applied sciences

MDPI

Article

A Path Planning Strategy for Multi-Robot Moving with Path-Priority Order Based on a Generalized Voronoi Diagram

Sheng-Kai Huang [1], Wen-June Wang [1,*] and Chung-Hsun Sun [2]

[1] Department of Electrical Engineering, National Central University, Taoyuan 320, Taiwan; 105581006@cc.ncu.edu.tw
[2] Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 807, Taiwan; chsun@nkust.edu.tw
* Correspondence: wjwang@cc.ncu.edu.tw

Featured Application: This study proposes a new path planning method called the navigation strategy with path priority for multiple robots moving. It also completely designs the robot's action strategy. The proposed study is suitable for handling or moving more than one robot in a trackless factory environment.

Abstract: This paper proposes a new path planning strategy called the navigation strategy with path priority (NSPP) for multiple robots moving in a large flat space. In the space, there may be some static or/and dynamic obstacles. Suppose we have the path-priority order for each robot, then this article aims to find an efficient path for each robot from its starting point to its target point without any collision. Here, a generalized Voronoi diagram (GVD) is used to perform the map division based on each robot's path-priority order, and the proposed NSPP is used to do the path planning for the robots in the space. This NSPP can be applied to any number of robots. At last, there are several simulations with a different number of robots in a circular or rectangular space to be shown that the proposed method can complete the task effectively and has better performance in average trajectory length than those by using the benchmark methods of the shortest distance algorithm (SDA) and reciprocal orientation algorithm (ROA).

Keywords: Voronoi diagram; Dijkstra algorithm; multi-robot path planning; collision-free; path-priority order

Citation: Huang, S.-K.; Wang, W.-J.; Sun, C.-H. A Path Planning Strategy for Multi-Robot Moving with Path-Priority Order Based on a Generalized Voronoi Diagram. *Appl. Sci.* **2021**, *11*, 9650. https://doi.org/10.3390/app11209650

Academic Editor: Augusto Ferrante

Received: 13 August 2021
Accepted: 12 October 2021
Published: 15 October 2021

1. Introduction

Based on the technological development of industry 4.0 [1], robots can accomplish many tasks in smart factories. Therefore, many workers in smart factories have been replaced by robots. A single robot can accomplish some tasks, but others with high complexity and/or in large spaces may require multiple robots to work together [2]. Decreasing transit time, increasing system reliability, path planning, and collision avoidance are challenging problems for cooperative multi-robot groups. Path planning is one of the challenges, and it can be divided into single-robot path planning [3,4] and multi-robot path planning [5,6]. In this paper, we deal with the problem of path planning for multiple robots in a dynamic environment.

There have been a lot of papers studying the problem of single-robot or multi-robot path planning. For the path planning of a single robot, Reference [7] created a robot navigation system that presented a path generation method within a maze of arbitrary complexity and optimized the generated path with the appropriate boundary conditions. An immunity algorithm is proposed by [8], in which the robot could perform its task through optimal path planning and minimal rotation angle efficiency. Traditional and heuristic approaches to path planning were proposed, such as the potential field method and A* algorithm, as seen in papers [9,10]. Reference [11] chose the roadmap method

and utilized a Voronoi diagram to calculate the optimal path between a single source and a single destination with some polygonal obstacles. Reference [12] proposed a jump point search (JPS) A* algorithm to improve computational time and path optimality and compared it with other algorithms like basic Theta*, Phi*, and A* in different scenarios. A modified genetic algorithm (MGA) based on the Bezier curve method was proposed by [13], in which the robot's path is dynamically determined based on the obstacles' locations and improved the exploration capabilities. In the study of multi-robot path planning, Reference [14] proposed an improved gravitational search algorithm (IGSA) in a dynamic environment to optimize path trajectories for multiple robots. An improved particle swarm optimization (IPSO) algorithm and an improved gravity search algorithm (IPSA) are combined to achieve a collision-free smooth optimal path from the beginning position to the end position in [15]. A hierarchical model predictive (HMP) control was introduced by [16,17] to address the heuristic search methods that cannot provide path stability through theoretical guarantees for multiple robots. Reference [18] emphasized a team of robots in formation control mode for robot navigation in two-dimensional and three-dimensional spaces with mobile obstacles. Recently, there have been some works on coverage path planning, which has become a relevant topic. In paper [19], a flock of unmanned aerial vehicles (UAVs) was used to reconstruct rough terrains to plan safe trajectories for an unmanned ground vehicle (UGV). Reference [20] used UAV to identify terrain morphology and unstructured vineyards to develop an algorithm for generating a path for UGV.

The preliminary idea used in this study is a Voronoi diagram [21], which uses multiple points as the cores to divide a flat space into several areas for the path planning of the robot's moving. Although the Voronoi diagram is not a new concept used in path planning for robots' cooperation and exploration, it still plays a critical role in improving various algorithms for different purposes. For instance, Reference [22] used a Voronoi diagram and the particle swarm optimization algorithm to achieve multi-robot navigation and obstacles avoidance. In paper [23], the Voronoi diagram divides farmland into several areas for multiple robots to perform agriculture tasks. For multi-robot exploration problems, paper [24] used the distributed Voronoi partition to disperse the robot and then used the parametric algorithms to further control the robot's dispersion behavior to achieve the maximum area exploration. This paper considers the path planning problem in a space with multiple obstacles and multiple robots, and those obstacles and robots are not points but have body size. The generalized Voronoi diagram (GVD) [25] is used to deal with the problem. Reference [26] presents a nonuniform sampling technique to find an optimal path on the map, the map uses the generalized Voronoi graph to initialize, and the path is calculated by a heuristic path method. To improve the motion planning efficiency of the rapidly exploring random tree and its variants (RRTs), Reference [27] presents a GVD-based heuristic path planning algorithm to generate a heuristic path and guide the sampling process of RRTs.

In this paper, a navigation strategy with path priority (NSPP) is proposed to solve the problem of multiple robots moving in a large space with some static or/and dynamic obstacles. The main contributions of the paper are as follows. 1. The Voronoi diagram for a single-robot path planning is extended to multiple robots. 2. Each robot has its path-priority order to make the path of each robot more efficient. Suppose a control center manages an unmanned factory where each robot is set a unique path-priority order depending on the importance of its task. Each robot moves from its initial position to the target position without colliding with other robots or obstacles. Section 2 defines the problem to be solved and explains what the path-priority order of the robot is. Section 3 is the main session to build the navigation strategy with path priority (NSPP). Simulations and comparisons are carried out in Section 4. Finally, the conclusion is given in Section 5.

The main idea of the algorithm is that the Voronoi diagram for single-robot path planning is extended to multiple robots. Suppose a control center manages an unmanned factory where each robot is set a unique path-priority order. The path-priority order for

each robot depends on the importance of its task so that all robots with priority orders moving in the factory will be more efficient.

2. Problem Formulation

It is known that there have been many automated guided vehicles (AGV) or robots operating in unmanned factories. Guiding multiple robots to move quickly and safely to arrive at their destinations in a flat space without colliding with each other or hitting any obstacles is a required mission in an unmanned factory. This paper proposes an algorithm (called NSPP) to achieve the mission for differential drive robots. Suppose the space for robots moving is a two-dimension flat space, and all robots are moving with differential wheels. Suppose the localization of the robot is accurate and no sliding happens in the moving of robots. There may be some existing static or/and dynamic obstacles in the space. The proposed method should be feasible to any number of robots in the space. Here, each robot is set a unique number by the control center depending on the importance of the robot's task and the number represents its path-priority order. We also suppose each robot will do one job during a certain time. If the robot changes its job, then its priority order may be changed. The path-priority order is explained by an example as follows. The path-priority order of robot-1 is higher than that of robot-2, which means the path planning of robot-1 does not consider robot-2, even though robot-2 may appear on the path of robot-1. However, robot-2's path planning must consider robot-1 as a moving obstacle and plan a path to avoid hitting robot-1. That is to say, the robot with lower path priority will be made a detour to avoid hitting the robot with higher path priority. On the contrary, the robot with higher path priority will not change its original planned path to avoid the robot with lower path priority.

3. The Navigation Strategy with Path Priority

This section introduces the main algorithm of this paper called the navigation strategy with path priority (NSPP) to solve the problem. The NSPP contains five steps as follows and is summarized as the flowchart in Figure 1.

Step 1 Giving the flat space map for the robot moving includes obstacles and all wheeled robots initially.
Step 2 Setting the path-priority order for each robot and applying the GVD map division to each robot according to its path-priority order.
Step 3 Finding an efficient path for any robot from the starting point to the target point.
Step 4 Designing the velocity control strategy for all robots.
Step 5 To avoid any collision, the robot moving should follow the path-priority order defined in Step 2 to determine whether it moves continuously or stops and waits to avoid a collision.

