

Robot Motion Planning
Prof. Ashish Dutta
Department of Mechanical Engineering
Indian Institute of Technology – Kanpur

Lecture – 11
Cell Decomposition Methods

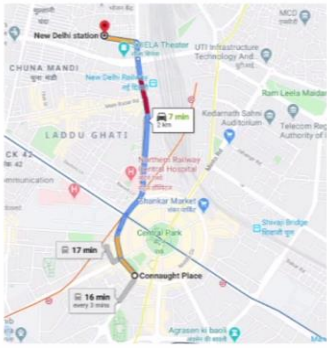
Hello and welcome to lecture number 11 of the course Robot Motion Planning. In the last class we looked at road map methods that is we make a set of pathways and then we connect the initial point to this network of paths and then find the more suitable paths that will take the robot from the initial point to the goal point avoiding obstacles. Today, we will continue with that discussion and then move on to the next topic which is cell decomposition methods which is how do you break the free space and the obstacle size space into a number of cells.

And then how do you find the path which will take here from the initial point to the goal point. Today, we will be looking at cell decomposition methods which is the next method after road map. So, very quickly I will complete a couple of points which are still left for the road map method and then move on to the next topic which is cell decomposition methods.

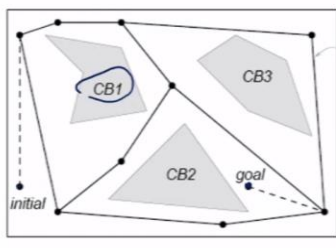
(Refer Slide Time: 01:04)

The Basic Idea of Roadmap Methods *feasible.*

Capture the connectivity of free space by a graph or network of paths.



Connaught Place to New Delhi Railway station in Google maps



Obstacle avoidance

Reference: Principles of Robot Motion, Howie Choset, et al.

So, in the last class we saw that road map methods basically captures the connectivity of free space by a graph or a network of paths. Like the example is Google maps which everybody is familiar these days that if you want to go from some point to some other point basically what it does is it finds a network of paths and then find which is the shortest path in terms of

distance and also maybe time to take you from one point where you are to the desired goal position.

Similarly, if you look at this example in the right hand side where we have work space and there are obstacles so CB 1, CB 2, CB 3 are obstacles and we have a robot in initial point and we want to go to the goal point which is marked here. So, what the road map method does is essentially it generates a network of feasible paths mean it should not go inside an obstacle and then what it does it connects the initial points to the network of paths and the goal point to that network and then find different ways of going there.

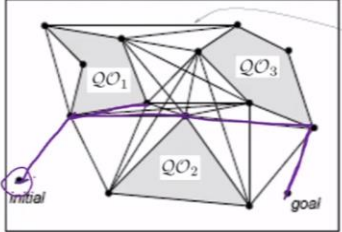
So, this is the basic idea of road map methods and from road map methods what we do get is we can get the most best path which will take you from the initial points to the goal point.

(Refer Slide Time: 02:21)

Visibility Graph methods

- Defined for polygonal obstacles
- Nodes correspond to vertices of obstacles
- Nodes are connected if
 - they are already connected by an edge on an obstacle
 - the line segment joining them is in free space
- Not only is there a path on this roadmap, but it is the *shortest* path
- If we include the start and goal nodes, they are automatically connected
- Algorithms for constructing them can be efficient

Reduced Visibility graph



Reduced Visibility Graph.
→ (Gurobo)

Now there are other methods also like visibility graph is also road map method in which we generate a set of pathways and then we find the connection between initial point and the path and the network of paths for the goal point and the network of paths and then as we can see here there are multiple paths which can take the robot from the initial point to the goal point and then we find the shortest length path etcetera.

So, these are all called road map methods because they capture the connectivity of free space by a network of feasible paths and then basically find the path which we will take it from the initial point to the goal point. Visibility graph is another method in which you can imagine

that you are standing at the initial point and which vertices are visible to you connect with straight line.

And then all the vertices are connected to the other vertices by straight lines and then it is a question of going from one point to another point simply by connecting finding the path which is the shortest length, for example, this one probably would be the shortest length path. Now visibility graph also has reduced visibility graph something we looked at in the last class.

So, reduced visibility graph now in this reduced visibility graph we basically remove the edges which are there in the concave paths to make it faster. So for this please refer to the previous lecture now the other method after this or rather the thought of the next method is that basically when human beings are going to avoid obstacles and go from one point to another point we do not actually go and go very near the obstacle and almost go touching the obstacle and then try and go to the goal point like it is being done here.

So, in this case we are going from the initial point to the node then we go to the next node then we are going to this node and then we go to the goal point. Now that is something which human beings do not took why because there is a chance of an accident because of wrong data so the distance data from the robot and the obstacles is not very accurate.

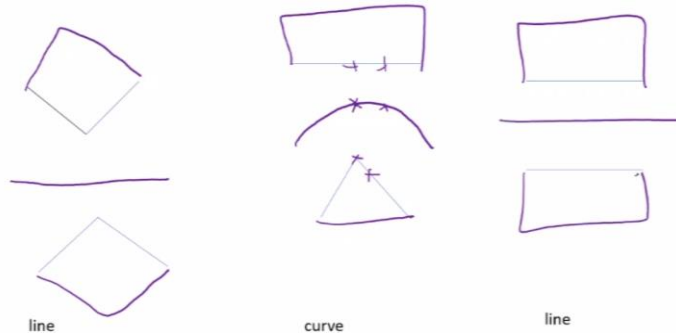
There is a chance the robot will hit the obstacle. Now to avoid that what we do is we try and find the path which will go in between the obstacles like this keeping a safe distance between two obstacles that is what we do or any biological entity does.

(Refer Slide Time: 04:46)

Polygonal spaces

In a polygonal environment, obstacles have two features, vertices and edges.

Equidistance relations are easy. Set of points equidistant are:

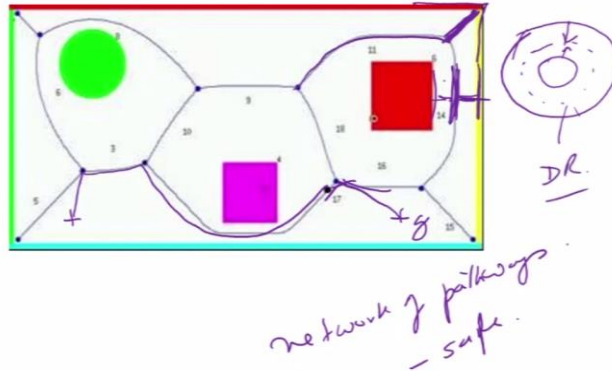


Now the idea that is basically to find equidistant between obstacles and the work space and boundaries, for example, in this particular case we are basically going to find equidistance relations between an edge and edge. So, you can guess there are two obstacles like this so this is one edge and this is another edge one vertex another vertex. Now what will be the equidistance path?

Equidistance path will be something like this in between them. Now on one side you have a flat straight line other side you have a vertex like this what will be the equidistance path it will be something like this. How do we start? We will basically start by finding the shortest distance here so between this point and this point that is the equidistance point then similarly between this point and this point that is the equidistance point and continue like that. Now if a surface has two edges like this then the equidistant path will be in between.

(Refer Slide Time: 05:38)

Deformation Retracts: Generating the Voronoi diagram and going from start to goal.



And this is basically the idea of deformation retracts where we are trying to generate a Voronoi diagram that is going from the start to the goal point by using a network of paths which are equidistant from the obstacle and the work space boundaries. So, this is a network of paths and how did we get them? They are basically equidistant. So, this line is equidistance between this and this.

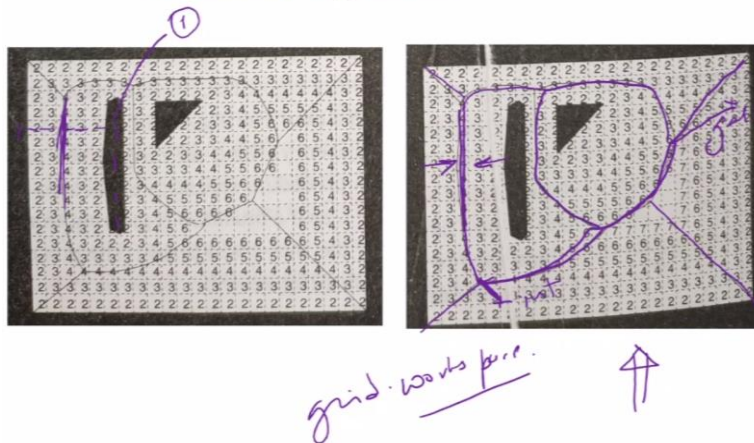
Now this here is equidistance between this side and this side. So, now if you want to go from some point to some other point basically what we do is we connect to these graph follow a path way and suppose this is my goal and I go from here and I simply go to the goal. So, again you can see that this is a network of paths or feasible pathways and the advantage of this is that it is safe it does not go too near the obstacles so there is no chance of hitting the obstacle over the wall.

