

Physics-Based Robot Swarms For Coverage Problems

Diana SPEARS, Wesley KERR, and William SPEARS

Abstract—One of the biggest issues preventing the acceptability of large, multi-robot systems (i.e., swarms) is the predictability of aggregate behavior. This is an especially thorny issue due to the current multi-robot philosophy of designing swarm behaviors that emerge spontaneously from local robot interactions, without the invocation of any global engineering principles. Although this paper agrees with the notion of emergent swarm behavior, it adopts a more rigorous approach to multi-robot system design that is based on physics principles. By using physics for multi-robot design, traditional physics analysis techniques are easily applied to predict swarm behavior. This paper demonstrates that by using a physics-based swarm approach, one can design systems that are both effective on coverage tasks and predictable in the aggregate.

Index Terms—swarms, predictable behavior, physics-based, coverage tasks.

I. INTRODUCTION

This paper addresses an unusually challenging coverage task that requires surveillance of a region about which there is no prior knowledge and for which movement is needed to achieve maximum coverage (because there are too few robots, with local sensing, for good static coverage). Our solution involves a swarm of simulated robots that successfully achieve this task. Furthermore, characteristics of aggregate swarm behavior are demonstrated to be predictable using a theoretical framework.

The paper begins with a description of our coverage task, along with related prior research. Then, we present a novel approach to solving this problem, founded on physics-based principles of modeling gaseous fluids. Next, physics-based theoretical laws are derived, which make stochastic predictions about aggregate swarm behavior. These physics laws are tested in simplified settings, and shown to be predictive of simulated swarm behavior. Finally, rigorous experimental comparisons between our approach and state-of-the-art alternatives are run on the full simulated task. Our physics-based approach performs competitively, while making fewer assumptions than the alternatives.

II. THE COVERAGE TASK

The coverage task being addressed is to sweep a large group of mobile robots through a long bounded region. This could be a swath of land, a corridor in a building, a city sector, or an

underground passageway/tunnel. The goal of the robots is to perform a search, thereby providing maximum coverage. This search might be for enemy mines (de-mining), survivors of a collapsed building or, alternatively, the robots might act as sentries by patrolling the area. All of these different examples require the robots to search the entire region to guarantee coverage.

For practical reasons, the robots in the swarm are assumed to be simple and inexpensive, and they have a limited sensing range for detecting other robots or objects. The robots have no GPS or global knowledge, other than that they can sense a global direction to move, e.g., with light sensors. Robots need to avoid obstacles of any size, possibly the size of large buildings. This poses a challenge because with limited sensors, robots on one side of large obstacles cannot visually/explicitly communicate with robots on the other side.

It is assumed that the robots need to keep moving because there are not enough of them to simultaneously view the entire length of the region. All robots begin at the corridor entrance, and move to the opposite end of the corridor (considered the “goal direction”). This constitutes one *sweep*. A sweep terminates when all robots have reached the goal end of the corridor, or a time limit is reached. The primary objective of the robots is to maximize the coverage in one sweep; a secondary objective is to minimize the sweep time.

Due to our two objectives, we are concerned with two primary forms of coverage: *spatial coverage* and *temporal coverage*. Two forms of spatial coverage are considered here: *longitudinal* (in the goal direction) and *lateral* (orthogonal to the goal direction). Longitudinal coverage can be achieved by moving the swarm in the goal direction as a whole, and lateral coverage is achieved by a uniform distribution of the robots between the side walls of the corridor. In particular, as robots move toward the goal they increase longitudinal coverage, and as they move or disperse orthogonal to the goal they increase lateral coverage.

Temporal coverage is determined by the time spent performing a sweep of the corridor. Longer sweep time implies worse temporal coverage. Greater temporal coverage can be achieved by increasing the average robot speed, for example.

Note that it is difficult to optimize both temporal and spatial coverage simultaneously. Optimum spatial coverage takes time, whereas quick sweeps will necessarily miss areas. Poor spatial or temporal coverage is potentially problematic because in this case it is easier for an intruder to remain undetected.

Part of this paper has been presented in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Edmonton, August, 2005.

D. Spears and W. Spears are with the Department of Computer Science, University of Wyoming, Laramie, WY 82071. They can be reached via email at: {*dspears,wspears*}@*cs.uwyo.edu*. W. Kerr is with the Department of Computer Science, University of Southern California, Los Angeles, CA 90089. He can be reached via email at: *wkerr@isi.edu*.

