UAV Path Planning based on Bidirectional Sparse A* Search Algorithm

Meng Bo-bo, Gao Xiaoguang

Northwestern Polytechnical University, Xian, Shaanxi, 710129, China
mbbmj@hotmail.com

***Abstract*—Nowadays, unmanned aerial vehicle (UAV) as an effective platform has been widely applied to military and civil fields. This paper focuses on UAV path planning problem. Sparse A* Search algorithm (SAS) proposed by R.J.SZCZERBA for route planning is introduced firstly. Then the Bidirectional Sparse A* Search algorithm (BSAS) is presented after the discussion of the deficiency of the application of SAS algorithm in UAV path planning. BSAS algorithm expands the start point and goal point respectively for searching path and calculates whether there is a Line of Sight (LOS) between every two different expansion nodes produced by the two expansion processes to get UAV path. The results show that the BSAS algorithm has stronger robustness and obtains better planning path than the SAS algorithm.**

Keywords- UAV; A* algorithm; path planning

I. INTRODUCTION

In recent years, Unmanned Aerial Vehicle (UAV) path planning has received considerable research attention. There are many studies on UAV path planning using various approaches, such as A* algorithm[1-4], Genetic Algorithm (GA)[5, 6], Ant Colony Algorithm (ACA)[7], Particle Swarm Optimization (PSO)[8, 9] and etc. Among all these algorithms, A* algorithm has been widely used because of its simplicity of realization.

This paper firstly describes the path planning based on Sparse A* Search algorithm (SAS), and then improves such algorithm to solve the UAV path planning problem. The rest of the paper is organized as follows. In section 2, we describe the SAS algorithm proposed by Robert J. Szczerba. Our Bidirectional Sparse A* Search algorithm (BSAS) for UAV path planning which is improved based on SAS algorithm is discussed in detail in section 3. In section 4, relevant simulation results are presented. Finally the main conclusion and further research directions are outlined in section 5.

II. PATH PLANNING BASED ON SAS ALGORITHM

First of all, the evaluation function for path planning is described as[4]:

$$J = \sum_{i=1}^{n} (w_1 l_i^2 + w_2 f_{i,threat}) \qquad (1)$$

Where J is evaluation function for path planning, n is the total number of the path edges, l_i is the length of the i'th edge and f_i is the threat cost when UAV flying from the i'th edge. w_1 and w_2 are the weights of the two parts.

A* is an optimal, best-first search heuristic that computes a cost function for various locations in an environment. Equation (2) is the cost function that is minimized at each step of the A* algorithm[4]:

$$f(n) = g(n) + h(n) \qquad (2)$$

Where $g(n)$ is the actual cost from the start point to the intermediate point n and $h(n)$ is the estimated cost from the intermediate point n to the goal point. At each step in the A* propagation, the minimized $f(n)$ value is selected and inserted into a sorted list which named Open to store the possible path. It has been proved that if the actual cost from point n to the goal point is greater than or equal to the estimate ($h(n)$) of this cost, then the A* algorithm is guaranteed to be a minimum cost solution[4].

When using the A* algorithm to search path, the algorithm searches only its neighborhood grid cells, but may take a very long time and use an unbounded amount of memory to coverage to an optimal solution. Moreover, the expansion nodes do not always satisfy the kinetic constraints for unmanned aerial vehicles, and UAV can not fly according to such flight path in reality. To reduce the search space and search time for A* algorithm, Robert J. Szczerba has improved A* algorithm and proposed the Sparse A* Search algorithm (SAS) for aircraft route planning.

The main idea of SAS is to integrate the constraints proposed by Robert J. Szczerba into the search process in order to reduce the search space and search time for A* algorithm. These constraints include minimum route leg length, maximum turning angles, route distance constraint and fixed approach vector to goal position.

Minimum route leg length constraints the route to be straight for a predetermined minimum distance before initiating a turn. Maximum turning angles constraints to turn with an angle less than or equal to a preset maximum turning angle in the generated path. Considering the two constraints, the area can be searched by A* algorithm for one step is showed in Figure 1.Where l_{min} is the minimum route leg length, and θ_{max} is the maximum turning angles.

