Computer Life
ISSN: 1819-4818 | Vol. 13, No. 2, 2025

A Path Planning Algorithm for Unmanned Overhead Cranes Integrating Improved A* and D* Lite with Kinematic Optimization

Wubo Zhang

School of Anhui University of Technology, Anhui 10360, China

Abstract: Aiming at the problems of excessive redundant nodes, unsmooth paths, and poor adaptability to dynamic environments in the path planning of unmanned cranes in coil warehouses, this paper proposes a hybrid path planning algorithm (AD* Lite) integrating improved A* and D* Lite, along with a quintic polynomial kinematic optimization method. Firstly, a collision detection mechanism is designed to prevent paths from passing through obstacle vertices, and a redundant node deletion strategy is introduced to enhance path smoothness. Secondly, a multi-scenario adaptive hybrid algorithm is constructed by combining the static planning advantages of the A* algorithm and the dynamic re-planning capability of the D* Lite algorithm. Furthermore, quintic polynomials are used to optimize the path, integrating kinematic constraints such as speed, acceleration, and turning radius. Experiments show that under the satisfaction of kinematic constraints, the improved fusion algorithm improves various indicators such as path length, search time, number of turns, and number of searched grids by approximately 40%.

Keywords: Path Planning; A* Algorithm; D* Lite Algorithm; Quintic Polynomial; Kinematic Constraints.

1. Introduction

In the automation process of the steel industry, path planning for unmanned cranes in coil warehouses is a core technology. Traditional algorithms have problems such as paths passing through obstacle vertices, excessive redundant nodes, and single application scenarios. Improved fusion algorithms based on traditional algorithms provide new ideas for path planning. Salzmann et al. (2022)[1] proposed a Transformer-based multimodal trajectory prediction model, which solves the problem of generating dynamically feasible paths by fusing visual, radar, and map data. However, the self-attention mechanism leads to high inference latency, making it difficult to meet the requirements of real-time path planning. Yan Jianhong et al. (2024)[2] proposed a mobile robot path planning algorithm that improves A* by fusing the Dynamic Window Approach (DWA), addressing the issues of dense turning points and excessive traversed nodes in the A* algorithm's path planning process. Shi Gaojian et al. (2024) proposed a greedy strategy-improved RRT* algorithm for robotic arm path planning[3], which solves the problems of suboptimal path quality and slow convergence speed but may have deficiencies in path optimality. Ren Qingxin et al. (2025)[4] proposed a robot 2D path planning method based on Dijkstra's algorithm and the Arctic Puffin Optimization (APO) algorithm, which optimizes path quality and planning efficiency but has poor scalability and environmental adaptability. Currently, although A* and D* Lite algorithms are mature, they still have issues in adaptability to static and dynamic environments and path turning smoothness, and neither considers crane kinematic constraints (such as speed and acceleration limits). Through analyzing actual scenario requirements, it is found that existing algorithms have room for improvement in complex scenario adaptability and feasibility under multiple constraints.

2. Hybrid Path Planning with Improved A* and D* Lite Algorithms

Traditional A*[5] and D* Lite algorithms[6] are respectively suitable for static and dynamic scenarios, with single application scenarios. Additionally, these algorithms have problems such as excessive redundant nodes in planned paths, unsmooth paths, and failure to consider kinematic constraints. To solve these problems, this paper proposes a hybrid path planning method based on improved A* and D* Lite algorithms, incorporating multiple constraints to ensure algorithm reliability.

