

Physicomimetics for Mobile Robot Formations

William M. Spears
Rodney Heil
Diana F. Spears
Dimitri Zarzhitsky
Computer Science Department
University of Wyoming
wspears@cs.uwyo.edu

©ACM, (2004). This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version will be published in the proceedings of AAMAS'04.

Abstract

In prior work we established how physicomimetics can be used to self-organize hexagonal and square lattice formations of mobile robots. In this paper we extend the framework to moving formations, by providing additional theoretical analysis and showing how this theory facilitates the implementation of seven robots in a hexagonal formation moving towards a goal.

1. Introduction

The focus of our research is to build aggregate sensor systems, specifically, to design rapidly deployable, scalable, adaptive, cost-effective, and robust networks (i.e., swarms, or large arrays) of autonomous distributed mobile sensing agents (e.g., robots). This combines sensing, computation and networking with mobility, thereby enabling deployment, assembly, reconfiguration, and disassembly of the multi-agent collective. Our objective is to provide a scientific, yet practical, approach to the design and analysis (behavioral assurance) of aggregate sensor systems. Our target applications for multi-agent networks include the monitoring of various air pollutants dispersed by the extraction of fossil fuels, and tracing biological and chemical hazards to their source [32].

For such applications each agent forms a grid point for performing computational fluid dynamics (CFD) calculations to follow a chemical/biological plume. Hexagonal grids have been proven to be superior to traditional rectangular grids for numerical solutions of partial differential

equations, as needed for CFD. In particular, they are more efficient and effective at handling boundary conditions [3]. For applications with few agents, seven agents create an excellent hexagonal grid for CFD computations.

Agent vehicles could vary widely in type, as well as size, e.g., from nanobots or micro-electromechanical systems (MEMS) to micro-air vehicles (MAVs) and micro-satellites. Agents are assumed to have sensors and effectors. An agent's sensors perceive the world, including other agents, and an agent's effectors make changes to that agent and/or the world, including other agents. It is assumed that agents can only sense and affect nearby agents; thus, a key challenge of this project has been how to design "local" control rules. Not only do we want the desired global behavior to emerge from the local interaction between agents (i.e., self-organization), but we also would like there to be some measure of fault-tolerance i.e., the global behavior degrades very gradually if individual agents are damaged. Self-repair is also desirable, in the event of damage. Self-organization, fault-tolerance, and self-repair are precisely those principles exhibited by natural physical systems. Thus, many answers to the problems of distributed control can be found by studying the natural laws of physics.

In prior work we have shown how this framework can be used to self-organize large numbers of mobile robots into hexagonal and square lattices. We now extend the framework to include motion of a hexagonal lattice towards a goal. First, we summarize the general *physicomimetics* framework. Then we provide a "force balance" analysis which will allow us to set the magnitude of a goal force, allowing motion towards the goal while assuring formation cohesion. Finally, details are provided regarding the implementation of our framework on a team of seven small robots with minimal sensing capabilities.

2. The Physicomimetics Framework

In our physicomimetics framework virtual physics forces drive a multi-agent system to a desired configuration or state. The desired configuration is one that minimizes overall system potential energy, and the system acts as a molecular dynamics ($\vec{F} = m\vec{a}$) simulation. We also refer to our framework as “artificial physics” or “AP”.

At an abstract level, physicomimetics treats agents as physical particles. This enables the framework to be embodied in vehicles ranging in size all the way from nanobots to satellites. Particles exist in two or three dimensions and are considered to be point-masses. Each particle i has position \vec{p} and velocity \vec{v} . We use a discrete-time approximation to the continuous behavior of the particles, with time-step Δt . At each time step, the position of each particle undergoes a perturbation $\Delta\vec{p}$. The perturbation depends on the current velocity, i.e., $\Delta\vec{p} = \vec{v}\Delta t$. The velocity of each particle at each time step also changes by $\Delta\vec{v}$. The change in velocity is controlled by the force on the particle, i.e., $\Delta\vec{v} = \vec{F}\Delta t/m$, where m is the mass of that particle and \vec{F} is the force on that particle. F and v denote the magnitude of vectors \vec{F} and \vec{v} . A frictional force is included, for self-stabilization. This force is modeled as a *viscous friction* term, i.e., the product of a viscosity coefficient and the agent’s velocity (independently modeled in the same fashion by [10]).

