Pheromone inspired Morphogenic Distributed Control for Self-Organization of Unmanned Aerial Vehicle Swarm

Kiwon Yeom

Human Systems Integration Division, NASA Ames Research Center, Moffett Field, CA 94035-0001, USA

Abstract: Distributed formation of swarming with no coordinated agreement or positioning information is an interesting research area because the global behaviors must emerge from many diverse local interactions. A central issue in distributed formation of swarm is enabling agents with only a local view of their environment to take actions that advance global system objectives (emergence of collective behavior). This paper describes a bio-inspired control algorithm using pheromone for coordinating a swarm of identical flying agents to spatially self-organize into arbitrary shapes using local communication maintaining a certain level of density. Different from most existing distributed control, the proposed approach considers the topological structure of the organization, supports dynamic reconfiguration and self-organization. This paper presents the experimental results on simulating in the forming of arbitrary shape, and simulating wireless communicated swarm behavior forming communication networks, self-repairing, and avoiding pitfalls in mission execution.
[Kiwon Yeom. **Pheromone inspired Morphogenic Distributed Control for Self-Organization of Unmanned Aerial Vehicle Swarm**. *Life Sci J* 2013;10(3):979-991] (ISSN:1097-8135). http://www.lifesciencesite.com. 143

Keywords: Distributed formation, self-organization, intelligence, UAV, self-reconfiguration, modular flying agents, federation of agents

1. Introduction

In the areas of robotics and distributed systems, a lot of research effort has been developed towards controlling autonomous agents with low power requirements and simple sensor capabilities [1], [2], [3]. We aim at investigating flying agents based on minimal aerial swarm systems which have minimal capabilities for sensing and communications, which can be deployed in real-life scenarios. We believe that this approach could be applicable in most environments in a rapid, inexpensive, scalable and simple manner. This research work is inspired by an application whereby several flying agents have to self-organize autonomously to establish an emergency communication network to detect multiple users located in disaster areas and relay their position information [4], [5], [6].

As the cost of robotic hardware has come down and availability has gone up, there has been growing interest in robotic systems that are composed of multiple simple robots rather than one highly-capable robot. The technology enables new applications, and large numbers of simple, self-directed robots show high potential for use in sensor grids, resource harvesting, and group transport [7]. This trade-off reduces the design and hardware complexity of the robots and removes single point failures, but adds complexity to the algorithm design. There are strong challenges to control and coordinate individual agents, or enable a swarm of agents with minimal communication capability to perform a useful task as a collective behavior.

Nature has shown that complex collective behaviors can be made possible by very simple interactions among large number of agents which are relatively unintelligent [8]. For example, schools of fish swim in unison, and are able to execute large scale collective behaviors to avoid a predator. Termite colonies can build very large and complex nests. Ants collectively search a very large area and are capable of returning food to the nest [9]. In these examples, there is no central leader with all the information-making decisions for each individual. This non-supervised behavior is a central aspect of distributed systems.

Ultimately we would like to control global or collective behavior in a distributed way using flying agents that have limited communication ability and act autonomously. Flying agents can fly over difficult terrain such as flooded or debris areas [10]. Rather than relying on positioning sensors which depend on the environment and are costly, flying agents rely on proprioceptive sensors and local communication with neighbors (see Fig.1).

The two-dimensional formation task is the starting point of this research, as it is simple to describe swarm applications in terms of theory or practice. For instance, given a set of flying agents and a set of points, the problem to solve is to arrange the agents on the points without being piled up on one another (i.e., one to one correspondence). In real world tasks, often flying agents can be involved to form a particular shape in examples such as sensor grids (or sensor networks) and group transport.

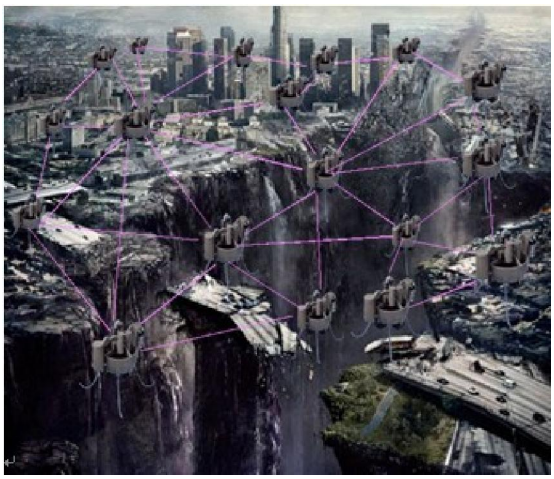


Figure 1. Artistic view of the use of a swarm of UAVs for establishing communication networks between users located on ground

Keeping a specific formation of flying agents is important for many real world tasks, especially when individual agents have limited abilities or the task requires global action. For example, flying agents may aggregate for coordinated search of survivors in a disaster area. Imagine a large group of small unmanned autonomous aerial vehicles that can fly with the agility of a flock of starlings in a city square or of a swarm of donut-like shape avoiding many obstacles [11]. A global shape, which is much larger than an individual agent's communication range, can be formed by collective behavior. It enables a swarm of agents to gather information from a wide area and transport it.

In this paper, we focus on a control algorithm in which flying agents self-organize and self-sustain arbitrary 2D formations. An approach to form an arbitrary shape in two dimensional space called the 'ShapeBugs' is depicted by [12]. It only requires agents to have the ability of local communication, two approximate measures for relative distance and motion, and a global compass. The proposed algorithm proceeds by synchronizing agents' coordinate system and enables agents to form an arbitrary shape. The algorithm is robust but the global compass requires much computational resources to synchronize local agents' coordinate system.

Our work is motivated to modify and extend the ShapeBugs algorithm. We present a modified and fast formation control approach by substitute the global compass with continuous calculation of the error in estimation of local agent's movement. In addition, we introduce pheromone based density control mechanism to manage and keep the overall shape as any failure in agents happens.

We propose a decentralized formation algorithm, which can not only accomplish arbitrary shapes by self-organization but also produces resulting the formed global shapes that are highly robust to varying numbers of agents from agent death. In addition, it can compensate for practical hardware limitations like sensor and movement error.

In this work, we assume that our flying agents are equipped with imperfect proprioceptive sensors and a short-range wireless communication device with which agents can exchange information with only nearby neighbors. Briefly, our algorithm works as follows: first, flying agents initially wander with no information about their own coordinates or their environment. However, they have a programmed internal knowledge of the desired shape to be formed. Next, a small seed group of agents are initially located in a shape. As non-seeded agents move, they continually perform local trilaterations to figure out their location by continuous local communication. At the same time, agents maintain a certain density level among themselves using pheromones and flocking-rule-based distance measurements [13]. This enables flying agents to disperse within the specific shape and fill it efficiently.

This approach has several salient contributions. It only requires agents to have local communication ability, an approximate measure of relative distance, an approximate measure of relative motion. Technically, our system can distribute agents through a specified shape, not merely place them in particular positions on the two dimensional plane. Therefore, agents can easily aggregate into arbitrary pre-defined shapes. The proposed algorithm enables agents to form many arbitrarily connected formations maintaining a certain density regardless of map size, the number of agents, and obstacles.

We show through simulation experiments not only that flying agents can easily aggregate into arbitrary user-defined shapes but also that the formed shape is robust to varying numbers of agents and independent of the number of agents.

