Mathematical Biosciences and Engineering

MBE, 18(1): 1–21.
DOI: 10.3934/mbe.2021001
Received: 09 September 2020
Accepted: 06 November 2020
Published: 17 November 2020

http://www.aimspress.com/journal/MBE

Research article

Application improvement of A* algorithm in intelligent vehicle trajectory planning

Xiaoyong Xiong, Haitao Min, Yuanbin Yu* and Pengyu Wang

State Key Laboratory of Automotive Simulation and Control, Jilin University, No. 5988, Renmin Street, Changchun, Jilin 130022, China

* **Correspondence:** Email: yyb@jlu.edu.cn.

Abstract: Trajectory planning is one of the key technologies for autonomous driving. A* algorithm is a classical trajectory planning algorithm that has good results in the field of robot path planning. However, there are still some practical problems to be solved when the algorithm is applied to vehicles, such as the algorithm fails to consider the vehicle contours, the planned path is not smooth, and it lacks speed planning. In order to solve these problems, this paper proposes a path processing method and a path tracking method for the A* algorithm. First, the method of configuring safe redundancy space is given considering the vehicle contour, then, the path is generated based on A* algorithm and smoothed using Bessel curve, and the speed is planned based on the curvature of the path. The trajectory tracking algorithm in this paper is based on an expert system and pure tracking theory. In terms of speed tracking, an expert system for the acceleration characteristics of the vehicle is constructed and used as a priori information for speed control, and good results are obtained. In terms of path tracking, the required steering wheel angle is calculated based on pure tracking theory, and the influence factor of speed on steering is obtained from test data, based on which the steering wheel angle is corrected and the accuracy of path tracking is improved. In addition, this paper proposes a target point selection method for the pure tracking algorithm to improve the stability of vehicle directional control. Finally, a simulation analysis of the proposed method is performed. The results show that the method can improve the applicability of the A* algorithm in automated vehicle planning.

Keywords: intelligent vehicle; trajectory planning; motion control; A-star; pure pursuit

1. Introduction

The development of autonomous driving technology can gradually liberate drivers from driving behaviors, reduce traffic accidents caused by improper driving behaviors, thereby improve road traffic efficiency and ensure traffic safety. The progress of artificial intelligence technology in recent years has made the application of autonomous driving technology in practical scenarios possible. The key issues of autonomous driving vehicle lie in three aspects: location and perception, decision-making and planning and control and execution. Among them, trajectory planning determines the expected trajectory, which is related to the safety, comfort, efficiency of the vehicle. At the same time, trajectory planning has been a research challenge due to the complexity of the traffic environment and vehicle systems.

Currently, research on trajectory planning has focused on grid search-based algorithm [1], random sampling-based planning algorithm [2], artificial potential field method [3], genetic algorithm [4] and other artificial intelligence planning algorithms.

Grid search-based planning algorithms have been widely used in path planning in recent years due to their simple principle and parsimony optimality. The most typical search algorithms are Dijkstra search algorithm [5], A* algorithm [6], D* algorithm [7]. These algorithms require pre-processing of perceptual information to generate a search map containing obstacles and free areas, and then apply a graph search algorithm to find the shortest path [8]. The A* algorithm combines the idea of mathematical search and heuristic search, which not only improves the efficiency of the algorithm, but also ensures the optimality of the results [9]. Therefore, it is widely used in path planning.

Traditional A* algorithms are usually applied to robot path planning. Most researchers are concerned with the real-time performance of the algorithm. The WA* (Weighted A*) algorithm is a variant of the A* algorithm [10]. By assigning a variable weight to the heuristic term of the evaluation function of the original A* algorithm, it enhances the influence of heuristic information and improves the computational efficiency. The A*-connect algorithm speeds up the search process by bidirectional search [11].

At the same time, the optimality of the A* algorithm is also a concern for researchers. The ARA* (Anytime Repairing A*) algorithm can improve the optimality of planning results as much as possible within a given time constraint [12]. MHA* (Multi-Heuristic A*) algorithm avoids falling into local optimal results by setting multiple heuristic functions [13]. DMHA*(Dynamic Multi-Heuristic A*) algorithm ensures the global optimality of the algorithm by setting up a jump-out mechanism to make the algorithm jump out of the locally optimal results [14].

In addition, the application of the A* algorithm in dynamic environment is also a problem to be solved. The LPA* (Lifelong Planning A*) algorithm applies the incremental search idea to A* algorithm [15], which can reuse the previous search results in path re-planning, thus improve the computational efficiency of the algorithm in dynamic environment. The D* algorithm can avoid large-scale re-planning after map changes through reverse search, so as to improve efficiency [16].

The aforementioned researchers are more concerned with the path optimization and efficiency of the A* algorithm. However, if the algorithm is applied to the path planning of automobiles, the smoothness of the path cannot be ignored. Some researchers use the method of increasing search dimension or grid search scope to smooth the path. For example, study [17] added heading angle state, which changed the state space from two-dimensional space to three-dimensional space. Some

researchers have increased the search scope from 8 neighborhoods to 24 neighborhoods [18]. These methods can smooth the path to some extent, but are prone to increase the computational effort.

Another important issue is that the contours of the vehicle must be taken into account during path planning. Existing literature usually uses larger grid sizes to avoid this problem, but this affects planning accuracy. In addition, speed planning is also indispensable. Existing literature usually deals with speed planning on structured paths [19,20], while there is little literature related to speed planning in unstructured environments.

Trajectory tracking also has an important impact on the application of the A* algorithm. MPC [21] and pure pursuit algorithm [22] are commonly used algorithms for vehicle trajectory tracking. MPC can achieve good tracking results, but often requires a large amount of computational resources. The pure pursuit algorithm has better real-time performance, but the tracking error is larger when the curvature changes. The selection of the target point in the pure pursuit algorithm has a great impact on the tracking effect, but few papers describe this in detail.

Therefore, a set of map pre-processing and path post-processing methods are proposed to address the shortcomings of the A* algorithm in vehicle trajectory planning, and a target point selection method for pure pursuit algorithms is proposed to improve the usefulness of the A* algorithm in vehicles. The main contributions of this paper are as follows:

(1) A set of processing methods is proposed to address the safety, path smoothing and speed planning problems of the A* algorithm in the practical application of vehicle trajectory planning. The redundancy space of the raster map is set up to avoid vehicle collisions. The path planning is done based on the A* algorithm, then the path is smoothed by Bezier curve and the velocity planning is done based on the curvature of the path. The novelty of the proposed process is that multiple problems are considered and solved simultaneously in a continuous and complete set of processes. These problems are not apparent when the algorithm is applied to a robot, but are crucial when it is applied to a vehicle.

(2) A feedforward control method based on an expert system is constructed for speed control, and a pure pursuit algorithm is used for steering control. The target point selection method of pure pursuit algorithm is proposed, which is rare in the previous literature.

The sections of the paper are organized as follows. Section 2 describes the whole architecture of automatic driving algorithm in this work, then proposes a set of trajectory planning and processing process based on A* algorithm. Section 3 puts forward a trajectory tracking algorithm based on expert system and pure pursuit algorithm. Section 4 gives the simulation results and analysis of the algorithm, which is followed by the conclusions in section 5.

2. Trajectory planning and processing

A* algorithm is usually used in unstructured environment where obstacles are relatively cluttered and vehicle speeds are low. In order to avoid collision of vehicle contours, redundant spaces are set up based on vehicle size when the map is rasterized. In order to meet the vehicle steering constraints, the path is smoothed using Bessel curves after the initial planning of the A* algorithm. In addition, the planning of vehicle speed is based on curvature.

2.1. Automatic driving system architecture

As shown in Figure 1, we use the "perception, planning, control" automatic driving algorithm architecture in this work. First, the perception system generates a local map by positioning and sensor information. Then, trajectory planning is performed based on the A* algorithm. Finally, the trajectory tracking algorithm is used to calculate and output control signals to the vehicle.

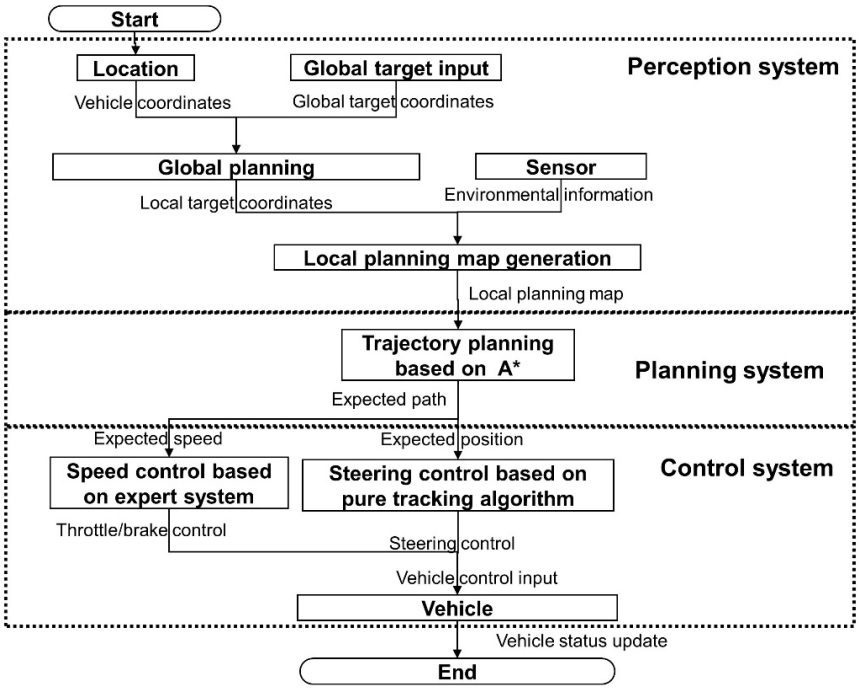


Figure 1. Automatic driving system architecture.

