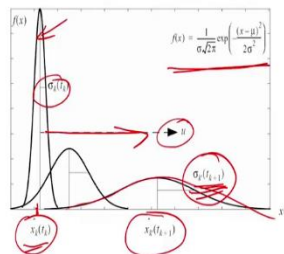


Wheeled Mobile Robots
Prof. Asokan Thondiyath
Department of Engineering design
Indian Institute of Technology, Madras

SLAM
Lecture - 5.5
Kalman Filter Localisation for Mobile Robots

(Refer Slide Time: 00:13)

Lecture 5. 5
Kalman Filter Localisation for Mobile Robots



Hello everyone, welcome back to the discussion on localisation of Mobile Robots. So, in the last lecture we briefly talked about Kalman filter based localisation. So, I explained about the logic of having a Kalman filter and then how this Kalman filter can be used for localizing the mobile robots. So, as you can see here.

So, if you have the basic principle is that if you have two estimates you can actually combine these two estimates or you can fuse these two estimates using a Kalman filter and get a better estimate. And the estimated the new estimate will be having reduced covariance; that means, the error will be much less than the two individual estimates that you had earlier.

So, that is the basic principle of Kalman filter. And we saw that there are some basic assumptions that the previous position and its covariance is not. For example, you know that the robot is at t_k the robot is at x_k and its deviation is standard deviation σ is also known and it is actually a white noise distribution. And it can be represented using this

density function $f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$. So, that is the way how we can represent this distribution.

And there is a fundamental assumption that the error can be explained expressed using a distribution of this form. Now, we assume that the robot is actually moving from here with the control input u and reaches the next position. So, you will be having a prediction update because of u that is a controlled input and the predicted position can be given as $x_k(t_{k+1})$ and then it will be having a deviation $\sigma_k(t_{k+1})$.

And this can actually be calculated using the error propagation law we discussed earlier because we know $x_k(x_k, t_k)$ and then σ_k and we know the control input and the covariance in the control input or the sensor inputs. And using these two we will be able to get the new that is the first prediction update. And then some of the prediction update you will be able to.

So, as you can see on the and the during the prediction update the uncertainty increases and this uncertainty need to be controlled using a perception update. So, in the perception update we use the sensors to get the information and then we map the match the sensor data as well as the map data and use that one to calculate the Kalman gain and then use the Kalman gain to get the new position estimate as such as its coverage.

(Refer Slide Time: 03:12)

Kalman Filter Localisation for Mobile Robots

- Prediction update and Measurement update
- Prediction update: The robots position at time step t is predicted based on its old location (time step $t-1$) and its movement due to the control input $u(t)$ (odometric estimation)
- Measurement update
 - Observation : Sensor measurements Z
 - Measurement prediction \hat{Z}
 - Predicted observations as what the robot expects to see at present location "xt"
 - Prediction expressed in sensor frame
 - Matching
 - Produce an assignment from the observation to the prediction
 - Innovation: measure of difference between predicted and observed feature
 - Validation gate
 - Estimation
 - $x_t = x_{t-1} + K_t V_t$, $K_t = P_t H_t^T (\Sigma_{z_t})^{-1}$ Co - variance , $P_t = P_{t-1} - K_t (\Sigma_{z_t}) K_t^T$

$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

So, that is basically the Kalman filter localisation. So, the steps involved in the Kalman filter localisation as I explained in the last class I will just go through once again. So, go for the prediction update ok. So, there are two update prediction and measurement updates. So, we will first go for the prediction updates.

So, that is the robots position at time step t is predicted based on its location old location t minus 1 and its movement due to the control input u_t . So, that is basically the prediction updates and we saw that it can be done using the control input the if you know the kinematics of the robot and using the error propagation law we will be able to get the new position the prediction update position which is \hat{x}_t and p_t that is the covariance that can be obtained.

And then the next step is the measurement updates. So, what we will do? We will update the this position that is this position that it has already obtained using the control inputs we will try to update this using the measurement. So, measurement is basically the perception update you measure the surroundings get some information and then try to update it that is basically measurement updates ok.

So, we have two things in this one is the observation using the sensors. So, we call this as z_t that is the sensor measurement z_t and then we have the measurement prediction \hat{z}_t . So, we call this as the \hat{z}_t . So, this is \hat{z}_t . So, the prediction update is measurement prediction is basically what the robot is supposed to see from this position if it is actually exactly at that position what the robot is supposed to see based on the map information.

So, map has got some features and if the robot is at that location then as per the map it is supposed to see something and that is this \hat{z}_t and this is the z_t which is basically what the sensor is actually seeing. So, there are two things one is what actually the robot is seeing and what the robot is supposed to see that is the z_t and \hat{z}_t and as I mentioned the predictor observations are what the robot expect to see and it has to be expressed in sensor frame.

Because the map will be having its own frame, but all these features need to be transferred to the sensor frame; so, we use a transformation h_j to transfer all those images

all those objects what the robot is seeing as per the map and then represent it as $z_{t_j} h$.

There is a prediction expression sensor frame.

And now we will try to find out what is the difference between these two what is the supposed to see and what it is actually seeing and that is what we call it as the innovation $v_{t_{ij}}$ ok. So, that is the matching stage.

So, produce an assignment from the observation to the prediction measure of difference between predicted and observed feature. So, you have a predicted feature and an observed feature. So, what is the difference between these two can be obtained using here $z_{t_i} - \hat{z}_{t_j}$ because, there may be m objects the robot is seeing there may be n object what it is supposed to see and now we need to see match them each one of these and then find out the different which we called it as the innovation.

So, if there is a map like this and the robot is actually I mean it is seeing an object here. Now this is what actually the robot is seeing I call this as Z_1 and then the robot is actually supposed to see something here. So, I call this as \hat{Z}_1 . So, this is 1, similarly there will be 2 ok, then there may be 3 like that 1 2 3 and some of them will be visible some of them would not be visible, but the robot will not be knowing which one is what.

