Robotica (2014) volume 32, pp. 611–624. © Cambridge University Press 2013
doi:10.1017/S0263574713000738

# A global path planning method for mobile robot based on a three-dimensional-like map

Yaonan Wang and Wenming Cao*

*College of Electrical and Information Engineering, Hunan University, Changsha 410082, China*

(Accepted June 17, 2013. First published online: October 7, 2013)

**SUMMARY**

This paper presents a novel global path planning method for mobile robots. An improved grid map, called three-dimensional-like map, is developed to represent the global workspace area. The new environment model includes not only contour information of obstacles but also artificial height information. Based on this new model, a simple but efficient obstacle avoidance algorithm is developed to solve robot path planning problems in static environment. The proposed algorithm only requires simple distance calculations and several comparison operations. In addition, unlike other algorithms, the proposed algorithm only needs to deal with some obstacles instead of all. The research results show that this method is computationally efficient and can be used to find an optimal or near optimal path.

KEYWORDS: Three-dimensional-like map; Global static environment; Obstacle avoidance algorithm.

## 1. Introduction

A fundamental robotics task is to plan a collision-free path as efficiently and reliably as possible from a start to goal position in an obstacle-cluttered workspace. The degree of difficulty of path planning in robots varies greatly depending on several factors: whether all information regarding the obstacles (i.e., sizes and locations) is known before the robot moves and whether these obstacles move around or stay in place as the robot moves. The different possible scenarios are shown in Table I.

In partially or completely unknown environment, the main purpose of planning is to enhance the robot's obstacle avoidance ability, and then take into account other criteria such as the path length and movement time, since a robot has no priori knowledge about the environment. However, in global path planning, the goal of planning is not only to avoid obstacles successfully but also produce an optimum path. The focus of this paper is on path planning in global static obstacles. This problem is usually solved in the following two steps:

**Step 1:** In the initial stage, the environment is mapped using an appropriate algorithm that represents the global workspace area.
**Step 2:** Perform a graph search to find a collision-free path from a start to a goal position.

The environment model differs depending on which approach is used to solve the problem, and the three most common approaches are the roadmap approach, the cell decomposition approach, and the potential field approach. The roadmap approaches include the visibility graph,[1] the Voronoi diagram,[2–3] and the freeway net,[4] and this approach is dependent upon the concepts of configuration space and a continuous path. A set of line segments, each of them connects two nodes of different polygonal obstacles, lies in the free space and represents a roadmap. If a continuous path can be found in the free space of roadmap, the initial and goal points are then connected to this path so as to achieve the final solution. Conversely, the basic idea of cell decomposition approach[5] is that a path between the initial and goal configuration can be determined by subdividing the free space of the robot's configuration into small regions called cells. Thus, a connectivity graph is constructed according to

* Corresponding author. E-mail: caowenming8@gmail.com

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press

Table I. Path planning environment category.

| Environment | Static obstacles | Dynamic obstacles |
|---|---|---|
| Completely known | Case I | Case II |
| Partially or completely unknown | Case III | Case IV |

the adjacency relationships between the cells after decomposition. From this connectivity graph, a continuous path can be determined by following adjacent free cells from the initial point to the goal point. If more than one continuous path can be found based on above algorithms, some search methods are used to find the best path, such as A* algorithm,[6] D* algorithm,[7] and their variations. Furthermore, the potential field method[8] involves modeling the robot as a particle moving under the influence of a potential field that is determined by the set of obstacles and the target destination. This method is usually very efficient but prone to local minima, and this issue is usually resolved by coupling the method with techniques to escape from local minima, or by constructing potential field functions that contain no local minima. Other solving algorithms include neural network method,[9] simulated annealing method,[10] ant colony optimization method,[11] fuzzy logic method,[12] genetic algorithm,[13] and their variations and extensions.

As we know, almost all paths planning methods in two-dimensional space just utilize obstacle contour information. How to further explore obstacle information to improve planning efficiency is an interesting problem. This paper describes a new path planning method for robot in the presence of static obstacles, and the proposed method can be summarized as follows:

**Step 1:** Define a grip map to represent a geometric structure of the environment.
**Step 2:** Assign specified value to each grid point according to certain rules. In our program, these values represent height values of each point. Therefore, the improved grip map includes not only obstacle contour information but also height information.
**Step 3:** Construct path planning algorithm based on foregoing improved map.

It should be noted that the "height" in our program is just an artificial concept to simulate the concept of the natural height. These values do not represent the true height of the obstacles. In fact, there is no any connection between the specified "height values" and the true heights of the obstacles. In order to differentiate from real three dimension map, the improved grid map is called three-dimension-like map in the context of this paper. Based on this new environment model, this paper develops a simple but efficient algorithm for path planning in global static environment. Unlike traditional algorithms, the proposed algorithm only requires simple distance calculations and several comparison operations. Furthermore, it does not need to deal with all obstacles. Our research results show that this method is computationally efficient and can be used for finding an optimal or near optimal path.

This paper is organized as follows. Section 2 introduces how to build a three-dimension-like map. Section 3 proposes the new path planning algorithm. Some examples are presented to evaluate the proposed algorithm in section 4. The conclusions from this work are provided in the last section.

## 2. Environment Modeling

In the initial stage, the environment is mapped using an appropriate mapping algorithm that represents the global workspace area where the robot works. This map will represent the start point, goal point, and differentiate the area that consists of obstacles and free space. In this paper, the environment is first modeled as a conventional grid map. That is, the workspace is tessellated into a square grid as illustrated in Fig. 1.

For each element representing obstacle space, a 1 is stored in it, and a 0 is assigned to other elements. According to different positions, the obstacle can be divided into three types: connecting to one or two sides of the boundary or not, as shown in Fig. 1.

Next, the foregoing grid map can be improved including height information. As we know, the contour line is usually used to describe the height information of mountain in geographical studies. In other words, if the plane contour is known, the height information of mountain is determined.

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press

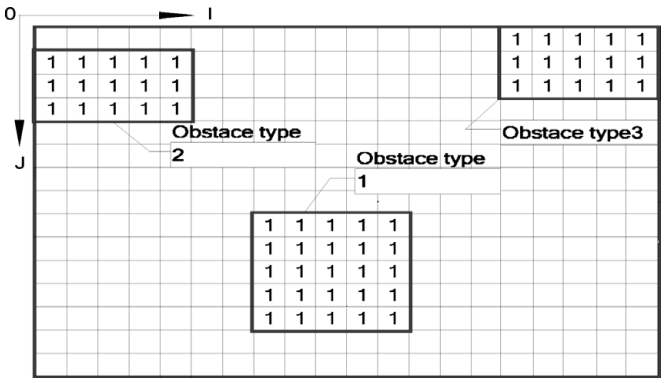



Fig. 1. A grid map including three obstacle types.