The rest of the paper is organized as follows: we present related work in Section 2, and Section 3 describes our flying agent model followed by a detail explanation of the self-organizing formation algorithm in Section 4. We present simulation experiments that investigate efficiency, robustness in Section 5, and draw conclusions and discuss in Section 6.

2. Related Work

The theory and practice of the distributed control of swarms of agents was introduced by Reynolds [14]. The essence of his research work was that coordinated collective behavior can be achieved by a simple distributed interaction between neighbors. Each agent must move according to a small set of simple rules, which take into account the range

bearing, and preferably the orientation, of its neighbors.

There are several literatures that can be considered relevant to robotic self-assembly [15], [16], [17], [18], [12]. These researches addressed the problem of building arbitrary shapes by self-assembling robot swarms. [19] investigated micro aerial vehicles to establish a positionless communication network. However, frequent replacement of node MAVs in the network may drift the swarm from original position.

The idea behind the aerial swarm had its origins in the Beowulf Project [20]. The Beowulf Project demonstrated the possibility of creating a distributed parallel computer interconnecting through an Ethernet network using a number of cheap Linux boxes. [21] remarked that it should be possible to configure a fleet of LinuxBots to operate across the wireless LAN as a Beowulf cluster.

Several projects are aimed at getting UAVs to fly in formation, usually under remote but high-level control [22]. This type of project is therefore different from the biologically-inspired flexibility and responsiveness of flocking pursued within a swarm. However many of the required technologies are similar. The MinuteMan project at UCLA builds a reconfigurable architecture for highly mobile multi-agent systems [23]. The intention is that the computationally capable autonomous vehicle would be able to share information across a wireless fault tolerant network. Study of formation-flying were undertaken at MIT, within the Autonomous blimps project [24]. The University of West England developed the flying flock project slightly different from previous work [25]. The work is conceived with a minimalist approach.

Currently, UAVs are designed to achieve tasks such as the surveillance of an area of interest or searching for targets and subsequently destroying, tracking or sensing them [10] [26] [13]. Other possible applications include environmental monitoring and more specifically toxic plume characterization or forest fire detection, and the deployment of mobile communication networks [27].

Several map-based UAV applications are proposed in [28] and [29]. In map-based applications, UAVs know their absolute position which can be shared locally or globally within the swarm. Each agent then decides where to navigate based on its interpretation of the map. UAVs can deposit and sense virtual pheromones, location information visited by robots over time, or areas of interest in the environment.

However, obtaining and maintaining relative or global position information is challenging for UAVs or mobile robot systems. A possible advance is to adopt a global positioning system (GPS). However, GPS is not reliable and rarely possible in cluttered areas [30]. Alternatively, wireless technologies can be used to estimate the range or angle between agents of the swarm. In this case, beacon agents can be used for a reference position to other moving agents. However, depositing beacons is generally not practical for swarm systems in unknown environments [31]. Off-the-shelf sensors such as cameras, laser range finders, radars, ultrasound and infrared sensors are capable of providing relative positioning, but this equipment is typically expensive and heavy and hence incompatible with the scalable nature of swarms composed of large numbers of simple and inexpensive aerial robots.

Our system attempts to achieve connected arbitrary formation using a decentralized local coordinate system of agents with relative distance and density model.

3. Flying Agent Model

We assume a simple aerial robot that moves and turns in continuous space, which is motivated by the pieces of capabilities of real autonomous UAVs. Each robot has several simple equipment such as distance and obstacle-detection sensors, a magnetic compass, wireless communication, etc. (see Table I)

Table 1: Flying agent model

Distance sensor	provides estimated distance of each neighbor
Detection sensor	detects obstacles in direct proximity to robot
Wireless comm.	allows agents to communicate with each other
Locomotion	moves agents in the world but has error
Internal shape map	is specified by user as a target shape

We assume that agents move in 2D continuous space, all flying agents execute the same program, and agents can interact only with other nearby agents by measuring distance and message exchange. We assume that the simulation world is finite for simplifying the handling of agent trajectories, and agents that wander off one side will reappear on the other side. The agents' communication architecture is based on a simple IR ring architecture because we assume that agents can interact only with nearby neighbors. The robots have omnidirectional transmission, and directional reception. This means that when a robot receives a transmission, it knows roughly which direction the transmission came from (see Fig.2). An example of such communication hardware is described in [32] (see Fig. 3).

In order to test the algorithm, we developed a simulation environment based on Swarm (Swarm 2.2 http://www.swarm.org). The simulator models a continuous world in which the robots and obstacles exist and each object occupies some physical extent.

The reason we chose this simulation environment is that the swarm simulator provides a gridded world environment so that the proposed algorithm can face real-world problems.

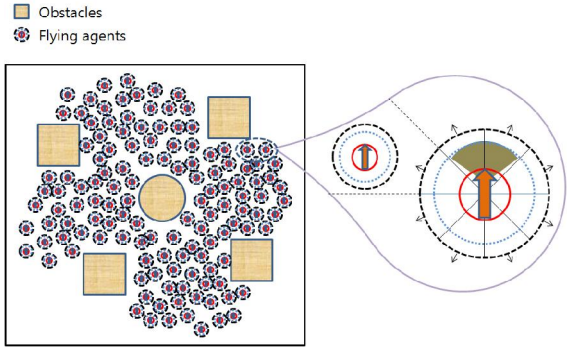
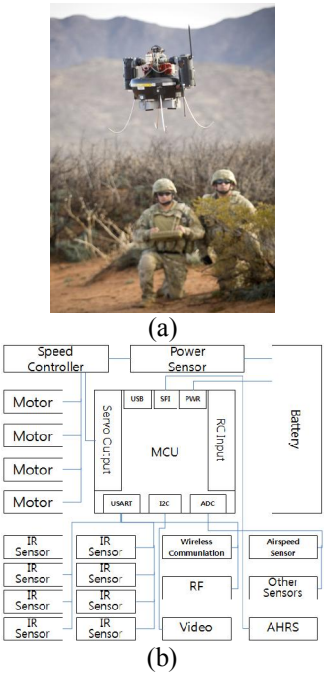


Figure 2. Agent model inspired from capabilities of real UAVs



(a)

(b)

Figure 3. (a) An example of UAVs hardware. (b) UAV hardware architecture.

The agent's dynamic model is implemented using a first order flight model for simple and low-cost airframe. We assume that our UAVs can fly at a speed of approximately 0.5 m/s and are able to hover or make sharp turns as an example in Figure 3. The minimum turn radius of the UAVs is assumed to be as small as possible with respect to the communication range.

Having a realistic communication model is essential for credibility because of the real-life challenge brought on by highly dynamical systems, signal propagation uncertainties and network topologies that are prone to packet collisions.

Later we will consider wireless communication based on the IEEE 802.11b specification, allowing a communication range of around 3 m. This medium might be enough for realistic scenarios because in most potential networks, ground users can use wireless communication devices which are embedded on laptops, smart phones, PDAs etc.

4. Self-Organizing Formation Algorithm

The decentralized control of UAVs has some advantages. First it is highly scalable, i.e., there is no difference between controlling 10 or 100 UAVs. Secondly, no need of information with respect to formation is required. Therefore higher flexibility and higher capacity of manoeuvre can be guaranteed. Thirdly, the swarm control mechanism itself is intrinsically fault tolerant. The failure of an agent or a small group of agents does not compromise the swarm flocking.

