Proceedings of the 18th World Congress
The International Federation of Automatic Control
Milano (Italy) August 28 - September 2, 2011

Online Path Planning for UAV Using an Improved Differential Evolution Algorithm

Xing Zhang*, Jie Chen*, Bin Xin*, Hao Fang*

**School of Automation, Beijing Institute of Technology, Beijing 100081, China;*
Key Laboratory of Complex System Intelligent Control and Decision, Ministry of Education, Beijing, 100081, China
(Tel:86-10-68912464-22;;email:zx218705@163.com, chenjie@bit.edu.cn, brucebin@bit.edu.cn, fangh@bit.edu.cn)

Abstract: This paper presents a 3D online path planning algorithm for unmanned aerial vehicle (UAV) flying in partially known hostile environment. In order to provide a smooth fight route for UAV, the algorithm adopts B-Spline curve to describe UAV's path whose control points are optimized by an improved differential evolution algorithm. The planner gradually produces a smooth path for UAV from starting location to its target. Regarding path planning performance, the proposed improved diferential evolution based planner is compared with its three competitors which are based on classical differential evolution, genetic algorithm and particle swarm optimization, respectively. Numerical simulation demonstrates that the proposed UAV path planner not only produces high-quality feasible flight routes for UAV, but also obviously outperforms its competitors.

Keywords: UAV, online, path planning, differential evolution, B-Spline.

1. INTRODUCTION

As versatile tools, UAVs play a more and more important role in modern warfare. Successfully application of UAVs depends on many factors, such as appropriate combating strategy[e.g. weapon-target assignment (chen et al. 2009a)] and reconnaissance strategy (e.g. path planning schemes). The path planning of UAV (abbr., UAV-PP) aims at searching for the best flight route from its starting location to predefined target. In terms of diverse classification criteria, UAV-PP can be divided into different categories, such as offline UAV-PP vs. online UAV-PP (w.r.t. whether the environment is known in advance or not), three-dimensional (3D) UAV-PP vs. two-dimensional (2D) UAV-PP (w.r.t. whether the flight altitude of UAV changes or not), and single UAV's path planning vs. that of multiple UAVs. Many effective methods have been developed over the past few years to solve the UAV-PP problem. Chasparis et al. (2005) employed linear programming to solve the path planning problem of multi-vehicle coordination in an adversarial environment. Yang et al. (2010) employed a rapidly-exploring random tree (RRT) to solve the path planning problem in an unknown and cluttered environment. Zhang et al. (2006) used a method based on Voronoi and B-spline and Li et al. (2009) proposed an improved A* algorithm to solve the single UAV offline path planning problem. Genetic algorithm (GA) (Ruan et al. 2008), ant colony optimization (ACO) (Wang et al. 2008), and particle swarm optimization (PSO) (Foo et al. 2009) were also applied to the UAV-PP problem. As a burgeoning powerful numerical optimization method, differential evolution is promising to provide a new efficient way to solve the UAV-PP problem. In order to adapt differential evolution to solving online UAV-PP problems, we proposed an improved mutation operator which favours a better tradeoff between exploration and exploitation in the problem space of UAV-PP.

In this paper, UAV is assumed to be equipped with on-board radar to scan its surroundings and GPS to get its position in real time. In general, these fields of enemy radars are known in advance, but the accurate information of environment is unknow. UAV scans its surroundings and generates a local optimal path. UAV flies along current local path and implements next scan. The process is repeated until the UAV reaches its target.

2. PROBLEM FORMULATION OF ONLINE UAV-PP

In the process of UAV's flight to the target, it may face obstacles (e.g. mountains) and threats (e.g. enemy radars). Its route should avoid these obstacles and threats, and at the same time satisfy the UAV's manoeuvrability requirement. At any time, UAV can only sense its surroundings and generate a local path. In this paper, we describe the local path by B-Spline curve with four control points. Meanwhile, we use guide function to direct UAV to its target.