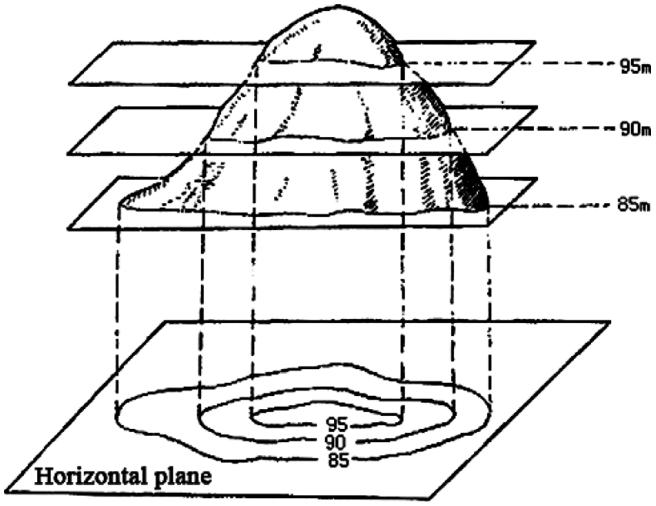



Fig. 2. The contour of mountain.

Figure 2 shows an illustration of mountain contours. From Fig. 2, the value of the contour gradually reduces from the highest point to outside.

Similarly, each grid point can be assigned height value by following rules:

**Rule 1:** The height value of grid points belong to free space is equal to zero.
**Rule 2:** The height value of grid points belong to obstacle space is a positive integer.
**Rule 3:** Starting from the highest point, the height gradually reduces by 1 from inside to outside.

It should be noted that the positions of the highest points are different for different obstacles. The highest point of type 1 that does not touch the boundary of map is located at its center. For other two types, the highest point is located at grid points connecting with the boundaries. Taking Fig. 1 as an example, a detailed description is given about how to build a three-dimension-like map. The procedure for dealing with the obstacle type 1 can be summarized as follows.

**Step 1:** Find the *I*- and *J*-coordinates of the minimum and maximum grid points, which are denoted by $I_{\min}$, $I_{\max}$, $J_{\min}$, and $J_{\max}$, respectively.
**Step 2:** As shown in Fig. 3, assign value 1 to the grid points which *I*-coordinates are $I_{\min}$ or $I_{\max}$. Assign value 2 to grid points which *I*-coordinates are $I_{\min} + 1$ or $I_{\max} - 1$. And so on until all grid points are successfully processed.
**Step 3:** Similarly to step 2, plus value 1 to the grid points which *J*-coordinates are $J_{\min}$ or $J_{\max}$. Plus value 2 to the grid points which *J*-coordinates are $J_{\min} + 1$ or $J_{\max} - 1$. And so on until all grid points have been successfully processed. Then a complete three-dimensional-like map about obstacle 1 has been obtained, as shown in Fig. 4.

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press

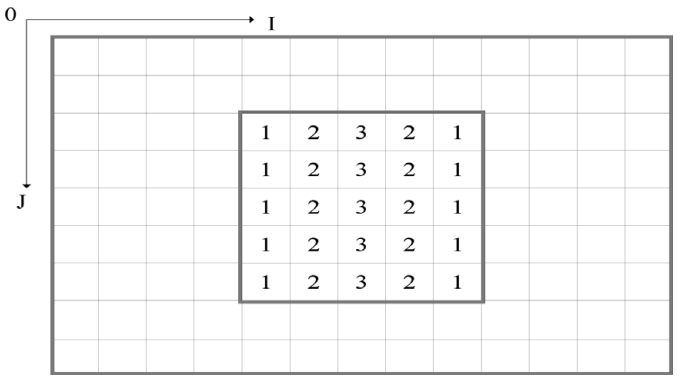



Fig. 3. Assign height value to obstacle 1 according to row order.

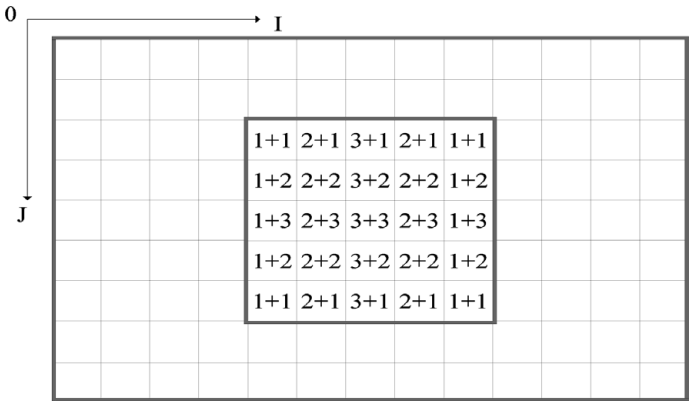



Fig. 4. Assign height value to obstacle1according to column order.

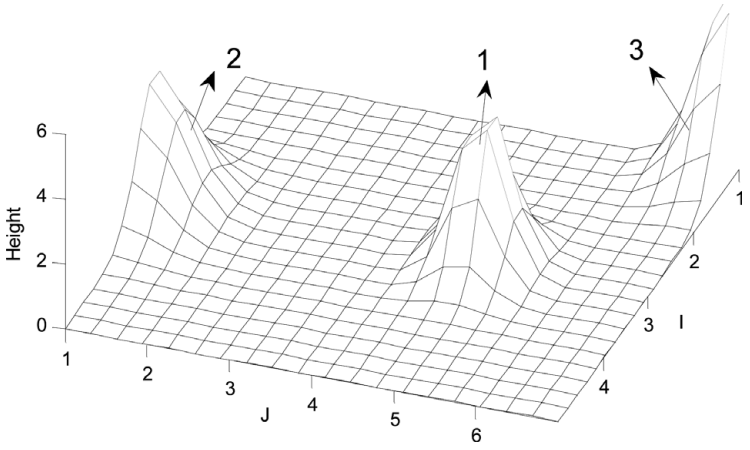



Fig. 5. An illustration of three-dimensional-like map for different obstacle category.

