Hindawi
Journal of Robotics
Volume 2017, Article ID 8796531, 11 pages
https://doi.org/10.1155/2017/8796531

Research Article

A Global Path Planning Algorithm Based on Bidirectional SVGA

Taizhi Lv,[1] Chunxia Zhao,[1] and Jiancheng Bao[2]

[1]*School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China*
[2]*School of Information Technology, Jiangsu Maritime Institute, Nanjing 211170, China*

Correspondence should be addressed to Taizhi Lv; lvtaizhi@163.com

Received 3 August 2016; Revised 29 November 2016; Accepted 4 January 2017; Published 2 February 2017

Academic Editor: Yuan F. Zheng

For path planning algorithms based on visibility graph, constructing a visibility graph is very time-consuming. To reduce the computing time of visibility graph construction, this paper proposes a novel global path planning algorithm, bidirectional SVGA (simultaneous visibility graph construction and path optimization by A*). This algorithm does not construct a visibility graph before the path optimization. However it constructs a visibility graph and searches for an optimal path at the same time. At each step, a node with the lowest estimation cost is selected to be expanded. According to the status of this node, different through lines are drawn. If this line is free-collision, it is added to the visibility graph. If not, some vertices of obstacles which are passed through by this line are added to the OPEN list for expansion. In the SVGA process, only a few visible edges which are in relation to the optimal path are drawn and the most visible edges are ignored. For taking advantage of multicore processors, this algorithm performs SVGA in parallel from both directions. By SVGA and parallel performance, this algorithm reduces the computing time and space. Simulation experiment results in different environments show that the proposed algorithm improves the time and space efficiency of path planning.

1. Introduction

Intelligent mobile robots have been widely used not only in military activities but also in civil life. Navigation is the key problem of the mobile robot technology. The quandary of navigation could be summarized: "Where am I? Where do I go? How do I get there? [1]" The first two problems are about localization and mapping, and the last problem is about path planning.

Path planning is to determine a collision-free path between the start and target position by a performance criterion such as the distance, time, and energy consumption [2]. According to whether an environment is known or not, there are two categories of path planning algorithms, namely, local and global path planning. A mobile robot plans a feasible path in an unknown or dynamic environment by the information which is gotten from sensors. This is known as local or online path planning. Global path planning is to search for an optimal path based on complete information about stationary obstacles and is also known as offline path planning.

Classical local path planning approaches include artificial potential field, vector field histogram, dynamic windows, and bug. However these algorithms suffer from some drawbacks, such as trapping in local minima and unsmooth planned path. To overcome such limitations, there have emerged some proposals combining evolutionary or heuristic algorithms with classical path planning approaches, such as parallel evolutionary artificial potential field (PEAPF) [3] and Egress-Bug [4]. Global path planning consists of two steps: environmental modeling and path optimization. Configuration space (C-Space) is a fundamental approach to environmental modeling problem. Cell decomposition and roadmap are well known environmental modeling approaches based on C-Space. The simplest cell decomposition is grid with a fixed resolution. The main difficulty of this approach is how to determine the size of cell. Some improved approaches are proposed to solve this difficulty, such as fan-shaped grid map [5] and quad-tree grid map [6]. Voronoi diagram and visibility graph are two main roadmap approaches. Voronoi diagram is proposed by Dunlaing and Yap. This approach constructs a roadmap by using points equidistant from two

or more obstacles [7]. Artificial intelligent path planning methods apply modern artificial intelligent technology to mobile robot path planning [8]. Artificial neural nets, fuzzy logic, genetic algorithm, and particle swarm optimization are widely applied technologies in the path planning research. These artificial intelligent technologies overcome many drawbacks of classic approaches, and they reinforce the robot intelligence. But these technologies are difficult to implement path planning independently, and they need to be combined with classic approaches [9]. Rapidly exploring random tree (RRT) is a sampling-based path planning algorithm and is being implemented heavily in recent years [10].

Visibility graph modeling is to construct a compact, undirected graph that registers visibility among vertices of obstacles [11]. For its simplicity, visualization, and completeness, visibility graph is still useful in many applications. However, constructing a visibility graph is very time-consuming. Some algorithms aiming efforts at reducing computing time are proposed by scholars. Tran et al. proposed a parallel-oriented visibility graph construction algorithm. This algorithm divides the environment into some parts and constructs the visibility graph for each part in parallel [12]. Zhang et al. use a simplified visibility graph suitable for path planning algorithm to model environment [13]. By ignoring redundant obstacles which do not affect the result of path planning, it improves the efficiency of path planning. Some techniques are used to improve the time complexity such as reducing the amount of visibility edges, simplifying obstacles to rectangles, and combining the tiny obstacles [14, 15]. Some intelligence algorithms such as quantized algorithm [16] and ant colony [17] are used in path planning based on visibility graph to improve the computational efficiency.

The above improved algorithms all firstly construct a visibility graph and then search for an optimal path based on this visibility graph. Constructing a visibility graph is very time-consuming and the optimization efficiency is drastically decreased with the increasing of edge number. To improve the computational efficiency, this paper proposes an improved global path planning algorithm, bidirectional SVGA (simultaneous visibility graph construction and path optimization by A*). Two approaches to increase the performance of global path planning are introduced to the proposed algorithm:

(i) It does not construct the visibility graph before the search process, but it constructs the visibility graph and searches for the path at the same time. Whether a direct line between two vertices is collision-free or not is the main process of the visibility graph construction. This process is called visibility judgment in short. In order to improve the efficiency of global path planning, the main purpose of the proposed algorithm is to reduce the executions of visibility judgment. Two strategies are used in this approach. One is to make use of the relationship between the start position, target position, and all vertices. Many vertices unrelated to the path planning result are ignored and visibility judgment between them does not need be executed. The other is to make use of the heuristics search. If a line between two vertices is more related to the optimal path, the visibility judgment of this line is executed earlier. It causes that the visibility judgment less related to the optimal path may not be executed.

(ii) SVGA is performed in parallel from both directions. By taking advantage of multicore processors, it improves the efficiency of global path planning.

