Mechatronics and Automation Technology
J.-Y. Xu (Ed.)

doi:10.3233/ATDE231128

A Local Path Planning Method of Multi-AGV Systems

Yi LIU[a], Haiyan YAO[a], Zhipeng TANG[a], Qiang GUO[a], Xufeng ZHANG[a] and Haofeng SHEN[b,1]

[a] *Hangzhou Power Equipment Manufacturing Co., Ltd., Yuhang Qunli Complete Electrical Manufacturing Branch, Hangzhou 311100, China*

[b] *School of Automation, Hangzhou Dianzi University, Hangzhou 310018, China*

Abstract. The dynamic window approach (DWA) algorithm is a local path planning algorithm based on velocity space sampling. Compared with the traditional DWA algorithm, this study employs visual sensors to construct a two-dimensional map, the relative distance and relative angle are obtained by transmitting each other's speed and position information through the mutual communication between AGVs (Automated Guided Vehicles). The traditional evaluation function is improved to adapt to the local path planning in the case of multiple AGVs, and some special cases are processed in detail. Finally, the path planned by the DWA algorithm is optimized and tracked via designing model predictive control (MPC) algorithms for the AGVs.

Keywords. Multiple AGV; Local path planning; Dynamic window approach; Model predictive control.

1. Introduction

In recent years, with the continuous development of science and technology, autonomous mobile robot technology has gradually become mature and reliable, and has been widely used in manufacturing, industry, agriculture, warehousing and transportation, medical assistance, security inspection and other fields [1]. As autonomous mobile robots perform a wide variety of tasks in a variety of environments, the increasing performance requirements pose challenges for autonomous mobile robots to adapt to increasingly complex environments and task collaboration capabilities. In order to ensure that the robot can avoid obstacles while maintaining reliability, completeness, optimal energy consumption, optimal speed and other performance indicators to complete tasks in strange environments, the path planning capabilities of autonomous mobile robots need to be further improved and developed [2].

During the development of AGV path planning, numerous scholars have proposed many methods, such as Dijkstra [3], A* [4], Ant colony optimization (ACO) [5]. In order to cope with more complex and changeable environment, the traditional AGV path planning algorithm using global environment information cannot meet the needs. Fox et

[1] Corresponding Author, Haofeng SHEN, School of Automation, Hangzhou Dianzi University, Hangzhou 310018, China, Email: shf@hdu.edu.cn.

al put forth dynamic window approach (DWA) in 1997, which transformed the conventional path planning to sampling of velocity space [6]. It considers the dynamics of the robot to determine the best sampling speed, and controls the robot movement in real time with the best linear and angular velocity. Although the DWA algorithm can only use part of the environment information for local path planning, it is prone to local oscillation or local minimum problems [7].

To address some of the shortcomings and deficiencies in the DWA algorithm, many researchers have combined the DWA algorithm with other algorithms to improve its performance. The DWA algorithm with Virtual Manipulators (DWV) was proposed in [8]. This method utilizes virtual manipulators (VM) and environmental information to generate velocities for reflective motion, enabling the algorithm to generate obstacle-avoidable paths that include non-straight line and non-arc paths in environments with dynamic obstacles. The DWA algorithm and the Dijkstra algorithm are frequently combined. In reference [9], the successful integration of these two algorithms allows intelligent obstacle avoidance vehicles to navigate from the planned initial position, avoiding obstacles, and reaching the specified destination. Furthermore, in reference [10], by utilizing the Dijkstra algorithm and the DWA method to plan the shortest and conflict-free path, it significantly addresses the path planning problem in small-scale multi-AGV systems. By combining the excellent global path search capability of Ant Colony Optimization (ACO) with the advantages of the DWA algorithm in local obstacle avoidance, the algorithm in reference [11] enhances the navigation, exploration, and dynamic obstacle avoidance capabilities of robots in unknown and complex dynamic environments. References [12] and [13] applied the DWA algorithm in the A-star algorithm to achieve obstacle avoidance, reducing the average path length and enhancing the algorithm's performance in complex environments. And in 2019 Zeya Zhu proposed a novel method based on fusion of A* algorithm and DWA algorithm, so as to improve the global optimization and real-time obstacle avoidance ability of mobile robots[14]. To enhance the performance of the DWA algorithm when dealing with dynamic obstacles, reference [15] introduces two improved algorithms based on the classical DWA algorithm to handle dynamic obstacles. These algorithms are known as the Dynamic Window for Dynamic Obstacles (DW4DO) and the Dynamic Window for Dynamic Obstacle Trees (DW4DOT).