In our self-organizing formation algorithm, each flying agent has a shared map of the shape to be constructed and this should be overlaid on the agent's learned coordinate system. Initially, flying agents are randomly scattered into the simulation world without any information about the environment. Then agents begin to execute their programmed process to decide their position using only data from their proximity sensors (i.e., distance and density) and their wireless communication link with nearby neighbors.

The programmed process consists of two continuous and concurrent thread modules. One is for defining the agent's coordinate system. In this module, each agent can adjust its learned coordinate system so that it corresponds to another agent's coordinate system. This is can be accomplished by continuous trilateration using distance. The other is for controlling the agent's movement. If an agent is within a shape, then it plays a role as beacons with a density sensor. Otherwise agents randomly move around in the world until finding an appropriate position.

4.1. Agent Transition Cycle

Agents are simulated in an asynchronous and autonomous manner with finite time required for the calculation of both position and movement. In our model, agents have a simple transition cycle model as shown in Fig.4.

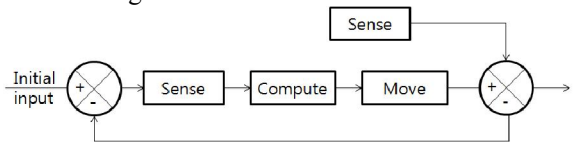


Figure 4. Agent's process cycle

The second sense step is necessary because agents should compare the data before and after

movement to determine distance and orientation from positioning error. After each transition of agent, the time until the next transition is set randomly from [Tmin, Tmax), where Tmin is 0 to the time for computation (see Fig. 4) and Tmax is approximately the time of wait or move. Agent can move only one unit or 0 unit during Move and Sense transition process. Therefore, if agent does not move, the agent's Move does not have any time code.

In a more realistic scenario, agents would move varying distances over time because of both distance measurements and movement error. The positioning process largely relies on the ability of agents to estimate the magnitude of their motions.

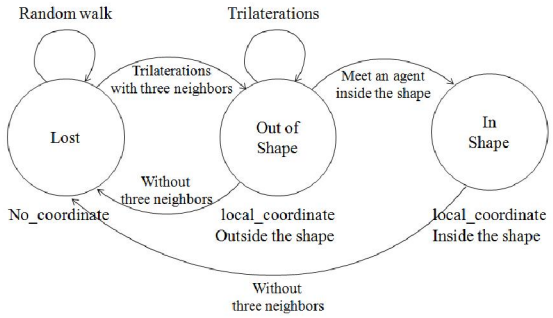


Figure 5. An example of agent's state machine. In this work, the agent does not perform more steps to keep residing inside when it is inside the shape. However, if it cannot do trilateration with other three neighbors, its state change into 'Lost'.

In this model, agents have three computational states such as *lost, out of shape*, and *in shape* (see Fig. 5 as describe in [12]. Although [12] is robust, it does not provide any stable state because there is no definition of simulation complete state. Therefore, in our work, we define the simulation iteration will be complete when all agents make transition and are back to *StanbyQueue* state queue.

Agent's state will be set the *StanbyQueue* as virtual final step if whole agents do not require any more sense because they are inside the shape and has at least three neighbor agents with coordinate system. In addition, agents continuously broadcast their estimated position (x, y), state, and local density. This information therefore is always available for any other agent which is in Sense step. While agent moves, the position is generated in the step of Compute. Any adjacent two agents have the relative distance as well by acquiring its own position and other agent's position.

Although agent which is in the *Lost* state does not have coordinate system, any agent in *In Shape* or *Out of Shape* has a shared coordinate system. Initially, agents are in the lost state because there is no given coordinate system. An agent in *Lost* state will randomly wander through the world until it senses three nearby neighbors which have a coordinate system (see Fig.8). When this condition is satisfied, the lost state agent tries to trilaterate to calculate its position by comparing the neighbor's distance and the relative position. Unlike [12], our model uses 8 IR sensors to approximately sense the direction of the referenced agent. As mentioned earlier, this enables for agent to easily and fast approach towards inside the target shape. Once agents acquire a shared coordinate system, they begin to fill a formation shape by each agent diffuses pheromone with repulse range R_{rep} (see Fig. 10). The pheromone emission mechanism allows the formation shape to be robust against agents' death, while agents evenly spread out throughout the shape.

4.2. Hybrid Agent Positioning Algorithm

In geometry, trilateration is the process of determining absolute or relative locations by measurement of distances using the geometry of circles or triangles. Trilateration does have practical applications in navigation including global positioning systems (GPS).

In two-dimensional geometry, when it is known that a point lies on two curves such as the boundaries of two circles then the circle centers and the two radii provide sufficient information to narrow the possible locations down to two [33].

Trilateration allows an agent to find its perceived position (x_p, y_p) on the connected coordinate system (see Figure 5). It is also used subsequently to adjust its position. In this work, the trilateration process occurs only if there are at least three neighbors that are not in the lost.

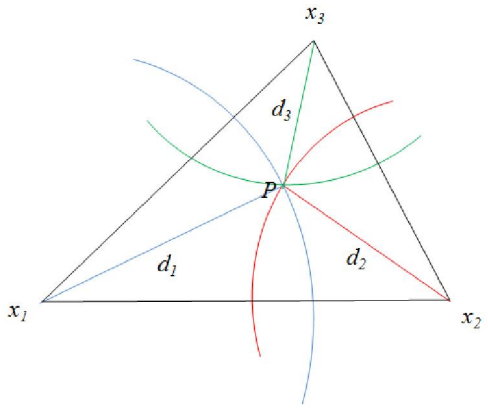


Figure 6: Agent's trilateration

An agent uses its distance sensor to estimate its distance to each neighbor agent and also to request their own learned coordinate systems by wireless communication.

Let the positions of the three fixed anchors be defined by the vectors x_1, x_2, and $x_3 \in \Re^2$. Further, let $x_p \in \Re^2$ be the position vector to be determined. Consider three circles, centered at each anchor, having radii of d_i meters, equal to the distances between x_p and each anchor x_i. These geometric constraints can be expressed by the following system of equations:

$$\|\vec{x}_p - \vec{x}_1\|^2 = d_1^2 \quad (1)$$
$$\|\vec{x}_p - \vec{x}_2\|^2 = d_2^2 \quad (2)$$
$$\|\vec{x}_p - \vec{x}_3\|^2 = d_3^2 \quad (3)$$

$$2(\vec{x}_2 - \vec{x}_1)\cdot\vec{x}_p = d_2^2 - d_1^1 - \|\vec{x}_2\|^2 + \|\vec{x}_1\|^2 \quad (7)$$
$$2(\vec{x}_3 - \vec{x}_1)\cdot\vec{x}_p = d_3^2 - d_1^1 - \|\vec{x}_3\|^2 + \|\vec{x}_1\|^2 \quad (8)$$
$$\|\vec{x}_p\|^2 - 2\vec{x}_1\cdot\vec{x}_p + \|\vec{x}_1\|^2 = d_1^2 \quad (4)$$
$$\|\vec{x}_p\|^2 - 2\vec{x}_2\cdot\vec{x}_p + \|\vec{x}_2\|^2 = d_2^2 \quad (5)$$
$$\|\vec{x}_p\|^2 - 2\vec{x}_3\cdot\vec{x}_p + \|\vec{x}_3\|^2 = d_3^2 \quad (6)$$

$$\|\vec{x}_p - \vec{x}_i\|^2 = \|\vec{x}_p\|^2 - 2\vec{x}_i\cdot\vec{x}_p + \|\vec{x}_i\|^2$$