The procedures for obstacle 2 and 3 just make some minor modifications to the above. When all obstacles have been processed, a complete three-dimensional-like map is obtained. Figure 5 shows an illustration of three-dimensional-like maps about obstacles 1–3 in Fig. 1.

It should be noted that the different obstacle shapes will be transformed into the polygon forms in the above grid map. Take the circular obstacle in Fig. 6 as an example, the grids intersected with edge of the circular are regarded as obstacle regions, and number 1 is assigned to these elements. Finally, the circular obstacle can be approximated with a polygon. Moreover, the shape error can be reduced by increasing the density of the grids. A similar process can be used to other obstacle shapes.

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press

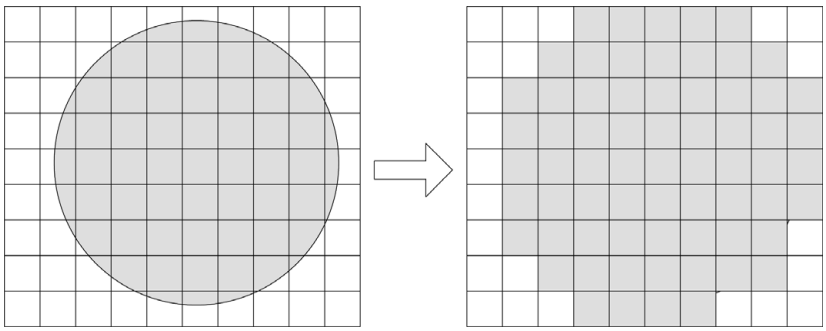


Fig. 6. An illustration of how to handle the obstacles that have no corner points.

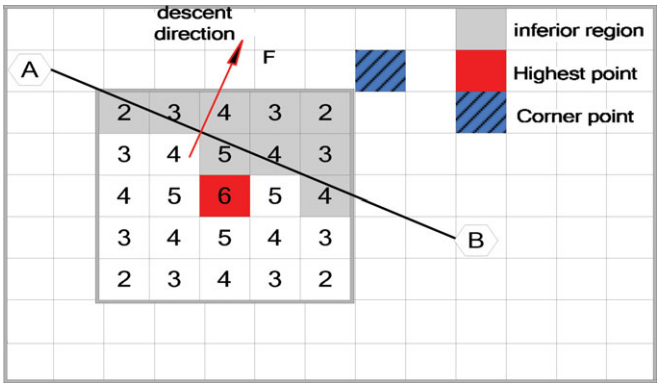



Fig. 7. (Colour online) This figure illustrates how to determine descent direction of obstacle.

## 3. Path Planning Based on a Three-Dimensional-Like Map

For convenience, some related concepts are defined and introduced as follows:

**Definition 1: Initial Planning.** Joining start and goal point without considering if it is a feasible path or not.

The coordinates $(I_x, J_x)$ of grid points lying in the initial path is

$$\begin{cases} I_x, & I_m \le I_x \le I_n, \\ J_x = [k(I_x - I_m) + J_m],\ k = \frac{J_m - J_n}{I_m - I_n}, \end{cases} \tag{1}$$

where $I_m$, $J_m$ are the $I$- and $J$-coordinates of the start point. $I_n$, $J_n$ are the $I$- and $J$-coordinates of the goal point. The expression $k$ represents the slope. The symbol **[]** denotes rounding up and down operation.

**Definition 2: Descent Direction.** It is defined as vertical direction starting from the highest of the each obstacle regions to the path.

The highest point and the points in the inferior region of obstacle are on the different side of the initial path, as shown in Fig. 7. Then, their coordinates satisfy the following inequality:

$$\{(I_y - I_m)k - (J_y - J_m)\} \times \{(I_{\max} - I_m)k - (J_{\max} - J_m)\} \le 0, \tag{2}$$

where $I_{\max}$, $J_{\max}$ are the $I$- and $J$-coordinates of the highest point of the obstacle. According to Definition 2, the descent direction just points to the inferior region of obstacle. Therefore, the descent direction can be obtained indirectly via Eq. (2).

**Definition 3: Basic Planning.** Modify the parts of path lines intersecting with obstacle regions. The modified paths can go along the inferior contours of the obstacles.

Next, we demonstrate the procedures of the proposed algorithm with some examples.

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press

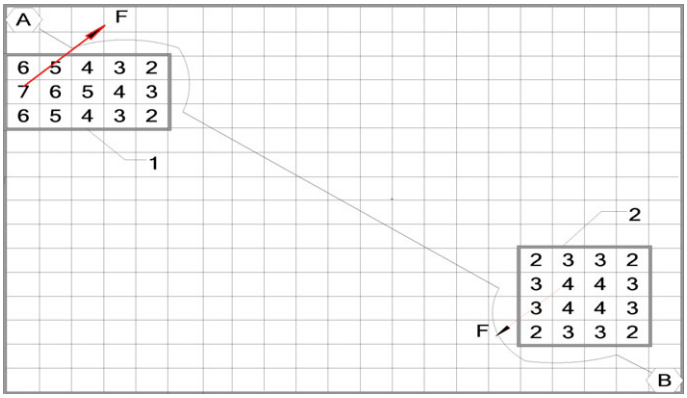



Fig. 8. (Colour online) An access but unoptimal path between points A and B.

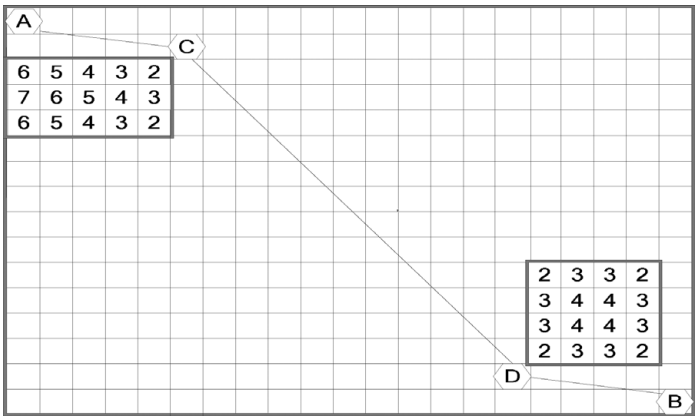



Fig. 9. The shortest possible path between points A and B.