In order to ensure the accuracy of the DWA algorithm, it is necessary to construct an accurate two-dimensional map to obtain the surrounding environment information. In the past, for the construction of environmental maps, visual sensors and inertial sensors were usually alone or combined, this technology is called visual simultaneous localization and mapping (SLAM) systems. On this basis, Carlos Campos [16] proposed ORB-SLAM3, the first system able to perform visual, visual-inertial and multimap SLAM with monocular, stereo and RGB-D cameras, using pin-hole and fisheye lens models. ORB-SLAM3 is mainly composed of three parts [17]: tracking thread, local mapping thread and loop and map merging thread. The sparse point cloud map obtained by ORB-SLAM3 is not able to provide navigation for omni-directional robots because of the small number and discrete distribution of feature points [18]. Therefore, it is necessary to convert a sparse point map into a semi-dense map by adding depth information and overlaying each frame of image. Semi-dense point cloud maps require a lot of storage space to store data, and with the passage of time, the number of local environment map constructions is also increasing, resulting in the space required for storing maps will continue to grow [19]. More importantly, the semi-dense point cloud map cannot store the interconnection information between various spatial points, and is not directly applicable to path planning and navigation. Therefore, this paper employs the approach proposed in references [20] and [21], based on octrees and probabilistic occupancy estimation, to model octree maps, addressing the shortcomings of semi-dense maps and constructing two-dimensional maps.

To make the path generated by the DWA algorithm smoother and more controllable, this paper follows the approach outlined in references [22-24] and utilizes the Model Predictive Control algorithm to take the optimal trajectory as a reference input. By controlling the linear velocity and angular velocity, it optimizes the motion posture of the AGV.

Most of the current DWA algorithms can only deal with the path planning problem of a single AGV. In view of the above problems, this paper proposes an improved DWA algorithm. Main contributions of this paper are the following three aspects:

1) By converting semi-dense maps to octree maps, the problems of information redundancy and storage difficulties are solved. Next, the octree map is flattened and transformed into a two-dimensional map suitable for AGV path planning

2) An improved DWA algorithm with a new evaluation function is proposed to deal with the complex working environment in which multiple AGVs perform tasks at the same time. New constraints are added to avoid collisions, and special cases that may occur are handled.

3) Combining the DWA algorithm with the MPC algorithm, the MPC algorithm optimizes the linear velocity and angular velocity of each local motion path planned by the DWA algorithm, so that the AGV can track the planned path as much as possible with the minimum control input.

2. Traditional Dynamic Window Approach Algorithm

2.1. The Robot Motion Model

The motion model of AGV can be expressed as

$$x_t = f(x_{t-1}, u_t) + \xi_t \tag{1}$$

where $f(x_{t-1}, u_t)$ represents the state transition function of the system under the control signal, ξ_t indicates system noise.

Build the dynamic model: assuming that AGV drive wheel radius is r, the distance between the two driving wheels is L ,the sampling period is ΔT , the number of revolutions of the motor is n which is calculated by the number of pulses of the photoelectric encoder and the reduction ratio of the reducer within the time ΔT .The distance traveled by the left wheel and the right wheel are s_l and s_r , the average speeds of the left and right wheels are v_l and v_r .

The distance traveled by the wheel is:

$$d = 2\pi r * n \tag{2}$$

The average linear velocity of the AGV is:

$$v = \frac{v_l + v_r}{2} \tag{3}$$

Assuming that the steering angle of the AGV turns $\Delta\theta$ within the time ΔT, and ΔT is small enough, then the angle of rotation $\Delta\theta$ can be approximated:

$$\Delta\theta \approx \sin\theta = \frac{s_l - s_r}{L} \tag{4}$$

Therefore, the angular velocity ω of the AGV around its trajectory rotation center is:

$$\omega = \frac{\Delta\theta}{\Delta T} = \frac{\mathrm{v}_l - v_r}{L} \tag{5}$$

Through the linear velocity v and angular velocity w of the AGV, the trajectory rotation radius R of the AGV can be obtained:

$$R = \frac{v}{\omega} = \frac{L*(v_l + v_r)}{2(v_l - v_r)} \tag{6}$$

Assume that the pose of the AGV at time $t-1$ is:

$$X_{t-1} = [x_{t-1}, y_{t-1}, \theta_{t-1}]^T \tag{7}$$

Then the pose of the AGV at time t can be obtained as:

$$X_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + v\Delta T\cos(\theta_{t-1}) \\ y_{t-1} + v\Delta T\sin(\theta_{t-1}) \\ \theta_{t-1} + \omega\Delta T \end{bmatrix} \tag{8}$$

