Research on Path Planning Problem of Smart Car Considering Kinematic Constraints

Zhanhong Liu

YK Pao Middle School, Shanghai, China

elowoo_d@outlook.com

Abstract. With the development of artificial intelligence, intelligent cars have gradually appeared in people's lives and brought many conveniences to people's lives and work. On the street, we can see many busy figures of unmanned delivery cars; In some factories, smart cars shuttle back and forth between product production processes for material distribution; In some hospitals, we can also find that some smart cars replace medical staff to undertake the task of sending medicines and tests. For intelligent cars, autonomous collision-free path planning and trajectory tracking of the planned path are the most basic and core tasks of intelligent vehicles. The existing path planning algorithms are mainly divided into search-based algorithms (Dijkstra, A*, etc.) and sample-based algorithms (PRM, RRT, etc.), and different types of algorithms have good performance in their respective applicable scenarios. However, these path planning algorithms all use the same simplification assumption, that is, the intelligent car is regarded as a freely moving point or sphere in the path planning task, and the driving experience in life tells us that the movement of the wheel-driven intelligent car is constrained by the turning radius of the vehicle, so the path planned by the simplified intelligent car for a point is often not suitable for the driving of the intelligent car in reality. Considering this constraint, this project models the kinematics of the intelligent trolley with four-wheel steering, and adds kinematic constraints to the path planning, so as to plan the path that conforms to the kinematics model of the intelligent trolley. The path planning algorithm proposed in this topic will be tested in ROS and Gazebo simulation environments, and compared with the traditional search-based A* algorithm, the results of multiple scenarios verify the effectiveness of the algorithm. Finally, this project summarizes and looks forward to this research, which has completed the construction of low-cost physical intelligent vehicles, and plans to complete the verification of the real vehicle function by combining the designed path planning algorithm with the real-time mapping and positioning system of the car in the future.

Keywords: Intelligent trolley, Path planning, Kinematic constraints.

1. Introduction

The intelligent car is an unmanned car equipped with sensors that can perceive the environment, interact with the environment, plan sports independently, avoid collisions and avoid obstacles. Due to its high degree of intelligence, intelligent cars can bring many conveniences to people's lives. At the moment of the epidemic, human transportation is extremely tight, and many takeaway delivery and express logistics staff are sick and unable to quickly deliver the required items to families and companies in need. In this environment, smart cars have great use, such as Meituan's unmanned delivery car (see Figure 1(a)), which can quickly find the optimal delivery route according to the present map and real-time road conditions, and give the required items to those people. In addition, such trolleys can be mass-produced and do not require a break, which can continue to serve couriers. The popularity of smart cars can also reduce the number of people-to-person encounters, thus reducing the risk of virus transmission. Smart cars can also replace human patrols in cities and replace police patrols with video recordings (see Figure 1(b)), providing more evidence to maintain social order and punish those who break the law. In addition to being popularized and applied in cities, smart cars can also be applied to rural agriculture after receiving certain mechanical transformation (see Figure 1(c) Plant Protection Agricultural Machinery). During the rice harvest season, farmers must harvest the rice before the rainy day, but they also need plenty of rest and no outside influence that causes farmers to stop working. In this way, farmers will not only be overworked, but the efficiency

of rice harvesting will also be low. But with the help of smart cars, the work that would have taken several days can be done by robots in a day, saving a lot of manpower and time. In the industrial sector, people still rely heavily on mineral resources such as coal and oil, but there are also certain dangers in mining such mines. A sudden collapse can lead to many unimaginable consequences, but with the help of unmanned exploration vehicles (see Figure 1(d)), the structure and shape of the mine cave can be detected in advance, and the mining plan can be set in advance, reducing the risk of surveyors being killed.

