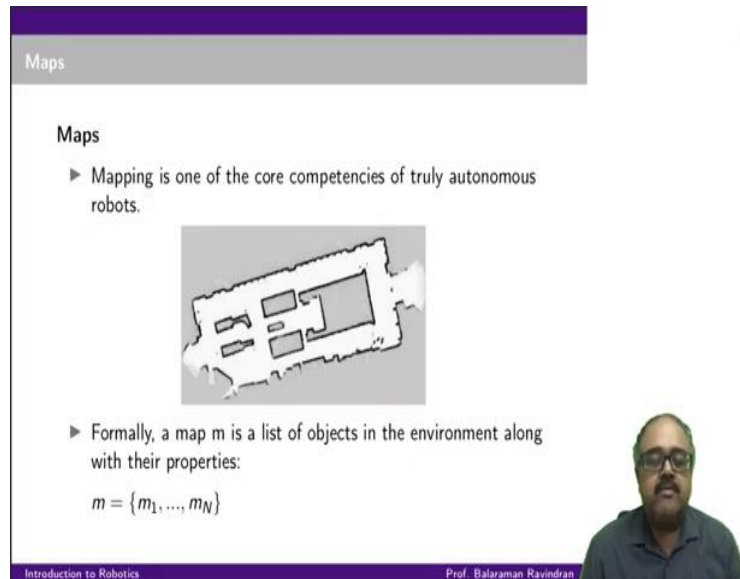


Introduction to Robotics
Professor Balaraman Ravindiran
Department of Computer Science
Indian Institute of Technology, Madras
Lecture - 39
Occupancy Grid Mapping

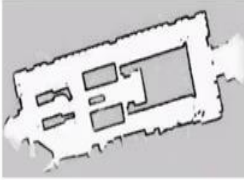
(Refer Slide Time: 00:14)



Maps

Maps

- ▶ Mapping is one of the core competencies of truly autonomous robots.



- ▶ Formally, a map m is a list of objects in the environment along with their properties:
$$m = \{m_1, \dots, m_N\}$$

Introduction to Robotics Prof. Balaraman Ravindiran

Hello, everyone, and welcome to the 3rd lecture in a week 11 in the Intro to Robotics course, where we are going to continue looking at probabilistic robotics. So we have looked at various estimation problems and various kinds of models so far. So in the last lecture, we were talking about how the motion model has to be corrected in the presence of a map.

So where does this map come from? So you can assume that the map is given to you a priori, but quite often, being able to estimate the map of the environment in which you are operating in, is one of the really an ability that we want an autonomous robot to have. So one way of thinking about it is, it is a core competency for autonomous system being able to detect, estimate the map of the region in which they are operating.

So formally, what we are going to assume is, a map m , has essentially just a list of objects in the environment, along with their properties. You could say, okay, there is an obstacle here, it is in location x y , and then the orientation of θ , so that could be m_1 . So I am going to look at this as a collection of objects, along with their properties.

In some sense, this whole map estimation problem is a lot more difficult than what we have looked at so far. All the problems in terms of state estimation and even estimating the motion models, and so on, so forth are slightly easier because even if you are operating in a

continuous space, so we are at any point of time keeping track of only one location when we are looking at state estimation. So all the Bayesian filter problems and all are slightly simpler.

But in the case of mapping, we are literally looking at so many different locations. In fact, if the, if you are operating in a continuous space, then you are really looking at being able to represent this continuous workspace in some kind of an abstract form. So make the job easier. So think of this as a more of a discrete estimation problems, we move it down into this kind of a list of objects and their properties.

(Refer Slide Time: 02:36)

Maps

▶ Maps are usually indexed in one of two ways:

- **Feature-Based:** In feature-based maps, n is a feature index. The value of m_n contains, next to the properties of a feature, the Cartesian location of the feature.

Feature-based maps only specify the shape of the environment at the specific locations, namely the locations of the objects contained in the map.

Feature representation makes it easier to adjust the position of an object, e.g., as a result of additional sensing

Introduction to Robotics Page 30 of 48 Prof. Balaraman Ravindran

And so the mapping is done in two ways. So typically, so this list of objects that we talked about, they could either be some kind of a feature-based map. So in a feature-based map, n is some kind of a feature index. So features could be the different objects in the environment, could be wall, positions of walls and partitions, and so on, so forth.

