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

Cooperative Defensive Surveillance using Unmanned Aerial Vehicles

A. Matlock, R. Holsapple, C. Schumacher, J. Hansen and A. Girard

Abstract— **The paper presents a cooperative control algorithm for a team of Unmanned Aerial Vehicles (UAVs) used in the surveillance of the area around a military base to protect against potential threats. The UAVs are required to search an area of interest, while efficiently allocating their time between zones of varying degrees of importance. Irregular routes are preferred, to reduce the ability of an adversary to predict the patrol routes of the UAVs. In this paper, we consider a team of potentially heterogeneous, dynamically constrained UAVs with constant velocities. The problem is approached as a finite horizon optimization to account for possible alarms as they occur. This approach seeks to optimize the amount of information obtained by the UAVs, with surveillance of pop-up alarms a high but not sole priority. Particle Swarm Optimization (PSO) is used to search the control space and optimize the reward function. This approach guarantees feasible trajectories, without smoothing, in addition to unpredictable paths.**

I. Introduction

Aerial surveillance is a key technology for a wide range of civilian and military applications. Civilian uses include, but are not limited to: forest fire monitoring, wildlife tracking, oil spill detection, traffic monitoring, and search and rescue. Military applications are many and varied, including both strategic and tactical uses, with examples ranging from target detection, identification, and tracking, to perimeter monitoring, area surveillance, and intelligence gathering.

The algorithmic results presented in this paper are independent of any specific UAV platform. Hence, the UAVs may be heterogeneous or homogeneous, but must work cooperatively. By cooperatively, we mean that the UAVs optimize a common reward function. The UAVs could fly any of uncountably many feasible paths to perform the desired surveillance mission, and the extra degrees of freedom will be exploited to remove predictability from the vehicle flight paths. The inclusion of pop-up targets and/or alarms as a stochastic event makes the cooperative planning problem more complex, and also contributes to the unpredictability of vehicle routes. Due to the extreme complexity of the resulting optimization problem, and the requirement to compute new routes quickly when pop-up alarms occur, computing an exact solution to the optimization problem is not feasible. Instead, a heuristic approach will be used to compute a suitable and acceptable sub-optimal solution within the allowable computation time.

A. Matlock is a graduate student in the Department of Aerospace Engineering at the University of Michigan, Ann Arbor, MI 48109, USA amatloc@umich.edu

R. Holsapple is a Mathematician at the Air Force Research Laboratory's Control Science Center of Excellence, Wright-Patterson AFB, OH 45433, USA raymond.holsapple@wpafb.af.mil

C. Schumacher is a Senior Aerospace Engineer at the Air Force Research Laboratory's Control Science Center of Excellence, Wright-Patterson AFB, OH 45433, USA corey.schumacher@wpafb.af.mil

J. Hansen is an Aeronautical Engineer at the Air Force Research Laboratory's Control Science Center of Excellence, Wright-Patterson AFB, OH 45433, USA john.hansen@wpafb.af.mil

A. Girard is an Assistant Professor of Aerospace Engineering at the University of Michigan, Ann Arbor, MI 48109, USA anouck@umich.edu

A. Nomenclature

We begin by listing the nomenclature that will be used throughout the paper. For ease of notation, we also note that all vectors are considered to be row vectors; column vector transpose is not indicated. In addition, functional dependence on time is suppressed in equations where the notation might seem cumbersome.

$$\begin{aligned}
\text{number of UAVs} &= N \in \mathbb{N} \\
\text{time horizon} &= T \in \mathbb{N} \\
\text{position of vehicle } i &= (x_i, y_i) \in \mathbb{R}^2 \\
\text{orientation angle of vehicle } i &= \psi_i \in \mathbb{R} \\
\text{control input for vehicle } i &= u_i \in \mathbb{R} \\
\text{UAV footprint radius of vehicle } i &= r_i \in \mathbb{R} \\
\text{velocity of vehicle } i &= v_i \in \mathbb{R} \\
\text{environment reward function} &= E : \mathbb{R}^3 \to \mathbb{R} \\
\text{instantaneous reward for all UAVs} &= R : \mathbb{R}^{2N+1} \to \mathbb{R}
\end{aligned}$$

