applied sciences

MDPI

Article

High-Efficiency Navigation of Nonholonomic Mobile Robots Based on Improved Hybrid A* Algorithm

Zhaozhan Chi [1], Zhenhan Yu [2], Qianyu Wei [3], Qiancheng He [1], Guangxian Li [1,4,*] and Songlin Ding [4,*]

[1] School of Mechanical Engineering, Guangxi University, Nanning 530004, China
[2] School of Public Policy and Management, Guangxi University, Nanning 530004, China
[3] School of Journalism and Communication, Guangxi University, Nanning 530004, China
[4] School of Engineering, RMIT University, Victoria 3083, Australia
* Correspondence: liguangxian@gxu.edu.cn (G.L.); songlin.ding@rmit.edu.au (S.D.)

Abstract: With the development of automation technologies, autonomous robots are increasingly used in many important applications. However, precise self-navigation and accurate path planning remain a significant challenge, particularly for the robots operating in complex circumstances such as city centers. In this paper, a nonholonomically constrained robot with high-precision navigation and path planning capability was designed based on the Robot Operating System (ROS), and an improved hybrid A* algorithm was developed to increase the processing efficiency and accuracy of the global path planning and navigation of the robot. The performance and effectiveness of the algorithm were evaluated by using randomly constructed maps in MATLAB and validated in a practical circumstance. Local path planning and obstacle avoidance were carried out based on the model predictive control (MPC) theory. Compared with the conventional A* + DWA (dynamic window approach) method, the average searching time was reduced by 12.62~24.5%, and the average search length was reduced by 9.25~9.5%. In practical navigating tests, the average search time was reduced by 18~24%, and the average search length was reduced by 10.3~12%, while the overall path was smoother. The results demonstrate that the improved algorithm can enable precise and efficient navigation and path planning of the robot in complex circumstances.

Keywords: nonholonomic mobile robots; self-navigation; mapping; path planning; hybrid A* algorithm

Citation: Chi, Z.; Yu, Z.; Wei, Q.; He, Q.; Li, G.; Ding, S. High-Efficiency Navigation of Nonholonomic Mobile Robots Based on Improved Hybrid A* Algorithm. *Appl. Sci.* **2023**, *13*, 6141. https://doi.org/10.3390/app13106141

Academic Editors: Oscar Reinoso García, Fabrizio Giulietti, Nadjim Horri and William Holderbaum

Received: 29 March 2023
Revised: 11 May 2023
Accepted: 12 May 2023
Published: 17 May 2023

1. Introduction

With the development of artificial intelligent and automation technologies, mobile robots with self-navigation capabilities are gradually replacing conventional machines such as manually operated vehicles. However, the working areas and traffic conditions for autonomous mobile robots could be complex, e.g., in the central business district. To work efficiently in these environments, an autonomous robot should have three functions [1]: (1) global planning capability under different working conditions, which enables the robot to plan a feasible global path efficiently according to the local surroundings; (2) tracking and controlling capability to ensure high precision in the movement based on the path planning; and (3) rapid response to environmental constraints and planning of new feasible paths in a time when the local environments are changed due to the passing-by vehicles or pedestrians [2].

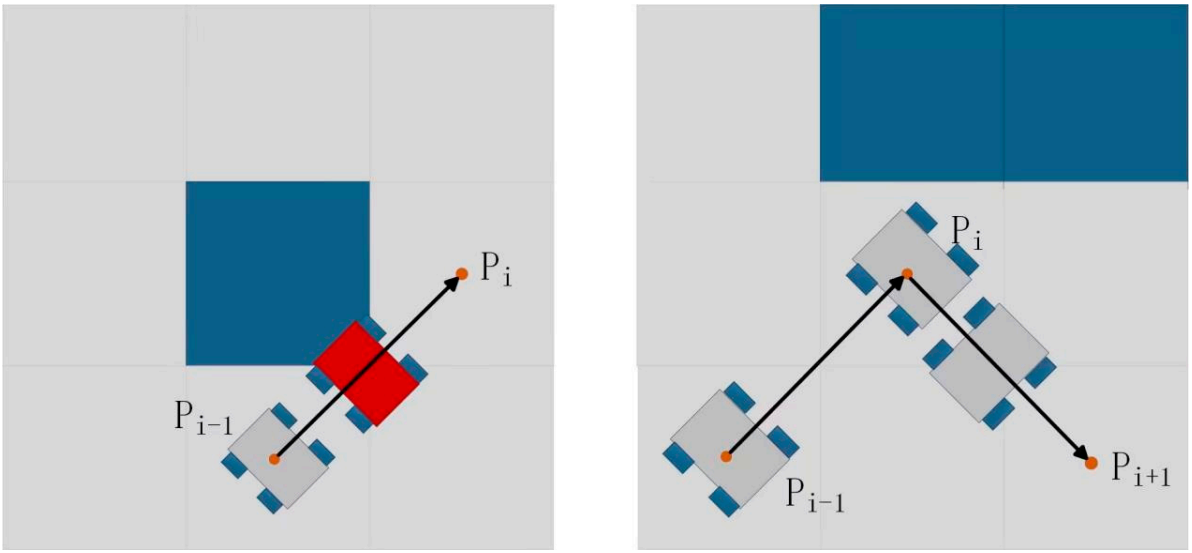
The planning of the path can be divided into global path planning and local path planning. Global path planning is designing the route of the robot in a fully perceived environment. It is the process of constructing a two-dimensional or three-dimensional map of the known environment with specific constraints [3] and target information [4]. For the global path planning, two classical searching methods are often used to detect the positions and areas of obstacles: (1) searching algorithms based on random samplings, such as the probabilistic road-map (PRM) [5], rapidly exploring random tree (RRT) method [6],

and RRT algorithm based on convolutional neural network (CNN) [7]; and (2) graph search-based algorithms, such as the Dijkstra [8] and extended Dijkstra algorithms [9]. The Dijkstra algorithm is characterized by its ability to guarantee that the obtained path will be the shortest. However, it needs to scan all the nodes in the map, and the number of nodes is generally immense. This results in a significant increase in computation and a potential decrease in solution rate, while also consuming significant memory space.

To increase the scanning/searching efficiency, A* algorithm was proposed by using a heuristic function as the search-guided approach [10,11]. The mathematical principle of the A* algorithm is presented in Equation (1). To determine the priority of the next node f(n) in the scanning/searching process, two cost functions, $g(n)$ and $h(n)$, are used in the equation: $g(n)$ represents the cost from the current node to the start node of the path, and $h(n)$ represents the estimated cost from the next node to the end node of the path.

$$f(n) = g(n) + h(n) \tag{1}$$

Based on the A* algorithm, some improved algorithms were developed recently. For example, Harabor and Grastien developed the jump point search (JPS) algorithm [12] by storing nodes in the open list to improve search efficiency. Li et al. proposed self-adaptive learning particle swarm optimization (PSO) with different learning strategies [13] to improve the search capability of PSO. With the development of nonholonomically constrained robots, path planning becomes more and more important for robots, such as the Ackermann mobile robot chassis. However, the A* algorithm is not effective in planning path for nonholonomically constrained mobile platforms. When planning the paths for such robots, post-processing and smooth optimization processes are required before the application of the paths because such robots are only feasible to paths that satisfy their kinematic constraints, and they are not capable of tracking all the continuous paths. As shown in Figure 1, the Ackermann robots cannot traverse angles greater than 90° (such as the red robot in Figure 1a) in a limited range when confronting such obstacles (the blue grid) in real-world scenarios. Therefore, flexible path planning should be used to avoid potential collisions (Figure 1b).



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

Figure 1. The collision schematic diagram. (**a**) The collision of Ackermann chassis due to the limitation of transverse angle. (**b**) Avoiding collision via flexible path planning.

In addition to the global path planning, local path planning is also critical when the working environment keeps changing dynamically. The local dynamic path planning is mainly divided into two approaches [14]. The first approach involves the detection

of new obstacles in the environment and obtaining space information. The robot then conducts replanning at the navigation planning level to obtain a new collision-free path that is sent to the underlying tracking controller for executing movement. The typical methods in this category include the artificial potential field (APF) method, grid planning method, and others. However, these methods require replanning and outputting control commands, resulting in a large amount of computation, and cannot meet the real-time requirements. The other approach to local path planning at the motion control level involves considering the nonholonomic-constraint characteristics of the robot. The primary concept is to establish an optimal control problem in the unit-time step when facing new obstacles and solve the optimization problem while obtaining the optimal trajectory and the corresponding control input. The primary representative methods of this approach include the dynamic window approach (DWA) [15], time-elastic belt (TEB) approach [16], and local planning based on model predictive control (MPC) [17] which is widely used for avoiding collisions [18]. Yi Gan et al. proposed the A1-0 Bg-RRT algorithm to reduce computational time in complex environments [19]. Zhang et al. proposed the algorithm for ships with INS-aided shipborne GNSS, which considered the measurement update from one-dimensional speed [20]. Zhang et al. introduced the Bézier curve method to smooth the routes [21]. However, when applied to nonholonomic constraints, these algorithms show slow navigation speed and can generate non-optimal paths.

