Presented at the Second 2ND AIAA Unmanned Unlimited Systems Technologies and Operations Aerospace Land and Sea Conference and Workshop & Exhibit, San Diego, CA, 15-18 Sept 2003

SWARMING COORDINATION OF MULTIPLE UAV'S FOR COLLABORATIVE SENSING

H. Van Dyke PARUNAK Sven A. BRUECKNER James J. ODELL

Altarum Institute
3520 Green Court, Suite 300
Ann Arbor, MI 48105 USA

{van.parunak, sven.brueckner}@altarum.org

James Odell Associates
3646 West Huron River Drive
Ann Arbor, MI 48103-9489 USA

email@jamesodell.com

ABSTRACT

Some imaging tasks and modalities (e.g., interferometric SAR) require managing a dynamic spatio-temporal configuration of sensors (whether electro-optic or RF) over a wide area. One promising approach is to mount each sensor on a separate unpiloted vehicle, and endow the population of such vehicles with the ability to configure themselves and coordinate their actions to create and maintain the required sensor configuration. This paper describes some scenarios where such a capability would be useful, identifies technical issues that need to be addressed, suggests general principles and techniques that we have found useful in dealing with such scenarios, and describes a specific example that we have constructed and tested in a simulation environment.

SCENARIOS

A number of important military scenarios would benefit greatly from a swarm of sensor-bearing UAV's. These include:

Situation awareness.—Maintaining situation awareness requires finding, locating, identifying, and tracking potential targets and other relevant objects. Initial object discovery requires a diffuse view of the battlespace, and can be supported by spreading the swarm out widely. Locating objects requires constructing a map of the local environment, a task that can be done collaboratively by position-aware UAV's. Identification can be aided by bringing together multiple sensors to provide more refined imagery. Tracking (especially in urban environments) requires the ability to move with the target.

Urban tomography.—A critical enabler for military operations in urban terrain (MOUT) is identifying the number and distribution of people in a building from the outside (R07, "Through-Wall Sensor"[3]). Human bodies have a very different radar cross-section than building materials. A ring of UAV's flying up the outside of a building could collect reflection and transmission data from different angles at each floor, and use mathematical techniques from medical tomography to map the building interior and locate humans. This vision requires the ability to arrange the UAV's in a robust formation around the building, maintain this configuration as it moves vertically, repair it if individual UAV's are disabled, and assign roles (sender vs. receiver) dynamically.

FOPEN.—L-band radars can penetrate foliage to detect substantial objects (such as trucks or artillery) hidden under the tree canopy, but at these long wavelengths, most reflection is specular, and target identification becomes difficult. This difficulty can be greatly reduced by interferometry among signals from multiple coordinated UAV's. Such a technique permits the reconstruction of the height of detected objects (or, with a sufficient number of sensors, full 3D imaging).

These scenarios, and others like them, have a number of common requirements that swarms of autonomous UAV's can address.

- Sensory input is needed from different spatial locations under tight temporal constraints.
- The separation of these locations is often greater than could be achieved by mounting sensors at fixed locations on a single platform (e.g., at opposite ends of an airplane wing).
- The relative alignment of the sensors, and their roles, often needs to change in response to events in the battlefield. A static configuration of sensors will not meet the needs of the dynamically changing environment anticipated in these scenarios.

ISSUES

Coordinating multiple UAV's for such sensing scenarios requires spatial and temporal coordination and the alignment of distinct roles within the team. *Spatial coordination* distributes units over the area being observed, and includes such tasks as determining the maximum spread between vehicles and the minimum acceptable number of revisits per unit area, assigning sectors to each unit, causing a team to converge in a

specific location, or stationing UAV's in a particular formation. *Temporal coordination* ensures that all UAV's act at the right time or with the right frequency, provide their input at the right moment, and assume their designated locations and operating roles at the right time for the constellation to work as a whole. *Team coordination* seeks to optimize the assignment of individual vehicles to roles in terms of their preferences or constraints (e.g., the configuration of individual vehicles), managing the formation, coordination, maintenance, and dispersion of groups of vehicles.