2.2. Add Constraints Related to Robot

In practice, AGV is constrained by its own physical structure, and there is a limit to the range of speed. It is defined V_m as the speed set of the maximum speed and the minimum speed of the AGV.

$$V_m = \{(v,\omega) \mid v \in [v_{\min}, v_{\max}], \omega \in [\omega_{\min}, \omega_{\max}]\} \tag{9}$$

At the same time, due to the limited torque of motor tools such as motors, the acceleration of AGV is also limited. Let t be the time during which the acceleration v' and the angular acceleration ω' act, v_c and ω_c represent the current linear velocity and current angular velocity of the AGV.

Define V_d as the set of velocities that can be achieved within a time interval t.

$$V_d = \{(v,\omega) \mid v \in [v_c - v' \cdot t, v_c + v' \cdot t] \wedge \omega \in [\omega_c - \omega' \cdot t, \omega_c + \omega' \cdot t]\} \tag{10}$$

On the actual road ahead, there are many obstacles, and the closer obstacles have certain restrictions on the rotational acceleration and translation speed of the AGV. $dist(v,\omega)$ indicates the trajectory corresponding to the speed, that is the closest distance of an arc to the obstacle. $v_b{}'$ and $w_b{}'$ are the linear velocity braking acceleration and the angular velocity braking acceleration. The speed set V_a is defined as a feasible speed set whose distance to the obstacle is less than the braking distance after discarding the simulated trajectory.

$$V_a = \{(v,\omega) \mid v \le \sqrt{2 \cdot dist(v,\omega) \cdot v_b{}'} \wedge \omega \le \sqrt{2 \cdot dist(v,\omega) \cdot \omega_b{}'}\} \tag{11}$$

Define V_r as the feasible region of the velocity dynamic window, that is, the feasible velocity set.

$$V_r = V_m \cap V_d \cap V_a \tag{12}$$

2.3. Establish Evaluation Function

The traditional DWA evaluation function consists of three parts:

$$G_1(v,\omega) = \sigma(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot vel(v,\omega)) \tag{13}$$

where $heading(v,\omega)$ represents the direction angle evaluation function, which is used to characterize the degree to which the current direction of the AGV is aligned with the target direction. $dist(v,\omega)$ denotes the gap evaluation function, which is used to characterize the distance between the AGV and the closest obstacle on the current trajectory. If there is no obstacle on the current trajectory or the distance from the obstacle is greater than the set safety distance, set a larger constant value D. $vel(v,\omega)$ is the speed evaluation function, which is used to characterize the speed of the AGV on the corresponding trajectory. α, β, γ are the weighting coefficients of the three evaluation functions, which are usually constants in traditional DWA algorithms. σ represents the normalization coefficient used to eliminate the influence of different measurement units and smooth the trajectory, usually by dividing each item of the current trajectory by the sum of each item of all trajectories.

Different speed combinations are calculated by the evaluation function to obtain the corresponding evaluation values. The larger the evaluation value is, the better the trajectory performance of the speed combination is. Therefore, the sampling velocity combination trajectory with the largest evaluation value in the feasible velocity set is selected as the optimal path for local path planning.

3. 2D Map Construction

The octree map is a compressible and flexible transformation of the three-dimensional map representation, each node in the octree is called a voxel, which is the space contained in the cube. The octree is equivalent to the model of the binary tree in the data structure. It uses the entire space as the root node, and then continuously divides it. Each parent node can be divided into eight child nodes, the child nodes are recursively split again until the set octree resolution ratio is reached. The degree of occupation of the child nodes in the octree is represented by the probability, and the update process of the occupation probability is as follows:

$$\begin{cases} L(n \mid z_{1:h}) = L(n \mid z_{1:h-1}) + L(n \mid z_h) \\ L(n) = \log(\dfrac{P(n)}{1-P(n)}) \end{cases} \tag{14}$$

where $L(n \mid z_{1:h})$ is the logarithm of the occupancy probability of a voxel from start to time h , z_h denotes the occupancy probability of the n-th child node through the observation at time h , $P(n)$ represents the prior probability, unknown area is 0.5.

