AUV path planning for environment changes over time

Jianping Li[a *], Yuliang Shi[b]

Faculty of Information Technology, Beijing University of Technology 100124, Beijing, China.

Abstract. Path planning is an important research field in robotics, and it is also a key technology for realizing automatic navigation of aircraft. It is of great significance both in theory and in application. The task of planning trajectories for an aircraft has received considerable attention in the research literature. Most of the work assumes that the threat to the aircraft is static; less attention has been paid to the problem of the changing environment. However, the threat from the weather when the UAV is flying in the air is always changing. This paper introduces an improved BFS algorithm, which can find the best safe path when the threat is always changing.

1 Introduction

The use of Unmanned Air Vehicles (UAVs) has been increased recently[1]. Literature has addressed extensively the motion planning problem for one or more unmanned aerial vehicle moving through a field of obstacles to a goal. In most of the problem-solving approaches there are measures that specify the quality and goodness of the approach. In UAV path planning, these measures as defined by [2] are:

1. Completeness: the planner should find the solution (path from start to goal) if there is one, in some situations the planner can't guarantee to find the solution even if it exists; this is due to some problems like dead-lock.

2. Optimality: the solution or the path that is found should be the shortest between all the potential solutions exists.

3. Uncertainty: in some situations, the UAV may have little or no information about the environment or its work space, so how the planner can deal with such situation to find the path.

Some methods meet only a few of these requirements, and some meet all requirements. Traditional path planning methods include depth first search(DFS) [3], breadth first search(BFS) and swarm intelligence algorithms like genetic algorithms, ant colony algorithms, etc. However, these algorithms are often used to solve the problem that the solution to the problem with a constant risk range from the start point to the end point. In reality, especially in the field of UAV track planning, the threat is often changing with time. For long distance transport drones, weather is the most important threat. For example, a sudden hurricane will destroy unmanned aerial vehicles. The most important feature of weather is changing all the time. How to plan the optimal trajectory in a changing environment has become an important issue to be resolved. This paper improves on the basis of the traditional BFS algorithm to make it capable of planning of planning the drone route under variable weather.

2 Previous work

The objective of path planner is to move the unmanned aerial vehicle from some location in the world to a goal location, such that it avoids all obstacles and minimizes a positive cost metric (e.g., length of the traverse) [4]. Many excellent path planning algorithms have emerged in the path planning field. In 1959, Dijkstra proposed the famous Dijkstra algorithm [5]. The advantage of this algorithm is that the algorithm is concise and the optimal solution can be obtained. The disadvantage is that it is inefficient, and it takes up a lot of space when computing. In 1984, Bundy A, Wallen L invented the Breadth-first search algorithm, which is a layer-by-layer traversal layer blocking [6]. In 1996, Lydia E invented the random roadmap method (RPM) [7]. The advantage of this method is that it is not easy to fall into a local minimum when the computation is small. It can be applied to multi-MATEC Web of Conference degree-of-freedom robot path planning. Its disadvantage is that the path is unstable. The rapid expansion of the random tree method RRT [8] is based on a tree-structured search algorithm. The RRT algorithm expands a tree structure from the starting point outward, and the extension direction of the tree structure is determined by randomly taking points in the planning space. Similar to PRM, this method is not optimal. In 1994 Stentz A proposed an improved D* algorithm [9] based on the A* algorithm.

2.1 BFS algorithm for optimal trajectory

*Corresponding author: [a]lwtg_jpli@163.com [b] shiyl@bjut.edu.cn

The traditional breadth first search algorithm is often used to solve the maze problem that starting from the entrance, after leaving the entrance, visiting the nodes adjacent to the current position and the access from these adjacent nodes. The parent node of each cell is recorded until the exit to the maze is found, then an optimal solution to the maze problem is obtained, which means that the shortest path of the maze is found. A notable feature of BFS is that in the process of solving, the path from the starting point to all other reachable points is actually solved. Unlike the swarm intelligence algorithm [10], it is easy to converge to the local optimal solution. There are cases where there are paths in some places but they can't be found. BFS is very suitable to solve the problem of real-time drones doing express transportation is generally a starting point for multiple destinations.

2.1.1 Breadth first search general steps

1) Put the starting point S into the sources collection
2) Fails to exit if the sources collection is empty. Otherwise continue to the next step.
3) Take the first source of the sources collection
4) If source is the target node, the solution exits successfully; otherwise, skip to the next step.
5) If the source can't find the successor node, skip to step (2), otherwise continue.
6) Extend the node source, place its children in the sources, and the child parent points to the source.
7) If a node in a child node is target node. Then find a solution. Return successfully or skip back to step (2).