2.2. Principle of A algorithm*

The principle of A* algorithm is shown in Figure 2. The algorithm creates two lists: $openList$ and $closeList$. Where, $openList$ is used to store the grid to be extended and $closeList$ is used to store the processed grid. During initialization, the starting point is put into the $openList$ [23]. In subsequent iterations, the grid with the lowest path estimation cost is selected from the $openlist$ as the current grid cur, and the path estimation cost is calculated by Eq (1).

$$F = G + H \tag{1}$$

G is the actual cost, representing the current lowest cost of expanding from the starting grid to the current grid, and H is the heuristic cost, representing the estimated cost from the current grid to the end grid. The sum of the two is the estimated cost of the entire path. The actual cost G used in this paper refers to the cumulative distance. The cost of vertical and horizontal movement is 10, and the cost of diagonal movement is 14. The heuristic cost H is the Euclidean distance from the current grid to the final target grid.

Next, remove the current grid from the $openList$ and put it into the $closeList$. Traverse eight grids around the current grid. If the searched grid cannot pass or it is already in the $closeList$, skip it.

Otherwise, if it is not in the $openList$, add it to the $openList$ and calculate the actual cost, heuristic cost and estimated cost. If it is already in the $openList$, compare the actual cost of the grid with the current grid as the parent node with the original one, and keep the one with less value as the new parent node of the grid. When the end grid appears in the $openList$, it means that the search is completed. The planned path can be output by taking the parent node from the current grid.

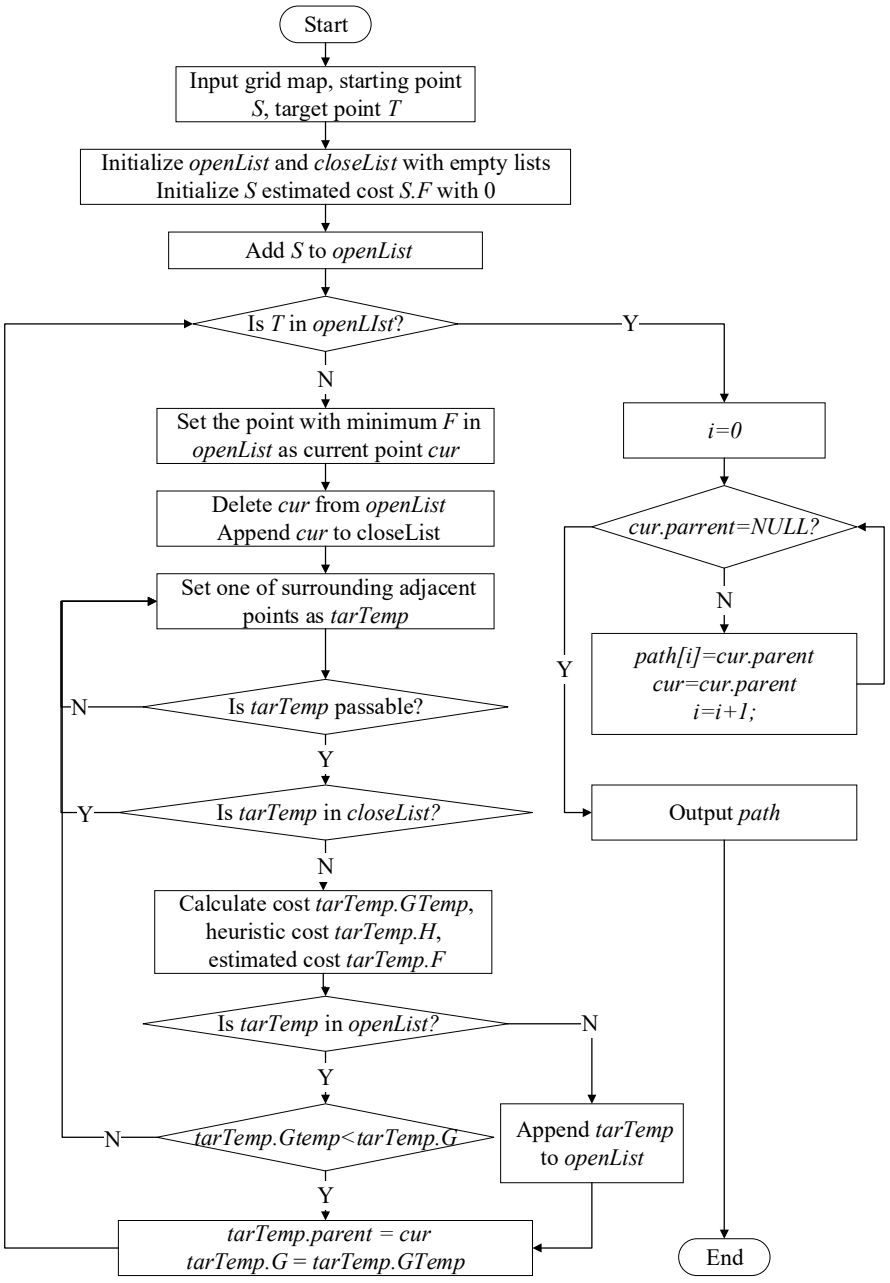


Figure 2. A* algorithm flowchart.

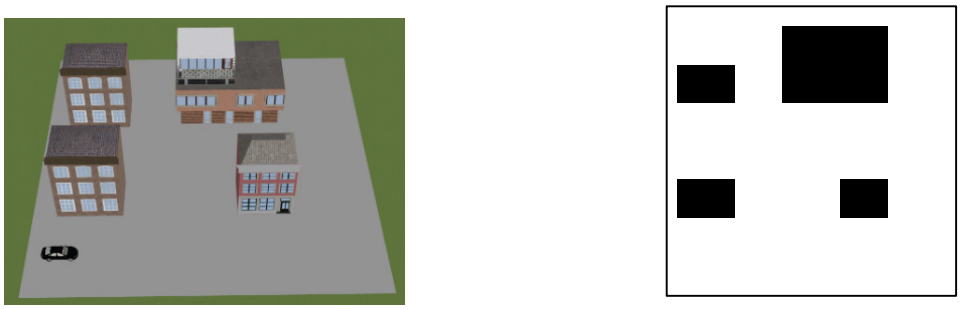
2.3. Gridding of map

The information output from the vehicle perception system is usually based on Cartesian

coordinate system. However, the A* algorithm needs to be run on a grid map, so the map must be gridded as shown in Figure 3. The grid map is actually a two-dimensional matrix. The matrix index indicates the position, and the matrix element value indicates the presence of obstacles. The correspondence between the position of the grid matrix elements and the coordinates in the local map is shown in Eq (2).

$$\begin{cases} n = \dfrac{x}{p} \\ m = \dfrac{y_{max} - y}{p} \end{cases} \tag{2}$$

Where, n is the column index of elements in the grid matrix, m is the row index of elements, p is the precision of the grid map, which is the actual distance indicated by side length of unit grid, (x, y) is the coordinate in the actual map, and y_{max} represents the maximum vertical coordinate of the local map. Since p and y_{max} will affect the accuracy and required computing resources, in order to balance accuracy and real-time performance, we set p to 0.25 m, and y_{max} to 50 m.



(a) (b)

Figure 3. Gridding of map: (a) Actual scenario; (b) Grid map.

Considering the vehicle profile, some redundant space must be reserved near obstacles. The redundant space around obstacles can leave some space between the planned path and obstacles, and ensure the safety when the vehicle tracks the path directly. The specific method of setting up redundant crash spaces on a grid map includes the following steps:

(1) Determine the maximum distance between vehicle tracking center and the outer contour of the vehicle D_s. This paper uses Audi a8 model, and the D_s is set to 2.5 m.

(2) Calculate the minimum number of grids to be retained based on the actual distance p represented by a single grid edge length as shown in Eq (3).

$$n_{ex} = Ceiling\left(\frac{D_s}{p}\right) \tag{3}$$

Where, $Ceiling()$ is the up-rounding function.

(3) During the binarization of the grid map, the n_{ex} grids in the up, down, left and right of obstacles are assigned as non-passable values. In this paper, n_{ex} is calculated to be 10.

After the above steps, Figure 3(b) can be further represented as a grid map with redundant spaces as shown in Figure 4.

