Path planning for UAV based on quantum-behaved particle swarm optimization

Yangguang Fu [a], Mingyue Ding [a,b], Chengping Zhou [*a], Chao Cai [a] , Yangguang Sun [a]

[a] Institute for Pattern Recognition and Artificial Intelligence, State Key Laboratory for Multi-spectral Information Processing Technologies, Huazhong University of Science and Technology, Wuhan 430074, China; [b] College of Life Science and Technology, "Image Processing and Intelligence Control" Key Laboratory of Education Ministry of China, Huazhong University of Science and Technology, Wuhan 430074, China

ABSTRACT

Based on quantum-behaved particle swarm optimization (QPSO), a novel path planner for unmanned aerial vehicle (UAV) is employed to generate a safe and flyable path. The standard particle swarm optimization (PSO) and quantum-behaved particle swarm optimization (QPSO) are presented and compared through a UAV path planning application. Every particle in swarm represents a potential path in search space. For the purpose of pruning the search space, constraints are incorporated into the pre-specified cost function, which is used to evaluate whether a particle is good or not. As the system iterated, each particle is pulled toward its local attractor, which is located between the personal best position (*pbest*) and the global best position (*gbest*) based on the interaction of particles' individual searches and group's public search. For the sake of simplicity, we only consider planning the projection of path on the plane and assume threats are static instead of moving. Simulation results demonstrated the effectiveness and feasibility of the proposed approach.

Keywords: path planning, UAV, inertial weight, particle swarm optimization, quantum-behaved particle swarm optimization

1. INTRODUCTION

Path planning for unmanned aerial vehicles (UAV) is one of the most important parts of Mission Planning. Path planning is to generate a path between an initial prescribed location and desired destination having an optimal or near-optimal performance under specific constraint conditions.

In recent years, path planning for UAV in both military and civilian applications has been a heavily research area. The existing approaches of path planning for UAV can be classified as follows: 1) graphic-based algorithms, such as Voronoi diagram [1] and probabilistic roadmap [2]; 2) heuristic algorithms, such as A* [3], Sparse A* [4], and D* [5]; 3) evolutionary computation algorithms [6-8]. The disadvantages of graphic-based algorithms and heuristic algorithms and the advantages of evolutionary computation algorithms are discussed in Ref. [7] for the path planning oriented applications.

Particle swarm optimization is a new evolutionary computing algorithm developed by Kennedy and Eberhart [9]. PSO is widely used in the function optimization with multi-variable, neural network training and fuzzy system control [10]. Many significant ameliorations of PSO algorithm are proposed to improve the performance of the algorithm and two of them are discussed in this paper. Sun [11, 12] put forward the quantum-behaved particle swarm optimization (QPSO) algorithm model from the perspective of Quantum Mechanics view. QPSO was tested on some benchmark functions such as Sphere function, Rosenbrock function. Experiment results provided by Sun showed that QPSO outperforms PSO.

In this paper, a novel quantum-behaved particle swarm optimization (QPSO) algorithm is applied to generate a path for UAV, and PSO with the inertial weight known as Standard PSO [13] is selected for comparison with the quantum-behaved PSO. The constraints [14] that the qualified path has to satisfy and include but not limit to: 1) minimize the path length to the target; 2) minimize the UAV' exposure to threats. Unfortunately, these two primary minimizations

*zhouchp@sina.com; phone 027-87544512

MIPPR 2009: Medical Imaging, Parallel Processing of Images, and Optimization Techniques,
edited by Jianguo Liu, Kunio Doi, Aaron Fenster, S. C. Chan, Proc. of SPIE Vol. 7497, 74970B
© 2009 SPIE · CCC code: 0277-786X/09/$18 · doi: 10.1117/12.832476

are impossible to be met simultaneously. We can obtain a shorter path with less regard of the exposure to radar, or the better exposure path with a long path length. Therefore, a suitable selection of the weight coefficient provides a trade-off between path length and threat avoidance.