2.1. Improvements to Traditional Algorithms

2.1.1. Collision Detection Mechanism

When traditional algorithms use 4-directional or 8-directional neighborhoods, paths often pass through obstacle vertices during turns, which is dangerous in practical scenarios. This paper adopts a collision detection mechanism[7] to solve this problem, which includes two key components: a collision detection function and neighbor node selection. The collision detection function undertakes the key task of collision detection, determining whether the path from the start to the end collides with obstacles. If the path passes through an obstacle vertex, the current node is replaced with a preset neighbor node. This is crucial for handling path corners and avoiding crossing obstacle vertices. The neighbor node function is used to obtain neighbor nodes of a given node s, considering obstacles during selection, which indirectly helps avoid crossing obstacle vertices at path corners. The specific method is as follows:

First, check whether the two endpoints of the start and end points are obstacles themselves. If either point is in the obstacle set, it is directly determined that there is a collision. Then, for the case where the non-endpoint is directly an

obstacle, further determine whether the line between the two points intersects with the obstacle. Especially when the two points are on different abscissas and ordinates (i.e., not adjacent points in the horizontal or vertical direction), it will indirectly determine whether the line intersects with the obstacle by judging whether the diagonal vertices s1 and s2 of the rectangle formed by these two points are in the obstacle set. If s1 or s2 is in the obstacle set, it means that the line collides with the obstacle; if no collision is found after all the above detections, it indicates that the path between the two points is collision-free. The performance of the collision detection mechanism is shown in Figure 1:

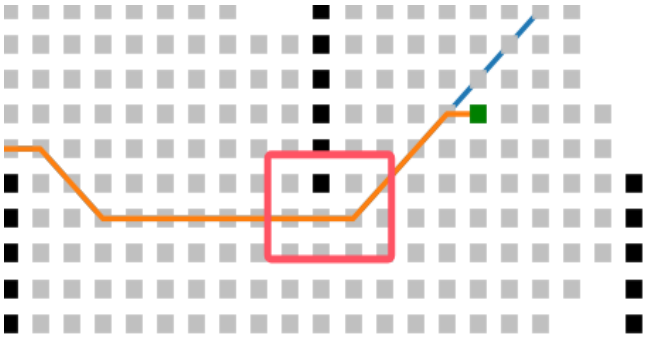

Fig.1 Performance of the collision detection mechanism

2.1.2. Redundant Node Deletion

When calculating paths using Manhattan distance[8], the search direction is limited to 4 directions, resulting in many right-angle inflection points. Although using 8-directional neighborhoods with Euclidean distance[9] effectively reduces right angles, paths may pass through obstacle vertices when moving diagonally. To solve this, this paper adopts a redundant node deletion mechanism[10]. The specific implementation involves using Manhattan distance and Euclidean distance in segments to construct a "transition route" for path compression: Euclidean distance is used in obstacle-free areas to shorten the path, while Manhattan distance is switched to in obstacle areas to avoid collisions. The implementation principle is shown in Figure 2:

Draw a straight line from the start point to each node on the path to establish a "transition route". If the straight line does not pass through obstacles, continue the same operation for the next node, and the "transition route" formed by the previous nodes becomes invalid; if the straight line passes through obstacles, the "transition route" is regarded as a "dangerous route", return to the previous node, use the "transition route" formed by the previous node as the "final route", and take this node as the new start node, continue to repeat the above operation to delete redundant nodes until the end point. The path optimized by the redundant point deletion mechanism gets rid of the limitation of traditional 8-directional neighborhood search, reduces unnecessary path node traversal, and makes the path look smoother and shorter. The implementation effect is shown in Figure 3:

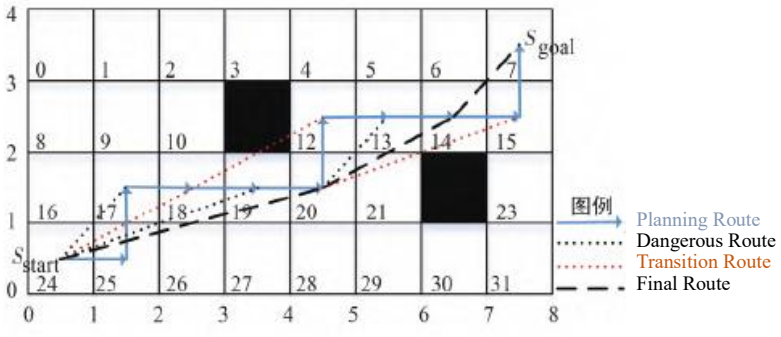

Fig.2 Redundant node deletion process

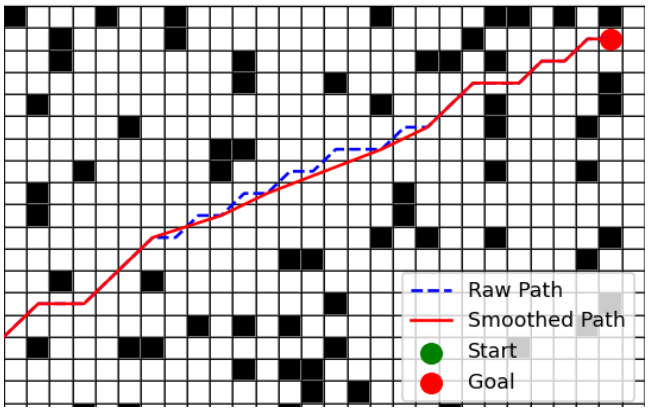

Fig.3 Effect of redundant node deletion

2.2. Multi-Scenario Algorithm Fusion Strategy

In actual operation scenarios of coil warehouses, the number of tasks is not fixed. When there is a single coil inbound/outbound task, the environment is static; in most cases, there are multiple cranes and multiple coils per crane, with fixed start or end points, making the environment a combination of static and dynamic. For non-single coil tasks, the completion of the previous coil inbound/outbound task is equivalent to adding or removing obstacles, causing local dynamic changes in the warehouse environment that require map updates. To address these actual scenarios, this paper proposes a multi-scenario hybrid strategy, implemented in three steps: initial planning phase, re-planning phase, and path optimization phase. These phases are respectively completed by the improved A*, D* Lite, and B-spline interpolation algorithms. In the initial path planning phase, the A* algorithm is used to quickly generate a feasible path. In the second phase, when new obstacles appear or the target position changes (indicating environmental changes in the warehouse), the D* Lite algorithm is used to dynamically re-plan the locally changed path segment. In the third phase, the path generated in the previous two phases is optimized using B-spline interpolation for smoothness. B-spline curve smoothing path[11] has the following advantages: local control, high flexibility, and adaptability. The implementation effect is shown in Figure 4 below:

