Autonomous Robots Path Planning: An Adaptive Roadmap Approach

Mohamed Elbanhawi[1, a], Milan Simic[1, b] and Reza Jazar[1, c]

[1]RMIT University, Melbourne, Australia

[a]s3322588@student.rmit.edu.au, [b]milan.simic@rmit.edu.au, [c]reza.nakahiejazar@rmit.edu.au

Keywords: Autonomous, Mobile Robots, Path Planning, Navigation, Algorithm, Roadmap

Abstract. Developing algorithms that allow robots to independently navigate unknown environments is a widely researched area of robotics. The potential for autonomous mobile robots use, in industrial and military applications, is boundless. Path planning entails computing a collision free path from a robot's current position to a desired target. The problem of path planning for these robots remains underdeveloped. Computational complexity, path optimization and robustness are some of the issues that arise. Current algorithms do not generate general solutions for different situations and require user experience and optimization. Classical algorithms are computationally extensive. This reduces the possibility of their use in real time applications. Additionally, classical algorithms do not allow for any control over attributes of the generated path. A new roadmap path planning algorithm is proposed in this paper. This method generates waypoints, through which the robot can avoid obstacles and reach its goal. At the heart of this algorithm is a method to control the distance of the waypoints from obstacles, without increasing its computational complexity. Several simulations were run to illustrate the robustness and adaptability of this approach, compared to the most commonly used path planning methods.

Introduction

Autonomous mobile robots are able to purposely and safely traverse an unfamiliar environment without human intrusion. An autonomous robot initially needs to gather information about its environment to be able to safely navigate through it. This is known as the perception stage. The perceived data is then treated in order for the robot to localize itself in the environment map. Knowing its current and goal location the robot will proceed to plan a collision free route. Finally, the mobile robot will trail that generated route.

The problem at hand is to develop a strategy that allows the robot to reach its desired position through a short, smooth and collision free path. Once the robot localizes itself in its environment and gathers information about obstacles, a path planner algorithm should be implemented to generate the desired path. The path planner additionally regulates features of the path such as smoothness and length.

Mobile robot path planning algorithms are categorized into classic and heuristic approaches. According to [1] more than fifty percent of all current robot planning algorithms are based on classical methods. Nevertheless, the application of classical methods is in constant decline in favour of heuristic approaches. The most commonly used classical methods are Potential Field, Voronoi Diagrams, VD, and Visibility graphs, Vgraphs, [1].

Classical methods are generally subdivided into roadmap, cell decomposition and potential field methods. Detailed analyses of these methods can be found in [2] and [3]. The most popular roadmap approaches are VD and Vgraph. These methods graphically analyse the map, in which the robot has localized itself in, to produce a network, or connectivity graph. A connectivity graph is a set of feasible routes from the current robot position, through sets of successive nodes, to the target position.

Voronoi diagrams generate nodes (waypoints) that are equidistant to two or more objects. They were first used by [4] in path planning. VD have been applied in several robot path planning approaches [5]. They have also been combined with heuristic methods [6]. The main drawback of VD is that they tend to generate long routes, as they maximize the distance between the robot and obstacles.

Visibility graph approaches consider obstacle vertices, in the map, to be the nodes through which the robot can reach its desired position. They proceed to connect vertices that are visible to each other. Visible nodes are nodes with the property that straight line joining them does not intersect any obstacles. Vgraphs were first used in robot motion planning by [7]. They guarantee that the robot will find the shortest path to its goal. A V*graph was introduced in [8], which reduced the number of considered vertices, thus reducing the computational complexity of the algorithm. In [9], a reduced Vgraph method was coupled with a curved weighed Dijkstra algorithm [10] and Simulated Annealing for autonomous mining applications. The main disadvantage of this approach is that it plans a route that forces the robot to pass, as close as possible, to any detected obstacles.

Autonomous robot localization and mapping data are computed by probabilistic approaches given in [11, 12]. Subsequently, planned routes that are close to any detected obstacles carry high risk of collision as they neglect the uncertainty of localization algorithms and consider the given data to be accurate.

Potential fields [13] consider the robot to be under the influence of several forces. These forces are generated towards the goal and away from obstacles. This approach is rather popular in robot motion planning [1]. However potential fields tend to get trapped in local minima and generate vibrating paths in narrow passages as discussed in [14].

