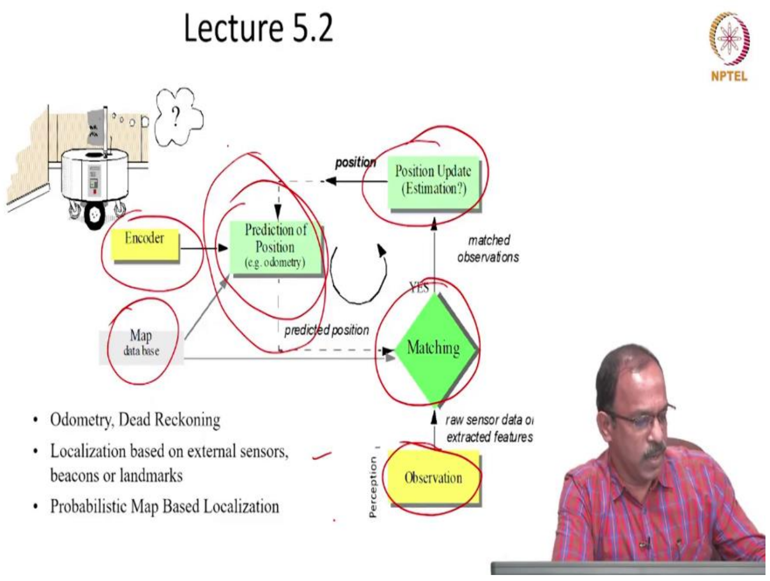
Wheeled Mobile Robots

Prof. Asokan Thondiyath
Department of Engineering design
Indian Institute of Technology, Madras

Lecture - 5.2
Map Based Localisation

(Refer Slide Time: 00:13)



Lecture 5.2

position
Position Update (Estimation?)
Encoder
Prediction of Position (e.g. odometry)
matched observations
YES
Map data base
predicted position
Matching
raw sensor data or extracted features
Perception
Observation

- Odometry, Dead Reckoning
- Localization based on external sensors, beacons or landmarks
- Probabilistic Map Based Localization

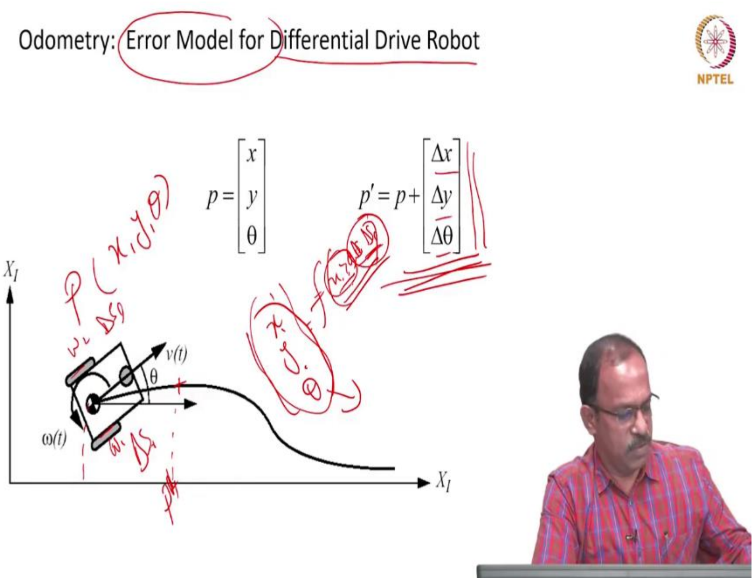
Hello everyone, welcome back. So, in the last class, we discussed about localization of mobile robots. We briefly talked about the importance of localization, the challenges involved in localization, how the sensors properties affect the localization, and the error propagation in the localization process. So, and also we mentioned about the five step process involved in the map based localization.

So, the first step in localization of mobile robot is basically the odometric, odometry based localization where we use the encoders on the robot. And then use the encoder data to find out how much the robot has moved over a period of time. And then use that information to find out the new position and orientation of the robot, so that is basically the odometry based localization where we call this the dead reckoning also where the position and orientation information is collected from the sensors.

So, that is the first step where you predict the predict the position using odometry. And then of course, we need to go to the sensors for collection of information from outside the robot that is camera or some other sensors will be used to get the information from outside the robot. And then check whether that is matching with the map information already available with the robot. So, there is a matching phase where the observation from the sensor and the map database will be mapped.

And then based on the error in these two information, the robot position will be recalculated, and you will be getting an updated position of the robot. So, this is the way how we go for the map based localization. So, the first step in map based localization is basically the odometry based prediction of the position of the robots. And I briefly mentioned about how we do this in the mobile robot.

(Refer Slide Time: 02:27)



Odometry: Error Model for Differential Drive Robot

$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \qquad p' = p + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$$

So, how we use the odometry based information to get the new position of the robot. So, I will just explain it once again. So, we have the current position of the robot as $p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$. So, we put this as the current position of the robot $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$. And then we assume that the robot is moving, and then it reaches the next position we call this as the $p' = p + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$, right.

The robot will be moving in x,y plane, and there will be an angular position angular position change also which will be given as $\Delta\theta$. So, this is the normal process especially for a when we consider it as a two wheeled robot with the differential drive. Now, we are estimating the robot position based on the information coming from the sensors.

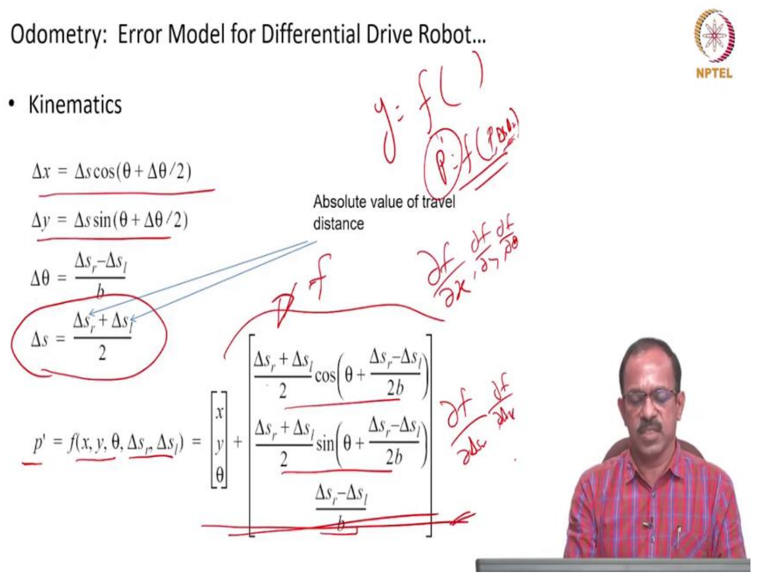
So, we have these two encoders which actually measures the distance travelled by the wheels. So, assuming that it is moving by ω_1 and ω_2, and we can say that it travels ΔS_r and ΔS_l that is the left and right wheels are travelling distances, ΔS_r and ΔS_l and period of Δt. Then we can say that that will actually lead to Δx, and Δy, $\Delta\theta$ in the Cartesian space.

So, how much will be sorry in the robot frame, you can say that it is $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ with respect to the frame of the robots. So, now, what happens is that, we are estimating the position x, y and θ in the new position we call this x', y', and θ', we are estimating this as a function of the previous position $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ because $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$. And then this previous position is $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$, and then we have a motion of ΔS_r and ΔS_l.