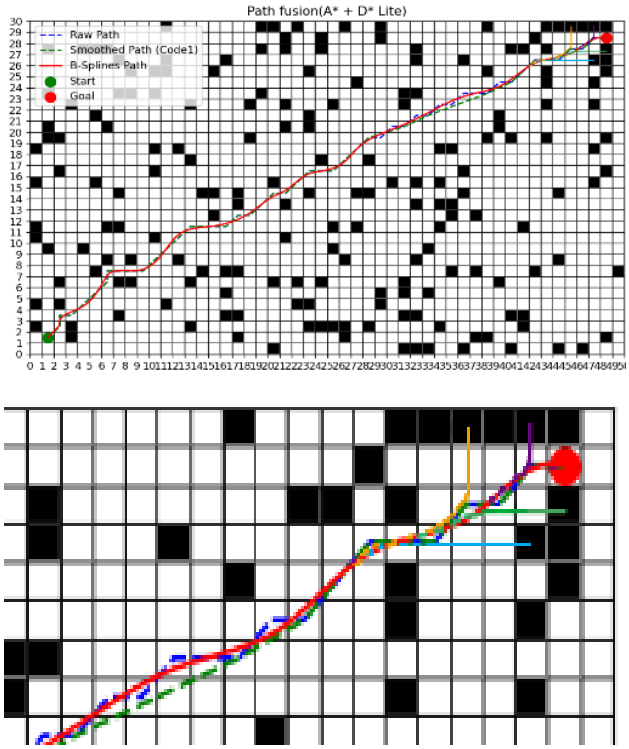

Fig.4 Implementation effect of the multi-scenario fusion algorithm

2.3. Quintic Polynomial Path Optimization

In actual operation scenarios of coil warehouses, various factors affect path planning. The overall quality of the crane's path depends not only on the algorithm but also on mechanical kinematic constraints such as speed range, acceleration limits, turning radius, and time cost. Therefore, this paper proposes a quintic polynomial[12] for further path optimization. The advantage of quintic polynomials is their ability to simultaneously consider movement speed, acceleration, turning radius, and algorithm time continuity, making crane operation safer and more reliable. Their basic forms are shown in equations (1) and (2):

X-axis:

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \quad (1)$$

Y-axis:

$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5 \quad (2)$$

Where a_0, a_1, a_2, a_3, a_4, $a_5$are polynomial coefficients, and t represents time. These functions describe the position changes of the crane in the X and Y axes over time; different coefficient combinations generate different routes. To determine the coefficients, six boundary conditions are required: initial position $s(0)$, end position $s(T)$, initial speed $s'(0)$, end speed $s'(T)$, initial acceleration $s''(0)$, and end acceleration $s''(T)$.

The first derivative of equations (1) and (2) gives the velocity function, and the second derivative gives the acceleration function. Combining velocity and acceleration values, the turning radius is obtained as in equation (3):

$$r = \frac{(1 + (y')^2)^{\frac{3}{2}}}{|y''|} \quad (3)$$

(The y − axis direction is the same as the x − axis direction)

Based on actual crane operation data, the kinematic constraint curves are shown in Fig. 5, Fig. 6, and Fig.7:

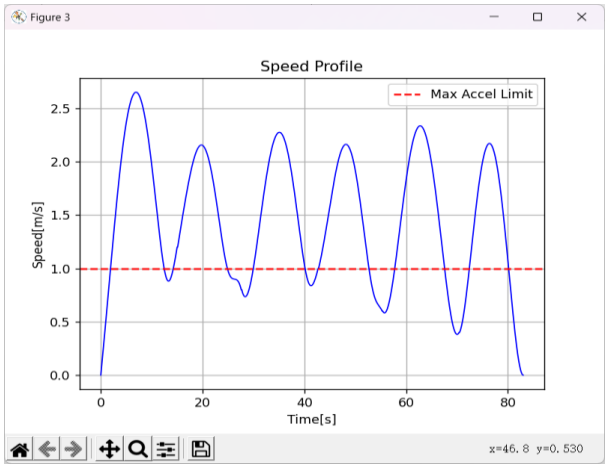

Fig.5 Speed-time curve

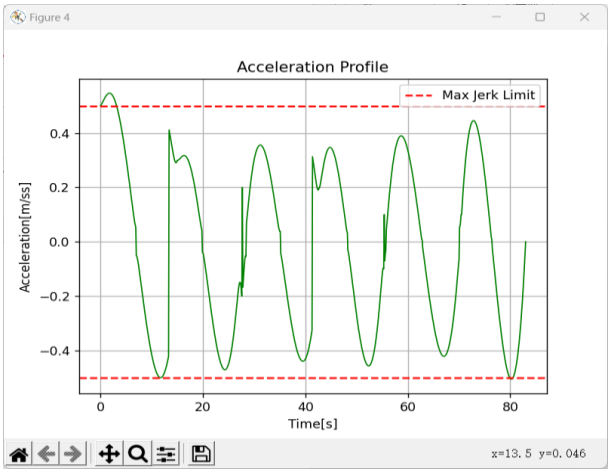

Fig.6 Acceleration-time curve

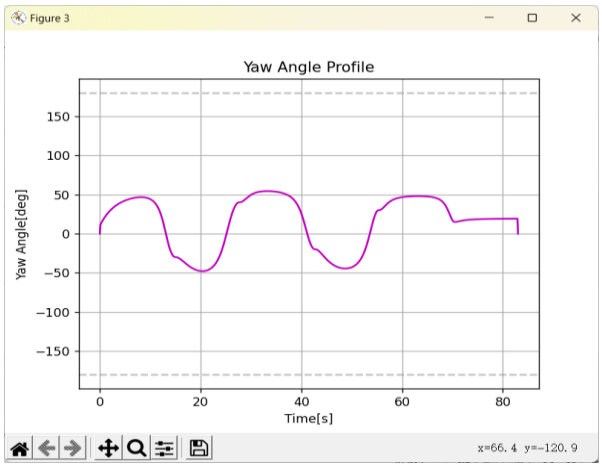

Fig. 7 Turning radius-time curve

The results of the above kinematic curves show that the improved fusion algorithm meets various kinematic constraints, which proves the effectiveness of the algorithm improvement.

3. Experiments and Result Analysis

3.1. Experimental Environment and Parameter Settings

To verify the effect of the improved and optimized algorithm, a corresponding warehouse model was constructed for testing. The experimental hardware environment is: Core i7-8750H CPU @ 2.20GHz 12 threads, 16GB memory;

software environment: Windows 11, Pycharm 2021, Python 3.8.

3.2. Modeling and Simulation

The warehouse layout is shown in Fig. 8, with saddles fixed on the ground and a total of 50*60 warehouse positions. The crane is installed above the warehouse, spanning the positions, and can move in three directions (X, Y, Z axes). Therefore, a 3D coordinate system model was established, with the corresponding 2D ground model shown in Fig.9:

Fig.8 On-site diagram of warehouse position layout

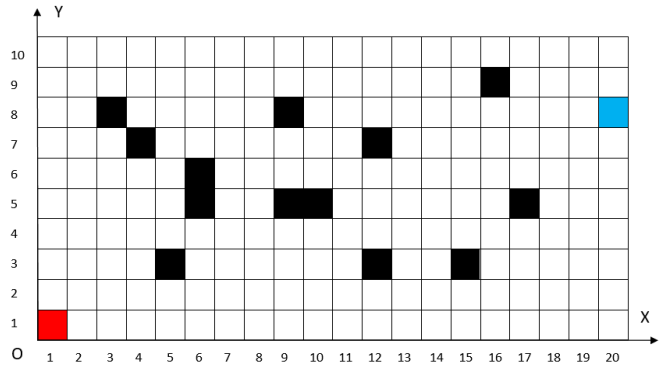

Fig.9 2D modeling diagram of warehouse positions

3.3. Experimental Result Analysis

Based on the above research, this section uses inbound and outbound scenarios to evaluate algorithm performance, with four performance metrics: path length, search time, number of turns, and number of searched grids. Experiments and analyses were conducted on five algorithms: traditional A* algorithm, traditional D* Lite algorithm, improved A* algorithm, improved D* Lite algorithm, and the improved fusion algorithm proposed in this paper. All data used in this paper are real data from actual warehouse scenarios. Specific simulation results are shown in Table 1 and Table 2:

Table 1. Algorithm performance in inbound experiments

Performance Parameter / Algorithm Type	Path length /cm	Search time /ms	Number of turns /times	Number of searched grids/count
Traditional A* algorithm	57.94	35.00	31	562
Traditional D* Lite algorithm	56.85	36.00	28	485
Improved A* algorithm	48.87	33.56	21	486
Improved D* Lite algorithm	52.30	21.64	20	309
Improved fusion algorithm	48.59	15.71	21	167.20

Table 2. Algorithm performance in outbound experiments

Performance Parameter / Algorithm Type	Path length /cm	Search time /ms	Number of turns /times	Number of searched grids/count
Traditional A* algorithm	56.47	34.17	31	576
Traditional D* Lite algorithm	57.52	37.50	27	480
Improved A* algorithm	46.92	32.70	22	479
Improved D* Lite algorithm	50.76	20.28	20	302
Improved fusion algorithm	47.34	14.63	19	156.62

