

Final Project Report

Project name

Research on cooperative path planning of dual agents based on large language model

Team members

Wu Xiyun 224040203, Sun Ruipeng 224040262

1. Project Background

In the current era of rapid technological development, the application of multi-agent path planning technology is increasingly widespread, and its research background is mainly reflected in the following three aspects:

The first is the urgent demand of application scenarios. In practical scenarios such as logistics warehouses, industrial workshops, and complex terrain exploration, intelligent vehicles carrying robotic arms are widely used to perform tasks such as transporting or picking up items. However, these environments are often full of obstacles, which poses high requirements for path planning.

The second is the limitation of the traditional single-agent transportation method. When performing the above tasks, the traditional single-agent transportation mode has gradually exposed many drawbacks, such as poor load-bearing capacity, low efficiency in obstacle avoidance path planning, difficulty in quickly finding the optimal path in complex environments, and limited obstacle avoidance ability, which easily leads to task failure or a significant reduction in efficiency.

The third is the advantages of multi-agent systems and the impetus of the development of large language model technology. With the continuous advancement of large model technology and the in-depth research on multi-robot collaboration, it has become possible to combine large models with path planning. Compared with single-agent systems, multi-agent systems have many significant advantages in task execution: On the one hand, multiple agents can handle different parts of the task simultaneously through division of labor and collaboration, greatly improving the efficiency and flexibility of task execution; On the other hand, they can adjust their path planning in real time through mutual communication and information sharing, enhancing their adaptability to complex environments and obstacle avoidance capabilities. In addition, multi-agent systems also have higher reliability and fault tolerance. Even if one agent fails, the others can still continue to complete the task. By combining large models with path planning, the path planning efficiency and decision-making ability of multi-agent systems can be further optimized. Meanwhile, using dual agents instead of single agents can better adapt to various complex environments and diverse task requirements, providing more efficient and reliable solutions for practical applications.

2. Research Objectives

In this study, we have set the following three specific research goals:

First, dual-agent collaboration for obstacle avoidance. We will construct a maze environment containing obstacles of different heights to simulate complex real scenarios. Two intelligent vehicles, Spark-T, equipped with mechanical arms are deployed in it, aiming to achieve the collaborative picking up of the same object by the two and successfully exit the maze while avoiding various obstacles. During this process, the focus is on studying how the dual agents can obtain real-time environmental information of surrounding obstacles, thereby adjusting their own movement trajectories and completing tasks in an efficient and precise collaborative manner.

Second, autonomous path identification and correction. We utilize the built-in camera of Spark-T and image processing technology to achieve autonomous recognition of the complete driving path under the condition that the current position and direction of the intelligent vehicle are known, and correct the errors of position and direction after each step of operation. We will enable the intelligent vehicle to accurately extract the position information of the drivable area and obstacles from the images captured by the camera through image processing methods, and transform the robot coordinate system perspective into the global world coordinate system perspective through mathematical function relationship conversion, thereby constructing a detailed map model. Subsequently, through the comparison of image angles and positions at each step, the positions and errors are corrected to ensure that the intelligent vehicle can move stably and accurately along the planned movable path, and at the same time, it has the ability to deal with unexpected situations on the path.

Third, integrate path planning with large language model technology. We will convert the map dot matrix scanned by the built-in camera of Spark-T into a 0,1 matrix and import it into the large language model (Deepseek,Kimi). Both large language models can handle it. The model will give the instruction for the next movement and return it to the robot through the program, thereby controlling the robot's next movement. The deep integration of path planning with advanced large language model technology aims to further optimize the performance of the path planning algorithm, improve the overall operational efficiency and reliability of the system, and enhance the adaptability of agents in complex and changeable environments.

3. Technical methods

In this section, we will introduce image processing in detail and the specific methods used to achieve the above three research goals.

3.1 Image Processing technology

In this section, we mainly introduce how to handle the pictures taken by the robot's built-in camera and the generated npy files, as well as how to identify obstacles based on the pictures.

