2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

Multiple UAV Path Planning using Anytime Algorithms

P. B. Sujit and Randy Beard

Abstract— **We address the problem of generating feasible paths from a given start location to a goal configuration for multiple unmanned aerial vehicles (UAVs) operating in an obstacle rich environment that consist of static, pop-up and moving obstacles. The UAVs have limited sensor and communication ranges, when they detect a pop-up or a moving obstacle that is in the collision course with the UAV flight path, then it has to replan a new optimal path from its current location to the goal. Determining optimal paths with short time intervals is not feasible, hence we develop anytime algorithm using particle swarm optimization that yields paths whose quality increases with increase in available computation time. To track the given path by the anytime algorithm in 3D, we developed a new uav guidance law that is based on a combination of pursuit guidance law and line of sight guidance law from missile guidance literature. Simulations are carried out to show that the anytime algorithm produces good paths in a relatively short time interval and the guidance law allows the UAVs to track the generated path.**

I. Introduction

Recently, unmanned aerial vehicles (UAVs) have been used for various search and surveillance missions. A mission usually consists of generating a path that the UAV uses as a reference and follows it. A path can be described as a *set of way-points* and the *path planning* problem is to produce a set of valid way-points taking the environmental and physical constraints of the UAVs. Designing a path planner for multiple UAVs flying at low altitudes in obstacle rich environments that contains static, pop-up and moving obstacles is difficult. The difficulty further increases as the environment is in three dimensions.

For example, consider a mission with multiple UAVs where, the initial and goal positions for each UAV is given and the UAVs need to plan optimal de-conflicted paths in the presence of static and dynamic obstacles. When the UAVs detect a change in environment then they have to re-plan an optimal path to their goal configuration. Generating optimal paths in a short interval of time is difficult hence they need to produce close to optimal paths. Since, the quality of the generated path depends on the time available to the UAV for computation, therefore hence we need to develop anytime algorithms. Anytime algorithms produce solutions for any given computational time interval and the solutions get better with increase in available computation time. In this paper, we develop a path planning algorithm based on particle swarm optimization (PSO), whose solution quality increases with increase in the computation time for multiple UAVs traveling in an obstacle rich environment. The choice of PSO is based on its low computational overheads and faster solution convergence compared to GA and other evolutionary algorithms [1]. Qin et. al. [3] used PSO to improve the path generated by the Dijkstra shortest path algorithm for a partitioned environment in 2D with static obstacles. In 3D it is difficult to partition the configuration space taking pop-up and dynamic obstacles into account. In this paper, we present 3D path planning algorithm using PSO and take pop-up and dynamic obstacles into account.

The path planning problem in three dimensional environment without any obstacles was addressed in [4], [5] and with only static obstacles in [6]- [8]. Saunders et al. [9] used RRTs for UAV path planning but do not consider the quality of the path produced. Ferguson and Stentz [10] develop anytime RRTs for a single robot traveling in an obstacle rich environment but in two dimensions. When RRTs are used in three dimensions, due to availability of larger solution space, the RRTs can produce high sub-optimal paths, hence RRTs are not suitable when close to optimal solutions are needed. All these papers address the issue of path planning, but address only a subset of the issues that our mission confronts.

In this paper, we develop a path planning algorithm for multiple UAVs traversing from a start location to the goal configuration in the presence of static and dynamics obstacles. When a new obstacle is detected, then depending on the time available to generate a new path we execute the anytime algorithm using PSO that produces a path avoiding the obstacle. In order to track the desired path, we developed a new guidance law for the UAVs in three dimension. The guidance law is based on a combination of pursuit and line of sight guidance laws from the missile guidance literature.