Table 3. Comprehensive evaluation indicators

Evaluation Indicator	Inbound Improvement	Outbound Improvement	Average Improvement
Path length	-15.65%	-16.91%	-16.28%
Search time	-55.06%	-57.18%	-56.12%
Number of turns	-32.26%	-38.71%	-35.49%
Number of searched grids	-70.25%	-72.81%	-71.53%
Comprehensive average	-43.31%	-46.40%	-44.86%

It can be seen from Table 3 that the improved fusion algorithm has significantly improved the comprehensive evaluation indicators compared with several other algorithms during batch operations. Among them, due to the excellent node reuse capability of the improved D* Lite algorithm, only the first path spends the maximum time searching for the number of grids during batch operations, so it has a significant improvement effect. The comprehensive analysis results show that the performance of various indicators of the improved fusion algorithm has increased by about 40%.

4. Conclusion

Aiming at the problems existing in the hoisting path planning of coil warehouses, this paper comprehensively considers the shortcomings of algorithms and hardware equipment, first improves the shortcomings of traditional algorithms, then adaptively fuses the scenarios studied in this paper, and proposes an AD* Lite fusion algorithm based on kinematic constraints. The fusion algorithm can be applied to scenarios where static and dynamic coexist. Additionally, a quintic polynomial optimization strategy is proposed for

crane hardware, considering factors such as speed, acceleration, and turning radius. Simulation experiments show that the AD* Lite algorithm meets crane kinematic constraints and improves various indicators such as path length, search time, number of turns, and number of searched grids by approximately 40%, successfully verifying its adaptability and advantages in mixed static-dynamic environments, making crane operation safer, more energy-efficient, and more efficient.

References

[1] Salzmann T, Ivanovic B, Chakravarty P ,et al. Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data[J]. 2020.

[2] Yan J H, Liu C Y, Sun H X, et al. Research on Mobile Robot Path Planning Algorithm Based on Improved A* Fused with DWA[J]. Journal of Yuncheng University, 2024, 42(06): 45-51.

[3] Shi G J, Wang X W, Liu Q, et al. Robotic Arm Path Planning Based on Greedy Strategy-Improved RRT* Algorithm[J]. Manufacturing Technology & Machine Tool, 2024, (09): 29-35.

[4] Ren Q X, Feng F. 2D Path Planning for Robots Based on Dijkstra Algorithm and APO[J]. Internet of Things Technologies, 2025, 15(11): 80-83.

[5] Bagheri SM, Taghaddos H, Mousaei A, et al. An A-Star algorithm for semi-optimization of crane location and configuration in modular construction[J]. Automation in Construction, 2021,121:103447.

[6] Wang J, Qiao L Y, Han H Z, et al. Mobile Robot Path Planning Based on Improved D* Lite Algorithm[J]. China Sciencepaper, 2023,18(7):699-705.

[7] Fu Y J. Research on Path Planning of 6-DOF Industrial Robots Based on Collision Detection[J]. Science & Technology Information, 2024,22(14):37-39.

[8] Chao Y W, Yidan X, Jie Z. Voronoi treemap in Manhattan distance and Chebyshev distance[J]. Information Visualization,2023,22(3):246-264.

[9] Xiaohu H, Dezhi H, Hsiung T W, et al. A localization algorithm for DV-Hop wireless sensor networks based on Manhattan distance[J]. Telecommunication Systems,2022,81(2):207-224.

[10] Fu L, Liu F, Liu S Y, et al. Continuous Dynamic Path Planning Algorithm in 2D Based on Improved D* Lite[J]. Radio Communications Technology, 2023,49(6):1042-1051.

[11] Nguyen N T, Gangavarapu P T, Sahrhage A, et al. Navigation with polytopes and B-spline path planner[C]//2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023: 5695-5701.

[12] Li S Q, Ding X M. Trajectory Planning for Intelligent Vehicles Based on Quintic Polynomials[J]. Journal of Jiangsu University (Natural Science Edition),2023,44(04):392-398.