From the start, we wished to have our framework map easily to physical hardware, and our model reflects this design philosophy. Having a mass m associated with each particle allows our simulated robots to have momentum. Robots need not have the same mass. The frictional force allows us to model actual friction, whether it is unavoidable or deliberate, in the real robotic system. With full friction, the robots come to a complete stop between sensor readings and with no friction the robots continue to move as they sense. The time step Δt reflects the amount of time the robots need to perform their sensor readings. If Δt is small the robots get readings very often, whereas if the time step is large readings are obtained infrequently. We have also included a parameter F_{max} , which provides a necessary restriction on the acceleration a robot can achieve. Also, a parameter V_{max} restricts the velocity of the particles.

Although our framework does not require them, our design philosophy also tends to reflect further real-world constraints. The first is that our framework be as distributive as possible and the second is that we require as little information as possible. To this end, we assume that sensors are quite minimal in information content and that the sensors (passive and active) are extremely local in nature.

Due to the particle-like nature of our simulation, one important aspect of the real world is not modeled, namely, collisions of robots with other robots or objects in the envi-

ronment. This was a deliberate design decision, since we wanted our general framework to be as platform independent as possible. Once a physical platform is selected, that aspect of the simulation must be modeled separately, and we assume that some other algorithm may be responsible for collision avoidance. As we shall see, the AP general framework tends to avoid collision through strong repulsive forces, but if additional guarantees are required then they must be modeled separately.

Also, we do not model the behavioral dynamics of the actual robot. Although our robots can stop and turn on a dime, other platforms, such as MAVs, will not have this capability. We consider AP to be an algorithm that will determine “way points” for the actual physical platforms. Lower-level software may be necessary also – to actually control the movement of the robots toward their respective desired locations.

Given a set of initial conditions and some desired global behavior, we define what sensors, effectors, and force F laws are required such that the desired behavior emerges.

3. Hexagonal Lattice Sensing Grids

Let us consider an example of design. In this example, AP is applied to a swarm of agents whose mission is to form a hexagonal lattice, which acts as a distributed sensing grid [12]. Since agents are assumed to have simple sensors and primitive CPUs, our goal is to provide the simplest possible control rules that require minimal sensors and effectors. At first blush, creating hexagons would appear to be somewhat complicated, requiring sensors that can calculate range, the number of neighbors, their angles, etc. However, it turns out that only range and bearing information are required. To understand this, recall an old high-school geometry lesson in which six circles of radius R can be drawn on the perimeter of a central circle of radius R (the fact that this can be done with only a compass and straight-edge can be proven with Galois theory). Figure 1 illustrates this construction. If the particles (shown as small circular spots) are deposited at the intersections of the circles, they form a hexagon with a particle in the middle.

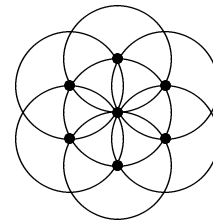


Figure 1. How circles can create hexagons.

The construction indicates that hexagons can be created via overlapping circles of radius R . To map this into a force law, imagine that each particle repels other particles that are closer than R , while attracting particles that are further than R in distance. Thus each particle can be considered to have a circular “potential well” around itself at radius R – neighboring particles will want to be at distance R from each other. The intersection of these potential wells is a form of constructive interference that creates “nodes” of very low potential energy where the particles will be likely to reside (again these are the small circular spots in the figure). Thus the particles serve to create the very potential energy surface they are responding to! Potential energy (PE) is never actually computed by agents. Agents only compute local force vectors for their current location. PE is computed for visualization/analysis purposes only.

With this in mind, we define a force law $F = Gm_i m_j / r^p$, where F is the magnitude of the force between two particles i and j , r is the range between the two particles, and p is some power (by default $m_i = 1.0$ for all particles). The “gravitational constant” G is set at initialization. The force is repulsive if $r < R$ and attractive if $r > R$. Each particle has a sensor that detects the range and bearing to nearby particles. The only effector is to be able to move with velocity \vec{v} . To ensure that the force laws are local in nature, particles have a visual range of only $1.5R$. Also, due to the discrete-time nature of the model, it is important to define a maximum force F_{max} that can be obtained.

Figure 2 shows how an initial universe of $N = 200$ particles that began in a single small, random cluster has evolved over 1000 time steps into a hexagonal lattice, using this very simple force law. The hexagonal lattice is not perfect – there is a flaw near the center of the structure. Also, the perimeter is not a hexagon, although this is not surprising, given the lack of global constraints. However, many hexagons are clearly embedded in the structure and the overall structure is quite hexagonal.