We use one host to control two Spark-T robots. The commands for each step of operation are transmitted from one host to the two robots through sockets. On the two robots, we have respectively established two servers. It will listen to the instructions sent by the host and run. At each step of the car's operation, it will save a captured image of the current position and an npy file to the local robot. Among them, the "captured image of the current position" is captured by the built-in camera of the spark-T robot. The "npy file" is an $n \times 3$ matrix, with each row containing three numbers representing the values of each point in the environment in the x-plane, y-plane, and z-plane. Subsequently, through the instructions of the host, we transfer the npy file and the captured pictures from the robot to the host. In the above operations, one robot is responsible for taking pictures, passing the npy file and performing the actions, while the other robot is only responsible for performing the actions.

After the host obtains the npy file, it converts the matrix into a 3D plane through the program host.py to form a dot matrix map, and projects it onto the x and y planes to form a top view plane map represented by the 0,1 matrix, and converts it into a visible top view through the save map function. As shown in Fig 1, there are two groups of pictures. On the left side of each group of pictures is the original image taken by the robot, in the middle is the dot matrix formed by processing the npy file, and on the right side is the top view projected onto the x and y planes. The dark areas in the top view file are the zones that are impassable due to obstacles.

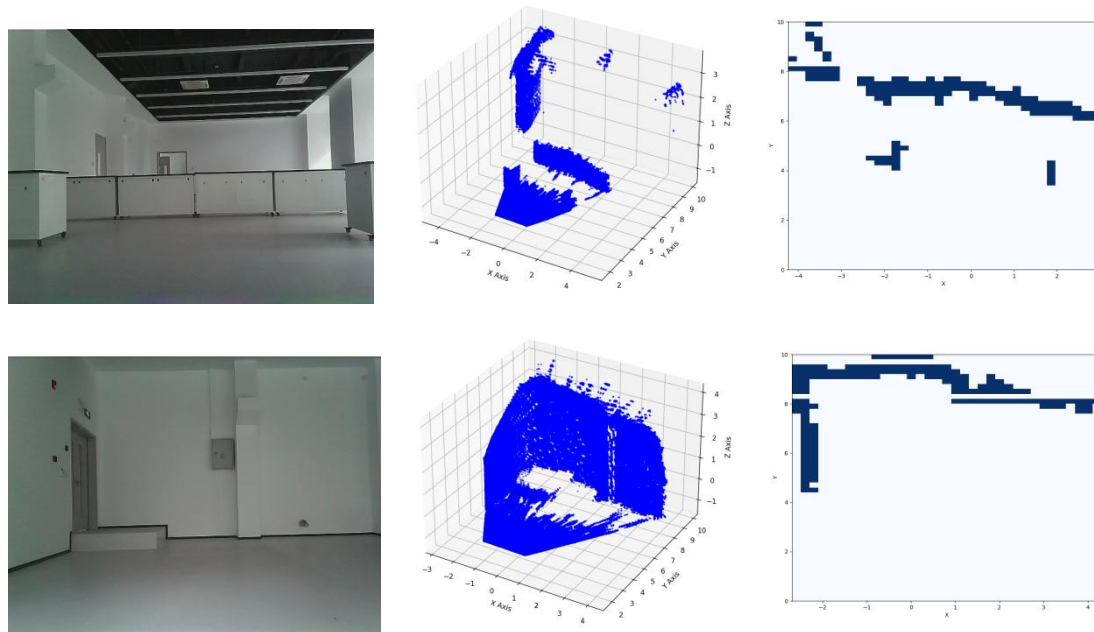


Fig 1

3.2 Dual-agent cooperative obstacle avoidance

In this part, we mainly introduce how two intelligent vehicles, Spark-T, equipped with robotic arms adjust the robotic arms and avoid obstacles according to the height of small obstacles on the map.