2.1 Environment Model

To simulate UAV-PP, flight environments of UAV are generated artificially. The environment model adopted here contains terrain model and threat model of enemy radars.

2.1.1 Terrain Model

Taking into account UAV's flight condition, terrain model mainly considers mountain terrain and is produced by using the following function for the convenience of simulation.

$$z(x,y) = h \cdot e^{(-\frac{(x-x_0)^2}{x_s^2} - \frac{(y-y_0)^2}{y_s^2})} \quad (1)$$

where (x_0, y_0) is the planar coordinate of the peak center, h is the peak height, and x_s and y_s are the coefficients which reflect the mountain slope.

Due to UAV's considerable flight speed, flying too close to peaks will raise the risk of collisions. For safety, it is usually required that the minimum distance between UAV and peaks should not be less than a predefined safety distance (denoted

by d_s). The probability of collision between UAV and a peak can be approximately measured in the following way:

$$p_m(d) = \begin{cases} 0 & d > d_s \\ \dfrac{d_s - d}{d_s} & d \le d_s \end{cases} \quad (2)$$

where d is the distance between the UAV and the peak. We assume that the damage value is w_m if collision occurs, so the damage the UAV faces is $w_m \cdot p_m(d)$.

2.1.2 Model of Enemy Radars' Threat

We assume the maximum detection radius of an enemy radar is R_{max} and the distance between UAV and the enemy radar is denoted by d . Then the probability that UAV is detected by the radar can be expressed approximately as follows (Zhang et al. 2006).

$$p_r(d) = \begin{cases} 0 & d > R_{max} \\ R_{max}^{\ 4} / (R_{max}^{\ 4} + d^4) & d \le R_{max} \end{cases} \quad (3)$$

We assume the threat value is l_r if the UAV is detected by the radar, so the threat the UAV faces is $l_r \cdot p_r(d)$.

2.2 Flight Constraints

To guarantee the generated path flyable, the planner must take into account the physical limitations of UAV such as turning angle, climbing/diving angle, and flight altitude.

2.3 Online Planning Procedure

UAV scans its surroundings via on-board sensors within a certain range (denoted by R). UAV is blind to the area which is within its scanning range but sheltered by obstacles. The sheltered area will not be considered into the planning space.

Each time UAV completes its scan, it generates a subpath in visible area. Then UAV will move along the subpath and do next scan when it has reached the point which is L from the end of current subpath. The process is repeated until the target point lies within UAV's scanning area. So the whole path is composed of several path segments. After UAV accomplishes next scan, it begins to plan next subpath whose first control point is d_s from the end of current subpath. The following relationship holds:

$$L \ge V \cdot T + d_s \quad (4)$$

where V is the speed of UAV, T is the time we use to program the subpath, and d_s is the safety distance mentioned above. Fig.1 provides an illustration of the online path-planning procedure. The curve from A to B is the current subpath and UAV flies along it. The planner implements next scan at the point p_{scan} , and then it begins to optimize the next subpath. After time T , UAV has reached the point p_{prog} and next subpath has been generated at the same time. So UAV will fly along the new subpath from the point p_{prog} .