Cell decomposition methods proceed to subdivide the map into smaller cells. The decomposition continues until a minimum resolution is reached (the size of the robot), a maximum number of iteration is reached, or all cells are either free of obstacles, or completely occupied by obstacles [2]. This method was first used by [15] in robot motion planning. Approximate cell decomposition proceeds to subdivide cells that are neither completely full nor completely occupied and thus reduce the computational complexity of that approach. In [16], an approximate cell decomposition algorithm was coupled with tesseral addressing to further reduce the computational time of cell decomposition. The free cells are then used as waypoints or nodes in a connectivity graph as explained earlier.

Heuristic methods employ certain assumptions to reduce the complexity of a problem. Arguably, the most commonly used heuristic method in robot motion planning is A* Algorithm which was introduced by [17] and used in [8], [18] and [19]. Other heuristic algorithms mimic biological ones, such Genetic Algorithms [6, 19] and Ant Colony Optimization [18, 20], then physical phenomena, such as Simulated Annealing [9] or human decision making such as Fuzzy Logic Control [21-23]. The drawback of these methods is that they use multiple variables and coefficients that must be chosen by the algorithm designer. There is no literature that defines a particular method for variable selection and thus results are not consistent for different scenarios. For instance, a genetic algorithm requires the selection of mutation and cross-over factors, encoding and decoding methods, selection criteria, fitness function design, number of generations, population size and finally number of individuals and their string length. All these variables require fine-tuning by the designer and they do not guarantee optimal solutions. As a consequence, heuristic methods do not produce general solutions. For different situation the variables of a heuristic algorithm may require adjustment.

We present here an adaptive roadmap-based path planning algorithm for mobile robots in two-dimensional maps. As any other path planner, it assumes a-priori knowledge of the robot position, its goal position and surrounding obstacles. It combines the ability of some algorithms to produce optimum short paths with the characteristic of other algorithms that route robots safely away from obstacles. This method also reduces the computational complexity of roadmap approaches, which is their main disadvantage, as it only considers obstacles that are in its path neglecting all others. Unlike most heuristic methods, the proposed planner has the ability to produce general solutions for different scenarios. A minimum safe distance is introduced to the algorithm in order to modify it based on the complexity of the map and the certainty of the robot's position. It can also be combined with any other algorithms to produce different planners. The algorithm is presented in section 2, algorithms used with the presented planner will be discussed in section 3 and finally some experiments are presented in section 4 to illustrate the implementation of this algorithm.

Proposed Path Planning Algorithm

Algorithm, proposed here, could be considered as a roadmap approach to path planning. It generates a set of waypoints, through which the robot can navigate without colliding with obstacles. All obstacles in the map are modelled as polygons. The algorithm analyses location of each obstacle's vertices. The start and target position of the robot are considered to be known relative to the obstacles in the surrounding environment. The operation of the algorithm, for a map that consists of one obstacle is as follows:

1- The algorithm is given the robot's start S and target T positions, and obstacles vertices number and location, as illustrated in Fig. 1 (a).
2- A straight-line path joining the start and target points is generated, as shown in Fig. 1 (b). The straight path is the shortest path between the two points, however, usually, it is not collision free, as in this case. This line divides the map into two halves.
3- The intersection point between the straight-line path and the obstacle are computed. They are highlighted in red in Fig. 1 (b).
4- Waypoints are calculated, for each half of the map, such that the line joining the waypoint and its corresponding intersection point is orthogonal to the original straight-line path, as shown in Fig. 1 (c). Since there are two intersection points with each point having two waypoints in each half of the map, a total of four waypoints are computed.
5- Successive waypoints are joined together to generate several possible collision free routes for the robot around the obstacle. In Fig. 1 (c), there are two possible collision free paths. Each path passes through two waypoints, both of which are highlighted in blue.
6- The algorithm is reiterated for every segment of the path until a collision free path is generated. The start and finish points of that segment are considered to be the intermediate start and target goals for the robot.

