






# Correspondence

## An Efficient and Accurate A-Star Algorithm for Autonomous Vehicle Path Planning

Zhi Lin , Kang Wu , Rulin Shen , Xin Yu ,  
and Shiquan Huang 

**Abstract**—Efficient and accurate path planning algorithm of unmanned vehicles for unstructured road is of great significant for field disaster relief. An improved A-star algorithm for unmanned vehicle path planning is proposed in this paper to achieve optimized path with shorter length and less inflection point. Redundant safety spaces are set up in the improved algorithm to filter out impassable narrow roads and to avoid collisions between vehicles without obstacles. The prejudgment planning strategy and redundant inflection point elimination strategy are integrated into the algorithm to obtain a better driving path. Furthermore, a safe corridor approach was used in the trajectory optimization to avoid repeated checks and corrections and save computer time. The simulation results show that the path planned with the improved algorithm has shorter length, less inflection point and smoother than the path planned with the initial algorithm, and can avoid collision between vehicle contours and obstacles.

**Index Terms**—Autonomous driving, A-star algorithm, path planning, redundant inflection point elimination, pre-judgment planning strategy.

### I. INTRODUCTION

#### A. Background

Autonomous driving technology has great prospects in various traffic environments. Path planning algorithms based on artificial intelligence is one of the main research focuses of this technology. Various path planning techniques have been developed, such as grid-based search algorithms [1], artificial potential field methods [2], random sampling-based planning algorithms [3], neural networks [4], [5], and deep learning methods [6], [7], [8]. Recently, a grid search-based path planning algorithm has gained popularity due to its simplicity and analytical optimality. Among them, Dijkstra algorithm [9], the A-star (A\*) algorithm [10], [11], [12], the D-star (D\*) algorithm [13], are the most popular algorithms.

A\* algorithm adds heuristic function on the basis of Dijkstra, combining graph search and heuristic search methods, which improves efficiency and greatly reduces the search workload. However, A\* algorithm still has potential for improvement in efficiency, optimality, and smoothness. Ebendt et al. proposed a weighted A\* (WA\*) algorithm [14], which increases the weight coefficient in the original function to improve the influence of the heuristic function on search efficiency,

but the results tend to fall into the local optimal value and cannot be optimized. Gochev et al. introduced an improved version of the weighted A\* algorithm, the anytime-tree-restoring-weighted A\* [15] (ATRA\*), to prevent results from falling into local optimization by sacrificing partial optimality. The algorithm also makes use of the past search data to speed up the search process and improve the search efficiency. However, there is still potential for optimization and smoothing. To meet the requirements for planning time limits in the real world, Likhachev et al. proposed an anytime repairing A\* (ARA\*) algorithm [16], which can adjust the performance limits of the planning according to the acceptable search time. However, the number of inflection points may still be reduced to improve the path continuity.

Swaminathan et al. introduced the multi-heuristic A\* (MHA\*) algorithm [17] to address the issue of unstable algorithm performance caused by the challenge of determining the heuristic function in the traditional A\* algorithm. The algorithm changes the influence of heuristic functions by combining different heuristic functions, and simplifies the design of heuristic functions to avoid falling into local minimums. Fahad et al. propose dynamic multi-heuristic A\* (DMHA\*) algorithm [18] on the basis of the multi-heuristic A\* algorithm, which retains the advantages of multi-heuristic function and can avoid manually setting heuristic function to improve planning performance. Islam et al. presented an A\*-connect algorithm [19] that utilizes multiple heuristics, resulting in faster pathfinding and better stability compared to the RRT-connect algorithm [20]. The A\*-connect algorithm conducts a bidirectional search from both the starting and target points. Ma et al. introduced a lifelong planning A\* (LPA\*) algorithm [21] based on the features of heuristic and incremental searches. The LPA\* algorithm exhibits improved search efficiency and generates better paths than that of the A\* algorithm, but path length and number of inflection points still have potential for optimization.

#### B. Motivations and Contributions

Most autonomous driving technologies target specific scenarios, such as enclosed areas, urban structured roads and highways, where the traffic environment and vehicle location information are relatively simple and vehicle trajectories can be achieved by restricting lane lines or road edges. However, in the post-disaster field environment, the situation is more complicated. Although the speed of the vehicle is not very fast, efficient and accurate optimal path planning algorithm are the guarantee of rescue success. In this paper, innovation strategies are integrated in an improved A\* algorithm to achieve high planning efficiency, path optimality, optimal passability and smoothness. Different from other published works, the main contribution of this research lies in the following three aspects.