So, it has to check with the 1 which is matching with the $\hat{1}$ or $\hat{2}$ or three hat it has to do the matching. So, it will do one by one each one of these will be checked and then it will try to find out which one is really matching and based on this matching it will use a validation gate to check whether it is the same object or not because, there is no point in comparing this object what they are supposed to see and then these objects.

They are not the same. So, if the difference is this one in a within a validation gate they define a small value and if it is within that then it just it will declare that it is the same object what the robot is seeing and what the robot is supposed to see. Only thing there is a small difference between the position what that is actually there and it is supposed to be there.

So, that way it will try to find a use a validation gate and then once it is validated then it will try to how much is the difference and that using that one it will try to find out an innovation covariance which is given as $H_j \hat{P}_t H_j^T$ and R_{t_i} where R is the sensor

uncertainty and P_t is the covariance at the current position and H is the Jacobian of this transformation j.

So, that you will get a σ innovation or we called it as the innovation covariance and once that is there then we will get the new position x_t as \hat{x}_t \hat{x}_t is the current position and that plus $K_t V_t$ where K_t is the Kalman gain and V_t is the innovation that is what you are getting the total innovation that is calculated from all the objects will be getting it as innovation and then you have this K_t which is the Kalman gain which can be written as

$$\hat{P}_t H_t \sigma_{innovation}^{-1}$$

So, that is basically the Kalman gain and then the covariance of the new position at this the estimated new position will be somewhere here and that will be something like this that is obtained like this. $\hat{P}_t - K_t \sigma_{innov} K_t^T$. So, that is the way how we get the new position estimates position as well as its covariance. So, this is basically the Kalman filter based localisation of mobile robots. So, that is the basic principle ok.

(Refer Slide Time: 09:46)



$\hat{x}_t = \hat{x}_t + K_t V_t$

The best estimate \hat{x}_t of the robot state at t is equal to the best prediction of the state at t (\hat{x}_t) before the new measurement, Z_t , plus a correction term of an optimal weighing value K_t times the difference between Z_t and best prediction \hat{Z}_t at time t.

$K_t = \hat{P}_t H_t^T (\Sigma_{IN_t})^{-1}$

Co - variance $P_t = \hat{P}_t - K_t (\Sigma_{IN_t}) K_t^T$

So, once we have this then this x_t is $K_t V_t$ and K_t can be obtained as this is the Kalman gain. So, the best estimate of x_t of the robots at states t is equal to the best prediction of states at t that is \hat{x}_t before the new measurement Z_t plus a correction term of an optimal weighing value K_t that is the Kalman gain times the difference between Z_t and \hat{Z}_t .

So, that is basically the Z_t and \hat{Z}_t are the predicted and the actual observations the difference between that minus the difference between that multiplied by the Kalman gain plus the previous estimate will be the new position estimate of the robots and covariance is this one P_t .

Because then again this P_t will be the covariance at that estimated position new position and you can see it will be smaller than \hat{P}_t because these are all positive. So, it will be always smaller than the previous covariance that is the predicted covariance or the movement covariance due to the motion of the robots it will be reduced to a new smaller value when we have a perception update.

(Refer Slide Time: 11:11)

Kalman Filter for Mobile Robot Localization

Robot Position Prediction: *Example*

NPTEL

Odometry

So, that is how we get the new estimated position ok. So, I will take an example and try to explain this to you of course, we will not go through all the numerical stages, but I will explain the logic of how we do this. So, assume that we have a arena like this what is shown here. So, this is an arena where a robot is moving ok. So, there is a arena assume that the robot is at $t - 1$ it is here.

So, that is the position of the robots. So, that the blue circle what you are seeing is the robots and you can see the two wheels also here and its got an axis here the forward axis. So, at $t - 1$ the robot position is call it as x_{t-1} and we say that it has got a covariance of P_{t-1} which is represented using this ellipse.

So, the ellipse basically tells that this is the uncertainty of the position of the robot the robot will be uncertain it is still not sure what is its position. So, there is an uncertainty, but its position is mean position is x_{t-1} and its covariance is P_{t-1} . Now the robot is getting a control input U_t . So, it is getting a control input U_t and then it moves to the new position here.

So, that is the new position; let me change the color to something ok. So, this is the new position of the robot. So, after getting control input U_t the robot has moved to x_t . And it has estimated the position \hat{x}_t . So, we know that \hat{x}_t that is the new position \hat{x}_t is a function of x_{t-1} and U_t and that is the odometry.

So, if you know the if it is a differential robot we know how to calculate the new position x_t . So, \hat{x}_t can be obtained and its P_t as we know that P_t which is the covariance at this. So, this is x_{t-1} sorry x at t and there will be a covariance \hat{P}_t also. So, this \hat{P}_t again we know that it can be obtained as $F_x P_t P_{t-1}^T, F_x P_{t-1} F_x^T + F_u Q F_u^T$.

So, we can actually write this as Q where Q is the covariance of the sensor used for measuring the travel or it is the encoder uncertainty. So, we will be able to get this x_t and P_t that is actually shown here. So, now, we have a x_t and P_t . Now, what we need to do is to ok.

(Refer Slide Time: 14:06)

Kalman Filter for Mobile Robot Localization

Robot Position Prediction: **Example**

$$\hat{x}_t = \begin{bmatrix} x_{t-1} + \frac{\Delta s_r + \Delta s_l}{2} \cos(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2b}) \\ y_{t-1} + \frac{\Delta s_r + \Delta s_l}{2} \sin(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$

$$P_t = F_x P_{t-1} F_x^T + F_u Q F_u^T$$

$$Q = \begin{bmatrix} k_l |\Delta s_l| & 0 \\ 0 & k_r |\Delta s_r| \end{bmatrix}$$

Odometry Z_t

So, if we if we really use a sensor sorry if you use the relationship for the differential robot then we will get x_t as like this. So, $\begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix}$. So, x_{t-1} will be $\begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \frac{\Delta s_r + \Delta s_l}{2} \cos \theta$.

