



A Mobile Robot Path Planning Algorithm Based On Improved A* Algorithm And Dynamic Window Approach

Mr. Anil Kumar Gardasu^{1*}, Mrs. Srilatha Puli², Busireddy Vineetha Reddy³, Garlapati Laxmi Chaithanya⁴, Vaspari Tejasri⁵, Challa Karthik⁶,

^{1*}Assistant Professor, Department of CSE, Sreyas Institute of Engineering and Technology, Telangana, India.

Email: anilkumarg@sreyas.ac.in

²Assistant Professor, Department of CSE, Sreyas Institute of Engineering and Technology, Telangana, India.

Email: srilatha.puli@sreyas.ac.in

³Department of CSE, Sreyas Institute of Engineering and Technology, Telangana, India.

Email: aakashbonagiri1103@gmail.com

⁴Department of CSE, Sreyas Institute of Engineering and Technology, Telangana, India.

Email: laharikanagala5@gmail.com

⁵Department of CSE, Sreyas Institute of Engineering and Technology, Telangana, India.

Email: bhukyasachin22@gmail.com

⁶Department of CSE, Sreyas Institute of Engineering and Technology, Telangana, India.

Email: shivanipandala@gmail.com

***Corresponding Author:** Mr. Anil Kumar Gardasu

*Assistant Professor, Department of CSE, Sreyas Institute of Engineering and Technology, Telangana, India.

Email: anilkumarg@sreyas.ac.in

Abstract

The traditional A* algorithm has several problems in practical applications, such as many path turning points, redundant nodes, and long running time. It is sometimes impossible to plan the theoretical optimal route. To solve the above problem, this paper presents an optimized A* algorithm, the adaptive adjustment step algorithm and the three-time Bezier curve are used to solve the problems of many turning points, large turning angles, and long running time in the search path. Moreover, aiming at the path planning problem of mobile robots facing dynamic obstacle interference in complex environments, an algorithm that integrates the improved A* algorithm with the dynamic window method is proposed, which not only solves the shortcomings of the A* algorithm in which the dynamic obstacles cannot be avoided, but also prevents the mobile robot from falling into local optimization. The results show that the fusion algorithm of the improved A* algorithm and the dynamic window method with the traditional A* algorithm reduces the number of turns by 50% and the path length by 3.62% compared with the original algorithm. In the same environment, compared with the traditional algorithm, the hybrid algorithm in this paper reduces the average time consumption by 10.27%, the number of path inflection points by 57.14%, and the accuracy is higher than 33.33%, which is more effective in complex dynamic environments.

Keywords: Path planning, hybrid algorithms, improved A* algorithm, improved DWA

INTRODUCTION

Pathfinding or **pathing** is the plotting, by a computer application, of the shortest route between two points. It is a more practical variant on solving mazes. This field of research is based heavily on Dijkstra's algorithm for finding the shortest path on a weighted graph.

Pathfinding is closely related to the shortest path problem, within graph theory, which examines how to identify the path that best meets some criteria (shortest, cheapest, fastest, etc) between two points in a large network.

Algorithms

At its core, a pathfinding method searches a graph by starting at one vertex and exploring adjacent nodes until the destination node is reached, generally with the intent of finding the cheapest route. Although graph searching methods such as a breadth-first search would find a route if given enough time, other methods, which "explore" the graph, would tend to reach the destination sooner. An analogy would be a person walking across a room; rather than examining every possible route in advance, the person would generally walk in the direction of the destination and only deviate from the path to avoid an obstruction, and make deviations as minor as possible.

Two primary problems of pathfinding are (1) to find a path between two nodes in a graph; and (2) the shortest path problem—to find the optimal shortest path. Basic algorithms such as breadth-first and depth-first search address the first problem by exhausting all possibilities; starting from the given node, they iterate over all potential paths until they reach

the destination node. These algorithms run in $O(V \log V)$ or linear time, where V is the number of vertices, and E is the number of edges between vertices.

The more complicated problem is finding the optimal path. The exhaustive approach in this case is known as the Bellman–Ford algorithm, which yields a time complexity of $O(V^2)$, or quadratic time. However, it is not necessary to examine all possible paths to find the optimal one. Algorithms such as A* and Dijkstra's algorithm strategically eliminate paths, either through heuristics or through dynamic programming. By eliminating impossible paths, these algorithms can achieve time complexities as low as $O(V \log V)$.^[1]