Now these are also called Voronoi diagram this is also Voronoi diagram and this process is basically called deformation retract it is like if this were to expand and this were to expand they will go and meet there. It is like both the sides are deforming and then they are going and meeting there. So, similarly if we look at a doughnut shape object like this if this is shrinking and that one is expanding where will it meet?

It will meet in one dimension circle like that. So, this is basically what is called as deformation retract.

(Refer Slide Time: 07:03)

Bush fire algorithm



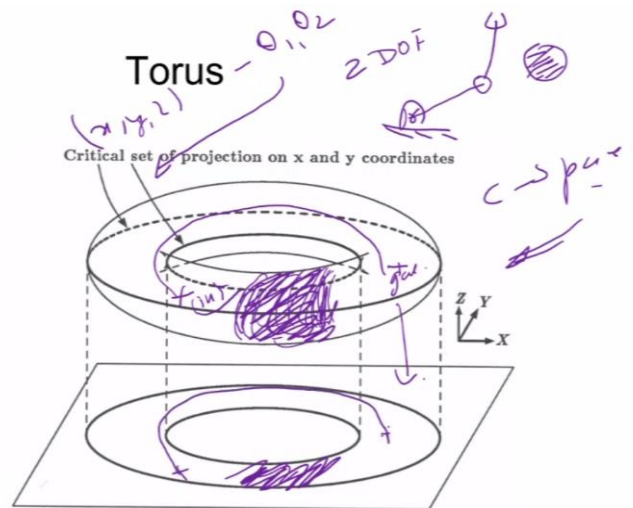
Now there are other methods fast methods by which you can make this deformation retract. Bushfire algorithm is one of them where the object is marked as 1 the number 1 and then the adjoining cells so this is basically a great kind of grid work space and my computer can compute this very, very quickly. So, this is given 1, 1, 1, 1 and the next grid to 1 is given 2 then 3 in that order then you can see this is 2 then 3, 4 in that order.

Similarly from the boundary of the work space we start with 1 this becomes 2 then 3. Now you can see it can go maximum till 4 and then it starts meeting. So, similarly what happens is if you look at this so the one on the right side is showing the completed assignment of all the grid numbers. So, whatever is the largest number that has been assigned that is the one where the two sides are meeting.

So, for example, here this is the case where if you expand from here and you expand from here this is what you will get. So, this is basically my deformation retract, but my computer would find this very easy to compute and do because this is in a great form and these are numbers. So, it can do you can write in algorithm very quickly to do this and find the deformation retract.

Now if you are here see I have initial point you simply can connect with a node suppose you want to go here this is my goal point we can simply follow this path like this and go there. So, this is basically called the Bush fire algorithm.

(Refer Slide Time: 08:40)



Now moving forward we are looking at a torus now we saw that for a 2 degree of freedom serial arm suppose we take this example we did a few classes back where you have an arm two degrees of freedom and there is an obstacle. Now the C space for this would look something like as shown here in a torus. Now there is an obstacle there so maybe part of the obstacle is going to come here on the surface of the torus now.

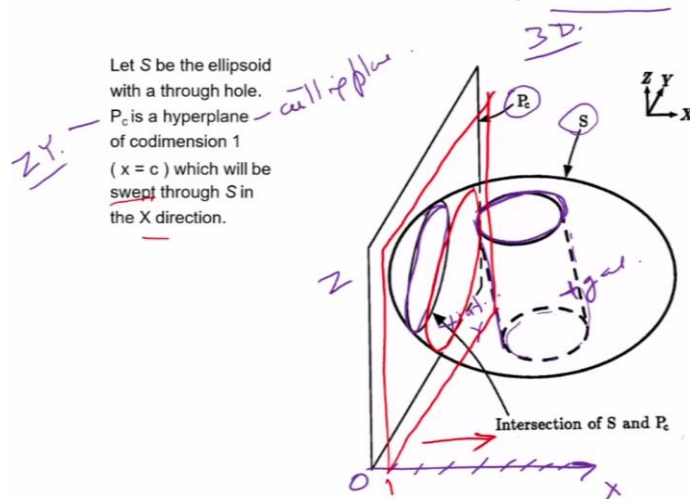
Now I wanted to do path planning on this torus. Now please note that the torus is 2D basically. The torus has two variables θ_1 and θ_2 , but when you draw it you require x, y and z in terms of dimensions. So, now if you want to do path planning let us say this is my obstacle I want to do path planning of a torus now. Now how to do the path planning on the torus?

Now again we have eyes so you can see very clearly suppose I want to go from here to here so this is my initial point this is my goal point how do I go. So, although we have eyes we can see it is not very clear how would you go, would you go around like this, would you go from angular here how would you go and if you take the projection it simply become the 2D to ellipses which is of not much use.

So, this part is fully blocked if I take a 2D projection. Now what about back side so if you want to go from here to here do you go like this or is there any other way?

(Refer Slide Time: 10:13)

Silhouette Methods: Ellipsoid structure



Now to look at this kind of a problem we move on to what is called the Silhouette method. Now we are taking this example of the Silhouette method using ellipsoid kind of structure so this is an ellipsoid that means an ellipsoid 3D so this is ellipsoid. Now this ellipsoid has a hole in there. It is a cylindrical cut which is inside there you can imagine that I hope. Now, I want to do on this particular surface I want to do path planning.

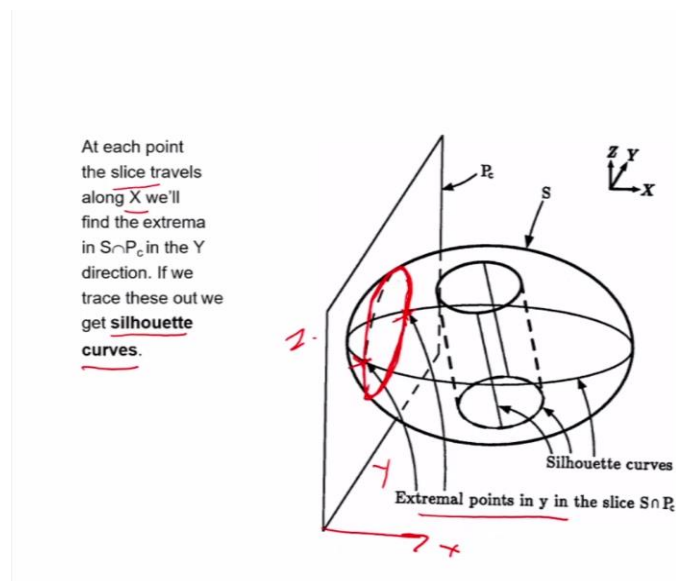
Suppose, I want to go inside here that is my goal point and my initial point is here. Now this is similar to this that this is a torus which is something like a 3D structure. Now this is also a 3D structure which is an ellipsoid with a hole and I want to do path planning here. Now in this Silhouette method what it basically does is we have the ellipsoid which is S with a hole. Now let us say that we have a hyperplane or a plane which is P_c this is a plane now we can call this a cutting plane hyperplane or cutting plane

Now this cutting plane will slice this ellipsoid at various points. So, let me draw my X axis like this, this is my X axis this is Y axis and this is my Z axis. So, this hyperplane is basically in the Z and Y plane. Now X axis what I will do is I am going to divide this X axis in the number of points and for each of these points I am going to pass the plane. So, when I pass the plane at 0 see this is my 0 then where does it cut the ellipse. It cuts the ellipse like this.

Then next what I would do I will increment this X by one unit which is 1 the plane is there now. So, the plane comes here now you can see that it is going to cut the ellipse. Where is it going to cut the ellipse? It cut the ellipse over here like this the red line. So, we keep moving this hyperplane in this direction and we keep taking slices. So, this hyperplane is going to

sweep through S in the X direction and then at every slice what we are going to do we are going to see where is the boundary of that ellipse.

(Refer Slide Time: 12:30)



So, at each point the slice travels along the X we will find the extrema points, what is the extrema points? They are basically the extrema points are these points so we are going to get this curve and what is the extrema point the extrema point is this point and that point on the curve. So, extrema point in Y in the slice is given by this one. So, this is my X axis, Y axis and that is my Z axis.

So, I took the slice what I got is this curve by cutting the ellipse. Now in that what I am going to do I am going to take the extrema point this point and this point and these are the extrema points in Y in the slice S intersection P_c . Now, if we trace these out we get what is called as Silhouette curve. So, what am I doing I am going to cut this using slices and every slice I find the extrema point and with the extrema points what we do is we get a so this is my these are the extrema points on the curve.

