Review Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
September-October 2020: 1–13
© The Author(s) 2020
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1729881420962263
journals.sagepub.com/home/arx
SAGE

An improved A-Star based path planning algorithm for autonomous land vehicles

Shang Erke[1,2], Dai Bin[1], Nie Yiming[1], Zhu Qi[1], Xiao Liang[1] and Zhao Dawei[1]

Abstract

This article presents a novel path planning algorithm for autonomous land vehicles. There are four main contributions: Firstly, an evaluation standard is introduced to measure the performance of different algorithms and to select appropriate parameters for the proposed algorithm. Secondly, a guideline generated by human or global planning is employed to develop the heuristic function to overcome the shortcoming of traditional A-Star algorithms. Thirdly, for improving the obstacle avoidance performance, key points around the obstacle are employed, which would guide the planning path to avoid the obstacle much earlier than the traditional one. Fourth, a novel variable-step based A-Star algorithm is also introduced to reduce the computing time of the proposed algorithm. Compared with the state-of-the-art techniques, experimental results show that the performance of the proposed algorithm is robust and stable.

Keywords

Autonomous land vehicles, path planning, improved A-Star, guideline, key point

Date received: 6 May 2020; accepted: 5 September 2020

Topic Area: Mobile Robots and Multirobot Systems
Topic Editor: Nak-Young Chong
Associate Editor: Genci Cap

Introduction

Autonomous land vehicle (ALV), as a typical robot, is widely researched recently.[1] The development of ALVs is based on core components, such as environment perception, path planning, vehicle control, position localization, and so on. Among these components, path planning is a crucial function to determine the maneuvers of ALVs because it provides the path motion operations of vehicles according to the environment perception information.[2]

In the last decades, lots of techniques for motion planning strategies have emerged. These path planning algorithms are generally classified into four classes[3]: graph search algorithms,[4,5] sampling algorithms,[2] interpolating algorithms,[6] and numerical optimization algorithms.[7] Among these presented algorithms, the A-Star algorithm and its various improved algorithms are widely studied and implemented.[4,5,8,9] The idea of the A-Star algorithm is first introduced in the article,[10] derived from the Dijkstra's graph search algorithm, which introduced a heuristic function to enable a fast node search. Several successful applications in real ALVs had been implemented, such as a hybrid A-Star was applied in Junior,[4,5] and some other

[1] National Innovation Institute of Defense Technology (NIIDT), Academy of Military Sciences, Beijing, People's Republic of China
[2] Tianjin Artificial Intelligence Innovation (TAIIC), Tianjin, People's Republic of China

Corresponding author:
Zhu Qi, National Innovation Institute of Defense Technology (NIIDT), Academy of Military Sciences, Beijing 100071, People's Republic of China.
Email: Zhuqi@163.com

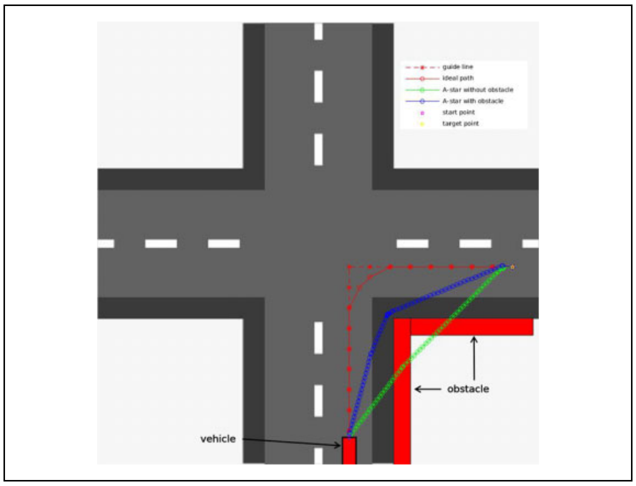

Figure 1. The problem of the classical A-Star algorithm when it is applied in ALVs (I). ALV: autonomous land vehicle.

variations were applied both in AnnieWAY[8] and Boss,[11] all of these vehicles had joined in DARPA challenges.

The advantage of the A-Star algorithm is simple and relatively fast (compared to the Dijkstra algorithm), while its limitation is as follows: the classical A-Star is not suitable when the vehicle turns a corner, as shown in Figure 1. As we know, the heuristic function $F = G + H$ is applied to accelerate the search process to the target. G means the cost that has finished. $H = \mathrm{dis}(C, T)$, in which C means the current position, and T means the target position. The function $\mathrm{dis}(A, B)$ means the distance between position A and position B. This classical heuristic function would quickly guide the search toward the target, as shown in Figure 1. Figure 1 shows a scene that the ALV would turn the right corner, the "guideline" is given by the global planning or human. The planning algorithm is expected to generate a result similar to the "ideal path." The real results planned by the classical A-Star algorithm are shown as "A-Star without obstacle" and "A-Star with obstacle," which are generated by the literature.[12] Although it is not A-Star's fault, in many scenes, indeed, the road edge is not easy to be detected by perception. Thus, both two results are unsatisfactory in Figure 1. Therefore, the classical A-Star algorithm is not suitable for application in such scenes.

Another problem with the classical A-Star algorithm is the selection of parameters. Several key parameters should be set before this algorithm is applied, such as the size of θ, the length of the search step, and so on. These parameters would greatly influence the performance both on the quality and the time-consuming. For example, different lengths of the search step would bring a different result, as shown in Figure 2. Under the same condition, four different steps ($\mathrm{step} = 1, 3, 6, 9$) are applied by the classical A-Star algorithm offered by the literature,[12] and their results are shown in Figure 2. In Figure 2, the position and direction of the start point and the target point are fixed, and an obstacle is located between the start point and the target point. Figure 2(a) to (d) shows the result under different steps. It is obvious that the parameter is not appropriate in Figure 2(d) since it brought a wrong result.

The computing time cost of the A-Star algorithm dramatically depends on the number of expanding points.[9] The number of expanding points under four different steps is listed in Table 1. It is shown that the larger the step length, the less the amount of expansion nodes.

Through the above analysis, it is difficult to choose a set of suitable parameters, though these parameters are essential for the A-Star algorithm to achieve a better outcome.

This article presents an improved A-Star based path planning algorithm for ALVs. Be aimed at those shortcomings above, several improvements are presented to raise the quality of the A-Star algorithm in this article. Firstly, a universal evaluation standard is introduced, which is employed to measure the performance of different kinds of graph search-based planning algorithms and also is used for selecting appropriate parameters. Then, a guideline-based A-Star algorithm is presented to overcome the problem of the traditional classical A-Star algorithm, in which the guideline is employed during the search process. Further, a key point-based A-Star algorithm is introduced to improve the quality during obstacle avoidance. Besides, a novel variable-step based A-Star algorithm is also introduced to reduce the computing time of the proposed algorithm. Based on these improvements, the experimental results show that the presented algorithm is valid and reliable.

The remaining of this article is organized as follows: the second section discusses some related works on path planning about ALVs, especially these graph search-based algorithms; the third section introduces the proposed evaluation standard for path planning algorithms; the fourth section describes the improved guideline-based A-Star algorithm; the fifth section describes the improved key point-based A-Star algorithm; the sixth section describes the improved variable-step based A-Star algorithm; the seventh section illustrates the results of real road environmental experiments and conclusions are drawn in the eighth section.

Related works

Path planning is a crucial module for ALVs, which greatly decides the safety, comfort, and energy optimization of ALVs. As stated in the "Introduction" section, four typical classes[3] of path planning algorithms have emerged in the last decades.