Since, the above equations can be rewritten as follows:

$$\mathcal{N} = \begin{pmatrix} \vec{x}_2 - \vec{x}_1 \\ \vec{x}_3 - \vec{x}_1 \end{pmatrix} \quad (9)$$

$$\vec{\xi} = \begin{pmatrix} d_2^2 - d_1^1 - \|\vec{x}_2\|^2 + \|\vec{x}_1\|^2 \\ d_3^2 - d_1^1 - \|\vec{x}_3\|^2 + \|\vec{x}_1\|^2 \end{pmatrix} \quad (10)$$

by subtracting the second and third equations from the first, results in the following two equations:

by solving the following linear system, the column vector can be determined:

Generally, the best fit for x_p can be regarded as the point that minimizes the difference between the estimated distance (ξ) and the calculated distance from x_p (x_p, y_p) to the neighbors reported coordinate system. That is,

$$\arg\min_{(x_p,y_p)} \sum_i \left| \sqrt{(x_i - x_p)^2} - \xi \right| \quad (11)$$

From this information, we can learn that this problem is closely related to the *sum-minimization problems* that arise in least squares and maximum-likelihood estimation. Therefore we suggest simple way of search of local minima (see Eq.12 Eq.13). However, in this paper, we do not consider finding any optimal or global solution but a local minimum, because it requires a lot of computational resources and it is not suitable for a small and inexpensive device. For simplification, formula 11 can be rewritten in the form of a sum as follows:

$$Q(w) = \sum_i^n Q_i(w) \quad (11)$$

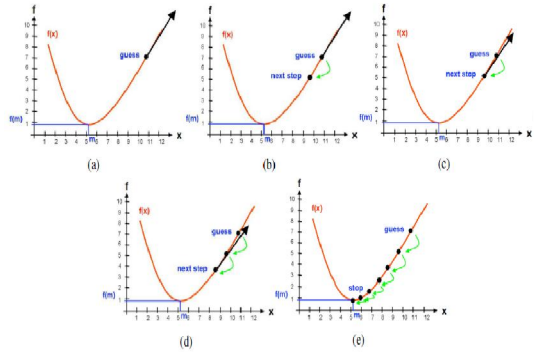
where the parameter w is to be estimated and where typically each summand function $Q_i()$ is associated with the i-th observation in the data set. We perform Eq.6 to minimize the above function:

$$w := w - \alpha\nabla Q(w) = w - \partial\sum_{i=1}^n \nabla Q_i(w) \quad (12)$$

where α is a step size. For easy understanding, we draw the ictures in Fig.7.

We found the success of the above iteration procedure pends on the initial starting position and search step size. Positioning is usually an approximate process and hence may have errors. Firstly, the measurement of measurement of distance by distance sensors has error. Secondly, the reported coordinate system from neighbors may be inaccurate. Thus, the agent can have a movement error after positioning. This means that an agent has moved distance d in some direction may actually move $d + \delta$. As a result, the consensus coordinate system accumulates errors over time. Therefore we readjust the coordinate system at certain intervals.

Figure 7. (a) Initial guessing for deciding direction to find local minima. (b)(e) stepwise searching the local minima.



4.3. Flocking Movement control

As described in the previous section, the agent has three states such as *Lost*, *Out of Shape*, and *In Shape*. Agents take different movement patterns according to their states. If an agent is in the lost state, it is assumed that the agent is located outside the shape or is in the initial simulation starting status. If they are outside the shape, they begin to wander randomly to find their way into the shape. For simplification we assume that agents can continuously walk in a random zig-zag path (see Fig. 8).

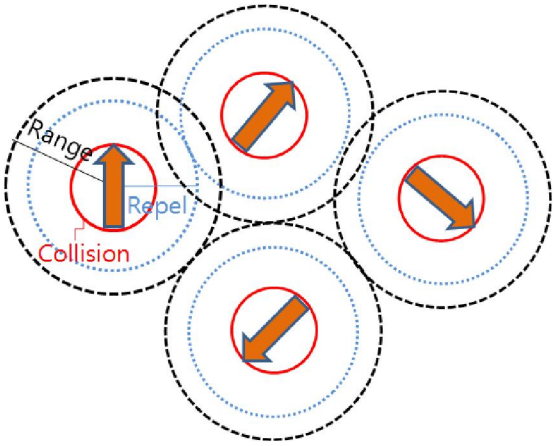


Figure 8. Agent's random walk in zig-zag path

When agents are inside the shape, they are considered as part of the swarm that comprises it. Once agents have acquired a coincident coordinate system in the shape, they should not take any steps so that place them outside of the shape. Then agents attempt to fill a formation shape. In this work, we achieve this control by modeling virtual pheromones in a closed container. Agents react to different densities of neighbors around them, moving away from areas of high density towards those of low density [34]. They finally settle into an equilibrium state of constant density level throughout the shape over time (see pseudo code in Fig. 9).

This mechanism is inspired by Reynold's flocking model and the pheromones of ants [35] and [36]. When agents die, surrounding agents quickly flood into the lower density area until equilibrium is restored.

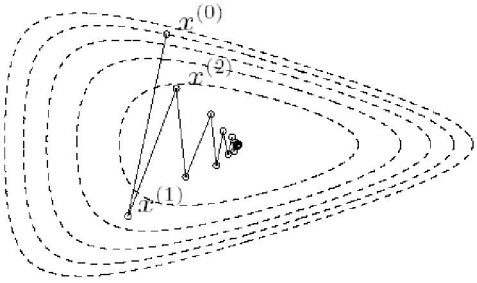


Figure 9. An example of agent's behavior for searching a target using pheromone.

However, based on our assumption, if the number of agents is not enough to compensate for the area, the agents cannot maintain the shape any more. In other words, the swarm can respond to any loss as long as there are enough agents left to maintain a certain level of density equilibrium (see Fig.10).

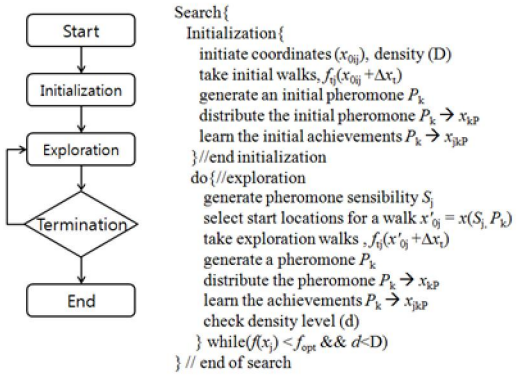


Figure 10. Pheromone robot's influence ranges

This is very a reasonable consideration when we deploy real aerial vehicles to some points because they have physical limitations in hardware like a short range of wireless link. If new agents are flooded somewhere into the swarm world, the density level is quickly increased and the agents adjust their position to maintain the density until they reach a given level of equilibrium again.