(Refer Slide Time: 13:34)

Observations:

- The silhouette curves are one-dimensional.
- This is not a roadmap, it's not connected.
- There are points where extrema disappear and reappear, these will be called critical points and the slices that go through these points are critical slices.
- A point on a silhouette curve is a critical point if the tangent to the curve at the point lies in P_c .

Now on the observation that we need to observe here is that the Silhouette curves are one dimensional. This is not a road map because they are not connected. Please note that these are not connected the way I do it even here. So, we will get one curve here and we get another curve there so these are not connected. So, if I take the extrema points one point is there one point is here, but these are not connected.

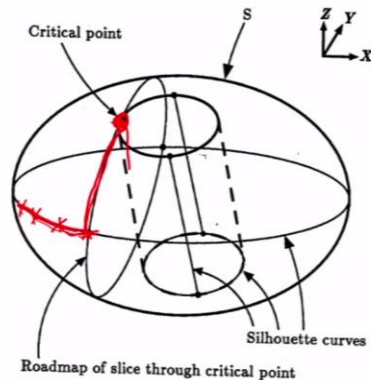
Now there are points where extrema disappears and reappear these are called critical points. Now because the structure is such that there is a hole in there so what will happen is there will be places where the extrema is going to disappear and reappear again and these points are basically what we call critical points. So, there are points where the extrema disappears and reappears and these are called the critical points and the slices that go through these points are called critical slices.

Now point on a Silhouette curve is a critical point if the tangent to the curve at the point lies in P_c . Now if you can imagine a little bit what am I trying to do? Maybe if I want to do path planning I want to go inside here that is my goal. So, basically I need to find this point I need to go inside that hole so I need to look at this point. Once I look at this point then I can go inside. So, what the critical points are doing is basically they are trying to locate where there is a change in the curve.

(Refer Slide Time: 14:55)

Critical points

We'll connect a critical point to the rest of the silhouette curve with a path that lies within $S \cap P_c$. This can be done by running the algorithm recursively. Each time, we increase the codimension of the hyperplane by 1.



So in this slice we are going from here what will happen is this is my critical point now. So, we will connect a critical point to the rest of the Silhouette curves and a path that lies within a sphere. So, now what we are doing till now is we are taking slices and at every slice I am going to get my critical points which are here, which are here. Now the case where there is a critical point what we do is for that particular slice we connect the critical point with the Silhouette curve here.

So, we will connect a critical point with the rest of the Silhouette curves with a path that lies in $S \cap P_c$. So, what we can do is these are my Silhouette curve if that is my Silhouette curve so what I do is now I want to go inside the cylinder. How do I go I can go like this now this is the critical point so I have to go like this and then I can go inside. So, I hope you can get the idea of why this critical point is important because in order to go inside I need to find where is the critical point and once I get the critical point now I know that I can go inside that hole now.

(Refer Slide Time: 16:03)

Final points

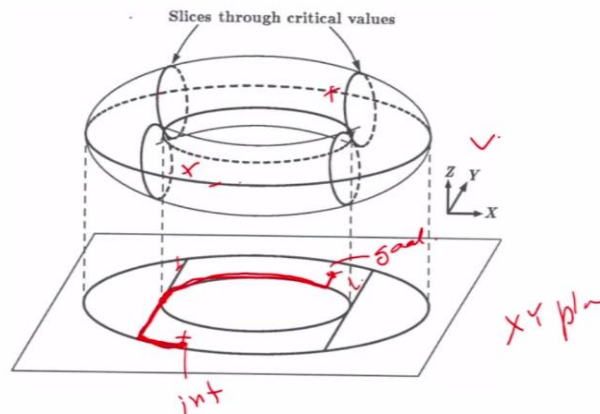
- The recursion is repeated until there are no more critical points or the critical slice has dimension 1 (it is its own roadmap)
- The roadmap is the union of all silhouette curves

Now, the recursion is repeated until there are no more critical points or the critical slices dimension 1. The roadmap is the union of all the Silhouette curves. So, what we are getting is the road map is the union of all the Silhouette curves. So, now suppose I want to go from here to there I have got my curves like this. Now these are my points on the Silhouette curve. So what I do is I basically go like this is the critical point.

So I can simply go like this now I can go inside them. So, now we understand how to go inside these ellipsoid which have a cylinder inside by using the Silhouette method this is called the Silhouette method. Now, let us come back to where we started that is the torus. Now can I use the same method for the torus? Now, if I use the same method for the torus what I am basically going to do is I will have the hyperplane which is cutting the torus like this.

So, if it cuts the torus it will be cutting here. So, I move this in this direction if this is my X direction that is my Y direction and that is my Z direction. So, in different intervals of X, I am going to move this slicing plane and at every point I am going to get the Silhouette curve this is the points where the curve is cutting. So, there is one here one there so this is my Silhouette curve.

(Refer Slide Time: 17:38)



Now if I take the projection of this now what is going to happen at the critical values because at this point we can see like there is a hole inside so this is my critical point now. So, in the slice through the critical values you will get one here if you can imagine in one here like this. So, when I cut using the plane here what would happen is here I am going to get one curve on this spontaneous, but at that critical points there will be two curves now one this side and one this side.

If I take the projection on my XY plane what I will be getting is these two curves are getting projected here on this line that is what we have got now.

(Refer Slide Time: 18:26)

Accessibility and Departability

In order to access and depart the roadmap we treat the slices which contain q_s and q_g as critical slices and run the algorithm the same way.

So we were able to get the projection of the ellipse on this curve now. Now suppose I want to go from some point to some other point now because now you can guess I am sure that what

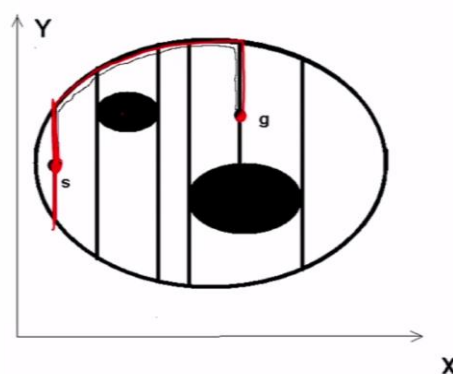
is the use of doing this. Now, suppose I want to go from this point to here or rather if I am here and I want to go here this is my initial point and this is my goal point now how do we go?

So, if this is my Silhouette curve the side boundary which I am getting these points I have got right. So, from here I can go like this I can go like this I can go like this and I can go there. So, this Silhouette curve is actually giving me the path which I can follow. This is the time for road map now. So, I am going to start from initial point go like this follow the curve here now that is a straight line which is having a critical point then I cross over here I go like this and I go there.

Now, if you simply look at the ellipse without doing this it will be difficult to figure out how do you go. So, what did I do I went from somewhere here I was here I went somewhere here now without doing that without doing this will be difficult to know how would we move the surface of the ellipse, but once you do this use the Silhouette method it becomes very clear that how you are going to go from one point to another point on this torus.

(Refer Slide Time: 19:57)

Accessibility and Departability



These are some more examples accessibility and departability some examples here. So, suppose you have an ellipsoid again like this and this has an obstacle. So, what we do is basically I am going to make my hyperplane I am going to pass my hyperplane and this hyperplane is going to touch it there and this is my critical point now. So, the first thing we do is slice it and get our hyper critical points.

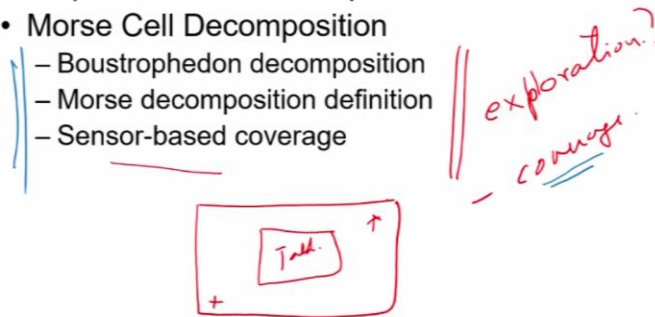
Now I have got my critical points here, here, here and here and then what I do is suppose I want to go from S which is my start point to my goal point. I can draw another slice through here and this is a path this is giving me a path here this is my path and I can go to my goal point here avoiding this and this. Now, I hope you understand what how this method works that by simply slicing this what we are doing is we are getting a series of connected paths which is taking from the start point to the goal point.

And in this method is called the Silhouette method. Now, so much for road maps. Road maps are basically generating network of pathways.

