Coordinated UAV Manoeuvring Flight Formation

Henry Hexmoor and Shahram Rahimi
Department of Computer Science
Southern Illinois University
Carbondale, IL. 62901, USA
E-mail: {hexmoor, rahimi}@cs.siu.edu

Jody T. Little
Sierra Nevada Corporation
4801 NW Loop, 410
San Antonio, TX 78205, USA

Keywords: agents, collaboration, UAV, formation flight, affect

Received: May 28, 2007

A methodology is presented for real-time control of unmanned aerial vehicles (UAV) in the absence of apriori knowledge of location of sites in an inhospitable flight territory. Our proposed hostile control methodology generates a sequence of waypoints to be pursued on the way to the target. Waypoints are continually computed with new information about the nature of changing threat. The Dijkstra algorithm is used to account for a weighted combination of threat measures arising from the probability of encountering hostile ground to air fire as well as the internal urgency to complete the mission in the shortest time. UAVs broadcast latest sensed data to their counterparts. The sequence of waypoints defines the trajectory of the UAV to its target. By varying components of cost function, paths are altered to obtain a desired performance criterion. Validation of our methodology is offered by a series of agent-based simulations.

Povzetek: Predstavljena je metoda za upravljanje brezpilotnega letala ali helikopterja v sovražnem okolju.

1 Introduction

A powered, aerial vehicle that does not need a human pilot and uses aerodynamics forces to provide vehicle lift is called Unmanned Aerial Vehicle (UAV), also called a drone. UAVs are able to fly autonomously or can be remotely piloted. They might be expendable and can be equipped to carry lethal payloads. Ballistic missiles, cruise missiles, and artillery projectiles are not considered UAVs. Three varieties are shown in Figures 1-3.

UAVs were first conceived in the aftermath of the First World War and were used during the Second World War to train anti-aircraft gunners. Subsequently, early UAVs were large, remote-controlled planes until the late 20th century. Technological advances allowed the military to develop more capable fighting machines that could be used in performing dangerous missions without posing a significant risk to human life.

Military strategists have envisioned developing a wide array of roles performed by unmanned aircraft including deployment of ordinance [25]. UAVs are primarily used for intelligence, reconnaissance, surveillance. For example, the Global Hawk UAC (shown in Figure 1) provides the U.S. Air Force and joint battlefield commanders with near real-time, high resolution intelligence, surveillance, and reconnaissance imagery. Predator (Figure 2) is a high endurance, medium altitude unmanned aircraft system used for surveillance and reconnaissance missions, as well as the refuelling of jets. Seiko Epson has developed the world's lightest and smallest UAV helicopter, the Micro Flying

Figure 1: A Global Hawk.

Figure 2: A Predator.

Figure 3: A Micro UAV.

Robot (shown in Figure 3). Sierra Nevada Corporation offers the most widely used UAV landing and takeoff software systems. Next, we outline the specific flight problem that is addressed.

Consider a ground to air protected hostile force zone that is of interest for surveillance by the friendly force. We need to compute the vulnerability in crossing the enemy region and find the safest trajectory across the region. Consider a swarm of coordinating reconnaissance UAVs, modelled as agents, which attempt to cross the hostile region. These UAV agents form a closely communicating friendly network and collectively produce estimates of threats needed to generate the safest possible flight trajectories.

The main objective of this work is to provide a novel approach for manoeuvring of a group of coordinating agents in a hostile environment. Due to the complexity of the problem involving formation and control of the UAVs in the real world, we are primarily concentrating on path generation [9].

2 Related work

The problem of multi-sensor surveillance involves detection of multiple intrusions and tracking of the intruders. Detection and target tracking have been explained from multiple viewpoints. To keep track of multiple moving objects researchers need to know all the joint probabilistic states of the all objects. These states grow exponentially with increase of number of moving objects. Sample-based variant of join probability data association filters technique is used to track moving objects [22]. Mobile service agents are designed to operate in a dynamic environment by estimating the state of the dynamic object using probabilistic techniques [23]. The probabilistic estimates accurately identify the most likely state of each dynamic object. Multiple targets can be tracked by using dynamic time stamps. In order to detect, track, and avoid targets, researchers have used cluster-based approaches [12]. The focus of all these techniques has been building reliable estimators and trackers. These approaches do not use distributed sensors and are not directly useful for the problem of large area surveillance.

Within the context of distributed task allocation and sensor coordination [20] proposed a scheme for delegating and withdrawing agents to and from targets through the ALLIANCE architecture. The protocol for allocation was one based on the "impatience" of the robot with respect to a target while withdrawal was based on "acquiescence." [18] presented a strategy for tracking multiple intruders through a distributed mobile sensor network. Researchers have made significant advances in the areas of distributed sensor networks [7] and sensor management [18].