Nonholonomic mobile robots such as Ackermann architecture mobile robots, single-wheeled inverted pendulum robots and three-wheeled omnidirectional mobile robots have fewer controlled variables than state variables during motion, which can easily lead to motion slipping problems. How to control nonholonomic mobile robots to reach the target location and face the target direction are a challenging task, and extensive research has been conducted in this area. For example, Karl Worthmann et al. [22] proposed a predictive control scheme based on tailored nonquadratic stage cost to address this issue. Dang et al. [23] proposed a novel evaluation function based on the Theta-star search algorithm. The new function can generate waypoints with an empirically averaged distance to obstacles. In 2020, Li et al. [24] proposed a graph-based multi-agent path planner. The planner can effectively generate optimal collision-free trajectories for multiple nonholonomic mobile robots in environments with multiple obstacles. Johnson et al. [25] proposed the dynamic motion planning networks (Dynamic MPNet) algorithm. They modified the planning network to achieve real-time planning for nonholonomic robot motion planning problems, instead of simply relaxing constraints. By adjusting the steering of the robot through its path planner or driver, Gonon et al. [26] developed the method to assist navigation in dense crowds.

However, although these path planning algorithms can accommodate nonholonomic constraints in robots, many inherent issues exist, e.g., suboptimal operating speed, failure in producing smoother paths with higher accuracy, and the randomly occurred excessive searches. This is because the heuristic function $h(x)$ of the conventional A* algorithm is generally the Euclidean distance, which traverses a very large number of unnecessary nodes in planning the path and significantly limits the efficiency of navigation. To optimize the path planning, an improved hybrid A* algorithm was proposed in this paper with the intension to improve the performance of autonomous navigation of mobile robots with nonholonomic chassis. The Chebyshev distance [27] was used to weight the heuristic function in the global path planning, and the local path planning algorithm was improved as an MPC local planner. The performance of the algorithm was tested in MATLAB and validated in practical navigation experiments, and the conventional A* algorithm was used as comparison to demonstrate the effectiveness of the new algorithm. By comparing the paths planned for maps with different obstacles, it was found that shorter and smoother paths were generated via the improved A* algorithm due to the application of the Chebyshev distance, which increased the efficiency and accuracy of the overall path planning. The results of this study provide a practical method for the efficient autonomous

navigation of robots with the nonholonomic-constraint mobile platform, e.g., the pit robot with Ackermann chassis.

2. Path Planning for Nonholonomic Robots

2.1. Map Construction and Robot Positioning

Simultaneous localization and mapping (SLAM) [28], which includes laser SLAM, visual SLAM, and multi-sensor fusion SLAM according to the type of sensors employed, was the commonly used method in the construction of maps for different working environments [29]. In this study, laser SLAM was chosen as the localization and mapping method due to its higher accuracy, faster speed, and wide-ranging capabilities with the using of LiDAR. The sensor emits laser signals to perceive the surrounding environment, and the obstacles reflected back to the receiver by the laser are used to construct the map based on the sizes, shapes, and positions of the obstacles. In plotting the map, the geometric mapping approach, the topological mapping approach, and the occupancy grid mapping approach are the commonly used methods. In this study, the occupancy grid map approach, which plots the map by small squares, was employed. The number of squares represents the size of the map, and the probability of the obstacle in a square is higher when the number is larger. Then, the plotted map was meshed, and the values in the range of 0–255 were defined to each cell presenting different occupied characters (vacancy, obstacles, etc.). The computer uses this feature to discriminate obstacles. A two-dimensional map was constructed digitally via the G-mapping, which is a widely used SLAM algorithm in ROS. This algorithm is an improvement upon the Rao-Blackwellized particle filter (RBPF) approach, which prioritizes positioning before map construction.

2.2. Robot Positioning

The positions of a robot can be determined via the SLAM algorithm which utilizes the probability distribution approach; particles are used to represent an estimation of the robot's position based on sensor measurements. However, the algorithm could generate a very large number of particles, leading to high computational load and time delay. Furthermore, the SLAM algorithm is less effective in addressing the robot kidnapping problem, i.e., when the robot is suddenly moved to a different location or experiences tire skidding. As a result, the positioning of robot was implemented via the AMCL algorithm, which utilizes a particle filter to calculate the robot's position according to the data from both the odometer and LiDAR sensors. Moreover, AMCL can be applied in solving both global and local localization problems to reduce the workload required for subsequent navigation algorithm. With regard to robot kidnapping, AMCL incorporates weights determined by the information from short-term observation and long-term observation. These weights enable more reasonable particle number and time allocation, as expressed in Equations (2) and (3), respectively:

$$w_{long\_t+1} = w_{long\_t} + a_{long}\left(w_{avg} - w_{long\_t}\right) \tag{2}$$

$$w_{short\_t+1} = w_{short\_t} + a_{short}\left(w_{avg} - w_{short\_t}\right) \tag{3}$$

where w_i is the observation average function, a_i is the average weight coefficient determined by the observed information, i has three states: long-term, short-term, and average. In the AMCL algorithm, the probability of increasing particle number is related to the two weight coefficients. The weights were updated by iteration, and the values were typically within the range of 0.0010~0.01. The probability of increasing the particle number can be calculated using the following equation:

$$P = \max\left\{0.0, 1.0 - \frac{w_{long}}{w_{short}}\right\} \tag{4}$$

It can be seen from Equation (4) that the increase in the particle number is related to the value of $\frac{w_{long}}{w_{short}}$. When the weight of short-term information is less than that of long-term information, which indicates that the robot has deviated from the predetermined trajectory, the system will increase the number of random particles. At this stage, the robot's self-positioning capability can be achieved more quickly and accurately by matching a larger number of particles. The visualized process of the robot's self-positioning process via AMCL algorithm is illustrated in Figure 2. Through multiple updates and resampling of the particle set, the particle filter can adjust the weight and position of the particles to reflect the probability distribution of the robot's position. The location where particles appear more frequently is where the robot is more likely to be located.

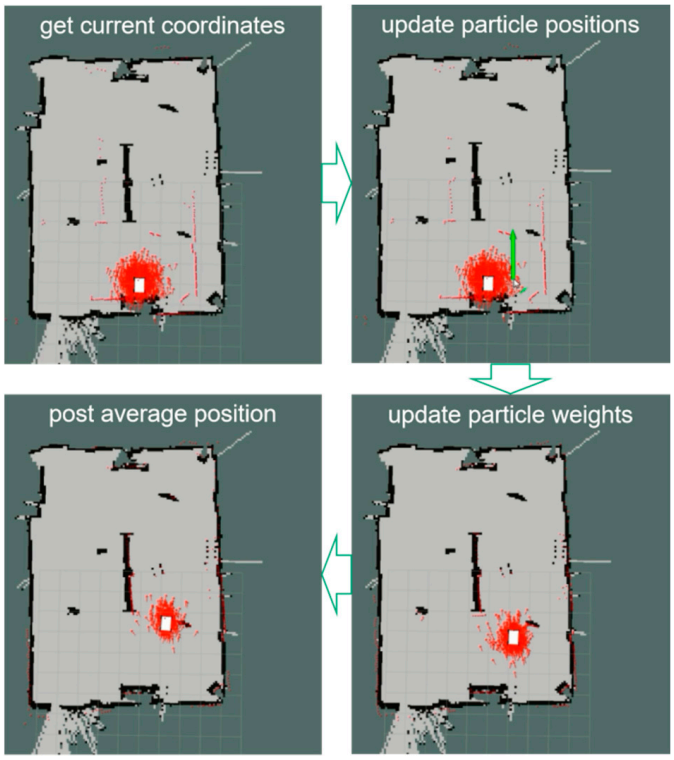


Figure 2. Robot self-positioning.