The primary objective of this paper is to describe a general path planning strategy based on quantum-behaved particle swarm optimization (QPSO). Usually, the resultant path is composed of path's projection on the XY plane and path's altitude. For the sake of simplicity, we only consider planning the projection of path on the plane and assume threats are static rather than moving. The standard particle swarm optimization (PSO) and quantum-behaved particle swarm optimization (QPSO) are presented and compared through a UAV path planning application. Simulation results are given to demonstrate the effectiveness and feasibility of the proposed approach.

The remainder of this paper is organized as follows: Section 2 described the application of PSO and QPSO algorithms to the UAV path planning problem. Experimental results were presented in Section 3. Finally, the paper was concluded in Section 4.

2. PATH PLANNING BASED ON PSO AND QPSO

2.1 Structure of particles

Each possible solution in the problem space can be represented by an individual in the population. The individual in the swarm is called particle. In fact, a particle with arbitrarily small mass and volume is a potential path in the search space. Consequently, the particles' representation builds a one to one mapping between particles and candidate paths. The terms 'particle', 'individual', 'path' and 'solution' are used interchangeably in this paper. For comparison, the particles of PSO and QPSO are encoded in the same way.

The population is denoted by a matrix $X=[X_1, X_2, \dots, X_m]^{\mathrm{T}}$, where m is the size of the population. The vector $X_i=[x_{i1}, \dots, x_{in}, x_{i,n+1}, \dots, x_{i,2n}]$ is the ith particle's position with $2n$ dimensions, x_{ij} is the jth dimension of the ith particle's position ($i=1, \dots, m$; $j=1, \dots, 2n$). The first half of X_i represents the abscissa value of each path node on the ith path while the rest represents the corresponding ordinate value of the first part. Thus, the jth path node on the ith path is specified by the coordinates $(x_{ij}, x_{i,j+n})$. The start point and endpoint for all particles are the same, meaning that the value of dimension is twice the size of the number of the path nodes on the generated path except the first node (start point) and the last one (endpoint).

At the beginning, each particle is initialized randomly in the search space although some more sophisticated initialization strategy (such as Sobol sequence generator, nonlinear simplex method, and skeletonization approach) can improve the overall performance of algorithms. The length of a particle, i.e., the dimension, which is equal for each particle, is a parameter determined by user. Note that the value of dimension makes a trade-off between accuracy of the path and computational efficiency.

2.2 Evaluation of particles

The performance of each particle is measured by a predefined fitness function, which in general is application-dependant. The cost function put forward below is employed to evaluate whether a particle is good or not. Consequently, the fitness value of a particle is equal to the its cost.

The representation of cost function is important to the path planning for UAV. From the path planning perspective, there are two types of costs corresponding to two main constraints which have to be considered. One is the cost of path length while the other is the cost of threats. In addition, the maneuverability of UAV is also needed to be added. The cost of turn angle introduced into the cost function will make the path smoother and more flyable.

To combine these three types of costs, the cost function for a specific path X_i is weighted by the path length, threats and turn angel:

$$J(X_i) = w_1 J_{length}(X_i) + w_2 J_{threat}(X_i) + w_3 J_{trun\_angle}(X_i)\,, \tag{1}$$

where w_1,w_2 and w_3 are the weight coefficients of path length, threats and turn angel respectively and they are satisfied $w_1+w_2+w_3=1$.

For a given path X_i, the coordinates of the path nodes can be represented as $\{(x_{i1}, x_{i,n+1}), (x_{i2}, x_{i,n+2}), \dots, (x_{in}, x_{i,2n})\}$. The path length cost J_{length} is defined as the sum of all path segments from the start point to the endpoint,

$$J_{length}(X_i) = d_{i,0} + d_{i,n+1} + \sum_{j=1}^{n-1}[(x_{ij} - x_{i,j+1})^2 + (x_{i,n+j} - x_{i,n+j+1})^2]^{1/2},$$

where $d_{i,0}$ denotes the distance from the start point to the first node $(x_{i1}, x_{i,n+1})$ on the path X_i while $d_{i,n+1}$ represent the distance between the last node $(x_{i,n}, x_{i,2n})$ on the path X_i and the end point.