And so the value of m_n , remember m is just, m_n is an n th objects, contains other than the properties of the feature, it also contains the Cartesian location of the features in the, in the workspace. So you can just say, okay, here is the $x y$ location, I have a feature, it is like, it could be a table, so in which case, I would have some kind of dimensions of the table, and so on, so forth.

So the feature-based maps specify the shape of the environment at specific locations namely, the locations of these objects that you have on the map. It does not necessarily give you a complete description of the shape of the environment where there are no objects. So it,

sometimes it is actually useful to have this feature-based maps when I am trying to do localization in a world.

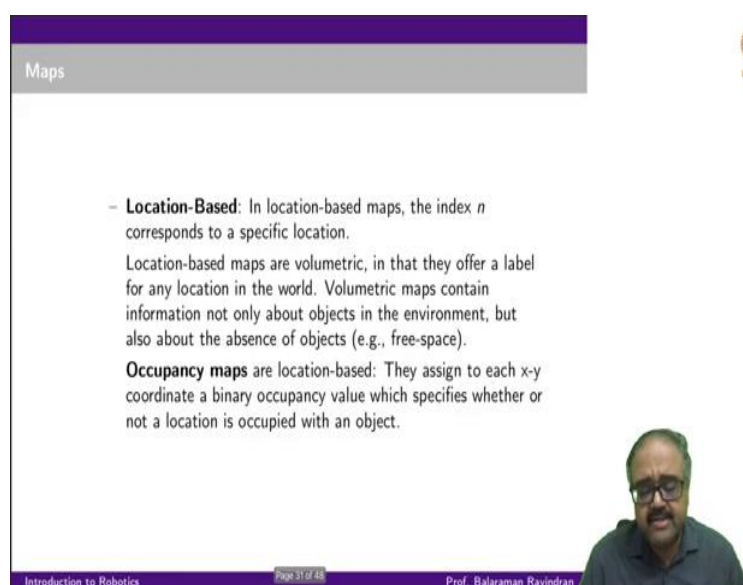
So if I am actually next to a specific feature, if I am next to, for example, if I am next to a particular building, when I am trying to do navigation, it is easier for me to know where I am, oh I am next to that building. So that way I get, I get more information when I have this kind of feature-based representation.

And likewise, another advantage with feature-based representation is it is easy to add additional features, and also it is easy to move an object around. So I can adjust the position of an object on the map just by doing additional sensing or getting closer to the object and therefore being able to refine my position estimate of the object, and so on, so forth.

So feature-based maps are easier to update in some sense. And, and I can do this without having to worry about actually verifying any of the other locations or other objects that are there in my, in my map. So I can just independently adjust the position of the new features.

So that way, it is convenient in certain conditions. But in other ways when I want to figure out if a particular location is free for me to move into, suppose, do I have an obstacle or not? Like we were talking about in the motion model. In such a case, I would really have to run through all my objects, all my features to figure out if there is a feature that intersects with the location that I want to move to.

(Refer Slide Time: 05:15)



Maps

– **Location-Based:** In location-based maps, the index n corresponds to a specific location.
Location-based maps are volumetric, in that they offer a label for any location in the world. Volumetric maps contain information not only about objects in the environment, but also about the absence of objects (e.g., free-space).

Occupancy maps are location-based: They assign to each x - y coordinate a binary occupancy value which specifies whether or not a location is occupied with an object.

Introduction to Robotics Page 21 of 46 Prof. Balaraman Ravindran

In such cases, it is easier for us to maintain what are called location-based maps. Like can feature-based map, each m_i correspond each m_i ; corresponds to a specific object or a specific feature that I want to map. In the location-based maps, each m_i corresponds to a specific location. So we call these location-based maps as volumetric maps in the sense that they offer a label for any location in the world.

So in the feature-based maps, you do not get that, in volumetric maps, they not only contain information about the actual objects, they also contain information about absence of objects. For example, free-space. So if I want to know if there is a particular location, to which I can move to having a location-based map and having seen that location as marked as free-space, is a useful thing to have.

So in many cases, when I want to do these kinds of planning, path planning, and so on, so forth, I would like to have this kind of location-based maps. When I want to do localization and I want to figure out where I am in the map, with respect to landmarks, and so on and so forth, feature-based maps are easy. And when I want to do planning, and I want to do some kind of prediction in terms of where I will end up with, location-based maps are more useful.