So, this relationship we saw in the previous derivation. So, we will be getting x_t like this and then we will be getting the P_t like this.

And where Q_t which is basically the sensor covariance $K_r |\Delta s_r|$ and $K_l |\Delta s_l|$. So, that is what we get. So, from here to here we can actually do once we know the kinematics of the robots and if you have the covariance of the sensor the sensor uncertainty is known we will be able to calculate.

So, we will be getting the new position \hat{x}_t and the \hat{P}_t . So, that is what we do in the prediction updates. So, once that is done the next one is the perception update in perception update what we need to do is to see what the robot is actually seeing from this from here.

So, we will assume that there is a sensor attached to the robot and the robot is using its sensor and using that to collect all the information from the surroundings. So, if it is a laser sensor. We assume that the laser sensor will be able to identify this wall, this wall, this wall and this wall.

So, all these walls it will be able to identify using the sensor. So, that is what actually the robot will be getting as an information Z_t . So, Z_t will be what the robot is seeing using its sensor. So, it will say that from my position current position here I am able to see a wall at a radius r_1 at a at the distance r_1 and then angle α_1 with respect to its own reference frame it will tell.

So, robot will be having a reference frame. So, using the reference frame you can say I am seeing a wall at $r_1 \alpha_1$. I am seeing a wall at $r_2 \alpha_2$ and I am seeing a wall at $r_3 \alpha_3$ and that is the way how the wall can actually be characterized you have distance and an orientation at what orientation robot is seeing.

So, that is what you can see. So, that is basically the Z_t which is the observed observations of the robots. So, you have the Z_t . Now we have the map already given to

the robots and the robot actually check with the map ok. So, this is the map and map has got a coordinate reference frame here and the map has got all the information from this reference frame you have a wall here at one r and α you have another wall here you have another wall here all these informations are there in the map.

But you cannot really compare this one and this one because all these measurements are with respect to the reference the robot frame. So, what the robot will do? Will actually convert the look at the position where the robot is currently and see from that position what are the images it is supposed to see what are the objects it is supposed to see.

(Refer Slide Time: 17:15)

Kalman Filter for Mobile Robot Localization

Robot Position Prediction: *Example*

$$\hat{x}_t = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$

$$P_t = F_x P_{t-1} F_x^T + F_w Q F_w^T$$

$$Q = \begin{bmatrix} k_x / \Delta s_r & 0 \\ 0 & k_y / \Delta s_l \end{bmatrix}$$

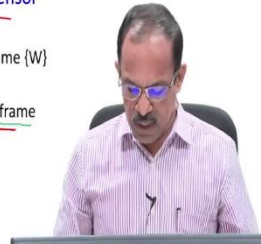
Odometry

And that is basically the \hat{z}_t ok. So, z_t we have I is equal to 1 2 3 in this case and then there are there are many things robot supposed to see if it is at this position ok, if it is exactly at this position it should have actually seen this also or this also like this also I mean there can be many things which the robot will be able to see if it is exactly some location.

(Refer Slide Time: 17:42)

Observation

- The second step is to obtain the observation Z_t (measurements) from the robot's sensors at the new location at time t
- The observation usually consists of a set n_o of single observations z_j extracted from the different sensors signals. It can represent *raw data scans* as well as *features like lines, doors or any kind of landmarks*.
- The parameters of the targets are usually observed in the sensor frame $\{S\}$:
 - Therefore the observations have to be transformed to the world frame $\{W\}$ or
 - the measurement prediction have to be transformed to the sensor frame $\{S\}$.
 - This transformation is specified in the function h .

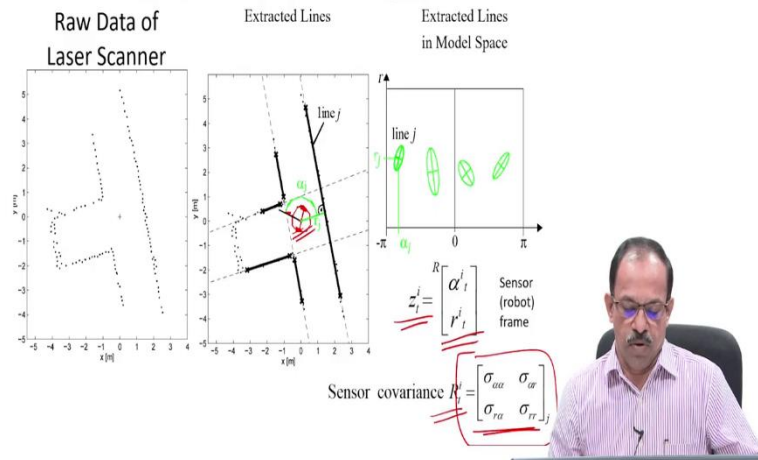


So, we will look that on that one and then try to predict what are the things that robot is seeing ok. So, that is the Z_t measurements. And then the observation usually consists of set of single observations extracted from the different sensor signals like features like lines doors any kind of things ok. So, we will put that as sensor ah. So, these are the robot will be able to see from the sensor ok, these are all shown in the sensor frame ok.

So, either you need to change everything into the world frame or we can change everything into the sensor frame which any one of these can be done and this transformation we call this as the transformation h ok. So, the measurement prediction have to be transform to the sensor frame. So, you have prediction based on the map. So, that all will be converted to the sensor frame using the transformation h ok.

(Refer Slide Time: 18:36)

Observation: *Example*



So, how do we do this? So, now, this is what actually the robot is seeing using the sensor and then it will actually extract the lines. So, these are the lines that will be extracting using some image processing algorithm it will actually identify the edges and then each one of these will be mapped to $r_j \alpha_j$.

