Path Planning For Autonomous Vehicles Dr. Assadian - Future Mobility Lab

Jonathan Dorsey *Member: No Sleep Club: est. 2017*
https://github.com/JonnyD1117/ros2-py-path-planning-server

***Abstract*—The objective of this project is.....???**

***Index Terms*—A*, Dijkstra, PRM, RRT, path planning, mobile robots, autonomous vehicles**

I. Introduction

THE world of robotics is full of constraints, demands, and performance trade-offs that humans handle naturally on a daily basis.

II. Path Planning for Mobile Robots

A. Motivation

The motivation for path planning in the domain of mobile robotics becomes very apparent when naively asking the questio of "How do we get a mobile robot to get from a starting location to a end location?".

Before any control actions can be taken, the robot must first determine its path which can then be fed as an input into the which ever control scheme the robot implements.

The goal of path planners is to systematically break down the problem of where the robot should go next in a computationally tractable, physically possible, and desireable way.

III. Occupancy Grid Maps

The first in any in any path planning problem is to define the domain over which a path is to be planned. One of many solutions used to make this a computationally tractable task is to the partition space into a finite number of discrete cells, rather than planing directly from the continuous configuration space, which is often of either two or three dimensions.

One of the most popular approaches to spatial discretization is that of the **occupancy grid map**. In this formulation, a configuration space is discretezed into a uniform 2D/3D grid. Each cell or pixel holds a single value between $[0, 1]$, which represents the probability at which that cell is occupied. A value of zero would indicate that the cell is 100% free and does not contain any objects or obstacles, where as, a value of one indicates the cell is 100% occupied and contains no free space. Naturally, it follows that values between zero and one indicate the probability that the cell is more of less occupied.

A. Assumptions & Tradeoffs

It is important to note that the assumptions and compromises that applying such a discretization scheme imposes upon the path planning algorithms which must use this grid map as an input.

1) Benefits: The first benefit of using these grid maps is there spatial effiency. Due to their ability to be store as either a 2d grid of cells (exactly like a picture is comprised of pixels), or a 3d volumetric tensor of cells, these grid maps can efficiently be stored into a computers RAM and queried.

The second benefit of grid maps comes from the fact these maps can literally be stored on a computer as a gray-scale image.

2) Limitations: The first limitation of grid maps comes directly from the process of uniformly discretizing the configuration space. By definition grid maps degrade the resolution and fidelity of the configuruation space by forcing its representation is an irreversible lossly manner. Much like the the discretization of analogue to digital signals, discretization will indeed degrade the fideility of the original signal; however, it is often sampled at a high enough frequency to yield the resulting losses insignificant when considering the gains achieved by being able to encapsulate and query finite data-structures with high effiency.

This implies that by using grid map, we are necessarily missing information that exists in the actual configuration space which which is too fine to be resolved into our grid map. This problem can usually be addressed by creating a grid map of high enough density that any aliasing if minor enough to be ignored by the path planning algorithms that use that map.

B. Grid Map Generation with Lidar

In general the problem of creating a occupancy grid map is handled by the Simulaneous Localization & Mapping (SLAM) algorithm which is being used. However, for explanations sake, we can illustrate the creation of a grid map in a static environment with a static planar lidar sensor.

1) Invserse Sensor Model:

2) Bresenham's Line Algorithm:

IV. Path Planning Algorithms

This section will cover several of the most popular path planning algorithms such as A, Dijkstra's Shortest Path,

Probablistic Road Maps, and more. The objective of each of these algorithms is to leverage different computational tools that are available to us such that we can extract (in finite time) a feasible path that the robot could physically travel to obtain its goal.

A. Graph Search

The first categories of algorithms to investigate are among the most popula. These are graph search algorithms. The primary mechanism of these planners is to construct a graph of the configuration space. Once this has been achieved different algorithms, heuristics, and operations can be applied to the graph to endeavor to find the shortest path between two nodes that exist in that graph.

Occupancy grid maps are ideal for the such path planners in that it is trivial to translate the grid map into a graph data structure, with only a few simple rules for legal moves between cells of the grid map. The typical connectivity of these spatial graphs are either **connected-4** or **connected-8**.

1) Dijstra's Shortest Path: The grandfather of almost all graph search planners is Dijkstra's shortest path algorithm. This algorithm implented by Dutch mathematician Edsger Dijkstra in Norweh in 1956. This algorithms implements a uniform cost best first search that will traverse weighted undirected graph, and extract the shortest path between two nodes.