And quite often, in many practical scenarios, we would like to go back and forth between the two. And so for the rest of the lecture, I will look at a specific kind of location map, location-based map, which is very popular, which are called occupancy maps; occupancy grids, or occupancy maps.

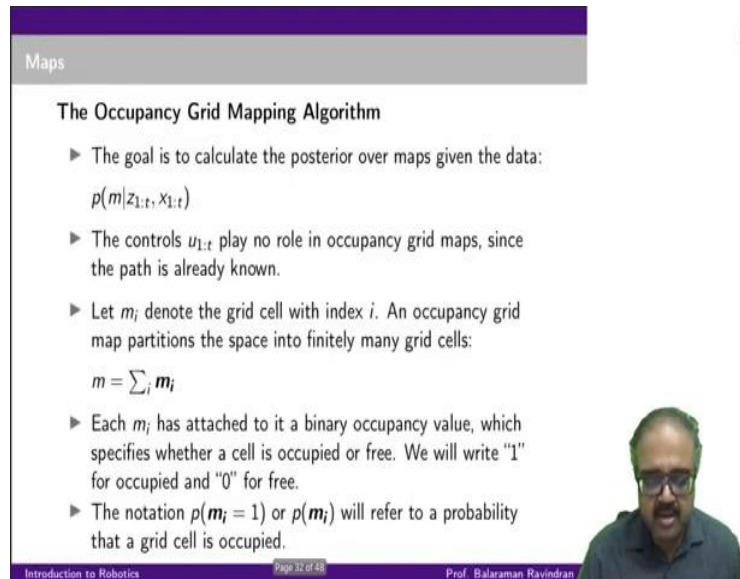
And basically, the idea here is the assigned to each x y coordinate, a binary occupancy value, it is either 0 or 1. 0 means that its location is free, 1 means locations occupied. That is why it is called occupancy maps. So 1 means a location is occupied.

So I can query any location in a world, and then figure out whether that location has a object or does not have an object. So if you actually, you can even store it as a bitmap as 0 on 1. And for, sometimes even for large spaces, you can have a compact occupancy grid or occupancy map.

So when you are operating in a continuous space, so we typically do not look at each continuous x y coordinate value for occupancy grid, occupancy maps. What we do is we actually make a grid space around it, so we make like a grid line. So we will see it later on one of the slides, and for each cell in the grid, we assign a occupancy value.

As supposed to assigning it for each x y coordinate, because if we are operating in a continuous space, then this is not really a practical approach. So what we do is we discretize the state space and make it into a grid and that is when occupancy maps are also called occupancy grids.

(Refer Slide Time: 08:06)



Maps

The Occupancy Grid Mapping Algorithm

- ▶ The goal is to calculate the posterior over maps given the data:
$$p(m|z_{1:t}, x_{1:t})$$
- ▶ The controls $u_{1:t}$ play no role in occupancy grid maps, since the path is already known.
- ▶ Let m_i denote the grid cell with index i . An occupancy grid map partitions the space into finitely many grid cells:
$$m = \sum_i m_i$$
- ▶ Each m_i has attached to it a binary occupancy value, which specifies whether a cell is occupied or free. We will write "1" for occupied and "0" for free.
- ▶ The notation $p(m_i = 1)$ or $p(m_i)$ will refer to a probability that a grid cell is occupied.

Introduction to Robotics Page 32 of 48 Prof. Balaraman Ravindran

So looking at the occupancy grid algorithm, so in some sense, the goal is to compute the following posterior. So given a sequence of observations and the sequence of states, in each state, I am making some kind of an observation, I would like to estimate a map m . What is the probability over the map m ?

So remember, m is actually a collection of m_i s. For each, m_i denotes a particular index in the map. So it is a collection of these objects. So it could, so in the occupancy grid, each m_i is going to correspond to a location.

So and the controls really play no role in the occupancy grid maps. Since we already know where we are? So for each location, I make a measurement, I really need to know whether I am, it is occupied or not occupied. So I do not really need to know, what is the control that will get me through that space?

In some sense, I can assume that the path that I am actually following for doing this mapping is already known to me. So we are going to say that each m_i denotes a grid cell with index i . So which is each, so I am going to say m_1 means index 1.