- 1) A redundant safety space method is proposed for improving passability in the post-disaster field road environment.
- 2) A comprehensive strategy is proposed to achieve an effective path with shorter length, less inflection points, and smaller turning angles, which does not reduce the map resolution or change the original search method. It can avoid planning impassable narrow roads which vehicle body contours colliding with obstacles.
- 3) A safe corridor is set up during the path optimization process, which avoids repeated detection and correction through strong constraints and reduces optimization time.

Manuscript received 6 June 2022; revised 23 November 2022, 26 May 2023, and 25 September 2023; accepted 17 December 2023. Date of publication 29 December 2023; date of current version 20 June 2024. This work was supported by the National Key Research and Development Program of China under Grant 2019YFC1511503. The review of this article was coordinated by Dr. Ajeya Gupta. (Corresponding author: Rulin Shen.)

Zhi Lin, Rulin Shen, Xin Yu, and Shiquan Huang are with the College of Mechanical and Electrical Engineering, Central South University, Changsha 410083, China (e-mail: 472845137@qq.com; shenrl@csu.edu.cn; yuxin0816@163.com; huangshiquan@csu.edu.cn).

Kang Wu is with Sunward Intelligent Equipment Company Ltd., Changsha, Hunan 410100, China (e-mail: wukangjk@foxmail.com).

Digital Object Identifier 10.1109/TVT.2023.3348140

### C. Organization

The remainder of this article is organized as follows. Section II describes the improved path planning algorithm in theory. Section III presents the simulation experiment of the proposed algorithm and analysis the results. Section IV draw a conclusion of this paper.

## II. PATH PLANNING

A\* algorithm is a typical heuristic search method that applying evaluation function to direct and determine the search direction. The evaluation function  $F(n)$  is applied to scan all the nodes of the OpenList (Traversable points) from the starting node. The node with the lowest  $F(n)$  value is considered the current node, and the algorithm identifies the surrounding eight nodes as extension nodes for exploration. The process is repeated in a loop until the target node is added to the OpenList. To obtain the minimum cost path, the algorithm conducts a backward search from the parent node to the starting node, initiating the search process from the target node. The evaluation function  $F(n)$  is defined as:

$$F(n) = G(n) + H(n) \quad (1)$$

$$G(n) = \sqrt{(x_n - x_s)^2 + (y_n - y_s)^2} \quad (2)$$

$$H(n) = \sqrt{(x_t - x_n)^2 + (y_t - y_n)^2} \quad (3)$$

Where  $G(n)$  denotes the cost function that reflects the real cost incurred from the initial node to the current node  $n$ ;  $H(n)$  is a heuristic function that provides an estimate of the cost from the current node  $n$  to the target node; The horizontal and vertical coordinates of the starting node and target node are denoted as  $x_s, y_s, x_t$ , and  $y_t$ , respectively. The current node  $n$  horizontal and vertical coordinates are represented as  $x_n$  and  $y_n$ .

Based on the heuristic search method and evaluation function, A\* algorithm is improved by combining three strategies to achieve high efficiency and accuracy.

### A. Redundant Safety Space Setting

For unstructured road, especially the post-disaster environment, obstacles were scattered randomly across the road. Therefore, it is necessary to filter out narrow and impassable narrow roads to avoid collision of vehicle and obstacles. Setting redundant safety space around obstacles is the way to keep a certain distance between the planned path and the obstacles, ensuring the safety of the vehicles following the path directly. The process of setting the redundant safety space on the grid map primarily involves the following steps:

- 1) Identify the farthest distance between the center of the vehicle and its contour, denoted as  $D_s$ .
- 2) Determine the minimum number of grid cells needed based on the actual length represented by a single grid side:

$$n_{ex} = \text{Ceiling} \left( \frac{D_s}{p} \right) \quad (4)$$

Where  $\text{Ceiling}()$  is the function for upward rounding;  $p$  is the length of a single grid.

- 3) Add  $n_{ex}$  non-passable grids in the four directions of up, down, left and right of the obstacle when setting redundant safe space on the grid map.

The comparison of with or without redundant safety space around the obstacles is shown in Fig. 1. For paths in open space, such as paths 1 and 2 in Fig. 1, redundant safe spaces can separate paths from obstacles

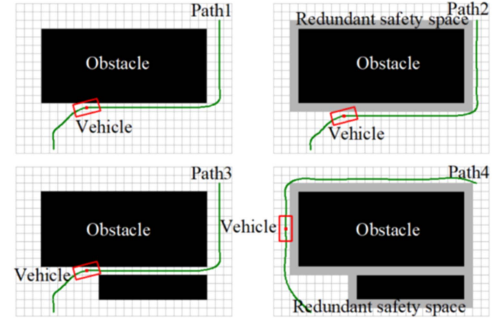


Fig. 1. Effect of redundant safety space. Path 1 and path 3: Without safety space, the vehicle collides with the obstacle or squeezed into an impassable road. Path 2 and path 4: With safety space, the vehicle find the right path to avoid collision.

