



Improving A-Based Path Planning through Interpolation and Curve Smoothing with Collision-Aware Correction

Weiting Song

College of Electrical and Information Engineering, Hunan University, Changsha, Hunan, 410082, China

swte@hnu.edu.cn

Abstract. With the development of automatic navigation technology, path planning algorithms play a vital role in mobile robots and intelligent transportation systems. Although the traditional A* algorithm is stable and efficient, the paths it generates often contain numerous turning points and lack smoothness, making it difficult to satisfy the continuity and feasibility requirements of real-world trajectory tracking. To address these limitations, this study proposes an integrated optimization framework that combines interpolation, B-spline smoothing, and legality verification based on the traditional A* path. The approach begins with initial path generation using the A* algorithm on a discrete grid map, followed by floating-point interpolation and curve fitting to improve continuity, and a grid-level collision detection mechanism to guarantee path feasibility. Experimental evaluations compare the original and optimized paths in the light of length, number of turning points, and computation time. Results prove that the proposed method markedly shortens the path length, reduces turning points by more than 50%, and produces high-quality, smooth trajectories suitable for practical navigation applications.

Keywords: Path Planning, A-star, B-Spline Smoothing, Autonomous Navigation.

1 Introduction

In autonomous platforms like mobile robots, autonomous surface vessels (ASVs), and intelligent vehicles, path planning is really crucial. It serves as the connection between environmental perception and motion execution. Its performance has a direct impact on navigation efficiency and operational safety. Another thing worth mentioning is that generating a collision-free and smooth path in cluttered environments is a really tough challenge. Among the many planning techniques, the A* algorithm has been prominent for a long time due to its optimality and computational stability in grid-based search spaces. However, even with its advantages, the discrete nature of A* outputs often results in paths that have abrupt turns and redundant waypoints. These features are not good for continuous control.

To deal with these shortcomings, people have done a lot of research to improve A* in aspects like accuracy, continuity, and physical feasibility. For example, some studies have brought in guideline-driven heuristics and variable-step strategies. This has led to paths that are shorter and also fit better with vehicle kinematics [1]. Another thing worth mentioning is that others have looked into geometric refinements, like arc-based local smoothing. They did this to reduce too much turning and node redundancy [2]. More recently, interpolation and secondary curve-fitting techniques have been added. This way, higher continuity can be achieved while still keeping the feasibility [3].

Based on these insights, this study puts forward an integrated optimization framework. It combines floating-point interpolation, B-spline smoothing, and also does post-processing legality checks. First of all, the whole process starts with an A*-derived baseline. Then, it gets refined through interpolation and curve fitting. Finally, it does point-wise validation to get rid of any interference from residual obstacles. When it comes to the experimental evaluations, they show that this method can achieve really good trajectory quality. And what's more, it only needs minimal computational overhead. This paves the way for it to be used in real-world navigation systems.

2 Methods

This study brings in floating-point interpolation and B-spline smoothing. It does this based on the traditional A* planning. And it aims to build an optimization method for creating paths with high continuity in complex environments. Another thing, the approach has four stages. First is the initial path planning. Then comes the sparse optimization. After that, there's the interpolation. And finally, there's the legality verification.

2.1 Initial path planning (A* Algorithm)

The initial path gets generated by using the classic A* algorithm on a discrete grid map. It's a strategy that's often used in global navigation planning techniques. That's because it's efficient and can guarantee optimality under an acceptable heuristic [4]. The A* algorithm works like a heuristic search method. It assesses nodes by means of a defined cost function.

$$f(n) = g(n) + h(n) \quad (1)$$

Here, $g(n)$ stands for the cumulative cost starting from the beginning point and going all the way to the current node n . And $h(n)$ means the heuristic estimate of the remaining cost from the node to the target point. Usually, it's calculated by using the Euclidean distance or Manhattan distance [5]. Another thing is that this cost structure enables the algorithm to find a balance between exploration and exploitation. So, it can improve the computational efficiency a lot when compared to uninformed search methods like Dijkstra [6].

The environment gets modeled as a 2-D grid. In this grid, each cell is either free or occupied by an obstacle. Nodes are connected according to adjacency relationships. An eight-neighborhood expansion strategy is utilized, which allows each node to have as many as eight neighbors. There are four neighbors in the horizontal and vertical directions and another four along the diagonals [7]. Such a configuration makes diagonal movements possible, and it results in shorter and more natural paths in comparison with a strict four-neighborhood (orthogonal-only) approach where movements are restricted to going up, down, left, and right. During the search procedure, the point having the minimum value is repeatedly chosen from the open set. Then the neighbors of this node are expanded and their cost values are updated until the final is arrived at or the search space is completely used up. This process guarantees that the generated path is globally optimal within the given grid and cost.