The following figure summarizes the above steps and is explained in detail below.

3.1. Step 1 and Step 2: Giving an Environment Map and Generalized Voronoi Diagram (GVD) Map Division

Suppose a map with fixed obstacle positions is shown in Figure 2a. To avoid the robot hitting any obstacle, the gray boundary for each obstacle is dilated from the periphery of the obstacle, as shown in Figure 2b. The size of the dilatation is based on the radius of the considered robot's body. Let the Voronoi corners [28] for each obstacle be shown as the red dots in Figure 2c, where the Voronoi corners include the detected corners and some assigned extra corners. In this study, the extra corners on the boundary of the map are equidistantly set with twice the diameter of a robot and those on the edges of the obstacle are equidistantly set with the diameter of a robot. Those extra corners are added on the workspace boundary and the edges of obstacles. Due to adding the extra corners, navigation points will not be easily influenced by a Voronoi corner and there will be more paths to be chosen by the robot. In this paper, the locations of extra corners are set by a designer's experience. The segmented areas divided by blue lines using a Voronoi diagram

are shown in Figure 2d, where any intersection point of blue lines is called a navigation point of robots. Those blue lines passing through obstacles are deleted, as shown in Figure 2e, called the generalized Voronoi diagram (GVD) [29]. The moving path of a robot consists of several links, where each link connects two adjacent navigation points. It is known that if there are few extra corners, then there are few blue lines too (see Figure 2f). In this study, the number of each robot is the path-priority order number of the robot. For instance, robot-1 has higher path priority than robot-2 does.

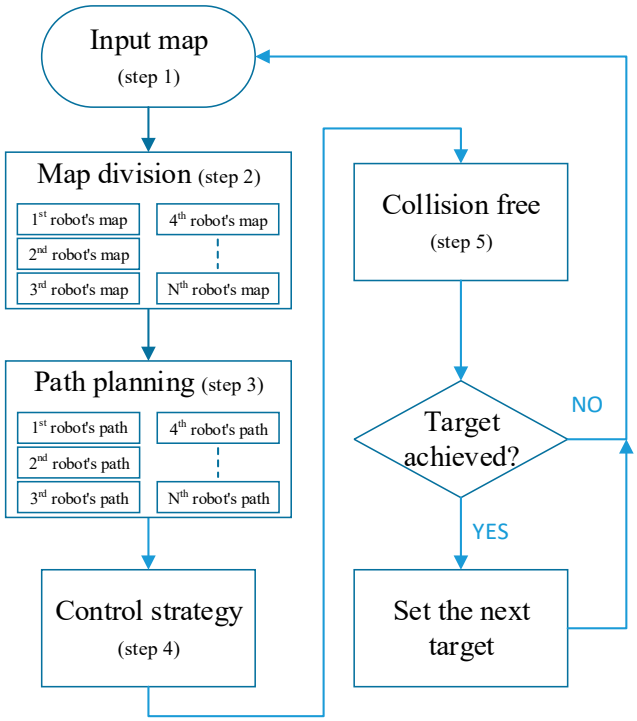


Figure 1. Flow chart of multi-robot path planning.

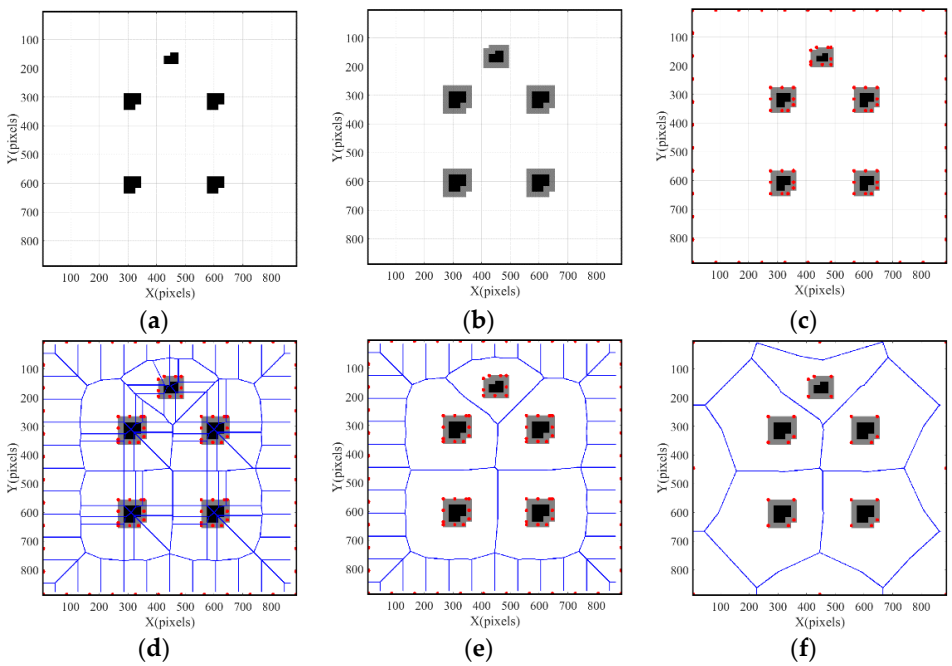


Figure 2. (**a**) The input obstacle map; (**b**) safety margins are added; (**c**) Voronoi corners are detected; (**d**) VD division, (**e**) GVD division, and (**f**) GVD division with few extra corners.

There are some navigation points in the GVD divided map, as shown in Figure 2e, and each robot can move from any navigation as an initial position to the target point along the blue lines. However, if there are additional fixed or dynamic obstacles in the space, we will immediately update the divisions in Figure 2d depending on the movements of the considered robot. For instance, as shown in Figure 3a,b,e, there are seven obstacles and two robots (represented by green circles) in the space, and these maps are in Figure 3 based on robot-2's viewpoint. Figure 3a,b shows the positions of robot-1 and robot-2 between adjacent time frames. Figure 3c,d are the local enlargements of Figure 3a,b, respectively, and show many feasible paths (the red arrows) to the navigation points around robot-2. It is noted that some feasible paths may be blocked by higher path-priority robots later. To avoid blocking situations happening, let us have an extra assigned Voronoi corner at the center of the lower path-priority robot (robot-2), such that an extra Voronoi cell is created around robot-2 as shown in the yellow areas of Figure 3e,f, respectively. It is seen that the feasible paths of robot-2 in the yellow Voronoi field shown in Figure 3f will not block the higher path-priority robot's moving.