In [7], distributed robots cross a region using density estimates in a manner that facilitated maximal tracking of targets in that region. The decision for a robot to move to another region or to stay in its current region was based on certain heuristics. The method presented did not address collaborative or shared reasoning strategies for decision-making and action selection, such as the decision to move to a new area. Coordination between sensors was restricted to communicating their respective positions. Lesser's group used sensor coordination for the purpose of tracking only one target [14].

In solving path planning and resource allocation problems researchers have considered UAVs to be independent of one another. Path planning methods focus on sequencing UAVs to arrive at specified locations or target locations [15]. Resource allocation methods have concentrated on target assignment, [16] and classification [5]. In the research of Ken Nygard [19], a scheme based on hierarchical decomposition was presented to assign a sub-team of UAVs to a particular task. McLain used a methodology that allowed the UAVs to reach the target simultaneously by avoiding pop-up threats whose locations were known a-priori [17].

This article presents a strategy that is a variant of methods described in [17] in that the UAVs can be coordinated to arrive at a desired target location or respective target locations within a specified time period. However, unlike McLain's [17], locations of targets are not known beforehand. A set of waypoints are generated through a reinterpreted Dijkstra algorithm, and the computation is updated every time a *UAV agent* discovers new information. Our contribution is largely in our unique interpretation of path costs. There are numerous applications of this algorithm in robotic motion planning [8]. However, there are no known adaptations of this algorithm that suggest an emotional interpretation of path costs. This is a novel contribution in our approach. We have redefined the interned distance function from the standard algorithm, e.g., given in [6].

Simulations can be greatly simplified when the agent's mission is executed using formation control strategy [1]. Formation control strategy is used to study aerodynamic effects involving multiple-aircraft [15] and to explore large areas with the aid of UAVs whose sensor capabilities are limited [19]. The aspect ratio[1] of the formation can be increased by using a formation control strategy. Two or more closely spaced agents can be treated as a single unit, which will decrease the workload of a remote operator and the cost of communication with

[1] The aspect ratio of two-dimensional shape is the ratio of its longest dimension to its shortest dimension.

other aircraft because the distance between aircrafts is decreased.

In many missions, UAVs fly in formations that are thought to increase chances of success. There are three different formation control strategies: leader following approach, behavioral approach, and virtual structures approach.

In the leader following approach, one of the agents acts like a leader, while the rest of the agents act like followers. The follower agents track the position and orientation of the leader. The problem with this approach is that the motion of the agent is controlled by the motion of the leader; if the leader agent fails in its motion then the following agent will not make the best move. Researchers have proposed variations of the *leader following approach* in order to prevent the centralized control of the agents' motion. In these approaches, agents designate multiple leaders, form chains, and create tree topologies. These approaches only solve the centralized motion problem partially, because subsets of the agents follow the designated leader agent. There have been a number of studies of leader-following techniques in the mobile robotics community. Using leader-following technique [5], robots can cooperatively move a box.

The goals of the behavioral approach are to prescribe several desired behaviors for each agent [1] and to make the control action of each agent a weighted average of the behaviors. Possible behaviors include collision avoidance, obstacle avoidance, goal-seeking and formation-keeping. There are also numerous variations on the behavioral approach to multi-agent coordination, most of which are derived by novel weight of the behaviors. In the behavioral approach agents use the decentralized motion method, which is desired in Multi Agent Systems. Botelho and Alami (1999) applied the behavioral approach to the problem of satellites in an equally distributed ring formation [2].

In the virtual structure approach, the entire agent formation is treated as a single structure. This approach defines the desired motion of the structure. Next, the motion of the virtual structure is translated into the desired motion for each agent. Finally, tracking controls for each agent are derived. It is very easy to implement this approach, which is its strength. However, applications in which we can use this approach are very limited.

Agent interaction in multiagent systems is associated with some form of communication [3]. Researchers rely on agent communication to solve standard multi-agent problems, like coordination and negotiation. The communication primitives that are exchanged among agents are typically referred to as communicative acts or speech acts. Some communicative acts are informing about the environment, querying about environment, telling about an action, advising other agents, and directing agents. These communicative acts help agents to make decisions. Agents have to use a standardized language format for exchanging information so that agents can easily understand each other. Several agent communication languages have been proposed by researchers aiming at standardizing the multi-agent communication process. The two most notable ones are KQML and FIPA ACL [24], each using a slightly different syntax and set of communicative acts.

Intelligent agents, having incomplete knowledge of the operating environment, must learn the structure of the environment to better accomplish their tasks. This exploration may be performed in a different phase, or it may be combined with the task at hand. Algorithms which guide agents in unknown physical environments can be classified as full exploration algorithms or navigation algorithms [26]. Full exploration algorithms are used when the entire environment is mapped out a-priori. Navigation algorithms are used when a specific target location has to be reached. Modified search algorithms are widely used in solving these types of problems.