Although the octree map can realize the path and planning of the mobile robot, the three-dimensional octree map has a relatively large amount of calculation and poor real-time performance compared with the two-dimensional map. The redundant 3D information is not important for the mobile robot to avoid obstacles and complete tasks in the 2D plane. Therefore, the octree map is transformed into a two-dimensional map through a three-dimensional oblique projection transformation. The principle of oblique projection transformation is given in figure 1.

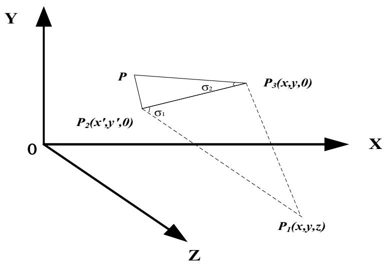

Figure 1. The principle of oblique projection transformation

The point P_1 is the coordinate of a voxel in the octree map, the point P_2 is the two-dimensional map coordinate obtained after oblique projection, and the point P_3 is the coordinate of the voxel after orthogonal projection.

Projected Ground Equation:

$$\begin{cases} L = z\cot\sigma_1 \\ x' = x - L\cos\sigma_2 = x - z\cot\sigma_1\cos\sigma_2 \\ y' = y - L\sin\sigma_2 = z - z\cot\sigma_1\sin\sigma_2 \end{cases} \tag{15}$$

where σ_1 is the angle between P_1, P_2, P_3 and σ_2 is the angle between P $, P_3, P_2$.

According to the projected ground equation, the voxel points in the octree map within a certain range from the ground are projected to a two-dimensional plane by oblique projection transformation. In the process of oblique projection, the occupancy state of the generated map points is updated in real time, and the final two-dimensional map of the local environment is obtained.

4. Evaluation Function Improvement

In order to make the dynamic window algorithm applicable to more complex environments and multiple AGVs. It is necessary to add a function about the distance from other AGVs to the evaluation function so that the AGV can stay away from other AGVs during driving to avoid collisions.

Assume that the effective perception radius of the AGV's vision sensor is r_s and the communication radius of the wireless communication module is r_c , $r_c > 2r_s$, the relative distance between the current AGV and other AGVs is d_s and the relative angle is θ_s.

The evaluation function of the improved dynamic window algorithm is:

$$G_2(v,\omega)=\sigma(\alpha\cdot heading(v,\omega)+\beta\cdot dist(v,\omega)+\gamma\cdot vel(v,\omega)+\lambda\cdot hide(v,\omega)) \qquad (16)$$

where $hide(v,\omega)$ represents the avoidance evaluation function, which is used to characterize the distance between the current AGV and other AGVs, λ is the weighting coefficient.

In the process of AGV driving, if there is no AGV within the wireless communication range, that is, $d_s > r_c$, it means that there is no possibility of conflict in the surrounding environment of the current AGV, the $hide(v,\omega)$ function does not work, and is set to a larger constant H.

When other AGVs enter the communication range, but have not been detected by the visual sensor, that is, $r_s < d_s < r_c$, it means that there are AGVs in the surrounding environment of the current AGV, and there is a possibility of conflict. At this time, the hide function will come into play. This function obtains the relative distance d_s and relative angle θ_s of other AGVs and the current AGV through the sensor. From this two information, the accurate position of other AGVs relative to the AGV can be obtained. Because the time interval is very small, in order to simplify the calculation, it can be considered that each trajectory is approximate to a straight line, so as to calculate the distance between the current trajectory and other AGVs. After all distances are summed, the larger the value, the farther away from other AGVs, and the higher the evaluation score, which also requires normalization.

$hide(v,\omega)$ function is defined as:

$$hide(v,\omega)=\sum_{i=1}^{N}\sqrt{d_{si}{}^2+(v_c t)^2-2d_{si}v_c t\cos\left|\theta-\theta_{si}\right|} \qquad (17)$$

where N represents the number of other AGVs within the current AGV threshold r_c, d_{si} and θ_{si} is the relative distance and angle of the ith AGV and the current AGV, θ represents the angle of rotation of the current AGV, and v_c denotes the forward speed of the current AGV.

If the distance between AGVs reaches or is even smaller than the perception range r_s of the visual sensor, that is, $d_s < r_s$, it means that there is a risk of conflict, and the soft constraint of the $hide(v,\omega)$ function to keep the AGVs away is not enough to eliminate this risk. Therefore, at this time, AGVs need to communicate their respective motion states, such as speed, direction, and their respective actual achievable speed sets V_d through wireless communication modules.