by a safe distance (path 2). The presence of redundant safety space allows the vehicle to pass through obstacles while maintaining a certain distance from them, ensuring the safety of the vehicle during travel. Although the vehicle may come across the redundant safety space, collision with obstacles will never occur. For paths in narrow spaces, as shown in paths 3 and 4 in Fig. 1, impassable roads will be directly filtered out and suitable paths will be planned to pass the obstacles. Narrow paths that are impassable by vehicles will be fully overlapped with redundant safety space.

### B. Strategies for Shorter Length, Less Inflection Point and Turn Angle

To achieve a path with shorter length, less inflection point and turn angle, the improved A\* algorithm is strengthened in three areas, which are Turning cost setting, Prejudgment planning, and Redundant inflection point elimination.

1) **Turning Cost Setting:** In the post-disaster road environment, a large number of obstacles leads to lots of turns of vehicle. The A\* algorithm, in its traditional form, doesn't account for the cost of turning for autonomous vehicles during path planning. As a result, the planning path may have numerous turning points that makes the turn much, leading to significant energy and time consumption. The evaluation function  $F(n)$  in A\* algorithm is redefined in this paper to calculate the effect of turning cost.

To take turning cost into account, it is necessary to determine collinear among the extended node, the current node, and the parent node of the current node before calculating the  $F$  and  $G$  values of the current node. If the three points do not lie on the same line, the current node is considered as an **inflection point**, indicating that the unmanned vehicle needs to change direction when approaching this node. In this case, the turning cost constant  $C$  should be added to the  $G$  value of the current node during path planning. On the other hand, if the three points lie on the same line, the current node is not an inflection point. The unmanned vehicle does not need to turn when driving to this node, and the turning cost constant  $C$  is not included. The improved cost function  $G'(n)$  and evaluation function  $F'(n)$  are defined as:

$$G'(n) = \begin{cases} G'(n-1) + P(n) \\ G'(n-1) + P(n) + C \end{cases} \quad (5)$$

$$F'(n) = G'(n) + H(n) \quad (6)$$

Where  $G'(n)$  is the cost function of the parent node of the current node;  $P(n)$  is the cost function from the current node to its parent node;

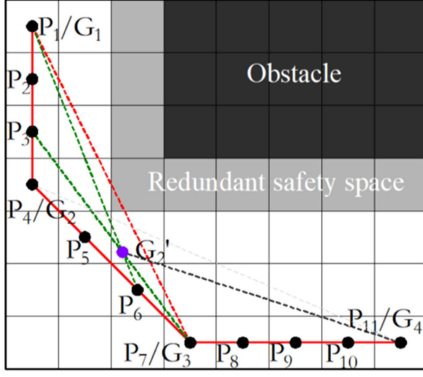


Fig. 2. Path optimization diagram. Red solid line: Original algorithm path. Gray dashed line: Path generated with the pre-judgment strategy.  $G_1G_2G_3G_4$ : Initial extraction inflection point. Green dotted line: Line for looking new inflection point  $G'_2$ .  $G_1 - G'_2 - G_4$ : Final optimized path.

$F'(n)$  is the formula for the changed A\* algorithm;  $C$  is the turning cost.

When calculating the evaluation function  $F'(n)$  of the current node, the cost function  $G'(n-1)$  is known, so only the cost function  $P(n)$  needs to be calculated:

$$P(n) = \sqrt{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2} \quad (7)$$

The horizontal and vertical coordinates of the parent node of the current node are denoted as  $x_{n-1}$  and  $y_{n-1}$ , respectively.

2) **Prejudgment Planning**: A **short path** with **less inflection** and **turning angles** facilitates quick post-disaster rescue. However, the conventional A\* algorithm finds the current node by traversing the OpenList with the smallest  $F$  value and continues until the target node is included in the OpenList, which result in long path lengths, **numerous inflection points**, and **large turning angles**. This paper introduces a prejudgment planning strategy that allows for early termination of the planning algorithm under certain conditions. After searching for the current node with the lowest  $F$  value in OpenList, the algorithm program will check if an obstacle existed between the current node and the target node. If no obstacle blocks the path between the current node and the target node, the algorithm connects them directly and ends the path planning. If obstacle is detected, the algorithm continues until the path is clear between the current node and the target node, and then ends the path planning.