We deployed two intelligent vehicles Spark-T equipped with robotic arms in the map, aiming to achieve the collaborative picking up of the same object by the two and successfully exit the maze while avoiding various obstacles. For the two small vehicles jointly transporting an object, we need to conduct constraint modeling on the reachable space of their joints, as shown in Figure 2. Figure 2 is a simplified diagram of two Spark-T robots grasping the same object. The main constraints are as follows in four aspects:

- ①The end effectors of the two robots remain horizontal
- ②The end effectors jointly lift one object horizontally
- ③The angles of each joint range from 0 to 90 degrees
- ④The object and the two robots do not touch the obstacles

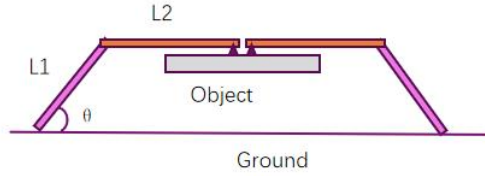


Fig 2

Based on the geometric relationship, we can calculate the height H of the passable obstacles

$$H = L1\sin\theta - h1 - h2$$

Here, $h1$ represents the height of the suction cup, and $h2$ represents the height of the object being held.

The distance from the bottom of the suction cup to the ground is approximately 30cm in reality. Due to the large diameter of the base of the spark mobile platform, on the premise of ensuring the level of the end joints, the movement space of the robotic arm is not large. Therefore, in order to simplify the experiment, when the obstacle was higher than 20cm, two intelligent vehicles bypassed and passed through the obstacle. When the obstacle is less than or equal to 20cm, the two intelligent vehicles avoid the obstacle by the lifting of the mechanical arm, as shown in Figure 3. The left picture shows the lifting of the mechanical arm to avoid the obstacle, and the right picture shows the two intelligent vehicles bypassing the obstacle.

Therefore, when conducting image processing, for the obstacles in the middle of the road, we will first filter out the targets lower than 0.2 meters and only select the scattered points ranging from 0.2 meters to 1.2 meters for projection, thereby achieving the function of obstacle avoidance through the collaboration of dual agents.



Fig 3

3.3 Autonomous Path Recognition and Correction

In this section, we mainly introduce how to convert the current perspective map captured by the robot into a global map and perform error correction in motion control.

3.3.1 Autonomous Path Recognition and Generation

We use the code `client.py` to control the motion of the car. The car moves forward one step by 1 meter and turns one step by 30° . During the rotation process, Robot No. 1 keeps rotating in place, and Robot No. 2 rotates with Robot No. 1 as the center. When it is necessary to turn left, Robot No. 1 is the robot on the left. When it is necessary to turn right, Robot No. 1 is the robot on the right. Figure 4 is a simple demonstration diagram of the rotation. Therefore, we used different file controls for the left and right turns of the two robots respectively.

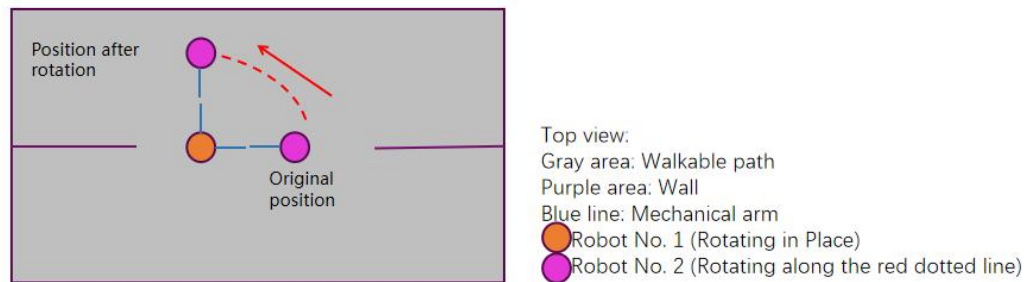


Fig 4

At each step of the robot's operation, it will take a photo of the current position and generate an npy file to save locally. Then, it will transfer it to the host through the host command and generate a dot matrix and top view of the npy file through code for the analysis of the current position. However, all these pictures are based on the robot coordinate system. We need to convert them into the world coordinate system to generate a complete map from the local map. This is because the robot coordinate system is relative to the position and direction of the robot itself and will change as the robot moves, while the world coordinate system is a fixed global reference framework. Converting the image information to the world coordinate system can eliminate the coordinate differences caused by the change of the robot's position and direction, ensuring that all data have a consistent reference benchmark. Thus, the results of multiple local scans can be accurately integrated together to construct a coherent and complete environmental map. Provide accurate spatial information for the global path planning, environmental modeling and subsequent navigation tasks of the robot.