Figure 1. A series of applications of intelligent trolleys: (a) Meituan unmanned delivery trolley, (b) unmanned patrol police vehicle, (c) unmanned agricultural machinery, (d) unmanned detection mining vehicle

Path planning is one of the contents of robot motion planning. Motion planning consists of path planning and trajectory planning. The curve that connects the start and end points of the robot in the environment is called the path, and the strategy of finding the path is called path planning. Path planning has many uses in life, such as vehicle route planning based on GPS navigation system [1], express logistics distribution route planning [2], and sweeping robot route planning [3]. The general steps of path planning are as follows: First, according to the sensor data (lidar or camera, etc.) that can perceive the environment, a model of the environment around the robot is made, and at the same time, the environment model is rasterized to facilitate the robot to carry out path planning tasks. Second, the optimal collision-free path is searched by a specified algorithm in the map of the modelling environment. Third, with the help of the controller, the trolley can follow the path planned by the algorithm as much as possible, and if there is a situation where it cannot follow or the movement obstacle interferes, the path is replanned.

Depending on the planning strategy, path planning algorithms can be divided into two broad categories: search-based methods and sampling-based methods. Search-based methods include graph search methods [4], Dijkstra [5], and A* [6] algorithms. The most commonly used in graph search are breadth-first search, depth-first search, and heuristic search (greedy algorithm). These kinds of graph search algorithms have obvious defects, breadth-first search and depth-first search often need to visit a large number of nodes to search for the target point, heuristic search can search for the target point faster, but at the same time, there is a greater probability of falling into the local optimal value, resulting in the inability to search the global optimal path. Dijkstra and A* algorithms combine breadth-first search and heuristic search to jointly consider, under the premise of ensuring that the

optimal path is searched, the efficiency of search is improved, but the path planned by Dijkstra and A* algorithm has quite a lot of corners, the corners are generally polylines, the path is not smooth, resulting in the robot turning at the inflection point is difficult, resulting in the robot tracking the path when tracking the difficult tracking, easy to collide at the corner. Sampling-based methods include probabilistic road map (PRM) [7] and fast random tree (RRT) [8] algorithms; The PRM algorithm transforms the continuous space into a discrete space by adoption, and uses the graph search method to search for paths in the discrete space to improve efficiency. Although in small scenes, a relatively small number of sample points can cover most of the feasible space, when there are too few sample points or the sample points are not evenly distributed, the PRM algorithm will not be able to get a feasible path. Compared with the PRM algorithm, the RRT algorithm is simpler and more versatile, the search process of the RRT algorithm is equivalent to a tree growing continuously from the beginning and spreading around, until the tree spreads to the end, because the RRT algorithm adopts a completely random process, the final path obtained is not optimal, and even very twisted and winding. The methods summarized above each have certain defects, and the methods have a basic assumption: "treat the car as a point and move freely in any direction", and most of the intelligent cars in life use wheeled steering geometric models.

In order to solve the technical problems of these path planning algorithms, this study will propose to improve the A* algorithm, after adding kinematic constraints, the improved A* algorithm avoids the shortcomings of the traditional A* algorithm with large turning angle and planning path close to obstacles, while retaining the advantages of high search efficiency and planning success rate of A* algorithm. In addition, the size of the robot itself is usually not considered in the trajectory search, so there is a chance that the planned route will not collide but the collision will occur in the actual situation due to the robot volume, so the improved A* algorithm also introduces the function of obstacle expansion, so that the safety of path planning is guaranteed.

The remainder of the article is as follows: the second part introduces the software framework, system platform, and implementation details of the improved A* algorithm for this study; The third part introduces the simulation environment and experiments for testing and improving the A* algorithm, and analyses the experimental results. Part IV summarizes the entire topic and provides an overview of ongoing research for future extensions.