2.2 Path optimization (sparse processing)

Trajectories produced by the A* algorithm often contain many unnecessary nodes. To reduce turning points and simplify the path structure, the research introduces a path sparse strategy. The procedure begins at the initial node and iteratively selects the most distant point that can be connected by an unobstructed straight line, thereby forming a reduced sequence of key waypoints. This step effectively reduces unnecessary turns in the path, which helps to improve the quality of subsequent curve interpolation.

2.3 Interpolation processing (continuous space mapping)

Traditional grid-based path planning produces a sequence of integer grid points $((x, y) \in N^2)$, which often lacks sufficient resolution for practical trajectory tracking and motion control. To overcome this limitation and achieve higher-resolution trajectory representation with improved continuity, piecewise linear interpolation is implemented for the refined path.

For any two consecutive nodes (x_i, y_i) and (x_{i+1}, y_{i+1}) on the sparse path, a set of intermediate points is inserted based on a predefined interpolation step size Δs (e.g., 0.1--0.2 grid units). The coordinates of each interpolated point are determined using the parametric linear interpolation equations:

$$x(t) = (1 - t)x_i + tx_{i+1}, y(t) = (1 - t)y_i + ty_{i+1}, \quad t \in [0,1] \quad (2)$$

where t is the normalized interpolation parameter. For each segment, the number of interpolated points is determined as:

$$n = \max\left(1, \left\lfloor \frac{d_{i,i+1}}{\Delta s} \right\rfloor\right) \quad (3)$$

where $d_{i,i+1} = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$ is the Euclidean distance between two adjacent nodes.

This interpolation procedure transforms the discrete A* output into a quasi-continuous representation by generating floating-point coordinates rather than fixed integer grid points. The refined trajectory, now exhibiting higher spatial resolution, serves as the foundation for subsequent smoothing operations. At this stage, B-spline fitting is employed to further enhance continuity and reduce excessive curvature variations.

2.4 Smoothing Processing (B-spline)

Interpolation can lessen those abrupt transitions between nodes. Often, there are still residual angular discontinuities hanging around at the junctions of consecutive segments. Here's what we do to deal with this. We bring in a B-spline (Basis Spline) curve-fitting approach. This approach actually gives us both geometric flexibility and curvature continuity, which is really useful.

B-splines are widely used in path planning and computer-aided geometric design. They have two main advantages: one is the local control of shape adjustments, and the other is global smoothness. They manage to do this by blending piecewise polynomial segments through basis functions that are defined over a knot vector. The parametric form of a B-spline curve of a certain degree k can be written like this:

$$C(u) = \sum_{i=0}^n N_{i,k}(u)P_i, u \in [0,1] \tag{4}$$

where P_i are the control points and $N_{i,k}(u)$ denote the corresponding basis functions, ensuring smoothness across segment boundaries.

$$N_{\{i,0\}}(u) = \begin{cases} 1, & u_i \leq u < u_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

$$N_{(i,k)}(u) = \left((u - u_i) / (u_{(i+k)} - u_i) \right) N_{(i,k-1)}(u) + \left((u_{(i+k+1)} - u) / (u_{(i+k+1)} - u_{(i+1)}) \right) N_{((i+1),k-1)}(u) \tag{5}$$

where $\{u_i\}$ denotes the knot vector.

Fitting the interpolated trajectory with a B-spline curve effectively suppresses sharp angular transitions while retaining the global structure of the original path. This property is particularly advantageous in robotic navigation, where curvature continuity underpins dynamic feasibility and facilitates high-precision motion control.

2.5 Detection of legality and its repair

When generating B-splines, some of the curve segments might run into obstacles or go beyond the boundaries of the map. In order to make sure that the path is feasible, a mechanism for detecting legality point by point is put into practice. This mechanism carries out collision checks at the grid level for each floating-point node along the

smoothed path. Should a node happen to be inside an obstacle or outside the permitted area of the map, it will be either removed or replaced by means of linear interpolation. The purpose of this is to ensure that the final trajectory stays completely within the free space.

3 Results

To evaluate the performance of the proposed A*-based algorithm incorporating interpolation and path smoothing, many simulations were implemented in a Python environment. The evaluation focused on several indicators, including path generation quality, smoothness of curvature, total distance, frequency of turning points, and computational time. All tests were performed on a 15×15 grid map populated with randomly distributed obstacles. The start was fixed at the corner $(0, 0)$, and the final at the corner $(14, 14)$, both guaranteed to lie within reachable regions.