B. Literature Review

The literature in UAV cooperative control is vast and suggests many ways to control a team of cooperative UAVs for aerial surveillance and area coverage. Past research for aerial surveillance can be grouped into at least two categories. One approach looked at the surveillance problem as a coverage problem, and optimized the coverage of a given area, such as the lawnmower pattern. In [1], an exhaustive search algorithm, similar to a lawnmower pattern, is developed to search for targets. In this algorithm the UAVs fly in lanes and turn into subsequent lanes after the entire current lane has been traversed. Ahmadzadeh et al. [2] considered the problem of optimizing the coverage of an area while satisfying hard constraints, such as initial and final positions. Delima [3] considered optimizing coverage while using metrics such as dynamic coverage, heterogeneity of coverage, and energy consumption. Others [4], [5], [6], [7], [8] have also investigated similar techniques to optimize coverage.

Another aspect of aerial surveillance focuses on control algorithms that observe areas of highest interest in a region. Girard et al. [9] and Beard et al. [10] develop control algorithms to track the perimeter of a known area of interest.

978-1-4244-4524-0/09/$25.00 ©2009 AACC

In this paper, we consider a team of UAVs used in the surveillance of a two dimensional space (the base and/or some surrounding region), while still responding (visiting) to pop-up alarms during the surveillance period. Our algorithm surveils the entire region over time, while emphasizing the areas that have a high potential for intrusions. The algorithm also generates irregular paths to reduce the ability of an intruder to predict the paths of the UAVs. Due to the inherent randomization of the technique, there is no guarantee of complete coverage of a given area in finite time.

This paper is organized as follows. Section II describes the objective and the problem formulation. Section III briefly gives an overview of PSO. Section IV describes the notations used in the algorithm, and Section V describes the algorithm itself. Finally, results are given in Section VI for several example problems, and conclusions are found in Section VII.

II. Problem Formulation

A team of UAVs is assigned for surveillance of a military base and the surrounding area to protect against potential threats. We define surveillance as simply the monitoring of a given area. The goal of the UAVs is to find feasible paths through the regions of most importance to a human operator at a given time *t*. The flight trajectory control sequence for each vehicle is determined by solving the optimization problem described below. A reward function quantifies the amount of information the UAVs have accumulated over a given time T, with $R(t)$ denoting the reward at time instant $t \in [0, T]$. The sensor footprints of the cameras are assumed to be cones that intersect the plane with radius r_i, centered directly below the UAVs. When a region is within a UAV's footprint, the reward over the entire region is collected and is instantaneously set to zero, so the UAVs will tend not to revisit previously viewed areas. Conversely, when regions are outside the UAVs' footprint, the reward grows.

We begin by defining a couple of new variables. Let

$$p_i(t) = (x(t)_i, y(t)_i) \tag{1}$$

be the cartesian coordinates of vehicle i, in an inertial frame. We will omit the dependence on time from this point on. Define

$$p = [p_1 \;\; p_2 \;\; \cdots \;\; p_N] \tag{2}$$

to describe the position coordinates of all N UAVs. Then, the problem can be stated simply. Find

$$\max \int_0^T R(t)\, dt, \tag{3}$$

subject to:

$$f(p_i) = \{\hat{p_i} = (\hat{x_i}, \hat{y_i}) \mid d(\hat{p_i}, p_i) \leq r_i\}, \tag{4}$$

$$\widetilde{R}_i(p_i, t) = \int_{f(p_i)} E(p_i, t)\; dA, \tag{5}$$

$$R(p, t) = \sum_{i=1}^{N} \widetilde{R}_i(p_i, t). \tag{6}$$

In Equation (3), the optimization is performed over all feasible trajectories. In Equation (4), we define the footprint of the i-th vehicle, which is centered at p_i, where $d(\hat{p_i}, p_i)$ denotes the Euclidean metric space. Equation (5) describes the computation of the local reward information at time t for vehicle i. In this equation, $\hat{p}$ is the variable the integration is computed over; it ranges over all points in the set $f(p_i)$. Moreover, $E(x, y, t)$ is the environment reward function that describes the relative importance of viewing a given region of the base at time *t*, where *x,y* are cartesian coordinates. Equation (6) is the total reward for all N UAVs at time t.