2. Methods

2.1. Formatting the title

In this project, a robot operating system (ROS) is used. With the rapid development and complexity of the field of robotics, the need for code reuse and modularization is becoming increasingly strong, and the existing open source system can no longer adapt well to the needs, and in 2010, Willow Garage released the open source robot operating system ROS [9]. ROS provides the services engineers expect from an operating system, including hardware abstraction, low-level device control, implementation of common functions, messaging between processes, and package management. It also provides tools and libraries to get, build, write, and run code on multiple computers. At the same time, ROS supports a variety of programming languages. It is easy to communicate between Python and C++ nodes, and C++ programming is mainly used in this topic. ROS is a framework for robot programming that couples otherwise loose components together and provides them with a communication architecture in which ROS nodes are connected to the different modules used by the robot and transferred data to and from the rest of the nodes through subscription-publish and client-service frameworks. ROS topics are one-way communication mechanisms, when an ROS node is a publisher, it usually converts data into message data and publishes it to a topic, while other nodes need to subscribe to the topic if they want to get the message packet. ROS's service is a synchronous two-way communication mechanism, the server only responds when there is a request, the client accepts the response after the request is made, and when the service request and response are completed, the two connection points are automatically disconnected. The communication method of

ROS action is similar to that of the service, except that it takes a long time for the server to complete the response after receiving the request, and it needs to feedback the current completion to the client and report the current status in the middle. This article summarizes the communication mechanism of ROS in Figure 2. Although ROS is called an operating system, it is not an operating system in the usual sense of Windows and Mac, it is only connected to the operating system and the ROS application developed by the user, so it is also a middleware, ROS-based applications have established a bridge of communication, on the ROS platform, the robot perception, decision-making, control algorithms can be better organized and run. In addition, the ROS platform is highly compatible with the Gazebo simulation platform [10] mentioned below, and based on these advantages, most robotics algorithm research and development will use the ROS platform.

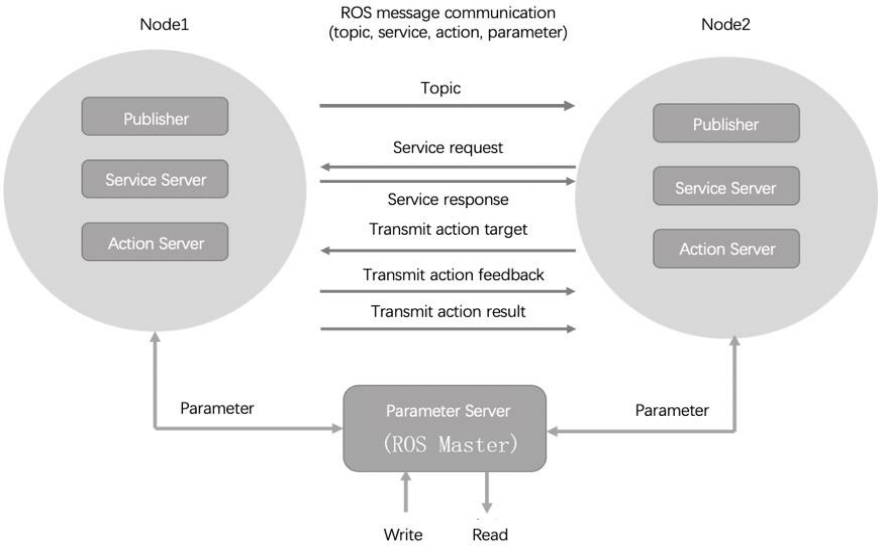


Figure 2. Schematic of the communication mechanism of ROS

2.2. Traditional A* algorithm

The A* algorithm is a very common path finding and graph traversal algorithm, and because it is guided by heuristics, the A* algorithm generally has better performance than ordinary search algorithms. Heuristic exploration is to use the heuristic information (such as the distance from the end point) to guide the search, so as to reduce the scope of exploration and reduce the complexity of the problem. The general idea of the A* algorithm is: starting from the starting point node, traversing the search for neighbouring points around the starting point (usually eight nodes around the current node, see Figure 3) through the heuristic function, and selecting the best advantage as the next expansion point, and then performing the same heuristic search traversal operation on the next expansion point, gradually spreading outward until the end point is found, and finally returning to the starting point from the end node to obtain the final path.