Hill climbing is a heuristic method of searching for solutions to problems that have huge solution spaces [21]. In the hill climbing approach, the problem space is converted to a graph, where each solution corresponds to a node associated with a value. The current path is extended with a successor node, which is closer to the solution than the end of the current path. In the simple hill climbing approach, the node closest to the present node is added to the solution space. In steepest ascent hill climbing, all successors are compared, and the one closest to the solution is selected. A fitness function is defined for evaluating the effectiveness of the node newly added to the solution. Unless the fitness function is smooth and effective, these two approaches will fail to a reach global maximum. This happens when there are local maxima in the search space, which are not solutions. This can be partially overcome using varied hill climbing methods such as iterated hill climbing, stochastic hill climbing, random walks, and simulated annealing [26].

To avoid getting stuck on the first local maximum, several hill climbs are repeated, each time starting from different randomly chosen points. This method is known as iterated hill climbing. This approach increases the probability of reaching the global maximum value by detecting different local maximum points. If there are several local maximum points in problem space, it is not a good method to implement. This approach surprisingly gives optimal results in many applications.

In the stochastic hill climbing approach, a randomly chosen neighbor node is evaluated using the fitness function. This node is only retained if it increases the value of the fitness function; otherwise, another neighbor node is selected randomly for evaluation of the fitness function. Stochastic hill climbing usually starts from a random point. By combining iterated hill climbing with stochastic hill climbing, solution paths avoid getting stuck on local optimum values.

Simulated annealing is an optimization technique proposed by [13] by extending the Monte Carlo method to determine the equilibrium state of a collection of atoms at any given temperature T. Simulated annealing is inspired by a technique involving the heating and controlled cooling of a material that increases the size of its crystals and reduces their defect. The heat makes

atoms move randomly due to the high energy state. The slow cooling gives them more chance of finding more stable configuration than the initial one. Simulated annealing consists of randomly choosing a solution from the neighboring nodes of the current node. If the value of the fitness function increases the current one, the new node is accepted as the new current node. If the fitness function is not improved, the new solution is retained with probability. Simulated annealing differs from hill climbing algorithms in that it allows the possibility of going downhill if the temperature is high enough. When the temperature is high, the motion is more random. Because the probability of going downhill is inversely related to temperature, when temperature decreases, it is hard to go downhill. Simulated annealing generally starts with high temperature values. The slower the cooling, the easier it becomes to find the global solution. Infinitely slow cooling certainly produces a global optimum solution, although it might take infinite time. The main difficulty of simulated annealing is to find an appropriate temperature decrease rate.

Thus far, we have discussed the research related to coordination and communication in multiagent systems. In the following sections, we will present details of our approach.

3 Implementation

We have made certain abstractions while transforming the real world situation to a simulated environment. We also make assumptions about the modelling of inter-agent communication. We define our simulation system by describing agents present in the system, their capabilities, and the missions assigned to the agents. There are two different types of agents present in our system: friendly UAV agents and hostile, ground-based agents. UAV agents are autonomous and compute their trajectory dynamically. They perform surveillance over a square-shaped surveillance zone. The main objective for UAV agents is to move from a location outside the surveillance area to a designated target zone. These agents use a variety of algorithms to achieve their objective. They continually compose and update a trajectory to reach the target location consistent with perception of their surroundings. Our simulation allows for user-defined number of UAV agents to be modelled. These agents communicate with one another to effectively analyze the surveillance zone.

Hostile agents protect the surveillance area from intrusion. From the perspective of UAV agents, hostile agents are considered to be threatening. Hostile agents are static and reactive. Hostile agents operate independently of one another. The objective of these agents is to harm unknown vehicles within their perceptual range. Although they are an integral component of our system, they are not considered to be part of the multi-agent component as they do not communicate or coordinate among themselves. They remain in the same location throughout the simulation. Therefore, once UAV agents identify the location of a hostile agent, that hostile agent need not be sensed again.

Hostile agents are capable of firing anti-aircraft missiles. The success of hitting UAV agents depends on the distance of the UAV agent from the hostile agent. In our simulation, UAV agents cannot be completely destroyed by anti-aircraft missiles. Each UAV agent will continue to travel the surveillance area even after it has sustained a strike by an anti-aircraft missile. This assumption is made so that we can compute the effectiveness of the path algorithm used for the reconnaissance mission.

We introduce a number of assumptions in order to define our model. Hostile agents are guarding the surveillance area in which UAV agents are attempting to cross. UAV agents divide the surveillance zone into number of square cells for the trajectory computation. The firing strength of the hostile agent is the probability that it can shoot down a UAV agent when the UAV agent flies over the cell occupied by it. The probability of hitting a UAV agent decreases as the distance of the UAV agent from the cell occupied by hostile agent increases. As shown in Figure 4, in an eight-connected sense, the occupied cell by a hostile agent is denoted by O, the four nearest neighbors are denoted by N1 and the four diagonal neighbors by N2. If the distance between the centers of two four-connected cells is termed a unit, cells that are two units away are denoted by N2. Figure 4 is used to compute actual firing strength of hostile forces. In contrast, Figure 8 will be used to reflect the perspective of a friendly UAV flying over the hostile territory and how they perceive threats from hostile agents. Since UAVs share information, subsequent figures show cumulative perceptions of threat.