Using the actual achievable speed set V_d without using the feasible speed dynamic window set V_r is to prevent the sudden detection of an obstacle from changing the feasible set of the speed dynamic window, leaving room for AGV calculation and avoidance.

After the current AGV obtains the actual reachable speed set V_d of other AGVs and the sensor obtains the relative distance d_s and relative angle information θ_s, the current AGV can calculate the area that other AGVs can reach at the next moment, and this area can be approximated into a triangle Obstacle area. The current AGV compares the triangular area with the feasible set of its own speed dynamic window, discards the speed sampling of the overlapping area, and selects a path from the remaining safe area to avoid collision and conflict between AGVs.

In the face of some special circumstances that may arise, additional responses and adjustments are required.

When there are multiple AGV robots passing through the intersection, because the surrounding terrain limits the change of angular velocity, the feasible set of speed dynamic windows of the two may lead to large-scale overlap, which is not conducive to staggered passing. When this happens, AGVs with lower task priorities will slow down or even stop themselves, thereby narrowing the feasible set of speed dynamic windows, allowing AGVs with higher task priorities to pass quickly.

When two AGVs are facing each other on a narrow straight road, if the width of the straight road cannot allow the two vehicles to pass in parallel, congestion will occur, resulting in neither vehicle being able to pass. When there is an impassable area ahead, the AGV with lower task priority will turn in place and drive out of the impassable area, while the AGV with higher priority can continue to move along the straight road.

5. Evaluation Function Improvement

Controlling the input of the AGV through the MPC model prediction algorithm can enable the AGV to better track the optimal trajectory. The MPC algorithm takes the optimal trajectory planned by the DWA algorithm as the reference input, and controls the AGV's pose during the actual driving process by controlling the AGV's linear velocity and angular velocity. The ultimate optimization goal of MPC is to optimize the input of AGV linear velocity and angular velocity, so that the path trajectory of the reference input can be tracked as much as possible under the smallest input.

The control signal is:

$$u = \begin{bmatrix} v & \omega \end{bmatrix}^T \tag{18}$$

Convert the motion model of the AGV into discrete form:

$$\begin{bmatrix} x_c(k+1) \\ y_c(k+1) \\ \theta_c(k+1) \end{bmatrix} = \begin{bmatrix} x_c(k) \\ y_c(k) \\ \theta_c(k) \end{bmatrix} + \begin{bmatrix} \cos(\theta_c(k)) & 0 \\ \sin(\theta_c(k)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(k) \\ \omega(k) \end{bmatrix} \Delta T \tag{19}$$

According to the discrete form of the AGV motion model, the incremental equation can be obtained:

$$\begin{aligned} \Delta x(k+1) &= A\Delta x(k) + B\Delta u(k) \\ y_c(k) &= C\Delta x(k) + y_c(k-1) \end{aligned} \tag{20}$$

where:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \tag{21}$$

Set the prediction step size to P and the control step size to M.

$$x(k+1|k)=\begin{bmatrix} CA \\ \vdots \\ \sum_{i=1}^{P} CA^{i} \end{bmatrix}\Delta x(k)+\begin{bmatrix} I_n \\ \vdots \\ I_n \end{bmatrix} y_c(k)+\begin{bmatrix} CB & 0 & 0 \\ \vdots & \vdots & \vdots \\ \sum_{i=1}^{P} CA^{i-1}B & \dots & \sum_{i=1}^{P+M-1} CA^{i-1}B \end{bmatrix}\Delta U(k) \tag{22}$$

where:

$$\Delta U(k)=\left[\Delta u(k) \quad \cdots \quad \Delta u(k+M-1)\right]^{T} \tag{23}$$

In order to make the difference between the actual driving trajectory of the AGV and the optimal trajectory route of the reference input as small as possible, the driving route is smoother, and the tracking ability is stronger, a cost function needs to be established to constrain the trajectory deviation value and the change range of the control input.