2.4 Subpath Link

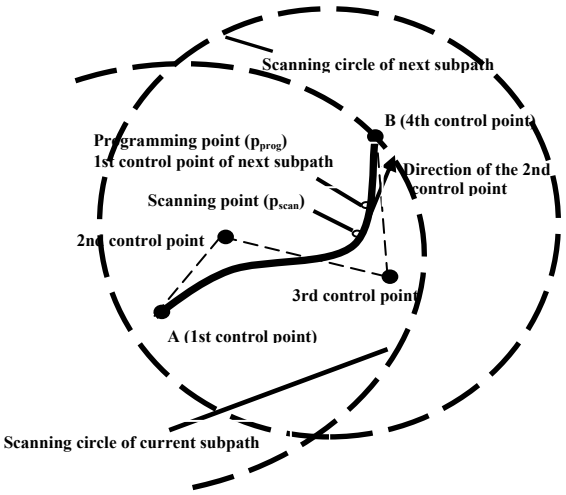

Fig.1. Schematic diagram of subpath planning

To ensure the generated path smooth and flyable, we should make the curvature continuous. Since B-Spline curve is tangential to its control polygon at the starting point and ending point, the starting direction of the curve is determined by its first control point and second control point (Nikolos et al. 2003). As shown in Fig.1, p_{prog} is the first control point of the next subpath, the starting direction of the next subpath is determined by the point p_{prog} and its second control point. Because the slope at point p_{prog} on current subpath is deterministic, the slope at this point on next subpath must be the same. So the second control point of the next subpath must lie in the tangential direction at the point p_{prog} (the direction indicated by the arrow in Fig.1).

2.5 Path Description

We express each control point of a B-Spline curve in the cylindrical coordinate (r, θ, z) . From the analysis in section 2.4, the second control point of each subpath must be selected in required direction. Therefore, we can determine its position as long as we know its distance from the 1st control point along the direction. To facilitate calculation, we limit the range of the latter three control points. The distance from the 1st control point to the second point along the direction is limited within the range $(0, 0.4R)$, the radial value of the third point is limited within the range $(0.4R, 0.8R)$, and the fourth control point is selected on the scanning circle. So the parameters to be optimized are simplified to only six decision variables: the distance from the 1st control point to the second point along the direction r_2 , the cylindrical coordinates of the third point (r_3, φ_3, z_3) and the azimuth and height of the fourth point (φ_4, z_4) .

2.6 Objective Function

In order to generate a feasible path, the constrains of UAV are taken into account in the objective in the form of a penalty function. The objective value of a subpath is computed as follows :

$$f = \sum_{i=1}^{6} a_i \cdot f_i \tag{5}$$

where $f_i (i = 1, 2, ..., 6)$ are the optimization terms described below and $a_i (i = 1, 2, ...6)$ are the corresponding weights.

Term f_1 represents the length of the subpath. For convenience, we compute f_1 as follows:

$$f_1 = \sum_{i=1}^{3} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \tag{6}$$

where x_i , y_i and z_i are the Cartesian coordinates of the ith control point.

To calculate other terms, we select 10 points uniformly from the subpath.

Term f_2 represents the height of the subpath. We compute it by averaging the height of the ten points.

$$f_2 = \frac{1}{10} \sum_{i=1}^{10} z_i \tag{7}$$

Term f_3 is the guide function which guides UAV to fly towards its target. We design it under two different conditions:

Case 1: If the line connecting scanning point and the target point (abbr., S-T) does not pass through any radar threat area, the target is used to guide UAV.

Case 2: If the line S-T passes through threat area of radars, we should guide UAV to fly along the tangential direction of the threat circle. For convenience, we take the radar which is passed through by S-T and nearest to the scanning point as a reference and inspect all the radars which have overlapping detection area with it. As shown in Fig. 2, radar O1 is passed through by the line S-T and nearest to scanning point p_{scan} , and radar O2 has overlapping detection area with it, so we will take into account the two radars to guide UAV.

If the overlapping area between two radars is big, UAV will not pass through the intersection area between the two threats.

As shown in Fig. 2, $\overrightarrow{SM}$ and $\overrightarrow{SN}$ are the candidate directions, but $\overrightarrow{SN}$ is closer to target than $\overrightarrow{SM}$, so we will guide UAV to fly along $\overrightarrow{SN}$. If the overlapping area is small, UAV can pass through the intersection area between the two radars, we will only consider the radar traversed by the line S-T. As shown in Fig.2, in this case $\overrightarrow{SN}$ and $\overrightarrow{SC}$ are candidate directions and we will select the direction along which UAV can arrive at the target faster.