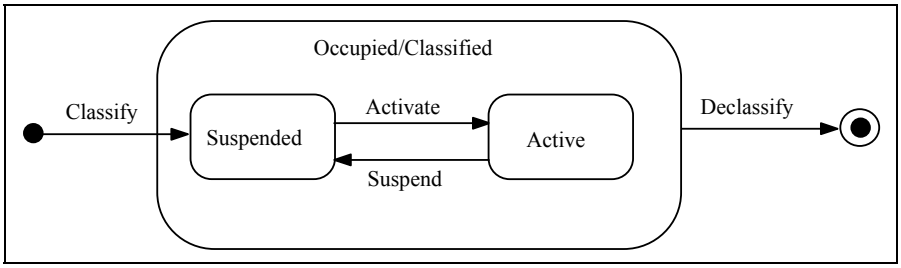

Figure 1.—Statechart depicting some of the permitted states and transitions of an agent in a role.

Conventional (non-autonomous) approaches require humans to fly each UAV. This approach is costly both in terms of manpower and (since all communications must go through the control center) bandwidth, and in addition can be a difficult cognitive task. Autonomous coordination among the aircraft permits the use of local nearest-neighbor communications (reducing bandwidth), and our experiments show that simple local algorithms can yield robust self-organization sufficient to satisfy these missions.

GENERAL PRINCIPLES AND TECHNIQUES

We have found three classes of principles and techniques to be useful in addressing problems of this nature. The first class concerns the relation between individual agents and the groups of which they are a part, and falls under the general rubric of "roles." The second concerns mechanisms for optimizing such systems in the face of resource constraints. The third is the broad area of mechanisms inspired by natural systems.

Issues in Team and Role Coordination

Effective modeling and design of agents in emergent swarms is greatly facilitated by identifying distinct "roles," or patterns of behavior, that agents can "play" in different mission settings.[13] A role is a class that defines a normative behavioral repertoire of an agent. Roles provide both the building blocks for agent social systems and the requirements by which agents interact. Each agent is linked to other agents by the roles it plays by virtue of the system's functional requirements—which are based on the expectations that the system has of the agent. The static semantics of roles, role formation and configuration, and the dynamic interactions among roles have been examined closely in recent years,[4-7, 13] and an initial axiomatization has been proposed.[6] However, little work has been done on formalizing the temporal aspects of dynamic role assignment. Role modelers refer only informally to actions such as "taking on a role," "playing a role," "changing roles," and "leaving roles." The ambiguities inherent in these terms pose difficulties for applications such as ours, in which dynamic role change is a pervasive feature of the system's behavior.

To understand these issues better, we distinguish two aspects to a change in role, summarized in Figure 1 and discussed more fully elsewhere.[9] Role *classification* gives an agent the methods necessary to execute the behaviors in a role). Role *activation* captures the sense that an agent is currently executing in a role.

Dynamic Classification

Dynamic classification refers to the ability to change the classification of an entity. Consistent with the proposed axiomatization,[6] we insist that each agent have at least one role at all times. Dynamic classification deals with adding additional roles or removing roles beyond the minimum of one. This requirement is analogous with the notion that every human must play the "person" role, whatever other roles they may have. In the case of humans, this minimal role persists throughout the agent's life. It is conceivable that an artificial agent might begin with the minimal role A, add role B, then remove role A, leaving it with the minimal role B. Whether such a fundamental redefinition of the agent is possible will depend on such features as physical equipment associated with the agent and the nature of the platforms on which the agent can run. An alternative approach is to define a basic role AgentId that belongs to every agent, whatever other roles it may play. (Id in AgentId recalls the Freudian notion of primal basic urges, not "Identity.") Having AgentId as a role is a controversial point. However, elsewhere[8] we have defined *role* as a class that defines a normative behavioral repertoire of an agent. The basic class AgentId defines the normative behavioral repertoire for agenthood.

Table 1: Operators for Dynamic Classification

Operator	Pre-state	Post-state
Classify	A and not B	A and B
Declassify	A and B	A and not B
Reclassify	A	B
Create	Null	A
Terminate	A	Null

To become an instance of a given role, the agent is *classified* as an instance of, or *occupies*, that role. Once classified, the agent occupies the new role and possesses all of its features. In the opposite process, if an agent is *declassified*, it is removed as an instance of a particular role—and no longer occupies the role nor possesses features unique to that role. An agent is said to be *reclassified* when it is both declassified in one role and classified as another. Agent instantiation and deletion are limiting cases of changes in classification, and we describe their consequences at the role level with *create* and *delete* operators. Table 1 summarizes roles held by an agent before and after each of these operators.