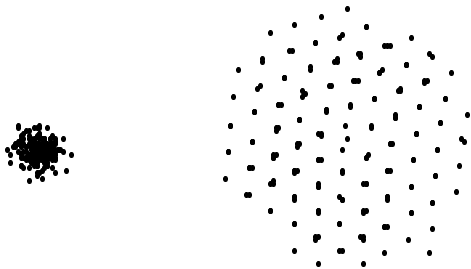


Figure 2. The evolution of the particles from $t = 0$ to 1000.

Note that in Figure 2 we observe a clustering effect, i.e., each node in the lattice may contain multiple particles. Clustering was an emergent property that we had not expected, and it provides increased robust behavior, because the disappearance (failure) of individual particles (agents) from a cluster will have minimal effect. The pattern of particles shown in Figure 2 is quite stable, and does not change to any significant degree as t increases past 1000.

Clustering results when the repulsion of two or more particles at a particular lattice node is overcome by neighboring particles that force the cluster together. Hence clustering is unlikely at the perimeter of the formation, while very likely at interior nodes. Clustering is a result of the setting of G . Simply put, high G produces deep potential wells that can contain multiple particles. As G is reduced, those wells reduce in depth. Finally, at a certain point, the well vanishes and only one particle can occupy that lattice node position. A phase transition occurs and the lattices contain no clusters. If we denote G_t as the value of G where the phase transition occurs, it can be shown that, for hexagonal lattices:

$$G_t^{\Delta} = \frac{F_{max} R^p}{2\sqrt{3}} \quad (1)$$

As shown in [27, 28], it is simple to extend the physi-comimetics framework from hexagonal lattices to square lattices, by the addition of a one bit attribute to each robot. In this case the phase transition occurs at:

$$G_t^{\square} = \frac{F_{max} R^p}{2\sqrt{2} + 2} \quad (2)$$

Note that the only difference between the two equations is the denominator, which reflects the difference in geometry between hexagonal and square lattices [9]. To summarize, if $G < G_t$, formations occur without clustering. If $G > G_t$, formations occur with clustering. These equations are quite accurate (as confirmed via simulation) for arbitrary N . However, in this paper we will focus on a hexagonal formation of seven robots (with one in the center), and refine the theory for that specific situation.

4. Refined Phase Transition Theory

If we have seven robots, then it is possible for there to be a small cluster of two robots in the center, with five robots (instead of six) along the perimeter. This situation is depicted in Figure 3 (having more than two particles at the central node is almost impossible and we ignore that situation in this paper). The open circle represents the unoccupied node in the formation.

Let us focus on one of the two particles in the center, and call this particle “A.” Intuition would argue that the most

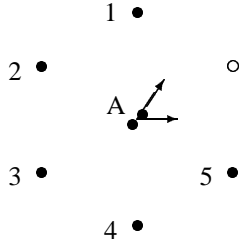


Figure 3. How particles escape.

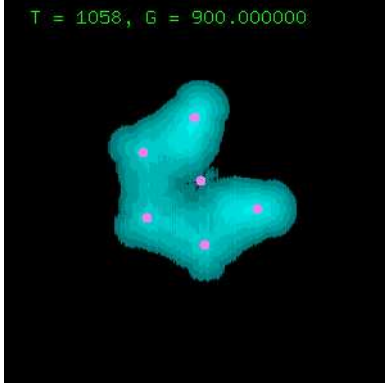


Figure 4. Visualization of potential field, showing escape paths.

likely escape path for 'A' would be directly towards the unoccupied node. Unfortunately, this intuition is incorrect, due to the geometry of the situation. In fact the arrows depict the two most likely escape paths (as the particle escapes further from the center, it eventually curves back towards the unoccupied node, but we need not concern ourselves with that for this analysis). We can confirm this by visualization of the potential field for this situation. By definition the potential field is computed using the path integral $-\int_s \vec{F} \cdot d\vec{s}$, where $\vec{s} = x\vec{i} + y\vec{j}$ is the path. A path integral may be used to calculate the potential field if the force is (or is approximately) conservative, which is true for our framework.