So, the new position $\dot{x}$ x' y' θ' is actually a function of the previous position $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ and the movement of the robot the wheel displacement ΔS_r and ΔS_l. Now, we know that the new position is related to or depending on this. And therefore, if there is any error in this x y the previous position and the error and any error in the ΔS_r and ΔS_l, that will actually lead to an error in the estimate of new position also.

So, this is what we need to estimate how much will be the error in the new position when there is an error in ΔS_r , ΔS_l as well as the previous position estimate also. So, we will see how can we actually generate an error model for this particular robot differential drive robot based on the information that we collect from the encoder. So, that is what we are going to do in this exercise.

(Refer Slide Time: 05:24)



Odometry: Error Model for Differential Drive Robot...

- Kinematics

$$\Delta x = \Delta s \cos(\theta + \Delta\theta/2)$$

$$\Delta y = \Delta s \sin(\theta + \Delta\theta/2)$$

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b}$$

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2}$$

Absolute value of travel distance

$$p' = f(x, y, \theta, \Delta s_r, \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2}\cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2}\sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$

$y = f(\ )$
$p' = f(p, \Delta s_r, \Delta s_l)$
∇f
$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial \theta}$
$\frac{\partial f}{\partial \Delta s_l}, \frac{\partial f}{\partial \Delta s_r}$
NPTEL

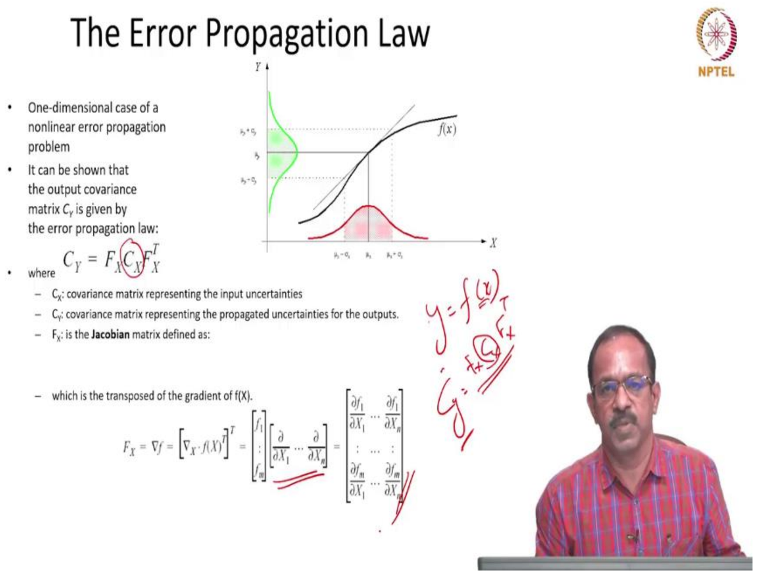
So, we can write this Δx based on the kinematics of the differential drive robots. We will be able to write that $\Delta x = \Delta s \cos(\theta + \Delta\theta/2)$; $\Delta y = \Delta s \sin(\theta + \Delta\theta/2)$; and $\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b}$, where b is the distance between the two wheels. And we will define $\Delta s = \frac{\Delta s_r + \Delta s_l}{2}$ over the average distance travelled by the wheel.

Now, we can see that the p' as I mentioned is a function of $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$, and ΔS_r and ΔS_l, ok. And this function can be written as this way. So, the function f is actually this one. So, this is the function f (x, y, θ) + $\frac{\Delta s_r + \Delta s_l}{2}\cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right)$.

So, that is basically the distance I mean we can write this $\Delta x = \Delta s \cos(\theta + \Delta\theta/2)$ that will be written like this. Similarly, Δy is this, and Δθ is this. So, this is the function f. So, we now write it as y is a function of this, or we can write it as p' is a function of p, and ΔS_r , ΔS_l.

Now, we need to know how much will be the error in p' if there is an error in p, and ΔS_r , ΔS_l, so that is basically the error propagation. There is an error in the previous position estimate and the estimate from the sensors. How much the wheel has travelled that also has got some error. If that is the case, how much will be the error in p dash need to be understood.

(Refer Slide Time: 07:13)



The Error Propagation Law

- One-dimensional case of a nonlinear error propagation problem
- It can be shown that the output covariance matrix C_Y is given by the error propagation law:
- where $C_Y = F_X C_X F_X^T$
  - C_X: covariance matrix representing the input uncertainties
  - C_Y: covariance matrix representing the propagated uncertainties for the outputs.
  - F_X: is the **Jacobian** matrix defined as:
  - which is the transposed of the gradient of f(X).

$$F_X = \nabla f = \left[\nabla_X \cdot f(X)^T\right]^T = \begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial X_1} & \cdots & \frac{\partial}{\partial X_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial X_1} & \cdots & \frac{\partial f_1}{\partial X_n} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_m}{\partial X_1} & \cdots & \frac{\partial f_m}{\partial X_n} \end{bmatrix}$$

$y = f(x)$

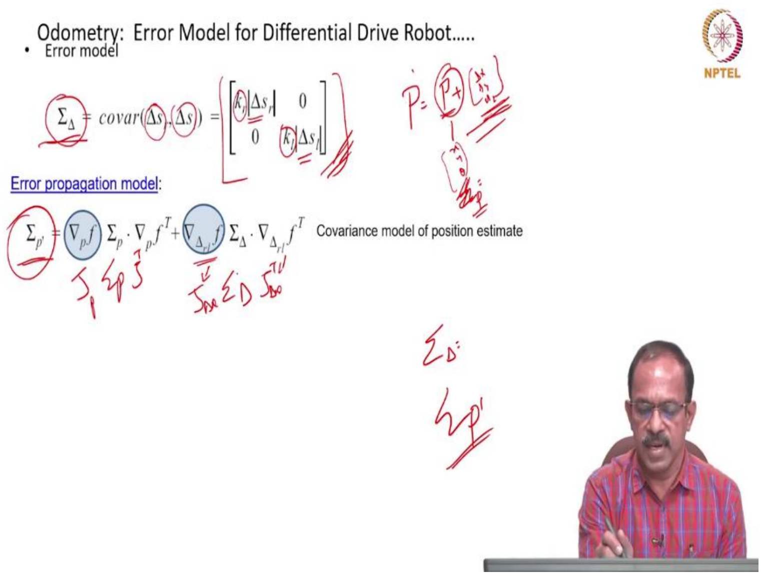
$C_y = F_x C_x F_x^T$

So, we can actually do this by looking at the error propagation law that we discussed in the previous class. So, the error propagation law says that if y = f (x), then we can find out the error in y for the $C_Y = F_X C_X F_X^T$, where C_x is the covariance in the measured the previous one f f(x).

And C_y is the covariance of the new value y is y = f (x). So, x has got an error C_x, then y will be having an error $C_Y = F_X C_X F_X^T$, where F_x is the Jacobian, which is defined by this. So, like the partial derivative of the function with respect to the variable x you will be getting the Jacobian.

So, this was this is the error propagation law. Now, we can use the same error propagation law and then find out what will be the error in the position of the new position estimate of the robots. When we know the previous position and the measurement from the encoders, ok.

(Refer Slide Time: 08:19)



Odometry: Error Model for Differential Drive Robot…..