Breadth first search is a blind search. Blind search is less efficient and consumes more computation space and time. Contrary to blind search is heuristic search. Heuristic search introduces heuristic information in the search process, evaluates each search position in the state space, obtains the best position, and then searches from these positions to the target, which can eliminate a large number of invalid search paths and improve efficiency [11]. The A* algorithm is the most famous heuristic search algorithm. However, the A* algorithm cannot be used for the path planning problem directly in a multi-variable environment (the specific content of A* algorithm will be introduced later). Because it will put the searched places into the close list, and nos longer search for these points. With the change of the environment, the points in the open list may become dangerous points, so that no path can be found. Obviously, this is wrong, but we can transform the A* algorithm to improve the BFS algorithm so that it will find a path in the changing environment.

A* is an informed search algorithm, or a best-first search, meaning that it solves problems by searching among all possible paths to the solution (goal) for the one that incurs the smallest cost (least distance travelled, shortest time, etc.), and among these paths it first considers the ones that appear to lead most quickly to the solution. It is formulated in terms of weighted graphs: starting from a specific node of a graph, it constructs a tree of paths starting from that node, expanding paths one step at a time, until one of its paths ends at the predetermined goal node. At each iteration of its main loop, A* needs to determine which of its partial paths to expand into one or more longer paths. It based on an estimate of the cost (total weight) still to go to the goal node. Specifically, A* selects the path that minimizes. The following is the cost formula of A* algorithm.

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

where n is the current node on the path, g(n) is the cost of the path from the start node to current node, and h(n) is a heuristic that estimates the cost of the cheapest path from n to the goal. The heuristic is problem-specific. For the algorithm to find the actual shortest path, the heuristic function must be admissible, meaning that it never overestimates the actual cost to get to the nearest goal node[12].

3 Discussion

Because the heuristic part of the A* algorithm h(n) is the evaluation of the distance of the target point, the algorithm can only find one target point at a time, and if it wants to find the path of multiple target points, it needs to cycle multiple times. For a problem in which the environment changes over time, the time consumption is mainly in the cycle from the start time to the end time. If all the paths can be found in one cycle, the efficiency will be greatly improved. The BFS algorithm will find the path of all reachable points in each round of each cycle, so the increase of the number of target points will not increase the time complexity of the algorithm. With the change of time, the reachable point in the current stage may become a dangerous point and be discarded in the next stage. Therefore, when searching for a path, the searched point cannot be placed in the close list and the search is no longer performed as in the A* algorithm. We need to save the reachable points found in the previous phase as possible starting points for the next phase so that we can ensure that every reachable route is not missed.

4 Algorithm frameworks

4.1 Threat model

It is important to plan the path of the unmanned aerial vehicles before it takes off. It is necessary to comprehensively consider the various threats. For the sake of simplicity, only one wind force threat is considered in this paper. The area where the wind force is greater than the threshold is set as a dangerous area. Although the weather changes over time, if the time taken is short enough, it can be considered that there the weather does not change in a short time. This article assumes that the weather does not change in one hour and put every hour's danger points into danger matrix.

4.2 This article algorithm ideas

Construct a hazard area matrix for each hour, treating each hour as a stage. The first stage visits its neighbour from the starting point and sets the starting point as the parent of its neighbour. Then add the new point to the next step source queue as the starting point for the next visit. In this experiment, the UAV takes one step every two minutes. It can take 30 steps per hour (that is, each stage), and there are 18 stages to go. It will through these 18 stages then exit.

It is necessary to ensure that the UAV flight route has a high safety factor and a short flight time. Because weather forecasting cannot be 100% accurate, safe-prone areas also have a little probability of being dangerous. Obviously, the farther away from the danger zone, the safer it is. In order to comprehensively consider such factors as time distance and security, it is necessary to define a scoring function for each node. If there are some successor nodes then the node with the highest score is selected as the successor node, and if it is not, it stays in place. The scoring function is as follows.

$$f=\alpha*D+\beta*\bar{D}-\gamma*L-\delta*T \quad (2)$$

Among them, α, β, γ, δ are adjustable parameters, which are used to adjust the proportion of each component in the total score. D is the shortest distance between the current node and the dangerous area. $\bar{D}$ is the average distance from the starting point to the current point at the shortest distance from the hazardous area. L is the length of the path that has been travelled, and T is the length of time that has elapsed.