II. Problem Formulation

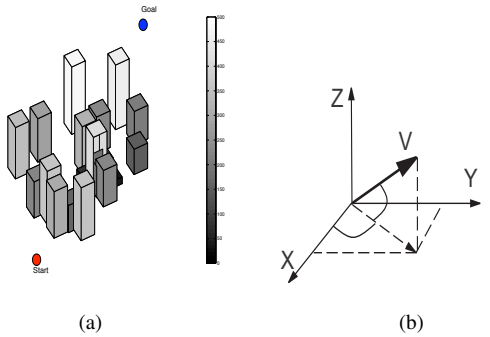
A. Scenario

We consider a mission where multiple UAVs need to travel from their assigned start location to goal configurations

This work was partially funded by the National Science Foundation under Information Technology Research Grant CCR-0313056 and by NASA under STTR contract number NNA04AA19C to Scientific Systems Inc, and Brigham Young University and by the Air Force Office of Scientific Research award no. FA9550-04-0209.

P. B. Sujit is a Research Scientist in the Department of Electrical and Computer Engineering, University of Porto, Portugal. sujit@fe.up.pt

Randy Beard is a Professor in the Department of Electrical and Computer Engineering, Brigham Young University, Provo, Utah - 84602. beard@byu.edu



(a) (b)

Fig. 1. (a) UAV has to determine its optimal de-conflicting path from the start location to the goal in the presence of static and pop-up obstacles. The color of the obstacles determines their height as shown in the color bar. (b) UAV coordinate geometry in three dimensions.

through an environment consisting of obstacles of different sizes and heights. The UAVs have to design an optimal path avoiding the obstacles and also other UAVs during flight. The environment also consists of pop-up obstacles and if the flight path is on a collision course with the pop-up obstacles then the UAV has to replan a new optimal path from its current location to the goal. A typical scenario for one UAV is shown in Figure 1(a).

B. Assumptions and constraints

We assume that the UAVs are subjected to limited sensor and communication ranges, physical constraints and have constant velocity during flight. Since the UAVs have limited sensor range, they can detect the pop-up obstacles only within its sensor range. If the obstacle is in the collision course of the flight path, then the UAV has to change the path to avoid collision. Depending on the velocity of the UAV, it can predict the time to collide T_i^c (where i represents the i^{th} UAV), and it needs to determine a new path before T_i^c.

We assume that the UAVs have physical constraints in terms of maximum descent and accent rates, and maximum turn radius constraints. Although the time to collide is T_i^c, due to the UAV physical constraints, it has to start its collision avoidance maneuver much earlier than T_i^c, which is denoted as T_i^m (time to maneuver) and $T_i^m < T_i^c$. Therefore the time given to the anytime algorithm to produce a feasible path is T_i^m.

The UAVs are also subject to limited communication ranges and hence they cannot inform the other UAVs about their planned path that may be in conflict. In this case also, the UAVs determine their T_i^c and T_i^m and plan a new path given the time constraint. Designing path planning algorithms with these constraints in three dimensions is a difficult problem that we address in this paper.

C. Solution concept

UAVs like Predators or Global Hawk have been used for surveillance missions. For these missions a pre-defined path is generated and the UAV has to follow the path. This path is a sequence of way-points that the UAV uses as a reference path. Similarly, we also consider a path to be a set of way-points and optimize their locations to obtain an optimal path length. However, generating an optimal path with a short interval of time and in obstacle rich environment is difficult.

Let $\mathcal{S}$ be the environment that contains M obstacles and each obstacle is a cube (as shown in Figure 1(a)). The j^{th} obstacle is represented as $O_j, j = 1, \ldots, M$ and located at x_j^o, y_j^o with width w_j^o, length l_j^o and height h_j^o. The free space $\mathcal{F} = \mathcal{S} \setminus \mathcal{O}$ is the space available for the UAVs to generate the optimal path, where $\mathcal{O} = \bigcup_{j=1}^{M} O_j$ and $'\setminus'$ is the set difference operator.