- Error model

$$\Sigma_\Delta = covar(\Delta s_r, \Delta s_l) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix}$$

Error propagation model:

$$\Sigma_{p'} = \nabla_p f \, \Sigma_p \cdot \nabla_p f^T + \nabla_{\Delta_{rl}} f \, \Sigma_\Delta \cdot \nabla_{\Delta_{rl}} f^T$$ Covariance model of position estimate

$P = P + \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$
$J_p \Sigma_p J^T$
$J_{\Delta} \Sigma_D J_{\Delta}^T$
Σ_D:
$\Sigma_{p'}$

So, for that we assume that we know few things that is, so we have this P. So, P is given as $p' = p + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix}$. So, this is the; now, we assume that we know the covariance of the sensor. So, we know that the sensor has got some error. And we model that error. And we assume that that error is already known or the how much error that the sensor will be having is known is again not a what do you call a predicted one. It is more like a random error.

So, we model this as probability function, and then say that the covariance sensor covariance ΣΔ. So, we will put it as ΣΔ. We will write that it is known and it is actually depending on ΔS_r and ΔS_l. So, this can be written as a covariance matrix can be given as $\Sigma_\Delta = covar(\Delta s_r, \Delta s_l) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix}$.

So, we assume that this k_r and k_l are known for the sensor. So, how much will be the error that the sensor can give for right and left is given by this k_r and k_l a coefficient, and that multiplied by the actual travel distance will be the error in that sensor. So, this is what actually we assume that we know this. The sensor means since we have since we are choosing a a non sensor we assume that we have this information.

Now, we have this p' is p plus this one. So, the error will be basically coming from this one also. So, assume that p = $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ that was the initial value of θ. And we assume that the initial covariance of the position or the previous covariance is Σ_p. This can be if we know exactly from where the robot start, then this will be 0; otherwise, this will be some having some value, and as the robot moves the Σ_p will keep on change.

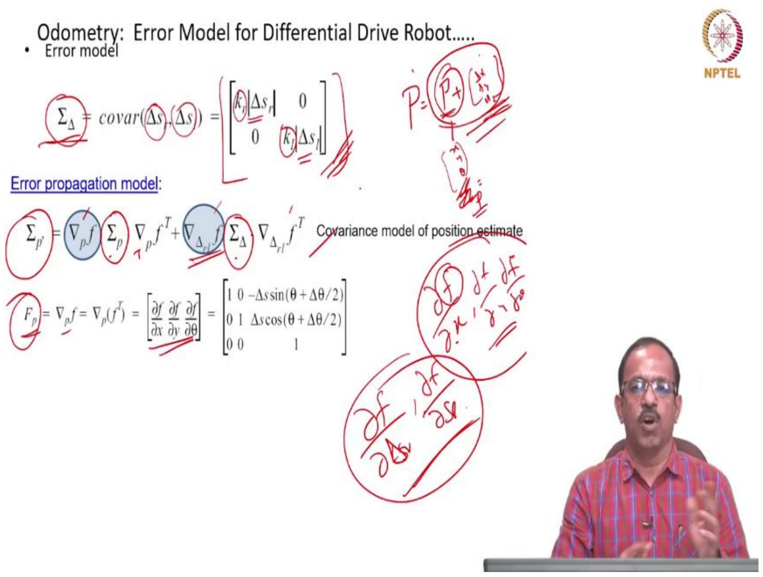
So, we assume that we have this Σ_p known that is the previous covariance previous uncertainty is known and the uncertainty the sensor is known. Then we can write this error propagation model as like this, that is the new covariance of p p' that is the covariance in the estimated position which we call it as $\Sigma_{p'} = \nabla_p f \Sigma_p . \nabla_f^T + \nabla_{\Delta_{rl}} f \Sigma_\Delta . \nabla_{\Delta_{rl}} f^T$. Now, this is basically the Jacobian.

So, we have this Jacobian j Σ_p j^T this is for the position and then we have Δ_{rl} Σ_Δ and $j\Delta_{rl}^T$. So, this is the covariance. So, this is the Jacobian. So, we can the error propagation model tells you that since there is a there are two things one is p and then the measured value.

So, the p also will be introducing some error, and then $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ also this measured are also introducing some error. So, the total error $\Sigma_{p'}$ can be written as j Σ_p j^T + j Δ_{rl} Σ_Δ and $j\Delta_{rl}^T$. So, this is the way how we can write the error propagation model for this.

Now, since we know this, so we assume that we know the previous estimated error in the estimated position, and we assume that we know the sensor error. Once we know these two, we will be able to calculate the Jacobians, and then find out what is the error in the estimated position p', so that is basically what we need to do.

(Refer Slide Time: 12:13)



Odometry: Error Model for Differential Drive Robot.....

- Error model

$$\Sigma_\Delta = covar(\Delta s_r, \Delta s_l) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix}$$

Error propagation model:

$$\Sigma_{p'} = \nabla_p f \, \Sigma_p \, \nabla_p f^T + \nabla_{\Delta_{rl}} f \, \Sigma_\Delta \, \nabla_{\Delta_{rl}} f^T$$ Covariance model of position estimate

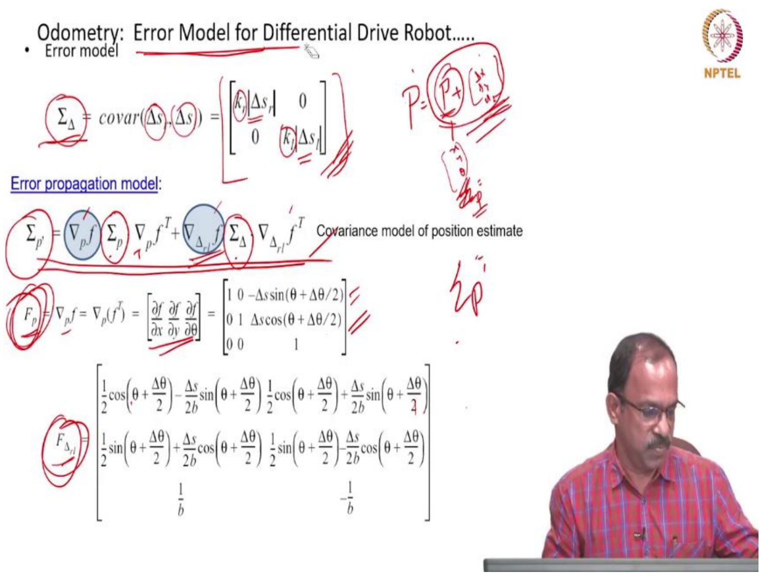
$$F_p = \nabla_p f = \nabla_p(f^T) = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta s \sin(\theta + \Delta\theta/2) \\ 0 & 1 & \Delta s \cos(\theta + \Delta\theta/2) \\ 0 & 0 & 1 \end{bmatrix}$$

So, F p which is basically the Jacobian is del f they can be obtained as partial derivative of f with respect to $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ ok, so because the position p = $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$. So, we will say that the f f by x y and θ will be the Jacobian for the position estimates. So, your F is known. So, the F is the function which actually you get $p' = p + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$. This is the function F.

So, we take the partial derivative of this F with respect to x, y, θ we will get F_p. Similarly, if you take the partial derivative of F with respect to Δr and sorry Δs_r and Δs_l, we will be getting the Jacobian corresponding to this also. So, this is the way we get these two Jacobians.