III. PRIOR RESEARCH ON COVERAGE TASKS

Although swarms of insects and flocks of birds are adequate at solving our coverage task, we prefer a *physicomimetic* over a biomimetic approach. Our rationale for this decision is that although biomimetics is effective for swarm design, it produces systems that are difficult to analyze [17]. Furthermore, many of the biomimetic approaches to coverage rely on the emission and subsequent detection of environmental markers for explicit communication. For example, one biomimetic method (with which we compare in the experiments section, below) is the Ant robot approach of Koenig and Liu [15]. This approach is theoretically guaranteed to yield complete regional coverage. Nevertheless, to achieve such coverage with the Ant approach, information must be communicated by leaving traces in the environment. This is a practical concern for three reasons: it requires additional sensors and effectors for detecting and emitting traces, it eliminates stealth, and it allows for deception by an adversary.

Other swarm-based approaches investigate the usefulness of formations while navigating obstacle courses [3], [10]. Unfortunately, geometric robot formations are poorly suited for maximum coverage tasks. The work by Bayazit et al. [5] attempts to combine both behavior-based models and global roadmaps for addressing coverage tasks. This work successfully navigates the world, but assumes that a lot of global information is known *a priori*.

Ad-hoc sensor networks are also often designed to achieve complete coverage. Batalin and Sukhatme [4], for example, created control algorithms to maximize coverage by increasing robot-robot distances. Each robot moves away from the other robots until coverage is maximized. Megerian et al. [18] provide different approaches guaranteeing coverage by using theoretical graph proofs. Whereas these prior approaches are successful at maximizing sensor coverage, they implicitly assume that the number of robots (with a given, fixed sensor range) is sufficient to cover the area statically, which may be an impractical constraint. Also, many of these approaches require a preliminary geolocation step requiring GPS, or some type of advance knowledge of the environment, e.g., [18].

Another approach examines different decompositions of the corridor to cover [20]. In these decompositions, space is separated into cells that can be covered by sweeping back and forth. By creating simple cell decompositions, these approaches guarantee complete coverage. To accomplish this, the robots must maintain an internal representation of the world in order to determine different decomposition points. This is potentially problematic for robotic swarms, because internal representations can require a great deal of memory, coupled with localization.

IV. PHYSICS-BASED SWARM SYSTEM DESIGN

As the previous section shows, prior research is limited in its ability to fully address maximum coverage problems because it tends to have overly restrictive assumptions. Our approach overcomes these restrictions; it does *not* require environmental traces, any prior information about the environment to be surveilled, multi-robot formations, global information (other

than the ability to detect the longitudinal direction of the corridor and to recognize that the far end has been reached), a sufficient number of robots or a sufficient sensing range to cover the region statically, or internal maps. Rather than a biomimetic, network-based, or decomposition approach, our approach was inspired by natural physics-based systems. In particular, we have designed a swarm of robots to model a fluid composed of particles, e.g., molecules. Each robot in the swarm adopts the role of a particle in a fluid. Both liquids and gases are considered to be fluids. Here, we focus on gases. The reason we selected gas-like swarm behavior for our model for a coverage and obstacle-avoidance task is that: gases are easily deformed, they are capable of coalescing after separating to go around objects, and they fill volumes. Gases also have desirable properties when they are in motion. For example, consider releasing a gas from a container at the top of a room with obstacles. Assume that the gas is slightly heavier than the ambient air. As the gas slowly falls to the floor, it will naturally separate to go around obstacles. Also, it will diffuse - by either laminar or turbulent processes - to cover areas under the obstacles. Therefore, robots capable of mimicking gas flow will be successful at avoiding obstacles and moving around them quickly and efficiently.

As just mentioned, a physics-based gas model approach is well-suited to coverage tasks. However, there is also another compelling reason for adopting a physics-based approach. Physics-based algorithms are easily amenable to physics-based analyses. Whenever a swarm solution is applied to a task, the question arises as to whether the emergent swarm behavior will be predictable, unpredictable, or even chaotic. Unfortunately, the vast majority of swarm researchers are unable to answer this question. This is highly troublesome since *predictable aggregate swarm behavior is critical for swarm acceptability*. Therefore, we have turned to particle physics approaches, which have a long history of theoretical analyses and accurate multi-particle predictions. Our research was also inspired by Jantz and Doty [11], who designed a very simple multi-robot system using *kinetic theory (KT)*. Because they used KT for design, they were able to directly apply standard KT analysis techniques. Robots behaved as particles. Jantz and Doty focused on the analysis aspect, rather than the system design. They developed simple, elegant formulas that were quite predictive of robot collision frequency, collision cross section, and the rate of multi-robot effusion through a hole in a wall. Nevertheless, other than our earlier work, e.g., [13], [14], the research presented here is the first of which we are aware that has explored kinetic theory primarily for swarm *design*.

We designed and implemented a physics-based approach based on kinetic theory (KT). The next section describes the details of this approach. KT is considered to be a *physicomimetic*, rather than a biomimetic approach. *Swarm systems designed with physicomimetics are easily analyzed using physics-based analysis methods* [23]. This enables one to control, predict, and guarantee that the behavior of a swarm will remain within desirable bounds.