So, it will be Z_{t_i} will be $\alpha_i r_i$. So, you can i can be any number of numbers can be there and we assume that the sensor has got some kind of a covariance σ_α and σ_{rr} ; that means, there is an error in the measurement of α and r . So, there can be some kind of an uncertainty in the measurement also and that is the sensor covariance r . So, you will be having multiple observation I can be many things because there are many lines.

So, all these can actually be mapped to a model space we can see there can be 1 2 3 4 measurements. So, $\alpha_j r_j$. So, there will be 4 things the robot sends the sensor is actually seeing 4 objects or 4 features are identified with its parameters $r_j \alpha_j$ are these 4. So, that is the observation example it is all from this location it is able to identify the get the features. So, that is the observation.

(Refer Slide Time: 19:53)

Measurement Prediction

- In the next step we use the predicted robot position \hat{x}_t and the map M to generate multiple predicted observations

$$\hat{z}_t$$

- They have to be transformed into the sensor frame

$$\hat{z}_t' = h^t(x_t, M^t)$$

- We can now define the measurement prediction as the set containing all n_t predicted observations

- The function h^t is mainly the coordinate transformation between the world frame and the sensor frame

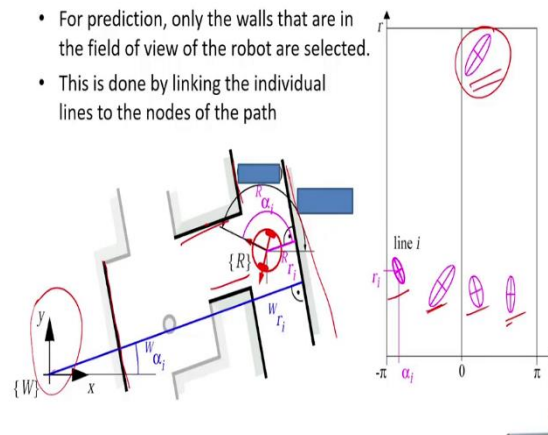


Now, the next one is the measurement prediction is \hat{z}_t ok. So, as I mentioned it will check from the map and then based on the map you will try to find out the predicted observations if the robot is at that location. So, this is something which the algorithm will try to do the position is given of the robot position is given it will try to find out what are the features that near to that and then say that these features are supposed to be seen by the robots.

And since we transfer them to the sensor frame we use the transformation h_j and then get the (Refer Time: 20:32) \hat{z}_j ok. So, there will be predicted observations this h_j is mainly the coordinate transformation between the world frame and the sensor frame that is the h_j .

(Refer Slide Time: 20:46)

Measurement Prediction: *Example*



So, in this particular case you will see that this is the world frame W is the world frame. So, you have this world frame and the features are given like this ok, each one of these features can be given as like this. Now, we convert that into the sensor frame and then see what are the features of this the robot is supposed to see and you will see that all those lines which the sensor actually saw are there, but in addition it says that this wall also supposed to see by the robot.

If it was actually in this location it should have been seeing this wall also. So, that is what actually you can see here. So, there is a feature here also. So, 1 2 3 4 5 features the robot is supposed to see, but it is seeing only 4 features by using the sensor. Now, we will try to map each one of the match each one of these features and then try to see corresponding to this what is the robot actually seeing corresponding to this is the ranging scene like that we will check each one of these ok.

(Refer Slide Time: 21:53)



- The generated measurement predictions have to be transformed to the robot frame $\{R\}$

$${}^W z_{t,i} = \begin{bmatrix} \alpha_{t,i} \\ r_{t,i} \end{bmatrix} \rightarrow {}^R z_{t,i} = \begin{bmatrix} \alpha_{t,i} \\ r_{t,i} \end{bmatrix}$$

- According to the figure in previous slide the transformation is given by

$$Z_t^j = \begin{bmatrix} \alpha_t^j \\ r_t^j \end{bmatrix} = h^j(x_t, m^j) = \begin{bmatrix} {}^{(w)}\alpha_t^j - \theta_t \\ {}^{(w)}r_t^j - x_t \cos({}^{(w)}\alpha_t^j) + y_t \sin({}^{(w)}\alpha_t^j) \end{bmatrix}$$

and its Jacobian by

$$H_t^j = \begin{bmatrix} \frac{\partial \alpha}{\partial x} & \frac{\partial \alpha}{\partial y} & \frac{\partial \alpha}{\partial \theta} \\ \frac{\partial r}{\partial x} & \dots & \dots \\ \frac{\partial r}{\partial x} & \dots & \dots \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ \cos({}^{(w)}\alpha_t^j) & \sin({}^{(w)}\alpha_t^j) & 0 \end{bmatrix}$$



So, that is the stage where we try to do the matching ok. So, the prediction basically all these will be converted to the robot frame and there will be a transformation matrix h_j which can be given like this. Because you have α and θ while ϕ with respect to the world frame and θ with respect to the robot frame and therefore, this $\alpha_t - \theta$ is the orientation of the robots.

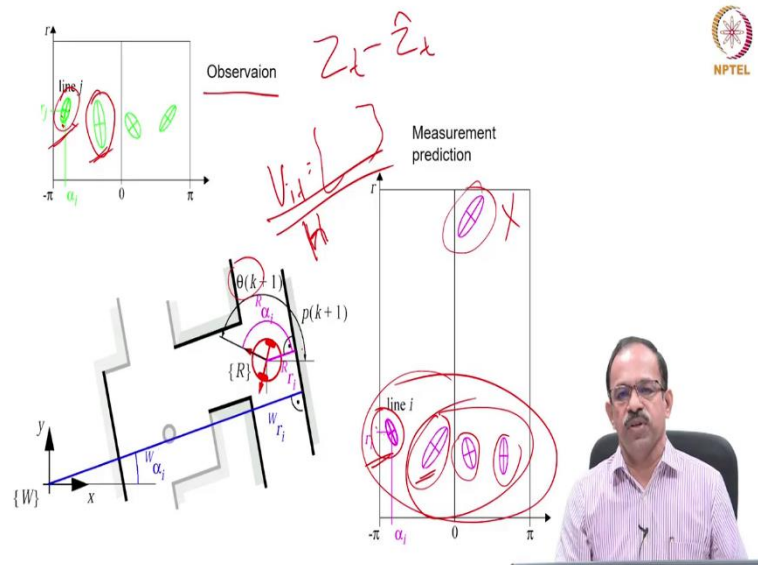
So, $\alpha_t - \theta$ will be the orientation difference between the world frame and the sensor frame similarly the position will be $r_t - x_t \cos \alpha + y_t \sin \alpha$. So, if you see the difference the position difference r_t is the distance of the feature from the reference frame or the world frame and x is the position of the robot x_t y_t .