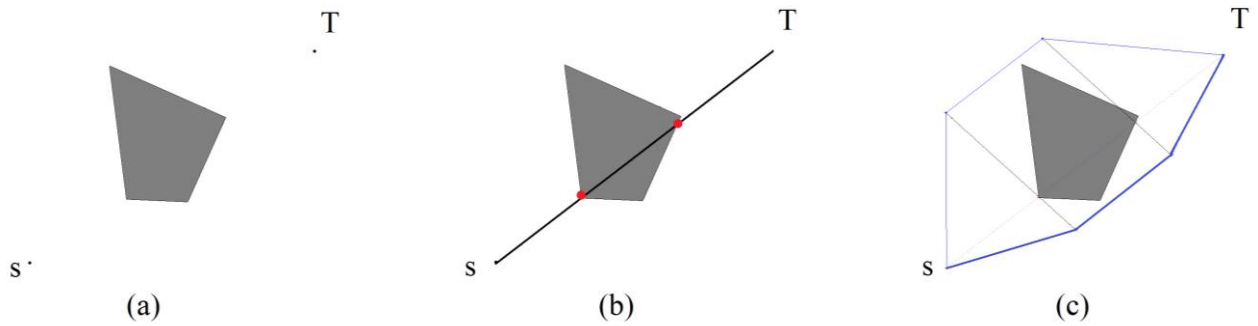

Fig. 1 The different stages of waypoint generation, around an obstacle in the robot's path, using the proposed algorithm

Waypoint Calculation. The advantage of this method over classical roadmap approaches, such as VD and Vgraph, is that the distance between the generated route and obstacles can be controlled. VD methods ensure that the robot follows a route that is far from any obstacles and Vgraph generated routes as close to the obstacles. The proposed method can be better controlled, i.e it enables the robot to approach obstacle within a certain, acceptable, distance while also minimizing the travelled path. This allows for the algorithm to adapt based on the complexity of the obstacles and the certainty of the information about the obstacle and robot locations.

Fig. 1(b) shows the third step in the algorithm, where the intersection points between the straight-line path and obstacles are known and it is then required to calculate the waypoints for those intersection points. Consider Fig. 2, the line joining the start S and goal G points intersect an obstacle, whose edges are shown as dotted lines, at point P. It is required to calculate a point P' that creates an orthogonal line to the straight-line path at point P. The normal distance between the point P' and the straight line must exceed the maximum distance n between any vertex in that object and the straight line by a safe distance of δ. The normal distance n is calculated using Eq. 1 and the waypoints P' on

either side of the intersection point are calculated using Eq. 2 and 3. In the given equations **G, S, V, P** and **P'** are position vectors for the goal, start, vertex, intersection and waypoint locations.

$$n = \frac{|(\mathbf{G}-\mathbf{S})\mathrm{X}\,(\mathbf{S}-\mathbf{V})|}{|\mathbf{G}-\mathbf{S}|} \tag{1}$$

$$\mathrm{P}' = \mathrm{P} \pm (\mathrm{n}+\delta)\begin{bmatrix}-\mathrm{Cos}\,(90-\alpha)\\ \mathrm{Sin}\,(90-\alpha)\end{bmatrix} \tag{2}$$

$$\begin{bmatrix}x'\\ y'\end{bmatrix} = \begin{bmatrix}x\\ y\end{bmatrix} \pm (\mathrm{n}+\delta)\begin{bmatrix}-\mathrm{Cos}\,(90-\alpha)\\ \mathrm{Sin}\,(90-\alpha)\end{bmatrix} \tag{3}$$