The vehicles are dynamically constrained by the Dubins' vehicle model where ψ is the angular orientation relative to the positive x-axis. The UAVs are assumed to fly at constant altitude, implying the dynamics of the UAVs evolve in a plane, with a constant positive velocity and minimum turning radius enforced by the control input constraints (see below). When the vehicles are notified of a target, the vehicles should visit the site of the target in finite time. The system dynamics are given by the following equations:

$$\dot{x}_i = v_i \cos\psi_i, \tag{7}$$

$$\dot{y}_i = v_i \sin\psi_i, \tag{8}$$

$$\dot{\psi}_i = u_i. \tag{9}$$

In addition the system has the following constraints:

$$\dot{\psi}_i \leq \omega_i, \tag{10}$$

$$\bigcup_{i=1}^{N} \bigcup_{t\epsilon[0,T]} f(p_i(t)) = \Gamma, \tag{11}$$

where Γ is the set of points (region) that is desired to be visited.

III. Review of Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an evolutionary optimization heuristic that is based upon the social behavior of birds looking for optimal food sources. The algorithm was developed by Russell Eberhart and James Kennedy. It was developed after several attempts to simulate the movement of organisms in a bird flock, such as flocking synchronously, changing directions suddenly, to scattering and regrouping [12]. Since then, PSO has been implemented to optimize nonlinear and piecewise continuous functions. PSO is a non-gradient optimization heuristic, that can be used to search non-convex spaces. One of the reasons for its growing notoriety is because of its easy implementation compared to other evolutionary algorithms and the quality of its results.

PSO consists of a number of particles ($P \in \mathbb{R}^m$), where *m* is the number of optimization variables. The entries of the particle is known as its position vector, which represents a solution to the optimization problem. Each particle also has a velocity ($V \in \mathbb{R}^m$). The particles have memory of their own optimal solution position (P_b) as well as the global optimal position of all the particles (G_b). There are examples in the literature that break the particle into neighborhoods, the difference being that the particles only remember P_b and the best in its neighborhood (L_b), which can consist of several

"close" particles. Each particle iteratively adjusts its velocity based on P_b, G_b and a weighting (α) of its previous velocity. The new velocity is then added to its current position and the new position offers a new solution to the optimization problem. By tweaking the constants c_1 and c_2, the user can alter the balance of each particle's exploration versus exploitation. Typical values for the constants are $c_1 = c_2 = 1.49$ and $\alpha = 0.72$ [14], [15]. To start the particle swarm optimization technique a random position and velocity is given to each particle. To evaluate the solution, particles' positions are then placed into the function to be evaluated. The PSO algorithm is shown in the following equations:

$$\begin{aligned} V_{k+1} &= \alpha V_k + c_1\mu_1\left(P_b - U_k\right) + c_2\mu_2\left(G_b - U_k\right), \\ U_{k+1} &= U_k + V_{k+1}, \end{aligned}$$

where $\mu_1, \mu_2 \in [0\ 1]$ are random variables with a uniform distribution.

IV. Approach

To simplify the problem at hand, we assume the UAVs have one of three control inputs: turn left, turn right or go straight. Mathematically, this enforces a discrete constraint on u_i. For each $i = 1, \ldots, n$ we have

$$\begin{aligned} u_i &\in \{-\omega_i, 0, \omega_i\}, \quad (12) \\ \omega_i &> 0, \quad (13) \end{aligned}$$

where ω_i is the maximum turning rate of vehicle i. Discretizing the control input allows us to significantly reduce the number of feasible solutions from uncountably many to 3^{NT} feasible solutions.

A. Control Input Sequence Array

First we form the control input sequence for each vehicle:

$$U_i = \begin{bmatrix} u_i^1 & u_i^2 & \cdots & u_i^T \end{bmatrix}. \quad (14)$$

The control input sequence $\overline{U} \in \mathbb{R}^{NT}$ contains the control input sequence for all the UAVs:

$$\overline{U} = [U_1\ \ U_2\ \ \cdots\ \ U_N]. \quad (15)$$

The control input sequence array is set up so that the first T elements correspond to the control input for the first vehicle; elements $T+1$ through $2T$ correspond to the second vehicle and so on. For each vehicle, the first element of U_i is the first control input, and the control inputs are sequentially implemented with the final control input being the T^{th} component of U_i.