2.3. Path Planning Based on Improved Hybrid A Algorithm*

2.3.1. Grid Environment Modeling

The planning of paths was determined according to the positions of vacancies and obstacles on the constructed maps. As introduced in Section 2.1, grid mapping approach was used for the construction of the map, and the map was meshed into grids with certain numbers presenting starting point, end point, obstacles, security areas, and other relevant information. As shown in Figure 3, the black regions in the grid map represent obstacle areas, while the white regions represent areas which are available for travel, and both areas can be updated in real time according to the positions of obstacles and vacancies when the environment is changed.

2.3.2. Global Path Planning Based on Improved Hybrid A* Algorithm

Robot path planning consists of global path planning and local path planning. As mentioned before, the processing efficiency of conventional A* algorithm is limited when the areas of the obstacles are complex. Therefore, the hybrid A* algorithm [30] was used for the global path planning. The algorithm improved the specification of the heuristic function and the searching method to achieve better results. For the heuristic search-based path planning, the effectiveness of the heuristic function $h(x)$ is critical. In

conventional A* algorithm, $h(x)$ is usually the Euclidean distance between the two nodes (x_n, y_n) and (x_g, y_g), as presented in Equation (5).

$$h(x) = \sqrt{(x_n - x_g)^2 + (y_n - y_g)^2} \tag{5}$$

However, this approach causes the traversal of unnecessary nodes and an increase in the redundancy of the path planning when the distance calculated via $h(x)$ is smaller than the actual path. To address the limitations of the Euclidean-distance heuristic function, Chebyshev distance d_q, which was expressed in Equation (6), was applied. The improved algorithm is presented in Equation (7).

$$d_q = max\{|x_n - x_g|, |y_n - y_g|\} \tag{6}$$

$$(\mathrm{n}) = \mathrm{g}(\mathrm{n}) + \mathrm{w} \cdot h(\mathrm{n}) \tag{7}$$

$$\mathrm{w} = \exp\left(\frac{1}{\alpha \mathrm{d_n} + \beta}\right) \tag{8}$$

$$d_n = \frac{max\{|x_n - x_s|, |y_n - y_s|\}}{max\{|x_g - x_s|, |y_g - y_s|\}} \tag{9}$$

Value d_n in Equation (9) is the ratio of the Chebyshev distance from the starting point of the current node to the full Chebyshev distance. Value w is the weighting coefficient, which dynamically weights the heuristic function. Values (x_s, y_s) are the horizontal and vertical coordinates of the starting point, (x_g, y_g) are the horizontal and vertical coordinates of the target point, and (x_n, y_n) are the horizontal and vertical coordinates of the current node. The weighting coefficient α (typically 5~9) is used to obtain better results, depending on the distribution of obstacles in the map. The obstacle ratio β represents the proportion of the area occupied by obstacles to the total area of the map. By weighting the heuristic function with the Chebyshev distance (Equation (8)), a balance between planning efficiency and effectiveness can be achieved; it reduced the gap between the heuristic function $h(x)$ and the actual path length, the number of nodes traversed, and eventually the search time.

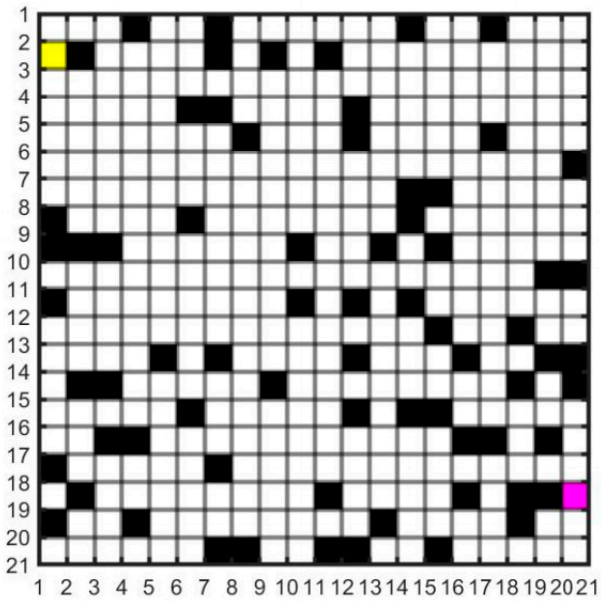


Figure 3. A virtual map generated via the grid mapping approach: the dark squares are obstacles; yellow and purple squares are the starting and end points, respectively.

The smoothing of the path is necessary for the vehicle (the Ackermann-steered mobile platform) and the position of each target point, especially the position of the vehicle at the starting and ending points on the path, which should be appropriately designed in order to minimize the total length of the path. The determination of the position of the vehicle at each point is presented in Figure 4. If the angle formed by a target vertex and its two adjacent points is less than 90° (Figure 4), the scheme of entering and exiting the opposite direction with an angular bisector position will be adopted. On the other hand, if the angle is greater than 90° (Figure 4), the one-way entry–exit method with the stop position being the vertical line of the angular bisector line will be adopted. This approach allows for smoother movement and efficient transverse of the vehicle.

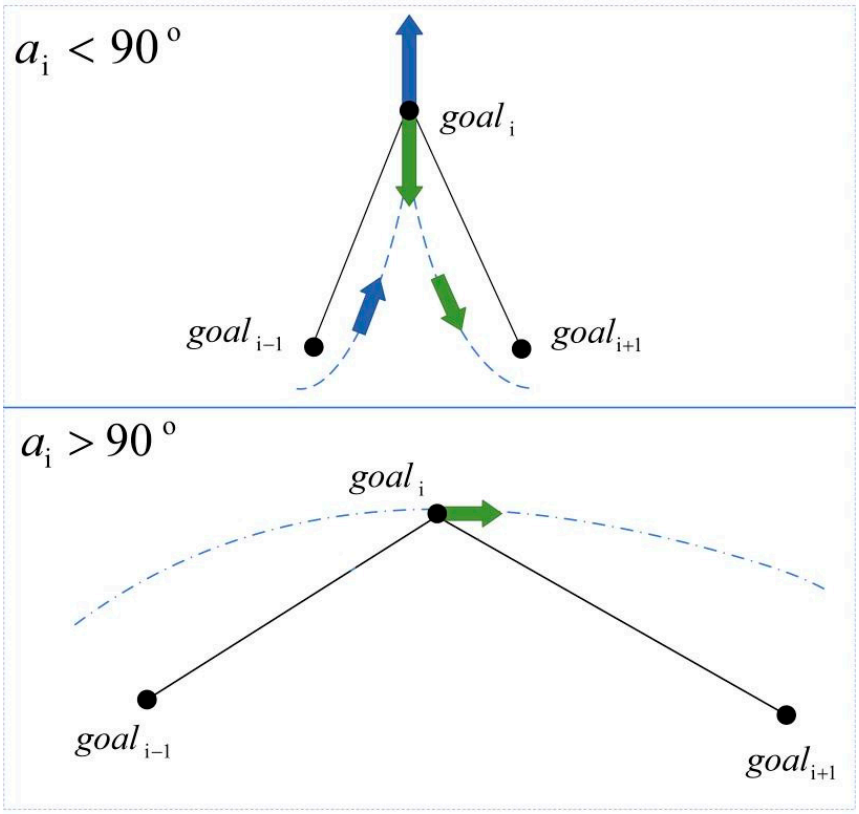


Figure 4. Robot motion path diagram in global planning.

To improve the searching efficiency and meanwhile satisfy the condition of the heuristic function, Reeds-Shepp curve was applied at the end of the global path planning process. The Reeds-Shepp curve is a cost-optimized path curve that has been proven to be optimal for nonholonomically constrained robots in a general-purpose environment free from obstacles [31]. During the global path planning, the improved hybrid A* algorithm with Chebyshev distance attempted to generate a Reeds–Shepp curve from the current node to the target node if a node was expanded during the loop search. If such a curve was generated between the two points without encountering any obstacles, the curve would be combined with the result in the closed list to generate the optimal path. The adoption of Reeds–Shepp curve significantly increased the efficiency of the algorithm.

The flowchart depicting the final global planning algorithm is presented in Figure 5. Figure 6 shows the pseudocode of the global path planning algorithm in the improved hybrid A* algorithm.

