frontiers | Frontiers in Plant Science

TYPE Original Research
PUBLISHED 10 September 2024
DOI 10.3389/fpls.2024.1393541

OPEN ACCESS

EDITED BY
Daobilige Su,
China Agricultural University, China

REVIEWED BY
Xiangjun Zou,
South China Agricultural University, China
János Botzheim,
Eötvös Loránd University, Hungary

*CORRESPONDENCE
Ranbing Yang
yangranbing@163.com

RECEIVED 29 February 2024
ACCEPTED 16 August 2024
PUBLISHED 10 September 2024

CITATION
Xu K, Xing J, Sun W, Xu P and Yang R (2024) Multi-robot collision avoidance method in sweet potato fields.
Front. Plant Sci. 15:1393541.
doi: 10.3389/fpls.2024.1393541

Multi-robot collision avoidance method in sweet potato fields

Kang Xu[1], Jiejie Xing[2], Wenbin Sun[1], Peng Xu[1] and Ranbing Yang[2*]

[1]College of Information and Communication Engineering, Hainan University, Haikou, China, [2]College of Mechanical and Electrical Engineering, Hainan University, Haikou, China

Currently, precise spraying of sweet potatoes is mainly accomplished through semi-mechanized or single spraying robots, which results in low operating efficiency. Moreover, it is time-consuming and labor-intensive, and the pests and diseases cannot be eliminated in time. Based on multi robot navigation technology, multiple robots can work simultaneously, improving work efficiency. One of the main challenges faced by multi robot navigation technology is to develop a safe and robust collision avoidance strategy, so that each robot can safely and efficiently navigate from its starting position to the expected target. In this article, we propose a low-cost multi-robot collision avoidance method to solve the problem that multiple robots are prone to collision when working in field at the same time. This method has achieved good results in simulation. In particular, our collision avoidance method predicts the possibility of collision based on the robot's position and environmental information, and changes the robot's path in advance, instead of waiting for the robot to make a collision avoidance decision when it is closer. Finally, we demonstrate that a multi-robot collision avoidance approach provides an excellent solution for safe and effective autonomous navigation of a single robot working in complex sweet potato fields. Our collision avoidance method allows the robot to move forward effectively in the field without getting stuck. More importantly, this method does not require expensive hardware and computing power, nor does it require tedious parameter tuning.

KEYWORDS

multi-robot, accurate spraying, collision avoidance, sweet potatoes, itinerary table

1 Introduction

Sweet potato is an important food crop after rice, wheat, and corn. With the development of large-scale sweet potato cultivation, sweet potato diseases and pests are becoming increasingly serious. Traditional prevention and treatment methods are high in cost and low in efficiency, and there are problems such as prevention and treatment of chemical abuse and overuse. Harmful pesticide chemicals will spread into the air, causing pollution to the environment and sweet potato crops, and may enter farmers' bodies

through the respiratory system (Meshram et al., 2022). Precision spraying technology can solve the problems of pesticide waste and overuse, improve pesticide utilization, and reduce pesticide pollution to the environment (Xiongkui, 2020). However, in the current precision spraying operations, a single robot is mainly used for spraying operations, such as Danton et al., 2020; Baltazar et al., 2021; Oberti and Schmilovitch, 2021, and Liu et al., 2022. A single robot has the disadvantages of limited operating capacity and long operating time. Especially for sweet potato fields with a relatively large area, using a single robot for spraying will take a lot of time (Kim and Son, 2020), and pests and diseases cannot be eliminated in time, causing irreparable economic losses to agriculture. In addition, if the robot fails, the entire spraying operation will be delayed, resulting in greater economic losses. Therefore, applying a multi-robot system to sweet potato spraying can effectively solve the problem of low operating efficiency of a single device, reduce agricultural maintenance costs, and indirectly improve agricultural economic benefits. There are many benefits to using a multi-robot system instead of a single robot (Parker et al., 2016; Jawhar et al., 2018): (1) Multiple robots can perform tasks simultaneously to complete tasks faster. (2) Multiple robots can effectively handle tasks that are essentially distributed over a wide area. (3) When any robot fails, multiple robots that can perform similar processes can be used to compensate.

As the number of agricultural workers continues to decrease around the world, the use of multi-robot systems to perform agricultural tasks has become increasingly common in large-scale fields with fewer people (Carbone et al., 2018), and multi-robot systems have gradually become a hot topic of research. However, one of the main challenges facing multi-robot systems is to develop a safe and robust collision avoidance strategy that enables each robot to navigate safely and efficiently from the starting position to the desired target (Long et al., 2017). The robot's obstacle avoidance method can be divided into local obstacle avoidance and global path according to the operation requirements (Tang et al., 2024), such as Genetic algorithm (Zhao et al., 2021), dynamic window approach (Chang et al., 2021), RRT* (Hu et al., 2024), A* (Zhang et al., 2024) and other algorithms. However, these obstacle avoidance methods are all for a single robot and are not suitable for multi-robot systems (Cai et al., 2007). At present, the collision avoidance methods of multi-robots are mainly divided into centralized control and decentralized control.

Initially, scholars used a centralized control method to avoid collisions between multiple robots. This approach assumed that the behavior of all robots was determined by a central controller that had comprehensive knowledge of all robots' intentions (e.g., initial states and goals) and their workspaces (e.g., 2D grid maps). Schwartz and Sharir (1983) proposed to find the continuous motion of two given structures connecting objects within a given region, during which they avoided collisions with walls and each other. He et al. (2016) used the cloud platform to calculate local collision-free paths for each robot, which improved the processing capabilities of the device. However, this method needed to ensure that each robot had networking capabilities. Yu and LaValle (2016); Tang et al. (2018); Matoui et al. (2020) and Cheng et al. (2023) proposed a centralized trajectory generation algorithm to find trajectories that navigate robots from starting positions to non-interchangeable target positions in a collision-free manner by planning optimal paths for all robots. In addition, Keshmiri and Payandeh (2009) used a centralized method to keep multiple robots in formation to avoid collisions between robots, but it was not suitable for the agricultural field and cannot control multiple robots to maintain formation during precise spraying. Centralized approaches can ensure multi-robot system safety, integrity, and near-optimality, but they are difficult to scale to large systems with many robots due to the high computational cost of multi-robot scheduling, heavy reliance on reliable synchronous communication, and low tolerance for failures or interference.

Compared with centralized methods, some existing studies proposed decentralized collision avoidance strategies, where each robot made decisions independently by considering the observable states (such as shape, speed, and position) of other robots. Cai et al. (2007) developed an advanced method for collision avoidance in multi-robot systems based on established techniques of omnidirectional vision systems, automatic control, and dynamic programming. However, each robot in this system must be an autonomous robot, equipped with equipment or systems such as omnidirectional vision systems, target recognition systems, communication systems, and control systems. Claes et al. (2012); Hennes et al. (2012), and Godoy et al. (2016) designed inter-agent communication protocols to share position and velocity information among nearby agents. However, communication systems posed additional difficulties, such as delays or blocking of communication signals due to obstruction by obstacles. Althoff et al. (2012) proposed a probabilistic threat assessment method for reasoning about the safety of robot trajectories. In this method, the trajectories of other dynamic obstacles needed to be sampled and then the global collision probability was calculated. Lacked of possibilities for safe navigation in dynamic multi-robot environments. Sun et al. (2014) proposed a behavior-based multi-robot collision avoidance method in large robot teams inspired by the concept of swarm intelligence, which could solve the coordination problem of multi-robots by using group behavior. Fiorini and Shiller (1998), Van den Berg et al. (2008); Snape et al. (2009); Hennes et al. (2012); Alonso-Mora et al. (2013); Claes and Tuyls (2018), and Zhu et al. (2020) designed a robot collision avoidance strategy based on the speed obstacle framework. This algorithm effectively inferred the local collision-free motion of multiple agents in a chaotic workspace. However, these methods have several serious limitations that hinder their widespread application in practical scenarios. First of all, it is difficult to meet the requirement that each agent has perfect perception of the surrounding environment. This requirement does not hold true in real scenarios. A second limitation is that speed barrier-based strategies are controlled by multiple adjustable parameters that are sensitive to scene settings and therefore must be set carefully to achieve satisfactory multi-robot locomotion.

Inspired by methods based on the speed obstacle framework, Chen et al. (2017b), and Chen et al. (2017a) trained a collision avoidance strategy using deep reinforcement learning. This strategy explicitly mapped the agent's own state and that of its neighbors to collision-free actions, but it still required perfect awareness of the

surrounding environment. Long et al. (2017); Pfeiffer et al. (2017); Wang et al. (2020); Huang et al. (2022); Xiao et al. (2022), and Han et al. (2022) used deep neural network modeling and trained using supervised learning on large datasets. However, there are several limitations to supervised learning policies. First, it requires a large amount of training data, which should cover different interaction situations of multiple robots, and the training time cost is high. Secondly, the expert trajectories in the data set are not guaranteed to be optimal in interactive scenarios, which makes it difficult for training to converge to a robust solution. Third, it is difficult to manually design a suitable loss function to train a robust collision avoidance strategy. Fourth, the hardware requirements required for model deployment on robots are relatively high. When the number of robots increases, the hardware cost increases exponentially. Multi-robot systems based on reinforcement learning or transfer learning require a large amount of data sets for training, which requires relatively high time and economic costs. However, the economic benefits obtained in agriculture are obviously not suitable for the large-scale application of such robots.

Ünal et al. (2023) proposed a robot collision avoidance algorithm for traveling along tangent paths to solve the problem of multi-robot collision avoidance in precision agriculture. However, they only considered two robots and did not introduce the specific application environment. In addition, they did not consider whether the robot would damage the surrounding crops when avoiding collision. Currently, research on multi-robot collision avoidance strategies mainly focuses on confined space scenes or in relatively open and empty spaces, and no scholars have considered how to avoid collisions among multi-robots in large field (Raibail et al., 2022). Due to the special field environment, the robot can only move along the ridges, and can only move forward or backward. It cannot move freely in all directions, otherwise it will damage the crops and the robot, causing unnecessary economic losses. The current multi-robot collision avoidance strategy cannot be directly applied to this situation. In order to overcome the shortcomings of the above method, we propose a low-cost multi-robot collision avoidance method, which uses position and known environmental information to predict the possibility of collision, and changes the robot path in advance to achieve the purpose of robot collision avoidance. The method outperforms existing agent- and sensor-level methods in terms of navigation speed and safety. It can be directly and smoothly deployed on physical robots, without the need for expensive laser radars, cameras, and terminal controllers, and without the need for cumbersome parameter adjustments. Based on this, we propose a multi-robot collision-free method suitable for the special environment of field. The main innovations and contributions are as follows:

1. The multi-robot collision avoidance method we proposed only requires the robot to have a communication module, a positioning module and a low-cost control chip. It does not require the robot to have vision, radar and other sensors, nor does it require the ability to deploy reinforcement learning models. Reduce farmers' purchase costs and field maintenance costs, thereby increasing farmers' economic income.
2. We proposed a rapid robot path generation method suitable for farmland. The global information of the farmland was generated through the two endpoints of the crop rows and the row spacing. The global information included the straight-line equation and the two endpoints of the crop rows, occupying very little storage space. Then based on the requirements of farm operations and the robot's target location, we could quickly generate a global path for the robot.
3. We proposed a robot collision avoidance method that could determine its own direction and working status only based on global information and the robot's position. Then the robot's position and global information were used to predict the possibility of collision in advance. Finally, the robot's local path was adjusted according to the robot's priority to avoid collisions. This method could quickly find collision-free paths for multi-robot systems and can be safely generalized to other work scenarios.

The rest of this paper is as follows: Section 2 introduces the working environment of multi-robot systems, related definitions and collision types. In Section 3, we discussed in detail how to predict the possibility of robot collision through maps and robot positions, and how to adjust local paths in the special environment of field to avoid collisions among multiple robots. The fourth section introduces the simulation environment and simulation settings of the multi-robot collision avoidance method, and verifies the method proposed in the third section. The final section presents the summary and prospects of this study.

2 Materials and methods

2.1 The working environment of robots

The sweet potato field is simplified to facilitate the description of the robot's working conditions, as shown in Figure 1. The rectangle $F_aF_bF_cF_d$ represents the field, and R_n $(n \in N^*)$ represents the working robots (Figure 1A). The line segments A_jB_j $(j \in N^*)$, A_sA_u, and B_sB_u $(s \neq u,\ 0 < s \leq j, 0 < u \leq j)$ represent the robot's paths in the field (Figure 1B). The line segment A_jB_j is called the working path, the points A_j and B_j are the endpoints of the A_jB_j, respectively, and j is the serial number of the working path. Line segments A_sA_u and B_sB_u are called the transition path. We set the sweet potato field to be planted in a standardized manner, with working paths running parallel to each other. The distance between each working path is represented by D.

2.1.1 Definitions of robots

R represents a group of working robots, as shown in Equation 1. R_n has the same dynamic model in vertical and horizontal dimensions. The real-time location of R_n is represented by P_{R_n}, and P_{R_n} is represented by coordinates (x_{R_n}, y_{R_n}).

$$\boldsymbol{R} = \left\{ \boldsymbol{R_n} | \boldsymbol{n} \in \boldsymbol{N^*} \right\} \tag{1}$$

10.3389/fpls.2024.1393541

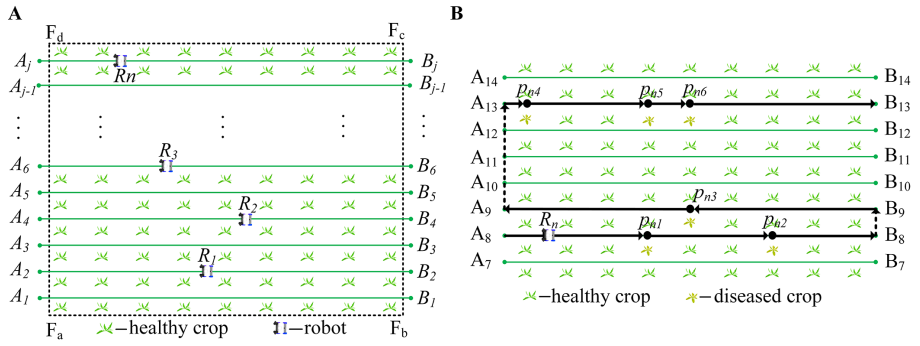

FIGURE 1
A simplified diagram of the robot's working environment. **(A)** Simplified diagram of field. **(B)** The working path of the robot.

R_O represents other robots except for R_n, as shown in Equation 2.

$$\boldsymbol{R_O = R - R_n} \tag{2}$$

A group of target points of robot R_n is represented by P_n, as shown in Equation 3. Robot R_n moves at a constant speed along the target point. After robot R_n reaches target P_n, it stops for 3 seconds and performs precision spraying operations.

$$\boldsymbol{P_n = \{p_{ni} | p_{ni} \in R^2, n \in N^*, i \in N^*\}} \tag{3}$$

where the coordinates of p_{ni} are represented by (x_{ni}, y_{ni}), and x_{ni} and y_{ni} are the horizontal and vertical coordinates on a two-dimensional plane, respectively. r^n represents the global path of the robot, and r_j^n represents the local path of the robot, as shown in Equation 4. The local path comprises a series of points, such as $r_8^n = \{A_8, p_{n1}, p_{n2}, B_8\}$ and $r_9^n = \{B_9, p_{n3}, A_9\}$, as shown in the solid black line in Figure 1B. r^n comprises a series of local paths, such as all-black paths in Figure 1B.

$$\boldsymbol{r^n = \{r_j^n | n \in N^*, j \in N^*\}} \tag{4}$$

There are two types of paths: the working and transition. In Figure 1B, the solid black lines A_8B_8, A_9B_9, and $A_{13}B_{13}$ are the working path, and the dotted black lines B_8B_9 and A_9A_{13} are the transition path.

