

Safety-Distance-Constrained Improved A* Path Planning for Mobile Robots with Cubic-Spline Smoothing

Youxu Wang

*School of Mechanical Engineering, Anhui University of Technology, Ma'anshan, China
2937678201@qq.com*

Abstract. This study develops an enhanced A* path-planning method for mobile robots operating on two-dimensional grid maps, with the aim of reducing jagged trajectories, sharp heading changes, and paths that pass too close to obstacles. The method redesigns the A* evaluation process by combining a weighted Manhattan-Euclidean heuristic, direction-aware node states, a turning-cost term, and a safety-distance cost. A cubic-spline module is then applied to the simplified path to improve trajectory continuity. Experiments are carried out in MATLAB under simple-obstacle, complex-obstacle, and narrow-passage maps, and the proposed algorithm is evaluated against conventional A*, Dijkstra, and RRT methods. Relative to conventional A*, the proposed approach reduces the average path length by 4.93%, turns by 83.34%, searched nodes by 71.46%, planning time by 56.77%, and smoothness index by 84.85%, while increasing the average minimum distance to obstacles by 80.47%. The results show that the revised evaluation strategy and smoothing procedure can enhance search efficiency, path continuity, and safety clearance while retaining collision-free planning.

Keywords: mobile robot, path planning, improved A* algorithm, directional state, turning penalty

1. Introduction

Mobile robots are now used in warehouse distribution, inspection tasks, healthcare-related transportation, campus delivery, and indoor service scenarios. These applications place stricter demands on autonomous navigation because a robot must move from a start position to a goal position without collision while also accounting for travel cost, computational burden, motion stability, and clearance from surrounding obstacles [1].

Existing path-planning approaches are commonly organized into graph-search, sampling-based, intelligent optimization, and learning-based categories. Graph-search algorithms are particularly suitable for grid representations because their search logic is explicit and their outputs are relatively stable. Sampling-based planners are useful in complicated spaces, whereas optimization-based and learning-based planners emphasize global exploration and adaptation [2]. In practical grid-map planning, however, several issues remain, including dense obstacle layouts, routes that lie close to obstacle edges, polyline trajectories caused by discrete grid nodes, and the need to balance search efficiency with path quality [3].

A* employs a heuristic term to guide node expansion and reduce unnecessary search, so it has become a common global planner for mobile robots [4]. Nevertheless, the standard form of A* mainly evaluates nodes through distance-related cost. It does not fully use information about incoming direction, accumulated turns, obstacle-clearance margins, or the trackability of the final trajectory. Therefore, the resulting route may contain many short line segments, abrupt changes in heading, or insufficient safety distance.

To address these limitations, this work focuses on two-dimensional grid maps and modifies the evaluation mechanism of conventional A*. A weighted combination of Manhattan and Euclidean distances is adopted to strengthen the orientation of the search. A direction state is then added to record the motion direction of each expanded node. On this basis, turning and safety-distance penalties are introduced to discourage sharp direction changes and nodes close to obstacles. Finally, cubic-spline interpolation is used to smooth the simplified path, and the method is compared with conventional A*, Dijkstra, and RRT algorithms in simple-obstacle, complex-obstacle, and narrow-passage scenarios.

2. Related work

Studies on mobile robot path planning cover graph-search methods, sampling-based strategies, intelligent optimization algorithms, and learning-based schemes. In grid maps, graph-search methods expand nodes according to a predefined connectivity structure; they are easy to implement and their results are generally reproducible. Dijkstra performs an exhaustive shortest-path search but usually visits many nodes, while A* uses heuristic guidance to improve the trade-off between path quality and search cost [1, 2]. RRT-type planners can rapidly find feasible paths in complex spaces, but their randomness often leads to unstable paths that require subsequent smoothing [3, 4]. Intelligent optimization and learning-based methods can improve adaptability, yet their computational cost, parameter sensitivity, and interpretability should be carefully considered in small static two-dimensional maps.