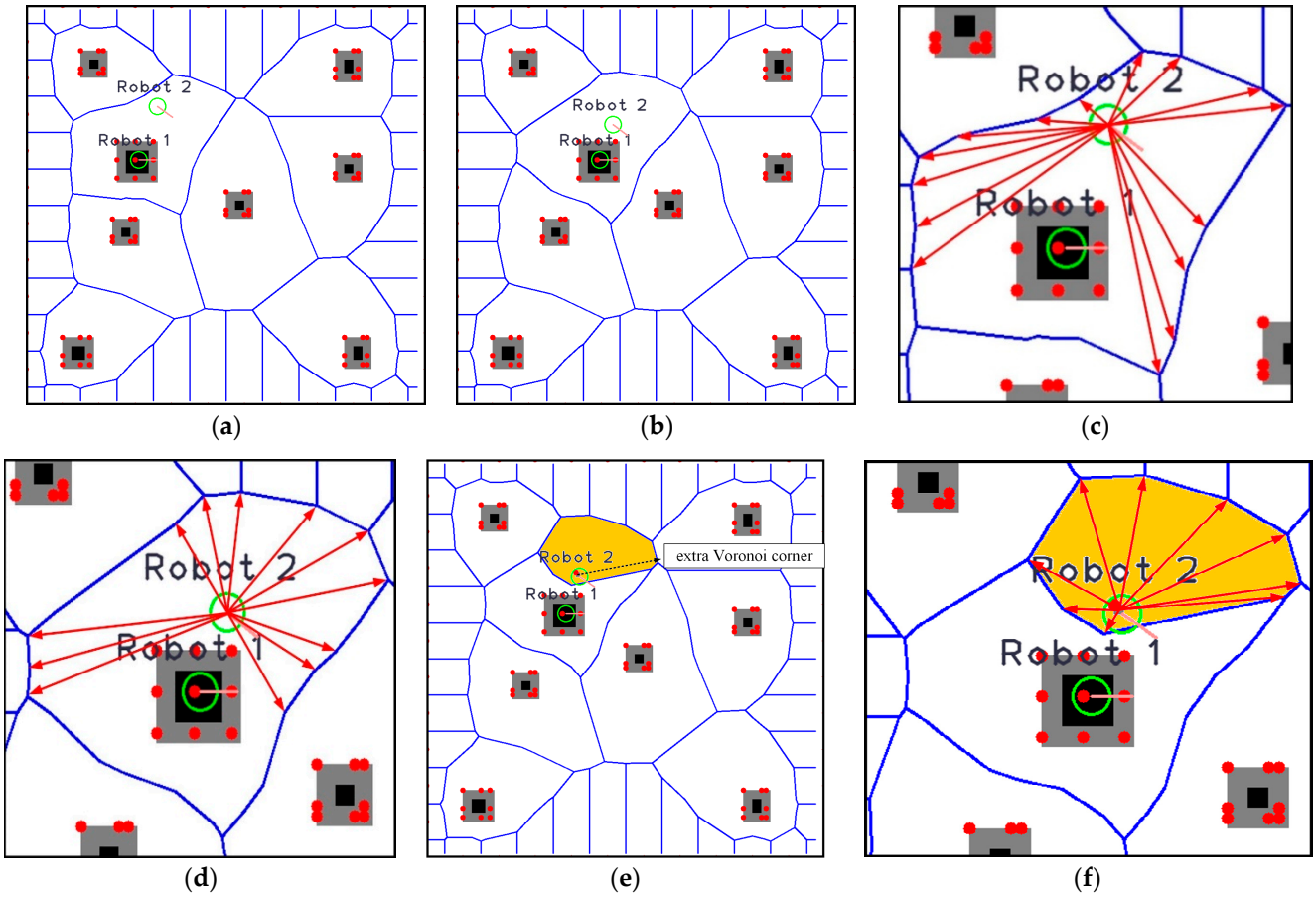
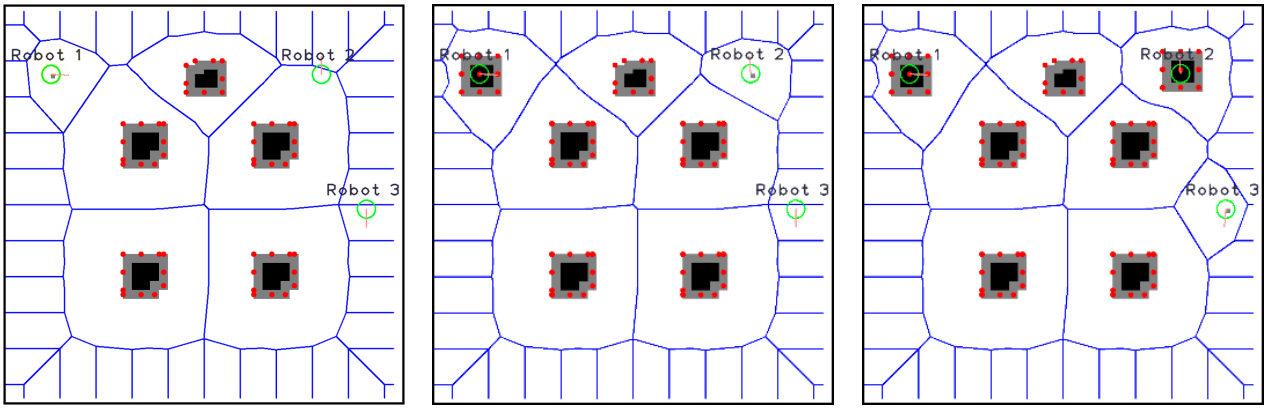


Figure 3. (**a**) The divided map from robot-2's view point without the assigned extra Voronoi corner at $time = t$. (**b**) The divided map from robot-2's view point without the assigned extra Voronoi corner at $time = t + 1$. (**c**) Local enlargement of (**a**) where the red arrows are the feasible paths; (**d**) local enlargement of (**b**) where the red arrows are the feasible paths; (**e**) robot-2's map with the assigned extra Voronoi corner at $time = t + 1$; and (**f**) local enlargement from (**e**) at $time = t + 1$, where the yellow region is the extra Voronoi cell.

From now on, we can set path priorities for all robots depending on the importance of each robot's task such that the robot transit will be more efficient. In other words, the higher path-priority robots can plan a path ahead of the lower path-priority robots, and the lower path-priority robot will regard the higher path-priority robot as a moving obstacle to be avoided. To increase the safety factor, the dilated range of the moving obstacle depends

on the radius of the higher path-priority robot. Then we will use GVD to detect all obstacles and create the Voronoi corners to update the divided map.

Figure 4a shows the map of robot-1 (from robot-1's point of view), Figure 4b shows that robot-1 is the dynamic obstacle in the map of robot-2, and Figure 4c shows that robot-1 and robot-2 are the dynamic obstacles to robot-3 in its map, and so on. It is noted that Figure 4 shows the different robots' points of view at the same point in time.



(**a**) (**b**) (**c**)

Figure 4. (**a**) The map of robot-1, (**b**) the map of robot-2, and (**c**) The map of robot-3.

3.2. Step 3: Path Planning

This subsection desires to find the efficient path for each robot in the space. In the beginning, the initial starting points and target points for all robots are set. Each robot moves from the starting point to the target point along links of the navigation points in the divided map, many feasible paths may be reached. Here, a Dijkstra algorithm [30] is used to calculate the lengths of all possible paths and find the shortest path between the starting point and target point for each robot using the links between navigation points on the map. As shown in Figure 5, if the robot has to pass between two obstacles, by using GVD, the feasible path between the two obstacles will pass through the midpoint of two obstacles with equal distance to both obstacles i.e., $O_1 = O_2$, where O_1 and O_2 are the distances between the path line and the two obstacles, respectively. In Figure 6a, the red line indicates a feasible path for robot-1 to reach the target point using the navigation points generated by the GVD. However, it is obvious that this path is not a short enough path to the target. To shorten the length of the red path, we may reduce the number of navigation points to find another shorter path using the algorithm in Algorithm 1. Note that both the starting and target points are navigation points that the robot can follow. Algorithm 1 defines two parameters, "scan" and "check", to simplify the path that is generated by Dijkstra algorithm. The "scan" begins from the starting point to the previous navigation point from target, but the "check" starts from the target point to the next numbers of "scan". For each iteration of "scan" and "check" parameters, we need to detect the obstacle on the connecting line between two navigation points corresponding to the "scan" and "check" numbers, respectively. If there is no obstacle on the connecting line, then the connecting line is a shortcut. Once a shortcut generates, the "scan" number will be changed to the number of "check" and the "check" will return to the beginning point (target point). Then repeat the above operations until the "scan" number reaches the target point. Then the final paths for robot-1, robot-2, and robot-3 are shown as the green paths in Figure 6a–c, respectively.

Algorithm 1. Path planning algorithm.

Input: All navigation points (NP_k, $k = 1, 2, \ldots$ n) generated by Dijkstra algorithm
Output: The better navigation points ($BNP_r, r = 1, 2, \ldots m$) depends on the input navigation points

```
Begin
  BNP_1 = NP_1, r = 2      # Save the target point
    For scan = 1 to n − 1      # "scan" from the starting point to the previous navigation point from target
      For check = n to scan + 1      # "check" from the target point to the next number of "scan"
        If (there is no obstacle on the connecting line between NP_scan and NP_check)
          BNP_r = NP_check      # save the shortcut point number
          scan = check, r = r + 1      # the next "scan" number is from the "check" numbers
          Break      # force into next round
        End If
      End For
  End For
End
```

