INTELLIGENT CONTROL OF ROBOTIC SYSTEMS

Prof. M. Felix Orlando

Department of Electrical Engineering

Indian Institute of Technology Roorkee

Lecture 18: Introduction to Path Planning

Good morning, everyone. Today, we are going to have a lecture on Introduction to Path Planning. What is path planning? Path planning involves determining a feasible route from a start to a goal while avoiding obstacles, which is crucial for autonomous systems. It is essential for autonomous systems to operate independently without human assistance.

Path planning ensures safe navigation in complex environments while minimizing time, distance, and energy consumption. Path planning is fundamental for robotic systems to perform tasks in structured and unstructured environments, whether on land, air, sea, or space. Coming to the applications of path planning, it is applicable for route planning in self-driving cars, considering traffic obstacles and road conditions, which are highly dynamic in nature. Flight path optimization in 3D space for tasks like delivery, surveillance, and mapping. Then, efficient movement of robotic arms in manufacturing, minimizing collisions and optimizing cycle time.

The figure here shows the workspace in recent robotic path planning competitions, where the obstacles and the target locations are highly dynamic. Evolution in path planning methods first occurred in the year 1956, when Dijkstra introduced the shortest path planning algorithm. This laid the foundation for later developments. Next, in the year 1968, Peter Hart, Niels Nilsson, and Bertram Raphael combined Dijkstra's algorithm with heuristics for improved efficiency. Then, in the year 1985, Lydia Kavraki and Jean-Claude Latombe enabled the path planning algorithm in high-dimensional spaces using sampling-based methods. Then, in the 1990s, Steven M. LaValle facilitated real-time path planning in complex environments. Next, in the year 2008, Sertac Karaman and Emilio Frazzoli ensured optimal paths in asymptotic conditions with

Rapidly Exploring Random Tree, RRT Refinement. In the year 2013, Sergey Levine used deep reinforcement learning for complex navigation tasks. Now let us see the types of

path planning. First one is single agent path planning. And it involves finding a path for a single autonomous agent from a start to a goal location.

while avoiding obstacles example is determining a route for a mobile robotic system in a warehouse and the challenges involved here are in handling dynamic obstacles or navigating through complex or high dimensional spaces next we have multi-agent path planning And this multi-agent path planning addresses path planning tasks for multiple agents that must coordinate their movements to avoid collisions and reach their goals. An example here is in coordinating multiple drones for simultaneous delivery tasks. And the challenges involved are in avoiding obstacles, or avoiding the collisions between the agents and ensuring efficient coordination and communication between or among agents.

Next, we have static path planning. Static path planning focuses on path planning in environments where obstacles and goals do not change over time. Example solving a maze with fixed walls and goal locations. Solving a path for a mobile robot to reach a fixed target while avoiding obstacles where the target is fixed static in nature and the obstacle locations are also static in nature. The challenges in this case of static path planning

In optimizing the chosen path, whether the path obtained through the static path planning approach is optimal or not. This is the biggest challenge. Next, we have dynamic path planning. It involves planning paths in environments where both obstacles and target locations can change over time. The severity lies in multiple obstacles that are dynamic in nature and multiple targets that are also dynamic.

For example, navigating a car through traffic where other vehicles and pedestrians move unpredictably. The challenges lie in adapting to real-time changes and uncertainties, as well as re-planning and adjusting paths on the fly. Another example of dynamic path planning is robotic smart needle interventions for minimally invasive surgical applications. Here, we have the initial point where the needle can enter, and there is a target location inside the human body.

The flexible bevel-tip needle, where the bevel angle (theta) could be 30 degrees or 45 degrees in certain applications. This flexible needle can enter through the entry point, which is the initial point, and by avoiding obstacles, it can be steered to reach the target. This path is formed to avoid dynamically moving obstacles as well as the target region, because this environment is inside the human body and consists of tissues, fluids, blood

vessels, and bones. There is a high chance of these obstacles changing, as well as the target changing over time. As you can see here in this simulation,

This is how it is. The needle enters here, and the obstacles keep changing dynamically over time. The needle tip moves towards the path found optimally without colliding with the dynamic obstacles and reaches the target. In real time, the target also will change location due to the interactive forces caused by the needle inserting into the tissue domain. As you can see, by avoiding the dynamic obstacles, the needle is inserted to reach the target. Here, we have a static target, but in real time, the target may also vary because of dynamic tissue movement.

Now, let us revisit the search algorithms that we have seen in the previous class quickly. We have studied three types of forward search algorithms. Breadth-first search method, depth-first search method, and A* search method, where first-in-first-out and last-in-first-out are the approaches or strategies used in the first two techniques. A* outperformed these two techniques that we have seen. Now, sophisticating the priority function f(x) makes a general search problem into a path planning problem.