Each UAV is represented as $A_i, i = 1, \ldots, N$ with start location as $S_i = \{x_i^s, y_i^s, z_i^s\}$ and goal configuration as $G_i = \{x_i^f, y_i^f, z_i^f\}$. We assume that a path P_i of agent A_i from S_i to G_i consists of $Q-1$ number of way-points. Therefore, path $P_i = \{W_i^0, \ldots, W_i^Q\}$ and each way-point is represented by a tuple $W_i^k = \{x_i^w, y_i^w, z_i^w\}, k = 0, \ldots, Q$. When $k = 0$, the way-point W_i^0 represents the start configuration S_i and when $k = Q$, way-point W_i^Q represents the goal configuration G_i. In order to obtain an optimal path, the way-points $W_i^k, k = 1, \ldots, Q-1$ should be optimized such that the path cost C_i is minimized. While constructing the cost function we should also take the vertical climb and descent of the way-points. When the UAV is climbing, it has to spend additional fuel and hence its cost should be increased, on the other hand when the UAV is on the descent phase, it spends less fuel therefore the segment cost should be reduced. Hence, the path cost C_i is evaluated as

$$C_i = \sum_{k=0}^{Q-1} (||W_i^k - W_i^{k+1}|| + k_c V_c - k_d V_d) \tag{1}$$

where $||W_i^k - W_i^{k+1}||$ is length of path segment between way-points W_i^k and W_i^{k+1}, V_c and V_d represent the vertical climb and descent of the segment, while k_c and k_d represent the gains associated with climb and descent phases.

We use particle swarm optimization (PSO) technique to determine the optimal path P_i. At the beginning of the mission we assume that the location of the obstacles, the number of obstacles and their shapes are known. With this information the UAV plans a path. If the start locations of the UAVs are within the communication range then they can share their plans. If the plans have potential conflict that can result in a collision then the UAVs need to coordinate with each other to change their plans and avoid collision. Let T_m^i be the time allowed to compute a new optimal path before a collision can occur. In that case, we execute the PSO algorithm with the updated environment for T_m^i seconds. The solution obtained from the time constrained optimization is

the new path P_i. The determination of the solution using PSO is described in the next section.

III. Particle Swarm Optimization Solution

A. Basics of PSO

PSO is a population based stochastic optimization technique developed by Eberhart and Kennedy [2] that simulates the social behavior of bird flocks, fish schools, etc. Each particle in a swarm is a potential solution in the search space. The particle adjusts its velocity according to its own flying experiences and its flock's experiences.

Assume the optimization problem to be of D-dimension, then each particle in the swarm Ω can be represented as $X_l = (x_{l1}, \ldots, x_{lD}), l = 1, \ldots, \Omega$. The best previous position attained by the particle is represented as $B_l = (b_{l1}, \ldots, b_{lD})$, and the velocity of the particle is $V_l^p = (v_{l1}, v_{l2}, \ldots, v_{lD})$. The best global position achieved by the swarm is represented by symbol Γ, and the iteration number is represented as f. The particles in the swarm are updated according to the following equation:

$$v_{ld}^{f+1} = \lambda(\omega v_{ld}^f + c_1 r_1 (b_{ld}^f - x_{ld}^f) + c_2 r_2 (b_{\Gamma d}^f - x_{ld}^f)) \quad (2)$$

$$x_{ld}^{f+1} = x_{ld}^f + v_{ld}^{f+1} \quad (3)$$

where $d = 1, \ldots, D$, r_1 and r_2 are uniformly distributed random numbers between $[0,\ 1]$, c_1 and c_2 are positive constants representing cognitive and social parameters, ω is the inertia weight, and λ is the constriction factor. The role of inertia weight ω is to create a balance between global exploration and local exploitation. Initially, it is necessary to explore the search space and then reduce ω as the solution reaches the optimum value [1]. The constants c_1 and c_2 aid in convergence of the solution. The random parameters r_1 and r_2 are used to maintain the diversity of the population, while the constriction factor controls the effect of velocity on the particles.

