A New Multirobot Path Planning With Priority Order Based on the Generalized Voronoi Diagram

SHENG-KAI HUANG1, WEN-JUNE WANG1, (Fellow, IEEE), AND CHUNG-HSUN SUN2

1Department of Electrical Engineering, National Central University, Taoyuan 320, Taiwan
2Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 82444, Taiwan

Corresponding author: Wen-June Wang (wjwang@cc.ncu.edu.tw)

ABSTRACT This paper proposes a new path planning method called the priority order navigation algorithm (PONA) for multi-robot navigation in a large flat space. The PONA can guarantee collision-free and efficient travel in the space with fixed or/and dynamic obstacles. The priority order of robots is assigned by the user based on the importance degree of the robots’ tasks and the objective is to make the higher priority robot reach its target faster than the lower priority robot. This study uses the generalized Voronoi diagram (GVD) to establish the initial map for PONA and links the navigation points in GVD to plan the path for each robot. Further, we modify the navigation point links to shorten feasible paths for the lower priority robot and its shortest two feasible paths can be switched to each other based on a certain condition to avoid hitting the higher priority robot. The proposed PONA is compared to several benchmark path planning methods, which are the shortest distance algorithm (SDA) and reciprocal orientation algorithm (ROA), in the simulation section and it is found that the PONA can reduce the average length of the trajectory by more than 10% compared with ROA and SDA.

INDEX TERMS Voronoi diagram, Yen’s algorithm, multi-robot path planning, collision-free, path-priority order.

I. INTRODUCTION

In Phase 4 of Industry 4.0, one of the critical points for vertical integration of factories is conveying system. In paper [1], the conveying system used linear and right-angle arc modules to combine multiple interconnected cyclic tracks. The robot of AGV (Automated Guided Vehicles) traveled along these cyclic tracks and used them to prevent collisions. Many international vendors, such as Amazon and Ally logistic property, use similar technologies to achieve automatic transportation. Some conveying systems without cyclic tracks are used, but path planning, collision-free, and transportation time reduction will be the problems to be dealt with in multi-robot transportation.

The problem of single robot path planning in a space with obstacles has been studied in many papers [2]–[11]. However, a single robot is not enough to handle the complicated work in a factory today and multi-robot moving and working simultaneously in a factory are very common in recent years. Therefore, studying the path planning for multi-robot in a space with obstacles is necessary and very important. In the study of multi-robot path planning [12], a navigation method of a group of formation robots in a dynamic environment was proposed. An improved gravitational search algorithm (IGSA) based on memory information, social interaction, and particle swarm optimization (PSO) was proposed in [13]. A hierarchical model predictive control (HMPC) based method was studied in [14] to solve the problem of multi-robot navigation control without reference trajectory. The paper [15] presented a two-level optimal multi-agent pathfinding algorithm to find paths with collision-free for multiple agents. The authors claimed that the proposed two-level algorithm examines fewer states than the A* algorithm [16] in many cases and still maintains the optimality. The paper [17] provided a method called the shortest distance algorithm (SDA) for multi-robot navigation in a dynamic environment. The method can calculate the collision-free trajectory for each robot based on the robot’s current position and other robot positions. The paper [18] proposed a conceptual approach to improve the classical krill herd (KH) [19] to find a collision-free path with minimum energy utilization. An intelligent control algorithm based on centroidal Voronoi tessellation (CVT) was presented.
by [20] to do the path planning for a swarm of robots. The studies [12], [15], [20] provided collision-free path planning for multi-robot in a dynamic environment, but they did not concern the length of the trajectory for each robot. The paper [14] solved the multi-robot navigation control problem by a theoretical method on stability and feasibility, but there is no way to guarantee the feasibility in dynamic environments. The papers [13], [18] also provided collision-free path planning for multi-robot. However, when there are many robots, and the paths to the corresponding targets, respectively, of all robots are intersected with each other, the performance may be doubted. The paper [17] provided good performance in a circle scenario where the paths of many robots conflict with each other. However, their simulation did not provide the performance result under obstacle avoidance. One more thing to be mentioned is that the above studies did not consider the priority mechanism for multi-robot. In other words, all robots have equal priority in their work.

If we consider the priority mechanism for all robots, there have been many studies investigating the problem of multi-robot path planning. The priority mechanism makes the robots do their work with priority order such that the robots moving will be more efficient. The randomized search with hill-climbing was used by [21] to find the optimal priority order for all robots to solve the coordination of the motions of multi-robot. The paper [22] solved the dynamic conflicts between robots by using the heuristic adjustment of priority values for cooperative path planning in the multi-robot system. A prioritization method in [23] provides consistent, conflict-free path plans by formalizing a robot’s path prospects to reach its goal from its current location. The paper [24] provided a priority-based approach, which gives priority to each robot, to bypass other robots for collision avoidance. Path planning is achieved by the communication between multi-robot and collision checks. In the above papers [21]–[23], the priority order for each robot is not assigned in advance. Those approaches dynamically adjust the robot’s priority order to achieve conflict-free path plans. Moreover, they cannot deal with the problem that multi-robot with different tasks may need a variety of requirements for their traveling. The paper [24] dealt with multi-robot path planning problem with obstacles in which the robot’s priority order depends on the distance length between its initial point and target point. It is pity that the paper did not provide a comparison between their proposed method and other algorithms. In paper [25], the navigation strategy with path priority (NSPP) algorithm assigns the priority to each robot such that the path of high path priority robot is shorter than that of low path priority robot in the same space and collision-free is guaranteed too. However, a drawback is that the higher path priority robot sometimes may need to stop and wait for a lower path priority robot to pass through for avoiding a collision.