To measure the overlapping degree of two radars, we introduce a new parameter $d_{overlap}$. We define that $d(O1, O2)$ is the distance between two radar centers, $R1$ and $R2$ are the corresponding detection radius. If $d(O1,O2) < R1 + R2 - d_{overlap}$, then we should consider the two threats together; otherwise, we will only consider the radar which is traversed by the line S-T. If there are more overlapping radars, we take the same treatment.

After determining the guide direction, term f3 will be obtained as follows:

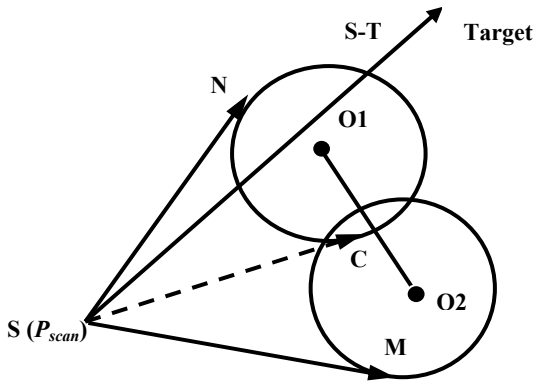

Fig.2. Schematic diagram of the guide direction for case 2

$$f_3 = \left\langle \vec{\alpha}, \vec{\beta} \right\rangle \tag{8}$$

where $\vec{\alpha}$ is the vector from scanning point to the last control point, $\vec{\beta}$ is the vector corresponding to the guide direction. $\left\langle \vec{\alpha}, \vec{\beta} \right\rangle$ is the intersection angle between $\vec{\alpha}$ and $\vec{\beta}$.

We can see that the guide function provides a guidance for UAV's flight direction. If there is no threat between UAV and its destination, it will lead UAV to approach the target; otherwise, the guide function can guide UAV to avoid potential threats in advance.

Term f_4 is the crash cost when UAV flies near peaks

$$f_4 = \sum_{i=1}^{m} \sum_{j=1}^{10} w_i \cdot p_{ij}(d) \tag{9}$$

where m is the number of peaks within the scanning area, w_i is the damage value if collision occurs, and $p_{ij}(d)$ is the probability that UAV collides with the ith peak at the jth point.

Term f_5 is the total threat value of enemy radars:

$$f_5 = \sum_{i=1}^{n} \sum_{j=1}^{10} l_i \cdot p_{ij}(d) \tag{10}$$

where n is the number of enemy radars, l_i is the threat value of the ith radar, $p_{ij}(d)$ is the probability that the UAV is detected by the ith enemy radar at the jth point.

Term f_6 penalizes the nonfeasible curves that violate the constraints. If a subpath does not meet the flight constrains, a punishment will be given, and the value of punishment is proportional to the degree of constraint violations.

$$f_6 = k_1 \cdot \hat{\alpha} + k_2 \cdot \hat{\beta} + f_v \tag{11}$$

where k_1 and k_2 are the penalty coefficients, $\hat{\alpha}$ is the violation value of turning angle, $\hat{\beta}$ is the violation value of the climbing / diving angle, and f_v is the penalty value when there are subpath points that are not within the planning space.

As the ranges of different terms are not the same, the values from term f_1 to f_5 are normalized to the common range (0,100).

3. DIFFERENTIAL EVOLUTION

Differential evolution (DE) is a powerful evolutionary algorithm developed on the framework of genetic algorithm and inspired by Nelder-Mead Simplex method (Price et al. 2005). It is a simple yet efficient population-based algorithm to solve global optimization problems. It generates new individuals based on the individual difference to achieve the evolution of population.

3.1 Classical DE— DE/rand/1/bin

The classical DE variant, DE/rand/1/bin first randomly generates a population $\{\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,D}) \mid i = 1,2,...NP\}$ where D is the dimension of the problem and NP is the population size. After initialization, DE enters a loop of evolutionary operations: mutation, crossover and selection.