The direct output path of A* algorithm is expressed in the form of matrix indexes, so it is necessary to convert the index representation of path in grid map to the coordinate representation of rectangular coordinate. The conversion formula is shown in Eq (4).

$$\begin{cases} x = np \\ y = y_{max} - mp \end{cases} \tag{4}$$

Take the vehicle position as the starting point and the destination as the end point to run A* algorithm for path search, and the path can be obtained as shown in Figure 5.

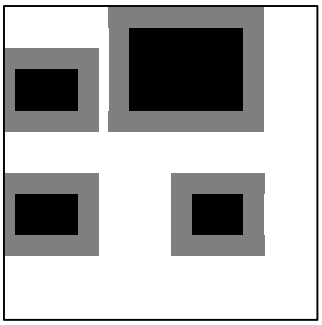


Figure 4. Grid map with redundant space.

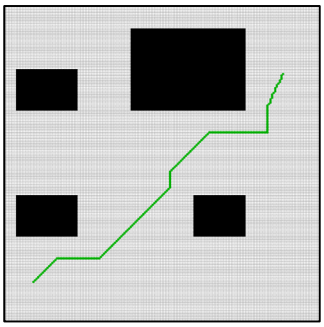


Figure 5. Planning results.

2.4. Path smoothing

It can be seen from Figure 5 that there are many turning points in the path directly output by A* algorithm, which is not conducive to the stable following of the vehicle. Therefore, further smoothing is needed. We use Bezier curve to smooth the path in this work.

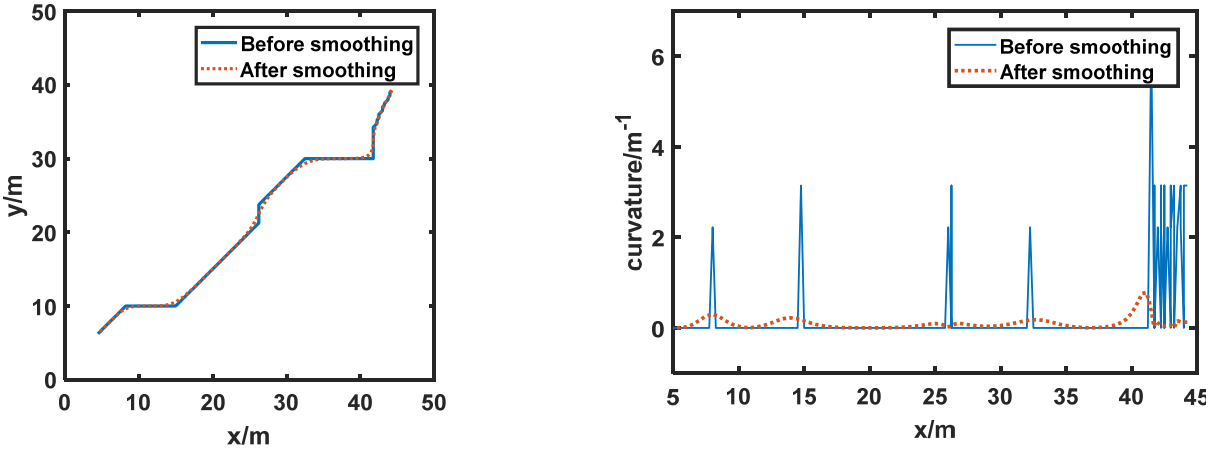


Figure 6. Smoothing results: (a) Path; (b) Curvature.

Bezier curve is widely used in two-dimensional graphic design because of its simple control and smooth curve [24]. Figure 6(a) shows planned path before and after smoothing, Figure 6(b) shows curvature of the path before and after smoothing. The curvature is calculated by the curvature

formula as shown in Eq (5).

$$\kappa = \frac{\Delta\alpha}{\Delta s} \tag{5}$$

Where κ is the curvature, $\Delta\alpha$ is the tangent direction difference of the sampling points, and Δs is the arc length of the sampling points.

As can be seen in the figure, the curvature of the smoothed path and its variation is greatly reduced, which facilitates the smooth tracking of the vehicle.

2.5. Speed planning

The curvature of the planned path is highly variable and has significant constraints on the speed. Therefore, we plan the velocity based on the average curvature. The average curvature of the 10 points before and after the current path point is taken as the influencing factor. As shown in Eq (6), when the curvature is zero, the instantaneous velocity $\tilde{v}_i$ is $20km/h$, which decreases with the increase of curvature. The final output velocity v_i is the average speed within 10 points before and after the current point as shown in Eq (7).

$$\tilde{v}_i = 20 - 30 \cdot \frac{1}{21} \sum_{j=i-10}^{i+10} \kappa_j \tag{6}$$

Where, $\tilde{v}_i$ is the instantaneous value of velocity of the ith point, κ_j is the curvature of the jth point.

$$v_i = \frac{1}{21} \sum_{j=i-10}^{i+10} \tilde{v}_j \tag{7}$$

Using the above method to plan the velocity of the path in Figure 6(a), the planning results are shown in Figure 7. It can be seen from the figure that the speed is relatively smooth and the planned speed can be reduced in time when the curvature is large. The results show that the planned velocity is conducive to the safe driving of vehicles.

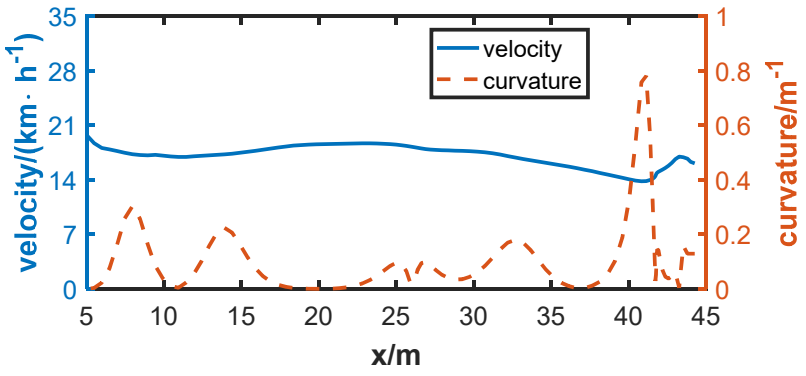
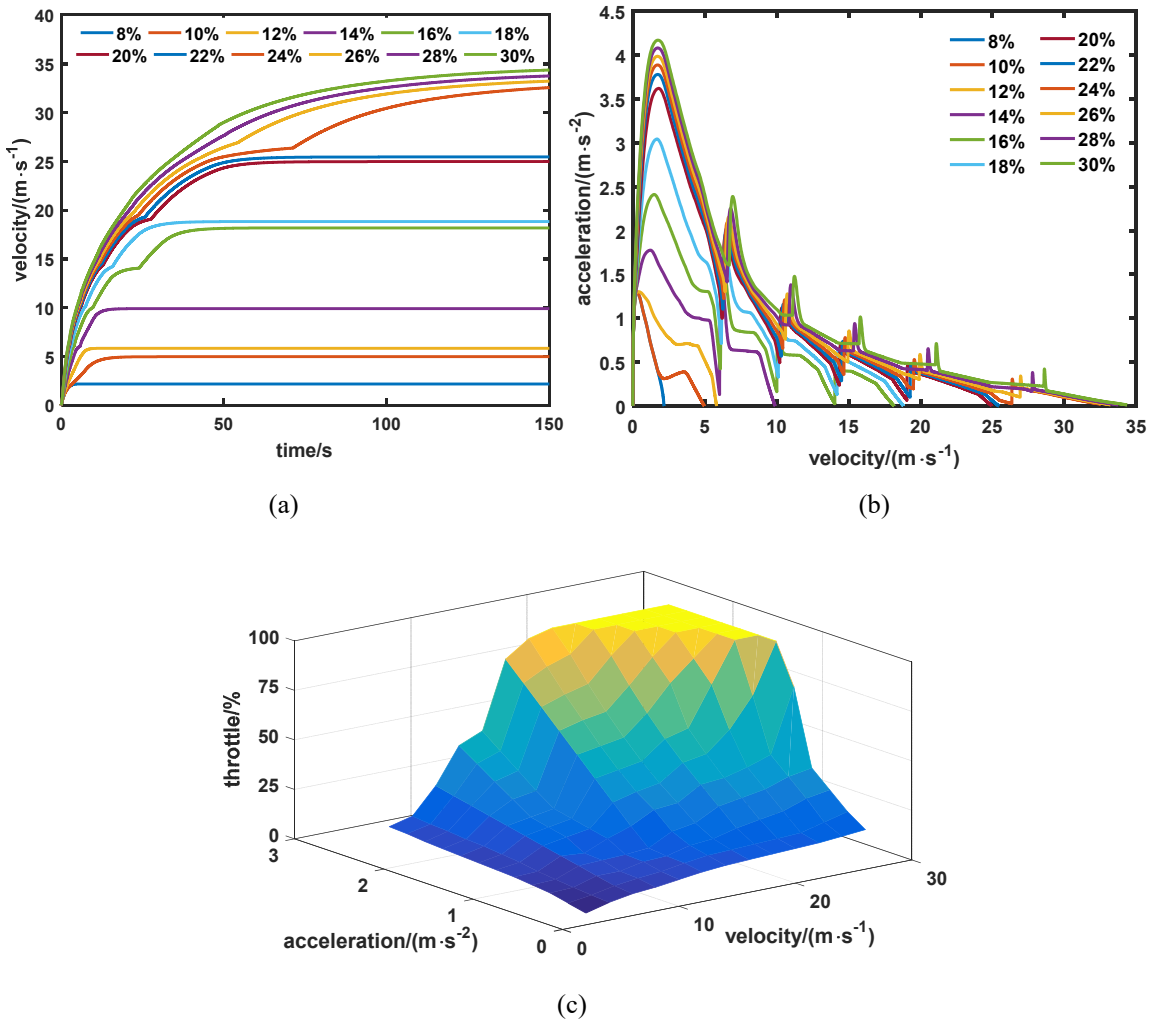


Figure 7. Speed planning results.