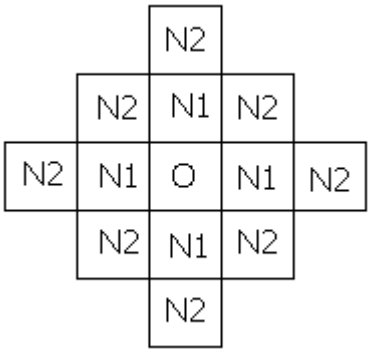

Figure 4: Labeling of cells with respect to a central hostile cell denoted by O; i.e., the cell that is occupied by a hostile agent.

Direct firing strength is defined as the probability of hitting a UAV agent when the UAV agent is flying over cell O. First neighboring firing strength is defined as the probability of hitting UAV agent from the cells labelled N1. Second neighbor firing strength is defined as the probability of hitting UAV agent from the cells labelled N2.

For a UAV agent that flies over a cell i, the probability that it is hit by a hostile agent is computed by Equation 1.

$$p_{fi} = \begin{cases} fs_O; i \in O \\ fs_{N1}; i \in N1 \\ fs_{N2}; i \in N2 \\ 0; elsewhere \end{cases} \quad fs_O > fs_{N1} > fs_{N2}$$

(Equation 1)

fs_O is direct firing strength, fs_{N1} is firing strength at the first neighbor's level, and fs_{N2} is firing strength at the second neighbor's level.

If a UAV agent flies directly over an occupied cell, the UAV agent has the maximum probability of being hit, but its probability of being struck decreases if it flies over cell N1 and further decreases if it flies over cell N2. p_{fj} is the probability that a hostile agent situated at cell j can fire successfully at the UAV agent flying over cell i.

Thus, for a UAV agent flying over cell i, the cumulative probability of being shot is the sum of the individual probabilities of being successfully fired at by the hostile agents positioned at the cell beneath or any of the N1 or N2 cells. The cumulative probability of being shot is given by P_{si}, i.e.,

$$P_{si} = \sum_j p_{fj},\ j \in O\ or\ j \in N1\ or\ j \in N2$$

(Equation 2)

In the absence of prior information regarding the occupancy of the cells in the region, the prior probability of cell j being occupied by a hostile agent is denoted by p_{Oj}. The cumulative probability of being shot when flying over cell i and lacking such information regarding the occupancy of the cells is given by Equation 3.

$$P_{si} = \sum_j p_{Oj} p_{fj},\ j \in O\ or\ j \in N1\ or\ j \in N2$$

(Equation 3)

It is assumed that the a-priori information regarding the number of hostile agents, n_{HA}, is known while their coordinates or the cells that they occupy are unknown. If the total number of cells is n_C, then $p_{Oj} = \frac{n_{HA}}{n_C}$. As the UAV agents move over the habitat they obtain information regarding the occupancy or non-occupancy of a cell. This information is then broadcast to the other UAV agents. The occupancy probability of a cell for which there is no information yet is recomputed as in Equation 4.

$$p_{Oj} = \frac{n_{HA} - n_V}{n_C - v} \qquad \text{(Equation 4)}$$

In Equation 4, n_V represents the number of cells that are occupied by hostile agents. Here, v represents the total number of cells for which the UAV agent has obtained information about.

Distance anxiety denotes the anxiety experienced by the UAV agents to travel unit distance over the hostile area. We use w to denote distance anxiety. When $w = 0$ the UAV agent has the maximum innate distance anxiety. This encourages UAV agent to seek the path with the shortest possible distance. At $w = 1$ the UAV agent will not have any distance anxiety. At this value, UAV agent will seek paths of least probabilistic resistance that need not be optimal in terms of distance. It is to be noted, however that increasing w does not always imply increasing path lengths – it only implies paths with reduced chances of being fired at or a search that is biased towards least probabilistic paths. Decreasing w does not always imply paths along which there are increased chances of getting fired at – it only implies a search that is biased towards shortest distance paths. In other words, a path of least probabilistic resistance obtained for $w = 1$ could well be a path that is shortest in terms of distance.

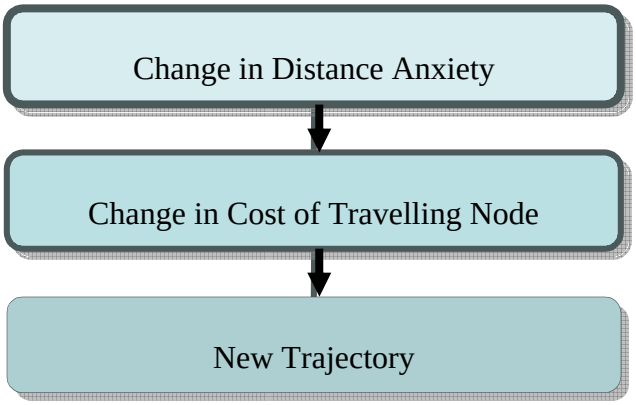

Figure 5: The effect of distance anxiety on trajectory computation.