(Refer Slide Time: 21:19)

Types of Cell Decompositions

- Polygonal Cell Decomposition
- Trapezoidal cell Decomposition
- Morse Cell Decomposition
 - Boustrophedon decomposition
 - Morse decomposition definition
 - Sensor-based coverage

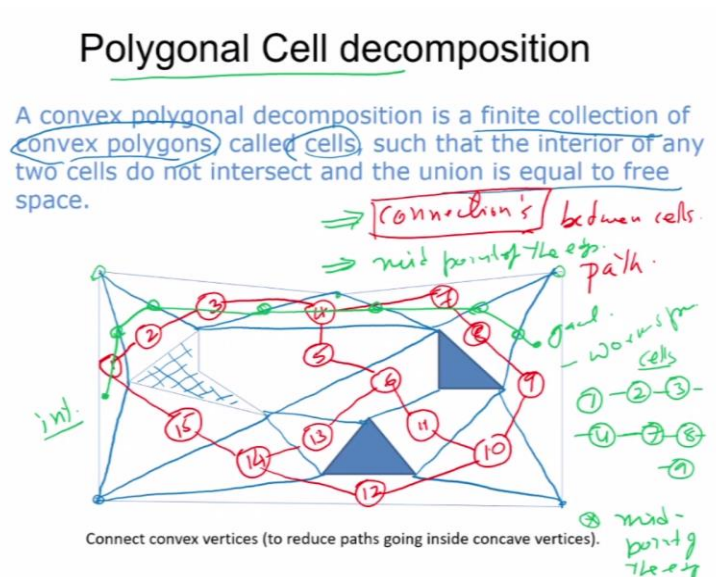


Now the next method is basically called cell decomposition method. Now as the name indicates the cell decomposition method basically decomposes the cell the work space into a number of cells and there are different methods which are used here number one is polygonal cell decomposition, second is trapezoidal cell decomposition then we have the Morse cell decomposition then we have this Boustrophedon cell decomposition.

Then we have Morse decomposition and sensor based coverage. These are more or less useful exploration. So, exploration is one coverage is the other. So, the point is not only that suppose I have a work cell and I have an obstacle here robot has to go from here to here that is just one problem now it could be such that the robot actually has to explore everywhere and not simply go from one point to another point suppose there it is mapping.

The other is coverage suppose there is a vacuum cleaner which is cleaning this room. Suppose this is a room and this is a table then it has to cover all the points that means it should not leave any point un-cleaned. So, that is where basically this coverage algorithms come inside and these three are basically coverage algorithms how the robot can cover all points not only our obstacles.

(Refer Slide Time: 22:43)



So, let us start up with a first one which is polygonal cell decomposition. Now as the name indicates the polygonal cell decomposition basically works like this. So, a convex polygonal cell decomposition is a finite collection of convex polygons called cells such that the interior of any two cells do not intersect and the union is equal to the free space. Now what this means is that this is my work space and there are vertices and these are my obstacles in here.

So, this one is also green obstacles this one is also green obstacles. Now convex polygonal decomposes this total work space into a number of cells by a collection of finite collection of polygons such that the interior of any two cells do not intersect and the union is equal to free space. So, let us start up by how this proceeds is let us connect all these vertices, put the vertices of the object so this is one here, there is one there and one here and from there is one here there is one here.

Then this vertices is this vertex is coming here this one is coming here I am just connecting all the convex vertices please note that they are convex polygons. So, the other connection we are making is here, here then we are making a connection here, here. So, these are all

connections I am connecting the vertices I have made by connections. So, I have divided the whole work space into a connection of cells by connecting the nodes and the vertices.

Now what we can do is we can number these cells let us number these cells this is 1, this is 2, this is 3, this is 4, this is 5 then this is 6, this one is 7 that is 8, this is 9 then let us say this is 10 then 11, 12, 13, 14 and 15 I will connected all the cells. Now, if I want to make a connection diagram now what is the objective? The objective is to take the robot from some initial point to some goal point.

So, what I can do is I can make a connection diagram 1 is connected to 2, 2 is connected to 3, 3 is connected to 4, 4 is connected to 7, 4 is also connected to 5, okay 7 is connected to 8, 8 is connected to 9, 9 is connected to 10, now 5 is also connected to 6, 6 is connected to 11, 11 is connected to 10. Now, 1 is also connected to 15 and 14. So, we have got there is a connection here also one here.

So, these are all the connections in between the cells. So, I have put numbers into all the cells and I have connected the cells. Now I know exactly what the connection diagram would like in fact the red line is showing you the connection diagram. So, if you want to go from cell number 1 to cell number 11 how would you go? So, you can choose two ways you are going to 1, 2, 3, 4, 5, 6 and 11 or you will go for 1, 15, 14, 13, 6 and 11 there is a third path also 1, 15, 14, 12, 10, 11.

So, this is basically giving us I have divided the total work space in to convex polygons called cells and I can find the connections between cells now. So, I have done my connections and this is my connections between the cells. Now once we have got the connections between the cells. Now suppose I want to find the path so these are the connections red line. Now I want to find a path to take me from one point to another point.

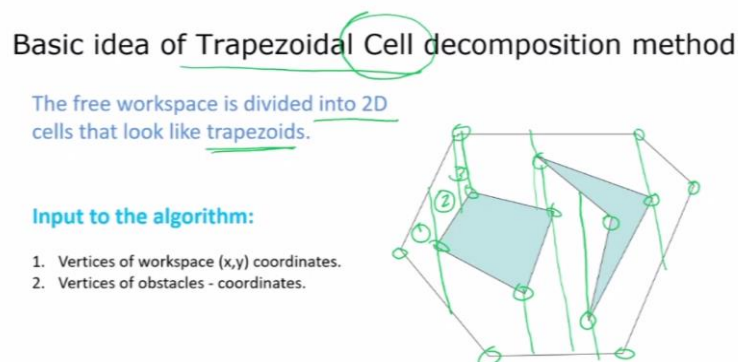
So, how do I do that is very simple and very interesting in the sense that if you look at this now. Now which will be the path? Now what I do is suppose I want to go from my initial point here this is my initial point and this is my goal point let us say goal point is right here that is my goal. So, from initial to goal I see that I can go from 1, 2 then 3 then 4 and then 7, 4, 7, 8 and then it is 9 that is the path.

This is the different cells that are there to take me from the start point to the goal point, but now what I can do is I can very cleverly take the mid points of the path, midpoint of the sides or the edges. So, I take the midpoint here, midpoint here, midpoint here, midpoint here, midpoint point here, midpoint here and I simply connect it this one, this one, this one, this one, this one, this one I have got my coming to my goal.

So, first what I do is I make my connections then I take the midpoints of the edges and connect by the straight lines and that is how we get our I have to go from one point to another point. This is called polygonal cell decomposition because we are dividing the total work space into a number of convex polygons and then I find number the different cells and the connection between the cells.

And to find a path I simply take the midpoint of the edges which are making up the polygon and I simply connect it and I get my point. Now computer can do this very, very quickly and this method is called the polygonal cell decomposition.

(Refer Slide Time: 29:27)



Now then at the other method in cell decomposition trapezoidal cell decomposition. In the case of trapezoidal cell decomposition as the name indicates and you would be guessing by now. So, we are going to divide it into a number of cells. So, the free workspace is divided into 2D cells that look like trapezoids. So, the input to the algorithm is the vertices of work space coordinates and vertices of obstacle coordinates.

So, this is my coordinates of the vertices of the work space and these are coordinates of the vertices of the object. So, this is the input to the algorithm. Now what the algorithm basically does is it has to divide this into a number of cells. In the previous case this polygonal cells here it is trapezoidal cell how it does that is it takes every vertex and draw the straight line. So, from this vertex it is going to come down here.

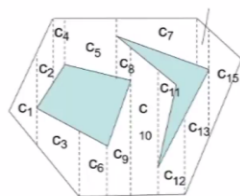
And this vertex is going to go up, this vertex will come down, this vertex will come here and from there this vertex will go there. So, I hope we can see what is going on. So, what is going on is that we are dividing this into a number of cells. Once I have divided this into a number of cells then what I am going to do is I am going to number them just like before so 1, 2, 3 in that order.

And then find the connection between the cells and once we have found the connection between the cells how do you find the path? You will simply see the connection of the cells to form the path and take the midpoints of those straight lines and then you have your path.

(Refer Slide Time: 31:09)

Adjacency Graph

- Node correspond to a cell
- Edge connects nodes of adjacent cells



So, let us see how this is done. So, how this is done is in this particular case we have seen that.