Now we describe the process used to ensure that each $u_i \in \{-\omega_i, 0, \omega_i\}$. The absolute value of each element in the control sequence array is divided by the maximum element of the array and then multiplied by three. Let

$$U_a = \left\{ \left|\overline{U}_k\right| : \overline{U}_k \text{ is the } k^{\text{th}} \text{ element of } \overline{U} \right\}_{k=1}^{NT}. \quad (16)$$

We then compute the temporary control sequence U^{temp} componentwise. For each $k = 1, \ldots, NT$, let

$$U_k^{\text{temp}} = \frac{3\left|\overline{U}_k\right|}{\max U_a}. \quad (17)$$

Finally, the actual control sequence is redefined as follows: for each $i = 1, \ldots, NT$. Redefining the control sequence allows us to approximate a discrete control sequence based upon the relative magnitudes of the elements of $\overline{U}$

$$\overline{U}_k = \begin{cases} -\omega, & \text{if } U_k^{\text{temp}} \le 1 \\ 0, & \text{if } 1 < U_k^{\text{temp}} \le 2 \\ \omega, & \text{if } U_k^{\text{temp}} > 2 \end{cases}. \quad (18)$$

B. Position Vectors

Now that the control sequence has been defined for a vehicle, we can numerically integrate the system to approximate the states of the vehicles, given the initial conditions of each vehicle. The sequence of the state vectors is defined exactly like the sequence of the control sequence array. First we define the state sequence of each vehicle as follows: for each $i = 1, \ldots, n$

$$\begin{aligned} X_i &= \begin{bmatrix} x_i^1 & x_i^2 & \cdots & x_i^T \end{bmatrix}, \quad (19) \\ Y_i &= \begin{bmatrix} y_i^1 & y_i^2 & \cdots & y_i^T \end{bmatrix}, \quad (20) \\ \Psi_i &= \begin{bmatrix} \psi_i^1 & \psi_i^2 & \cdots & \psi_i^T \end{bmatrix}. \quad (21) \end{aligned}$$

Then we define the state sequence arrays for all vehicles:

$$\begin{aligned} X &= [X_1\ \ X_2\ \ \cdots\ \ X_N], \quad (22) \\ Y &= [Y_1\ \ Y_2\ \ \cdots\ \ Y_N], \quad (23) \\ \Psi &= [\Psi_1\ \ \Psi_2\ \ \cdots\ \ \Psi_N]. \quad (24) \end{aligned}$$

The first element of each X_i, Y_i, Ψ_i corresponds to the initial condition of the i-th vehicle, while the last element of each is the final state of the vehicle. The second element of each X_i, Y_i, Ψ_i corresponds to the states of the vehicle after the control input U_i^1 has been implemented, and so on until the final state is calculated. A second-order Runge-Kutta method was used for the numerical integration.

Once the vehicle states are known, the reward for the UAV being at that location can be calculated from equations (4) - (6). The total reward for the system at the final time T is calculated by summing the rewards for each vehicle at each state in the discretized state array.

V. Algorithm

In our algorithm, UAVs communicate via *blackboard*, that is, each UAV has access to the environment **information** $E(x, y, t)$. The algorithm is as follows: assume a particle gives a solution/control sequence U. U is then transformed into a discrete form using equations (16) - (18), yielding $\overline{U}$. At time-step t, vehicle i implements its control. Numerically integrating the vehicle's dynamics for the implemented control allows us to identify the states' of vehicle i at time-step t, to evaluate equation (5). A dummy environment $E_d(x, y, t)$ is then updated by setting $E_d(f(p_i), t) = 0$, that is the region within the a UAVs' footprint equal to zero. Vehicle $i + 1$ then implements its control for the same time-step t, and updates the dummy environment once again. This is done for all vehicles. $E_d(x, y, t)$ is updated after each implemented

control, to ensure no two vehicles receive the same reward. At the end of time-step t, we use equation (6) to realize the total reward obtained by all vehicles for that time instant. The dummy environment for the next time-step, $E_d(x, y, t+1)$, is calculated using equation (25), where $C(x, y)$ is a constant value that depends on the coordinates. After $E_d(x, y, t)$ has been updated for all the control inputs, a total of $N \cdot T$ times, the total reward over the entire time horizon using equation (3) is then compared to those of all other particles for the global best solution and local best solution. At the end of the algorithm, the $E_d(x, y, t)$ corresponding to the optimal control sequence becomes the environment for all time, $E(x, y, t) = E_d(x, y, t)$.

$$\dot{E}(x, y, t) = C(x, y) \tag{25}$$

The feasibility constraint is programmed into the algorithm. Because we are performing a search of the control space, implementing any control in the allowable control space will produce a feasible state and hence, a feasible trajectory. Collision avoidance is implicitly defined within the optimization problem, as it is not optimal for the footprint of any two UAVs to intersect at a given time *t*.