Figure 4 illustrates the PE field. Lighter shading represents high positive potential energy, while black represents low (zero or negative) potential energy. Positive PE indicates that work is required to push a virtual particle to that position. Negative PE indicates that work is required to push a particle *away* from that position. A virtual particle placed in this field moves from regions of high potential energy to low potential energy. Note that a virtual particle that is close to the center will not want to move directly towards the unoccupied node (and one can see a small bulge of positive PE along that direction). Instead, it would follow one of the two directions depicted in Figure 3.

Due to symmetry, we can focus on either of the escape paths for 'A'. Let us arbitrarily focus on the escape path along the horizontal axis. Particle 'A' can be expelled from its cluster along this axis by the other central particle, which exerts a repulsive force of F_{max} , because the range between particles, r , is very small. Therefore, the fragmentation force upon 'A' is equal to F_{max} .

Next, we derive an expression for the cohesion force on 'A'. Particle 'A' is held near the center by the outer perimeter particles. As above, without loss of generality we focus on the horizontal axis as the escape path of 'A'. Consider the force exerted by the outer perimeter particle '2' on 'A'. Because nodes are R apart, the magnitude of this force is G/R^p . The projection of this force on the horizontal axis escape path is $\sqrt{3}/2$ times the magnitude of this force – because the angle between the chosen outer perimeter particle and the horizontal axis is 30 degrees. Since there are three outer perimeter particles ('2', '3', and '5') exerting this force (the remaining two have a force of 0 after projection), we multiply this amount by three to get a total cohesion force of $3\sqrt{3}G/2R^p$.

When the cohesion force is greater than the fragmentation force, the central cluster will remain intact. When the fragmentation force is greater, the central cluster will separate. Thus, our law states that the phase transition will occur roughly when the two forces are in balance: $F_{max} = 3\sqrt{3}G/2R^p$. We can now state that the phase transition will occur when $G = 2F_{max}R^p/3\sqrt{3}$. Hence, for our specific seven particle system, the phase transition occurs at:

$$G_t^{\Delta} = \frac{4G_t^{\Delta}}{3} \quad (3)$$

This refinement is intuitively understandable. If instead of the situation in Figure 3 we had an eight particle system, with two in the center and six in the perimeter (i.e., the unoccupied node is now occupied), four outer particles would attempt to hold the cluster together (along some chosen escape path). A derivation of the phase transition point would yield exactly G_t^{Δ} . However, in Figure 3 one of those four outer particles is missing, producing a 4/3 ratio. The net effect is that we can raise G to $1.33G_t^{\Delta}$ (hence increasing the speed of self-organization and the strength of the formation), while still avoiding clustering. This will be verified in our robotic experiments.

5. Movement of the Hexagonal Formation

Let us assume that each robot attempts to sense the goal. However, we can not assume that such sensing will always be accurate. Hence, on occasion robots may attempt to move in different directions towards their incorrectly sensed goal. Furthermore, if one or more robots are temporarily

halted in their movement (due to environmental or hardware problems), we would like the formation to maintain its cohesion, at least until a decision is made that the cohesion is no longer desirable.

Strength of cohesion is in opposition to the strength of the goal force acting on the robots. Stronger goal forces will tend to pull the formation apart. In this section we derive an upper bound on the goal force, which we will use in our empirical experiments.

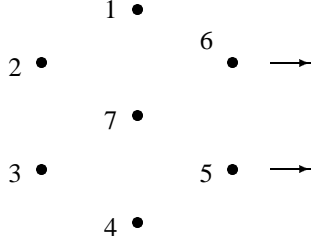


Figure 5. The formation is moving.

Consider Figure 5, which shows a formation of seven robots moving towards a goal that is to the right. Let us assume that all robots other than the rightmost two have temporarily halted. We desire both of the rightmost robots ('5' and '6') to maintain position. Hence, the force of cohesion holding them into the formation must exceed the goal force F_{goal} .

We can use a "force balance" analysis similar to that shown in the previous section. Consider either of the two rightmost particles. Particle '6' is held back by '1' and '7', while '5' is held back by '4' and '7'. Particle '1' holds '6' in position with force magnitude $\sqrt{3}G/2R^p$ (after projection). Similarly, particle '7' holds '6' in position with force magnitude $\sqrt{3}G/2R^p$. Hence the total force holding '6' in place (in opposition to the goal force) has magnitude $\sqrt{3}G/R^p$. The analysis is identical for particle '5'. Also, the identical analysis holds if all robots other than '2' and '3' have halted and we wish to prevent the formation from collapse. In all cases we require the goal force to be less than the force of cohesion:

$$F_{goal} < \frac{\sqrt{3}G}{R^p} \quad (4)$$

If $G = G_t^{\Delta}$ this can be simplified to:

$$F_{goal} < \frac{2F_{max}}{3} \quad (5)$$

Although there are other situations which we could consider (such as the formation moving upwards or at some other angle), the situation we have just analyzed is the most stringent. In this situation both of the two rightmost particles are kept from moving forwards only by two other parti-

cles (which are in the center "column" of the formation). In other situations more particles participate in the cohesion. In essence we have analyzed the weakest link in the chain of force bonds and if cohesion can be guaranteed, then it will be guaranteed for all other reasonable situations. One exception is if a robot is "dangling" and is connected to the formation via only one force bond. This situation has never occurred in our experiments; however, if we needed to deal with it we merely have to stipulate that F_{goal} must be less than G/R^p , a more strict constraint than before.

6. Application to a Team of Mobile Robots

The current focus of this project is the physical embodiment of the physicomimetics framework on a team of robots. Our choice of robots and sensors clearly expresses a preference for minimal expense and expendable platforms. For our initial experiments with robots we have used inexpensive Lego kits from the KISS Institute for Practical Robotics (we thank Alan Schultz of NRL for the loan of five of these kits – two of them were used in the following experiments). These kits come with a variety of useful sensors and effectors, and two processors, the RCX Lego processor and the Handy Board. Due to its generality and ease of programming (it is programmed using Interactive C), we are currently using the Handy Board. The Handy Board has a HC11 Motorola processor with 32K of static RAM, a two line LCD screen, and the capacity to drive several DC motors and servos. It also has ports for a variety of digital and analog sensors.

Our robotic platform has two independent motors (drive trains) and two casters, allowing the platform to turn on a dime and move forward and backward. Slot sensors are incorporated into the drive trains to function as shaft encoders, giving us reasonably precise measures of the angle turned by the robot and the distance moved. The transmissions are geared down 25:1 to help minimize slippage with the floor surface.

The "head" of the robot is a sensor platform used for the detection of other robots in the vicinity. For range information we use Sharp GP2D12 IR sensors. This sensor provides fairly accurate readings based on the distance of the sensed object (10% error over a range of 6 to 50 inches). The readings are relatively non-influenced by the material sensed, unless the material is highly reflective. However, the angle of orientation of the object does have significant effects, especially as the object became more reflective. As a consequence, the "head" is a circular cardboard (non-reflective) cylinder, allowing for accurate readings by the IR sensors.

The head is mounted horizontally on a servo motor. With a 180 degrees of motion of the servo, and two Sharp sensors mounted opposite each other, the head provides a simple "vision" system with a 360 degree view. Once a full 360

```

#define GOAL_FORCE 1.0

// Compute the force exerted by the goal.
// Normalize to have magnitude GOAL_FORCE.
void goalforce (float *x, float *y) {
    float mag;
    *x = *x + (float)(256 - analog(2))/256.0;
    *x = *x - (float)(256 - analog(3))/256.0;
    *y = *y + (float)(256 - analog(4))/256.0;
    *y = *y - (float)(256 - analog(5))/256.0;
    mag = sqrt(*x * *x + *y * *y);
    *x = GOAL_FORCE * *x / mag;
    *y = GOAL_FORCE * *y / mag;
}

// The main AP code.
void ap() {
    int theta, index = 0;
    float r, F, fx, fy;
    float sum_fx = 0.0, sum_fy = 0.0;
    float vx = 0.0, vy = 0.0; // Full friction
    float delta_vx, delta_vy, delta_x, delta_y;

    // Get force from goal
    goalforce(&sum_fx,&sum_fy);

    // Add in force by neighbors
    while ((robots[index][0] != -1)) {
        theta = robots[index][0];
        r = robots[index][1];
        if (r > 1.5 * R) F = 0.0;
        else { // The force law
            F = G / (r * r);
            if (F > F_MAX) F = F_MAX;
            if (r < R) F = -F;
        } // Break into x and y components
        fx = F * cos(theta);
        fy = F * sin(theta);
        sum_fx += fx;
        sum_fy += fy;
        index++;
    }

    // Compute new velocity
    delta_vx = delta_T*sum_fx;
    delta_vy = delta_T*sum_fy;
    vx = vx + delta_vx;
    vy = vy + delta_vy;
    delta_x = delta_T*vx;
    delta_y = delta_T*vy;

    // Compute distance/bearing to move
    distance = (int)(sqrt(delta_x*delta_x +
                          delta_y*delta_y));
    turn = (int)(atan2(delta_y, delta_x));
}

```

Figure 6. The main AP code.

degree scan is done, object detection is performed. We use a simple first derivative filter that detects object boundaries, even under conditions of partial occlusion. Simple width filters are used to ignore objects that are too narrow (chair legs) and too wide (walls). The resulting algorithm does a good job of detecting nearby robots, producing a “robot” list which gives the bearing and range to the nearest robot.