Neighboring agents inside the shape with distance < *Repel* (see Fig.8) will repulse each other, leading to an average density of agents throughout the shape. This mechanism allows the shapes to be robust against agent death or addition while spreading agents evenly throughout the shape.

This goal is accomplished by giving each agent a varying pheromone level which has a maximum value at the center of the agent and dwindles at a constant rate.

4.4. Pheromone Model for Density Control

Pheromone model is inspired by following factors: (1) biological discoveries about how cells self-organize into global patterns, and (2) distributed control systems for self-reconfigurable agent [14], [18]. Pheromone provides the common mechanism that makes it possible for agents to communicate without identifiers or addresses. The basic idea of pheromone is that a swarm is a network of agents that can dynamically communicate in the network. Agent will react to pheromone according to their local topology and state information. There is no guarantee that every agent in the network will receive the same copy of the original pheromone because a pheromone may be modified and dissipated during its propagation.

Dynamic Network of Swarm Flying Agents is specified as a network of N autonomous agents. Each agent has a set of connectors through which the agent can dynamically connect to other agents to form a kind of edges for communication or physical coupling. The connectors of agents are the channels it can be used to communicate with others. A channel of

an agent has to be connected to the other channel of another agent to communicate. Because connectors of agents can be dynamically joined and disjoined, agents can make a sort of dynamic and reconfigurable communication network.

Let $Agent_i$ and $NetEdge_i$ denote the number of agents and the number of netwoked edges, respectively. Then the dynamic network can be mathematically written as follows:

$$DN = (Agent_i, NetEdge_i) \quad (14)$$

Note that both $Agent_i$ and $NetEdge_i$ can be dynamically changed because agents can autonomously join, leave, or be failed and died.

The diffusion and dissipation of pheromone of a given agent is denoted by $P(x, y)$, where x and y are 2D space. We simply introduce the mechanism of diffusion and dissipation of pheromone as follows:

$$\frac{\partial P}{\partial t} = \left(\alpha \frac{\partial^2 C}{\partial x^2} + \beta \frac{\partial^2 P}{\partial y^2}\right) - \Delta E \quad (15)$$

The first term on the right is for diffusion, and α and β represent the rate of diffusion in x and y directions, respectively. The second term is for dissipation and the constant δ is the rate of dissipation. Eg.15 can be considered as a part of environment function which responsible for the implementation of the dynamic communication and other effects.

4.5. Density Control of Swarm

The density control is based on Payton approach [34], but is also similar in nature to the flocking rules proposed by Reynolds [14].

Our density control is to equalize overall density of agents at any situation. To this end, as shown Fig.10, the agent has three different influence ranges. Each agent has a varying repellant (or repulsive power) that has a maximum value near center which is described as Collision area and a minimum value around Range zone (see Fig.10) regarding any adjacent agent. The repellant decreases at a constant ratio from center and it becomes the smallest value when it reaches a Range zone (black dotted circle area). The agent's movement vector is weighted inversely by distance. Therefore, if any two agents are close, they push away one another. This allows agents to disperse evenly at any density.

4.6. Error Correction Method

In this work, our positioning method of agent is approximate technique and might have error for several reasons. First, distance (or proximity) sensors have inherited sensing error by themselves. Second, the gradient descent search algorithm, which we adopted to find appropriate position, may get local minima. Third, coordinates reported by neighbors may be inaccurate. Finally sensing and movement errors of robots are very common in real world.

Thus, it is suitable to take the averaged coordinates from several trilaterations instead of relying on a single trilateration [37]. In our work, if an agent moves distance d in some direction, it may actually move $d \pm \delta$. As a result, agent's perceived coordinate systems accumulate errors over time. Thus, it should be recalculated and readjusted. To this end, we accumulate previous trilaterations performed by agents in memory and average them with the recent coordinate at a certain interval of time steps (e.g., every 10 steps). We handle these issues as follows. Each agent performs trilateration and gets a coordinate at every time step. Agents keep some previous trilateration information (m) and every 10 steps, they average their recent coordinates with the last m trilaterations. (

5. Experimental Results

We show that the proposed formation control algorithm can form any arbitrary shape while autonomically compensating for various errors and maintaining the shape against agent death. The system is implemented in Java 1.5.2 based on the architecture of SWARM2.2. The user specified shape maps are represented by bitmap images. A group of 10 agents which are in 25x25 pixels are seeded to trigger the first round of trilaterations. Distance sensors have a range of 10 units (around 20m in the real world) and agents move a discrete 1 unit at every time step.

5.1. Experiments

The scenario consists of having a swarm of UAVs form shapes while maintaining a wireless connection and avoiding obstacles. This is based on a real world situation. For example, when an earthquake occurs and a lot of buildings are destroyed, it is very difficult to approach some positions. In addition, there may be a second danger like an additional building collapse. Therefore avoiding obstacles is a very important issue for gathering information in a disaster area.

In this section we show several experimental examples. Of course all experiments are inspired by [38]. They described simple heuristic algorithms for shape generation using barycenter, in which each particle senses gradients propagated by all other particles. However, as shown in their results, they had to frequently change (adding or removing something) to form arbitrary patterns. It means that

every time it requires the agents to be modified to form a specific shape.

In contrast, our algorithm can form arbitrary shapes without any human intervention or frequent modification of agents, and achieves restoration of formation from agent death or damage because each agent forms (from any starting configuration) and holds a swarm in a class of shapes. With centralized information, the distributed self-organization is possible while agents are sharing their connected coordinate system using a wireless link.

5.2. Formation of Arbitrary Shapes

Figure 9 shows several formation examples which are made by flying agents, and also shows that the same shape can be formed with different density levels that agents can accommodate. In this experiment, we basically set the initial density level of agents as 16 neighbors in target shapes. As shown in Fig.11, at any density our virtual pheromone model causes flying agents to disperse evenly throughout user-specified shapes.

As shown in Figure 9, only connected formations are possible due to using a consensus coordinate system between agents in our formation control algorithm, and shapes have a tendency to be harder to form well (i.e., organic growth).

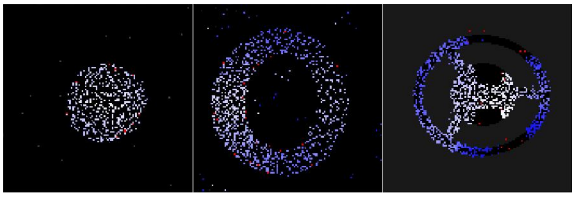


Figure 11. Examples of formations

The complete shape formation took about 1500 time steps, depending on the number of agents and the density level.

5.3. Circular Shaping

In this example, flying agents run the distributed algorithm to assume a circle shape. Several seeded agents (bright colored) will serve as the circle center. At each step, all the other agents sense their positions and they move along the direction of the circle shape. Eventually, agents outside the intended circle radius will collapse toward it.

As shown in Fig.12, it runs like a random walk (see Fig. 8). There is no thick part or high density around the boundary. The result makes a rather regular shape.

