# Comparison of Path Algorithms in Driverless Vehicles and Application to Complex Paths

Yilin Zhang[1], Yuhan Zhang [2,*]

[1]Shanghai No.2 High School Shanghai, 200000, China
[2]Hefei New Oriental School, Hefei, 230000, China
*sr3255599@gmail.com

**Abstract.** This thesis focuses on path planning for driverless vehicles and discusses various path algorithms in depth. Driverless vehicles need path algorithms that can plan the optimal route from the starting point to the endpoint for the vehicle based on the map and the destination, taking into account the traffic conditions, real-time road conditions, etc., avoiding congestion and dangerous areas, such as planning routes around congested roads in the morning and evening peaks, and improving the travel efficiency. This paper concludes that traditional algorithms are more efficient in simple structured environments, but have poor adaptability in the face of complex paths; deep learning algorithms, although computationally complex and requiring a large amount of data, show strong adaptability and learning ability in complex dynamic environments, such as mountainous harsh environments. Comprehensive analysis of the advantages and disadvantages of different algorithms in specific complex path scenarios provides a basis for the selection and optimization of path algorithms for unmanned vehicles, which helps to improve the safety, efficiency and reliability of unmanned systems driving on complex paths, and promotes the further development and improvement of unmanned technology.

**Keywords:** Driverless Vehicles; Dijkstra's Algorithm; A* Algorithm; Learning Algorithm

## 1 Introduction

With the development of science and technology, unmanned systems are changing day by day, so it also puts forward higher requirements for vehicle navigation systems. Currently, traditional driverless algorithms include graph-based path planning algorithms such as Dijkstra and A* algorithms, among which Dijkstra algorithm, which is an algorithm used to find the shortest path between the nodes in a graph, is commonly used to solve the problem of calculating the shortest path between two points in a map [1].

A* algorithms are developed from Dijkstra evolution, which is a relatively mature heuristic algorithm, through which driverless vehicles are able to choose the optimal route in complex traffic environments, alleviate urban traffic congestion, improve traffic efficiency, and enhance the convenience of traveling [2]. It can also improve


J. Iqbal (ed.), *Proceedings of the 2025 2nd International Conference on Mechanics, Electronics Engineering and Automation (ICMEEA 2025)*, Advances in Engineering Research 272,
https://doi.org/10.2991/978-94-6463-821-9_20

traffic safety. Path algorithms enable vehicles to respond quickly and accurately to various road conditions and traffic participants, avoiding traffic accidents caused by human driving errors, reducing accident rates, and protecting the lives of passengers and pedestrians. For logistics, cab and other industries, the combination of efficient path algorithms can optimize driving routes, reduce fuel consumption and vehicle wear and tear, improve operational efficiency, and promote the development of related industries. However, its working principle is to walk along all possible paths until the path is the shortest, so it takes more time and is not favorable for special environments such as hilly areas and complex highways.

Hilly and mountainous areas tend to have more random and complex terrain than plains operations, with steeper gradients in the road surface. In this terrain, the use of ordinary agricultural transportation operations is easily restricted by the terrain conditions, and not only the problem of low operating efficiency and poor quality, but the stability of the machinery is also poor, prone to tipping over accidents, endangering the safety of the operator. Therefore, the application of mountain unmanned systems is particularly important. Due to the complexity and variability of the mountain environment, the traditional unmanned driver can not flexibly respond to unexpected situations, to realize obstacle avoidance, and therefore requires more sophisticated learning algorithms to assist in improving the unmanned driver navigation system [3].

Therefore, this paper adopts the method of dimensionality reduction, through the establishment of the Frenet coordinate system on the driving level, the obstacles are clustered and analyzed, and the principle of Python is used to realize obstacle avoidance, so as to realize the expansion of the application from ordinary paths to complex paths. This continuous expansion of the application scenarios of the path algorithms for driverless vehicles from simple structured roads to complex urban roads, rural roads, etc., can realize further enhancement of the algorithms, as well as the extension in specific fields such as logistics and distribution, mining, agriculture, etc., in order to adapt to the different complex environments and task requirements. Secondly, good algorithms can quickly re-plan routes when encountering unexpected situations such as road construction and vehicle breakdowns, while some simple algorithms may not be able to react in time. So it is necessary to compare different types of path algorithms, by comparing different algorithms, to clarify the advantages and disadvantages of each algorithm, as well as their applicability in different application scenarios.