The robot has different moving directions on the working and transition paths. When the robot is moving on the working path, the robot's direction at points A_j and B_j are 0 and 1, respectively. When going from point A_j to B_j, the direction of the robot is $0 \rightarrow 1$. When the robot goes from point B_j to A_j, the robot's direction is $1 \rightarrow 0$. Therefore, we can get the direction of the robot only through the two endpoints of the local path without using other sensors. The expression method of the robot's moving direction in the working path is shown in Equation 5.

$$\begin{aligned} \boldsymbol{r_8^n} &= \boldsymbol{\{A_8, p_{n1}, p_{n2}, B_8\}(0 \rightarrow 1)} \\ \boldsymbol{r_9^n} &= \boldsymbol{\{B_9, p_{n3}, A_9\}(1 \rightarrow 0)} \end{aligned} \tag{5}$$

When the robot moves on the transition path, the direction is determined according to the serial number of the current and last working paths, as shown in Equation 6. The robot's direction is up when the *sign* is a positive number. When the *sign* is negative, the direction of the robot is down. The direction on the transition path is only judged by the j, and no other sensors are needed.

$$\boldsymbol{sign = j_c - j_l} \tag{6}$$

where j_c represents the serial number of the current working path, and j_l represents the serial number of the last working path.

2.1.2 The working rules of robots

The width of the working path A_jB_j in the field is relatively narrow, and it does not support operations such as moving side by side or turning; otherwise, the robots will crush or knock down the crops. Therefore, robots with different moving directions cannot simultaneously exist in the working path A_jB_j; otherwise, the robots will collide.

The robots move in a 'U' shape in the field (Hameed et al., 2013), and the working method is shown in the black path in Figure 1B. For example, R_n starts from point A_8, moves along the working path A_8B_8, reaches the end point B_8, and moves along B_8B_9 to A_9B_9. In addition, robots start from the garage, go to the field to work, and then return to the garage.

2.2 Collision and conflict scenarios of robots

During the research, we found that the possibility of a robot collision can be judged based on the robot's path type. When robots move on the same path type, we can determine whether the robots will collide by simply comparing their serial number of the working paths and directions. The path type and serial number are the same, indicating that they are in the same working path. At this time, if they are moving in opposite directions, it means that they are moving toward each other and a collision will inevitably occur. If they are moving in the same direction, as long as their speeds are the same, there will be no collision. Therefore, we divide the robot collision scenarios into four types based on the different positions of R_1 and R_2: 1) working path and different movement directions; 2) working path and same movement direction; 3) transition path and same movement direction; 4) transition path and different movement directions.

As shown in Figure 2, it describes the possibility of collision when the robot is working. The green line segments represent the working paths, and the two working robots are represented by R_1 and R_2; their target points are p_{1a} and p_{2b}.

2.2.1 Working path and different movement directions

Figure 2A describes that robot R_2 has been moving in the working path A_uB_u, and robot R_1 is about to enter the working path A_uB_u. Robots R_1 and R_2 move in opposite directions. According to section 2.1, there cannot be two robots with different moving directions in the same working path. If robot R_1 continues to enter the working path A_uB_u without changing its path, it will inevitably collide with R_2.

Figure 2B describes that R_1 and R_2 are about to enter the same working path A_uB_u. Robots R_1 and R_2 are moving in opposite directions. If the path of robot R_1 or R_2 is not changed, R_1 and R_2 will eventually collide together.

2.2.2 Working path and same movement direction

Figure 2C describes that robot R_2 is already moving in the working path A_uB_u, and robot R_1 is about to enter the working path A_uB_u. Figure 2D describes that the robots R_1 and R_2 are already moving in the working path A_uB_u. Robots R_1 and R_2 move in the same direction. According to Section 2.1, two arobots with the same moving direction can exist in the same working path. Since the speeds of the robots are the same, there will be a constant distance between robots moving in the same direction. However, when robot R_2 arrives at target point p_{2b}, the spraying operation takes time, and robot R_1 is still moving normally. R_1 needs to keep a safe distance from R_2 to stop moving and then wait for R_2 to complete its work. Otherwise, R_1 and R_2 will collide together.

2.2.3 Transition path and same movement direction

Figure 2E describes that R_1 and R_2 move in the same direction and go to different working paths. Since R_2 and R_1 move in the same direction and speed, they will not collide. However, when R_2 goes to the working path A_sB_s where the target point p_{2b} is located, it will turn around, which takes a certain amount of time. If R_1 does not stop moving and waits for R_2 to complete its turn, R_1 and R_2 will collide.

2.2.4 Transition path and different movement directions

Figure 2F describes that R_1 and R_2 move in different directions and go to different working paths. When R_1 goes to the target point p_{1a}, R_2 goes to the target point p_{2b} from the opposite direction. If the path of R_1 or R_2 is not changed, they will collide at a particular moment.

3 Approach

3.1 Global map generation method

In order to get the serial number j of working path and the robot's moving direction, the geographic information is needed. Use straight lines A_jB_j and the endpoints A_j and B_j to generate a field map. Represent this map as a point-line map (*plm*). Use straight line A_1B_1 as the baseline to generate other parallel line segment. As described in Algorithm 1, according to the straight-line equation of A_1B_1 and the distance D between each working path, the straight-line equations of all A_jB_j and the coordinates of the endpoints A_j and B_j are obtained. Finally, all working paths A_jB_j and endpoints A_j and B_j are stored in the *plm*.

Input: the coordinates (x_{A_1}, y_{A_1}), and (x_{B_1}, y_{B_1}) of the two endpoints A_1 and B_1
Output: *plm*

1: Initialize $k_{A_1B_1}$ and $b_{A_1B_1}$

2: **for** $j = 2, 3, \ldots, N^*$ **do**

3: $\quad k_{A_jB_j} = k_{A_1B_1}$

4: $\quad b_{A_jB_j} = b_{A_1B_1} + (j-1)D\sqrt{k_{A_1B_1}^2 + 1}$

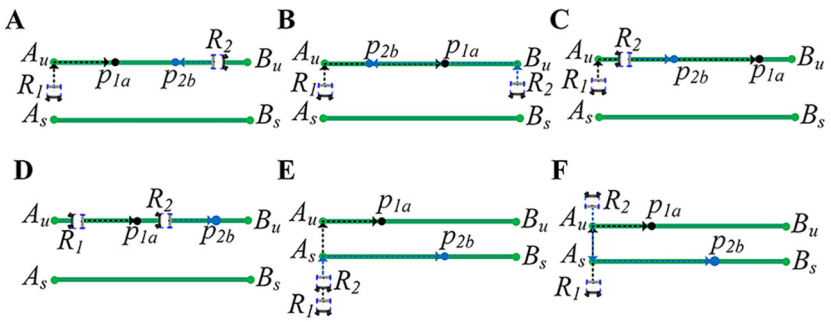

FIGURE 2
Types of robot conflicts. **(A, B)** Working path and different movement directions. **(C, D)** Working path and same movement direction. **(E)** Transition path and same movement direction. **(F)** Transition path and different movement directions.

5: //Update straight-line equation of A_jB_j

6: $y_{A_jB_j} = k_{A_jB_j}x_{A_jB_j} + b_{A_jB_j}$

7: //Update straight-line equations of A_1A_j and B_1B_j

8: $k_{A_1A_j}, k_{B_1B_j} = -1/k_{A_1B_1}$

9: $y_{A_1A_j}, y_{B_1B_j} = -x_{A_1A_j}/k_{A_jB_j} + b_{A_1A_j}, -x_{B_1B_j}/k_{A_jB_j} + b_{B_1B_j}$

10: $b_{A_1A_j}, b_{B_1B_j} = y_{A_1A_j} + x_{A_1A_j}/k_{A_jB_j}, y_{B_1B_j} + x_{B_1B_j}/k_{A_jB_j}$

11: //Update coordinates of A_j and B_j

12: $A_j(x_{A_j}, y_{A_j}) \leftarrow (b_{A_1A_j} - b_{A_jB_j})/(k_{A_jB_j} - k_{A_1A_j}), k_{A_1A_j}x_{A_j} + b_{A_1A_j}$

13: $B_j(x_{B_j}, y_{B_j}) \leftarrow (b_{B_1B_j} - b_{A_jB_j})/(k_{A_jB_j} - k_{B_1B_j}), k_{B_1B_j}x_{B_j} + b_{B_1B_j}$

14: **end for**

15: $plm \leftarrow \{A_jB_j, A_j, B_j | j \varepsilon N^*\}$

16: **return** plm

Algorithm 1. The plm generation method.

3.1.1 Calculating the equation of the working path

As shown in Figure 1, take A_1B_1 as the baseline, and make j parallel line segments A_1B_1, A_2B_2,…,. Suppose the slope of line segment A_1B_1 is $k_{A_1B_1}$, and the intercept is $b_{A_1B_1}$, then the straight-line equation of line segment A_1B_1 is expressed as:

$$y_{A_1B_1} = k_{A_1B_1}x_{A_1B_1} + b_{A_1B_1} \tag{7}$$

The coordinates (x_{A_1}, y_{A_1}), and (x_{B_1}, y_{B_1}) of the two endpoints and B_1 of A_1B_1 are obtained through the positioning device. Then the slope $k_{A_1B_1}$ and intercept $b_{A_1B_1}$ of A_1B_1 are expressed as:

$$k_{A_1B_1} = (y_{B_1} - y_{A_1})/(x_{B_1} - x_{A_1}) \tag{8}$$

$$b_{A_1B_1} = y_{A_1} - k_{A_1B_1}x_{A_1} \tag{9}$$

Since the straight line A_1B_1 is parallel to the straight line A_2B_2, the slope $k_{A_1B_1}$ of the straight line A_1B_1 is equal to the slope $k_{A_2B_2}$ of the straight line A_2B_2. According to the distance formula between two parallel lines, the intercept $b_{A_2B_2}$ of the straight line A_2B_2 is expressed as:

$$b_{A_2B_2} = b_{A_1B_1} + D\sqrt{k_{A_1B_1}^2 + 1}, b_{A_2B_2} > b_{A_1B_1} \tag{10}$$

According to Equation 10, the straight-line equation of A_2B_2 is expressed as:

$$y_{A_2B_2} = k_{A_1B_1}x_{A_2B_2} + b_{A_1B_1} + D\sqrt{k_{A_1B_1}^2 + 1} \tag{11}$$

The distance from straight-line segment A_3B_3 to straight-line segment A_1B_1 is 2 D. Then according to Equation 10, the intercept $b_{A_3B_3}$ of the straight line A_3B_3 is expressed as:

$$b_{A_3B_3} = b_{A_1B_1} + 2D\sqrt{k_{A_1B_1}^2 + 1}, \ b_{A_3B_3} > b_{A_1B_1} \tag{12}$$

According to Equation 12, the straight-line equation of A_3B_3 is expressed as:

$$\begin{aligned} y_{A_3B_3} &= k_{A_3B_3}x_{A_3B_3} + b_{A_3B_3} \\ &= k_{A_1B_1}x_{A_3B_3} + b_{A_1B_1} + 2D\sqrt{k_{A_1B_1}^2 + 1} \end{aligned} \tag{13}$$

According to Equations 10-13, the straight-line equation of A_jB_j is expressed as:

$$y_{A_jB_j} = k_{A_1B_1}x_{A_jB_j} + b_{A_1B_1} + (j-1)D\sqrt{k_{A_1B_1}^2 + 1}, j \in N^* \tag{14}$$

According to the coordinates of the endpoints A_1 and B_1, the straight-line equation of A_1B_1 is obtained, and then the straight-line equations of all other working paths are obtained according to D.

3.1.2 Calculating endpoint coordinates

According to the straight-line equation of A_jB_j obtained from Equation 14, determine the coordinates of the endpoints A_j and B_j. Make perpendicular lines A_1A_m and B_1B_m from point A_1 and point B_1 to line segment A_mB_m $(1 < m \leq j)$, and the foot points are A_m and B_m. Let the slopes of line segments A_1A_m and B_1B_m be $k_{A_1A_m}$ and $k_{B_1B_m}$, and the intercepts be $b_{A_1A_m}$ and $b_{B_1B_m}$, respectively. Then the straight-line equations of A_1A_m and B_1B_m are expressed as,

$$\begin{cases} y_{A_1A_m} = k_{A_1A_m}x_{A_1A_m} + b_{A_1A_m} \\ y_{B_1B_m} = k_{B_1B_m}x_{B_1B_m} + b_{B_1B_m} \end{cases}, k_{A_1A_m} = k_{B_1B_m} = \frac{-1}{k_{A_1B_1}} \tag{15}$$

Substitute the coordinates (x_{A_1}, y_{A_1}) of point A_1 into the straight-line equation of A_1A_m to obtain the intercept $b_{A_1A_m}$ of A_1A_m, as shown in Equation 16.

$$b_{A_1A_m} = y_{A_1} + \frac{x_{A_1}}{k_{A_1B_1}} \tag{16}$$

The straight-line equation of A_1A_m can be obtained according to the intercept $b_{A_1A_m}$ and slope $k_{A_1A_m}$. Then combine the linear equations of A_1A_m and A_mB_m to obtain the coordinates of the foot point A_m as (x_{A_m}, y_{A_m}), as shown in Equations 17, 18.

$$x_{A_m} = (b_{A_1A_m} - b_{A_mB_m})/(k_{A_mB_m} - k_{A_1A_m}) \tag{17}$$

$$y_{A_m} = k_{A_mB_m}x_{A_m} + b_{A_mB_m} \tag{18}$$

The coordinate (x_{B_m}, y_{B_m}) of the foot point B_m is represented as,

$$x_{B_m} = (b_{B_1B_m} - b_{A_mB_m})/(k_{A_mB_m} - k_{B_1B_m}) \tag{19}$$

$$y_{B_m} = k_{A_mB_m}x_{B_m} + b_{A_mB_m} \tag{20}$$

Finally, according to Equations 14, 17-20 and the coordinates of the endpoints A_1 and B_1, the coordinates (x_{A_m}, y_{A_m}) and (x_{B_m}, y_{B_m}) of all other endpoints can be obtained.

3.1.3 Fusion of endpoints and lines

The straight-line equation of A_jB_j and the coordinates of endpoints A_j and B_j are obtained from Sections 3.1.1 and 3.1.2, where the straight-line equation of A_jB_j includes information such as slope $k_{A_jB_j}$, intercept $b_{A_jB_j}$, and serial number j. Use *plm* to represent endpoints A_j, B_j, and working path A_jB_j, as shown in Equation 21.

$$\boldsymbol{plm} = \left\{\boldsymbol{A_j}, \boldsymbol{B_j}, \boldsymbol{A_jB_j} | \boldsymbol{j} \in \boldsymbol{N^*}\right\} \tag{21}$$

The *plm* provides detailed field information for the robot, supporting the robot to obtain its own direction, serial number of working paths, and path.