In general, the Voronoi diagram [26] uses points or blocks as the inputs to divide the space into several regions, and it is often used for the path planning of the multi-robot moving. In paper [27], the Voronoi diagram was used to divide the farmland into several areas such that humans spray pesticides in agriculture more efficiently. To plan the paths for multi-robot in an unknown environment, the paper [28] uses the Voronoi diagram to find the safe path for each robot and then uses PSO to find the shortest path to the next target point. In paper [29], the authors proposed a two-phased approach with the distributed Voronoi partitioning to deal with the problem of partitioning the space in an even way between several autonomous mobile robots. In paper [30], the generalized Voronoi diagram (GVD) was developed from the traditional Voronoi diagram, and it is always used to solve the segmented polygonal regions containing obstacles with body size. In paper [31], the environment map is initialized with GVD and a nonuniform sampling technique, based on the pipeline of rapidly exploring random tree (RRT), is presented to efficiently calculate collision-free paths. To improve the efficiency of dynamic path planning for fast exploration of random trees and their variants, the paper [32] proposed a GVD feature matrix-based efficient heuristic path planning for a robot in dynamic environments.

In this paper, we study the path planning problem for multi-robot moving in a two-dimensional flat space with obstacles. Each robot has its priority order depending on the importance degree of the task done by the robot. The objective of this study is to plan the paths such that the higher priority robot moves and reaches its target point with shorter paths and less time spent than the lower priority robot. Moreover, there will be collision-free when multiple robots simultaneously move in the same space. This paper is organized as follows: Section 2 presents the generation of a generalized Voronoi diagram for each robot. Section 3 describes the path planning for all robots. Section 4 proposes the collision avoidance strategy between two robots. The moving trajectory derivative of the robot and some simulations of PONA are presented in Section 5. The comparison between the proposed PONA and some benchmark algorithms is stated in Section 6. At last, a conclusion is given in the final section.

II. GENERALIZED VORONOI DIAGRAM GENERATION
This section introduces the generation of the generalized Voronoi diagram, which is the first stage of the path planning for multi-robot. This stage includes three parts to be described as follows.

A. DATA PRE-PROCESSING
Let the space where wheeled robots are moving be regarded as a map. The map includes several obstacles and all-wheeled robots initially. Suppose the map with known obstacle (including other robots) positions is shown in Fig. 1, where black areas denote those obstacles. These obstacles include static (such as fixed objects) or/and dynamic obstacles (such as other moving robots). The gray boundary of each obstacle shown in Fig. 2 is dilated from around the obstacle to be the buffer area to avoid being hit by moving robots. The dilation width depends on the size of the obstacle, and
the larger the size of the obstacle, the wider the width of the dilation. Then referring to [25], we can generate many red points called Voronoi corners around the gray boundary of each obstacle and the boundary of the map shown in Fig. 3.

![The map with obstacles.](image1.jpg)

FIGURE 1. The map with obstacles.

![Obstacles dilation.](image2.jpg)

FIGURE 2. Obstacles dilation.

![Red dots denote Voronoi corners.](image3.jpg)

FIGURE 3. Red dots denote Voronoi corners.

**B. GENERALIZED VORONOI DIAGRAM**

After assigning the Voronoi corners, a Voronoi diagram is generated as shown in Fig. 4, where any intersection point of red lines is called the navigation point (NP) for the robot moving. A robot will move along line segments connected between any two neighboring navigation points (NPs). However, many red lines that go through the obstacle need to be removed; the final diagram is called the GVD as shown in Fig. 5.

![The Voronoi diagram.](image4.jpg)

FIGURE 4. The Voronoi diagram.

![The generalized Voronoi diagram.](image5.jpg)

FIGURE 5. The generalized Voronoi diagram.

Next, we need to assign priority orders for all robots based on the importance degree of all robot tasks to make the robot transport more efficiently. The robot with higher priority (higher important task) has higher priority to plan its path than the robot with lower priority. In other words, the robot with lower priority will regard the robot with higher priority as a dynamic obstacle to be avoided, but not vice versa. Therefore, each robot can detect all fixed and dynamic obstacles in the space and assign the Voronoi corners on the map. Then the GVD map corresponding to the robot is generated.

Suppose that Fig. 6 is the GVD map of robot-1 (from the point of view of robot-1), Fig. 7 is the GVD map of robot-2 and shows that robot-1 is the dynamic obstacle in the GVD map of robot-2, and Fig. 8 shows that robot-1 and robot-2 are the dynamic obstacles in the GVD map of robot-3, and so on. Here, we define robot-$i$ has higher priority than robot-$j$, $j \geq i$.

![The GVD map of robot-1.](image6.jpg)

FIGURE 6. The GVD map of robot-1.

In addition, each robot adds a Voronoi corner inside its own body to allow itself to be surrounded by a Voronoi cell in its
GVD map [25] so that the robot can find its first NP easily without being affected by other obstacles (see Fig. 9). For instance, there are five obstacles and three robots in the space shown in Fig. 9 and the map in Fig. 9 is based on robot-2’s viewpoint. The NPs around robot-2 depends on the locations of obstacles or higher priority robot (robot-1). Because of the extra Voronoi corner inside robot-2, robot-2 is always inside the Voronoi cell (yellow area) corresponding to the extra Voronoi corner. Therefore robot-2 can easily have the closest NPs on the border of the cell such that the first NP for robot-2 moving will be on the border of the cell.

