

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

Lecture – 20
Optimization in Motion Planning

Hello and welcome to lecture number 20. So, this is the last lecture in the course robot motion planning. So, we started this course by looking at the very introduction of robot motion planning, why motion planning is required, where it is used for serial arms and mobile robots. Then we started off with bug algorithms bug 1, bug 2. Then we looked at configuration space that is we saw that when the robot is a point robot.

And the robot can move in any direction, we do the planning and C-space or configuration space and from that we moved on to the various roadmap methods. So, in roadmap methods, we looked at the Voronoi diagram, we looked at visibility graph methods. Then from there we moved on to cell decomposition and then potential free methods, right. Now, we also saw that normally when you look for a path or rather when you generate a network of paths, you get a large number of paths to go from the start point to the end point.

For example, in sampling based methods you can get very large number of paths which will take you from the start point to the goal point if of course there are large number of paths. So, in cases where there were very large number of paths, we have to find which path is the optimal path terms of path lengths or in terms of energy consumed. So, what you notice or what you understand by now is that some kind of optimization is required.

So, today's class basically we will look at this concept of optimization that why is optimization required in robot motion planning and when we are trying to optimize how do we particular try and get a particular solution. So, today's class we will look at multi agents where we have multiple mobile robots which are for example capturing our object and pushing it somewhere, we can look at multifinger robot hands.

It is almost like the multi fingers of the robot hand or like multiple serial robot arms which are manipulating an object. Then from there, we will move on to bipedal locomotion. Again, we will see that when we are using our path planning methods, then what happens is we end

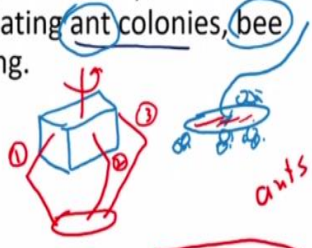
up with a large number of solutions and some kind of optimization is required. So, today's lecture is mainly to do with optimization motion planning and with this lecture we will be ending the course on robot motion planning. So, let us start off with the last lecture. So optimization in motion planning that is the topic of our last lecture that is lecture number 20.

(Refer Slide Time: 02:30)

Multi agent systems – mobile robots, manipulators, multi finger grasping

multiple solutions

- The idea of Multi agents (swarm robots) has been borrowed from nature, emulating ant colonies, bee hives, etc. for motion planning.
- Multifinger manipulation.
- The basic idea is to be able to perform a cooperative task that cannot be performed by one individual robot.



Now, when you have multiple agents doing a particular task for the multi agents like mobile robots, manipulators, multifinger grasping, they all can be looked upon as multiple robots doing a particular task for example moving an object, constraining an object. now the idea of multi agents has been borrowed from nature because you must have seen this that in ant colonies or bees basically it is like multiple bees are together performing a particular task or if you look at an object say for example this is a piece of food.

And there are ants who are carrying this object. So these are ants, so you can see multiple ants who are actually carrying this last object. So, they are cooperating with each other through a particular task. For example, there are many ants and these ants are lifting this object and taking it in some trajectory or taking in that direction. Now, that can be a case of multi agent system where you have multiple agents doing a particular task.

And hence they can be ants, many ants doing a task, they can be bees, multiple bees are flying in formation, collecting honey and doing a particular task. Now, so this idea of multi agent or what we also call swarm robotics is basically borrowed from nature. Another example is multifinger manipulation. So, we have an object like this. So, let us say I have a cube. This is a cube and the cube is held by fingers, multiple fingers.

We have multifinger hand which is holding this cube and I want to move this object in a particular orientation. So, I basically want to manipulate this object, so I am going to rotate or translate this object in a particular way. So, it is like there are multiple robots, each of these fingers can be looked upon as a robot, a serial arm robot, so this is 2 and that is 3. So these three robots are together performing a task to manipulate this object in a particular desired way.

So, the basic idea here is to be able to perform a cooperative task that cannot be performed by one individual robot. So, in the case of the bees or the ants, so in the case of the ants that we are seeing here, now one ant cannot lift this object and take it somewhere. So there are multiple ants which are actually catching it and taking it somewhere. So the very basic idea is cooperation when one individual robot cannot do this task, then multiple robots can actually come together and do this task.

In the case of multifinger manipulation also, we have the same concept. We have multiple fingers which are actually are applying a force of motion to the object and is manipulating the object in some way. Now, something that you note here is that there can be multiple solutions. So this basically means that if the ants want to catch this object and lift it and take it somewhere, there are multiple ways of doing this task.

Similarly, in multifinger manipulation, if you want to manipulate the object in a particular way, then there can be very many solutions to this problem. That means the fingertips can apply different forces or different motions to orient the object in a particular way.

(Refer Slide Time: 05:40)

Mobile robots as multi agents



Fig. Mobile robot as agent.

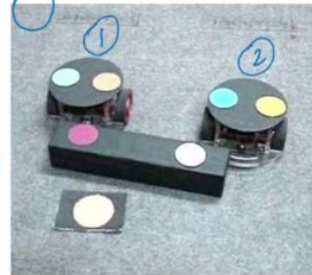


Fig. Two or more agents pushing an object.

Now, these are some examples of small experiments which are done using mobile robots as multi agents. So, this is a very small mobile robot. Now you might have made these kinds of robots in robotics club and things like that. So, inside this robot there are two DC motors, there is a DC motor there and a DC motor there. It is driven by a microcontroller and it can move in a particular direction. So this is basically what we call a differential drive robot. So, I have two wheels, one wheel here, one wheel here.

This has ω_1 that has ω_2 and there is a caster wheel in front. So, the ratio $\frac{\omega_1}{\omega_2}$ this ratio I control for the robot to go straight, to turn right or to turn left. And if you have one robot, for example I want this robot to pick up an object or push it somewhere, it may not be able to do it, it may not have enough torque or power to do that. So suppose as shown in this figure on the right side, so we have this black object and we want to push this object and take it to some location.

So, one robot may not be able to push, so I am using two robots, there is one robot here and another robot there. This is an example of multi robots doing a particular task. Now, this black object could be a moving object also. For example, I leave the black object itself, suppose it is moving having a particular trajectory and I have mobile robots which want to capture this object, first go and catch it and then maybe push it somewhere. So, these are examples of multi agent kinds of applications.

(Refer Slide Time: 07:00)

From mobile robots to Humanoids as agents

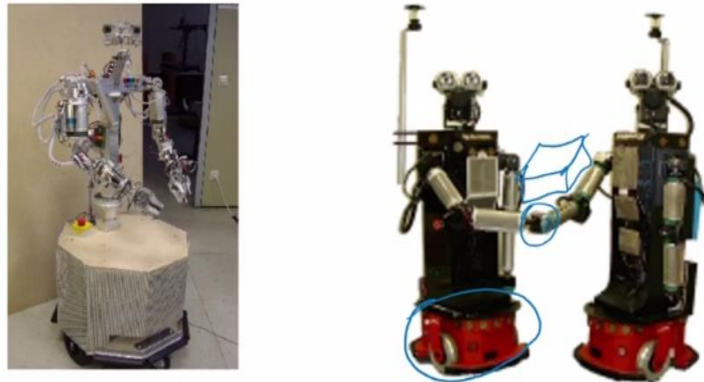
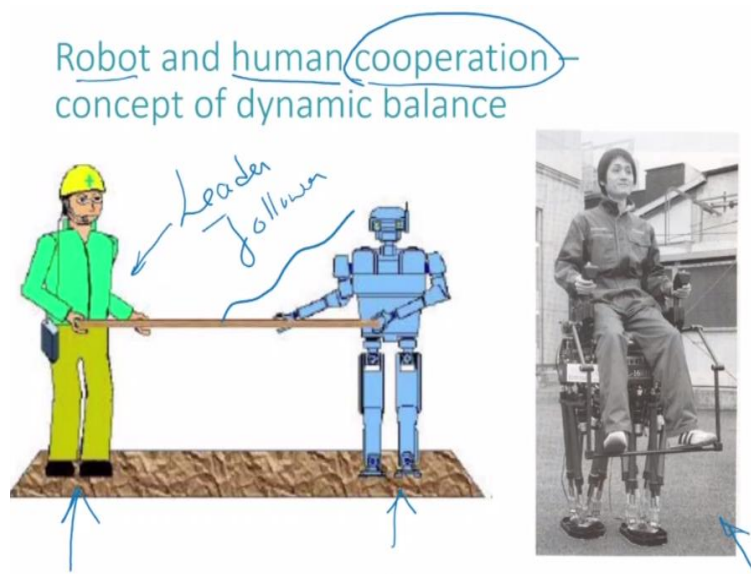


Fig. Upper torso Humanoid that does not require balance.

Now, multi agents can be extended to many, many applications. For example, two humanoid robots, one robot which is a wheel mounted a humanoid robot. So, this robot is is mounted on a mobile base. And there are two robots which are supposed to do a particular task. For example, both the robots you can see are shaking hands here or they could be carrying an object, for example this is a very heavy object which one robot cannot carry, then both the robots will come together and carry an object.

(Refer Slide Time: 07:31)

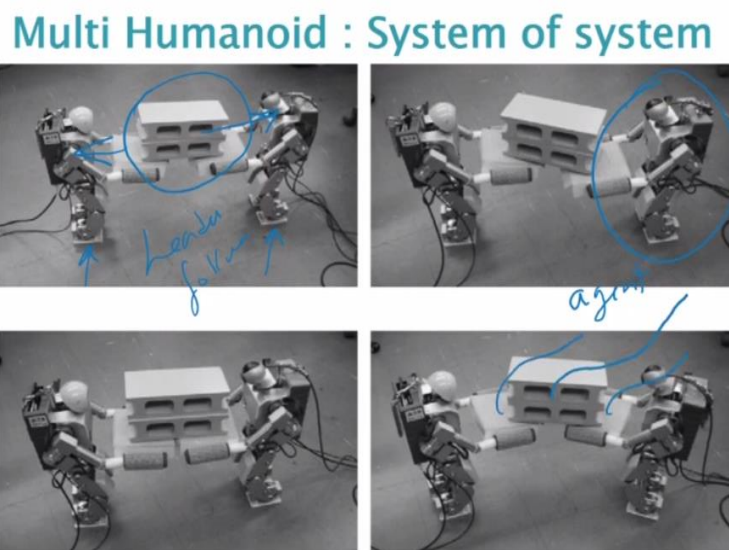


For example, here, you can also have a human who is cooperating with a robot. Say for example, here there is a human on one side, there is a human here and the robot on the other side is carrying this block of wood in some direction. Here, we come up with the concept of what we call leader, follower. So in this case, the human is the leader. The human pushes or

pulls the object in a particular direction and the robot just follows. So, the idea here is that the robot is cooperating with the human in doing a particular task.

Now, obviously, this has to be moved in a particular way. So, this is a motion planning and path planning. Now, this is another example shown on the right hand side where it is a human being who is actually sitting on this legged robot, again this legged robot is going in a particular direction. So, this is a robot who is cooperating with the human in carrying the human from one location to another location by following a particular path. So, this is a case of robot human cooperation. There can be cases of robot-robot cooperation also.