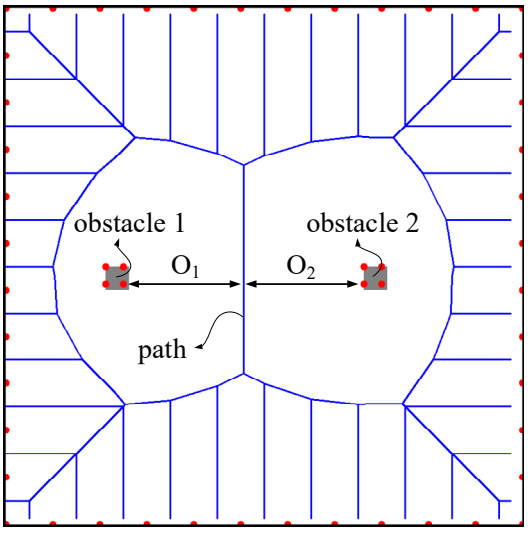
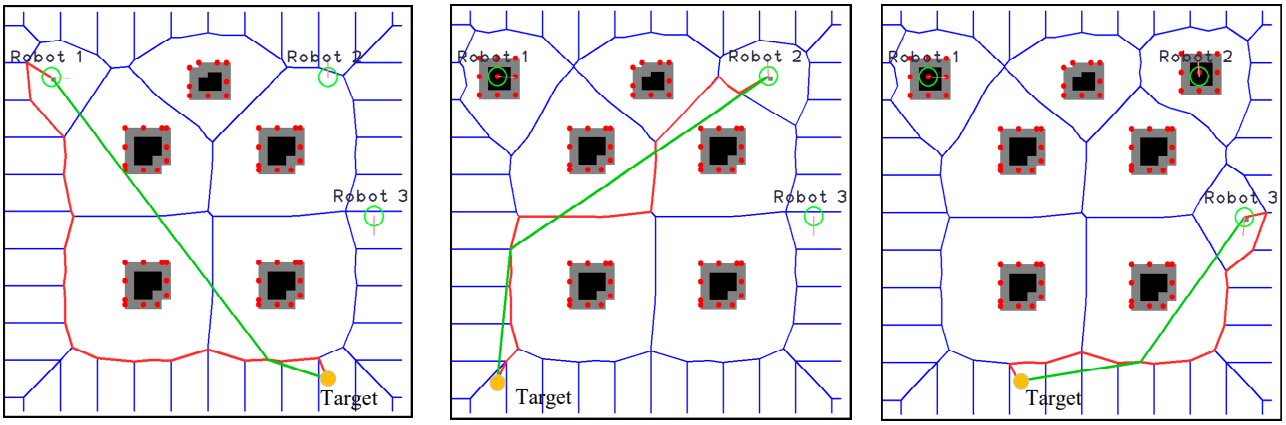


Figure 5. All feasible paths generated by the GVD.



(**a**) (**b**) (**c**)

Figure 6. (**a**) Path planning for robot-1 (the red line is the shortest path based on the GVD, and the green line is the result of path planning), (**b**) path planning for robot-2, and (**c**) path planning for robot-3.

3.3. Step 4: Velocity Control Strategy

To have a real simulation to verify the feasibility of the proposed NSPP, we have the following arrangement for each robot. Let us establish a robot polar coordinate [31,32] between the current position of the robot and the next navigation point (or the target) position as shown in Figure 7, where the origin of the coordinate is the next navigation point (or the target) position and the position of the robot is denoted by $\mathrm{R}(\rho, \theta)$, where ρ is the distance between the robot and the next navigational point (or the target), and θ is the orientation angle of the robot in the polar coordinate. Let ϕ be the angle between the moving direction of the robot and the horizontal axis, and $\beta = \pi - \theta$ and $\alpha = -\beta - \phi$ are defined. Here, u_1 is the speed of the robot moving, u_2 is the rotational speed of the robot, l is the half-length of the distance between two wheels, and v_l and v_r indicate the velocities of the left and right wheels, respectively, of the robot. Therefore, we have Equation (1) [31,32].

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \sin\alpha/\rho & -1 \\ -\sin\alpha/\rho & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{1}$$

where

$$u_1 = (v_r + v_l)/2, \tag{2}$$

and

$$u_2 = (v_r - v_l)/2l. \tag{3}$$

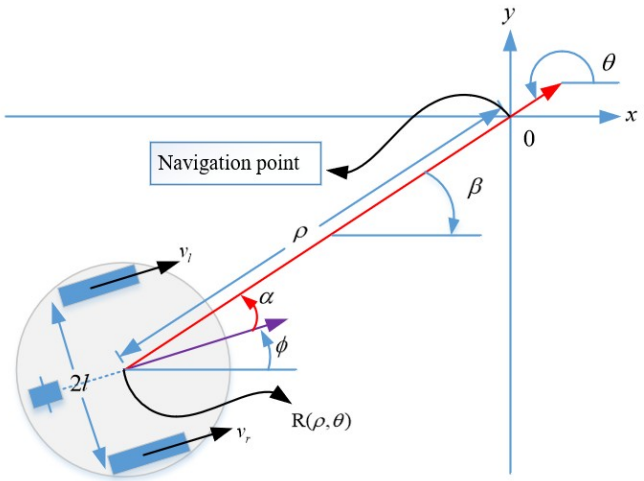


Figure 7. The polar coordinate of the wheeled robot (the purple arrow denotes the moving direction of the robot).

Furthermore, let $u_1 = k_\rho \times \rho$ denote that the forward moving speed depends on the distance from the robot to the next navigational point (target), and $u_2 = k_\alpha \times \alpha$ means the rotational speed is determined by the angles of α. From Equations (2) and (3), we can find the velocities of the left wheel as $v_l = k_\rho \times \rho - l \times k_\alpha \times \alpha$ and of the right wheel as $v_r = k_\rho \times \rho + l \times k_\alpha \times \alpha$, respectively. Finally, let the left and right wheel speeds be normalized as Equations (4) and (5) to avoid speed reduction when the robot moves closer to the navigational point.

$$v_{r\_nor} = k \times \frac{v_r}{|v_r| + |v_l|}, \tag{4}$$

$$v_{l\_nor} = k \times \frac{v_l}{|v_r| + |v_l|}, \tag{5}$$

where k is a constant.

Since v_l and v_r depend on the value of ρ, the speed reduction happens when the robot is close to the navigation point (such as point 1). However, the robot still needs to move to the next navigation point (such as point 2), and then it will start to speed up its speed due to new ρ from point 1 to point 2. Therefore, the speed of the robot will be changed frequently between any two navigation points. This situation is not efficient for the robot moving and should be avoided. Since the speed of the robot is $v = v_r + v_l$, then by the normalization in Equations (4) and (5), the final v will be $v_{r\_nor} + v_{l\_nor} = k$, which is a constant speed.

There are many parameters to be set for the above equations. We set $k_\rho = 0$ and $k_\alpha = 1$, which can be called the mode 1 (rotation mode) with $|\alpha| > 0.03$. Let $k_\rho = 1$ and $k_\alpha = 0.01$, which can be called the mode 2 (forward mode) with $|\alpha| \leq 0.03$. In mode 1, we are only concerned about the direction in which the robot is moving. Mode 2 forces the robot to move toward the target regardless of its orientation on reaching the target point.

3.4. Step 5: The Collision Avoidance Strategy

First, we define a sector in front of a certain robot (for instance, robot-*j*) as shown in Figure 8, where the sector denotes the dangerous area. The robot-*j* has the sector to robot-*n*, but robot-*n* does not have such sector to robot-*j*, where $j < n$. If robot-*n* has obstacles on that area, they may be hit by the moving robot-*j*. The reasons for the collision are as follows. (1) Robot-*j* will not change its planned path because of robot-*n*, (2) the sector of robot-*j* is too small, and (3) robot-*n* does not have such a sector to detect robot-*j*. It is noted that if each robot has its sector to detect other robots, then all robots may be stuck moving. In Figure 8, R_j is the radius of the sector, which is h times robot-*j*'s radius, i.e., $R_j = h \times L_j$, $4 \leq h$, where L_j is the radius of robot-*j*. Θ_j is the center angle of the sector of robot-*j*, and it is larger than 30 degrees. D_{jn} is the distance between robot-*j* and robot-*n*. G_{jn} is the angle for robot-*j* rotating to face robot-*n*. Therefore, when robot-*n* is inside the dangerous area (sector) of robot-*j* as in Expression (6),

$$R_j > D_{jn} \text{ and } \Theta_j > G_{jn}, \tag{6}$$

where $v_r^j = 0$ and $v_l^j = 0$ for robot-*j*. Then $u_1 = 0$, which denotes robot-*j*, stops to wait for robot-*n* to avoid the collision. The reason to choose $\Theta_j = 30$ and $R_j = 4L_j$ is based on our simulation experience to guarantee the sector is big enough to avoid robot-*j* hitting robot-*n*.

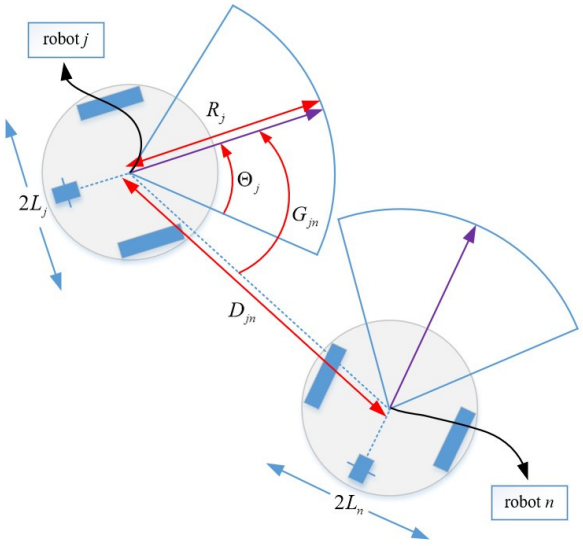


Figure 8. The sectors in front of the robots (the purple arrow denotes the moving direction of the robot).