The path planning process using the traditional A\* algorithm and the A\* algorithm with prejudgment planning strategy are compared in Fig. 2. The black rectangular grid represents an obstacle; The gray rectangular grid represents redundant safe space; The path nodes are denoted by  $P_1P_2 \dots P_{10}P_{11}$ , with  $P_1$  being the initial node and  $P_{11}$  representing the destination. In Fig. 2, the path planning using traditional A\* algorithm involved 10 search iterations from  $P_1$  to  $P_{11}$  before finding the target node. However, the A\* algorithm incorporating the prejudgment planning strategy only need four steps to find the path ( $P_1P_2P_3P_4P_{11}$ ) without obstacles and complete the path planning. The latter path has a shorter distance, fewer turning points, and smaller turning angles compared to the former. Meantime, the search time also be shortened.

3) **Redundant Inflection Point Elimination**: A redundant inflection point elimination strategy is applied to improve A\* algorithm to optimize the planning in path length, inflection points and turning angle.

As shown in Fig. 2, the initial path is marked as  $P_1P_k$ , which  $P_k(k = 1, 2, \dots, 11)$  are nodes along the path, where  $P_1$  is the starting node and  $P_{11}$  is the target node. Denote the straight line formed by the path node  $P_k$  and the path node  $P_{k+1}$  as  $L_k$ , and then judge whether  $L_k$  and  $L_{k+1}(k = 1, 2, \dots, 10)$  are collinear. If the two lines are collinear,  $P_{k+1}$  is not an inflection point; otherwise, each subsequent node  $P_{k+1}$  along the path will be considered as an inflection point until the target node. Denote the inflection point in the path as  $G_j(j = 1, 2, 3, 4)$ .

Connect the two separated inflection points  $G_j$  and  $G_{j+2}$  to identify the valid inflection point. If there is an obstacle on the line connecting the two inflection points, such as the inflection points  $G_1$  and  $G_3$  in Fig. 2, the point is invalid. If there is no obstacle on the line connecting the two inflection points, such as the inflection points  $G_2$  and  $G_4$  in Fig. 2, the point is valid. For the invalid point, a new inflection point  $G'_{j+1}$  need to be found to replace the original inflection point  $G_{j+1}$ .

A new inflection point needs to be found when the initial inflection point is invalid. The green line in Fig. 2 is an example of search a new inflection point. Because an obstacle locates on the connecting line between the separated inflection points  $G_1$  and  $G_3$ , a new inflection point  $G'_2$  need to be found to replace the original inflection point  $G_2$ . First, taking point  $G_1$  as the starting node, and connecting it to the previous point  $P_6$  before point  $P_7/G_3$  with a line. If there is an obstacle on the line which connecting point  $G_1$  and point  $P_6$ , then connecting point  $G_1$  and point  $P_5$ . Until none obstacle locates on the connecting line between point  $G_1$  and point  $P_k$ , find the expression of the straight line  $G_1P_k$ . In this example,  $k = 6$ , so the expression of straight line  $G_1P_6$  is obtained by the transformation of point  $G_1$  and point  $P_6$  with the two-point straight line equation, namely:

$$y = (P_{6y} - G_{1y})(x - G_{1x}) / (P_{6x} - G_{1x}) + G_{1y} \quad (8)$$

Where  $x$  is the horizontal coordinate of any point on the line;  $y$  is the vertical coordinate corresponding to the point.

Then, connecting point  $G_3$  and point  $P_2$  behind point  $P_1/G_1$ . If there is an obstacle on the line between point  $G_3$  and point  $P_2$ , connecting point  $G_3$  and point  $P_3$  until no obstacle on the line between point  $G_3$  and point  $P_k$ . Finally figuring out the expression for the straight line  $G_3P_k$ . In this example,  $k = 3$ , and the expression of the straight line  $G_3P_3$  is:

$$y = (P_{3y} - G_{3y})(x - G_{3x}) / (P_{3x} - G_{3x}) + G_{3y} \quad (9)$$

The intersection of the straight line  $G_1P_6$  and the straight line  $G_3P_3$  is the new inflection point  $G'_2$ . As can be seen from Fig. 2 that the length of path  $G_1 - G'_2 - G_3$  is significantly shorter than the original path  $G_1 - G_2 - G_3$ .

During the iterative process, eliminate path nodes and update inflection points. In fig. 2, the path nodes in the original path are removed, leaving only the inflection points. Besides, the original inflection point  $G_2$  is replaced with a new inflection point  $G'_2$ . Consequently, the revised inflection point path shown in Fig. 2 becomes  $G_1 - G'_2 - G_3 - G_4$ .

