Path Planning Algorithms and Their Use in Robotic Navigation Systems

IOP Conf. Series: Journal of Physics: Conf. Series **1207** (2019) 012018 doi:10.1088/1742-6596/1207/1/012018

Path Planning Algorithms and Their Use in Robotic Navigation Systems

P E Teleweck[1,a] and B Chandrasekaran[1,b]

[1] Computer Engineering dept. Florida Polytechnic University 4700 Research way, Lakeland Fl 33805

E-mail: [a] Pteleweck3868@floridapoly.edu; [b] bchandrasekaran@floridapoly.edu

Abstract. This paper serves as an introduction to the concept of path planning and navigation. Contents include path planning algorithms and their many applications. Specific applications include navigation systems in autonomous/ semi-autonomous systems. Often these autonomous systems rely on several layers of sensor data, however at the root is a search-algorithm-based navigation system. This paper is to serve as an introduction to these algorithms and the use cases where young/ new roboticists can develop path finding/ path planning applications to fit their educational robotics needs. The A*, (A-Star), search algorithm and its applications will be the primary algorithm used within this paper. This paper is a portion of an ongoing research project for a robotic vehicle navigation system.

1. Introduction

Robotics is often referenced as an autonomous system or some system of electronic components that accomplish some task. Robotics is a very broad term that encompasses many interdisciplinary fields including, programming, mechatronics, electronics, and computer architecture. A topic of conversation in recent years has been autonomous systems such as autonomous vehicles. Countless millions of dollars have been poured into the research of autonomous systems, either in industrial applications or those regarding the safety of fully autonomous vehicles. With a market leaning towards autonomy in our roadways, education of these systems is paramount. Path planning and robotic navigation are at the forefront of the autonomous vehicle debacle [1]. Various search algorithms are used in part to develop these autonomous navigation systems alongside systems such as object detection and avoidance measures. This paper will go in depth as to where our current understanding of robotic navigation, path planning, and object avoidance are. Alongside results of testing, examples and further works will be discussed.

2. Robotic Navigation

Navigation of a robotic system incorporates various techniques and means of gathering information. Most notably, any one of many path finding/ path planning algorithms are likely to be used to assists in initializing the location of the robot and determining the next move. Navigation in the realm of robotics also includes object detection and avoidance. Object avoidance is often referenced when discussing Robot human interaction, or HRI. Being able to determine whether an object will interfere in the path that the robot is taking is imperative to the safety of the equipment and anyone around, as well as for the success of the navigation. Obstacles such as walls are referred to as static obstacles,

whereas those that are moving, such as humans on a sidewalk, are considered dynamic obstacles. We will discuss techniques for determining and avoiding both classifications of obstacles within this paper.

3. Path Planning

The planning of a path taken by a robot in a closed environment, can be done in various ways. Methods of determining a path include, but is not limited to, landmark based navigation, reactive planning, and various path planning algorithms. In a predetermined environment, a path planning algorithm could easily retrieve a set of coordinates for the robot to follow. Typically, as a means of testing various algorithms, a grid of a predetermined size is generated showing where on the map is "traversable". It is safe to assume for testing, that all borders of the grid are reachable by the robot. Within the scope of this project, we will also assume that a solution can always be reached from a given starting position.