We used parameters to control the communication behavior in the system. Communication range denotes the number of UAV agents to whom UAV agent can send messages. Communication penalty is the cost of communicating in the system for a single unit of distance. It is directly related to the distance that the UAV agent travels. If UAV agents travel a long distance the cost of communication will increase.

3.1 Path Generation

In this model, *UAV agents* have to traverse a square hostile surveillance area. While crossing this area, *UAV agents* compute trajectory by considering the hostile elements on the ground. This trajectory is also influenced by the *UAV agent's* distance anxiety value, which can cause the computed trajectory to have fewer steps. We used the standard Dijkstra algorithm to create our path generation algorithm, for algorithm consult (Knuth, 1997). The nodes in Dijkstra's algorithm

correspond to our *UAV agent* surveillance cells. Similarly, the edges correspond to adjacency among our surveillance cells. For an edge directed from cell m to cell n, the cost of traversing the edge C_{mn} is given by the weighted combination given in Equation 5.

$$C_{mn} = (1-w)\psi d_{mn} + wP_{sn} \quad \text{(Equation 5)}$$

P_{sn} is the same as the left-hand side of Equation 3; i.e., the cumulative probability of being shot at cell n. The distance between the cells in the Euclidean distance denoted by d_{mn}, while ψd_{mn} represents the fatigue accrued by the *UAV agent* after traveling a distance of d_{mn}. ψ is a normalization constant that allows d_{mn} to be scaled to similar values as P_{sn}. w is the weighing factor in the range of 0.0 to 1.0. To recapitulate, Equation 5 suggests that the cost of traveling from cell m to cell n is the sum of the overall probability of being fired at while over cell n as well as the fatigue developed with distance.

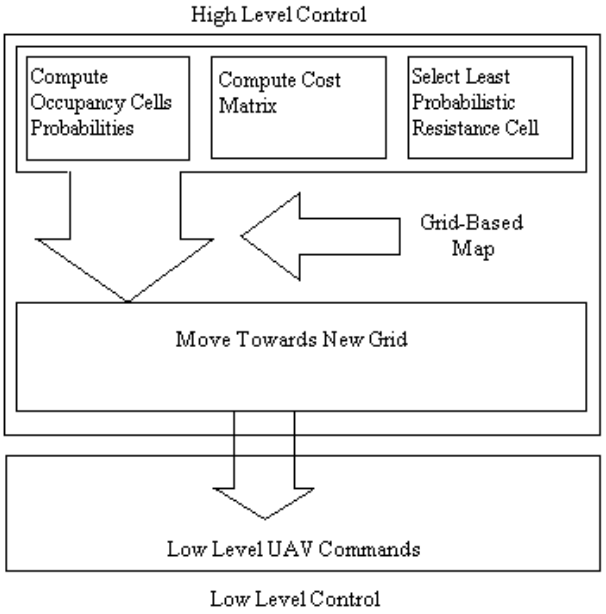

Figure 6: High level flow of algorithm.

Figure 6 depicts the high level flow of the algorithm. Paths are updated whenever information regarding a new cell in the form of either presence or the absence of a *hostile agent* occupying it is discovered. The occupancy probabilities of unobserved cells are recomputed by using Equation 4, and the cost matrix is updated by using Equation 5. Change in the distance anxiety value triggers the change in the cost of traversing two nodes as shown in Figure 5. This change will affect the waypoints generated by the path generation algorithm. The algorithm we devised for computation of the waypoints for reaching target locations uses a hill climbing technique. The waypoints are always computed from the current position to the target position. The neighbor node, which produces the optimal solution from the current node, is always added to solution space. This is similar to the steepest ascent hill-climbing search.

The overall algorithm given a specific parametric value *w* is given in Figure 7.

For all UAV agents alive in the system, do steps 2 and 3 until the last waypoint or target is reached.

If new information is available about the presence or absence of a hostile agent occupying a cell, continue with the following:

Update occupancy probabilities at all cells.

Compute the new cost matrix based on the most recent probabilities.

Search in the space of *w* for all paths that satisfy the time upper bound.

Select the path with the least probabilistic resistance or distance metric.

Move towards the next waypoint.

Figure 7: Overview of system architecture

4 Simulation details

A piece of surveillance area is shown in Figure 8. We divided the surveillance area into cells. By doing so, we can observe the enemy location and *UAV agent* motion. We also can understand the reason for *UAV agent's* motion in the surveillance area. In contrast to Figure 4, Figure 8 depicts the UAV perspective.