B. Implementing PSO for path planning

The PSO algorithm has the ability to explore higher dimensional solution spaces to generate optimal solutions given sufficient number of iterations. However, due to lack of sufficient time to achieve optimal solution, the UAVs have to settle for the best available solution based on the constrained computation time. We design an algorithm to generate the best solutions for a given computation time T_i^m.

We consider a swarm Ω number of particles to determine the path. Each particle has a dimension $D = 3(Q-1)$, that is, the path consists of $Q-1$ way-points and each way-point has three components. Therefore, the dimension of the particle is $3(Q-1)$. The path formed by each particle is evaluated to determine the cost. If any segment of the path intersects an obstacle then that path cost is forced to infinity otherwise it is determined using Equation 1. The implementation of the anytime algorithm using PSO is described using Algorithm 1.

Algorithm 1 Path planning algorithm using anytime algorithm based on PSO

```
% Function to determine PSOpath
1: function PSOpath(S, S_i, G_i, T_i^m)
2: Initialize: swarm Ω, numParticles, λ, ω, c_1, c_2
3: bestParticleValue ←∞; bestParticle ← []
4: while t ≤ T_i^m do
5:    Pval ← evaluateSwarm(Ω, numParticles);
6:    [minParticleValue, minParticleIndex]← min(Pval)
7:    if minParticleValue ≤ bestParticleValue then
8:       bestParticleValue ← minParticleValue;
9:       bestParticle← Ω(minParticleIndex);
10:   end if
11:   update Ω using Equations 2 and 3
12:   update t
13: end while
14: return(bestParticle)
```

Algorithm 2 Function to evaluate the particles

```
% Evaluate each particle of the swarm
1: Function cost = evaluateSwarm(Ω, numParticles)
2: for l = 1 to numParticles do
3:    cost(l) ← 0;
4:    if Any ξ_d ∉ S, ξ_d ∈ Ω(l,:) then
5:       cost(l) ← ∞;
6:    else if Any ξ_d ∈ O, ξ_d ∈ Ω(l,:) then
7:       cost(l) ← ∞;
8:    else
9:       for d = 1 : 3 : 3(Q − 1) do
10:         if (line([Ω(l, (d : d+2))], [Ω(l, (d+3 : d+5))]) ∈
            O then
11:            cost(l) ← ∞;
12:         else
13:            cost(l) ← ||Ω(l, (d : d + 2)), Ω(l, (d + 3 : d +
               5))|| + k_c V_c + k_d V_d;
14:         end if
15:      end for
16:   end if
17: end for
```

We initialize the Swarm Ω and the constants c_1, c_2, ω, and λ. The PSO algorithm uses *evaluateSwarm* function (described in Algorithm 2) to determine the cost of a path that is represented by a particle. For each evaluation, we check if all the dimensions of the l^{th} particle are in the environment or not (Algorithm 2, line 4). If the value of any $\xi_d \notin \mathcal{S}$ then we consider the particle to be invalid and make its value as infinity. We also carry out other verifications on the validity of the particle before determining the cost of the path, these checks are (i) whether a particle is inside an obstacle ($\xi_d \in \mathcal{O}$, line 6) and (ii) if the line segment joining the two way-points does not intersect any obstacle

(Algorithm 2line 10). If the particle passes these checks then we evaluate the cost using Equation 1. We determine the minimum value particle for the swarm and check if the minimum value is less than the previously recorded cost by the swarm. If the current minimum value is less than previous best then we modify the bestParticleValue to the current minimum value and the minimum attained particle in bestParticle (Algorithm 1, line 8 and 9). If the computation time is available then we update the swarm using Equations 2 and 3. This process continues till t reaches T_i^m and we use the path for the agent A_i represented by bestParticle. Once the path is generated, the UAV has to follow the way-point segments of the path. In order to carry out this operation we developed a guidance law for the UAV which is described in the next section.