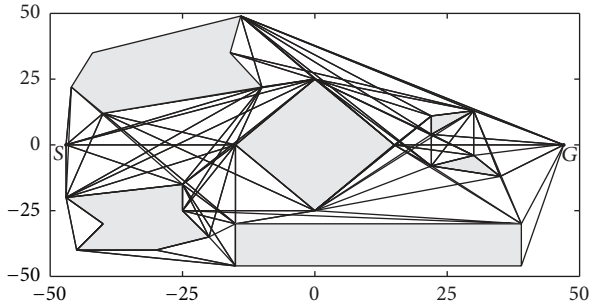

Figure 1: Complete visibility graph.

The rest of this paper is organized as follows. Section 2, at first, introduces the key concepts of global path planning based on visibility graph. The global path planning algorithm based on bidirectional SVGA is presented in Section 3. The performance analysis is presented in Section 4. The experimental results are provided in Section 5. Finally, Section 6 concludes this paper.

2. Global Path Planning

Global path planning can be described as a search problem in a four-atom formulation $\langle \mathbf{X}, x_{\text{start}}, x_{\text{goal}}, \mathbf{X}_{\text{obst}} \rangle$:

(i) $\mathbf{X}$: search space.

(ii) x_{start}: start position.

(iii) x_{goal}: target position.

(iv) $\mathbf{X}_{\text{obst}}$: obstacle set.

If a path series $\{u_1, u_2, \ldots, u_k\}$ is a solution, $u_1 = x_{\text{start}}$, $u_k = x_{\text{goal}}$, and $\{u_1, u_2, \ldots, u_k\} \cap \mathbf{X}_{\text{obst}} = \Phi$. Global path planning aims to search for an optimal solution in all solutions.

2.1. Visibility Graph. Visibility graph is constructed by joining the lines which connect the start position, target position, and vertices of obstacles. These lines do not intersect obstacles. In other words, these lines are visible. The vertices and lines form a visibility graph VG $= \{V, E\}$. Figure 1 shows a visibility graph where grey polygons represent obstacles, S is the start position, and G is the target position. After a complete visibility graph is constructed, the shortest path is then identified by some search algorithms.

2.2. A Algorithm.* Dijkstra is a goal-directed search algorithm. Based on Dijkstra, A* adds a potential function to the priority key of each node in the queue [18, 19]. The potential

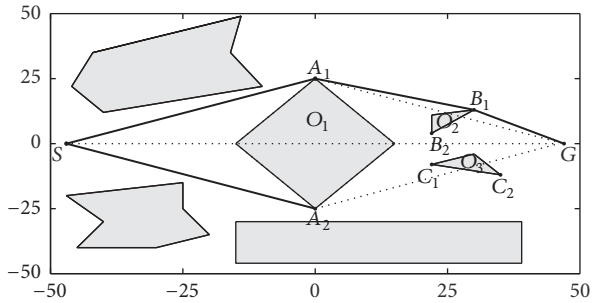

Figure 2: Unidirectional SVGA.

function is an estimation of the path length through the vertex v.

$$f(v) = g(v) + h(v). \tag{1}$$

$f(v)$ represents the estimated shortest path length between the start and target position through the vertex v. $g(v)$ is the real path length from the start position to the vertex v, and $h(v)$ is heuristics function which estimates the shortest path length from the vertex v to the target position.

3. Bidirectional SVGA

Classic global path planning algorithms based on visibility graph search for a path after the complete visibility graph roadmap is constructed. The time complexity of visibility graph construction is $O(N^3)$, where N is the number of vertices. To reduce the visibility graph construction time, this paper proposes the bidirectional SVGA algorithm which does not construct a complete visibility graph. This algorithm simultaneously constructs the visibility graph and searches for the optimal path by A*. In the optimization process only related edges are added to the visibility graph, and the added edges equally affect the optimization process. From both directions, SVGA is performed in parallel, for it can take advantage of multicore processors [20]. One direction is from the start position to the target position, and it is called forward SVGA. The other is from the target position to the start position, and it is called backward SVGA.

For convenience, unidirectional SVGA is firstly introduced. It performs SVGA from one direction. Figure 2 shows this process.

At each step of the optimization process, one through line is drawn. The through line is a straight line which can pass through obstacles to connect two vertices. In other words, the line would be the shortest path between the two vertices supposing the robot can pass through obstacles. According to this line, related vertices are added to the OPEN list for expansion.

Firstly, a through line from the start position S to the target position G is drawn. It can be seen that if the robot wants to go to G, it must pass by the vertex A_1 or A_2. A_1 and A_2 are added to the OPEN list. The node A_1 with the lowest estimation cost is expanded, and a through line from S to A_1 is drawn. Because this line does not pass through any obstacle, it is added to the visibility graph. Then a through line from A_1 to G is drawn, and B_1 and B_2 are added to the OPEN list. Next A_2 is selected to be expanded. The line from S to A_2 is added to the visibility graph, for the line is free-collision. A through line from A_2 to G is drawn. C_1 and C_2 are added to the OPEN list. B_1 is selected for its lowest estimation cost. A through line from A_1 to B_1 is drawn. Because this line is free-collision, it is added to the visibility graph and B_1 is continued to be expanded. A through line from B_1 to G is drawn, and it is also added to the visibility graph for there is no collision. Because the optimization arrives at G, this algorithm is finished and the optimal path $S \to A_1 \to B_1 \to G$ is output.

Bidirectional SVGA is similar to unidirectional SVGA, and the difference is that bidirectional SVGA is performed in parallel from both directions. For parallel performance, there are two OPEN lists, OPENF and OPENB. They are used in forward and backward SVGA, respectively.

$$\text{OPENF}(i) = \{index, prev, status, gn, hn, priority\}, \tag{2}$$

where OPENF(i) represents the ith node. Each node is related to a vertex, and *index* represents the index of this vertex in the vertex set V. *prev* represents the preview node. *status* represents whether there is a visible edge between this node and preview node or not. Different from the classic A* algorithm, $g(v)$ in this algorithm is not the real path length but an estimated path length from the start position to the current node. *gn* represents $g(v)$ and it is calculated as

$$gn = prev.gn + D_{pp}, \tag{3}$$

where D_{pp} represents the straight-line distance from the preview node to the current node. *hn* represents the heuristics function $h(v)$. It is the straight-line distance from the current node to the target position. Nodes of OPENF are not deleted in this algorithm for the same node might be visited repeatedly. *priority* represents the access priority and visit count. A dead circle in the optimization process can be avoided by the *priority* property. SVGA is based on the graph search and each node is identified by *index* and *prev*. It means that when *index* and *prev* are both equal, the nodes are the same. A node is represented as {*index*, *prev*} in a simplified form. The structure of OPENB is the same as OPENF.