If there is no obstacle on the connecting line between point  $G_1$  and point  $G_3$ , continue to connect point  $G_1$  and point  $G_4$  until there is an obstacle on the connecting line between point  $G_1$  and point  $G_j$ , then connect point  $G_1$  and point  $G_{j-1}$ . At the same time remove intermediate redundant inflection points and update the path. Repeat the above operations in turn from inflection point  $G_j$  until none redundant inflection points locate in the path. The final path obtained by updating the path is shown as  $G_1 - G'_2 - G_4$  in Fig. 2.

Fig. 2 shows both the optimization path achieved by applying redundant inflection point elimination strategy and the original path. In the Fig. 2,  $G_1 - G_2 - G_3 - G_4$  is the path planned with the initial

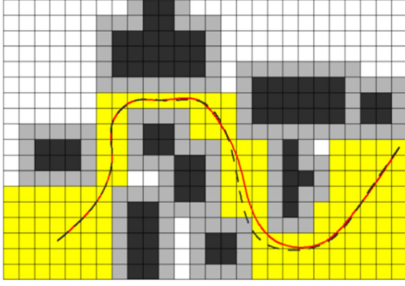


Fig. 3. Path with and without safe corridor in trajectory optimization. Black dashed line: Without safety corridor constraint. Solid red line: With safety corridor constraint.

algorithm,  $G_1 - G'_2 - G_4$  is the path planned with the improved algorithm. The optimization algorithm shortens the path length and reduces the number of turning points and turning angles, as shown in the figure.

### C. Safe Corridor Settings

A planning path with shorter length, less inflection points and turning angle is not enough for post-disaster rescue, because it may still fail to satisfy the mobility limitations of the vehicle [22]. Therefore, further optimization of the initial path is required to produce a continuous and smooth final path. During the optimization process of trajectory, the curve fitting disregards obstacles, leading to the presence of collision risks. Therefore, it is necessary to conduct collision detection after smoothing and modify any inadequate smoothing parameters. This may trigger repeated smoothing and collision detection. Safe corridor is applied to the trajectory optimization in this paper by add constraint, which can avoid the collision between the final planned curve and the obstacle, and avoid the repeated smoothing and detection processing of the program. The procedure of setting safe corridor on the grid map is list as follows:

- 1) Generate points uniformly along the path connecting the start and end points.
- 2) Expand each point around and take the largest rectangular space it can get.
- 3) Connect the rectangular space in pairs, and finally form a continuous safety corridor with spatial constraints.
- 4) Trajectory optimize the original path within the safety corridor with Bézier [23], [24], [25] curves.

Fig. 3 shows the effects of trajectory optimization with and without safe corridor. The advantages of trajectory optimization with safe corridor mainly include:

The trajectory-optimization path does not collide with obstacles. As shown by the red solid line, the safety corridor completely restricts the spatial extent of trajectory optimization and the planned path will not collide with obstacles.

Since the collision between the path and the obstacle is avoided, repeated smoothing and detection process can be exempted to reduce the running time to some extent. An accuracy path can be planned in high efficiency.

## III. SIMULATION AND RESULTS ANALYSIS

Simulation was carried to verify the advantages of the algorithm. A computer with an i5 processor and 12G running memory was used for the simulation. The planning simulation environment is the test site for trajectory tracking at the later stage of the project, which have been mapped with simultaneous localization and mapping (SLAM)

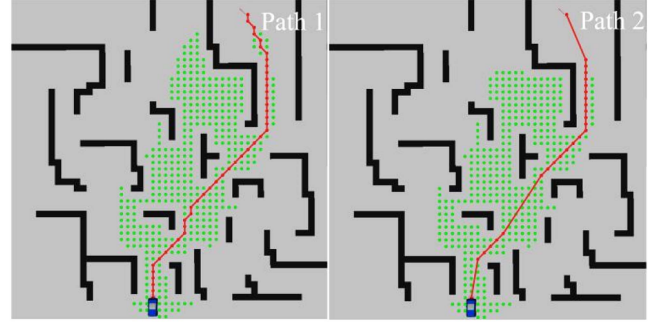


Fig. 4. Comparison of the paths planned by the two algorithms. Path 1: Paths planned with the initial algorithm. Path 2: Paths planned with the improved algorithm (green grid: Search area; red line: Planned path; black object: Obstacle).