V. KINETIC THEORY

There are two traditional methods available when designing an algorithm to model a gas: an Eulerian approach, which models the gas from the perspective of a finite volume fixed in space through which the gas flows (which includes computational fluid dynamics), and a Lagrangian approach (which includes kinetic theory), in which the frame of reference moves with the gas volume [1]. Because we are constructing a model from the perspective of the robots, we choose the latter.

When modeling a real gas, the number of particles is problematic, i.e., in a gas at standard temperature and pressure there are 2.687×10^{19} particles in a cubic centimeter. Utilizing today’s computational power, we are unable to model a gas deterministically. Therefore, a typical solution is to employ a stochastic model that calculates and updates the probabilities of the positions and velocities of the particles, also known as a Monte Carlo simulation. This is the basis of KT [8]. One advantage of this model is that it enables us to make stochastic predictions, such as the average behavior of the ensemble. The second advantage is that with real robots, we can implement KT with probabilistic robot actions, thereby avoiding predictability of the *individual* robot, e.g., for stealth. It is important to point out that although KT is based on having large numbers of particles (e.g., 10^{19}) we will demonstrate that the theoretical analysis is quite accurate with only hundreds of robots, and the actual task performance is quite adequate with as few as ten robots.

In KT, particles are treated as possessing no potential energy, i.e., an ideal gas, and collisions with other particles are modeled as purely elastic collisions that maintain conservation of momentum. The system consists entirely of kinetic energy. KT typically does not deal with potential energy or forces, which are central to Newtonian physics. Instead, with KT, increased particle movement is driven by collisions and/or a temperature increase.¹

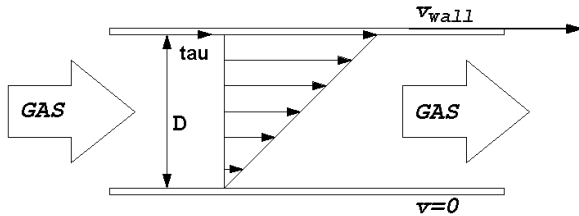


Fig. 1. Schematic for a one-sided Couette flow.

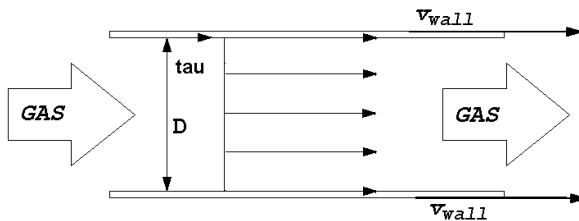


Fig. 2. Schematic for a two-sided Couette flow.

Our KT simulation algorithm is a variant of the particle simulations described in [8]. We substantially modified the algorithms in [8] to tailor them to simulated robots with local views. Robots, modeled as particles, behave in the aggregate like “Couette flow.” Figure 1, from [8], depicts one-sided Couette flow, where a fluid moves through some environment between two walls – one wall moving with velocity v_{wall} , and the other stationary (with respect to the environment as the frame of reference). In this Couette, fluid is assumed to move in the positive y -direction (i.e., longitudinally toward the goal end of the Couette corridor), and the positive x -direction goes from the stationary wall to the moving wall (i.e., laterally across the Couette corridor). Because the fluid is Newtonian and has viscosity, there is a linear velocity distribution across the system. Fluid deformation occurs because of the shear stress, τ , and the wall velocity is transferred (via kinetic energy) because of molecular friction on the particles that strike the wall. On the other hand, the particles that strike either wall will transfer kinetic energy to that wall. This does not cause the wall to change velocity, since in a Couette flow the walls are assumed to have infinite length and depth and therefore infinite mass. We chose a Couette flow in order to introduce kinetic energy into the system and to give the particles a direction to move.

Our 2D simulated world models a modified (two-sided) Couette flow in which both Couette walls are moving in the same direction with the same speed (see Figure 2). We invented this variant as a means of propelling all robots in a desired general direction, i.e., the large-scale fluid motion is approximately that of the walls. See Appendix A and [12] for details.

Robots’ velocities are randomly initialized (where initial velocities are directly proportional to a system temperature). These robot velocities remain constant, unless robot-robot or robot-wall collisions occur. (Note that with actual robots, robot-robot collisions would be virtual, i.e., they would be considered to occur when the robots get too close to each other. Robot-wall collisions and Wall velocity are also virtual.) The system updates the world in discrete time steps, Δt . We choose these time steps to occur on the order of the mean collision time for any given robot. Finer resolution time steps produce inefficiency in the simulation, while coarser resolution time steps produce inaccuracies.