Dynamic Activation

In addition to changing roles over time (dynamic classification), an agent may have multiple roles that apply to it at any one moment, a condition that we describe as *multiple classification* (not to be confused with multiple inheritance). Role *activation* seeks to capture the intuition that an agent may hold multiple roles concurrently while not actively executing them at the same instant.

Formalizing such a notion of "activity" is problematic. In some sense, even a quiescent agent that is waiting for a message or some signal to awaken could be considered active in its role, because alertness can be thought of an activity. UML 2.0[10] offers a useful refinement by distinguishing between user-defined actions (which are represented explicitly in sequence diagrams and activity diagrams) and fundamental system actions such as i/o, invocation, and data flow (which are not represented as actions in these diagrams). In UML, each activation, or *execution occurrence*, has some duration and is bounded by a start and stop point. We propose to take advantage of this refinement in the following unification:

- *We adopt the UML 2.0 definition of action.* Any unit of behavior that has started and has not yet ended is considered "active." Otherwise it is "inactive."
- *We use the basic role of* AgentId *to specify primitive behavior.* Behaviors such as controlling, handling data flows, and waiting for messages and signals are "primitive" actions that all entities must possess to be agents. Therefore, any entity playing the role of AgentId can exhibit this basic behavior, deferring "higher-level" behavior to user-specified actions in more specialized roles. Furthermore, these basic behaviors are themselves actions. For example, actions that support listening for messages and signals, *by definition,* begin the moment an entity is classified an AgentId and cease when the entity is no longer an AgentId.
- *We consider roles other than* AgentId *to be active only when their user-defined actions are active.* Activity of primitive actions is attributed to the concurrently executing AgentId role, not to the user-specified role.

Table 2: Operators for Dynamic Activation

	Prestate		Poststate	
Operator	Active	Suspended	Active	Suspended
Activate	A	B	A and B	
Suspend	A and B		A	B
Shift	A	B	B	A

Dynamic activation involves the operators *activate*, *suspend*, and *shift*. Table 2 summarizes the role assignments as affected by these operators.

Resource Constrained Local Optimization

In the execution of a particular mission, we deploy many simple UAV's in a mission swarm and task the swarm (not the individual UAV) with the mission goal (e.g., imaging of a particular target with a minimum image quality). Once tasked, the UAV's of the swarm coordinate their individual activities (see Team and Role Coordination) to achieve the mission goal.

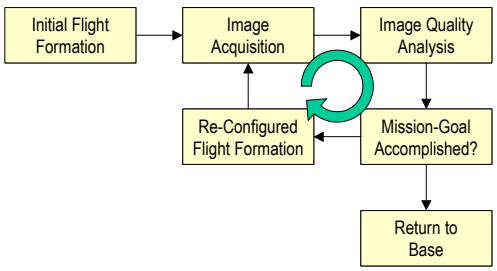

Figure 2. A swarm needs to adjust its execution to meet the mission goal.

Especially if a mission takes a long time to complete, we need to endow the swarm with the ability to judge the current quality of the mission achievement (e.g., the quality of the best image it acquired thus far) and to adjust its execution accordingly. In the case of distributed imaging of a target, the UAV may

have acquired a low-quality image in a particular flight formation already, and now the swarm needs to re-configure the formation to improve the image quality until the requirements of the mission goal are met. Figure 2 shows the resulting closed adjustment loop.

Adapting the swarm's mission execution to the quality of the mission goal achievement requires a) that the swarm is able to perceive the current quality, and b) that the swarm knows how to reconfigure to improve this quality. Of course, since there is not really a "swarm" entity, these two issues immediately translate to the need for the individual UAV to know whether and how it needs to change its behavior to improve the overall (global) system performance.

In the following paragraphs, we will contrast three approaches to the re-configuration problem. We evaluate the applicability of these approaches with respect to the effort involved in the specification of the knowledge of the individual UAV and the processing and communication effort that arises in the execution.

All three cases assume local autonomous control by the UAV (no central controller). The re-configuration process is iterative: each UAV runs through multiple cycles of perception (learning the locations of other UAV's), planning (selecting a new location), and execution (approaching the selected location). The planning of the new location assumes that if no other UAV were to relocate, the resulting new formation would achieve a better image quality. As movements of other UAV's invalidate this assumption, the process repeats.

Implicit Local Model

The implicit local model is a (numerical) function given to the UAV that takes as an input the state of the swarm (locations) as perceived by the UAV and returns the new location that the UAV should assume.