Two CLOSED lists, CLOSEDF and CLOSEDB, are used in this algorithm. Each node is defined as

$$\text{CLOSEDF}(i) = \{index, prev, gn\}, \tag{4}$$

where *index* is the index in V. *gn* is the real path length between the start position and the current node.

The proposed algorithm consists of two steps: initialization and optimization. In the initialization process, two OPEN lists and two CLOSED lists are initialized. The start position, target position, and vertices of all obstacles are added to the vertex set V.

$$V = \{S, G, vertices\}. \tag{5}$$

Only edges between adjacent vertices of some obstacles are added to the edge set E.

$$E = \{edges_{ajacent}\}. \tag{6}$$

The target position G is added to OPENF, and the start position S is added to OPENB.

$$\begin{aligned} \text{OPENF}(1) &= \{G, S, 0, D_{SG}, 0, 0\}, \\ \text{OPENB}(1) &= \{S, G, 0, D_{SG}, 0, 0\}, \end{aligned} \tag{7}$$

where D_{SG} is the Euclidean distance between S and G, $priority = 0$ represents that the node has not been expanded, and $status = 0$ represents that the visibility between the two nodes is unknown.

Two CLOSED lists are initialized as

$$\begin{aligned} \text{CLOSEDF}(1) &= \{S, 0, 0\}, \\ \text{CLOSEDB}(1) &= \{G, 0, 0\}. \end{aligned} \tag{8}$$

The node is the initial node if *prev* equals 0.

Only forward SVGA is specified. Backward SVGA is the same as forward SVGA. At each step of the optimization process, the node with the lowest estimation cost is selected to be expanded. The estimation function is defined as

$$f(v) = gn + hn - priority \times \text{MAX}, \tag{9}$$

where MAX represents the max possible distance from S to G. After this expansion, the *priority* value of this node minuses one.

Different through lines are drawn according to the *status* value of this node. A through line from the preview node to this node is drawn provided that *status* equals 0. A function visible(v_i, v_j, E_s) is used to determine whether a line between the vertices v_i and v_j is visible or not. E_s is the set of boundaries of obstacles. If the line intersects any obstacle, the function visible(v_i, v_j, E_s) returns false. If this line is visible, it is added to E, and this node is added to CLOSEDF, and *status* is assigned to 1.

$$\begin{aligned} &E = \{E, (cNode.prev.index, cNode.index)\}, \\ &\text{CLOSEDF}(size + 1) \\ &\quad = \{cNode.index, cNode.prev, cNode.gn\}, \\ &cNode.status = 1, \end{aligned} \tag{10}$$

where *cNode* represents the current node. If this node exists in CLOSEDB, it means that forward and backward SVGA meet the current node. The optimization is finished and the optimal path is output. If this line passes through an obstacle, no more than two vertices of this obstacle are added to OPENF. If a vertex of this obstacle is farthest from the line in any direction of two directions and is not in OPENF, it is added to OPENF.

$$\begin{aligned} &\text{OPENF}(size + 1) \\ &\quad = \{index_p, cNode.prev, 0, cNode.prev.gn \\ &\quad + D_{PP}, D_{PG}\}, \end{aligned} \tag{11}$$

where $index_p$ is the index of this vertex in V, D_{PP} is the straight-line distance from preview node to the vertex, and D_{PG} is the straight-line distance from the vertex to G. The pseudo-code of adding vertices of the crossed obstacles to the OPEN list are shown below.

```
Procedure addAllVertices(node, line, crossedObs)
(01) For (o in crossedObs)
(02)     P_high = maxDistanceFromTop(o, line);
(03)     If (exists(P_high, node.prev) == false)
(04)         addVertex(P_high, node.prev);
(05)     End If
(06)     If (moreFarLineS2G(P_high, node) == true
             && exists(P_high, S) == false)
(07)         addVertex(P_high, S);
(08)     End If
(09)     P_low = maxDistanceFromBotton(o, line);
(10)     If (exist(P_low, node.prev) == false)
(11)         addVertex(P_low, node.prev);
(12)     End If
(13)     If (moreFarLineS2G(P_low, node) == true
             && exists(P_low, S) == false)
(14)         addVertex(P_low, S);
(15)     End If
(16) End For
End Procedure
```

If the *status* value of this node equals 1, a through line from the current node to G is drawn. If this line has been in E or is free-collision, the optimization is finished and the optimal path is output. If the line passes through an obstacle, no more than two vertices are added to OPENF.

$$\begin{aligned} &\text{OPENF}(size + 1) \\ &\quad = \{index_p, cNode, 0, cNode.gn + D_{CP}, D_{PG}\}, \end{aligned} \tag{12}$$

where D_{CP} is the distance from the current node to the new node.

The flowchart of forward SVGA is shown as Figure 3. Parallel performance causes different stop criterions. If the following four stop criterions are satisfied, bidirectional SVGA is over and the optimal path or no path is output.

(i) Forward SVGA arrives at G.

(ii) Backward SVGA arrives at S.

(iii) Forward and backward SVGA meet the same node.

(iv) The *priority* value of the node with the lowest estimation cost is less than 0.