(Refer Slide Time: 08:30)



See, for example here we have a concept of robot and robot cooperation. There are two humanoid robots; one robot here, another over here and these two robots are carrying this very heavy brick and taking it in some direction. Again, the concept of leader follower has to come here. Why leader follower? Because suppose this robot pulls in this direction and that robot also pulls in that direction, then the task is going to fail, they will drop the object, right.

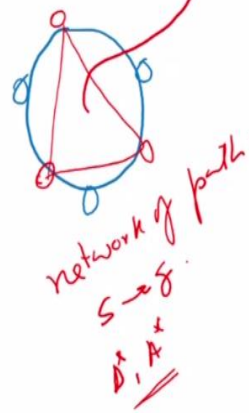
So there has to be a leader who is doing the task and there has to be a follower who is simply following. This is an example of two robots doing a particular task. Now, each of this can be called an agent. So each robot here can be called an agent. Now this is a bipedal robot agent. Here it is mobile robot agent, but essentially they are doing the same job. They are having some task plan and they are doing some particular task.

So, we can see that these agents can be of different kinds. They can be bipedal agent or they can be torso which is mounted on a mobile robot or they can be mobile robots itself, who are doing a particular task. So now what we are seeing here is that again to do this task, if I am talking about a path plan if I want to carry the object in a particular direction, there has to be a path. So, these robots also have to follow a particular path and they must have some motion to it.

(Refer Slide Time: 09:49)

Need for optimization ?

- Multiple solutions – many ways of grasping an object ;
force closure / form closure



Now, if there are multiple ways of doing the task, there is need for optimization. For example, we looked at cases of form closure and force closure. In the last two classes, I think we looked at the concept of force closure and form closure where the object is in closure by using three fingers with friction or we have four fingers or more without friction. So, suppose these three finger tips are holding an object like this and I want to rotate this object in a particular way, then these fingers have to apply forces.

Now there can be different combination of forces which can give you the same motion which basically means that first of all I need to ensure that there is a closure, force closure or form closure so the object is in follow, then I have to optimize it, optimize the motion. So, this means I need to have some kind of optimization which is going to optimize the motion. For example, I have an object here, this object, I have three fingers as shown in this figure.

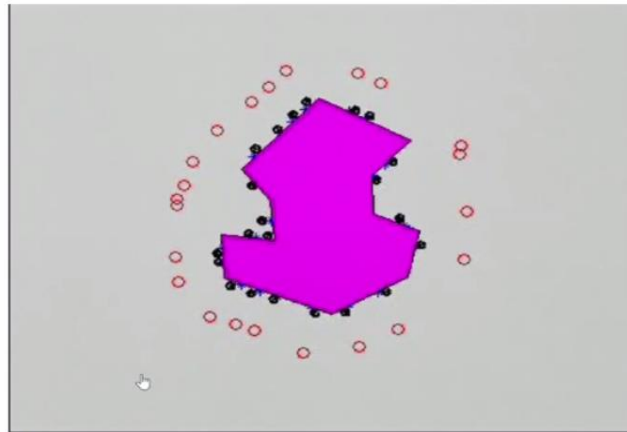
Where do I place the fingers? Do I place them here, here and here or do I place the fingers in some other way? Suppose I can place the fingers here, I can place the fingers let us say here, here, here and here also. So, this is one way of placing fingers that is the other way of placing

fingers. So, there are multiple solutions here now. So to get form closure or force closure, which is the optimal way and hence the question of optimization comes into picture.

Now, when we are choosing in the case of roadmap methods where we are making a network of paths, so we are made network of paths from the start point to the goal point. We were choosing the path by means of algorithms like the D * or A * algorithms which give us the length of the path. Now, that is not enough here because we also have to ensure form closure or force closures. For example, if the object is moving I have to catch it and then I have to do something.

(Refer Slide Time: 11:46)

Multiple objectives – minimum number of mobile robots, minimum force application, etc.



So this is another example where we have multiple mobile robots. So, these small black, black dots that you are seeing here these are mobile robots like the one I showed you. And this is a moving object, this object can move. Now I want to catch this movement object, there are so many robots you can see, there are a set of robots here and a set of robots here. Now the function of this set of robots is to go and capture this object.

This is a moving object, right? So, it has to go and capture the object and then maybe push it and take it somewhere that is the objective of this path learning with multi agents. Now this object is moving. So, if this robot tries and goes and sits here what will happen is this object is going to hit the robot, so it does not go and sit on the object straightaway, it has an expanded boundary like this.

So there is an expanded boundary like this on which the robot goes and sits that is what we are seeing here. So, what the robot will do is it will first go and sit on the boundary, then it will go and capture the object and stop it from moving. Otherwise, the object is going to hit the robots. So here, there are a couple of questions. Number one, what is the minimum number of robots required? What is the minimum force to be applied to do this task?

So this is where the question of optimization comes in. Let us look at this video. **(Video Starts: 13:02)** So you can see that the object is moving and the robots are going in positioning themselves on the expanded boundary on the blue line. And then once all of them go and put themselves on the boundary, then they go and try and capture the object because this is a moving object, what will happen is otherwise the robots will get hit.

Even then, you see they just caught the object and there are some robots which are still trying to position. So the question that we are trying to use here, let us look at the video again. You can see multiple robots are going sitting on the expanded boundary of the object, then they will jump and go and capture the object. So now, what is the question we are asking here? The question we are asking here is how many minimum robots are required?

Where should we go and capture the object? Is it in force closure? Is it form closure? These are two terms that we used in the last couple of classes. **(Video Ends: 13:53)** So that is something which we will look at in this lecture. So first, we will start off with form closure. **(Refer Slide Time: 13:59)**

Multiple force possibilities for constants in potential field method

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$

$$U_{\text{att}}(q) = \frac{1}{2} \zeta d^2(q, q_{\text{goal}}),$$

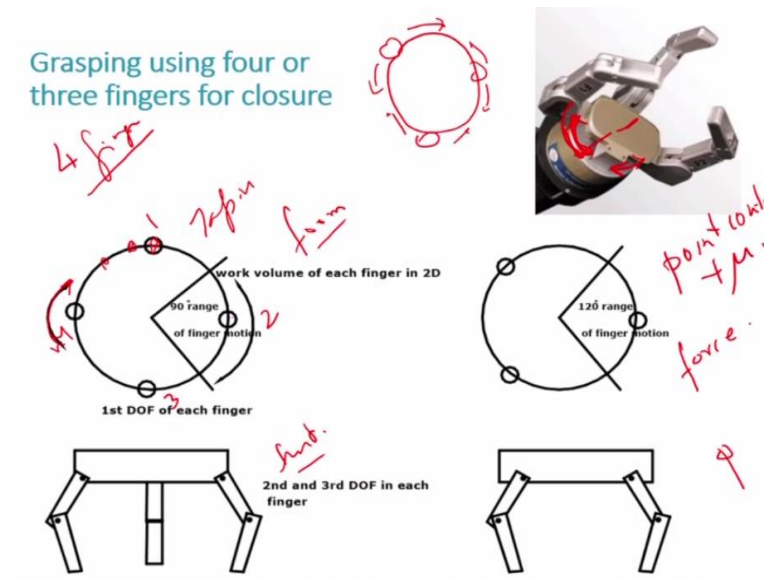
ζ → Const.

$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2} \eta \left(\frac{1}{D(q)} - \frac{1}{Q^*} \right)^2, & D(q) \leq Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

η · Const.

Also, in the case of potential field methods, we have seen that when you are using an attractive potential and repulsive potential, then this constant zeta and eta are there. So, these are constants and you have to put numerical values. So, how do you put the numerical values of these two constants that is also one need for optimization in potential field methods.

(Refer Slide Time: 14:21)



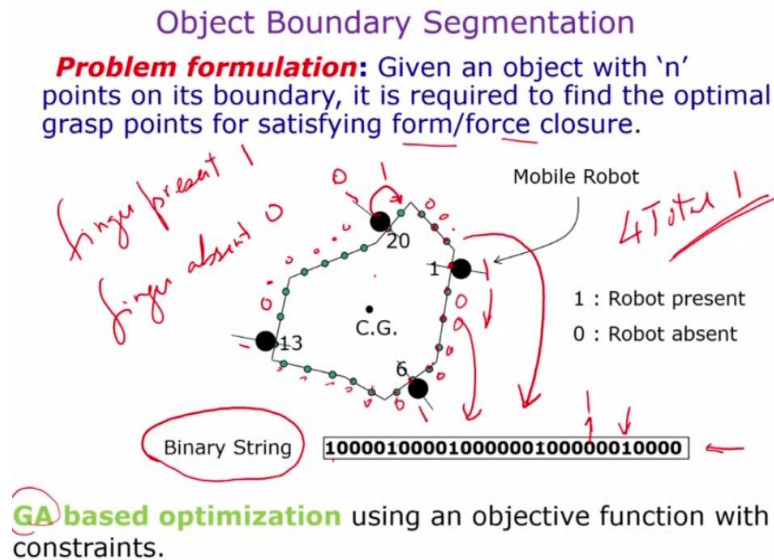
Now, let us start off here by looking at this case where we are trying to grasp an object using three fingers or four fingers for closure, either form closure or force closure. This is an industrial gripper. We said that all industrial grippers or most industrial grippers have three fingers because of which they can have force closure grasps. So, suppose you have four fingers, on the left hand side we are seeing four finger case.

Now how this gripper works is this finger can rotate about this axis. Now the finger rotates about this axis so it can rotate, so the finger can place itself, the finger can come this side. So if I see the top view, if this is the top view of the hand if this is my finger, these fingers can move in this direction. That means they can go and place their fingertips in any location. So in the figure on the left hand side we have four fingers.

The gripper having four fingers and we want to place the fingers, these are four fingers let us say 1, 2, 3, 4 fingers and this is showing the top view. This is the top view and this is the front view let us say, so there are four fingers, front view and top view. Now these fingers can move in this direction, so I can place them in different points on that circle. So, the question being asked here is in order to have form closure of this object, how would you place your fingers? Where would you place your fingers? What is the motion plan?

Now there are four fingers point contact without friction. So, we can obtain form closure, for form closure we can obtain four fingers. Now, on the right hand side we have point contact with friction, so point contact plus friction. So in this case, three fingers is enough. So, three fingers can be used to obtain force closure. So, this is we are looking at force closure, this is we are looking at form closure. So, the question being asked is where would you place these fingertips such that the object is in closure?

(Refer Slide Time: 16:25)



Now, let us look at this as an optimization problem. Now, this object let us say is, I can have any shape. So this object that is shown here this is my object and this object can have any shape issue. So, what we do first of all is that we segment the boundary. Now, this is a continuous object, in order to be able to check where the fingertip can be placed that is discrete, so if I say that I take this side and I divide it into these points that these points the fingertip can be placed.

For example here the finger cannot be placed on a corner. So, what I do first of all is I segment the boundary. So, given an object with n points on its boundary we are required to find the optimal grasp points for satisfying form closure or force closure. So, that is the problem. This is the object of some shape. Now, I want to convert this into a digital problem. So, what I do is I segment the boundary of this object, I segment the boundaries by putting these points.

So when I put this I break up the boundary into number of points where the fingertip can be placed. And then what I do is I convert this into a binary string. Now, wherever the fingertip

is present is 1, wherever there is no fingertip it is 0. So finger present means 1, finger absent means 0. So there is a finger sitting here, so I start from here. Let me start from this point, so there is a finger sitting there so this is 1 and if you are going clockwise direction then it is 1, 2, 3, 4 are there.