And then based using these two Jacobians, we will be calculating the covariance model of the position estimate or how much will be the error in the estimated position, so that is what we do in this. So, this Δ_{rl}, so this will be the Jacobian.

(Refer Slide Time: 13:21)



Odometry: Error Model for Differential Drive Robot…..

- Error model

$$\Sigma_\Delta = covar(\Delta s_r, \Delta s_l) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix}$$

Error propagation model:

$$\Sigma_{p'} = \nabla_p f \, \Sigma_p \, \nabla_p f^T + \nabla_{\Delta_{rl}} f \, \Sigma_\Delta \, \nabla_{\Delta_{rl}} f^T$$ Covariance model of position estimate

$$F_p = \nabla_p f = \nabla_p (f^T) = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta s \sin(\theta + \Delta\theta/2) \\ 0 & 1 & \Delta s \cos(\theta + \Delta\theta/2) \\ 0 & 0 & 1 \end{bmatrix}$$

$$F_{\Delta_{rl}} = \begin{bmatrix} \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta s}{2b}\sin\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{\Delta s}{2b}\sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{2}\sin\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{\Delta s}{2b}\cos\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\sin\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta s}{2b}\cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix}$$

So, if you take this F function that is what is shown in the previous slide. So, this is the f function. So, you take the partial derivative of this with respect to x, because this is $x + \frac{\Delta S_r \Delta S_l}{2cos\theta} + \Delta S_r - \frac{\Delta}{2b}$. So, partial derivative of F with respect to x will be this will be 1 and then you will not be getting anything for the y, then x will be obtained then the F by y you can obtained and θ also you can obtain.

Similarly, partial derivative of this with respect to S_r and S_l, you can take the partial derivative of this with respect to S_r and S_l, you will be getting the Jacobian. So, that is what actually you are getting yeah. So, this is what you will be getting as the Jacobian. And as you know that this will be function of the θ, current θ and the Δ θ how much the robot has travelled I mean change its orientation.

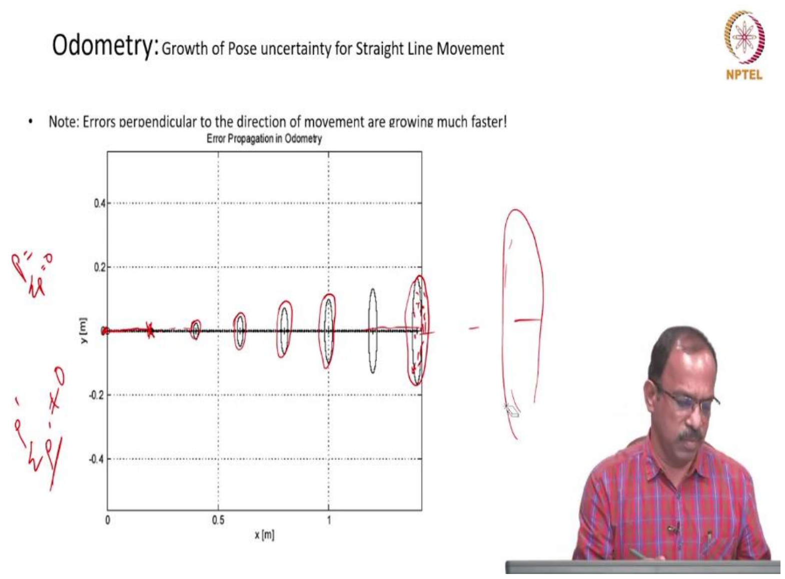
So, this Jacobian will depend on these values k_θ for the partial this Jacobian F_p will be functioning of function of θ as θ changes this also will change. Similarly, F Δ_{rl} Jacobian will be function of θ and Δ θ. So, current position and that changes in position also changes in the orientation will also be function.

So, we now get this F Δ_{rl}, and F_p, and then what we do is we will just try to find out how what is a new estimated position and it is covariance. So, position the mu value will be simply x + Δx and y + Δ y and θ + Δ θ that will be the position p. And its covariance Σp will be this one. So, you will be able to get the Σp.

So, at every step as the robot moves so, every step we calculate what is the new position and the new covariance of that position, and then use that information to calculate the next position and the next covariance. So, this is the way how we move forward and keep on estimating the robot position as the robot moves.

So, this is the odometry based estimation where we use the error model to find out the estimated covariance of the new position. So, this, these steps need to be continuously carried out for the robot in order to estimate the new position of the robot position and its covariance.

(Refer Slide Time: 16:05)



Odometry: Growth of Pose uncertainty for Straight Line Movement

- Note: Errors perpendicular to the direction of movement are growing much faster!

So, for example, suppose the robot is starting from here, ok. So, it is an x y plane the robot is in the x y plane. So, we do not consider the θ motion for the time being. We assume that the robot starts from here and the position is completely known. So, the p is given, and it Σp is given as 0, I mean there is no uncertainty in the beginning.

Now, at the robot starts moving, it will calculate at this position we will calculate what is the new position p’ and this covariance p’. So, as the robot has moved, then there was an uncertainty in the sensor measurement. There will be small uncertainty. So, this Σp’ would not be equal to 0 now because of the sensor error.

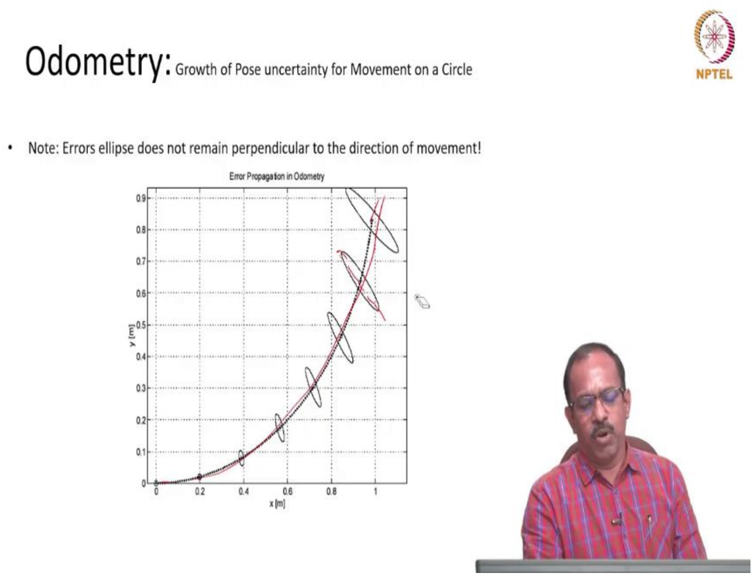
And we can actually represent that like a I mean this is with a small ellipse over here. You know as the robot moves this uncertainty the covariance can actually be represented using this ellipse. So, you can see that the uncertainty will keep increasing – the ellipse size increases because the covariance increases the uncertainties increasing. And after some time, you will see that the it is uncertainty is very high.

So, the robot will not be knowing exactly where it is. It says that it can be anywhere in this ellipse. The position of the robot could be anywhere within this ellipse, but it has actually moved in a straight line, it could be anywhere here. So, that much of uncertainty will be there. And as the robot moves it is only for a short duration you will see that it.