Recent improvements to A* have mainly concentrated on heuristic design, neighborhood expansion, obstacle-aware constraints, and post-processing. Modified heuristic functions and greedy mechanisms can strengthen search directionality and reduce invalid expansions [5, 6]. Geometric constraints and fusion pathfinding strategies incorporate map-structure information into the planning process and help improve route quality [7, 8]. Studies on grid-map planning and medical-laboratory robots also show that obstacle clearance and safety constraints are important for path executability [9, 10]. Adaptive neighborhoods and motion-block mechanisms further adjust expansion modes, shorten search time, and reduce the number of searched nodes [11, 12]. These findings indicate that A* performance is influenced not only by distance measurement but also by direction continuity, turn suppression, obstacle clearance, and smoothing after search.

In summary, existing algorithms can usually obtain feasible routes, but static two-dimensional grid maps still tend to produce jagged paths, sudden turns, near-obstacle trajectories, and redundant search. For mobile robots, a useful path should satisfy collision-free requirements while also considering length, computational efficiency, smoothness, and safety distance. For this reason, this paper embeds a weighted heuristic, direction states, a turning penalty, and a safety-distance penalty into the A* framework, and then uses cubic-spline interpolation to improve the continuity of the output trajectory.

3. Improved A* algorithm

In response to redundant path length, poor trajectory smoothness, possible near-obstacle motion, and limited search efficiency in conventional A*, an improved algorithm is designed. The proposed method adjusts the heuristic term, records direction-related node states, introduces turning and safety-distance costs, and then smooths the generated route through cubic-spline interpolation.

3.1. Optimization of the heuristic function

The baseline A* algorithm often adopts Euclidean distance as its heuristic estimate. Although this metric measures the straight-line distance between the current node and the goal, it does not sufficiently reflect the directional characteristics of grid-based movement. To provide stronger target guidance, Manhattan distance and Euclidean distance are combined with weight coefficients, and an additional heuristic weight is introduced. The optimized heuristic function is defined as follows:

$$h(n) = w_h [w_1 h_M(n) + w_2 h_E(n)] \quad (1)$$

where w_h is the heuristic weight, which is set to 1.15 in this paper. w_1 and w_2 denote the weight coefficients of Manhattan distance and Euclidean distance, respectively, satisfying $w_1 + w_2 = 1$. In this paper, $w_1 = 0.45$ and $w_2 = 0.55$. The Manhattan distance $h_M(n)$ is expressed as:

$$h_M(n) = |x_n - x_g| + |y_n - y_g| \quad (2)$$

where (x_n, y_n) denotes the coordinate of the current node, and (x_g, y_g) denotes the coordinate of the target node. The weighted heuristic function considers both geometric distance and grid-direction constraints, which helps improve search efficiency and maintain stable path quality under different obstacle distributions.

3.2. Search state considering direction

In the original A* search process, a node is usually represented only by its row and column coordinates, namely $n=(r,c)$. Such a representation cannot indicate the direction from which the robot enters the current grid. To distinguish continuous motion directions and prepare for the subsequent turning-cost calculation, a direction variable is included in the node state. The improved state is written as follows:

$$n = (r, c, d) \quad (3)$$

where r denotes the row of the current node, c denotes the column of the current node, and d denotes the direction from which the robot enters the current grid. In this paper, directions are divided into eight types: up, down, left, right, upper left, upper right, lower left, and lower right. At the start point, $d=0$ indicates that there is no incoming direction. By recording the direction vector between the current node and its parent node, the algorithm can determine whether the next

expansion involves a turn, thereby making the planned path more likely to choose routes with smaller directional changes.

3.3. Turning penalty

Node selection in traditional A* is mainly governed by distance cost, so frequent turning segments may still appear in passable regions. In real mobile-robot operation, repeated direction changes increase control difficulty and energy consumption and may reduce tracking stability. Therefore, this paper incorporates a turning penalty into the cost function. When two adjacent path segments have different directions, the additional cost is computed as follows:

$$P_t(d, dt) = \frac{1}{\pi} \arccos \left(\frac{\mathbf{v}_d \cdot \mathbf{v}_{dt}}{\|\mathbf{v}_d\| \|\mathbf{v}_{dt}\|} \right) \quad (4)$$

where $\mathbf{v}_d$ denotes the current incoming direction vector of the robot, and $\mathbf{v}_{dt}$ denotes the next movement direction vector. This expression characterizes directional change by the angle between vectors and normalizes it by π , so the penalty value approximately falls within [0,1]. A larger directional change produces a larger penalty. If the current node is the start point or has no directional state, the turning penalty is set to 0.