Minimize

$$\begin{gathered}\min \sum_{n=1}^{P}\left\| Y(k+n)-Y_{ref}(k+n)\right\|_{Q}^{2}+ \\ \sum_{n=1}^{M}\left\|\Delta u(k+n-1)\right\|_{R}^{2}\end{gathered} \tag{24}$$

subject to

$$\begin{gathered}u_{\min}\le u(k+n)\le u_{\max} \\ \Delta u_{\min}\le \Delta u(k+n)\le \Delta u_{\max}, \quad n=0,1\cdots M\end{gathered} \tag{25}$$

where Q and R are the weight matrix, $Y_{ref}(k)$ are the output reference trajectory, $\Delta u(k+n-1)$ are the change values of the input at the time of $k+n-1$, $u_{\min}$ and $u_{\max}$ are the thresholds of the control input, $\Delta u_{\min}$ and $\Delta u_{\max}$ are the thresholds of the control input changes.

6. Simulation

In the context of the aforementioned content, several experiments are presented in this chapter to demonstrate the effectiveness of the proposed strategies. The similation environment is MATLAB R2021a and Windows 10-64 bit system (CPU AMD Ryzen 7 5800H and RAM 16 GB).

To validate the effectiveness of the improved DWA algorithm proposed in this paper, we construct random obstacles and introduce three agents to compare the performance of the proposed algorithm with the original DWA algorithm under the same environment and parameters. Here are the coordinates for the randomly generated ten circular obstacles with a radius of 0.5 within the range of $x \in [1, 9]$ and $y \in [1, 9]$: [0,2], [2,4], [2,5], [4,2], [5,4], [5,6], [5,9], [7,9], [8,8], [8,9]. Agent 1 moves from the starting point [10, 8] to the target point [0, 0], Agent 2 moves from the starting point [9, 10] to the target point [4, 0], Agent 3 moves from the starting point [7, 8] to the target point [2, 0]. Evaluation function parameter values: Weight of heading score $\alpha = 0.5$, weight of obstacle distance score $\beta = 0.2$, weight of velocity score γ = 0.1, Weight of avoidance score $\sigma = 0.2$. In figure 2, it can be observed that both algorithms have the same initial states for the agents and share the same target points. In figure 3, the left image clearly illustrates that the original DWA algorithm fails to handle scenarios with multiple agents, resulting in collisions between the agents during the path planning process and ultimately leading to task failure. On the other hand, the right image demonstrates the improved DWA algorithm, which takes the relative distances between agents into account. It actively avoids collisions by moving away from nearby agents when there is a potential risk of collision. As a result, the improved algorithm successfully achieves the task objective without any collisions.

In certain specific environments, the improved DWA algorithm also demonstrates its superiority over the original DWA algorithm. In figure 4, the special case is a narrow straight path with obstacle distribution as follows: [3,3.75], [3,6.25], [4,3.75], [4,6.25], [6,3.75], [6,6.25]. Agent 1 moves from the starting point [7, 5.5] to the target point [0, 4.5], Agent 2 moves from the starting point [0, 4.5] to the target point [7.5, 5.5].

In figure 5, the onther special case is a crossroad with the following distribution of obstacles: [3,3.75], [3,6.25], [4,2.75], [4,3.75], [4,6.25], [4,7.25], [6,2.75], [6,3.75], [6,6.25], [6,7.25], [7,3.75], [7,6.25]. Agent 1 moves from the starting point [5, 8] to the target point [1, 4.5], Agent 2 moves from the starting point [2, 4.5] to the target point [8.5, 5.5].

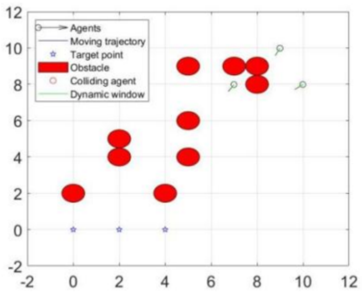

Figure 2. At t=0, the initial states of both algorithms are the same.

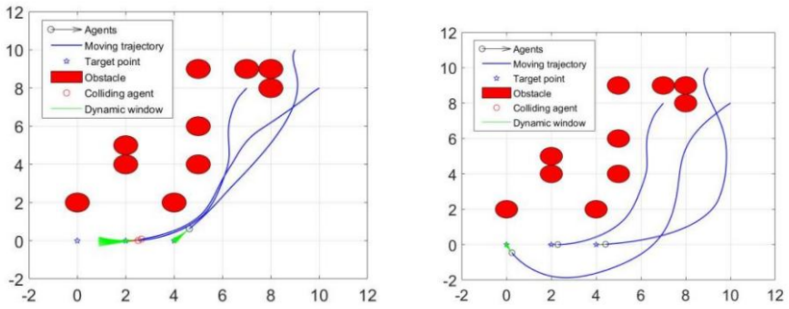

Figure 3. The left figure shows the original DWA algorithm where collision occurs between agent 1 and agent 3 at 40s, while the right figure represents the improved DWA algorithm successfully completing the task.