Firstly, this chapter mainly introduces the research background and significance of unmanned systems, and proposes the introduction of Python to assist in avoiding obstacles, and secondly, this chapter mainly introduces the working principle, application scenarios and advantages and disadvantages of the mainstream unmanned algorithms, Dijkstra's algorithm and A* algorithm, nowadays. At the same time, this chapter mainly introduces Frenet coordinates and their transformation with Cartesian coordinates, simplifies the research process through dimensionality reduction and clustering methods, and establishes algorithm models in Python environment to realize the transformation of complex path algorithms. Next, this chapter introduces the application of the design algorithm in real vehicle testing, and its combination with

navigation algorithms to improve on the traditional algorithms. Finally, it summarizes and outlines the content of this paper and gives an outlook on the future of navigation.

## 2 Traditional Algorithms for Global Path Planning

### 2.1 Dijkstra algorithm

Dijkstra's algorithm was developed by Dutch computer scientist (Edsger Wybe Dijkstra) in 1956. The algorithm was introduced in 1956 by Dutch computer scientist Edsger Wybe Dijkstra and is mainly used to solve shortest path problems in graphs, such as calculating the shortest route between two locations in a mapping application.

#### 2.1.1 Working Principle.

It divides the nodes in the graph into a set of visited nodes and a set of unvisited nodes. At the beginning, the distance of the starting node is marked as 0 and the other nodes are marked as infinity. Then, each time, the closest node to the start node is selected from the unvisited nodes, it is added to the visited set, and the distance of the unvisited nodes adjacent to this node is updated. This distance is updated by comparing the original distance with the distance past the current node to the neighboring node (the original distance plus the weight of the edge from the current node to the neighboring node). This process is repeated until the target node is added to the visited set, at which point the shortest path from the start node to the target node is obtained[4].

#### 2.1.2 Dijkstra's algorithm advantages and disadvantages.

Dijkstra's algorithm The advantages of this algorithm are that it can guarantee to find the shortest path from the starting point to the endpoint, and the results are accurate and reliable; the logic of the algorithm is relatively simple, easy to understand and implement.

The disadvantage of the algorithm is that the time complexity is high, in large-scale maps or complex environments, the computation time will be longer, the amount of computation is also particularly large, and it is a static algorithm, once the environment changes (such as the emergence of new obstacles, road construction resulting in road impassability, etc.), the need to replan the path.

### 2.2 A* algorithm (A-star algorithm)

A* algorithm was published in 1968, combining Dijkstra's algorithm and breadth-first search algorithm, the principle of this algorithm is simple and easy to implement, with the help of heuristic function to find the optimal path faster, the search range is small, high efficiency, but not applicable to dynamic environments and high-dimensional space, the target point is unreachable when the performance consumption is large[5].

#### 2.2.1 algorithm summary.

The start and end points are determined, and the start node is put into a container called an "open list", which is used to store the nodes to be examined. Each node in the open list has an evaluation function value f(n), g(n) (the actual cost from the start node to the current node) of the start node is 0, and h(n) (the predicted cost from the current node to the target node) is usually calculated using a heuristic method, such as the Manhattan distance or the Euclidean distance:

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

At the same time, a "closed list" is created to hold the nodes that have already been examined [6].

Then the node with the smallest f(n) value is selected from the open list as the current node to be explored.

Iterate through the neighboring nodes of the current node, and judge each neighboring node: if the node is an obstacle or already in the "closed list" (which holds the nodes that have been explored), it is skipped; if it is not in the open list, calculate its g(n), h(n), and f(n) values, and set its parent node as the current node, and put it into the open list [7].

The A* algorithm selects nodes by synthesizing an evaluation function (1). Where g(n) denotes the actual cost from the start node to the current node n, e.g., the length of the path traveled; h(n) denotes the heuristically estimated cost from the current node n to the goal node, e.g., the straight line distance between two points; and f(n) is the integrated evaluation cost of the path through node n [8].

Finally, put the current node into the closed list and keep repeating steps 2-3 above until the endpoint is found or the open list is empty. If the endpoint is found, the complete path can be obtained by backtracking through the parent node of each node record.

### 2.2.2 Advantages and disadvantages of the algorithm.

The advantages of the A* algorithm incorporate heuristic information to guide the search direction, which usually allows rapid advancement towards the target node and reduces the search space. For example, in map navigation, the A* algorithm is able to find the path from the start point to the end point faster than blind search algorithms. Moreover, the A* algorithm is guaranteed to find the optimal solution when the heuristic function satisfies certain conditions.