IV. Guidance Law

The UAV has to navigate along the path taking the kinematic constraints into account in three dimensions given by the anytime PSO algorithm. We develop a guidance which is a combination of pursuit guidance law and Line of Sight (LOS) guidance law from the missile guidance literature. The choice of using pursuit and LOS guidance laws is motivated from the fact that pursuit allows the UAVs to catch the desired way-points fast and is simple to implement but its trajectory may not be along the LOS. Hence, we also use LOS guidance law that will steer the UAV towards the LOS.

We use a five UAV state model that can take the course angles, flight path angles and the height into account during its flight. These state equations are given in Equation 4 and the geometry of the coordinate system is shown in Figure 1(b).

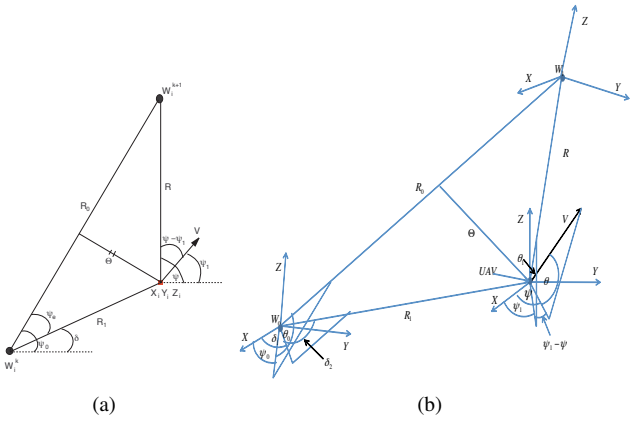
$$\begin{aligned} \dot{x}_i &= V_i \cos\psi_i \cos\theta_i \\ \dot{y}_i &= V_i \sin\psi_i \cos\theta_i \\ \dot{z}_i &= V_i \sin\theta_i \\ \dot{\psi}_i &= \frac{g}{V_i}\tan\phi_i\eta_i \\ \dot{\theta}_i &= \frac{g\cos\theta_i}{V_i}(\eta_i - 1) \end{aligned} \tag{4}$$

where V_i represents velocity, ψ_i the heading angle, θ_i the pitch angle, g the gravitational acceleration, ϕ_i the roll angle and η_i the load factor of the UAV A_i.

Consider the engagement geometry of the UAV A_i in two-dimension as shown in Figure 2(a). Assume that the UAV is located at the x_i, y_i, z_i and it has to follow the path formed by the way-points W_i^k and W_i^{k+1} with $z_i = z_i^k = z_i^{k+1} = 0$. For the UAV to move towards the way-point W_i^{k+1}, it has to apply vertical acceleration and horizontal acceleration to move along the LOS (give by R_0).

The vertical acceleration is determined using the pursuit guidance law and given as:

$$a_p = -K(\psi_1 - \psi) \tag{5}$$



(a) (b)

Fig. 2. (a) UAV engagement geometry in two-dimension (b) UAV engagement geometry in three-dimensions.

where, K is the gain, ψ_1 is the current heading angle, and ψ is the desired heading angle. The horizontal acceleration is determined using the LOS guidance law and given as:

$$a_l = R_1 \sin(\psi_o - \delta) \tag{6}$$

The combined acceleration that A_i has to apply to move towards the way-point W_i^{k+1} is

$$a = -K(\psi_1 - \psi) - R_1 \sin(\delta - \psi_o) \tag{7}$$

The analysis applied to a two-dimensional system can now be extended to a three dimension coordinate frame for the UAV. Unlike in 2-D, now the UAV has to consider the heading and the pitch angles. The coordinate geometry of the UAV is shown in Figure 2(b). Since, the UAV is moving in three dimensions, we also need to apply acceleration in the horizontal (a_h) and in the vertical (a_v) directions. These accelerations are given as:

$$a_h = -K_1(\psi_1 - \psi) - R_1 \sin(\delta_1 - \psi_o) \tag{8}$$

$$a_v = -K_2(\theta_1 - \theta) - R_2 \sin(\delta_2 - \theta) \tag{9}$$

where, $\psi_1, \psi, \delta_1, \delta_2, \theta,$ and θ_1 are determined using the coordinate geometry figure as shown in Figure 2(b).