In the early decades, the computing resource is limited on the robotic platform. Thus, the computational complexity is much considered. The main idea of a sampling-based planner is trying to solve timing constraints. The most commonly used method in robotics is the rapidly exploring random tree (RRT) method.[13] It allows fast planning in semistructured spaces by executing a random search

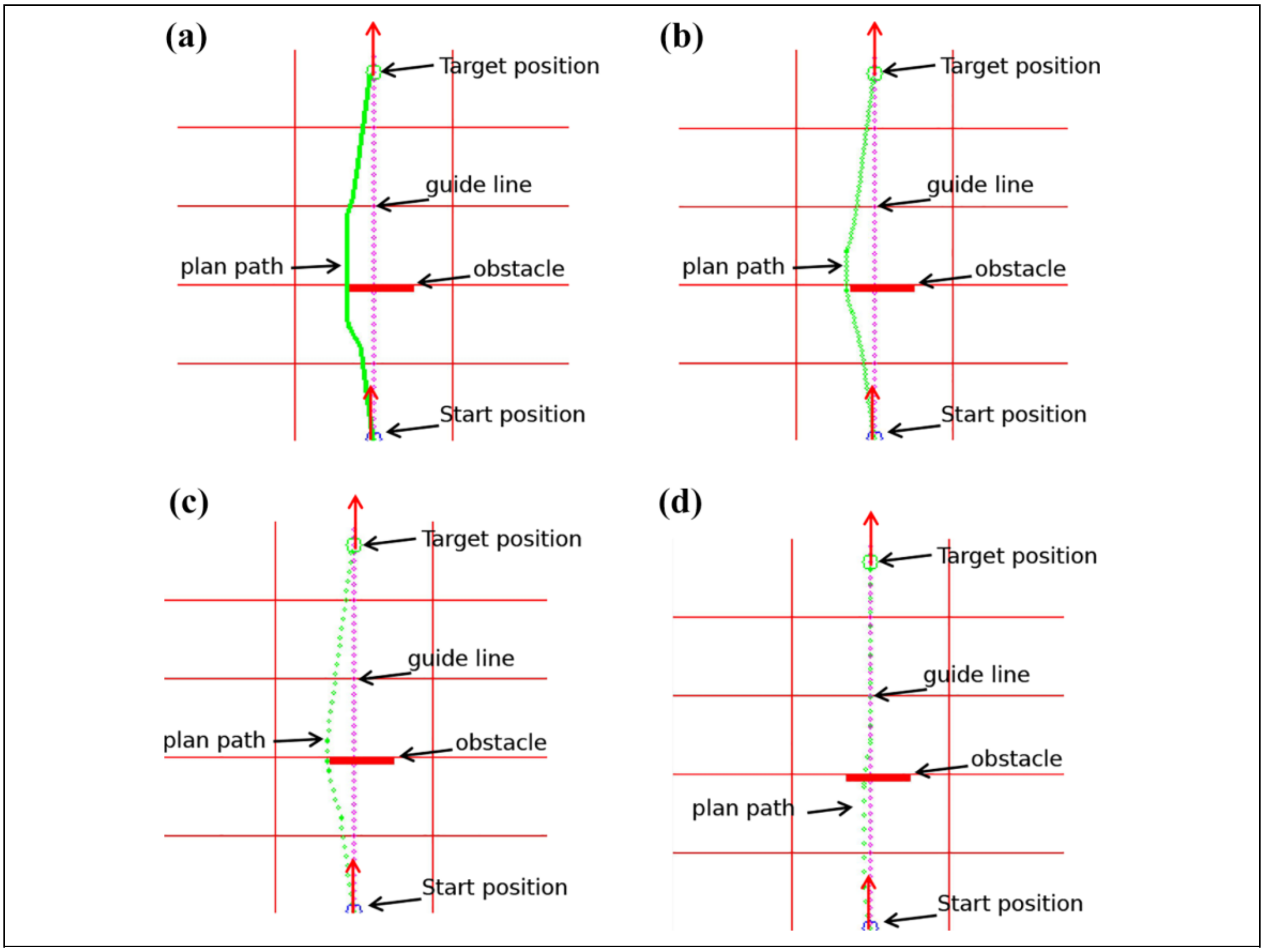

Figure 2. (a–d) The problem of the classical A-Star algorithm when it is applied in ALVs (II). ALV: autonomous land vehicle.

Table 1. The number of expansion nodes under different parameters.

Parameters	Step = 1	Step = 3	Step = 6	Step = 9
Number of expansion nodes	62,866	10,345	8439	3408

through the navigation area. However, the resulting path is not optimal, which would significantly affect its application on ALVs. Thus, a series of developments are presented, such as RRT*.[14] In the literature,[14] two critical extensions to the traditional RRT are introduced: committed trajectories and branch-and-bound tree adaptation. Those two improvements are thought to enable the algorithm more efficiency of computation time online.

Another idea of solving the planning path problem is to build a vehicle model. In the literature,[15] a multibody modeling technique is used to develop a four-wheeled vehicle model. Two coupled controllers are proposed in that article: The first controller is designed by the Lyapunov control technique, while the second one is designed by an immersion and invariance approach. Both of the controllers aim to ensure a robust tracking of the reference trajectory while taking into account the strong coupling between the lateral and the longitudinal vehicle dynamics.

It is also easy to think that different segment road networks can be represented by interpolating known way-points with straight lines and circular shapes, such as cubic splines,[16] intrinsic splines,[17] quintic Bezier splines,[18] clothoids,[6] and so on.

In ALV applications, the planning problem is also easy to be transformed into a state space to get a path from start point A to target B. Therefore, graph search-based planners are also widely used to solve this planning path problem.[5] For example, Dijkstra's graph search algorithm, A-Star algorithm, and their various improved algorithms are extensively studied and implemented in the ALV field.[4,5,8,9]

The article[5] describes a variant of the A-Star algorithm for ALVs and experimentally validated in the 2007 DARPA urban challenge. Their algorithm consists of two phases. In first phase, a hybrid-state A-Star search is used,

Algorithm 1. Evaluation standard for path planning algorithms.

```
Require:
    The test environment and parameters, such as the size of search space, position of start point and target point, obstacle map, search step and so on.
Ensure:
    The cost time $T$ and the number of expansion nodes $N$.
1: Set the test scene, including the size of search space, position and direction of start and target, obstacle map, search step and so on;
2: Set the parameter of evaluation standard, such as the mapping relationship between path curve and virtual speed;
3: Apply the path planning algorithm and comes out the result: a sequence of path points $P = P_i$, and the number of expansion nodes $N$;
4: Compute the distance between two adjacent points, $L_i = dis(P_{i-1}, P_i)$, where the function $dis(a, b)$ means the distance between point $a$ and point $b$;
5: Compute the angle between two adjacent points, $\theta_i = angle(line(P_i, P_{i-1}), line(P_i, P_{i+1}))$, where the function $line(a, b)$ means the line between point $a$ and point $b$, the function $angle(A, B)$ means the angle between line $A$ and line $B$;
6: According to the mapping relationship table, mark a virtual speed $v_i$ for each path point $P_i$, where the bigger $\theta_i$ is, the smaller $v_i$ is. Once the $\theta_i$ is bigger than the real vehicle can be implemented, the virtual speed $v_i$ would be $zero$;
7: According to the equation $T = L \div v$, the total time $T$ that the vehicle would cost the path is computed: $T = \sum T_i$, and $T_i = dis_i \div v_i$, where $dis_i = L_i/2 + L_{i+1}/2$, and $T_i$ means the cost time for $i^{th}$ line;
8: return $(T, N)$.
```

in which a 4D search space $\langle x, y, \theta, r \rangle$ is represented. In the second phase, the quality of the solution is improved by nonlinear numeric optimization.

The article[19] also employed the A-Star algorithm in their application on their embedded device. The motion model is considered as follows: three directions, including forwarding, turning left, and turning right model build beforehand. The most important contribution of this article is that the open-source of their A-Star algorithm is public on.[12]

Recently, several new methods are emerged.[20,21] In the literature,[20] a dynamically integrated spatiotemporal-based trajectory planning and control algorithm is presented for ALVs. A two-level dynamically integrated structure is first introduced. It is said that the best trajectory is selected among a group of candidate time-parameterized trajectories in the upper level, while the trajectory controller based on the vehicle dynamics model is employed to control the vehicle to follow the desired trajectory in the lower level. In the literature,[21] a novel idea of planning algorithm by considering inverse vehicle handling dynamics is introduced. In their article, a tyre model and optimal preview time are established. And a path-tracking controller based on linear quadratic regulator method is designed and calculated for the path-tracking problem.

To address the limitation of the classical A-Star algorithm mentioned above, this article also presents an improved A-Star based path planning algorithm. An evaluation standard for evaluating each path planning algorithm is first introduced. Then, a guideline is employed to improve the heuristic function of the proposed algorithm. A key point method is also employed to improve the ability to avoid collisions. Experimental results show that the proposed algorithms are effective and stable.

Table 2. The mapping relationship between path curve and virtual speed.

	Straight	Turn left	Turn right
Straight	1	α	α
Turn left	α	α	β
Turn right	α	β	α

Evaluation standard

Path planning algorithm is required as safety, comfort, and energy optimization, but there is a lack of evaluation rule to quantitative analysis of those requirements. To estimate various of path planning algorithms and their results, a novel evaluation standard algorithm is first introduced as follows.

The path planning algorithm offered by the literature[12] is employed to describe the usage of the presented evaluation method. The parameter of the search step is employed to show its influence. As mentioned in the article,[12] three directions, including turn left, turn right, and go straight, are used during the search process. Thus, the mapping relationship between path curve and virtual speed is given in Table 2: suppose the virtual speed from straight road to straight road is set unit 1, the virtual speed from straight road to turn left or turn right is set α, and the virtual speed from turn left to turn left or from turn right to turn right is also set α. The virtual speed from turn left to turn right or from turn right to turn left is set β, where $0 < \beta < \alpha < 1$.