So and an occupancy grid map is essentially just a collection of many of these grid cells, m_i ; so for all the grid cells m_i . And each m_i is going to have either a 1 or a 0, so m_i equal to 1 means that it is occupied; 0 means free. And I am going to use the probability of m_i to refer to the probability that m_i is equal to 1 that it is actually occupied.

So whenever you see, when I, whenever we write p of m_i , it actually means the probability that the i th cell is occupied.

(Refer Slide Time: 10:15)

Maps

Using the ideas on the previous slide, the standard occupancy grid approach breaks down the problem of estimating the map into a collection of separate problems, namely that of estimating:

$$p(m_i | z_{1:t}, x_{1:t}) \forall \text{ grid cells } m_i$$

Thanks to our factorization, the estimation of the occupancy probability for each grid cell is now a binary estimation problem with static state.

A filter for this problem has already been discussed - **The Binary Bayes filter**

As in the original Binary-Bayes filter, our occupancy grid mapping algorithm uses the log-odds representation of occupancy:

$$l_{t,i} = \log \frac{p(m_i | z_{1:t}, x_{1:t})}{1 - p(m_i | z_{1:t}, x_{1:t})}$$

Introduction to Robotics Page 23 of 88 Prof. Balaraman Ravindran

So, and then what we do here is instead of looking at this probability, probability of m given z_1 to z_t , I am going to just say that, what is the probability of m_i given z_1 , z_1 to t and x_1 to t , and then look at the, look at this for each one of these cells independently.

So I am going to assume that the probability that a cell is occupied does not influence or gets influenced by any of the other cells in the state space. It is influenced only by the measurements I am making and the actual state values. So only, it is influenced only by z_1 to t and x_1 to t . So this makes my problem much easier. Otherwise, I will have to look at this very large joint estimation problem. So instead of that, we will assume that each cell is estimated independently.

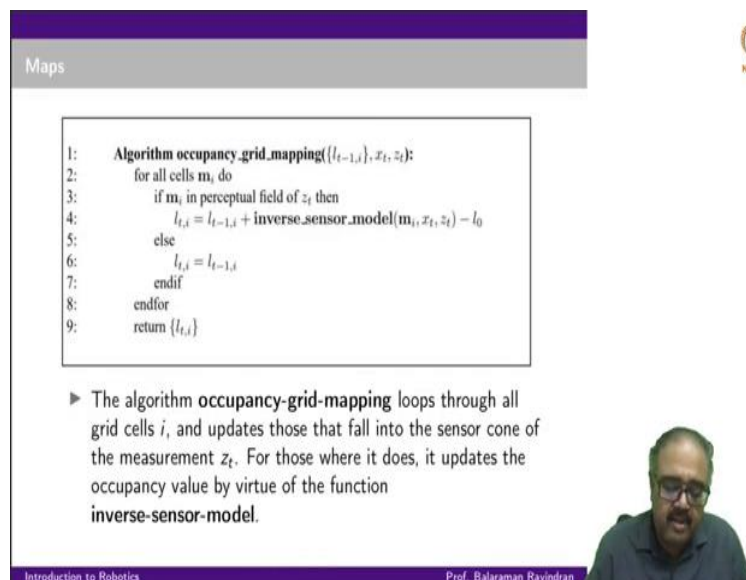
Now, if you think about it, so why are we getting here? So I have a single variable that I am trying to estimate. And this single variable is static. I am assuming that the cell is static, it is not changing, it is not going to get affected by my actions or my movement because we are talking about the map, so the map is either clear there or it is occupied.

There is a single variable taking on one binary value 0 or 1 and it does not change. So we already looked at this. We already looked at the binary Bayes filter with static state earlier and we already looked at the algorithm.

So all we really need to do now is to apply that algorithm. So where, instead of looking at x , and so looking at the state x given a sequence of observations, I am going to look at what is the value of m_i , probability of m_i given the sequence of observations, and the sequence of states I have seen.

It is exactly the same algorithm. So instead of looking at it as a state estimation algorithm, I am going to run the binary Bayes filter with static state algorithm to solve the occupancy grid mapping problem and using the same kind of log-odds representation of occupancy.