So, we will try to find out the difference between the actual r and then $x_t \cos \alpha$ and $y_t \sin \alpha$ that will give you the variation from the world frame to the sensor frame. So, this is basically the transformation matrix which transforms the images from the world frame to the robot frame and you can get this Jacobian also.

You can take the partial derivative of this with respect to α and you will be getting the Jacobian which will be obtained like this oh sorry I will partial derivative of this with respect to x y θ you will be getting this as the Jacobian. So, this H capital H the Jacobian

will be this one for $\begin{bmatrix} 0 & 0 & -1 \\ \cos({}^{(w)}\alpha_t^j) & \sin({}^{(w)}\alpha_t^j) & 0 \end{bmatrix}$ will be the Jacobian of the transformation. So, you have this Jacobian obtained.

(Refer Slide Time: 23:50)



Now, we look at this is the observation the green one is observation and this is the measurement. Now, what the robot has to do is to match each one ok. So, it will check the first one it will take ij is equal to 1 and $i = 1$ here and see if what is the difference between this one and this one.

Similarly it will check what is the difference between this one and this one and this one and this one similarly all these. So, you will see that if this measuring or finding out the different that is $z_k - \hat{z}_k$ this first one this one and this one the difference will be very small, but the difference between all others all other measurement prediction will be very large.

So, if it is smaller than a particular value then only it will be considering that these are the same ok. So, if the difference between this one and this one is smaller than a particular value that is decided by the programmer then we will say that what we are seeing and what is observed are the same what it is supposed to see and what it is seeing are the same.

Then we say that these two are matching and the other others are not matching with this. Similarly we will take the next one and then say find out which one is the matching one and the difference is smaller than a particular value then say that these two are the matching ones. So, we will say that this one and this one are matching.

Like that we will take and we will say that this one there is nothing to match so, we will discard it. Now, for each one of this matched one we will try to find out what is the V_k^i

that is what is the innovation between all these matched one that is the innovation that we get for only for the matched one and for the matched one we take the H also or we will find out the transformation the Jacobian H corresponding to those matched images. So, that is what we do.

So, we can see here this is the well coordinate frame and this R and here this is α and that θ is the orientation of the robot and W_α is the orientation of the feature with respect to world frame and R_α is the orientation of the feature with respect to the robot frame. So, you will be able to get that H the transformation matrix and then we will be getting the innovation V_t^i also for the matched features ok.

(Refer Slide Time: 26:18)


Kalman Filter for Mobile Robot Localization

Matching

- Assignment from observations z^j (gained by the sensors) to the targets z_t^j (stored in the map)
- For each measurement prediction for which a corresponding observation is found we calculate the innovation:

$$v_t^j = \begin{bmatrix} z_t^j - z_t^j \\ r_t^j \end{bmatrix} = \begin{bmatrix} z_t^j - h_j(x_t, m_j) \\ r_t^j \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_t^j \\ r_t^j \end{bmatrix} - \begin{bmatrix} {}^{(w)}\alpha_t^j - \theta_t \\ {}^{(w)}r_t^j - x_t \cos({}^{(w)}\alpha_t^j) + y_t \sin({}^{(w)}\alpha_t^j) \end{bmatrix}$$




and its innovation covariance found by applying the error propagation law:

$$\sum_{N_i} = H^j P H^{jT} + R_i^j$$

Sensor covariance $R_i^j = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix}$

- The validity of the correspondence between measurement and prediction can e.g. be evaluated through the Mahalanobis distance:

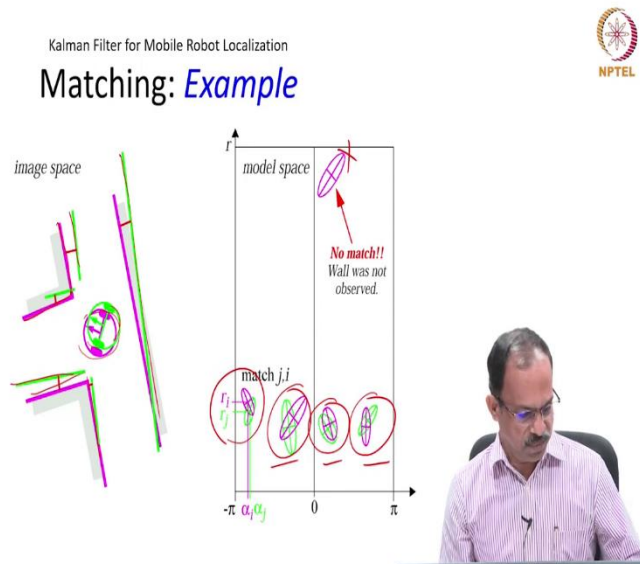


So, as I mentioned the V_t^i will be the difference and this is the H that is the sorry this is the H which is the transformation matrix in this case ok. So, the validity of the corresponding between the measurement and prediction can be evaluated through something called in a Mahalanobis distance. So, basically Mahalanobis distance is a small value which is defined.

So, we will see that if the difference is smaller than this then we take it as a matching feature if it is more than that we say that they are not matching. So, we cannot really take the difference between those two and we will using that Mahalanobis distance we will try to find out all the innovations ok.

