Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/33/20230262

Modified A* algorithm for path smoothing and obstacle avoidance

Hui Geng

School of Mechanical Engineering and Automation, Dalian Polytechnic University, Dalian, Liaoning, 116034, China

S20004310@MAILGLYNDWRAC.onmicrosoft.com

Abstract. With the rapid development of robotics technology, path planning is a crucial aspect of autonomous robot systems. Among them, planning paths involves using the A* algorithm, which is a common method. However, traditional A* algorithm has several limitations in path planning, such as poor real-time performance, large amount of computation per node, long computation time, low algorithmic search efficiency. Based on this, two improved approaches for the A* algorithm are proposed. The first is expanding the obstacles in the map by increasing their expansion radius. The second is the Hybrid A* algorithm, which optimizes the A* algorithm by modifying its heuristic function. Specifically, the Hybrid A* algorithm combines two heuristic functions: one based on non-holonomic constraints and the other based on dynamic programming. Experimental tests are conducted under various map expansions and branching parameters to compare the performance of these two algorithms in terms of path length, execution time, and path smoothness at corners. The results demonstrate that, with smaller branching parameters, the Hybrid A* algorithm generates shorter paths. However, in highly complex mazes, the path length of the Hybrid A* algorithm may be longer, but it exhibits smoother movements at corners.

Keywords: path-planning, A* algorithm, Hybrid A* algorithm, optimization methods.

1. Introduction

With the continuous development of related technologies, robots have been utilized in a wide range field such as manufacturing, aerospace, deep-sea exploration, and healthcare. To enhance the intelligence and autonomy of robots in task handling, researchers have proposed various path-planning methods. After years of research, there are two types of route planning techniques: conventional path-planning algorithms and clever biomimetic path-planning algorithms. A*, D*, artificial potential field method, and RRT are examples of traditional route planning techniques. Biomimetic path planning techniques include ant colony algorithm, particle swarm optimization algorithm, and genetic algorithm, among others. Among these methods, one of the most established and widely used path-planning techniques is the A* algorithm. It is a Dijkstra algorithm extension that has several uses in manufacturing and pathfinding in video games., among other areas [1].

The A* algorithm, by considering both actual costs and estimates from heuristic functions, can find the optimal path in graph search problems. It has been widely applied in the fields of computer science and artificial intelligence for path-planning, game AI, and decision-making in intelligent systems [2].

However, when dealing with complex issues and continuous spaces, the conventional A* method has several drawbacks. These limitations include constraints on the search space size, challenges in selecting appropriate heuristic functions, and concerns regarding completeness and optimality guarantees. Additionally, traditional A* algorithm may encounter issues such as an excessive number of turning points, lack of path smoothness, and longer path distances [3].

This paper aims to overcome the limitations of traditional A* method. An optimization method is proposed in this study to enhance the performance and efficiency of the A* algorithm. These optimization techniques include improving heuristic functions and adjusting obstacle distances in the map. By optimizing the A* algorithm, the quality and efficiency of path planning can be improved, enabling robots to intelligently plan their trajectories and achieve better performance in real-world tasks.

2. The traditional A* algorithm

An efficient algorithm for finding the optimal path in the same environment is the traditional A* algorithm, which is derived from the Dijkstra algorithm [4]. While ensuring the optimal planned path, it swiftly approaches the goal guided by the evaluation function $f(n)$ [5]. The A* algorithm's heuristic function is written as follows:

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

$f(n)$ - The heuristic function that estimates the cost from a node through node n to the destination [6];

$g(n)$ - The actual cost function from a point to node n [6];

$h(n)$ - The actual cost function from a point to node n [6];

If $h(n)$ = 0, the A* algorithm becomes equivalent to the Dijkstra algorithm, and it may be utilized to find the shortest distance. The A* method can still look for the shortest path in circumstances when the anticipated value of $h(n)$ is smaller than the actual cost from the current node to the destination node. However, as $h(n)$ decreases, the efficiency of the search decreases due to an increase in expanded nodes [7]. If the estimated distance $h(n)$ is equal to the actual cost between a node and the destination node, the A* algorithm can efficiently and accurately search for the optimal path without generating additional nodes. This results in high search efficiency and yields the optimal path. However, if the value of $h(n)$ exceeds an alternative value between the node and the target location, it may not find the optimal path but can still improve relative search efficiency. The A* algorithm transforms into the Breadth-First Search (BFS) algorithm when $g(n)$ = 0. Considering the main characteristics of the A* algorithm's scoring function, striking a balance between $g(n)$ and $h(n)$ is crucial for refining the A* algorithm [8].