III. PATH PLANNING

A. SHORTEN THE FEASIBLE PATHS

This subsection introduces how to find the efficient path for each robot in the space. Any path must be one set of many navigation links. The initial position and the target point of all robots are set initially. Many possible paths can reach the target if each robot moves along links of NPs in its GVD map. To make the figure clearer, the Voronoi corners will be not shown in the following figures. For the GVD map of robot-3 shown in Fig. 10, robot-3 has many possible paths (only five links are drawn with blue color in the figure) to reach its target point. Generally, only the shortest path will be selected based on the Dijkstra algorithm [33] when GVD is used, and then the shortest path is called the first shortest navigation link (SNL-1). The remaining links are sorted to find the second one called the second shortest navigation links (SNL-2), the third shortest navigation links (SNL-3), etc. Let SNL-1 and SNL-2 be called the feasible paths. For instance, Yen’s algorithm [34] can find all possible paths (navigation links) between the starting point and the target point for each robot on its GVD map. In Fig. 11, Fig. 12, and Fig. 13, only both SNL-1 (blue) and SNL-2 (purple) are shown for the maps of robot-1, robot-2, and robot-3, respectively. In Fig. 11, robot-1 has two paths to be shown, one is moving around the obstacle on the upper path (SNL-2), and the other is on the lower path (SNL-1). In Fig. 12 and Fig. 13, the purple line overlaps with the blue line for the most part so that only a small section of the blue line can be seen. It means that the difference between SNL-2 and SNL-1 is tiny in this case. Whether SNL-1 or SNL-2, the moving path maybe not be efficient since the robot must move along the navigation links. Then the moving path may be far away from the obstacle to be avoided, so that the moving distance to the target may become longer.

Therefore, the moving path along the navigation links should be improved to shorten the entire path length. The flow chart of the improvement for the navigation linked path is shown in Fig. 16. Let Fig. 14 be an illustration, and the steps of Fig. 16 are as follows.

Step 1: Consider SNL-1 at the start of the flowchart. Set a parameter PP = 0 initially.
Step 2: Connect the link from the current robot’s position to the first NP and increase PP to 1, where the first NP is at the border of the cell containing the robot, and this link will not pass through the obstacle.
Step 3: Connect the link from the current robot’s position to the next NP along SNL-1 and check the link. If the link does not pass through the obstacle, set PP = original PP + 1, and go to Step 4. Otherwise, go to Step 5.

Step 4: Go to Step 3 until the link passes through the obstacle.

Step 5: Then the current PP is recorded, and the present link is a shortcut from the current position to the current NP.

Step 6: The robot moves to the current NP and goes to Step 3 until the next NP does not exist and is replaced by the target point. Then the path connected by many shortcuts will be the modified path shortened from SNL-1. Then go to Step 1 and repeat the algorithm but change SNL-1 to SNL-2.

Finish the above steps, and finally, we can get two modified paths shortened from SNL-1 and SNL-2, respectively, for each robot such as the green lines in Fig. 14 and Fig. 15. However, two modified paths shortened from two feasible paths for robot-2 (or robot-3) are the same in Fig. 12 (or Fig. 13). Those green modified paths are called the modified feasible paths or we call them modified SNL-1 and modified SNL-2.

B. SWITCH BETWEEN TWO FEASIBLE MODIFIED PATHS

When multiple robots are moving simultaneously in the same space, a lower priority robot moving along its modified feasible path may block higher priority robots’ movement. For example, in Fig. 17, there are three robots in a space without any obstacle, and each robot only uses its modified SNL-1, where any pink arrow indicates its robot moving direction. Suppose Fig. 17 is the GVD map of robot-2, and the green line is the modified SNL-1 of robot-2 towards its target point. It is seen in Fig. 17 that robot-1 will move to the right according to
its pink arrow and follows the modified SNL-1 (green dotted line), and robot-3 will move to the bottom. There is an intersection point of two modified SNL-1 of robot-1 and robot-2 shown in Fig. 18. If robot-2 is a little closer to the intersection point than robot-1, then robot-2 will arrive at the intersection point a litter earlier than robot-1. Since robot-1’s priority is higher than that of robot-2, robot-1 moves forward and ignores the moving obstacle (i.e., robot-2). Then there may be a collision between robot-1 and robot-2 at the intersection point. If the same situation happens in [25], robot-1 will wait until robot-2 leaves to avoid a collision. After that, robot-1 keeps going in the study since the higher priority robot has a higher priority to remain on its planned path than the lower priority one, but if the latter arrives at the point a litter earlier than the former, then the former may have to wait for the latter leaving the intersection point. Since the higher priority robot does not change its planned path but has to avoid hitting the lower priority robot, then the waiting situation of the higher priority robot may happen in [25]. On the other hand, if robot-1 arrives at the intersection point earlier than robot-2, the latter will bypass the intersection point to avoid hitting robot-1. This paper likes to resolve the waiting problem of the higher priority robot to improve the efficiency of higher priority robots’ operation.