Mutation: For each target vector $\mathbf{x}_i^G$ (i=1,2,…,NP, G is the current generation index), a mutant vector is generated as shown in (12).

$$\mathbf{v}_i^G = \mathbf{x}_{r1}^G + F \cdot (\mathbf{x}_{r2}^G - \mathbf{x}_{r3}^G) \tag{12}$$

where $r1$, $r2$ and $r3$ are chosen randomly from the index set $\{1,2,...,NP\}$ and different from i, and F is the scaling factor.

Crossover: A binomial crossover operation operates on the target vector and mutant vector to form the trial vector $\mathbf{u}_i^G$.

$$\mathbf{u}_{i,j}^G = \begin{cases} \mathbf{v}_{i,j}^G & rand_j(0,1) \le Cr \text{ or } j = j_{rand} \\ \mathbf{x}_{i,j}^G & \text{otherwise} \end{cases} \tag{13}$$

where j_{rand} is an integer randomly chosen from 1 to D and generated for each i, and Cr is the crossover probability.

Selection: The selection operation selects the better one from the target vector $\mathbf{x}_i^G$ and the trial vector $\mathbf{u}_i^G$ according to their fitness value, and the better one will become a member of the population in the next generation.

$$\mathbf{x}_i^{G+1} = \begin{cases} \mathbf{u}_i^G & f(\mathbf{u}_i^G) \le f(\mathbf{x}_i^G) \\ \mathbf{x}_i^G & \text{otherwise} \end{cases} \tag{14}$$

3.2 Improved Differential Evolution Algorithm

The biggest drawback of classical differential evolution is its slow convergence in later search stage, and sometimes it is easily trapped in local optima when solving some complex problems. Online path planning requires that the algorithm is concise and easy to implement, and at the same time has desirable convergence performance and robustness. In this paper, we propose a new mutation operator which can achieve a better tradeoff between exploration and exploitation (Chen et al. 2009b).

1) Improved mutation operator

To improve the efficiency of the classical differential evolution, researchers have proposed many differential mutation strategies such as the DE/best/1, DE/best/2, DE/current-to-best/1, and DE/rand-to-best/1 (Price et al. 2005).

Inspired by the idea of combining the respective advantages of different mutation strategies in Xia et al. (2009), the paper presents a new hybrid mutation strategy as follows:

$$\mathbf{v}_i^G = a \cdot \mathbf{x}_{r1}^G + (1-a) \cdot \mathbf{x}_{best}^G + F1 \cdot (\mathbf{x}_{best}^G - \mathbf{x}_{r1}^G) + F2 \cdot (\mathbf{x}_{r2}^G - \mathbf{x}_{r3}^G) \tag{15}$$

We can get different strategies by adjusting the value of parameters. If $a = 1$ and $F1 = 0$, it will be equivalent to DE/rand/1 which has better diversity. If $a = 0$ and $F1 = 0$ it will be equivalent to DE/best/1 which has better convengence rate. If $a = 1$ and $F1 \neq 0$, it will be equivalent to DE/rand-to-best/1. Better tradeoff between exploration and exploitation can be achieved by blending different strategies using different paremeters.

2) Dynamic parameter setting scheme

The scaling factor F controls the amplitude of the difference vector. A larger F results in a wider search range but slower convergence speed. On the contrary, a smaller F benefits faster convergence speed, but DE in this case can be easily trapped into local optima. Parameter a determines the position of the base vector between x_{r1} and x_{best}. A bigger a results in a base vector close to x_{r1} and a smaller a results in a base vector close to x_{best}. When the base vector is near x_{r1}, it is advantageous to take a larger $F2$ and a smaller $F1$ to improve the exploration ability. When the base vector is near x_{best}, it is advantageous to take a smaller $F2$ and a larger $F1$ to enhance the exploitation ability. According to the analysis above, we take the following parameter setting scheme:

$$F1 = k1 \cdot (1 - a), F2 = k2 \cdot a \tag{16}$$

The mutation strategy is transformed into the following form:

$$\mathbf{v}_i^G = a \cdot \mathbf{x}_{r1}^G + (1-a) \cdot \mathbf{x}_{best}^G + k1 \cdot (1-a) \cdot (\mathbf{x}_{best}^G - \mathbf{x}_{r1}^G) + k2 \cdot a \cdot (\mathbf{x}_{r2}^G - \mathbf{x}_{r3}^G) \tag{17}$$