4.3 Algorithm steps

1） add the starting point to the sources queue.
2） Stage=0;
3） If stage<NUM_OF_STAGE-1
4） Stage++;
a) Step=0
b) If step<NUM_OF_STEP-1 and source que is not empty
- Take a point form open as the current point
- Calculate the highest-scoring point among the currently adjacent four points.
- Judge where the node is superior to the best node in history. If it's true, we will record the current Node score, and its parent node.
- Update the best node in history
- Add this point to the next sources queue of the next step
- Step++

5) Otherwise, if you reach the last step in this phase, update the corresponding variables in this phase to prepare for the next phase of the search.

6） Check if there is a path to the target at the current stage. Save it if it exists.

7) If it gets to the last stage, exit the program

Detailed algorithmic steps are shown in the following flow chart.

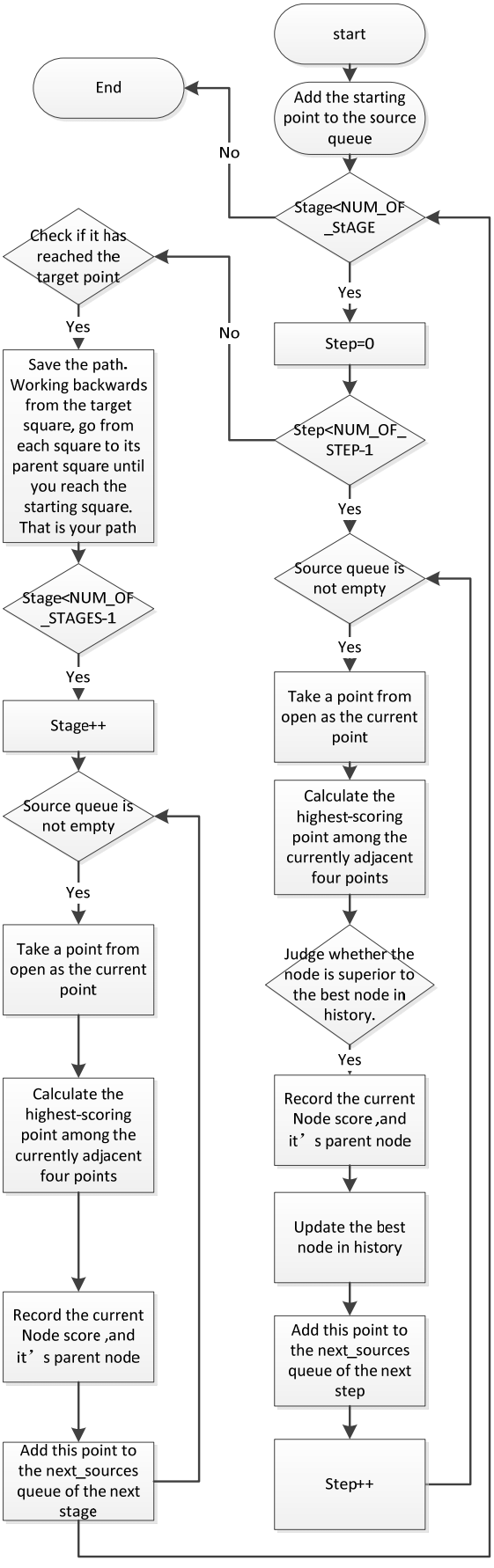

Figure 1. Flowchart of this algorithm

5 Experiments

In order to investigate the feasibility and effectiveness of the proposed method, I did the following experiment.

5.1 Experimental comparison

The figure below depicts the comparison of two algorithms for a route within an hour. The direction of progress is from left to right. The black area represents the dangerous area, the dotted line represents the path of the traditional BFS algorithm plan, and the solid line represents the route of the improved BFS algorithm plan.

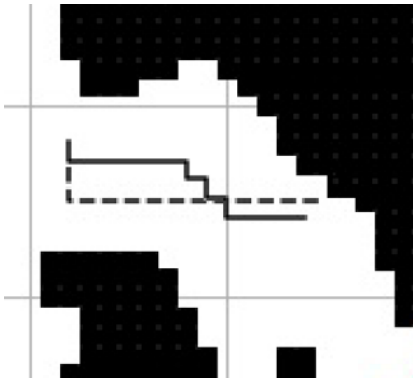

Figure 2 Path comparison chart

It can be seen that the path obtained by the traditional BFS algorithm will very close to the dangerous area when it encounters the dangerous area. However, the improved route will consider various factors such as the distance to the dangerous area and the time, so as to select a better route and enhance the safety of the route while ensuring the route that can be found.