(Refer Slide Time: 31:20)

Path Planning

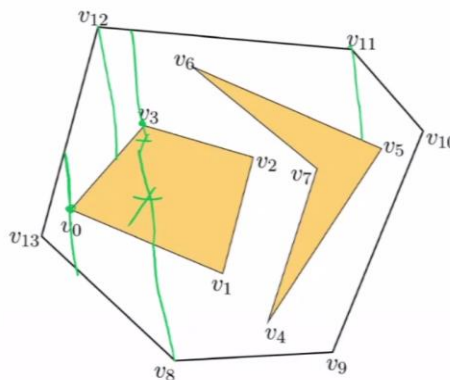
- Path Planning in two steps:
 - Planner determines cells that contain the start and goal
 - Planner searches for a path within adjacency graph

So, the path planning is done in two steps. The path planner determines cells that connect the start on the goal then the planner searches for a path within the adjacency graph.

(Refer Slide Time: 31:29)

Trapezoidal Decomposition

- At each vertex draw two line segments one going up and the other going down.
- Lines do not go through obstacles.



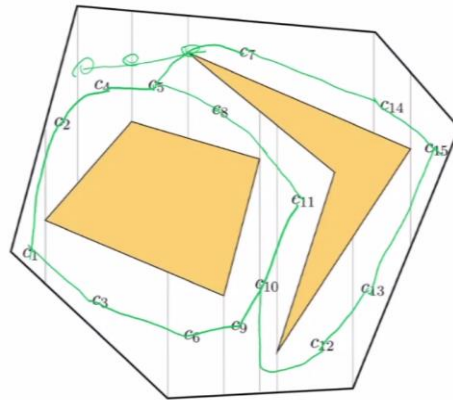
Let us see how it is done. So, at each of these vertices what we are going to do is we are going to draw a straight lines. Now at each vertex draw a line segments one going up and the other going down, lines do not go through obstacles. I will just explain. So, what we are going to do at a vertex is I am going to draw a vertex like this down and from here I am going to draw it up.

It will not go through obstacles should not go through obstacles. Now wherever there is an edge like this in this particular case it is going to go up and down at every vertex. So, in this vertex we are going to draw line up, but not down because we are going through an obstacle.

In this vertex we are going to draw line down, but we does not go through an obstacles. So, this is what we are going to do and divide this into a number of cells.

(Refer Slide Time: 32:19)

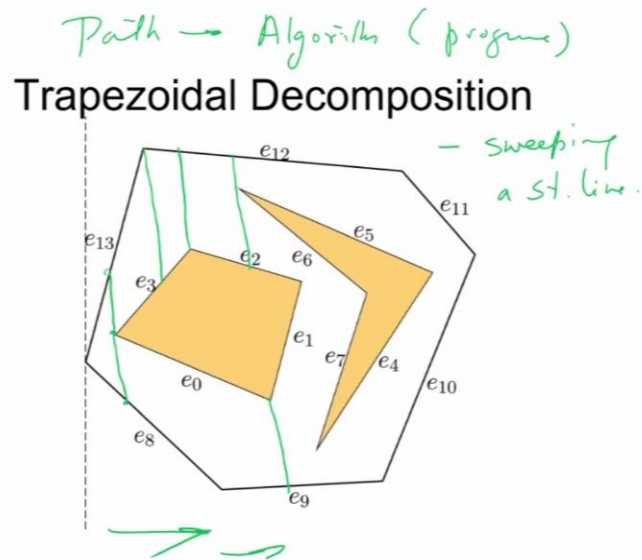
Trapezoidal Decomposition



Now once we have divided this into a number of cells I am going to number the cells 1, 2, 4, 5 in this way. So, each cell has been given a number. Now we know the connection between the numbers so for example C is connected 2, 3, 4, 5. 5 is connected to 8 this is connected to 10, 10 is connected to 11 this is another one and the one going on top this is connected that is connected, that is connected this is connected, this is connected and this one is also connected.

So, now I know which cells are connected to which cells and now if I want to find a path what I do I simply take the midpoints of those lines and I simply connect it thus the path which is going to be the equidistant from the obstacles in the work space and that is going to give me a connected graph now.

(Refer Slide Time: 33:10)



Now how do we run the algorithm? Now in path planning or motion planning we are worrying about algorithm. Now algorithms basically means a program, now this program has to automatically decide now we have eyes so we can see I have said this many times I guess, but we have eyes so it is very easy for us to see okay suppose you want to go from here to here there is a path like this, like this, but the poor robot does not have eyes and the program does not have eyes of course so how would it basically divide the cells?

So, what we do is we start off with a straight line. So, we start off with a straight line and sweep the straight line so this is basically sweeping some time back we also looked at the sweep line algorithm in the case of Voronoi diagram. Now here also we have a straight line like this which you are sweeping. Now as we are moving whenever the straight line hits the vertex geometrically we know the equation of straight line.

We can find out if the vertex is lying on the straight line. So, the moment it hits the straight line we know that is a vertex and it is a line which is going up, which is going down. So, I move the line little bit more on the right until so I am sweeping this line in this direction. The moment you hits that I know he is hitting a vertex and then I take it up and I take it down now I know it is hitting here and it is hitting there.

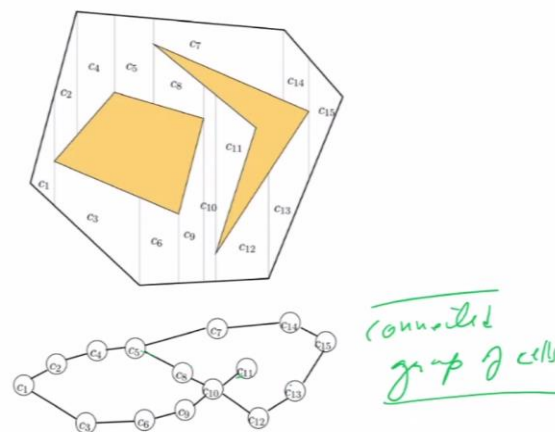
So that is my cell I will repeat it. So, there is a straight line which is being swept in this direction. So, here when it is hitting the vertex I know this line is going up and this line is coming down this is the straight line. So, I get this point, I get this point, I get this point. Now

I sweep it more this side so what happens it hits I get this one. Now it should not go through an obstacle.

So, it does not go through an obstacle it just ends there then I get this one then at the sweep line gets this one on top it gets that one.

(Refer Slide Time: 35:17)

Trapezoidal Decomposition

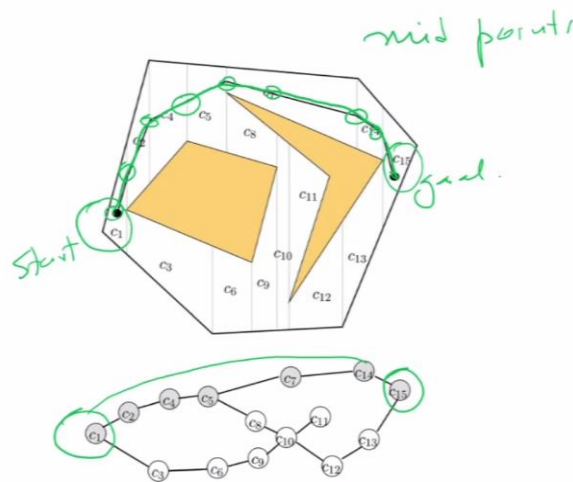


So, in this way what it is doing is it is sweeping through the space so it is sweeping through the space and we are getting this connected graph. So, then what we do is we draw this connected graph of cells. So, this is 1, 2, 3, 4 we can see that this is 1, 2, 3, 4 then 7, 14 this in one branch 7, 14, 15.

Now from 5, 8, 10, 11 is one branch and the other branch is coming this side C 3, 6, 9, 10, 11 and from this side it is coming to 13, 14, 12, 10. So, what we are seeing is we are getting the connection between the different cells the connected graph of cells. Now, suppose I want to go from some point to some other points. What we need to do is I simply need to join the midpoints as usual from the connected graph. Suppose I want to go from the one point to another point.

(Refer Slide Time: 36:24)

Trapezoidal Decomposition Path



Let us have a look at this I want to go from the start point to the goal point. What I will do is I will look at c 1 the start point is in c 1 so it is in c 1. The goal point is in c 15. Now I know which is the path so this is the path in terms of the cells. Now how do I get the path actually in terms of points what I do I take the midpoints of these lines. So, I will take midpoint here, midpoint of these straight lines and I simply connect it like this and that is going to give me the path which we will take it from the start point to the goal point.

(Refer Slide Time: 37:05)

Sweep line method: what is the ultimate goal?

Program / Algorithm.