(Refer Slide Time: 12:03)



```
1: Algorithm occupancy_grid_mapping( $\{l_{t-1,i}\}, x_t, z_t$ ):
2:   for all cells  $m_i$  do
3:     if  $m_i$  in perceptual field of  $z_t$  then
4:        $l_{t,i} = l_{t-1,i} + \text{inverse\_sensor\_model}(m_i, x_t, z_t) - l_0$ 
5:     else
6:        $l_{t,i} = l_{t-1,i}$ 
7:     endif
8:   endfor
9:   return  $\{l_{t,i}\}$ 
```

► The algorithm `occupancy-grid-mapping` loops through all grid cells i , and updates those that fall into the sensor cone of the measurement z_t . For those where it does, it updates the occupancy value by virtue of the function `inverse-sensor-model`.

Introduction to Robotics Prof. Balaraman Ravindran

So this algorithm should look very familiar to you. This is you can go back and look at the binary Bayes filter statistic algorithm. And here for each, so what do I have here? So I have the likelihood of the occupancy that I have seen so far, starting from the beginning.

So basically, t minus 1 is like my belief state on the map, it is like the belief state on the map. So what is the likelihood that cell i is occupied at time based on the measurements I have made till time t minus 1, so that is what the set is.

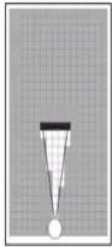
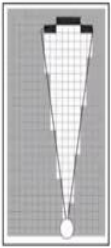
So we will have one entry for each value for i and then you have your current state and your current measurement. Now, for all cells m_i , now if m_i is in the perceptual field of z_t , so what do I mean by that? That means that when in x_t , I am making a measurement z_t , so how much

of the world can I see at z_t ? Depending on what kind of sensing do I have, it is not that I am only looking at the current state I am in. So I could look at a region of state around it, as we will see in the next couple of slides.

(Refer Slide Time: 13:17)

Maps

▶ The function **inverse-sensor-model** implements the inverse measurement model $p(m_i|z_t, x_t)$ in its log-odds form.

(a)  (b) 

A somewhat simplistic illustration of such a function for range finders for two different measurement ranges. The darkness of each grid cell corresponds to the likelihood of occupancy.

Introduction to Robotics Page 25 of 88 Prof. Balaraman Ravindran

Maps

```

1: Algorithm occupancy_grid_mapping( $\{l_{t-1,i}\}, x_t, z_t$ ):
2:   for all cells  $m_i$  do
3:     if  $m_i$  in perceptual field of  $z_t$  then
4:        $l_{t,i} = l_{t-1,i} + \text{inverse\_sensor\_model}(m_i, x_t, z_t) - l_0$ 
5:     else
6:        $l_{t,i} = l_{t-1,i}$ 
7:     endif
8:   endfor
9:   return  $\{l_{t,i}\}$ 

```

▶ The algorithm **occupancy-grid-mapping** loops through all grid cells i , and updates those that fall into the sensor cone of the measurement z_t . For those where it does, it updates the occupancy value by virtue of the function **inverse-sensor-model**.

Introduction to Robotics Prof. Balaraman Ravindran

Maps


Using the ideas on the previous slide, the standard occupancy grid approach breaks down the problem of estimating the map into a collection of separate problems, namely that of estimating:

$$p(\mathbf{m}_i | z_{1:t}, x_{1:t}) \forall \text{ grid cells } \mathbf{m}_i$$

Thanks to our factorization, the estimation of the occupancy probability for each grid cell is now a binary estimation problem with static state.

A filter for this problem has already been discussed - **The Binary Bayes filter**

As in the original Binary-Bayes filter, our occupancy grid mapping algorithm uses the log-odds representation of occupancy:

$$l_{t,i} = \log \frac{p(\mathbf{m}_i | z_{1:t}, x_{1:t})}{1 - p(\mathbf{m}_i | z_{1:t}, x_{1:t})}$$


Introduction to Robotics Page 23 of 66 Prof. Balaraman Ravindran

So, for example, I will just go, just look at this for the measurement model. So even though I am here in this particular state right now. So I am actually able to see a lot more states based on my field of vision. So I have this field of vision, so I am able to see a lot more states and I can actually make estimates about whether these states are also occupied or not just by making a measurement at X_t .

So when I say I am at X_t and I am making a measurement z_t that does not mean that z_t returns information only about the current cell occupied by the robot, it could actually give me information about more parts of the state space. So that is basically what we are trying to take into account here.