3.1 Results of visualizing path generation

The experiment made a comparison among four types of paths, namely the original A* path, the optimized sparse path, the interpolated path, and the smoothed path. In Fig. 1, the black cells stand for the regions with obstacles, and the white cells signify the free space within the grid map. The green marker is used to denote the starting point, and the red marker serves to indicate the goal point.

The orange dashed line is in correspondence with the original A* path. This original path usually looks jagged as it has numerous redundant nodes and sharp turns. The cyan polyline stands for the interpolated path. This interpolated path is obtained by mapping discrete nodes into continuous space. The way to do this mapping is through the insertion of intermediate floating-point points. By doing so, it can enhance the resolution of the trajectory and also offer a foundation for curve fitting. The blue solid curve shows the final smoothed path that is generated by B-spline fitting. This fitting effectively gets rid of the abrupt angle changes while still keeping the overall shape of the original path intact. As a result, it creates a trajectory that looks natural visually and is smooth. Such a trajectory is more fitting for robotic controllers that demand continuous motion. When comparing the original path with the optimized one, it can be seen that there is a notable improvement in the quality of the path. This improvement is evident in both the geometry and the smoothness aspects.

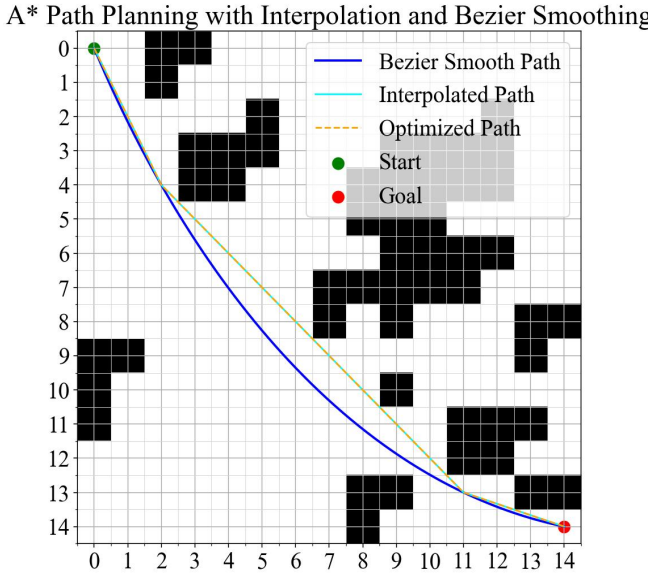


Fig. 1. Visualization of the original A* path (orange dashed), interpolated path (cyan), and smoothed B-spline path (blue) on a grid map with obstacles (black) and free space (white) (Photo/Picture credit: Original).

3.2 Comparison of quantitative evaluation indicators

The total path length, number of turning points, and planning time were recorded for three types of paths: the original A* path, the optimized path, and the smoothed path. A representative set of results is summarized in Table 1.

Table 1. Numerical comparison of different path types

Path Type	Path Length (grid units)	Number of Turns	Planning Time (s)
Original A* Path	25.66	12	0.0043
Optimized Path	21.19	5	0.0045
Smoothed Path	20.92	—	0.0046

Results indicate that applying path optimization and continuity enhancement decreases the overall path length by approximately 18%, while the number of turning points drops by over 50%, with only a marginal increase in planning time.

3.3 Path legality analysis

To ensure the feasibility of the smoothed path, a point-wise legality detection module was applied following B-spline processing. All invalid points, such as those located within obstacle regions, were identified and removed. As a result, the final smoothed path remains entirely within the traversable area, without intersecting obstacles, thereby guaranteeing path safety in complex environments.

In conclusion, the proposed approach significantly improves trajectory continuity and curvature smoothness while ensuring better path quality and efficient planning, accompanied by only a minimal rise in computational overhead. These findings validate the effectiveness and real-world applicability of the optimization framework for practical path planning scenarios.

4 Discussion

4.1 Comparison with other path smoothing methods

The proposed framework combines floating-point interpolation with B-spline smoothing within the A* paradigm. It manages to strike a balance between continuity and feasibility. Some methods do multi-segment splicing. They repeatedly backtrack and replan. This approach is different. It can deliver a continuous trajectory in just one pass. This way, it avoids those redundant global searches. Also, it cuts down on the computational overhead [8]. Another thing worth mentioning is about the sliding window adjustments. They rely on dynamic cost surfaces. But the proposed strategy is not like that. It anchors the smoothing at the global stage. This design choice allows for offline orchestration before the actual execution. It lessens the reliance on real-time sensory feedback. What's more, it enhances the system's deploy ability [8].