3.2 Robots global path generation method

When a robot detects a work conflict, it needs to re-plan the path for the low-priority robot. In order to quickly plan new paths, we first use *plm* and target points to generate the local path of each working path. Then, all local paths are merged into the global path of the robot, as shown in Algorithm 2. When the robot needs to adjust its path, it only needs to change the order and direction of the local path. Figure 3A shows all target points of the robot. Figure 3B shows global path and all local paths of the robot.

Input: the target point p_n of the robot R_n, plm

Output: global path r^n

1: **for** $i = 1, 2, \ldots, N^*$ **do**

2: **for** $j = 1, 2, \ldots, N^*$ **do**

3: $d_{p_{ni}A_jB_j} \leftarrow$ the distance from point p_{ni} to working path A_jB_j

4: **end for**

5: //Get the j and target point when the distance is the minimum value

6: $j_{d_{min}}$, $p_{nd_{min}} \leftarrow \min\{ d_{p_{ni}A_jB_j} \}$

7: $r^n_{j_{d_{min}}} \leftarrow \left\{A_{j_{d_{min}}}, p_{nd_{min}}, B_{j_{d_{min}}}\right\}(0 \rightarrow 1)$

8: //Add local path to global path

9: $r^n \leftarrow r^n_{j_{d_{min}}}$

10: **end for**

11: **for** r^n_j **in** r^n **do**

12: //Merge local paths with the same j

13: $r^n_j \leftarrow \{A_j, p_{n1}, p_{n2}, \ldots, p_{nv}, B_j\}(0 \rightarrow 1)$

14: **end for**

15: Update global path r^n according to the modified r^n_j

16: **for** r^n_j **in** r^n **do**

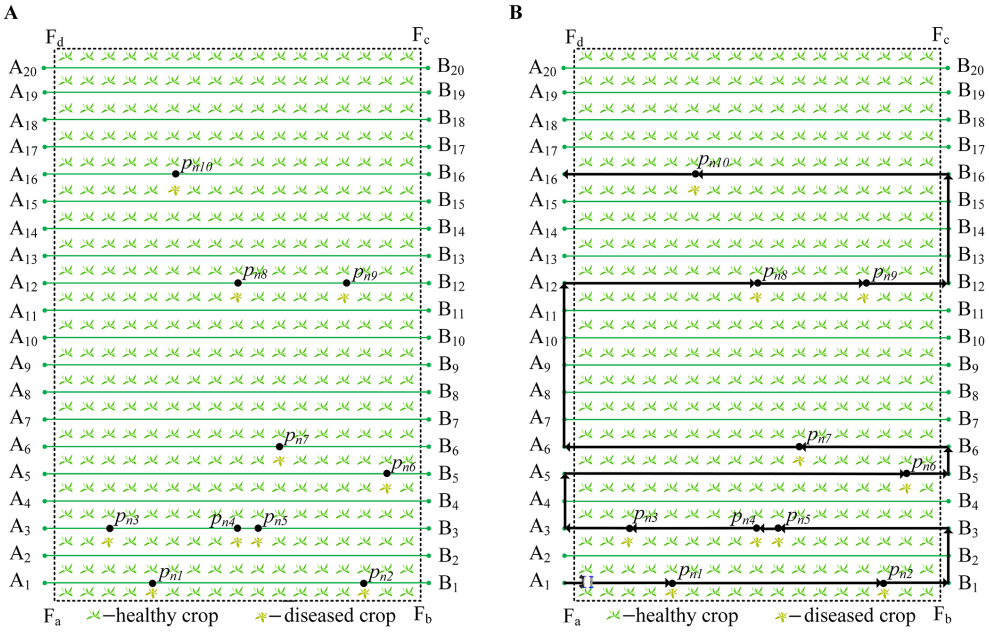

FIGURE 3
Target point and global path. **(A)** Target point. **(B)** Global path.

17: $d_{p_{nv}A_j}$ ← the distance from point p_{nv} to working path A_j

18: //Sort the target points $\{p_{n1}, p_{n2}, \ldots, p_{nv}\}$ according to the size of $d_{p_{nv}A_j}$

19: $r_j^n \leftarrow \{A_j, p_{n1}, p_{n2}, \ldots, p_{nv}, B_j\}(0 \rightarrow 1)$, $d_{p_{n1}A_j} < d_{p_{n2}A_j} < \cdots < d_{p_{nv}A_j}$, $0 < v \le i$

20: **end for**

21: Update global path r^n according to the modified r_j^n

22: //According to the requirements of direction $(0 \rightarrow 1)(1 \rightarrow 0)(0 \rightarrow 1)\cdots$, adjust the r^n

23: $r^n \leftarrow \left\{r_1^n(0 \rightarrow 1), r_2^n(1 \rightarrow 0), r_3^n(0 \rightarrow 1), \ldots, r_j^n(1 \rightarrow 0)\right\}$

24: **return** r^n

Algorithm 2. Robots global path generation method.

3.2.1 The local path of the robot

According to the *plm* and target point $p_n = \{p_{n1}, p_{n2}, \ldots, p_{ni}\}$ of the robot R_n, plan the local path r_j^n of the robot. Let the coordinates of p_{ni} be $(x_{p_{ni}}, y_{p_{ni}})$, and calculate the distance $d_{p_{ni}A_jB_j}$ from point p_{ni} to straight line A_jB_j according to Equation 22.

$$\boldsymbol{d_{p_{ni}A_jB_j} = \frac{\left|k_{A_jB_j} - y_{p_{ni}} + b_{A_jB_j}\right|}{\sqrt{1+k_{A_jB_j}^2}}, A_jB_j \in plm, i \in N^*} \quad (22)$$

Let $j_{d_{min}}$ represent the serial number of the working path when $d_{p_{ni}A_jB_j}$ takes the minimum value, as shown in Equation 23. When $d_{p_{ni}A_jB_j}$ takes the minimum value, the target point p_{ni} is in the working path

$A_{j_{d_{min}}}B_{j_{d_{min}}}$.

$$\boldsymbol{j_{d_{min}} = \left\{j \middle| min\left\{d_{p_{ni}A_jB_j} \middle| i \in N^*, A_jB_j \in plm\right\}\right\}} \quad (23)$$

The endpoints $A_{j_{d_{min}}}$ and $B_{j_{d_{min}}}$ of the working path $A_{j_{d_{min}}}B_{j_{d_{min}}}$ and point p_{ni} constitute the local path $r_{j_{d_{min}}}^n$, as shown in Equation 24. Put p_{ni} between points $A_{j_{d_{min}}}$ and $B_{j_{d_{min}}}$ because the robot starts from the endpoint $A_{j_{d_{min}}}$ or $B_{j_{d_{min}}}$ and then passes through the target point p_{ni}.

$$\begin{aligned} \boldsymbol{r_{j_{d_{min}}}^n} &= \left\{\boldsymbol{A_{j_{d_{min}}}, p_{ni}, B_{j_{d_{min}}}}\right\} \\ \boldsymbol{r_{j_{d_{min}}}^n} &= \left\{\boldsymbol{B_{j_{d_{min}}}, p_{ni}, A_{j_{d_{min}}}}\right\} \end{aligned} \quad (24)$$

Calculate the distance $d_{p_{nv}A_j}(0 < v \le i)$ from point p_{ni} to A_j according to Equation 25 when two or more target points are in the same working path.

$$\boldsymbol{d_{p_{nv}A_j} = \sqrt{\left(x_{p_{nv}} - x_{A_j}\right)^2 + \left(y_{p_{nv}} - y_{A_j}\right)^2}, (0 < v \le i), A_jB_j \in plm} \quad (25)$$

Sort the target points $\{p_{n1}, p_{n2}, \ldots, p_{nv}\}$ according to the size of $d_{p_{nv}A_j}$, and then update the local path r_j^n, as shown in Equation 26.

$$\begin{aligned} \boldsymbol{r_j^n} &= \left\{\boldsymbol{A_j, p_{n1}, p_{n2}, \ldots, p_{nv}, B_j}\right\}, \boldsymbol{d_{p_{n1}A_j} < d_{p_{n2}A_j} < \ldots < d_{p_{nv}A_j}} \\ \boldsymbol{r_j^n} &= \left\{\boldsymbol{B_j, p_{n1}, p_{n2}, \ldots, p_{nv}, A_j}\right\}, \boldsymbol{d_{p_{n1}A_j} > d_{p_{n2}A_j} > \ldots > d_{p_{nv}A_j}} \end{aligned} \quad (26)$$

In Figure 3A, p_{n1} and p_{n2} are in the working path A_1B_1. Since $d_{p_{n1}A_1} < d_{p_{n2}A_1}$, the path $r_1^n = \{A_1, p_{n1}, p_{n2}, B_1\}$ is obtained.

According to the target point p_n, a series of local paths r_1^n, r_2^n, r_3^n,…, r_j^n are obtained. For example, Figure 3A contains target points $\{p_{n1}, p_{n2}, \ldots, p_{n10}\}$, and six groups of local paths are obtained, as shown in Equation 27.

$$\begin{aligned} \boldsymbol{r_1^n} &= \boldsymbol{\{A_1, p_{n1}, p_{n2}, B_1\}(0 \rightarrow 1)} \\ \boldsymbol{r_3^n} &= \boldsymbol{\{A_3, p_{n3}, p_{n4}, p_{n5}, B_3\}(0 \rightarrow 1)} \\ \boldsymbol{r_5^n} &= \boldsymbol{\{A_5, p_{n6}, B_5\}(0 \rightarrow 1)} \\ \boldsymbol{r_6^n} &= \boldsymbol{\{A_6, p_{n7}, B_6\}(0 \rightarrow 1)} \\ \boldsymbol{r_{12}^n} &= \boldsymbol{\{A_{12}, p_{n8}, p_{n9}, B_{12}\}\ (0 \rightarrow 1)} \\ \boldsymbol{r_{16}^n} &= \boldsymbol{\{A_{16}, p_{n10}, B_{16}\}(0 \rightarrow 1)} \end{aligned} \quad (27)$$

3.2.2 Robots global path generation method

According to the local path r_j^n of the robot, it is known that the serial number of the working path is j. Sort the local paths r_j^n according to j and then get the path $r^n = \left\{r_1^n, r_2^n, r_3^n, \ldots, r_j^n\right\}$. For example, the path sorting in Figure 3A results in $r^n = \{r_1^n, r_3^n, r_5^n, r_6^n, r_{12}^n, r_{16}^n\}$.

However, the sorted r^n cannot be directly used as a robot's global path because the connection between local paths is not continuous. For example, r_1^n to r_3^n in Equation 27 is from point B_1 to A_3 and finally to B_3, which does not conform to the 'U' shape path of the robot. The robot cannot go directly from B_1 to A_3 but should go from B_1 to B_3, then to A_3. Therefore, it is necessary to adjust r_3^n to $\{B_3, ., A_3\}$, and at the same time, reorder the target points inside r_3^n according to Equation 25 to obtain the updated $r_3^n = \{B_3, p_{n5}, p_{n4}, p_{n3}, A_3\}(1 \rightarrow 0)$. Similarly, update r_5^n, r_6^n, r_{12}^n, r_{16}^n according to this rule, and finally update the six groups of local paths in Equation 27 to get Equation 28.

$$\begin{aligned} \boldsymbol{r_1^n} &= \boldsymbol{\{A_1, p_{n1}, p_{n2}, B_1\}(0 \rightarrow 1)} \\ \boldsymbol{r_3^n} &= \boldsymbol{\{B_3, p_{n5}, p_{n4}, p_{n3}, A_3\}(1 \rightarrow 0)} \\ \boldsymbol{r_5^n} &= \boldsymbol{\{A_5, p_{n6}, B_5\}(0 \rightarrow 1)} \\ \boldsymbol{r_6^n} &= \boldsymbol{\{B_6, p_{n7}, A_6\}(1 \rightarrow 0)} \\ \boldsymbol{r_{12}^n} &= \boldsymbol{\{A_{12}, p_{n8}, p_{n9}, B_{12}\}(0 \rightarrow 1)} \\ \boldsymbol{r_{16}^n} &= \boldsymbol{\{B_{16}, p_{n10}, A_{16}\}(1 \rightarrow 0)} \end{aligned} \quad (28)$$

Therefore, to ensure that the local paths are continuous, the direction between the local paths must satisfy $(0 \rightarrow 1)\ (1 \rightarrow 0)\ (0 \rightarrow 1)\cdots$ or $(1 \rightarrow 0)\ \ (0 \rightarrow 1)\ \ (1 \rightarrow 0)\cdots$. r^n satisfies Equation 29 or 30.

$$\boldsymbol{r^n = \left\{r_1^n(0 \rightarrow 1), r_2^n(1 \rightarrow 0), r_3^n(0 \rightarrow 1), \ldots, r_j^n(1 \rightarrow 0)\right\}} \quad (29)$$

10.3389/fpls.2024.1393541

$$r^n = \left\{ r_1^n(1 \to 0), r_2^n(0 \to 1), r_3^n(1 \to 0), \ldots, r_j^n(0 \to 1) \right\} \quad (30)$$

Combining the six groups of local paths in Equation 28, the global path r^n is obtained, as shown in Equation 31. Finally, according to the target point $p_n = \{p_{n1}, p_{n2}, \ldots, p_{n10}\}$ of the robot R_n in Figure 3A, plan the black path r^n in Figure 3B.

$$r^n = \{r_1^n(0 \to 1), r_3^n(1 \to 0), r_5^n(0 \to 1), r_6^n(1 \to 0), r_{12}^n(0 \to 1), r_{16}^n(1 \to 0)\} \quad (31)$$

3.3 Itinerary table with all global information

Based on the *plm* and its own position, the robot can calculate the direction of the robot, the number of the current working path, and the type of path. Combined with the global path and priority, the robot can learn all global information. When robot R_n learns its own information, it needs to send its global information to other robots so that other robots can understand the status of R_n, thereby judging the relationship between robots and predicting the possibility of collision. We designed an itinerary table (as shown in Table 1) that contains the robot's serial number, priority, path type, moving direction, real-time position, target position, current serial number of the working path and last serial number of the working path to facilitate the robot's sending and receive. When robot R_n is started, it generates an itinerary table H_{R_n}.` H represents the itinerary table of a group of working robots, and H_{R_n} represents the itinerary table of robot R_n, as shown in Equation 32.

$$\boldsymbol{H} = \left\{ \boldsymbol{H}_{\boldsymbol{R_n}} | \boldsymbol{n} \in \boldsymbol{N}^* \right\} \quad (32)$$

H_O represents other itinerary tables except for H_{R_n}, as shown in Equation 33.

$$\boldsymbol{H_O} = \boldsymbol{H} - \boldsymbol{H_{R_n}} \quad (33)$$

Robot R_n constantly updates the data in the itinerary table while working and broadcasts its H_{R_n} to other robots R_O. At the same time, robot R_n receives H_O and makes decisions.

TABLE 1 The itinerary table of robot R_n.

