A mobile robot path planning algorithm based on improved A* 

Jinzheng Shi¹, Yifan Su²* Chunguang Bu³ and Xiaoliang Fan ⁴

¹School Of Automation And Electrical Engineering, Shenyang Ligong University, Shenyang, Liaoning, 110159, china
²State Key Laboratory of Robotics, Shenyang Institute of Automation, Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang, Liaoning, 110016, china
³Corresponding author’s e-mail: 1023442559@qq.com

Abstract. Global path planning of mobile robot aims to provide a safe and smooth path for mobile robot navigation. Traditional A * algorithm is planning the path of more turns, and not smooth. Moreover, for the u-shaped terrain, the path stick to the obstacles. Aiming at the shortcomings of A* algorithm, cosine distance is selected as the heuristic function, and direction information is added and normalized. The 36-order neighborhood search matrix is selected to solve the problem of fitting u-bend. In addition, a post-processing method based on Bessel curve is proposed. Make the planned path curvature change continuously. Simulation results show the effectiveness of the improved algorithm.

1. Introduction
Path planning is one of the key technologies of mobile robot autonomous navigation. It is to search for an optimal non-collision path from the starting point to the target point[1]. Grid-based path planning algorithms mainly include: Dijkstra algorithm[2], A*[3], D*[4], where Dijkstra algorithm can search for the optimal path, but the search’s efficiency is low. D* series algorithms are mainly aimed at autonomous navigation in dynamic environment. Global path planning for a known static environment, A* algorithm can more efficiently and quickly solve the shortest path between the starting point and the target point [5]. It is widely used in mobile robots [6].

2. A* algorithm
A* algorithm as classic heuristic search algorithm, the core of A* algorithm is the valuation function, in which heuristic value is added, so that the search process from heuristic substitution value to the target state continuously. Therefore, it may save a lot of computational process and improve the efficiency of the algorithm[7].

2.1 Traditional A*

\[ f(s) = g(s) + h(s) \] (1)

Where, \( f(s) \) is the estimated value of node \( s \). \( g(s) \) represents the connected generation value from initial state \( s_{\text{start}} \) to state \( s \), and \( h(s) \) represents the heuristic value from state \( s \) to target state \( s_{\text{goal}} \).

Search process as shown in figure 1, A * algorithm, A is the starting point and B is the target point. The path search starts from the starting point A, and the current node is the parent node in the process,
search its child nodes connected area, through the formula 1 each child nodes are calculated separately, and the value f, select the minimum value f of the search for the next node, and calculate its parent, until the search for the target point B, finally from the target point B reverse back its parent to obtain a path.

![Fig1. A* algorithm search process](image)

Usually, the heuristic function selects the straight-line distance (Euclidean distance) from the node to the target point as the heuristic cost[8],

$$h(s) = \sqrt{(X_G - X_s)^2 + (Y_G - Y_s)^2}$$  \hspace{1cm} (2)

Where, \((X_G, Y_G)\) is the coordinate of target point G, and \((X_s, Y_s)\) is the coordinate of node S.

2.2 Problems of A* algorithm

A* algorithm still has the following problems.

- The search path has lots of turns;
- in the search process, the robot is simplified into particle points and the actual size of it is ignored;
- although the planned path can effectively avoid impassable obstacles, it is short of obstacles. In practical applications, the robot cannot follow the path well and is prone to collision;
- the planned path smoothness is poor.

3. Improved A* algorithm

3.1 Improved heuristic function

Traditional heuristic functions of A* only have distance information[9], as shown in formula 2. On this basis, cosine similarity and direction information are introduced to make the search direction more close to the target direction. The improved heuristic function is shown in formula 3.

$$h(s) = u \cdot L + v \cdot \cos \alpha + w \cdot \sin \alpha$$  \hspace{1cm} (3)

Where u, v and w represent static weights.

L represents the distance information contained in the traditional A*. As shown in formula 2 above, Euclidean distance function is selected to estimate the distance information of two nodes.

\(\alpha\) is the vector from the starting point to the target point, and the Angle between the parent node and the vector of the current child node.

\(\cos \alpha\) represents the cosine similarity of two vectors. Cosine similarity measures the similarity between two vectors by measuring the cosine of the Angle between them. Cosine of 0 degrees is 1, and cosine of any other Angle is less than 1; And its minimum value is negative 1. So the cosine of the Angle between the two vectors determines whether the two vectors point roughly in the same direction. When two vectors have the same direction, the cosine similarity value is 1; When the Angle between the two vectors is 90°, the cosine similarity value is 0. When two vectors point in opposite directions, the cosine similarity is negative 1.

\(\sin \alpha\) indicates direction information.

Taking the combination of distance information, cosine similarity and direction information as heuristic functions, there exists the problem of unit disunity, so the distance and Angle need to be
normalized. After processing, only the contribution of the two to the path is considered to avoid the problem of unit disunity.