***Example 1:*** Figure 8 shows a simple obstacle avoidance example. The points A and B are start and goal point, respectively. Line AB is initial planning path that intersects with obstacles 1 and 2. F represents the descent direction. For obstacle such as obstacle 1 connecting with boundary, the descent direction F points out a successful obstacle avoidance direction. For obstacle 2, the descent direction F determines the inferior contour of obstacle. When the descent direction F is determined, an access path can be obtained after applying basic planning. However, it obviously is not the optimal path. As shown in Fig. 9, to obtain a shorter path, we can find the points C and D, which we called the *corner point* in our algorithm. Connect points A, C, D, and B in turn. A desired feasible path ACDB can be obtained.

***Example 2:*** In Fig. 10, the initial path AB crosses three obstacles. The points C, D, and E are the *corner points*. If we connect the points A, C, D, E, and B, in turn, and an access path ACDEB can be obtained. Also, this path is not an optimal. To obtain a shorter path, we can use the following procedures: first, determine the relationship between the positions of the adjacent corner points C and D. If two points are on the same side of the initial path AB, compare the distances from two points to the line AB. Point A will be connected with the further point between C and D. In this example, the point D is further from the line AB than point C. Then, the path ACD is replaced by the shorter path AD. Next, determine the relationship between point D and the next corner point E. If two points are also on the same side of the line AB, the process is similar to the previous. Contrarily, they are connected directly. Finally, a path ADEB can be obtained. Obviously, this is an access and optimal path.

For the general cases, the procedures are composed of the foregoing steps. The complete solution procedure is summarized in Fig. 11.

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press

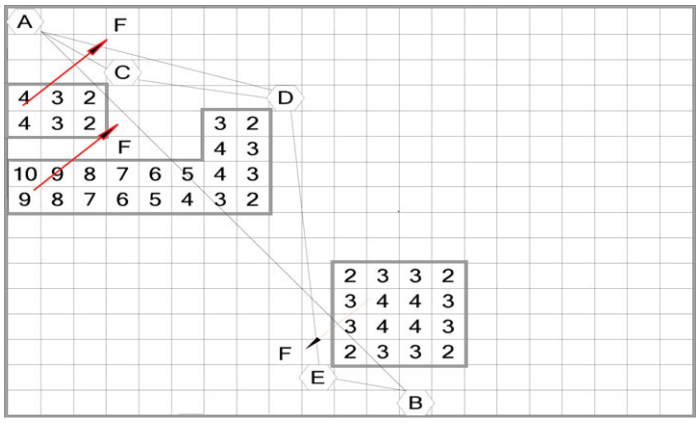



Fig. 10. (Colour online) This example illustrates the procedures of the proposed algorithm in multiobstacles environment.

During the initialization phase in Fig. 11, there exist four procedures: Modeling the environment as a three-dimensional-like map, initial planning, find all corner points of the obstacles intersecting with initial path and join the starting point to the first corner point.

An optimal or suboptimal path can be obtained by using the proposed algorithm. A brief proof is given below. In Fig. 12, the symbols S and G denote the starting point and goal, respectively. From this figure, the following conclusions can be derived:

**Case 1:** If the initial path SG does not intersect with any obstacles, it is obviously the shortest feasible path.

**Case 2:** If the initial path SG intersects with obstacles and path SBG is a feasible path, it can be proved that the path SBG is the shortest as follows:
Assume that there exits other feasible paths between the starting point and goal that pass through the point O in region 1. Obviously, the path SOBG is the shortest path among them. As we know, the sum of the lengths of any two sides of a triangle always exceeds the length of the third side. Therefore, the length of line SB is always less than the broken line SOB, as shown in Fig. 11. Similarly, assume that the point O in region 2 is located at some feasible paths, the path SOG is the shortest among them. From Fig. 11, it is clear that the length of broken line SBG is less than SOG. Notice that the positions of point O are arbitrary in the above discussions. That means the path SBG is the shortest among all feasible paths.

**Case 3:** If the path SB or BG in Fig. 12 still intersects with obstacles, similar to the above, the shortest path joining the points S and B or the points B and G can be further determined. However, it is noted that a combination of these shortest paths is not necessary to be the shortest from the points S to G. Therefore, if the global path combined these shortest paths is feasible, it is a suboptimal path. Furthermore, if the obtained path is still not feasible, repeat similar steps until a suboptimal collision-free path is obtained. Similar conclusion can also obtained intuitively by the following simulation results.

## 4. Simulations and Discussions

In this section, some typical maps are constructed to examine the properties of the proposed algorithm. The algorithm is coded in VC++6.0. All simulations are conducted on a Celeron (R) CPU 2.80-GHz PC.

From Fig. 13, test results show that an access path can be obtained for all cases. In fact, as long as the access paths exist, one of them must be successfully obtained using our algorithm. Although we cannot prove that they must be shortest possible paths, it is clear from these figures that the results are feasible and desirable. In order to evaluate the run-time performance, the computations required by the proposed method are given in Table II.

In Table II, $0 \leq k_1 \leq (I_m - I_n)$, $1 \leq k_2 \leq N_{ob}$, where $I_m$, $I_n$ are the $I$-coordinate of the start and the goal point, and $N_{ob}$ is the total number of obstacles. Operation 1 is used to check whether the