3. Trajectory tracking algorithm

The planned path only works through precise control. Therefore, a speed control method based on expert system and a steering control method based on the pure pursuit algorithm are proposed. The acceleration and steering characteristics of the vehicle are considered. At the same time, the target point selection method is proposed.

3.1. Speed tracking algorithm based on expert system



(a) (b)

(c)

Figure 8. Vehicle acceleration characteristics: (a) Velocity curve; (b) Velocity-acceleration curve; (c) Velocity-acceleration-throttle map.

The acceleration characteristics of different vehicles are different, so the acceleration characteristics of the vehicle need to be tested. The current speed, desired speed and desired acceleration of the vehicle are taken as inputs to the speed control system. Several equal gradient fixed throttle acceleration tests are performed, the vehicle speed is recorded and the acceleration is calculated to obtain the acceleration curve of the vehicle at each throttle, as shown in Figure 8. Based on the results of Figure 8(a),(b), the map of velocity-acceleration-throttle as shown in Figure 8(c) can

be obtained.

The actual control process of vehicle speed is shown in the Figure 9. Firstly, calculate the difference between the current speed and the target speed, and judge whether to enter the throttle control mode or the brake control mode according to the difference. When entering the brake control mode, the throttle output is set to zero, and the required brake pedal opening is calculated by PID algorithm. When entering the throttle control mode, the brake pedal opening is set to zero, then calculate the theoretical acceleration demand through the speed error and acceleration time. Next, add the acceleration error value at the current time for correction to get the actual acceleration demand. The throttle opening is obtained by looking-up the map shown in Figure 8 through the current speed and actual acceleration. Finally, the throttle is filtered and output.

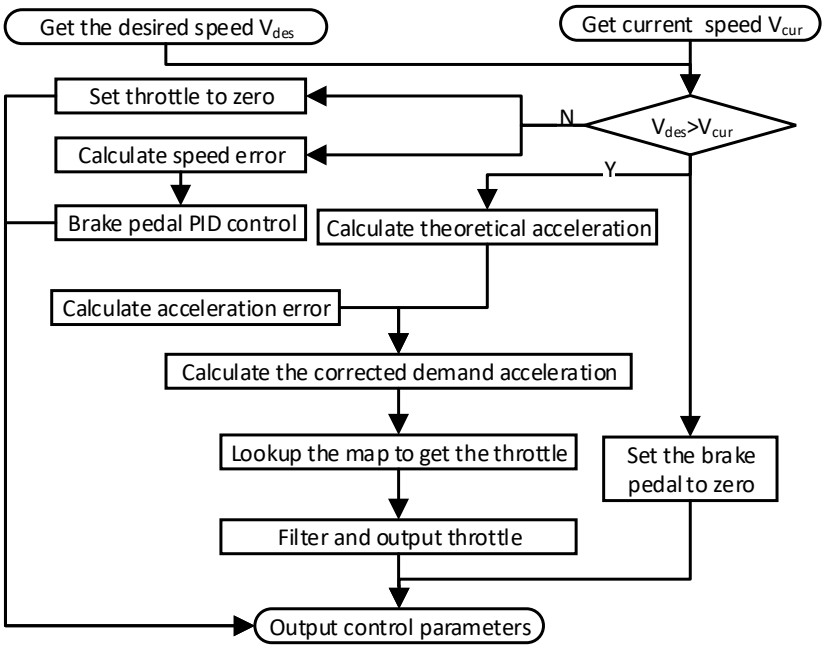


Figure 9. Speed control process.

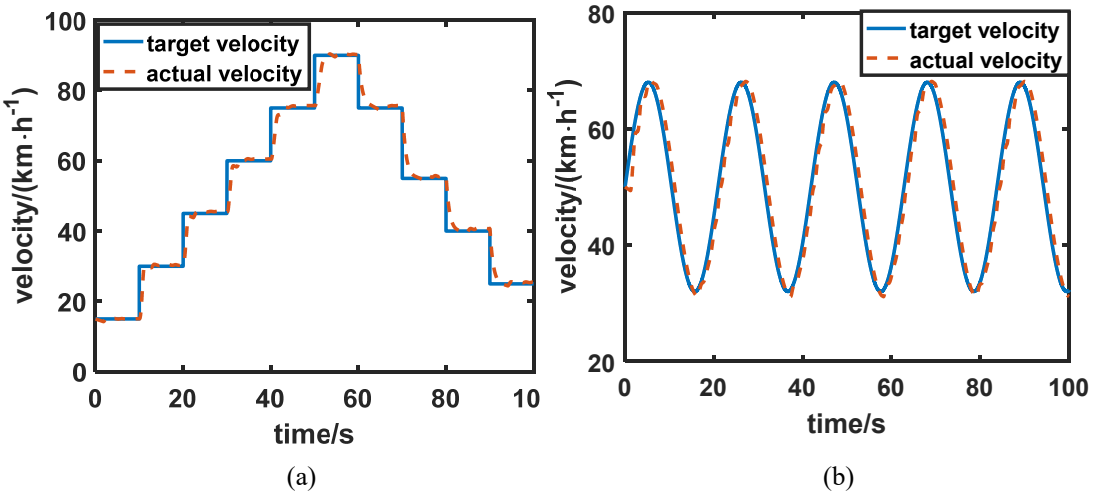


Figure 10. Speed control result: (a) Step target speed; (b) Sine wave target speed.

The velocity control results for step and sinusoidal input conditions are shown in Figure 10. It can be seen that the algorithm is able to track the target velocity quickly and steadily.

3.2. Path tracking algorithm based on pure pursuit algorithm

Pure pursuit algorithm is a path following algorithm. A point on the path is selected as the target point, and the currently required control input is computed based on the position of the target point. At the same time, with the iteration of the algorithm, the target point moves forward on the expected trajectory, making the vehicle approach the expected trajectory with a smooth trace [25].

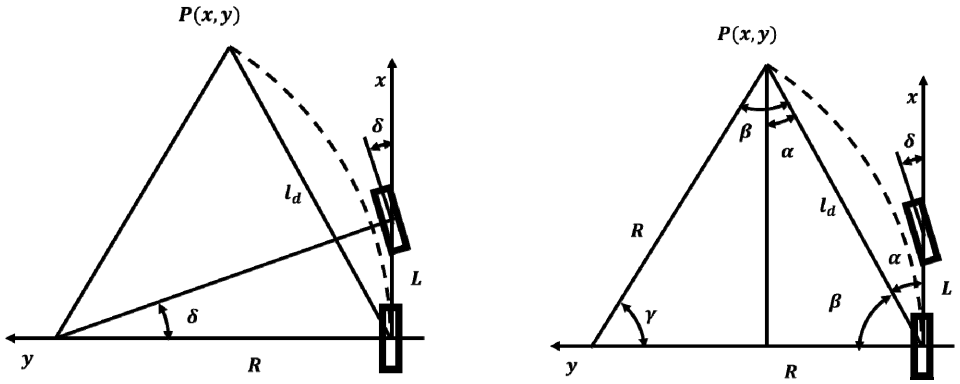


Figure 11. Principle of pure pursuit algorithm: (a) Vehicle steering geometry; (b) Pure pursuit algorithmic geometry.

Figure 11 shows the principle of pure pursuit algorithm. In the vehicle coordinate system, $P(x, y)$ is the target point, l_d is the distance between the target point and the rear wheel center of the vehicle, δ is the front wheel angle, L is the vehicle wheelbase, R is the expected path curvature radius, α is the azimuth of the target point relative to the vehicle. Figure 11(a) shows the geometric relationship of vehicle steering, which conforms to the Ackerman steering geometry as shown in Eq (8).