To speed up the convergence in later stage, we gradually increase $k1$ and decrease $k2$ as evolution proceeds. (Xiao et al. 2009).

$$k1 = k_{1\min} + (k_{1max} - k_{1min}) \cdot (\frac{g}{N})^2, k2 = k_{2max} - (k_{2max} - k_{2min}) \cdot (\frac{g}{N})^2 \tag{18}$$

where g is the current generation and N is the maximum number of generations.

There remain two important parameters a and Cr to be settled in the algorithm. Here, we adopt an adaptive scheme for the setting of the two parameters (Qin et al. 2005). Taking a as an example, we assume a is normally distributed in a range with mean a_{mean} and standard deviation 0.2(a_{mean} is initialized to 1). In every generation different values conforming to normal distribution are generated for each individual. After a generation, get all the values of a associated with trial vectors that enter next generation and

update a_{mean} of next generation using the mean value of them. The similar approach is adopted to regulate Cr .

4. NUMERICAL SIMULATION

We simulate the improved DE based online UAV-PP algorithm in Matlab environment. It is assumed that the problem space is 100km × 100km, the starting location is (0,0,0) and the target point is (100,100,0). The flight altitude of UAV is limited within the range (3, 5).

4.1 Influence of Parameters on the Planning Result

In addition to the parameters of the improved DE, there are many other important parameters in the path planner such as the weights in the objective function, $d_{overlap}$, $k1$, and $k2$. The values of the weights reflect the importance of corresponding terms, $d_{overlap}$ affects the choice of flight direction in some conditions, $k1$ and $k2$ determine the penalty degree. All these parameters have effects on the planning result of the algorithm largely. Here we will make an analysis.

The parameter setting of the improved DE is set to: $NP = 60$, $k_{1\min} = 0$, $k_{1\max} = 0.5$, $k_{2\min} = 0.5$, $k_{2max} = 1$. Fig. 3 shows the planning results with different weights and Fig. 4 shows the planning results with different values of $d_{overlap}$.

It is obvious that UAV can choose climbing or bypassing the mountain by weighing the relative size of a_2 and a_3 . When there are multiple radars that have overlapping area, the value of $d_{overlap}$ will determine the flight direction of UAV. As shown in Fig. 3, the UAV chooses bypassing the mountains when we increase the proportion of a_2 . We can see from Fig. 4 that when the value of $d_{overlap}$ is small UAV will bypass all radars, otherwise it may traverse through the intersection of two radars.

4.2 Performance Comparison

In order to test the performance of the improved DE algorithm proposed in this paper, we also employ genetic algorithm (GA), particle swarm optimization algorithm (PSO), and classical differential evolution (DE) to solve the online UAV-PP problem.

In this paper, two different environments are established. Environment 1 contains small number of sparsely distributed obstacles and threats and environment 2 is just the opposite. For each environment, we run each algorithm 20 times and statistic the mean and standard deviation of fitness value from starting point to target point. Numerical results are shown in Table 1. We can see that compared with PSO, GA and the classical DE, the improved DE proposed in this paper leads to better result (lower mean and standard deviation).The classical DE performs better than PSO and GA. For environment 1, since the problem is simple, we can get better results only if the time is enough. We can get feasible result with with T=3s (T is the time mentioned in Section 2.3), and the performance with T=5s has much improvement than it. But for environment 2, because of its complexity, the results of PSO and GA are not stable even when T=5s, the improved DE and classical DE can get stable result with T=5s, and the improved DE performs better than classical DE. When T=3s, the superiority of the improved DE is much evident. We can conclude that the improved DE proposed in this paper has feasible convergence performance and robustness for the problem. We got similar results in other circumstances. To save space, the results are not presented here.