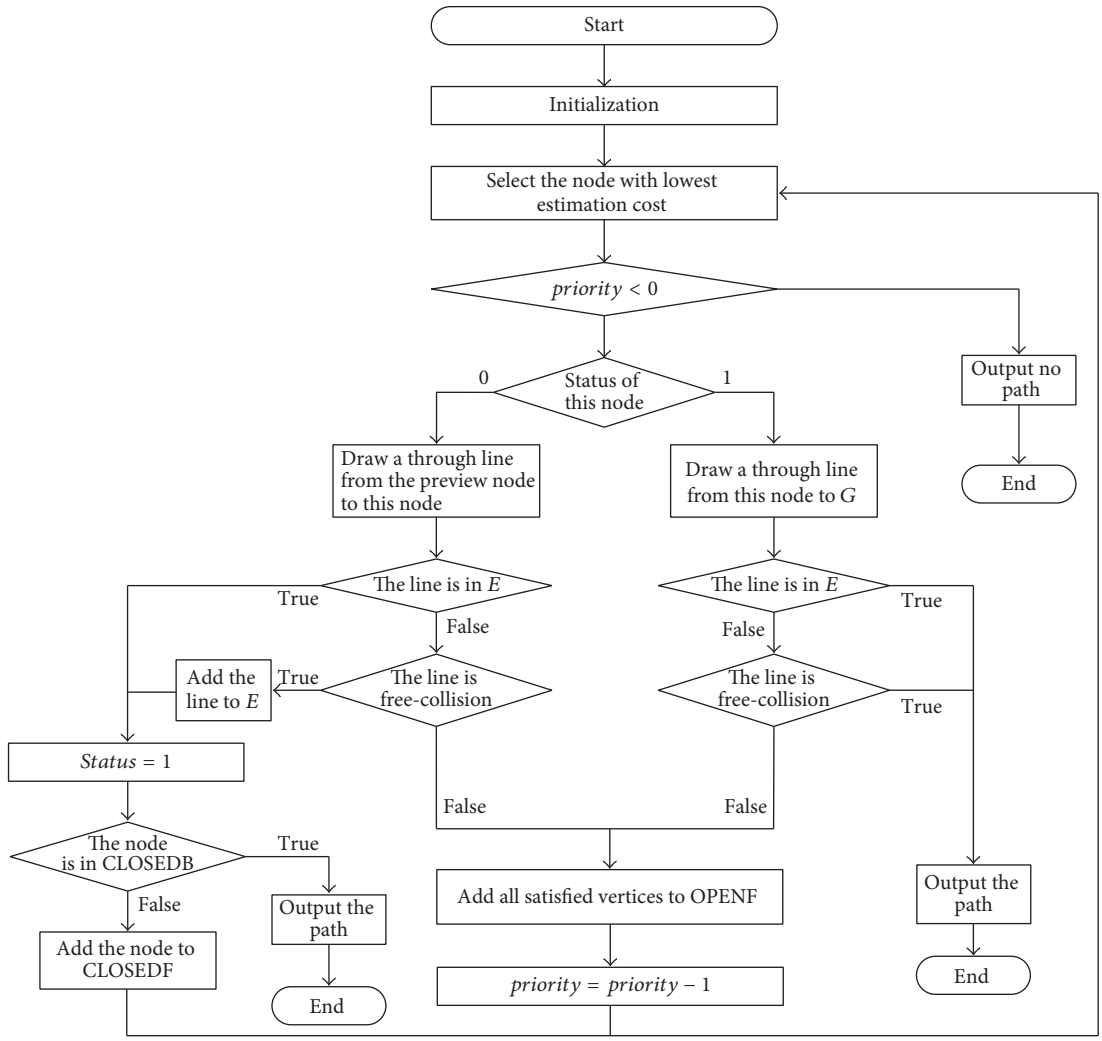

Figure 3: Forward SVGA.

4. Performance Analysis

4.1. Completeness. If SVGA satisfies the following conditions in an environment with limited vertices, it is complete.

(1) SVGA is guaranteed to find a solution when there is a feasible path between the start and target positions.

(2) SVGA is guaranteed to be finished in finite steps when there is no path.

SVGA can plan a path by through lines when there is a feasible path between the start and target positions. At each step, a robot passes through the vertices far away from the through line to avoid the obstacles. These vertices are added to the OPEN list for expansion. If there are obstacles between these vertices and the start position or the target position, SVGA continues drawing the through lines and the vertices of these obstacles are also added to the OPEN list. By the finite through lines, a mobile robot can avoid the obstacles to arrive at the target position.

When all possible paths are explored, the *priority* of every node is less than zero. It indicates there is no path between the start and target positions and SVGA is over.

Because bidirectional SVGA executes unidirectional SVGA at most twice, it is complete when unidirectional SVGA is complete.

4.2. Optimality. Firstly, the path planned by SVGA is proved to be optimal, and then the path planned by bidirectional SVGA is proved to be optimal.

Two conditions are required for optimality. One is that the incomplete visibility graph constructed by SVGA includes all edges on the optimal path. The other is that SVGA can find the optimal path from this incomplete visibility graph.

The first step is to prove that the incomplete visibility grapy includes all edges on the optimal path. If the through line $S \rightarrow G$ does not cross any obstacle, it is an optimal path. This line is added to the visibility graph as a visible edge and incomplete visibility graph includes the edge of this optimal

path. If this line is invisible, there are some obstacles between S and G. There are only two ways which let a robot skirt an obstacle through the optimal path. The first is that the robot goes through a vertex of this obstacle which is the farthest from the line in any direction of two directions. The second is that the robot goes through a position which is far away from this obstacle. It can be seen from Figure 2 that if the robot skirts the obstacle O_1, it must pass through the vertex A_1 or A_2 or a position which is farther from the through line than A_1 or A_2.

(a) The robot skirts an obstacle by the vertex P which is the farthest from the trough line in any direction of two directions. The vertex P is added to the OPEN list when SVGA draws the through line $S \rightarrow G$. Because P is on the optimal path, it must be selected for expansion. The through line $S \rightarrow P$ is drawn. If this line is visible, it is added to the visibility graph. If this line is invisible, the robot should skirt the crossed obstacles to arrive at P. The robot must go through the vertices of these obstacles which are farthest from the through line in any direction of two directions, or far away from these obstacles. By this recursive inference based on the two ways, it can be concluded that the vertices on the optimal path from S to P are added to the OPEN list, and the edges on the optimal path are added to the visibility graph.

(b) The robot skirts an obstacle by a position which is far away from this obstacle. It is based on the following two cases.

(1) The first case is that there is a vertex Q of the other obstacle which is farther away from the through line than the vertex P of this obstacle. The robot goes from S to Q or goes from Q to G by skirting this obstacle. When the through line $S \rightarrow G$ is drawn, the vertices P and Q are both added to the OPEN list. The through lines from S to Q and from Q to G are drawn. If these lines are visible, they are added to the visibility graph. If these lines are invisible, the robot skirts these obstacles by the two ways.

(2) The other case is that there is an obstacle between S and P or between P and G, and the vertex Q of this obstacle is farther away from the through line $S \rightarrow G$ than P. If this obstacle is between S and P, the through lines $S \rightarrow Q$ and $Q \rightarrow G$ are drawn. The edges between these vertices are added to the visibility graph by through lines. If this obstacle is between P and G, the nodes $\{Q, P\}$ and $\{Q, S\}$ are both added to the OPEN list for expansion. If there are some obstacles between these vertices, that the first condition is satisfied can be proven by the recursive inference based on the two ways.