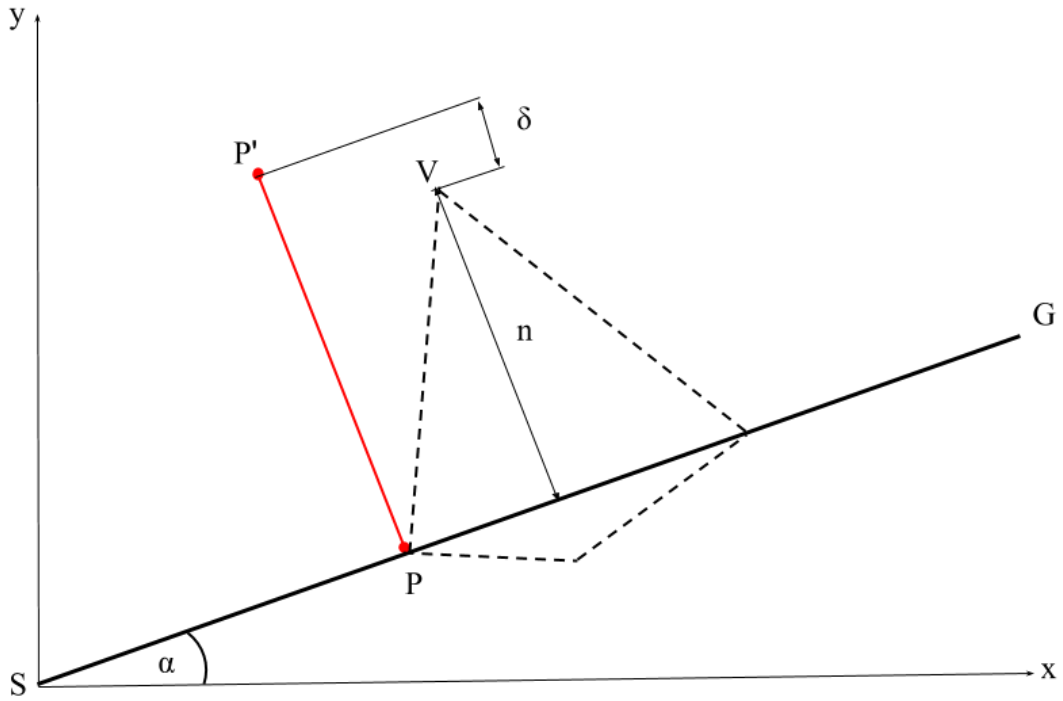

Fig. 2 Calculation of waypoint (P') at an intersection point (P)

Architecture of the Proposed Planner

Path planning for autonomous robots is the stage that follows localization. The features of the surrounding environment are detected and processed using algorithms that are presented in [11]. The path planner generates a route based on the localization data.

The proposed path planner consists of two main components, a global planner and a path optimizer. The global planner gathers the information about the robot's current and goal positions in addition to all information about obstacles. It then proceeds to generate sets of possible routes through which the robot can reach its goal. The optimizer analyses all the possible routes and selects the most suitable path, based on the predefined weight evaluation function.

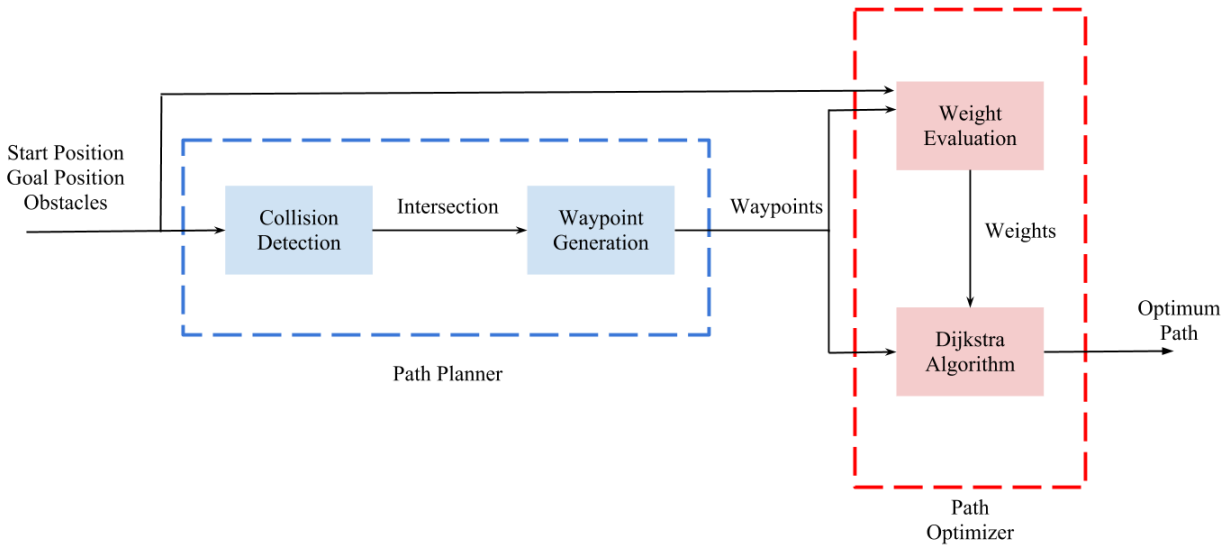

Fig. 3 The proposed algorithm consists of two main components, planning and optimization

The different components of the planner are shown above in Fig. 3. The details of the first component (path planning) are provided in the previous section. The planner generates a set of possible routes through which the robot can reach the desired goal.

There are multiple paths generated by the path planning part of the algorithm. They are evaluated based on the cost, or weight function. The task can be treated as a single source shortest path mathematical problem. A Dijkstra algorithm is employed to select the safest path with the least cost, referred to as the optimum path in this paper.