The disadvantage of the A* algorithm is that it has a high memory requirement and needs to maintain an open list and a closed list, which may take up a lot of memory space when the search space is large. For example, in large-scale map navigation, it may lead to memory shortage. And the heuristic function has a large impact, if the heuristic function is not reasonably designed, the search efficiency of the A* algorithm may be reduced, and may even degrade to a situation similar to breadth-first search, resulting in a long search time. It is also not suitable for dynamic environments, when the environment changes dynamically, such as the sudden appearance of obstacles on the path, the A* algorithm may need to recalculate the entire path, and poor adaptability[9].

# 3 Analysis of obstacle avoidance in mountain unmanned systems

The traditional algorithms of unmanned driving systems are more suitable for the common routes of dissolving various obstacles, however, in the process of real vehicle driving, special conditions are various and unpredictable. As a result, it is especially crucial to design an algorithm specifically applicable to the complex road conditions which can achieve the purpose of avoiding barriers. This chapter takes mountainous road conditions as an example in order to study the applicable algorithms as well as their limitations.

The special characteristics of mountain road conditions are significant in the undulating terrain, rugged mountain roads, and slippery, muddy dangerous conditions such as boulders rolling down and landslides, which may seriously affect the stability and accuracy of the unmanned system. Although the traditional Dijkstra and A* algorithms can effectively calculate the shortest path on flat roads, they are mainly based on two-dimensional models and do not consider the complex three-dimensional topography, terrain undulation, and dynamic obstacles in mountainous areas. That's the reason why they have greater limitations in actual mountain driving. Huang Shuai et al. transformed it into a 2D visual model by introducing the Frenet coordinate system and combining it with the ground obstacles on the mountain road. Next, he converted the non-convex problem into a convex optimization problem through the obstacle clustering and boundary transformation strategy. Eventually, he set the relevant constraints and constructed a quadratic programming problem by taking lateral offset, speed, and acceleration as the optimization objectives, and solved it to obtain the obstacle avoidance path. In this chapter, this thesis will refer to its dimensionality reduction idea and explore it in Python runtime environment.

## 3.1 Introduction to Frenet coordinates

The establishment of Frenet coordinates requires a given reference line, the establishment of a planar right-angle coordinate system in the plane, the location of the car is given the coordinates and then projected to the reference line, the distance to the projection point is the transverse displacement, the projection point and the origin of the line is the longitudinal displacement, which will be the state of motion in the horizontal roadway into a visual image, which is convenient for analysing the state of motion [10].

## 3.2 Cartesian and Frenet Coordinate Conversion

In the Cartesian coordinate system, the motion law of the vehicle is usually described by the variation of important information such as displacement, velocity, acceleration, and curvature of motion of the vehicle.

### 3.3 Reference path, obstacle transformation to Frenet coordinate system

In this paper, the mountain unmanned driver travels according to the ground path planning trajectory, with the initial path on the ground as the reference line under the Frenet coordinate system, and its starting point is the planning starting point obtained from the previous decision-making task, the longitudinal displacement of a ground global trajectory point on the reference path can be expressed as the cumulative arc length between the point on the reference line and the starting point, and the transverse displacements of the reference path on the ground are all 0. The sizes of ground obstacles are expressed by point coordinates in the coordinate system is expressed in point coordinates, and spatial falling obstacles are introduced into the pyfluent database to achieve obstacle detection and avoidance using hydrodynamics-related principles.

### 3.4 Obstacle clustering method

Mountainous areas have many obstacles such as stones, which are usually densely distributed. In this paper, an obstacle aggregation method is proposed, which sets two extreme values k1 and k2 to represent the maximum aggregation distances in the vertical and horizontal directions, respectively. When the distances of neighbouring obstacles in the horizontal or vertical directions are smaller than these two extreme values, the maximum boundary values of them in the horizontal and vertical directions are selected under the Frenet coordinate system to achieve obstacle aggregation, so that the neighbouring obstacles are merged into a bigger obstacle, and the degree of discrete can be effectively reduce the degree of discretization. Similarly, rockfall clusters, can be considered as a whole, which is easy to analyze.

### 3.5 Obstacle Detection Algorithm

Obstacles and their types are more numerous in mountainous areas than on roads, therefore, it is necessary to classify obstacles into roadblocks and falling obstacles based on dimensionality. Due to the significant difference between the two types of obstacles, it is difficult to overcome the traditional analysis methods, whereas the Python database can be used to analyze different situations by introducing different modules, and the results of the analysis can be presented in a visual form. Therefore, this thesis used Python for analysing the coding.

