Path Planning Algorithims

September 2023

1 Introduction

The use of mobile robots is growing every day. Path planning algorithms are needed to allow the coordination of several robots, and make them travel with the least cost and without collisions. With this emerged the interest in studying some path planning algorithms, in order to better understand the operation of each one when applied in this type of robots.

2 Types of Path Planning Algorithms

2.1 A*

The A star (A*) is a search method that uses a heuristic function, h(n), where n represents a node n. To each node n is associated an approximation h(n) of the cost of a path from n to a goal node, while h*(n) corresponds to the real distance (cost) from n to a goal node. A heuristic h is consistent if and only if: (i) h(n) = 0 (if n is the goal node); and (ii) for all nodes and their successors n 0 , the estimated cost of moving from node n to the goal node is not greater than the cost of moving from node n to node n 0 plus the estimated cost of moving from node n 0 to the goal node, as can be seen in Equation $h(n)6c(n, n0)+h(n0)$

A heuristic h is admissible when h(n) underestimates h*(n), that is, it respects Equation h(n) 6 h*(n) (2). The heuristic to be used may be the straight line distance, or the euclidean distance.

The other functions of this method are g(n), that denotes the cost of the path from the start node to node n, and f(n), which represents the estimated cost of the path passing through n to reach the goal node. f(n) is defined as the sum of g(n) with h(n), as in Equation f(n) = g(n) + h(n) (3)

The main advantage of this algorithm results from the fact that using the heuristic, the algorithm can quickly converge

The disadvantages include

- not consider obstacles for preventing collisions
- be slow in searching speed and be of poor applicability in the large scale paths search

.

The A* algorithm has been applied in several areas, among which can be mentioned the areas of automation / robotics (for trajectory planning of AGV in Smart Park), medicine (for needle penetration during surgery procedures) and games (for determining the paths in games).

2.2 Time Enhanced A*

Time Enhanced A* The Time Enhanced A* (TEA*) is an extension of the A*, used when there are multiple vehicles. It contains an additional component - time. This component allows a better prediction of the vehicles' movements during the run time. TEA* consists of an incremental algorithm that builds the path of each vehicle considering the movements of other mobile robots. This feature allows the algorithm to produce conflict free routes and, at the same time, deal with deadlock situations, since the paths are constantly recalculated and the map information is updated at each iteration. This way, the unpredictable events are considered in the input map, allowing to avoid the main challenges of any multi-robot approach, such as collisions and deadlocks. Each node on the map has three dimensions: the Cartesian coordinates (x, y) and a representation of the discrete time. The time is represented through temporal layers, $k = [0; T_{Max}], onwhich T_{Max}$ represents the maximum number of layers. Each temporal graph is composed of a set of free and occupied/obstacles nodes

The path for each robot is calculated during the temporal layers. In each temporal layer, the position of each vehicle is known and shared with the other vehicles. This way, it is possible to detect possible future collisions and avoid them at the beginning of the paths' calculation. Each robot can only start and stop in nodes and a node can only be occupied by one vehicle on each temporal layer. The operation of the TEA* algorithm is similar to A*, since for each AGV, during the path calculation, the next neighbor node analyzed depends on a cost function, f(n), given by the sum of two terms: the distance to the initial vertex, g(n), and the distance to the end point, h(n). The main difference between the two algorithms is that time is considered in TEA*, resulting in two definitions, according to:

- Definition 1: The neighbor vertices of a vertex j in the temporal layer k belong to the next temporal layer given by k + 1, that is, the total number of temporal layers depends of the number of iterations required to reach the intended destination. The more complex the map, the more iterations are needed.

- Definition 2: The neighbor vertices of vertex $j(v^j_{adj})$ include the vertex containing the AGV current position, and all adjacent vertices in the next time component, that is, the set of neighboring nodes includes not only the adjacent nodes, but also the node corresponding to the position in analysis. This property allows a vehicle to maintain its position between consecutive time instants if no neighbour node is free. In this case, it is

not considered a zero value for the euclidean distance; instead a constant heuristic value corresponding to the stopped movement is assigned.

2.3 Rapidly exploring Random Tree

The Rapidly exploring Random Tree (RRT) algorithm incrementally constructs a search tree in the configuration space until the goal configuration can be connected to one of its nodes. The operation of the RRT, involves the iterative execution of the following steps [10]:

1. A random configuration, q_{rand}, is sampled in the configuration space.
2. The tree is searched for a configuration q_{near}, which is the nearest node in the tree to q_{rand}.
3. A new configuration q_{new} is created by moving a predefined distance d from q_{near} in the direction of q_{rand}.
4. If q_{new} is a valid configuration that falls in C_{free} (unobstructed space), and if the local path between it and q_{near} is collision-free, then q_{new} is added to the tree as a new node and an edge is created between q_{new} and q_{near}. However, if q_{new} falls in C_{obs} (obstacle space), and if the local path between it and q_{near} has collisions, then is not created an edge between q_{new} and q_{near}.

These steps are repeated until the goal configuration can be connected to the tree or is reached a maximum number of iterations, or a given number of nodes in the tree, or a given running time. That is, the goal is to execute the whole process from q_{init} (starting point) to q_{goal} (end point). The most common metric for the nearest-neighbor selection is the Euclidean distance between points. In this case, the expansion pattern of the tree is modeled by the Voronoi diagram over the nodes within the tree. The probability of a node being expanded is directly proportional to the volume of its corresponding Voronoi region. Nodes that have a larger Voronoi region (i.e. the portion of the space that is closer to the node than to other nodes of the tree) are more likely to be chosen for expansion and are referred to as major nodes. This way, the tree is pulled towards unexplored areas, spreading rapidly in the configuration space (as the Voronoi regions of samples become approximately equal in size, the exploratory behavior gradually shifts from expansion of the tree to refinement). In the case of the Euclidean metric, these nodes tend to lie on the outside of the tree during the initial exploration. Conversely, inner or minor nodes have smaller Voronoi regions and often lie on the inside of the tree. Once the tree has explored the state space, these become major nodes as the algorithm begins to fill in the space. This phenomenon of favoring some nodes over others is referred to as the Voronoi bias, and yields an initial preference towards the exploration of the state space.