3. Improvement methods for the A* algorithm

3.1. Expansion of obstacle

The traditional A* algorithm may generate paths that are in close proximity to obstacles, and it can potentially lead to situations of collision or "pass-through" in simulations. If a mobile robot relies solely on A* for path planning, it poses a significant risk. Applying such a method in real-world scenarios would likely result in persistent collisions or frequent encounters with obstacles [9]. Hence, in path planning, it is essential to introduce an expansion radius for obstacles in the map. This not only enables the mobile robot to navigate more precisely toward the target but also reduces the computational time required. The mobile robot operates within a binary maze, as depicted in Figure 1, where it traverses and adapts to the environment. For this experiment, the robot model chosen is circular, and the grid size was selected as the extent of the expansion distance.

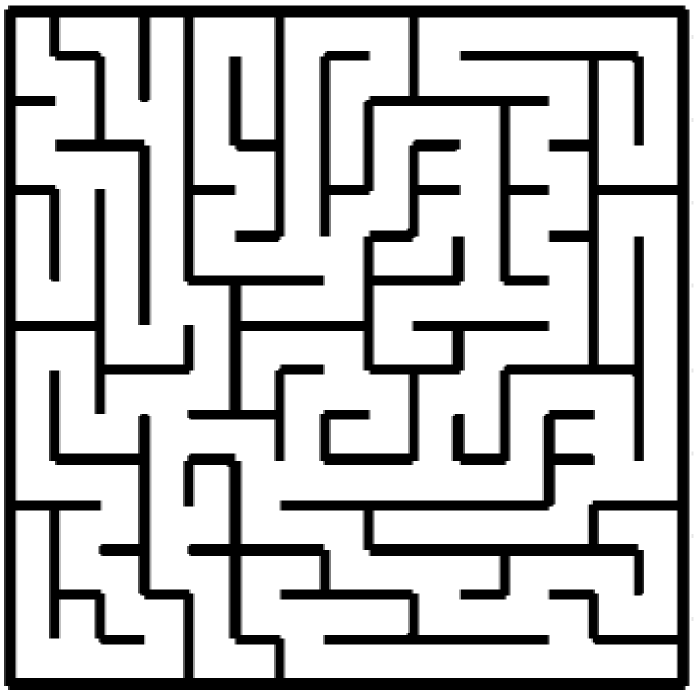

Figure 1. Binary occupancy grid.

Expanding the distance of obstacles essentially alters the sensing range of the mobile robot. This allows for a more precise path between the robot and the obstacles. During this process, it is important to ensure that the robot has sufficient space to maneuver while maximizing efficiency in the chosen path. The size of the expansion radius is determined by the radius of the robot itself. Figure 2 illustrates the map after radius expansion.