Consider for example distributed SAR imaging. We know that a formation that approximates a regular array results in good images. So, an implicit local model for the UAV would be a function that computes from the current formation a location for the UAV that would make the whole arrangement more regular.

To provide the UAV with an implicit local model, we need to understand the characteristics of those emergent UAV patterns that lead to good performance and we need to craft the local transformation function that determines the best change for one UAV given the overall arrangement. Furthermore, since the swarm reconfiguration emerges from the decisions of the individual UAV without any representation of the global state or goals, the design process must include extensive validation of the implicit model. Thus, the approach requires rather complex knowledge engineering before deployment. In compensation, the cost incurred by the approach during execution is minimal, since the UAV only needs to execute the transition function once to complete the planning process.

Fitness Evaluation

If the construction of a function that generates the new location directly is not feasible, a local search approach may be followed. Our second approach constructs a fitness function that translates a given state of the swarm (UAV locations) into a fitness value (e.g., a number in [0,1]). Thus, the individual UAV may perform a search for a nearby peak in the fitness landscape that is spanned by the (virtual) variation of the UAV's location in the overall swarm arrangement.

In the case of distributed SAR imaging, a useful fitness function might evaluate how much a given set of UAV locations resembles a regular array. The individual UAV would then seek to improve the image quality achieved by the formation by assuming a new location that results in a more regular array.

The reasoning process of the individual UAV is still partially implicit. Though the use of the fitness function now explicitly analyses the global state of the swarm (how close to a regular array is the formation), it still includes the implicit assumption that a particular UAV arrangement will result in good image quality. So, from a knowledge engineering perspective, the designer still has to understand the link between the state of the swarm and the quality of the mission performance.

The use of a fitness evaluation of possible swarm states results in a higher computational effort during the execution of the mission. The UAV must generate and evaluate possible formations that only vary its own location relative to the currently perceived arrangement of the UAV. Once it finds a formation with a sufficiently improved fitness, it will take its own location in this formation as the goal for its subsequent relocation process.

Quality Evaluation

Specifying a metric that explicitly measures the quality of the mission achievement for a given state of the swarm is the most explicit approach. Thus, instead of assuming that a certain state results in a certain quality (e.g., regular array for SAR imaging), this approach translates a given state into an expected outcome and then explicitly measures the quality of the result.

In the case of distributed SAR imaging, we would have to construct an image simulator that translates a UAV formation into the expected image. Then we would ap-

ply standard image-quality metrics that measure, for instance, the contrast or spatial resolution that would be achieved.

The knowledge engineering process for this approach is relatively simple and completely driven by the final goal of the respective mission. The design includes modeling the process of mission execution (the image acquisition and SAR processing) and the specification of mission quality metrics.

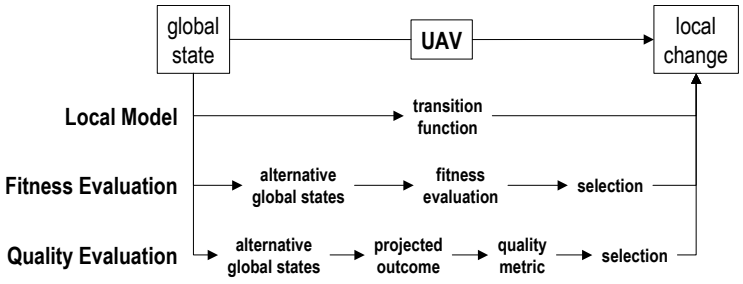

Figure 3. Three approaches to mission quality optimization.

Depending on the computational cost of predicting the outcome for a particular swarm state, the execution of the mission optimization process can be very expensive. As in the case of the fitness evaluation of a given state, the UAV performs a search across possible formations, but rather than applying a single function, the UAV must first estimate the mission outcome and then apply the quality metric.

Summary of Optimization Methods

If we deploy swarms of many simple UAV to perform an extended mission, we need to endow the swarm with the ability to adapt its mission execution based on the quality of the quality of the goal achievement. The autonomy of the UAV and the limited availability of processing and communications resources as well as the uncertainty and noise in the interactions with the physical environment (sensing and acting) require that the individual UAV change its behavior locally to improve the overall mission performance.