Since the robot with lower path priority will regard the robot with higher path priority as an obstacle to be avoided, the path of the robot with the lower path priority will frequently change to avoid the moving obstacles (i.e., the robots with higher path priority). However, the planned path of the robot with higher path priority does not change often, but it may stop to wait for lower path-priority robots to pass through its path. The above two points will be verified in the next section.

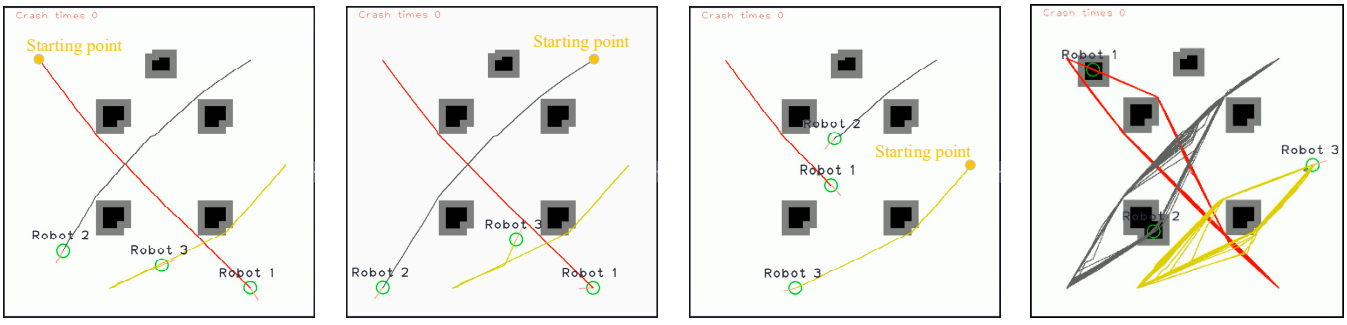
To emphasize the importance of the collision avoidance strategy, we provide a video on the website https://youtu.be/E40GOXhh-QQ (accessed on: 24 September 2021) to illustrate the robots' moving status with or without the avoidance strategy, respectively.

4. Simulation and Comparison

To evaluate the effectiveness of the proposed NSPP, this section presents the simulations of the robot navigation by using NSPP and makes a comparison between the proposed NSPP and some previous algorithms.

4.1. Simulation Results

The simulation was performed in an 880 × 880 (pixels) space, as shown in Figure 9a, and the radius of a robot is 20 pixels. In the first simulation, there are three robots, and each robot is moving from its starting point to its target point, as shown in Figure 6. The path priority of these robots is that robot-1 is the first, robot-2 is the second, and robot-3 is the third. Let $R_j = 4.5L_j$, $\Theta_j = 45°$, and $L_j = L_n$. The requirement is that once a robot reaches its target point, it must return to its original starting point immediately and start the next round trip. A collision counter is set in the upper left corner in the simulation, as shown in Figure 9a. The count increased by one means that the collision between two robots happens once. Therefore, in this experiment, the counter remaining at zero means all collisions are avoided. One more thing to be noted here, in mode 2 of simulation, we set $k_\alpha = 0.01$ because, considering the hardware limitation or uneven surface of the floor in the real experiment, it is impossible to have $\alpha = 0$ exactly at the sampling time; therefore, we used $\alpha = 0.03$ in the simulation and the sampling time of the simulation in this study was set at 38 ms.



(**a**) (**b**) (**c**) (**d**)

Figure 9. (**a**) The first trajectory (red line) of robot-1 arrives at the target point; (**b**) the first trajectory (gray line) of robot-2 arrives at the target point; (**c**) the first trajectory (yellow line) of robot-3 arrives at the target point; (**d**) the trajectories of all robots running between the starting point and the target point more than 50 times, respectively.

The trajectories of the first trip to the target points of robot-1, robot-2, and robot-3 are shown in Figure 9a–c, respectively. Figure 9d shows the trajectories of each robot moving between the starting point and the target more than 50 times (25 round trips). In Figure 9a, the path of robot-1 crosses the paths of robot-2 and robot-3, whereas the path of robot-2 does not cross the path of robot-3. In Figure 10a, the original path (green line) of robot-2 is blocked by robot-1 at a certain time, then Figure 10b–d shows that robot-1 is regarded as a moving obstacle by robot-2 and the real trajectory of robot-2 is altered to avoid hitting robot-1 on its way. A detailed description is presented in the caption of Figure 10. The same situation occurs for robot-3. A detailed description is presented in the

caption of Figure 11a–d, but it occurs much later than for robot-2. It is noted that robot-2 and robot-3 are of lower path priority than robot-1, so that their paths have turning points near the cross points with robot-1's path to avoid robot-1, respectively. It is concluded that robot-1 has the highest path priority, and its path plan focuses only on avoiding the fixed obstacles and disregards the collisions with robot-2 or robot-3. Thus, the trajectory of robot-1 is almost unchanged on its round trips (see Figure 9a). No collision occurs, so the collision number counter shows "0". A video of this simulation result can be seen at https://youtu.be/rNPp7bCDbUg (Accessed on: 16 September 2021).

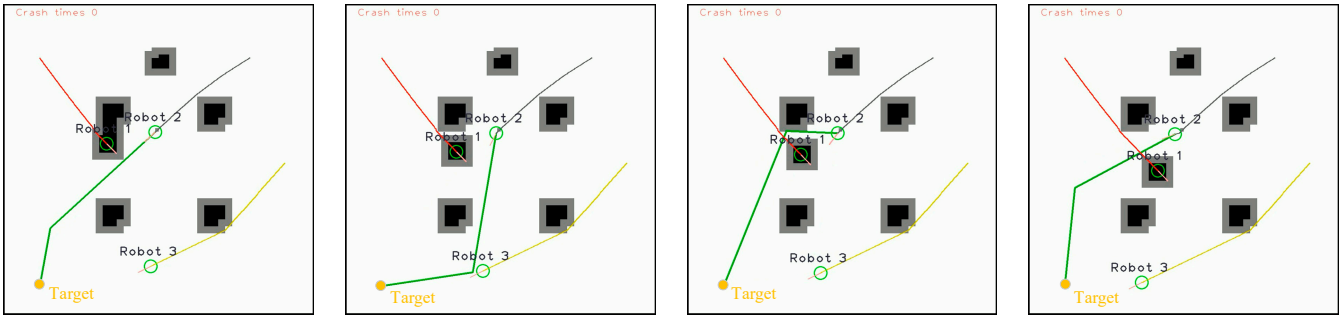


Figure 10. The path planning (green line) of robot-2 in the experiment. (**a**) Robot-1 has not yet become an obstacle to robot-2. (**b**) Robot-2 plans its path to avoid being blocked by robot-1. (**c**) Robot-1 continues to move toward the target point and then robot-2 changes its planned path to a shorter one, which passes behind robot-1. (**d**) Robot-1 gradually moves away from the area that affects robot-2.

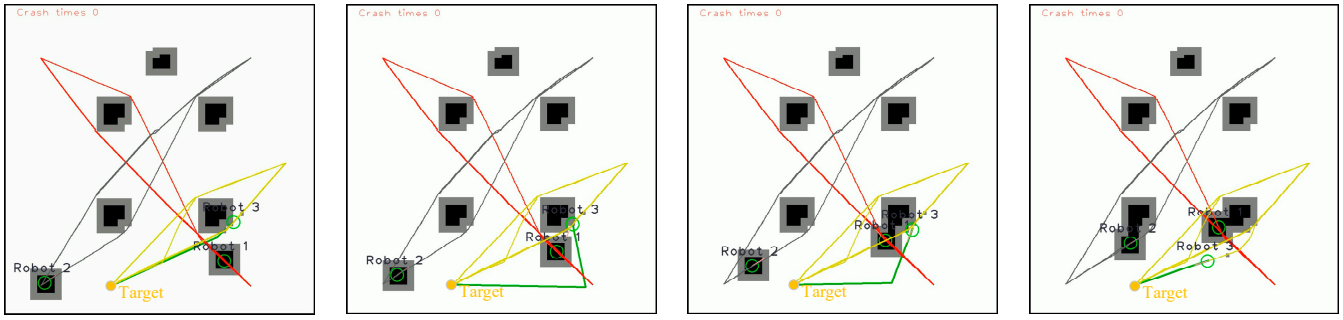


Figure 11. The path planning (green line) of robot-3 in the experiment. (**a**) Robot-1 has not yet become an obstacle to robot-3. (**b**) Robot-3 plans its path to avoid being blocked by robot-1 on its return trip. (**c**) Robot-1 continues to move back to its starting point, and then robot-3 changes its planned path to a shorter one that is behind robot-1. (**d**) Robot-1 gradually moves away from the area that affects robot-3. Robot-3's path becomes similar to the planned path in (**a**).