As mentioned earlier, the hitting probability for a cell can be computed by adding the likelihood of *hostile agent* being present in the vicinity. Cells that are located either in the corners or at the sides of the surveillance area have fewer neighbors. Consequently, these cells have the least probabilistic values, so the path computed by *UAV agents* using the algorithm will be always along these cells. We assigned shading for cells to depict their perception of levels of threat.

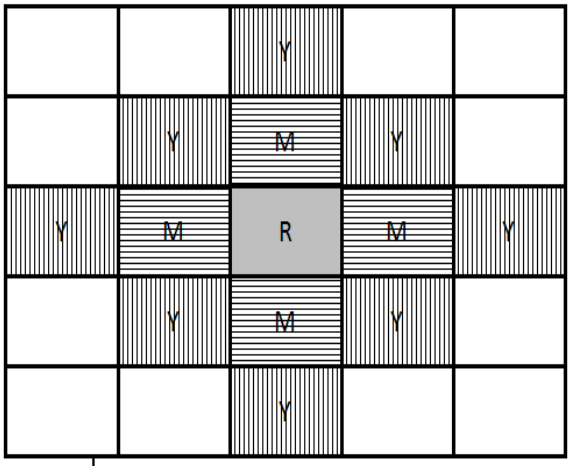

Figure 8: Shading assignment for UAV perception of the surveillance zone: white shade denotes no threat, solid gray denotes maximum threat, horizontal shade denotes moderate threat, vertical shade denotes minimal threat.

We identified the perception of severity of threats for each cell by using a specific shade. White shaded cells are neutral areas, i.e., these are perceived to be free of hostile agents. These cells are not used for computing paths. Perception of a *hostile agent* at a cell is denoted by

solid gray shade. The first neighboring cells of a hostile agent are shaded horizontally. The second neighboring cells of a hostile agent are shaded vertically. Figure 9 shows our GUI where we can assign different parameter values for the simulation system. These values can be set using the sliders shown in Figure 9. We can change *firing strength* values and *distance anxiety* values for the simulation. These values can be changed at run time.

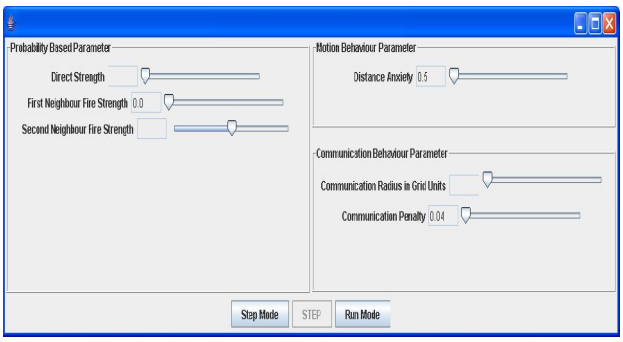

Figure 9: Simulation system controls: The GUI.

5 Results

We simulated the environment of a UAV surveillance *Multi Agent System* and carried out experiments. All of the assumptions that we mentioned in the beginning of Section 5 were made for this simulation.

The figures 10, 11 and 12 show paths of *UAV agents* for $w = 1, 0.5, and\ 0$ respectively. *UAV agents* themselves are not aware of their own locations beforehand until one of them identifies a *hostile agent* during flight. Since UAVs share information, subsequent figures show cumulative perceptions of threat. A *UAV agent's* sensing range at any instant is defined as the area covered by 9 surrounding cells, three along the length and three along the breadth. In Figure 10, the cells are shaded.

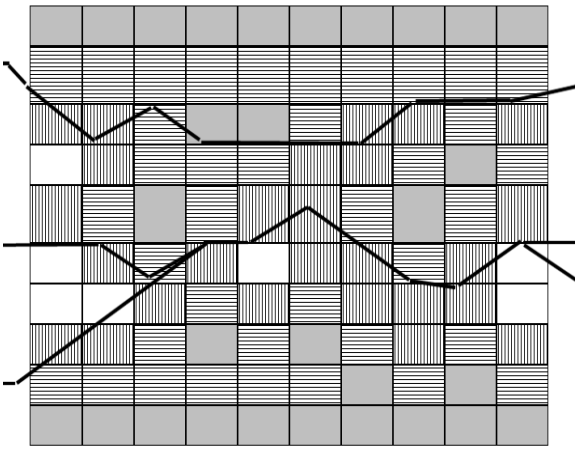

Figure 10: The paths traced by the *UAV agents* for w=1. Threat perceptions for cells are indicated by shadings: gray for most threat, vertical for the least threat, and horizontal for moderate threat.