Figure 3 summarizes our three approaches to local optimization of the mission performance. They vary in the degree to which the individual UAV must be aware of the current mission quality and means to improve it. Endowing the UAV with an implicit local model results in the fastest and simplest execution of the swarm, but requires a significant effort in designing and validating the model. Using a fitness function that evaluates the quality of a given formation with respect to the implicitly assumed mission performance relieves the designer of the burden of constructing a complete model since it applies a heuristic search on potential alternatives, but still requires off-line specification of what a "good" swarm state would be. The simulation of the outcome produced by alternative states restricts the knowledge engineering to the description of the operation of the system, but then requires the UAV to figure out the link between the process and appropriate optimization strategies.

The specific characteristics of the three proposed approaches suggest the following simulation-based transition path. In the first stage, we apply our knowledge of the general process to optimize the mission performance based on the quality evaluation approach (simulate outcome and measure quality). We then perform extensive simulations of the intended mission and characterize the swarm states that result in good mission performance (e.g., regular arrays in SAR). Based on this characterization, we construct a fitness function that measures the degree to which a given state deviates from the description of a "good" state. We replace the prediction and quality metric with the fitness function and validate its performance in the simulation. Finally, we analyze the nature of the relocation decisions that the individual UAV make and construct and evaluate the implicit local model from which the "good" global states emerge. At this point, the computational effort required for the execution of the mission optimization is sufficiently small to deploy the decision logic onto a real resource-constrained UAV.

Table 3: Natural Examples of Swarming

Swarming Behavior	Entities
Pattern Generation	Bacteria, Slime Mold
Path Formation	Ants
Nest Sorting	Ants
Cooperative Transport	Ants
Food Source Selection	Ants, Bees
Thermoregulation	Bees
Task Allocation	Wasps
Hive Construction	Bees, Wasps, Hornets, Termites
Synchronization	Fireflies
Feeding Aggregation	Bark Beetles
Web Construction	Spiders
Schooling	Fish
Flocking	Birds
Prey Surrounding	Wolves

Inspiration from Natural Systems

Our approach to autonomous coordination among multiple entities is based on principles observed in biological communities, which we have outlined elsewhere.[11] Table 3 gives examples of the kinds of behavior that these techniques can support. Broadly speaking, these techniques achieve self-organization in

multi-agent systems by way of local interactions.

Elsewhere, we have reported how these techniques can be applied to such practical problems as coordinating manufacturing operations,[1] planning paths and deconflicting airspace for UAV's,[14] and recognizing patterns in a distributed sensor network without central processing,[2] and compared our approach (based on agent interactions through a shared environment) with others.[12]

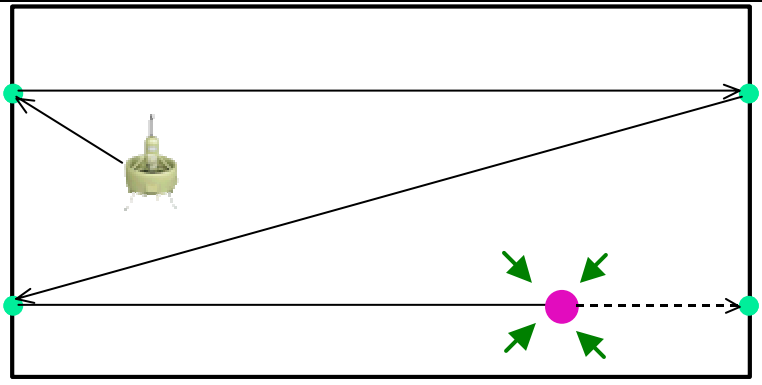

Figure 5: Individualistic search strategy

A SPATIAL COORDINATION EXAMPLE

As an example, we describe a swarm that we have configured to do target location and imaging in a FOPEN scenario. In this scenario, the swarm must achieve three objectives that require different behaviors on the part of individual UAV's.

In **searching**, it must effectively cover a large search space and revisit locations regularly, maximizing detection probability based on known characteristics of the target (e.g., visibility angle), while not exhibiting any obvious systematic search patterns that would permit mobile targets to execute simple avoidance strategies.

When a vehicle detects a target, it announces the location of the target, and vehicles that receive this announcement begin **imaging.** In this phase, a vehicle must collect data from varying angles along linear trajectories (box) while minimizing both the effort (the number of required vehicles and the distance they must move) and the data collection time (by collecting data in parallel).

In addition, individual vehicles require periodic refueling or other **maintenance**, and the swarm must ensure that individual vehicle requirements are met without compromising the ability of the overall swarm to continue functioning.