No matter in any case, the visibility graph constructed by SVGA includes all edges on the optimal path by this recursive inference based on the two ways.

The next step is to prove that SVGA can find the optimal path based on this visibility graph. The estimation function of SVGA is nondecreasing. x_n is supposed as the successor of node x.

$$f(x_n) = g(x_n) + h(x_n) = g(x) + D_{x\_x_n} + h(x_n), \quad (13)$$

where $D_{x\_x_n}$ is Euclidean distance between x_n and x. By the general triangle inequality, it can be concluded that

$$h(x) \leq D_{x\_x_n} + h(x_n) \Longrightarrow \quad f(x) \leq f(x_n). \quad (14)$$

Because the estimation function is nondecreasing, SVGA determines a node to be expanded according to a nondecreasing sequence. The first selected target node for expansion is the optimal solution [21]. The estimation cost of all succor nodes is not less than the first selected target node.

It is assumed that forward and backward SVGA meet the same node when the expanded node of one directional SVGA is in the CLOSED list of the other directional SVGA. Bidirectional SVGA is over when forward and backward SVGA meet the same node. This node may be the start position, target position, or the vertex of an obstacle. The path from the start position to this vertex found by forward SVGA and the path from target position to this vertex found by backward SVGA are both optimal, so the found path is optimal.

4.3. Complexity

4.3.1. Time Complexity. SVGA uses A* algorithm to determine a node to be expanded and through lines to determine vertices to be added to the OPEN list. The time complexity of A* algorithm is $O(N^2)$. SVGA adds new nodes by visibility judgment of a through line. The time complexity of the function visible(v_i, v_j, E_s) is $O(N)$, so the time complexity of SVGA is $O(N^3)$, the same grade as complete visibility graph construction. But a construction of the complete visibility graph involves all vertices. This construction needs to call the function visible$(v_i, v_j, E_s)N \times (N-1)$ times. SVGA ignores the vertices which are independent of the optimal path, and the involved vertices are much fewer than the vertices of complete visibility graph construction. In Figure 1, there are 28 vertices. The complete visibility graph construction needs to call the function visible(v_i, v_j, E_s) 756 times. By ignoring the most vertices, SVGA only call the function visible(v_i, v_j, E_s) 7 times. It can be seen that the computing time of SVGA is far lower than the complete visibility graph construction and bidirectional SVGA is faster than SVGA for parallel performance.

4.3.2. Space Complexity. The huge memory requirements of global path planning algorithms based on visibility graph are the visible edges. The space complexities of complete visible graph and SVGA are both $S(N^2)$, but memory requirements of SVGA are fewer than the complete visibility graph. The complete visibility graph keeps all visible edges in memory, but SVGA ignores the most visible edges and keeps a few visible edges related to the optimal path. As shown in Figures 1 and 2, complete visibility graph includes 107 visible edges and SVGA only includes 33 visible edges. The number of visible edges in bidirectional SVGA is no more than simply twice the number of SVGA.

5. Experimental Results and Discussion

The simulation experiments are carried out to validate the effectiveness of SVGA and to compare SVGA with some global path planning algorithms based on visibility graph. The first algorithm searches for an optimal path by A* after a complete visibility graph is constructed. It is called complete