In this estimate process, we set $\alpha = 0.5$ and $\beta = 0.25$, nine different search steps (step $= 1 - 9$) are employed, and their results are shown in Figure 3. The cost time T and their middle results are listed in Table 3. In Table 3, "Epoints N" means the number of expansion points, and "Ppoints" means the number of path points. It is obvious that under the parameter step $= 4$, the cost time is the smallest, which means it gets the best performance under the parameter step $= 4$.

The advantage of the proposed evaluation standard is as follows: first, the cost time T that a vehicle passed the planned path is employed to measure the ability of these planning algorithms, which has a physical meaning and is consistent with the real one. Second, virtual distance and virtual speed are used in the proposed algorithm to overcome the difficulty of getting the vehicle model but keeping the tendency consistently with the real one.

Guideline-based A-Star algorithm

The classical A-Star algorithm is not suitable when directly applied in ALVs, such as when the vehicle turns a corner

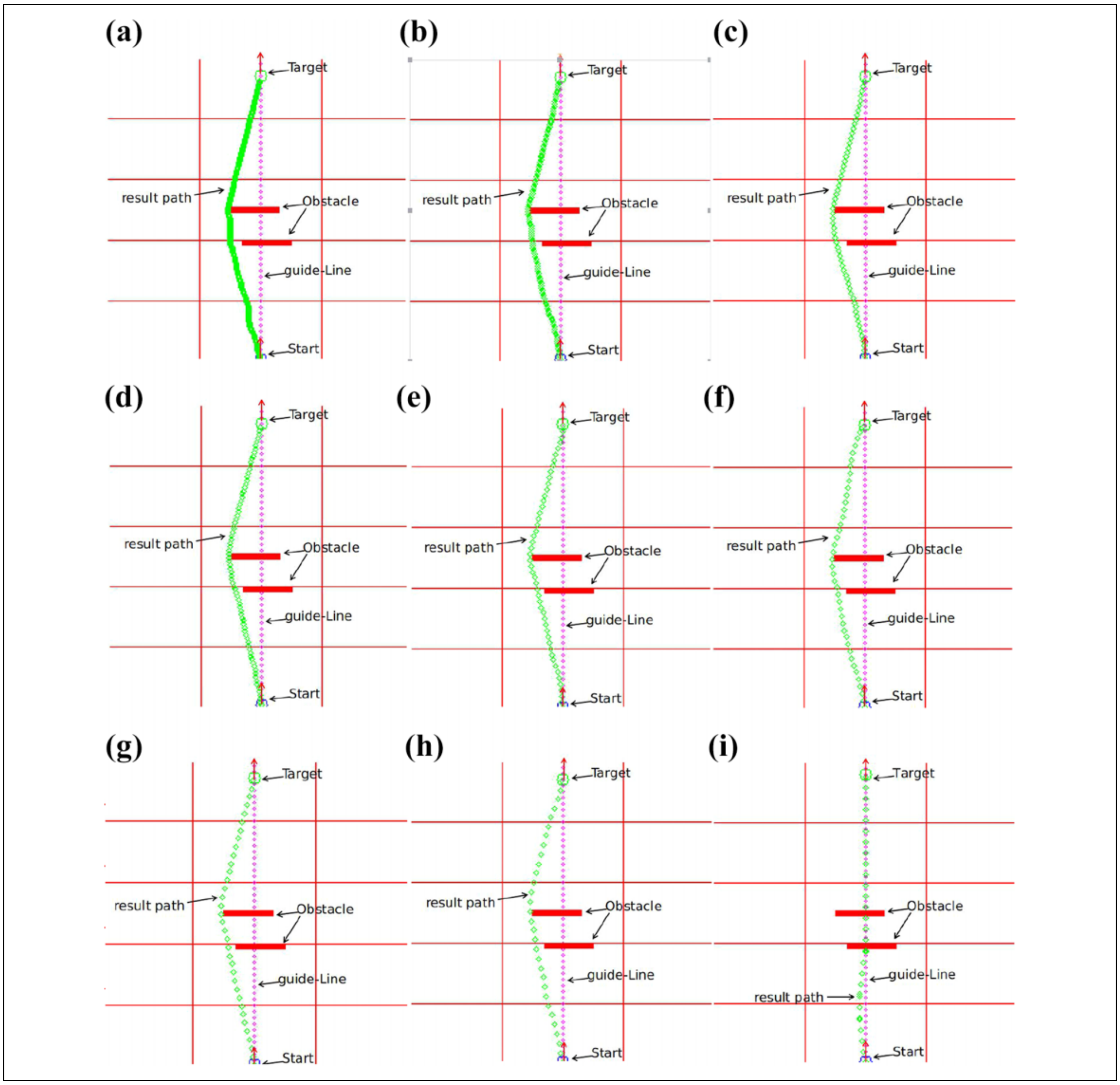

Figure 3. (a–i) The result of planning path under different steps.

Table 3. The results of estimating under different parameters.

Parameters	Epoints *N*	Ppoints	Length	Cost time *T*
Step = 1	89,879	238	249.8131	298.6263
Step = 2	51,922	119	248.8820	293.7640
Step = 3	48,318	88	246.0100	271.0201
Step = 4	16,117	66	245.0794	264.1587
Step = 5	14,168	54	245.8635	269.7270
Step = 6	12,207	46	247.9927	277.9854
Step = 7	9631	40	245.2979	285.5958
Step = 8	7226	36	248.2427	270.4855
Step = 9	—	—	—	—

while the roadside is not detected, as mentioned in Figure 1. In these cases, the purposes of the human driver are not well implemented. Thus, a guideline-based A-Star algorithm is presented in this article to overcome those problems. Guideline, which was generated by a global planner or by humans, expressed the driver's intention. Thus, in our guideline-based algorithm, the guideline is employed to develop the heuristic function to express the driver's intention. The improved heuristic function is defined as follows

$$F(i) = G(i) + H1(i) \times \alpha 1 + H2(i) \times \alpha 2 \quad (1)$$

The heuristic function F is combined by two parts, as the same as in classical one: G means the cost that had finished, and H means an estimated value from the current position to the target. In our algorithm, H is divided into two components: $H1$ is the distance between the point i and the guideline, and $H2$ means the distance from $g(i)$ to the target, where $g(i)$ is on the guideline and is decided by point i. $g(i)$ is computed as follows: when the point i is fixed, distance between the point i and each point in guideline is computed. Then, the minimum distance and its corresponding point on guideline are found, marked as $g(i)$.

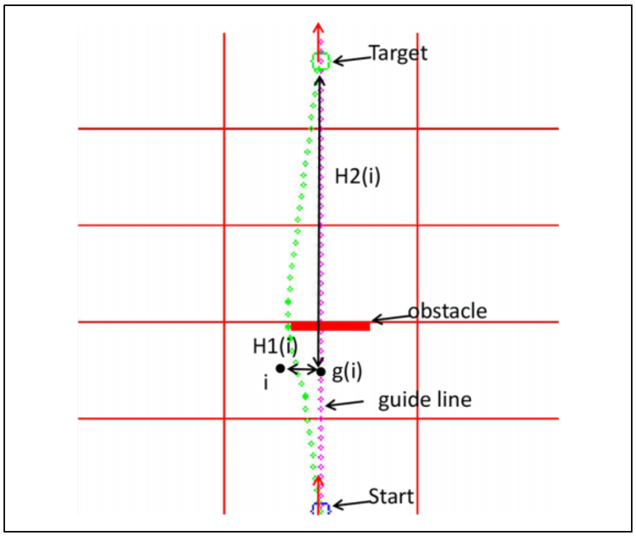

Figure 4. The illustration of physical meaning for $H1$ and $H2$.

The physical meaning of $H1$ and $H2$ is shown in Figure 4. $\alpha1$ and $\alpha2$ are coefficients.

Several compare experiments are carried out to show the performance of the proposed guideline-based A-Star algorithm. The classical algorithm offered by the article[12] is employed as a comparison, named as "Autoware A-Star." Our algorithm is developed based on the "Autoware A-Star," named as "guideline-based A-Star." Experimental results are shown in Figure 5. Two scenes, a crossroad scene and a curve road scene, are employed to test the quality of these two algorithms. In Figure 5(a), two results, with obstacles and without obstacles, are both listed by the "Autoware A-Star," since maybe roadside would be detected or not. The scene in Figure 5(b) shows a typical curve road, which is very common in both country road and highway. In both these scenes, the results offered by the "Autoware A-Star" are not suitable, even would bring danger to the ALV. Contrary to "Autoware A-Star," the proposed algorithm is well done in these testing scenes.