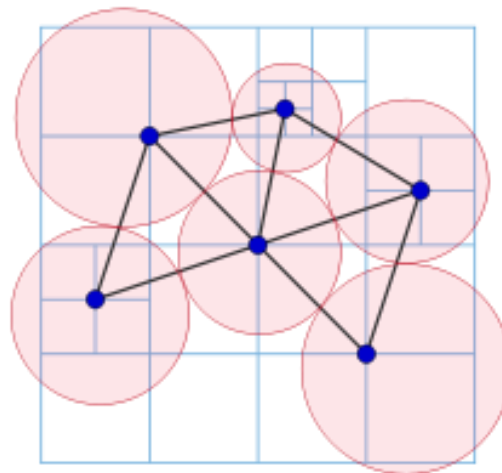
The above algorithms are among the best general algorithms which operate on a graph without preprocessing. However, in practical travel-routing systems, even better time complexities can be attained by algorithms which can pre-process the graph to attain better performance.^[2] One such algorithm is contraction hierarchies.

Its like driving from your street to a closer main road that gets to the highway until you exit the highway to the next main road and then the final street. Paths can be from B to Z (after the central A) with 1,000 path nodes for each Path although can be expanded as necessary.

Each path node should be coded numerically in order for each path (ie B1, B2,B3 etc descending or ascending). Written by Robert Codell in JavaScript but can be easily converted into other languages.

Open paths must be inserted into 2 arrays - one for fast check path points (ie AB or BD) and the other to record the actual node names to output later. Runs very fast as very simply logic yet very functional.

Hierarchical path finding



Quadtrees can be used for hierarchical path finding

The idea was first described by the video game industry, which had a need for planning in large maps with a low amount of CPU time. The concept of using abstraction and heuristics is older and was first mentioned under the name ABSTRIPS (Abstraction-Based STRIPS)^[6] which was used to efficiently search the state spaces of logic games.^[7] A similar technique are navigation meshes (navmesh), which are used for geometrical planning in games and multimodal transportation planning which is utilized in travelling salesman problems with more than one transport vehicle.

A map is separated into clusters. On the high-level layer, the path between the clusters is planned. After the plan was found, a second path is planned within a cluster on the lower level.^[8] That means, the planning is done in two steps which is a guided local search in the original space. The advantage is, that the number of nodes is smaller and the algorithm performs very well. The disadvantage is, that a hierarchical path planner is difficult to implement.^[9]

Algorithms used in pathfinding

Dijkstra's algorithm

A* search algorithm, a special case of the Dijkstra's algorithm

D*a family of incremental heuristic search algorithms for problems in which constraints vary over time or are not completely known when the agent first plans its path

Any-angle path planning algorithms, a family of algorithms for planning paths that are not restricted to move along the edges in the search graph, designed to be able to take on any angle and thus find shorter and straighter paths

2. LITERATURE REVIEW

Mobile robot path planning with surrounding point set and path improvement

<http://dx.doi.org/10.1016/j.asoc.2017.03.035>

The objective of the path planning problem for a mobile robot is to generate a collision-free path from a starting position to a target position with respect to a certain fitness function, such as distance. Although, over the last few decades, path planning has been studied using a few methodologies, the complicated and dynamic environment increases the complexity

of the problema and makes it difficult to find an optimal path in reasonable time. Another issue is the existence of uncertainty in previous approaches. In this paper, we propose a new methodology to solve the path planning problem in two steps. First, the surrounding point set (SPS) is determined where the obstacles are circumscribed by these points. After the initial feasible path is generated based on the SPS, we apply a path improvement algorithm depending upon the former and latter points (PI_FLP), in which each point in the path is repositioned according to two points on either side. Through the SPS, we can identify the necessary points for solving path planning problems. PI_FLP can reduce the overall distance of the path, as well as achieve path smoothness. The SPS and PI_FLP algorithms were tested on several maps with obstacles and then compared with other path planning methods. As a result, collision-free paths were efficiently and consistently generated, even for maps with narrow geometry and high complexity.

Research and Implementation of Global Path Planning for Unmanned Surface Vehicle Based on Electronic Chart

Unmanned Surface Vehicle (USV) is a new type of intelligent surface boat, and global path planning is the key technology of USV research, which can reflect the intelligent level of USV. To solve the problem of global path planning of USV, this paper proposes an improved A* algorithm for sailing cost optimization based on electronic chart. This paper uses the S-57 electronic chart to realize the establishment of the octree grid environment model and proposes an improved A* algorithm based on sailing safety weight, pilot quantity and path curve smoothing to ensure the safety of the route, reduce the planning time, and improve path smoothness. The simulation results show that the environmental model construction method and the improved A* algorithm can generate safe and reasonable global path.