As shown in Figure 11(b), according to the geometric relationship, the Eqs (9)–(11) can be listed.

$$tan\,\delta = \frac{L}{R} \tag{8}$$

$$\alpha + \beta = \frac{\pi}{2} \tag{9}$$

$$\gamma + 2\beta = \pi \tag{10}$$

$$l_d \cdot cos\,\alpha = R \cdot sin\,\gamma \tag{11}$$

By combining Eq (9) with Eq (10), the Eq (12) can be obtained.

$$\gamma = 2\alpha \tag{12}$$

The Eq (13) can be obtained by combining Eq (12) with Eq (11).

$$R = \frac{l_d \cdot cos\,\alpha}{sin\,2\,\alpha} = \frac{l_d}{2\,sin\,\alpha} \tag{13}$$

By combining Eq (13) with Eq. (8), the Eqs (14) and (15) can be obtained.

$$tan\,\delta = \frac{2L\,sin\,\alpha}{l_d} \tag{14}$$

$$\delta = \arctan\frac{2L\,sin\,\alpha}{l_d} \tag{15}$$

Due to the different steering characteristics of vehicles at different speeds, the Eq (16) is finally obtained by adding the speed-related correction factor K in Eq (15).

$$\delta = K \cdot \arctan\frac{2L\,sin\,\alpha}{l_d} \tag{16}$$

The correction factor K in Eq (16) can be obtained through experiments, and finally the $K - v$ fitting curve as shown in Figure 12 can be obtained.

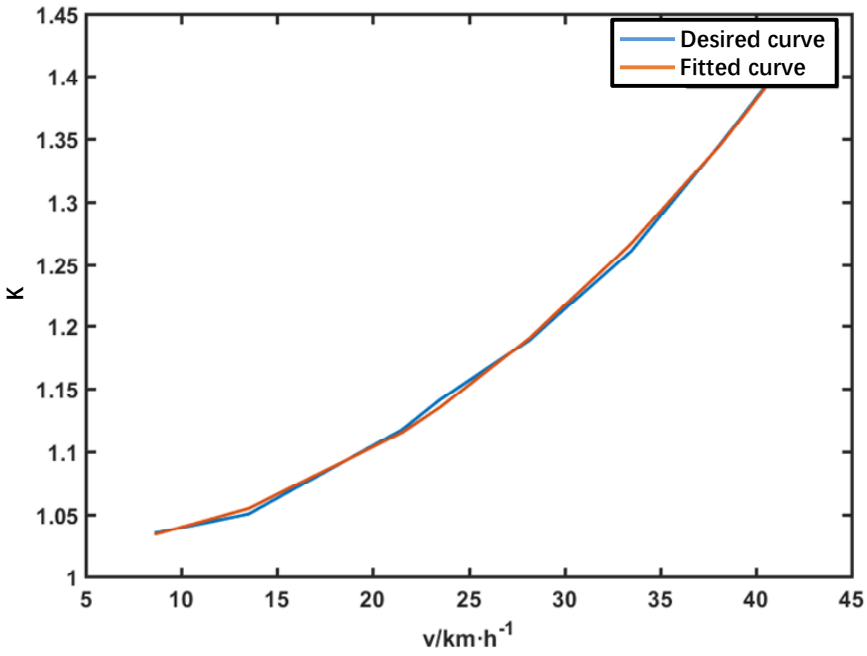


Figure 12. K-v fitting curve.

The steering control process is shown in Figure 13. Firstly, the coordinates of the target point and the current vehicle coordinates are obtained, and then the difference between the target heading angle and the current heading angle of the vehicle is calculated. Next, the distance of the target point and the correction coefficient K are calculated, which are substituted into the formula for calculating

the front wheel angle and multiplied by the correction coefficient to obtain the steering wheel angle.

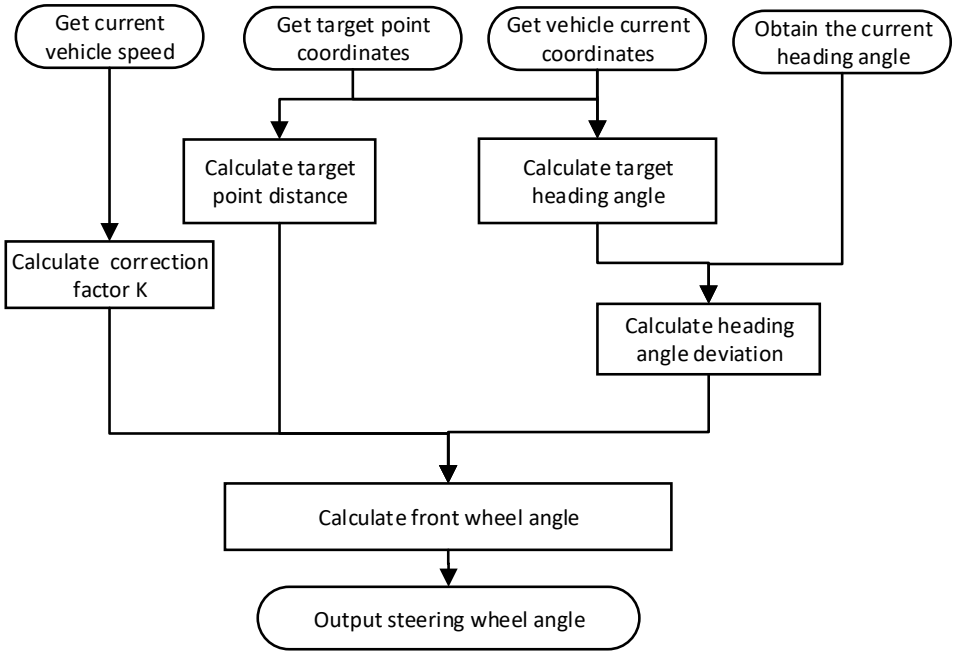


Figure 13. Steering control process.

3.3. *Target point selection*

The selection of target point has a great influence on the tracking accuracy, especially when the path curvature and speed change. Therefore, it is necessary to give a target point selection method. In this paper, the target point distance d_{tar} is used to select target points. Among the preceding path points, the first one whose distance from the vehicle is not less than d_{tar} is selected as the target point.

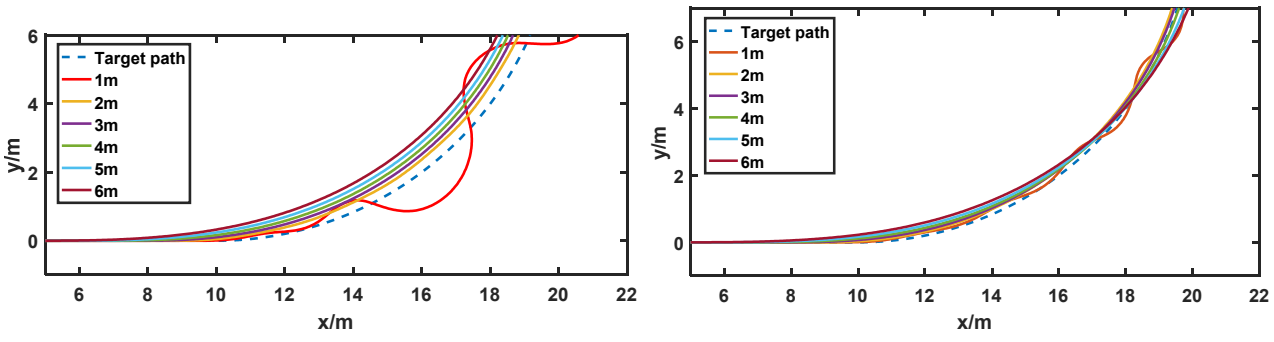


Figure 14. The influence of different correction coefficients on tracking effect when path curvature is 0.1: (a) $\varphi = 1$ (b) $\varphi = 0.6$.

Figure 14 shows the tracking results when the expected path curvature is 0.1, and the target point distance between target points is from 1 m to 6 m. Figure 14(a) shows that the longer the target distance, the easier it is to produce the "shortcut" phenomenon, and the shorter the target distance, the easier it is to cause oscillation. As shown in Eq (17), in order to avoid "short cut" phenomenon, a correction coefficient φ is used to correct original angle δ to get the output angle δ_{out}. When the φ is set to 0.6, the tracking result is obtained as shown in Figure 14(b). Although the "short cut" phenomenon is improved, it also brings the “understeer” phenomenon.

$$\delta_{out} = \varphi \cdot \delta \tag{17}$$

If the algorithm is used to track the expected path of other curvatures, as shown in Figure 15, the results are similar. Also, it can be found that with the increase of curvature, it is not so sensitive to the distance of the target point.

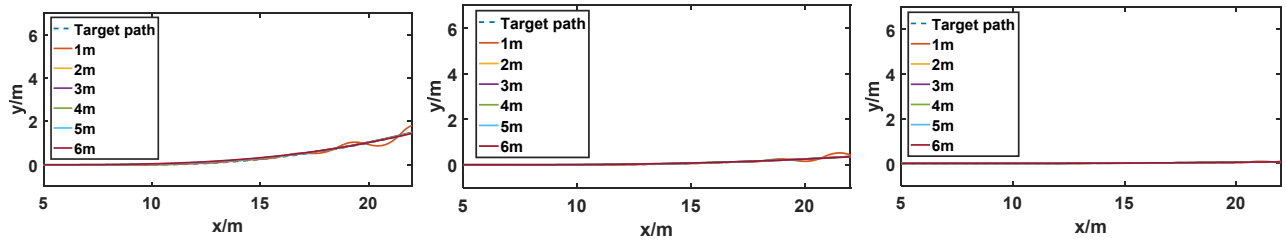


Figure 15. Tracking results under different curvature: (a) curvature is 0.02 m^{-1}; (b) curvature is 0.005 m^{-1}; (c) curvature is 0.001 m^{-1}.