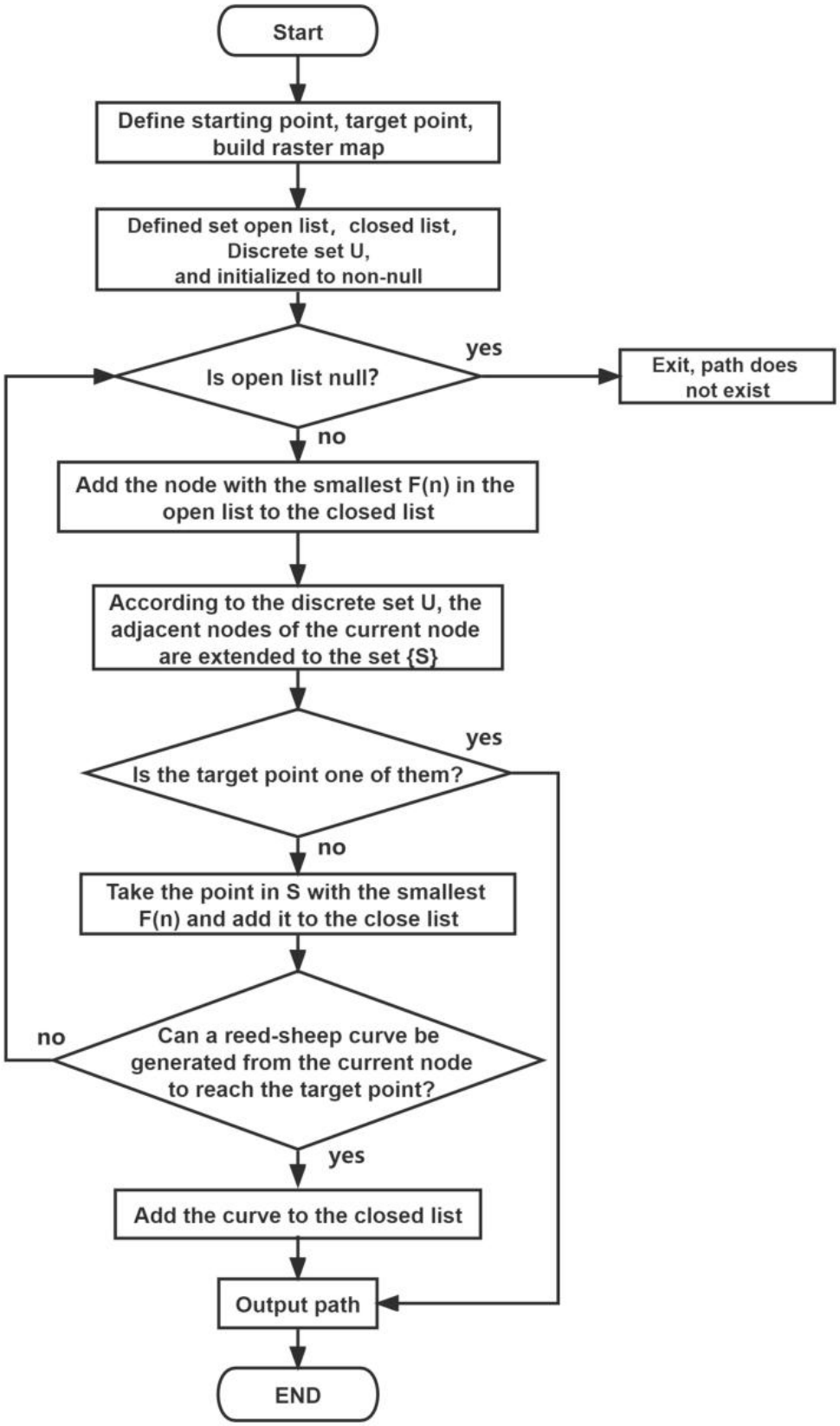


Figure 5. Flowchart of global path planning algorithm.

input:

Cumulative cost function $g(n)$, Heuristic function $h(n)$, Dynamic weighting coefficient w, Origin state S_t, Target point state S_e, Child nodes extend the discrete space set $\{U\}$

Output:

Optimal globally planned path node sequence closed list

initialization:

A matrix set is built to store all nodes to be expanded open list←$\{S_t\}$

Build a heuristic function for each node in the map {H}

Initializes the cumulative cost function for all nodes g(S_t)←{0}

Child nodes extend the discrete space set $\{U\}$←$\{u\}$&$\{w\}$

A collection of adjacent nodes $\{S\}$

Start of algorithm:

```
While openlist ≠ φ do
    The F(n) minimum cost node in the open list is displayed
    Expand adjacent nodes according to discrete space set
  If the target point belongs to a set of adjacent nodes，return closed list;
        break;
  END If
    For i=1 to size(closed list) do
      if g(S_n)=infinite
        Push node S_n into the closed list
      END If
    END For
    If S_n can be connected to S_e by a reed-sheep curve
      RES=reed-sheep(S_n, S_e, R_min)
    return {closed list+RES};break;
    END IF
  End while
Return {s}
```

Figure 6. Pseudocode of global path planning algorithm.

2.3.3. Local Path Planning Based on MPC Theory

Global path planning is generally constructed for one time at the initial of the path planning. In practice, the environment could be dynamically changed during the operation of the robot. Therefore, it is necessary to update the local path according to the changed environment and incorporate such local path planning into the total path planning pro-

cess [32]. The real-time update of the environment changes (e.g., new obstacles or changes in the existed trajectory) can be realized by the MPC-based local path planning. MPC is a specialized form of control that generates the current control action by solving an open-loop optimal control problem within a finite time horizon [33,34]. It provides a more adaptive and robust control strategy for mobile robots, ensuring safe and efficient navigation in dynamic environments. In the improved hybrid A* algorithm, the total cost $f(n)$ of a node n is expressed as $f(n) = g(n) + h(n)$, where $g(n)$ and $h(n)$ are the actual cost from the start of the path to node n and the estimated cost from node n to the goal. Specifically, $g(n)$ is the stage cost determined by the current state and controlling input, which is calculated using the following formula: $g(n) = g(\text{parent}) + c(\text{parent}, n)$, where g(parent) is the actual cost from the start of the path to the parent of node n, and c(parent, n) is the actual cost from the parent to node n. Value $h(n)$ is the terminal cost according to the predicted future state, and it is obtained by using a heuristic cost estimate function that estimates the cost from node n to the goal. As a result, it can be seen that the optimal control via MPC is directly influenced by $g(n)$ and $h(n)$. With regard to Ackermann steering chassis, the MPC is particularly suitable for local path planning because it can effectively handle the nonholonomic constraints that are unique to the chassis. As shown in Figure 7, MPC can plan a path that takes into account these constraints and also has a feedforward system that includes a prediction function, allowing for dynamic obstacle prediction during local path planning.

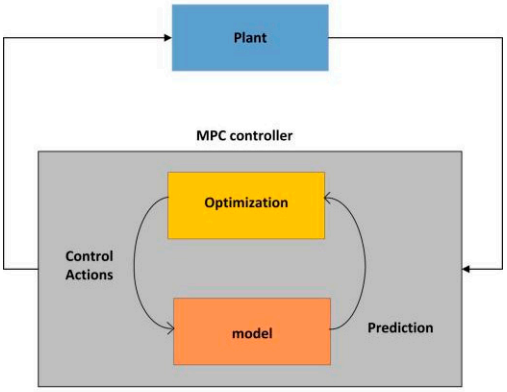


Figure 7. Schematic diagram of the MPC principle.

In the MPC strategy, the constraint conditions of dynamic obstacles are fully considered, reflecting the optimization of the robot motion state and prediction of possible collision situations between the robot and dynamic obstacles at future time steps. Adjustments are made based on these constraints, ensuring safe and efficient navigation. At a given sampling moment, the geometric visualization of the entire constraint condition of dynamic obstacles is illustrated in Figure 8.

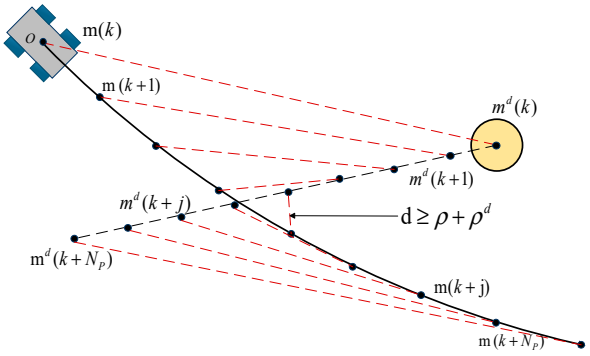


Figure 8. Local path planning diagram based on MPC.

The mathematical expression of the obstacle constraint on the robot position space $m(k)$ at the current time k can be obtained. Equations (10) and (11) show the criteria of collision elimination of a robot when confronting the static obstacle and the dynamic obstacle with constant motion state: the distance d between the center points of the two obstacles should be larger than the sum of the expansion radius in all predicted states.

$$\|m(k+j) - m^{os}\|_2 \geq r^{os} + \rho, \forall j = \{0, 1, 2, ..., N_p\} \tag{10}$$

$$\left\|m(k+j) - m^{d}(k+j)\right\|_2 \geq \rho^{d} + \rho, \forall j = \{0, 1, 2, ..., N_p\} \tag{11}$$

where r^{os} and ρ^{d} are the expansion radius for the static and dynamic obstacles, respectively, j represents a certain time, m^{os} and m^{d} represent the position space of static obstacle and dynamic obstacle, m represents the spatial position of the robot, and ρ is the expansion radius of the robot.