3.4. Safety-distance penalty

Standard A* only prevents the path from crossing obstacle cells; it does not necessarily keep enough clearance from obstacle boundaries. Since a real robot has physical dimensions and may be affected by localization error, a safety-distance penalty is added to the evaluation cost. The penalty depends on the distance between a candidate node and the nearest obstacle. If this distance is smaller than the preset safety radius, the cost of the candidate node is increased:

$$P_s(n') = \max \left(0, \frac{R - D_{obs}(n')}{R} \right) \quad (5)$$

where R is the safety radius, which is set to 4.0 in this paper, and $D_{obs}(n')$ denotes the Euclidean distance from candidate node n' to the nearest obstacle. The closer the candidate node is to an obstacle, the larger the penalty value becomes. When the distance is not smaller than the safety radius, the penalty value is 0. This design can guide the planned path to actively stay away from obstacle boundaries and improve passage safety in complex-obstacle and narrow-passage environments.

3.5. Integrated improved A* algorithm

Based on the above improvements, the weighted heuristic function, directional state, turning penalty, and safety-distance penalty are integrated into a unified evaluation function. The evaluation function of the improved A* algorithm is:

$$f(n', d') = g(n', d') + h(n') \quad (6)$$

where $h(n)$ is calculated according to equation (1), and $g(n', d')$ denotes the cumulative cost considering direction, the turning penalty, and the safety-distance penalty. It is expressed as:

$$g(n', d') = g(n, d) + c_{move} + \lambda_t P_t + \lambda_s P_s \quad (7)$$

where $g(n, d)$ denotes the cumulative cost of reaching the current state. c_{move} is the basic movement cost, which is 1 for horizontal or vertical movement and $\sqrt{2}$ for diagonal movement. $P_t(d, d')$ is the turning penalty, $P_s(n')$ is the safety-distance penalty, and λ_t and λ_s are penalty weights. In this paper, $\lambda_t=0.35$ and $\lambda_s=0.45$. By adjusting these two penalty weights, a balance can be achieved among path length, search efficiency, smoothness, and safety distance.

3.5.1. Pathfinding procedure of the improved A* algorithm

The improved A* algorithm still uses an open set and a closed set to maintain the search process. Let the current state be $q=(r, c, d)$, where (r, c) denotes the grid coordinate and d denotes the incoming direction. The candidate neighborhood is denoted as $N_8(q)$. In each iteration, the state with the minimum evaluation function in the open set is selected as the current expansion node:

$$q^* = \arg \min_{q \in open \ set} f(q) \quad (8)$$

For any candidate neighbor $q' \in N_8(q^*)$, if the node is an obstacle or has already been included in the closed set, it is skipped. Otherwise, the basic movement cost is calculated according to horizontal, vertical, or diagonal movement:

$$c_{move} = \begin{cases} 1, & \text{straight} \\ \sqrt{2}, & \text{diagonal} \end{cases} \quad (9)$$

Then, a temporary cost containing the turning penalty and safety-distance penalty is calculated:

$$g_{temp} = g + c_{move} + \lambda_t P_t + \lambda_s P_s \quad (10)$$

If the temporary cost is smaller than the original cumulative cost, the parent node, incoming direction, and cumulative cost of q' are updated, and the corresponding evaluation function is recalculated. When the target node is selected as the current expansion node, the algorithm stops searching and obtains the original path by backtracking from the target point along the parent-node chain. This process allows the algorithm to consider distance, directional changes, and obstacle safety margins during the search stage rather than passively correcting the path after generation.

In the code implementation, the open set records nodes to be expanded, and the closed set records nodes that have already been expanded. parentRow and parentCol are used to save parent-node coordinates. obstacleDistanceMap is precomputed to store the distance from each feasible grid to the nearest obstacle, thereby reducing repeated computation of the safety-distance penalty in the loop.