The threats cost J_{threat} is computed using the following rules. Without loss of generality, the different threats are represented by the circles with different centre point and radius. The length of the radius indicates the cover range of a threat. For a given path segment, the threats cost is proportion to the length of path segment which is contained in threat circles.

$$J_{threat}(X_i) = \sum_{k=1}^{N} s_k (L_{i,0}^{(k)} + L_{i,n+1}^{(k)} + \sum_{j=1}^{n} L_{ij}^{(k)}),$$

where N is the number of threats, and s_k indicates the threat intensity of the kth threat. $L_{i,0}^{(k)}$ and $L_{i,n+1}^{(k)}$ represent the length of the segment including the start point and the endpoint contained in the kth threat circle respectively. $L_{ij}^{(k)}$ is the length of the jth segment from node $(x_{ij}, x_{i,n+j})$ to node $(x_{i,j+1}, x_{i,n+j+1})$ of the ith path X_i contained in the kth threat circle. If the jth segment of the ith path X_i is out of the kth threat, then $L_{ij}^{(k)}$ is zero. In other words, UAV is allowed to intersect a point on the boundary of a threat but with high threat cost to fly through the interior of a threat.

In the view of physical limitation of UAV, it can only turn with an angle less than or equal to a predetermined maximum turning angle. Therefore, the turn angle constraint should be enforced at each path node where UAV is unlikely to be making a sharp turn at such a point. Considering two adjacent path segments AB and BC formed by three successive path nodes $A(x_{ij}, x_{i,n+j})$, $B(x_{i,j+1}, x_{i,n+j+1})$ and $C(x_{i,j+2}, x_{i,n+j+2})$, the turn angel formed by these two segments at point B is defined to be the complementary angle of $\angle ABC$ and can be calculated by the cosine theorem. Then, the turn angle cost is measured by the difference between the current turn angle and the predetermined maximum turning angle $theta_{max}$. When the current turn angle is less than or equal to $theta_{max}$, the cost of this turn angle is set to zero.

$$J_{turn\_angle}(X_i) = c_{i,0} + c_{i,n+1} + \sum_{j=1}^{n-2} const(\theta_{current}^{j} - \theta_{\max}),$$

$$\theta_{current}^{j} = \arccos\left(\frac{(x_{i,j+1} - x_{i,j})(x_{i,j+2} - x_{i,j+1}) + (x_{i,n+j+1} - x_{i,n+j})(x_{i,n+j+2} - x_{i,n+j+1})}{\sqrt{[(x_{i,j+1} - x_{i,j})^2 + (x_{i,n+j+1} - x_{i,n+j})^2][(x_{i,j+2} - x_{i,j+1})^2 + (x_{i,n+j+2} - x_{i,n+j+1})^2]}}\right),$$

where $c_{i,0}$ and $c_{i,n+1}$ are the first and the last turn angle cost respectively.

2.3 Velocity and position updating of particles in PSO

Many revised versions of PSO algorithm are proposed to improve the performance of PSO algorithm. Two most significant ameliorations are the introduction of inertial weight [13] and quantum-behaved mechanism. PSO with an inertial weight is known as the Standard PSO. The inertial weight plays the role of balancing the global search (large inertial weight) with the local search (small inertial weight).

Each particle adjusts its position in the search space according to its own flying experience and companions' flying experience. The ith particle is associated with a velocity vector $V_i=[v_{i1}, v_{i2}, \dots, v_{i,2n}]$. During the search process, every particle keeps track of the personal best (*pbest*) position $P_i=[p_{i1}, p_{i2}, \dots, p_{i,2n}]$ found by itself and the global best (*gbest*) position $P_g=[p_{g1}, p_{g2}, \dots, p_{g,2n}]$ achieved by any particle in the population. During the iteration procedure, the velocity and position of the particle are updated according to Eq. (2) :

$$\begin{cases} v_{ij}^{k+1} = wv_{ij}^{k} + c_1 r_{1i}^{k}(p_{ij}^{k} - x_{ij}^{k}) + c_2 r_{2i}^{k}(p_{gj}^{k} - x_{ij}^{k}) \\ x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1}, \ (i = 1,2,...,m; j = 1,2,...,2n) \end{cases}, \tag{2}$$