First, we drew a simulation map through Gazebo, as shown in Figure 5. Then, we imported this map through code, along with the current position and direction, to generate the robot's perspective. Next, convert them into the world coordinate system, thereby generating a complete map from the local map. In this process, we need to perform coordinate transformation. There are mainly two transformation steps, namely rotation and displacement. The rotation part uses a two-dimensional rotation matrix, while the displacement part is simply the addition of vectors. The main steps are as follows:

Suppose there is a local coordinate of a point as $(local_x, local_y)$ (That is, the coordinates of the robot in the robot coordinate system). The rotation Angle is α . Then the coordinate of the point obtained after its rotation is $(rotated_x, rotated_y)$:

$$\begin{aligned}\text{rotated_x} &= \text{local_x} \cdot \cos(\alpha) - \text{local_y} \cdot \sin(\alpha) \\ \text{rotated_y} &= \text{local_x} \cdot \sin(\alpha) + \text{local_y} \cdot \cos(\alpha)\end{aligned}$$

Subsequently, the coordinates of the rotated points are converted to global coordinates (global_x, global_y), suppose the global coordinate of the origin is (x, y), then the calculation formula is:

$$\begin{aligned}\text{global_x} &= x + \text{rotated_x} \\ \text{global_y} &= y + \text{rotated_y}\end{aligned}$$

After each step, perform this conversion, and the local map can be transformed into a global map. As shown in Figure 6, there are 4 groups of pictures. The left side of each group is the local map at the current position, and the right side is the current global map generated based on the current position and all the positions passed before. The complete generated video has been placed on GitHub.