Each robot can be described by a position vector, \vec{p} , and a velocity vector, \vec{v} . At each time step, the position of every robot is reset based on how far it could move in the given time step and its current velocity:

$$\vec{p} \leftarrow \vec{p} + \vec{v}\Delta t$$

Our KT robot algorithm is distributed, i.e., it allows each robot to act independently. Figure 3 shows the main pseudo-code for this algorithm. Robots are modeled as holonomic objects with a size of $6''$, speed and heading. At every time step, t , each robot needs to determine an angle, θ , to turn and a distance, δ , to move. At the beginning of the time step, the robot determines its proximity to the goal end of the corridor (e.g., based on light intensity). If its proximity to the goal

¹This is *virtual* system heat, rather than an actual heat, designed to increase kinetic energy and thereby increase particle motion.

```

float distance, turn, vx, vy;
float mpv = 1.11777f; (T=0.0030)
float stdev = 0.79038f;
float wall = 2.0f;
float collision-zone = 15.0f;
float Vmax = 6.0f;
boolean[] collision;
int[] timeCounter;

void move()
  incrementCollisionCounters();
  Read Goal Sensor (Light Sensor)
  float bearing = sensor.getBearing();
  Read Robot Localization Sensor
  int[] robotIds = localization.getRobotIds();
  float[] robotd = localization.getDistances();
  float[] robotθ = localization.getBearings();
  Read Sonar Sensors
  float[] sonard = sonar.getDistances();
  float[] sonarθ = sonar.getBearings();
  if ( !robotDirectlyInFront() )
    updateForWalls(bearing, sonarθ, sonard);
  updateForRobots(robotIds, robotθ, robotd);
  distance = sqrt(vx*vx + vy*vy);
  if (distance > Vmax) distance = Vmax;
  turn = atan2(vy, vx);

```

Fig. 3. KT pseudo-code.

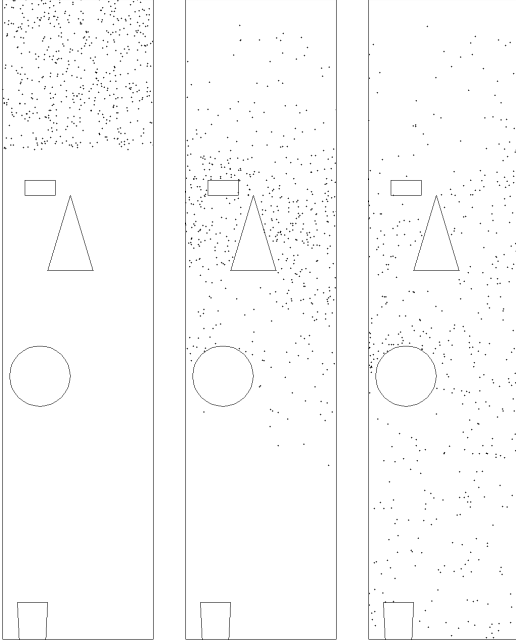


Fig. 4. KT performs a sweep downward toward the goal. The three snapshots show the swarm at three different moments in time.

exceeds a threshold, then one sweep has been completed; if one sweep is the robot’s objective, then it halts. If the robot is not at the goal, it will continue moving. In the case of no collisions, the robot maintains its current heading and speed.

However, if there are any imminent collisions, then they affect the robot’s turn and distance to move (i.e., speed). There are two possible types of collisions, robot-wall and robot-robot collisions (again, all collisions are virtual). The former are processed first. For imminent collisions of the robot with a wall, the robot gets a new velocity vector. It is important to note that due to local sensing constraints, a robot is unable to distinguish between a Couette wall and any obstacle wall

that is parallel to it. Hence we treat all such walls as Couette walls. If the robot-wall collision is “Couette,” the (virtual) wall velocity, v_{wall} , is added to the y -component of the velocity. Thus, obstacle walls can also increase the velocity of the robots toward the goal. Further details are given in Section V-A.

Next, robot-robot collisions are processed. The probability of a virtual collision with another robot is based on its proximity to that other robot, and it is independent of the angle between the two robots’ velocity vectors. In other words, the two robots may not actually be on a collision course. The concept of virtual collisions based on proximity is central to the kinetic theory framework. The new velocity vectors that result from this virtual collision are based on a center of mass vector, coupled with a random component. Further details are given in Section V-B.

To prevent repeated consecutive virtual collisions with the same robot/wall, collisions between the same entities are only processed again after a predetermined number of time steps have occurred. Because this process could in fact produce robots on actual collision courses, lower-level algorithms ensure that real physical collisions do not occur. The final robot velocity is the vector sum of all velocity vectors. Due to realistic assumptions concerning the maximum speed of a robot, the magnitude of this vector can be no higher than $V_{max} = 6''/\text{second}$. This KT algorithm halts when a sufficient percentage of the robots have completed a full sweep of the corridor. Figure 4 shows an example of a gas dynamically moving down an example obstacle-laden corridor. The corridors used in our experiments are $1000'' \times 5000''$.