And once we got the innovation will try to find out the innovation covariance using this relationship $H^j P_t h^j + R_t$ where R is the sensor covariance and P_t is the covariance at the current position and H is the Jacobian of the transformation matrix. So, we get the σ in σ innovation covariance ok.

(Refer Slide Time: 27:26)



So, now ah; so, as I mentioned there is no match between these and there is a match between these two this one this one and this one. So, we just look at what are the features matched. So, only the matched one will be used for calculation of innovation. So, we know that these two are matching though there is a difference between the r and α in both cases.

So, that is basically tells that the position of the robot is not the same as what actually it is supposed to be and therefore, each one of these will try to find out the innovation. And as you can see here; so, you will see that this is the position of the robot and as per the robot position it will take the measurement it will and it will see the features.

So, that these are the features you just see, but as per the map it was supposed to be like this the robot was supposed to be like this if there is a real actual position of the robot is here should I have actually seen it in this format ok or in this position and orientation. So, that is the difference. So, there is a difference between what the robot is seeing and what the robot is supposed to see. So, we will take that information for correcting the position of the robots ok.

(Refer Slide Time: 28:41)

Kalman Filter for Mobile Robot Localization

Estimation: Applying the Kalman Filter



- Kalman filter gain:

$$K_t = \hat{P}_t H_t^T (\Sigma_{P_t})^{-1}$$

- Update of robot's position estimate:

$$\hat{x}_t = \hat{x}_t + K_t V_t$$

- The associated variance

$$\text{Co-variance, } P_t = \hat{P}_t - K_t (\Sigma_{IN_t}) K_t^T$$



So, we get this Kalman filter gain using the information σ innovation and P_t and H_t and then update the robots position using $x_t = \hat{x}_t + K_t V_t$ where K is the Kalman gain and V_t is the innovation that the difference between observed and action and the covariance associated with this $P_t - K_t \sigma_{innovation} K_t^T$ where K is the Kalman gain.

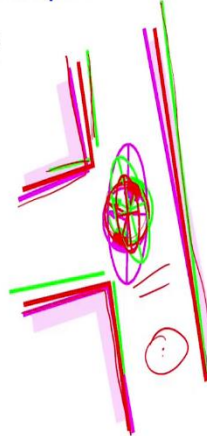
So, we know Kalman gain can be obtained from the covariance and the σ innovation and once you have this we will be able to calculate the new x_t which is the position of the predicted or the estimated position after the observation will be x_t which will be $\hat{x}_t + K_t V_t$.

So, as I mentioned it will be the previous position after the prediction plus a gain multiplied by V_t that is the difference between the observation and the prediction. So, you will be getting the covariance here. So, covariance and the position. So, that is what we needed. So, we want to get the x_t and P_t at the location.

(Refer Slide Time: 30:01)

Estimation: *Example*

- Kalman filter estimation of the new robot position
 - By fusing the prediction of robot position (magenta) with the innovation gained by the measurements (green) we get the updated estimate of the robot position (red)



So, now this is how it will be finally. So, you can see that the prediction of robot position was using the magenta. So, prediction is basically what the robot predicted using its odometry. So, that is the prediction here. Then the measurements actually showed that if the this was the what the robot is seeing, but supposed to see and the robot position is somewhere here.

So, that green one is the position and its covariance also you can see. So, you can see the magenta the ellipse with magenta color is very large. And then the prediction one also the green one it again shows it is a large green circle ellipse and now when you fuse these two we are actually getting the new position as the red one.

So, the red one is the new position and the red ellipse is the new uncertainty. So, you can see from this position it was initially here it and is once it moved and it was only the prediction then there was a large uncertainty which represented by the magenta ellipse. And then when we use the sensors it actually predicted something saying that it is seeing something different from the what is actually the robot is seeing not correct.

But as per the map it should have been somewhere here and the robot should position should have been here. And now we have fuse these two information then we will get a new estimate which actually says that the robot position is somewhere in between these two.

And its uncertainties now reduced which is shown by the red ellipse which says that the uncertainty has come down drastically. So, this is the step that the robot will be taking at

every moment every step where the robot moves it will calculate using the odometry and then it will update it using the sensor and again it will move and again keep on doing this.

So, this is why the robot needs to have a bit of computation as the as it has to move. So, it has to observe, then it has to identify the edges or identify the features do the matching and then recalculate its position and correct its position and then it move forward.

So, that is why robots are bit slow because just to do a lot of computation as it has to move it is not just like you command the robot to move and then robot keeps moving it has to recalculate its position and then only it will be able to get a direct and decide path for it to move. So, that is the basic principle of Kalman filter based localisation.



(Refer Slide Time: 32:43)

Autonomous Map Building

Starting from an arbitrary initial point, a mobile robot should be able to autonomously explore the environment with its onboard sensors, gain knowledge about it, interpret the scene, build an appropriate map and localize itself relative to this map.

SLAM : Simultaneous Localization and Mapping

- SLAM is a process by which a mobile robot can build a map of an environment and at the same time use this map to deduce its location. In SLAM, both the trajectory of the platform and the location of all landmarks are estimated online without the need for any a priori knowledge of location.

I hope you understood this. So, these two methods that is the we discussed about two methods for localisation. So, the Kalman filter was the method where actually we do the localisation if we know the previous position and its covariance. And this assumption in this both the methods were that we know the map and there is a map available for the robot to check or the robot to verify its position or I mean cross check with what it is seeing.