For implementing the five-state model of the UAV, we need to determine the load factor η_i and the roll angle ϕ_i. From the three dimensional missile engagement geometry, we can determine these values as

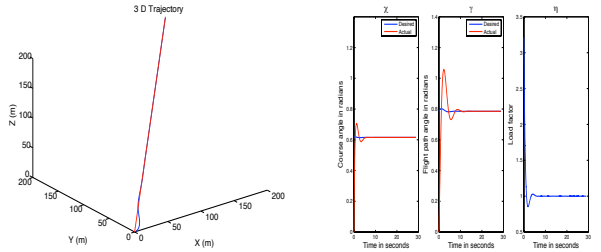
$$\dot{\psi}_1 = \frac{g}{V}\tan\phi\eta = \frac{a_h}{V\cos\theta_1} \tag{10}$$

$$\dot{\theta}_1 = \frac{g\cos\theta_1}{V}(\eta - 1) = \frac{a_v}{V} \tag{11}$$

solving Equations (10) and (11), we get

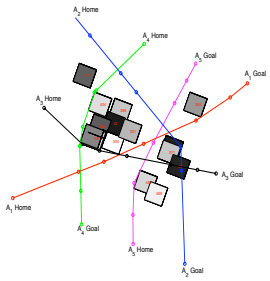
$$\begin{aligned} \eta &= \frac{a_v}{g\cos\theta_1} + 1 \\ &= -K_2(\theta_1 - \theta) - R_2\sin(\delta_2 - \theta_o) \end{aligned} \tag{12}$$

$$\phi = \tan^{-1}\frac{a_h}{\eta g\cos\theta_1} \tag{13}$$



(a) (b)

Fig. 3. (a) UAV 3-D path for a given way-point segment with $K_1 = 10$, and $K_2 = 10$ (b) Plots of heading angle, pitch angle and the load factor.



(a)

Fig. 4. The initial start and the goal locations of the UAVs with their paths.

Using the values obtained from equations (12) and (13) we simulated an example scenario where the UAV is initially located at [0,0,0], and has to follow the path formed by the way-point segment [0,0,0] and [200,200,200]. Figure 3(a) shows the trajectory of the UAV following the way-point path. From the figure we can see that the UAV settles along the path in less than 8 seconds. This settling time is due to low gains on the accelerations. In Figure 3(b) we can see that the heading and pitch angles settle to the desired values determined by the way-points of the path. In the figure we can also see that the load factor settled to 1, which means that the UAV has attained its desired pitch angle. Thus it follows the given path. We can significantly decrease the settling time with increase with acceleration gains that correspondingly requires high load factor. Hence, by tuning these gains we can achieve the desired performance of the guidance law.

Apart from the gain, the initial pitch and heading angles of the UAV also contribute to the behavior of the trajectory. For a given path by the PSO, the UAV has to follow the path using the developed guidance law. During the transition from one way-point to another, the UAV trajectory may be off the path and there is a possibility that the UAV can collide with the obstacles. Hence, before starting the mission, we fix the gains and determine the worst deviation and use this deviation in the PSO path cost. If the deviated path intersects with an obstacle then that path is not considered. Thus, using the developed guidance law we can track any given path.