where v_{ij}, whose representation is similar to x_{ij}, p_{ij} and p_{gj}, is the jth dimension of the ith particle's velocity and is usually confined to the closed interval $[-v_{\max}, v_{\max}]$ to prevent the explosion of the particles. Coefficient r_1 and r_2 are two pseudo-random, scalar values drawn from a uniform distribution on the unit interval. The superscript *k* denotes the *k*th iteration. The acceleration coefficients c_1 and c_2 are 2.0 for almost all applications [15]. *w* is the inertial weight.

Shi and Eberhart [16] proved that a time decreasing inertial weight could significantly improve the performance of the algorithm. The variation of inertial weight in PSO is analogous to that of temperature in simulated annealing. If the inertial weight is equal to 1, then the Standard PSO turned into the original PSO. The maximum velocity is often set to be the dynamic range of each variable on each dimension in Standard PSO, whereas about 10-20 percent of the dynamic range of each variable on each dimension in original PSO [10, 17]. It is believed that the maximum velocity is not even needed in the PSO with constriction factor, but can get better solutions by setting the maximum to be the dynamic range of each variable on each dimension [17, 18].

The position of *pbest* and *gbest* are computed from Eqs. (3) and (4) respectively,

$$P_i^{k+1} = \arg\min_{1\le j\le k+1} J(X_i^j) = \begin{cases} P_i^k, & if\ J(P_i^k) \le J(X_i^{k+1}) \\ X_i^{k+1}, & if\ J(P_i^k) > J(X_i^{k+1}) \end{cases}, \tag{3}$$

$$P_g^{k+1} = \arg\min_{1\le i\le m} J(P_i^{k+1}) = \begin{cases} P_g^k, & if\ J(P_g^k) \le J(P_i^{k+1}) \\ P_i^{k+1}, & if\ J(P_g^k) > J(P_i^{k+1}) \end{cases}, \tag{4}$$

where *J*( ) is the cost function defined in Eq. (1).

2.4 Position updating of particles in QPSO

Deform the velocity updating formula as shown in Eq. (2), we have:

$$\begin{cases} v_{ij}^{k+1} = wv_{ij}^k + (c_1 r_{1i}^k + c_2 r_{2i}^k)[(\frac{c_1 r_{1i}^k}{c_1 r_{1i}^k + c_2 r_{2i}^k} p_{ij}^k + \frac{c_2 r_{2i}^k}{c_1 r_{1i}^k + c_2 r_{2i}^k} p_{gj}^k) - x_{ij}^k] \\ x_{ij}^{k+1} = x_{ij}^k + v_{ij}^{k+1},\ (i = 1,2,...,m; j = 1,2,...,2n) \end{cases}. \tag{5}$$

In Eq. (2), the velocity is adjusted by adding two terms: one is the randomly weighted distance between *pbest* and the *i*th particle's current position, the other is the randomly weighted distance between the swarm's *gbest* and the *i*th particle's current position, while the velocity is updated by adding only one term as shown in Eq. (5). Because c_1, c_2 are deterministic constant and r_1, r_2 are stochastic constant, Eq. (5) can be rewritten as:

$$\begin{cases} v_{ij}^{k+1} = wv_{ij}^k + (\xi_1 + \xi_2)[(\frac{\xi_1}{\xi_1 + \xi_2} p_{ij}^k + \frac{\xi_2}{\xi_1 + \xi_2} p_{gj}^k) - x_{ij}^k] \\ x_{ij}^{k+1} = x_{ij}^k + v_{ij}^{k+1},\ (i = 1,2,...,m; j = 1,2,...,2n) \end{cases}. \tag{6}$$

Define $\xi = \xi_1 + \xi_2$ and $\xi_1 / (\xi_1 + \xi_2) = \zeta$, then

$$v_{ij}^{k+1} = wv_{ij}^k + \xi(q_{ij}^k - x_{ij}^k),\ \text{where}\ q_{ij}^k = \zeta p_{ij}^k + (1-\zeta) p_{gj}^k. \tag{7}$$

Eqs. (5)-(7) are mathematically identical to Eq. (2). Clerc and Kennedy [18] has indicated that as the system iterates, each particle is pulled toward its local attractor Q_i=[q_{i1},q_{i2},…,$q_{i,2n}$] based on the interaction of particles' individual searches and the group's public search to ensure convergence. And the local attractor Q_i is the stochastic weighted mean of the personal best position (*pbest*) and global best position (*gbest*).