The core formula of the A* algorithm is its heuristic function:

$$F(n) = G(n) + H(n) \tag{1}$$

Where $F(n)$ is the estimated cost of moving from the start to the end of the current node n; $G(n)$ represents the cost of moving from the origin to the current node n; $H(n)$ represents the estimated cost of moving from the current node n to the end point. The G-value of the child node is the G-value of the parent node plus the cost of moving the parent node to the current node. The H value is typically calculated from the distance from Manhattan [11] from the current node n to the end point. Manhattan Distance (MD), also known as taxi distance, is used to calculate the sum of the absolute axis distances of two points in a standard coordinate system. The distance of Manhattan between a point (x_2, y_2) and a point (x_1, y_1) in two-dimensional space is defined as follows:

$$d = |x_1 - x_2| + |y_1 - y_2| \tag{2}$$

The A* algorithm usually uses two linked lists to record the nodes to be traversed and the nodes that have been traversed, which are called open linked lists and closed linked lists, respectively. The full A* algorithm description is described as Algorithm 1.

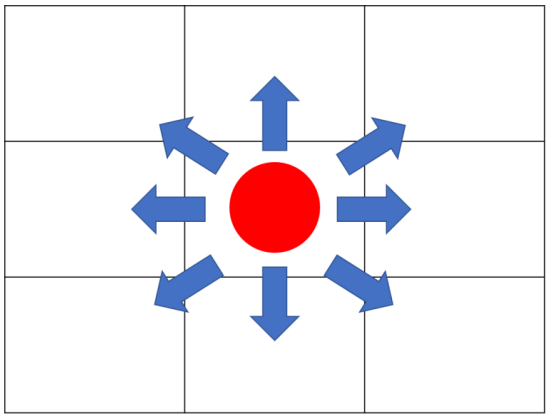


Figure 3. A* algorithm extends the node graph

2.3. Improved A* algorithm that considers kinematic constraints

2.3.1 Kinematic constraints of wheeled mobile robots

Assuming that the basic configuration space of the robot is $q = (x, y, \theta)$, in the traditional A* algorithm, only the x and y states of the robot are considered, and there is no constraint on the speed of the robot $v = (\dot{x} + \dot{y})$; However, the experience of car driving tells us that in the actual driving process of the four-wheel mobile robot, the vehicle cannot directly translate to the left or right, that is, the speed perpendicular to the front of the vehicle is 0. Using Figure 4 as an example, assuming that the vehicle's velocity perpendicular to the front of the vehicle is $v_\perp$, the following constraints apply:

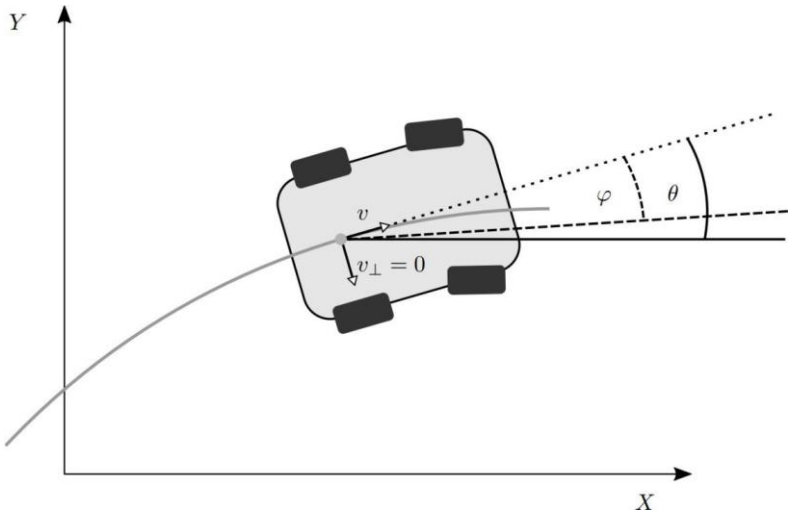


Figure 4. Schematic diagram of kinematic constraints of a four-wheeled steering mobile robot

$$v_\perp = \frac{\dot{x}}{cos(\theta - \pi/2)} \tag{3}$$

$$v_\perp = -\frac{\dot{y}}{sin(\theta - \pi/2)} \tag{4}$$

Simultaneous (3) and (4) yield:

$$\frac{\dot{x}}{cos(\theta - \pi/2)} = -\frac{\dot{y}}{sin(\theta - \pi/2)} \tag{5}$$

$$\dot{x}\, sin(\theta - \pi/2) + \dot{y}\, cos(\theta - \pi/2) = 0 \tag{6}$$

Finally, the kinematic constraints of the four-wheeled mobile robot are:

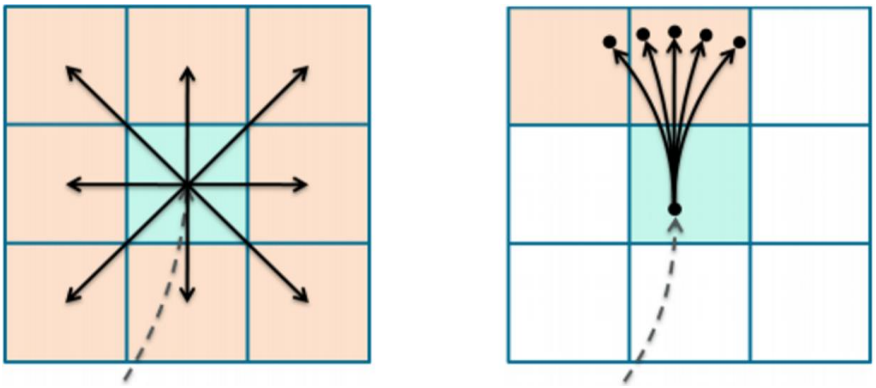
$$\dot{x}\,cos(\theta) - \dot{y}\,sin(\theta) = 0 \tag{7}$$

2.3.2 Improved A* algorithm

In practice, the trajectory of the robot is smooth, while the A* algorithm is discrete when expanding nodes, and some nodes that do not meet the kinematic constraints of wheeled robots are also taken into account when searching for nodes, so that the resulting path is usually said to be unexecutable because the direction change of the vehicle is sudden rather than smooth. To solve this problem, this topic will improve the traditional A* algorithm.

First, consider the kinematic constraints introduced in 2.3.1 when searching for surrounding nodes. The search space of the improved A* algorithm not only considers the search in the x and y directions, but also considers the search element in the θ direction, we adopt (x, y, θ) to retain the state that conforms to the kinematic constraints in equation (7) as the search point, compared with the A* algorithm only for the x and y direction search, the search node of the improved algorithm will be more in line with the kinematic constraints of the vehicle and smoother, the specific node expansion comparison chart is shown in Figure 5. Due to the different ways of node expansion, improving the heuristic function of the A* algorithm also needs to be modified in the heuristic function formula (1) of the traditional * algorithm. While the traditional A* algorithm heuristic function uses the cost of movement between lattices, the improved A* algorithm replaces $G(n)$ with a kinematically constrained path length from the starting point to the node n.

In addition, in the traditional A* algorithm, for the simplicity of the algorithm, the robot is usually idealized as a particle without volume, and the path planned under this idealization premise is usually attached to the obstacle, and there will be certain safety risks, especially when passing between the obstacles of the two oblique streets, because the robot has its own volume, the robot has a high probability of colliding with the obstacle, resulting in the inability to travel or even the robot damage. In order to solve this problem, the improved A* algorithm rasterizes and expands all obstacles, expands with the approximate radius of the robot, and then treats the robot as a particle for path planning algorithm processing, which can effectively reduce the risk of obstacle collision while retaining the simplicity and applicability of the algorithm to the greatest extent.



(a) Traditional A* node extension (b) Improved A* node extension

Figure 5. Node expansion comparison chart for traditional A* and improved A*

3. Experimental testing and analysis

The path planning method proposed in this topic will be simulated and tested with the help of the Gazebo simulation platform; the Gazebo platform is highly compatible with ROS and supports the DART dynamic physics engine, which can highly simulate indoor and outdoor environments. In addition, Gazebo also supports the simulation of sensors. This article will design two simulation maps dedicated to testing, and use a four-wheel mobile car as the execution body of the path planning task.