Because the priority function—how we design it—is the key strategy in these three search algorithms. Now, by sophisticating this priority function f(x), we can turn a general search problem into a path planning problem. Among these three approaches, A* has the most sophisticated and optimal priority function. Now, let us learn about A* search. A* search is not just interested in finding paths but also the optimal path.

The star is meant for indicating optimal path. or the optimal solution obtained from this algorithm. For that we use heuristic function h of x along with the cost function g of x. So, the next point. is given by $x' = f(x) + h(x)$

case the new state can be obtained by the inclusion of f(x) and h(x) or you can see that precisely it is the heuristic function and the cost function f(x) let us say this is f(x) or g(x) being the cost function both considering the cost as well as the heuristic the next state is obtained in the a star approach In most of the cases h of x is the Manhattan distance from the current location to the goal location or by expression it is given by

$$h(x) = |x_{\text{goal}} - x_{\text{current}}| + |y_{\text{goal}} - y_{\text{current}}|$$

Now let us talk about the greedy search. In path planning. A search algorithm. That selects the most promising node. Based on a heuristic.

Evaluation to reach the goal. As quickly as possible. The greedy search works in such a way that From the initial point, you reach a node. Let us say from that, you try to reach the goal as quickly as possible based on a heuristic evaluation. This is called the greedy search approach, and here we are minimizing the cost or estimated cost from the current state to the goal using a heuristic function h(x). And this approach, unlike the A* approach, here we have $f(x) = h(x)$ always.

The example here is, let us say we have to find the shortest path on a 2D grid from the location (0, 0) to the goal location (5, 5). This is (0, 0) location, and (5, 5) location is the goal location. Initially, h equals 10 here, and for (1, 1), h becomes 8. The heuristic function takes the value 8, which is the least possible value. Accordingly, it reaches here where h becomes 6, and here it reaches where the heuristic function takes the value h equal to 4, then 2 accordingly.

Similarly, it will go to (2, 2) after that and so on. That is the way it reaches. Precisely, it will look something like in this figure. It started from the initial location and step by step reaches the goal location, which indicates that it is not only finding a path which is the general approach to reach the target, but also it may show that the path is the optimal path.

The algorithm here has the structure of a general search algorithm, where the path obtained is the optimal path to reach the target location. Now, let us compare the greedy search approach and the A-star search approach in the perspective of path planning. A-star is generally more reliable for finding the optimal path than the greedy search approach, given an admissible heuristic. Greedy search can be faster than in some cases because it only considers the heuristic and does not keep track of the path cost.

However, this comes at the cost of potentially missing the optimal path. Both algorithms, greedy as well as A-star approaches, rely on heuristics and but the A-star approach uses a more comprehensive evaluation function, making it less dependent on the heuristic alone. Now, let us see the potential fields method—the introduction to this approach. It uses artificial potential fields to guide the

robot from the start location to the goal location. As you can see from the schematic, this is the work volume of the robotic system. The robot is somewhere in the target location and has to reach the initial location, then proceed to the target location. The obstacles are given higher potential fields, so the robot will follow a path avoiding the higher potential field areas. Thereby, it forms a path between the obstacles to reach the target location.

Precisely, the potential fields method uses artificial potential fields to guide robotic systems. to maneuver over the path.

To reach the target. Here, an attractive force. Is generated. Towards the goal. And a repulsive force.

Is generated. To move away from the obstacles. Near the obstacles. Once the robot is moving. It will have a repulsive force.

From the obstacles. So that the robotic system. Avoids the. Collision with the obstacles and towards the goal, the robot finds an attractive force to reach the target. The attractive fields pull robotic systems toward the goal, whereas the repulsive fields push the robot away from the obstacles.

So, we model here the attractive and repulsive potential fields as follows, which are the key attributes for this algorithm. So, the attractive potential field is given by $U_{\text{att}} = \frac{1}{2} k_{\text{att}} \|x - x_{\text{goal}}\|^2$

and the repulsive potential field $U_{\text{rep}} = \frac{1}{2} k_{\text{rep}} \left(\frac{1}{\|x - x_{\text{obs}}\|} - \frac{1}{d_0} \right)^2$

where x is the current location, x_goal is the goal location, and k_att and k_rep are the gain factors used in attractive potential field computation and repulsive potential field computation, respectively. And d_0 is the distance at which the repulsive force or repulsive potential starts to take effect.