3.1. Search Algorithms

Search algorithms in computer science are programs that try to solve any variation of tasks assigned to it. Examples of these would be Dijkstra's Algorithm, iterative search, backtracking, and A* to name a few. Given a list of data or values, a search algorithm will iteratively search for the required or requested value within the list [2]. The lists can be in the form of sequential number lists or in our case a list of coordinates forming a "grid", featured in Figure 1.

```
// 00010203040506070809101112131415161718 19
  1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,   // 00
  1,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,1,   // 01
  1,9,9,1,1,9,9,9,1,9,1,9,1,9,1,9,9,9,1,1,   // 02
  1,9,9,1,1,9,9,9,1,9,1,9,1,9,1,9,9,9,1,1,   // 03
  1,9,1,1,1,1,9,9,1,9,1,9,1,1,1,1,9,9,1,1,   // 04
  1,9,1,1,9,1,1,1,1,9,1,1,1,1,9,1,1,1,1,1,   // 05
  1,9,9,9,9,1,1,1,1,1,1,9,9,9,9,1,1,1,1,1,   // 06
  1,9,9,9,9,9,9,9,9,1,1,1,9,9,9,9,9,9,9,1,   // 07
  1,9,1,1,1,1,1,1,1,1,1,9,1,1,1,1,1,1,1,1,   // 08
  1,9,1,9,9,9,9,9,9,9,1,1,9,9,9,9,9,9,9,1,   // 09
  1,9,1,1,1,1,9,1,1,9,1,1,1,1,1,1,1,1,1,1,   // 10
  1,9,9,9,9,9,1,9,1,9,1,9,9,9,9,9,1,1,1,1,   // 11
  1,9,1,9,1,9,9,9,1,9,1,9,1,9,1,9,9,9,1,1,   // 12
  1,9,1,9,1,9,9,9,1,9,1,9,1,9,1,9,9,9,1,1,   // 13
  1,9,1,1,1,1,9,9,1,9,1,9,1,1,1,1,9,9,1,1,   // 14
  1,9,1,1,9,1,1,1,1,9,1,1,1,1,9,1,1,1,1,1,   // 15
  1,9,9,9,9,1,1,1,1,1,1,9,9,9,9,1,1,1,1,1,   // 16
  1,1,9,9,9,9,9,9,9,1,1,1,9,9,9,1,9,9,9,9,   // 17
  1,9,1,1,1,1,1,1,1,1,1,1,9,1,1,1,1,1,1,1,   // 18
  1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,   // 19
```

Figure 1. Display of proposed grid. 20 x 20 grid shown

Through a literature review, there have been applications of several familiar path finding/ path planning algorithms within the scope of robotic navigation. Numerous search algorithms were reviewed to determine which would best fit the needs of this research including D* (Dynamic A*), D* Lite, A* (A-star), the hill climbing algorithm, and more [3]. Each algorithm displays its own benefits and drawbacks mainly determined by the heuristic set by the user.

There were four predominant trade-off criteria that needed to be accounted for, each one posing a question to be determined [4].

1) Optimality: Does the heuristic guarantee the best solution can be found?
2) Completeness: Can the heuristic find all solutions for the path/ problem at hand?
3) Accuracy/ precision: Can the heuristic provide a confidence interval for the solution?
4) Execution Time: is the heuristic the best-case scenario for solving the given problem?

After considering the trade-off criteria we have determined that the A* path planning algorithm would be the best fit for the simulated environment we are proposing to test. A* is technically a modified version of the original Dijkstra's algorithm in which the value of the heuristic, h(n), is zero. A* is what is considered a Best-First search algorithm, meaning that the lowest cost per iteration move will be taken. This means that the closest most efficient path that is immediately visible is chosen, parsing through all possible "shortest paths". Being an informed search algorithm, it solves a given problem by evaluating all possible paths to determine the path that appears to lead most quickly to the intended solution. A* will always find a solution given one does exist, making this an optimal search algorithm. Given the heuristic value is admissible, then the algorithm itself is admissible, or optimal, if a closed set is not used.

A* aims to minimize the cost for the path to be taken [5]:

$$f(n) = g(n) + h(n) \quad (1)$$

Where g(n) is the cost of each move from the beginning node to the final node, and h(n) is the heuristic in which estimates future costs. A heuristic is an estimation or evaluation for the cost of future moves. In the interest of simplicity, this specific example uses the Euclidean distance cost evaluation. Adding the estimated cost of future moves and steps already taken, we can estimate the least cost path to travel.