(a) Simple test scenario (b) Complex test scenario (c) Simulation of moving four-wheeled car

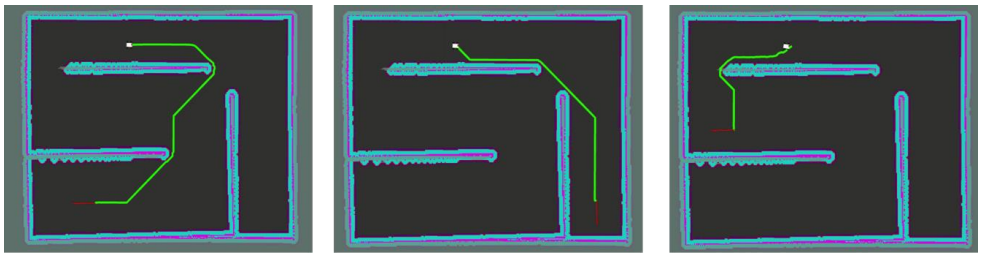
Figure 6. Gazebo path planning test scenario with simulation trolley

In order to reflect the comparison and improvement of the improved A* algorithm to the traditional A* algorithm, this paper designs two maps with different degrees of difficulty. Figure 6(a) is a simple map, and Figure 6(b) is a complex map. The complex map restores author's family building structure. In order to avoid the contingency of a single planning result, this paper tests the planning paths of traditional A* and improved A* under multiple different end points in simple maps and complex maps, and the time required for the robot to reach the end point from the starting point in the simulation. Under the premise of the same destination and using the same controller, the smoother and more efficient the path, the easier the mobile robot will follow and the less time it will take to reach the destination.

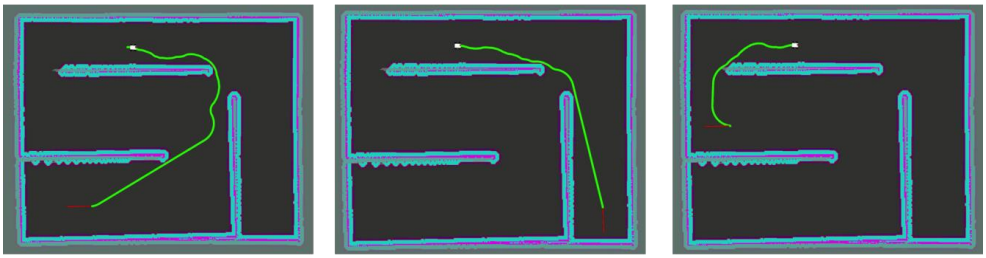
The comparison of planned paths between the traditional A* algorithm and the improved A* algorithm under a simple map is shown in Figure 7; the comparison of planned paths under a complex map is shown in Figure 8; the time-consuming summary of each planned path tracking is shown in Table 1. In Figures 7 and 8, the white rectangle represents the robot (at the starting position), the red arrow is the target point, and the green route is the trajectory; in order to avoid collisions, the maps used by both improved A* and traditional A* have performed obstacle expansion operations (The inflated part is the blue area).

Table 1. Comparison of the tracing time of the traditional A* algorithm and the improved A* algorithm planning path

Method	Easy test 1	Easy test 2	Easy test 3	Complex test 1	Complex test 1	Complex test 1	Complex test 1
Traditional A* algorithm	39.06s	32.61s	26.34s	28.93s	43.28s	47.95s	18.63s
Improved A* algorithm	19.82s	13.21s	9.86s	23.12s	21.59s	24.63s	16.47s