And as it moves after some time it you will see that the uncertainty will be too high. And within the map itself, you will see that the robot is completely uncertain about its position. And therefore, it there is no, no more a meaningful localization can be done if you are using only the odometry. For short duration, yes, we can actually get some kind of an estimate. But for a long duration, you will see that it actually goes completely uncertain and the localization fails.

So, this is the problem with odometry. So, this shows that how we can use the odometric information to estimate the position using the uncertainty measurements, and using the error propagation model. So that is one simple example to show that how odometry can be used for localising a robot, and what problems you face if you are simply doing it only using odometry. So, this was on a straight line.

(Refer Slide Time: 18:31)



Odometry: Growth of Pose uncertainty for Movement on a Circle

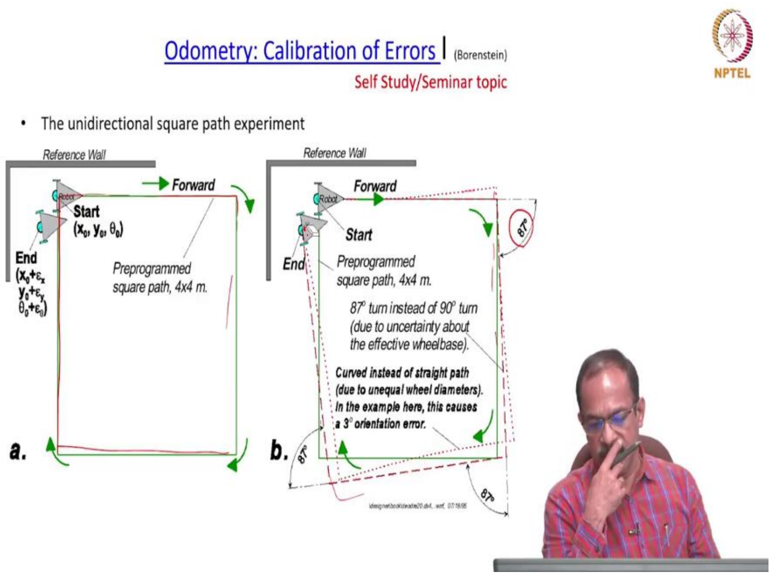
- Note: Errors ellipse does not remain perpendicular to the direction of movement!

But if you go it in a curved line, so obvious the robot is actually commanded like this to move. You will see that the uncertainty increases, but always the in the direction of motion the uncertainty will be less, because you are actually getting a lot of measurements in the direction of motion.

But perpendicular to that the uncertainty will be increasing because you do not have anything to measure in that direction. So, your measurement is always along the direction of motion. So, the along the direction of motion, your uncertainty will be less, but perpendicular to the direction of motion you will have large uncertainty. So, this is the common in odometry based localization which cannot be solved easily. The only way to do it is to go for better sensors with very less uncertainty or error errors.

So, if the sensor covariance is very small, then the error also will be small estimated position error also will be small. So, this is why how we do the odometry based localization. And this is quite old. We would have been using this for a long time. And there were lot of experiments carried out on based on odometry.

(Refer Slide Time: 19:35)


Odometry: Calibration of Errors (Borenstein)

Self Study/Seminar topic

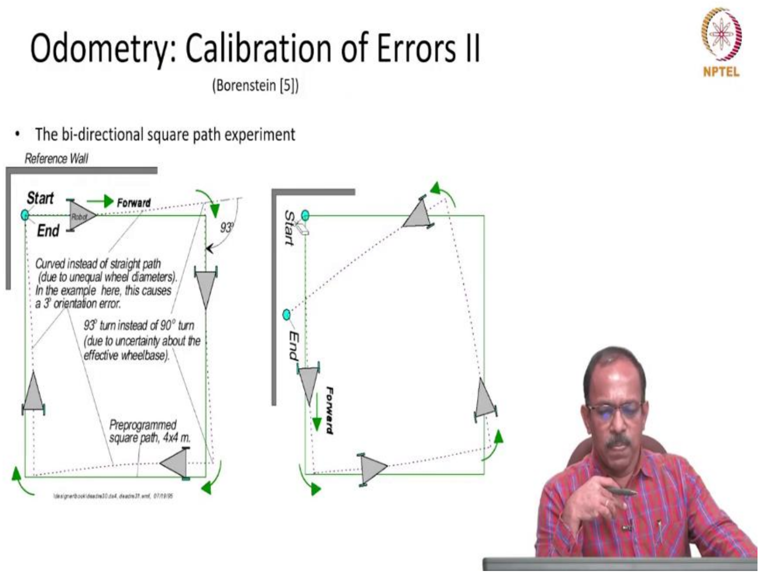
- The unidirectional square path experiment

NPTEL
Reference Wall
Forward
Robot
Start
(x_0, y_0, θ_0)
End
$(x_0+\varepsilon_x$
$y_0+\varepsilon_y$
$\theta_0+\varepsilon_\theta)$
Preprogrammed square path, 4x4 m.
a.
Reference Wall
Forward
Robot
Start
End
Preprogrammed square path, 4x4 m.
87° turn instead of 90° turn (due to uncertainty about the effective wheelbase).
Curved instead of straight path (due to unequal wheel diameters). In the example here, this causes a 3° orientation error.
87°
87°
87°
b.

So, if you simply use the robot to simply command the robot to move along this paths in the commander position path is like this. You will see that the actual path the robot has move maybe something like this. So, it will never be reaching the same position if you do this. This is because of the errors in many things.

One is the errors in the travel, then the errors in the rotation, it is you want 90° to be rotated, but it may not rotate 90°. So, like this you will be having lot of errors and the robot may be reaching here. So, this you will see that this is the actual path the robot will be travelling.

(Refer Slide Time: 20:22)



Odometry: Calibration of Errors II

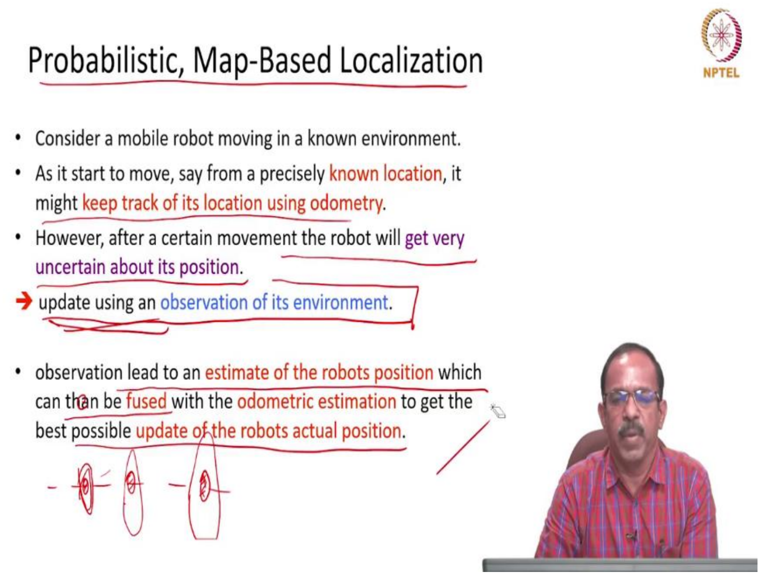
(Borenstein [5])

- The bi-directional square path experiment

NPTEL
Reference Wall
Start
Forward
Robot
End
93°
Curved instead of straight path (due to unequal wheel diameters). In the example here, this causes a 3° orientation error.
93° turn instead of 90° turn (due to uncertainty about the effective wheelbase).
Preprogrammed square path, 4x4 m.
Start
End
Forward

And of course, if you do this you many times you will see that the robot has completely lost its position, and it will never be following a a defined path or a predicted paths, ok.