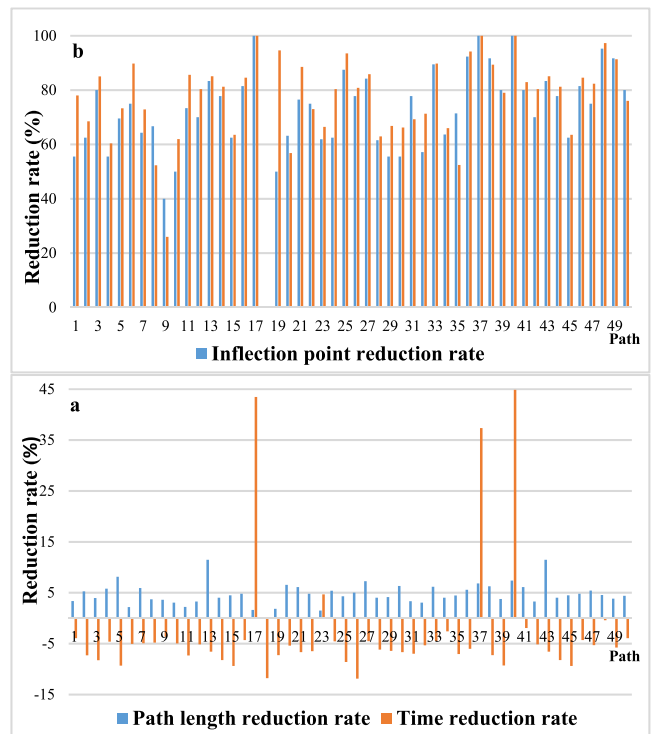


Fig. 5. Reduction rate of path length, time, number of inflection points, and turning angle. 50 sets of data are calculated as  $\frac{\text{preimprovement data} - \text{improvement data}}{\text{preimprovement data}} \times 100\%$ .

techniques. The path planning framework was built with robot operating system (ROS). Fifty simulation experiments were conducted to verify the effectiveness of the algorithm on unstructured roads.

Fig. 4 shows one set of the experiments, which path 1 is the result of initial algorithm with many turning points, large turning angles, and a relatively longer path. Path 2 is the result of improved algorithms which include turn cost, prejudgment planning strategy and redundant inflection point elimination strategy. The results demonstrate a substantial reduction in the number of turning points and turning angles in the path. Besides, the curve is smoother compared to that planned with the initial algorithm. When no obstacle is found between a node and the target point, a direct connection can be established, thus reducing both the search time and path length of the algorithm.



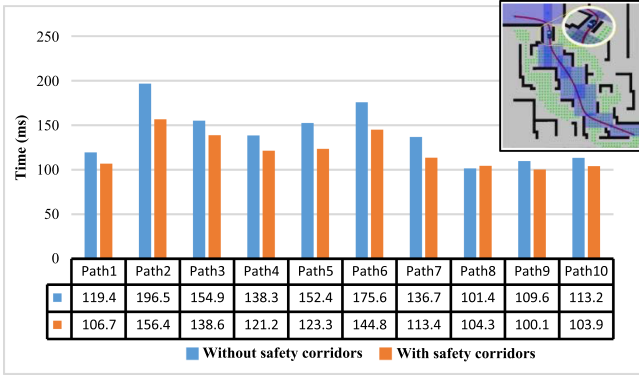


Fig. 6. Comparison of computation time of trajectory optimization with and without safety corridor constraints. The trajectory diagram in the upper right corner shows the final result after trajectory optimization.

Fig. 5 illustrates the results of 50 simulation experiments conducted to evaluate the algorithm's effectiveness in reducing path length, time, number of inflection points, and turning angle. On average, the path length was reduced by 4.74%, with a maximum reduction of 11.46% (Fig. 5(a)). When there are no obstacles between the start and end points, the prejudgment strategy can directly connect them to save a significant amount of time (Path 17, path 37 and path 40 in Fig. 5(a)). However, in all other cases, the time spent increases. Consequently, the average increase in time is 3.08%, with a maximum increase of 11.79% and a minimum decrease of -44.87% (Fig. 5(a)).

The average reduction of the number of inflection points is 67.98%, the maximum is 100% and the minimum is 0% (Fig. 5(b)). The average reduction of the turning angle is 71.01%, the maximum is 100%, and the minimum is 0% (Fig. 5(b)). When no obstacle locate between the start and end points and the slope between the two points is  $0^\circ$  or  $90^\circ$ , the improved algorithm does not work, so the number of inflection points and the turning angle are reduced by 0%. When the slope between two points is not equal to  $0^\circ$  or  $90^\circ$ , the starting point and the ending point will be directly connected, and the number of turning points and the turning angle are reduced by 100%.

Simulation results show that the improved algorithm can effectively shorten the path length, reduce the number of inflection points and turning angles, and achieve a more concise planning path, but the computer time has increased slightly.

Inflection points still existed in the current planning path, therefore, further trajectory optimization is necessary carried out to achieve smooth path with continuous curvature. In this paper, the boundary of the safety corridor is attached to the trajectory optimization of the Bessel curve as a strong constraint to form a convex optimization process, which effectively avoids the repeated planning part of the trajectory optimization and saves time greatly.