3. Simulation and Experiment

3.1. Simulation of Map Construction

To evaluate the effectiveness and efficiency of the improved hybrid A* algorithm, a computer simulation was carried out via MATLAB 2018b. The hardware/software environments are listed in Table 1. The following reasonable assumptions were considered in the simulation:

(1) The robot can perform turns within the reaction time, and the turning angle does not exceed its maximum turning angle while moving in the map, without considering factors such as friction and other resistances.
(2) The robot's orthographic projection area occupies one quarter of a grid. In a map with an obstacle ratio of 0.2, the value of variable α is 5, and the value of variable β is 0.2. In a map with an obstacle ratio of 0.3, the value of variable α is 6, and the value of variable β is 0.3.

These assumptions fit the definition of our model. The Chebyshev distance from start to finish is 8 m.

Grid maps with obstacle proportions of 0.2 and 0.3 (the ratio of the obstacle number to the total number of grids) was used as the comparative experiments. The simulation process is the random generation of a grid graph by MATLAB using the A* algorithm and improved algorithm for path planning. Four experiments were carried out on each map with different ratio of obstacles, and the time and length of each planned path were calculated. The grid map used in the simulation was 20 × 20, with a size of 0.5 m × 0.5 m per grid. The information of the hardware is listed in Table 1. The model of the GPU used in this study is NVIDIA GTX1650 with 896 CUDA cores and the core frequency of 1485 MHz. The SLAM and relevant machine-learning algorithms were processed via CUDA toolkit, and the processing efficiency is directly determined by the core frequency and core number of the GPU. Since GPUs have a large number of cores, which allow for better computation of multiple parallel processes, they can process multiple computations simultaneously. Therefore, the calculation running on a high-performance GPU can have a significantly shorter time compared to those running on the CPU.

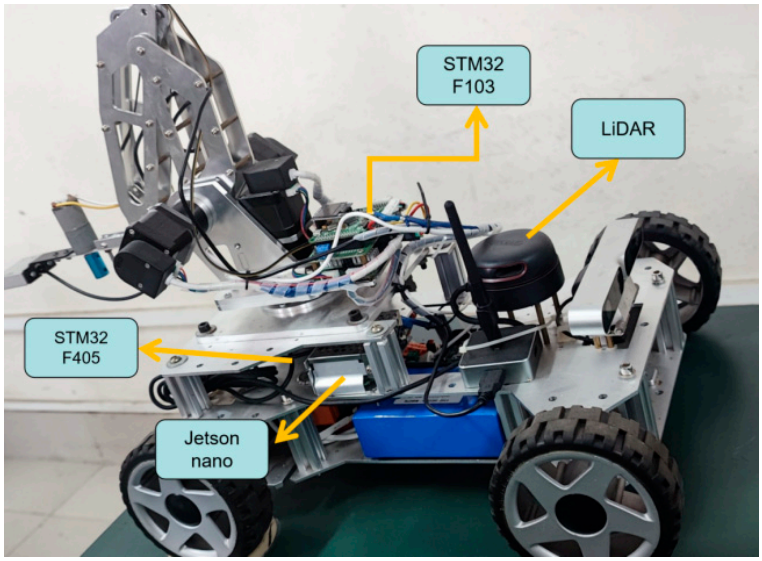
Table 1. Hardware environment for simulation.

Operating System	CPU	RAM	GPU
64 bit Windows 10	Intel(R) Core (TM) i5-9300H	8.0 GB	GTX1650

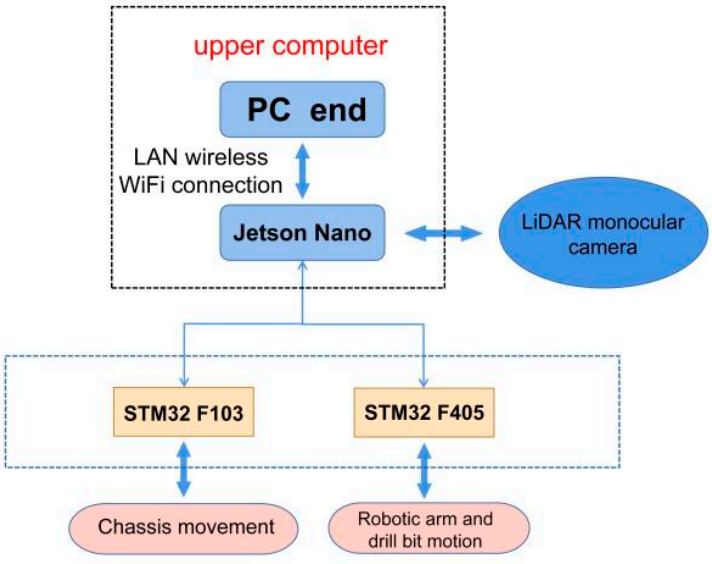
3.2. Physical Mock-Ups and the Controlling System

A robot which consisted of an Ackermann steering chassis, a three-degree-of-freedom mechanical arm, and an actuator was fabricated (Figure 9a). Specifically, the chassis was

responsible for tasks such as target navigation and path planning, while the mechanical arm performed track planning assisted with an auxiliary actuator and camera-based target positioning. The Ackermann chassis was equipped with LiDAR, a binocular camera, a control box, a manipulator, a motor, and a drive which facilitated the autonomous driving and operation of the robot. The controlling system of the autonomous robot was made up by a upper computer and a lower computer, as shown in Figure 9b. The path planning algorithm was stored and performed in the upper computer which consisted of a PC terminal with an Intel (R) Core (TM) i5-9300H CPU @ 2.30 GHz processor and a microcontroller with the Jetson Nano demo board. The PCD terminal performed decision-making and information processing, whereas the Jetson Nano controlled the radar and the camera and facilitated communication between the components. The lower computer comprised two STM32 demo boards. The STM32 F103 received commands sent by the upper computer and controlled the movement of chassis. This demo board was also responsible for information collection from odometer and responded back to the upper computer for navigation. The STM32 F405, along with the related drive, received commands from the upper computer to control the mechanical arm according to the track.



(**a**)



(**b**)

Figure 9. (**a**) The assembly of the robot. (**b**) The working scheme of the controlling system.

3.3. Navigation Experiment

The practical navigation experiment was conducted at a 10 m × 10 m testing site, and cardboard boxes and chairs were allocated within the site to imitate the obstacles (Figure 10). In the experiment, the obstacle ratio of the whole map is 0.15, so β is 0.15, α is 5, and the Chebyshev distance d_q from the starting point to the end point is 7 m.

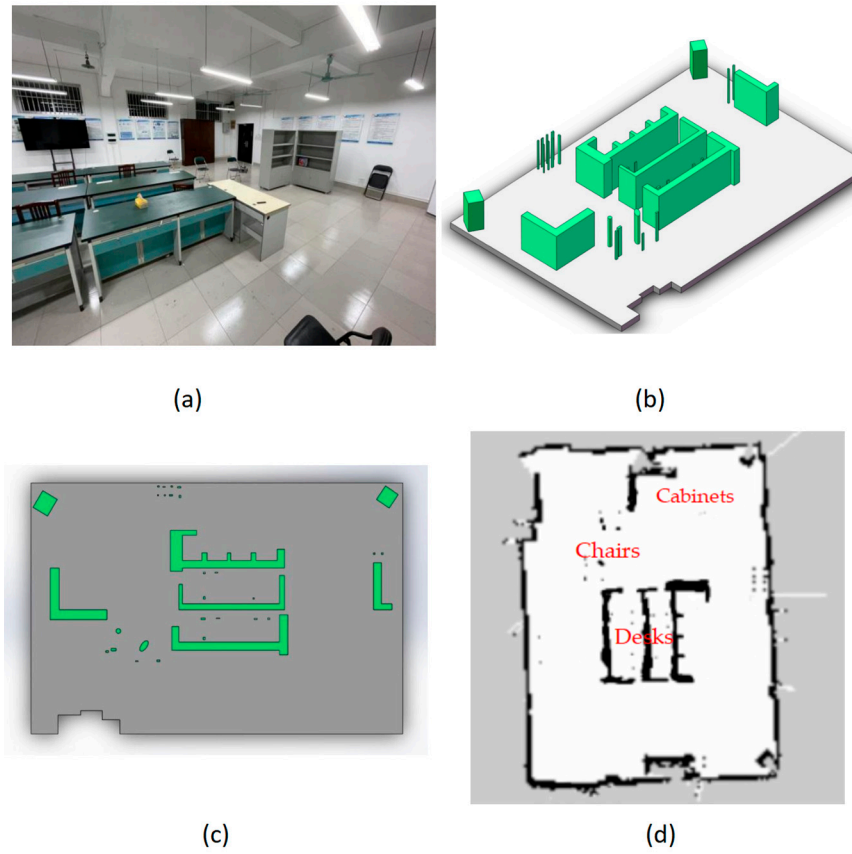


Figure 10. The practical environment. (**a**–**c**) are for navigation experiment and the constructed map. (**d**) practical environment.