Key point-based A-Star algorithm

The proposed guideline-based A-Star algorithm would work well on crossroad and curve road as analyzed above, but it would take troubles when the guideline cross an obstacle. The guideline would lead the potential path toward the obstacle according to the improved heuristic function F, which is not only dangerous to the ALV but also contrary to the human driver's intention. Thus, a key point-based idea is presented to develop the guideline-based algorithm in this section. The main idea is that when

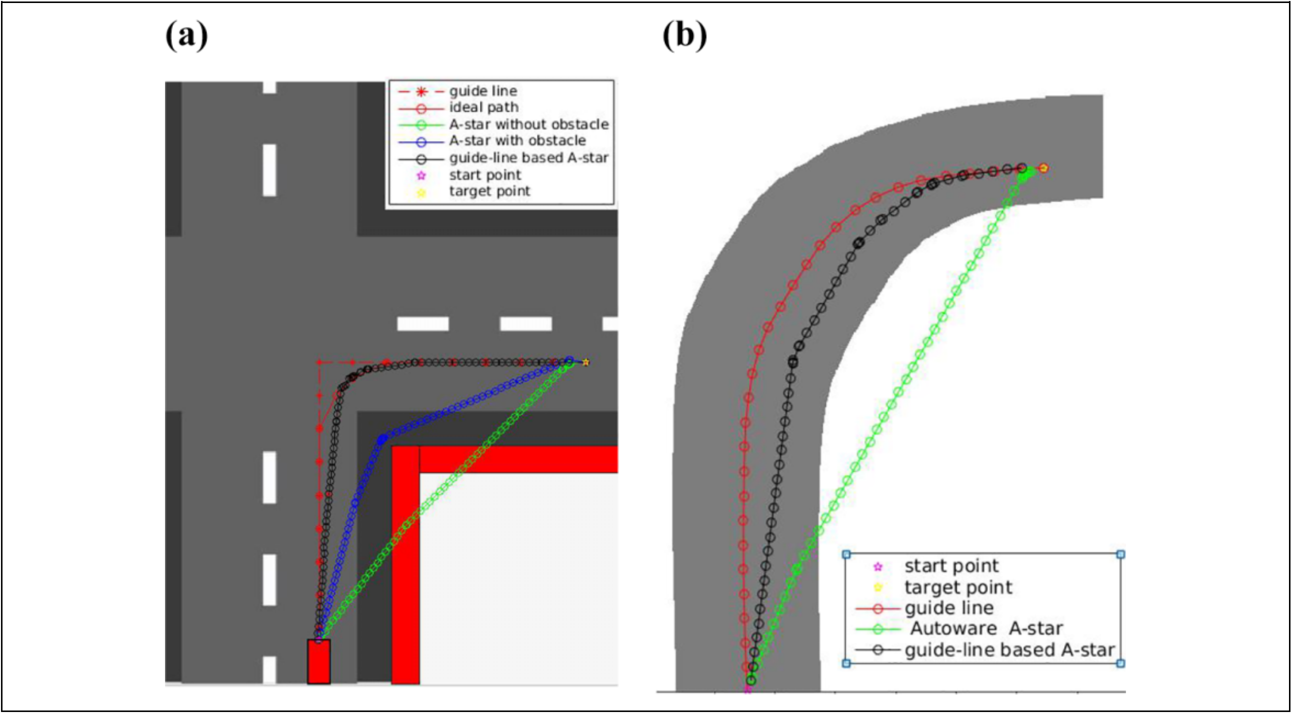

Figure 5. (a, b) The results of the proposed guideline-based A-Star algorithm.

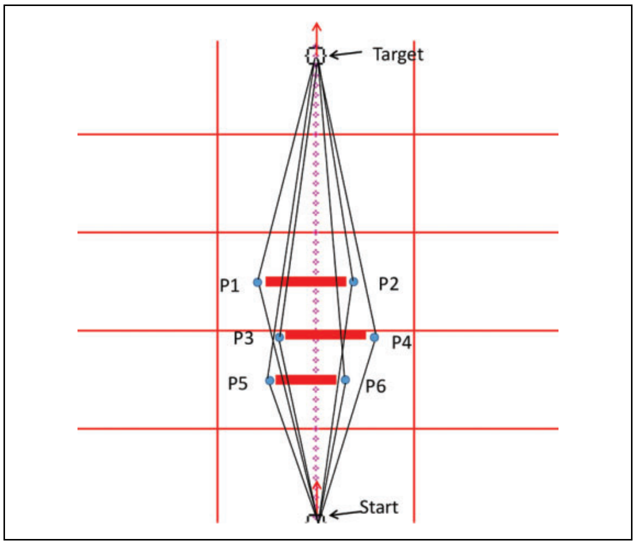

Figure 6. The process of the proposed key point-based A-Star algorithm.

the guideline is on an obstacle, then find a new guideline that not on obstacles. The proposed key point-based A-Star algorithm is described as follows:

Step 1: Set the test scene, including the size of search space, guideline, the start point and target point, obstacle map, and so on;

Step 2: Check the guideline from the start position to the target, find whether there are obstacles on the guideline;

Step 3: If there is no obstacle on the guideline, go to return, otherwise, go to next step;

Step 4: Find all candidate key points according to those obstacles crossed by the guideline. Two sides of the obstacle are found as candidate key points. In order to describe this process more clear, an example is shown in Figure 6: there are three obstacles on the guideline, and "P1–P6" are six candidate points;

Step 5: Generate "n-order ($n = 2$)" candidate guidelines as follows: link these candidate points with the start point and the target point to generate new lines, such as the line "S-P6-T" means the line from the start point to key point $P6$, and finally to the target point. Thus, there are six "2-order" candidate guidelines: "S-P1-T, S-P2-T, S-P3-T, S-P4-T, S-P5-T, S-P6-T." The "n-order" means there are "$n-1$" key points between the start to target. For example, "two-order" candidate guideline means there is one key point, such as the line "S-P1-T," "3-order" candidate guideline means there are two key points, such as the line "S-P5-P1-T," and so on;

Step 6: Check whether there are obstacles on these new candidate guidelines, and compute their cost times by the proposed evaluation standard;

Step 7: To find a candidate line, first, meet the requirement that there is no obstacle to it, and then the cost time is as short as possible. If there is no candidate line that meets the condition, go to next step; otherwise, this new candidate guideline would instead of the original one, then go to return; In this example shown in Figure 6, only one line "S-P4-T" meets the requirement that there is no obstacle on it. Thus, $P4$ is the key point, and line "S-P4-T" would become the new guideline.

Step 8: Make $n = n + 1$, then generate "n-order" candidate guidelines, and go to step 6;

Return: Apply the guideline-based A-Star algorithm and return the result.

Several compared experiments are carried out to show the performance of the proposed key point-based A-Star algorithm. The "Autoware A-Star" algorithm is employed as a comparison again, and the guideline-based A-Star algorithm is employed as a comparison, too. All of the parameters in three algorithms are set as the same, and their search step is fixed as step = 4, which is thought the optimum one. The experimental result is shown in Figure 7, and their quality estimated by the proposed estimate standard is listed in Table 4. In Table 4, "Epoints N" means the number of expansion points, and "Points" means the

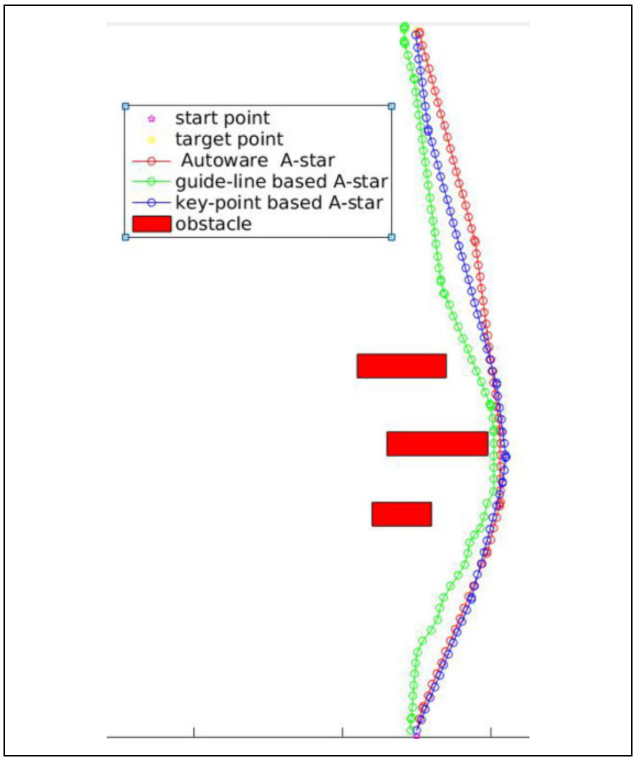

Figure 7. Experimental results of the proposed key point-based A-Star algorithm, and comparison with other algorithms.