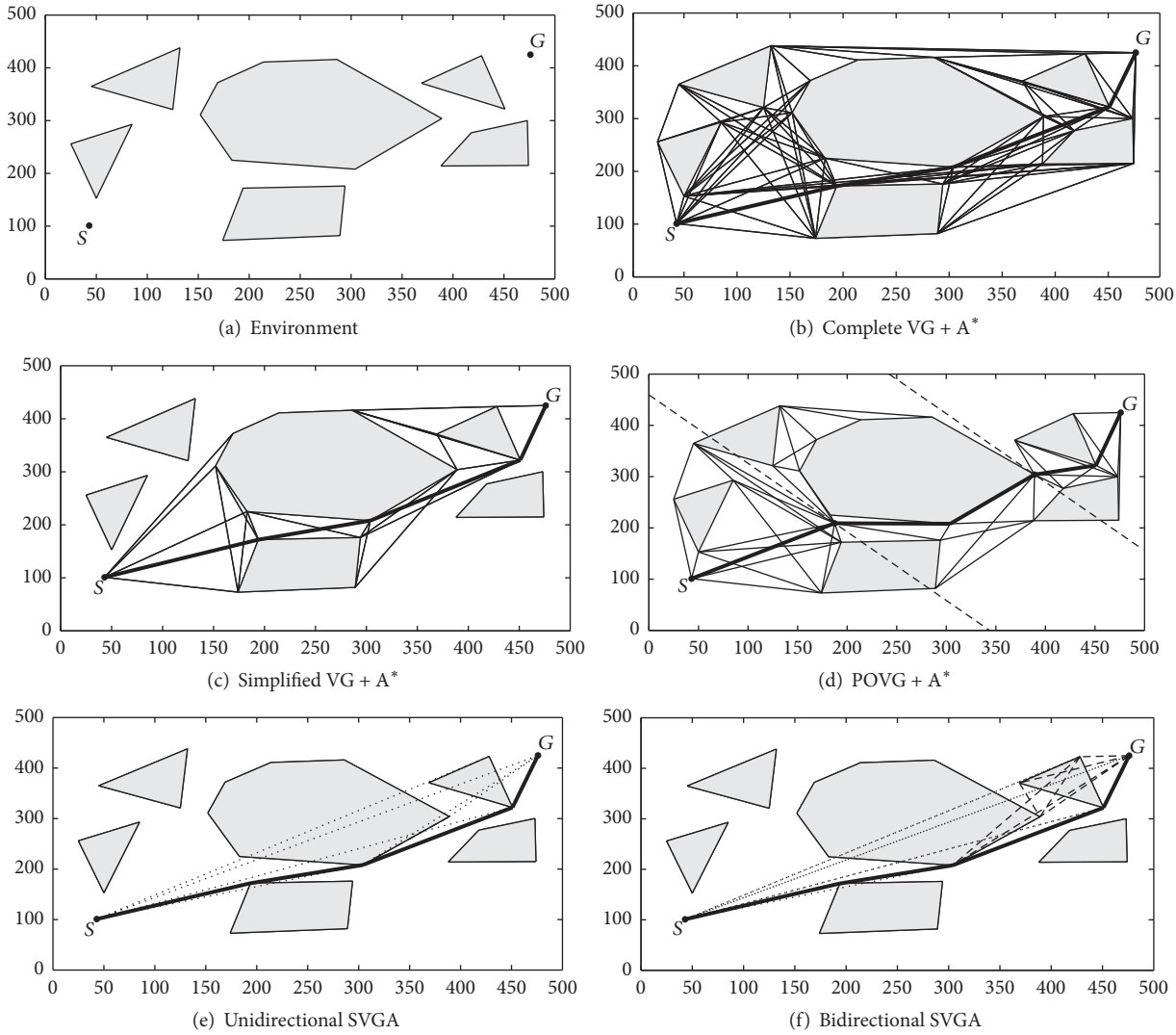

Figure 4: Comparison between five algorithms in an environment.

VG + A* in short. The second algorithm uses a simplified visibility graph to implement environmental modeling and A* to search for an optimal path [13]. It is called simplified VG + A* in short. In this algorithm, redundant obstacles which do not affect the path planning result are removed by considering positions of obstacles, the start and target points. The third algorithm uses parallel-oriented visibility graph to implement environmental modeling and A* to plan a path [12]. This algorithm combines the modified visibility graph and parallel computation to improve computing time. It is called POVG + A* in short. The fourth is unidirectional SVGA and the last is bidirectional SVGA. For a fair comparison between these algorithms, they are tested by using the same 2D environment with obstacles.

Firstly, the environment shown in Figure 4(a) is used to be tested for the comparison between five algorithms. There are 6 obstacles and 29 vertices including S and G. The position of S is at (43, 101), and the position of G is at (476, 425). All algorithms except POVG + A* can find the optimal path, and the length of this path is 574.61.

The complete VG + A* algorithm constructs a complete visibility graph shown as Figure 4(b). The construction of this visibility graph takes 0.66 s and 106 visible edges are added to the visibility graph. The function visible(v_i, v_j, E_s) is called 812 times. This algorithm uses A* algorithm to search for an optimal path based on the visibility graph. The optimization process takes 0.05 s. At the end of the optimization, the size of the OPEN list is 67. It can be seen that the main phase of this algorithm is to construct the visibility graph, and it is very time-consuming. Simplified VG + A* algorithm constructs a simplified visibility graph shown as Figure 4(c). By ignoring the redundant obstacles for the optimal path, the number of visible edges is reduced to 38. This algorithm takes 0.39 s, including the visibility graph construction time 0.36 s and the optimization time 0.03 s. In the visibility graph construction process, the function visible(v_i, v_j, E_s) is called 240 times. The size of the OPEN list is 13 at the end of the optimization. POVG + A* takes 0.31 s to find the optimal path, and it is shown in Figure 4(d). This algorithm divides the environment into the three regions. In each region,

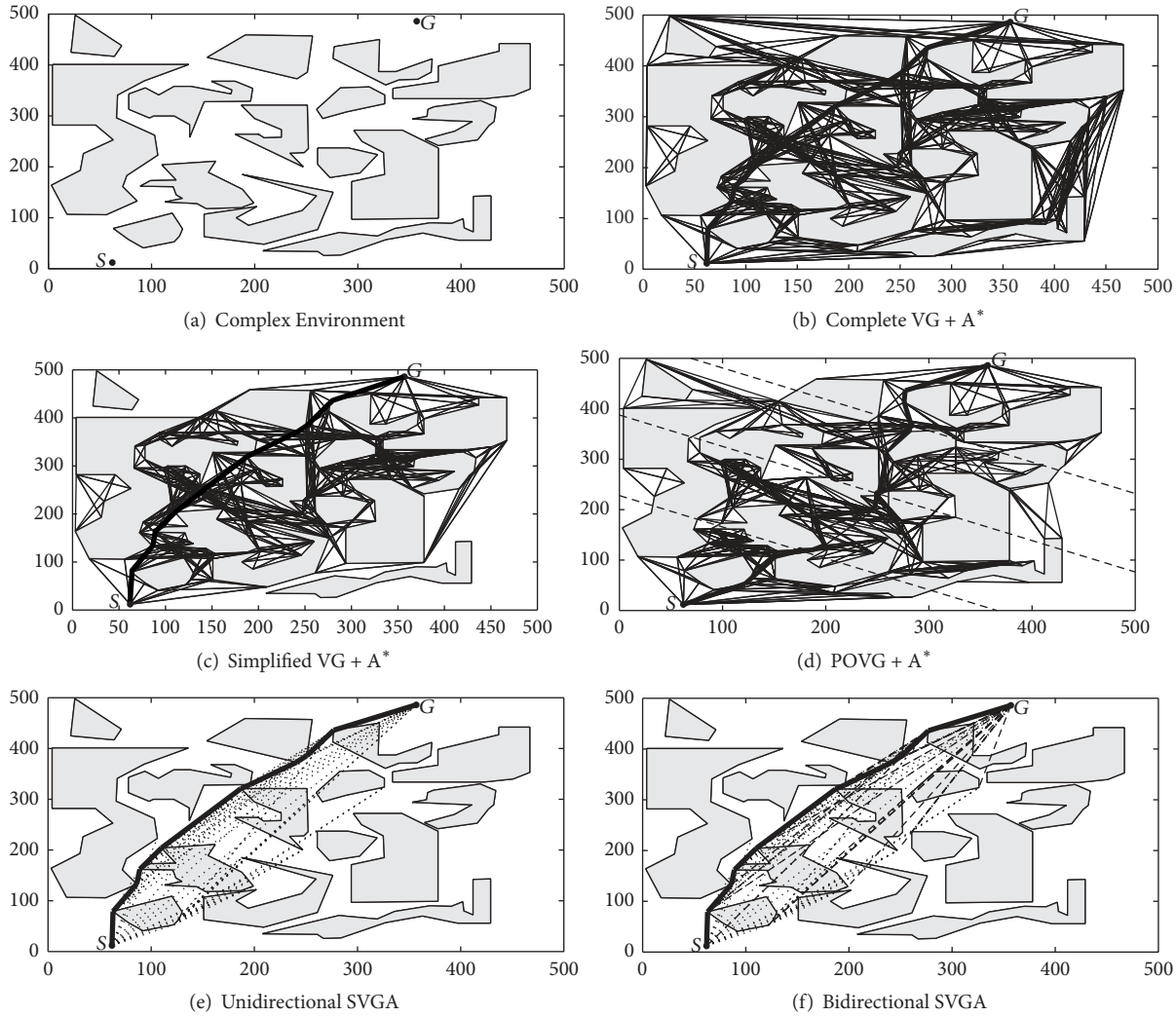

Figure 5: Comparison between five algorithms in a complex environment.

this algorithm constructs the visibility graph and finds the path, respectively. At last, three subpaths are combined to a complete path. POVG + A* adds 70 edges to the visibility graph and calls the function visible(v_i, v_j, E_s) around 90 times for each region.