(Refer Slide Time: 20:36)



Probabilistic, Map-Based Localization

- Consider a mobile robot moving in a known environment.
- As it start to move, say from a precisely known location, it might keep track of its location using odometry.
- However, after a certain movement the robot will get very uncertain about its position.

➔ update using an observation of its environment.

- observation lead to an estimate of the robots position which can than be fused with the odometric estimation to get the best possible update of the robots actual position.

So, that is the pure odometry based prediction of the robot position. And as I mentioned it is not at all feasible to use it for a long duration and that is where we go for the probabilistic and map based localization. So, in the probabilistic, map based localization, what we do we, still use the odometry, but then we tried to correct the odometric error by some other method, so that is basically the probabilistic map based localization, ok.

So, this one I already mentioned that there will be lot of errors in the odometry. So, it may keep from non-location it may be able to move based on the odometry, but after a certain moment the robot will get very uncertain about its position. So, in the probabilistic map based localization what we do is, we update the position using an observation of its environment. So, this is what actually we do.

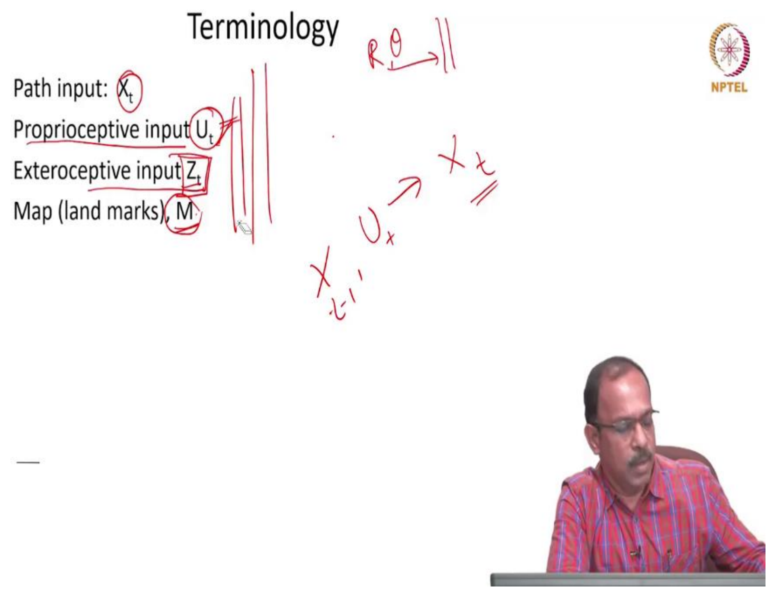
We try to update the position based on observation of its environment. As I mentioned earlier, so we use a camera or some other sensor to observe the surroundings of the robot. And then use that information along with the map information and using that information we try to update the robot position. So, that is basically the map based localization.

So, this observation lead to an estimate of the robot position which can then be, which can then be fused with the odometric estimation to get the best possible update of the robots actual position. So, what we do? We will fuse these two information- one is the: information collected from the odometry, and the other one is the information collected from the sensors, and then we will fuse these two to get a better estimate of the robot.

So, we use some kind of a fusion algorithm to get the better estimate of the robot position, so that is basically the map based localization. So, we do not depend only on odometry. We use some other method also, and then fuse these two, and then reduce the error in the estimated position. So, at every stage, you try to reduce the error. So, when the error is actually going increasing like this as I mentioned earlier.

So, at every stage, we will try to at this stage if this is the estimated position using odometry. We will try to use the information from the sensors and then reduce it to a smaller value. And again this will actually go up, and then again we will try to bring it down, and we will try to bring it down. So, the error estimated error will be much less compared to the odometry alone localization. So, that is basically the map based localization.

(Refer Slide Time: 23:29)



Terminology

Path input: X_t

Proprioceptive input U_t

Exteroceptive input Z_t

Map (land marks), M

So, before we talk about map based localization, let me introduce some of the terminologies we normally commonly you encounter in this map based localization ok. So, we assume that there is a path commanded to the robot which is given as X_t. So, X_t is the path inputs. So, at every time instant t, you have a position $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$, commanded. So, X is the with the vector which represents the pose of the robot at t.

Now, we assume that there is a proprioceptive input U_t. So, there is the control input or the input that we actually given to the robot, which is measured by the sensors. So, the proprioceptive that what is sensored sensed by the sensors within the robot. So, for example, the encoder will measure the speed of the wheel that is basically the proprioceptive input U_t.

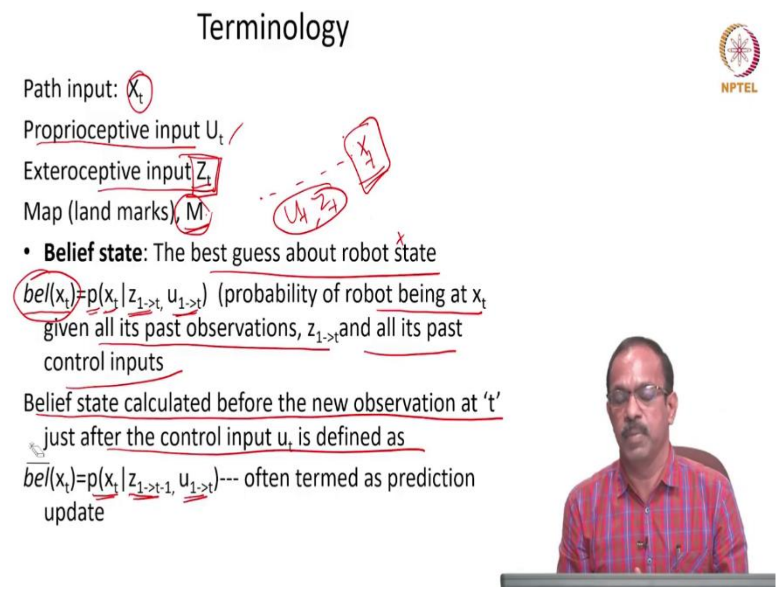
So, when the robot is in that position X_t, suppose it is at initial at X_{t-1}, and then you give a control input U, then it moved to X_t. So, that is the control input that you give U_t for the to move to X_t. So, U_t is the proprioceptive input which is basically the control input given to the robot and that input is measured using the sensors what is the how much the wheel has rotated or how much the wheel has travelled is measured that is basically the U_t.

Then we assume that there is a exteroceptive input that you are getting from the sensors which are attached to the robot. Exteroceptive sensors are the sensors, which collect information from outside the robot. So, we use a exteroceptive sensor, and then collect the information which we call it as Z_t. So, Z_t will be information about some object in the vicinity of the robot.

The robot will say oh I can see a wall at a distance R and at an angle θ from my position. So, that may be an input which is given coming from the sensor, so that is basically a exteroceptive inputs Z_t. And then we assume that there is a map available to the robot, or the map information is already available.