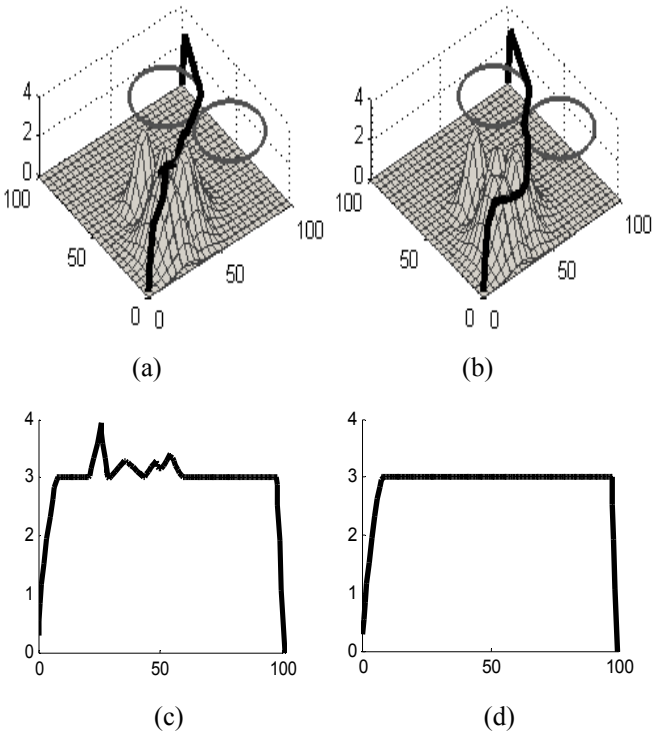

Fig. 3. 3D path displays and corresponding vertical profiles. (a) Planning result with $a_1 = 0.1$, $a_2 = 0.15$, $a_3 = 0.15$, $a_4 = 0.3$, $a_5 = 0.3$, $a_6 = 1$; (b) Planning result with $a_1 = 0.1$, $a_2 = 0.25$, $a_3 = 0.05$, $a_4 = 0.3$, $a_5 = 0.3$, $a_6 = 1$; (c) Vertical profile w.r.t. (a); (d) Vertical profile w.t.r. (b).

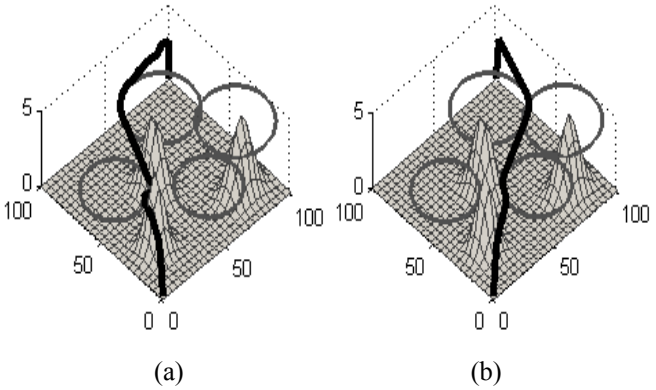

Fig. 4. 3D path with different value of $d_{overlap}$. (a) Planning result with $d_{overlap} = 2$; (b) Planning result with $d_{overlap} = 6$.

Table 1. Performance comparison of four algorithms on test environments

Environ / Time \ Algorithm / Performance		Proposed DE		Classical DE		PSO		GA	
		Mean Cost	St Dev	Mean Cost	St Dev	Mean Cost	St Dev	Mean Cost	St Dev
Environ 1	T=3s	88.3	4.9	97.9	8.2	94.4	8.2	125.4	27.4
	T=5s	76.6	3.7	79.4	3.7	84.5	7.6	110.7	16.5
Environ 2	T=3s	123.2	9.0	146.0	18.3	157.2	23.3	166.7	22.8
	T=5s	117.5	8.0	123.3	11.1	146.7	27.6	148.5	16.4

5. CONCLUSIONS

UAV's path planning is a complex optimization problem involving many constraints. Especially in the online path planning, the sensors can only get the local environment information, so we can only implement local optimization. The proposed method based on improved DE represents the paths using B-Spline curve which can ensure the generated path smooth and flyable. The simulation result shows that the improved DE algorithm can get feasible path and has stronger robustness and better convergence performance than other optimization algorithms (GA, PSO, and classical DE).