To further quantify the relationship between curvature and target distance, the average error for each condition was calculated and the results are shown in Table 1.

Table 1. Average tracking error under different curvature and different target point distance conditions.

Curvature (m^{-1}) / d_{tar}(m)	0.005	0.01	0.02	0.05	0.1
1	0.012	0.013	0.020	0.041	0.075
2	0.011	0.011	0.016	0.038	0.069
3	0.010	0.010	0.018	0.043	0.074
4	0.009	0.012	0.021	0.049	0.077
5	0.011	0.014	0.026	0.061	0.113
6	0.013	0.018	0.033	0.079	0.153

Obviously, different curvature corresponds to different optimal target point distances, so a variable curvature road was used for further testing. The expected trajectory is shown in Figure 16(a). The curvature of the road is shown in Figure 16(b), where the horizontal axis is the distance from the starting point.

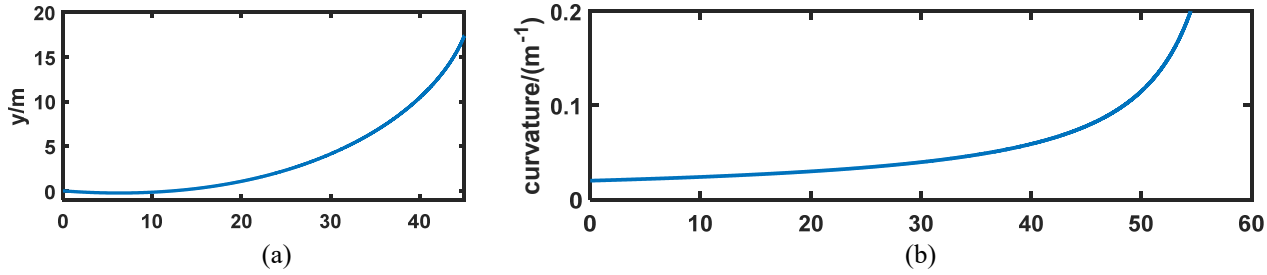


Figure 16. Expected path of changing curvature: (a) path; (b) curvature.

The test results at a speed of 15 km/h are shown in Figure 17. It can be seen that in the first half of the path, the curvature of the path is smaller, and the error is smaller when the target point distance is short. In the second half of the path, the curvature of the path increases, the vibration occurs when the target point distance is short, and the error decreases when the target point distance is long.

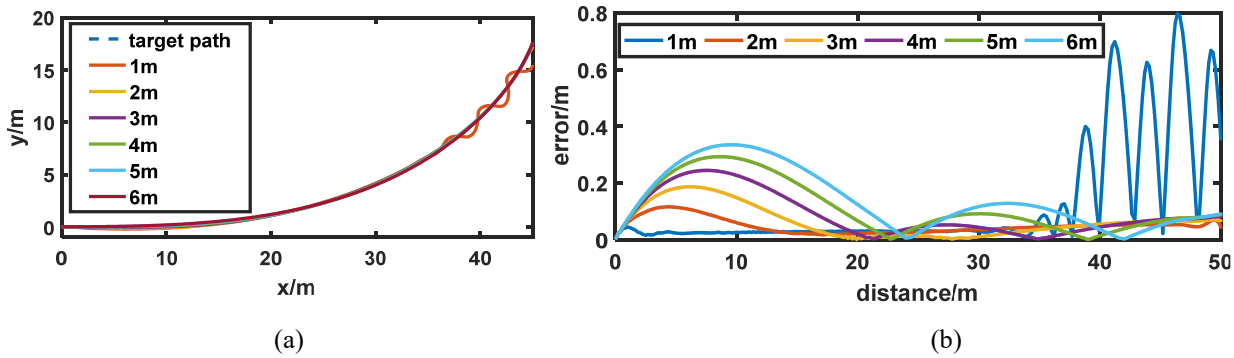


Figure 17. Tracking results of different target point distances at 15 *km/h*: (a) path tracking comparison; (b) path tracking error comparison.

The target point distance is fixed at 6 m and the path tracking test is performed at different speeds. The results are shown in the Figure 18. The higher the speed, the more prone to understeer.

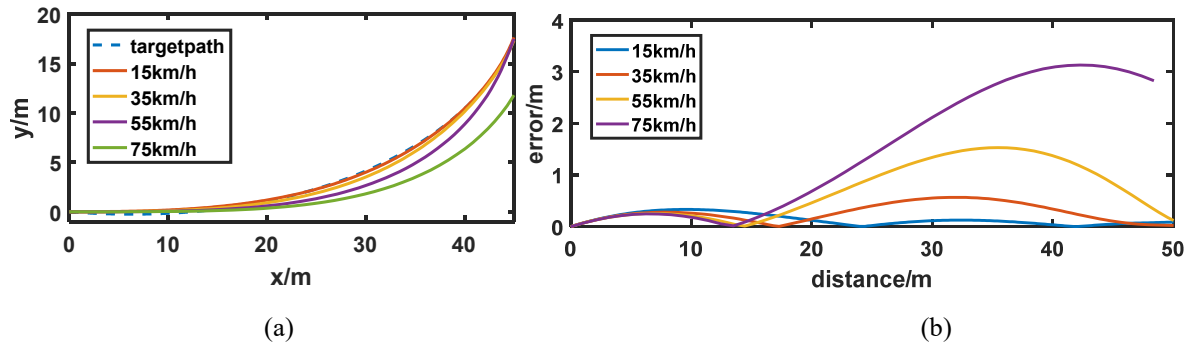


Figure 18. Tracking results of different velocity when the target point distances is 6 *m*: (a) path tracking comparison; (b) path tracking error comparison.

In summary, the greater the curvature, the smaller the optimal target point distance. With the increase of curvature, the influence of target point distance on tracking effect decreases. In addition, the greater the speed, the more prone to understeer, and the larger the target point distance is needed. Therefore, Eq (18) can be used to calculate the target point distance:

$$d_{tar} = \lambda \cdot \sqrt{v} \cdot \ln\left(\frac{1}{\kappa}\right) + \mu \tag{18}$$

Where, d_{tar} is the target point distance; v is the vehicle velocity; κ is the curvature; λ is correction factor; μ is correction item.

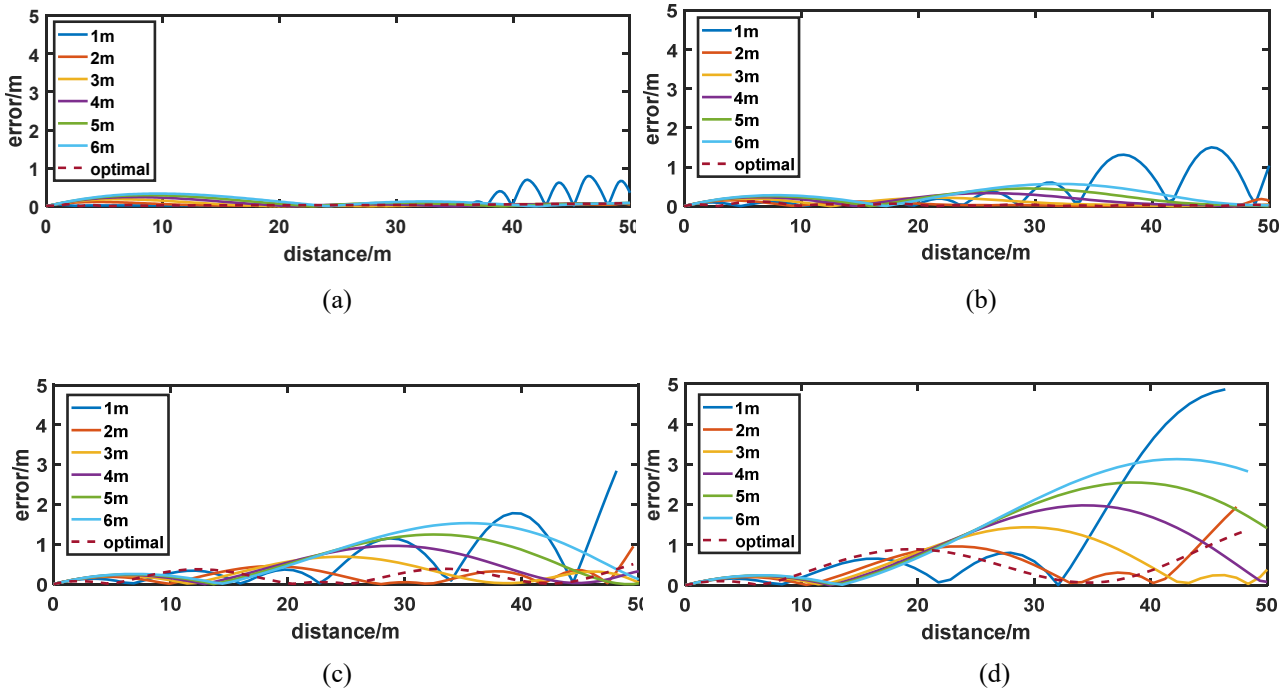


Figure 19. Comparison of tracking error of fixed target point distance and changed target point distance at different speeds: (a) 15 *km/h*; (b) 35 *km/h*; (c) 55 *km/h*; (d) 75 *km/h*.