But in many cases the robot will not be having the map of an environment especially when the robot is going to an unknown area there is no map available to the robots. So,

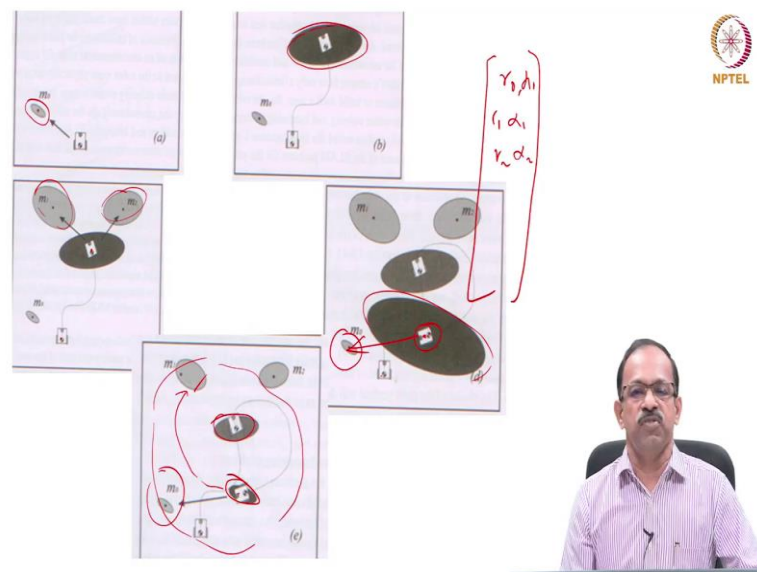
in that case in such cases the robot needs to do a localisation at the same time it has to do the map building also. So, it has to build the map and then localize itself in that map.

So, this is basically known as autonomous map building. And then localizing itself. So, you need to do both map building and localisation ok. So, we this is basically the simultaneous localisation and mapping. So, once the map is known it can do the localisation using the Kalman filter or other methods, but when the map is not there it would not be able to use the method.

So, we need to go for a simultaneous localization and mapping method which is known as the SLAM which is again a very well known and popular terminology in robotics field. So, it is a process by which a mobile robot can build a map of an environment and at the same time use this map to reduce its location. So, it has to build the map and then use the map to reduce its location.

So, both the trajectory of the platform and the location of the landmarks are estimated online without the need for any a priori knowledge of location. So, it does not need to have a priori knowledge of location because the robot will be using the information collected using sensors to create the map and the same time localize it. So, this is basically known as the slam.

(Refer Slide Time: 35:00)



Just to explain how it works. So, then we look at look into the mathematical formulation. So, the basic principle is that the robot starts with a known position assume that the position is known the initial position and uncertainty is known then it starts with the location and then it looks around and then see something.

Suppose the robot is seeing an object m_0 it will actually create a an information it will create a database I am seeing an object at a distance of r_0 and α_0 something like that it will create a map from its current location. So, the current location is known. So, it will know that from this location I am seeing an object which is $r_0 \alpha_0$.

And there is an uncertainty because the robot position is uncertain and the sensor usually is uncertain I mean it has got uncertainties errors. So, there will be an error in the object position also. So, that also will be recorded by how much will be the error that in the position of the object and the robot starts moving further and as it moves its uncertainty increases its position uncertainty increases because its only having the odometry.

So, at every odometry calculation the uncertainty increases the robot is very much uncertain about its position. So, that is why it is a large ellipse. And again it will be seeing some objects. So, it will actually see two objects in this case any objects number of objects. So, we will record it as $m_1 m_2$ it will say I am seeing an $r_1 \alpha_1, r_2 \alpha_2$, but from my current position this is my current position based on this position it will calculated what is the current position and from that position it is seeing two objects.

And it just got uncertainty because the position is uncertain and the sensor uncertainty is there therefore, there will be large uncertainty in the object also ok the position of the object. And it keeps on doing this it keeps moving around and then as it reaches here its uncertainty is very huge because it has traveled all the distance. So, it has become highly uncertain about its position.

But then from this position it sees the same object m_0 ok. So, now, we have two observations for the same object from one from this position and one from this position and there will be a difference or then it can calculate if it does from here what should when being the should have been the position of the object if it is in this robot is in this position.

Because it is already there in the map this object is already there in the map and now the robot can actually use that information to recalculate its position using some kind of a data fusion and once it does that then the uncertainty will come down drastically then this position uncertainty will come down drastically.

Now, this uncertainty has come down because it has seen the same object, but the same feature it has seen and therefore, since it is already there in the database the robot will be able to use that information. Fuse these two information and then recalculate its position similar to the Kalman filter localisation we saw it will be able to recalculate. And once it recalculate this position it will back calculate and find out what is the uncertainty in this also.

So, similarly it will be able to recalculate this position. Like this the robot keeps on moving and if it moves further from here it again it will see the something here it will recalculate this position and record the position of the object new object based on the information.

And after going through this many times the robot will be having a much more clarity on are the objects in the arena. So, it will be creating a much better map and then it will be able to localize itself using the fusion method or some algorithms that we already use for localisation.

So, this is the basic principle of SLAM that is only two to explain how what is the logic of using the SLAM algorithm for getting the map and localisation. So, this way the robot will be able to identify or create a map of the environment and then say that within this map I am in this position. So, that is the localisation. So, this is possible for the robot to do. So, that is the basic principle of simultaneous localisation and map.

(Refer Slide Time: 39:28)



Mathematical formulation of SLAM

X_T Robot path
 U_T Robot motion
 M True map
 Z_t Observations in the sensor frame

$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$

The SLAM problem is recovering the model map M and robot path X from odometry U and observations Z .

Online SLAM

$P(x_t, M | Z_t, U_t)$



So, to go to the mathematical formulation of SLAM. So, we can actually say that X_T will be the path of the robots I mean we can actually say it is a set of points the points will be and X_T will be having its own space. So, that will be having like you know x y and θ will be the parameters at every t it will be finding out what is the new position that is basically the robot path.

And then you have the control input U_T which is the motion robot motion control input similar to the previous case we discussed and then we have a M which is the map of the environment the map which is being created by the robots which is the M and Z_T is the observation what the robot is seeing in the sensor frame. So, the SLAM problem is to recover the model map M and path X from odometry and observations. So, we have only odometry and observations.