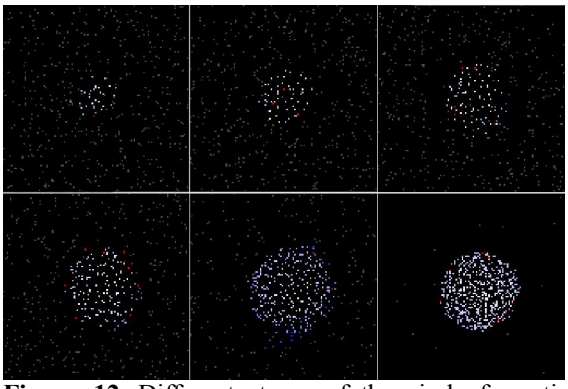


Figure 12. Different stages of the circle formation. As several agents start propagating the circle gradient as seed, other agents gradually collapse toward the circle circumference.

5.4. Formation of Ring

We consider a formation similar to a ring network architecture. In particular, we imagine a difficult terrain with large obstacles so that agents can make an emergency communication network between multiple survivors located on the ground and a rescue team (refer Fig.1). In this case, UAVs can fly over a difficult area such as flooded or collapsed terrain, or building debris and could replace damaged, nonexistent or congested networks. Our endeavor is motivated by this scenario. As shown in Fig.13, our algorithm is well adapted to making ring architectures.

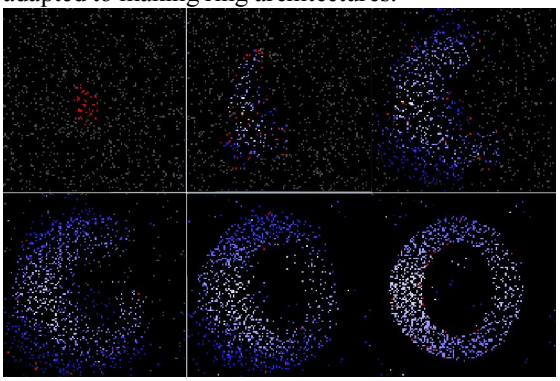


Figure 13. Ring formation. As the agents self-recognize to be there, they start making a ring shape.

5.5. Connection to Outside Swarm

Similar to the previous section, we consider a more complicate situation with a lot of obstructions. In addition, during connection several sets of agents are destroyed. The separate swarm groups should connect to each other to share information about the task area. The gray circle in Fig.14 shows the disconnection to the inner circle. Agents try to connect to the outside by the network bridge which we assume they could find. During moving, several groups died from some event and the other agents should connect to each other to avoid the debris area. As

shown in Fig. 14, two agent groups are well connected in spite of some damage. It is worth noting that we do not apply self-repairing in this case.

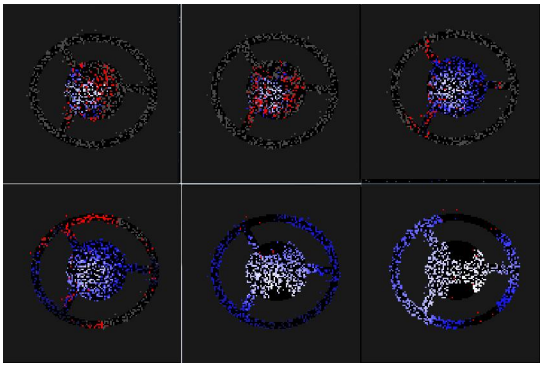


Figure 14. Connection to outside swarm group.

5.6. Self-Repairing

Whatever the shape being formed, it is of fundamental importance to preserve and maintain it. In this section, we describe experiments aimed at testing the ability to recover shape deformation from damage such as regional death of agents. We show that the connected coordinate system can be re-stabilized and that the agents can successfully adapt to death without any explicit detection or monitoring for failures.

It is a challenge to maintain the overall shape that a misinformed group of agents should stabilize into in relation to the whole aggregate. For example, suppose that the shape is correctly created and in a certain zone some agents get destroyed by an impact, opening a void space.

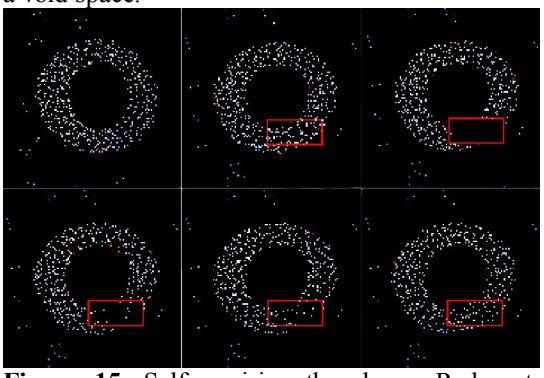


Figure 15. Self-repairing the shape. Red rectangle part is destroyed but agents self-maintain the shape without any modification.

To test this case, we first allowed agents to stabilize into the aggregate shape. Then, we selected a large region of agents and uniformly displaced their coordinate systems. On the one hand, agents are able to estimate their local density, and thus they can sense a sudden drop in their neighborhood, revealing a change. On the other hand, all the agents close to the space previously occupied by the destroyed particles now have the possibility to move.

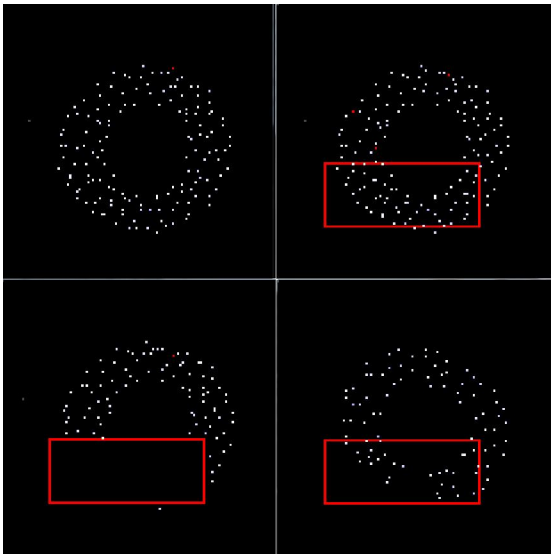


Figure 16: Unrecovered shape because of the lack of agents

Fig.15 shows experiments on the ring architecture. Some of agents in the lower right corner are destroyed and got rid of from the system. The displaced agents start to move to the corresponding region on the grid. As agents interact with their neighbors from the original grid, they consequently correct the error on the shape and the collapsed shape can be reverted into the original shape. However, as described in section 4.3, if there are not enough agents to maintain the shape, the distorted shape may not be restored from damage as shown in Fig.16.

6. Discussion

In the proposed approach, when an agent moves, it should move to another place without negatively affecting the stability of the coordinate system for adjacent agents. To demonstrate that agent movement does not negatively affect the stability, we examined the following experiment.

First, we set 1000 agents in a given 100X100 world. After 150 steps, these agents are to converge on a consistent coordinate system. Then, we assign each agent a probability to move randomly with a probability. After 220 steps, the agents are no longer allowed to move. For every 10 steps, the consistency is recorded. This experiment is repeated 5 times and Fig.19. The consistency is sum of the difference of the actual distance between the two agents and the distance between their locations in the coordinate system.