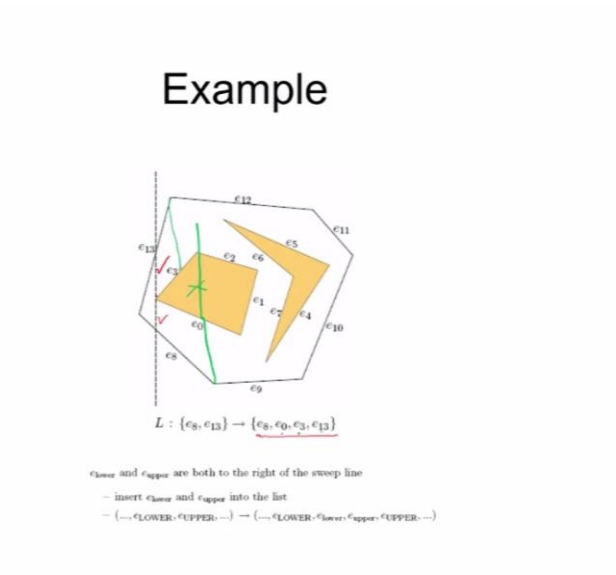
Find the mid point of vertical line segments.

- Sweep a line through the free space stopping at vertices.
- Maintain a list L of all the current edges the slice intersects.
- Update the main list depending on the way the upper and lower lines intersect.
- Depending on whether the vertical lines are above, below or through an obstacle, decide the mid point of the segments.

Now the algorithm is basically using the sweep line method and the ultimate goal of the sweep line method is to divide into trapezoidal cells and then once we get the trapezoidal cells the length of the straight line then we can find the midpoint. Now the sweep line it sweeps through the free space stopping at the vertices so we sweep the straight line this is my straight line so I start off here I sweep here until it hit the vertex.

The moment I hit the vertex it goes up or down depending on whether it is going through an obstacle or not going through an obstacle. So, if it is going through an obstacle it does not go anymore it stops there then next is as we are going we have to maintain the list of all current edges and slices intersects and we keep updating that main list depending on the way the upper lines are intersecting depending whether the vertical lines are above, below or through the obstacles decide on the midpoint of the segments.

(Refer Slide Time: 38:05)



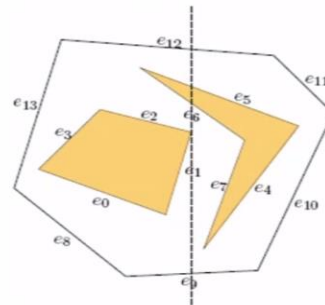
This is basically the working at the algorithm. So, basically what it is doing this is a straight line. So, it is starting so what it is doing is we have got the straight line hitting the vertex consisting of e 8 and e 13. So, e 8 is here and e 13 is here. So, what we have done in terms of the algorithm is that we are noted the vertices plus the edges which are marked then we are going to maintain this set it starts sweeping and here it is e 8 and e 13 so e 8 and e 13 is the first vertex here.

Then next it goes a little bit more what happens if it hits this one. So, it hits e 8, e 0, e 3 and e 13. So, it sweeps a little bit more you can see it is hitting now e 8, e 0, e 3 and e 13. Now when it is hitting the vertex we also check if it is going through an edge if it is going through e 3 like this then we describe the upper part. Similarly from the top when it is coming down hitting an edge then is it going through the edge that we can check very easily.

And then we describe that part we just stop it here. So, we keep updating this list as the line is sweeping and we then came up with the total list of vertices and the length of the lines which are doing that.

(Refer Slide Time: 39:30)

Find mid points of vertical line segments.



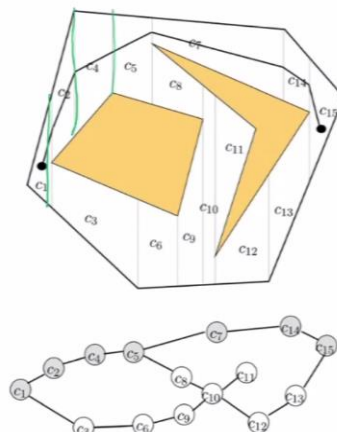
$\{e_9, e_1, e_2, e_6, e_5, e_{12}\} \rightarrow \{e_9, e_6, e_5, e_{12}\}$

delete e_{lower} and e_{upper} from the list
 $(\dots, e_{LOWER}, e_{lower}, e_{upper}, e_{UPPER}, \dots)$
 $(\dots, e_{LOWER}, e_{UPPER}, \dots)$

So, after doing that what we get is this is the final list that will come out will only consist of the lines the length of the lines which are being drawn.

(Refer Slide Time: 39:40)

Examine the adjacency graph to get possible paths

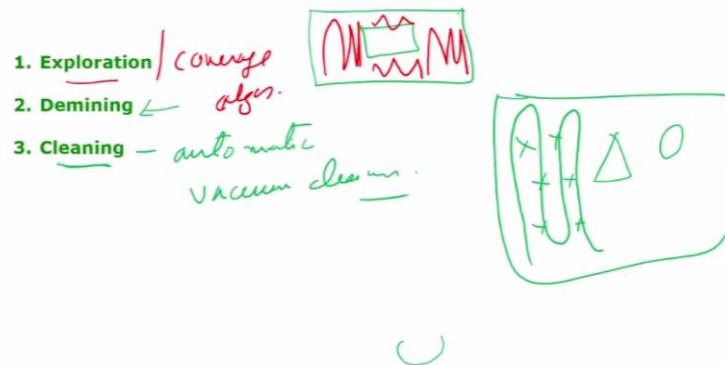


So, for this one we will have the full one. This we will have only from here to here this we will have from here to here that is how we are maintaining our list.

(Refer Slide Time: 39:47)

Path planning is just one application:

Other requirements are for covering or exploring the full free workspace.



Now this is basically to show the how to find the path from an initial point to a final point by a connected set of roads and this method is called the cell decomposition method because we are breaking it up into cells. Now path planning is not just about finding a path if for example as I said that we have work space and an obstacle space and I want to go from this point to this point.

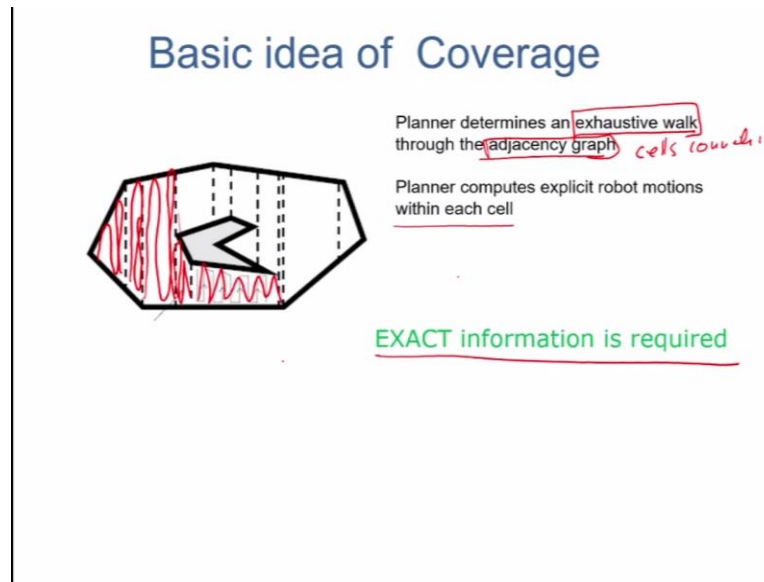
So, one of the applications is just to find a path avoiding obstacles, but there are other requirements for covering or exploring the full work space. Exploration means suppose the robot does not know that there is an obstacle here, where is the obstacle? It has to explore this area and find out where are the obstacles, where are the paths. Other applications are places like demining.

These are different applications where mines are laid in particular ways and the robot has to actually find out these mines. So, this robot in order to do that the; robot has to cover all the space, it cannot leave any space. The third is very common these days are cleaning so automatic vacuum cleaners, for example, so automatic vacuum cleaner. So, automatic vacuum cleaner is supposed to vacuum clean.

Let us say I have a room like this and I have a table and the vacuum cleaner is supposed to clean everywhere. So, the question that is being asked here is it is not a question of going from one point to another point it is a question that I have to cover now how do I cover, do I go like this, do I go there, do I go here and do I come back like this is that a better way by covering everything and not leaving any path or should I take some other path.

Should I do this part first then I do this part then I do this part then I do this part, how does it guarantee that I have covered everything and I have not missed out any part that is the meaning of what we call exploration and coverage algorithms.

(Refer Slide Time: 42:00)



So, the basic idea of coverage is that the planner determines an exhaustive walk through adjacency graph. Now this adjacency graphs are basically cells which are connected so connected cells. Now the planner computes explicit robot motions within each cell and in this particular case exact information required. For example we have a work space this is my work space and this is my obstacle.