3.5.2. Smoothing post-processing based on cubic spline interpolation

Although the turning penalty reduces unnecessary heading changes, the route produced by grid search is still composed of discrete nodes and may contain local polyline segments. To improve trajectory continuity, line-of-sight checking is first used to remove redundant intermediate nodes. Cubic-spline interpolation is then applied to the simplified point sequence. Let the simplified path be $P_i = (x_i, y_i)$, $i=0,1,\dots,m$, and define the chord-length parameter as follows:

$$s_0 = 0, \quad s_i = s_{i-1} + \|P_i - P_{i-1}\|_2 \quad (11)$$

Cubic spline functions $x=S_x(s)$ and $y=S_y(s)$ are constructed for the row and column coordinates, respectively. In the interval $[s_i, s_{i+1}]$, the spline function can be expressed as:

$$S_i(s) = a_i + b_i \Delta s + c_i \Delta s^2 + d_i \Delta s^3, \quad \Delta s = s - s_i \quad (12)$$

To ensure curve continuity, the spline function must satisfy node interpolation conditions as well as first- and second-order derivative continuity conditions:

$$\begin{cases} S_i(s_i) = P_i, & S_i(s_{i+1}) = P_{i+1} \\ S_i'(s_{i+1}) = S_{i+1}'(s_{i+1}) \\ S_i''(s_{i+1}) = S_{i+1}''(s_{i+1}) \end{cases} \quad (13)$$

During implementation, no fewer than 80 interpolation samples are generated according to the total path length, and the start and goal points are kept fixed. Collision checking is then performed on the interpolated polyline segments to prevent the smoothed curve from crossing obstacles. If the collision test is passed, the spline trajectory is retained; otherwise, the simplified polyline is used. This procedure reduces abrupt turns while maintaining collision-free feasibility.

4. Simulation experiment verification

To evaluate the improved A* method described in Section 3, comparative simulations are performed on MATLAB-based two-dimensional grid maps. Conventional A*, Dijkstra, and RRT are selected as baseline algorithms. Dijkstra is used as a deterministic shortest-path benchmark and is also common in recent studies of mobile-robot path planning and obstacle avoidance [13]. RRT and related variants represent sampling-based planners, making them suitable for comparing the proposed deterministic search method with random-sampling approaches in terms of path length and smoothness [14].

4.1. Simulation experiment and comparative analysis

The simulation platform is Matlab R2022b. The computer configuration includes Windows 11 Home, an Intel Core i5-13500HX processor at 2.50 GHz, and 16 GB of memory. The comparison algorithms include traditional A*, Dijkstra, and RRT. Recent RRT-based methods often improve convergence speed and path quality through bidirectional search, goal bias, or node optimization [15], and some studies combine artificial potential fields with RRT* to enhance search efficiency in obstacle environments [16].

Six criteria are used in the evaluation: path length, number of turns, searched nodes, planning time, minimum obstacle distance, and smoothness. Path length measures movement cost; turns and smoothness describe trajectory continuity; searched nodes and planning time characterize computational efficiency; and minimum obstacle distance reflects safety clearance. Prior studies on path smoothing indicate that curve-fitting post-processing can reduce angular discontinuities in discrete routes [17]. Cubic splines are also widely used to improve the trackability of mobile-robot paths [18]. The simple-obstacle environment is first used as an illustrative case, and the planned routes are shown in Figure 1.

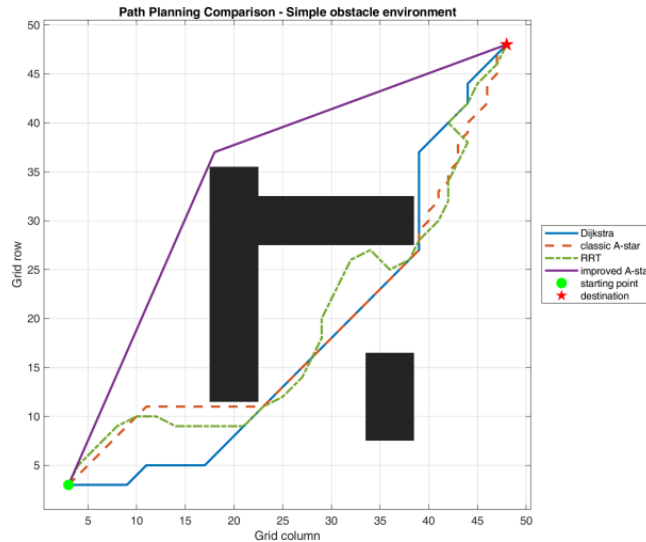


Figure 1. Path planning comparison in a simple-obstacle environment

Using the evaluation function, the indicators of each algorithm in this environment are obtained, as shown in Table 1. Table 1 presents the six indicators of each algorithm under the same starting point and the same grid environment.