By ignoring the vertices independent of the optimal path, SVGA improves the path planning efficiency. By 6 through lines and 4 edges, unidirectional SVGA finds the optimal path. It only calls the function visible(v_i, v_j, E_s) 10 times. It takes 0.06 s, and the size of the OPEN list is 6. This process is shown in Figure 4(e). Because parallel performance causes different execution sequences, the time and space consumption of each performance for bidirectional SVGA are a little different from one another even though in the same environment. Figure 4(f) shows the result of one performance. In this performance, the running time is 0.05 s, and the sum size of OPENF and OPENB is 15. Comparing with the other three algorithms, whether unidirectional or bidirectional SVGA improves the efficiency of global path planning based on visibility graph enormously. For computational time, there are ninety percent improvement comparing with the first algorithm, eighty-four percent improvement comparing with the second algorithm, and eighty percent improvement comparing with the third algorithm. Comparing unidirectional SVGA, bidirectional SVGA takes less computational time and more computational space.

Next, another comparison is carried out in a complex environment. This environment is shown in Figure 5(a). There are 15 obstacles and 165 vertices in the environment. S is at (62, 12) and G is at (375, 486).

As shown in Figure 5(b), complete VG + A* draws 790 visible edges. The total running time is 24.79 s, including the visibility graph construction time 23.59 s and the optimization time 1.20 s. The time of visibility graph construction increases quickly for the function visible(v_i, v_j, E_s) is called 27060 times with the growth of the vertex number. In fact, many visibility judgments are independent of the optimal path searching. A large number of visible edges cause the more time and space consumption on the path optimization. At the end of the optimization, the size of the OPEN list

Table 1: Experimental results in 2D environment with randomly generated obstacles.

Algorithms	Number of obstacles	Average number of vertices	Average number of edges in E	Average running time/s	Best running time/s	Worst running time/s
Complete VG + A*	6	41	171	0.82	0.67	1.38
Simplified VG + A*			76	0.35	0.002	1.19
POVG + A*			131	0.32	0.19	0.45
Unidirectional SVGA			36	0.05	0.002	0.19
Bidirectional SVGA			40	0.04	0.002	0.17
Complete VG + A*	9	60	288	2.84	1.38	5.58
Simplified VG + A*			183	1.87	0.002	5.18
POVG + A*			174	0.87	0.64	1.15
Unidirectional SVGA			59	0.12	0.002	0.34
Bidirectional SVGA			64	0.09	0.002	0.31
Complete VG + A*	12	79	405	6.75	4.87	7.32
Simplified VG + A*			308	5.06	0.27	6.50
POVG + A*			203	1.85	1.44	3.12
Unidirectional SVGA			84	0.18	0.01	0.85
Bidirectional SVGA			92	0.14	0.01	0.76
Complete VG + A*	15	95	535	11.96	7.38	16.42
Simplified VG + A*			410	9.18	6.24	17.45
POVG + A*			235	2.73	1.54	4.74
Unidirectional SVGA			117	0.40	0.17	1.63
Bidirectional SVGA			131	0.32	0.14	1.47
Complete VG + A*	18	113	601	17.28	14.82	31.18
Simplified VG + A*			516	15.02	8.73	32.29
POVG + A*			269	4.12	2.89	7.26
Unidirectional SVGA			140	1.03	0.21	4.07
Bidirectional SVGA			167	0.81	0.18	4.39

is 325. Simplified VG + A* takes 21.32 s and constructs 663 visible edges. The running time of the simplified visibility graph construction is 20.17 s, and the running time of the path optimization is 1.15 s. At the end of the optimization, the size of the OPEN list is 310. From Figure 5(c), it can be seen that there is a little improvement comparing with complete VG + A* for most obstacles are related to the optimal path. POVG + A* takes 7.84 s to find a feasible path. It divides the environment to four regions and is shown in Figure 5(d). This algorithm adds 323 edges to the visibility graph and calls the function visible(v_i, v_j, E_s) 7918 times for all regions. Unidirectional SVGA takes 1.34 s to find the optimal path, and the size of the OPEN list is 129. Figure 5(e) shows the optimization process. By the through lines and the heuristics search, the optimal path can be quickly found by SVGA. This algorithm only called the function visible(v_i, v_j, E_s) 65 times. Bidirectional SVGA takes 0.97 s to find the optimal path and it shown in Figure 5(f). At the end of the optimization, the sum size of OPENF and OPENB is 173. Whether in time or space consumption, unidirectional and bidirectional SVGA are far better than the other three algorithms.

To further validate the efficiency of the proposed algorithm, five algorithms are compared in a 150 × 150 area. The max length of edges is 30, and the vertex number of each obstacle is no more than 10. Five categories of environments are used in this comparison. The numbers of obstacles in different categories are not the same, and they are 6, 9, 12, 15, and 18, respectively. Each category of environment is randomly generated 100 times. The results are shown in Table 1.

It can be seen whether unidirectional or bidirectional SVGA is better than the other three algorithms in any environment. Even though in the environment with 18 obstacles, the proposed algorithm can find an optimal path in one second for most test cases. However the average running time of complete VG + A* is 17.28 s and 31.18 s in the worst condition.

Unidirectional SVGA, bidirectional SVGA, and simplified VG + A* all firstly draw a direct line between the start and target position. If this line is collision-free, the search processes of the three algorithms are over. If there are very few obstacles or no obstacles which affect the robot going to the target position, unidirectional SVGA, bidirectional SVGA, and simplified VG + A* all can find a path quickly. Based on these environments, the running times of three algorithms are all around 0.002. It can be seen from Table 1 that the best running time of the three algorithms are all 0.002 which is accurate to the third-decimal place. But, in most

conditions, the running time of bidirectional SVGA is less than unidirectional SVGA and simplified VG + A*.