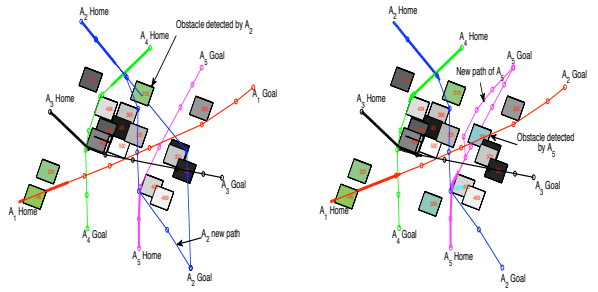
V. Simulation Results

The validity of the anytime path planning algorithm using PSO is determined using simulations. We consider an environment of 1000m $\times$ 1000m that contains 15 static obstacles and 5 dynamic obstacles that appear randomly. All the obstacles are randomly generated with random heights and the maximum height of any obstacle is 500m (see Figure 4(a)). The grey scale shows the heights of the obstacle in the environment. We assume the velocity of the UAV is constant at 12m/s and can detect any obstacle within a sensor range of 100m. Therefore, the UAV has less than 100/12 seconds to produce the path as the obstacle can pop-up inside the sensor range of 100m. The pop-up obstacles appear at random interval of times and they are represented in a different color to distinguish between the static and the pop-up obstacles (see Figure 5(a)).

We consider five UAVs who need to plan their paths from start to goal configurations without colliding against the static, pop-up, and moving obstacles (other UAVs). A path is considered as a sequence of way-point segments and when one way-point is reached the next way-point is considered. If the UAV detects a pop-up obstacle which is in the collision course then it has to generate a new path. The number of way-points that the new path may consists of depends on the remaining number of way-points to traverse.

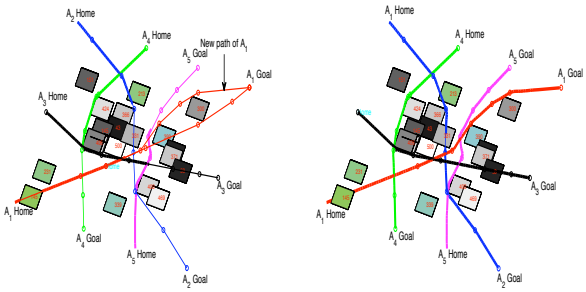
Initially, each UAV is allowed to generate its path for a computational period of 60 seconds. Since, the UAVs are at the base station, we can allow sufficient amount of time to compute the best path, but to know the feasibility of the PSO algorithm in providing reasonable solutions, the computation time was limited to 60 seconds. The paths of each UAV is shown in Figure 4(a). The parameters used the UAVs to generate paths are $K_1 = 10$, and $K_2 = 10$ for the guidance law, $c_1 = 0.5, c_2 = 0.5, \lambda = 0.5, \Omega = 20$ and $\omega = 0.95$ for the PSO algorithm, and $k_c = 2$ and $k_d = 0.5$ for the cost function.

Selecting the number of way-points is a trade-off for a mission that depends on the number of obstacles present in the environment and the available computational time. If the number of way-points are low then the algorithm will generate paths that travel above the obstacle and not into the environment. The computational complexity of the algorithm is also cheap. However, increasing the number of way-points increases the complexity of the algorithm that will result in low cost paths. Another difficulty in the selection of the way-points is that, the available computational time is dependent on the current UAV sensing environment, hence having large number of way-points may not result in better solution than the low number of way-points. After carrying out several experiments with different number of way-points, we selected five way-points, whose computational complexity is reasonable to the produced paths in relatively short intervals (around 5-6 seconds).



(a) (b)

Fig. 5. (a) A_2 encounters a pop-up obstacle and the new replanned path (b) A_5 detects a pop-up obstacle on its route and the new re-planned path.



(a) (b)

Fig. 6. (a) The new path of A_1 that was replanned after encountering a pop-up obstacle (b) Paths of all the UAVs from start location to the goal.

At time $t = 23$s, A_2 detects a pop-up obstacles as shown in the Figure 5(a). In the same figure we can see the trajectories of all the five agents and also the pop-up obstacles that have appeared. The UAV A_2 has 6.59 seconds to compute a new path before reaching the maneuvering way-point. The computed new path also shown in the Figure 5(a). The path of A_2 in the plot appears to cross over the obstacle with height 331 which is not true since the third way-point height is 368m (before the obstacle)and the fourth way-point height is 331.86m (after the obstacle). Although the path is sub-optimal but this is the best path that the algorithm was able to generate in the given computation time.