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press

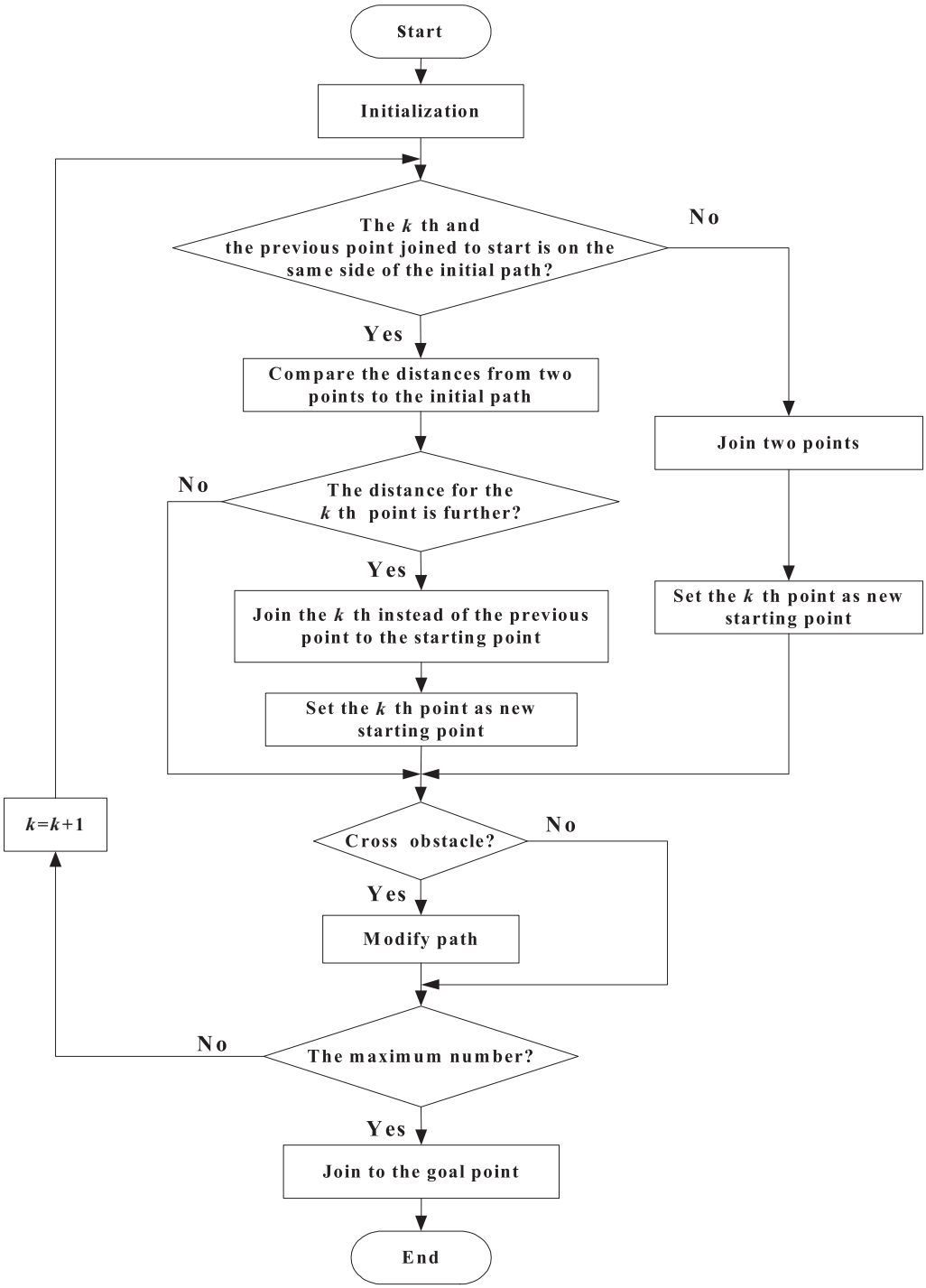



Fig. 11. The flow chart of the proposed algorithm.

current path intersects with obstacles (see Eq. 1). The aim of Operation 2 is to find the corner points of obstacles that lie in the current path (see Eq. 2). The distances from corner points to current path are determined by Operation 3, where the distance formula between point and line is used.

It is well known that path planning problem is NP-complete. Therefore, the exact computational requirements are difficult to be obtained in advance. For the sake of convenience, we consider the worst case for the proposed method. That is, to find an available path, all obstacles have to be dealt with. From Table II, a feasible path (if exists) can be obtained by using the proposed method under a finite number of computations. It should be noted that there are needs for other several

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press

Table II. The computational requirements for the proposed method.

| Operation | Computations required |
|---|---|
| 1 | Add. $(2k_1k_2)$ ; Mult. $(k_1k_2)$ |
| 2 | Add. $(6k_2 \times 4)$ ; Mult. $(2k_2 \times 4)$; Cross product. $(k_2 \times 4)$ |
| 3 | Add. $(4k_2 \times 2)$; Mult. $(3k_2 \times 2)$; Root operation. $(k_2 \times 2)$ |
| *Total* | Add. $(\leq [32N_{\text{ob}} + 2(I_m - I_n)N_{\text{ob}}])$; Cross product. $(\leq 4N_{\text{ob}})$ |
| | Mult. $(\leq [14N_{\text{ob}} + (I_m - I_n)N_{\text{ob}}])$; Root operation. $(\leq 2N_{\text{ob}})$ |

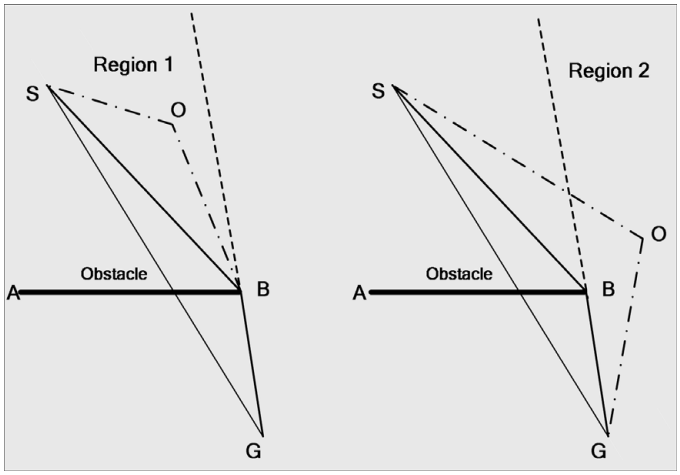



Fig. 12. The schematic diagram illustrates the shortest path SBG.

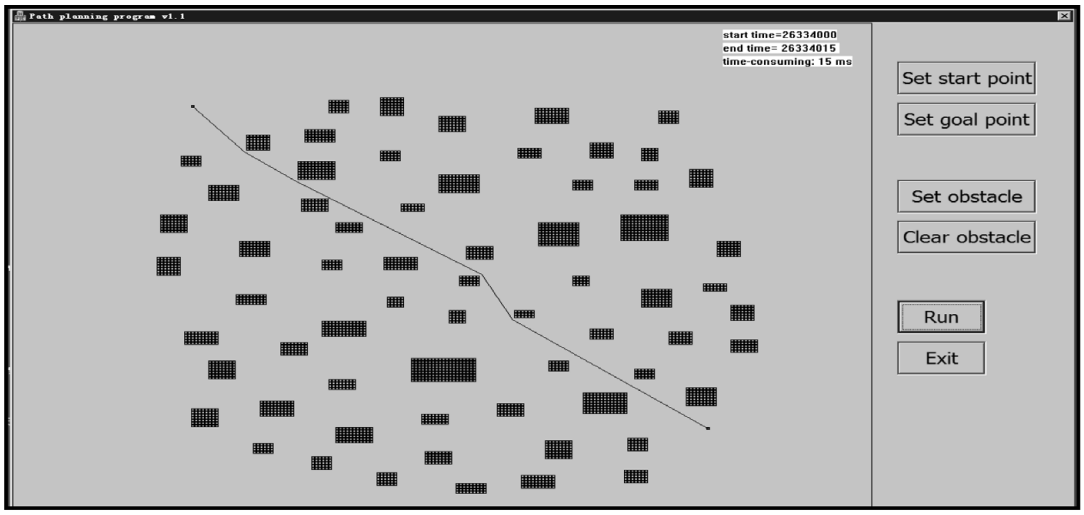
comparison operations in computations. Moreover, the operations for environment modeling are inevitable. Fortunately, these processes of comparison and assignment operations are very simple and fast. Therefore, the approximate results in Table II can demonstrate the efficiency of the proposed method.