So, we need to have the position of the robot, and what is the control input given to the robot for it to move forward. And what is the sensor that is seeing that is exteroceptive input Z_t – what is what the robot is seeing, what the sensor is seeing in the surroundings, and then the actual map of the environment. So, these are the requirements for map based localization.

(Refer Slide Time: 26:13)



Terminology

Path input: X_t

Proprioceptive input U_t

Exteroceptive input Z_t

Map (land marks), M

- **Belief state**: The best guess about robot state

bel$(x_t)=p(x_t \mid z_{1\to t}, u_{1\to t})$ (probability of robot being at x_t given all its past observations, $z_{1\to t}$ and all its past control inputs

Belief state calculated before the new observation at 't' just after the control input u_t is defined as

$\overline{bel}(x_t)=p(x_t \mid z_{1\to t-1}, u_{1\to t})$--- often termed as prediction update

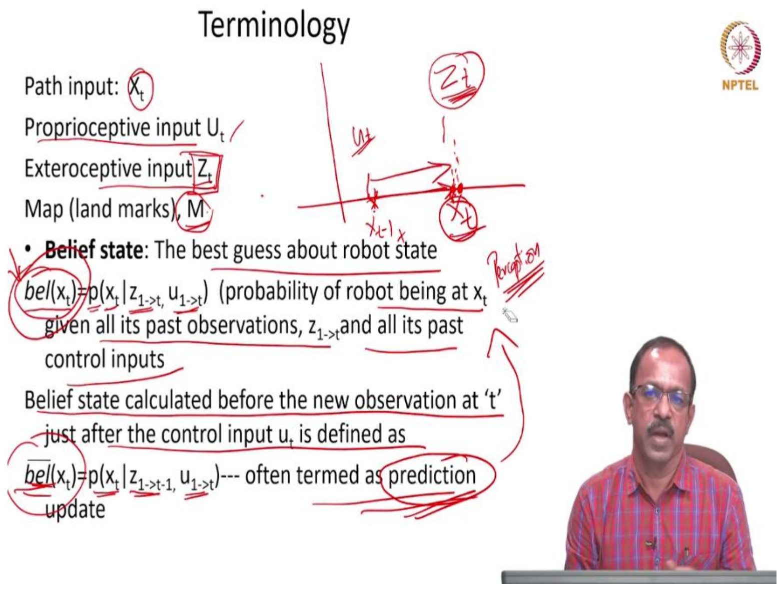
Now, we define something called a belief state. So, a belief state is defined as the best guess about robot state. So, the robot is believing that it is in this particular position we call that as the belief state of the robot. Now, this belief state can actually be having two types of belief states ok.

So, the belief state can be defined as the probability that the robot is at x t probability that the robot is at X_t given the Z_t that is the exteroceptive input, and the control input up to t that is known as the belief state of the robot belief X_t. So, the probability of robot being at X_t given all its past observations and all its past control inputs, so that is basically the belief state that is the robot was robot say that it is this t, it has moved from somewhere else.

So, all its control input is given, and all its map information is provided. And based on all this information, the robot says that my position is this X_t, so that is basically the belief state of the robot. Now, we can have a belief state calculated before the new observation at t just after the control input U_t, which is defined as $\overline{bel}(X_t)$.

So, we call this $\overline{bel}(X_t)$, which is probability of the robot is that it is the X_t provided the observations before reaching this position, ok. Before reaching this position, all the observations are there, and the control input till that t position t instant is provided. So, that is basically known as a $bel(X_t)$ $\overline{bel}(X_t)$. So, to explain this, so we will take a again a one-dimensional case.

(Refer Slide Time: 28:24)



Terminology

Path input: X_t

Proprioceptive input U_t

Exteroceptive input Z_t

Map (land marks), M

- **Belief state**: The best guess about robot state

bel(x_t)=p(x_t | $z_{1\text{->}t,}$ $u_{1\text{->}t}$) (probability of robot being at x_t given all its past observations, $z_{1\text{->}t}$ and all its past control inputs

Belief state calculated before the new observation at 't' just after the control input u_t is defined as

$\overline{bel}$(x_t)=p(x_t | $z_{1\text{->}t-1,}$ $u_{1\text{->}t}$)--- often termed as prediction update

So, assume that the robot was initially here X_{t-1}. Now, you give a control input u t here, so the robot moves to X_t. Then it uses the information from the sensor and then collect the information Z_t, and recalculate its position and then says that ok now this is the Z X_t that is the new position X_t. So, the robot believes that it is at this X_t when all the U_t is known and Z_t is known, then we call this as the $bel(X_t)$.

$\overline{bel}(X_t)$ is it has it was in X_{t-1} and then it has got a control input U_t, and then reached at this position X_t and did not did not get the information from the sensor, and that belief state is known as the $\overline{bel}(X_t)$. So, $\overline{bel}(X_t)$ is the belief state just before getting the update from the sensor exteroceptive sensor.

So, it has got the control input, it has moved to its new location, but it has not corrected its position based on the sensor input, so that is basically known as the $\overline{bel}(X_t)$. So, $\overline{bel}(X_t)$ is basically a prediction update based on the odometry; and $bel(X_t)$ is basically a estimated position after getting the input from the sensor.

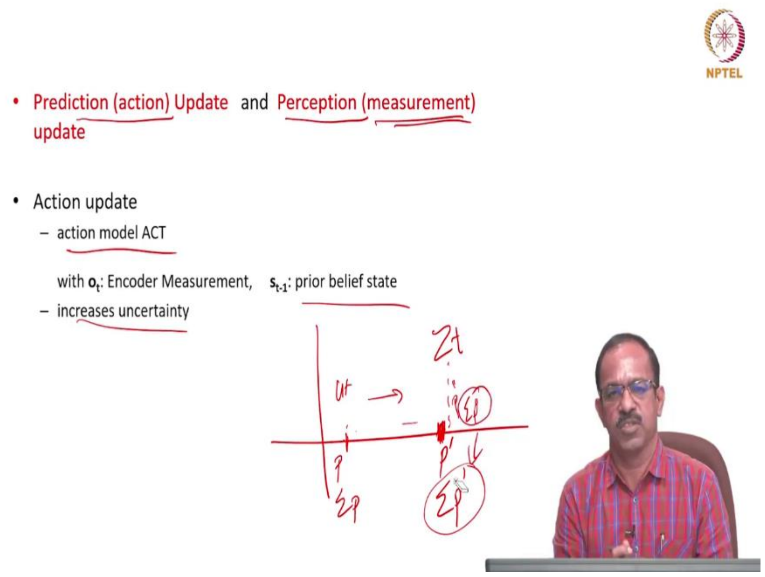
So, we call this as the prediction update because we are predicting the new position based on the odometry is belief bar. And then this is known as the perception update. So, you are actually updating the new position based on the perception that is the data that you are collected from the environment, so that is basically the perception updates.

So, you have two stages one is known as prediction update. Prediction update is purely based on the control input. And perception update is you have a updates of the prediction update based on the observations surrounding the robot, and that update is known as prediction update sorry perception update.

So, the final belief state that we are interested is this one $bel(X_t)$, and the intermediate stage is the $\overline{bel}(X_t)$, which is a prediction update. So, we will be having first state prediction update. So, the robot starts from a non-position. It update its position based on the prediction, and then it updates its position once again based on the observation, and finally, you will get the new position estimate of the robot.