Next the UAV A_3 encounters a pop-up obstacle as shown in Figure 5(b) at time $t = 32$ seconds and it has 6.71 seconds to compute a new path. The new path generated by the path planning algorithm is shown in the same figure. The UAV A_1 detects a pop-up obstacle at time $t = 50$ and has 7.53 seconds to compute the path. The new path generated by A_1 is shown in Figure 6(a). All the UAVs reach their respected destination by 126 seconds and their paths are shown in Figure 6(b).

The pursuit plus LOS guidance law was able to track the generated path without colliding into any of the objects. Thus allowing us to use the guidance law for more complex scenarios. Due to the stochastic nature of the PSO, the solution may be different for different runs with the same initial conditions. In order to determine the variation in the simulations, we ran the setup for 50 times. The results shown in Figures 4(a)-6(b) were the same for 48 out of 50 runs, while for two runs the solutions were different from that shown in the figures 4(a)-6(b). Hence, the anytime PSO can produce consistently good results even with the stochastic nature of the PSO.

VI. Conclusions

We proposed an anytime algorithm using PSO for multiple UAV path planning problem in an environment that consist of both static and pop-up obstacles. The algorithm can improve its quality of solution given sufficient amount of the time. In order to track the path produced by the anytime PSO, we developed a new guidance law that combines the pursuit guidance law and the LOS guidance law. From the simulations we can observe that the anytime algorithm has generated paths that are reasonably good for a short interval of computation time.

References

[1] K. E. Parsopoulos and M. N. Vrahatis: Recent approaches to global optimization problems through particle swarm optimization, *Natural Computing*, Springer, 2002, Vol. 1, pp. 235-306.

[2] R. C. Eberhart and J. Kennedy: A new optimizer using particle swarm theory, *Proc. of the Symposium on Micro Machine and Human Science*, Piscataway, NJ, 1995, pp. 39-43.

[3] Y. Qin, D. Sun, N. Li, and Y. Cen: Path planning for mobile robot using the particle swarm optimization with mutation operator, *Proc. of the Internation Conference on Machine Learning and Cybernetics*, Aug 2004, pp. 2473-2478.

[4] S. Kanchanavally, R. Ordonez, C.J. Schumacher: Path Planning in Three Dimensional Environment Using Feedback Linearization, *Proc. of the American Control Conference*, Mineapolis, MN, June 2006.

[5] M. Shanmugavel, A. Tsourdos, R. Zbikowski and B. A. White: 3D path planning for multiple UAVs using pythagorean hodograph curves, *Proc. of the AIAA Guidance, Navigation and Control Conference and Exhibit*, Hilton Head, SC, August 2007, AIAA 2007-6455.

[6] N. Vandapel, J. Kuffner and O. Amidi: Planning 3-D path networks in unstructured environments, *Proc. of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005, pp. 4624 - 4629.

[7] Y. Kuwata and J. How: Three dimensional receding horizon control for UAVs, *Proc. of tha AIAA Guidance, Navigation, and Control Conference*, Aug 2004, AIAA-2004-5144.

[8] I. Hasircioglu, H. R. Topcuoglu and M. Ermis: 3-D path planning for the navigation of unmanned aerial vehicles by using evolutionary algorithms, *Proc. of the Conference on Genetic and Evolutionary Computation*, Atlanta, GA, 2008, pp. 1499–1506.

[9] J. Saunders, B. Call, A. Curtis, R. Beard, and T. McLain: Static and dynamic obstacle avoidance in miniature air vehicles, *Proc. of that AIAA Infotech@Aerospace Conference and Exhibit*, Arlington, VA, Sep. 2005, AIAA-2005-6950.

[10] D. Ferguson and A. Stentz: Anytime RRTs, *Proc. of the IEEE International Conference on Intelligent Robots and Systems*, Beijing, China, Oct. 2006, pp.5369-5375.