Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/34/20230303

Research on path planning technology of mobile robot in the restaurant scene based on A* and D* algorithm

Jiahao Han

School of Energy and Power Engineering, Huazhong University of Science and Technology, Wuhan City, Hubei Province, 430074, China

u202011356@hust.edu.cn

Abstract. Intelligent robots have found widespread applications in various fields including medical treatment, industrial production, and social services. These robots are specifically designed to fulfill diverse tasks in different environments, the design and control of these robots play a crucial role in ensuring their efficiency and security. In response to the high demand for fast and convenient service, the catering field is a crucial and promising application field for intelligent robots. This article is primarily focused on the robot in the restaurant scene where the robot could automatically deliver food to the customers, thus reducing the burden on human service providers. In this special environment, the delivery robot must possess the capability to identify its surroundings within the restaurant and autonomously calculate an obstacle-free path from the starting point to the desired destination without colliding with either obstacles or individuals. In this article, the restaurant scene is built in the Gazebo, and map scanning and conversion methods are utilized to enable the robots to recognize the environment effectively. Furthermore, A* and D* algorithms are employed in Python to achieve global and local path planning, respectively, thus validating the feasibility of the methods. The combination of these two algorithms demonstrates a successful path-planning application within the context of a restaurant scene.

Keywords: SLAM, MAP scanning, MAP conversion, A* algorithm, D* algorithm.

1. Introduction

In the past century, the development of information technology, machinery manufacturing, and electronic control have dramatically stimulated technical progress. Some machines have taken over repetitive and heavy-loaded work, the efficiency and quality of production improved, reducing the burden on workers and improving the living standards [1]. In the last decade, society has witnessed the continuous advancement of artificial intelligence and the integration of multiple disciplines and technologies. The integration of these technologies boosted the advent of intelligent robots, and the production process and our lives have been transformed. According to the differences in application environments, robots can be classified as industrial robots, service robots, and specialized robots [2].

The global robot market is expanding constantly and the growth rate of service robots reached 40% annually in 2020, including cleaning robots, delivery robots, educational robots, and so on [2]. In the restaurant scenario, there are a large number of simple and repetitive tasks that have been gradually replaced by corresponding service robots in recent years. Especially during the COVID-19 pandemic,

DOI: 10.54254/2755-2721/34/20230303

service robots were commonly adopted by restaurants to transfer and deliver food directly to customers without human interference, thus reducing the possibility of virus transmission [3]. Although robots have been applied in some restaurant environments, there is still space for improvement in their intelligence. The robots should be able to identify the restaurant environment and coordinate diversified sensors to navigate and move to their destination [4]. While moving, it is essential for the robots to not only avoid crashing obstacles but also avoid colliding with humans. Therefore, information perception and path planning are two important technical foundations for service robots to complete tasks in restaurant environments.

At present, many sensors with different mechanisms have been designed for robots to perceive the surrounding environment. Recent research has primarily focused on robot mapping, localization, and navigation. A promising solution to these challenges is the employment of Simultaneous Localization and Mapping technology (SLAM). The SLAM technology involves the determination of the robot's position within an unknown environment and simultaneously constructing a map of this environment [5, 6]. Multiple sensors are utilized to enhance the robot's environmental perception and convert sensory input into actionable signals [7]. In complex environments, the integration of vision-based SLAM and laser-based SLAM through multi-sensor fusion is commonly employed to eliminate accumulated errors [8]. Sensors play a vital role as they gather crucial information that enables subsequent robot actions, such as navigation and dynamic control.

Robot navigation can be categorized as global navigation and local navigation [8]. Global navigation involves specifying the relationship between objects in the environment and a reference axis, enabling movement toward predetermined destinations. Local navigation, on the other hand, focuses on recognizing dynamic environmental changes and establishing positional correlations between distinct elements [9].

To guide robots to their intended destinations, path-planning methods are essential. Obstacle-free paths are the priority of path planning, so various strategies and algorithms have been introduced to address navigation challenges [9, 10]. The Dijkstra algorithm and its modifications are frequently employed in applications, whereas A* and its variants are suitable for static contexts due to their ability to overcome Dijkstra's computationally intensive blind search problem by utilizing heuristic functions [10]. However, both of these algorithms face limitations when dynamic environments are involved. D* and its variants demonstrate the effectiveness and facilitate path planning in dynamic settings [9]. Additionally, path planning in vast and complicated ecosystems has been addressed by meta-heuristic algorithms including RRT, Genetic, Ant Colony, and Firefly algorithms [10, 11].