Itinerary table	
serial number of the robot	0, 1, 2, 3, …, *n*
priority	0, 1, 2, 3, …
path type	working or transition path
moving direction	$0 \to 1$, $1 \to 0$ or *up*, *down*
real-time position	P_{R_n}
target position	P_n
current serial number of the working path	j_c
last serial number of the working path	j_l

3.4 Multi-robot collision avoidance method

Based on the point-line map, global path and itinerary table, the robot can understand the global information of the field and the status of other robots. With this information, the robot can make reasonable decisions and avoid collisions with other robots. The multi-robot collision avoidance algorithm is shown in Algorithm 3. We calculate the direction of the robot R_n in different path type through Equations 5, 6. Then obtain the moving direction of other robots through the received itinerary table H_O. Finally, we can compare whether the robots have the same direction.

Input: r^n, H_{R_n}

1: **while** true **do**

2: Calculate the directions of robots according to Equations 5, 6.

3: **if** the robots move in the same direction **then**

4: $d_{R_nR_o} \leftarrow$ the distance from robot R_n to R_O

5: **if** $d_{R_nR_o} < 1$ **then**

6: The robot at the rear stops moving

7: **else**

8: The robot keeps moving

9: **end if**

10: **else**

11: //According to the local path r_j^n, query the path type

12: If the robot path points from A_j to B_j or B_j to A_j, the path type is working path.

13: If the robot path points from A_s to A_u or B_s to B_u, the path type is transition path.

14: **if** two or more robots have the same path type **then**

15: **if** path type is working path **then**

16: $j_c \leftarrow$ get the serial number of the working path based on the local path r_j^n

17: **if** j_c is the same **then**

18: //Adjust low-priority robot paths

19: $r^n = \left\{ r_j^n | j \in N^* \right\} \rightarrow r^n = \left\{ r_j^n, r_s^n | j \in (N^* - s) \right\}$

20: Then, the robot continues to move along the new path

21: **else**

22: The robot continues to move along the global path

23: **end if**

24: **else if** path type is transition path **then**

25: $d_{R_n R_o}$ ← the distance from robot R_n to R_O

26: **if** $d_{R_n R_o} < 1$ **then**

27: //Add obstacle avoidance paths to the r^n of low-priority robot.

28: $r^n = \left\{ r_j^n | j \in N^* \right\} \rightarrow r^n = \{..., r_s^n, r_t^n, r_u^n, ... | t > j, s \neq u, 0 < s \leq j, 0 < u \leq j\}$

29: Then, the robot continues to move along the new path

30: **else**

31: The robot continues to move along the global path

32: **end if**

33: **end if**

34: **else**

35: The robot continues to move along the global path

36: **end if**

37: **end if**

38: **if** P_{R_n} is the last point in the target point P_n **then**

39: **break**

40: **end if**

41: **end while**

Algorithm 3. Multi-robot collision avoidance method.

If the direction is the same, query the real-time position of the robot with the same direction in the itinerary table H_O, and calculate the distance $d_{R_n R_o}$ between R_n and R_O. When $d_{R_n R_o} \geq 1$, R_n and R_O maintain the current moving state. When $d_{R_n R_o} < 1$, the robot at the rear stops moving, and when $d_{R_n R_o} \geq 1$, the robot at the rear resumes moving.

But if the directions are different, the robot must make different decisions depending on the path type. We determine the path type of the robot based on its local path. If the robot path is from A_j to B_j or B_j to A_j, the path type of the robot is a working path. If the robot's path is from A_s to A_u or B_s to B_u, the path type of the robot is a transition path.

3.4.1 On the working path

If the path types of multiple robots are working path, we query the current serial number of the working path of the robot through the local path r_j^n.If the serial number j_c is different, R_n and R_O are not in the same working path, and there is no possibility of conflict. Robots continue to move according to the global path and do not need to do anything.

If the serial number j_c is the same, R_n and R_O are in the same working path, and conflicts or collisions will occur. There are two conflict cases: The first case is that one robot is moving in the working path, and another robot is moving from the transition path to the working path; The second is that all robots are entering the working path from the transition path.

For the first case, for example, when R_O is already moving in the working path A_sB_s, R_n is moving from A_sA_u to A_sB_s. Since they are going in different directions, they must collide. We do not consider the priorities of the robots at this time, directly let the robot R_n give up the current working path, and then re-plan the global path. In order not to hinder the robot's work and quickly generate the remaining path of the robot, we move the current conflicting local path r_s^n to the end of the global path, as shown in Equation 34. Then, using the content of Section 3.2.2, according to the direction continuity between local paths, modify the target point order of the local path, and then update r^n. The robot R_n continues to move according to the updated r^n.

$$\boldsymbol{r^n = \left\{ r_j^n | j \in N^* \right\} \rightarrow r^n = \left\{ r_j^n, r_s^n | j \in (N^* - s) \right\}} \quad (34)$$

For the second case, for example, both R_O and R_n are moving on the transition path and have not yet reached the working path. At this time, consider the priority of R_O and R_n, assuming that the priority of R_n is lower than R_O. The robot R_n with low priority abandons the current working path A_sB_s and re-plans the path. The remaining path re-planning rules are the same as in the first case.

3.4.2 On the transition path

If multiple robots are on a transition path, we first query the real-time positions of the robots on the same path type. Then, calculate the distance $d_{R_n R_o}$ between the robots. When $d_{R_n R_o} > 1$, robots maintain the current moving state. When $d_{R_n R_o} \leq 1$, the robot with low priority avoids the one with high priority. Since the width of the transition path is generally relatively wide, multiple robots can be accommodated simultaneously in the lateral direction. We can add some extra paths to avoid robot collisions.

In Figure 4A, robot R_1 goes from point A_s to P_{11}, robot R_2 goes from point A_u to P_{21}. It is obvious that R_1 and R_2 have different

directions. R1 and R2 are close together and are about to collide. Assuming that robot R_1 has a lower priority than robot R_2. The low-priority robot translates the current local path a certain distance to the left or right to avoid collision. For example, in Figure 4B, robot R_1 translates the local path $P_{R_n}A_u$ to P_aP_b to avoid collision with R_2. After translating the local path, the path P_aP_b is updated to $\{P_{R_n}, P_a, P_b, A_u\}$ according to the continuity of the path. We set the new path to r_t^1, as shown in Equation 35.

$$\boldsymbol{r_t^1} = \left\{\boldsymbol{P_{R_n}}, \boldsymbol{P_a}, \boldsymbol{P_b}, \boldsymbol{A_u}\right\}, \boldsymbol{t > j} \tag{35}$$

The conflict occurs on the transition path A_sA_u, and A_sA_u is located between the local paths r_s^1 and r_u^1, so insert r_t^1 between r_s^1 and r_u^1, as shown in Equation 36. The moving direction of the robot on the transition path A_sA_u is *up* or *down*, and it is $0 \rightarrow 1$ or $1 \rightarrow 0$ on the working paths A_sB_s and A_uB_u, so insert r_t^1 between r_s^1 and r_u^1, and the direction between other local paths in r^1 still satisfies $(0 \rightarrow 1)$ $(1 \rightarrow 0)$ $(0 \rightarrow 1)$ $\cdots$ or $(1 \rightarrow 0)$ $(0 \rightarrow 1)$ $(1 \rightarrow 0$ $)$ $\cdots$. Therefore, we only need to simply insert the local obstacle avoidance path r_t^1 between r_s^1 and r_u^1 without doing other operations.

$$\boldsymbol{r^1} = \left\{\ldots, \boldsymbol{r_s^1}, \boldsymbol{r_t^1}, \boldsymbol{r_u^1}, \ldots\right\}, \boldsymbol{s \neq u, 0 < s \leq j, 0 < u \leq j} \tag{36}$$

When a new robot is detected, the low-priority robot continues to translate the local path left or right. In Figure 4C, robot R_3 is detected on path P_aP_b after R_1 passes through point P_a, and robot R_1 plans local avoidance path $r_{t_1}^1$, as shown in Equation 37.

$$\boldsymbol{r_{t_1}^1} = \left\{\boldsymbol{p'_{R_n}}, \boldsymbol{P_c}, \boldsymbol{P_d}, \boldsymbol{P_b}, \boldsymbol{A_u}\right\} \tag{37}$$

Update r_t^1 according to the local avoidance path $r_{t_1}^1$, as shown in Equation 38. Then insert the updated r_t^1 into the global path r^1.

$$\boldsymbol{r_t^1} = \left\{\boldsymbol{P_{R_n}}, \boldsymbol{P_a}, \boldsymbol{p'_{R_n}}, \boldsymbol{P_c}, \boldsymbol{P_d}, \boldsymbol{P_b}, \boldsymbol{A_u}\right\} \tag{38}$$

The updated local avoidance path is shown in Figure 4D. If R_1 conflicts with other robots again, plan the local obstacle avoidance path according to this rule and then update r_t^1 and r^1.

4 Experiment and result

4.1 Simulate initial conditions

In order to verify our proposed multi-robot collision avoidance method, we used Gazebo software to design the robot working environment. The working environment includes field, crops, roads, and four working robots, as shown in Figure 5. Each working robot contains a medicine storage barrel, a medicine spraying device, communication device, positioning device and controller.

In order to facilitate the description of the working process of the robot, a GUI display interface is designed to display the garage, field, crops, and robot, as shown in Figure 6. Set up a group of working robots $R = \{R_1, R_2, R_3, R_4\}$, with priorities of 0, 1, 2, and 3, respectively. Set the field length to 20 meters and width to 16 meters. The field has 20 rows of crops ($1 \leq j < 20$), and the distance between each row of crops is 1 meter ($D = 1$). The coordinate system is established with point O as the origin, and the coordinates of the two endpoints A_1 and B_1 of the baseline A_1B_1 are set to (0, -9) and (16, -9), respectively.

4.2 Generating a point-line map

We first generate all working path A_jB_j and endpoints A_j and B_j based on the baseline A_1B_1. Calculate the slope $k_{A_1B_1}$ and intercept $b_{A_1B_1}$ of the working path A_1B_1 based on points A_1 and B_1 and the straight-line equation of A_1B_1, as shown in Equation 39.

$$\boldsymbol{y_{A_1B_1}} = \boldsymbol{-9 (0 \leq x_{A_1B_1} \leq 16), k_{A_1B_1} = 0, b_{A_1B_1} = -9} \tag{39}$$

According to Equations 14, 39, calculate the straight-line equation of the working path A_jB_j, as shown in Equation 40.

$$\boldsymbol{y_{A_jB_j}} = \boldsymbol{-9 + (j-1) * D, k_{A_jB_j} = 0, b_{A_jB_j} = -9 + (j-1) * D} \tag{40}$$

Then according to Equations 17-20, the coordinates of points A_j and B_j are obtained as $(0, -9 + (j-1)*D)$, $(16, -9 + (j-1)*D)$. Finally, according to Equations 39, 40, the *plm* is obtained as shown in Equation 41.

$$\begin{aligned} \boldsymbol{plm} = \{&\boldsymbol{A_j(0, -9 + (j-1) * D), B_j(16, -9 + (j-1) * D)}, \\ &\boldsymbol{y_{A_jB_j} = -9 + (j-1) * D | 1 \leq j < 20, D = 1}\} \end{aligned} \tag{41}$$

4.3 Generating the global path

We randomly generated the coordinates of 35 diseased crops, and then used them as the target points of the robot (Meshram et al., 2022). The target point is denoted as $P = \{p_1, p_2, p_3, \ldots, p_{35}\}$. Then, the coordinates P are randomly assigned to R_1, R_2, R_3, and R_4, as shown in Figure 7. Randomly divide the targets into four groups: P_1 =

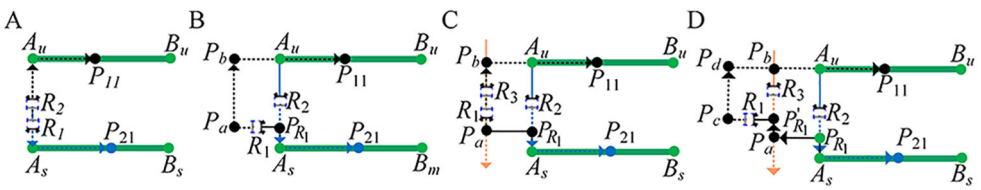

FIGURE 4
Conflict detection for robots on transition paths. **(A)** R_1 and R_2 meet at the transition path. **(B)** R_1 plans a new path. **(C)** R_1 meets R_3 on the new path. **(D)** R_1 plans a new path again.

10.3389/fpls.2024.1393541

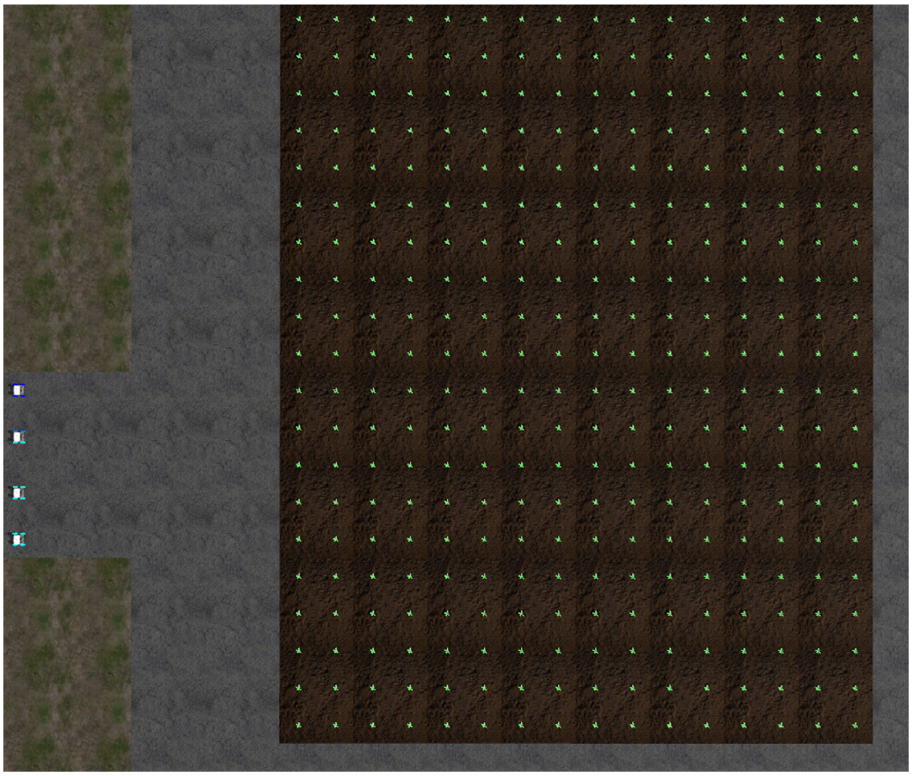

FIGURE 5
The robot's simulation work environment.