Attending to pop-up alarms is the most difficult requirement to implement, and was dealt with using the reward function. At the instant the UAVs are notified of a potential threat's coordinates $\tilde{p}\,(\tilde{x}_q, \tilde{y}_q)$, the reward in the vicinity of $\tilde{p}$ is substantially increased beyond the value of area search. The reason for dealing with targets in this manner is to have the UAVs balance between target visiting and searching for new targets. If the UAVs are in an area of very high reward, it is undesirable to leave that area immediately to visit pop-ups in very low areas of interest. So we instead entice the vehicles to visit the targets with a higher reward.

The following is pseudo-code for the described algorithm.

```
Step 1: Initialize U & V
for pt = 1 to particles
   for i = 1 to N
      for t = 1 to T
         integrate f(x(t),u(t))-system dynamics
      end
   end
   evaluate R(t)
end
pbest is the initialized control sequence
preward is the reward of each particle
gbest is the particle with highest reward
greward is the reward of the gbest particle

for iter=1:iterations
   Evaluate PSO
      for pt = 1 to particles
         for i = 1 to N
            for t = 1 to T
               integrate f(x(t),u(n,t))
            end
         end
         evaluate R(t)
         if reward > preward
            preward = reward
            pbest = the new particle
         end
         if reward > greward
            greward = reward
            gbest = the new particle
         end
      end
   end
end
```

Psuedo-code 1: UAV surveillance PSO algorithm

VI. Results

For the simulations used in this section, the following parameters were held constant:

$$\begin{aligned} N &= 3, \\ T &= 20 \text{ (time horizon)}, \\ p &= 20 \text{ (number of particles)}, \\ v &= [30\ 30\ 30], \\ r &= [3\ 3\ 3] \\ \omega &= \frac{\pi}{2}. \end{aligned}$$

There are four scenarios considered in this section. Each scenario adheres to the above parameters and was simulated in MATLAB. The first three subsections show the results of PSO using three different reward functions. The results shown in these subsections show the initial reward function and the number of times an area is within the footprint of any vehicle. These scenarios do not consider random targets within the base. The UAVs were allowed to optimize over 2000 time steps, running the optimization technique 100 times, yielding one run. There were 10 runs with each UAV starting at a random initial condition, with the averages shown in the results. The fourth scenario shows the trajectory of three UAVs optimizing the reward as well as attending to pop-up alarms. The average time for optimization using these parameters was 133.7 seconds, which can be greatly reduced using C++ or Fortran.

The reward functions and their rate of growth were chosen to be continuous semi-positive definite functions. The functions were chosen specifically such that the regions of high interest would have higher rewards and grow faster than regions of low interest. It was important to make sure that the rate at which the reward function grows was not extremely rapid so that the reward function does not "blow up"/grow unbounded, and also to ensure the UAVs do not stay in the areas where the reward function grows rapidly, hence the $\frac{1}{20}$ constant in Equation (26).

$$\begin{aligned} E_1(x, y, t_0) &= \max\left(|50 - x|, |50 - y|\right), \\ E_2(x, y, t_0) &= 10\left(\sin\frac{x}{7} + \sin\frac{y}{7} + 2\right), \\ E_3(x, y, t_0) &= 50 - \max\left(|50 - x|, |50 - y|\right). \end{aligned}$$