Once sensing and object detection is complete, the AP algorithm computes the virtual force felt by that robot. In response, the robot turns and moves to some position. This “cycle” of sensing, computation and motion continues until we shut down the robots or they lose power. Figure 6 shows the AP code. It takes a robot neighbor list as input, and outputs the vector of motion (in terms of a turn and a distance to move).

For our experiment we built seven robots. The objective was to form a stable hexagon that moves towards a light source. Each robot ran the same piece of software. The desired distance R between robots was 20 inches. Using our theory, $G_t^\Delta = 231$ ($p = 2$ and $F_{max} = 2$), but as our refined theory suggests, we should be able to use the higher value $G_t^{7\Delta}$ of 308 without obtaining clustering (as stated earlier this increases the speed of self-organization and the strength of the formation).

Each robot has a local coordinate system, with the “front” of the robot representing the positive x axis. We placed four photo-diode light sensors on each robot, one per side. The front, back, left, and right sensors are connected to Handy Board analog ports 2, 3, 4, and 5. Each light sensor produces values from 0 to 255, where 0 represents the brightest light. Each sensor value is normalized to the range (0,1], (where a 1 represents the brightest light) and their values are combined to produce the x and y components of the goal force. Finally, this force is normalized once again to have magnitude $F_{goal} = GOAL_FORCE$ (see Figure 6). According to theory, with $G = 308$, $R = 20$, $F_{max} = 2$, and $p = 2$, $GOAL_FORCE$ must be less than 1.33. For our experiment $GOAL_FORCE$ is conservatively set to 1.0.

The “goalforce” function produces a force vector that moves the robots towards a light source (a window). Note that the *reflection* of the window on the floor is not noticed by the robots and is not the light source. The “ap” function first computes the goal force and then adds in the force components from neighboring robots that create and maintain the formation. The results are shown in Figure 7, and were consistent over ten runs, maintaining formation and never showing clustering.

7. Summary and Related Work

This paper has presented “force balance” quantitative analyses of our physicomimetics framework. The first anal-

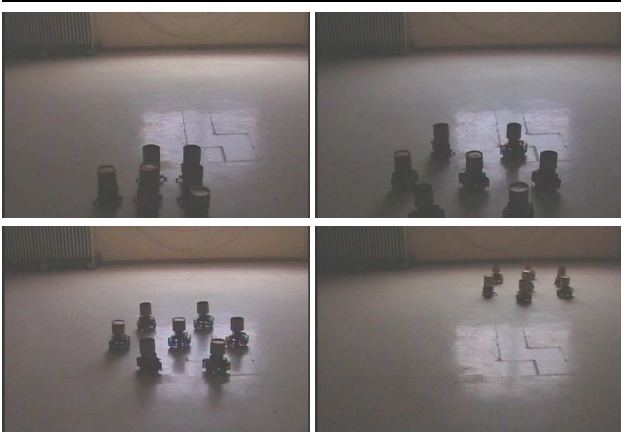


Figure 7. Seven robots form a hexagon, and move towards a light source.

ysis was a refinement of a prior general “phase transition” analysis specifically for seven robots, which allows us to control the presence of “clustering” in hexagonal formations. The second analysis allows us to set upper bounds on a goal force that is felt by the formation, allowing movement towards the goal while preserving cohesion of the formation. It is important to note that this second analysis holds in the more general situation of N robots.

Related work includes the potential field (PF) literature (e.g., [13]). Generally, this deals with a small number of robots (typically just one) that need to navigate through a field of obstacles to get to a target location. Recently, [10] and [31] have extended the PF approach to include inter-agent repulsive forces. Although this work was developed independently of AP, it affirms the feasibility of a physics-force-based approach.