![FIGURE 17. The status on using the modified SNL-1 only in the GVD map of robot-2 at the moment t.](image1)

The above waiting problem happens because we only use the modified SNL-1 for robot-1 and robot-2. The modified SNL-2 may be helpful in solving the problem. Let Fig. 19 denote the status at the moment between $t$ (Fig. 17) and $t + h$ (Fig. 18). In Fig. 19, robot-2 and robot-1 are far away enough to change their paths early. If robot-2 can choose its modified SNL-2 (the lower-left green line) instead of modified SNL-1 (upper right green line), then robot-1 will not be blocked by robot-2. In Fig. 20, the paths are the modified SNL-1 of robot-$j$ and robot-$n$, respectively, where $D_{j_{-ip}}$ is the distance between robot-$j$ and the intersection point, $\gamma_{jn}$ is the angle between the two modified SNLs of robot-$j$ and robot-$n$, and $\Psi_n$ is the vertical distance from the position of robot-$n$ to the path of robot-$j$. Suppose $j < n$, which means robot-$j$ has higher priority than robot-$n$. The following inequality is the condition for the path change. Once

$$\frac{D_{j_{-ip}}}{D_{n_{-ip}}} > 1.1 \quad (1)$$

or

$$D_{j_{-ip}} - D_{n_{-ip}} + (n - j) \times 5 > 0 \quad (2)$$

is satisfied, robot-$n$ must switch its modified SNL-1 to SNL-2 to avoid blocking robot-$j$. In addition, if the term $(n - j) \times 5$ in (2) is larger, the robot-$n$ switches its modified SNL-1 to SNL-2 earlier. The condition (1) or (2) is set according to the experience of many experiments.

![FIGURE 18. The status on using the modified SNL-1 only in the GVD map of robot-2 at the moment $t + h$.](image2)

![FIGURE 19. In the status of using the modified SNL-1 and modified SNL-2 in the GVD map of robot-2.](image3)

![FIGURE 20. Diagram of modified SNL between two robots.](image4)

For instance, there are three different time figures frames 48, 77, and 98 for multiple robots moving in a GVD map.
from robot-2’s viewpoint. Fig. 21 is the status of the moment after Fig. 18. At the moment in Fig. 21, robot-2 chooses the modified SNL-2 (green line) where (1) or (2) holds. As time goes by, robot-1 moves to the upper right of robot-2, and robot-2 moves left, as shown in Fig. 22, where robot-2 switches back to modified SNL-1 at this moment because there is no possibility of blocking robot-1. Finally, as shown in Fig. 23, robot-1 reaches the target point without being blocked, and robot-2 continues moving up.

**FIGURE 21.** Robot-2 chooses the modified SNL-2 to avoid blocking robot-1 at frame 48.

**FIGURE 22.** Robot-2 chooses the modified SNL-1 again to move towards the target point at frame 77.

**FIGURE 23.** Finally, robot-1 reached the target without being blocked at frame 98.

IV. COLLISION PREVENTION

If the lower priority robot blocks the higher priority robot probably, the former will switch paths from the modified SNL-1 to the modified SNL-2 to avoid blocking. However, if the lower priority robot does not have the modified SNL-2 or its modified SNL-2 and SNL-1 are the same, the switch is not available. For instance, in Fig. 24, robot-2 should do the switch to avoid blocking robot-1 when (1) or (2) happens at frame 25, but the two feasible modified paths of robot-2 are the same, so that the switch is useless. After a few seconds, the status in Fig. 25 occurs, where robot-1 can continue to move forward after waiting for a moment. Since robot-2 arrives at the intersection earlier than robot-1, it is better to let robot-2 move first in this status.

**FIGURE 24.** Robot 2 arrived at the intersection point earlier at frame 25.

**FIGURE 25.** Robot 2 moves first in the status at frame 50.

However, there are also different statuses under which we have to decide who needs to wait. This will be illustrated and explained in the following. Based on GVD, any robot will not hit fixed obstacles, but we need to have the following collision-free strategy to avoid hitting other moving robots. We give a sector in front of each robot, as shown in Fig. 26, where the sector denotes the area where the robot may hit other robots on moving forward. Furthermore, \( R_j \) is the sector’s radius and is set as multiples of the diameter of the robot-\( j \) such as \( R_j = h \times L_j, h \geq 2 \), and \( L_j \) is the diameter of robot-\( j \). \( \Theta \) is the center angle of the sector of a robot and it must be larger than 30 degrees. \( H_{jn} \) is the distance between robot-\( j \) and robot-\( n \). \( C_{jn} \) is the angle to which robot-\( j \) rotates to face robot-\( n \). In addition, \( D_{j,ip} \), \( D_{n,ip} \), \( Y_{jn} \), and \( \Psi_n \) have been defined in Fig. 20. We will explain the moving strategy for robots by the flow chart in Fig. 27 or Fig. 28 for different situations. Suppose \( j < n \), when the higher priority robot-\( j \) appears in the sector of the lower priority robot-\( n \),
the flow chart in Fig. 27 will be used by robot-\( n \) to make the moving decision. In addition, assume \( R_n > H_m \) and \( \Theta > G_{nj} \), which means that robot-\( j \) exists inside the sector of robot-\( n \). If both robots’ modified SNL-1 or SNL-2 are intersected and \( \gamma_{jn} < 120^\circ \), it means that the lower priority robot (robot-\( n \)) needs to stop and wait until the higher priority robot (robot-\( j \)) passes. It is noted that the upper bound of \( \gamma_{jn} \) is set by the designer’s experience; if the upper bound of \( \gamma_{jn} \) is more than \( 120^\circ \), it will have a collision risk. The larger the upper bound, the easier the lower priority robots block the higher priority robots. On the other hand, when the lower priority robot-\( n \) appears in the sector of the higher priority robot-\( j \), the flow chart in Fig. 28 will be used by robot-\( j \) to make the moving decision. If \( \Psi_n < L_j \) is satisfied, robot-\( n \) is too close to robot-\( j \) that robot-\( j \) needs to stop to avoid hitting robot-\( n \).