Comparing D*, D* Lite, and A*, we see the main advantages and disadvantages to these path-finding algorithms. A* is considered complete, given the path cost is greater than some positive constant δ, and optimal regarding path costs [6]. A*, however, is considered memory inefficient. D* is considered optimal, complete, and is generally more efficient than A* in complex environments. D* Lite is functionally similar to D*, using a simpler algorithm.

Given a set of coordinates, a robot with adaptive A* path planning could traverse an unknown "plane" and find the shortest path from the origin to the final position. Robotic navigation can be accomplished using any combination of multiple techniques. Options include landmark based navigation, index-based a*, and the potential field method [7].

Landmark based navigation is similar to how humans interact with the real world, meaning that when traveling, there is normally a destination known. Often, we can determine the location of our intended goal by its surroundings or a known, fixed point. The robot will have this "landmark" given as a set of coordinates that it may use as a point of relativity. After an optimal path is created, the robot will traverse the grid, or area, to its intended location relative to the established landmark. An example within pop culture of a landmark based system being implemented is it the augmented reality mobile phone game, "Pokemon Go" [8].

4. Object detection/ avoidance

A method of Object avoidance that can drastically affect the outcome of robotic navigation is the potential field method. This method simulates that the robot and all obstacles, or non-traversable terrain, act as "charged particles". Given specific tolerances determined by the heuristic, the robot can plan a path determined by its location relative to other objects or "charged particles". This is especially helpful in situations where the robot may not interact directly with other objects such as walls or dynamic obstacles such as people walking. This would improve the safety of the system if it were implemented into an environment where humans or sensitive equipment are present [9].

4.1. Object Detection via Sensors

The accurate detection of objects in a space is the first step to successfully avoiding obstacles. Computer vision is a broad term that encompasses object detection, facial recognition, and other facets of sensor recognition. Examples of these techniques include lane detection and adaptive cruise control on intelligent transportation systems. The Face ID security system within apple uses multiple sensors in order to detect and recognize the user as a means of unlocking a device [10]. There are several tools available to assist in the detection of objects. The inclusion of LIDAR into systems that utilize object detection has increased accuracy and decreased costs. Light detection and ranging, LIDAR, is

primarily used in settings were the distance and location of objects is important. It works in a similar way to SONAR, sound navigation and ranging, using propagating light rather than sound waves, and receiving the information as the light is reflected/ refracted back to the sensor unit. Using a LIDAR unit, it is possible to map an entire area and use that corresponding sensor data to assist in object avoidance. In further testing we will be using the "360 laser distance sensor" LIDAR on the robot.

Dynamic obstacles are those that have movement [11]. Whether or not that movement is periodic, cyclical, or unpredictable, is important to distinguish. It is important to anticipate the movement of dynamic obstacles to assist in the avoidance of collisions. Static obstacles are normally the first to be distinguished. The majority of static obstacles will be walls; as other vehicles are considered dynamic. In the case of an autonomous car, sidewalks, signs, and medians can all be considered static obstacles. Their lack of movement makes it simpler to avoid.

Another example of a sensor used in computer vision/ object detection is the ZED Stereo camera by Stereolabs. This sensor uses two camera sensors to imitate human vision and give depth to an image that goes through image processing [12]. This allows for more information to be read from a video source. Being able to determine the depth of an object allows for time to process and determine what the next move may be. For systems that need to determine a path on the fly, knowing what obstacles stand in your way is important for selecting a proper direction to travel.

A combination of LIDAR and a depth sensing camera allow for a very accurate visualization of the environment. This allows for more precise movements and drastically reduces the likelihood of a collision. In Autonomous vehicles, these systems and more are present to do precisely that. Avoiding obstacles as well as avoiding oncoming collisions is important in any scenario where the humans are at risk.