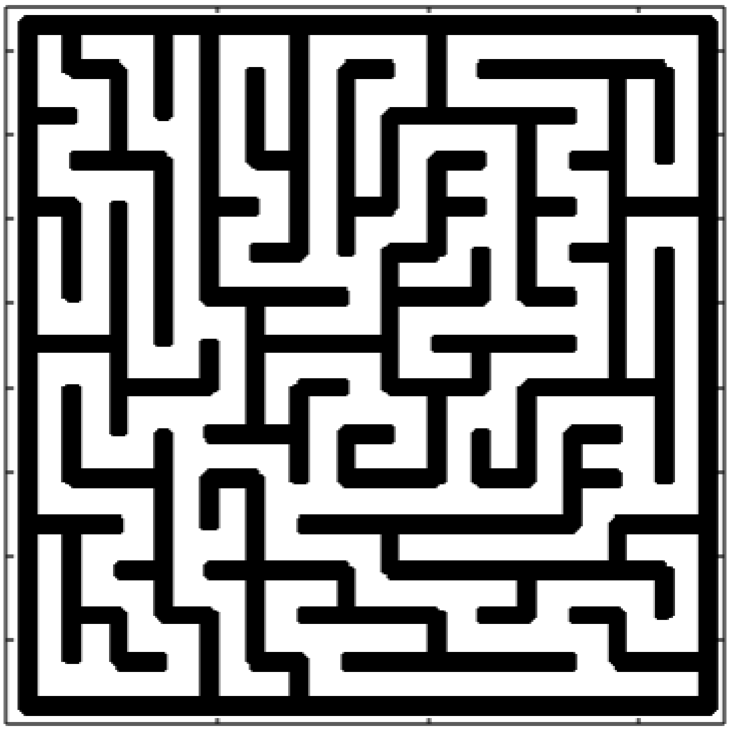

Figure 2. Map of the expanded radius.

3.2. *Optimized search algorithm*

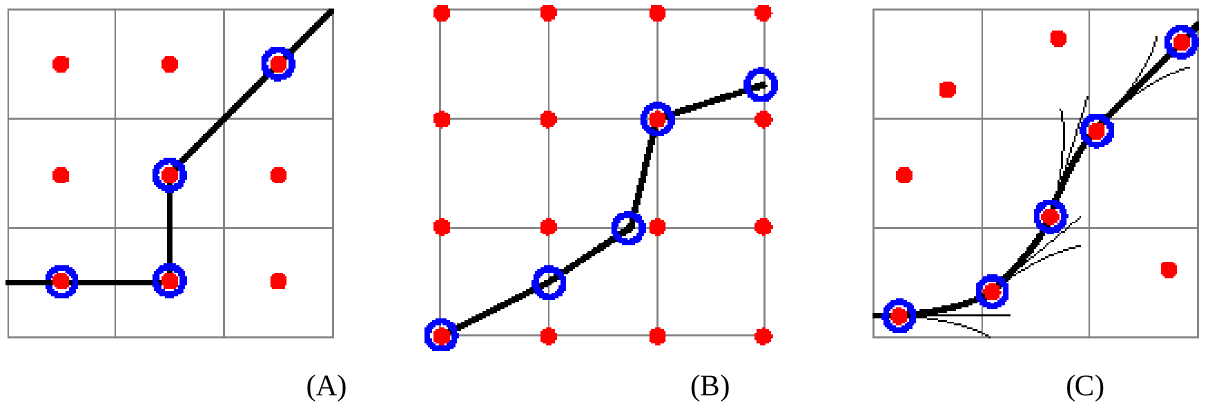

Figure 3. Comparison of search algorithms visually. (A): A* only travels to states that have grid-cell centers in common with their expenses. (B): Field D* (Ferguson and Stentz 2005) and Theta* (Nash et al. 2007) enable arbitrary linear pathways between cells and correlate costs with cell corners. (C): Hybrid A* assigns a continuous state to each cell, and the value of that continuous state's cost determines the cell's score [10].

A path-planning method utilized in autonomous driving is called the Hybrid A* algorithm. It combines both continuous state space and discrete state space to solve path-planning problems. The fundamental idea behind this approach is to include continuous state data into the discrete search nodes of the A* algorithm, allowing for the use of continuous state data throughout the search process. The method entails discretizing the search space and linking each grid cell to the vehicle's continuous 3D state. The related continuous state receives numerous steering actions when a node from the open list of A* is popped, and new child states are created using the kinematic model of the moving vehicle. The matching grid cell for each continuous kid state is calculated. The continuous status of the node is updated and the node is reinserted into the open list for consideration if a node with the same grid cell already exists and the new node has a lower cost [11]. In order to follow the vehicle's minimal turning radius and avoid obstructions, this method fixes the length of the motion primitives that must be created each time.