So there is 0 0 0 0, again there is 1, then it is 0 0 0 0 0 0, then there is a 1, 0 0 0 0 0 0, 1, 0 0 0 0. So if I start counting from here in a clockwise direction, then it is 1, 0 0 0, 1, 0 0 0 1 that way. So I have converted this analogue problem into a digital problem now. So this binary string is actually telling me; on that object where the fingertips are present. So wherever there is a 1, the fingertips are there. Therefore four 1, 2, 3, 4.

There should be four, I think there is a mistake somewhere, anyway there should be four fingers, So wherever finger is there it is 1, where it is not there it is 0. Now, this is a binary string. Now, a computer program can very easily handle this binary string. So, you can guess that now suppose I want to change the position of the fingers, what will, I do I will simply change zeros and ones. If I change the 0 to make it 1, the finger has come there now.

Suppose the finger from here goes here, what will happen? This will become 0 that will become 1. So what I can do is I can use optimization like genetic algorithm for example, what it does? It takes this binary string and changes those bits. It makes some of them one some of them zeros, ensuring that there are four total ones because this is a four finger gripper. So, there is going to be four robots present always.

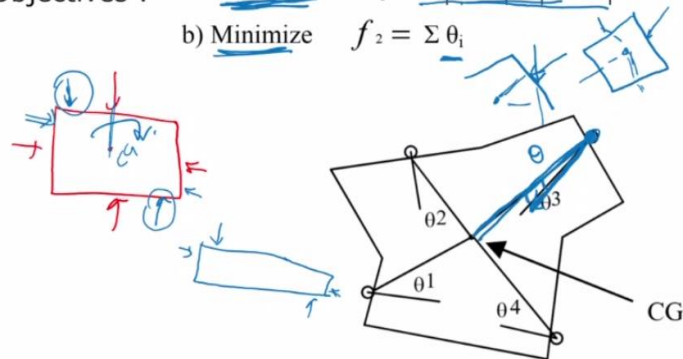
Now if there is more than four it can be one, you can make it 1, it does not matter. So, what GA would do essentially it is going to switch those bits and that essentially means in the physical system that is where the fingertip is present now. So that is equivalent to saying that I am placing my fingertip in different points on that object.

(Refer Slide Time: 19:48)

Grasp satisfying form closure with four fingers

Finger tip contact : point contact without friction

- Objectives :
- a) Maximize $f_1 = |M_{cw}| + |M_{ccw}|$
 - b) Minimize $f_2 = \sum \theta_i$



Moments cause by fingertip placement: CW , CCW

Now, how do you satisfy form closure? So we are saying that we are keeping the fingertip at different locations, but we have to satisfy form closure. So, what is the optimal way of satisfying form closure with four fingers? So one way is the fingertip contact is without friction, so it is point contact without friction, so you can apply only one force. So, one way would be to maximize the clockwise and anti-clockwise moments.

Let us look at this case. Suppose I have a rectangle and I have four point contacts. I can place my four fingers like this or what I could do is I could place the finger such that the; I place two fingers here, two fingers here, two fingers here and two fingers here. In that case, what would happen is I can maximize the moments, why? In this case, this fingertip is going through the CG, so it is not generating any moment.

So, it cannot resist moments whereas if I put my fingers here it is not only resisting the forces, but it is also resisting the moments now. Suppose there is a moment in this direction, this force and this force will resist that moment. So we can place the finger such that you maximize the clockwise and anti-clockwise moments. So for example, in this particular case I gave you have a very long object, it is a good way of catching it here and here and here and here, why?

From statics point of view you are maximizing the moments that means you have to apply less force to take care of any moment disturbance that is coming. So, this is the first one maximizing clockwise and counter clockwise moments. So, these are the objectives. Now, we

are doing GVS optimization, so we need to have objectives. Now, the objective can be single objective, which you can maximize or minimize, it can be multi objective.

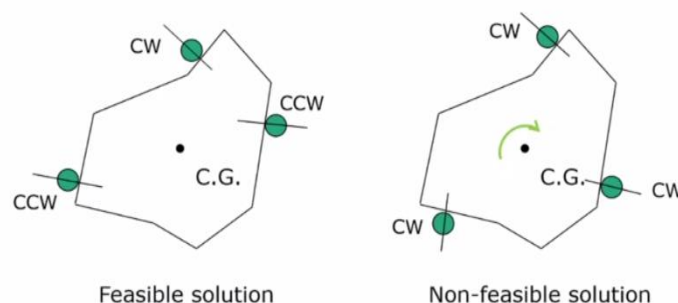
In this case, it is multi objective, there are two objectives, so we are maximizing this and we are minimizing this one. Now, what is the minimization problem? Minimization is we are minimizing the total angle subtended in the centre. For example, if I have a fingertip here and this pointing in this direction, so the fingertip force is going to be normal to the surface, right. And this is the line connecting the fingertip contact to the CG, then what I am saying is that is my angle θ and I want to minimize it.

Why do I want to minimize because if you know that if you have an object like this and you are placing fingertips all of which pass through the CG, then the system is going to be static equilibrium. That means the fingertip contact is, just let me give an example like this, if this is my CG and the fingertip is pointing in this direction, so what is happening is it is generating a moment.

Whereas if the; fingertip is pointing in this direction through the CG, then it is not generating a moment. So, we are maximizing this clockwise anti-clockwise and we are minimizing the angle θ so that the object is in static equilibrium. So, we have two objectives, maximize clockwise anti-clockwise moments and minimize the angle subtended in the centre.

(Refer Slide Time: 22:43)

There should be robots creating moments in both the direction about C.G.



This constraint takes care of rotation inaccessibility.

Now, we need to worry about the accessibility angle. So, the first constraint we saw is that the robot should be creating moments in both the directions, clockwise and anti-clockwise so

in order to balance the external torques, the external disturbance torques that may come on the system.

(Refer Slide Time: 23:02)

Other constraints

- Fingers will not overlap.
- Total number of finger is 4. *form closure of 2D object 4 finger*
- Area formed by the finger tip contact is not zero.
- Number of finger producing CW and CCW moments are equal.
- Accessibility angle is zero. *escape in any direction.*

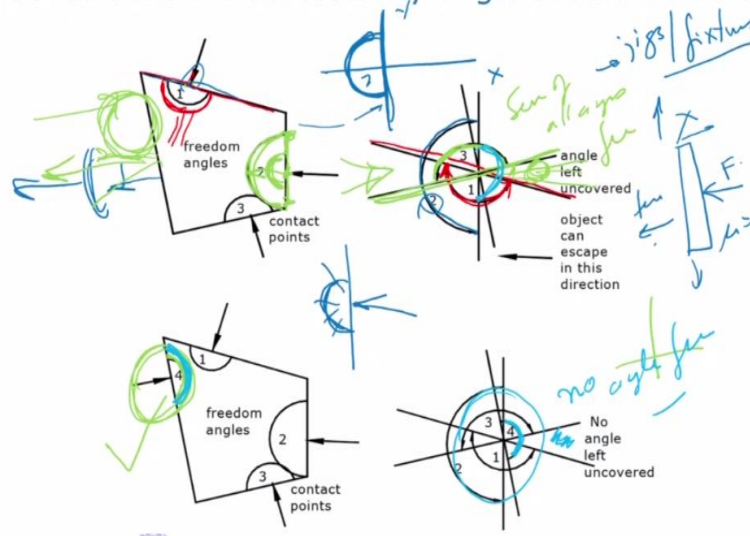
Now, the second thing is other constraints are, so this is one constraint, the other constraint is the fingers should not overlap, the total number of fingers should be four because we know that for form closure for 2D objects, so form closure of a 2D object we need four fingers that we have seen before. Now, the area formed by the fingertip contact is not zero and the number of finger producing clockwise anti-clockwise moments are equal and the accessibility angle is 0. Now, what is this last one, which is very important for us here?

Now, suppose I have an object like this and I place all the fingers here, what will happen? The object can go this side, can go this side, the object can go that side. It can, it is not constrained, it is not a form closure. So, simply maximizing and minimizing the moments may not give you the solution. So, what we need to do is we need to also look at the accessibility angle. The accessibility angle essentially means that if you have caught an object by four fingers, it should not be able to escape in any direction.

So what this means is that if I put four fingers here, here, here and here this object if it is a moving object it should not be able to escape or move in any direction, it should be constrained, then it is in form closure. So how do I get, how do I ensure form closure?

(Refer Slide Time: 24:27)

Constraints 1 : accessibility angle should be zero



So what we check is what we call the accessibility angle or the freedom angles. Now the concept is very simple and this is also used in jigs and fixtures. For example, if you are looking at an object, this is my object and I put my finger here, this is my finger. Now if I put my finger there it means that I am applying a force F in that direction. So the object cannot move in this direction, it cannot move, but the object is still free to move that side, this side and this side.

There is no μ there, so this is $\mu=0$. So the object is free to move in the other direction. So this is like saying if I have a fingertip here, this is my fingertip. This is my freedom, the object is still free to move 180° from here to here, any direction it can move. Similarly, if I put my finger here then the object is free to move in any direction like this, so this is called my freedom angle.

So, if I put one finger here, one finger here, one finger here the object is still free to move in this direction. How can I check that? That is actually very easy because please remember that this is a computer program which is doing this. As I said many times during this course that we have eyes so we can see, so I can guess there are three fingers here, the side is still free, so that we can go that side. But the computer program does not have eyes, it needs to figure this out geometrically.

So how does it geometrically figure this out? So what it does is basically it does a sum of the accessibility angles. So what this means is suppose let us say coordinates this is my x and y , this is x and this is y , this side is here, this side is here, and this angle 2 is like this, the same

direction. So wherever direction the force is opposite to that force this is the freedom angle, which is on the other side. So if I have a flat surface like this, there is the force here, this is my freedom angle because it can move in that direction.

Now, so that is the first one, so this is two. So you can see here I am saying 2 is here, that is my number 2. Next what I do is I look at 1, so where is 1, it has this inclination. So, 1 has this inclination, this surface so I draw this here, now where is the freedom angle? Freedom angle is here, that means the object can move in that direction as human the only this fingertip is there. So, this is my freedom angle where this is for 1, now there is another one that is 3.

So for number 3, what we can say is this is my freedom angle. So, number 3 freedom angle is here. Now if I take the sum of all these angles, then what I do get is this part is still free, you can see here that this part is free, this part is not covered. So, if I sum all the angles, I see that this part is still free, what this means is the object can escape in this direction and which direction is this? It is has this direction, so it is here, so the object can actually go that side.

So, if you have three fingers or three point contacts, the object is still free to go this side. Now, the program can very easily find this out. How the program can find it out? It is at every point, now how does it know at which point the finger is there? Basically, it goes back to the string, the binary string actually tells us that at which point on the boundary the fingertip is there, for example one finger is here.

So it knows this is the first point, this is the second point and these are vectors, right? So, what it does is it finds this angle, it can find that angle very easily and now it knows that what is the angle and converts it into the total diagram. So it is very easy to find the accessibility angle geometrically. All the program has to do is at every point it has to find the normal, so at this point it finds the normal and then finds this angle.