Ten sets of experiments with different path were conducted to demonstrate the effect of safety corridor and the result are shown in Fig. 6. The simulation result shows that the final trajectory is continuous and smooth, which meets the vehicle motion requirements. And under the constraint of safety corridor, the paths are planned only once, which avoid the original repeated checking and correction and reduce the computer time. From the experimental data, it can be seen that, except for path 8, all the paths save the calculation time of planning to some extent, depending on the complexity of the road conditions. The 10 sets of data save an average of 13.25% of the calculation time, but in the rare cases where the road conditions are particularly simple instead, a tiny amount of calculation time is added.

## IV. CONCLUSION

In this paper, an improved A\* algorithm is proposed to achieve high efficiency and accuracy for path planning on post-disaster field roads. The algorithm introduces redundant safety spaces to filter impassable narrow roads and is able to avoid collisions between vehicles and obstacles. Prejudgment strategy, redundant inflection point elimination strategy and inflection cost function were combined to reduce the path length, inflection points and turning angle. Furthermore, we also introduce safety corridor into trajectory optimization to achieve smoother path and save computer time. According to the simulation results, the improved algorithm can effectively avoid collisions with obstacles and vehicle contours. The path is smoother and the turns of the path are reduced significantly, and the total length of the path has been reduced to a certain extent. Repeated planning is avoided in the trajectory optimization, and time is effectively saved. In a random 50 path planning, the length of the path, number of inflection points and turning angle is reduced by about 4.74%, 67.98% and 71.01% averagely respectively, while the computation time only improve by about 3.08%. Ten sets of trajectory optimization show that the strong constraint on the trajectory optimization with the safe corridor method reduces the time by 13.25% on average.

This approach proves highly effective for path planning in fixed environments with unstructured roads. Yet, in dynamic scenarios, it becomes imperative to integrate dynamic windowing methods to align with practical needs. Addressing this aspect constitutes the next research objective for the authors.

## REFERENCES

- [1] F. H. Ajeil, I. K. Ibraheem, A. T. Azar, and A. J. Humaidi, "Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments," *Sensors*, vol. 20, no. 7, 2020, Art. no. 1880, doi: [10.3390/s20071880](https://doi.org/10.3390/s20071880).
- [2] C. Yuan, S. Weng, J. Shen, L. Chen, Y. He, and T. Wang, "Research on active collision avoidance algorithm for intelligent vehicle based on improved artificial potential field model," *Int. J. Adv. Robot. Syst.*, vol. 17, no. 3, pp. 1–10, 2020, doi: [10.1177/1729881420911232](https://doi.org/10.1177/1729881420911232).
- [3] M. P. Strub and J. D. Gammell, "Advanced BIT\* (ABIT\*): Sampling-based planning with advanced graph-search techniques," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 130–136, doi: [10.1109/ICRA40945.2020.9196580](https://doi.org/10.1109/ICRA40945.2020.9196580).
- [4] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized path planning," in *Proc. 19th Int. Conf. Auton. Agents MultiAgent Syst.*, 2020, pp. 1901–1903.
- [5] S. Cheng, Z. Wang, B. Yang, and K. Nakano, "Convolutional neural network-based lane-change strategy via motion image representation for automated and connected vehicles," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 18, 2023, doi: [10.1109/TNNLS.2023.3265662](https://doi.org/10.1109/TNNLS.2023.3265662).
- [6] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. L. P. Chen, "Design and implementation of deep neural network-based control for automatic parking maneuver process," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 4, pp. 1400–1413, Apr. 2022, doi: [10.1109/TNNLS.2020.3042120](https://doi.org/10.1109/TNNLS.2020.3042120).
- [7] R. Chai, D. Liu, T. Liu, A. Tsourdos, Y. Xia, and S. Chai, "Deep learning-based trajectory planning and control for autonomous ground vehicle parking maneuver," *IEEE Trans. Automat. Sci. Eng.*, vol. 20, no. 3, pp. 1633–1647, Jul. 2023, doi: [10.1109/TASE.2022.3183610](https://doi.org/10.1109/TASE.2022.3183610).
- [8] R. Chai, H. Niu, J. Carrasco, F. Arvin, H. Yin, and B. Lennox, "Design and experimental validation of deep reinforcement learning-based fast trajectory planning and control for mobile robot in unknown environment," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 10, 2022, doi: [10.1109/TNNLS.2022.3209154](https://doi.org/10.1109/TNNLS.2022.3209154).
- [9] Y. Sun, M. Fang, and Y. Su, "AGV path planning based on improved Dijkstra algorithm," *J. Phys.: Conf. Ser.*, vol. 1746, no. 1, 2021, Art. no. 012052, doi: [10.1088/1742-6596/1746/1/012052](https://doi.org/10.1088/1742-6596/1746/1/012052).
- [10] 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.*, vol. 17, no. 5, pp. 1–13, 2020, doi: [10.1177/1729881420962263](https://doi.org/10.1177/1729881420962263).