As expected, despite their lack of apriori knowledge regarding the locations of *hostile agents*, for $w = 1$, the *UAV agents* manage to find their paths through the cells where the probability of getting fired upon is minimal. On the contrary, for $w = 0$, the *UAV agents* move bravely through hostile cells to minimize their distance. The shortest distance is not a straight line between start and target locations since the graph search is through an eight-connected lattice. For $w = 0.5$, the paths turn out to be neither the shortest distance paths nor the shortest probability paths but paths that minimize $C_{mn} = 0.5(\psi d_{mn} + P_{sn})$. The sum of probabilities can exceed 1.0 since it represents the total chances of being fired upon whenever the *UAV agent* visits the cell and based upon the path computed at that instant. While it is this sum that gets minimized during the Dijkstra search for paths of least probability, this is different from the computation that evaluates the probability that a *UAV agent* gets past λ cells safely. That computation is given by $\prod_{i=1}^{\lambda}(1-P_{si})$ and is the safety factor of a path that traverses λ cells. However, it can be shown that at any given instant based on the knowledge of the environment, the path that minimizes the sum of probabilities would also be the one that gives the maximum value of the safety factor.

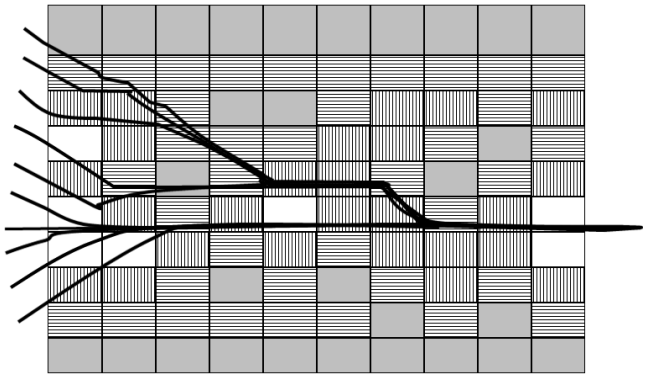

Figure 11: The paths traced by the *UAV agents* for w=0.5

It is seen that for w = 0 the paths with least prior and posterior probability sums, but with maximum distances, are obtained. This is due to the fact that the paths of least probabilities go through the center of the environment and the *UAV agents* that enter the habitat at its top and bottom search their way to the center due to lack of prior information about the *hostile agents* locations. For w=1, *UAV agents* take the shortest path with the least amount of fatigue. While computing paths with w=1, *UAV agents* will travel in an area with a high density of *hostile agents* to reduce the distance. This will increase their chances of being fired upon by *hostile agents*.

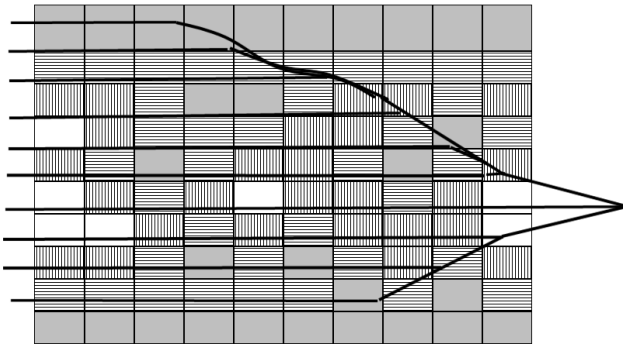

Figure 12: The paths traced by the *UAV agents* for w=0.

6 Conclusion

A methodology for parametric flight control of multiple *UAV agents* is presented so that a desired criterion is met. The desired criterion can be one that minimizes distance fatigue or the path that minimizes the chances of being fired upon. The method works for situations where a-priori knowledge of the hostile habitat is not available. It also lends itself to situations where the trajectories of *UAV agents* can be modified dynamically by adjusting a weighing factor w , since the paths are recomputed every time new information about the location of hostile agents is discovered. The effect of communication on the simulation system is observed. Communication in the simulated *multiagent system* improved the performance of the system by rapidly increasing the number of known cells to the agents at any time. The path computed by the algorithm chose waypoints towards the cells whose information is known. If *UAV agents* start at different intervals of time, the computed trajectory moves towards the known cells whose information is gathered by the UAV agent, which started the mission earlier. Due to this, *UAV agents* may not explore the surveillance area in which they are moving. The main objective was to find the safest path based on a given set of constraints.

Future scope of this work includes incorporating communication constraints such as latency, minimum distance to be maintained between *UAV agents* for information exchange, and an investigation into the role for embedding the *UAV agents* with social notions like autonomy and benevolence that yielded useful results in [13].

Acknowledgements

This research is sponsored by the Air Force Research Laboratory (AFRL) under contract FA8750-06-C-0138. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of AFRL or the United States Government. Special thanks go to the Net Centric SIGINT Focused Information Enterprise (NCSFIE) at AFLR for its support.

References

[1] Balch, T., & Arkin, R.C. (1998). Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation, 14(6),* pp 926-939.

[2] Botelho, S., & Alami, R. (1999). A scheme for multi-robot cooperation through negotiated task allocation and achievement. *In Proceedings of the International Conference on Robotics and Automation,* pp 1234-1239.