3.2.36 Search the neighborhood matrix

The search neighborhood of A* algorithm is designed as the matrix shown in the figure, and an 11*11 matrix is established centering on the search of child nodes. At this time, if all the points marked as 1 in the matrix are free nodes, the current node is considered passable, and vice versa. In practical applications, search matrix of appropriate size should be selected according to the robot's own size and map resolution to determine the distance between the path and obstacles, so as to ensure the safety of the robot during its operation.

![Search the neighborhood matrix](image)

Fig. 2 36 Search the neighborhood matrix

4. Path smoothing method

Bézier curve [10] by Bernstein polynomial (Bernstein polynomial) evolved. Bézier curve was used to smooth the path of the improved A* algorithm.

The N order of Bernstein polynomials has the following form:

\[ b(t) = \sum_{i=0}^{n} \beta_i \cdot B_{ni}(t), \quad t \in [0, 1] \]  

Among i = 0, 1, ..., n, and

\[ \binom{n}{i} = \frac{n!}{i!(n-i)!} \]  

It's the binomial coefficient. Bernstein n - order polynomials form a basis for exponential polynomials of n - order. In general, Bernstein polynomials can be expressed as

\[ B_n(t) = \sum_{i=0}^{n} \beta_i \cdot B_{ni}(t) \]  

Where the \( \beta_i \) is called the Bernstein coefficient, the Bernstein polynomials become bézier curve when the Bernstein coefficients are a series of fixed points in a two-dimensional plane[11].

Combining the bézier curve with global path planning, the total number of path points generated by the global path planning A* algorithm is n+1, and the \( \beta_i \) Bernstein coefficient is set as the ith path point Pi generated by the global path planning A* algorithm, where i = 0, 1, ..., n.

\[ B_n(t) = \sum_{i=0}^{n} p_i \cdot b_{ni}(t) \]  

Bézier curve has the following properties:

- the starting and ending points of the fitting curve coincide with the starting and ending points of the characteristic polygon;
- once the position of the points of the feature polygon is defective, the fitting curve is unique

After smoothing, the efficiency and flexibility of the actual movement of the robot can be improved.
5. Simulation Results
As shown in figure 3, the figure is a two-dimensional static global environment with a map size of 400*400. The dark areas are walls and obstacles, the upper right dot is the starting point A, and the lower right dot is the target point B.

![Static global map](image)

Fig3. Static global map

Fig4. The result of A* algorithm planning
As shown in figure 4, it is the path searched by the traditional A* algorithm. It can be clearly seen that this path is relatively close to the obstacle and has many turns.

![A* algorithm planning](image)

Fig5. Improved path planning of A* algorithm
6. Conclusion

It can be seen from the simulation experiment that the improved A* algorithm can well consider the relationship between the robot body and obstacles, solve the problem that the traditional A* path fits the obstacles, and process the generated path through the bézier curve, achieving the smooth path effect. The improved A* can better play a global guiding role in robot autonomous navigation.

References

[1] Jeddisaravi K, Alitappeh R J, A. Pimenta L C, et al. Multi-objective approach for robot motion planning in search tasks[J]. Applied Intelligence, 2016, 45(2):305-321.

[2] Kang H I, Lee B, Kim K. Path Planning Algorithm Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm[C]// Workshop on Computational Intelligence & Industrial Application. 2009.

[3] Hart P E, Nilsson N J, Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths[J]. IEEE Transactions on Systems Science and Cybernetics, 1968, 4(2):100-107.

[4] Stentz A. Optimal and efficient path planning for partially-known environments[C]// Robotics and Automation, 1994. Proceedings. 1994 IEEE International Conference on. IEEE, 1994.

[5] Yap P. Grid-based path-finding[C]// Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence. Springer-Verlag, 2002.

[6] Warren C W. Fast path planning using modified A* method[C]// IEEE International Conference on Robotics & Automation. 1993.

[7] Wang D. Indoor mobile-robot path planning based on an improved A* algorithm[J]. Journal of Tsinghua University, 2012.

[8] Ferguson D, Stentz A. Using interpolation to improve path planning: The Field D* algorithm[J]. Journal of Field Robotics, 2010, 23(2):79-101.

[9] Yao J, Lin C, Xie X, et al. Path Planning for Virtual Human Motion Using Improved A* Star Algorithm[C]// Seventh International Conference on Information Technology: New Generations. 2010.

[10] Chen C, He Y Q, Bu C G, et al. Feasible trajectory generation for autonomous vehicles based on quartic Bézier curve[J]. Zidonghua Xuebao/Acta Automatica Sinica, 2015, 41(3):486-496.

[11] Forrest A R. Interactive Interpolation and Approximation by Bézier Polynomials[J]. Computer-Aided Design, 1990, 22(9):527-537.