Table 1. Evaluation indicators in the simple-obstacle environment

Algorithm	Path length	Turns	Searched nodes	Planning time (s)	Minimum obstacle distance	Smoothness
Dijkstra	70.669	7	2241	0.08325	1.000	5.498
Traditional A*	70.669	18	838	0.02396	1.000	14.137
RRT	76.469	24	162	0.01769	1.000	13.853
Improved A*	69.115	1	173	0.00756	2.000	0.804

As indicated by Figure 1 and Table 1, the improved A* path length in the simple-obstacle environment is 69.115, which is 2.20% shorter than that of conventional A*. The number of turns is

reduced from 18 to 1, corresponding to a 94.44% decrease. The searched nodes are reduced from 838 to 173, and the planning time decreases from 0.02396 s to 0.00756 s, with reductions of 79.36% and 68.44%, respectively. In addition, the minimum obstacle distance increases from 1.000 to 2.000, and the smoothness index decreases from 14.137 to 0.804. These results suggest that the turning cost, safety-distance cost, and smoothing step improve path quality in the simple-obstacle case.

Similarly, the grid map is changed to a complex-obstacle environment. Some dead-end regions are designed in the map to increase planning and exploration difficulty, and the differences among the planned paths of the algorithms are observed. The path planning comparison obtained by executing the program is shown in Figure 2.

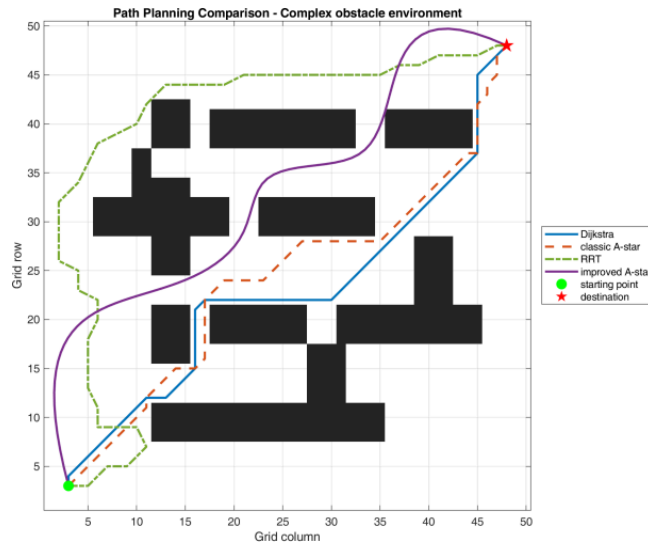


Figure 2. Path planning comparison in a complex-obstacle environment

The evaluation indicators of each algorithm in this environment are calculated in the same way and are listed in Table 2. The algorithms are compared under the same starting point and the same environment.

Table 2. Evaluation indicators in the complex-obstacle environment

Algorithm	Path length	Turns	Searched nodes	Planning time (s)	Minimum obstacle distance	Smoothness
Dijkstra	72.426	9	1965	0.04516	1.000	7.069
Traditional A*	72.426	17	868	0.01979	1.000	14.137
RRT	94.619	28	444	0.00672	1.000	19.777
Improved A*	68.451	5	242	0.00842	1.414	3.689

Figure 2 and Table 2 show that, in the complex-obstacle environment, the improved A* algorithm obtains a path length of 68.451, which is 5.49% shorter than conventional A*. The number of turns decreases from 17 to 5, a reduction of 70.59%. The searched nodes decrease from 868 to 242, and the planning time decreases from 0.01979 s to 0.00842 s, corresponding to reductions of 72.12% and 57.44%. Although the more complicated obstacle distribution increases detour difficulty, the proposed method still yields a shorter path, fewer searched nodes, and a smaller smoothness index. This result indicates that the weighted heuristic and safety-distance term help suppress ineffective search and local jagged segments.

