So I need an approximation of g at time t minus 1, a linear approximation of g at time t minus 1 so that I can compute what would be the state at time t by using that linear function. I will take t minus 1, whatever is the state at time t minus 1, I will compute the state at time t.

And remember, according to our beliefs, the most likely state at time t minus 1 is mu t minus 1, correct? So the most likely state at time t minus 1 because that is what, that is how our belief is determined. Remember, belief is a Gaussian distribution with mean mu and variance Sigma. The belief is a Gaussian distribution with mean mu and variance Sigma., so the most likely state under the belief is going to be the mean. So for the belief at time t minus 1, the most likely state is going to be mu t minus 1.

So I will take the most likely state, which is mu t minus 1 and I will do a linear approximation at that point. So what it really means is, I will evaluate the Taylor Series at mu t minus 1, but I will use only the first term here. So, so this is so, the approximation for g of ut comma xt minus 1, this approximation is good at mu t minus 1 is equal to g of ut comma mu t minus 1.

Remember that now this quantity, so this quantity, so this quantity is a constant because I know what ut is, ut is not, no longer a function now. I know what ut is I have already applied ut and mu t minus 1 is, is again a specific value for xt minus 1. That is what I think is the most likely state. It is no longer variable, it is actually an assignment xt minus 1.

And what you should remember, even though g is a function of u and x, both, but in this particular instant, I, when I am applying my model g, I have already picked my ut, I am not using g to pick ut; already picked my ut. So ut is a constant because it is already given to me when I am doing my belief updates. And mu t minus 1 is a constant that comes from my previous belief.

So now that I have plugged in specific values for x and u, I can actually compute what g is at this point. So this is actually a specific, specific quantity. It is no longer a function. So this is basically some quantity that is in the x space, that is, in the state space; it is a specific state, this is no longer a, this is no longer function.

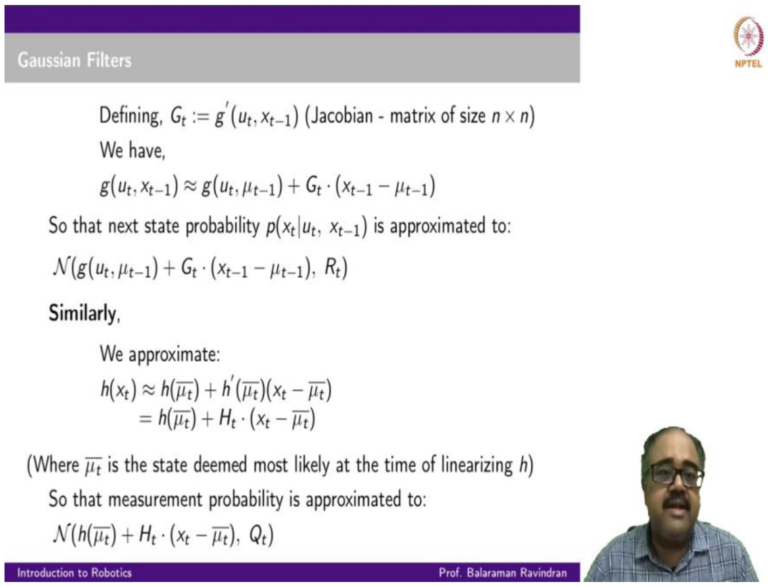
And then, what I am doing here is this is the first term in the Taylor Series expansion. So I am going to take the derivative of g with respect to x. And then, I will multiply that with a difference of xt minus 1 minus mu t minus 1. So this is the part that gives me the functional form. So it is now, it is a functional form in terms of xt minus 1.

And so, notice that I am using a specific notation here. So I use g prime of ut comma mu t minus 1, what it really means is that I take the derivative, I take this, I compute this derivative and I substitute mu t minus 1 for xt minus 1 here, and I compute the value of the gradient at that point. So this also is a constant. So that also is a constant.

So now I converted my nonlinear function g ut xt minus 1 to some kind of a constant plus g prime, which is a constant times xt minus 1 minus mu t minus 1 which is a constant. So basically, I have converted that into some kind of equation, which is linear in xt minus 1. So all the higher-order powers of xt minus 1 that could have been there in this have vanished because of the Taylor Series approximation.

Remember that this is actually an approximation and it is good to go at mu t minus 1 for a very, very short interval. Not throughout the function. I hope that is clear what we are doing with the linear approximation. So now that we have this approximation, we can go ahead and try to start writing our update equations.