Figure 1 shows the search area is a fan-shaped region. The SAS Algorithm divides the fan-shaped region into N sector regions and uses the N+1 fan-tail nodes as the expansion nodes. Using this method, SAS reduces the search space and search time.

Route Distance constraints the length of the route to be less than or equal to a preset maximum distance. Route distance constraint is described as[4]:

$$D(x) + L(x) \le L_{max} \qquad (3)$$

Where $D(x)$ is the actual distance of the calculated path from the start point to the point x and $L(x)$ is the straight-line distance from point x to the goal. L_{max} is the maximum distance as mentioned above.

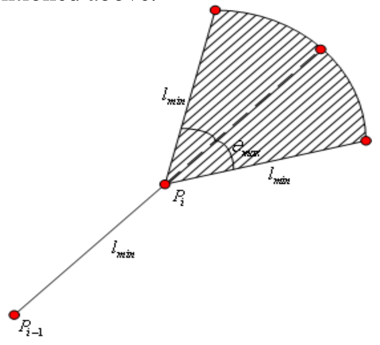

Figure 1. A* search area by one step

Fixed approach vector to goal position constraints the route to approach the goal position from a predetermined approach angle. The SAS algorithm solves this constraint by adding a high cost offset to MC array in a "bucket-shaped" area around the goal position[4].

The detailed description about the SAS algorithm can be found in [4].

III. Path Planning based on BSAS Algorithm

As mentioned above, the SAS algorithm which integrate the constraints into the search process can reduce the search space and search time for A* algorithm by adjusting three parameters. The three adjustable parameters include: l_{min}, θ_{max} and N. When the three parameters change, the path obtained will change correspondingly.

Although the SAS algorithm reduces the search space, but the method by adjusting the parameters for quick search will give rise to certain risks. Figure 2 is a "snapshot" which shows a failure search by using SAS algorithm. The SAS algorithm can not obtain a valid expansion node for path nodes 3 because all of its expansion nodes will be in the threat areas by using the current parameters settings. However, the line segment from path node 3 to goal point does not pass through any threat regions and the line segment satisfies the constraint of maximum turning angles, so UAV can reach the goal point along this line segment. That means there is a flyable path in the searching space which will not be searched by SAS algorithm.

In fact, Figure 2 has demonstrated a method to solve such problems. We can get a flyable path by connecting path node 3 and goal point. Through this idea, we modify the SAS algorithm as follows. Before expanding the current node, the algorithm checks whether or not the line segment between the current node and goal node passes through every threat areas If the line segment is out of threat areas, algorithm will add the goal point as an expansion node for the current node and finish the search.

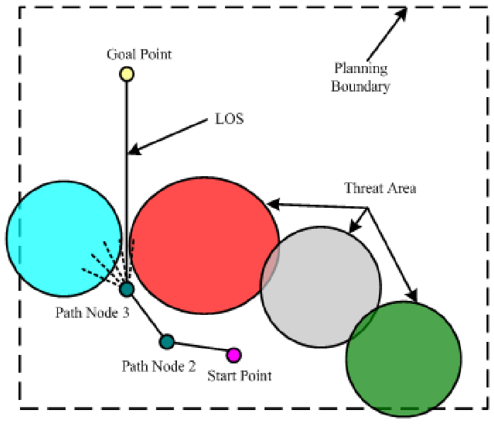

Figure 2. LOS in current node and goal node

We name the flyable line segment between path node 3 and goal node as "Line of Sight (LOS)", and assume that UAV has a global vision. When the goal point is "visible", UAV can jump out the search mode and fly directly from the current point to the goal point.

However, the LOS has the same constraints like the general expansion nodes. The length of LOS can not be less than l_{min} and LOS must satisfy the maximum turning angles constraint.

In this way, by adding the LOS, SAS algorithm has stronger robustness. However, not all of the goal points are globally visible. Figure 3 shows an example of this situation. For the start point, the path node 2 and path node 3, there is not any LOS between the goal point and each one of them and the improved SAS algorithm can not search a flyable path by using LOS.