A narrow-passage grid map is then established. Obstacles are arranged to form a map with a surrounding trend and limited channels, and the width of passage points is appropriately reduced to increase the difficulty of safety-aware planning. The path planning comparison of each algorithm is shown in Figure 3.

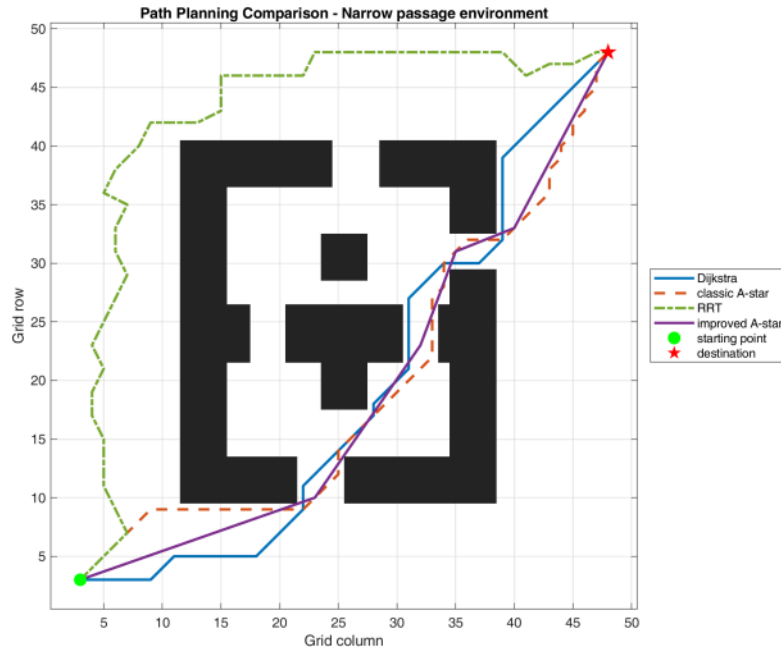


Figure 3. Path planning comparison in a narrow-passage environment

The indicators of different algorithms in the narrow-passage environment are obtained in the same way, as shown in Table 3.

Table 3. Evaluation indicators in the narrow-passage environment

Algorithm	Path length	Turns	Searched nodes	Planning time (s)	Minimum obstacle distance	Smoothness
Dijkstra	73.012	13	2033	0.03452	1.000	10.210
Traditional A*	73.012	20	717	0.01562	1.000	15.708
RRT	90.327	25	346	0.00173	2.000	20.486
Improved A*	67.834	3	266	0.00868	2.000	2.149

As shown in Figure 3 and Table 3, the improved A* algorithm achieves a path length of 67.834 in the narrow-passage environment, which is 7.09% shorter than conventional A*. The number of turns decreases from 20 to 3, giving an 85.00% reduction. The searched nodes decrease from 717 to 266, and the planning time decreases from 0.01562 s to 0.00868 s, corresponding to reductions of 62.90% and 44.41%, respectively. The minimum obstacle distance reaches 2.000, twice that of conventional A*, which shows that the safety-distance penalty can guide the path away from obstacle boundaries in narrow channels.

Across the three test maps, the improved A* algorithm reduces the average path length by 4.93%, the average turns by 83.34%, the average searched nodes by 71.46%, the average planning time by 56.77%, and the average smoothness index by 84.85% compared with conventional A*. At the same time, the average minimum obstacle distance increases by 80.47%. Relative to Dijkstra, the proposed method reduces the average searched nodes by 88.96% and planning time by 82.37%.

Relative to RRT, it reduces average path length by 20.72%, average turns by 88.66%, and average smoothness index by 88.35%. These results demonstrate that the proposed method can improve search efficiency, trajectory continuity, and passage safety while preserving collision-free planning.