(Refer Slide Time: 15:12)



Gaussian Filters

Defining, $G_t := g'(u_t, x_{t-1})$ (Jacobian - matrix of size $n \times n$)

We have,

$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t \cdot (x_{t-1} - \mu_{t-1})$

So that next state probability $p(x_t | u_t,\ x_{t-1})$ is approximated to:

$\mathcal{N}(g(u_t, \mu_{t-1}) + G_t \cdot (x_{t-1} - \mu_{t-1}),\ R_t)$

Similarly,

We approximate:

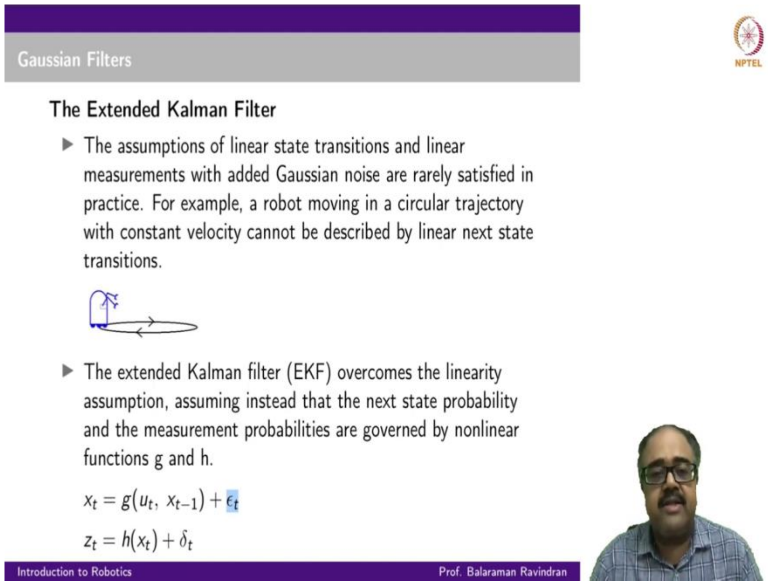
$$h(x_t) \approx h(\overline{\mu_t}) + h'(\overline{\mu_t})(x_t - \overline{\mu_t})$$
$$= h(\overline{\mu_t}) + H_t \cdot (x_t - \overline{\mu_t})$$

(Where $\overline{\mu_t}$ is the state deemed most likely at the time of linearizing h)

So that measurement probability is approximated to:

$\mathcal{N}(h(\overline{\mu_t}) + H_t \cdot (x_t - \overline{\mu_t}),\ Q_t)$

Introduction to Robotics   Prof. Balaraman Ravindran



Gaussian Filters

The Extended Kalman Filter

- The assumptions of linear state transitions and linear measurements with added Gaussian noise are rarely satisfied in practice. For example, a robot moving in a circular trajectory with constant velocity cannot be described by linear next state transitions.
- The extended Kalman filter (EKF) overcomes the linearity assumption, assuming instead that the next state probability and the measurement probabilities are governed by nonlinear functions g and h.

$x_t = g(u_t,\ x_{t-1}) + \epsilon_t$

$z_t = h(x_t) + \delta_t$

Introduction to Robotics   Prof. Balaraman Ravindran

So I am going to assign capital Gt, right, can be equal to g prime ut comma xt minus 1. So that is basically a matrix of size n cross n because I am going to take the partial derivative with respect to all the components of x. So n is the size of x, and so g prime of ut comma xt minus 1.

Now, I can rewrite my G, which is what we did in the previous slide, I am just writing it using my Gt notation. I am going to rewrite my g of ut comma xt minus 1 is equal to or approximately equal to g of ut comma mu t minus 1. And as we saw that this was a specific evaluation of the function g plus Gt, which is the Jacobian times xt minus 1 minus mu t minus 1.

So I mean, I presume all of you are familiar with the Jacobian, you must have seen it multiple times already in the in the course. And so, this is essentially the Jacobian times xt minus 1 minus mu t minus 1.

So the next state probability, which is what is our motion model, which is probability of xt given ut comma xt minus 1; can be approximated as follows as a Gaussian distribution. Remember that our next-state distribution was given by, our next-state xt was given by g of ut xt minus 1 plus a Gaussian noise epsilon t.

And if you remember from our previous lecture, so epsilon t is a Gaussian with 0 mean and covariance matrix Rt. So that is essentially the same thing that we are using here. And therefore, since it has a 0 mean, so I will be basically adding this value to it. So that is the mean. So it is the next state probability distribution is basically a Gaussian with the mean given by g of ut comma mu t minus 1 plus Gt times xt minus 1 minus mu t minus 1.

So that is essentially the mean of the Gaussian for the next state distribution and the covariance is Rt as we had earlier. So I hope that is clear. Now that we have the state transition model, we had something similar like this. So what did we have for the Kalman Filter? So we had A plus, so we had A xt xt minus 1 plus B, B ut, ut, there was the mean and the covariance was Rt.

Similarly, we can approximate the observation function h of xt as h of mu t bar. Why are we getting a mu t bar here, why not mu t. We use mu t when we did the next state probability but you are using mu t bar here. So why is that? So you should remember that we use the observation function or we use the measurement function only with bel bar, not with bel.

So when they come to bel bar, I have already updated my most likely location from mu t minus 1 to mu t bar. So I will be doing the linearization of my h at mu t bar because that is where I actually end up using h. I use it with updating from a bel bar to bel; bel bar of xt to bel of xt. So that is what I am using it for. And therefore, we will use mu t bar as the point around which I will linearize the function.