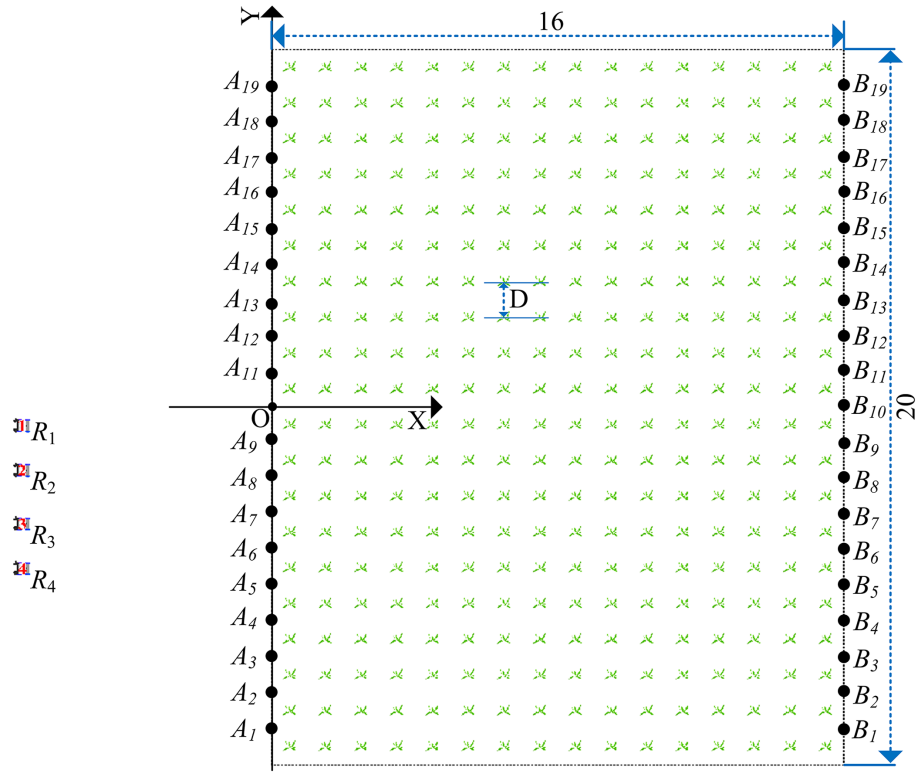

FIGURE 6
The initial diagram of the robot's working environment.

$\{p_{11}, p_{12}, \ldots, p_{18}\}$ is represented by orange, $P_2 = \{p_{29}, p_{210}, \ldots, p_{216}\}$ is represented by black, $P_3 = \{p_{317}, p_{318}, \ldots, p_{324}\}$ is represented by blue, and $P_4 = \{p_{424}, p_{425}, \ldots, p_{435}\}$ is represented by purple.

According to Section 3.2, use the target points and map to generate the robots' global path, as shown in Equations 42-45.

$$\boldsymbol{r^1} = \left\{\boldsymbol{r^1_{15}}(\boldsymbol{0 \rightarrow 1}), \boldsymbol{r^1_9}(\boldsymbol{1 \rightarrow 0}), \boldsymbol{r^1_6}(\boldsymbol{0 \rightarrow 1}), \boldsymbol{r^1_4}(\boldsymbol{1 \rightarrow 0}), \boldsymbol{r^1_3}(\boldsymbol{0 \rightarrow 1})\right\} \tag{42}$$

where $r^1_{15} = \{A_{15}, p_{11}, B_{15}\}$, $r^1_9 = \{B_9, p_{12}, A_9\}$, $r^1_6 = \{A_6, p_{17}, p_{15}, B_6\}$, $r^1_4 = \{B_4, p_{18}, p_{14}, A_4\}$, $r^1_3 = \{A_3, p_{13}, p_{16}, B_3\}$.

$$\boldsymbol{r^2} = \left\{\boldsymbol{r^2_2}(\boldsymbol{0 \rightarrow 1}), \boldsymbol{r^2_{11}}(\boldsymbol{1 \rightarrow 0}), \boldsymbol{r^2_{14}}(\boldsymbol{0 \rightarrow 1}), \boldsymbol{r^2_{15}}(\boldsymbol{1 \rightarrow 0}), \boldsymbol{r^2_{17}}(\boldsymbol{0 \rightarrow 1}), \boldsymbol{r^2_{19}}(\boldsymbol{1 \rightarrow 0})\right\} \tag{43}$$

where $r^2_2 = \{A_2, p_{29}, B_2\}$, $r^2_{11} = \{B_{11}, p_{213}, A_{11}\}$, $r^2_{14} = \{A_{14}, p_{211}, p_{210}, B_{14}\}$, $r^2_{15} = \{B_{15}, p_{215}, A_{15}\}$, $r^2_{17} = \{A_{17}, p_{214}, B_{17}\}$, $r^2_{19} = \{B_{19}, p_{212}, p_{216}, A_{19}\}$.

$$\begin{aligned}\boldsymbol{r^3} = \{&\boldsymbol{r^3_1}(\boldsymbol{0 \rightarrow 1}), \boldsymbol{r^3_2}(\boldsymbol{1 \rightarrow 0}), \boldsymbol{r^3_4}(\boldsymbol{0 \rightarrow 1}), \boldsymbol{r^3_7}(\boldsymbol{1 \rightarrow 0}),\\ &\boldsymbol{r^3_{10}}(\boldsymbol{0 \rightarrow 1}), \boldsymbol{r^3_{11}}(\boldsymbol{1 \rightarrow 0}), \boldsymbol{r^3_{12}}(\boldsymbol{0 \rightarrow 1})\}\end{aligned} \tag{44}$$

where $r^3_1 = \{A_1, p_{322}, B_1\}$, $r^3_2 = \{B_2, p_{318}, A_2\}$, $r^3_4 = \{A_4, p_{317}, B_4\}$, $r^3_7 = \{B_7, p_{320}, A_7\}$, $r^3_{10} = \{A_{10}, p_{324}, p_{319}, B_{10}\}$, $r^3_{11} = \{B_{11}, p_{321}, A_{11}\}$, $r^3_{12} = \{A_{12}, p_{323}, B_{12}\}$.

$$\begin{aligned}\boldsymbol{r^4} = \{&\boldsymbol{r^4_3}(\boldsymbol{0 \rightarrow 1}), \boldsymbol{r^4_5}(\boldsymbol{1 \rightarrow 0}), \boldsymbol{r^4_8}(\boldsymbol{0 \rightarrow 1}), \boldsymbol{r^4_{10}}(\boldsymbol{1 \rightarrow 0}),\\ &\boldsymbol{r^4_{13}}(\boldsymbol{0 \rightarrow 1}), \boldsymbol{r^4_{16}}(\boldsymbol{1 \rightarrow 0}), \boldsymbol{r^4_{17}}(\boldsymbol{0 \rightarrow 1}), \boldsymbol{r^4_{18}}(\boldsymbol{1 \rightarrow 0})\}\end{aligned} \tag{45}$$

where $r^4_3 = \{A_3, p_{426}, B_3\}$, $r^4_5 = \{B_5, p_{433}, p_{432}, A_5\}$, $r^4_8 = \{A_8, p_{425}, B_8\}$, $r^4_{10} = \{B_{10}, p_{428}, A_{10}\}$, $r^4_{13} = \{A_{13}, p_{427}, B_{13}\}$, $r^4_{16} = \{B_{16}, p_{434}, A_{16}\}$, $r^4_{17} = \{A_{17}, p_{435}, p_{431}, B_{17}\}$, $r^4_{18} = \{B_{18}, p_{429}, A_{18}\}$.

Figures 8A–D show the global paths of R_1, R_2, R_3, and R_4.

4.4 Simulation experiment of multi-robot collision avoidance method

After planning the global path, the robots start spraying operations and simultaneously create an itinerary table H, as shown in Table 1. During the operation, the robot determines the position, path, and direction of moving of other robots by querying their itinerary tables. Then, the robots detect four types of collision and conflict according to section 2.2. Next, the multi-robot collision avoidance method is simulated considering four types of collisions and conflicts.

4.4.1 Working path and different movement directions

When the serial number j_c of the working path is the same, and the moving direction is different, there are two conflict cases: The first case is that one robot is moving in the working path, and another robot is moving from the transition path to the working path; The second is that all robots are entering the working path from the transition path.

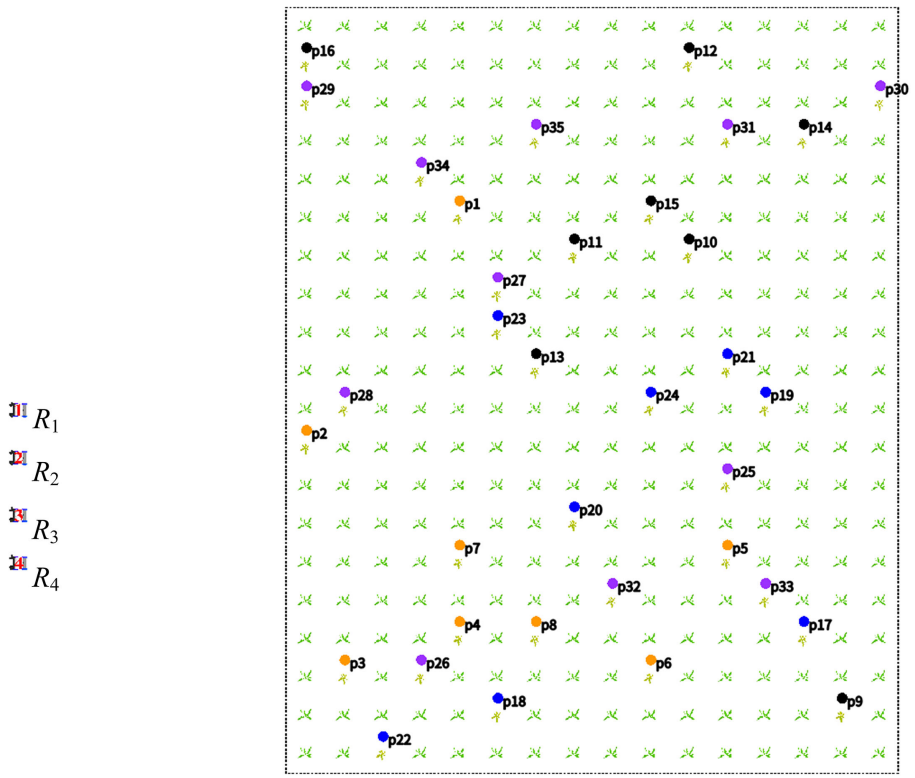

FIGURE 7
Random distribution of target points.

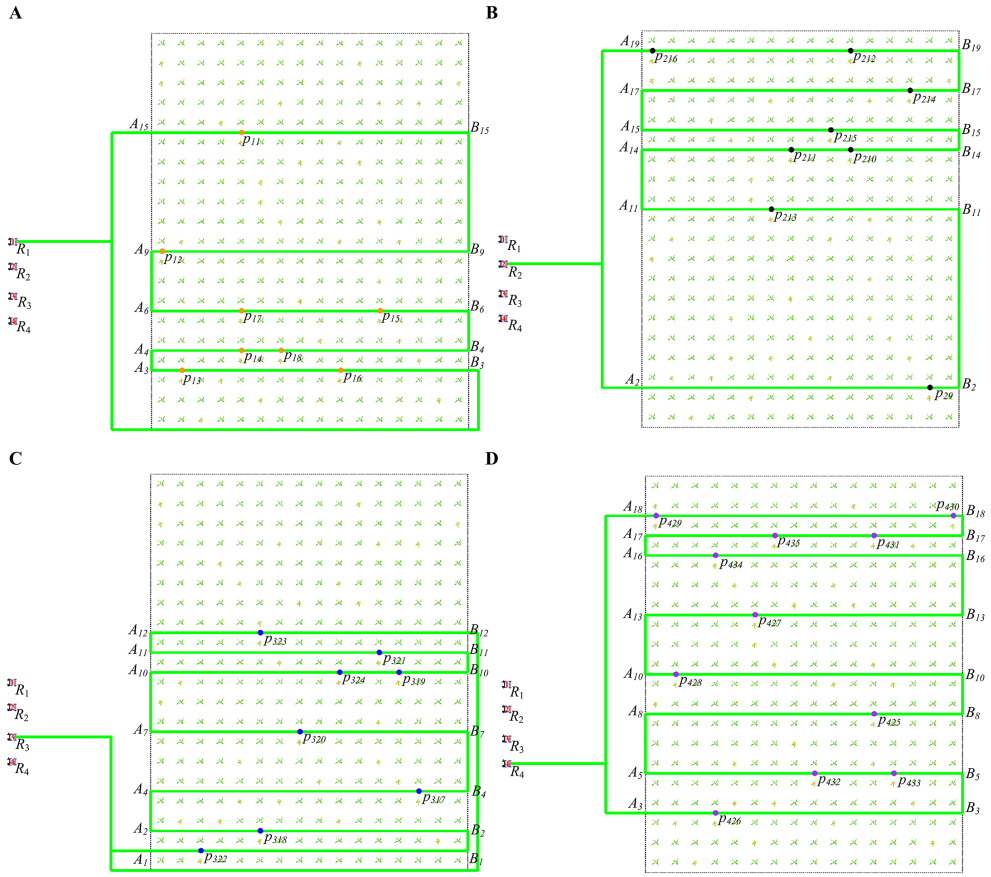

FIGURE 8
The global path of four robots. **(A)** The global path of R_1. **(B)** The global path of R_2. **(C)** The global path of R_3. **(D)** The global path of R_4.

Figure 9 shows the first case (The green path represents the path that has not been passed, while the black path represents the path that has already been passed). Robot R_2 is moving in the working path A_2B_2, and robot R_3 is moving from the transition path B_1B_2 to the working path A_2B_2. Figure 9A shows that at 121 seconds, robot R_3 is about to leave its current working path A_1B_1 and move to the next working path A_2B_2 for work. Figure 9B shows that at 125s, robot R_3 moves to the transition path B_1B_2 and detects a robot R_2 (At 25 seconds, R_2 works on working path A_2B_2, as shown in Figure 9C) with a different direction of moving in the working path A_2B_2 according to the itinerary table.

In order to avoid collision, robot R_2 will abandon the current local path r_2^3 and then re-plan the remaining path. We follow the method proposed in Section 3 and first move the current local path r_2^3 to the end of the global path r^3, as shown in Equation 46. But in the global path r^3, r_1^3 and r_4^3 do not satisfy the continuity of direction, and $(0 \rightarrow 1)$ to $(0 \rightarrow 1)$ appear, so we need to change the direction of the remaining path r_4^3, r_7^3, r_{10}^3, r_{11}^3, r_{12}^3 and r_2^3.

$$\begin{aligned} \boldsymbol{r^3}=\{&\boldsymbol{r_1^3(0\rightarrow1)}, \boldsymbol{r_4^3(0\rightarrow1)}, \boldsymbol{r_7^3(1\rightarrow0)}, \boldsymbol{r_{10}^3(0\rightarrow1)},\\ &\boldsymbol{r_{11}^3(1\rightarrow0)}, \boldsymbol{r_{12}^3(0\rightarrow1)}, \boldsymbol{r_2^3(1\rightarrow0)}\} \end{aligned} \tag{46}$$

where $r_1^3 = \{A_1, p_{322}, B_1\}$, $r_4^3 = \{A_4, p_{317}, B_4\}$, $r_7^3 = \{B_7, p_{320}, A_7\}$, $r_{10}^3 = \{A_{10}, p_{324}, p_{319}, B_{10}\}$, $r_{11}^3 = \{B_{11}, p_{321}, A_{11}\}$, $r_{12}^3 = \{A_{12}, p_{323}, B_{12}\}$, $r_2^3 = \{B_2, p_{318}, A_2\}$

We change the direction of the local path by modifying the order of the points of the local path, as shown in Equation 47. The robot continues driving along the updated path r^3. Figure 9D shows that at 522 seconds, robot R_3 moves to the last working path, and the black path in the figure clearly shows the updated new path.

$$\begin{aligned} \boldsymbol{r^3} = \{&\boldsymbol{r_1^3(0\rightarrow1)}, \boldsymbol{r_4^3(1\rightarrow0)}, \boldsymbol{r_7^3(0\rightarrow1)}, \boldsymbol{r_{10}^3(1\rightarrow0)},\\ &\boldsymbol{r_{11}^3(0\rightarrow1)}, \boldsymbol{r_{12}^3(1\rightarrow0)}, \boldsymbol{r_2^3(0\rightarrow1)}\} \end{aligned} \tag{47}$$

where $r_1^3 = \{A_1, p_{322}, B_1\}$, $r_4^3 = \{B_4, p_{317}, A_4\}$, $r_7^3 = \{A_7, p_{320}, B_7\}$, $r_{10}^3 = \{B_{10}, p_{319}, p_{324}, A_{10}\}$, $r_{11}^3 = \{A_{11}, p_{321}, B_{11}\}$, $r_{12}^3 = \{B_{12}, p_{323}, A_{12}\}$, $r_2^3 = \{A_2, p_{318}, B_2\}$.