So, when I say if m_i is in the perceptual field of z_t , that means that the measurement z_t gives me some information about the cell i , then I update the log-likelihood using the inverse-sensor-model that I have earlier.

So if you remember this expression, so this is $l_{t,i}$ minus 1 comma i which is my old belief in the likelihood form, and then the inverse-sensor-model. So the likelihood through the inverse-sensor-model and minus 1 naught which is the initial likelihood that I start off with. So this is my update equation.

And if it is outside the perceptual field of z_t , that means, I cannot make any estimate about whether cell i is occupied or not. If it is outside the perceptual field of z_t , then I leave the likelihood as it is. So the new belief state is same as old belief state because I have not made any measurement on cell i for me to change this. And to do this for all cells m_i , at the end of it, I get the new map. This is the new belief over the map.

Remember, what we are looking for is a probability of m , given x_1 to t and z_1 to t , and this is the new probability of m given z_1 to t x_1 to t . So I hope that is clear, I am going through this a little faster because we already seen the binary Bayes filter with static state and this is exactly the similar kind of an algorithm, except that instead of looking at the belief over your current location, you are looking at the belief over a particular cell in the map whether in the occupancy grid.

So now, the interesting question is, how do I come up with this inverse-sensor-model to use in this occupancy grid algorithm? So how do we come in the occupancy grid algorithm? So what we do is as follows. So just think about this for a minute. So if the robot is here and there is an obstacle either here or here, there are two different situation where we are talking about.

Both, in both cases, the robot has the same kind of field of view. Robot has like a 15-degree cone of vision, let us say. And in one case, because there is an obstacle nearby, the robot is able to look at all these states, all the states marked in white and it can tell you these are clear and it can look at all the states marked in black and can tell you there is an obstacle there and all the grey cells, it cannot give you any information over and above what you already know. So this is not something that the robot, the current measurement can give you any information on.

And here is a case where there is an obstacle that is a little farther away, the robot could potentially, so it could potentially look at a lot more states here. The robot could potentially look at a lot more states here, and then it can give you information about all of these saying that they are clear because this is a, in the field of view there is nothing that stops it until it hits the obstacle.

Beyond the obstacle the robot does not know what is there, before the obstacle, the robot knows what are the cells and these are all free and anything outside of this field of view or behind the obstacle, it cannot give you any additional information.

So the inverse model is basically implemented in the log-odds form directly so that I can just use the inverse model here instead of saying the log-odds of the inverse model. So that is what we had earlier. So this is the log-odds of the inverse model. So instead of saying that I could sorry, yeah, I have to cancel that. So instead of saying the log-odds of the inverse

model I can just say the inverse model here and learn the inverse model in terms of the log-odds.

And so, here you can see that everything that is white means it is free, everything black means it is occupied, everything grey means I do not know, really. So how do we go about doing this?


(Refer Slide Time: 18:17)

Maps

The algorithm corresponding to the previous slide's illustration is shown below:

```

1: Algorithm inverse_range_sensor_model( $i, x_t, z_t$ ):
2:   Let  $x_i, y_i$  be the center-of-mass of  $m_i$ 
3:    $r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$ 
4:    $\phi = \text{atan2}(y_i - y, x_i - x) - \theta$ 
5:    $k = \text{argmin}_j |\phi - \theta_{j,\text{sens}}|$ 
6:   if  $r > \min(z_{\text{max}}, z_t^k + \alpha/2)$  or  $|\phi - \theta_{k,\text{sens}}| > \beta/2$  then
7:     return  $l_0$ 
8:   if  $z_t^k < z_{\text{max}}$  and  $|r - z_{\text{max}}| < \alpha/2$ 
9:     return  $l_{\text{occ}}$ 
10:  if  $r \leq z_t^k$ 
11:    return  $l_{\text{free}}$ 
12:  endif
  
```

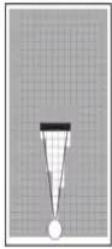


Introduction to Robotics Prof. Balaraman Ravindran

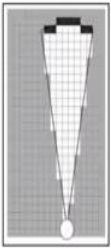
Maps

► The function **inverse-sensor-model** implements the inverse measurement model $p(m_j | z_t, x_t)$ in its log-odds form.


(a)



(b)




A somewhat simplistic illustration of such a function for range finders for two different measurement ranges. The darkness of each grid cell corresponds to the likelihood of occupancy.