Additional simulations were conducted and are shown in Figure 12, where the starting points of all robots are evenly distributed at the top of the space and their target points are at the bottom corresponding to the respective starting points. In addition, each number above the green circle (robot) in Figure 12 represents the robot's number. In Figure 12a, there are three fixed obstacles in the middle of the space, and the robots can navigate smoothly from the starting point to the target point without any collision. In Figure 12b, there are more obstacles in the space. Nevertheless, each robot can still reach its target point successfully, although the paths of robot-2 and robot-5 have some curved parts. In Figure 12c,d, there are a concave polygon obstacle and two specifically shaped obstacles, respectively, in the space. Using the proposed NSPP, each robot reaches its target, and its trajectory is shown too.

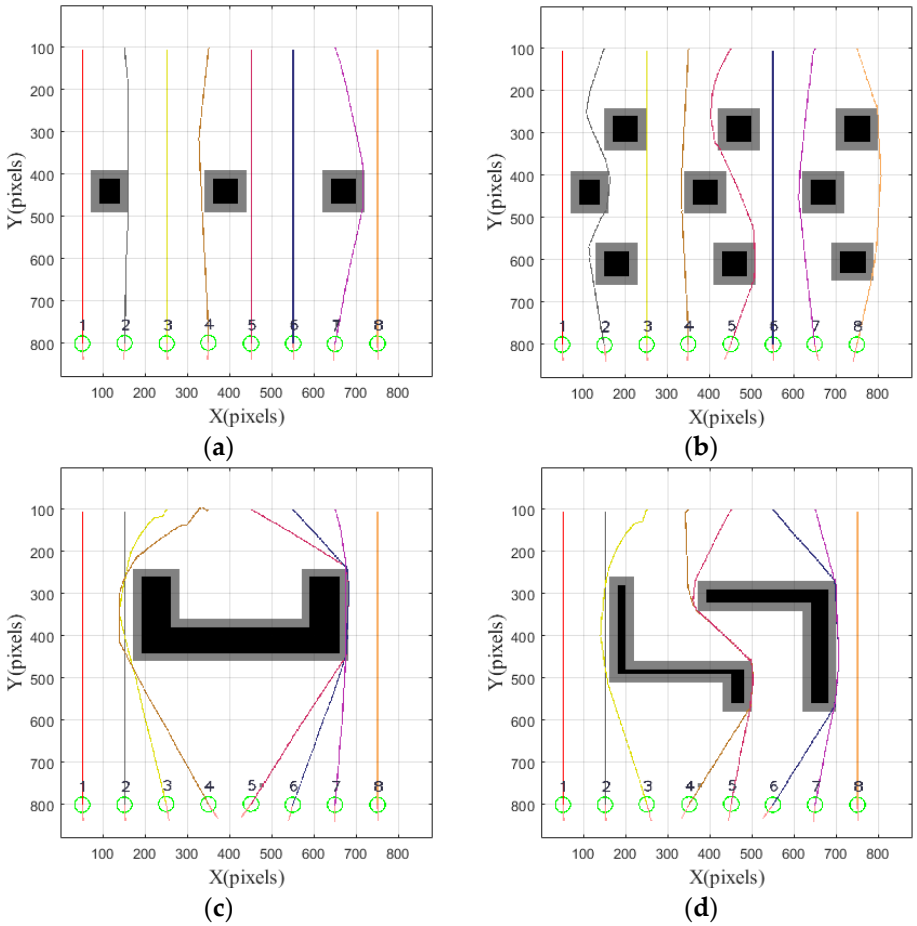


Figure 12. (**a**) A simple test with three obstacles, (**b**) nine obstacles interlaced test, (**c**) concave polygon obstacle test, and (**d**) passage-type obstacle test.

4.2. Comparison Results

The proposed NSPP was implemented and compared with the other algorithms mentioned in [33]. The comparison was implemented in a circular space, in which four or eight robots are distributed equally on the circle, as shown in Figure 13 [33]. Each robot has its own target at the opposite side of the circle, and its initial orientation is toward the target point. For example, the robot at P2 (or P6) will go forward to and stop at position P6 (or P2). Figure 13 is the configuration diagram of the eight robots in which all robots can be randomly assigned from P1 to P8. In the simulation of four robots, only P1, P3, P5, and P7 are needed in the configuration. Other hypotheses in the simulation are as follows. (a) The circle radius is 260 pixels, (b) the radius of a robot is 20 pixels, and (c) the speeds of all robots are the same.

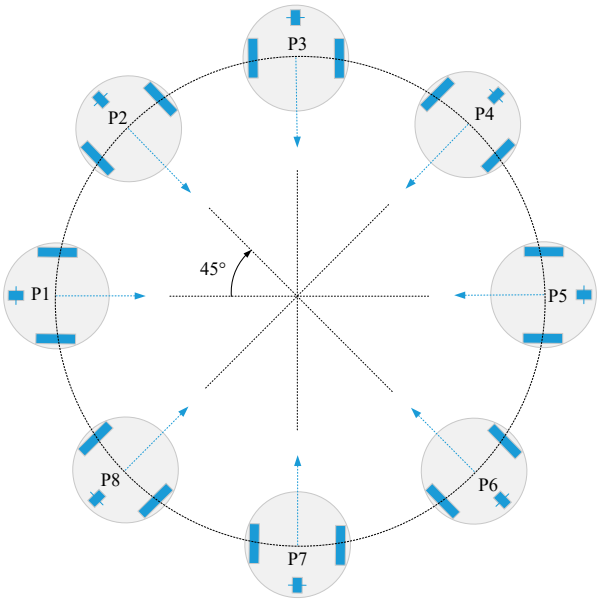


Figure 13. The configuration diagram of eight robots for the benchmark comparison.

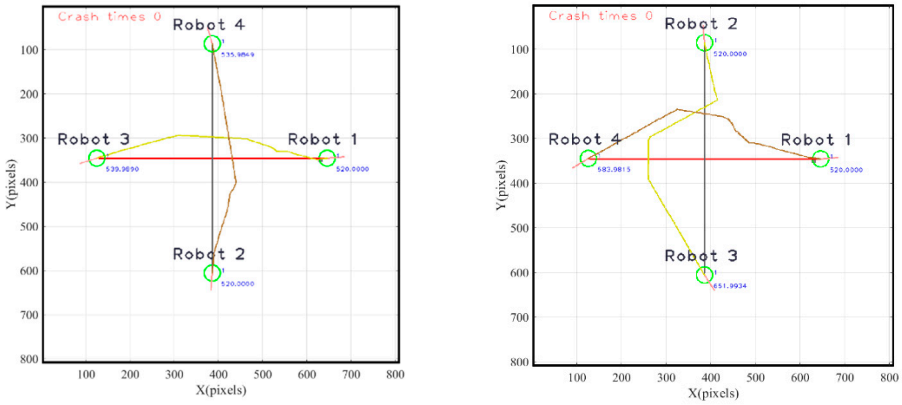
First, we consider the simulation with four robots. Because of the different priorities and different positions of the four robots, there are six (i.e., $3 \times 2 \times 1$) configurations and six different simulation results. There are 5040 configurations and 5040 different simulation results for eight robots. We chose six configurations randomly for both simulation cases.