Figure 10 shows the second case. Robots R_1 and R_2 move to the same working path $A_{15}B_{15}$ from other paths. Figure 10A shows that at 10 seconds, robot R_2 is about to leave its current working path $A_{14}B_{14}$ and move to the next working path $A_{15}B_{15}$. Figure 10B shows that at 13 seconds, robot R_2 reaches the transition path B_{14} B_{15} and begins to move toward the working path $A_{15}B_{15}$. At this time, it is detected that another robot R_1 (Figure 10C shows the operational status of R_1 at 13 seconds) is also moving toward the working path $A_{15}B_{15}$ according to the itinerary table.

Since R_1 has a higher priority than R_2, in order to avoid collision, R_2 abandons the current local path r_{15}^2 and re-plans a new path. The global path update method of r^2 is the same as

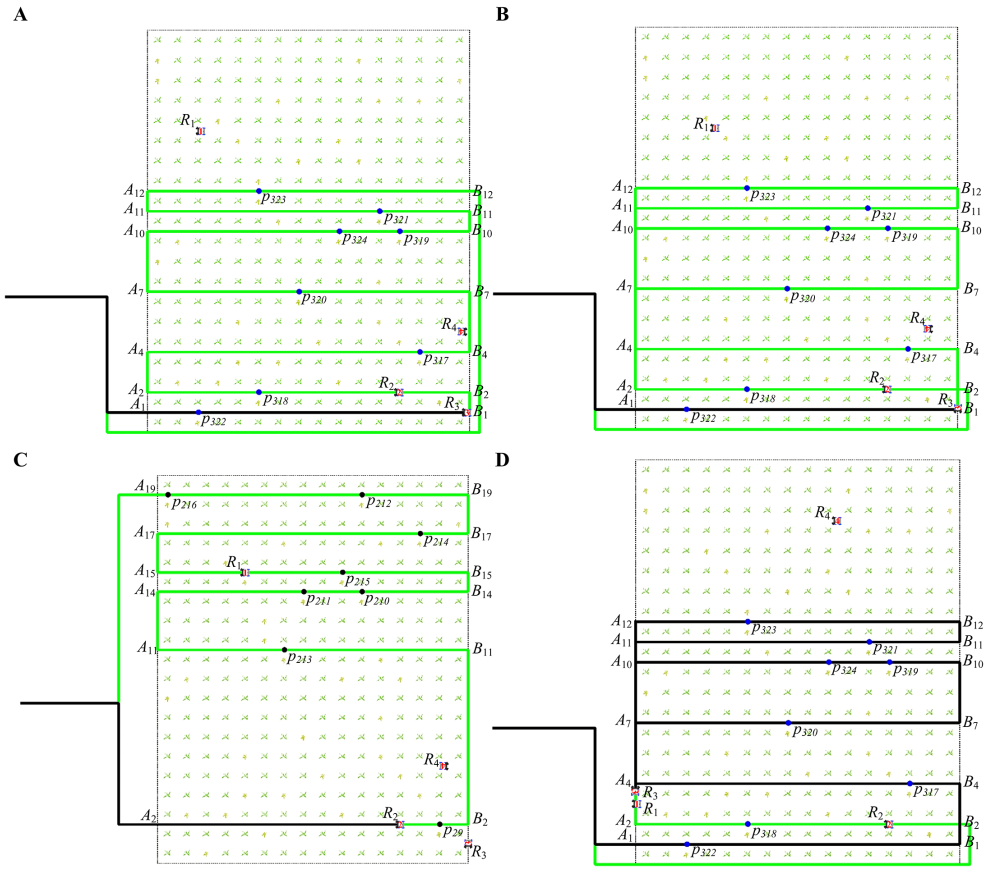

FIGURE 9
The paths of robots moving in different directions on the same working path at different times (the first case). **(A)** The path of robot R_3 at 121 seconds. **(B)** The path of robot R_3 at 125 seconds. **(C)** The path of robot R_2 at 125 seconds. **(D)** The path of robot R_3 at 522 seconds.

Equations 46, 47. The updated path r^2 of R_2 is as shown in Equation 48. Figure 10D shows that at 173 seconds, the robot R_2 moves to the last working path, and the black path in the figure clearly shows the updated new path.

$$\boldsymbol{r^2} = \left\{\boldsymbol{r_2^2(0\rightarrow 1)}, \boldsymbol{r_{11}^2(1\rightarrow 0)}, \boldsymbol{r_{14}^2(0\rightarrow 1)}, \boldsymbol{r_{17}^2(1\rightarrow 0)}, \boldsymbol{r_{19}^2(0\rightarrow 1)}, \boldsymbol{r_{15}^2(1\rightarrow 0)}\right\} \tag{48}$$

where $r_2^2 = \{A_2, p_{29}, B_2\}$, $r_{11}^2 = \{B_{11}, p_{213}, A_{11}\}$, $r_{14}^2 = \{A_{14}, p_{211}, p_{210}, B_{14}\}$, $r_{17}^2 = \{B_{17}, p_{214}, A_{17}\}$, $r_{19}^2 = \{A_{19}, p_{216}, p_{212}, B_{19}\}$, $r_{15}^2 = \{B_{15}, p_{215}, A_{15}\}$.

4.4.2 Working path and same movement direction

When the robots have the same serial number j_c and the same moving direction, there are two cases: The robots maintain the current speed and move in the order of one after the other. The second is that when one of the robots stops working at the target point, the other robots at the back must wait for the front robot to complete the work.

Figure 11 shows the first case. Robots R_2 and R_4 maintain a certain speed and distance, and move in a front-to-back sequence. Figure 11A shows the states of robots R_2 and R_4 at 11 seconds. Figure 11B shows that at 31 seconds, robot R_2 and R_4 keep moving forward in this state.

Figure 12 shows the second case. Robot R_4 stops moving and performs the spraying operation after reaching the target point p_{431}. Robot R_2 stops moving, waiting for robot R_4 to complete the operation. Figure 12A shows that at 33 seconds, robot R_2 moves toward R_4 along the path. As depicted in Figure 12B, robot R_2 detects the halt of R_4 at 36 seconds, and R_2 maintains a safe distance from R_4 when it stops. Figure 12C shows that at 36 seconds, robot fpls.2024.1393541 stops at the target point p_{431}. Figure 12D shows that at 39 seconds, robot R_2 stops moving until R_4 completes the work.

4.4.3 Transition path and same movement direction

When the robot is on the transition path and moves in the same direction as other robots, there are two cases: One is that the robot keeps the current speed and follows the others. The other is that when one of the robots turns to the working path, the robots behind stop at a safe distance and wait for the turn to finish.

The first case is the same as the first case in 4.4.2. As long as the safe distance between robots is ensured, no other operations are performed.

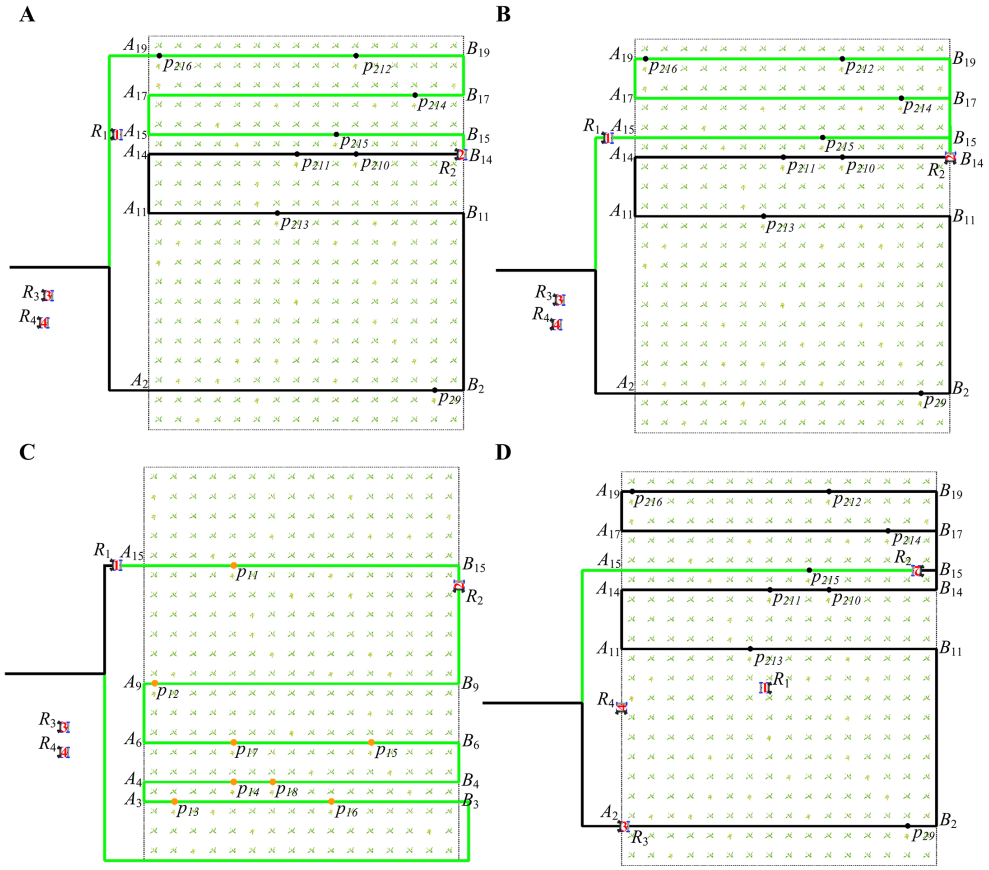

FIGURE 10
The paths of robots moving in different directions on the same working path at different times (the second case). **(A)** The path of robot R_2 at 10 seconds. **(B)** The path of robot R_2 at 13 seconds. **(C)** The path of robot R_1 at 13 seconds. **(D)** The path of robot R_2 at 173 seconds.

Figure 13 shows the second case, robots R_2 and R_3 are moving on the transition path. Figure 13A shows that at 11 seconds, robot R_2 detects that R_3 is moving from the working path to the transition path B_2B_{11}, and R_2 stops moving and waits for R_3 to complete the turn. Figure 13B shows that at 15 seconds, after R_3 completes its turn, R_2 and R_3 maintain a certain safe distance and move along the transition path B_2B_{11}. Figure 13C shows that at 22 seconds, robot R_2 maintains a safe distance from R_3 and moves in the same direction. Figure 13D shows that at 30 seconds, robot R_3 switches from the transition path B_2B_{11} to the working path A_7B_7, and robot R_2 stops at a safe distance, waiting for robot R_3 to finish the turn.

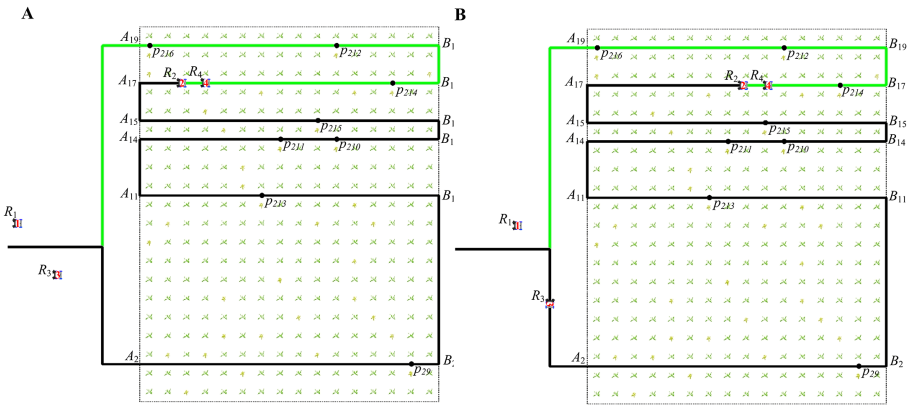

FIGURE 11
The paths of robots moving in the same direction on the same working path at different times (the first case). **(A)** The path of robot R_2 at 11 seconds. **(B)** The path of robot R_2 at 31 seconds.

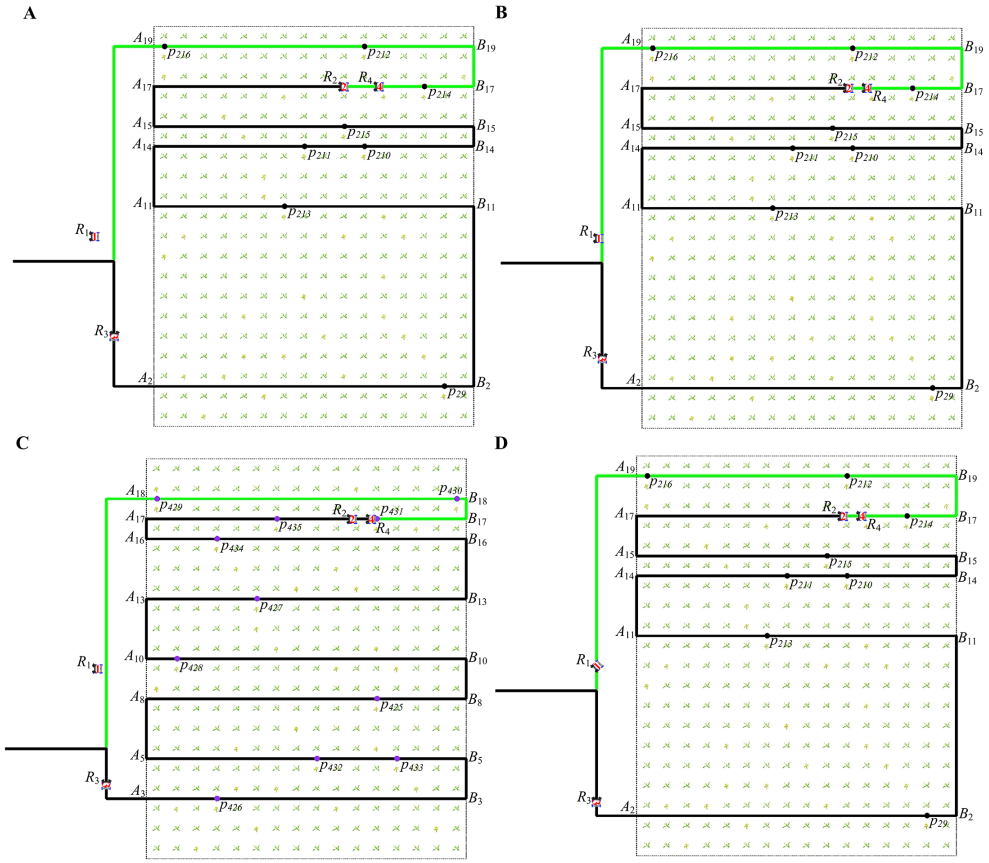

FIGURE 12
The paths of robots moving in the same direction on the same working path at different times (the second case). **(A)** The path of robot R_2 at 33 seconds. **(B)** The path of robot R_2 at 36 seconds. **(C)** The path of robot R_4 at 36 seconds. **(D)** The path of robot R_2 at 39 seconds.