Summing up, the efficiency of **RRT** stems from the Voronoi bias property which promotes tree growth towards unexplored regions. Therefore, the key is

the determination of the distance metric which computes the nearest-neighbour in the **RRT** algorithm. In holonomic planning, the Euclidean distance is an ideal metric to generate a Voronoi bias because any node which is the closest from the sampled points can be expanded. If there exists differential constraints, however, which limit the evolution of the system states, the Euclidean distance measure fails to capture the true distance. Here, X_s is an initial state of the system and X1...X8 are existing nodes in the current tree. If Euclidean distance is used for the distance metric, X2 is chosen as the closest node from X_r as shown in Fig. 4b. However, this is not true for the system which has differential constraints. Instead, X4 is the closest node from Xr if the nonholonomic constraints are considered. From this example, it can be seen that the true distance metric is extremely important to the **RRT** planner under differential constraints.

The relative advantages of this algorithm are that it:

- is successful at solving path-planning problems in highdimensional spaces.
- can be implemented for real-time, online planning
- avoids wandering around in explored regions.

The disadvantages are that:

- it is not appropriate when road planning involves narrow passages.
- the solution obtained is sub-optimal, since the planning process is merely a random exploration of the space.

The **RRT** algorithm has already been applied in various areas, such as, molecular biology, automation / robotics, including path planning for mobile robots (medicine (for steerable needles in 3D environments with obstacles) and humansystem interaction. The practical applications just referred are based on the **RRT** algorithm; however, they do not use their standard/basic version, but variants of this or the conjugation with other algorithms, in order to overcome their disadvantages and/or to be able to acquire characteristics that fit to each problem under consideration.

2.4 Time Windows

In dynamic routing a calculated path depends on the number of currently active AGV missions and their priorities. The Time Windows (TW) method allows to determine the shortest path using time windows. This method checks the mission candidate paths by using the time windows to verify if certain paths are feasible. Viability of a particular path is evaluated by a time windows insertion followed by a time windows overlap (conflict) test. In the case of conflict, the algorithm iteratively reinserts time windows until conflicts disappear or an overlap is present only on the paths origin arc, indicating that the candidate path is not feasible. The procedure is repeated for all candidate paths and the result is a set of executable paths. The final task of the algorithm is to choose the

shortest one among executable paths in terms of a time required for a vehicle in mission to get from the origin to the destination arc. When a new mission is requested at a given time, is searched a idle vehicle to assign it to that mission (with an initial mission priority). As a goal of dynamic routing is to determine the shortest path for certain mission under the current state of the system, all candidate paths should be checked for feasibility.

The applications areas for Time Windows encompass AGV (for dynamic routing in multi-AGV systems), logistics (in vehicle routing problem, pickup and delivery problems, Home Health Care (HHC), and petrol station replenishment problems).

3 Local and Global Planner

Path planning algorithms are a crucial component of robotics and autonomous systems. Two main types of path planners are local and global planners, which differ in terms of their scope, computation, and adaptability.

1. Local Planner A local planner computes the robot's immediate path and adjusts it in real-time to react quickly to changes in the environment. Local planners are often used for navigation in dynamic environments, as they are better suited to quickly adjust for obstacles in the immediate vicinity.

Example: Potential field-based path planning is a common technique used to guide the robot around obstacles in the environment. The robot will move in the direction of a region of low potential, similar to a particle moving down a gradient field.

2. Global Planner A global planner computes a complete path from the robot's current position to the target position, taking into account the entire environment's layout. Global planners are often used in applications where the environment is static or only changes occasionally and where the robot's path must optimize a specific criteria, such as shortest distance or least travel time.

Example: The A* algorithm is a widely used global planner that finds the shortest path between two points in a graph. It works by expanding the search through the graph from the initial point to the goal point, estimating the distance to the goal at each point and selecting the path with the lowest estimated cost.

4 Challenges and Future Trends

This paper presented some algorithms used in the path planning of mobile robots. For each one, its operation was briefly described, some of its advantages and disadvantages were analyzed, and were presented some of its possible application areas. It was possible to conclude that there are several modifications that can be made to make each of these algorithms more efficient, and they can be associated with other algorithms to solve particular problems, as exemplified in several of the referred case studies. Next is planned to implement

some of these algorithms in AGV fleets, to find out their relative advantages and disadvantages and in which situations their implementation is more adequate.

5 References

1. D. Floreano, P. Husbands, and S. Nolfi, Springer Handbook of Robotics, 01 2008.

2. S. M LaValle, Planning Algorithms, 01 2006.

3. J. Denny, E. Greco, S. Thomas, and N. Amato, "Marrt: Medial axis biased rapidly-exploring random trees," 09 2014, pp. 90–97.

4. I. Al-Bluwi, T. Simon, and J. Corts, "Motion planning algorithms for molecular simulations: A survey," Computer Science Review, vol. 6, p. 125143, 07 2012.

5. https://repositorio.inesctec.pt/server/api/core/bitstreams/2547b96c-cbd9-4cb8-81b3-b9e86cdf5b56/content