We have defined two mechanisms to address this problem. The first, an individualistic approach, has a minimum of inter-UAV coordination, and permits any UAV to communicate with any other. The other approach, a team approach, relies on local inter-UAV communication, and increases the coordination among the platforms. Different mechanisms can be applied to different behaviors in the same swarm.

Individualistic Approach

During **search**, each vehicle executes its own search pattern. These patterns are specified off-line to result in maximum global dispersion. In our current implementation, each vehicle selects a point at random on the western side of the search area, and flies toward it in a straight line (Figure 5), then flies due east, then repeats, until it detects a target or receives a message from another vehicles that has detected a target. When a vehicle detects a target, it broadcasts the location of the target to all the other vehicles.

During **imaging**, each vehicle chooses a linear trajectory that passes by the announced target location, executes a data collection flight, and communicates results, then resumes search behavior.

To accomplish **maintenance**, each time a UAV reviews its current role, it evaluates an exponential probability distribution over its fuel level. Full UAV's almost never return for refueling, while almost empty ones are highly likely to return. The stochastic nature of the decision breaks the symmetry among UAV's with similar fuel levels, and the swarm stabilizes in a state where a fixed proportion is engaged in maintenance at any time, assuming similar fuel consumption on the part of all UAV's.

The individualistic approach has the benefit of requiring fewer messages and less reasoning effort. However, it cannot adjust to changing conditions (e.g., the loss of a team member), overlap among the vehicles makes search and data collection sub-optimal, and the use of broadcast communications requires higher comms power levels.

Team Approach

The team approach uses a digital pheromone mechanism inspired by coordination in insect systems and similar to one we have used for real-time path planning for UAV's.[14] An important difference is that while that application envisioned a network of unattended ground sensors maintaining the pheromone field *externally* to the agents, in this case each agent maintains an *internal*

pheromone map that tiles the search space into discrete cells. Each cell is a place in a pheromone infrastructure, which means that the agent that controls the vehicle may deposit and sense digital pheromones of different flavors in that cell. The place aggregates deposits of the same flavor, propagates (shares) deposits to neighboring places, and evaporates (reduces) pheromone concentrations over time. The general dynamics of our pheromone infrastructure model are described in more detail elsewhere.[1]

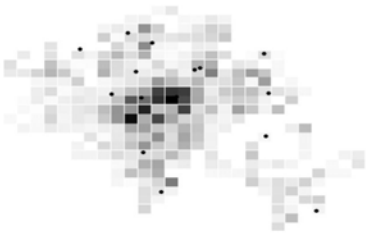

Figure 6: Visitation Pheromone Map of one UAV in the Swarm

During **search,** when a vehicle passes through the area in the search space that is assigned to a particular cell, it deposits a unit of the visitation pheromone into that cell in its internal map. In addition, the agent broadcasts its location, and the agents of any other vehicle within communications reach then deposit a visitation pheromone into their maps too. Thus, the agents mark cells that some member of the swarm has already visited. Figure 6 shows a snapshot of the visitation pheromone map of one agent in the swarm.

Local concentrations of pheromones lose strength over time, which enables the swarm to "forget" visitations to locations that occurred a long time ago. This knowledge management process ensures that the search process keeps revisiting locations in case targets have moved in.

The individual agent decides its vehicle's trajectory based on its internal map of visitation pheromones. Once it has reached its previous goal, the agent probabilistically selects a new location. The probability of the selection of a particular location is inversely proportional to its distance to the vehicle's current location and to the strength of the visitation pheromone concentration in the cell that covers this location. Thus the agents tend to prefer nearby locations that have not been visited recently, and collectively explore the whole search space.

An agent that detects a potential target dynamically forms an **imaging** team. Team formation is a collaborative process in which agents bid for a role in the team depending on the match of the vehicles' imaging capabilities with the role's requirements (hard constraint) as well as the current distance of the eligible vehicles to the detected target (soft constraint).

Once roles are assigned, the team members plot the optimal trajectories for their respective data acquisition flight and execute the imaging task. Depending on the imaging modality (coherent vs. non-coherent), the data acquisition may be executed individualistically or synchronized across the team. Once the task is completed, the team disbands and the agents resume their search behavior.