![FIGURE 26. The sectors of two robots.](image)

![FIGURE 27. When the higher priority robot appears in the sector of the lower priority robot.](image)

A. THE CALCULATION FOR A ROBOT MOVING

To simulate the real path and verify the feasibility of the proposed algorithm, this section wants to calculate the next position from the robot’s current position with the known next navigation point. Let the polar coordinate be set between a robot’s position and its next navigation point as shown in Fig. 29, where the navigation point is the origin and the robot position is denoted by \( R(\rho, \theta) \); where \( \rho \) is the distance between the robot and the navigation point, and \( \theta \) is the orientation angle of the robot in the polar coordinate. Let \( \phi \) be the angle between the moving direction of the robot and the horizontal axis, and \( \beta = \pi - \theta \) and \( \alpha = -\beta - \phi \) are defined. Suppose \( \hat{u}_1 = \frac{r_\rho}{l} \cdot \rho \) means the forward speed of the robot depends on the distance from the robot to the navigation point, and \( \hat{u}_2 = \frac{l}{k_\rho} \cdot \alpha \) means the rotational speed which is determined by the angles of \( \alpha \). Assume \( v_l \) and \( v_r \) indicate the velocities of the left and right wheels of the robot, respectively. It is noted that we also know \( \hat{u}_1 = (v_r + v_l)/2 \) and \( \hat{u}_2 = (v_r - v_l)/2l \). Based on the above parameter definitions, we can get the velocities of the left wheel as \( v_l = k_\rho \cdot \rho - l \cdot k_\alpha \cdot \alpha \) and the right wheel as \( v_r = k_\rho \cdot \rho + l \cdot k_\alpha \cdot \alpha \), respectively, where \( k_\rho \) and \( k_\alpha \) are two constants, respectively. Next, we adopted the kinematics model (3) of the robot moving from the papers [35], [36].

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

\[u_1 = (v_{r,\text{nor}} + v_{l,\text{nor}})/2,
\]

and

\[u_2 = (v_{r,\text{nor}} - v_{l,\text{nor}})/2l.
\]

with

\[v_{r,\text{nor}} = k \times \frac{v_r}{|v_r| + |v_l|}.
\]

V. ROBOT CONTROL AND SIMULATION

In the above sections, path planning and obstacle avoidance have been processed. This section will discuss the robot moving control and give a simulation for illustrating the robot moving.
FIGURE 29. The polar coordinate of the wheeled robot [25].

\[ v_{l\,nor} = k \times \frac{v_l}{|v_r| + |v_l|}. \]  

(7)

The left and right wheel speeds \(v_l\) and \(v_r\) are normalized as eq. (6) and (7) to avoid speed reduction when \(\rho\) is smaller, i.e., when the robot gets closer to the navigation point. If we have \(v_l\) and \(v_r\), and the current position \(\rho\) and \(\theta\) of the robot corresponding to the next navigation, then based on (3), we can calculate the next position of the robot. We do the above calculations continuously for the robot, then we can have the moving trajectory of the robot in the simulation of the next subsection. Finally, we give a flow chart as Fig. 30 to summarize the above processes for the proposed PONA.

B. SIMULATION DEMONSTRATIONS

The simulation environment is based on the visual studio 2019 platform using C++ programming language, and simulation result figures are drawn using the OpenCV. The following assumptions are for the simulation below.

1. The radius of a robot is 20 pixels.
2. The speeds of all robots are the same fixed constant, i.e., \(v_l\) and \(v_r\) are constants.
3. The distance between the starting point and the target point is 520 pixels for all robots.
4. \(R = 2.5 \times L_j, L_j = L_n, \Theta = 45^\circ\).

In the simulation, we set \(k_\rho = 0\), \(k_\alpha = 1\), and \(k = 0.3\) when \(|\alpha| > 0.03\) in mode 1 which is also called the rotation mode. \(k_\rho = 1\), \(k_\alpha = 0.01\), and \(k = 200\) when \(|\alpha| \leq 0.03\) in mode 2 which is called the straight mode. In mode 1, the robot turns its orientation and in mode 2, the robot moves forward straightly. The threshold of 0.03 for \(\alpha\) is set, because in a computer simulation or real experiment, it is impossible to have \(\alpha = 0\). If we do not have such a threshold, the robot may swing its head left and right in mode 1 and never becomes mode 2. It is noted that \(k_\alpha\) can be set to any positive value in mode 1 and \(k_\rho\) can be set to any positive value in mode 2 since they will be canceled in (6) and (7). The speed of the robot is determined by \(k\) only. In mode 2, \(k_\rho\) only needs a small value based on \(\alpha\) to assure the robot moves toward the NP.

FIGURE 31. The initial positions of all robots at frame 1 (in the GVD map of robot-2).