So, it is not only the position the it is covariance also will be estimated as the robot moves. So, we will be having a prediction update and perception update at every stage for the robot to move forward and localise itself. So, this is what we do in the map based localization, ok.

(Refer Slide Time: 31:17)



- Prediction (action) Update and Perception (measurement) update
- Action update
  - action model ACT

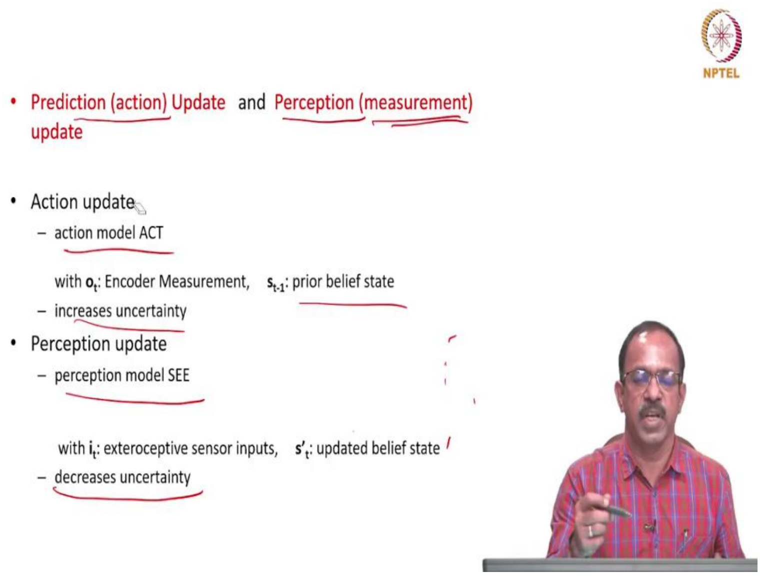
    with $\mathbf{o}_t$: Encoder Measurement, s_{t-1}: prior belief state
  - increases uncertainty

So, you have a prediction update which we call it as an action update because you are actually commanding the robot to move, there is the prediction update. And then there is a perception update and this is based on the measurement that you make from the robots using the robot sensor, so that is basically the measurement update or the perception update, ok.

So, you have an action update and a perception update. So, prior belief state and then it increases the uncertainty. Because from the previous position, so if you have the position like this, so this is the position p we say there is a uncertainties Σp also. Now, as the robot moves with the control input U_t, it reaches the new position P', and the uncertainty actually increased to Σp' because now because of the control input and the errors in the sensor you will be having a large uncertainty.

So, this uncertainty – large uncertainty will be reduced using the observation Z_t, and it will calculate a new position we call this as new position estimated as using the sensor as P'', and then we get a new covariance estimate also. And this new covariance will be smaller than this covariance. So, that is the (Refer Time: 32:37) here.

(Refer Slide Time: 32:37)



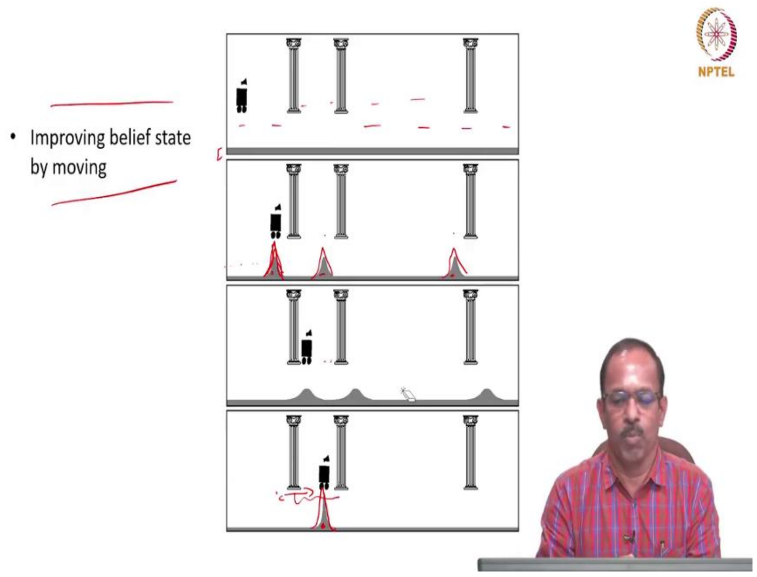
- Prediction (action) Update and Perception (measurement) update
- Action update
  - action model ACT

    with o_t: Encoder Measurement, s_{t-1}: prior belief state
  - increases uncertainty
- Perception update
  - perception model SEE

    with i_t: exteroceptive sensor inputs, s'_t: updated belief state
  - decreases uncertainty

So, the perception update is basically the SEE that is that you are actually seeing using a sensor and then updating the information. And that actually decreases the uncertainty because of this updated belief state. So, you perception updates will reduce the uncertainty, while action update will increase the uncertainty.

(Refer Slide Time: 32:58)



- Improving belief state by moving

So, this can be explained with a simple diagram. Consider this is a robot moving in a one-dimensional space, I mean it is just moving in a straight line. And assume that the robot does not know where it is. It is initial position and covariance is unknown. So, it assume that it could be anywhere in this, this location, anywhere, it could be anywhere

here. And it equal probability, so this actually this shaded region shows that it equal probability for the robot to be anywhere in this region.

Now, the robot starts moving. And assume that the robot has got a map, and the map shows that there are three poles like this and their distance between these are also known. Now, the robot starts moving. As the robot starts moving its uncertainty will keep on increasing. So, the uncertainty will keep increasing till here.

And it sees one pole here ok, so that means, it has seen one of these, but it could be the robot could be here, or it could be here also. So, robot does not know where it is in the map. So, it says that I could be here or here or here. So, I uncertainty my probability that the robot, robot is here, here, and here are much higher than any other place. So, the robot has got some information saying that it could be anywhere in these three places, so that is the uncertainty.

So, basically it has use the initially the odometry to update its position, and then using the sensor it has actually updated its position, and it has got a much more certainty that it could be here, here, or here, than any other place. So, that is the way how we reduce the estimate error. Now, the robot still starts moving because it could not really localise. It starts moving.

And as it starts moving, again this probability decreases because now the robots uncertainty again increases. So, this probability decreases. And then it could be all the three probability decreases the robot does not know where it is. Now, as it moves it actually sees the next pole after some time after travelling a particular distance, it sees that another pole is there.

Now, the robot is almost sure that it could be at this location only because from this location it has travelled this much distance and there, and it seeing the next one that means, it could be only at this position. It could not be any other location because no other location will be able to give you this information.

So, it is use the sensor information as well as the odometry information to estimate its new position, and then it has actually got a very high probability that the robot is here. So, the robot localise itself and saying that I could be in this position. And there is a small uncertainty, but otherwise it is more or less sure about its position and so, it localises itself.

So, so, it actually uses the prediction update and the perception update to update its current position and its uncertainty. And once it is able to use the map information and the sensor of information, and combine them together, and then identify its location then the robot localises itself. And then says that I could be in this position with maybe 99 percent probability. So, that is the way how the robots do this map based localization.

Of course, there should be some algorithms running to do this. So, we look into those aspects in the next class. How we can actually use some standard algorithms in order to do this matching of the features, and then use that information to get the covariant or the uncertainty estimate of the robots position. So, we will discuss that in the next class.

Thank you.