We used SLAMTEC A2 radar to build the map (Table 2). The accuracy of the radar is 1 cm to 5 cm, the scanning radius is 12 m, and the scanning frequency is 5 Hz to 15 Hz. The global map was established with the data collected via LiDAR and processed by the mapping algorithm. It can be seen that the constructed map is roughly the same as the practical environment, with a relative error of about 1%. The Rviz visualization interface was launched to verify the normality of map information, robot model, LiDAR, and other relevant data. Moreover, the planned path, such as global and local paths, can be visualized in the Rviz interface to monitor the results of the tests. After the generation of the path according to the global map, various speed instructions and other relevant data such as steering angle and rotating speed of the front wheel were generated in real-time. The autonomous navigation system in the upper computer constantly received and released topics (i.e., data and instructions) in the ROS communication mode. The lower computer received relevant instructions, performed the necessary tasks, and ultimately completed the navigation task.

Table 2. Specific parameters of the radar.

Model Number	Accuracy	Scanning Radius	Scanning Frequency	Sampling Frequency	Angular Resolution
SLAMTEC A2	1–5 cm	12 m	5–15 Hz	16 k	0.9°

4. Results and Discussions

Figure 11 shows the simulation results of mapping and path planning in MATLAB. The results of the improved hybrid A* algorithm were evaluated and compared to those of the traditional A* algorithm. The obstacle ratios of the maps used in the simulation were 0.2 and 0.3, respectively. The blue lines in the figures represent the paths planned by the conventional A* algorithm, and the red lines stand for those from the improved hybrid A* algorithm. It was found that the conventional A* algorithm searched the concave obstacles by scanning the eight surrounding nodes. This increased the search time and distance and reduced the algorithm's efficiency, although it did not result in the local minimum problem. Moreover, the sharp corners of the paths generated by the conventional A* algorithm were not suitable for nonholonomically constrained Ackermann chassis. In contrast, the path generated by the improved hybrid A* algorithm was shorter and smoother. This significantly reduced the total length of the path as well as the turning radius at the boundary of the obstacles, facilitating the movement of the Ackermann chassis.

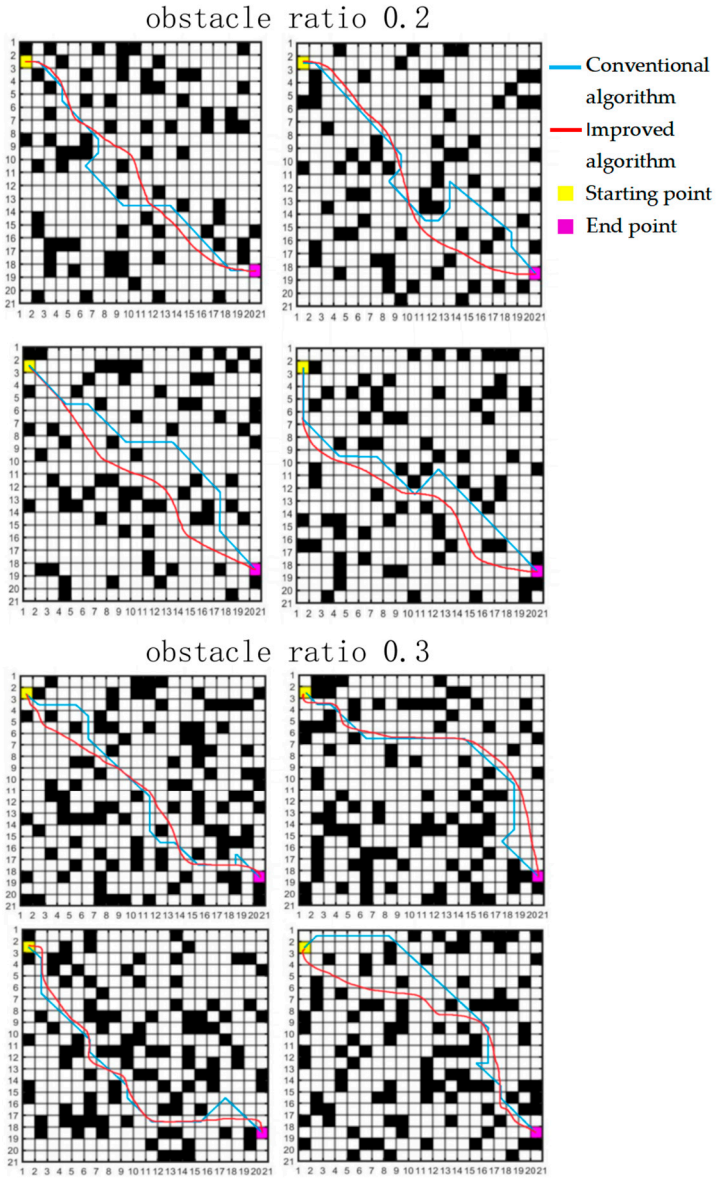


Figure 11. MATLAB simulation experimental results.

Figure 12 shows the difference between the A* algorithm and improved algorithm in the simulation experiments. It can be seen that the path charted by the A* algorithm possesses a large corner, displaying a tendency to collide with the boundary of the obstacles. This problem was caused by the algorithm's manner of node expansion in eight directions without considering motion constraints and the actual size of the robot. This makes the A* algorithm unsuitable for actual navigation. Conversely, the path planned by the improved algorithm is notably smoother with a small corner, presenting a more feasible performance than that of the conventional A* algorithm. This is because the application of the Chebyshev distance optimized the node expansion when the transverse angle was larger than 90°.

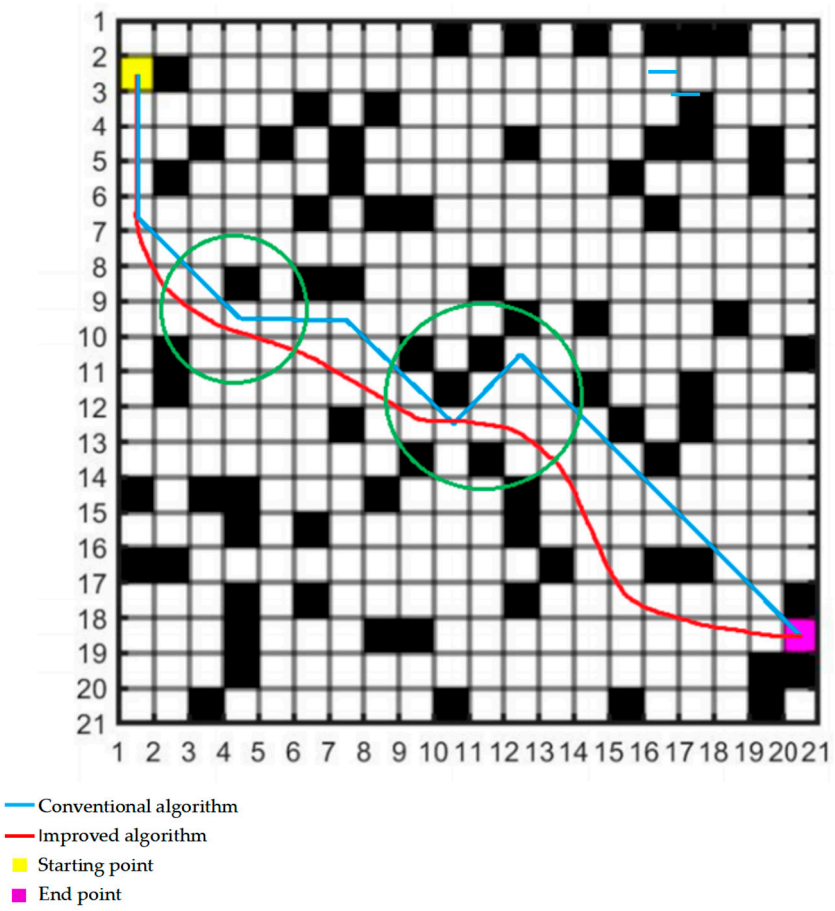


Figure 12. The difference between A* algorithm and improved algorithm in simulation experiments.