Table 4. The results of evaluation for different algorithms.

Algorithm	Epoints N	Ppoints	Length	Cost time T
Autoware	33,654	68	246.6846	286.3693
Guideline	93,888	75	251.9219	305.8438
Key point	11,102	67	246.6445	275.2890

Algorithm 2. Describe the strategy of variable step.

```
Require:
    current point , obstacle map.
Ensure:
    search step $ReT$.
1: Set parameter MaxStepValue, MinStepValue;
2: Search in MinStepValue radius with current point as the center
   in obstacle map, if there is existing some obstacle, set $ReT = MinStepValue$ and go to return, else go to next step;
3: Search in MaxStepValue radius with current point as the center
   in obstacle map, if there is not existing any obstacle, set $ReT = MaxStepValue$ and go to return, else go to next step;
4: Compute the distance $dis$ between current point and the obstacle, and
   let $ReT = dis \times \gamma$, where $\gamma \in (0, 1)$;
5: return search step $ReT$.
```

number of path points. It is obvious that the performance of the proposed key point-based A-Star algorithm is best.

Variable-step based A-Star algorithm

The proposed key point-based A-Star algorithm could optimize the planning path for ALVs, while the drawback of high time and space complexity is still existence, which is the main limitation of real application for the A-Star algorithm. Under the same search space, the number of expanding points means the time complexity, which is smaller, the performance is better. To reduce the time complexity, a variable-step based A-Star algorithm, based on the proposed key point one, is presented in this article. In the variable-step based A-Star algorithm, the search step is variable according to the distribution of obstacles. The advantage of this algorithm is that the search step can be big and stride in the open area, and the search step can be smaller, avoiding during near obstacles. The strategy of variable step is introduced in Algorithm 2 as follows.

To rise the computational efficiency, the article[12] offered a calculation template for computing expanding point during the initialization of the algorithm in its open source. This method would greatly reduce these sin and cos computing operation repeatedly. Therefore, a discrete variable step strategy is used in the proposed algorithm. For example, MinStepValue $= 1$ and MaxStepValue $= 9$ are set for the biggest and smallest search step, and ReT $= \text{round}(\text{dis} \times \gamma)$, where the function round(a) means to take an integer of a. Therefore, ReT $\in [2, 3, 4, 5, 6, 7, 8]$. Similar to the article,[12] calculation templates under all of the different variable steps are firstly generated during the initialization stage.

The same compared experiment is also carried out to show the performance of the proposed variable-step based A-Star algorithm. The result is shown in Figure 8, and details are listed in Table 5. In Table 5, "Epoints N" means the number of expansion points, and "Points" means the number of path points. Though the "cost time" of the proposed variable-step based A-Star algorithm is a little less than that of the key point-based A-Star algorithm (280.6477 : 275.2890), the computational efficiency is greatly increased (1426 : 11102). Thus, this improvement would be meaningful when the algorithm is applied under a limited computing resource environment or that require real-time performance.

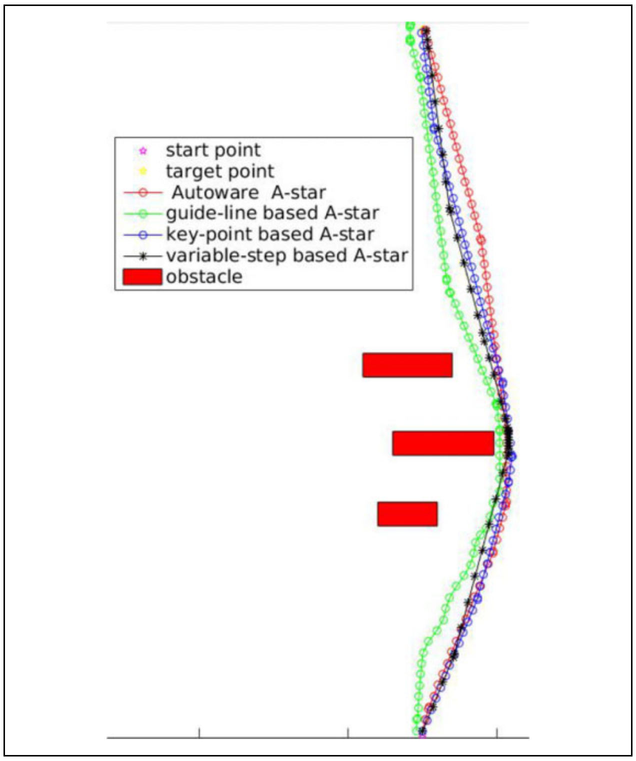

Figure 8. Experimental results of the proposed variable-step based A-Star algorithm, and comparison with other algorithms.

Table 5. The results of estimating for different algorithms.

Algorithm	Epoints N	Ppoints	Length	Cost time T
Autoware	33,654	68	246.6846	286.3693
Guideline	93,888	75	251.9219	305.8438
Key point	11,102	67	246.6445	275.2890
Variable step	1426	46	246.3238	280.6477

Experimental results and analysis

To evaluate the proposed algorithm, experiments are designed in various kinds of environments. Our experiments are composed of three parts: In part I, an open square with several fixed obstacles is employed to compare the proposed algorithm with the traditional A-Star algorithm. In part II, a real structured city scenario is employed to analyze the effectiveness of the presented algorithm. Sequence performances are also listed to check its ability to avoid obstacles. In part III, the proposed algorithm is widely applied in city scenarios with long distances to test its robustness.

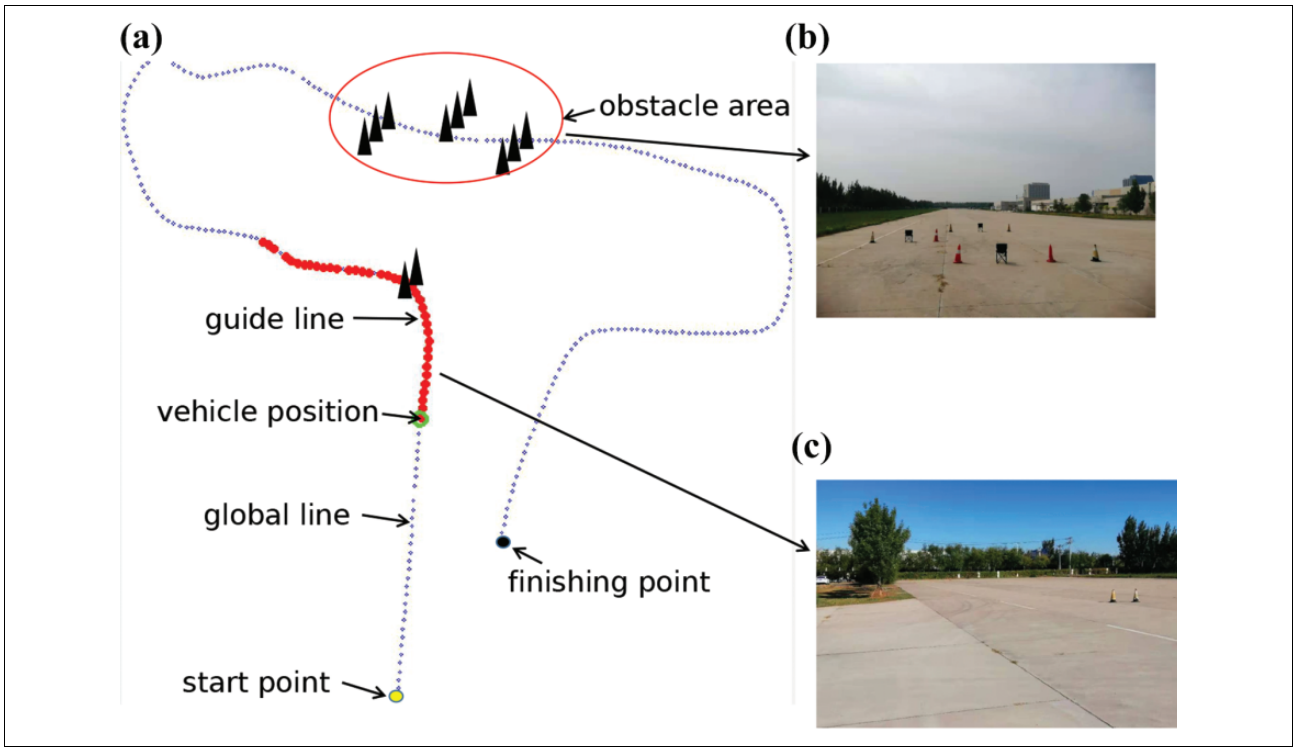

Figure 9. The first real scene is designed to check the performance of the proposed planning algorithm. (a) The global line in an open square and (b, c) two parts of obstacles to check the avoiding performance of the proposed algorithm.