QPSO algorithm [11, 12] assumes that there is one-dimensional delta potential well on each dimension at the local attractor point Q_i, and every particle in the swarm has quantum behavior. The quantum state of a particle is depicted by a wave function instead of the position X_i and velocity V_i used in PSO. The probability density function of the position that each particle fly to can be deduced from the Schrodinger equation:

$$f(x_{ij}^{k+1}) = \frac{1}{l_{ij}^k} \exp(-\frac{2\,|\,x_{ij}^{k+1} - q_{ij}^k\,|}{l_{ij}^k}),$$

where q_{ij} and $l_{ij}/2$ are the location parameter and scale parameter of the symmetric exponential distribution accordingly. It is easily to be proved that the mathematical expectation of x_{ij} is equal to q_{ij}.

However, the wave function or probability density function provides nothing more than the state of a particle in quantized search space, while the position of a particle in classical solution space is needed. Using Monte Carlo simulation method, the measurement of the position for each particle from the quantum state to the classical one can be implemented and the position of the ith particle can be determined as below:

$$x_{ij}^{k+1} = q_{ij}^{k} \pm l_{ij}^{k} / 2\ln(1/u)\,, \tag{8}$$

where q_{ij} and l_{ij} are the jth dimension of the ith particle's local attractor and delta potential well characteristic length, and u is a uniformly distributed pseudorandom number on the unit interval.

A high light of the QPSO is the introduction of mean best (*mbest*) position of the population on the basis of the *pbest* and *gbest* in PSO. The mean best position can be calculated by the following equation:

$$mbest^{k} = \frac{1}{m}\sum_{i=1}^{m} P_i^{k} = (\frac{1}{m}\sum_{i=1}^{m} p_{i1}^{k}, \frac{1}{m}\sum_{i=1}^{m} p_{i2}^{k}, \cdots, \frac{1}{m}\sum_{i=1}^{m} p_{i,2n}^{k})\,. \tag{9}$$

After obtaining the mean best position, the characteristic length of delta potential well can be expressed by the particle's personal current position [12] using the following equation:

$$l_{ij}^{k} = 2b\left|mbest_j^{k} - x_{ij}^{k}\right|, \tag{10}$$

where b is the contraction-expansion coefficient, which is similar to the inertial weight in the PSO.

Substitute Eqs. (7) and (9)-(10) into (8), the position of particle can be updated as follows:

$$x_{ij}^{k+1} = \zeta p_{ij}^{k} + (1-\zeta) p_{gj}^{k} \pm b\left|\sum_{i=1}^{m} p_{ij}^{k} / m - x_{ij}^{k}\right| \ln(1/u)\,. \tag{11}$$

2.5 Description of QPSO algorithm

The QPSO algorithm is implemented in detail as follows:

1) choose appropriate parameters such as population size *popsize*, particle dimensions and maximum iteration;
2) input the environmental information of war field and the selected parameters;
3) initialize the particles randomly in the solution space to generate *popsize* initial coarse paths;
4) evaluate each particle based on the aforementioned cost function defined in Eq. (1);
5) compute the personal best position *pbest* of each particle and global best poison *gbest* of the population using Eqs. (3) and (4);
6) compute the local attractor from Eq. (7);
7) compute *mbest*, which is defined as the average point of all particles' *pbest* positions, according to Eq. (9);
8) renew the position of each particle in the swarm from Eq. (11);
9) repeat Steps 4) to 8) until the maximum number of iteration allowed has been reached;
10) output the *gbest* as the optimal path when the loop ends.