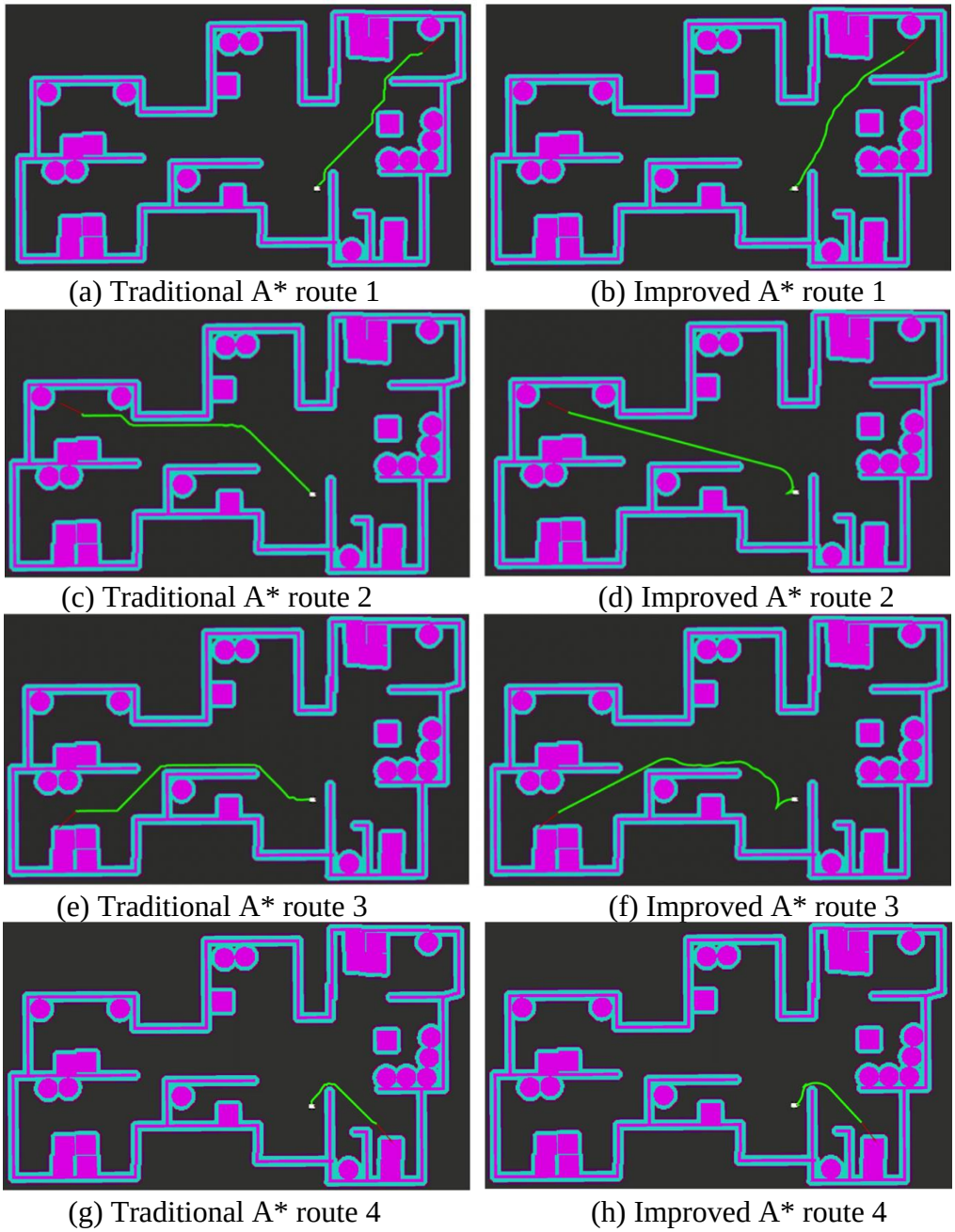


(a) Traditional A* path 1 (b) Traditional A* path 2 (c) Traditional A* path 3



(d) Improved A* path 1 (e) Improved A* path 2 (f) Improved A* path 3

Figure 7. Comparison of simple scene path planning results



(a) Traditional A* route 1 (b) Improved A* route 1

(c) Traditional A* route 2 (d) Improved A* route 2

(e) Traditional A* route 3 (f) Improved A* route 3

(g) Traditional A* route 4 (h) Improved A* route 4

Figure 8. Comparison of path planning results for complex scenarios

It can be seen from the planning path diagram that the path planned by the improved A* algorithm is smoother and more continuous than the traditional A* algorithm, and at the same time there are few sudden corners (the corners of the route in the figure are when the car moves forward or backward or turns backward. Forward); and the traditional A* algorithm, due to its eight-direction search strategy, will generate a kink at the beginning of each turn, resulting in discontinuity of the path at the turn. It can be seen from the path tracking time-consuming graph that when the improved A* algorithm plans a path with a similar length to the traditional A* algorithm, the path planned by the improved A* algorithm is easier to be tracked by the robot, and the tracking time is less; on average, the path tracking time of the improved A* algorithm is about half that of the traditional A* algorithm, which is mainly due to the fact that the improved A* algorithm takes the kinematic constraints of the car into consideration, and the planned path conforms to the kinematic constraints of the vehicle, so that it is easier to be tracked by the car.

4. Summary and Outlook

This paper proposes an improved path planning algorithm based on the traditional A* algorithm. This paper first analyses the defects of the traditional A* algorithm, including; the path is not smooth, and the corners are full of broken lines, which makes it difficult for the robot to turn around the corner; and then a new algorithm is proposed. The solution idea: consider the kinematic constraints of the mobile robot when expanding the nodes in path planning, and model the kinematics of the four-wheeled mobile robot to improve the A* algorithm; in addition, this paper also introduces obstacle rasterization Expansion processing increases the safety of path planning. In the path planning tests of simulation experiments, no collision case occurred. Finally, multiple experimental results in simple and complex scenes have verified that the improved A* algorithm is more effective than the traditional A* algorithm, the planned path is smoother, and it is easier to be tracked by the robot.