In the context of restaurant scenario, robot navigation involves both global and local navigation. The restaurant's map and configurations are static environment, enabling the creation of an optimal path between the starting point and destination. However, the presence of crowds requires the inclusion of local navigation, allowing the robots to identify obstacles within a dynamic environment. This article incorporates mapping scanning in Gazebo environment and map conversion methods in Python, thus providing robots with information on setting obstacles. Furthermore, path planning algorithms are verified and tested in Python. The A* algorithm is utilized for global navigation before the restaurant operates, some cleaning and preparation tasks could be done. During business hours, D* algorithm addresses the challenges posed by potential moving obstacles along the path toward the destinations. With the combination of these two algorithms, optimal path could be simulated and visualized in Python.

2. Methodology

2.1. Analysis of restaurant scenarios

Generally, restaurant scenarios are complex and distinct, different types of restaurants have distinctive layouts, as each restaurant type necessitates a distinct layout tailored to meet its specific requirements [12, 13]. While in this article, the focus will be put on the most basic settings, including the kitchens, scattered customer tables, and private rooms. To effectively simulate the path planning algorithm, a comprehensive restaurant model has been developed using Gazebo, a powerful 3D modeling software.

DOI: 10.54254/2755-2721/34/20230303

This Gazebo-based 3D model serves as a detailed restaurant map, which can be readily accessed and processed using Python for efficient calculations.

For optimal performance of robots within a restaurant setting, their path planning holds paramount significance, with safety and efficiency emerging as the two most critical considerations. First, robots are not allowed to cause damage to either human beings or themselves. As a result, the movement of the robots should avoid any collision with humans or obstacles. In the restaurant case, the mobile robots should select an obstacle-free path. At the same time, when human beings move on the selected path, the robots should have the capability to detect such an emergency and adjust their path. Additionally, the robots should possess the capability to swiftly detect emergencies caused by human movement and promptly adjust their path to prevent any incidents.

2.2. Scanning and conversion of restaurant scene maps

2.2.1. Map scanning based on sensors. SLAM technology employs multiple sensors to collect information of its surroundings and process the information, enabling it to build a map and track its own position in an unknown environment. Commonly used sensors for SLAM include monocular cameras, binocular cameras, depth cameras, and LiDAR. In the restaurant scene, the model being used is the turtlebot3 model with a lidar mounting on it [14]. The lidar scanner emits a beam of light that rotates around the robot, and the sensor measures the distance to the nearest object that the beam hits. The environment information will be sent to the Rviz platform node while the robot moving around. This instant communication allows the creation of a scanning map.

2.2.2. Map conversion for path planning. The map generated by the sensors cannot be directly utilized for path planning calculations due to the immense pixel count in the image. To reduce computational costs and interfering factors, Python Imaging Library (PIL) is used to process images to a black-and-white version. Every pixel of the original image is traversed, and its color is judged and changed by setting the color threshold. Additionally, Matplotlib and Numpy are included to achieve Multidimensional array object and matrix operation functions. The data from each pixel in the black-and-white image is stored in a matrix, forming a grid image [15]. Each grid cell in the grid map has fixed neighbors, which enables search algorithms to quickly find adjacent grid cells. In grid maps, obstacles can be handled by marking the corresponding grid cells as impassable. In this way, the search algorithm can avoid these obstacles when planning the path. Thus, this grid image becomes a convenient and reliable basis for path planning simulations.

2.3. Path Planning Methods in Restaurant Scenarios

2.3.1. Global path planning based on A Algorithm.* The restaurant scene is a fixed environment because the tables and any other obstacles are placed in specific positions. Therefore, the A* algorithm is well-suited for calculating the best route from the beginning point to the goal point in a static environment where obstacles do not change. Each position of the obstacles is provided by the grid map, and the A* algorithm employs a heuristic search algorithm that evaluates each node by combining the actual price from the start position to the current node and the estimated cost from the current node to the endpoint. This method of evaluation allows the A* algorithm to swiftly identify the shortest route from the origin to the destination.