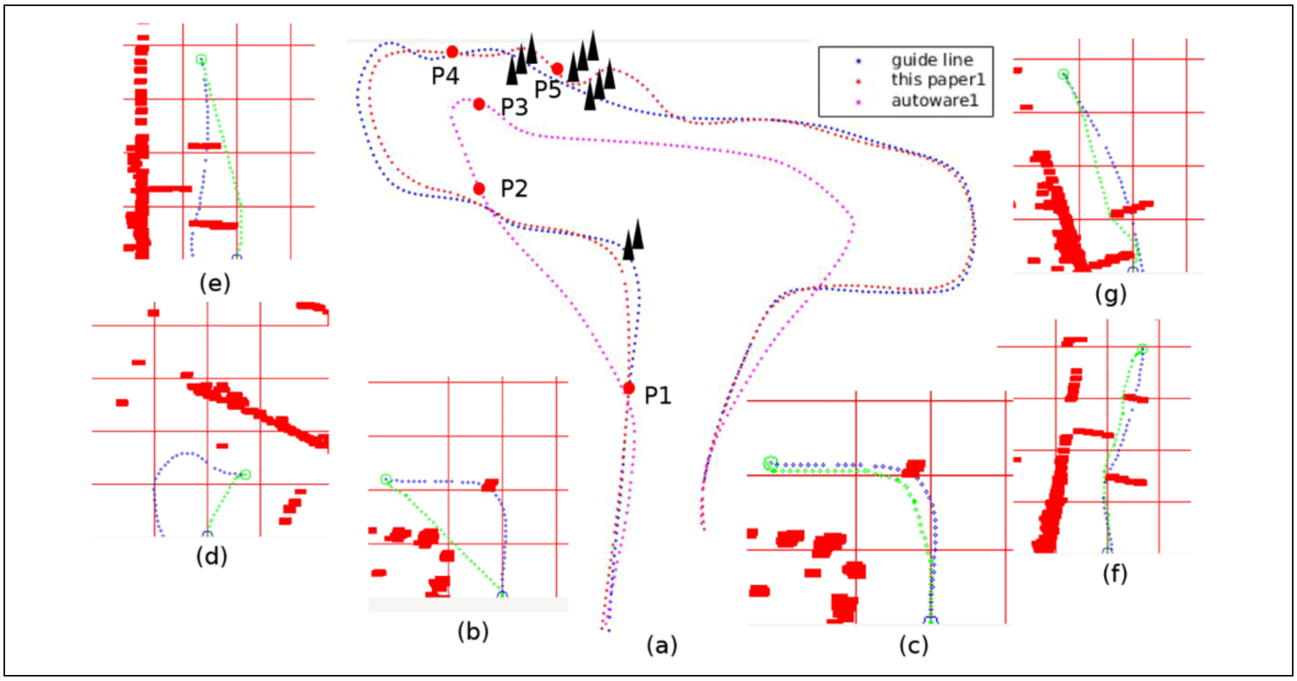

Figure 10. The results of both our algorithm and the article[12] tested in the first designed scene. (a) The global line (blue trajectory), trajectories generated by our algorithm (red trajectory), and the compared algorithm (pink trajectory) are shown. (b)–(g) Results at different positions generated by each algorithm are shown.

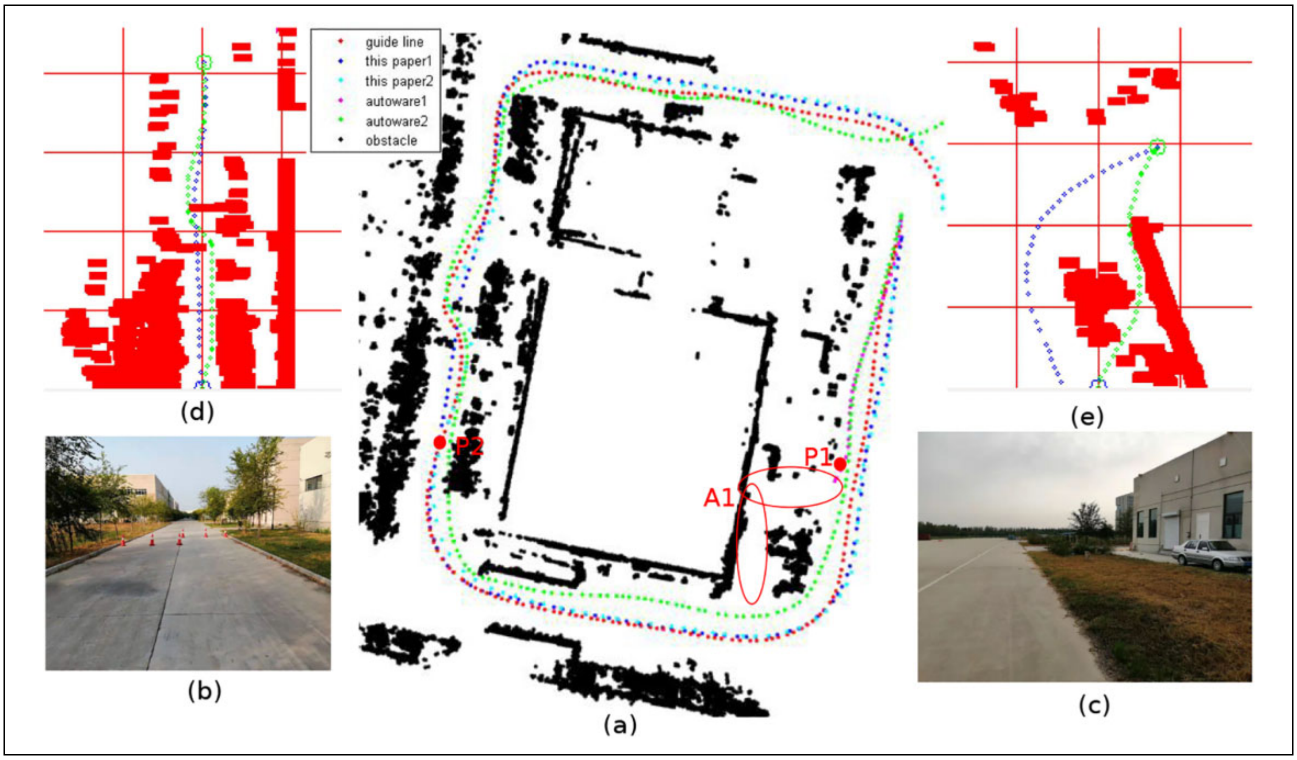

Figure 11. The results of both our algorithm and the article[12] tested in a real structured city scenario. (a) The global line, trajectories generated by our algorithm (the red and cyan trajectories), and trajectories generated by the compared algorithm (the pink and green trajectories) are shown. (b) Two groups of obstacles that placed in the middle of the road are shown. (d) The planning result that generated by our algorithm in scene (b) is shown; (c) the corresponding place of area $A1$; (e) the planning result that generated by the compared algorithm at position $P1$ is shown.

Experimental results in an open square

An open square with several obstacles is designed to verify the performance of the proposed algorithm. The scene is designed as follows (shown in Figure 9): global line is generated in an open square, and then two parts of obstacles are placed on the global line (which is represented by the black triangle). The position of the vehicle is marked as a green circle, and the guideline front of the vehicle is marked as a red circle. The distribution of obstacles of two parts is shown in Figure 9(b) and (c). In Figure 9(b), three groups of obstacles are S bending distributed, which is often used to check the performance of planning algorithms. In Figure 9(c), a group of obstacles is placed at a turn position to check the turning performance of the proposed planning algorithm.

Figure 10 shows the results by the proposed algorithm and by the compared one.[12] Their trajectories are listed in Figure 10(a) by red trajectory (generated by the proposed algorithm) and by pink trajectory (generated by the compared algorithm), where the blue trajectory is the guideline. Figure 10(b) shows the planning result generated by the compared algorithm at position $P1$, while Figure 10(c) shows the planning result generated by the proposed algorithm at the same position. It is obvious that the compared algorithm would bring the wrong way when at a turning place. Figure 10(d) shows the planning result generated by the compared algorithm at position $P2$, and Figure 10(e) shows the planning result generated by the compared algorithm at position $P3$. Figure 10(f) shows the planning result generated by our algorithm at position $P4$, and Figure 10(g) shows the planning result generated by our algorithm at position $P5$. From this experiment, we can see that the compared algorithm would bring trouble during turning place, while the proposed algorithm would greatly drive according to the global line. And our algorithm has a good performance when avoiding those groups of obstacles.