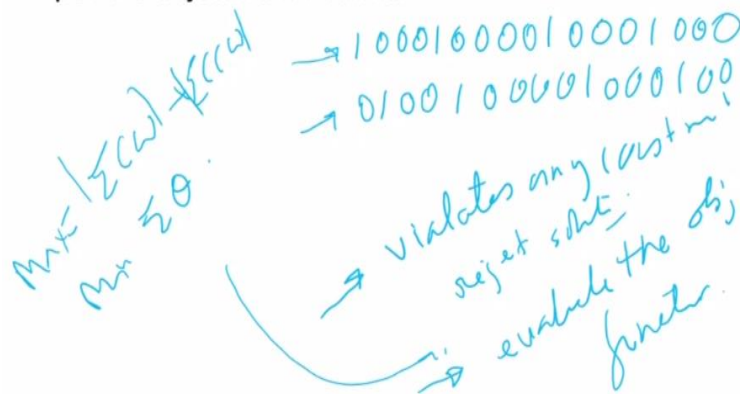
So this is a point here, this is the point here, this is the point here, you know the coordinates of these points. So this is a vector, this is a line, these are the lines, so you can find the angle between those lines. So, you can find this angle geometrically, then from here we come here and what we do is we find the accessibility angle. So, if 360^0 is fully covered, for example if you look at this case here and you look at the case at the bottom, so in the top case this location is free, in the bottom case one finger is put there.

So the moment you put a finger there this freedom angle is on this side now. So, what has happened is this freedom angle, this angle has come, this side now. So, this is my angle number 4. So this angle which was free before here that is covered now. So, if you look at this one now, there is no annual fee. This means the object cannot go in any direction. That means the fingertips have been placed in such a way that all the directions of escape of the object are covered.

(Refer Slide Time: 29:29)

Multi objective GA for optimization (NSGA II)

- Generate a large pool of solutions and rank them as per the objective functions.



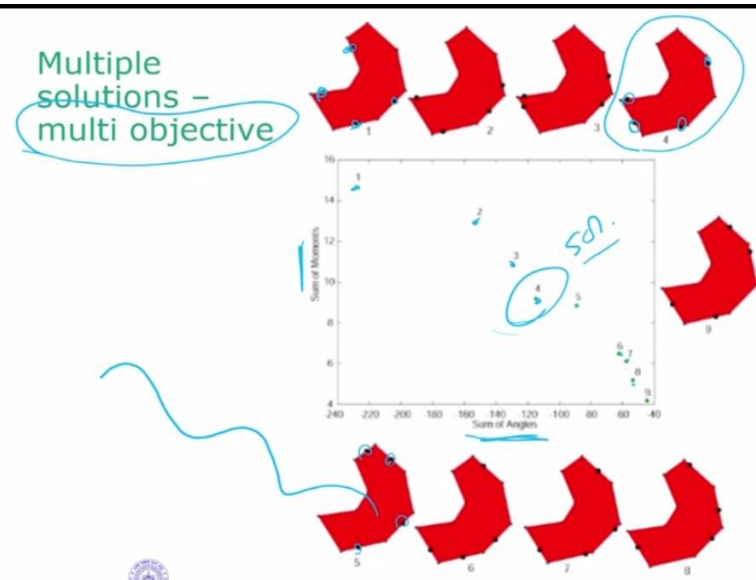
Now, how does this work? For example, we are using genetic algorithm. What genetic algorithm does is basically it takes that pool, say we had this binary string that we saw. So what GA will do is basically to switch some of this, it will make this 1, makes this 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0. So it generates another one by doing crossover and mutation. So that is what it does. Then what it does it evaluates the objective functions with the constraints.

So, suppose it violates any constraint, for example what is the constraint if the accessibility angle is not 0 that means the object can still escape, then it will reject that solution. Now, if it is not violating any constraint then what it will do is it will evaluate the objective functions. We have shown two objective functions which it does not evaluate, one was the moments, so sum of clockwise moments counter clockwise moment plus clockwise moment.

The mode of this and mode of that so it is maximizing this. And the other was it was minimizing the angle known θ . So, what it will do is it will evaluate this function and then give a particular value to that. This is how GA basically works. It is very simple to

understand that genetic algorithms basically look at this digital string and keep changing the bits in the string seeing solutions which do not violate any constraint, find the objective and then proceed.

(Refer Slide Time: 31:09)

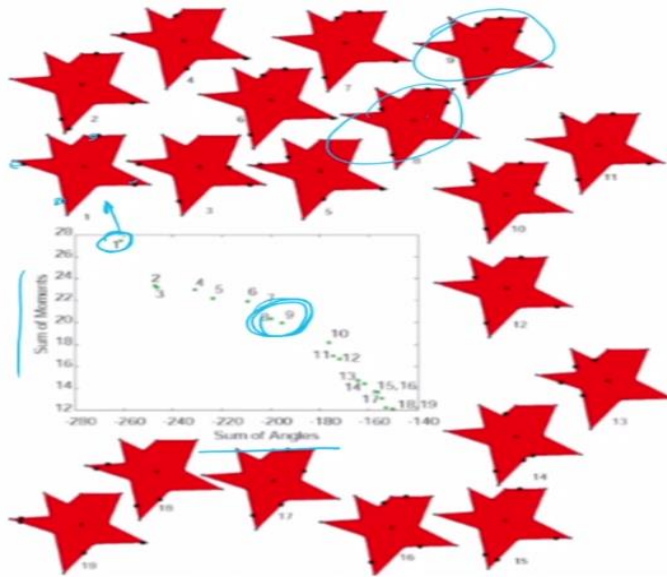


So, by doing that these are some examples. For example, the red one is an object and you are placing fingers at different positions. So, this is one finger, another one, another one, another one. So, in each of these you can see that those fingers are at different locations and the objectives are being evaluated. And if you plot them the sum of angle versus sum of moments, then these are some of the solutions that we are getting, this is multi objective.

So, multi objective will get multiple solutions. Then from one of these, you can choose one of the solutions, for example, I can choose this solution. So, this means solution number 4 is this one, I place my fingers here, here, here and there. So, this is the case of multi objective optimization, how we can find where the fingers should go to catch the object. Now, in the case of mobile robot we can say this is where the fingers should be placed, sorry where the mobile robot should go and catch the object.

So, suppose this object is moving like that, and I want to go and capture it, then the mobile robot should go and sit in these places for form closure. So, that is what this is telling us.

(Refer Slide Time: 32:13)



Now, this is another example star kind of an object. Again, you can see the sum of the angles are here and the sum of moments is there and these are locations of the fingertips. Wherever the fingertip is going and sitting, this is for number 1, so this is a solution where the location of the fingertip is. So again, this is multi objective, so there are multiple solutions, so you can choose the solution somewhere here which is giving equal weight to both the objectives.

So 8 and 9, where is 8 and 9? So 8 and 9 are here, 9 is here and 8 is here. So this would be good solutions which give equal weightage to both of them. So, this is how we can find where a multifinger robot hand can place its fingers or a mobile robot can go and capture an object, this is the same thing.

(Refer Slide Time: 32:54)

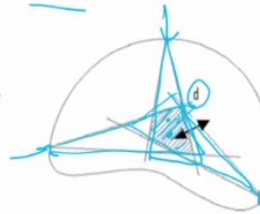
Force closure using three fingers

• Finger tip contact : point contact with friction

- Objectives :
- Max area of triangle
- Min force focus
- Force line (Min θ)



(a)



(b)



(c)

3 fr

M_i

Now, for three fingered hands, it is force closure, not form closure. So, in force closure we know that it is point contact with friction, in the previous case there was no friction. And in this case, we need three fingers that we have seen before. So, in this case again the question is the same, I have three fingers, I want to get this object where should I place my fingertips and we say to obtain force closure from statics point of view that the objectives are three of them maximum area of triangle.

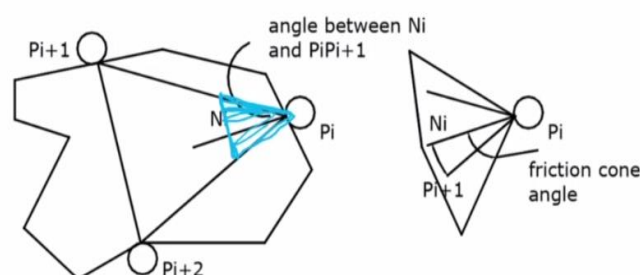
So the fingertips should be placed such that this triangle is maximized, why you can guess again so that your grasp is going to be stable. So, when you catch an object if you catch it on the boundaries, what will happen you have most stable grasp so that is one. The second is minimum force focus. So, suppose my finger is here, finger is there, finger is there now because this has friction, there is a friction cone which is there.

So, these are friction cones because there is friction there now. So, this intersection of the friction cones gives us this area and what we can do is we can say minimum force focus. So, what we can do is I can take the CG of this area and the CG of the object, I can take the distance d and try and minimize that. The third is force line which is same as the previous case. So, this object is here, it is pointing in this direction, so it is making an angle θ there.

So, we should place the fingers at locations where the normal at that point goes through the CG that means this θ should be minimized. So in this case for force closure, we have three objectives, maximum area of triangle, minimum force focus and force line minimum θ .

(Refer Slide Time: 34:40)

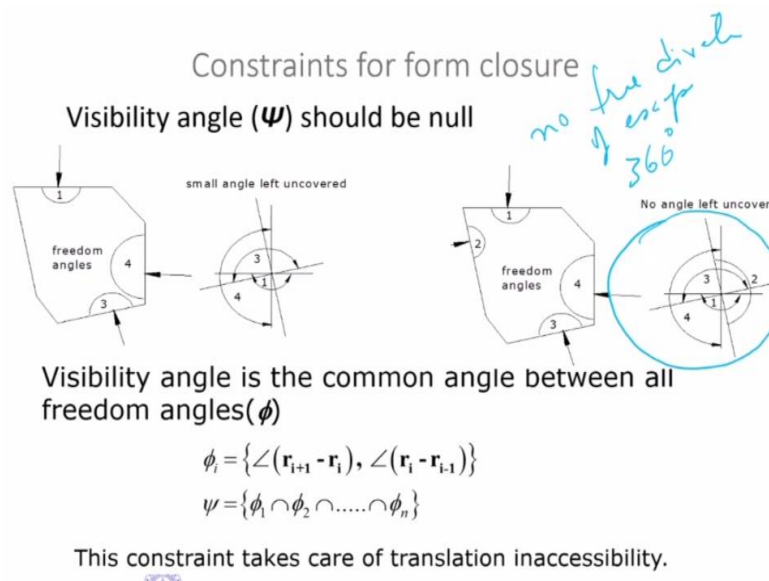
- Constraints
 - Fingers = 3
 - Friction cone should be satisfied
- 10000/000 / 00000
- (CA)



Now, what are the constraints? The constraints are that there should be three fingers. And why is this required? For example, I will use the same string again. So it is 10000100001000 that is three fingers now. So when I am using GA here and GA changing those bits. Any case where there is more than three ones it will get rejected, there is no solution, it is violating a constraint. And number two the friction cone should be satisfied.

So, if I have a friction cone here like that then the; normal force should be inside the friction cone, otherwise it will sit there. So, these are the two constraints that are kept. Now, accessibility angle also is used.