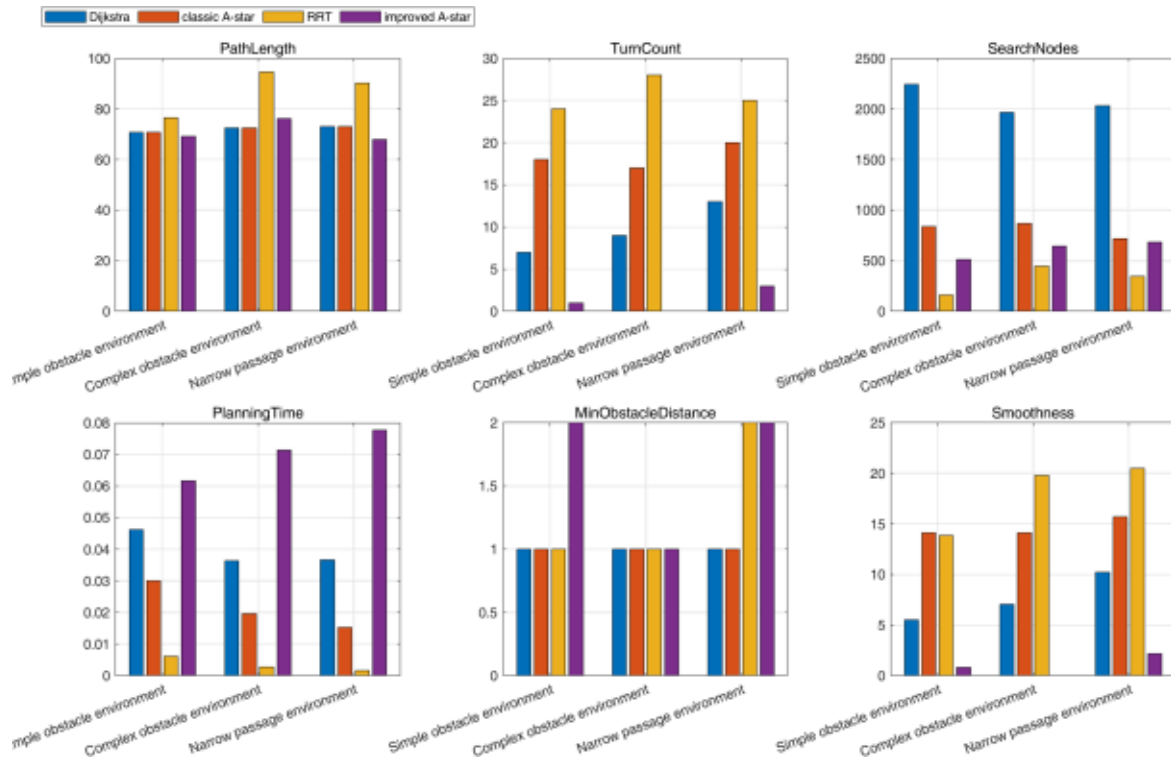


Figure 4. Comprehensive comparison of evaluation indicators

5. Conclusion

This paper presents a safety-distance-constrained improved A* algorithm for mobile robot path planning in two-dimensional grid maps. The method combines a weighted heuristic, direction-aware node states, turning-cost suppression, safety-distance constraints, and cubic-spline smoothing to reduce jagged segments, abrupt heading changes, redundant search, and near-obstacle routes. Simulations in simple-obstacle, complex-obstacle, and narrow-passage environments show that, compared with conventional A*, the proposed method reduces average path length by 4.93%, turns by 83.34%, searched nodes by 71.46%, planning time by 56.77%, and smoothness index by 84.85%, while increasing the average minimum obstacle distance by 80.47%. These results verify that the method improves path continuity and safety clearance while maintaining efficient collision-free search. The present study is still limited to static two-dimensional grid maps and treats the robot as a point-mass model. Future research will consider dynamic obstacles, robot-size inflation, chassis kinematic constraints, and real-platform experiments to improve practical applicability in complex environments.