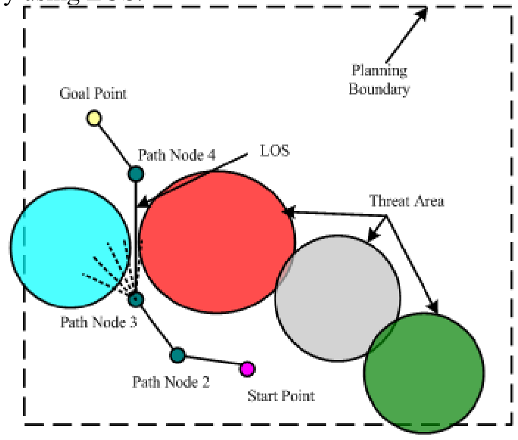

Figure 3. LOS in current node and other node

Therefore, the key to solve this searching problem is to expand the start point and goal point respectively and calculate whether there is a LOS between every two different expansion nodes produced by the two expansion processes.

However, if we expand the goal point and get its expansion point, we are able to search a flyable path. As is shown in Figure 3, the path node 4 is an expansion node of goal point, and there is a LOS between path node 3 and path node 4. In this way, we can get a flyable path by connecting path node 3 and path node 4.

We name this improved SAS as a Bidirectional Sparse A* Search (BSAS) Algorithm based on LOS.

The process about BSAS is shown as below:

Step1: Check whether there is a LOS between the start point and goal point. If so, then switch to Step8, otherwise, the turn to next step;

Step2: Insert the start point and goal point separately into an empty list *S* and list *T*;

Step3: Remove the node x from the list S, and expand the node x by using SAS algorithm, If there is not any suitable expansion point for node x , then algorithm turns Step5, otherwise, algorithm get a suitable expansion node x^* and perform the next step;

Step4: Check whether there is a LOS between the node x^* and each node in list *T*, if so, turn to Step8, otherwise, turn to the next step;

Step5: Remove the node y from the list *T*, and expand the node y by using SAS algorithm, If there is not any suitable expansion point for node y , then algorithm turns to Step7, otherwise, algorithm get a suitable expansion node y^* and perform the next step;

Step6: Check whether there is a LOS between the node y^* and each node in list *S*, if so, turn to Step8, otherwise, turn to the next step;

Step7: If the above-mentioned Step3 and Step5 both can not expand the node, algorithm judges no suitable route exists and search ends, otherwise, algorithm turns to Step3;

Step8: Searching processes find a route, and the algorithm ends.

Through the above analysis, we can see that BSAS algorithm has stronger robustness than SAS algorithm, which will be illustrated in the following simulations.

IV. Simulation Results

In this section some experiment results are given to show the effectiveness of the path planning strategy outlined in the previous section. The simulation is carried out as follows: the length unit is km, the planning space is set as 100km × 100km two-dimensional plane, the start point is located at (87,25) and the goal point at (25,90). The parameters mentioned above are set as below: $l_{\min} = 8$, $\theta_{max} = 60^o$, $N = 6$ and $w_1 = w_2 = 0.5$.

Simulation 1: There are six threat areas and the information about the threat area is shown in Table Ⅰ:

Figure 4 and Figure 5 show the results of SAS algorithm and BSAS algorithm respectively.

TABLE I. THREAT AREA INFORMATION

No.	Center Coordinate (km)	radius(km)
1	(32,60)	15
2	(57,81)	9
3	(79,72)	12
4	(40,31)	11
5	(62,54)	15
6	(15,32)	10