Behavior-based (e.g., [17, 2]) and rule-based (e.g., [25]) approaches are alternatives to a physics-based approach to swarm behavior. However, these alternatives are heuristic, and therefore in general they are more difficult to analyze. Control-theoretic approaches have also been applied effectively (e.g., [1, 4]). Nevertheless, predictive analyses of any of these approaches are scarce. Multi-robot swarms with emergent behavior are notoriously difficult to predict, and few researchers have tackled such an endeavor. Our advantage in this respect is that AP is physics-based, and thus we are able to employ traditional physics analysis techniques.

The work that is most related consists of other theoretical analyses of swarm systems. Our comparisons are in terms of the goal and method of analysis. There are generally two goals: stability and convergence/correctness. Under stability is the work by [24, 6, 16, 15, 20]. The first three apply Lyapunov methods. Liu et al. [16] use a geometric/topological approach, and Lerman [15] uses

differential equations to model system dynamics. Convergence/correctness work includes [29, 21, 16]. Geometry, topology and graph theory techniques are applied. Other goals of theoretical analyses include time complexity [19], synthesis [23], prediction of movement cohesion [16], coalition size [15], number of instigators to switch strategies [18], and collision frequency [11].

Methods of analysis are also diverse. Here we focus only on physics-based analyses of physics-based swarm robotics systems. We know of four methods. The first are the Lyapunov analyses by [24, 6, 20]. The second is the kinetic gas theory by [11]. The third is the minimum energy analysis by [23]. The fourth develops macro-level equations describing flocking as a fluid-like movement [30].

To the best of our knowledge, the only analyses mentioned above that can be used to set system parameters are those of [15, 18, 30]. The first two analyses are of behavior-based systems, while the latter is of a “velocity matching” particle system.

8. Future Work

We are currently working on improving our mechanism for robot localization. This work is an extension of the work by Navarro-Serment et. al. [14], using a combination of RF with acoustic pulses to perform trilateration. This extension will allow us to distinguish robots from obstacles in a straightforward fashion, and will be much faster than our current “scan” technique.

In addition, we are adding the capability to deal with obstacles. We are pursuing multiple approaches, including a “force balance” theoretical approach similar to that used in this paper for small obstacles (those that can be seen in their entirety), a kinetic theory approach (for very large obstacles), and an evolutionary algorithm approach for learning the appropriate force laws governing robot-robot, robot-goal, and robot-obstacle interactions.

It is important to point out that we consider AP to be one level of a more complex control architecture. The lowest level controls the actual movement of the platforms. AP is at the next higher level, providing “way points” for the robots to move toward, as well as providing simple repair mechanisms. Our goal is to put as much behavior as possible into this level, in order to provide behavioral assurances and the ability to generate laws governing important parameters. However, clearly the current AP paradigm will not solve more complex tasks, involving planning, learning, repair from more catastrophic events, and global information. For example, certain arrangements of obstacles (such as cul-de-sacs) will require the addition of memory and planning. Hence, even higher levels will be required [26, 8]. Learning is especially interesting to us, and we would like to add it to

AP. Learning has already been demonstrated to be advantageous in the context of behavior-based [5, 7] and rule-based [25, 22] systems, but its value has not yet been explored in the context of a physics-based system.

