Path Planning

September 6, 2023

1 introduction:

Path planning is the most important issue in vehicle navigation. It is defined as finding a geometrical path from the current location of the vehicle to a target location such that it avoids obstacles. This path must be navigable by the vehicle and optimal in terms of at least one variable so that it can be considered a suitable path. For different target distance situations, the smoothest path, the shortest path, or the path along which the vehicle can move with the highest speed can become the most important path. In other words, the optimal path is determined concerning these characteristics.

Path planning lets an autonomous vehicle or a robot find the shortest and most obstacle-free path from a start to goal state. The path can be a set of states (position and/or orientation) or waypoints. Path planning requires a map of the environment along with start and goal states as input. The map can be represented in different ways such as grid maps, state spaces, and topological roadmaps. Maps can be multilayered for adding bias to the path.

Path planning is one of the most crucial research problems in robotics from the perspective of the control engineer. Many problems in various fields are solved by proposing path planning. It has been applied in guiding the robot to reach a particular objective from very simple trajectory planning to the selection of a suitable sequence of action. Path planning cannot always be designed in advance as the global environment information is not always available a priori. By proposing a proper algorithm, path planning can be widely applied in partially and unknown structured environments.

2 Types of Path Planning Algorithm:

Path planning algorithms are differentiated based upon available environmental knowledge. Often, the environment is only partially known to the robot/AV.

An effective path planning algorithm needs to satisfy four criteria:

1. the motion planning technique must be capable of always finding the optimal path in realistic static environments.

2.it must be expandable to dynamic environments.

3.it must remain compatible with and enhance the chosen self-referencing approach.

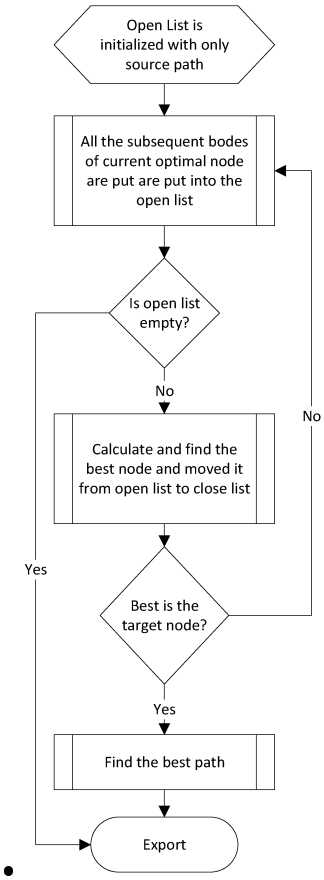
4.it must minimize the complexity, data storage, and computation time.

2.1 Dijkstra Algorithm:

The Dijkstra algorithm works by solving sub-problems computing the shortest path from the source to vertices among the closest vertices to the source [8]. It finds the next closest vertex by maintaining the new vertices in a priority-min queue and stores only one intermediate node so that only one shortest path can be found.

Dijkstra finds the shortest path in an acyclic environment and can calculate the shortest path from starting point to every point. We find many versions of Improved Dijkstra's algorithm, this is based on specific applications we find around us. Every Improved Dijkstra's algorithm is different, to reflect the diversity of use cases and applications for the same. Variants of the improved Dijkstra's algorithm are discussed in this article.

The traditional Dijkstra algorithm relies upon a greedy strategy for path planning. It is used to find the shortest path in a graph. It is concerned with the shortest path solution without formal attention to the pragmatism of the solution. The modified Dijkstra's algorithm aims to find alternate routes in situations where the costs of generating plausible shortest paths are significant. This modified algorithm introduces another component to the classical algorithm in the form of probabilities that define the status of freedom along each edge of the graph. This technique helps to overcome computational shortcomings in the reference algorithm, supporting its use in novel applications.



Another improved Dijkstra algorithm reserves all nodes with the same distance from the source node as intermediate nodes, and then searches again from all the intermediate nodes until traversing successfully through to the goal node, see Figure 1. Through iteration, all possible shortest paths are found and may then be evaluated.

2.2 A* Algorithm

The A* Algorithm is a popular graph traversal path planning algorithm. A* operates similarly to Dijkstra's algorithm except that it guides its search towards the most promising states, potentially saving a significant amount of computation time . A* is the most widely used for approaching a near optimal solution with the available data-set/node.

It is widely used in static environments; there are instances where this algorithm is used in dynamic environments. The base function can be tailored to a specific application or environment based on our needs.

A* is similar to Dijkstra in that it works based on the lowest cost path tree from the initial point to the final target point. The base algorithm uses the least expensive path and expands it using the function shown below

$$f(n) = g(n) + h(n)$$

is the actual cost from node n to the initial node, and h(n) is the cost of the optimal path from the target node to n.

The A* algorithm is widely used in the gaming industry, and with the development of artificial intelligence, the A* algorithm has since been improved and tailored for applications, including robot path planning, urban intelligent transportation, graph theory, and automatic control. It is simpler and less computationally-heavy than many other path planning algorithms, with its efficiency lending itself to operation on constrained and embedded systems.