4.4.4 Transition path and different movement directions

Figure 14 shows that robots R_1, R_2, and R_3 are moving on the transition path. The j_c and j_l of R_1 are 4 and 6 respectively. The j_c and j_l of R_2 are 11 and 2 respectively. The j_c and j_l of R_3 are 4 and 6 respectively. According to Equation 6, their directions on the transition path are down, up, and down respectively. The moving direction of R_2 is different from that of R_1 and R_3. Change the priority of R_1, R_2, and R_3 to 0, 2, and 1, respectively.

Figure 14A shows that at 31 seconds, robot R_2 detects that R_1 (Figure 14B shows the global path of R_1 at 31 seconds) is approaching from the opposite direction. Since the priority of R_2 is lower than that of R_1, R_2 plans the obstacle avoidance path according to Section 3.4.2. R_2 translates the current local path a certain distance to the right and adds a new obstacle avoidance path r_t^1, as shown in Equation 49.

$$\boldsymbol{r_t^1} = \left\{\boldsymbol{P_{R_2}}, \boldsymbol{P_a}, \boldsymbol{P_b}, \boldsymbol{B_{11}}\right\} \tag{49}$$

Then, we directly insert r_t^1 between r_2^2 and r_{11}^2 in r^2, as shown in Equation 50. R_2 moves according to updated r^2. Figure 14C shows the new path of R_2.

$$\boldsymbol{r^2} = \left\{\boldsymbol{r_2^2(0 \to 1)}, \boldsymbol{r_t^1}, \boldsymbol{r_{11}^2(1 \to 0)}, \boldsymbol{r_{14}^2(0 \to 1)}, \boldsymbol{r_{17}^2(1 \to 0)}, \boldsymbol{r_{19}^2(0 \to 1)}, \boldsymbol{r_{15}^2(1 \to 0)}\right\} \tag{50}$$

where $r_2^2 = \{A_2, p_{29}, B_2\}$, $r_t^1 = \{P_{R_2}, P_a, P_b, B_{11}\}$, $r_{11}^2 = \{B_{11}, p_{213}, A_{11}\}$, $r_{14}^2 = \{A_{14}, p_{211}, p_{210}, B_{14}\}$, $r_{15}^2 = \{B_{15}, p_{215}, A_{15}\}$, $r_{17}^2 = \{A_{17}, p_{214}, B_{17}\}$, $r_{19}^2 = \{B_{19}, p_{212}, p_{216}, A_{19}\}$.

R_2 detects a new robot R_3 (Figure 14D shows the global path of R_3 at 47 seconds) approaching from the opposite direction at 47 seconds in Figure 14C. Since R_2 has a lower priority than R_3, R_2 plans an obstacle avoidance path to avoid colliding with R_3. Just like the rules for avoiding R_1, move the current local path r_t^1 a certain distance to the right, and then update the local path r_t^1, as shown in Equation 51.

$$\boldsymbol{r_t^1} = \left\{\boldsymbol{P_{R_2}}, \boldsymbol{P_a}, \boldsymbol{P'_{R_2}}, \boldsymbol{P_c}, \boldsymbol{P_d}, \boldsymbol{B_{11}}\right\} \tag{51}$$

Then, we directly insert the updated r_t^1 between r_2^2 and r_{11}^2 in r^2, as shown in Equation 52. R_2 moves according to updated r^2. Figure 14E shows the new path of R_2 at 56 seconds.

$$\boldsymbol{r^2} = \left\{\boldsymbol{r_2^2(0 \to 1)}, \boldsymbol{r_t^1}, \boldsymbol{r_{11}^2(1 \to 0)}, \boldsymbol{r_{14}^2(0 \to 1)}, \boldsymbol{r_{17}^2(1 \to 0)}, \boldsymbol{r_{19}^2(0 \to 1)}, \boldsymbol{r_{15}^2(1 \to 0)}\right\} \tag{52}$$

where $r_t^1 = \left\{P_{R_2}, P_a, P'_{R_2}, P_c, P_d, B_{11}\right\}$.

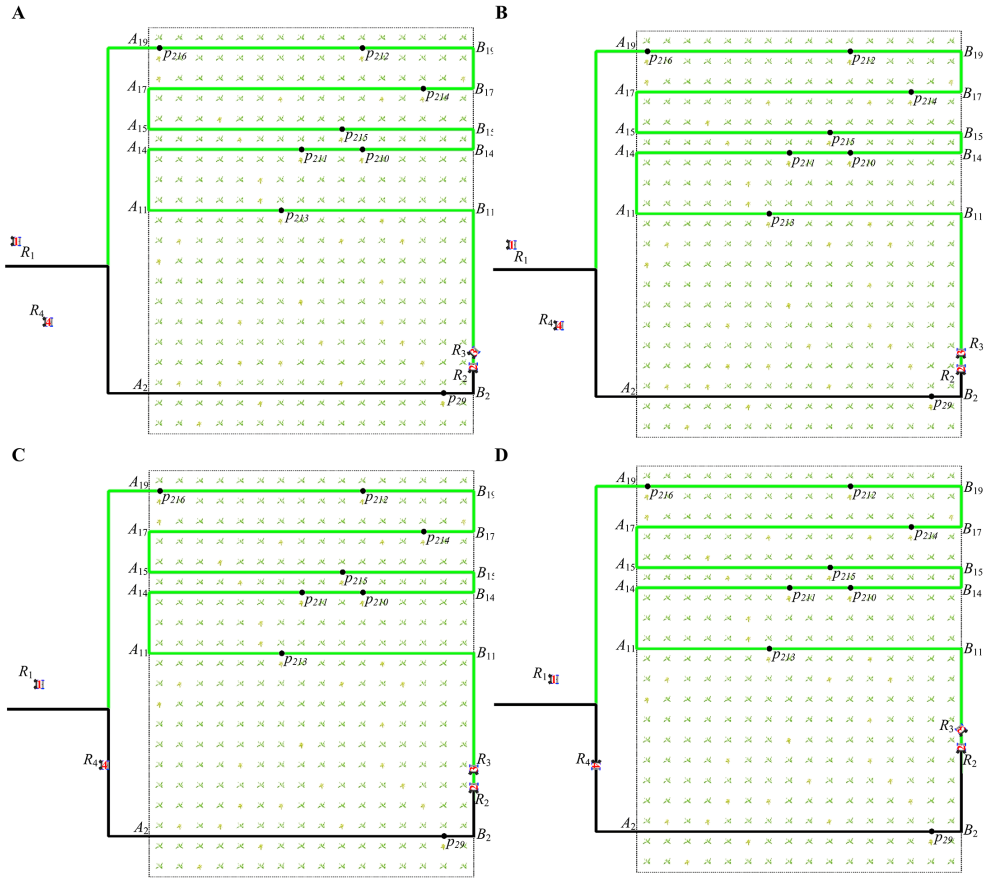

FIGURE 13
The paths of robots moving in the same direction on different working paths at different times. **(A)** The path of robot R_2 at 11 seconds. **(B)** The path of robot R_2 at 15 seconds. **(C)** The path of robot R_1 at 22 seconds. **(D)** The path of robot R_2 at 30 seconds.

4.5 Analysis of experimental results

4.5.1 Analysis of multi-robot collision avoidance results

We recorded the number of times the multi-robot system avoided collisions and the time spent executing tasks for various numbers of target points, as shown in Table 2. We randomly generated sets of 15, 25, 35, and 45 target points and conducted 10 experiments for each set, with target points randomly generated in each experiment. For the multi-robot system, we recorded the total number of collisions avoided and the total time spent on task execution for each robot. For a single robot, we recorded only the total time spent on the spraying task.

As shown in Table 2, under different numbers of target points, the time taken by the multi-robot system to complete the task is much shorter than that of a single robot. Specifically, for 15 target points, the multi-robot system avoids collisions 119 times in total, saving 55.6% of the time compared to a single robot. For 25 target points, the multi-robot system avoids collisions 130 times in total, saving 57.9% of the time compared to a single robot. For 35 target points, the multi-robot system avoids collisions 140 times in total, saving 48.9% of the time compared to a single robot. For 45 target points, the multi-robot system avoids collisions 145 times in total, saving 40.7% of the time compared to a single robot. The experimental results show that under the same number of target points, the multi-robot system can complete the spraying task while avoiding collisions or conflicts between robots, and the completion time can be reduced by more than 40%.

4.5.2 Comparison experiment with other methods

To evaluate our method, we compare it with the classic Reciprocal Velocity Obstacles (RVO) (Van den Berg et al., 2008) and Optimal Reciprocal Collision Avoidance (ORCA) (Niu et al., 2021) algorithms. In terms of time complexity, the time complexity of RVO and ORCA algorithms is $O(n^2)$. As shown in Algorithms 1–3, their time complexity are $O(n)$, $O(n^2)$, and $O(n)$, respectively. Therefore, the overall time complexity of our algorithm is $O(n^2)$.

Since the width of the working path in the field is relatively narrow, it cannot support two robots moving side by side. When two robots move toward each other on the working path (as shown in Figure 2B), if the low-priority robot performs an evasive action, it will inevitably collide with the crops. Therefore, we made some modifications to the RVO and ORCA algorithms. If they are detected to be moving in opposite directions on the working path, the low-priority robot stops and waits for the other robot to

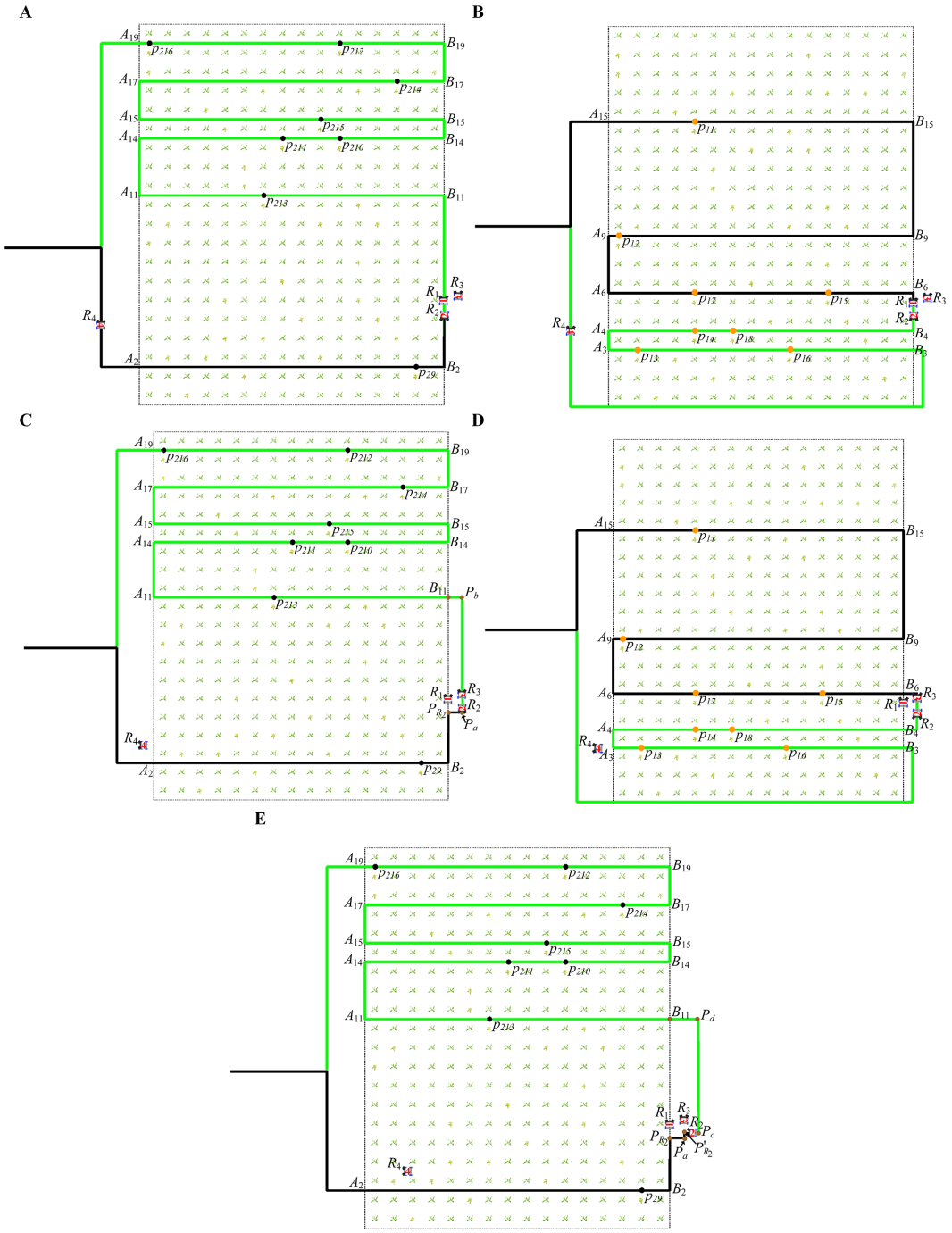

FIGURE 14
The paths of robots moving in different directions on different working paths at different times. **(A)** The path of robot R_2 at 31 seconds. **(B)** The path of robot R_1 at 31 seconds. **(C)** The path of robot R_2 at 47 seconds. **(D)** The path of robot R_3 at 47 seconds. **(E)** The path of robot R_2 at 56 seconds.

complete the work before continuing the work. Under the same settings, ten spraying tests with the same number and the same order are conducted for each collision avoidance method with different numbers of target points. The completion time mainly reflects the efficiency of different methods, and the number of collisions avoided reflects the performance from the side.

The experimental results are shown in Table 3, demonstrating that the proposed collision avoidance algorithm outperforms the RVO and ORCA algorithms. Specifically, for 15 target points, the completion time of our algorithm is reduced by 5.2% and 2.9% compared with RVO and ORCA, respectively. This is because when the number of target points is relatively small, the probability of collision or conflict between robots in a large field is relatively small, so the total completion time is not much different. For 25 target points, the completion time is reduced by 9.8% and 3.8% compared with the RVO and ORCA algorithms, respectively. For 35 target

TABLE 2 Experimental results (The total number of collisions avoided and the total time spent on the task for each robot).