So, using this we try to create the map and get the path X . So, when you say create the map that we are actually making the map and get the path X means we are trying to find out the location of the robot. So, the SLAM problem is basically recurring the model map and robot path from observations ok.

So, the online SLAM is basically getting the position current position of the robot using the map information current information and the control inputs that is basically known as the online SLAM problem offline SLAM you need to have the complete path identified at every point it has to be updated ok.

(Refer Slide Time: 41:20)



Extended Kalman Filter SLAM

- Extended state vector: $y_t = (x_t, m_0, m_1, \dots, m_{n-1})^T$
 $m_i = \text{features in the map}$
- Prediction
- Measurement model (observation)
- Estimation

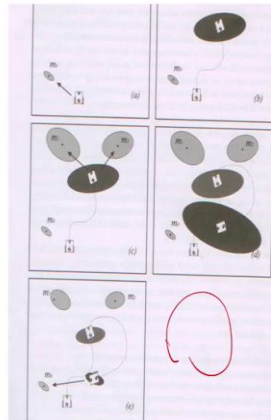


So, now we will see how can we use this Kalman filter for SLAM. So, we call this an extended Kalman filter because in Kalman filter earlier we discussed we were just looking at the position of the robot and its covariance. I mean the covariance, but in this case we need to have an extended state vector which actually talks about the positions as well as the features in the environment.

So, we have many features which can be given as $m_1 m_2$ etcetera. So, we have an extended state vector in the in this case which are the features of the m_i are the features in the map. So, we do the same procedure what we did earlier we go for the prediction and then using the measurement model observation we create the state vector, I mean this vector will be updated at every stage it will be updated.

And then whenever there is a matching between the features observed we use the Kalman filter and then update the position x_t and this x_t will be all the previous positions also will be extended. So, the there is state vector will be updated whenever there is a matching observation. So, that is basically the estimation. So, this way you will be able to use the Kalman filter principle in order to get the SLAM done ok, there are many methods to do the SLAM.

(Refer Slide Time: 42:46)



So, one is the extended Kalman filter SLAM there are many other methods also. So, this one I already explained. So, if you go through this you will see that the ok the extended vector you will be having. So, as I mentioned every stage it will try to find out the vector. So, you will be having a large vector.

So, initially you will be having an x_0, y_0, θ_0 and then you will see that an $r_1 \alpha_1$ and then it will be x_1, y_1, θ_1 then you will be giving an $r_2 \alpha_2, r_3 \alpha_3$. So, every stage you will be having this these vectors I am sorry these parameters to be updated. So, you will be having a state vector which will be having the dimensions depending on the space and the number of images.

So, this vector will be kept on updating and after many travels like this as you can see the robot keeps moving, you will see that the this vector kept on updating and at some stage you will see that there is no more updating happens then it says that that is the correct map of the environment and the correct position of the robot. So, this is the way how the extended Kalman filter will be used for localisation and mapping.

(Refer Slide Time: 43:59)

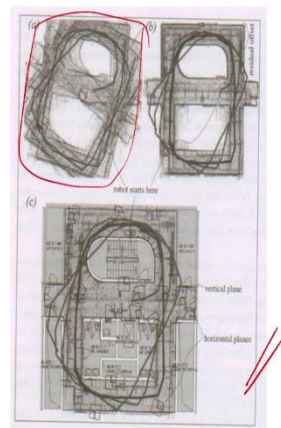


Figure 5.44 EKF SLAM using 3D laser scanner. (a) The robot starts at the center and makes three rounds. (a) Aligned 3D scans using odometry only leading to an inconsistent map. (b) Aligned 3D scans after using scan matching. The accumulated odometry error can be drastically reduced, but a small residual error remains (see offset). (c) The result of the EKF SLAM. The map built has been superimposed on a building plan for visual comparison. Notice that the offset is no longer present.



So, this shows an example. So, suppose you leave the robots in this kind of a area I mean this is the there is a building or some arena where the robot is allowed to move freely. So, the robot will start moving and then collect the information and because of the error there will be lot of uncertainty in the beginning and as we can see at the robot position and the map will be completely in a disarray.

And after some time the robot will be able to get much more clarity because it will be able to seeing the same objects again and again. So, it keeps on updating the state vector and finally, the robot will be able to get a clear map of the environment and then the robot will be able to know its position wherever it is in that particular map.

So, it will be able to do the simultaneous localisation and mapping. So, that is the basic philosophy of SLAM ok. This is basically just to introduce the SLAM concept to you of course, you need to if you are to if you are interested more.

(Refer Slide Time: 45:07)



Summary

- Localisation problem
- Challenges
- Odometry and error models
- Probabilistic, Map based localisation
- Markov and Kalman Filter localisation
- Map building
- SLAM
- EKF SLAM



You can actually refer to lot of literature there are lot of literature available in SLAM if you just search for SLAM you will see hundreds of papers on various SLAM strategies. So, you can go through it and then or you can go for some advanced courses you will be able to learn more about SLAM.

So with that let me conclude the discussion on localisation and mapping. So, we talked about the localisation problem and what are the various challenges in localising a robots in a known or unknown environment. And then we talked about the primary method of odometry and how the odometry can be used for localisation also we talked about the error models.

So, how the error propagation happens and then how we can use the error propagation to predict the errors in the position of the robot or when we do the localisation using odometry how can we predict the errors. Then we talked about the probabilistic of map based localisation where we follow a five step process to improve the localisation or accuracy by using the perception update.

It is not only the prediction update we use the perception update also to get the localisation done. And we talked about two methods Markov and Kalman filter based localisation and briefly talked about the map building and SLAM and then how EKFs can be used for SLAM applications. So, these are the things that we discussed in the last few lectures.

I hope you understood the basic philosophy of localisation and mapping and the challenges and as I mentioned if you are interested further to know about this particular area you need to go through the literature and then try to implement some of these concepts in the real field you will be an expert in localisation ok.

So, I will stop here next class onwards I will talk about path planning of mobile robots.

Thank you very much.