DOI: 10.54254/2755-2721/34/20230303

2.3.2. Local path planning based on D Algorithm.* In the restaurant scene, it's inevitable that human beings may cross paths with robots, potentially leading to collisions. Hence, it's imperative for robots to possess the capability to continuously detect their surrounding environment and adjust their path accordingly to cope with moving obstacles. An incremental search algorithm is the D* algorithm, which can quickly correct the existing path when the environment changes. The D* algorithm achieves this by maintaining a graph of the shortest paths from destinations to origins. When the environment changes, the D* algorithm only needs to update the affected parts without recomputing the entire path. Therefore, the D* algorithm stands out as an excellent choice for path planning in dynamic restaurant environments.

3. Experimental results and analysis

3.1. Experiment setting

To simulate the restaurant scenario, a configuration is created in the Gazebo by using the building editor. Figure 1 describes several fundamental infrastructures in the restaurant environment. The brown wall restricts the total area of the restaurant, the light blue walls represent several private rooms in the restaurant. The light green walls indicate the kitchen area and gray squares depict the scattered tables outside the rooms.

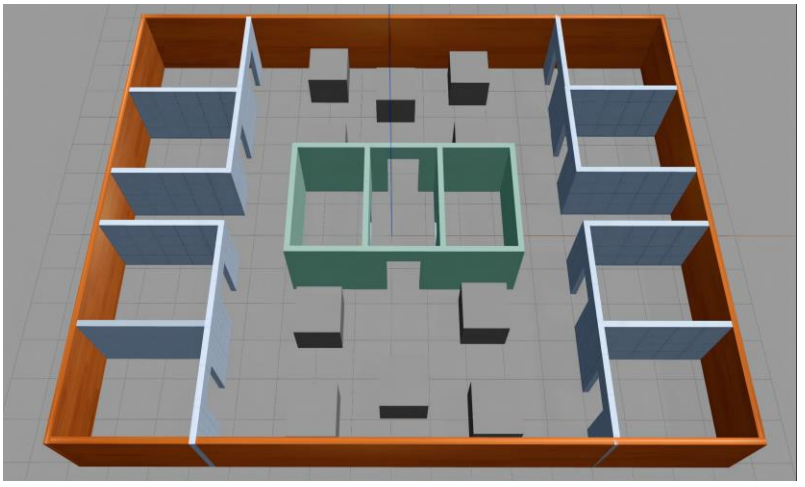


Figure 1. Environment settings in the Gazebo.

Relevant parameters of the settings are listed in Table 1. Moreover, to ensure smooth movement of the robot without any collisions, each aisle's width in the experimental environment is set to be at least 1m. The Gazebo environment provides a platform for the robot to move and scan the restaurant scenario. The turtlebot3 model can be imported into the restaurant environment and the sensors mounted on it provide information about the settings.

Table 1. Environment settings and parameters in the Gazebo.

Settings	Color	Size	Number
Wall	Brown	24m*20m	1
Private rooms	Light blue	4m*3m	8
Kitchen	Light green	7m*4m	1
Scattered tables	Gray	2m*2m	10

The Gazebo setting is primarily designed for accessing the map of the restaurant. Subsequently, the path planning task is mainly carried out in the Python environment. Utilizing the map scanned by the turtlebot3, the data can be converted and processed to generate a grid map, as shown in Figure 2.

DOI: 10.54254/2755-2721/34/20230303

Similarly, the size of the settings remains consistent, with each color representing the same infrastructure, as indicated in Table 1. Notably, the green and red points on this grid map represent the starting point and goal point, respectively. In Python, the path planning algorithms will be tested and simulated using this map.