For the convenience of comparison, we need to define some notations as follows. In Figure 13, the symbol NSPP(1432) means there are four robots and robot-1 is at position P1, robot-4 is at P3, robot-3 at P5, and robot-2 at P7, initially. In the case of eight robots, the symbol NSPP(14785623) means robot numbers 14,785,623 are initially at P1–P8, respectively. We show the simulation results of a four-robot configuration, and Figure 14a,b shows the shortest and longest average length of four robot's paths, respectively, in which the number under each robot is the robot's trajectory length. In Figure 15a,b, the simulation results are shown for the eight-robot configuration in which NSPP(14632875) has the shortest trajectory and NSPP(18526743) has the longest one. It is well known that the shortest distance algorithm (SDA) [33] and the reciprocal orientation algorithm (ROA) [33] are two benchmark methods in this issue. Let us define

$$\text{ATL}(\text{average trajectory length}) = \frac{\text{sum of all robot trajectory lengths}}{\text{number of robots}} \tag{7}$$

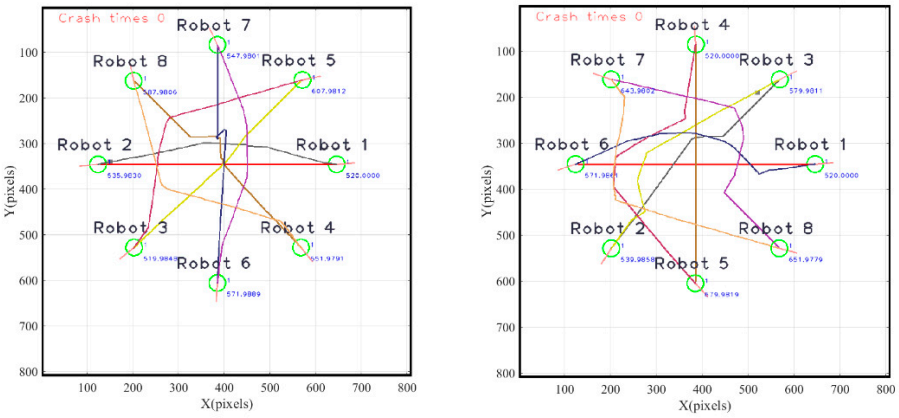
and

$$\text{MTL}(\text{maximum trajectory length}) = \text{maximum trajectory length in all trajectories of robots} \tag{8}$$



(**a**) (**b**)

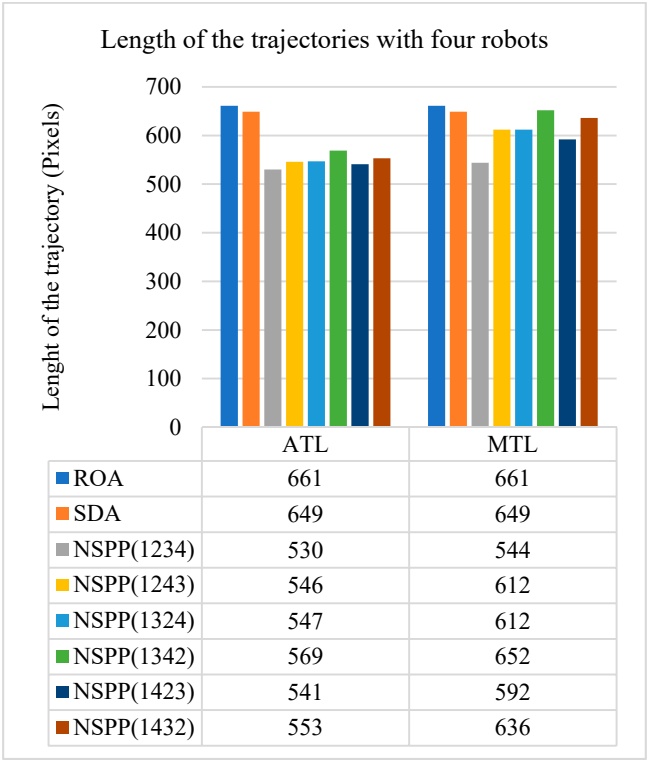
Figure 14. The results of four robots. (**a**) The best result of NSPP(1234) and (**b**) the worst result of NSPP(1342).



(**a**) (**b**)

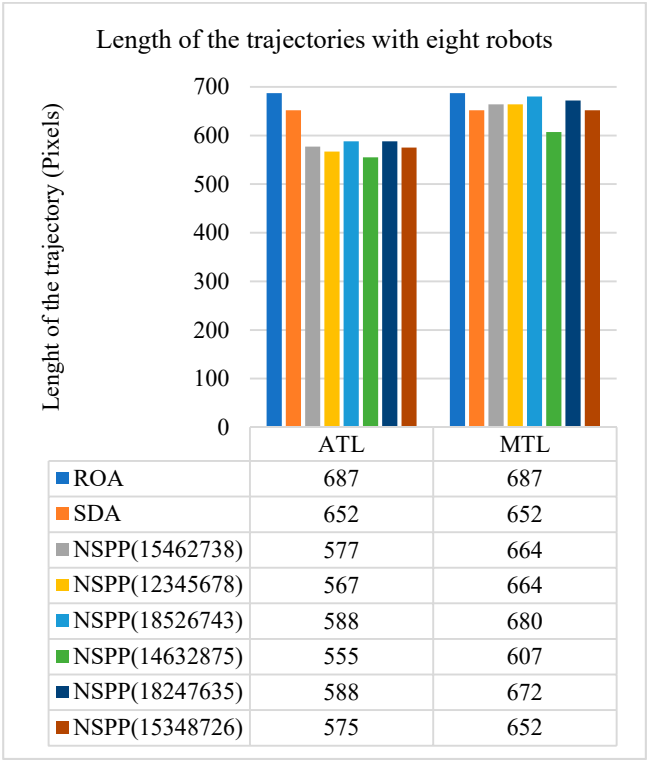
Figure 15. The results of eight robots. (**a**) The best result of NSPP(14632875) and (**b**) the worst result of NSPP(18526743).

The comparison is shown in Figure 16a, where all data of the NSPP in the ATL column are better than the results of both the SDA and ROA; the data of the NSPP are almost better than those of the SDA and ROA in the MTL column except that the NSPP(1342) is worse than SDA. The case of the NSPP(1342) has the maximum trajectory length in robot-3 since robot-1 is waiting for robot-2 and robot-2 is waiting for robot-3 at a certain time, simultaneously. Then robot-3 must make a big turn around robot-1 and robot-2. When the number of robots is eight, it is hard to tell which one is better compared to the MTL in Figure 16b. However, the NSPP performs better than the SDA and ROA in the ATL. It is noted that the data of the ATL and MTL are the same in the ROA and SDA, which means all robot trajectory lengths are the same in their methods, since the methods of ROA and SDA do not give the path-priority order for all robots.



	ATL	MTL
ROA	661	661
SDA	649	649
NSPP(1234)	530	544
NSPP(1243)	546	612
NSPP(1324)	547	612
NSPP(1342)	569	652
NSPP(1423)	541	592
NSPP(1432)	553	636

(**a**)



	ATL	MTL
ROA	687	687
SDA	652	652
NSPP(15462738)	577	664
NSPP(12345678)	567	664
NSPP(18526743)	588	680
NSPP(14632875)	555	607
NSPP(18247635)	588	672
NSPP(15348726)	575	652

(**b**)

Figure 16. (**a**) Trajectory length comparison of ROA, SDA, and NSPP for four robots; (**b**) trajectory length comparison of ROA, SDA, and NSPP for eight robots.

From the above comparison, the ROA and SDA achieved collision-free navigation for multiple robots, but the NSPP provides a shorter average trajectory length for the four- and eight-robot scenarios in the ATL column. In the MTL column of the four-robot case, most results by the proposed NSPP are better than the ROA and SDA. However, in the MTL column of eight robots, it is difficult to tell which is the winner.

5. Conclusions

To improve factory operation efficiency, the engineers can determine the path priority for each robot depending on the importance of the robot's task and then request the shorter average path for total robots. In this paper, an efficient navigation algorithm with path priority (NSPP) for multi-robot moving is proposed. Multiple robots can reach their respective target points with highly efficient paths and without collisions in a flat space with obstacles. The methods of ROA and SDA also achieved collision-free navigation for multiple robots, but in this study, we used the path-priority concept, GVD and Dijkstra, and obstacle avoidance strategies to develop an efficient NSPP to achieve the same objective. The performance of the NSPP was compared with those of the SDA and ROA, and it was found that the NSPP has a better average trajectory length than the others in four- or eight-robot configurations, as shown in Figure 16.

We have to mention that the proposed NSPP can be applied to any number of robots; however, it is not guaranteed that the performance of the proposed NSPP is the best compared to other methods when the number of robots increases or the working environment is changed. We have to admit that the NSPP still needs some improvement such as having a path priority that can be replaced by moving priority, in other words, maybe the robot with higher moving priority can move without avoiding any other lower priority robot; on the contrary, the lower moving priority robot should avoid hitting higher moving priority robot on the way. In other words, the robot with higher moving priority can follow its planned path to move, and the robot with lower moving priority has to change its planned

path frequently or maybe uses speed adjustment to slow down so that the robot with higher moving priority is not hit by the robot with lower moving priority. However, whether moving priority or path priority has a more efficient performance will be studied in the near future.