The performance of both algorithms was evaluated using a grid map at the two obstacle ratios. The comparison of the improved algorithm and the traditional algorithm is shown in Figure 13. The values of the average searching length and time consumption of the two algorithms are listed in Tables 3 and 4. The results obtained from the simulation and experiment demonstrate that the improved algorithm is advantageous in the path selection. Specifically, the average searching length with the improved hybrid A* algorithm was about 9% less at both obstacle ratios of 0.2 and 0.3. This is because the improved hybrid A* algorithm avoids the generation of nodes that were generated by the conventional A* algorithm. The smoother path generated via the improved hybrid A* algorithm contributed to the overall increase in the efficiency of the algorithm, which was reflected by the average searching time. As listed in the tables, the average search times using the improved hybrid A* algorithm were 12.62% and 24.5% when the obstacle rates were 0.3 and 0.2, respectively. This indicates the better efficiency of the improved algorithm, but the time reduction became less significant when the obstacle rate was increased.

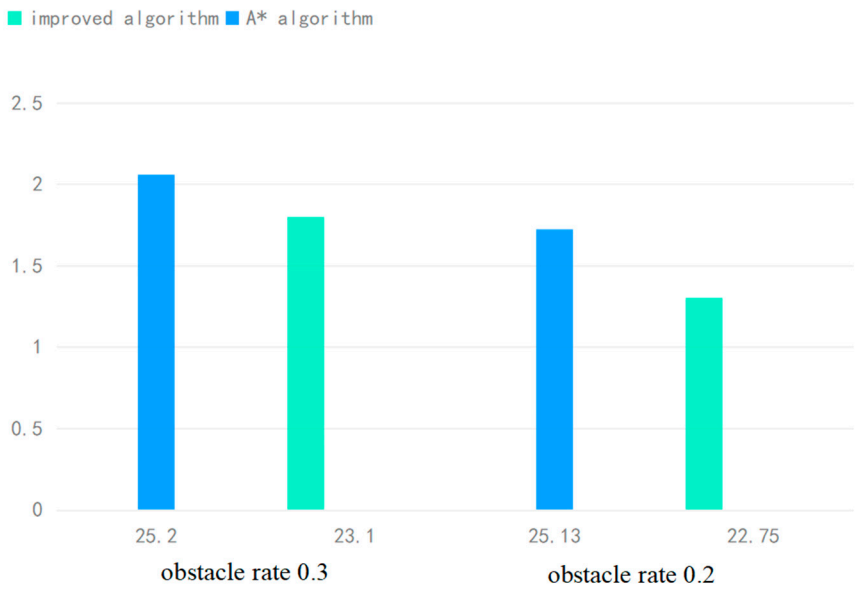


Figure 13. Comparison of the improved algorithm and the traditional algorithm.

Table 3. Comparison between the algorithms at the obstacle rate of 0.3.

Indicators	A* Algorithm	Improved Hybrid A* Algorithm
Path state	Unsmoothed	Smooth and continuous
Average searching length	25.2	23.1 (9.25% reduction)
Average searching time	2.06 s	1.8 s (12.62% reduction)

Table 4. Comparison between the algorithms at the obstacle rate of 0.2.

Indicators	A* Algorithm	Improved Hybrid A* Algorithm
Path state	Unsmoothed	Smooth and continuous
Average searching length	25.13	22.75 (9.5% reduction)
Average searching time	1.722 s	1.3 s (24.5% reduction)

Figure 14 shows a map of passable and impassable areas and obstacles. In this context, the black regions represent impassable areas and obstacles, whereas the gray regions signify passable areas. The starting point and end point for the comparison experiment remain the same. The path planned by the A* algorithm is shown in Figure 14a, while the path planned by the improved hybrid A* algorithm is shown in Figure 14b. It is evident from the image that the improved hybrid A* algorithm effectively circumvents the robot's entry into concave areas while searching for the optimal path. This approach results in a shorter path length, smoother trajectory, and reduced likelihood of collisions. In practical navigating tests, the average time of search was reduced by 18~24%, and the average length of search was reduced by 10.3~12%. Accounting for the positioning error of the robot, the present study suggests that the improved algorithm generates superior paths compared to that of the A* algorithm.

The improved algorithm uses the Chebyshev distance to weight the heuristic function, and under the guidance of global programming; it avoids the concave area of obstacles, improves the redundancy and inefficiency of the traditional A* algorithm, and can quickly generate an optimal curve in the early and middle stages.

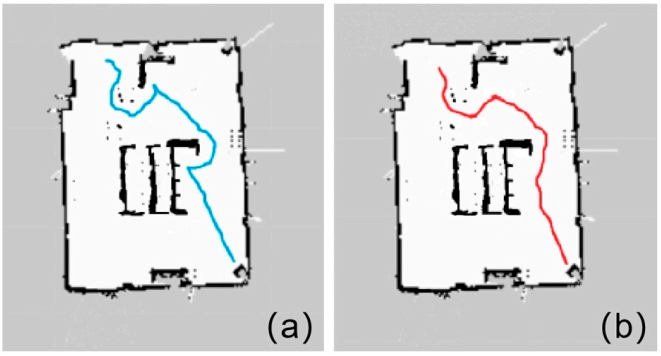


Figure 14. Path generated via different algorithms in the practical navigation experiment: (**a**) conventional A* algorithm and (**b**) improved hybrid A* algorithm.

5. Conclusions and Further Work

The nonholonomic robot with autonomous navigation was developed in this paper. The improved hybrid A* algorithm with Chebyshev weighting was applied in the path planning process to increase the efficiency of the navigation. The development of the self-path planning method was conducted in the following processes: (1) the construction of maps via the Robot Operating System (ROS), (2) the self-positioning of the robot via the adaptive Monte Carlo (AMCL) algorithm, (3) the global path planning via the improved hybrid A* algorithm, and (4) the local path planning via the model predictive control (MPC) strategy. By testing the performance of the algorithm in the simulation and practical environments, it was found that the improved hybrid A* algorithm outperformed the conventional A* algorithm in both search time and search length. These improvements make it an ideal choice for mobile chassis with nonholonomic constraints. Compared with the conventional A* algorithm, the average search time was reduced by 12.62~24.5%, and the average search length was reduced by 9.25~12% at different obstacle rates. In the practical navigating tests, the average time of search was reduced by 18~24%, and the average length of search was reduced by 10.3~12%, while the overall path was smoother. These results demonstrate the superior performance of the improved algorithm in various working environments.

Consideration for moving obstacles is an inevitable aspect of future work. With the evolution and increasing accessibility of autonomous navigation technologies, the rapid detection and avoidance of mobile objects become an imperative concern [35]. Future research focus will be on the development of new sensors, construction of more accurate obstacle-prediction models, and implementation of faster path-planning algorithms. Furthermore, applying artificial intelligence and deep learning in navigation algorithms can result in more precise prediction and avoidance of moving obstacles. The widespread implementation of these technologies will advance the maturity and expansion of robotics.

Author Contributions: Conceptualization, Z.C. and G.L.; methodology, Z.C.; validation, G.L.; investigation, Z.C. and Q.H.; resources, Z.Y. and Q.W.; data curation, Q.H.; writing—original draft, Z.C.; writing—review & editing, G.L. and S.D.; visualization, Z.Y., Q.W. and G.L.; supervision, G.L. and S.D.; funding acquisition, S.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

Nomenclature

Symbol	Definition
$f(n)$	Priority of the next node
$h(n)$	Estimated cost from the next node to the end
$g(n)$	Cost from the current node to the start
w_{long}	Long-term observation average function
w_{short}	Short-term observation average function
a_{long}	Average weight coefficient of the long-term observation information
a_{short}	Average weight coefficient of the short-term observation information
w_{avg}	Long-term observation average function
w_i	Observation average function
w	Weighting coefficient
α	Distribution coefficient
β	Obstacle ratio
d_n	Ratio of Chebyshev distances
d_q	Chebyshev distance
x_n, y_n	Horizontal and vertical coordinates of the current node
x_g, y_g	Horizontal and vertical coordinates of the target point
x_s, y_s	Horizontal and vertical coordinates of the starting point
r^{os}	Expansion radius of static obstacle
ρ^d	Expansion radius of dynamic obstacle
m^{os}	Position space of static obstacle
m^d	Position space of dynamic obstacle
m	Spatial position of the robot
ρ	Expansion radius of the robot