By connecting each grid cell with the continuous 3D state of the vehicle, the Hybrid A* algorithm, an upgraded version of the conventional A* method, increases the path planning accuracy. The Hybrid A* method, in contrast to the conventional A* algorithm, uses a continuous motion model to expand nodes and provide workable pathways. It is distinguished by creating extremely viable pathways but does not ensure the discovery of the overall ideal solution. The algorithm utilizes the A* search in the discrete state space and leverages the continuous state space to address discretization errors. Specifically, the Hybrid A* algorithm utilizes a heuristic function that is the maximum of two heuristic functions, one based on nonholonomic constraints and the other on dynamic programming. Furthermore, the algorithm utilizes analytical expansions to enhance search speed and precision.

In autonomous driving path planning, the Hybrid A* algorithm can be applied to plan vehicle trajectories that avoid obstacles and satisfy other constraints. By combining the advantages of discrete and continuous state spaces, the Hybrid A* algorithm can effectively handle the requirements of continuity and precision in path planning. It generates smooth and efficient trajectories by considering the robot's dynamic constraints and dynamic programming information, while avoiding path deviations caused by discretization errors. The mapping relationship between the vehicle's continuous state and the discrete grid cells in the Hybrid A* algorithm ensures more accurate and precise path planning. By strengthening the A* algorithm's heuristics and utilizing continuous state space for path expansion, the Hybrid A* algorithm holds significant value in autonomous driving path planning.

4. Simulation experiments

4.1. Creating maze map

Prior to path planning, robots should first ascertain environmental information and construct an environmental map. A well-defined environmental map facilitates the establishment of planning methods and the selection of search algorithms, ultimately reducing the time wasted in searching for satisfactory paths [12]. Based on this study, binary maps were employed for experimentation. Binary maps represent map information in binary form (0s and 1s) and serve as a data structure. The binary map utilized in this research has dimensions of 263×263 and a resolution of 10 units per grid cell.

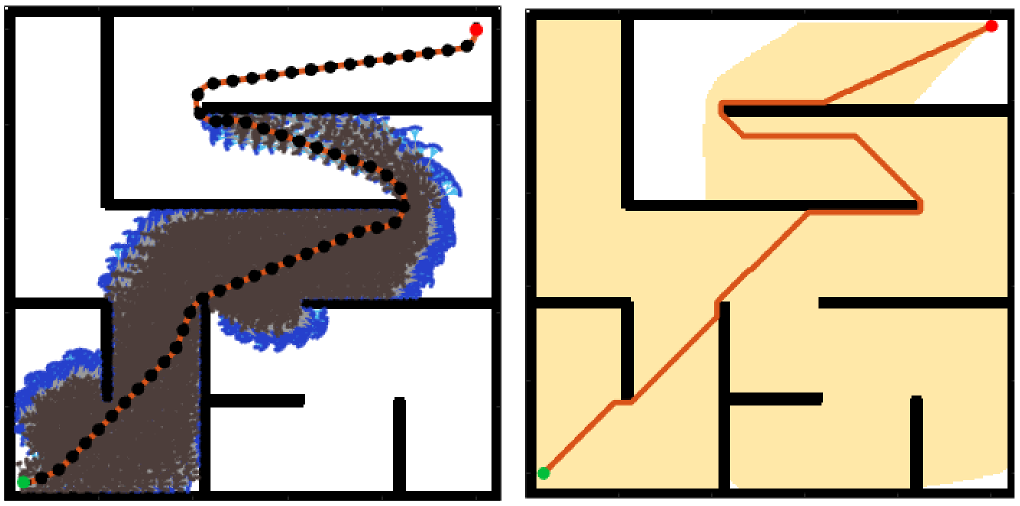

Figure 4. Simulation results of different method.

4.2. Experimental procedure

In the initial simulation, the performance of the A* algorithm and the Hybrid A* algorithm was primarily compared under two conditions: without inflating obstacles and with a 0.1 inflation radius for obstacles. Figure 4 illustrates the simulation results, using a binary map of size 263×263. The obstacles were randomly generated by altering the branching of the maze, while still maintaining some level of regularity. For the first experiment, a branching parameter of 6 was chosen. To approach real-world data, the runtime was obtained by recording three instances and averaging the values, thereby eliminating the influence of computer performance. The map resolution was set to 10 for easy recording and calculation of the robot's path and turning angles within the maze.