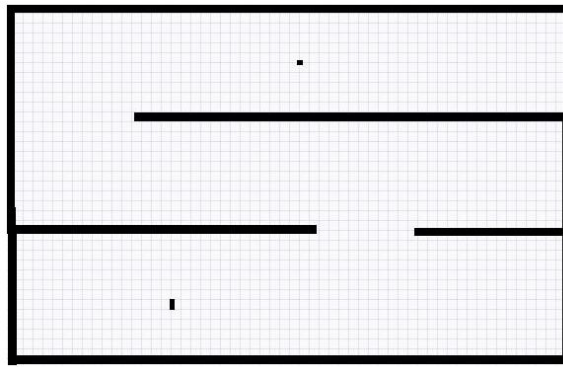


Fig 5

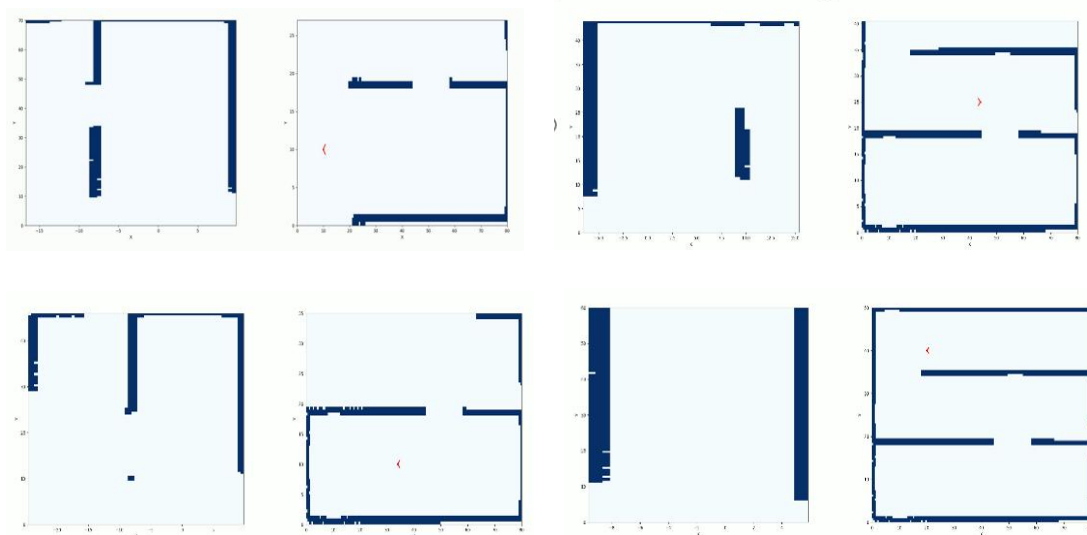


Fig 6

In the actual machine experiment, we returned the npy file at each step. Based on this file, we generated the complete map shown in Figure 7.1 from the local map. Among them, the blue dots at positions (11,6) are the successfully identified low-height but insurmountable obstacles, that is, the two intelligent vehicles will bypass them instead of passing by lifting the mechanical arms. Figure 7.2 is a real-scene map shot, which is roughly the same as the map we generated.

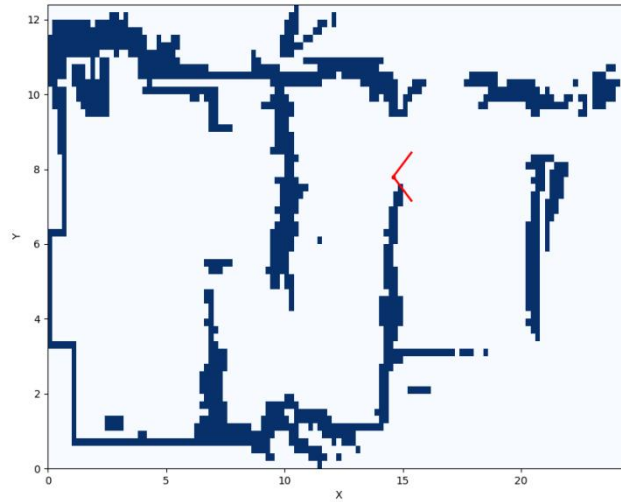


Fig 7.1



Fig 7.2

3.3.2 Autonomous Path Correction

The prerequisite for coordinate transformation is to know the current position and direction of the robot. However, errors may occur during operation due to some reasons, such as the limited accuracy of the wheel speed sensor, which leads to errors in the measurement of the robot's movement speed and displacement, and the wheels may experience slippage and idling. This causes the actual driving distance and direction to be inconsistent with expectations, as well as the simplification of the kinematic model and the changes in environmental factors such as ground friction and slope, resulting in the deviation between the actual motion state of the robot and the model prediction. In addition, since the global map is unknown before the entire task is completed, we cannot determine the position of the small robot on the map. Therefore, after each step of the robot's operation, we need to correct the robot's position and direction based on the information in the image.

After each step of operation, we will intercept the range of $0.2\text{m} \times 0.2\text{m}$ directly in front of the robot, and determine the current position and direction of the robot based on the generated graphic shape.

Take location as an example, as shown in Figure 8. We know that the previous step is 1m. Then, if the robot was 8m away from the wall in the previous step and 6m away from the wall in the current step, but theoretically it should be 7m away from the wall, then the deviation is 1m. After knowing the deviation, we corrected the actual forward distance to 2m instead of the stipulated 1m.

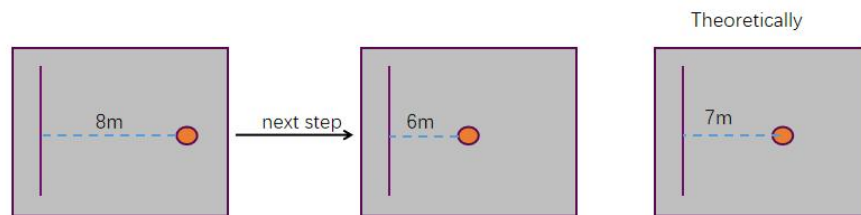


Fig 8

Take the direction as an example, as shown in Figure 9. We know that turning one step is 30° . Then, if in the previous step the robot was directly facing the wall, that is, the Angle with the wall was 0° , but in the current step the Angle between the robot and the wall is 45° , that is, it turned 45° . However, theoretically, after the robot turns 30° , the Angle with the wall should be 60° , so the deviation is 15° . After knowing the deviation, we corrected the actual turning degree to 45° instead of the specified 30° .