(Refer Slide Time: 35:36)

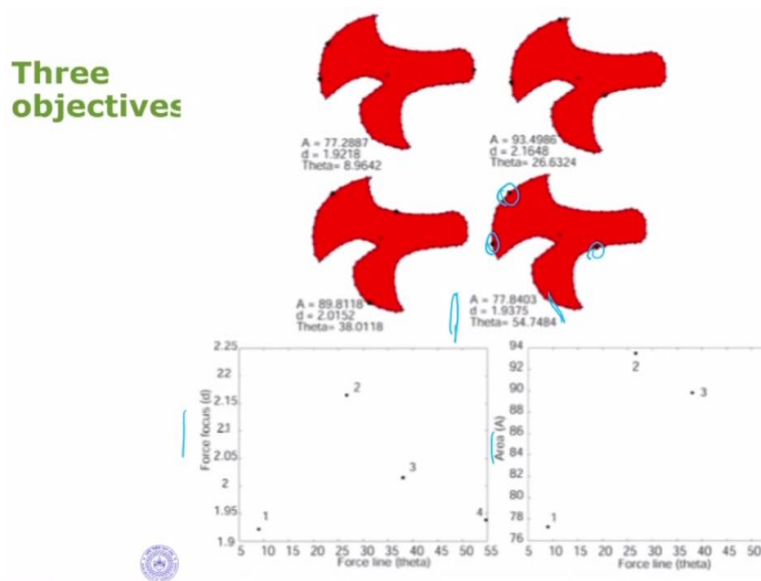


So accessibility, how this presents is essentially we do boundary segmentation, exactly the same way I divide the boundary into discrete points, make my binary string, check for accessibility angle, the freedom angle and ensure that the freedom angle covers 360° in space. That means there is no free direction in which it can go. So, no free direction of escape. So after placing the fingertips, we see that the objects cannot go in any direction, cannot escape, so that is a constraint.

So what GA will do is this is going to change the strings, it is going to change the binary string and then place the fingertip at locations where constraints are not satisfied and it will evaluate the function, it will evaluate the objectives.

(Refer Slide Time: 36:10)

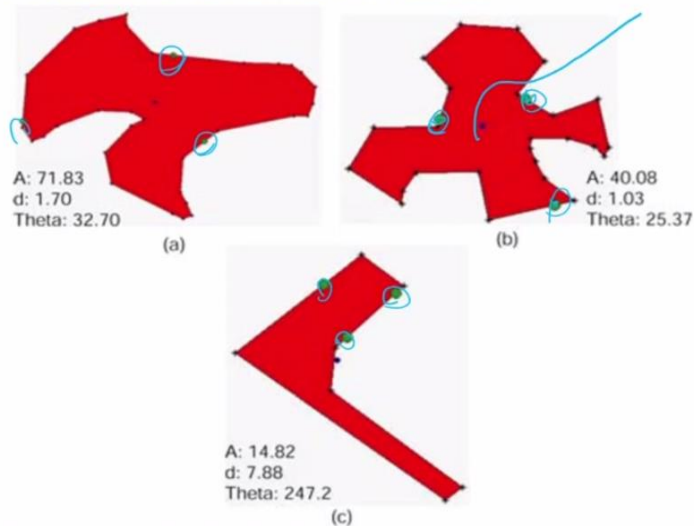
Three objectives



Now there are three objectives. One is force focus, force line, and area of triangle. So we get multiple solutions. This is actually a surface of solutions. So, you can choose a solution. For example, in this particular case A , d and θ has this particular value. Fingertips are here, here and here. So depending on what you want, you want to give more weightage to which part to force focus or to force line or to area, depending on that we can place our fingers accordingly.

(Refer Slide Time: 36:42)

Best solutions for different cases



These are another two cases where if you want to cast the subject, where is the best place to place your fingers, probably here, here and here. So here, here, here for this object is here. So, this is basically giving you an idea that if you want to do optimization here, you have multiple solutions. And for each of these multiple solutions, you need to evaluate the objectives and then come up with which is that grasp that you want.

Now, I am using the word grasp but this can also be done by multiple mobile robots. For example, if this is a moving object and these are mobile robots, then essentially it is going there and getting the object.

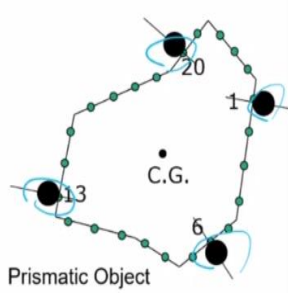
(Refer Slide Time: 37:21)

Objective Function 40%

Maximize $f = \left(\frac{|M_{cw}| + |M_{ccw}|}{N^k} \right) \left(\frac{1}{|N_{cw} - N_{ccw}| + \epsilon} \right)$

Moment is calculated by assuming unit normal force applied by robots.

N : Total No. of Robots
 M_{cw} : Sum of CW moments
 M_{ccw} : Sum of CCW moments
 k : parameter for controlling No. of agents.



Prismatic Object

If you want to do single objective optimization, we can still do a single objective. In single objective optimization what we have done is we have used clockwise moment plus anti-clockwise moment,

$$f = \left(\frac{|M_{cw}| + |M_{ccw}|}{N^k} \right) \left(\frac{1}{|N_{cw} - N_{ccw}| + \epsilon} \right), \text{ where } N \text{ is the number of robots.}$$

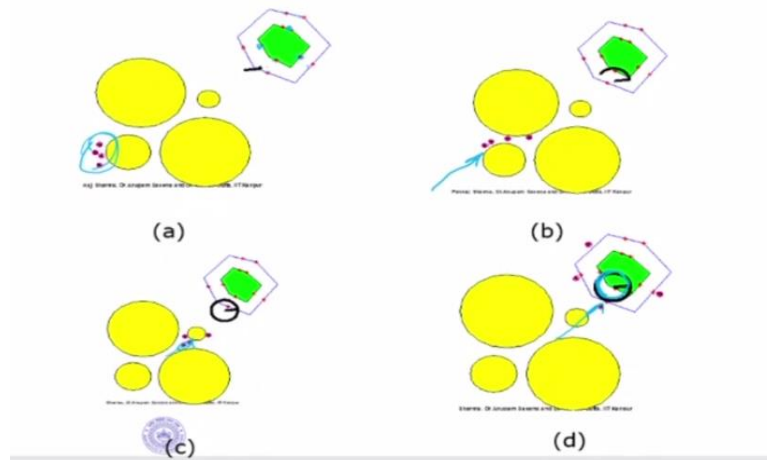
Suppose we are looking at a case where we do not know how many robots are required to capture the object, then I say N is the number of robots. Now I want to minimize the number of robots used and I want to also satisfy this surjective function.

Now this surjective function is a sum of the moments here, number of robots come in the denominator. So what GA will be doing is basically it will be switching the number of robots also. So, it can start off maybe with 50 robots and start coming down and seeing what is the minimum and the minimum is going to be for four because we know that force closure and form closure we need four robots. So you can have single objective, you can have multi objective optimization.

(Refer Slide Time: 38:16)

Obstacle avoidance

capture this obj - 0.



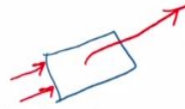
Now, this is an example of object capture where you can see that the object is. So, these are the robots, four robots, this object is moving. The green object is the obstacle which is moving and it is moving in a circle, you can see it here it is moving in a circle. So, these robots move by using a path plan basically. So basically, what the question is here capture this object by mobile robots.

So the first thing it has to do, it has to find out on the expanded boundary where it will go and sit first, then where to go and catch the object by using what we just discussed optimization. So, from the optimization, the answer would be that the robots need to go and sit on the boundary first here, here, here and here. And then on the object itself these are the points where the robot will go and sit to satisfy form closure, then it is caught.

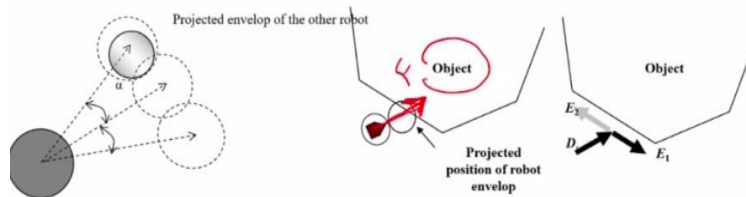
So this is the first you can see the robots are here, then the robots start moving there, then the object is moving, then goes here and then finally it goes and captures the object. So, this basically shows us the four stages in which the robot is going and capturing the object.

(Refer Slide Time: 39:29)

Projective path planning



- Each mobile robot image is projected a few instants of time ahead and then checked for collision.
- In case of collision it is moved either in the right or left directions.



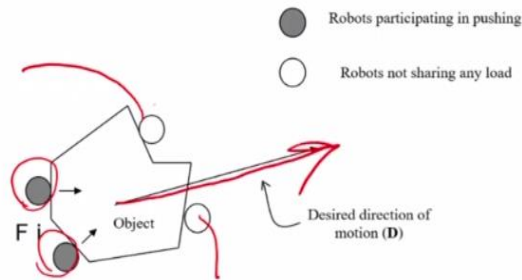
Now, after capturing the object the robot is expected to go and push the object and take it at some point. Now to push the object, say for example I have an object like this a rectangle and I have four robots. So after capture, the robot is supposed to go and push the objects in a particular direction. So, let us say that we have an object like this shown in red colour. So, this is my object and what I want to do is I want to push it in some direction.

So if I want to push it, to capture it please understand that I have to place my robots like this. Now suppose I want to push it in that direction, what I must do is I have to reposition this robot such that the robot pushes in this direction now, otherwise it would not go in that direction. So, the robot has to change its position again and have a new path plan to push it somewhere else. So for example, in the object here the robot sits here and pushes and applies a force in this direction.

(Refer Slide Time: 40:26)

Optimal pushing of the object to the desired goal point

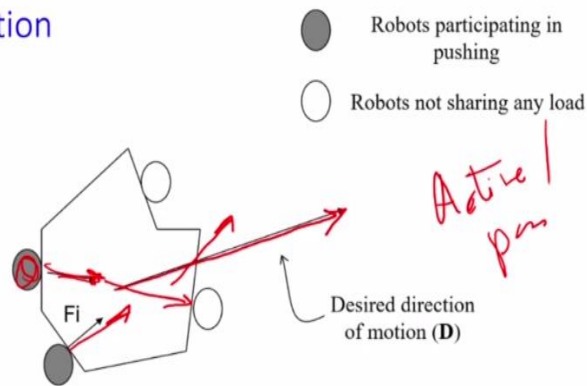
- The robots are reoriented for optimal pushing by switching between passive and active robots.



So, what we need to do is now we need to reposition the robots and after repositioning it then we can push in some direction. Say for example, after capture we have four robots which are positioned like this, now it has been captured, now I want to push in some direction. So, this robot should have to move away and then allow this fellow to push it and take it in this direction. So, I have to reposition the robots again.