In the second experiment, under various situations of obstacle inflation, the effectiveness of the A* algorithm and the Hybrid A* algorithm were compared. Multiple tests were conducted by repeating the following code, observing the performance of both algorithms with varying degrees of map inflation. The same map was used in this experiment, with different inflation radii applied to obtain more accurate results. Tables 2 and 3 present the results of the inflation ranging from 0.1 to 0.4.

The maximum turning angle for the A* algorithm was set to 90 degrees, while for the Hybrid A* algorithm, it was set to 45 degrees. The path length and runtime were recorded for each scenario, as shown in Tables 2 and 3.

In the third experiment, four sets of experiments were conducted to compare the performance of the A* algorithm and the Hybrid A* algorithm under different branching factors: 8, 10, 12, and 14. Based on the results of Experiment 2, an inflation radius of 0.1 was selected as a consistent variable. The runtime, maximum turning angle, path length, and other variables were compared between A* and Hybrid A* algorithms as the basis for evaluating the two algorithms. The average results from each trial were utilized as the experimental data in a similar manner to account for the variability in computer performance.

5. Experimental Results

5.1. First group of experiments

Table 1. Randomized 263×263 map with branching of 6 and map extension of 0.1.

Indicators	A*	A* in map expansion	Hybrid A*	Hybrid a* in map extension
Running time/s	0.246683	0.273854	12.033537	12.933158
Maximum steering angle/°	90	90	45	45
Path length/m	55.5269	56.3955	54.5201	55.3844

From Table 1, it can be observed that, overall, the map with expanded obstacles requires more runtime and results in longer path lengths. However, during the simulation, it was observed that without incorporating the robot's perception parameters, the robot would not be able to recognize these obstacles and would pass through them, resulting in a phenomenon known as "passing through walls". This issue becomes particularly pronounced during turns.

5.2. Second group of experiments

Table 2. A* in a randomized 263×263 map with branching of 10.

Map Extension	0.1	0.2	0.3	0.4
Maximum steering angle/°	90	90	90	90
Path length/m	42.4434	43.3605	44.2777	45.1948
Running time/s	0.122202	0.123718	0.118807	0.118825

Table 3. Hybrid A* in a randomized 263×263 map with branching of 10.

Map Extension	0.1	0.2	0.3	0.4
Maximum steering angle/°	45	45	45	45
Path length/m	43.7483	43.9688	44.7554	46.1682
Running time/s	11.797235	10.231245	10.588308	10.076580

The following conclusions may be reached from the study of the data in Table 2 and 3:

Path Length: As the extent of obstacle expansion increases, both algorithms show an increasing trend in path length. For the A* algorithm, the path length increases from the initial value of 42.4434 to 45.1948 when the obstacles are expanded to 0.4. Similarly, for the Hybrid A* algorithm under the same conditions, the path length increases from 43.7483 to 46.1682. It can be observed that, at the same level of expansion, the Hybrid A* algorithm tends to have longer path lengths.

Runtime: In terms of runtime, both the A* algorithm and the Hybrid A* algorithm exhibit relatively stable performance across different levels of expansion. The runtime of both algorithms fluctuates within a similar range, without any noticeable trend. For example, when the obstacles are expanded to 0.1, the average runtime for the A* algorithm is 0.122202 seconds, while for the Hybrid A* algorithm it is 11.797235 seconds.

The following tentative conclusions can be made in light of this analysis: as the extent of obstacle expansion increases, both algorithms exhibit an increase in path length, but the Hybrid A* algorithm tends to have longer path lengths. In terms of runtime, both algorithms demonstrate relatively stable performance without significant differences.

It is important to note that the results of this experiment may be influenced by the experimental setup and the choice of maps. To provide a more comprehensive and accurate assessment of algorithm performance, further experiments can be conducted with multiple trials and consideration of additional factors such as different map structures and parameter settings. The statistical significance and dependability of the findings would enable a more thorough assessment of the benefits and drawbacks of the A* algorithm and the Hybrid A* algorithm for path planning.