We use weather data predicted by the Bureau of Meteorology for the first to fifth days as test data, and then evaluate the number of routes that arrive safely every day based on actual weather data every day. The chart is as follows.

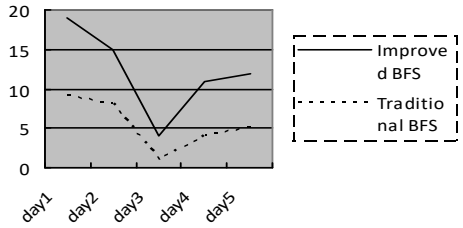

Figure 3. Arrived path

It can be seen from the figure that the 5-day test data shows that the improved BFS algorithm predicts that the number of safe paths is higher than that predicted by the traditional BFS algorithm, which means that the improved BFS algorithm greatly improves road safety.

According to formula (1) we calculate the flight costs of the two algorithms, as shown in the following figure.

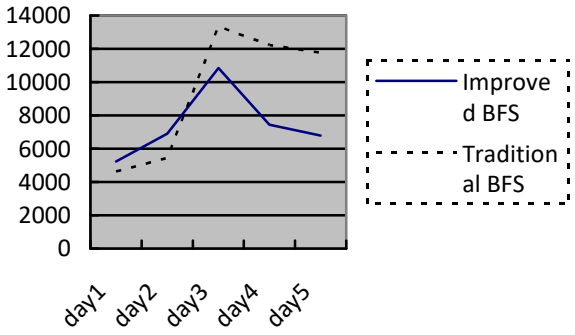

Figure 4. Cost score comparison chart

It can be seen that the planned path of the improved BFS algorithm is much smaller than the traditional BFS algorithm except that the cost of the previous two days is slightly higher than that of the traditional BFS algorithm. In general, the path of the improved BFS algorithm planning is less expensive than the path planned by the traditional BFS algorithm.

6 Conclusion

On the basis of the BFS algorithm, this paper compares the scores of extensible nodes when each node extends and compares it with the historical optimal value. If the node score is greater than the optimal value, the optimal value is updated. Thus, the optimal path for the current scoring function can be guaranteed. For the time dependence of the problem, by circling according to time, the position of the path at a specific point in time is determined, and a path highly related to time is planned. We only need to change the node scoring function appropriately for different time related issues. Therefore, the construction of the scoring function is also a very important task, and its quality will directly affect the path finally found.

References

1. Rathbun D, Kragelund S, Pongpunwattana A, et al. An evolution-based path planning algorithm for autonomous motion of a UAV through uncertain environments[J]. Algebra Universalis, 2002, **2**:551--607.
2. J. C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1991.
3. H. Shwail S, Karim A, Turner S. Probabilistic Multi Robot Path Planning in Dynamic Environments: A Comparison between A* and DFS[J]. International Journal of Computer Applications, 2013, **82**(7):29-34.
4. Stentz A. Optimal and efficient path planning for partially-known environments[C]// IEEE International Conference on Robotics and Automation, 1994. Proceedings. IEEE, 2002:3310-3317 vol.**4**

5. Skiena S. Dijkstra's algorithm[J]. Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley, 1990: 225-227.
6. Bundy A, Wallen L. Breadth-first search[M]//Catalogue of Artificial Intelligence Tools. Springer, Berlin, Heidelberg, 1984: 13-13.
7. Amato N M, Wu Y. A randomized roadmap method for path and manipulation planning[C]//Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on. IEEE, 1996, **1**: 113-120.
8. Kuffner J J, LaValle S M. RRT-connect: An efficient approach to single-query path planning[C]//Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on. IEEE, 2000, 2: 995-1001.
9. Stentz A. Optimal and efficient path planning for partially-known environments[C]//Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on. IEEE, 1994: 3310-3317.
10. Ahn C W, Ramakrishna R S. A genetic algorithm for shortest path routing problem and the sizing of populations[J]. IEEE transactions on evolutionary computation, 2002, **6**(6): 566-579.
11. Jensen R M, Bryant R E, Veloso M M. SetA*:an efficient BDD-based heuristic search algorithm[C]// Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, July 28 - August 1, 2002, Edmonton, Alberta, Canada. DBLP, 2002:668-673.
12. Zeng W, Church R L. Finding shortest paths on real road networks: the case for A*[J]. International Journal of Geographical Information Systems, 2009, **23**(4):531-543.