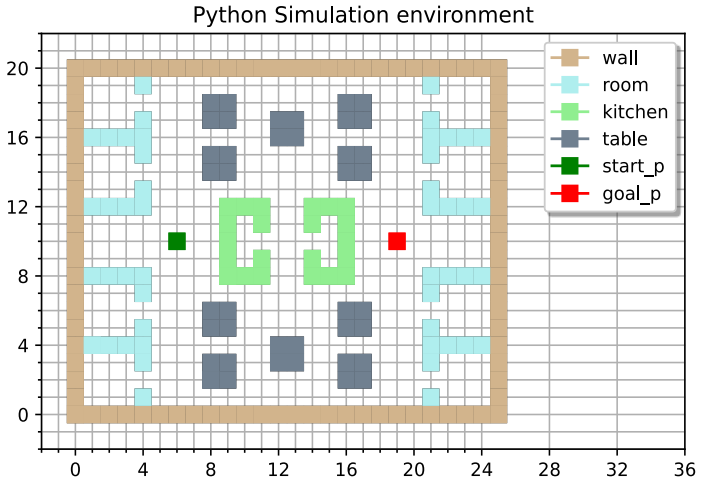
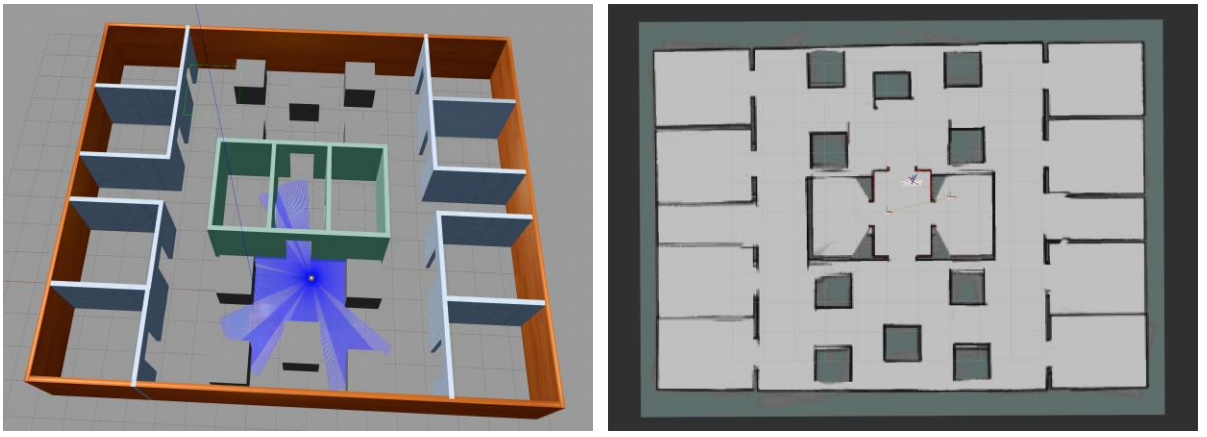


Figure 2. Python Simulation environment.

3.2. The scanning and conversion of the map

Figure 3. (a) describes that Turtlebot3 burger model is utilized to scan the restaurant scenario. The scan map is a package of turtlebot3 and this ros package provides a node that could transfer lidar data into a 2D map. The scan map publishes map data to a topic, and the Rviz platform subscribes to this topic to receive map data and display it in Figure 3. (b).



(a) (b)

Figure 3. Map scanning in the Gazebo using Turtlebot3 burger and Rviz platform.

A grid map is a method of dividing an environment into grids, where each grid represents a state. Path planning algorithms can use a grid map to represent the environment and search for optimal paths within it. The advantage of the grid map is that it is easy to implement and calculate. The conversion from the scanning map to the grid map is achieved in Python. Figure 4. (a) depicts the map scanned by turtlebot3 and exported via the Rviz platform. The color purity and clarity of the image are not enough, making it challenging to recognize and process. In Figure 4. (b), Python's PIL library processes the image to a black-and-white version by traversing each pixel and evaluating its average RGB value.