Experimental results in real structured city scenario

Another experiment is carried out in a structured city scenario to test the performance of the proposed algorithm, as shown in Figure 11, which is a part of the street block. Two algorithms, the proposed algorithm and the literature,[12] are employed again. The street scenario is a very common scene, and there are people, cars, trees, and edges beside the road in various forms (it may be not easy to be detected by computers), and the ALV would usually drive among these environments.

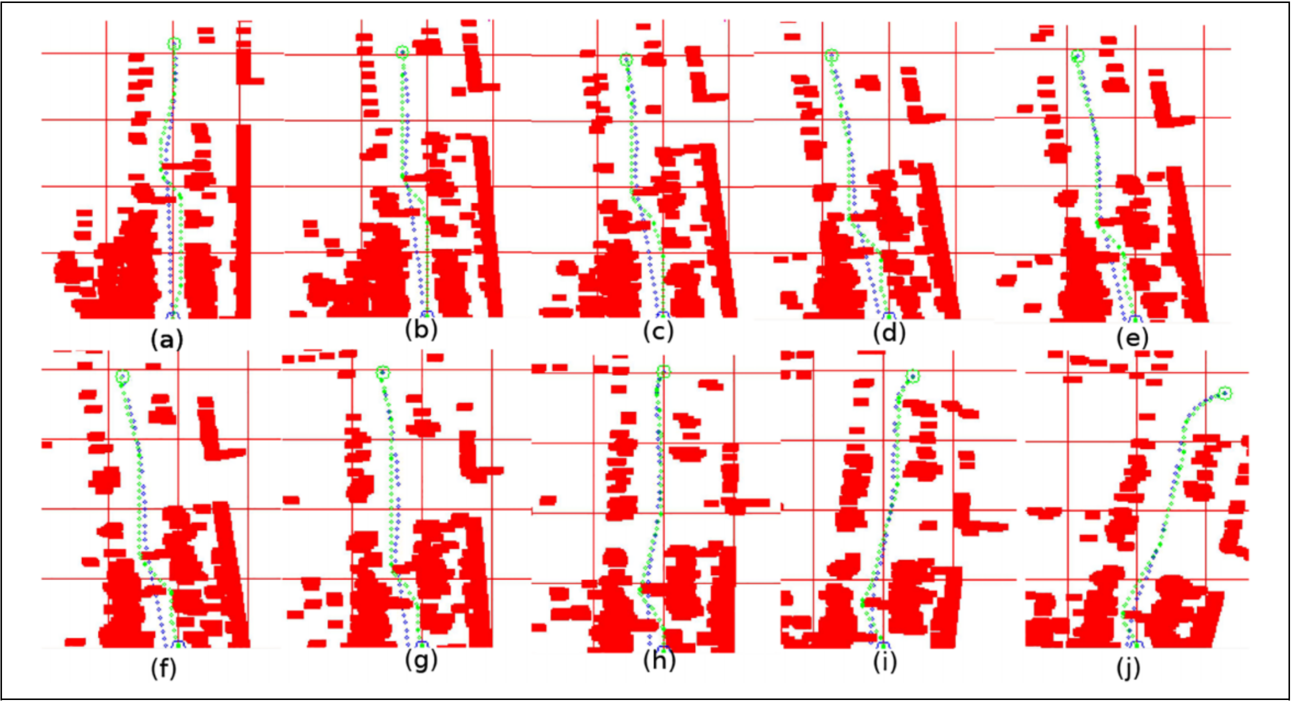

Figure 12. (a–j) A series of planning results to show the process of avoiding obstacles.

The results of both algorithms are also shown in Figure 11. Figure 11(a) shows the scene and trajectories of both algorithms, in which the blue trajectory is the guideline, the pink and green trajectories are generated by the compared algorithm, the red and cyan trajectories are generated by our algorithm. Two groups of obstacles are placed in the middle of the road to verify the avoiding ability of the planning algorithm, as shown in Figure 11(b). In this experiment, both algorithms are applied two times, and their results are listed by different colors in Figure 11(a). In these trajectories, the pink one, which is generated by the compared algorithm, is broken off at position $P1$. It was found that the compared algorithm would take a shortcut from area $A1$, which is marked in Figure 11(a). Figure 11(c) is the corresponding place of area $A1$, and Figure 11(e) shows the planning result that generated by the compared algorithm at position $P1$. It is not the traditional A-Star's fault since the perception not checks out the roadside correctly. But in a real application, these cases are very common that edges beside the road are not correctly detected. Thus, the traditional A-Star algorithm is not suitable under these cases. Under the condition that some obstacle was placed at area $A1$, the compared algorithm generated a new result, as shown in the green trajectory. From this experiment, we can see that the proposed algorithm in this article would deal with the problem that edges beside the road are not correctly detected, and it is better to show driver's purpose than the compared one (The blue trajectory and the cyan trajectory are much closer than the green one to the guideline). It is also found that the traditional A-Star algorithm has higher requirements for roadside detection than the proposed one.

In this designed scene, two groups of obstacles are specially placed in the middle of the lane, as shown in Figure 11(b). A series of planning results for avoiding these obstacles generated by our algorithm is shown in Figure 12(a) to (j). In Figure 12, the blue trajectory means the guideline, and the green trajectory is the planning result, and the red block means obstacles. Figure 12(a) to (j) shows the process that the vehicle passed obstacles. It is obvious that our algorithm is well done to avoid these obstacles.

Long-distance application in real environments

To verify the stable and reliable of the proposed algorithm, long-distance applications in structure environments are carried out. Lots of open scenes are employed, such as streets, campuses, villages, residential quarters, and so on. Among these scenes, pedestrians and moving vehicles are common, which would test the performance of our algorithm. Two typical results are shown in Figures 13 and 14. Figure 13 is a typical campus, and its total mileage is more than 5 km. In Figure 13, the blue line is the guideline, and the black one means obstacles. The ALV platform drives twice under the proposed algorithm, and their trajectories are shown as a red line and green line. Figure 14 is a typical industrial park, and its total mileage is

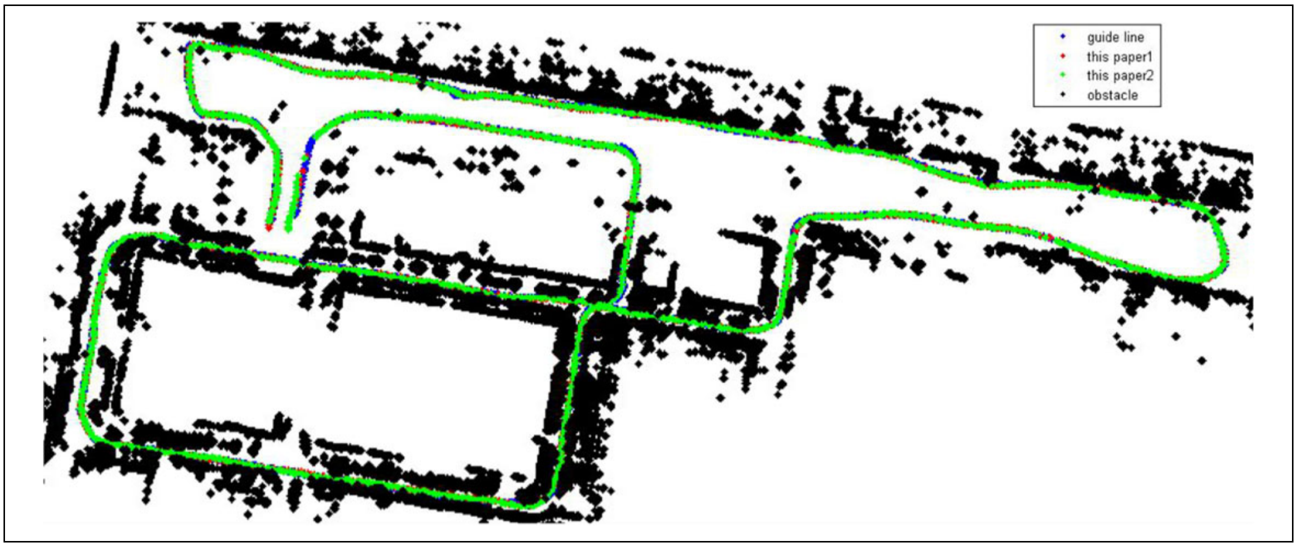

Figure 13. Long-distance application in street scene I.