A. Wall Collisions

As stated earlier, because robots cannot distinguish Couette walls from obstacle walls that are parallel to them, they assume that *any* wall parallel to the Couette walls (also parallel to the goal direction) is moving at speed v_{wall} . Therefore, the first portion of the wall collision algorithm is the determination of the angle of the wall that the robot has collided with. If the wall is determined to be approximately parallel to the goal direction (within ϵ degrees), the robot assumes that this wall is in motion. The user-defined threshold, ϵ , is based on the type of obstacles expected in the environment.

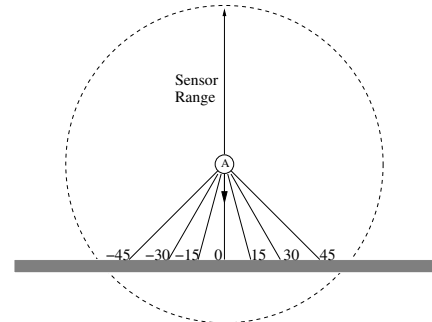


Fig. 5. The sonar information used to determine the orientation of a wall. The sonar at 0° represents the robot’s current heading.

We model each robot as having a ring of 24 sonar sensors, positioned every 15° around the robot. These sonars are used

for wall sensing and have a detection range of $60''$. For illustration, let us consider the processing from the perspective of robot A. Fig. 5 shows a potential collision of robot A with a wall. To determine if a wall is Couette, the robot has to determine the wall's orientation with respect to the goal direction. For this calculation, robot A is only concerned with the sonar sensors between -45° to 45° , because this is the direction that the robot is facing (see Fig. 5). First, the robot identifies two *endpoints*, which allow the robot to determine the wall orientation. The two endpoints are the left-most and right-most sonar sensors (from -45° to 45°) that register an object (which must be within $15''$ of the robot). It is assumed that the object is contiguous. The line segment between these two endpoints is considered to be a wall.

Robot A then determines if this line segment intersects a line from the robot to the goal, by calculating the intersection angle. If the lines do not intersect (again, within some threshold value ϵ), the wall is considered to be Couette. Although this process is error prone, it performs well in simulation. If a collision is imminent, then according to kinetic theory, the robot's new velocity must be drawn from a biased Maxwellian distribution [8], with an addition of v_{wall} to the y -component of the velocity. The biased Maxwellian distribution is obtained by using a distribution with a probability density function of $\lambda v_x e^{-\lambda v_x^2/2}$ (where $\lambda = m/kT$) in the x -direction, and a Gaussian $N(0, kT/m)$ distribution in the y -direction. Note that the temperature, T , plays an important role; increasing T increases the kinetic energy and hence the speed of the particles. Boltzmann's constant is k , and m is the particle mass. The result of the wall collision algorithm is a new velocity vector for A, in its local coordinate system.

B. Robot-Robot Collisions

Next, a robot must process (virtual) collisions with other robots. This involves robot localization. We have recently built a novel robot localization technique utilizing trilateration [9]. Each robot carries an RF transceiver and three acoustic transceivers. When a robot simultaneously emits an RF and acoustic pulse, other neighboring robots can localize the emitting robot. The RF is also used for communicating sensor information. Full details on this technique can be found in [22]. By using our trilaterative localization algorithm, a robot can sense the distance and bearing to all other neighboring robots, in its own local coordinate system. Furthermore, by communicating this information, robot A can know the orientation of robot B, and vice versa.

Let us consider the robot-robot collision algorithm in detail. If the distance between robots A and B is less than $15''$ (the "collision zone"), then it is assumed that there will be a robot-robot collision (Figure 6 – left). In this case, two steps are performed. In the first step, each robot computes the same *center of mass velocity* as follows:

$$\vec{v}_{cm} = \frac{1}{2} (\vec{v}_A + \vec{v}_B)$$

The center of mass velocity represents the velocity of the two particles if they actually collided (Figure 6 – middle). Because conservation of linear momentum holds, this center of

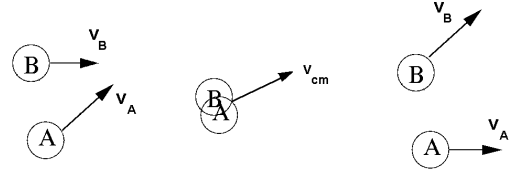


Fig. 6. How a robot-robot collision is processed. If two robots are too close, a collision is assumed (left). A center of mass vector is computed, conserving linear momentum (middle). Then the robots stochastically receive new velocity vectors (right).