3. EXPERIMENT RESULT

The path planning for UAV based on standard PSO algorithm and QPSO algorithm are both implemented in a Matlab programming environment. Suppose the war field to be a square region with a size of 700×700, the resolution is 100m×100m. There are six synthetic threats represented by six circles. Their centers and radii are (450,500,90), (300,150,80), (480,280,60), (200,450,70), (150,200,100) and (300,300,50) respectively. (450,500,90) means the center of the threat is (450,500) while the radius is 90, The coordinates of the start point and the endpoint are (40,500) and (600,300) respectively. The '*' in Figure 1 is the path node on the generated path.

The population size, dimensions and maximum number of iterations are set to 50, 10 and 200 respectively. Note that the dimension has a very close relationship with accuracy of the path and computational efficiency. Other conditions being equal, the larger the dimension, the more accurate the path will be, and the more complicate of the problem. Therefore, too large value of dimension may result in failure of planning. For PSO，the inertial weight coefficient varied from 0.9 to 0.4 linearly while for QPSO the contraction-expansion coefficient is dynamically decreasing linearly from 1.0 to 0.5. The weight coefficients in the cost function are all set to be 1/3.

For the purpose of comparison, the best paths generated by PSO (the dashed line) and by QPSO (the solid line) during 50 trails are showed in Figure 1. It is easy to find that the solid line generated by QPSO is shorter and smoother than the dashed one generated by PSO. In other words, the path produced by QPSO is with less cost value and more flyable. The relationship between number of iterations and cost value corresponding to the paths generated in Figure 1 is showed in Figure 2. After about 35 iterations, the QPSO has found the global optimal solution while PSO is trapped in the local optimal solution since the 140th iteration. QPSO converges to the global optima more quickly than PSO. During 50 times experiments, the minimum, maximum, average and standard deviation of cost value are showed in Table 1. In addition, the simulation results showed that PSO and QPSO are both computationally inexpensive in term of memory requirements and speed demands. The average time for per iteration is about 30 ms.

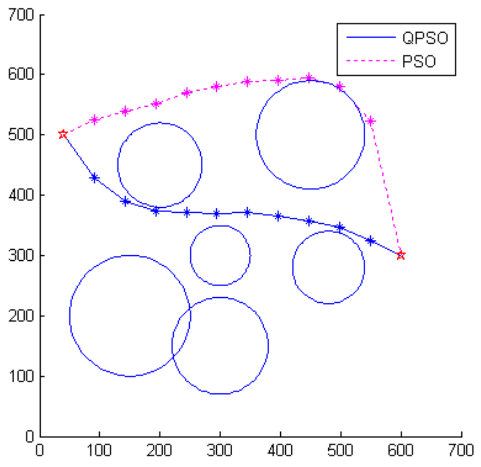

Fig. 1. Best Paths

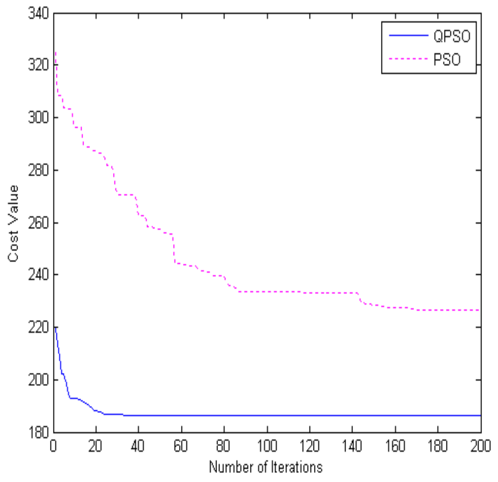

Fig. 2. Number of iterations and cost value

Table 1. Performance comparison of PSO and QPSO.