DOI: 10.54254/2755-2721/34/20230303

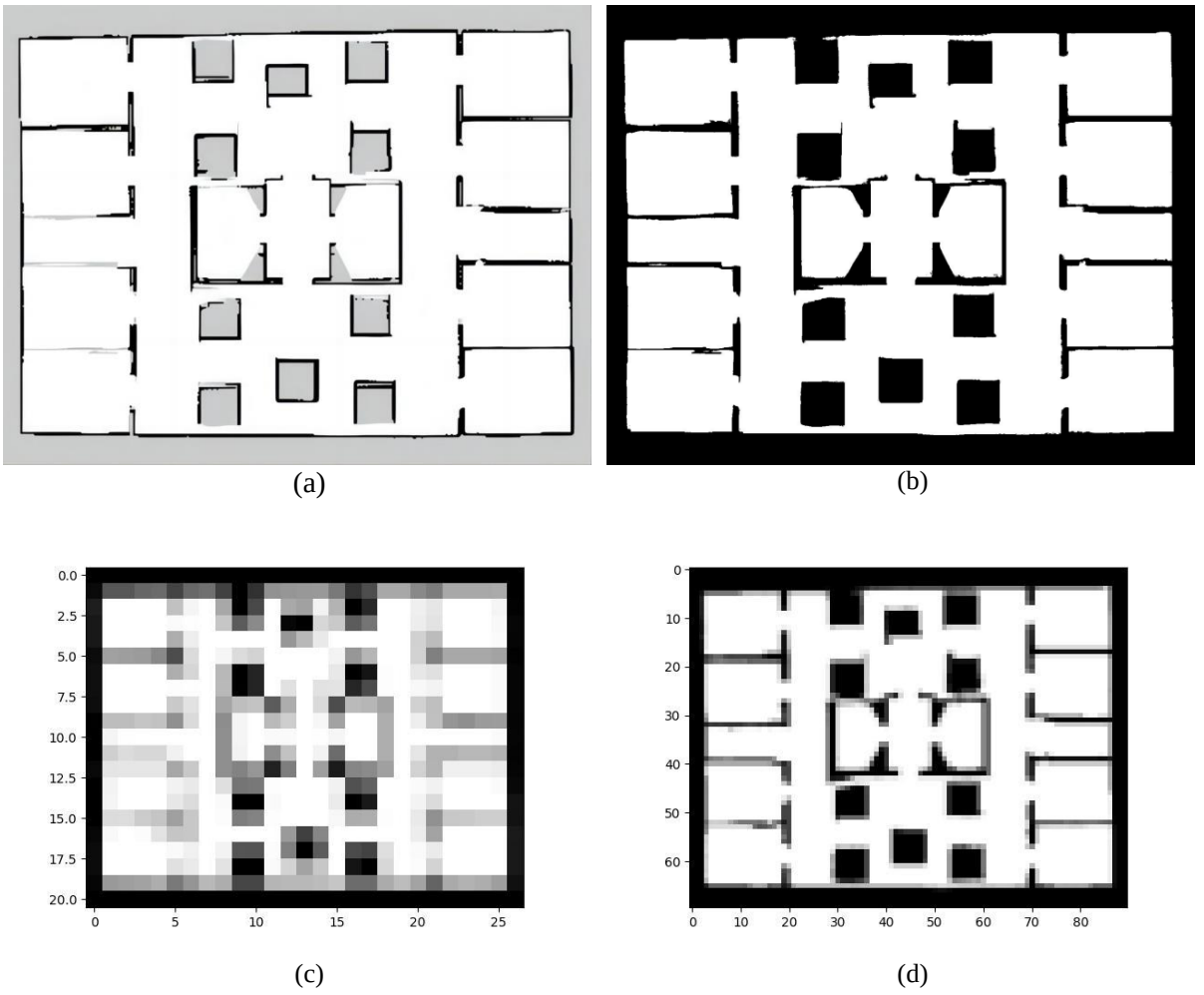


Figure 4. Map conversion in Python.

The data of each pixel traversed will be stored in an array after determining the proportion of black color. This array can be used to generate a grid image. To create a grid map, the grid size is a paramount factor. It represents the size of each separate part. When the grid size is larger, more pixels are encompassed within each grid, but this may lead to lower sharpness and increased distortion in the resulting grid image. Nevertheless, the smaller grid size means more calculations for the array. For instance, in Figure 4. (c), the grid size is 20, resulting in an array size of (24,32), while in Figure 4. (d), the grid size is 5, leading to a larger array size of (96,128). By comparing these two conditions, the image distortion of Figure 4. (c) is more serious than Figure 4. (d), while the calculations of Figure 4. (d) is more intensive. Therefore, during path planning simulations in Python, the grid size should be carefully and thoughtfully selected because the larger the array size is, the more calculation is required, and the worse the latency of the algorithm.