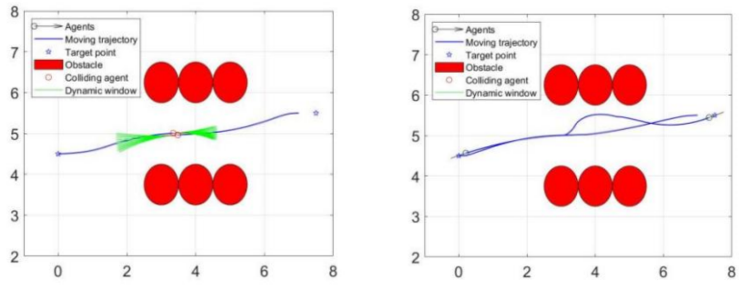

Figure 4. The left image shows that in the original DWA algorithm, there is a collision between the two agents as they pass through the narrow straight path. However, in the right image, the improved DWA algorithm allows one agent to decelerate and wait, allowing the other agent to pass through first, thus completing the task without a collision.

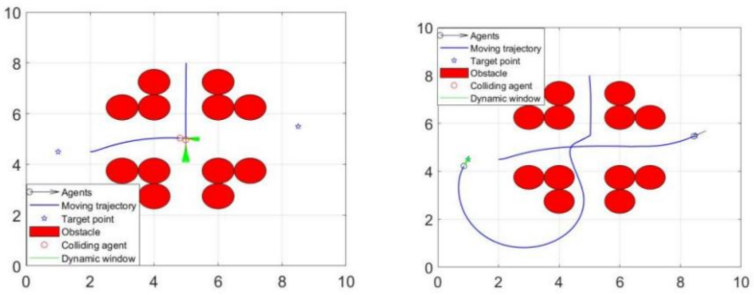

Figure 5. The left image shows that the original DWA algorithm leads to a collision between the agents while crossing the crossroad. On the other hand, the right image demonstrates that the improved DWA algorithm allows the lower-priority agent to wait at the intersection for the higher-priority agent to pass, thus successfully completing the task.

7. Conclusions

This paper proposes an improved DWA path planning algorithm combined with MPC algorithm, which can be applied to the working environment where multiple AGVs work together. According to the vision sensor and displacement sensor carried by the AGV itself, ORB-SLAM3 is used to construct a point cloud map, which is converted into an octree map, and then processed to construct a two-dimensional map suitable for path planning. In order to plan the path of multiple AGVs at the same time, this paper improves the original evaluation function, takes the relative distance and relative speed between different AGVs as a new evaluation index, and makes additional restrictions and countermeasures for possible collision situations. For possible special situations, corresponding solutions are also proposed. Finally, the path planned by the DWA algorithm is optimized by combining the MPC algorithm, so that the AGV can track the specified path with as little control input as possible.

References

[1] Rubio, F., Valero, F., Llopis-Albert, C. A review of mobile robots: Concepts, methods, theoretical framework, and applications[J]. Int. J. Adv. Robot. Syst. 2019, 16, 1729881419839596.