A team-based approach to **maintenance** can accommodate UAV's with different fuel consumption rates, as well as variations in the availability of maintenance resources at the base. UAV's deposit a pheromone flavor that communicates the intensity of their current desire for maintenance, while the base propagates a pheromone indicating its current level of load. A UAV's decision to shift into the maintenance role is promoted by its own desire for refuel and inhibited by the level of refueling pheromone it senses from neighboring UAV's and the load pheromone propagated from the base.

The team approach requires more computational effort, and the communications among platforms makes it more vulnerable to electronic countermeasures. However, it can adapt explicitly to the consequences of past actions and the state of the current problem, and reduce duplication among platforms.

Demonstration

To demonstrate these approaches, we distribute targets randomly under foliage in a rectangular search area. When a vehicle is near a target, it has a probability of detecting the target's presence, a probability that is symmetric over the center of the target.

Table 4 lists the approaches that we have implemented for each of the three roles in the problem. We can mix and match these mechanisms in a single configuration. Currently tested combinations are team search with individualistic imaging, and individualistic search with team imaging. In all cases, maintenance is currently done in individualistic mode.

Table 4: Implemented Mechanisms for UAV Behaviors

	Individualistic	Team
Search	X	X
Imaging	X	X
Maintenance	X	

Figure 7 shows the performance of the configuration using individualistic search and team imaging. An important benefit of team imaging over indi-

vidualistic imaging is that only the required number of UAV's needed for imaging are distracted from the search task. The others continue to search, and if another target is detected, multiple imaging teams can work concurrently.

Discussion

This example illustrates the utility of the three principles that we discussed earlier.

Roles Analysis

A UAV's shift among search, imaging, and maintenance is a showcase example of the usefulness of role-oriented design for high-level agent specifications. In both individualistic and team approaches, the code for each UAV includes all three roles, so there is no dynamic classification. In both cases, dynamic activation is invoked in the form of the *shift* operator between distinct roles. However, the invocation of *shift* is different in the two cases. In the individualistic approach, agents invoke the *shift* whenever they receive a report of a target. In the team approach, the *shift* from search to imaging takes place only after an agent wins the bidding process. In both approaches, the *shift* from imaging to search is unilateral, after the imaging run has been completed.

Optimization

The system demonstrated in Figure 7 uses an implicit local model for optimization. The algorithm used by a single UAV is based on our knowledge that flying a rectangle around the target gives a good image, so we can translate target location directly into recommended trajectories. The agents do not know anything about the quality of their mission performance, or how to translate such knowledge into waypoints. This level of optimization economizes run-time execution at the expense of up-front engineering to determine the parameters required for effective performance. Such implicit models are highly desirable for efficient deployment on UAV's especially in support of small inexpensive platforms, and methods for their systematic development are an important topic for research.

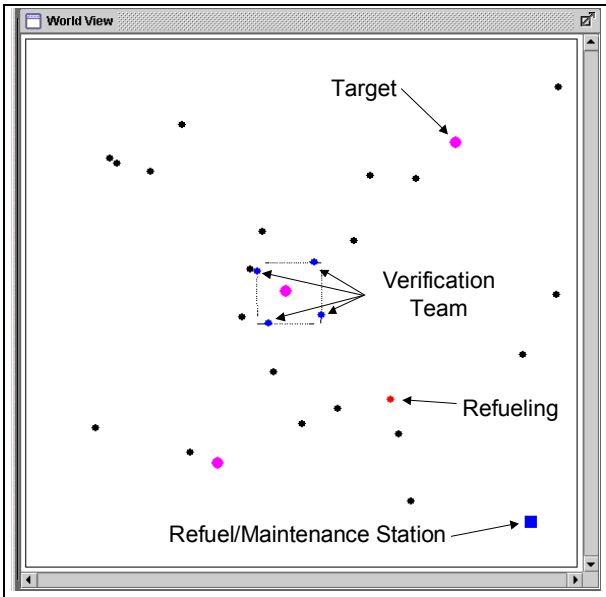

Figure 7: Swarm in Imaging Phase

Natural Systems Analogs

The use of digital pheromones in the team coordination approach is a direct application of environmental coordination mechanisms inspired by natural systems, in particular colonies of social insects. These systems are of particular value in applications that are discrete, distributed, decentralized, and dynamic, and all four characteristics are important for our application. These algorithms are appropriate systems of *discrete* entities, such as a swarm of UAV's. Digital pheromones enable *distribution* of system knowledge over the swarm, since each UAV maintains a local pheromone map of its vicinity, and *decentralization*, since all UAV's are peers in the coordination algorithm, making the system robust to loss of any individual platform. Perhaps the most important benefit of pheromone techniques in this application is their support for *dynamic* environments. A given UAV's environment (which includes the other members of the swarm) is constantly changing, and coordination mechanisms based on traditional knowledge bases face a huge challenge in maintaining consistency. Pheromone maps are constantly evaporating, disposing any information that is not being reinforced by current observation, and thus automatically discard obsolete information and keep themselves consistent.