5. Results

Through testing we were able to receive results from the A* algorithm. The grid that was established of 20 x 20 had to be traversed. Each move will output a set of coordinates that show each movement along the grid. This is shown in Figure 2. Using a predetermined set of traversable and non-traversable areas in the grid, the algorithm was able to determine the most efficient path through the grid. This proves A* works under controlled conditions. This however does not prove the consistency of A* when using sensor data as the input values. The continuing works will determine this use case. The algorithm used was based off and adapted from the open source works of Justin Heyes-Jones [13].

```
Search found goal state
Node position : (7,9)
Node position : (7,10)
Node position : (8,10)
Node position : (8,11)
Node position : (8,12)
Node position : (8,13)
Node position : (8,14)
Node position : (8,15)
Node position : (8,16)
Node position : (9,16)
Node position : (10,16)
Node position : (10,17)
Node position : (10,18)
Node position : (10,19)
Node position : (11,19)
Node position : (12,19)
Node position : (12,18)
Node position : (13,18)
Solution steps 17
SearchSteps : 32
```

Figure 2. Display of the output list of coordinates. This list is used as a path from the start point to the final location.

6. Future work

The future work for this paper includes an integration of the A* path planning algorithm developed, object avoidance algorithms, and sensor fusion with ROS so create a working robotic prototype. The robot used for development will be a "TurtleBot Burger". This modular robotic platform has built in support for raspberry-pi, a built in LIDAR, and expansion for further sensors such as the ZED Stereo camera. Using Gazebo Ros, a simulation tool, there will be a full simulation of the robot using the prewritten algorithms. The algorithms used for this paper were derived from a previous work [13].

For the full simulation, modifications to the A* algorithm will be made, including the development of a local map for the robot that feeds off sensor input. This gets rid of the need for the full "global" map and allows for faster processing of the path for "on-the-fly" applications. This will be done by referencing only the neighbour nodes to the robot and deriving information only from the most immediate inputs.

References

[1] http://www.driverless-future.com/Monitoring the self-driving car innovation process: California AV permits. (n.d.). Retrieved from http://www.driverless-future.com/

[2] Heyes-Jones, J. (2001). A* algorithm tutorial. Retrieved from http://heyes-jones.com/astar.php

[3] E. Fernandes, P. Costa, J. Lima and G. Veiga, "Towards an orientation enhanced astar algorithm for robotic navigation," *2015 IEEE International Conference on Industrial Technology (ICIT)*, Seville, 2015, pp. 3320-3325.

[4] Hayes, S. C., & Hofmann, S. G. (Eds.). (2018). *Process-based CBT: the science and core clinical competencies of cognitive behavioral therapy*. New Harbinger Publications.

[5] Heuristics. (1997). Retrieved from http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

[6] Choset, H., Mills-Tettey, A., Lee-Shue, V., Jr., Atkar, P. N., & Tantisevi, K. (n.d.). *Robotic Motion Planning: A* and D* Search*. Lecture. https://www.cs.cmu.edu/~motionplanning/lecture/AppH-astar-dstar_howie.pdf

[7] C. Qian, Z. Qisong and H. Li, "Improved artificial potential field method for dynamic target path planning in LBS," *2018 Chinese Control And Decision Conference (CCDC)*, Shenyang, 2018, pp. 2710-2714.

[8] Krukar, J., Schwering, A. & Anacta, V.J. Künstl Intell (2017) 31: 121. https://doi.org/10.1007/s13218-017-0487-7

[9] C. Ton *et al*., "LIDAR Assist Spatial Sensing for the Visually Impaired and Performance Analysis," in *IEEE Transactions on Neural Systems and Rehabilitation Engineering*. [7] https://ieeexplore.ieee.org/document/4459093/

[10] Z. Kim, "Robust Lane Detection and Tracking in Challenging Scenarios," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 16-26, March 2008.

[11] Y. Liu, D. Chen and S. Zhang, "Obstacle avoidance method based on the movement trend of dynamic obstacles," *2018 3rd International Conference on Control and Robotics Engineering (ICCRE)*, Nagoya, 2018, pp. 45-50.

[12] Stereolabs. (n.d.). Retrieved from https://www.stereolabs.com/

[13] J Hayes-Jones, astar-algorithm-cpp, (2001), GitHub Repository, https://github.com/justinhj/astar-algorithm-cpp