ACKNOWLEGEMENT

This work was supported by the National Science Fund for Distinguished Young Scholars (Grant No.60925011) and National Natural Science Foundation of China (Grant No. 60805035/F0306).

REFERENCE

Chasparis, G.C. and Shamma, J.S. (2005). Linear-programming-based multi-vehicle path planning with adversaries. In Proceedings of American Control Conference (ACC), pp.1072-1077. Portland, Oregon. Jun 8-10.

Chen, J., Xin, B., Peng, Z.H., Dou, L.H., and Zhang, J. (2009). Evolutionary decision-makings for dynamic weapon-target assignment problem. Science in China, Series F: Information Sciences, 52(11), pp.2006-2018.

Chen, J., Xin, B., Peng, Z.H., Dou, L.H., and Zhang, J. (2009). Optimal contraction theorem for exploration-exploitation tradeoff in search and optimization. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 39(3), pp.680-691.

Foo, J.L., Knutzon, J., Kalivarapu, V., Oliver, J., and Winer, E. (2009). Path planning of unmanned aerial vehicles using B-Splines and particle swarm optimization. Journal of Aerospace Computing Information and Communication, 6(4), pp.271-290.

Li, X., Xie, J., Cai, M.Y., Xie, M., and Wang, Z.K. (2009). Path planning for UAV based on improved heuristic A* algorithm. In Proceedings of International Conference on Electronic Measurement & Instruments, pp.3488-3493. Beijing. Aug 16-19.

Nikolos, I.K., Valavanis, K.P., Tsourveloudis, N.C., and Kostaras, A.N. (2003). Evolutionary algorithm based offline/online path planner for UAV navigation. IEEE Transactions on Systems Man and Cybernetics Part B: Cybernetics, 33(6), pp.898-912.

Price, K.V., Storn, R.M., and Lampinen, J.A. (2005). Differential Evolution: A Practical Approach to Global Optimization. Springer, Berlin.

Qin, A.K. and Suganthan, P.N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. In Proceedings of IEEE Congress on Evolutionary Computation, pp. 1785-1791. Sep 2-5.

Ruan, D., Montero, J., Lu, J., Martinez, L., Dhondt, P., and Kerre, E.E. (2008). Multiobjective path planner for UAVs based on genetic algorithms. In Proceedings of International Conference on Fuzzy Logic and Intelligent Technologies in Nuclear Science, pp.997-1002. Madrid, Spain. Sep 21-24.

Wang, Z.H., Zhang, W.G., Shi, J.P., and Han, Y. (2008). UAV route planning using multiobjective ant colony system. In Proceedings of IEEE Conference on Cybernetics and Intelligent System, pp.797-800. Chengdu, China. Sep 21-24.

Xia, H.M. and Zhou, Y.Q. (2009). Improved differential evolution strategy optimization algorithm for multiple hump functions. Computer Engineering and Applications, 45(32),pp.41-44.

Xiao, S.J. and Zhu, X.F. (2009). A modified fast and highly efficient differential evolution algorithm. Journal of Hefei University of Technology, 32(11),pp.1700-1703.

Yang, K., Gan, S.K., and Sukkarieh, S. (2010). An efficient path planning and control algorithm for RUAV's in unknown and cluttered environments. Journal of Intelligent and Robotic Systems: Theory and Applications, 57(1-4), pp.101-122.

Zhang, Y.F., Zhang, A., Zhang, Z.Y., and Zhang, J.L. (2006). Planning algorithm of tactics flight path. Journal of Traffic and Transportation Engineering, 6(4), pp.84-87.