Introduction to Robotics Page 23 of 48 Prof. Balaraman Ravindran

Maps




```

1:  Algorithm occupancy_grid_mapping( $\{l_{t-1,i}\}, x_t, z_t$ ):
2:    for all cells  $m_i$  do
3:      if  $m_i$  in perceptual field of  $z_t$  then
4:         $l_{t,i} = l_{t-1,i} + \text{inverse\_sensor\_model}(m_i, x_t, z_t) - l_0$ 
5:      else
6:         $l_{t,i} = l_{t-1,i}$ 
7:      endif
8:    endfor
9:    return  $\{l_{t,i}\}$ 

```

► The algorithm **occupancy-grid-mapping** loops through all grid cells i , and updates those that fall into the sensor cone of the measurement z_t . For those where it does, it updates the occupancy value by virtue of the function **inverse-sensor-model**.

Introduction to Robotics Prof. Balaraman Ravindran



And so here is a here is an algorithm that tells you what is being computed in the previous slide. So I am going to assume that the current state x_t is given by x , y , and θ ; so x , y , and θ . And the cell i , the index i denotes a cell m_i and which is centered at x_i y_i . So if you think of this as a cell, so the center of the cell is what I denote as x_i y_i .

Now r is the distance of the cell from the robot and ϕ is the angle the cell subtends with the robot; the direction, with the direction the robot is facing. So ϕ is the angle it subtends with the direction the robot is facing.

So now, among all the possible sensors that could be looking in that direction, remember z_t is a collection of all sensor readings, I pick the one that is pointed most in the direction of the object. So this basically this means that so this is the sensor, θ_j sens, is the sensor that is pointed as close to the direction of the object from the robot. So I pick that.

Now, what I do is if this object is farther away than the maximum range of the sensor, or where the sensor says there is an obstacle, if the object is farther away than the maximum range of the sensor or where the sensor says there is no obstacle, whichever is the minimum of the two, then I say that hey, I do not know anything, so I am going to return I naught.

Or if the object is too far away from the direction the sensor is looking, remember, this is the sensor that is closest to the object that is looking in the direction closest to the object but even then the object is too far away. In both these cases, I am going to say that, I do not know where it is.

So this, both these cases correspond to, so the first condition, the r condition corresponds to the cell being in one of these locations and the second condition, the one on the ϕ corresponds to the cell being in one of these locations. So in both cases, we say that we will return l_{naught} . So what does it mean to return l_{naught} ?

Remember, if you plug in l_{naught} here, I will have l_{naught} and this will become minus l_{naught} so they will cancel out. So I will basically leave the belief as it is without changing anything. So that is what I mean by saying, I will return l_{naught} there.

Then if z_{tk} , which is the distance of this, of the obstacle, z_{tk} is less than z_{max} ; that means that the robot has sensed an obstacle, and $r - z_{max}$ is less than $\alpha/2$. That means the distance to the cell m_i , the distance to the cell m_i is within the object; α is the object thickness, so it is within the object thickness from the maximum range, so in which case, I will return that the cell i is occupied.

Or on other hand, if the distance is actually less than, r as the distance to the cell if the distance is actually less than that z_{tk} , which means, z_{tk} is the distance at which I have sensed an obstacle. If r is less than z_{tk} , then I will say r is free. Then I will say the cell m_i is free; so l_{free} . So the cell m_i is free if r is less than z_{tk} .

But if r is not less than z_{tk} , if r is actually greater than z_{tk} then I am going to say it is $\alpha/2$. Then I am going to say that the cell is occupied. So what do we mean by cell is occupied? So this is an algorithm taken from the book and that is the error of the book itself.

So here this case, you should read that as so that means I have detected an obstacle that is within the maximum distance of the, within the maximum range of the sensor and the distance of the cell is within the obstacle thickness of this distance, in which case, I would say that it is occupied. So this should be z_{tk} or if it is less than z_{tk} , then I am going to say it is free. So that is basically the algorithm.

So this is how I build up a inverse range map. So I have talked about two kinds of maps. One, which is a feature-based map, which is typically more useful when I want to do localization, as we will see a little later. And the other is this kind of location-based map of which the most popular is the occupancy grid map. And we just saw that how we can use the binary Bayes filter with static state and an inverse-sensor-model in order to estimate the occupancy grid map.