3.3. Path planning results

To generate a perfect grid map, it is essential to select an appropriate grid size and utilize the data stored in the array. The array values of "1" represent the positions of obstacles or walls, while "0" denotes clear areas. By processing this information, a grid map in Figure 2 can be created.

A* algorithm is primarily designed for the static environment because any changes in the environment will lead to a complete recomputing, slowing down the algorithm or resulting in path errors. In a restaurant scene, where most infrastructures remain fixed, the A* algorithm could be employed to

DOI: 10.54254/2755-2721/34/20230303

guide the robot from the starting point to the goal point in the situation without human beings moving around. Figure 5. (a) demonstrates the feasibility and the outcome of the A* algorithm.

Additionally, during restaurant operations, the robot must avoid collisions with human beings while moving. In the Python simulation, a new additional obstacle is added to the map. The position of that obstacle is determined by where the mouse is clicked. The obstacles represent human beings and are shown in black color. When the newly added black obstacles interfere with the existing path, the D* algorithm updates the affected environment and handles such dynamic and variable environment. The robot's path execution when encountering different obstacles is shown in Figure 5. (b-d).

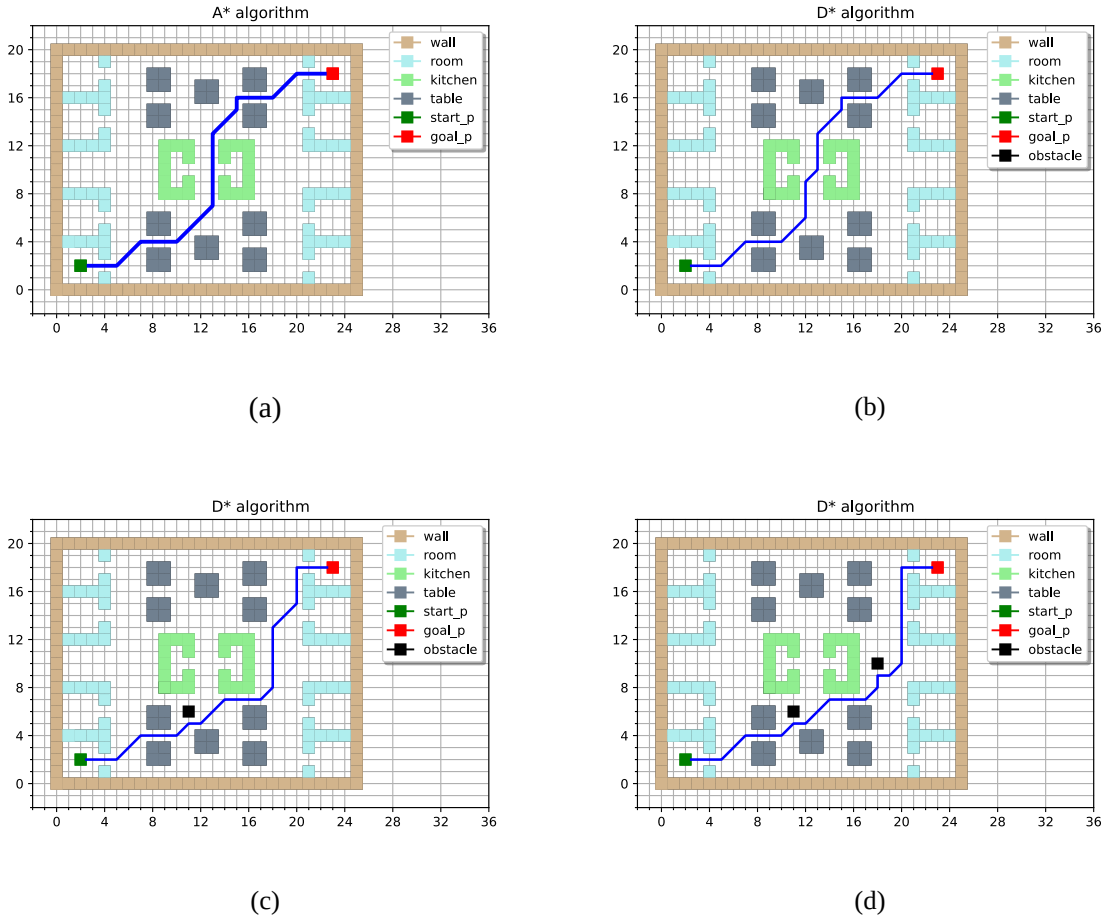


Figure 5. Path planning experimental results in Python.