Number of target points		15		25		35		45	
System	Robot	Time (s)	Number of collisions avoided	Time (s)	Number of collisions avoided	Time (s)	Number of collisions avoided	Time (s)	Number of collisions avoided
Multi-robot system	R_1	2562	27	3617	32	4473	37	4887	32
	R_2	2575	33	3282	42	4242	44	4972	40
	R_3	2782	35	3619	25	4468	29	5353	42
	R_4	2770	24	3509	31	4157	30	5157	31
Single robot system		6264	\	8589	\	8765	\	9024	\

points, the completion time is reduced by 14.9% and 8.2% compared with the RVO and ORCA algorithms, respectively. For 45 target points, the completion time is reduced by 16.8% and 9.5% compared with the RVO and ORCA algorithms, respectively. As the number of target points increases, the likelihood of collisions or conflicts between robots also increases. Our proposed algorithm demonstrates better performance, with a greater reduction in completion time compared to RVO and ORCA. In addition, the number of collision avoidance of our proposed algorithm is less than that of RVO and ORCA. This is because our collision avoidance algorithm can pre-judge potential collisions or conflicts on the working path, thereby reducing their occurrence. In summary, our proposed algorithm can complete the spraying task on large-scale fields in a timely and efficient manner, exhibiting high operational efficiency.

4.6 Summary

We designed a simulation environment for precise spraying of sweet potato fields, and then experimentally verified the collision avoidance method we proposed against four types of collisions or conflicts that may occur between multiple robots in the field. We demonstrate that our multi-robot collision avoidance method can be well deployed on robots even though they only have communication modules, positioning modules and low-cost control chips. We validate the collision avoidance strategy in various collision or conflict scenarios and show that our method can run robustly for long periods of time without collisions.

5 Conclusion

In this article, we propose a multi-robot collision avoidance method that only uses point-line maps and real-time robot positions to determine the relationship between robots, make reasonable decisions, and avoid collisions between robots. We evaluate the performance of our method through a series of comprehensive experiments and demonstrate that the collision avoidance method is simple and efficient in terms of success rate, safety, and navigation efficiency. For the current situation where only a single robot is used to complete the operation, the method we proposed can greatly improve the efficiency of farmland robot operations. At the same time, the method we proposed has extremely low requirements for robot hardware performance, which greatly reduces the cost of each robot, thereby reducing farmers' farmland management costs and labor intensity, and indirectly increasing farmers' economic income. It provides the theoretical basis and technical support for reducing the cost of multi-robot systems and accelerating the promotion and application of multi-robot systems in agriculture.

Our work is a first step toward reducing robot costs, avoiding robot collisions, and improving the efficiency of multi-agricultural robots. Although we are fully aware that as a local collision avoidance method, our approach cannot completely replace reinforcement learning-based multi-robot path planners when scheduling. Our future work will be how to extend our method to larger-scale robot systems at low cost and apply this method to real field environments to achieve a safe and efficient multi-robot collision avoidance method.

TABLE 3 Experimental results of different collision avoidance methods.

Number of target points	15		25		35		45	
Method	Time (s)	Number of collisions avoided	Time (s)	Number of collisions avoided	Time (s)	Number of collisions avoided	Time (s)	Number of collisions avoided
RVO	2935	142	4012	158	5255	192	6432	211
ORCA	2864	140	3761	152	4872	180	5917	185
Ours	2782	119	3619	130	4473	140	5353	145

Data availability statement

The original contributions presented in the study are included in the article/supplementary material. Further inquiries can be directed to the corresponding author.

Author contributions

KX: Writing – original draft, Writing – review & editing. JX: Supervision, Validation, Writing – review & editing. WS: Conceptualization, Investigation, Writing – review & editing. PX: Data curation, Methodology, Writing – review & editing. RY: Methodology, Supervision, Writing – review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This research was funded by the Key R&D Projects in Hainan Province (Grant No. ZDYF2023XDNY039) and National Talent Foundation Project of China (Grant No. T2019136).

Acknowledgments

The authors would like to thank their schools and colleges, as well as the funding of the project. All support and assistance are sincerely appreciated. Additionally, we sincerely appreciate the work of the editor and the reviewers of the present paper.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Alonso-Mora, J., Breitenmoser, A., Rufli, M., Beardsley, P., and Siegwart, R. (2013). "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed autonomous robotic systems: The 10th international symposium* (Berlin, Germany: Springer Berlin Heidelberg), 203–216.

Althoff, D., Kuffner, J. J., Wollherr, D., and Buss, M. (2012). Safety assessment of robot trajectories for navigation in uncertain and dynamic environments. *Autonomous Robots* 32, 285–302. doi: 10.1007/s10514-011-9257-9

Baltazar, A. R., Santos, F. N. D., Moreira, A. P., Valente, A., and Cunha, J. B. (2021). Smarter robotic sprayer system for precision agriculture. *Electronics* 10, 2061. doi: 10.3390/electronics10172061

Cai, C., Yang, C., Zhu, Q., and Liang, Y. (2007). "Collision avoidance in multi-robot systems," in *2007 International Conference on Mechatronics and Automation*. 2795–2800 (Harbin, China: IEEE).

Carbone, C., Garibaldi, O., and Kurt, Z. (2018). Swarm robotics as a solution to crops inspection for precision agriculture. *KnE Eng.* 3, 552–562. doi: 10.18502/keg.v3i1.1459

Chang, L., Shan, L., Jiang, C., and Dai, Y. (2021). Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. *Autonomous robots* 45, 51–76. doi: 10.1007/s10514-020-09947-4

Chen, Y. F., Everett, M., Liu, M., and How, J. P. (2017a). "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. (Vancouver, BC, Canada: IEEE), 1343–1350. doi: 10.1109/IROS.2017.8202312

Chen, Y. F., Liu, M., Everett, M., and How, J. P. (2017b). "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. (Singapore: IEEE), 285–292. doi: 10.1109/ICRA.2017.7989037

Cheng, P. D. C., Indri, M., Possieri, C., Sassano, M., and Sibona, F. (2023). Path planning in formation and collision avoidance for multi-agent systems. *Nonlinear Analysis: Hybrid Syst.* 47, 101293. doi: 10.1016/j.nahs.2022.101293

Claes, D., Hennes, D., Tuyls, K., and Meeussen, W. (2012). "Collision avoidance under bounded localization uncertainty," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. (Vilamoura-Algarve, Portugal: IEEE), 1192–1198. doi: 10.1109/IROS.2012.6386125

Claes, D., and Tuyls, K. (2018). Multi robot collision avoidance in a shared workspace. *Autonomous Robots* 42, 1749–1770. doi: 10.1007/s10514-018-9726-5

Danton, A., Roux, J. C., Dance, B., Cariou, C., and Lenain, R. (2020). "Development of a spraying robot for precision agriculture: An edge following approach," in *2020 IEEE Conference on Control Technology and Applications (CCTA)*. (Montreal, QC, Canada: IEEE), 267–272. doi: 10.1109/CCTA41146.2020.9206304

Fiorini, P., and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *Int. J. robotics Res.* 17, 760–772. doi: 10.1177/027836499801700706

Godoy, J., Karamouzas, I., Guy, S., and Gini, M. (2016). "Implicit coordination in crowded multi-agent navigation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30 (1). (Phoenix, Arizona, USA: AAAI Press). doi: 10.1609/aaai.v30i1.10131

Hameed, I. A., Bochtis, D., and Sørensen, C. A. (2013). An optimized field coverage planning approach for navigation of agricultural robots in fields involving obstacle areas. *Int. J. advanced robotic Syst.* 10, 231. doi: 10.5772/56248

Han, Y., Zhan, I. H., Zhao, W., Pan, J., Zhang, Z., Wang, Y., et al. (2022). Deep reinforcement learning for robot collision avoidance with self-state-attention and sensor fusion. *IEEE Robotics Automation Lett.* 7, 6886–6893. doi: 10.1109/LRA.2022.3178791

He, H., Kamburugamuve, S., Fox, G. C., and Zhao, W. (2016). Cloud based real-time multi-robot collision avoidance for swarm robotics. *Int. J. Grid Distributed Computing* 9, 339–358. doi: 10.14257/ijgdc

Hennes, D., Claes, D., Meeussen, W., and Tuyls, K. (2012). "Multi-robot collision avoidance with localization uncertainty," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. (Valencia, Spain: International Foundation for Autonomous Agents and Multiagent Systems), 147–154. doi: 10.5555/2343576.2343597

Hu, K., Chen, Z., Kang, H., and Tang, Y. (2024). 3D vision technologies for a self-developed structural external crack damage recognition robot. *Automation Construction* 159, 105262. doi: 10.1016/j.autcon.2023.105262

Huang, S., Zhang, H., and Huang, Z. (2022). Multi-UAV collision avoidance using multi-agent reinforcement learning with counterfactual credit assignment. *arXiv preprint arXiv:2204.08594*. doi: 10.48550/arXiv.2204.08594

Jawhar, I., Mohamed, N., Wu, J., and Al-Jaroodi, J. (2018). Networking of multi-robot systems: Architectures and requirements. *J. Sensor Actuator Networks* 7, 52. doi: 10.3390/jsan7040052

Keshmiri, S., and Payandeh, S. (2009). "A centralized framework to multi-robots formation control: Theory and application," in *International Workshop on Collaborative Agents, Research and Development* (Springer Berlin Heidelberg, Berlin, Heidelberg), 85–98.

Kim, J., and Son, H. I. (2020). A voronoi diagram-based workspace partition for weak cooperation of multi-robot system in orchard. *IEEE Access* 8, 20676–20686. doi: 10.1109/Access.6287639

Liu, L., Liu, Y., He, X., and Liu, W. (2022). Precision variable-rate spraying robot by using single 3D LIDAR in orchards. *Agronomy* 12, 2509. doi: 10.3390/agronomy12102509

Long, P., Liu, W., and Pan, J. (2017). Deep-learned collision avoidance policy for distributed multiagent navigation. *IEEE Robotics Automation Lett.* 2, 656–663. doi: 10.1109/LRA.2017.2651371

Matoui, F., Boussaid, B., Metoui, B., and Abdelkrim, M. N. (2020). Contribution to the path planning of a multi-robot system: centralized architecture. *Intelligent Service Robotics* 13, 147–158. doi: 10.1007/s11370-019-00302-w

Meshram, A. T., Vanalkar, A. V., Kalambe, K. B., and Badar, A. M. (2022). Pesticide spraying robot for precision agriculture: A categorical literature review and future trends. *J. Field Robotics* 39, 153–171. doi: 10.1002/rob.22043

Niu, H., Ma, C., and Han, P. (2021). Directional optimal reciprocal collision avoidance. *Robotics Autonomous Syst.* 136, 103705. doi: 10.1016/j.robot.2020.103705

Oberti, R., and Schmilovitch, Z. E. (2021). "Robotic spraying for precision crop protection," in *Innovation in agricultural robotics for precision agriculture: A roadmap for integrating robots in precision agriculture*. (Springer, Cham), 117–150. doi: 10.1007/978-3-030-77036-5_6

Parker, L. E., Rus, D., and Sukhatme, G. S. (2016). Multiple Mobile Robot Systems. In: B. Siciliano and O. Khatib (eds) *Springer Handbook of Robotics*. (Springer, Cham: Springer Handbooks), 1335–1384. doi: 10.1007/978-3-319-32552-1_53

Pfeiffer, M., Schaeuble, M., Nieto, J., Siegwart, R., and Cadena, C. (2017). "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *2017 ieee international conference on robotics and automation (icra)*. (Singapore: IEEE), 1527–1533. doi: 10.1109/ICRA.2017.7989182

Raibail, M., Rahman, A. H. A., AL-Anizy, G. J., Nasrudin, M. F., Nadzir, M. S. M., Noraini, N. M. R., et al. (2022). Decentralized multi-robot collision avoidance: A systematic review from 2015 to 2021. *Symmetry* 14, 610. doi: 10.3390/sym14030610

Schwartz, J. T., and Sharir, M. (1983). On the piano movers' problem: III. Coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers. *Int. J. Robotics Res.* 2, 46–75. doi: 10.1177/027836498300200304

Snape, J., Van Den Berg, J., Guy, S. J., and Manocha, D. (2009). "Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. (St. Louis, MO, USA: IEEE), 5917–5922. doi: 10.1109/IROS.2009.5354821

Sun, D., Kleiner, A., and Nebel, B. (2014). "Behavior-based multi-robot collision avoidance," in *2014 IEEE international conference on robotics and automation (ICRA)*. (Hong Kong, China: IEEE), 1668–1673. doi: 10.1109/ICRA.2014.6907075

Tang, Y., Qi, S., Zhu, L., Zhuo, X., Zhang, Y., and Meng, F. (2024). Obstacle avoidance motion in mobile robotics. *J. System Simulation* 36, 1–26. doi: 10.16182/j.issn1004731x.joss.23-1297E

Tang, S., Thomas, J., and Kumar, V. (2018). Hold or take optimal plan (hoop): A quadratic programming approach to multi-robot trajectory generation. *Int. J. Robotics Res.* 37, 1062–1084. doi: 10.1177/0278364917741532

Ünal, İ., Kabaş, Ö., Eceoğlu, O., and Moiceanu, G. (2023). Adaptive multi-robot communication system and collision avoidance algorithm for precision agriculture. *Appl. Sci.* 13, 8602. doi: 10.3390/app13158602

Van den Berg, J., Lin, M., and Manocha, D. (2008). "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE international conference on robotics and automation*. (Pasadena, CA, USA: IEEE), 1928–1935. doi: 10.1109/ROBOT.2008.4543489

Wang, D., Fan, T., Han, T., and Pan, J. (2020). A two-stage reinforcement learning approach for multi-UAV collision avoidance under imperfect sensing. *IEEE Robotics Automation Lett.* 5, 3098–3105. doi: 10.1109/LSP.2016.

Xiao, W., Yuan, L., He, L., Ran, T., Zhang, J., and Cui, J. (2022). Multigoal visual navigation with collision avoidance via deep reinforcement learning. *IEEE Trans. Instrumentation Measurement* 71, 1–9. doi: 10.1109/TIM.2022.3158384

Xiongkui, H. (2020). Research progress and developmental recommendations on precision spraying technology and equipment in China. *Smart Agric.* 2, 133. doi: 10.12133/j.smartag.2020.2.1.201907-SA002

Yu, J., and LaValle, S. M. (2016). Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Trans. Robotics* 32, 1163–1177. doi: 10.1109/TRO.2016.2593448

Zhang, D., Chen, C., and Zhang, G. (2024). "AGV path planning based on improved A-star algorithm," in *2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. (Chongqing, China: IEEE), Vol. 7, 1590–1595. doi: 10.1109/IAEAC59436.2024.10503919

Zhao, W., Wang, Y., Zhang, Z., and Wang, H. (2021). Multicriteria ship route planning method based on improved particle swarm optimization–genetic algorithm. *J. Mar. Sci. Eng.* 9, 357. doi: 10.3390/jmse9040357

Zhu, X., Yi, J., Ding, H., and He, L. (2020). Velocity obstacle based on vertical ellipse for multi-robot collision avoidance. *J. Intelligent Robotic Syst.* 99, 183–208. doi: 10.1007/s10846-019-01127-6