(Refer Slide Time: 40:47)

Optimization



Maximize

$$f = \mathbf{D} \cdot (\mathbf{F}_1 + \mathbf{F}_2 + \mathbf{F}_3 + \dots + \mathbf{F}_n)$$

$$f = \mathbf{D} \cdot \mathbf{F} \quad \dots (1)$$

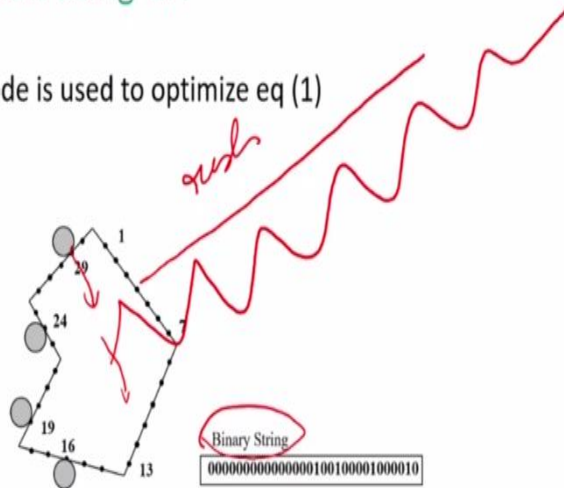
And how can we reposition the robots? Essentially, we can do optimization again where we can maximize this. Suppose this is the direction, some robots are going to push in that direction and some robots are going to be passive. So, it is a game between active and passive robots. So, when this robot is pushing the object will move in this direction, when this robot is pushing the object will move in that direction.

So, by making a combination of these two motions, the robot can be taking towards them in a particular direction. So, what we are optimizing is this basic function where D is the desired direction the vector $f = D \cdot (F_1 + F_2 + F_3 + \dots + F_n)$.

(Refer Slide Time: 41:25)

Optimization using GA

- Gordy GA code is used to optimize eq (1)

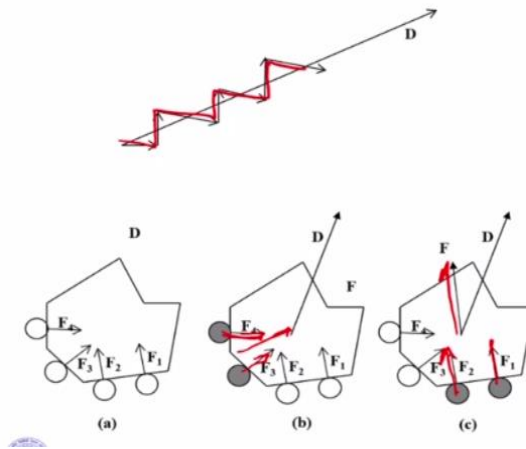


So, again we do exactly the same way. So, basically we use the binary string which indicates where the robot is sitting and then we make the robots active and passive. For example, this robot is active means it will push in that direction, so the object will go that side. Next, this robot will push, so the object will go this side now. So, by doing this kind of motion, you end up with this kind of motion.

But there is no other way because you will never get the resultant in this direction suppose I want to move an object like this. So, I will not get the resultant of forces in this direction by placing robots like this. So, I can make some of them active, some of them passive and then push them accordingly.

(Refer Slide Time: 41:59)

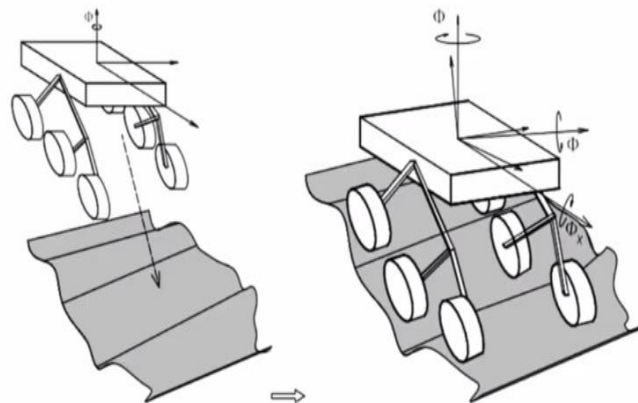
Motion of the pushed object



For example, in this case what the GA is doing? It is first using these robots to push, so what is happening is when these two push the resultant force will be in this direction. Then these two will push the resultant force will be in this direction now. So, what kind of motion would you get? The object will have this kind of motion depending on which two robots are active at which time. So, this is the way an object is pushed and taken to some other position.

(Refer Slide Time: 42:27)

Potential Fields for Rovers on 3D terrain



So, this is basically multiple agents guiding an object and taking it some someplace.

(Refer Slide Time: 42:33)

Multiple force possibilities for constants in potential field method in 2D – need for optimization

$$U(q) = U_{att}(q) + U_{rep}(q)$$

$$U_{att}(q) = \frac{1}{2} \zeta d^2(q, q_{goal}),$$

$$U_{rep}(q) = \begin{cases} \frac{1}{2} \eta \left(\frac{1}{D(q)} - \frac{1}{Q^*} \right)^2, & D(q) \leq Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

We had looked at the case of the potential field method in 3D where we said that or even in 2D that there are constraints, this is a constant, this is also a constant. So, again we have to do optimization of some kind.

(Refer Slide Time: 42:48)

Proposed Rough Terrain Path Planning Algorithm

Net force acting on the rover:

$$F_{total} = w_1 \times F_{att} + w_2 \times F_{rep} + w_3 \times F_{tan} + w_4 \times F_{grad}$$

where w_1, w_2, w_3, w_4 are the weight parameters for the attractive, repulsive, tangential and gradient forces respectively.

Attractive Force

$$F_{att}(q) = -\nabla U_{att}(q) = -\left[\frac{\partial U_{att}(q)}{\partial x} \quad \frac{\partial U_{att}(q)}{\partial y} \right]^T,$$

-where

- $U_{att}(q) = \frac{1}{2} \zeta d^2(q, q_{goal})$
- $\nabla U_{att}(q) = \zeta d(q, q_{goal}) \cdot \frac{\partial d(q, q_{goal})}{\partial q}$
- $F_{attx}(q) = \zeta d(q, q_{goal}) \cdot \frac{\partial d(q, q_{goal})}{\partial x}$
- $F_{atty}(q) = \zeta d(q, q_{goal}) \cdot \frac{\partial d(q, q_{goal})}{\partial y}$
- $d(q, q_{goal}) = \sqrt{(x - x_{goal})^2 + (y - y_{goal})^2}$ is the Euclidean distance between the robot q and the goal q_{goal} in 2-dimension and
- $c = 1$ or 2 .

GA randomly
 w_1, w_2, w_3, w_4
 $0.5 \quad 1 \quad 2 \quad 5$

Which means that in case of 3D this is something I have explained in a couple of classes back that if you are going from 2D to 3D, what we can do is we can put the gradient force also, so F_{grad} is there. Now, this w_1, w_2, w_3, w_4 are weights or constants. So, what we can do by GA, again we can use genetic algorithms here, what GA will do it will have $w_1, w_2, w_3,$ and w_4 and it will randomly switch these values, numerator values, and then it is going to evaluate some surjective function.

(Refer Slide Time: 43:23)

① **Objective function 1:** evaluated velocity ratios of the rover wheel for minimum energy. GF

$$f_1 = \min \sum_{l=1}^s (L_l) \left(\sqrt{\left(\left(\frac{\omega_{A_3}}{\omega_{A_1}} \right)^2 + \left(\frac{\omega_{A_5}}{\omega_{A_1}} \right)^2 \right) / \left(\left(\frac{\omega_{A_4}}{\omega_{A_2}} \right)^2 + \left(\frac{\omega_{A_6}}{\omega_{A_2}} \right)^2 \right)} \right),$$

where L_l is the length of the path between l -th and $(l+1)$ -th segment in the path, and s denotes the number of segments in the path.

② **Objective function 2:** to avoid terrain failure and wheel slip, algorithm minimizes the ratio of the traction force to the normal force along the path [8].

$$f_2 = \sum_{l=1}^s \max_i \frac{T_l}{N_l} \cdot l$$

③ **Objective function 3:** minimum power consumption along the path [9]

$$f_3 = \min \sum_{l=1}^s \frac{M_R g_m^2 r^2}{K_t^2} \sum_{i=1}^n T_i^2$$

where n denotes the number of wheels of the rover, M_R is the motor resistance, K_t is the motor torque constant and g_m is the gear ratio.

④ **Objective function 4:** minimum length of the path from the start to the goal position

$$f_4 = \min \sum_{l=1}^s (L_l),$$

where $L_l = \left(\sqrt{(x_{l+1} - x_l)^2 + (y_{l+1} - y_l)^2 + (z_{l+1} - z_l)^2} \right)$.

For example, the objective could be in this particular case, the objective function that is being evaluated is the ratio of the wheel velocities. So, you can imagine that if the robot is like this and there is a wheel here and there is a wheel there, then this is ω_1 this is ω_2 . Now, if it is on flat ground that ω_1 will be equal to ω_2 , $\frac{\omega_1}{\omega_2} = 1$. The moment it is on the plane ground what will happen is the velocities will change now.

So, if you want the robot to be in flat ground, then what we can do is I can take the ratio of the velocities, so this velocity left side velocity divided by the right side velocity and L_1 is the length. So, I am going to minimize this one that is one objective function. You can also have objective functions like minimizing the torque, torque the ratio of traction force to normal force. So, when this vehicle is moving in uneven terrain basically what we want to do is we want to minimize the ratio of the traction force to normal force.

You could have another objective like minimum power consumption. So, this is the torque at the wheels. So, torque square is the total power that has been consumed, I want to minimize that or you could minimize the length of the path itself. So, when we are doing A* or the D* algorithms, we are basically minimizing the length of the path. So, you can use that also as an objective here. So, what GA is doing is it is switching these values of the weights.

So, there are these weights w_1, w_2, w_3, w_4 . So, it is switching these values, says it is 0.5, this is 1, this is 2, this is 5, randomly keep switching them, evaluate the objective and find which one gives you the best value.

(Refer Slide Time: 44:57)

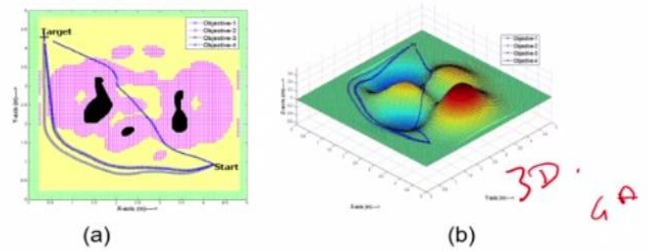


Figure :Comparative result for different objective functions: (a) 2D representation, (b) 3D representation.

Table :Comparative results for motion planning algorithm for each objective function.

Function number	Function value	W_1	W_2	W_3	W_4
Objective - 1	5.67	20.0	0.14	0.99	1.52
Objective - 2	61.79	15.5	1.89	0.34	0.91
Objective - 3	1.47	32.0	0.23	0.31	1.86
Objective - 4	5.35	57.0	0.71	0.96	1.78

Now, this is an example that gives us some idea. Now it is saying the w_1 which is the attractive force is the highest value that is expected. So, on the 3D terrain when we ran the GA algorithm to find the weights we found that the weight w_1 has the maximum weightage because it has attractive force and the next is the gradient force, then the third is the lowest value is the repulsive force and then it is the tangential force.