For this application, the cost function calculates the Euclidian distance between successive waypoints in the path. The problem at hand is now simplified into a single source shortest path problem that can easily be resolved using a Djkastra algorithm. The optimum path selected is the shortest path. It is possible to change the features of the path by varying the evaluation of the weight, or cost function. This variation allows for the option to select the path based on changes in direction, curvature or any other desired property. This is needed when it is required to minimize the time travelled not just the distance. Changing the evaluation function is used to account for any kinematic or dynamic constraints on the vehicle, or robot.

Experiments and Results

Several simulations were run to demonstrate the features of the proposed algorithm. All simulations were carried out using Matlab. Maps were created as images and loaded into the user interface. The user selects the start and goal positions in addition to the desired minimum distance. The dimension of the workspace is defined by the user as well. After the waypoints are generated, any possible path that lies outside workspace is eliminated.

Experiment 1 involved finding the shortest safe route around two obstacles that are arranged as shown in Fig. 4 (a). The algorithm proceeds to find the shortest path, as shown in Fig. 4 (b) while maintaining a safe distance away from the obstacles. Both Fig. 4 (c) and (d) illustrate the effect of adding another obstacle. The robot's route will not change unless the added object interferes with that route. As shown in Fig. 4 (d), the algorithm reroutes the robot, when an object intersect the shortest collision free path, to find the next shortest path.

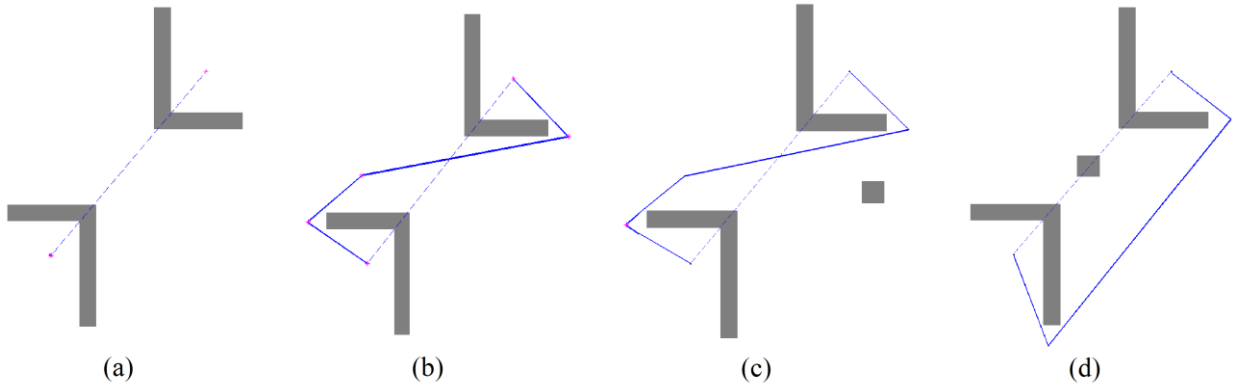

Fig. 4 Experiment 1 demonstrates the operation of the algorithm with multiple obstacles and the effect of placing obstacle on the generated path

Experiment 2 shows the effect of increasing the value of the safe distance from the obstacles δ. The algorithm will proceed to find the shortest possible distance, and will maintain the specified distance as shown in Fig. 5. In an area of 355,365 units squared, δ values of 0, 10, 20 and 80 units were used respectively in Fig. 5 (a), (b), (c) and (d). It must be noted that the generated path with a $\delta = 0$, shown in Fig. 5 (a), is identical to a path that would be generated by a Vgraph method as it forces the robot to move as close as possible to the obstacle.

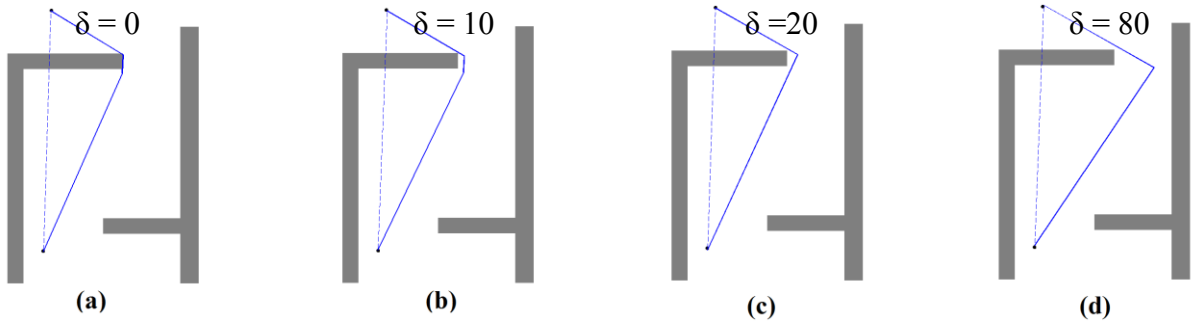

Fig. 5 Experiment 2 illustrates the effect of changing the safe distance δ on the generated path