CONCLUSION

An important application for swarms of UAV's is coordinated sensing tasks. We have developed a suite of design principles and algorithmic approaches to this coordination, and demonstrated their effectiveness in software simulations.

ACKNOWLEDGEMENTS

This work is supported in part by DARPA, contract F30602-02-C-0196 to Altarum, under DARPA PM Vijay Raghavan. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

REFERENCES

1 S. Brueckner. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Dr.rer.nat. Thesis at Humboldt University Berlin, Department of Computer Science, 2000. http://dochost.rz.hu-berlin.de/dissertationen/brueckner-sven-2000-06-21/PDF/Brueckner.pdf.

2 S. A. Brueckner and H. V. D. Parunak. Swarming Agents for Distributed Pattern Detection and Classification. In *Proceedings of Workshop on Ubiquitous Computing, AAMAS 2002*, Bologna, Italy, 2002. http://www.erim.org/~vparunak/PatternDetection01.pdf.

3 CALL. MOUT ACTD Overview Briefing. 2002. PDF file, http://call.army.mil/Products/mout/MOUT-ACTD.pdf.

4 C. Castelfranchi. Engineering Social Order. In, *Engineering Societies in the Agent World*, pages 1-18. Springer, Berlin, 2000.

5 M. Dastani, V. Dignum, and F. Dignum. Role Assignment in Open Agent Societies. In *Proceedings of Second International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS'03)*, Melbourne, Australia, pages (forthcoming), 2003.

6 J. Ferber, O. Gutknecht, and F. Michel. Agent/Group/Roles: Simulating with Organizations. In *Proceedings of Fourth International Workshop on Agent-Based Simulation (ABS03)*, Montpellier, France, 2003.

7 J. Odell, H. V. D. Parunak, and B. Bauer. Representing Agent Interaction Protocols in UML. In *Proceedings of Agent-Oriented Software Engineering*, 22nd International Conference on Software Engineering, pages 121-140, Springer, 2000. http://www.jamesodell.com/Rep_Agent_Protocols.pdf.

8 J. Odell, H. V. D. Parunak, and M. Fleischer. The Role of Roles in Designing Effective Agent Organizations. In A. F. Garcia and et al., Editors, *Software Engineering for Large-Scale Multi-Agent Systems*, Springer-Verlag, Berlin, 2003.

9 J. J. Odell, H. V. D. Parunak, S. Brueckner, and J. Sauter. Temporal Aspects of Dynamic Role Assignment. In *Proceedings of Workshop on Agent-Oriented Software Engineering (AOSE03) at AAMAS03*, Melbourne, AU, pages (submitted), Springer, 2003. http://www.erim.org/~vparunak/TemporalAspects03.pdf.

10 OMG. UML. 2003. Web Site, http://www.omg.org/uml/.

11 H. V. D. Parunak. 'Go to the Ant': Engineering Principles from Natural Agent Systems. *Annals of Operations Research*, 75:69-101, 1997. http://www.erim.org/~vparunak/gotoant.pdf.

12 H. V. D. Parunak. Making Swarming Happen. In *Proceedings of Swarming and Network-Enabled C4ISR*, Tysons Corner, VA, ASD C3I, 2003. http://www.erim.org/~vparunak/MSH03.pdf.

13 H. V. D. Parunak and J. Odell. Representing Social Structures in UML. In M. Wooldridge, G. Weiss, and P. Ciancarini, Editors, *Agent-Oriented Software Engineering II,*, *Lecture Notes on Computer Science*, pages 1-16. Springer, Berlin, 2002. http://www.erim.org/~vparunak/AOSE2001.pdf.

14 H. V. D. Parunak, M. Purcell, and R. O'Connell. Digital Pheromones for Autonomous Coordination of Swarming UAV's. In *Proceedings of First AIAA Unmanned Aerospace Vehicles, Systems,Technologies, and Operations Conference*, Norfolk, VA, AIAA, 2002. www.erim.org/~vparunak/AIAA02.pdf.