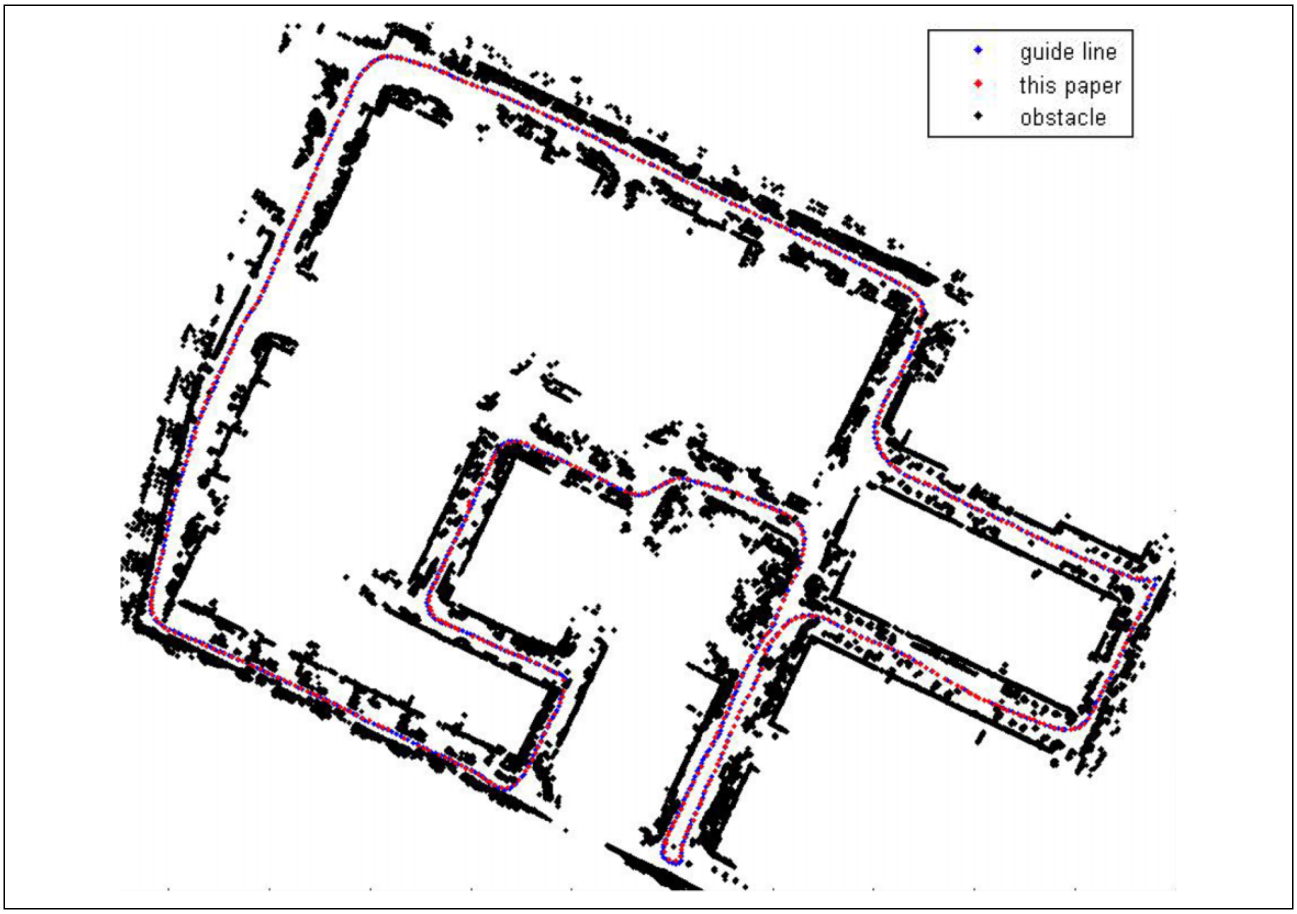

Figure 14. Long-distance application in street scene II.

more than 2 km. In Figure 14, the guideline is also shown as the blue one, and the real trajectory is shown as the red one. From those experiments, it is obvious that our algorithm is well done, stability, and reliability.

Conclusion

This article presents an improved A-Star based path planning algorithm for ALVs. On-road motion planning for ALVs is a challenging problem, and the traditional

A-Star algorithm is insufficient when road edge is not correctly detected, especially in turning a corner. Based on the traditional A-Star algorithm, four improvements are presented in this article. To measure the performance of algorithms and to choose the most appropriate parameter, an evaluation standard is first introduced. Then, a guideline-based A-Star algorithm is presented, in which the guideline is employed to develop the heuristic function to overcome the shortcoming of the traditional A-Star algorithm. Further, for improving the avoidance performance, the idea of key points besides obstacles is employed, which would guide the planning path to avoid the obstacle much earlier and more effective. In addition, a novel variable-step based A-Star algorithm is presented to reduce the computing time of the proposed algorithm. By combination of these improvements, this improved A-Star based path planning algorithm is well done by lots of experiments. Experimental results show that the proposed algorithm is robust and stable. Compared with the state-of-the-art techniques, the performance is better and more suitable in the real application.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by the National Natural Science Foundation of China under grant nos. 61803380 and 61790565.

ORCID iD

Shang Erke https://orcid.org/0000-0002-6669-3933

References

1. Shang E, An X, Wu T, et al. Lidar based negative obstacle detection for field autonomous land vehicles. *J Field Robot* 2016; 33(5): 591–C617.
2. Kim J, Jo K, Lim W, et al. A probabilistic optimization approach for motion planning of autonomous vehicles. *Proc Instit Mech Eng D J Automob Eng* 2018; 232(5): 632–650.
3. González D, Pérez J, Milanés V, et al. A review of motion planning techniques for automated vehicles. *IEEE Trans Intell Transp Syst* 2015; 17(4): 1135–1145.
4. Montemerlo M, Becker J, Bhat S, et al. Junior: The Stanford entry in the urban challenge. *J Field Robot* 2008; 25(9): 569–597.
5. Dolgov D, Thrun S, Montemerlo M, et al. Path planning for autonomous vehicles in unknown semi-structured environments. *Int J Robot Res* 2010; 29(5): 485–501.
6. Brezak M and Petrović I. Real-time approximation of clothoids with bounded error for path planning applications. *IEEE Trans Robot* 2013; 30(2): 507–515.
7. Gu T and Dolan JM. On-road motion planning for autonomous vehicles. In: *International conference on intelligent robotics and applications*. Montreal, QC, Canada, 3–5 October 2012, pp. 588–597. Berlin: Springer.
8. Kammel S, Ziegler J, Pitzer B, et al. Team AnnieWAY's autonomous system for the 2007 Darpa Urban Challenge. *J Field Robot* 2008; 25(9): 615–639.
9. Duchoň F, Babinec A, Kajan M, et al. Path planning with modified a star algorithm for a mobile robot. *Proc Eng* 2014; 96: 59–69.
10. Hart PE, Nilsson NJ, and Raphael B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans Syst Sci Cybern* 1968; 4(2): 100–107.
11. Ferguson D, Howard TM, and Likhachev M. Motion planning in urban environments. *J Field Robot* 2008; 25(11–12): 939–960.
12. Autoware: open-source software for urban autonomous driving, 2018-2-14. https://github.com/CPFL/Autoware/wiki.
13. LaValle SM and Kuffner JJ Jr. Randomized kinodynamic planning. *Int J Robot Res* 2001; 20(5): 378–400.
14. Karaman S, Walter MR, Perez A, et al. Anytime motion planning using the RRT. In: *2011 IEEE international conference on robotics and automation*. Shanghai, China, 9–13 May 2011, IEEE, pp. 1478–1483.
15. Chebly A. *Trajectory planning and tracking for autonomous vehicles navigation*. PhD Thesis, Université de Technologie de Compiègne, 2017.
16. Kanayama YJ and Hartman BI. Smooth local planning for autonomous vehicles. *Int J Robot Res* 1999; 16(3): 273–284.
17. Delingette H, Hebert M, and Ikeuchi K. Trajectory generation with curvature constraint based on energy minimization. In: *IEEE/RSJ international conference on intelligent robots and systems*. Osaka, Japan, 3–5 November 1991, IEEE, pp. 123–128.
18. Lau B, Sprunk C, and Burgard W. Kinodynamic motion planning for mobile robots using splines. In: *IEEE/RSJ international conference on intelligent robot and systems*. St. Louis, MO, USA, 10–15 October 2009, IEEE, pp. 2427–2433.
19. Kato S, Tokunaga S, Maruyama Y, et al. Autoware on board: enabling autonomous vehicles with embedded systems. In: *2018 ACM/IEEE 9th international conference on cyber-physical systems (ICCPS)*. Porto, Portugal, 1–13 April 2018, IEEE, pp. 287–296.
20. Li B, Du H, Li W, et al. Dynamically integrated spatiotemporal-based trajectory planning and control for autonomous vehicles. *IET Intell Transp Syst* 2018; 12(10): 1271–1282.
21. Liu Y and Cui D. Path tracking control for inverse vehicle handling dynamics. *Int J Veh Safe* 2019; 11(2): 120.