In the future, this study plans to apply the proposed algorithm to a physical mobile robot and conduct a test in a real environment. At present, this research has built a physical mobile robot (see Figure 9), and the follow-up will combine the radar/camera sensor installed on the mobile robot Complete real-time environment modelling and path planning tasks together.

Figure 9. Controllable physical four-wheeled trolley

References

[1] Xiaojian Lv. Research on path planning technology of vehicle navigation system. Master's thesis, Shanghai Jiaotong University, 2011.

[2] Yarong Shi, Di Wan, Shuangyan Li, Zhenyu Lv. Research on logistics distribution route planning based on GIS. Systems Engineering-Theory & Practice, 29(10): 76–84, 2009.

[3] Yi Jian, Bin Gao, Yue Zhang. Research on a full-traversal path planning method for indoor sweeping robot. Transducers and Microsystems, 37(1): 32–34, 2018.

[4] Qi Song, Yinghui Wu, Peng Lin, Luna Xin Dong, and Hui Sun. Mining summaries for knowledge graph search. IEEE Transactions on Knowledge and Data Engineering, 30(10): 1887–1900, 2018.

[5] Huijuan Wang, Yuan Yu, and Quanbo Yuan. Application of dijkstra algorithm in robot path-planning. In 2011 second international conference on mechanic automation and control engineering, pages 1067–1069. IEEE, 2011.

[6] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. Path planning with modified a star algorithm for a mobile robot. Procedia Engineering, 96: 59–69, 2014.

[7] David Hsu, Jean-Claude Latombe, and Hanna Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. The International Journal of Robotics Research, 25(7): 627–643, 2006.

[8] Steven M LaValle and James J Kuffner. Rapidly-exploring random trees: Progress and prospects: Steven m. lavalle, iowa state university, a james j. kuffner, jr., university of tokyo, tokyo, japan. Algorithmic and Computational Robotics, pages 303–307, 2001.

[9] Anis Koubâa et al. Robot Operating System (ROS)., volume 1. Springer, 2017.

[10] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), volume 3, pages 2149–2154. IEEE, 2004.

[11] Deepak Sinwar and Rahul Kaushik. Study of euclidean and manhattan distance metrics using simple k-means clustering. Int. J. Res. Appl. Sci. Eng. Technol, 2(5): 270–274, 2014.