References

- [1] R. Alur, J. Esposito, M. Kim, J. Kumar, and I. Lee. Formal modeling and analysis of hybrid systems: A case study in multi-robot coordination. In *Lecture Notes in Computer Science 1708*, pages 212–232. Springer-Verlag, 1999.
- [2] T. Balch and R. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Trans. on Robotics and Autom.*, 14(6):1–15, 1998.
- [3] E. Carlson, H. Sun, D. Smith, and J. Zhang. Second order accuracy of the 4-point hexagonal net grid finite difference scheme for solving the 2D Helmholtz equation. Technical Report 378-03, CS Dept, University of Kentucky, 2003.
- [4] J. Fax and R. Murray. Information flow and cooperative control of vehicle formations. In *IFAC World Congress*, 2002.
- [5] F. Fernandez and L. Parker. Learning in large cooperative multi-robot domains. *International Journal of Robotics and Automation*, 16(4):217–226, 2002.
- [6] R. Fierro, P. Song, A. Das, and V. Kumar. Cooperative control of robot formations. In R. Murphey and P. Pardalos, editors, *Cooperative Control and Optimization*, volume 66, pages 73–93. Kluwer Academic Press, 2002.
- [7] D. Goldberg and M. Matarić. Learning multiple models for reward maximization. In *Seventeenth International Conference on Machine Learning*, 2000.
- [8] D. Gordon, W. Spears, O. Sokolsky, and I. Lee. Distributed spatial control, global monitoring and steering of mobile physical agents. In *IEEE International Conference on Information, Intelligence, and Systems*, 1999.
- [9] D. Gordon-Spears and W. Spears. Analysis of a phase transition in a physics-based multiagent system. In *Formal Methods in Agent-Based Systems*, 2002.
- [10] A. Howard, M. Matarić, and G. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Sixth Int'l Symposium on Distributed Autonomous Robotics Systems*, 2002.
- [11] S. Jantz, K. Doty, J. Bagnell, and I. Zapata. Kinetics of robotics: The development of universal metrics in robotic swarms. In *Florida Conference on Recent Advances in Robotics*, 1997.
- [12] J. Kellogg, C. Bovais, R. Foch, H. McFarlane, C. Sullivan, J. Dahlburg, J. Gardner, R. Ramamurti, D. Gordon-Spears, R. Hartley, B. Kamgar-Parsi, F. Pipitone, W. Spears, A. Sciambi, and D. Srull. The NRL micro tactical expendable (MITE) air vehicle. *The Aeronautical Journal*, 106(1062):431–441, 2002.
- [13] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int'l Journal of Robotics Research*, 5(1):90–98, 1986.
- [14] L. L. Navarro-Serment, C. Paredis, and P. Khosla. A beacon system for the localization of distributed robotic teams. In *International Conference on Field and Service Robots*, 1999.
- [15] K. Lerman and A. Galstyan. A general methodology for mathematical analysis of multi-agent systems. Technical Report ISI-TR-529, USC Information Sciences, 2001.
- [16] Y. Liu, K. Passino, and M. Polycarpou. Stability analysis of m-dimensional asynchronous swarms with a fixed communication topology. In *IEEE Transactions on Automatic Control*, volume 48, pages 76–95, 2003.
- [17] M. Matarić. Designing and understanding adaptive group behavior. Technical report, CS Dept, Brandeis Univ., 1995.
- [18] C. Numaoka. Phase transitions in instigated collective decision making. *Adaptive Behavior*, 3(2):185–222, 1995.
- [19] S. K. O. Shehory and O. Yadgar. Emergent cooperative goal-satisfaction in large-scale automated-agent systems. *Artificial Intelligence*, 110:1–55, 1999.
- [20] R. Olfati-Saber and R. Murray. Distributed cooperative control of multiple vehicle formations using structural potential functions. In *IFAC World Congress*, 2002.
- [21] L. Parker. Alliance: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2), 1998.
- [22] M. Potter, L. Meeden, and A. Schultz. Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *Seventh International Conference on Artificial Intelligence*. Morgan Kaufmann, 2001.
- [23] J. Reif and H. Wang. Social potential fields: A distributed behavioral control for autonomous robots. In *Workshop on the Algorithmic Foundations of Robotics*, 1998.
- [24] D. Schoenwald, J. Feddema, and F. Opper. Decentralized control of a collective of autonomous robotic vehicles. In *American Control Conference*, pages 2087–2092, 2001.
- [25] A. Schultz, J. Grefenstette, and W. Adams. Roboshepherd: Learning a complex behavior. In *The Robotics and Learning Workshop at FLAIRS*, 1996.
- [26] R. Simmons, T. Smith, M. Dias, D. Goldberg, D. Hershberger, A. Stentz, and R. Zlot. A layered architecture for coordination of mobile robots. In A. Schultz and L. Parker, editors, *Multi-Agent Robot Systems: From Swarms to Intelligent Automata*. Kluwer, 2002.
- [27] W. Spears and D. Gordon. Using artificial physics to control agents. In *IEEE International Conference on Information, Intelligence, and Systems*, pages 281–288, 1999.
- [28] W. Spears, D. Spears, J. Hamann, and R. Heil. Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17(2-3), 2004.
- [29] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal of Computation*, 28(4):1347–1363, 1999.
- [30] J. Toner and Y. Tu. Flocks, herds, and schools: A quantitative theory of flocking. *Physical Review E*, 58(4):4828–4858, 1998.
- [31] D. Vail and M. Veloso. Multi-robot dynamic role assignment and coordination through shared potential fields. In *Multi-Robot Systems*. Kluwer, 2003.

- [32] D. Zarzhitsky, D. Spears, D. Thayer, and W. Spears.
Agent-based chemical plume tracing using fluid dynamics.
FAABS'04 workshop, 2004.