Table 2. Comparison of average tracking error of fixed target point distance and changed target point distance at different speeds.

Velocity(km/h) / $d_{tar}(m)$	15	35	55	75
1	0.137	0.400	0.616	1.1542
2	0.047	0.053	0.210	0.474
3	0.063	0.094	0.272	0.570
4	0.088	0.148	0.394	0.849
5	0.115	0.204	0.520	1.211
6	0.141	0.260	0.666	1.408
Optimal d_{tar}	0.038	0.032	0.179	0.461

When λ is 0.2 and μ is 0.5, the tracking result of the optimized target point distance using the

Eq (18) is compared with the fixed target point distance, and the results are shown in the Figure 19. The average error comparison is shown in Table 2. The tracking error is lower than that of any fixed target point distance when the target point distance is changed as Eq (18).

4. Simulation and analysis

To test the adaptability of the algorithm, several simulation tests are conducted in different scenarios. The simulation scene software is PreScan, and the planning and control algorithm is written in C++. The CPU of the simulation computer is core i5-9400F. The graphics card is NVIDIA GTX 1660 Ti.

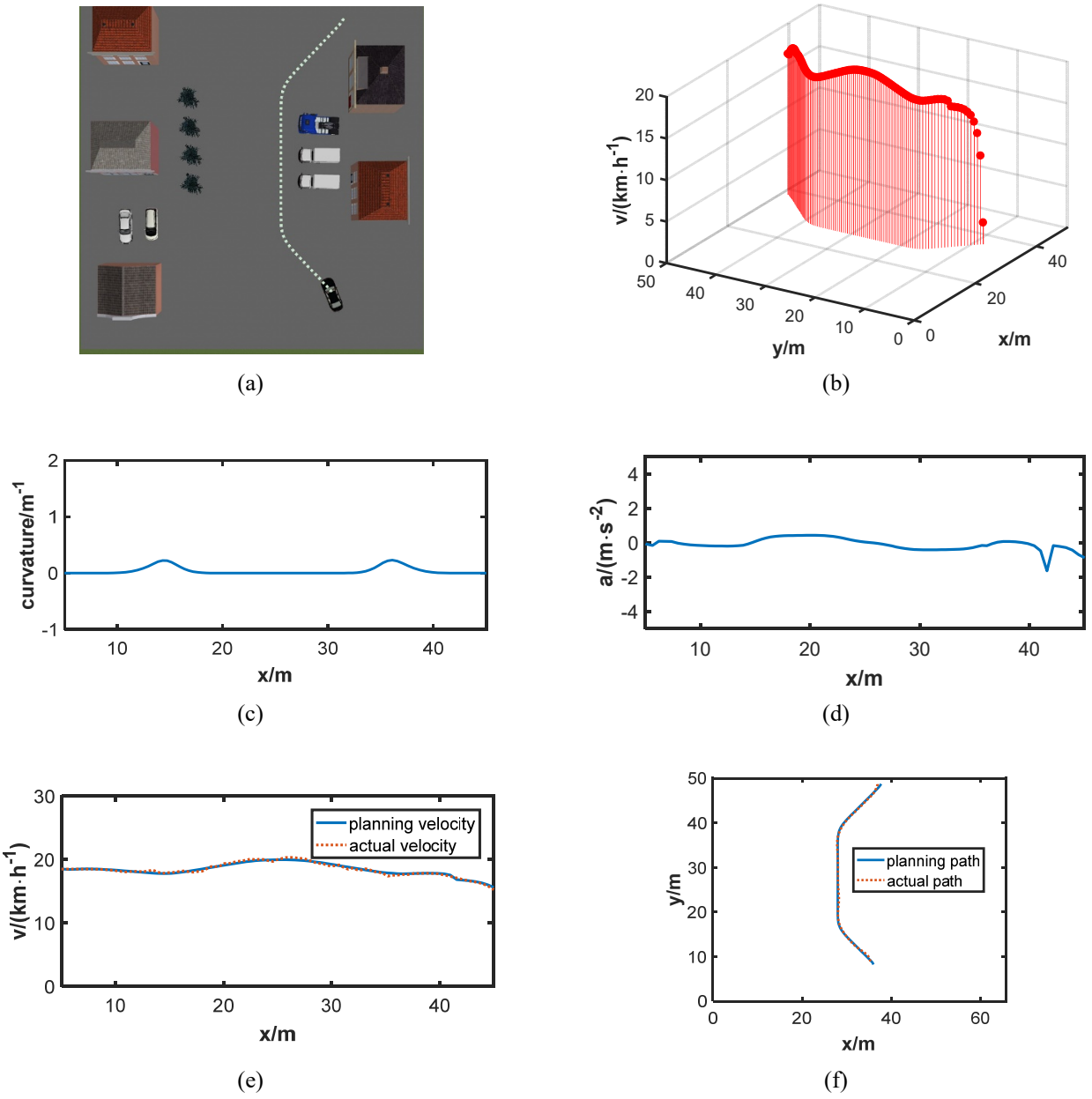


Figure 20. Simulation results of first scenario: (a) Planned path; (b) Velocity of path points; (c) Curvature; (d) Acceleration; (e) Velocity following; (f) Path following.

The simulation results of the first scenario are shown in Figure 20. The vehicle in the lower right corner need to plan its path to the upper right corner. The scene includes houses, cars and other obstacles. On average, it takes 153 ms per cycle for trajectory planning and tracking. It can meet the real-time requirements of vehicle computing platform.

Figure 20(a) shows the planned path. It indicates that the algorithm has worked out a safe path to avoid obstacles, and the overall path is relatively smooth. Figure 20(b) shows the velocity of each path point, which shows that the speed can be reduced at the corner and increased at the straight line. It also shows that both stability and traffic efficiency are guaranteed.

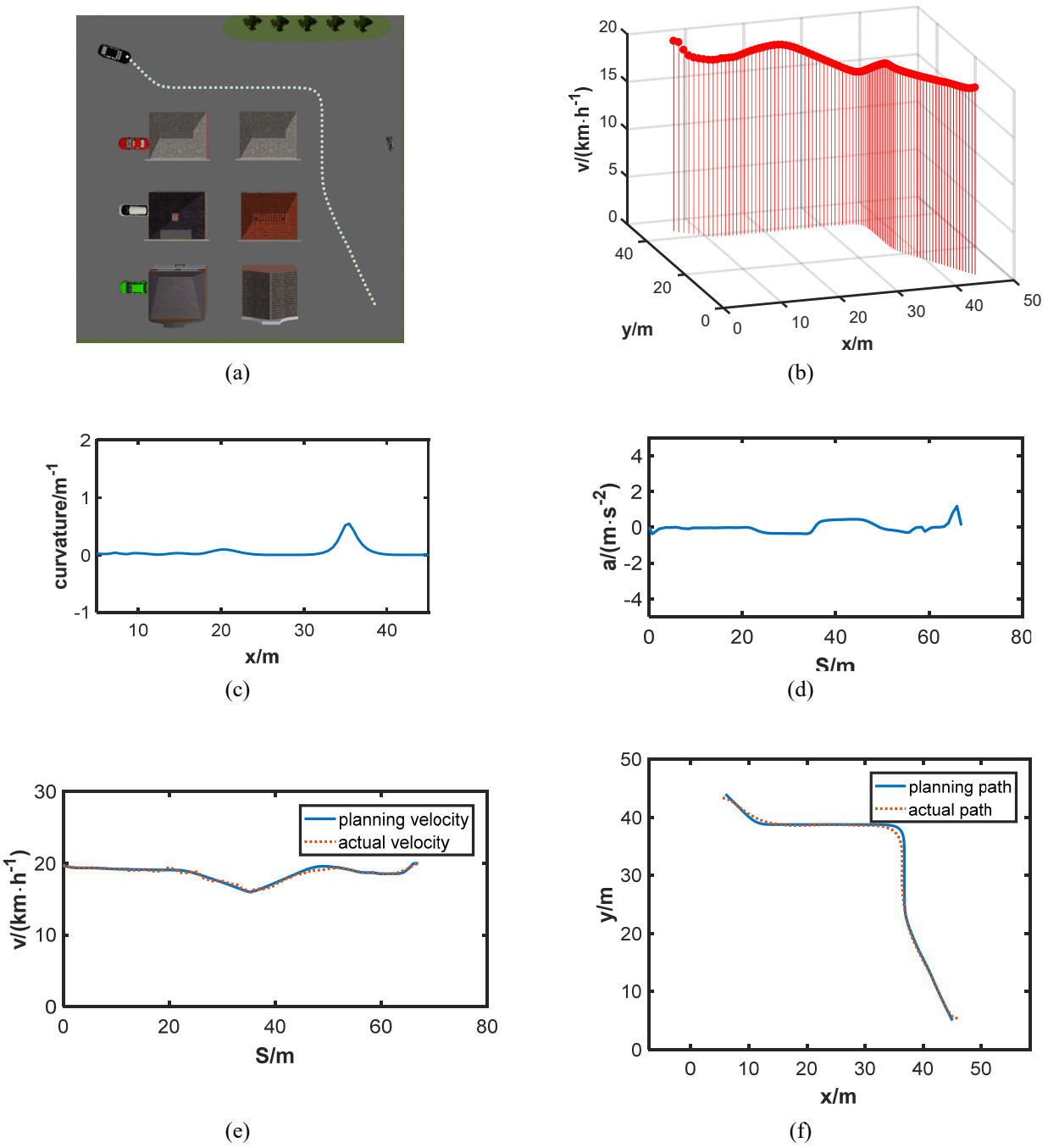


Figure 21. Simulation results of second scenario: (a) Planned path; (b) Velocity of each path point; (c) Curvature; (d) Acceleration; (e) Velocity following; (f) Path following.