In the first simulation, there are three robots on the map, such as Fig. 31, and they move round trips between their starting point and their target point, respectively. Robot 1 has the highest priority, and robot-3 has the lowest priority. Each robot has a number beside it that represents the number of round-trip times between the robot’s target point and starting point. A collision counter is set at the upper left corner of the map. If the robot hits any other robot once, its count will increase by one. Therefore, in this simulation, if the counter remains at zero, there is no collision between robots. In the following figures, all trajectories of each robot are recorded. Fig. 31 is the GVD map of robot-2, where the initial positions of all robots are shown and the target of robot-2 is the yellow dot in the figures. It is seen in Fig. 31, robot-1 is regarded as an obstacle for robot-2. As time goes by, in Fig. 32, robot-2 has chosen the modified SNL-2 to avoid blocking robot-1. In Fig. 33, robot-2 chooses modified SNL-1 again to move forward to the target point.
From Fig. 34-Fig. 36, if we consider only one trip, all robots reach their targets without collision, respectively, as shown in Fig. 37. Then let each robot keep moving for the round trips between the starting and target points many times. At the moment, shown in Fig. 38, where robots 1, 2, and 3 have moved 12, 11.5, and 10 round trips, respectively. Fig. 39 shows the moment that robot-1 has achieved 49.5 round trips, robot-2 has achieved 48.5 round trips, and robot-3 has achieved 35 round trips. Fig. 38 and Fig. 39 show that the proposed PONA is effective for three robots in the same space without fixed obstacles since the higher priority robot achieves more round trips than the lower priority robots at the condition of the same one trip distance and same time spent.

To highlight the efficiency of the proposed PONA, we gave one more simulation in the case of eight robots in different environments as shown in Figs. 39-42, where eight robots are initially set evenly distributed on the top of the map, and their target points are straightly on the bottom of the map corresponding to their starting points, respectively. In Fig. 40, eight robots move to the targets, respectively, with a straight line on an obstacle-free map. Fig. 41 and Fig. 42 show the moving trajectories of eight robots with different numbers of
obstacles on the map. Fig. 43 shows the moving trajectories of all robots with a large obstacle on the map.

VI. COMPARISON RESULTS

This section will give a comparison between the proposed PONA and other existing algorithms for the multi-robot navigation in the environment shown in Fig. 44 [17], where all robots are evenly distributed on a circle with a radius of 260 pixels. Those existing algorithms are ROA and SDA in [17], and NSPP in [25]. The start points of all robots are set arbitrarily from P1 to P8, and the target points are on the opposite side of the corresponding starting points, respectively. Initially, each robot is at its starting point and towards its target point. For example, the robot at P3 will move to and stop at P7, and the robot at P7 will move to and stop at P3. In the simulation, all robots start moving at the same time. In the first experiment, there are only four robots evenly distributed on the circle. Let PONA(1234) denote that PONA is used and robots-1, 2, 3, and 4 are at the starting points P1, P3, P5, P7, initially and respectively, and their target points are at P5, P7, P1, and P3, respectively. In other words, the number 1234 in the square bracket denotes the arrangement order (it can be called the configuration) of four robots. There will be six configurations for four robots such as PONA(1234), PONA(1243), PONA(1324), PONA(1342), PONA(1423), PONA(1432). In the case of eight robots, there are PONA(14567823) means robot numbers 1, 4, 5, 6, 7, 8, 2, and 3 are initially at P1, ..., P8, respectively, and totally there will be 5040 configurations (i.e., \(7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1\)). In the simulation below, we select all configurations for four robots but only six configurations (see Table 2) for eight robots in the comparison.

For the experiment of four robots, PONA(1324) in Fig. 45 and PONA(1342) in Fig. 46 are the best and the worst results, respectively, in all six configurations. For the experiment of eight robots, PONA(18526743) in Fig. 47 and PONA(15348726) in Fig. 48 are the best and the worst results, respectively, in the mentioned six configurations. The experiments have shown that higher priority robots usually have shorter trajectories. It is seen that the worse results always happen in the case that the lowest priority robots, such as robot-4 in Fig. 46 and robot-8 in Fig. 48, move the longest distance to avoid blocking the higher priority robots.
To compare the performance, define

\[ \text{ATL (average trajectory length)} = \frac{\text{Sum of the lengths of all robot trajectories}}{\text{Number of robots}} \]  
(8)

and

\[ \text{MTL (maximum trajectory length)} = \text{Maximum trajectory length in all trajectories of robots} \]  
(9)

We selected 12 configurations in four and eight robots to calculate their performance using (8) and (9), and the comparison results are shown in Table 1 and Table 2. In both figures, all data of PONA in the ATL column are smaller than those of ROA and SDA. In the MTL column of Table 1, all data of PONA are better than those of ROA and SDA too.
TABLE 2. Trajectory lengths comparison for different algorithms with eight robots.

| Algorithm       | ATL   | MTL   |
|-----------------|-------|-------|
| ROA             | 687   | 687   |
| SDA             | 652   | 652   |
| NSPP(15462738)  | 577   | 664   |
| NSPP(12345678)  | 567   | 664   |
| NSPP(18267453)  | 588   | 680   |
| NSPP(14632875)  | 555   | 607   |
| NSPP(18247635)  | 588   | 672   |
| NSPP(15348726)  | 575   | 652   |
| PONA(15462738)  | 585   | 708   |
| PONA(12345678)  | 599   | 764   |
| PONA(18267453)  | 567   | 609   |
| PONA(14632875)  | 563   | 614   |
| PONA(18247635)  | 583   | 679   |
| PONA(15348726)  | 618   | 834   |

The average MTL of PONA is 556 pixels which are less than the average MTL of NSPP (608 pixels) in Table 1. But in Table 2, the MTL of PONA is not good enough, since the lowest priority robot (robot-8) has to spend much more time to avoid blocking higher priority robots.