At last, we have to mention that there is also a method called "visibility graph" [34] for dealing with the path planning problem for a robot. The purpose of the visibility graph is to find the shortest path for a robot, which is similar to the GVD in the proposed method. However, comparing the time complexity between the GVD and the visibility graph, there are O (n log n) for the GVD [35] and O (n^2 log n) for the visibility graph [34], where *n* is the total number of input points. Hence, the time cost of using the GVD to find the feasible path is faster than using the visibility graph.

Author Contributions: Conceptualization, S.-K.H. and C.-H.S.; Methodology, S.-K.H. and C.-H.S.; Resources, W.-J.W.; Software, S.-K.H.; Supervision, W.-J.W.; Validation, S.-K.H. and W.-J.W.; Writing—original draft, S.-K.H.; Writing—review and editing, W.-J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Ministry of Science and Technology of Taiwan with the grant number 110-2634-F-008-005.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, S.; Wan, J.; Li, D.; Zhang, C. Implementing smart factory of industrie 4.0: An outlook. *Int. J. Distrib. Sens. Netw.* **2016**, *12*, 3159805. [CrossRef]
2. Matarić, M.J.; Sukhatme, G.S.; Østergaard, E.H. Multi-robot task allocation in uncertain environments. *Auton. Robot.* **2003**, *14*, 255–263. [CrossRef]
3. Kavraki, L.E.; Svestka, P.; Latombe, J.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
4. Belkhouche, F. Reactive path planning in a dynamic environment. *IEEE Trans. Robot.* **2009**, *25*, 902–911. [CrossRef]
5. Guo, Y.; Parker, L.E. A distributed and optimal motion planning approach for multiple mobile robots. In Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington, DC, USA, 11–15 May 2002; Volume 3, pp. 2612–2619.
6. Bellingham, J.S.; Tillerson, M.; Alighanbari, M.; How, J.P. Cooperative path planning for multiple UAVs in dynamic and uncertain environments. In Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas, NV, USA, 10–13 December 2002; Volume 3, pp. 2816–2822.
7. Keymeulen, D.; Decuyper, J. The fluid dynamics applied to mobile robot motion: The stream field method. In Proceedings of the 1994 IEEE International Conference on Robotics and Automation, San Diego, CA, USA, 8–13 May 1994; Volume 1, pp. 378–385.
8. Das, P.; Pradhan, S.; Patro, S.; Balabantaray, B. Artificial immune system based path planning of mobile robot. In *Soft Computing Techniques in Vision Science*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 195–207.
9. Ge, S.S.; Cui, Y.J. Dynamic motion planning for mobile robots using potential field method. *Auton. Robot.* **2002**, *13*, 207–222. [CrossRef]
10. Cui, S.G.; Wang, H.; Yang, L. A simulation study of A-star algorithm for robot path planning. In Proceedings of the 16th International Conference on Mechatronics Technology, Tainjin, China, 16–19 October 2012; pp. 506–509.
11. Bhattacharya, P.; Gavrilova, M.L. Roadmap-based path planning—Using the Voronoi diagram for a clearance-based shortest path. *IEEE Robot. Autom. Mag.* **2008**, *15*, 58–66. [CrossRef]
12. Duchoň, F.; Babinec, A.; Kajan, M.; Beňo, P.; Florek, M.; Fico, T.; Jurišica, L. Path planning with modified a star algorithm for a mobile robot. *Procedia Eng.* **2014**, *96*, 59–69. [CrossRef]
13. Elhoseny, M.; Tharwat, A.; Hassanien, A.E. Bezier curve based path planning in a dynamic field using modified genetic algorithm. *J. Comput. Sci.* **2018**, *25*, 339–350. [CrossRef]
14. Das, P.K.; Behera, H.S.; Jena, P.K.; Panigrahi, B.K. Multi-robot path planning in a dynamic environment using improved gravitational search algorithm. *J. Electr. Syst. Inf. Technol.* **2016**, *3*, 295–313. [CrossRef]
15. Das, P.K.; Behera, H.S.; Panigrahi, B.K. A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. *Swarm Evol. Comput.* **2016**, *28*, 14–28. [CrossRef]

16. Huang, X.C.C.; Zhang, Y.; Qin, S.; Zeng, Y.; Li, X. Switched linear multi-robot navigation using hierarchical model predictive control. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; pp. 4331–4337.
17. Huang, C.; Chen, X.; Zhang, Y.; Qin, S.; Zeng, Y.; Li, X. Hierarchical model predictive control for multi-robot navigation. In Proceedings of the International Joint Conference on Artificial Intelligence, New York, NY, USA, 9 July 2016.
18. Alonso-Mora, J.; Baker, S.; Rus, D. Multi-robot navigation in formation via sequential convex programming. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, Hamburg, Germany, 28 September–2 October 2015; pp. 4634–4641.
19. Guastella, D.C.; Cantelli, L.; Longo, D.; Melita, C.D.; Muscato, G. Coverage path planning for a flock of aerial vehicles to support autonomous rovers through traversability analysis. *ACTA IMEKO* **2019**, *8*, 9–12. [CrossRef]
20. Zoto, J.; Musci, M.A.; Khaliq, A.; Chiaberge, M.; Aicardi, I. Automatic path planning for unmanned ground vehicle using uav imagery. In *International Conference on Robotics in Alpe-Adria-Danube Region*; Springer: Kaiserslautern, Germany, 2019; pp. 223–230.
21. Imai, H.; Iri, M.; Murota, K. Voronoi diagram in the laguerre geometry and its applications. *SIAM J. Comput.* **1985**, *14*, 93–105. [CrossRef]
22. Gabbassova, Z.; Sedighizadeh, D.; Fini, A.S.; Seddighizadeh, M. Multiple robot motion planning considering shortest and safest trajectory. *Electromech. Energy Convers. Syst.* **2019**, *1*, 1–6.
23. Kim, J.; Son, H.I. A Voronoi diagram-based workspace partition for weak cooperation of multi-robot system in orchard. *IEEE Access* **2020**, *8*, 20676–20686. [CrossRef]
24. Slimane, N.B.; Tagina, M. Proposition of a distributed Voronoi partitioning approach enhanced with a dispersion phase for a multirobot system. *Int. J. Soc. Robot.* **2020**, *13*, 887–898.
25. Lee, D.T.; Drysdale, R.L., III. Generalized Voronoi diagrams in the plain. *SIAM J. Comput.* **1981**, *10*, 73–87. [CrossRef]
26. Wang, J.; Meng, M.Q.H. Optimal path planning using generalized Voronoi graph and multiple potential functions. *IEEE Trans. Ind. Electron.* **2020**, *67*, 10621–10630. [CrossRef]
27. Chi, W.; Ding, Z.; Wang, J.; Chen, G.; Sun, L. A generalized Voronoi diagram based efficient heuristic path planning method for RRTs in mobile robots. *IEEE Trans. Ind. Electron.* **2021**. [CrossRef]
28. Ho, Y.-J.; Liu, J.-S. Smoothing Voronoi-based obstacle-avoiding path by length-minimizing composite bezier curve. In Proceedings of the International Conference on Service and Interactive Robotics, Taipei, Taiwan, August 2009.
29. Takahashi, O.; Schilling, R.J. Motion planning in a plane using generalized Voronoi diagrams. *IEEE Trans. Robot. Autom.* **1989**, *5*, 143–150. [CrossRef]
30. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
31. Sun, C.-H.; Chen, Y.-J.; Wang, Y.-T.; Huang, S.-K. Sequentially switched fuzzy-model-based control for wheeled mobile robot with visual odometry. *Appl. Math. Model.* **2017**, *47*, 765–776. [CrossRef]
32. Siegwart, R.; Nourbakhsh, I.R. *Introduction to Autonomous Mobile Robots*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2011.
33. Ali, A.A.; Rashid, A.T.; Frasca, M.; Fortuna, L. An algorithm for multi-robot collision-free navigation based on shortest distance. *Robot. Auton. Syst.* **2016**, *75*, 119–128. [CrossRef]
34. Coleman, D. *Lee's O (n2 log n) Visibility Graph Algorithm Implementation and Analysis*; Department of Computer Science, University of Colorado at Boulder: Boulder, CO, USA, 2012.
35. Fortune, S. A sweepline algorithm for Voronoi diagrams. In Proceedings of the Second Annual Symposium on Computational Geometry, Yorktown Heights, New York, NY, USA, 2–4 June 1986.