As we know that the running time of most paths planning algorithms is proportional to number of obstacles. Furthermore, for some algorithms such as Visibility Graph method, the time complexity is increasing according to the law of exponential growth with the increasing of obstacles. Taking this into consideration, we analyze the influence of obstacle numbers on computation time of the proposed method. During the test, the shapes and positions of the obstacles are randomly generated, and the corresponding times for different number of obstacles are recorded and illustrated in Fig. 14. For the sake of fairness, the results are recorded in terms of the average times over 10 runs.

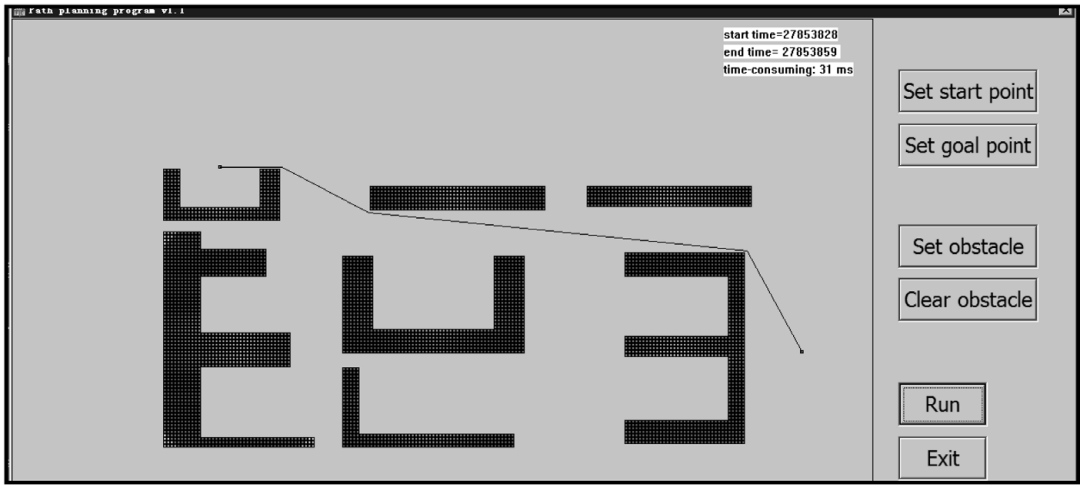
From Fig. 14, the search times for all cases are just tens of milliseconds. Notice that the running time is inevitably disturbed by other programs running on PC, these indicate that obstacle number does not greatly affect the running time of the proposed algorithm. Moreover, some effective features for the proposed method can also ensure the efficiency of planning. Take Fig. 15 for example, although the numbers of the obstacles are different, only one obstacle needs to be dealt with for two cases. Therefore, the major difference in computational time only comes from the process of modeling environment. This can greatly improve the efficiency of planning.

The proposed method and other common planning methods are compared and the results are summarized in the following. One of efficient solution methods for path planning is metaheuristic approach, including ant colony optimization (ACO) algorithm, genetic algorithm (GA), etc. The major advantages of these methods are not only capable to find an acceptable solution but are also adaptable and robust. However, the metaheuristic approaches seem too slow in most cases. For example, in ref. [11], ant colony algorithm is applied to find the shortest path in a simplest situation, i.e., a small size (20×20) of grid network is used and there is no obstacle. An optimal path is found by ACO after about 30 s. By comparison, the shortest path can be obtained by the proposed method just needing to directly connect the start and goal point, and the computational requirements needed for this simple example include operations of 40 additions and 20 multiplications in total. Although a

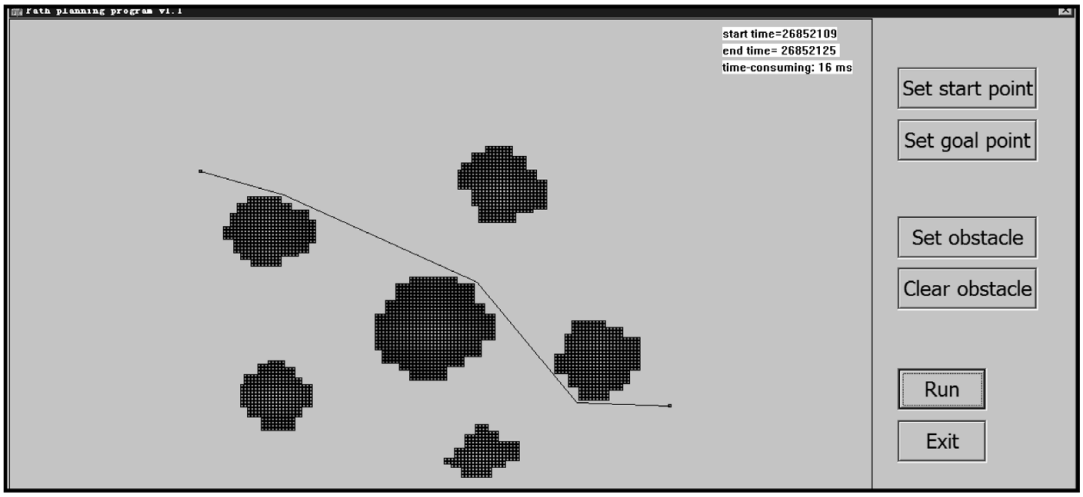
https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press