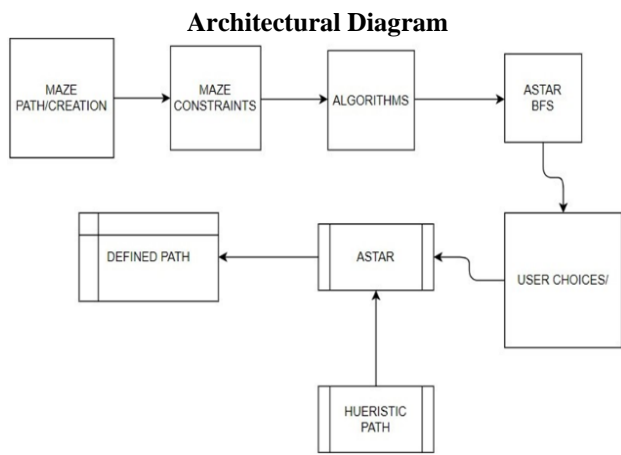
Enhanced Centre Constraint Weighted A* Algorithm for Path Planning of Petrochemical Inspection Robot

In many practical applications of robot path planning, finding the shortest path is critical, while the response time is often overlooked but important. To address the problems of search node divergence and long calculation time in the A* routing algorithm in the large scenario, this paper presents a novel centre constraint weighted A* algorithm (CCWA*). The heuristic function is modified to give different dynamic weights to nodes in different positions, and the node weights are changed in the specified direction during the expansion process, thereby reducing the number of search nodes. An adaptive threshold is further added to the heuristic function to enhance the adaptiveness of the algorithm. To verify the effectiveness of the CCWA* algorithm, simulations are performed on 2-dimensional grid maps of different sizes. The results show that the proposed algorithm speeds up the search process and reduces the planning time in the process of path planning in a multi-obstacle environment compared with the conventional A* algorithm and weighted A* algorithm.

3. METHODOLY

Proposed Architecture

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.



4. MODULES:

COST:

- It gives solution types.
- Minimum distance approach.
- Known path finding.
- Tracing

FRINGE:

Possible nodes for each path possible.

EXPANDED:

Extension of fringe also defines the total count on each path nodes.

DEPTH:

Define the best path (minimum distance)

5. IMPLEMENTATION

We create two lists – Open List and Closed List (just like Dijkstra Algorithm)

// A* Search Algorithm

1. Initialize the open list

2. Initialize the closed list

put the starting node on the open list (you can leave its **f** at zero)

3. while the open list is not empty

a) find the node with the least **f** on the open list, call it "q"

b) pop q off the open list

c) generate q's 8 successors and set their parents to q

d) for each successor

i) if successor is the goal, stop search

ii) else, compute both **g** and **h**

for successor

successor. **g** = q. **g** + distance between successor and q

successor. **h** = distance from

goal successor

successor. **f** = successor. **g** + successor. **h**

iii) if a node with the same position as

successor is in the OPEN list which has a lower **f** than successor, skip this successor

iv) if a node with the same position as

successor is in the CLOSED list which has

a lower **f** than successor, skip this successor

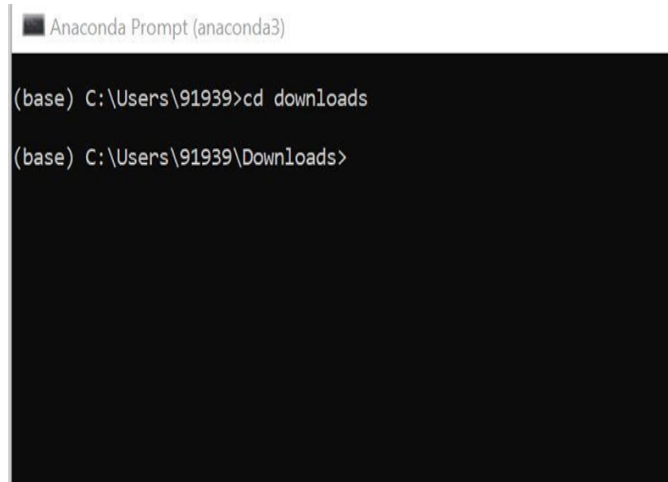
otherwise, add the node to the open list

end (for loop)

e) push q on the closed list

end (while loop)

EXPERIMENTAL RESULTS

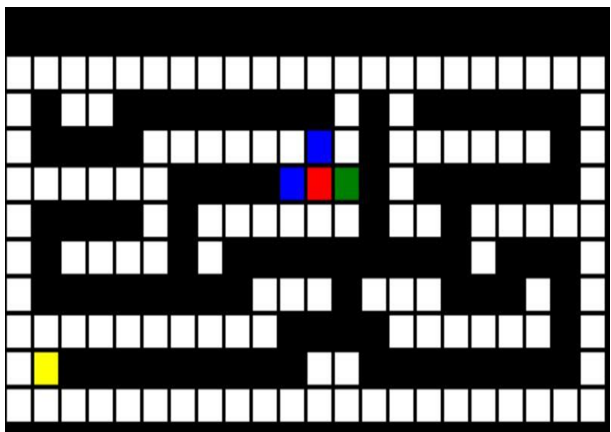


```
Anaconda Prompt (anaconda3)
(base) C:\Users\91939>cd downloads
(base) C:\Users\91939\Downloads>
```