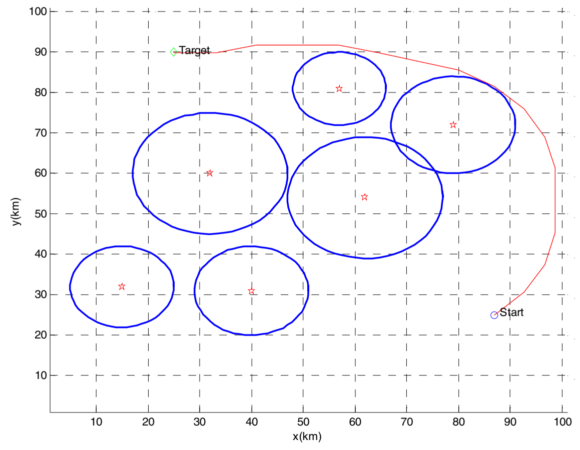

Figure 4. Simulation 1 result using SAS algorithm

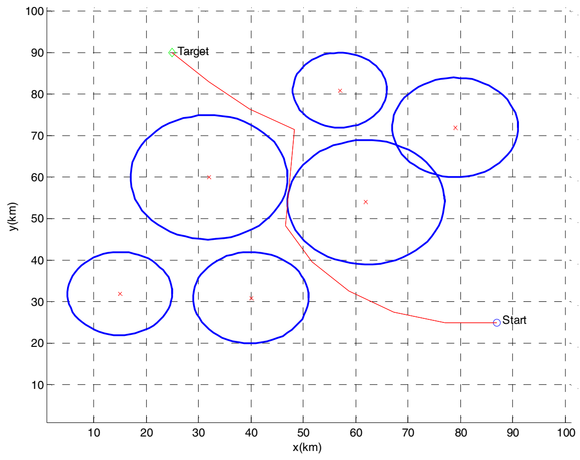

Figure 5. Simulation 1 result using BSAS algorithm

Simulation 2: The center coordinate of threat 1 is altered to (30,60) and other parameters remain the same as simulation 1. The results of SAS algorithm and BSAS algorithm are shown as Figure 6 and Figure 7 respectively.

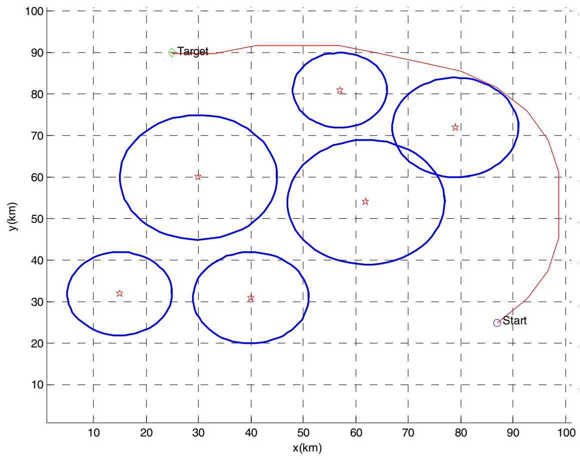

Figure 6. Simulation 2 result using SAS algorithm

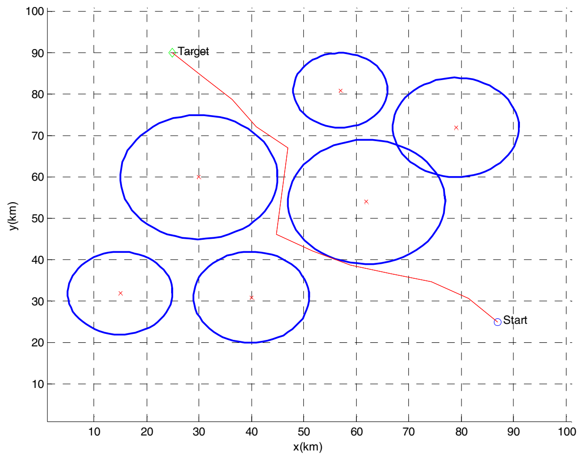

Figure 7. Simulation 2 result using BSAS algorithm

Simulation 3: l_{min} is altered to 10 km and other parameters remain the same as simulation 2. There is no path available using SAS algorithm and the result of BSAS algorithm is shown as Figure 8.