5.3. *Third group of experiments*

Table 4. A* in randomized 263×263 map with different branching.

Branching	8	12	14	10
Maximum steering angle/°	90	90	90	90
Path length/m	41.7948	89.1897	80.7144	38.3345
Running time/s	0.195021	0.141057	0.156715	0.118653

Table 5. Hybrid A* in randomized 263×263 map with different branching.

Branching	8	12	14	10
Maximum steering angle/°	45	45	45	45
Path length/m	41.3166	31.972064	17.740318	3.986916
Running time/s	5.516480	95.0422	82.5313	37.1491

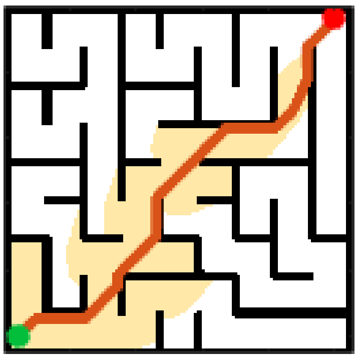
Figure 5. A* map of branching 10.

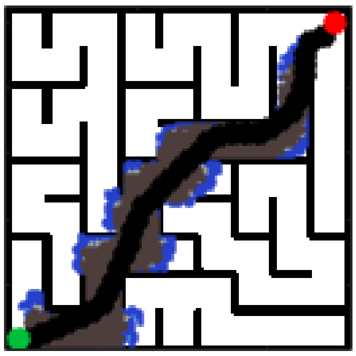
Figure 6. Hybrid A* map of branching 10.

In the first set of experiments with a maximum steering angle of 90 degrees, shown on Table 4, the overall path length was longer, and it showed an increasing trend as the branching factor increased. For example, when the branching factor was 12, the path length reached 89.1897, while for a branching factor of 10, the path length was 38.3345. Fig. 5 and Fig. 6 show the performance of the two algorithms in branching10. Furthermore, the runtime was relatively short and did not show any significant trend.

In the second set of experiments with a maximum steering angle of 45 degrees, shown on Table 5, the overall path length was shorter, and it gradually decreased as the branching factor increased. For instance, when the branching factor was 14, the path length was 17.740318, while for a branching factor of 16, the path length was only 3.986916. However, in contrast to the path length, the runtime exhibited noticeable fluctuations, especially when the branching factor was 12, with a runtime of 95.0422 seconds.

Based on the above analysis, the following observations and conclusions can be made:

- The choice of maximum steering angle has a significant impact on path length and runtime. A larger steering angle (90 degrees) leads to longer path lengths, while a smaller steering angle (45 degrees) results in shorter path lengths.
- The variation in the branching factor has a noticeable effect on path length. In the first set of experiments, increasing the branching factor led to longer path lengths, while in the second set of experiments, increasing the branching factor gradually decreased the path lengths.
- The runtime in the second set of experiments exhibited significant fluctuations, especially when the branching factor was 12, indicating increased computational complexity at larger branching factors.

6. Discussion

The size of the search space and the efficiency of the heuristic function have the most effects on the A* algorithm's time complexity. In small-scale maps, the A* algorithm can find the optimal path within a reasonable time. However, in large-scale maps or complex environments, the A* algorithm may need to search more nodes, resulting in increased runtime.

The hybrid A* algorithm improves upon the A* algorithm by combining continuous state space and discrete state space search, reducing the number of nodes to be searched. This makes the hybrid A* algorithm relatively more efficient in large-scale maps or complex environments, enabling it to find feasible paths more quickly.

The space complexity of the A* algorithm and the hybrid A* algorithm is relatively similar. Both algorithms require storing nodes and path information during the search process. In environments with sufficient memory resources, the space consumption of both algorithms is acceptable. However, the space utilization of the method must be considered in situations with limited resources, such as embedded systems or low-power devices.