So, for all the objectives you can see that the value is the same, I mean the trend is the same. So, w_1 attractive is always the highest and then followed by the gradient force. So, this gives us an example or tells us an idea that when you are doing a potential field method what should be the values of the numerical constants that you are putting. The attractive one should be at least ten times that of the repulsive force. So, the robot is always going towards the attractive force.

(Refer Slide Time: 45:58)

Atlas 2018



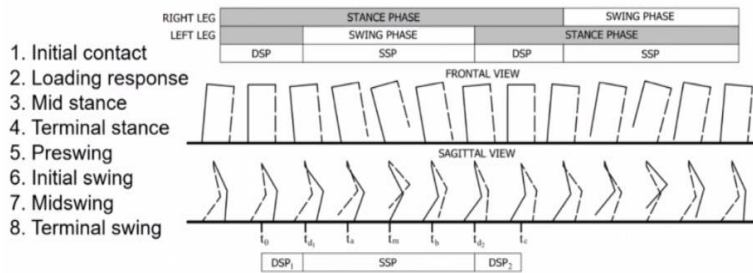
This is a very famous example of Boston Dynamics, **(Video Starts: 46:03)** one of the bipedal walking robots Atlas. So, this robot can walk around like it can go outside, it can walk in uneven terrain. Now, this is the case of path planning and motion planning and this is probably the most complex in the world today. So, we see this concept of path planning and it is also doing different tasks. It can pick up objects and put it somewhere, very interesting example.

It can pick up objects and for example here it has resistance to, it can reject disturbance. Now, it is even more unkind, it is the most unkind thing to do to a robot probably. So, the robot is being pushed and it goes and falls down. The next question is how do you get up? So, getting up in terms of motion planning is what is the motion plan the robot should have had to get up, so there is also motion planning problem.

We did not come this far because we did not discuss such complex systems. But today research in motion planning is in this direction that if the robot has to go outside, open a door, if the robot falls down it has to get up, how do you do this kind of tasks. **(Video Ends: 47:22)** Now, optimization is required because there are multiple solutions.

(Refer Slide Time: 47:23)

Walking Biomechanics



• Natural human walking gait is biphasic, and bilaterally symmetric

• The joint angles of the right leg in a step will be the same as the joint angles of the left leg in the previous step

$$\theta_{right}(t) = \theta_{left}(t - t_c)$$

Above θ s are the vectors of leg joint angles,

$$[\theta_{hip}, \theta_{knee}, \theta_{ankle\ roll}, \theta_{ankle\ pitch}]^T$$

I will very quickly explain what is done here. **(Video Starts: 47:30)** So, what we basically do is when a biped robot walks, you can see that it walks in two planes. One is the frontal plane, it moves from side to side and in the side plane it goes forward. So what we do basically is we assign, so this is in 3D, the biped robot is walking or balancing. **(Video Ends: 47:46)**
(Refer Slide Time: 47:48)

Balance : Zero Moment Point

Biped subjected to acceleration and grav

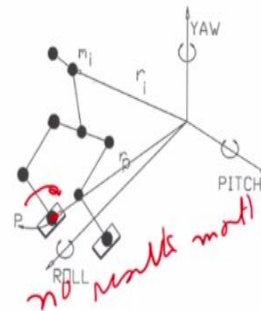
$$F^{g_i} = mg - ma_G$$

$$M_X^{g_i} = XG \times mg - XG \times ma_G - \dot{H}_G$$

Biped subjected to contact forces

$$F^c + mg = ma_G$$

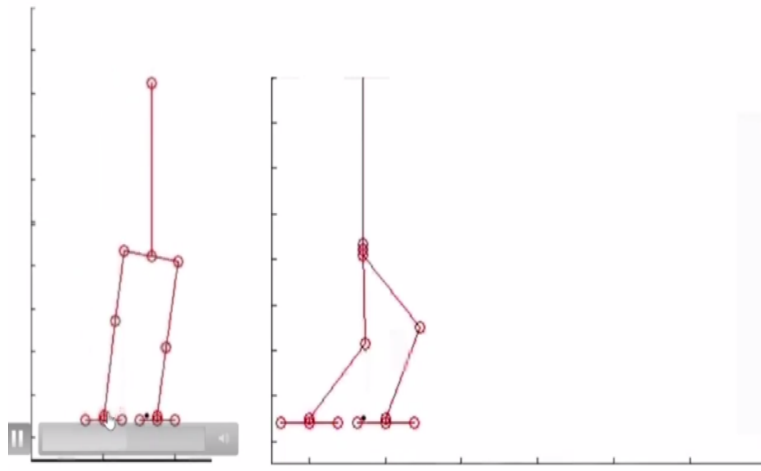
$$M_X^c + XG \times mg = \dot{H}_G + XG \times ma_G$$



Now what is balance? The balance is the constraint here. So, it basically means the robot should not fall down. What is the meaning of falling down? It basically means when it is standing on one foot, there should be no resultant moment here. So, no resultant moment. If there is a resultant moment anywhere, it will fall in that direction. Then how does the robot ensure that it bends on one side?

(Refer Slide Time: 48:06)

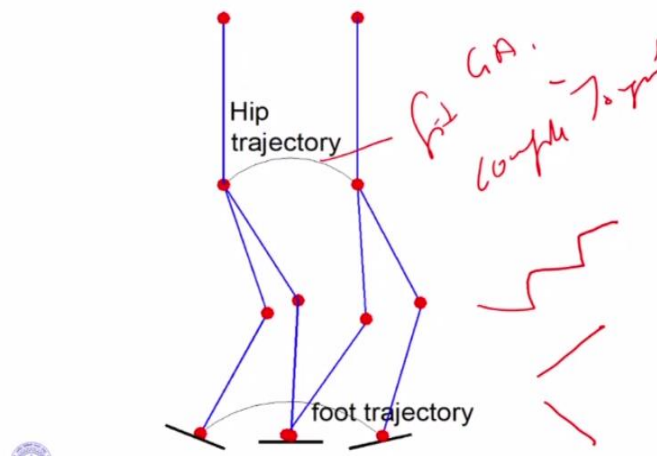
Motion in two planes



So when the robot bends on one side, when we walk with bend on one side that is essentially ensuring that this point called the 0 moment point is inside the foot, so we do not fall.

(Refer Slide Time: 48:15)

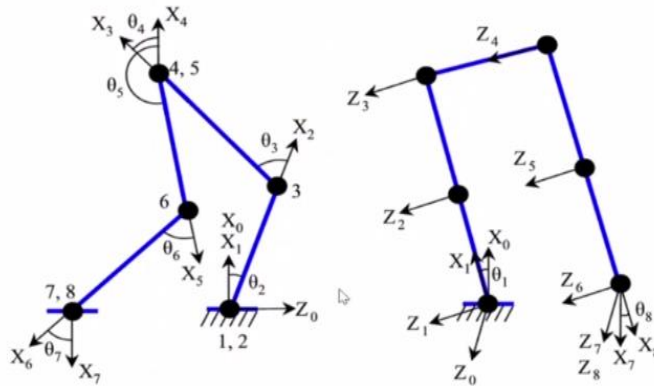
Forward motion of biped and inverse kinematics



Now that is the constraint. So, what is the way we solve this problem?

(Refer Slide Time: 48:19)

The Denavit-Hartenberg representation



Basically, we assign frames, a global frame and a local frame to each of the joints.

(Refer Slide Time: 48:27)

Hip and foot trajectory of the bip

Hip and foot trajectory is time parametrized cubic polynomials.
Splines knot vector \vec{t} is:

$$\vec{t} = [t_0, t_{d1}, t_m, t_{d2}, t_c]$$

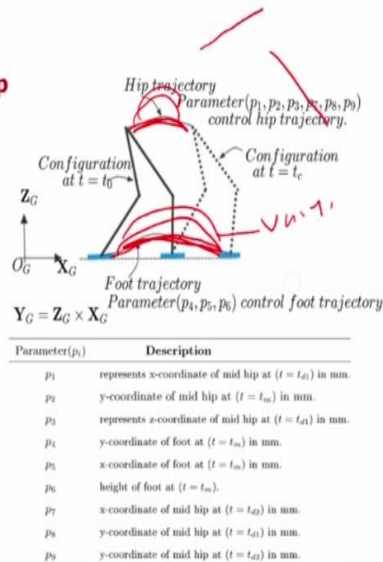
t_0 is start time of walk.
 t_{d1} is end time of (DSP-1).
 t_m is mid duration.
 t_{d2} is start time of (DSP-2).
 t_c is end time of the walk.

Optimal value of parameters

$$\mathcal{P} = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$$

is obtained by

optimization using genetic algorithm.



Then what we do is we find that if the robot has to move forward, there is a hip trajectory here and there is a leg trajectory for the robot to go forward. And you can have infinite trajectories like so if it is going on 3D what will happen is we have to vary this trajectory for different trajectories. For example, is going up a hill or is going down a hill what will happen is these trajectories will vary. So, what we can do is we can make a database.

(Refer Slide Time: 48:51)

Sample of Biped robot walking dataset

ML
DL
SUM

S.No.	S	L ₁	L ₂	ΔN	H ₁	H ₂	θ	φ	ψ	P ₁	P ₂	P ₃	P ₄	P ₅	P ₆	P ₇	P ₈	
1	1	93.7881	146.3118	-66.3829	27.7917	14.2356	0.2618	3.927	1.5708	0.58257	29.5385	303.428	-74.9505	-46.7407	0.55451	1.6809	-0.6942	12.725
2	0	122.221	144.2151	79.643	12.967	-37.3607	0.13063	0	4.7124	0.12761	-34.9353	180	43.1331	7.3761	0.82581	80.9746	-6.1683	-26.3795
3	1	51.6294	121.8075	-73.223	39.5285	-14.6423	0.2618	3.927	1.5708	0.57794	21.1941	339.2441	-72.9045	28.8711	0.57511	1.8892	13.5598	-21.5697
4	0	146.9517	71.8407	82.4094	-43.1007	0.81192	0.069813	5.4978	4.7124	-5.5154	134.2055	305.7247	42.0531	0.071149	0.59943	76.5843	15.2508	61.0084
5	0	132.4591	64.6563	76.1	33.2826	-48.7765	0.069813	1.5708	3.927	-33.8628	58.7834	259.6477	44.6265	68.7827	0.5	12.4567	-75.6052	-24.2835
6	0	40.4302	149.2564	64.4028	-13.297	30.1834	0.0174335	4.9785	4.978	-0.64393	-42.8165	334.5541	40	-8.2171	0.5	-56.1738	-0.77185	5.15
7	1	80.9032	115.1449	-42.3208	17.5524	-42.8403	0.13063	4.7124	0.7854	-7.9885	23.8866	221.5961	-56.5144	33.3424	0.51381	18.6139	-22.4867	71.3355
8	0	51.6336	145.3702	49.6654	15.7409	-45.5171	0.2618	3.927	1.5708	47.969	-93.4395	272.4543	54.1546	-47.8854	0.5	-2.4392	-77.735	-122.7397
9	0	138.5027	111.6976	79.4122	46.6562	14.8754	0.069813	0	3.927	-36.9658	-88.5041	322.9184	74.4407	-11.1407	0.84216	-31.1276	-36.2694	-105.326
10	0	104.4783	99.6654	81.372	-15.7012	-7.3527	0.13063	3.7854	2.3562	-36.0423	-98.7906	276.9617	79.1919	-32.0219	0.52843	-103.9607	-90.148	-99.4631
11	0	89.2968	33.5333	82.856	-32.6085	-49.533	0.069813	1.5708	0.7854	-8.4353	-146.4084	298.0288	40	-2.5392	0.5	1.248	-59.4975	-32.6105
12	0	1.2649	132.6832	45.0005	22.6114	19.863	0.2618	3.1416	3.1416	-75.9129	-98.829	297.5962	40	-69.3466	0.94735	55.5053	-33.6725	-59.0395
13	1	98.589	45.2429	-78.9644	24.1491	-34.9016	0.13063	5.4978	1.5708	-1.2173	29.3296	325.0992	68.5851	77.1007	0.5	0.017284	-4.227	14.8603
14	1	92.7205	51.6426	-79.4728	29.749	-33.5456	0.069813	3.927	1.5708	-37.9163	65.8479	318.4373	-78.03	-37.9527	0.3531	12.1226	0.5856	30.4069
15	1	110.7601	106.81	-112.7818	25.8541	26.0021	0.2618	3.927	5.4978	22.2412	77.3729	257.9825	74.8623	15.9184	0.75494	24.6773	68.357	51.4753