(a) Complex environment



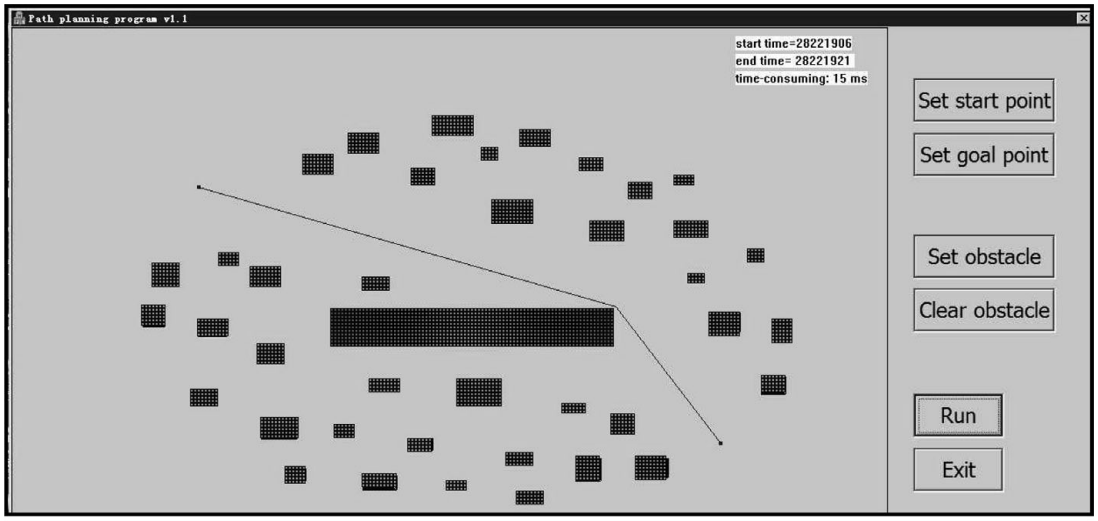

(b) A map with traps




(c) A map with irregular obstacles

Fig. 13. Some typical maps.

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press




(d) This example shows that, although there are several obstacles, only one need to be dealt with

Fig. 13. Continued

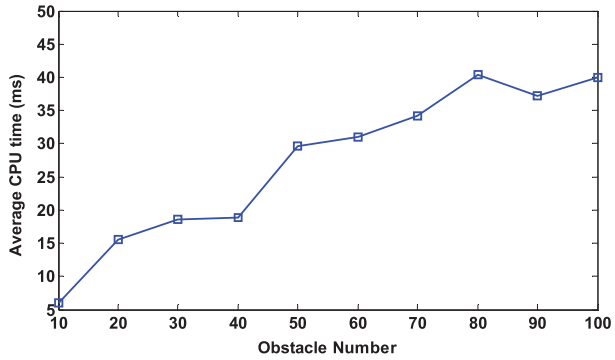



Fig. 14. (Colour online) The average running times for different obstacle numbers.

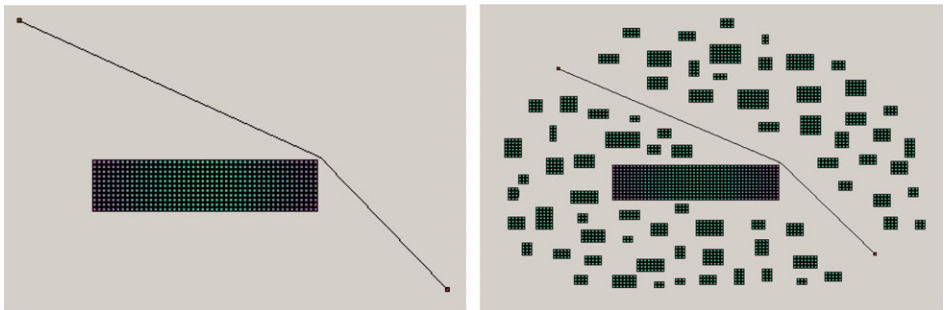


Fig. 15. (Colour online) A comparison of two path planning examples by using the proposed method.

simple example is used in ref. [14], to find an optimal path, the average computation times needed for Ant Colony System (ACS) algorithm and Genetic algorithm (GA) are 63.098 and 157.18 s, and with 8 and 4.4 average iterations, respectively. Therefore, the proposed method in this paper seems much faster than most of metaheuristic approaches except the method developed by TAN.[15] TAN presents a new and efficient globally optimal path planning method based on the ACS algorithm. The results show that the search time needed for generating the globally optimal path is just one-tenth of a second. This is an exciting result that indicates that the ACO-based method can be applied for a real-time path planning. However, although the researches have shown its great potential as a real-time motion planner, the algorithm efficiency is affected greatly by many parameters. Unfortunately, there is no

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press

Table III. Comparison of the proposed method with common path planning methods.

| Method | Environment | Local/Global | Real time performance | Algorithm complexity |
|---|---|---|---|---|
| The roadmap method | Static | Global | The time cost increases greatly with the increasing of the obstacles. | Need to combine shortest path search algorithms such as A* algorithm. |
| Potential field method | Static/dynamic | Local/global | Fast, but there exits trap situations due to local minima. | No reasonable mechanism to construct potential field functions. |
| Artificial intelligent method | Static/dynamic | Local/global | Slow in most cases | No reasonable mechanism to select suitable parameters such as network configuration. |
| Nonlinear programming method[17–21] | Static/dynamic | Local/global | Fast, the approach is suitable for online task planning. Local minima problems can be avoid. | The complexity is greatly reduced as the Cspace calculation is no longer needed. Mature search techniques can be applied directly. |
| The proposed method | Static | Global | Fast, time cost is not sensitive to obstacle number. | Only require a finite number of computations. |

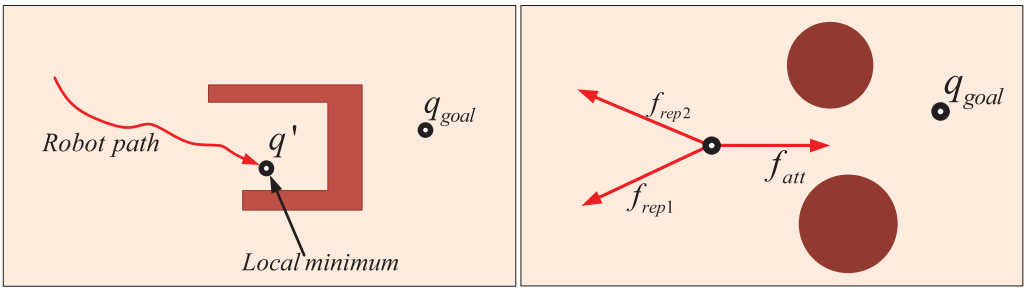



Fig. 16. (Colour online) Two examples of the local minimum problem with potential functions.

reasonable mechanism to choose suitable parameters in most cases. Moreover, the main parameters usually need to be adjusted for different maps.[16]

A different approach for path planning is potential field method. However, the major problem of this gradient-descent approach is that it does not guarantee a solution to the problem, as shown in Fig. 16. Another major challenge for potential-function-based approaches is constructing and representing the configuration space obstacles. By contrast, there are not such problems for the proposed method.