mass velocity remains unchanged by the collision, and it forms the basis for generating the new velocity vectors. The second step involves computing new velocity vectors for each robot after the virtual collision. To do this, each robot first computes the *relative* speed of the other robot with respect to itself, e.g., $v_r = |v_A - v_B| = |v_B - v_A|$. When the robots separate from their collision, not only should linear momentum be conserved, but so should the relative speeds. Because kinetic theory is stochastic and does not compute precise collision angles like Newtonian physics does, a post-collision angle, θ , is drawn randomly from $U(0, 2\pi)$. The new relative velocity vector is $\vec{v}'_r = \langle v_r \cos(\theta), v_r \sin(\theta) \rangle$. Note that this transformation preserves the relative speed. Finally, at the end of the second step, robots A and B split the difference of the new relative velocity, centered around the center of mass velocity (Figure 6 – right), i.e.,²

$$\begin{aligned} \vec{v}'_A &= \vec{v}_{cm} + \frac{1}{2} \vec{v}'_r \\ \vec{v}'_B &= \vec{v}_{cm} - \frac{1}{2} \vec{v}'_r \end{aligned}$$

This collision process is repeated for all robots within A's collision zone. The final resultant velocity of each robot is a vector sum of all the new velocities from every collision. Again, we assume that the robots have low-level routines to prevent *actual* collisions.

VI. THEORETICAL PREDICTIONS

One of the key benefits of using a physics-based multi-robot system is that extensive theoretical (formal) analysis tools already exist for making predictions and guarantees about the behavior of the system. Furthermore, such analyses have the added benefit that their results can be used for setting system parameters in order to achieve desired multi-robot behavior. The advantages of this are enormous – one can transition directly from theory to a successful robot demo, without all the usual parameter tweaking. For an example of such a success using physicomimetics, see [23]. To demonstrate the feasibility of applying physics-based analysis techniques to physics-based systems, we make predictions that support some of our claims regarding the suitability of gas models for our coverage task.

Recall that our objectives are to sweep a corridor and avoid obstacles. The ultimate objective is coverage. We utilize our previous definitions of spatial coverage: *longitudinal* (in the goal direction) and *lateral* (orthogonal to the goal direction). Longitudinal coverage can be achieved by movement of the

²Each pair of robots must synchronize in order to select the same θ and to determine which gets the negative fraction and which gets the positive.

swarm in the goal direction; lateral coverage can be achieved by a uniform spatial distribution of the robots between the side walls. The objective of the coverage task is to maximize both longitudinal and lateral coverage in the minimum possible time (maximize temporal coverage).

To measure how well the robots achieve the task objective, we examine the following three metrics:

- 1) The degree to which the **spatial distribution** of the robots matches a uniform distribution. This is a measure of lateral coverage of the corridor and provides confirmation of the equilibrium behavior of KT.
- 2) The **average speed** of robots (averaged over all robots in the corridor). This is a measure of all three types of coverage: lateral, longitudinal, and temporal. Velocity is a vector with speed and direction. The type of coverage depends on the direction. To control the average swarm speed, one can directly vary the value of the system temperature, T . Therefore, our experiment explores the relationship between T and average speed.
- 3) The **velocity distribution** of all robots in the corridor. This is a measure of longitudinal spatial coverage, as well as temporal coverage. For example, consider the one-sided Couette in Figure 1 again, and in particular focus on the line representing the velocity distribution. The slope of that line is inversely proportional to the longitudinal spatial coverage (and the temporal coverage). In other words, for a given Couette diameter, D , if you increase the wall speed, v_{wall} , then the slope will be reduced and the longitudinal and temporal coverage will increase. Below, we run an experiment to examine the relationship between the wall speed, v_{wall} , and the velocity distribution in one-sided and two-sided Couettes. The intention is to enable the system designer to select a wall speed for optimizing this swarm velocity distribution.

The error between the theoretical predictions and the simulation results, denoted *relative error*, is defined as:

$$\frac{|theoretical - simulation|}{theoretical} \times 100$$

The theory presented in this paper assumes simplified 2D environments with no obstacles. To develop theory for the full task simulation would require extensive theoretical physics analyses. That is beyond the scope of this paper, but will be addressed in the future.

We ran three sets of experiments, in accordance with the metrics defined above. For each experiment, one parameter was perturbed and eight different values of the affected parameter were chosen. For each parameter value, 20 different runs through the simulator were executed, each with different random initial robot positions and velocities. The average simulation results and relative error (over the 20 runs) were computed and graphed.

Although the theory assumes no obstacles, we also ran with six obstacle densities ranging from 0% to 50%, in order to determine the sensitivity of the theory to obstacle density. Surprisingly, some of the theory holds well, despite the presence of obstacles.

There are two complementary goals for running these experiments. The first goal is to determine how predictive the theory is. Derivations of all laws (predictive theoretical formulas) are in Appendix A. The second goal is to determine the relationship between parameter settings and system behavior. If a system designer understands this relationship, he/she can more easily set parameters to achieve optimal performance. Finally, and most importantly, the reason why these two goals are complementary is that if the theory is predictive of the system simulation behavior, then future system designers no longer need to run the simulation to determine the optimal parameter settings – graphs generated from theory alone will suffice. This can greatly reduce the time required for system design and optimization.