I want to clean the full area or I want to cover all the area or I want to perform what is called an exhaustive walk that means we want to do everywhere in the cell. So, what we can do is we can break it up into cells or adjacency graphs so I have broken it up into cells like in the previous case trapezoidal cell decomposition and then I start cleaning each of the cells one at a time that will ensure that I have covered all the cells and there is no cells which is left. Now how do we do that?

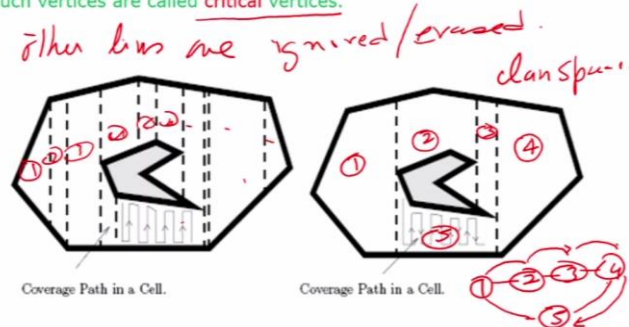
(Refer Slide Time: 43:02)

Boustrophedon Decomposition

Disadvantage of trapezoidal decomposition: many small cells can be formed

Boustrophedon / Moore decomposition: Only Vertical lines that can be extended up and down at vertices.

Such vertices are called critical vertices.



So, this is called Boustrophedon decomposition. The disadvantage of this trapezoidal decomposition is that there are too many small cells that are formed. So, whenever we looked at this trapezoidal cell decomposition we saw that let us just go back to this figure. So, we can see that there are so many cells which are being formed here. Now, if you start cleaning one cell here, one cell here there are too many cells and it gets in a way it gets a little confusing.

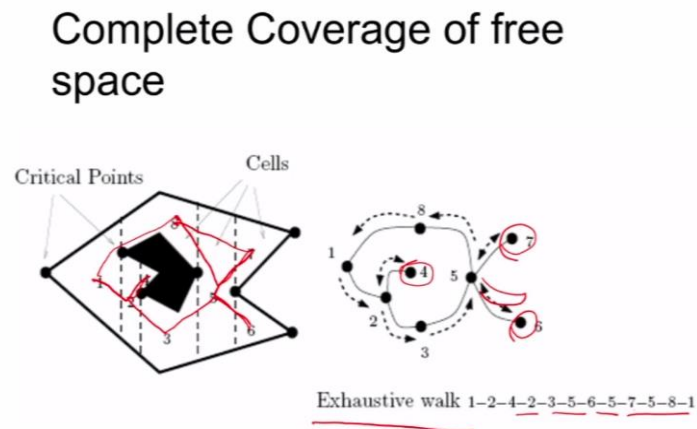
So, I just want to simplify it what was done here basically is that only vertical lines that can be extended upward down vertices are considered. Such vertical lines are called critical vertices and the rest of the other lines are ignored or erased. Now the lines which can only go up and down from a vertex is kept the other lines are all deleted. So, from this vertex you can see that this line can go right up and right down there is no vertex here.

So, this two can go which is not required. Similarly, this one can go up it cannot go down so this is also gone. Now this one the line can go up and the line can go down here the line can go up and the line can go down so this is deleted, this is deleted so what to get is this now. This is a much more clean space this is a better representation. Now if I want to do path planning I want to do the coverage algorithm basically.

So what I can do is I need to find the connection between the cells so I have found the connection let me go back to the previous one. I can find the connection between the cells let us say 1, 2, 3, 4 and then there is one more which is 5. So, there are actually 5 cells whereas if I had done the previous case that is this case how many cells would there be? So, there will be 1, 2, 3, 4, 5, 6 like that.

So here actually there is only 5 cells so I have a connection only between 1, 2, 3, 4 and 1 comes here 5 that is enough. So, if I want to now move the robot to clean itself I can go from 1 to 2, 2 to 3, 3 to 4, 4 to 5. So, I hope you understand how this is useful for a coverage algorithm whether robot has actually covered all the area which has not left anything.

(Refer Slide Time: 45:42)



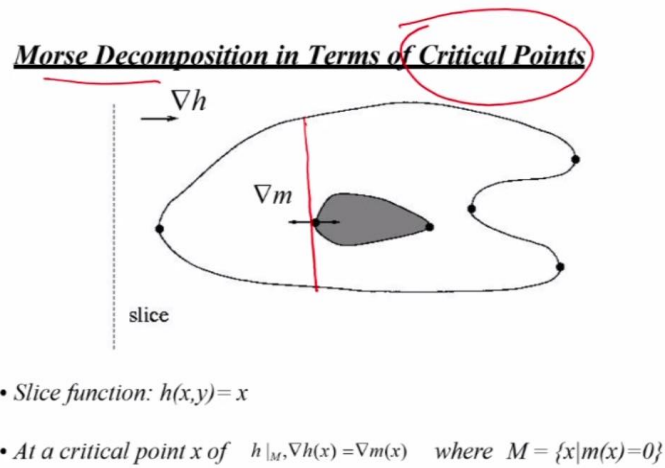
Now this is another example where we have divided it into number of cells and the exhaustive walk is following this cell 1, 2, 4, 2, 3 that is 1, 2, 4 so we are doing from 1 to 2, 1 to 4 then it is going from 4 it is going to 2 back and then comes 3 then 5, 6 it is coming 5, 6 then it goes back to 5 again then it goes to 7 then it goes to 8 sorry 7 to 5 then to 8 and then back to 1. As you can see that there could be more than one way of doing this.

So, which is the most efficient and the optimal way that is an interesting question that in this particular case suppose we are going like this or we are going like this which will be faster. In this particular case is a simple case there is only one loop whereas in this case there are two loops there are loops in here so which would be and there is a dead end here. So, if you go into 6 you have to come back this way.

So, this is an interesting question of this coverage algorithm. Now something that you would have noticed here is that when I said that I have to cover this which is okay, but then how do I cover it what can the motion do I have in one cell let me consider this cell will the robot move like this it has to cover every point remember or which start going round like that

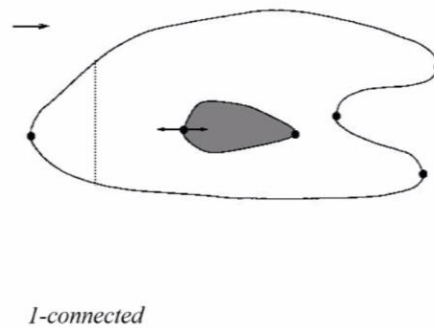
which is a better way of covering it that is an interesting question. Well there are different ways by which you can do.

(Refer Slide Time: 47:20)



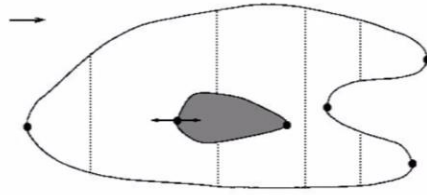
For example, Morse decomposition is one way of dividing the cells.

(Refer Slide Time: 47:26)



So, you have this number of connected cells so what I do is basically we look at the critical points that is the point at which the line is going to hit or change direction and then I divide it into number of cells. So, I have changed this into a number of cells.

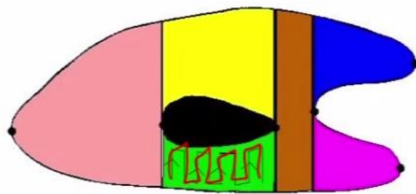
(Refer Slide Time: 47:50)



• *Connectivity of the slice in the free space changes at the critical points*

So, connectivity of the slice in the free space changes at the critical point that we know.

(Refer Slide Time: 47:54)



• *Each cell can be covered by back and forth motions*

And what we have done is these are my cells. Now with each cell can be covered by back and forth motions. So it simply does this and covers that out. Now let us see this further.

(Refer Slide Time: 48:12)

Incremental construction

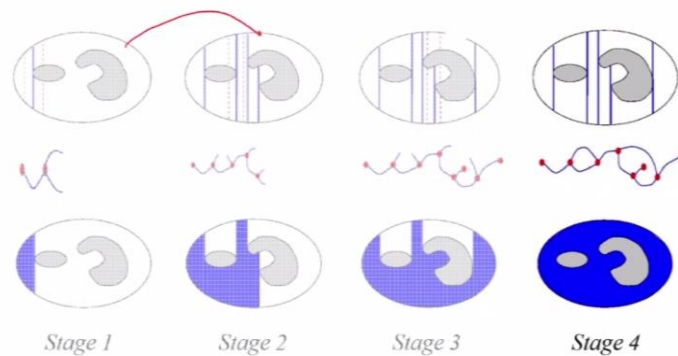
•While covering the space, look for critical points



So incremental construction how do we construct it? So, basically when we are constructing it we are starting with a straight line and the straight line is sweeping from the left side as you can see.