R_1 is for perimeter surveillance; R_2 is for lake surveillance; and R_3 is for protecting a center asset. Each of the three reward functions satisfies

$$\dot{E}_k(x, y, t) = \frac{1}{20} E_k(x, y, t_0), \tag{26}$$

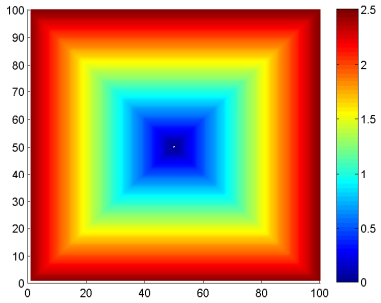

Fig. 1. Contour of $\dot{E}_1(x, y, t_0)$, for perimeter surveillance

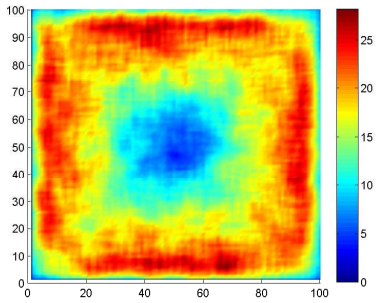

Fig. 2. Number of visits per 2000 steps, for perimeter surveillance

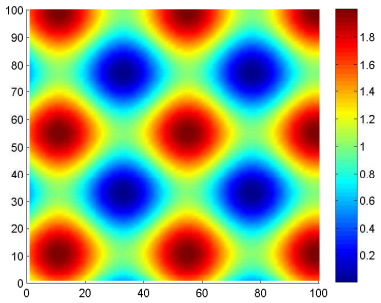

Fig. 3. Contour of $\dot{E}_2(x, y, t_0)$, for lake surveillance

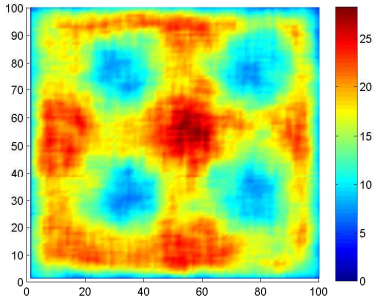

Fig. 4. Number of visits per 2000 steps, for lake surveillance

for $k = 1, 2, 3$.

A. Perimeter Surveillance

Figure 1 shows $\dot{E}_1$, the growth rate of the reward function. It is proportional to the distance away from the center, as expected for perimeter surveillance. In this scenario, the UAVs would like to stay near the perimeter of the base to prevent threats from reaching the interior. Figure 2 shows the number of times a specific area was within the footprint of any vehicle. The UAVs highly concentrated their time to the outer edges of the base, as desired in perimeter surveillance. The number of visits to an area is proportional to rate of growth of the reward. The regions along the outer edge of the base were within the radius of view on average 28 times. As the distance from the center of the base decreases so does the number of times the area is viewed. The relationship is almost linear, with the inner most region of the base being viewed approximately once.

B. Lake Surveillance

Figure 3 is a surveillance problem where there are regions of very little interest such as small lakes held within the base (dark blue regions centered at (30,30), (80,30), (30,80) and (80,80). The probability of a target being in these regions is very low, while regions outside the lakes are of greater interest. The goal in the scenario is to spend almost no time viewing regions over the lakes, but instead visit the hot spots such as roads. Figure 4 demonstrates the UAVs donate the majority of their time to the more important regions outside of the lakes, where the environment grows the fastest. The regions with highest growth rate were viewed on average 28 times, on 2000 time steps. The inner most regions of the lakes were viewed less than five times, over the 2000 time steps.

C. Center Asset

Another scenario to consider is the surveillance of a stationary object, at the center of the base. Figure 5 shows the reward function: the area of the highest interest lies in the center. Therefore the UAVs should focus their attention on the center of the base. Figure 6 shows the results from PSO. The inner most region is viewed on average a 33 times, and the remainder of the base is viewed a number of times inversely proportional to the distance from the center. The outer most areas of the base are viewed approximately once.