- [11] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, "Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment," *IEEE Access*, vol. 9, pp. 59196–59210, 2021, doi: [10.1109/ACCESS.2021.3070054](https://doi.org/10.1109/ACCESS.2021.3070054).
- [12] C. Ju, Q. Luo, and X. Yan, "Path planning using an improved a-star algorithm," in *Proc. IEEE 11th Int. Conf. Prognostics Syst. Health Manage.*, 2020, pp. 23–26, doi: [10.1109/PHM-Jinan48558.2020.00012](https://doi.org/10.1109/PHM-Jinan48558.2020.00012).
- [13] I. Maurovic, M. Seder, K. Lenac, and I. Petrovic, "Path planning for active SLAM based on the D\* algorithm with negative edge weights," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 48, no. 8, pp. 1321–1331, Aug. 2018, doi: [10.1109/TSMC.2017.2668603](https://doi.org/10.1109/TSMC.2017.2668603).
- [14] R. Ebendt and R. Drechsler, "Weighted A\* search - unifying view and application," *Artif. Intell.*, vol. 173, no. 14, pp. 1310–1342, 2009, doi: [10.1016/j.artint.2009.06.004](https://doi.org/10.1016/j.artint.2009.06.004).
- [15] K. Gochev, A. Safonova, and M. Likhachev, "Anytime tree-restoring weighted A\* graph search," in *Proc. 7th Annu. Symp. Combinatorial Search*, 2014, pp. 80–88, doi: [10.1609/socs.v5i1.18321](https://doi.org/10.1609/socs.v5i1.18321).
- [16] M. Likhachev, G. Gordon, and S. Thrun, "ARA\*: Anytime A\* with provable bounds on," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 767–774.
- [17] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A\*," *Int. J. Robot. Res.*, vol. 35, no. 1–3, pp. 224–243, 2016, doi: [10.1177/0278364915594029](https://doi.org/10.1177/0278364915594029).
- [18] F. Islam, V. Narayanan, and M. Likhachev, "Dynamic multi-heuristic A\*," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 2376–2382.
- [19] F. Islam, "A\* -connect : Bounded suboptimal bidirectional heuristic search," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 2752–2758.
- [20] S. Klemm et al., "RRT\*-connect: Faster, asymptotically optimal motion planning," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2015, pp. 1670–1677, doi: [10.1109/ROBIO.2015.7419012](https://doi.org/10.1109/ROBIO.2015.7419012).
- [21] H. Ma, W. Honig, T. K. S. Kumar, N. Ayanian, and S. Koenig, "Extended abstract: Lifelong path planning with kinematic constraints for multi-agent pickup and delivery?," in *Proc. 12th Int. Symp. Combinatorial Search*, 2019, pp. 190–191, doi: [10.1609/socs.v10i1.18485](https://doi.org/10.1609/socs.v10i1.18485).
- [22] S. Cheng, Z. Wang, B. Yang, L. Li, and K. Nakano, "Quantitative evaluation methodology for chassis-domain dynamics performance of automated vehicles," *IEEE Trans. Cybern.*, vol. 53, no. 9, pp. 5938–5948, Sep. 2023, doi: [10.1109/TCYB.2022.3219142](https://doi.org/10.1109/TCYB.2022.3219142).
- [23] B. Song, Z. Wang, and L. Zou, "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve," *Appl. Soft Comput.*, vol. 100, 2021, Art. no. 106960, doi: [10.1016/j.asoc.2020.106960](https://doi.org/10.1016/j.asoc.2020.106960).
- [24] Z. Duraklı and V. Nabiyevev, "A new approach based on Bezier curves to solve path planning problems for mobile robots," *J. Comput. Sci.*, vol. 58, 2022, Art. no. 101540, doi: [10.1016/j.jocs.2021.101540](https://doi.org/10.1016/j.jocs.2021.101540).
- [25] L. Zheng, P. Zeng, W. Yang, Y. Li, and Z. Zhan, "Bézier curve-based trajectory planning for autonomous vehicles with collision avoidance," *Inst. Eng. Technol. Intell. Transport Syst.*, vol. 14, no. 13, pp. 1882–1891, 2020, doi: [10.1049/iet-its.2020.0355](https://doi.org/10.1049/iet-its.2020.0355).