Experiment 3 is designed to compare performances of the proposed algorithm and Vgraph method. It can be seen that in both Fig. 6(a) and (b) the path generated by Vgraph method forces the robot to pass as close as possible to the wall. On the other hand, by increasing δ value the path generated by the proposed algorithm will change accordingly to increase its distance from the obstacle. This highlights the ability of this algorithm to generate safer paths than Vgraph especially when the obstacle locations are uncertain.

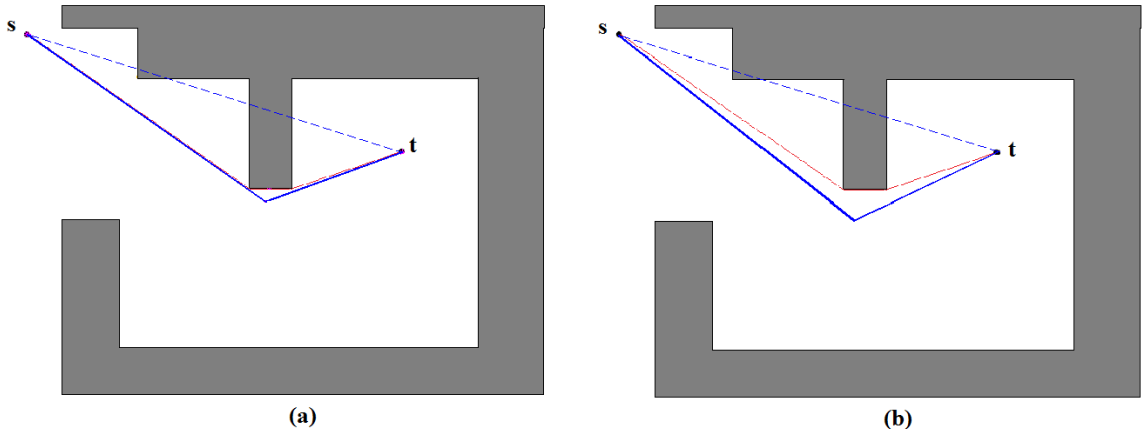

Fig. 6 Experiment 3 compares outcomes of the proposed method and Vgraph

A comparison between the path generated by VD and the proposed algorithm is shown in Fig. 7. Experiment 3 shows that the proposed algorithm generates shorter path, as it does not have to be equidistant from two obstacles. Additionally, the generated path is smoother since it only consists of one waypoint between the start and goal positions, thus requires fewer changes in robot's heading unlike the VD generated path.

Experiment 4 examines the problem presented in [18]. The generated path for the Simple Ant Colony Optimization algorithm is given in [18]. The proposed algorithm is used to plan a collision free path in the same map with different δ values. Both scenarios are shown in Fig. 8 (a) and (b). It can be seen than with a small δ value the path is shorter than the Ant Colony generated path, however it tends to pass close to the obstacles, as highlighted in red. On the other hand, the algorithm will generate a longer, yet safer path, when increasing δ. Another disadvantage of the Ant Colony algorithm, given in [18], like many heuristic methods, is that it treats the workspace as a grid. Subsequently, the position of waypoints cannot be modified with a resolution smaller than the grid size, unlike the presented algorithm, which does not require the environment to be represented as a grid. Additionally, information about each grid, or cell, must be stored. This will increase the computational complexity of the path planning and reduce its suitability for real time applications.