VII. EXPERIMENT 1: SPATIAL DISTRIBUTION

The first experiment examines the lateral spatial coverage provided by KT. Robots are placed in a square container in an initially tight Gaussian distribution and allowed to diffuse to an equilibrium. During this experiment, there is no goal direction or wall movement.

The purpose of this experiment is to confirm the theoretically expected behavior of a KT system in equilibrium, and verify the correctness of our implementation. The KT gas model predicts that, upon reaching equilibrium, the particles will be spatially uniformly distributed. To confirm this prediction, we divided the square container into a 2D grid of cells. Theory predicts that there should be (on average) n/c robots per cell, where n is the total number of robots and c is the total number of grid cells. We ran with six obstacle densities, ranging from 0% to 50%, to determine the sensitivity of the spatial distribution to obstacle density.

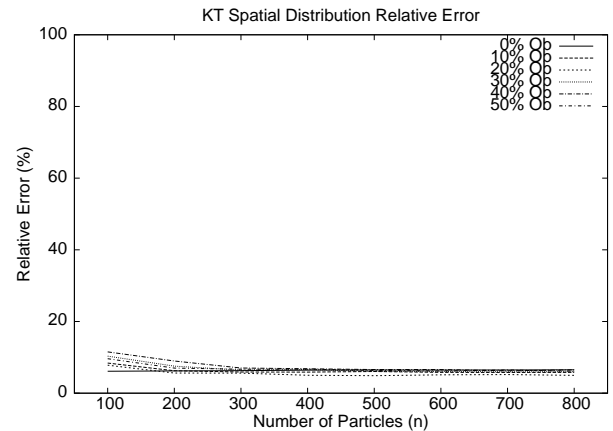


Fig. 7. Relative error for the KT spatial distribution.

Fig. 7 shows the results for the KT simulation. Note that despite the introduction of as much as a 50% obstacle coverage, we can still predict the spatial distribution with a relative error of less than 10%, which is surprisingly low. The error is very low once the number of robots is about 300, which is surprising considering that 300 robots is far less than the 10^{19} particles typically assumed by traditional kinetic theory.

VIII. EXPERIMENT 2: AVERAGE SPEED

For the second experiment, we examine the average speed of the robots in the system. There is no goal force or wall movement, and therefore no *externally-directed* bulk transport of the swarm. However, the robots will increase in speed if there is an increase in the system temperature, which causes an increase in kinetic energy. For example, the system temperature in the simulation directly affects the initial agent speeds, the new agent velocities after collisions, and the wall velocities, which are all drawn from a biased Maxwellian distribution that is a function of Boltzmann’s constant, the temperature, and the agent mass. The objective of this experiment is to examine the relationship between the temperature, T , and the average speed of the robots. The average robot speed serves as a measure of how well the system will be able to achieve complete coverage – because higher speed translates to greater lateral and/or longitudinal coverage, depending on the velocity direction. This experiment also serves to verify that our simulation code has been implemented correctly. Note that not all applications will require maximum coverage; therefore, we want to study the general question of precisely how specific choices of speed affect coverage.

Our predictive formula for 2D is (see Appendix A):

$$\langle v \rangle = \frac{1}{4} \sqrt{\frac{8\pi kT}{m}}$$

where k is Boltzmann’s constant ($1.38 \times 10^{23} J/K$), m is the robot mass (assumed to be one), and T is the system temperature (a user-defined system parameter analogous to real temperature).

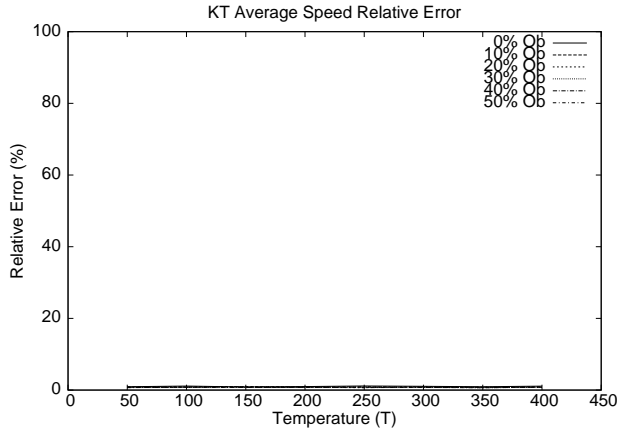


Fig. 8. Relative error for the KT average speed.

This theoretical formula is compared with the actual average speed, $\langle v \rangle$, of the robots in the simulation, after the system has converged to the equilibrium state. There are 300 robots in the simulation. Because temperature affects speed, temperature is varied from 50° Kelvin to 400° Kelvin. We ran with six obstacle densities ranging from 0% to 50%, in order to determine the sensitivity of the average speed to obstacle density.