We can make a database by making the robot walk in different conditions. And this is the database basically which is saying what is the inclination of terrain, what was the variation of angles. And this would be a very large database and once you make the database, we have to learn it. Now to make the database for each gait, we have to optimize, how do we optimize? (Refer Slide Time: 49:11)

Optimization of Hip and Foot trajectory

Objective function :

$$\text{Work done: } W = \sum_{i=1}^{14} \int_{t_0}^{t_c} |\tau_i(t) \dot{\theta}_i(t)| dt$$

Constraint of balancing:

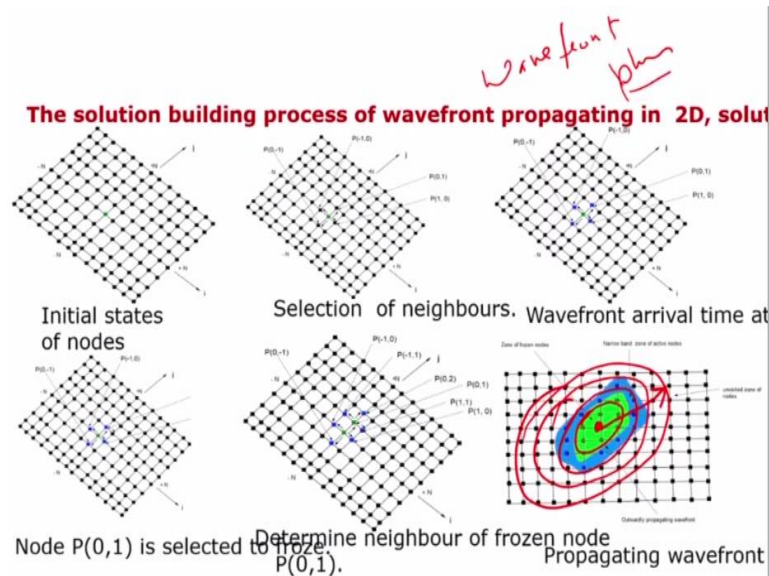
$$\text{ZMP constraint: } -L_{ab} \leq X_{ZMP} \leq L_{af}, \quad \frac{-L_{fw}}{2} \leq Y_{ZMP} \leq \frac{L_{fw}}{2}$$

- Its challenging to find the derivatives of the dynamic equation.
- This optimization problem is solved by Genetic Algorithm.
- It takes about 180 seconds to obtain energy-optimal and dynamically balanced gait for one step of walking.

We optimize by minimizing energy. So, we do energy minimization by doing the integral of torque into $d\theta$, so this is energy minimum. So, what I am saying here is that what GA will do here again is suppose it is a flat down, it is very easy to understand. Let us look at this figure, it is extremely easy to understand here. So, what GA will do is it will fit different trajectories. So, GA will fit different trajectories and will compute torques at joints by using dynamics. Then it will say that this trajectory gives you the minimum torque.

That means that is the energy optimal trajectory for this walk. By doing that for different terrain conditions, for example going upstairs, going downstairs, going upside, going downside GA can find what are the optimal hip and foot trajectories but energy optimal Using that it basically makes a database, this is the database which is showing the turning angle, going up, going down, what is the various parameters of the leg trajectory and the hip trajectory. Once it makes this database it has to learn it.

(Refer Slide Time: 50:16)



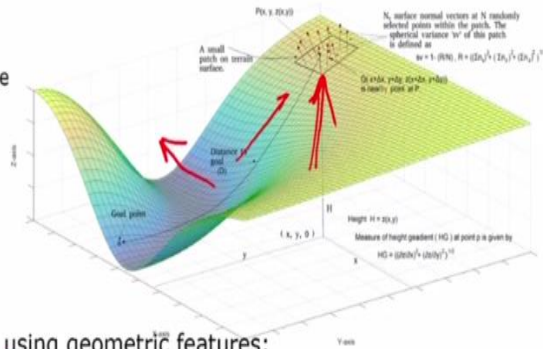
So, it learns this database by using some machine learning algorithms like neural net is example, deep learning is another one support vector machine. So, these can all be used to just learn the database and once you learn the database, next you have to go. So, this is an example of where we have used the wavefront planners. You have studied wavefront planners it is like dropping stone on the surface of a lake.

What will happen? Waves will be formed and it is going to go out like that. So it is like saying the robot is starting from here and is following the waves which are propagating in this direction.

(Refer Slide Time: 50:56)

Speed function based on geometric features of terrain

- Height map.
- Spherical variance
- Height gradient.
- Distance to go.

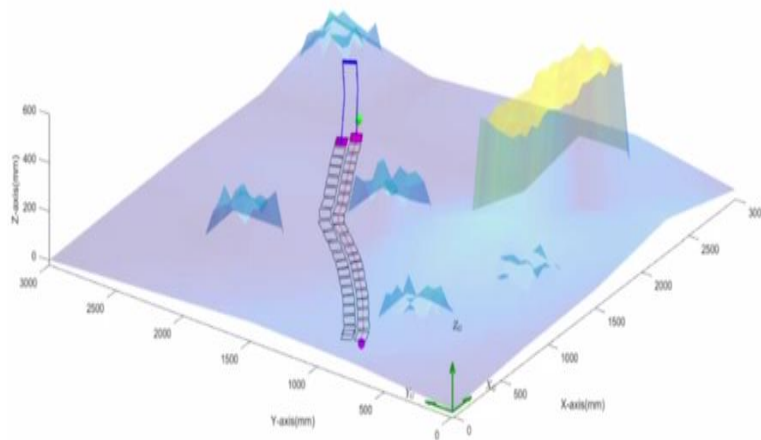


Speed function using geometric features:

$$F^{\text{geometric}} = f(HG, sv, H, D)$$

Now, we can say that the propagation of the waves can be a function of the gradient. In the case of a surface like this, if there is a high gradient here, so we are saying there is a gradient here. So, it is very high the wave will not go in that direction, it will go this side that is expected.

(Refer Slide Time: 51:11)



Biped robot traversing the path on uneven terrain.

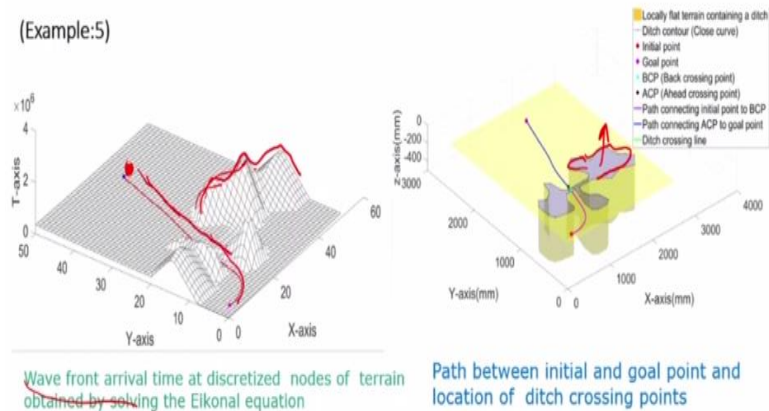
So by making this kind of a graph or making this kind of a surface, we can make the robot walk using a wavefront propagator.

(Refer Slide Time: 51:18)

Simulation Results for ditch crossing

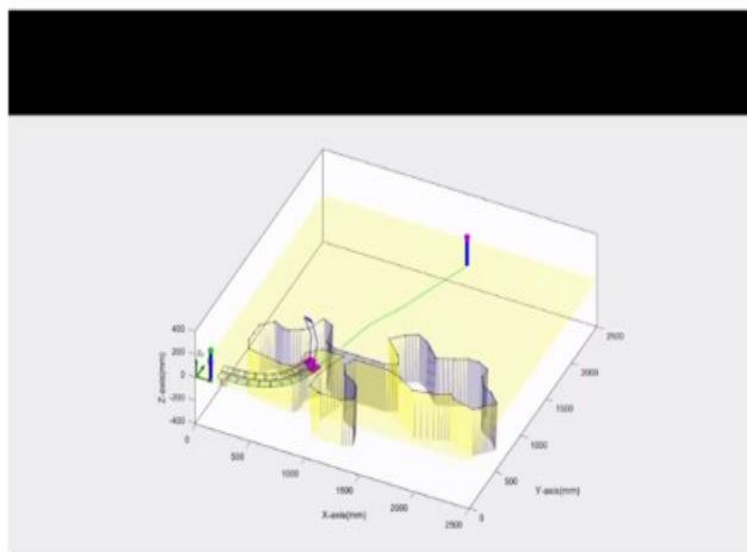
- Path and footsteps between initial point and goal point separated by an irregular shape ditch is planned for 14 DOF biped robot on flat terrain.

(Example:5)



See for example here, this is my wavefront arrival time, this is obstacle. So in this case, there is a ditch here, we can see there is a ditch. So, this ditch is coming up in the form of a high potential here. That means the wave will not go that side because it is high, so the wave which is propagating from here, this is my start point of the robot, the wave is propagating this way, so it will always try and keep it a low ground.

So you can see here that by using the wavefront panel, the robot is walking. **(Video Starts: 51:47)** So here you can see the robot is walking from one point to another point simply by trying to follow the flat ground. It is exactly the same as the wavefront planner that we studied. So, the waves try to keep to a flat ground, they do not try to climb gradients. So, this is a case of biped robot walking on 3D terrain. **(Video Ends: 52:07)**
(Refer Slide Time: 52:09)



Now we can have biped robot, somehow this is a ditch, so this place is a ditch. So, it should not go into the ditch. So the ditch manifests itself by a very high gradient. So the robot avoids that high gradient and tries to find a location where it can cross by using the wavefront planner and then crosses at that point and goes on the other side. So this was the last class of course robot motion planning.

I hope that you have got an idea of robot motion planning, especially if you are looking at applications of robots. And I would encourage you to take other courses in robotics also, introduction to robotics to give you an introduction, then AI in robotics. Now, I hope that you will be able to apply whatever you have learned in this course and then either do research or if you are in industry to be able to use it in industry. I hope this course has been useful. So, see you again in later courses. Thank you.