Therefore, the net potential field here is given by $U_{\text{total}}(x) = U_{\text{att}} + U_{\text{rep}}$

Now, we can calculate the lines of forces, which will be the paths from the initial location to the goal location for the robotic system. And these lines are given by the formula

$$\boldsymbol{F}(x) = -\nabla U_{\text{total}}(x, y, z) = -\left(\frac{\partial U_{\text{total}}}{\partial x} \hat{\boldsymbol{x}} + \frac{\partial U_{\text{total}}}{\partial y} \hat{\boldsymbol{y}} + \frac{\partial U_{\text{total}}}{\partial z} \hat{\boldsymbol{z}} \right)$$

So, the advantages of this approach are that it is simple and it facilitates real-time path planning for robotic systems and other autonomous systems, and it is flexible for dynamic environments where both obstacles and multiple targets are dynamic in nature, meaning their locations vary with time. And the disadvantages are that we can get stuck in a local minimum instead of reaching the global minimum. This approach requires careful tuning of the parameters associated with the update algorithm or the update law. Now, coming to the introduction to sampling techniques.

The definition of sampling-based path planning is such that Sampling techniques involve generating random samples in the environment to explore possible paths. Upon exploration, depending on the conception of time, we eventually reach the target. The purpose of this sampling-based path planning approach is that It is used when the environment is too complex or unknown for exact methods.

And it helps in finding feasible paths by randomly exploring the space. Random sampling. Let us talk about random sampling now. The concept of random sampling is that Generate random points in the search space and connect them to form a path, depending on the current situation, to reach the target while considering optimality and avoiding obstacles. The process here is to randomly select points within the workspace, check connectivity between points, and form a graph.

The usage of this approach of sampling or random sampling is that it is useful in simple environments where a complete map is not available. Now, let us have a brief introduction to RRT, which is a rapidly exploring random tree. The concept here is to incrementally build a tree by randomly sampling and expanding nodes. That means we start from the goal.

And then select a node. Connect it. And search for node which is closer to that. And connect it. And select some other node.

And see the distance which is closer. Connect it. Then select a node and select another node and see whether which is closer. Connect it. Thereby developing the tree to reach the target randomly.

The nodes are selected randomly and by having a metric check we can connect this nodes together to form a branch of the tree to finally reach the target. The process here is that start from an initial point and grow the tree by randomly sampling the space. Connect new nodes to the nearest existing node. The advantages of this RRT approach is that it is efficient in high dimensional spaces.

And can handle complex and high-dimensional environments easily. Now, having seen a brief overview of RRT, let us talk about RRT*. As the name indicates, the star means that the path found by this approach is optimal in nature. The concept here is: An improved version of RRT that optimizes paths by rerouting and connecting nodes.

So, we have a target here. We have the initial point here. And these are the obstacle locations. So, the process here is similar to RRT but focuses on refining paths for

optimality. So here, it goes for the random node, and from the first random node, it searches the nodes randomly and selects one based on the closeness to this node. Then it continues, and by having the nearby node selected, they have a radius around that node.

And in that radius around that node, we have so many random nodes selected. And see whether from this node to this, and from here to here, or from here to here. So, connecting this node to the nearest node will form the path: this and this and this. But this algorithm checks what is the path here and here because this path, say P2, and this path, say P1, the total length of this path is compared, and the minimum path is taken. By neglecting this, like that, another random sample point or the node is selected, around which the same search region is put upon, and then so many nodes.

From that, we choose one node and see how it is getting connected to this. Now, the path is here. This way to reach this or this. Accordingly, the shorter path is chosen. Likewise, this path is continued toward the target or the goal region.

By having the selected space region around the newly chosen random node. So, the advantages of this approach are that it provides better path quality compared to the basic RRT. And it balances exploration and optimization. In this way, it reconnects the nodes to improve path quality. The path that is obtained here is further tuned to provide a much more optimal path.

Now, We see the introduction to probabilistic roadmaps (PRM). As can be seen here in the graph, this approach builds a roadmap by randomly sampling points and connecting them to form a graph, as shown in the schematic. The process involved here is sampling points and connecting them to create a graph. Use graph-based search algorithms to find paths from this graph.

The advantage of this approach is that it provides better path quality compared to the basic RRT. Probabilistic roadmaps effectively handle complex high-dimensional environments by creating a roadmap through random sampling. Yes, so as a conclusion to this lecture, We have seen the introduction to path planning generally, and then we have seen various path planning approaches involving probabilistic roadmaps, rapidly exploring random trees, RRT*, A*, greedy search approaches, and how a search approach becomes a path planning approach by involving heuristic approaches. We have seen a couple of applications briefly toward mobile robot maneuvering to reach a target, and also we have seen a simulation of a robotic needle reaching the target in a drastically changing dynamic environment inside the human body. In the next class, we will be

seeing an elaborate lecture on RRT, that is, rapidly exploring random trees, and probabilistic roadmaps with applications in minimally invasive surgical robots. Thank you so much.