4.2 Legality assurance mechanism for smoothed paths

To deal with the possible issues of 'obstacle penetration' that might occur during the smoothing process, a mechanism for detecting legality point by point is introduced right after the generation of B-spline. Through this, any sampling points that are invalid and located within obstacles are removed, thus ensuring that the final path can be fully traversed. Meanwhile, other approaches try to ensure safety by making use of global cost maps and adjusting paths. However, they are still prone to collision risks in environments that are densely cluttered, as shown in [9]. The combination that is proposed, which involves both smoothing and legality correction, offers a strong safeguard for static scenarios.

4.3 The adaptability of continuous paths within complex environments

From a morphological aspect, the outputs of the traditional A* are restricted by the discrete grid structure. This brings about numerous turns that are unnecessary and also segments of backtracking, which don't fulfill the requirements of continuous control. Through the integration of interpolation along with curve fitting, the method put forward improves the smoothness of paths to a notable extent. This is especially the case in those regions where obstacles are distributed rather sparsely. Nevertheless, as previous studies have pointed out, in environments that are either dynamic or have a dense population of obstacles, when it comes to smoothing, one must take into account the real-time interaction with the avoidance of local obstacles [10]. The work

in the future will look into combining this method with algorithms like DWA or APF so as to boost the adaptability and robustness when there is uncertainty.

4.4 The applicability and scalability of this method

This method is mainly applicable to static global path planning tasks. It is suitable for autonomous vehicles, unmanned aerial vehicles (UAVs), and indoor mobile robots as well. Moreover, the interpolation and legality detection parts show strong modularity. This enables them to be transferred to other frameworks like D* or RRT. Based on the current research findings, it is clear that smoothed paths set higher demands on control precision. Future enhancements will include dynamic constraints and velocity profiling to achieve a closer integration between planning and control.

5 Conclusion

A path planning strategy that was improved by integrating interpolation along with B spline curve smoothing was put forward. The aim was to deal with the problems like discontinuity and having too many turns that are usually seen in traditional A* paths. When floating-point interpolation and continuous curve fitting were applied to the output of A* and a path legality verification module was incorporated, the optimized path managed to have a shorter overall length. At the same time, it also managed to considerably cut down the number of turning points. The experimental evidence really shows the framework's ability to create trajectories that match up with practical control needs. It turns out to be especially useful for static global planning situations. Another thing worth mentioning, its applications cover areas like autonomous ground vehicles, aerial platforms, and indoor robotic navigation. In these fields, path continuity is a crucial factor for stability and safety. Looking to the future, an exciting direction is to combine this methodology with real-time obstacle avoidance and adaptive replanning methods. This way, we can get a unified solution for dynamic and unpredictable environments..

References

1. S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei, An improved A-Star based path planning algorithm for autonomous land vehicles, *Int. J. Adv. Robot. Syst.*, **17**, 1729881420962263 (2020)
2. C. Ju, Q. Luo, and X. Yan, Path planning using an improved a-star algorithm, in 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan), IEEE, 23–26 (2020)
3. Y. Song and P. Ma, Research on mobile robot path planning based on improved A-star algorithm, in 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS), IEEE, 683–687 (2021)

4. Z. Tang and H. Ma, An overview of path planning algorithms, *IOP Conf. Ser.: Earth Environ. Sci.*, **804**, 022024 (2021)
5. N. Buniyamin, W. W. Ngah, N. Sariff, and Z. Mohamad, A simple local path planning algorithm for autonomous mobile robots, *Int. J. Syst. Appl. Eng. Dev.*, **5**(2), 151–159 (2011)
6. X. Lan, X. Lv, W. Liu, Y. He, and X. Zhang, Research on robot global path planning based on improved A-star ant colony algorithm, in *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, IEEE, **5**, 613–617 (2021)
7. H. Qin, S. Shao, T. Wang, X. Yu, Y. Jiang, and Z. Cao, Review of autonomous path planning algorithms for mobile robots, *Drones*, **7**(3), 211 (2023)
8. S. Sedighi, D. V. Nguyen, and K. D. Kuhnert, Guided hybrid A-star path planning algorithm for valet parking applications, in *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, IEEE, 570–575 (2019)
9. K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, A survey of path planning algorithms for mobile robots, *Vehicles*, **3**(3), 448–468 (2021)
10. H. Sang, Y. You, X. Sun, Y. Zhou, and F. Liu, The hybrid path planning algorithm based on improved A* and artificial potential field for unmanned surface vehicle formations, *Ocean Eng.*, **223**, 108709 (2021)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