[3] Cannan, J.W. (1999). Seeing more, and risking less, with UAVs. *Aerospace America, 37(10).*

[4] Chandler, P. R., Rasmussen, S. J. (2002). Military applications: Unmanned aerial vehicles: Multi UAV: a multiple UAV simulation for investigation of cooperative control. ***In Proceedings of the 34th Conference on Winter Simulation: Exploring New Frontiers***.

[5] Chandler, P., Rasmussen, S., & Pachter, M. (2000). UAV cooperative path planning. *In Proceedings of the AIAA Guidance Navigation, and Control Conference,* Denver, CO.

[6] Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C. (2001). *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill. ISBN 0-262-03293-7. Section 24.3: Dijkstra's algorithm, pp.595–601.

[7] Das, A.K., Fierro, R., Kumar, V., Ostrowski, J.P. (2002). A vision-based formation control framework. *IEEE Trans. Robot. and Automat.,18(5),* 813-- 825.

[8] Dudek, G., (2005). Computational Principles of Mobile Robotics, Cambridge University press.

[9] Hexmoor, H., & Pasupuleti, S. (2003). Institutional versus interpersonal influences on role adoption. *International workshop on autonomous agents (AAMAS-03),* Australia.

[10] Kirkpatrick, S., Gelatt Jr., C.D., & Vecchi, M.P. (1983). *Optimization by simulated annealing. Science, 220(4598),* pp. 670-680.

[11] Knuth, D.E. (1997). *The art of computer programming: Fundamental algorithms,* (Vol. 1, 3rd ed.). Addison-Wesley.

[12] Krishna, M., Hexmoor, H., & Pasupuleti, S. (2003). Avoiding collision logjams through cooperation and conflict propagation. *International Conference on Integration of Knowledge Intensive Multi Agent Systems (KIMAS-03),* Boston, Massachusetts.

[13] Krishna, M., Hexmoor, H., & Pasupuleti, S. (2004). Role of autonomy in a distributed sensor network for surveillance. *In proceedings of the International conference on artificial intelligence (ICAI),* Las Vegas, Nevada.

[14] Lesser, V., Horling, B., Vincent, R., Miller, R., Shen, J., Becker, R., & Rawlins, K. (2001). Distributed Sensor Network for Real Time Tracking. *In Proceedings of the 5th International Conference on Autonomous Agents*. pp 417- 424. Montreal, Canada.

[15] McLain, T., Chandler, P., Ramussen, S., & Pachter, M. (2001). Cooperative control of UAV rendezvous. *In Proceedings of the American Control Conference (ACC),* pp. 2309-2314, Arlington, VA.

[16] McLain, T., Beard, R., & Kelsey, J. (2002). Experimental demonstration of multiple robot cooperative target intercept. *In Proceedings of the AIAA Guidance and Control Conference and Exhibit,* Monterey, CA.

[17] McLain, T., Beard, R., Goodrich, M., & Anderson, E. (2002). Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*.

[18] Mataric, M., Sukhatme, G., & Ostergaard, E. (2003). Multi-robot task allocation in uncertain environments. *Autonomous Robots, 14(2-3),* pp. 255-263.

[19] Nygard, K., Chandler, P.,& Pachter, M. (2001). Dynamic network optimization models for air vehicle resource allocation. *In Proceedings of the American Control Conference (ACC),* pp. 1853-1856.

[20] Parker, L. (1993). Designing control laws for cooperative agent teams. *In Proceedings of IEEE International Conference on Robotics and Automation (ICRA),* pp 582-587, Atlanta, GA.

[21] Pierce, D., & Kuipers, B. (1991). Learning Hill-Climbing Functions as a Strategy for Generating Behaviours in a Mobile Robot. From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior, pp 327-336, Cambridge, MA: The MIT Press/Bradford Books.

[22] Schulz, D., Burgard, W., Fox, D., & Cremers, A.B., (2001). Tracking Multiple Moving Targets with a Mobile Robot using Particle Filters and Statistical Data Association.. *In Proceedings of IEEE International Conference on Robotics and Automation.* Seoul Korea, pages 1665-1670.

[23] Schulz, D. & Burgard, W. (2001). Probabilistic State Estimation of Dynamic Objects with a Moving Mobile Robot. *Robotics and Autonomous Systems,34(2-3).*

[24] Silva, V.T., Carlos, J. P., & Lucena, D. (2003). Doctoral papers: MAS-ML: a multi-agent system modeling language. *In* Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, Anaheim, CA.

[25] Williams, W., Harris, M. (2002). The Challenges of Flight-Testing Unmanned Air Vehicles, in Systems Engineering, Test & Evaluation Conference, Sydney, Australia.

[26] Yuret, D., and Maza M. (1993). Dynamic Hill-Climbing: Overcoming the limitations of optimization techniques. The Second Turkish Symposium on Artificial Intelligence and Neural *Networks,* pp. 208-221.