Algorithm	Min cost value	Max cost value	Average cost value	Standard deviation
PSO	226.4007	1.0170e+003	600.4782	267.8980
QPSO	186.1255	190.8166	188.6239	1.7612

4. CONCLUSION

A novel QPSO algorithm is used to implement the path planning for UAV. The effectiveness and performance of PSO and QPSO was compared based on the cost value of generated path. As have been demonstrated from the experiment results that the QPSO is prone to make progress toward improve the robustness of algorithm while PSO may lead to premature convergence and become trapped in local optima. The standard deviation also proved the robustness of QPSO. In the iteration, PSO is unable to move far enough to reach a better position in the solution space because the particle is limited to a finite sampling space, which decreased the global search ability of the PSO algorithm. In other words, since the convergence of particles is implemented in the form of trajectory in PSO, particles in PSO can only fly in a limited number of directions, while the particles in QPSO have the nature of aggregation state and can appear in any position with a certain probability distribution. Consequently, the particles in QPSO, by its nature, can search a much larger portion of the solution space than PSO. It can reach the global optimal solution much faster than PSO. The global search capability of QPSO is validated by our simulation results.

REFERENCES

[1] McLain, T. and Beard R., "Coordination variables, coordination functions, and cooperative-timing missions", Journal of Guidance Control and Dynamics 28(1), 150-161 (2005).

[2] Pettersson, P.O. and Doherty P., "Probabilistic roadmap based path planning for an autonomous unmanned helicopter", Journal of Intelligent and Fuzzy Systems 17(4), 395-405 (2006).

[3] Melchior, P., et al., "Consideration of obstacle danger level in path planning using A* and Fast-Marching optimisation: comparative study", Signal processing 83(11), 2387-2396 (2003).

[4] Szczerba, R., et al., "Robust algorithm for real-time route planning", IEEE Transactions on Aerospace and Electronic Systems 36(3), 869-878 (2000).

[5] Stentz, A., "Optimal and efficient path planning for partially-known environments. in Robotics and Automation", Proceedings of IEEE International Conference on Robotics and Automation, 3310-3317 (1994).

[6] Yi, M., Ding M. and Zhou C., "3D route planning using genetic algorithm", Proceedings of SPIE International Symposium on Multispectral Image Processing, 92-95 (1998).

[7] Zheng, C., et al., "Evolutionary route planner for unmanned air vehicles", IEEE Transactions on Robotics 21(4), 609-620 (2005).

[8] Zhu, H., et al., "Path planner for unmanned aerial vehicles based on modified PSO algorithm", Proceedings of IEEE International Conference on Information and Automation, 541-544 (2008).

[9] Kennedy, J. and Eberhart R., C., "Particle swarm optimization", Proceedings of IEEE International Conference on Neural Networks, 1942-1948 (1995).

[10] Eberhart, R., C. and Shi, Y., "Particle swarm optimization: developments, applications and resources", Proceedings of the Congress on Evolutionary Computation, 81-86 (2001).

[11] Jun, S., Bin F. and Xu W., B., "Particle swarm optimization with particles having quantum behavior", Proceedings of the Congress on Evolutionary Computation, 325-331 (2004).

[12] Jun, S., Xu W., B. and Bin F., "A global search strategy of quantum-behaved particle swarm optimization", Proceedings of IEEE Conference on Cybernetics and Intelligent Systems, 111-116 (2004).

[13] Shi, Y. and Eberhart R., C., "A modified particle swarm optimizer", Proceedings of IEEE World Congress on Computational Intelligence, 69-73 (1998).

[14] Beard, R., et al., "Coordinated target assignment and intercept for unmanned air vehicles", IEEE Transactions on Robotics and Automation 18(6), 911-922 (2002).

[15] Shi, Y. and Eberhart R., C., "Empirical study of particle swarm optimization", Proceedings of the Congress on Evolutionary Computation, 1945-1950 (1999).

[16] Shi, Y. and Eberhart R. C., "Parameter selection in particle swarm optimization", Proceedings of the 7th International Conference on Evolutionary Programming VII, 591-600 (1998).

[17] Eberhart, R.,C. and Shi Y., "Comparing inertia weights and constriction factors in particle swarm optimization", Proceedings of the Congress on Evolutionary Computation, 84-88 (2000).

[18] Clerc, M. and Kennedy J., "The particle swarm - explosion, stability, and convergence in a multidimensional complex space", IEEE Transactions on Evolutionary Computation 6(1), 58-73 (2002).