References

1. Gul, F.; Rahiman, W.; Alhady, S.S.N. A comprehensive study for robot navigation techniques. *Cogent Eng.* **2019**, *6*, 1632046. [CrossRef]
2. He, X.; Yang, H.; Hu, Z.; Lv, C. Robust Lane Change Decision Making for Autonomous Vehicles: An Observation Adversarial Reinforcement Learning Approach. *IEEE Trans. Intell. Veh.* **2023**, *8*, 184–193. [CrossRef]
3. Behringer, R.; Muller, N. Autonomous road vehicle guidance from Autobahnen to narrow curves. *IEEE Trans. Robot. Autom.* **1998**, *14*, 810–815. [CrossRef]
4. Skog, I.; Handel, P. In-Car Positioning and Navigation Technologies—A Survey. *IEEE Trans. Intell. Transp. Syst.* **2009**, *10*, 4–21. [CrossRef]
5. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
6. Cui, R.; Li, Y.; Yan, W. Mutual Information-Based Multi-AUV Path Planning for Scalar Field Sampling Using Multidimensional RRT*. *IEEE Trans. Syst. Man Cybern. Syst.* **2016**, *46*, 993–1004. [CrossRef]
7. Wang, J.; Chi, G.; Li, C.; Wang, C.; Meng, M. Neural RRT*: Learning-Based Optimal Path Planning. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1748–1758. [CrossRef]
8. Yao, Y.F.; He, N.; Zhang, M. The Application of Internet of Things in Robot Route Planning Based on Multisource Information Fusion. *Comput. Intell. Neurosci.* **2022**, *2022*, 1707259. [CrossRef]
9. Jiang, J.R.; Huang, H.W.; Liao, J.H.; Chen, S.Y. Extending Dijkstra's Shortest Path Algorithm for Software Defined Networking. In Proceedings of the 2014 16th Asia-Pacific Network Operations and Management Symposium (APNOMS), Hsinchu, Taiwan, 17–19 September 2014; IEEE: Piscataway, NJ, USA, 2014.
10. Chang, J.R.; Jheng, Y.H.; Chang, C.H.; Lo, C.H. An Efficient Algorithm for Vehicle Guidance Combining Dijkstra and A* Algorithm with Fuzzy Inference Theory. *J. Internet Technol.* **2015**, *16*, 189–200. [CrossRef]
11. Yang, D.; Xu, B.; Rao, K.; Sheng, W. Passive Infrared (PIR)-Based Indoor Position Tracking for Smart Homes Using Accessibility Maps and A-Star Algorithm. *Sensors* **2018**, *18*, 332. [CrossRef]
12. Harabor, D.; Grastien, A. Online Graph Pruning for Pathfinding on Grid Maps. In Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 7–11 August 2011; Volume 2.
13. Li, G.; Chou, W. Path planning for mobile robot using self-adaptive learning particle swarm optimization. *Sci. China Inf. Sci.* **2017**, *61*, 052204. [CrossRef]
14. Li, J.C.; Ran, M.P.; Wang, H.; Xie, L.H. MPC-based Unified Trajectory Planning and Tracking Control Approach for Automated Guided Vehicles. In *Proceedings of the 2019 IEEE 15th International Conference on Control and Automation (ICCA), Edinburgh, UK, 16–19 July 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 374–380.

15. Fox, D.; Burgard, W.; Thrun, S. The Dynamic Window Approach to Collision Avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33. [CrossRef]
16. Wu, J.; Ma, X.; Peng, T.; Wang, H. An Improved Timed Elastic Band (TEB) Algorithm of Autonomous Ground Vehicle (AGV) in Complex Environment. *Sensors* **2021**, *21*, 8312. [CrossRef] [PubMed]
17. Rasekhipour, Y.; Khajepour, A.; Chen, S.K.; Litkouhi, B. A Potential Field-Based Model Predictive Path-Planning Controller for Autonomous Road Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1255–1267. [CrossRef]
18. Ji, J.; Wang, H.; Ren, Y. *Path Planning and Tracking for Vehicle Collision Avoidance in Lateral and Longitudinal Motion Directions*; Synthesis Lectures on Advances in Automotive Technology; Springer: Cham, Switzerland, 2020; Volume 4, pp. 1–152. [CrossRef]
19. Gan, Y.; Zhang, B.; Ke, C.; Zhu, X.F.; He, W.M.; Ihara, T. Research on Robot Motion Planning Based on RRT Algorithm with Nonholonomic Constraints. *Neural Process. Lett.* **2021**, *53*, 3011–3029. [CrossRef]
20. Zhang, Z.Y.; Li, Y.; Wu, P.; Liu, Z.C.; Xiao, J.J.; Zhu, W.H. An INS-aided MASS autonomous navigation algorithm considering virtual motion constraints and the leeway and drift angle. *Ocean Eng.* **2023**, *272*, 113790. [CrossRef]
21. Zhang, Y.F.; Wen, Y.Y.; Tu, H.Y. A Method for Ship Route Planning Fusing the Ant Colony Algorithm and the A* Search Algorithm. *IEEE Access* **2023**, *11*, 15109–15118. [CrossRef]
22. Worthmann, K.; Mehrez, M.W.; Zanon, M.; Mann, G.K.I.; Gosine, R.G.; Diehl, M. Model Predictive Control of Nonholonomic Mobile Robots without Stabilizing Constraints and Costs. *IEEE Trans. Control Syst. Technol.* **2016**, *24*, 1394–1406. [CrossRef]
23. Dang, C.V.; Ahn, H.; Lee, D.S.; Lee, S.C. A Path Planning Method Based on Theta-star Search for Non-Holonomic Robots. In Proceedings of the 2022 Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCIS&ISIS), Ise, Japan, 29 November–2 December 2022; pp. 1–6.
24. Li, J.; Ran, M.; Xie, L. Efficient Trajectory Planning for Multiple Non-Holonomic Mobile Robots via Prioritized Trajectory Optimization. *IEEE Robot. Autom. Lett.* **2021**, *6*, 405–412. [CrossRef]
25. Johnson, J.J.; Li, L.; Liu, F.; Qureshi, A.H.; Yip, M.C. Dynamically Constrained Motion Planning Networks for Non-Holonomic Robots. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 6937–6943.
26. Gonon, D.J.; Paez-Granados, D.; Billard, A. Reactive Navigation in Crowds for Non-Holonomic Robots with Convex Bounding Shape. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4728–4735. [CrossRef]
27. De Oliveira, A.L.; Binder, B.J. Discrete Manhattan and Chebyshev pair correlation functions in k dimensions. *Phys. Rev. E* **2020**, *102*, 012130. [CrossRef] [PubMed]
28. Smith, R.; Cheeseman, P. On the Representation and Estimation of Spatial Uncertainty. *Int. J. Robot. Res.* **1987**, *5*, 56–68. [CrossRef]
29. Kang, Y.; Song, Y.; Song, Y.; Yan, D.; Li, D. Square-Root Cubature Kalman Filter and Its Application to SLAM of an Mobile Robot. *Robot* **2013**, *35*, 186–193. [CrossRef]
30. Dolgov, D.; Thrun, S.; Montemerlo, M.; Diebel, J. Practical search techniques in path planning for autonomous driving. *Int. Symp. Comb. Search SoCS* **2008**, *1001*, 18–80.
31. Reeds, J.A.; Shepp, L.A. Optimal Paths for a Car That Goes Both Forwards and Backwards. *Pac. J. Math.* **1990**, *145*, 367–393. [CrossRef]
32. He, X.; Liu, Y.; Lv, C.; Ji, X.; Liu, Y. Emergency steering control of autonomous vehicle for collision avoidance and stabilisation. *Veh. Syst. Dyn.* **2018**, *57*, 1163–1187. [CrossRef]
33. Shim, T.; Adireddy, G.; Yuan, H. Autonomous vehicle collision avoidance system using path planning and model-predictive-control-based active front steering and wheel torque control. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2012**, *226*, 767–778. [CrossRef]
34. Kayacan, E.; Kayacan, E.; Ramon, H.; Saeys, W. Distributed nonlinear model predictive control of an autonomous tractor–trailer system. *Mechatronics* **2014**, *24*, 926–933. [CrossRef]
35. He, X.; Lou, B.; Yang, H.; Lv, C. Robust Decision Making for Autonomous Vehicles at Highway On-Ramps: A Constrained Adversarial Reinforcement Learning Approach. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 4103–4113. [CrossRef]