Figure 20(c),(d) shows the curvature and acceleration of the path, which reflects the changes of the path and velocity, respectively. It can be seen from the figures that the curvature and acceleration are kept in a small range, which is a good proof that the path has good smoothness and continuity. It also proves that the trajectory output by the algorithm has good comfort.

Figure 20(e),(f) gives the results of velocity following and path following respectively. The errors shown in the two figures are both small. It not only shows the accuracy of the tracking algorithm, but also proves the practicability of the planning algorithm.

The simulation results of the second scenario are shown in Figure 21. The vehicle in the upper left corner are expected to reach the lower right corner. On average, it takes 141 ms per cycle for trajectory planning and tracking. It can meet the real-time requirements of vehicle computing platform.

Figure 21(a) shows the planned path. Figure 21(b) shows the velocity of each path point. Figure 21(c),(d) shows the curvature and acceleration of the path respectively. Figure 21(e),(f) gives the results of velocity following and path following respectively.

As can be seen from the figures, the algorithm also achieves good results in this scenario. The planned path is safe, efficient, smooth and stable. The planned speed is reasonable and efficient. Path tracking process is accurate and smooth.

5. Conclusions

The planning and control of intelligent vehicles are studied in this paper. A solution to the practical application of A* algorithm in automobile is proposed. In the process of trajectory planning, firstly, redundant spaces are set up to avoid the collision risk of the vehicle contours, so that A* algorithm can plan a safer vehicle path. Secondly, Bessel curve is used to smooth the path to eliminate the discontinuous points of the path, and reduce the curvature of the path, which is conducive to meeting the vehicle turning constraints. In addition, the vehicle velocity is planned based on the curvature, which ensures the driving stability. In the process of trajectory tracking, a speed control method based on expert system is established, and the feedforward information of expert system improves the stability of speed control; the direction control is based on the pure pursuit theory, and the selection method of target point in the pursuit model is given. In addition, considering the influence of speed, a correction coefficient is added to the pursuit model to improve the tracking accuracy.

The process proposed in this paper can solve some problems existing in the practical application of A* algorithm in vehicles and has good real-time performance. It has reference significance to improve the applicability of search-based algorithm on vehicles. Next, we will consider the application of search-based algorithm in intelligent vehicles on structured roads.

Acknowledgments

This work was supported by the National Key R&D Program of China (2018YFB0106200).

Conflict of Interest

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

References

1. Z. Wang, W. Du, J. Wang, J. Zhou, X. Han, Z. Zhang, et al., Research and application of improved adaptive MOMEDA fault diagnosis method, *Measurement.*, **140** (2019), 63–75.
2. B. Gao, Q. Yang, Z. Peng, W. Xie, H. Jin, S. Meng, A direct random sampling method for the Fourier amplitude sensitivity test of nonuniformly distributed uncertainty inputs and its application in C/C nozzles, *Aerosp. Sci. Technol.*, **100** (2020), 105830–105837.
3. H. Chen, H. Chen, Q. Liu, Three dimensional formation path planning of multiple UAVs based on improved artificial potential field method, *J. Syst. Simul.*, **1** (2019), 1–7.
4. Y. Li, R. Wang, Y. Liu, M. Xu, Satellite range scheduling with the priority constraint： An improved genetic algorithm using a station ID encoding method, *Chin. J. Aeronaut.*, **28** (2015), 789–803.
5. H. Wang, W. Mao, L. Eriksson, A Three-Dimensional Dijkstra's algorithm for multi-objective ship voyage optimization, *Ocean Eng.*, **186** (2019), 106131–106143.
6. B. Fu, L. Chen, Y. Zhou, D. Zheng, Z. Wei, J. Dai, et al., An improved A* algorithm for the industrial robot path planning with high success rate and short length, *Rob. Auton. Syst*., **106** (2018), 26–37.
7. F. Zhang, W. Bai, Y. Qiao, B. Xing, P. Zhou, UAV indoor path planning based on improved D*algorithm, *CAAI Trans. Intell. Syst.*, **14 (**2019), 662–669.
8. G. D. Goez, R. A. Velasquez, J. S. Botero, UAV route planning optimization using PSO implemented on microcontrollers, *IEEE Lat. Am. Trans.*, **14** (2016), 1705–1710.
9. H. Min, X. Xiong, P. Wang, Y. Yu, Autonomous driving path planning algorithm based on improved A* algorithm in unstructured environment, *Proc. Inst. Mech. Eng., Part D.*, **2020** (2020), 0954407020959741.
10. Y. Zhang, Y. Zhao, T. Wei, Improved A* algorithm for obstacle avoidance path planning strategy of the blind, *Aero Weaponry.*, **3** (2017), 86–92.
11. F. Islam, V. Narayanan, M. Likhachev, A*-Connect: Bounded suboptimal bidirectional heuristic search, *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016. Available from: https://ieeexplore.ieee.org/abstract/document/7487437.
12. Z. Zhang, T. Long, Z. Wang, G. Xu, Y. Cao, UAV dynamic path planning using anytime repairing sparse A* algorithm and targets motion estimation, *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, 2018. Available from: https://ieeexplore.ieee.org/abstract/document/9019099.
13. K. Mi, J. Zheng, Y. Wang, J. Hu, A multi-heuristic A* algorithm based on stagnation detection for path planning of manipulators in cluttered environments, *IEEE Access*, **7 (**2019**),** 135870–135881.
14. F. Islam, V. Narayanan, M. Likhachev, Dynamic multi-heuristic A*, *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015. Available from: https://ieeexplore.ieee.org/abstract/document/7139515.

15. Q. Wan, C. Gu, S. Sun, M. Chen, H. Huang, X. Jia, Lifelong multi-agent path finding in a dynamic environment, *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018. Available from: https://ieeexplore.ieee.org/abstract/document/8581181.
16. F. A. Raheem, U. I. Hameed, Heuristic D* algorithm based on particle swarm optimization for path planning of two-link robot arm in dynamic environment, *Al-Khwarizmi Eng. J.*, **15** (2019), 108–123.
17. S. Oh, H. W. Leong, Strict Theta*: shorter motion path planning using taut paths, *Proceedings of the Twenty-Sixth International Conference on International Conference on Automated Planning and Scheduling*, 2016. Available from: https://dl.acm.org/doi/abs/10.5555/3038594.3038626.
18. B. Cui, M. Wang, Y. Duan, Path planning for a* algorithm based on searching 24 neighborhoods, *J. Shenyang Univ. Technol.*, **40** (2018), 180–184.
19. F. Christ, A. Wischnewski, A. Heilmeier, B. Lohmann, Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients. *Veh. Syst. Dyn.*, **2019** (2019), 1–25.
20. Y. Zhang, B. Gao, L. Guo, H. Guo, M. Cui, A novel trajectory planning method for automated vehicles under parameter decision framework, *IEEE Access*, **7** (2019), 88264–88274.
21. Z. Wang, J. Zha, J. Wang, Flatness-based model predictive control for autonomous vehicle trajectory tracking, *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019. Available from: https://ieeexplore.ieee.org/abstract/document/8917260.
22. H. Wang, X. Chen, Y. Chen, B. Li, Z. Miao, Trajectory tracking and speed control of cleaning vehicle based on improved pure pursuit algorithm*, *2019 Chinese Control Conference (CCC)*, 2019. Available from: https://ieeexplore.ieee.org/abstract/document/8865255.
23. A. V. Le, V. Prabakaran, V. Sivanantham, R. E. Mohan, Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor, *Sensors*, **18** (2018), 2585–2612.
24. S. K. Renny, N. Uchiyama, S. Sano, Real-time smooth trajectory generation for nonholonomic mobile robots using Bezier curves, *Rob. Comput. Integr. Manuf.*, **41** (2016), 31–42.
25. J. Duan, C. Yang, H. Shi, Path tracking based on pure pursuit algorithm for intelligent vehicles, *J. Beijing Univ.* Technol., **42** (2016), 1301–1306.

AIMS Press