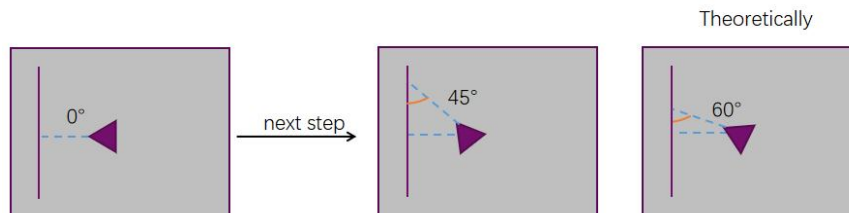


Fig 9

In this section, we mainly introduce how to use the LLM model for path planning of this task and control the movement of the trolley.

Figure 1 consists of four sequential images showing a robot in a room with a large window and a white wall. The robot is a small, black, two-wheeled vehicle with a green sensor or camera mounted on top. In the first image, the robot is positioned in the center of the room, and its shadow is cast onto the floor. In the second image, the robot has moved slightly to the right, and its shadow is cast onto the wall. In the third image, the robot has moved further to the right, and its shadow is cast onto the wall. In the fourth image, the robot has moved to the rightmost position, and its shadow is cast onto the wall. The room has a white floor, white walls, and a large window on the right side.

Fig 10

In each subsequent step when we start to move, the robot will take pictures of the current position and generate npy files, and send them back to the host through the host's instructions for image processing, converting the npy files into top views represented by a 0,1 matrix. First, we carry out the path recognition and path repair of 3.3.1 and 3.3.2. Then, we use the agent.py code to pass the 0,1 matrix to the LLM model. Among them, for the LLM model, we use R1, DeepSeek and Kimi. These three large language models can all achieve path planning. The LLM model will respond with a statement on how to act. We use code to extract the key instructions "Turn left", "turn right", and "go straight", and control the robot to implement the movement of these instructions. As shown in Figure 11, it is the key information we extracted after the response section of the LLM model.

response: <think>

好的，我现在需要帮助用户逃离这个迷宫。首先，理解地图结构是关键。地图由0和1组成，0是空地，1是墙。左下角是(0,0)，当前位置在(1,0)。方向方面，右是0度，上是90度，左是180度，下是270度，现在朝向120度，也就是向左上方偏左的方向。

首先，我需要确定当前位置周围的环境。我位于(1,0)，即第2列（从0开始数），第一行。查看地图，可以看到该点周围的情况。右边是(2,0)，左边是(0,0)，下方可能超出范围，上方是(1,1)。

接下来，分析地图的结构。看最后一行，大部分都是1，所以可能是个封闭的区域，但中间有空洞。我需要找到从边界离开的路径，也就是在迷宫边缘且值为0的地方。

当前朝向120度，这意味着我面对的是左上方，偏向于左侧。接下来，我需要判断是否可以继续直行，或者需要转弯。考虑周围的格子：

1. 直行方向：如果继续前进，下一步会到哪里？朝向120度的话，每移动一次，x坐标减少1，y坐标增加1，因为每45度一个单位，所以120度是左上方。从(1,0)出发，直行一步到(0,1)，检查该点的地图值。查看地图的第10行（假设从下往上数），第10行是“1111111111111111111111111111”，所以(0,1)应该是墙，所以不能直行。
2. 右转：当前朝向120度，右转后变为90度，也就是面朝正上方。检查前方的格子(1,1)，是“10000000000000000000000000000001”，所以(1,1)是0，可以移动。
3. 左转：当前朝向120度，左转变为150度，也就是更偏向左边。前方的格子会是(-1,1)，这显然超出了地图范围，所以不可行。