Visibility graph is another important path planning method. Each node in the visibility graph represents a point location, and each edge represents a visible connection between them. That is, if the line segment connecting two locations does not pass through any obstacle, an edge is drawn between them in the graph. Finally, the shortest paths among a set of polygonal obstacles can be found by applying the shortest path algorithm such as Dijkstra's algorithm to the graph. It should be point out that both visibility graph and the proposed method use the polygon obstacle's vertices. However, while all nodes in visibility graph are connected to one another, the method developed here only uses the optimal nodes. Consequently, the computations required by the proposed method are considerably less than those required by visibility graphs. The major advantage of visibility graph is that can guarantee an optimal solution to the problem. However, since computational efficiency is an important consideration in real practice, the proposed method in this paper can be seen as an effective alternative. Table III shows the performance comparison of the proposed method in this paper with other common methods.

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press

## 5. Conclusions

In this paper, we have presented a novel global path planning algorithm for mobile robot. A new environment model, called three-dimensional-like map, is developed to represent the global workspace area. The new model includes not only contour information but also height information of obstacles. Based on this new environment model, a simple but efficient obstacle avoidance rule is constructed to solve robot path planning problems in static environment. Our research results show that this method is computationally efficient and can be used for finding an optimal or near optimal path. Although this method does not guarantee to find the shortest path in all cases, since computational efficiency is an important consideration, it is still suitable. It should be noted that there is no essential difference between the global and local path planning. Therefore, a promising future research topic is to extend this method to other path planning areas. Moreover, as the geometry of mobile robot is ignored, how to keep the path generated by this method as far away from the obstacles as possible is still a problem in practical application. This problem may be resolved by coupling the method with techniques to avoid obstacle, or with the aid of obstacle avoidance sensors. Anyway, although simulation results show the effectiveness of the proposed method, there are still some open problems to apply this method in practice.

## Acknowledgements

The authors thank the anonymous reviewers for their invaluable suggestions, which have been incorporated to improve the quality of this paper dramatically.

## References

1. T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM* **22**, 560–570 (1979).
2. D. Leven and M. Sharir, "Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagrams," *Discrete Comput. Geom.* **2**, 9–31 (1987).
3. P. Bhattacharya and M. L. Gavrilova, "Roadmap-based path planning-using the Voronoi diagram for a clearance-based shortest path," *IEEE Robot. Autom. Mag.*, **15**, 58–66 (2008).
4. R. A. Brooks, "Solving the find-path problem by good representation of free space," *IEEE Trans. Syst. Man Cybern.* **13**, 190–197 (1983).
5. T. Arney, "An Efficient Solution to Autonomous Path Planning by Approximate Cell Decomposition," *Proceedings of the 3rd International Conference on Information and Automation for Sustainability*, Melbourne, VIC (4–6, 2007) pp. 88–93.
6. C. Alexopoulos and P. M. Griffin, "Path planning for a mobile robot," *IEEE Trans. Syst. Man Cybern.* **22**, 318–322 (1992).
7. A. Yahja, S. Singh and A. Stentz, "An efficient on-line path planner for outdoor mobile robots," *Robot. Auton. Syst.* **32**, 129–143 (2000).
8. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.* **5**, 90–98 (1986).
9. R. Glasius, A. Komoda and S. C. A. M. Gielen, "Neural network dynamics for path planning and obstacle avoidance," *Neural Netw.* **8**, 125–133 (1995).
10. P. Zhang, T. Lu and L. Song, "Soccer robot path planning based on the artificial potential field approach with simulated annealing," *Robotica* **22**, 563–566 (2004).
11. M. Brand, M. Masuda, N. Wehner and X. Yu, "Ant Colony Optimization Algorithm for Robot Path Planning," *Proceedings of the 2010 International Conference On Computer Design and Applications*, vol. 3 (2010) pp. 436–440.
12. N. Tsourveloudis, K. P. Valavanis and T. Hebert, "Autonomous vehicle navigation utilizing electrostatic potentialfields and fuzzy logic," *IEEE Trans. Robot. Autom.* **17**, 490–497 (2001).
13. A. Taharwa, A. Sheta and M. A. Weshah, "A mobile robot path planning using genetic algorithm in static environment," *J. Comput. Sci.* **4**, 341–344 (2008).
14. N. Buniyamin, N. Sariff, W. A. J. Wan Ngah and Z. Mohamda, "Robot global path planning overview and a variation of ant colony system algorithm," *Int. J. Math. Comput. Simul.* **1**, 9–16 (2011).
15. G. Z. Tan, H. He and A. Sloman, "Ant colony system algorithm for real-time globally optimal path planning of mobile robots," *Acta Autom. Sin.* **33**(3), 279–285 (2007).
16. X. Chen, Y. Kong, X. Fang and Q. Wu, "A fast two-stage ACO algorithm for robotic path planning," *Neural Comput. Appl.* **22**(2), 313–319 (2013).
17. Y. Wang and D. M. Lane, "Sub-sea vehicle path planning using nonlinear programming and constructive solid geometry," *IEEE Proc.* D *Control Theory Appl.* **144**(2), 143–152 (1997).
18. Y. Wang and D. M. Lane, "Solving a generalized constrained optimization problem with both logic AND and OR relationships by a mathematical transformation and its application to robot path planning," *IEEE Trans. Syst. Man Cybern. Appl. Rev.* **30**(4), 525–536 (2000).

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press

19. Y. Wang, H. Liu, M. Li, Q. Wang, J. Zhou and M. Cartmell, "A real-time path planning approach without the computation of Cspace obstacles," *Robotica* **22**(2), 173–187 (2004).
20. Y. Wang, M. Cartmell, Q. Tao and H. Liu, "A generalized real-time obstacle avoidance method without the cspace calculation," *J. Comput. Sci. Technol.* **20**(6), 774–787 (2005).
21. Y. Wang, D. M. Lane and G. J. Falconer, "Two novel approaches for unmanned underwater vehicle path planning: constrained optimization and semi-infinite constrained optimization," *Robotica* **18**, 123–142 (2000).

https://doi.org/10.1017/S0263574713000738 Published online by Cambridge University Press