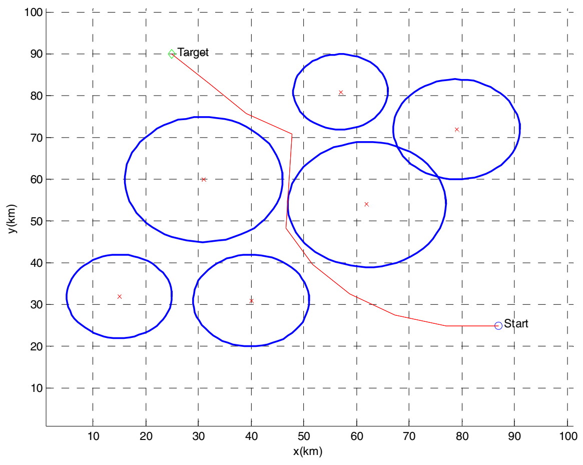

Figure 8. Simulation 2 result using SAS algorithm

Table II shows the information about each simulation. As the result shows in Table II, the BSAS algorithm has a stronger robustness than SAS algorithm and we can get a more excellent solution by BSAS algorithm.

TABLE II. SIMULATION INFORMATION

No.	Algorithm	Path length(km)	Time(s)	Expansion nodes
1	SAS	128.28	0.30	380
1	BSAS	101.97	0.46	268
2	SAS	128.28	0.27	330
2	BSAS	100.93	0.45	270
3	BSAS	102.57	0.28	145

V. CONCLUSION

In this paper, a novel algorithm called BSAS algorithm for path planning of UAV has been put forward. Compared with SAS algorithm, the algorithm we presented has its primary strengths because it has 1) better planning result and 2) stronger robustness. Furthermore, the BSAS algorithm can be extended to route planning in_other vehicles such as a pilot, unmanned airship and robot.

REFERENCES

[1] Bodhale D, Afzulpurkar N, Thanh N T, "Path planning for a mobile robot in a dynamic environment," Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics, IEEE Press, February 2009, pp. 2115-2120, doi: 10.1109/ROBIO.2009.4913329.

[2] Peng L, Xinhang H, Min W, "A hybrid method for dynamic local path planning," 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing, IEEE Computer Society Press, April 2009, pp. 317-320, doi: 10.1109/NSWCTC.2009.135.

[3] Sathyaraj B, Jain L, Finn A, et al, "Multiple UAVs path planning algorithms: A comparative study," Fuzzy Optimization and Decision Making, 2008, 7(3): pp. 257-267, doi: 10.1007/s10700-008-9035-0.

[4] Szczerba R J, Galkowski P, Glicktein I S, et al, "Robust algorithm for real-time route planning," Aerospace and Electronic Systems, 2000, 36(3): pp. 869-878, doi: 10.1109/7.869506.

[5] Obermeyer K J, "Path planning for a UAV performing reconnaissance of static ground targets in terrain," AIAA Modeling and Simulation Technologies Conference, AIAA Press, August 2009, pp. 1-11, AIAA-2009-5888.

[6] Pongpunwattana A, Rysdyk R, "Evolution-based dynamic path planning for autonomous vehicles," Innovations in Intelligent Machines - 1, 2007, Volume 70: pp. 113-145, doi: 10.1007/978-3-540-72696-8_5.

[7] Fridman A, Weber S, Kumar V, et al, "Distributed path planning for connectivity under uncertainty by ant colony optimization," 2008 American Control Conference, IEEE Press, June 2008, pp. 1952-1958, doi: 10.1109/ACC.2008.4586778.

[8] Foo J L, Knutzon J S, Oliver J H, et al, "Three-dimensional multi-objective path planning of Unmanned Aerial Vehicles using particle swarm optimization," 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA Press, April 2007, pp. 2170-2179, AIAA-2007-1881.

[9] Tsung-Ying Sun, Chih-Li Huo, Shang-Jeng Tsai, et al, "Optimal UAV flight path planning using skeletonization and parical swarm optimizer," 2008 IEEE Congress on Evolutionary Computation (CEC 2008), IEEE Press, June 2008, pp. 279-288, doi: 10.1109/CEC.2008.4630946.