(Refer Slide Time: 48:24)

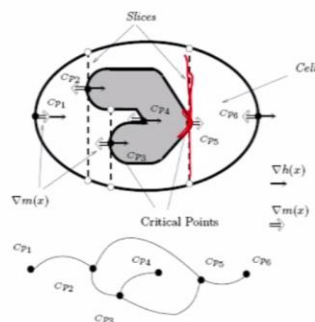
Incremental construction (cont'd)



And every time it gets a critical point it forms the next cell finally we have from here we go here the line is sweeping towards the right here, here and we have covered all the cells.

(Refer Slide Time: 48:39)

Detect Critical Points



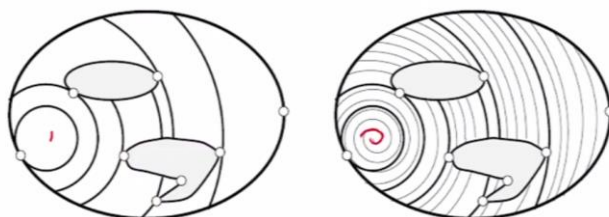
Now how do we detect the critical points? So, you can detect the critical points like in the previous case wherever it is hitting a vertex or there is a change in gradient of the surface profile there then is the line going up and also down that point will be a critical point here. Now most decomposition in the previous case was being done by using a straight line. So, what is the difference between what we have just done in the cell decomposition between here and the Morse way of doing things.

So, what is there see that this is a complete cover here this is what I was talking about. So, here what we are doing is basically we are dividing it by straight lines and then we are going having motion like this, but in the Morse case we can divide it by not by straight lines only straight line, but we can divide it by other kind of curves also.

(Refer Slide Time: 49:43)

Morse Decomposition

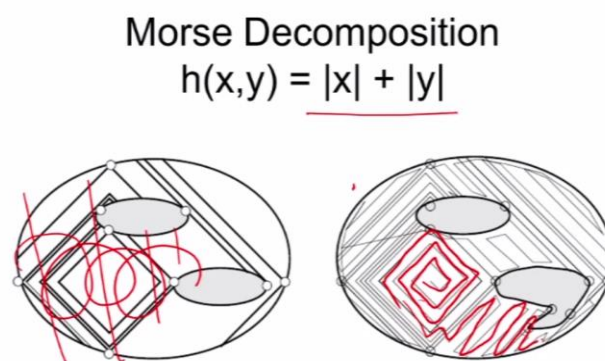
$$h(x,y) = x^2 + y^2$$



Now let us see what happens if you start using this kind of a Morse decomposition where you are dividing this by using this arcs. So, we start from here and do a circles $x^2 + y^2$. It is like the circle is increasing so we have a circle here. Now circle increases, has Morse point there then the circle increases further it has one more critical point there, increases further has a critical point there.

So, in the previous case it was straight lines which were moving now it is circles which are moving. So, circle is simply increasing and it is hitting at this point, this point, this point these are all the critical points now and now we have got different cells like before 1, 2, 3, 4, 5, 6, 7 and I can do the same thing to do the coverage algorithm. So, because they are circular my motion is circular now the robot is cleaning like this then it goes like this.

(Refer Slide Time: 50:39)



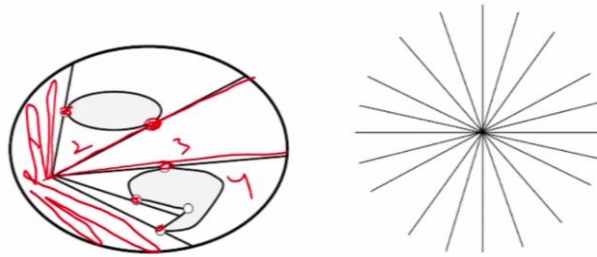
Now I can do it in some other way also say for example we can do $|x| + |y|$ in this particular case I am dividing my cell into this rectangular fashion and the robot is moving like this now if we can see. So, it comes here does this then goes there so these are all ways by which you can do efficient planning. So, it is interesting question whether using the straight line would have been faster or the circle would have been faster.

It depends on the geometry it is difficult to say. So, you actually have to see which one will give the better.

(Refer Slide Time: 51:15)

Morse Decomposition

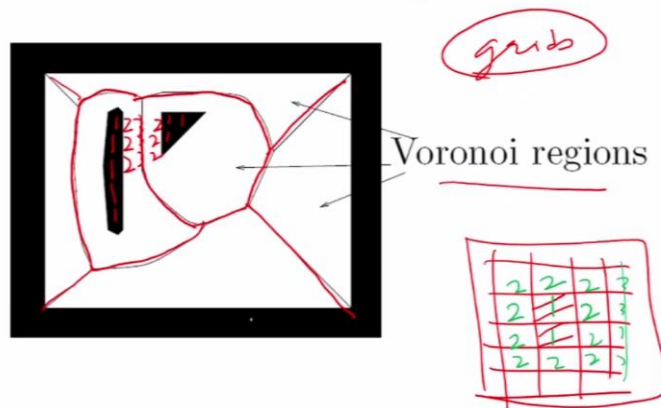
$$h(x,y) = \tan(y/x)$$



Now, you can also use a tangent $\tan\left(\frac{y}{x}\right)$ so from this points straight lines are going of tangent to the curves here and these are my different cells so this is 1, 2, 3, 4 like that and my coverage algorithm also is going to take it is going to move this way so it will move like this. So, these are different ways of dividing the cell they are all cell decomposition different ways.

(Refer Slide Time: 51:43)

Brushfire Decomposition

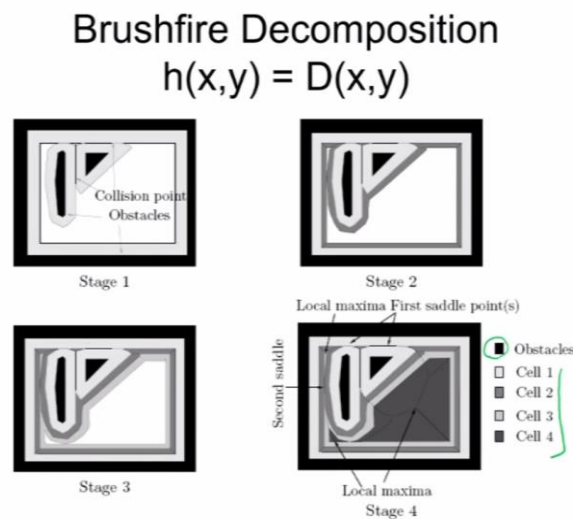


Bush fire algorithm is another way now with Bush fire algorithm is very interesting because we are dealing with grids. Now in grids you know that in a computer if you can divide this in your work space into grids and assign numbers then the computer works very, very fast it can do it very easily as compared to putting concave and concave objects finding the vertices, finding the intersection whereas in this case what we do in the Bush fire algorithm is

basically we assign this as 1 and the next grid will be assigned 2 then it will be assigned 3 in that way so this is 1, 1, 1 this is 2, 2, 2 this is my 3.

So, basically what I am going to get I am going to get this curves. So, the objective is to be able to break the free space into a number of cells and these are also called free cells or Voronoi regions using Bush fire algorithm. Bush fire algorithm I just explained in the beginning of the class today that if there is an obstacle here like this I name this as 1, the second one will become 2, the third one will become 3 in that order and wherever the highest number meet that is your region.

(Refer Slide Time: 53:00)



Now from here what we can see is that we have divided the full space into a number of points and so this is my first point second stage, third stage and fourth stage. So, how many cells are there? So, we can see that these are 1, 2, 3 and 4 cells and the black one is the obstacles. So, I have basically got using cell decomposition I have got 4 cells now I can clean each of the cells one at a time.

So, the robot has the coverage algorithm basically starts covering performing an exhaustive walk on each of the cells one at a time. So, today we looked at we started off with a road map method we just completed the Silhouette method that is pass the hyperplane give the Silhouette curve and see how to complete the road map and take the robot from some point to some other point.

Then after that we looked at cell decomposition. Cell decomposition is we looked at polygonal cell decomposition then we looked at trapezoidal cell decomposition then we look at the coverage algorithms. So, these are all algorithms which are used for the robot to go from one point to another point or for coverage. Now what you would notice here is that this is basically a geometrical problem.

Then what is the input to the algorithm or the program is the geometry of the work space that means the coordinates of the vertices of the work space and the vertices of the obstacles. Now so this gives you some idea that when you write a program for either a simulation or in a case of real robot which is going to go from some point to some other point how the program is written and what is the input of the program and what is the output of the program and how it works.

So, today we will stop here and next class we will move on to the next set of methods. Thank you.