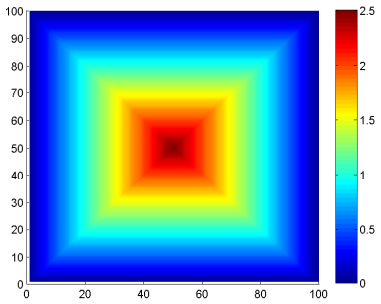

Fig. 5. Contour $\dot{E}_3(x, y, t_0)$, for center asset surveillance

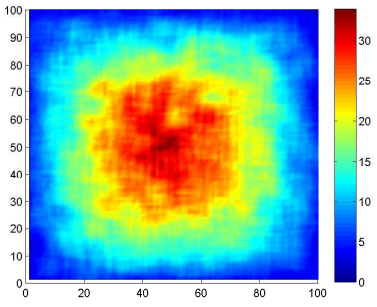

Fig. 6. Number of visits per 2000 steps, for center asset surveillance

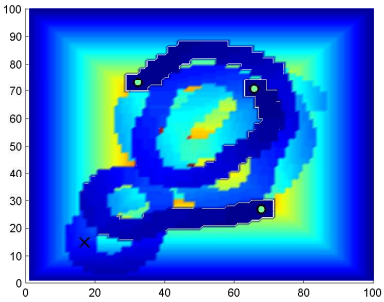

Fig. 7. Center Asset Surveillance with Target

D. Target Attendance and Trajectories

The last scenario considers the surveillance of a center asset, when there is an alarm that goes off to notify the UAVs of a potential threat, represented by an X in Figure 7. The UAVs must identify the object to decide whether indeed it is a target or not. In addition, the remainder of the UAVs must continue to watch the center asset. The results show one UAV goes to scout the object while the remainder of the UAVs continue their surveillance. After the UAV scouts the object, it returns back to surveillance of the center asset.

VII. Conclusion

The UAV control algorithm in this paper, can easily be implemented and used for surveillance in both military and civilian applications. The results shown in the previous section show that PSO is a viable approach to constructing trajectories for difficult cooperative surveillance problems. We have demonstrated that the PSO method combines both exploration and exploiting of important areas for surveillance. The algorithm at hand can be used to calculate trajectories online, and within reasonable time. Furthermore the PSO algorithm generates unpredictable paths online, that makes it difficult for adversaries to predict the paths of the UAVs.

References

[1] D. Enns, D. Bugajski and S. Pratt, "Guidance and Control for Cooperative Search", *in Proceedings of the American Control Conference*, Anchorage, AK, May 2002.

[2] A. Ahmadzadeh, A. Jadbabaie, V. Kumar, G. Pappas, "Stable Multi-Particle Systems and Application in Multi-Vehilce Path Planning and Coverage", *in Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, LA, December 2007.

[3] P. DeLima and D. Pack, "Toward Developing an Optimal Cooperative Search Algorithm for Multiple Unmanned Aerial Vehicles", *in Proceedings of the International Symposium on Collaborative Technologies and Systems*, Irvine, CA, May 2008.

[4] A. Ahmadzadeh, A. Jadbabaie, V. Kumar, G. Pappas, "Cooperative Coverage using Receding Horizon Control", *in Proceedings of the European Control Conference*, Kos, Greece, July 2007.

[5] M. Flint, M, Polycarpou, E. Fernandez-Gaucherand, "Cooperative Control for Multiple Autonomous UAVs Searching for Targets", *in Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, NV, December 2002.

[6] J. Ousingsawat and M. Earl, "Modified lawn-mower search pattern for areas comprised of weighted regions", *in Proceedings of the American Control Conference*, New York, NY, July 2007.

[7] P.F. Hokayem, D. Stipanovic, M.W. Spong, "On Persistent Coverage Control", *in Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, LA, December 2007.

[8] C. Cassandras and W. Li, "A Receding Horizon Approach for Solving Some Cooperative Control Problems", *in Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, NV, December 2002.

[9] A.R. Girard. A.S. Howell, J.K. Hedrick, "Border Patrol and Surveillance Missions using Multiple Unmanned Air Vehilces", *in Proceedings of the 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, December 2004.

[10] R.W. Beard, T.W. McLain, D.B. Nelson, D. Kingston, D.Johanson, "Decentralized Cooperative Aerial Surveillance Using Fixed-Wing Miniature UAVs", *Proceedings of the IEEE*, Vol. 94, Issue 7, July 2006.

[11] A. Banks, J. Vincent, K. Phalp, "Particle Swarm Guidance System for Autonomous Unmanned Aerial Vehicles in an Air Defence Role", *Journal of Navigation*, Cambridge Online Journals, Vol. 61, Issue 1, January 2008.

[12] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", *in Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia, November-December 1995.

[13] J. Kennedy and R. Eberhart, "A New Optimizer Using Particle Swarm Theory", *in Proceedings of the International Symposium on MicroMachine and Human Science*, Nagoya, Japan, October 1995.

[14] I. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection", *Information Processing Letters*, Vol. 85, No. 6, March 2003.

[15] F. van den Bergh, A. Engelbrecht, "A study of particle swarm optimization particle trajectories", *Information Sciences*, Vol. 176, No. 8, April 2006.