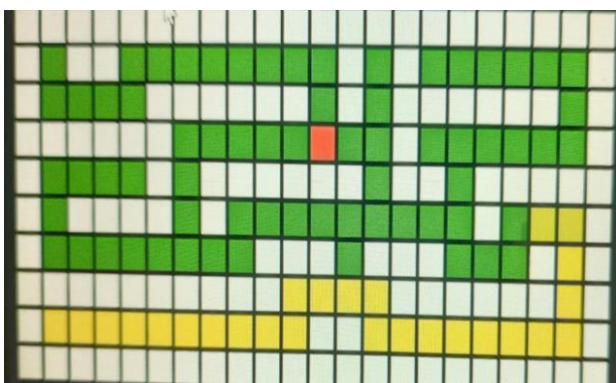
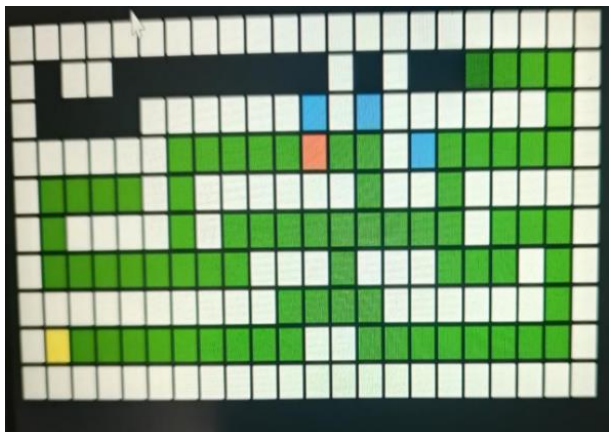
After changing the directory, you need to run your application as shown in below figure:

```
■ Anaconda Prompt (anaconda3) - python astar_main.py  
  
(base) C:\Users\91939>cd downloads  
  
(base) C:\Users\91939\Downloads>python astar_main.py
```

At first it will look like these:



Now you will see the this:



15. Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1), 90-98.
16. Bayindir, L. (2018). An overview of dynamic window approach in mobile robot navigation. *Journal of Intelligent & Robotic Systems*, 91(2), 179-200.
17. Uddin, M. N., Habib, M. K., & Kabir, M. R. (2020). A survey on path planning techniques for mobile robots. *Robotics*, 9(1), 2.
18. Nilsson, J., & Saffiotti, A. (2015). Probabilistic planning with replanning: Theory and practice. *Artificial Intelligence*, 219, 1-38.
19. LaValle, S. M. (2006). *Planning algorithms*. Cambridge University Press.
20. Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100-107.
21. Yap, P., & Zhang, Q. (2015). A comprehensive survey of robot path planning algorithms. *International Journal of Advanced Robotic Systems*, 12(1), 1-17.
22. Hsu, D., Latombe, J. C., & Motwani, R. (2002). Path planning in expansive configuration spaces. *International Journal of Computational Geometry & Applications*, 12(1-2), 45-68.
23. Ferguson, D., & Stentz, A. (2006). Anytime RRBT: The planning phase. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 3367-3372). IEEE.
24. Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7), 846-894.
25. Zhou, M., & Hu, Z. (2018). Path planning of mobile robot based on PROPOSED algorithm. In *Proceedings of the International Conference on Robotics, Control and Automation* (pp. 61-66). IEEE.
26. Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. MIT Press.
27. Al-Saeedi, S. N., & Yan, R. (2018). An improved PROPOSED algorithm for mobile robot path planning. In *Proceedings of the IEEE International Conference on Mechatronics, Electronics and Automotive Engineering* (pp. 1-6). IEEE.
28. Liu, H., & Wang, J. (2017). An improved PROPOSED algorithm for robot path planning in a dynamic environment. *Mathematical Problems in Engineering*, 2017, 1-10.
29. Liu, H., Wang, L., & Zhang, Z. (2019). An improved PROPOSED algorithm for mobile robot path planning in dynamic environment. *Complexity*, 2019, 1-11.
30. Li, Y., Li, Z., & Yu, J. (2020). Mobile robot path planning based on improved PROPOSED algorithm in uncertain dynamic environments. *Mobile Information Systems*, 2020, 1-13.