This is notable as it was the first time that this abstraction of a graph as a model of the configuration space of a robot had been used for the purposes of path planning.

For our implementation, Dijkstra's algorithm uses the grid map to generate a dense graph stucture of the configuration space. Grid map cells that have an occupancy value over a certain threshold are assumed to be occupied and the edges connecting an occupied node to a free node are deleted, thus making the obstacles untraversable on the graph and ensuring that the path planning will not use these nodes when searching over the best path for the planner to generate.

2) A:* A is an algorithm that was developed at the Stanford Research Institute in 1968, for the historically significant robot Shakey, being used in some of the earliest pioneering research into mobile robotics.

A is the spiritual successor to Dijkstra's Algorithm in so much as it too implements a graph search. The primary improvement that A* adds is the implementation of a heuristic to improve the ability of the graph to improve its search direction according to the given heuristic.

Previous with Dijkstra's algorithm, the graph is searched in a breadth-first manner, this means that effectively the entire graph must be search (until the goal node is reached). The addition of a heuristic proves the planner with the ability to preferentially search in a given direction that will traverse nodes that are more likely to be approaching the goal node fom the starting node. So long as the heuristic is quick to compute and admissible (an exact or under estimate of the actual distance to the goal node), we can use heuristic search to drastically improve the planners convergence rate.

B. Probabilistic Road Maps

Probablistic road maps (PRM) is a the combination of two techniques, random sampling, and Dijkstra's algorithm. The objective of PRM is to decrease the density of the graph structure, but uniformly sampling the configuration space and creating a graph from these samples, instead of creating a pixel level accurate graph as represented by our grid map.

This technique is not only faster for the graph generation porition of the algorithm, but since it's graph is orders of magnitutude less dense than the graph that we generated for A* or Dijkstra we can apply either of these algoithms over this "sampled" graph with the result that traversing this graph is MUCH faster due to decreases node density of the graph mentioned previously.

C. Rapidly Exploring Random Trees

Similar to PRM, Rapidly Exploring Random Trees (RRT), also attempt to leverage Monte Carlo method of random sampling to improve the sample efficiently and reduce the graph generation time. However, instead of creating a graph, RRT's create a tree data structure.

This structure has the benefit of being incredibly easy to determine the shortest path from the root of the tree to the desired goal node, (so long as a path actually exists).

1) RRT: Unlike the previous algorithms review so far, RRT is not garunteed to find the shortest path between two nodes even in that limit that we infinitely sample the configuration space. This is due to the static nature of the tree generation.

2) RRT:* Unlike RRT, RRT* is garunteed to find the shortest path in the limit, due to it's improved mechanisms of leaf generation and lowest cost tree re-wiring.

V. Path Smoothing

It should be noted that in our usecase, the paths that all of our planners have generated are piece-wise linear. This means that with the edge case of a singular straight line path, the shortest path generated by our path planners will always be discontinuous.

This fact is a consequence of our choice to discretize the configuration space into a discrete occupancy grid map. Unforunately, our robotic systems generally possess several incompatibilities with this sort of discontinuous shortest path.

The first incompatibility is that many robotic systems implement some form of continuous control algorithm or scheme and secondly even if a discrete path is desired, it is often more desirable to have a much higher sampling frequency of this path than the direct output of the path

planner.

This leads to the need to implement path smoothing over the set of points which our planner has output.

A. Bezier Path Smoothing

One of the most simple and straight forward method for path smoothing is that of Bezier curves. While the mech behind the Bezier curve is beyond the scope of this paper, sufficed to say that Bezier curves allow for simple and direct mathematically description outputting an infinitely resolvable contiuous curve from a list of discrete points.

This concept of path smoothing is simple and direct; however it suffers from the inability to applied custom constraints to the path smoothing that garuntee feasibility in the configuration space.

B. Optimzation Based Path Smoothing

Unlike Bezier path smoothing, optimizations based smoothing works on the principle of minimizing some continuous cost function associated with the points output by the planner. This methid is far more computationally expensive and far less intuitive than the implementation of Bezier curves; however, as with most optimizations problems it can be configured as a constrained optimization to obey certain required physically limiations/constraints such as minimum allowable curvature.

VI. Way Point Generation

VII. Acknowledgment

References

[1] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.