The A* algorithm is a heuristic algorithm that uses heuristic information to find the optimal path. The A* algorithm needs to search for nodes in the map and set appropriate heuristic functions for guidance, such as the Euclidean distance, Manhattan distance, or Diagonal distance. An algorithm is governed by two factors for efficiency resources used for performance of the task and response time or computation time taken for performance of the task.

There is a trade off between speed and accuracy when the A* algorithm is used. We can decrease the time complexity of the algorithm in exchange for greater memory, or consume less memory in exchange for slower executions. In both cases, we find the shortest path. One simple application for the A* algorithm is to find the shortest path to an empty space within a crowded parking lot.

In the hierarchical A* algorithm, path searching is divided into few processes, the optimal solution for each process is found, and then the global optimal solution is obtained, which is the shortest path to the empty spot. Another improvement is done on this algorithm, with the improved hierarchical A* algorithm using the optimal time as an evaluation index, and, introducing a variable, a weighted sum is used along with the heuristic:

$$fw(n) = (1 - w)g(n) + w.h(n)$$

The weighted co-efficient is defined with respect to each application and in most cases it might be used to represent linear distance (e.g., Euclidean distance). Improved hierarchical A* is claimed to have improved efficiency and precision. Another variation of this algorithm is used in valet parking where the ego-vehicle (cars that usually focus only on their local environment and do not take into account environmental context) has a direction of motion d that depends on the speed, gear, steering angle, and other parameters of the vehicle kinematics.

This algorithm is called the Hybrid A* Algorithm. The Hybrid A* search algorithm improves the normal A* algorithm when implemented on a non-holonomic robot (e.g., autonomous vehicles).This algorithm uses the kinematics of an ego vehicle to predict the motion of the vehicle depending on the speed,

gear, and steering angle, and other parameters of the vehicle will add costs to the heuristic function. An improved version of Hybrid A* was introduced to the same application with the help of a visibility diagram path planner.

A visibility diagram was one of the very first graph-based search algorithms used in path planning in robotics. This method guarantees finding the shortest path from the start to the final end position. The start, end, and obstacle positions are fed into as input to the Hybrid A* algorithm where A* is run on the results of the visibility diagram, which then provides the optimum distance. This is called the Guided Hybrid A* algorithm

3 local planner and global planner:

3.1 Global Path

Path planning may be either local or global. Global path planning seeks an optimal path given largely complete environmental information and is best performed when the environment is static and perfectly known to the robot. In this case, the path planning algorithm produces a complete path from the start point to the final end point before the robot starts following the planned trajectory [6]. Global motion planning is the high level control for environment traversal.

3.2 Local path

local path planning is most typically performed in unknown or dynamic environments. Local path planning is performed while the robot is moving, taking data from local sensors. In this case, the robot/AV has the ability to generate a new path in response to the changes within the environment. Obstacles, if any exist, may be static (when its position and orientation relative to a known fixed coordinate frame is invariant in time), or dynamic (when the position, orientation, or both change relative to the fixed coordinate frame)

4 Challenges and Future Trends in Path Planning

Apart from current regulations that prohibit drones' operation beyond the visual line of sight, safety concerns in instances of mechanical failure, and drone navigation in a GPS-denied environment, there are other limitations that need to be considered when developing a practical path planning algorithm. In this section, we go over four major limitations: the limited battery capacity of drones, onboard computational capabilities of drones, extreme weather conditions, and dynamic environments.

4.1 Energy Constraints

Although rotary-wing drones are highly maneuverable and versatile, they are still limited in terms of endurance. Thus, a single rotary-wing drone cannot be used for missions requiring long flight distances. Many factors can affect the battery performance of drones. Many maneuvers, such as hovering, horizontal and vertical movements, payload, speed, and flying in the wind direction, can impact drones' battery performance at varying levels. In many studies, it is often assumed that drones can navigate through a computed path on a single charge. However, in practical settings, drones may often need to stop and recharge in order to navigate to their destination.

Improvements have been made to reduce the energy consumption of drones. The overall weight of rotary-wing drones has been reduced by using carbon-fiber airframes. There have also been improvements in brushless direct-current motors' power-to-weight ratios, which contributes significantly to energy consumption. Since there are weight constraints to drones, additional batteries or batteries with larger capacities cannot be placed on a drone to improve its endurance. In, it has been shown that there is an optimal value of mass on a rotorcraft. Increasing a drone's weight after such a point begins to reduce the drone's endurance and causes the drone to become unstable.

4.2 Weather Conditions

An important factor to consider for drone path planning is the potential for navigating under extreme weather conditions. Although there are many improvements being made in the design of drones to make them more durable for such conditions, environments with precipitation, fog, and severe wind can have an adverse effect on a drone's endurance, visibility, and sensors for navigation and collision avoidance. Recently, there have been a few studies that have proposed solutions for such weather conditions. Some studies have proposed solutions where drones can evaluate risk and avoid regions that may have severe weather conditions when generating a path. Others, such as in, suggest using the Euler method to perform a numerical analysis to predict the trajectory of drones under different kinds of wind conditions. In, the authors suggest tilting a drone based on different wind directions can significantly help reduce the deviation of the camera on the drone when tracking the movement of a ground object. Finally, for vision-based path planning algorithms which rely on cameras to avoid obstacles, fog can severely reduce visibility.