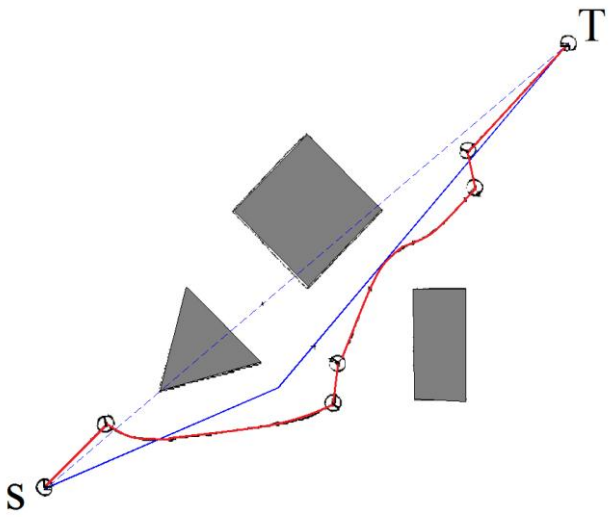

Fig. 7 Experiment 4 compares between the proposed algorithm and VD

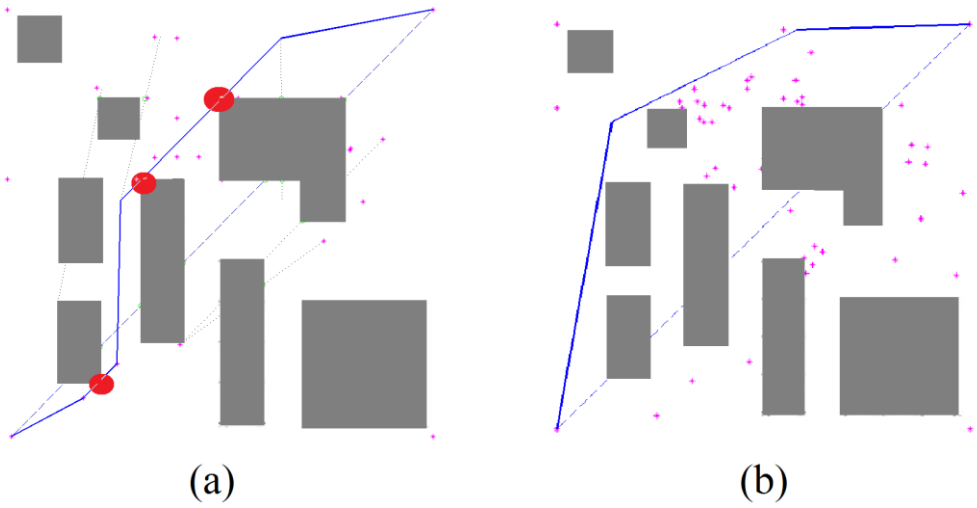

Fig. 8 Experiment 5 compares the proposed algorithm with a heuristic algorithm (a) Path generated using the proposed algorithm a low δ value (b) Path generated using the proposed algorithm and a high δ value

Conclusion

In this paper, a new path planning algorithm for autonomous vehicles is presented. It combines the advantages of two of the most commonly used roadmap algorithms, VD and Vgraphs. Unlike both VD and Vgraph approaches, this method provides a great extent of flexibility. The adaptability of the proposed system stems from varying the minimum allowable distance between the robot and any detected obstacle. The adaptability feature of this algorithm makes it more efficient and allows it to produce general solutions for different scenarios. It provides the shortest collision free path, similar to Vgraph, while maintaining a safe distance from obstacles, similar to VD.

Simulations have been conducted in order to compare the performance of the presented algorithm, with classical and heuristic approaches. Sets of experiments demonstrated the effect of changing the safe distance on the planned path. The results of the experiments, are promising, as they illustrated the effectiveness, computational efficiency and adaptability of the presented approach.

The shortcoming of this method is the lack of a proper procedure to selecting a suitable safe distance, but the variability of the minimum safe distance makes this algorithm very promising. At this stage, the minimum safe distance δ must be defined by the user prior to using the algorithm. A safe distance that produces acceptable results for one map may not generate a collision free path in other situations. We need to further investigate this promising approach and probably include a function that would be used to optimize selection and eliminate trial and error part of the algorithm. That is the subject of the further research.

Future Work

The presented algorithm has shown promising characteristics, and generated good results in numerous scenarios and experiments. Further investigation is required to improve the performance of this planner. Introducing a method to calculate the safe distance based on the given scenario would allow this algorithm to produce general solutions without any user input. Additionally, the current algorithm is coupled with a Dijkstra algorithm to calculate the shortest path through the generated waypoints. The weights between the points are calculated based on the Euclidian distance between them. Using a curve weighted Dijkstra algorithm presented in [9], a fuzzy evaluated cost function presented in [18], or a fitness function, would improve the control of the features of the path, based on the desired smoothness, curvature and length. Further experimentation using different algorithms such as A* Algorithm, Fuzzy Logic, Genetic Algorithms and Ant Colony Optimization is needed to optimize the performance of the presented planner. Further investigation into changing the position of the generated waypoints, within a certain area and its effect of the generated path is required. This is possible since the presented algorithm does not require grid representation of the workspace.