因此，最优选择是右转，面朝90度，然后前往(1,1)。接下来，再继续寻找出口。

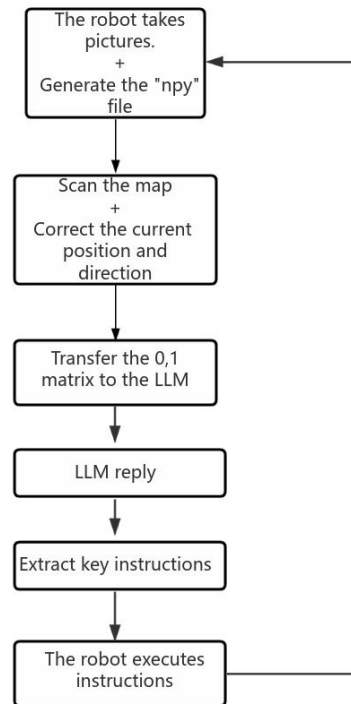
</think>

根据当前位置和方向，你应该向右转。

action: 右转

Fig 11

To sum up, the entire process of this project can be represented by the following flowchart:



4、Summary and Outlook

This project has conducted in-depth research on the path planning of dual-agent collaboration based on large language models, accomplished the established research goals and achieved substantive progress.

In terms of obstacle avoidance through the collaboration of dual agents, we have successfully constructed a maze environment containing obstacles of different heights, and achieved the collaborative picking up of the same object by two intelligent vehicles Spark-T equipped with mechanical arms and getting out of the maze while avoiding obstacles. By sensing the surrounding environment information in real time and adjusting the movement trajectory, the effectiveness of the collaboration of dual agents has been verified.

In terms of autonomous path recognition and correction, by using the built-in camera and image processing technology of Spark-T, error correction is carried out through the comparison of image angles and positions at each step, ensuring that the intelligent vehicle can move forward stably and accurately along the planned path. Meanwhile, the local map information is integrated into the global map through coordinate transformation, providing accurate environmental data for path planning.

In terms of path planning and the integration of large language models, the map dot matrix scanned and generated by the Spark-T camera is converted into a 0-1 matrix and imported into the large language model. The model returns the next motion instruction to control the movement of the robot. Experiments show that all three large language models can effectively handle this task and give reasonable motion instructions. The feasibility of applying large language model technology to path planning has been verified.

In our future work, we mainly have the following three improvements:

First of all, we will be committed to improving the efficiency of large language models. At present, the computational cost and resource consumption of the model remain the key factors restricting its further development. Each step of operation takes 2 to 3 minutes, and it takes a long time to complete the entire map, which is inefficient. Improving efficiency can reduce the resource consumption during the model's operation, accelerate its reasoning and learning speed in complex scenarios, and thereby achieve the effect of wider application.

Secondly, we will conduct experiments using more complex maps. The current map is too simple, with few obstacles and regular shapes of the maze walls. The use of complex maps can provide a test environment that is closer to the real scene, helping us better understand and optimize the collaborative ability and path planning algorithm of the dual-agent system, thereby comprehensively testing the adaptability and navigation ability of the agents.

Finally, since the maze we built used laboratory tables, experiments showed that the metal material on the tables would cause significant errors in the scanning results of the robot. This is because the metal surface may cause abnormal reflection or interference of the sensor signal, thereby affecting the accuracy of the robot's scanning results. To solve this problem, we will replace the materials of the maze by ourselves and re-conduct the experiment with materials such as cardboard that have a smaller scanning image, thereby reducing the adverse effect of metal obstacles on the scanning accuracy and improving the navigation ability of the robot in a complex material environment.

The contributions of the group members:

Wu Xiyun(224040203):

Preliminary research
The robotic arm can achieve spatial analysis
Simulation map drawing
Image processing of maps from the robot's perspective.
Conversion of the robot's perspective into a global map
Report writing

Sun Ruipeng(224040262):

Preliminary research
Real-machine control and operation of Spark-T robots
Establishment of communication between robots and host
Position and angle correction of robots
Path planning and interaction with large language models
Github code upload

Github Link: <https://github.com/passage2016/AIR-5021-Team2-FinalProject>