We also compared the arrival order of robots between NSPP and PONA in Table 3 and Table 4. It is noted that NSPP has good performance in both ATL and MTL, but the arrival order result of NSPP does not match the priority order of robots (see Table 1 and Table 3). In other words, the robot with higher priority in NSPP will have a shorter moving length than PONA but may not have a shorter time spent. Therefore, the higher priority robot may arrive at the target point later than the lower path priority robot since the former may often stop for a while to avoid hitting the latter.

TABLE 3. The order of the four robots’ arrival.

| Four robots in circular test | Order of robot arrival |
|-----------------------------|------------------------|
| NSPP(1234)                  | 2→3→4→1               |
| NSPP(1243)                  | 2→3→1→4               |
| NSPP(1324)                  | 3→4→1→2               |
| NSPP(1324)                  | 3→4→2→1               |
| NSPP(1432)                  | 4→2→1→3               |
| NSPP(1432)                  | 4→2→3→1               |
| NSPP(1234)                  | 3→4→1→2               |
| NSPP(1243)                  | 3→4→2→1               |
| PONA(1234)                  | 1→2→4→3               |
| PONA(1234)                  | 1→2→3→4               |
| PONA(1234)                  | 1→2→4→3               |
| PONA(1234)                  | 1→2→3→4               |
| PONA(1234)                  | 1→2→3→4               |

In Table 3, 1→2→3→4 represents robot-1 arriving at its target first and robot-4 arriving at its target last. Compared to NSPP, PONA proposes a much better arrival order of robots without increasing MTL in Table 3 and Table 4. In Table 3, the arrival order of four robots using PONA is much more consistent with their priority order than using NSPP. In Table 4, the arrival order of PONA seems a little better than that of NSPP too.

From the above comparison, all algorithms for multi-robot navigation can provide collision-free paths, but PONA can give the shortest ATL and MTL (except in one case) in four robots configuration. However, it is hard to decide who is better than the others for eight robots configuration. We only can say that PONA can improve the moving efficiency of the higher priority robots, but the lower priority robot’s trajectories may be longer since it needs to change its modified SNL frequently to avoid blocking the higher priority robots.

VII. CONCLUSION

This paper has proposed a priority order-based multi-robot path planning algorithm (PONA) established based on GVD, modified feasible paths planning, and robot moving control. The algorithm is developed from the idea of NSPP such that the travel efficiency of higher priority robots is improved. Of course, collision-free is guaranteed by PONA too. Table 5 shows the average ATL percentage reduction of PONA compared with ROA, SDA, and NSPP. It is seen that compared with NSPP in the eight robots case, the PONA has a little larger average ATL than NSPP, but PONA improves the waiting problem of the higher priority robot significantly. The proposed PONA can be implemented with any number of robots, but we cannot guarantee PONA will have better performance in more than eight robots configuration. However, this paper still has proposed a novel idea to improve the travel efficiency for multi-robot navigation.
REFERENCES

[1] S. Wang, D. Li, J. Wan, and C. Zhang, “Implementing smart factory of Industry 4.0: An outlook,” *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 1, 2016, Art. no. 3159805.

[2] N. V. Kumar and C. S. Kumar, “Development of collision free path planning algorithm for warehouse mobile robot,” *Proc. Comput. Sci.*, vol. 133, pp. 456–463, Jan. 2018.

[3] S. G. Cui, S.-K. Huang, and W.-J. Wang, “A simulation study of A-star algorithm for robot path planning,” in *Proc. 16th Int. Conf. Mechatronics Technol.*, 2012, pp. 506–509.

[4] D. Keymeulen and J. Decuyper, “The fluid dynamics applied to mobile robot motion: The stream field method,” in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, May 1994, pp. 378–385.

[5] P. Bhattacharya and M. Gavrilova, “Roadmap-based path planning—using the Voronoi diagram for a clearance-based shortest path,” *IEEE Robot. Autom. Mag.*, vol. 15, no. 2, pp. 58–66, Jun. 2008.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[7] M. Elleosyen, A. Tharwat, and A. E. Hassanien, “Bezier curve based path planning in a dynamic field using modified genetic algorithm,” *J. Comput.*.

[8] F. Duchoti, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Šenická, “Path planning with modified a star algorithm for a mobile robot,” *Proc. Eng.*, vol. 96, pp. 59–69, Jan. 2014.

[9] S. S. Ge and Y. I. Cui, “Dynamic motion planning for mobile robots using potential field method,” *Automo. Robots*, vol. 13, no. 3, pp. 207–222, 2002.

[10] P. Das, S. Pradhan, S. Patro, and B. Balabantaray, “Artificial immune system based path planning of mobile robot,” in *Soft Computing Techniques in Vision Science*, vol. 395, S. Patnaik and Y.-M. Yang, Eds. Berlin, Germany: Springer, 2012, pp. 195–207.