#### 3.5.1 Roadblock Detection.

Assuming that the car is travelling to position (x, y) with speed (vx, vy), this thesis construct the algorithm model in a Python environment. Firstly, first, import the math model library and define the current position coordinates (x1,y1) of the moving entity a that avoids collision according to the function (3):

$$F = ma \tag{3}$$

Calculate the acceleration and update the position and velocity of the vehicle. By detecting the coordinates of the vertices of the irregular obstacle group at the current position, calculate the distance to the obstacle, approximate it as an ellipse and specify

(r1,r2), and establish the function of collision avoidance, i.e., the straight line distance of the line connecting the car and the midpoint of the obstacle is greater than or equal to $\sqrt{r1^2 + r2^2}$. If the distance is equal to$\sqrt{r1^2 + r2^2}$, then the speed is immediately changed to reverse speed (-vx,-vy). In practice, there is a friction factor, so it is also based on

$$f = \mu mg \tag{3}$$

Calculate the friction force, i.e.,

$$F = Fdraw - fF = Fdraw - f \tag{4}$$

where F draw is the car's own property, which can be set according to the actual situation. The sensor state is used to control the vehicle movement along the obstacle radius until the next obstacle is detected, and so on.

### 3.5.2 Airborne Falling Object Detection.

The real mountain situation is variable and complex, so the type of obstacles should be considered in many ways, and the 3D spatial falling objects are taken as an example of irregular stones. Stone falling is subject to many conditions, such as air resistance, manoeuvrability, rotational convection, shape factor, etc. Therefore, the pyfluent module in Python is introduced for processing. pyfluent is a software library in Python for processing fluid dynamics, which can be used to study complex fluid flow models.

First of all, the computational model is initialized by creating the flying body and setting its friction parameters, velocity and mass, setting the environmental parameters (gravitational acceleration, air friction coefficient), setting the time parameters, time step and simulation maximum time. After that the dynamics simulation is executed, the solver type is set, the steady state mode is switched off, the maximum number of iterations is set and finally, the simulation is executed.

In this process, this thesis creates the flying body, considers its flight angle, speed and mass, combine the air friction coefficients, perform the simulation and updates the position of the flying body in each time step.

## 3.6 Recommendations and Outlook

This chapter classifies and studies different computer algorithms for 2D, 3D obstacles in mountain driving. The combined use of digital and hydrodynamic processing modules in the context of Python and the Frenet coordinate system can simplify the problem of obstacles and car traveling in complex road conditions, and pave the way for combining with the traditional driverless navigation system in the later part of the paper. However, this paper only considers the effects of mountain obstacle types and does not take into account more complex limitations in practical application, such as mountain road gradient, soil type, etc., which is a limitation in the theory-to-practice transition. Compared to real driving environments, most tests of reinforcement learning methods are conducted in ideal environments with a single driving environment, which cannot reflect the complexity of the driving environment in real scenarios [11, 12].

# 4 Practical use of algorithms

## 4.1 Example of Algorithm Integration

Using the A* algorithm can find the characteristics of the path from the starting point to the end point faster, this chapter will be about the A* algorithm combined with the Python module. Firstly using A* navigation to achieve the shortest path planning, this thesis formulated the vehicle body length of 5m, body width of 1m, travelling speed of 40km/h, friction coefficient of 0.5, and estimated friction force of about 5000N. When driving the complex terrain, the Python database is enabled, and when 5m away from the obstacle is monitored, the speed is immediately changed to -40km/h, and when backing up to a safe distance of 10m, the vehicle is driven along an arc with a radius of 10m from the center point of the obstacle, and this step is repeated when the next obstacle is encountered. When an airborne falling object is detected, the pyfluent library is called, the size of the object is measured with the estimated time of landing, the range, and its proposed projection on the ground is regarded as a road obstacle, which is dodged according to the two-dimensional roadblock algorithm.

## 4.2 Algorithm improvement

Using the A* algorithm before traveling to the complex road section ensures the shortest driving time for simple road conditions, and enabling Python to assist in the complex road section effectively improves the problem that the A* algorithm is unable to avoid obstacles and re-plan routes. This combination makes the overall driving time the shortest, improves the efficiency of unmanned driving, and eliminates the hidden dangers of the driving process, taking into account the constraints in the actual driving, and realizes the algorithm has the practical application value of the improvement[11].

# 5 Conclusion

In this study, A* and Dijkstra algorithms in path planning for driverless vehicles are deeply analysed and studied. By comparing the two algorithms, the strengths, weaknesses and differences of the two algorithms are clearly shown.