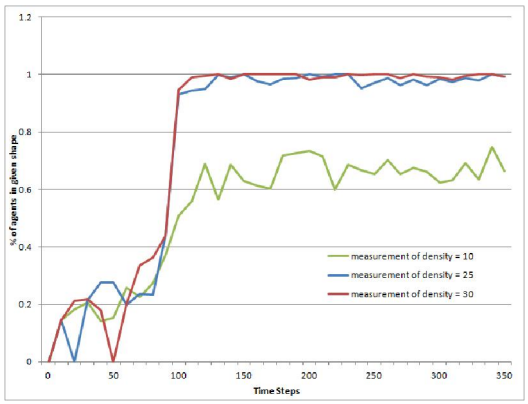


Figure 17. Percentage of agents in the shape with different measurement of density

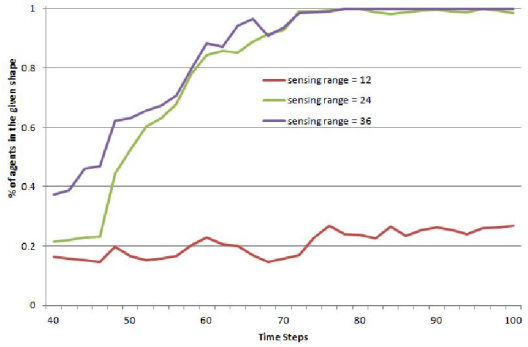


Figure 18. Percentage of agents in the shape with different angle of sensors

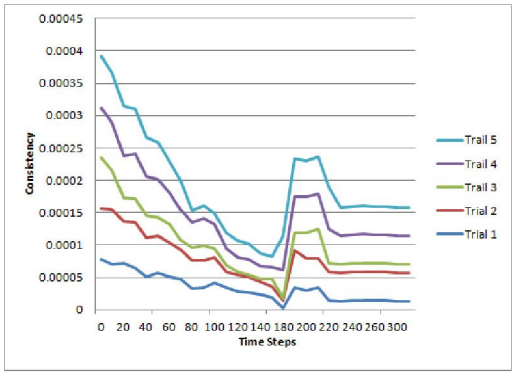


Figure 19. Consistency error for 5 simulation runs. Between time steps, 180 220, the agents were to move with probability. Immediately after Agents' movement stops, the error of consistency drastically drops back to the consistency levels of previous movement.

As shown in Fig.19, the instability of consistency jumps at time step 180. It is because when agents move, they still keep their old location information until they complete relocalization. This causes the instability during agent movement. However, after 240 steps having no more movement of agents, the consistency drops back to the level prior to movement. This indicates the movement does not negatively affect the coordinate system.

In our experiments, the average time required to complete a stabilized shape formation is about 300 time steps, depending on the number of agents and agent density. Fig.17 shows the percentage of agents in the given shape in a 150x150 world.

Most shapes are roughly formed in 100 time steps and converge after 300 time steps. In addition, rate of shape formation increases as the number of agents increases from 150. We also observe that coordinate systems very quickly propagate throughout agents when the agent density is high so that the time to stabilization is reduced.

We simply tested how the agents are affected by hardware limitations. As seen in Fig.18, the degree of angle of a sensor affects the performance of the agents. However, we did not evaluate agents' movement error or sensing error because those are related closely to making a consensus coordinate system among agents. Finally, we evaluated the variance of the coordinate system with respect to movement error and sensing error.

Fig.20 shows that the variance settles to a stable value after about 300 time steps.

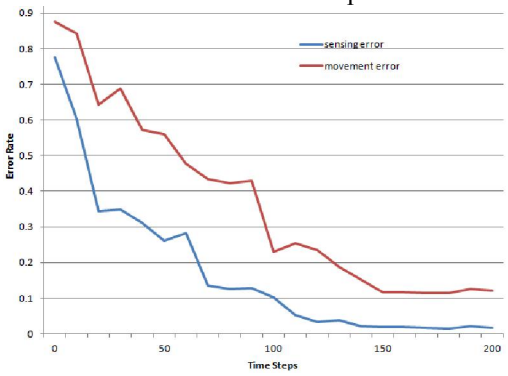


Figure 20. Average coordinate variance under movement and sensing errors

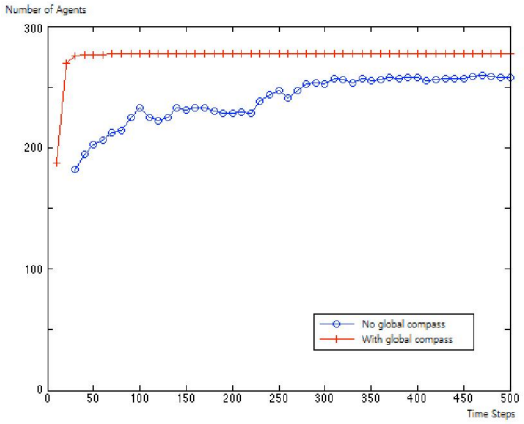


Figure 21. Average convergence comparison of global compass and without global compass

Fig.21 shows the difference between global compass and without global compass usage. As expected, the required time steps of without global compass are less than with global compass. However, it shows that when a global compass uses, it produces more stable convergence. The modified algorithm generally requests more agents to form a shape than the algorithm of [12]. As mentioned before, because our flocking model starts with random works to find other three agents which have a shared coordinate system, it shows erratic patterns until agents converge after 300 We believe, however, that reducing the cost of hardware has a benefit when the realistic UAVs are deployed to any harsh environments. In our modified algorithm

7. Conclusion and Future Works

This paper provides insight into the design of unmanned flying agent-based swarms capable of self-organizing using only local communication with inexpensive hardware.

The formation and maintenance of a swarm of UAVs for the creation of wireless communication networks in disaster areas is demonstrated in a 2D simulation with realistic scenarios. Because the development of local interaction between neighbors responsible for shape formation of a swarm is an unsolved problem, the overall inspiration is taken from the biological models of pheromones. When agents form a shape, the swarm is capable of building an efficient communication network between agents and other ground survivors.

We show that agents can self-organize into arbitrary userspecified shapes and maintain well the formed architecture by continuous trilateration-based on a consensus coordinate system and a virtual pheromone-based density model. When a set of agents is dead, destroyed, or displaced the resulting construction of swarms can also self repair it.

We also provide several quantitative evaluations to describe the effectiveness of the proposed control algorithm in terms of percentage of agents in shapes and the variance of learned coordinate systems according to agents' movement sensor errors. Future developments can focus on mitigating the effect of wind. In addition, agent's orientation control can also be investigated. Finally, scalability can be a useful approach.