So h of xt, I am approximating this by h of mu t bar plus h prime of mu t bar; h prime is as you can see, is a partial derivative of h with respect to x times xt minus ut bar. This is exactly like the Taylor expansion that we used for g except that it is, it is around mu t bar. And likewise, just like we did earlier of writing Gt as I mean, g prime of , as Gt. Similarly, we write h prime as capital Ht, and this is the partial derivative for the measurement function.

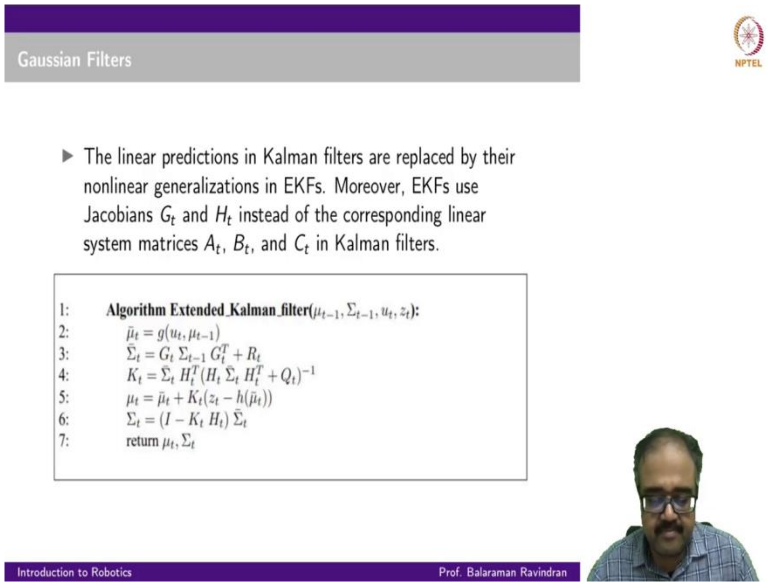
So I am approximating h of xt has h of mu t bar plus capital Ht times xt minus mu t bar. So that is my function. And now, remember, we are (use), you have to use mu t bar here because that is the most likely state at that time when I actually use my h.

So now, my measurement probability, which is probability of zt given xt, the probability of zt given xt can be approximated as a normal distribution with mean given by h of mu t bar plus capital Ht times xt minus mu t bar and Qt, which is a covariance of the noise term delta t. Remember, delta t is a Gaussian noise, additive Gaussian noise with mean 0 and covariance Qt.

So now, I will do the same thing here but the mean, now the new mean, instead of being Ct times xt as we had earlier, it is going to be h of mu t bar plus Ht times xt minus mu t bar and Qt would be the covariance as before.

So thus, second, see here is almost like the Kalman Filter, except that instead of using A, B, C, you are going to use the Ht and Gt which is essentially helping us linearize the function.

(Refer Slide Time: 21:01)



Gaussian Filters

- The linear predictions in Kalman filters are replaced by their nonlinear generalizations in EKFs. Moreover, EKFs use Jacobians G_t and H_t instead of the corresponding linear system matrices A_t, B_t, and C_t in Kalman filters.

```
1:    Algorithm Extended_Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2:        $\bar{\mu}_t = g(u_t, \mu_{t-1})$
3:        $\bar{\Sigma}_t = G_t\, \Sigma_{t-1}\, G_t^T + R_t$
4:        $K_t = \bar{\Sigma}_t\, H_t^T (H_t\, \bar{\Sigma}_t\, H_t^T + Q_t)^{-1}$
5:        $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6:        $\Sigma_t = (I - K_t\, H_t)\, \bar{\Sigma}_t$
7:        return $\mu_t, \Sigma_t$
```

Introduction to Robotics   Prof. Balaraman Ravindran

Now, once we have these linearized versions of the filter, of the motion model, and we basically rewrite the Extended Kalman Filter algorithm to look very much like the original Kalman Filter algorithm. Again, lines 2 and 3 are the update equations for go from bel xt minus 1 to bel bar xt. So as before, your bel xt is given by mu t minus 1, bel xt minus 1 is given by mu t minus 1 and Sigma t minus 1. I have the action ut I have the observations at t.

Now, I can use that to compute my mu bel bar of xt, which is again obtained by obtaining, computing mu t bar and Sigma t bar. And it looks, this expression looks very similar to what we had earlier, except that instead of A, B, and thing, so we are just using G now.

And likewise, we have the, so one thing note is that I am actually using the exact g here; g of ut comma mu t minus 1. So I am basically looking at, plugging in mu t minus 1 and computing what, mu t bar. So that is that is something to note.

And then again, I have the Kalman gain which is computed in the similar fashion. And I have lines 5 and 6 now compute bel of xt from bel bar. And likewise, very similar computation to what we had earlier and the Kalman Filter.

So this is essentially the Extended Kalman Filter algorithm. So it is more powerful than the Kalman Filter in the sense that it uses a nonlinear model for both the movement as well as the measurement but it retains the same convenience of Kalman Filter because I can assume that the belief is Gaussian, and it will stay Gaussian through my updates. So that is that is basically the Extended Kalman Filter. So we can stop here.