References

- [1] Yang, L., Li, P., Qian, S., Quan, H., Miao, J., Liu, M., et al. (2023). Path planning technique for mobile robots: A review. *Machines*, 11(10), 980. <https://doi.org/10.3390/machines11100980>
- [2] Abdulsahab, J. A., & Kadhim, D. J. (2023). Classical and heuristic approaches for mobile robot path planning: A survey. *Robotics*, 12(4), 93. <https://doi.org/10.3390/robotics12040093>

- [3] Tang, Y., Zakaria, M. A., & Younas, M. (2025). Path planning trends for autonomous mobile robot navigation: A review. *Sensors*, 25(4), 1206. <https://doi.org/10.3390/s25041206>
- [4] Tan, C. S., Mohd-Mokhtar, R., & Arshad, M. R. (2021). A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. *IEEE Access*, 9, 119310-119342. <https://doi.org/10.1109/access.2021.3108177>
- [5] Zhang, L., & Li, Y. (2021). Mobile robot path planning algorithm based on improved A Star. *Journal of Physics: Conference Series*, 1848(1), 012013. <https://doi.org/10.1088/1742-6596/1848/1/012013>
- [6] Tang, G., Tang, C., Claramunt, C., Hu, X., & Zhou, P. (2021). Geometric A-Star algorithm: An improved A-Star algorithm for AGV path planning in a port environment. *IEEE Access*, 9, 59196-59210. <https://doi.org/10.1109/access.2021.3070054>
- [7] Liu, Z., Liu, H., Lu, Z., & Zeng, Q. (2021). A dynamic fusion pathfinding algorithm using Delaunay triangulation and improved A-Star for mobile robots. *IEEE Access*, 9, 20602-20621. <https://doi.org/10.1109/access.2021.3055231>
- [8] Xiang, D., Lin, H., Ouyang, J., & Huang, D. (2022). Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot. *Scientific Reports*, 12(1), 13273. <https://doi.org/10.1038/s41598-022-17684-0>
- [9] Ou, Y., Fan, Y., Zhang, X., Lin, Y., & Yang, W. (2022). Improved A* path planning method based on the grid map. *Sensors*, 22(16), 6198. <https://doi.org/10.3390/s22166198>
- [10] Yin, C., Tan, C., Wang, C., & Shen, F. (2024). An improved A-Star path planning algorithm based on mobile robots in medical testing laboratories. *Sensors*, 24(6), 1784. <https://doi.org/10.3390/s24061784>
- [11] Huang, J., Chen, C., Shen, J., Liu, G., & Xu, F. (2025). A self-adaptive neighborhood search A-star algorithm for mobile robots global path planning. *Computers and Electrical Engineering*, 123, 110018. <https://doi.org/10.1016/j.compeleceng.2024.110018>
- [12] Kabir, R., Watanobe, Y., Islam, M. R., & Naruse, K. (2024). Enhanced robot motion block of A-Star algorithm for robotic path planning. *Sensors*, 24(5), 1422. <https://doi.org/10.3390/s24051422>
- [13] Alshammrei, S., Boubaker, S., & Kolsi, L. (2022). Improved Dijkstra algorithm for mobile robot path planning and obstacle avoidance. *Computers, Materials & Continua*, 72(3), 5939-5954. <https://doi.org/10.32604/cmc.2022.028165>
- [14] Xin, P., Wang, X., Liu, X., Wang, Y., Zhai, Z., & Ma, X. (2023). Improved bidirectional RRT* algorithm for robot path planning. *Sensors*, 23(2), 1041. <https://doi.org/10.3390/s23021041>
- [15] Li, X., & Tong, Y. (2024). Path planning of a mobile robot based on the improved RRT algorithm. *Applied Sciences*, 14(1), 25. <https://doi.org/10.3390/app14010025>
- [16] Wu, D., Wei, L., Wang, G., Tian, L., & Dai, G. (2022). APF-IRRT*: An improved informed rapidly-exploring random trees-star algorithm by introducing artificial potential field method for mobile robot path planning. *Applied Sciences*, 12(21), 10905. <https://doi.org/10.3390/app122110905>
- [17] Li, F.-F., Du, Y., & Jia, K.-J. (2022). Path planning and smoothing of mobile robot based on improved artificial fish swarm algorithm. *Scientific Reports*, 12(1), 659. <https://doi.org/10.1038/s41598-021-04506-y>
- [18] Ma, X., Gong, R., Tan, Y., Mei, H., & Li, C. (2022). Path planning of mobile robot based on improved PRM based on cubic spline. *Wireless Communications and Mobile Computing*, 2022(1), 1632698. <https://doi.org/10.1155/2022/1632698>