Corresponding Author:
Kiwon Yeom
Human Systems Integration Division, NASA Ames Research Center,
Moffett Field, CA 94035-0001, USA

References

1. G. Prencipe, N. Santoro, Distributed algorithms for autonomous mobile robots, in: *Fourth IFIP International Conference on Theoretical Computer Science-TCS 2006*, Springer, 2006, pp. 47–62.
2. G. Prencipe, Corda: *Distributed coordination of a set of autonomous mobile robots*.
3. P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, Gathering of asynchronous robots with limited visibility, *Theoretical Computer Science,* 2005, 337 (1-3) 147–168.
4. M. Flint, M. Polycarpou, E. Fernandez-Gaucherand, Cooperative control for multiple autonomous uav's searching for targets, in: Decision and Control, 2002, *Proceedings of the 41st IEEE Conference, IEEE*, 2002, Vol. 3, pp. 2823–2828.
5. L. Merino, F. Caballero, J. Martínez-de Dios, J. Ferruz, A. Ollero, A cooperative perception system for multiple uavs: Application to automatic detection of forest fires, *Journal of Field Robotics* 2006, 23 (3), 165–184.
6. D. Pack, G. York, Developing a control architecture for multiple unmanned aerial vehicles to search and localize rf time-varying mobile targets: part i, in: Robotics and Automation, 2005. ICRA 2005. *Proceedings of the 2005 IEEE International Conference*, IEEE, 2005, pp. 3954–3959.
7. F. Adler, D. Gordon, Information collection and spread by networks of patrolling ants, *The American Naturalist*, 1992, 140 (3) 373–400.
8. S. Camazine, Self-organization in biological systems, *Princeton Univ Pr*, 2003.
9. T. Sharpe, B. Webb, Simulated and situated models of chemical trail following in ants, in: From animals to animats 5: *Proceedings of the fifth international conference on simulation of adaptive behavior*, pp. 195–204.
10. P. Basu, J. Redi, V. Shurbanov, Coordinated flocking of uavs for improved connectivity of mobile ground nodes, in: *Military Communications Conference*, 2004. MILCOM 2004. IEEE, Vol. 3, IEEE, 2004, pp. 1628–1634.
11. R. De Nardi, O. Holland, Ultraswarm: A further step towards a flock of miniature helicopters, *Swarm Robotics*, 2007, 116–128.
12. J. Cheng, W. Cheng, R. Nagpal, Robust and self-repairing formation control for swarms of mobile agents, in: *Proceedings of the National Conference on Artificial Intelligence*, Vol. 20, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005, p. 59.
13. J. Elston, E. Frew, Hierarchical distributed control for search and tracking by heterogeneous aerial robot networks, in: *Robotics and Automation*, 2008. ICRA 2008. *IEEE International Conference on*, IEEE, 2008, pp. 170–175.
14. C. Reynolds, Flocks, herds and schools: A distributed behavioral model, in: *ACM SIGGRAPH*

Computer Graphics, Vol. 21, ACM, 1987, pp. 25–34.
15. L. Barnes, M. Fields, K. Valavanis, Swarm formation control utilizing elliptical surfaces and limiting functions, Systems, Man, and Cybernetics, Part B: Cybernetics, *IEEE Transactions on*, 2009, 39 (6) 1434–1445.
16. A. Breitenmoser, M. Schwager, J. Metzger, R. Siegwart, D. Rus, Voronoi coverage of non-convex environments with a group of networked robots, in: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, 2010, pp. 4982–4989.
17. J. Cortes, S. Martinez, T. Karatas, F. Bullo, Coverage control for mobile sensing networks, Robotics and Automation, *IEEE Transactions on,* 2004, 20 (2), 243–255.
18. M. Rubenstein, W. Shen, A scalable and distributed approach for selfassembly and self-healing of a differentiated shape, in: Intelligent Robots and Systems, 2008. IROS 2008. *IEEE/RSJ International Conference on*, IEEE, 2008, pp. 1397–1402.
19. S. Hauert, L. Winkler, J. Zufferey, D. Floreano, Ant-based swarming with positionless micro air vehicles for communication relay, *Swarm Intelligence,* 2008, 2 (2) 167–188.
20. D. Becker, T. Sterling, D. Savarese, J. Dorband, U. Ranawak, C. Packer, Beowulf: A parallel workstation for scientific computation, in: Proceedings, *International Conference on Parallel Processing*, 1995Vol. 95.
21. R. Dawkins, O. Holland, A. Winfield, P. Greenway, A. Stephens, An interacting multi-robot system and smart environment for studying collective behaviours, in: Advanced Robotics, 1997. ICAR'97. *Proceedings., 8th International Conference on*, IEEE, 1997, pp. 537–542.
22. K. Yeom, Distributed formation control for communication relay with positionless flying agents., in: *FGIT-MulGraB (1), Vol. 262 of Communications in Computer and Information Science*, Springer, 2011, pp. 18–27.
23. P. Yoxall, Minuteman project, gone in a minute or here to stay-the origin, history and future of citizen activism on the united states-mexico border, the U. Miami Inter-Am. L. Rev. 37 2005, 517.
24. R. van de Burgt, H. Corporaal, *Blimp positioning in a wireless sensor network*.
25. U. of West England, The flying flock. URL *http://www.ias.uwe.ac.uk/projects.htm*
26. A. Campo, M. Dorigo, Efficient multi-foraging in swarm robotics, in: *Proceedings of the 9th European conference on Advances in artificial life*, Springer, 2007, pp. 696–705.
27. P. Gaudiano, E. Bonabeau, B. Shargel, Evolving behaviors for a swarm of unmanned air vehicles, in: Swarm Intelligence Symposium, 2005. SIS, 2005. *Proceedings 2005 IEEE*, IEEE, 2005, pp. 317–324.
28. B. Kadrovach, G. Lamont, Design and analysis of swarm-based sensor systems, in: Circuits and Systems, 2001. MWSCAS 2001. *Proceedings of the 44th IEEE 2001 Midwest Symposium on*, Vol. 1, IEEE, 2001, pp.487–490.
29. M. Kovacina, D. Palmer, G. Yang, R. Vaidyanathan, Multi-agent control algorithms for chemical cloud detection and mapping using unmanned air vehicles, in: Intelligent Robots and Systems, 2002. IEEE/RSJ *International Conference on*, Vol. 3, IEEE, 2002, pp. 2782–2788.
30. R. Siegwart, I. Nourbakhsh, Introduction to autonomous mobile robots, *The MIT Press*, 2004.
31. L. Hu, D. Evans, Localization for mobile sensor networks, in: Proceedings of the 10th annual international conference on Mobile computing and networking, ACM, 2004, pp. 45–57.
32. L. Panait, S. Luke, A pheromone-based utility model for collaborative foraging, in: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1, IEEE *Computer Society*, 2004, pp. 36–43.
33. N. Patwari, J. Ash, S. Kyperountas, A. Hero III, R. Moses, N. Correal, Locating the nodes: cooperative localization in wireless sensor networks, *Signal Processing Magazine*, IEEE 2005, 22 (4), 54–69.
34. D. Payton, M. Daily, R. Estowski, M. Howard, C. Lee, Pheromone robotics, *Autonomous Robots*, 2001, 11 (3), 319–324.
35. H. Van Dyke Parunak, S. Brueckner, J. Sauter, Digital pheromones for coordination of unmanned vehicles, *Environments for Multi-Agent Systems*, 2005, 246–263.
36. J. Sauter, R. Matthews, H. Van Dyke Parunak, S. Brueckner, Performance of digital pheromones for swarming vehicle control, in: *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, ACM, 2005, pp. 903–910.
37. R. Nagpal, H. Shrobe, J. Bachrach, Organizing a global coordinate system from local information on an ad hoc sensor network, in: *Information Processing in Sensor Networks*, Springer, 2003, pp. 553–553.
38. M. Mamei, R. Menezes, R. Tolksdorf, F. Zambonelli, Case studies for self-organization in computer science, *Journal of Systems Architecture*, 2006, 52 (8-9), 443–460.

7/11/2013