The space complexity of the SVGA algorithms is also better than the other three algorithms. It can be concluded that the SVGA algorithms can improve the time and space complexity of the path planning based on visibility graph roadmap. The efficiency difference between unidirectional SVGA and bidirectional SVGA is that unidirectional SVGA takes more computational time and less computational space, and, on the contrary, bidirectional SVGA takes less computational time and more computational space.

6. Conclusions

This paper presents a global path planning algorithm based on bidirectional SVGA. This algorithm constructs the visibility graph and searches for an optimal path simultaneously. It takes advantage of the vertex positions and heuristics search, and most visibility judgments between two vertices are ignored. By reducing executions of visibility judgment, this algorithm improves the efficiency of path planning on time and space. Much of the recent improvement in computer speed is due to multicore processors. This algorithm takes advantage of multicore processors and adapts the path planning to parallel processing. From both directions, it executes SVGA in parallel.

To validate the efficiency of the proposed algorithm, different simulation environments are tested. These environments include a simple environment, a complex environment, and five different categories of environment in a 150 × 150 area. These simulation experiments all validate the effectiveness of the SVGA algorithm.

The proposed algorithm is to improve the global path planning based on visibility graph. It is assumed that the environment is known and static, and the vertex positions of all obstacles should be known in advance. In many applications, environments are dynamic or unknown. The further research is to apply SVGA in the dynamic environments.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work is financially supported by the National Natural Science Foundation of China (no. 61101197). Taizhi Lv would like to thank Jiangsu Overseas Research & Training Program for University Prominent Young & Middle-Aged Teachers and Presidents for financial support. Taizhi Lv also would like to thank the Qianfan project of Jiangsu Maritime Institute for financial support.

References

[1] M. Algabri, H. Mathkour, H. Ramdane, and M. Alsulaiman, "Comparative study of soft computing techniques for mobile robot navigation in an unknown environment," *Computers in Human Behavior*, vol. 50, pp. 42–56, 2015.

[2] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: a review," *International Journal of Physical Sciences*, vol. 7, no. 9, pp. 1314–1320, 2012.

[3] O. Montiel, R. Sepúlveda, and U. Orozco-Rosas, "Optimal path planning generation for mobile robots using parallel evolutionary artificial potential field," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 79, no. 2, pp. 237–257, 2015.

[4] K. R. Guruprasad, "EgressBug: a real time path planning algorithm for a mobile robot in an unknown environment," in *Proceedings of the International Conference on Advanced Computing, Network and Security*, pp. 228–236, Surathkal, India, 2011.

[5] T. Li, S. Sun, and Y. Gao, "Fan-shaped grid based global path planning for mobile robot," *Robot*, vol. 32, no. 4, pp. 547–552, 2010.

[6] L. J. Guo, W. X. Shi, Y. Li, and F. X. Li, "Mapping algorithm using adaptive size of occupancy grids based on quadtree," *Control and Decision*, vol. 26, no. 11, pp. 1690–1694, 2011.

[7] M. Shao and K. Shin, "Sensor-based path planning for planar two-identical-link robots by generalized voronoi graph," *Journal of the Korea Academia-Industrial Cooperation Society*, vol. 15, no. 12, pp. 6986–6992, 2014.

[8] Y. Gigras and K. Gupta, "Artificial intelligence in robot path planning," *International Journal of Soft Computing & Engineering*, vol. 2, no. 2, pp. 471–474, 2012.

[9] D. Q. Zhu and M. Z. Yan, "Survey on technology of mobile robot path planning," *Control and Decision*, vol. 25, no. 7, pp. 961–967, 2010.

[10] V. T. Huynh, M. Dunbabin, and R. N. Smith, "Convergence-guaranteed time-varying RRT path planning for profiling floats in 4-Dimensional flow," in *Proceedings of the Australian Conference on Robotics and Automation*, pp. 1–10, Melbourne, Australia, 2014.

[11] H.-P. Huang and S.-Y. Chung, "Dynamic visibility graph for path planning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, pp. 2813–2818, IEEE, Sendai, Japan, October 2004.

[12] N. Tran, D.-T. Nguyen, D.-L. Vu, and N.-V. Truong, "Global path planning for autonomous robots using modified visibility-graph," in *Proceedings of the 2nd International Conference on Control, Automation and Information Sciences (ICCAIS '13)*, pp. 317–321, NhaTrang, Vietnam, November 2013.

[13] Q. Zhang, J.-C. Ma, and L.-Y. Ma, "Environment modeling approach based on simplified visibility graph," *Journal of Northeastern University*, vol. 34, no. 10, pp. 1383–1391, 2013.

[14] L. I. Ping, J. Y. Zhu, F. Peng, and L. Yang, "Path planning based on visibility graph and A* algorithm," *Computer Engineering*, vol. 40, no. 3, pp. 193–195, 2014.

[15] T. T. N. Nguyet, T. V. Hoai, and N. A. Thi, "Some advanced techniques in reducing time for path planning based on visibility graph," in *Proceedings of the 3rd International Conference on Knowledge and Systems Engineering (KSE '11)*, pp. 190–194, Hanoi, Vietnam, October 2011.

[16] J. Kim, M. Kim, and D. Kim, "Variants of the quantized visibility graph for efficient path planning," *Advanced Robotics*, vol. 25, no. 18, pp. 2341–2360, 2011.

[17] J. Huang and Y. Cen, "A path-planning algorithm for AGV based on the combination between ant colony algorithm and immune regulation," *Advanced Materials Research*, vol. 422, pp. 3–9, 2012.

[18] F. Duchoň, A. Babinec, M. Kajan et al., "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[19] S. M. Persson and I. Sharf, "Sampling-based A* algorithm for robot path-planning," *International Journal of Robotics Research*, vol. 33, no. 13, pp. 1683–1708, 2014.

[20] M. Phillips, M. Likhachev, and S. Koenig, "PA∗SE: parallel A∗ for slow expansions," in *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS '14)*, pp. 208–216, Portsmouth, NH, USA, June 2014.

[21] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson Prentice Hall, Upper Saddle River, NJ, USA, 3rd edition, 2010.