[11] F. Belkhouche, “Reactive path planning in a dynamic environment,” *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 902–911, Aug. 2009.

[12] J. Alonso-Mora, S. Baker, and D. Rus, “Multi-robot navigation in formation via sequential convex programming,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 4634–4641.

[13] P. K. Das, H. S. Behera, P. K. Jena, and B. K. Panigrahi, “Multi-robot path planning in a dynamic environment using improved gravitational search algorithm,” *J. Electr. Syst. Inf. Technol.*, vol. 3, no. 2, pp. 295–313, Sep. 2016.

[14] C. Huang, X. Chen, Y. Zhang, S. Qin, Y. Zeng, and X. Li, “Switched linear multi-robot navigation using hierarchical model predictive control,” in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 4331–4337.

[15] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent pathfinding,” *Artif. Intell.*, vol. 219, pp. 40–66, Feb. 2015.

[16] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[17] A. A. Ali, A. T. Rashid, M. Frasca, and L. Fortuna, “An algorithm for multi-robot collision-free navigation based on shortest distance,” *Robot. Autonom. Syst.*, vol. 75, pp. 119–128, Jan. 2019.

[18] D. C. Rao, M. R. Kabat, P. K. Das, and P. K. Jena, “Cooperative navigation of multiple mobile robots using improved krill herd,” *Arabian J. Sci. Eng.*, vol. 43, no. 12, pp. 7869–7891, Dec. 2018.

[19] C. H. Sun, Y.-J. Chen, S.-K. Huang, W.-J. Wang, and J. H. Sun, “A path planning strategy for multi-robot moving with path-priority order based on a generalized Voronoi diagram,” *Appl. Sci.*, vol. 11, no. 20, p. 9650, Oct. 2021.

[20] H. Imai, M. Iri, and K. Murata, “Voronoi diagram in the Laguerre geometry and its applications,” *SIAM J. Comput.*, vol. 14, no. 1, pp. 93–105, Feb. 1985.

[21] J. Kim and H. I. Son, “A Voronoi diagram-based workspace partition for weak cooperation of multi-robot system in orchard,” *IEEE Access*, vol. 8, pp. 20676–20686, 2020.

[22] Z. Gabbassova, D. Sedighizadeh, A. S. Fini, and M. Seddighizadeh, “Multiple robot motion planning considering shortest and safest trajectory,” *Electromechanical Energy Convers. Syst.*, vol. 1, no. 3, pp. 1–6, 2019.

[23] N. Ben Slimane and M. Tagina, “Proposition of a distributed Voronoi partitioning approach enhanced with a dispersion phase for a multirobot system,” in *Int. J. Social Robot.*, vol. 13, no. 5, pp. 887–898, Aug. 2021.

[24] D. T. Lee, “Generalized Voronoi diagrams in the plain,” *SIAM J. Comput.*, vol. 10, no. 1, pp. 73–87, 1981.

[25] J. Wang and M. Q.-H. Meng, “Optimal path planning using generalized Voronoi graph and multiple potential functions,” *IEEE Trans. Ind. Electron.*, vol. 67, no. 12, pp. 10621–10630, Dec. 2020.

[26] W. Chi, Z. Ding, J. Wang, G. Chen, and L. Sun, “A generalized Voronoi diagram-based efficient heuristic path planning method for RRTs in mobile robots,” *IEEE Trans. Ind. Electron.*, vol. 69, no. 5, pp. 4926–4937, May 2022.

[27] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[28] J. Y. Yen, “An algorithm for finding shortest routes from all source nodes to a given destination in general networks,” *Quart. Appl. Math.*, vol. 27, no. 4, pp. 526–530, 1970.

[29] C.-H. Sun, Y.-J. Chen, Y.-T. Wang, and S.-K. Huang, “Sequentially switched fuzzy-model-based control for wheeled mobile robot with visual odometry,” *Appl. Math. Model.*, vol. 47, pp. 765–776, Jul. 2017.

[30] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. Cambridge, MA, USA: MIT Press, 2011.

**S.-K. Huang et al.: New Multirobot Path Planning With Priority Order Based on GVD**

---

**SHENG-KAI HUANG** is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, National Central University. He has been a Researcher with the Intelligent Control and Image Processing Laboratory, National Central University, since 2016. His research interests include multi-robot path planning, although he has concurrent research in visual odometry and image processing.

---

**WEN-JUNE WANG** (Fellow, IEEE) received the B.S. degree in control engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 1980, the M.S. degree in electrical engineering from Tatung University, Taipei, Taiwan, in 1984, and the Ph.D. degree in electronics from the National Chiao Tung University, in 1987. He has authored or coauthored more than 160 refereed journal articles and 160 conference papers in the areas of fuzzy systems and theorems, robotic and nonlinear control in large-scale systems, and neural networks. His most significant contributions are the design of fuzzy systems and the development of robotics. His current research interests include robot control, neural networks, and pattern recognition. He was an IFSA Fellow, in 2017.

---

**CHUNG-HSUN SUN** received the B.S. and Ph.D. degrees in electrical engineering from National Central University, Taiwan, in 1999 and 2006, respectively. From 2008 to 2017, he was with the Department of Mechanical and Electro-Mechanical Engineering, Tamkang University, Taiwan. He is currently an Associate Professor with the Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Taiwan. His current research interests include fuzzy control systems, robotics, and computer vision.