[2] Niloy, M.A.K., Shama, A., Chakrabortty, R.K., Sarker, M.R., Uddin, M.Z. Critical design and control issues of indoor autonomous mobile robots: A review[J]. IEEE Access 2021, 9:35338-35370.
[3] Wang, H., Yu, Y., Yuan, Q. Application of Dijkstra algorithm in robot path-planning[C]. In Proceedings of the 2011 Second International Conference on Mechanic Automation and Control Engineering, Hohhot, China, 15-17 July 2011, pp. 1067-1069.
[4] Duchoň, F., Babinec, A., Kajan, M., Hudák, M., Takács, B. Path planning with modified a star algorithm for a mobile robot[J]. Proc. Eng. 2014, 96:59-69.
[5] Brand, M., Masuda, M., Wehner, N., Xiao-Hua, Y. Ant colony optimization algorithm for robot path planning[C]. In Proceedings of the 2010 International Conference on Computer Design and Applications, Qinhuangdao, China, 25-27 June 2010.
[6] Fox, D., Burgard, W., Thrun, S. The dynamic window approach to collision avoidance[J]. IEEE Robot Autom Mag. 1997, 4: 23-33.
[7] Zhou, C., Huang, B., Fränti, P. A review of motion planning algorithms for intelligent robots[J]. Intell Manuf. 2022, 33:387-424.
[8] Kobayashi, M., Motoi, N. Local Path Planning: Dynamic Window Approach with Virtual Manipulators Considering Dynamic Obstacles[J]. IEEE Access 2022, 10: 17018-17029.
[9] Liu, L., Lin, J., Yao, J., Gao, Y., He, D., Zheng, J., Huang, J., Shi, P. Path planning for smart car based on Dijkstra algorithm and dynamic window approach[J]. Wirel Commun Mob Comput, 2021.
[10] Chen, T.J., Sun, Y., Dai, W., Tao, W., Liu, S. On the shortest and conflict-free path planning of multi-AGV system based on dijkstra algorithm and the dynamic time-window method[J]. In Proceedings of the International Conference on Industrial Engineering and Engineering Management, Bangkok, Thailand, 10-13 December 2013, pp. 267-271.
[11] Yang, L., Fu, L., Li, P., Mao, J., Guo, N. An Effective Dynamic Path Planning Approach for Mobile Robots Based on Ant Colony Fusion Dynamic Windows[J]. Machines 2022, 10, 50.
[12] Chi, X., Li, H., Fei, J.Y. Research on random obstacle avoidance method for robots based on the fusion of improved A* algorithm and dynamic window method[J]. Chin. J. Sci. Instrum. 2021, 42: 132-140.
[13] Zhong, X., Tian, J., Hu, H., Peng, X. Hybrid path planning based on safe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment[J]. Intell. Robotic Syst. 2020, 99: 65-77.
[14] Zhu, Z., Xie, J., Wang, Z. Global dynamic path planning based on fusion of A* algorithm and Dynamic Window Approach[C]. In Proceedings of the 2019 Chinese Automation Congress (CAC), Hangzhou, China, 22-24 November 2019, pp. 5572-5576.
[15] Molinos, E.J., Llamazares, A., Ocaña, M. Dynamic window based approaches for avoiding obstacles in moving[J]. Rob Auton Syst 2019, 118: 112-130.
[16] Campos, C., Elvira, R., Rodríguez, J.J.G., Montiel, J.M.M., Tardós, J.D. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM[J]. IEEE Trans. Robot. 2021, 37: 1874-1890.
[17] Mur-Artal, R., Montiel, J.M.M., Tardos, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System[J]. IEEE Trans. Robot. 2015, 31: 1147-1163.
[18] Lv, Q., Huican, L., Wang, G., Wei, H. ORB-SLAM-based tracing and 3D reconstruction for robot using Kinect 2.0[C]. In Proceedings of the 2017 29th Chinese Control and Decision Conference (CCDC), Chongqing, China, 28-30 May 2017, pp. 3319-3324.
[19] Li, X., Ao, H., Belaroussi, R., Gruyer, D. Fast semi-dense 3D semantic mapping with monocular visual SLAM[C]. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16-19 October 2017, pp. 385-390.
[20] Wurm, K.M., Hornung, A., Bennewitz, M., Stachniss, C., Burgard, W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees[J]. Rob Auton Syst 2013, 34:189-206.
[21] Wurm, K.M., Hornung, A., Bennewitz, M., Stachniss, C., Burgard, W. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems[C]. In Proceedings of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation, Anchorage, AK, USA, 3 May 2010.
[22] Xu, H., Yu, Z., Lu, X., Wang, S., Li, S., Wang, S. Model predictive control-based path tracking control for automatic guided vehicles[C]. In Proceedings of the 2020 4th CAA International Conference on Vehicular Control and Intelligence (CVCI), Hangzhou, China, 18-20 December 2020, pp. 627-632.
[23] Wu, Y., Li, S., Zhang, Q., Liu, Y., Sun-Woo, K., Yan, Y. Route Planning and Tracking Control of an Intelligent Automatic Unmanned Transportation System Based on Dynamic Nonlinear Model Predictive Control. IEEE Trans[J]. Intell. Transp. Syst. 2022.
[24] Kim, J.C., Pae, D.S., Lim, M.T. Obstacle avoidance path planning algorithm based on model predictive control[C]. In Proceedings of the 2018 18th International Conference on Control, Automation and Systems (ICCAS), Pyeongchang, Korea 17-20 October 2018, pp. 141-143.