References

1. Masehian, E. and D. Sedighizadeh, *Classic and heuristic approaches in robot motion planning-a chronological review.* World Academy of Science, Engineering and Technology, 2007. **23**: p. 101-106.
2. Siegwart, R., I.R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. 2011: Mit Press.
3. Latombe, J.-C., *ROBOT MOTION PLANNING.: Edition en anglais*. 1990: Springer.
4. Canny, J. *A Voronoi method for the piano-movers problem*. in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*. 1985.
5. Takahashi, O. and R.J. Schilling, *Motion planning in a plane using generalized Voronoi diagrams.* Robotics and Automation, IEEE Transactions on, 1989. **5**(2): p. 143-150.
6. Chien-Chou, L., C. Wei-Ju, and L. Yan-Deng. *Path Planning Based on Bezier Curve for Robot Swarms*. in *Genetic and Evolutionary Computing (ICGEC), 2012 Sixth International Conference on*. 2012.

7. Asano, T., et al. *Visibility-polygon search and euclidean shortest paths*. in *Foundations of Computer Science, 1985., 26th Annual Symposium on*. 1985.
8. Alexopoulos, C. and P.M. Griffin, *Path planning for a mobile robot.* Systems, Man and Cybernetics, IEEE Transactions on, 1992. **22**(2): p. 318-322.
9. Maekawa, T., et al., *Curvature continuous path generation for autonomous vehicle using B-spline curves.* Computer-Aided Design, 2010. **42**(4): p. 350-359.
10. Dijkstra, E.W., *A note on two problems in connexion with graphs.* Numerische Mathematik, 1959. **1**(1): p. 269-271.
11. Durrant-Whyte, H. and T. Bailey, *Simultaneous localization and mapping: part I.* Robotics & Automation Magazine, IEEE, 2006. **13**(2): p. 99-110.
12. Luettel, T., M. Himmelsbach, and H.J. Wuensche, *Autonomous Ground Vehicles: Concepts and a Path to the Future.* Proceedings of the IEEE, 2012. **100**(Special Centennial Issue): p. 1831-1839.
13. Khatib, O., *REAL-TIME OBSTACLE AVOIDANCE FOR MANIPULATORS AND MOBILE ROBOTS.* International Journal of Robotics Research, 1986. **5**(1): p. 90-98.
14. Koren, Y. and J. Borenstein. *Potential field methods and their inherent limitations for mobile robot navigation*. in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. 1991.
15. Brooks, R.A. and T. Lozano-Perez, *A subdivision algorithm in configuration space for findpath with rotation.* Systems, Man and Cybernetics, IEEE Transactions on, 1985. **SMC-15**(2): p. 224-233.
16. Arney, T. *An efficient solution to autonomous path planning by Approximate Cell Decomposition*. in *Information and Automation for Sustainability, 2007. ICIAFS 2007. Third International Conference on*. 2007.
17. Hart, P.E., N.J. Nilsson, and B. Raphael, *A Formal Basis for the Heuristic Determination of Minimum Cost Paths.* Systems Science and Cybernetics, IEEE Transactions on, 1968. **4**(2): p. 100-107.
18. Garcia, M.A.P., et al., *Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation.* Applied Soft Computing Journal, 2009. **9**(3): p. 1102-1110.
19. Cen, Z., Z. Qiang, and W. Xiaopeng. *Robotic Global Path-Planning Based Modified Genetic Algorithm and A* Algorithm*. in *Measuring Technology and Mechatronics Automation (ICMTMA), 2011 Third International Conference on*. 2011.
20. Dorigo, M., M. Birattari, and T. Stutzle, *Ant colony optimization.* Computational Intelligence Magazine, IEEE, 2006. **1**(4): p. 28-39.
21. Antonelli, G., S. Chiaverini, and G. Fusco, *A fuzzy-logic-based approach for mobile robot path tracking.* IEEE Transactions on Fuzzy Systems, 2007. **15**(2): p. 211-221.
22. Hagras, H.A., *A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots.* Fuzzy Systems, IEEE Transactions on, 2004. **12**(4): p. 524-539.
23. Zadeh, L.A., *Outline of a New Approach to the Analysis of Complex Systems and Decision Processes.* Systems, Man and Cybernetics, IEEE Transactions on, 1973. **SMC-3**(1): p. 28-44.