Dijkstra's algorithm, with its rigorous global optimality guarantee, highlights its value in scenarios that require high path accuracy, ensuring that the vehicle always travels on the theoretically optimal trajectory. However, due to its unguided blind search, the computational resources and time consumption are huge, especially in large-scale maps or complex traffic networks, and the real-time performance is difficult to meet the needs of unmanned vehicles.

The A* algorithm, with its efficient heuristic search strategy, shows excellent path planning speed in most conventional scenarios, significantly reduces computational complexity and time cost, and is able to quickly provide feasible paths for unmanned vehicles, which greatly improves the real-time response capability of the system. However, its path optimality is affected by the accuracy of the heuristic function in

complex environments, and if the heuristic function is not properly designed, it may lead to low efficiency or even fail to find the optimal solution.

When traditional algorithms are combined with learning algorithms, the disadvantage of their inability to deal with complex paths can be improved. The unmanned navigation system based on learning algorithms uses simulation experiments to validate the decision-making method, which is closer to real driving scenarios, and has controllability and safety. However, the learning algorithm still has certain limitations, and future research can consider more limiting factors based on this article to ensure the practicability of the experiment.

The future development of driverless technology should be based on specific application scenarios and needs to flexibly choose algorithms or explore the way of integration. A variety of traditional algorithms will be fused to complement each other. For example, the combination of the A* algorithm and Dijkstra's algorithm ensures that the optimal solution can be found while improving the search efficiency, so as to better cope with the complex and changing road conditions. In the pursuit of efficient real-time planning while taking into account the optimality of the path, in order to promote driverless vehicles in the complex and changing traffic environment, robust, safe, and intelligent driving, to lay a solid foundation for the comprehensive upgrading of the intelligent transport system.

## Authors Contribution

All the authors contributed equally and their names were listed in alphabetical order.

## Reference

1. Lu, F.: 'Research on path planning algorithm based on mobile robot', Anhui University of Technology, 2023. DOI:10.27763/d.cnki.gahgc.2023.000213
2. Gong, H., Ni, C., Wang, P., et al.: 'Smooth path planning method based on Dijkstra algorithm', Journal of Beijing University of Aeronautics and Astronautics, 2024, 50, (2), pp. 535-541. DOI:10.13700/j.bh.1001-5965.2022.0377
3. Tang, X.: 'Research on motion control of wheeled tractors with an attitude adjustment in hilly and mountainous areas', Northeast Electric Power University, 2024. DOI:10.27008/d.cnki.gdbdc.2024.000391
4. Zhang, H.: 'Design and implementation of the shortest path for train departure under emergency events based on Dijkstra algorithm', China Storage and Transportation, 2023, (12), pp. 104-105. DOI:10.16301/j.cnki.cn12-1204/f.2023.12.041
5. Li, X., Zhu, P., Zhang, Y., et al.: 'Research on path planning algorithm', Automation Applications, 2024, 65, (22), pp. 126-129. DOI:10.19769/j.zdhy.2024.22.037
6. Wei, Y., Jin, F., Dong, K., et al.: 'Improved global path planning A* algorithm based on node optimization', Computer Measurement and Control, 2023, 31, (6), pp. 143-148. DOI:10.16526/j.cnki.11-4762/tp.2023.06.022
7. Teng, L.: 'Research on path planning and trajectory tracking control algorithm for unmanned vehicles', Zhejiang University of Science and Technology, 2024. DOI:10.27840/d.cnki.gzjkj.2024.000072

8. Bai, J., Bai, Y., Xi, J., et al.: 'Workshop material distribution path planning based on improved A* algorithm', Journal of Jilin University (Science Edition), 2024, 62, (6), pp. 1401-1410. DOI:10.13413/j.cnki.jdxblxb.2023507
9. Dong, Q., Yang, W., Dong, G., et al.: 'Research on autonomous driving path planning based on A* algorithm', Heilongjiang Science, 2024, 15, (18), pp. 74-77
10. Huang, S., Wang, J., Huang, J., et al.: 'Research on local obstacle avoidance path planning algorithm for unmanned mining trucks', Control and Information Technology, pp. 1-7 [2024-12-23]. DOI:10.13889/j.issn.2096-5427.2024.06.500
11. Wang, S.: 'Research on autonomous driving trajectory planning algorithm based on Frenet coordinate system sampling', Lanzhou University of Technology, 2019
12. Yu, S., Ma, C., Chen, J.: 'Research progress of autonomous driving lane change decision algorithm based on learning', Automobile Practical Technology, 2023, 48, (24), pp. 189-194. DOI:10.16638/j.cnki.1671-7988.2023.024.037