Both the A* algorithm and the hybrid A* algorithm can be optimized by adjusting the heuristic function to improve efficiency. The number of nodes that must be searched can be decreased and the search direction can be efficiently guided by selecting the right heuristic function. In practical applications, algorithm optimization and adaptive improvements can be made based on different environments and task requirements to improve efficiency and path planning performance.

The route length of the hybrid A* method is often less than that of the A* algorithm in situations when the branching factor is low. This may be because the hybrid A* algorithm, with its advantages of combining continuous and discrete state space, can more efficiently plan shorter paths. However, in highly complex mazes, the path length of the hybrid A* algorithm may be much longer than that of the A* algorithm. This may be because the hybrid A* algorithm focuses more on generating smoother paths rather than just finding the shortest path.

In highly complex mazes, although the path length of the hybrid A* algorithm may be longer, it tends to have smoother turning points. This means that the hybrid A* algorithm can generate paths that are more in line with real robot motion, avoiding excessive large directional changes. This is because the hybrid A* algorithm considers the robot's kinematic constraints and dynamic programming information, placing more emphasis on path smoothness in path planning, resulting in more elegant and realistic paths.

7. Conclusion

By comparing and analysing the performance of these two algorithms in different scenarios, this paper proposes better optimization methods for the A* algorithm. The hybrid A* algorithm generally generates shorter paths when the branching factor is small, which may be attributed to its utilization of both continuous and discrete state spaces.

In highly complex mazes, the hybrid A* algorithm may have longer path lengths, but it exhibits smoother turning points, which are more in line with real robot motion. This indicates that the hybrid A* algorithm emphasizes path smoothness and realism. The hybrid A* algorithm is more efficient than the A* algorithm in large-scale maps or complex environments, enabling it to find feasible paths faster and generate paths that better match real robot motion. However, specific application scenarios and problem requirements still need to be considered for selection and optimization.

References

[1] S. a. J. J. a. S. H. a. Y. Y. Chen, 2023, Improved A-star Method for Collision Avoidance and Path Smoothing, in 2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT), pp. 32-35.
[2] H. a. Z. L. a. L. H. a. W. C. a. Q. Z. a. Q. Y. Zou, 2010, Optimized Application and Practice of A* Algorithm in Game Map Path-Finding, in 2010 10th IEEE International Conference on Computer and Information Technology, pp. 2138-2142.
[3] Z. a. W. S. a. Z. J. Zhang, 2021, A-star algorithm for expanding the number of search directions in path planning, in 2021 2nd International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), pp. 208-211.
[4] F. D. a. A. B. a. M. K. a. P. B. a. M. F. a. T. F. a. L. Jurišica, 2014, Path Planning with Modified a Star Algorithm for a Mobile Robot, Procedia Engineering, vol. 96, pp. 59-69.
[5] J. Y. J. L. H. T. X. &. G. M. Liu, 2017, An improved ant colony algorithm for robot path planning, Soft computing, vol. 21, pp. 5829-5839.
[6] D. a. Z. Y. a. L. Q. a. W. T. Huang, 2022, Research on Path Planning of Mobile Robot Based on Improved A-Star Algorithm, in 2022 International Conference on Informatics, Networking and Computing (ICINC), pp. 251-255.
[7] J. W. Z. a. H. X. Huiqun, 2018, Path planning based on improved particle swarm optimization algorithm, Journal of agricultural machinery, vol. 49, no. 12, pp. 371-377.
[8] Chang C, 2020, Research and Application on Path Planning Based on Improved A-Star Algorithm, Nanjing University.
[9] L. S. J. J. W. Y. L. W. L. T. Wang H, 2022 , The EBS-A* algorithm: An improved A* algorithm for path planning, PLoS ONE, vol. 17, no. 2.
[10] D. A. Dolgov, 2008, Practical Search Techniques in Path Planning for Autonomous Driving.
[11] D. a. T. S. a. M. M. a. D. J. Dolgov, 2010, Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments, I. J. Robotic Res., vol. 29, pp. 485-501.
[12] Y. a. W. Z. a. Z. S. Li, 2022, Path Planning of Robots Based on an Improved A-star Algorithm, in 2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), vol. 5, pp. 826-831.