Name: Alvin O'Garro
Advisor: Erik Verriest
Group Name: NavX

Graph-based Search in Robotic Path Planning: A Review

Introduction

Autonomous vehicles are becoming more and more prominent in the technology used today, with applications in defense, industrial automation, transportation, and others. An essential task that one of these autonomous vehicles must execute is finding an obstacle-free path from its current location to some destination using as few resources as possible. Thus, many path-planning algorithms have been developed for various robotics applications, with graph-based pathfinding algorithms being the subject of a large body of robotics research. This paper briefly reviews some of the software technologies currently being used in graph-based robotic path planning.

Graph-Based Path Planning Algorithms

The Graph and Dijkstra's Algorithm

Because of the limited computational resources available to a robot, a graph is often used as a map representation of the navigation space around a robot. A graph's nodes and edges offer a low-cost, discretized means to represent navigable points and straight-line arcs between points through which the robot can travel. Many algorithms for finding efficient paths on graphs have been developed in the past half-century, many of which are based on Dijkstra's Algorithm [1], created in 1959. Dijkstra's algorithm takes a starting node and goal node in a graph and systematically finds the list of edges connecting the two nodes with the lowest edge weights possible. In robotics, this algorithm is extremely useful for finding the optimal path, or the path with the lowest edge weights. The cost function, represented by the edge weights in the graph, can represent distance or difficulty of terrain navigation, among other metrics.

Other Algorithms

While Dijkstra's algorithm provides the optimal path deterministically, it does so by exhaustively searching many nodes in the graph before finding an optimal path [2]. This can be too costly and needlessly exact for many robotics usages, so more efficient, heuristic-based approaches were developed to combat this problem. A popular evolution of Dijkstra's is A-Star (A*), which uses a heuristic to reduce the search time at the expense of optimality of the final path. During graph search, the heuristic is used to choose the "closest" node to the goal node at each iteration of the algorithm, thus reducing the number of nodes that need to be explored. A* does not fix all the problems of Dijkstra's, as it still can only find

paths in a static environment [3]. Several A* derivatives, such as Dynamic A* (D*) and D* Lite have been formulated to efficiently navigate a world with a moving target and moving obstacles [3], [4].

Much of the modern research in robotics attempts to optimize the search time beyond the capability of Dijkstra-based algorithms with random sampling. One widely-used method is Rapidly-Exploring Random Tree (RRT) [5]. which generates a tree structure by randomly picking obstacle-free coordinates and connecting them to the nearest node in the already existing tree until the tree spans the navigable space. While the algorithm doesn't provide nearly the same optimality as A* and its derivatives, RRT does handle dynamic obstacle avoidance very well [3], [5].

Applications of Graph Search Algorithms

Consumer Products

The software tools that implement these algorithms are generally free to the public, with several open-source navigation libraries available on Github [6] and elsewhere. Free robotics frameworks, such as the Robot Operating System, provide the ability to implement a variety of path-planning algorithms [7]. Commercially-available robots that implement path planning algorithms, such as iRobot's Roomba and Create robots, can cost between $200 and $1000 [8].

Research and Development

In the domain of research and development, several navigation technologies are based on graph search. The technology that is used in the Roomba implements coverage path planning (CPP), in which a robot needs to explore all regions of a navigation space. CPP makes use of graph representations and searching [9] along with grid-based maps. In addition, swarm robotics applications must solve the problem of multiagent path planning, which heavily relies on A* and similar algorithms [10].

When deciding what methods to use for a new robot's navigation, researchers and engineers weigh the constraints of the navigation space and their optimization priorities. As discussed in [4], if the navigation space has few degrees of freedom and is easily discretized, it makes sense to use a graph representation and an optimal graph search algorithm like A*. If the space is more complex, a randomized algorithm like RRT may be more optimal to use instead. The degree to which obstacles and targets move will determine what kind of graph search method is used, with D* and D* Lite taking preference if the navigation space is more dynamic. In contrast, Dijkstra's algorithm is preferred if there is a greater need for optimal paths than there is for computation time.

[1] T. Abiy, H. Pang, J. Khim, "Dijkstra's Shortest Path Algorithm," brilliant.org, para. 2, Apr. 27, 2016. [Online]. Available: https://brilliant.org/wiki/dijkstras-short-path-finder/. [Accessed Mar. 4, 2018].

[2] N. Correll, Introduction to Autonomous Robots, 1.9th Ed. Magellan Scientific, 2016. [E-book] Available: https://github.com/correll/Introduction-to-Autonomous-Robots/releases/download/v1.9/book.pdf

[3] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, Y. Xia, “Survey of Robot 3D Path Planning Algorithms,” Journal of Control Science and Engineering, vol. 2016, April, 2016. [Online serial]. Available: https://www.hindawi.com/journals/jcse/2016/7426913/. [Accessed Mar. 4, 2018].

[4] D. Ferguson, M. Likhachev, A. Stentz, “A Guide to heuristic-based path planning,” In Proc. ICAPS Workshop on Planning under Uncertainty for Autonomous Systems, 2005, pp. 262-272.

[5] D. Connell, H. M. La, " Dynamic Path Planning and Replanning for Mobile Robots using RRT*,” Cornell University Library. Vol cs.RO, April, 2017 [Online serial]. Available: https://arxiv.org/pdf/1704.04585.pdf

[6] jslee02, “Awesome Robotics Libraries,” Mar, 2018. [Online]. Available: https://github.com/jslee02/awesome-robotics-libraries. [Accessed Mar. 4, 2018].

[7] N. Lamprianidis, “global_planner,” ros.org, para 2.4, Jan. 11, 2018 [Online]. Available: http://wiki.ros.org/global_planner [Accessed Mar. 4, 2018].

[8] “Roomba Robot Vacuum,” irobot.com, 2018 [Online]. Available: http://www.irobot.com/For-the-Home/Vacuuming/Roomba.aspx [Accessed Mar. 4, 2018].

[9] E. Galceran, M. Carreras, “A survey on coverage path planning for robotics,” Robotics and Autonomous Systems, vol. 61, no. 12, Sept, 2013. [Online serial]. Available: https://www.sciencedirect.com/science/article/pii/S092188901300167X [Accessed Mar. 4, 2018].

[10] J. Yu and S. M. LaValle, "Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics," in IEEE Transactions on Robotics, vol. 32, no. 5, pp. 1163-1177, Oct. 2016.