4. Conclusion

The restaurant scenario with essential infrastructures is constructed using the Gazebo environment, where the map scanning task is achieved. The robot's mounted sensors provide crucial information about the settings, enabling the robot to gather data. The map conversion and path planning tasks are executed in the Python environment, where the map scanned by the turtlebot3 is converted and processed into a grid map. To get the best results from path planning simulations, the grid size and computation requirements must be balanced properly. The carefully selected grid size ensures accurate path planning while minimizing computation requirements and algorithm latency.

In Python, the path planning algorithm is tested and simulated within that grid map, allowing the robot to efficiently navigate the restaurant while avoiding collisions. The D* algorithm is superior for

DOI: 10.54254/2755-2721/34/20230303

dynamic situations, such as a restaurant with changing obstacles, because it can quickly adjust to changes while the A* algorithm is effective for identifying the best paths in static surroundings. To be more specific, the A* algorithm is suited for cleaning and preparing before the restaurant operates because its path won't be affected by human activities. D* algorithm is perfect for delivering and serving while the restaurant is operating because the sensors will timely provide the information about obstacles around, thus shifting their path accordingly.

The combination of A* and D* algorithms could cover most service requirements in the restaurant scenario. Ultimately, this integrated approach enables successful path planning and navigation of the robot within the restaurant scenario, making it an effective and intelligent solution for real-world applications.

References

[1] Nayyar, Anand, and Akshi Kumar, eds. A roadmap to industry 4.0: Smart production, sharp business and sustainable development. Berlin: Springer, 2020.

[2] Wang, Tian-Miao, Yong Tao, and Hui Liu. "Current researches and future development trend of intelligent robot: A review." International Journal of Automation and Computing 15.5 (2018): 525-546.

[3] Zeng, Zhanjing, Po-Ju Chen, and Alan A. Lew. "From high-touch to high-tech: COVID-19 drives robotics adoption." Tourism geographies 22.3 (2020): 724-734.

[4] Yang, Guang-Zhong, et al. "Combating COVID-19—The role of robotics in managing public health and infectious diseases." Science Robotics 5.40 (2020): eabb5589.

[5] Durrant-Whyte, Hugh, and Tim Bailey. "Simultaneous localization and mapping: part I." IEEE robotics & automation magazine 13.2 (2006): 99-110.

[6] Bailey, Tim, and Hugh Durrant-Whyte. "Simultaneous localization and mapping (SLAM): Part II." IEEE robotics & automation magazine 13.3 (2006): 108-117.

[7] Chong, T. J., et al. "Sensor technologies and simultaneous localization and mapping (SLAM)." Procedia Computer Science 76 (2015): 174-179.

[8] Chen, Chin S., Chia J. Lin, and Chun C. Lai. "Non-contact service robot development in fast-food restaurants." IEEE Access 10 (2022): 31466-31479.

[9] Patle, B. K., et al. "A review: On path planning strategies for navigation of mobile robot." Defence Technology 15.4 (2019): 582-606.

[10] Karur, Karthik, et al. "A survey of path planning algorithms for mobile robots." Vehicles 3.3 (2021): 448-468.

[11] Majeed, Abdul, and Seong Oun Hwang. "Path planning method for UAVs based on constrained polygonal space and an extremely sparse waypoint graph." Applied Sciences 11.12 (2021): 5340.

[12] DiPietro, Robin. "Restaurant and foodservice research: A critical reflection behind and an optimistic look ahead." International Journal of Contemporary Hospitality Management 29.4 (2017): 1203-1234.

[13] Malekshahi, Armita. Investigation on restaurant layout design. Diss. Eastern Mediterranean University (EMU), 2013.

[14] Stan, Alexandru-Calin. "A decentralised control method for unknown environment exploration using Turtlebot 3 multi-robot system." 2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI). IEEE, 2022.

[15] Tsardoulias, Emmanouil G., et al. "A review of global path planning methods for occupancy grid maps regardless of obstacle density." Journal of Intelligent & Robotic Systems 84 (2016): 829-858.