The results are striking, as can be seen in Fig. 8. The difference between the theoretical predictions of the average

speed and the simulated average speed results in less than a 2% error, which is outstanding considering that as much as a 50% obstacle coverage has been introduced. Finally, note that we can use our theory to not only predict swarm behavior, but also to *control* it. Specifically, by setting T , a system designer can easily achieve a desired average speed.

IX. EXPERIMENT 3: VELOCITY DISTRIBUTION

The third experiment concerns longitudinal coverage and sweep speed via movement. Kinetic theory predicts what the velocity distribution will be for robots in a Couette. This theoretical prediction is compared with simulated behavior. Again, fluid flow is in the y -direction, which is toward the goal. The x -direction is lateral, across the corridor. This experiment examines the relationship between wall speed, v_{wall} , and the velocity distribution of the robots in the system.

We examine a subtask in which a traditional one-sided Couette flow drives the bulk swarm movement. Our predictive formula is (see Appendix A):

$$v_y = \frac{x}{D} v_{wall}$$

where v_{wall} is the velocity of the Couette wall, x is the lateral distance from the stationary wall, and D is the Couette width. In other words, the theory predicts a linear velocity distribution.

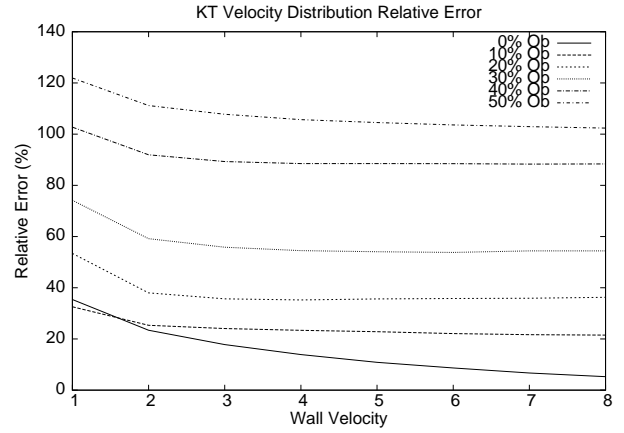


Fig. 9. Relative error for the KT velocity distribution.

We set up an experiment to measure the relative error between theory and simulation, consisting of 300 robots. The corridor is divided into eight discrete longitudinal sub-corridors. Theory predicts what the average speed will be lengthwise (in the goal direction) along each of the sub-corridors. Within each sub-corridor, the average y velocity of the robots is measured. The relative error between the theory and the experimental results is then calculated, for each sub-corridor. Finally, the relative error is averaged across all sub-corridors and plotted in Fig. 9 for eight different wall speeds and six different obstacle percentages. Although the error is reasonably low for 0% obstacles and high wall speeds, error increases dramatically as obstacles are added.

Why is there a discrepancy between the theory and experimental results? The reason is that theory predicts a certain

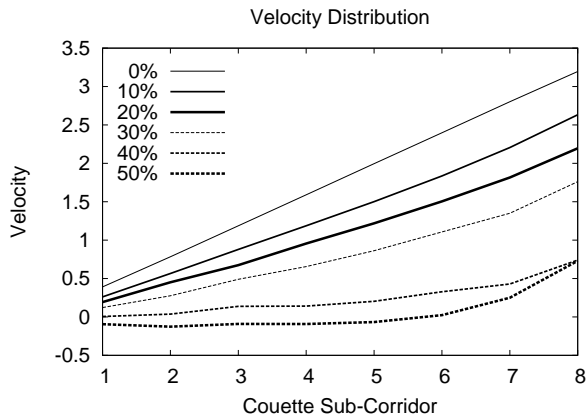


Fig. 10. The velocity distributions as the density of obstacles increases, for one-sided Couette flow.

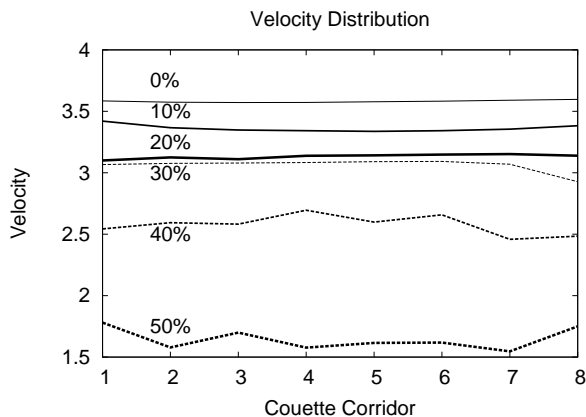


Fig. 11. The velocity distributions as the density of obstacles increases, for two-sided Couette flow.

linear velocity distribution, but assumes that there are no obstacles. For simplicity, the theory assumes that robots never move backward (back up the corridor). In the simulator, on the other hand, robots *do* move backward, regardless of whether or not there are obstacles – because the simulation has a random component. In fact, as obstacles are introduced into the simulated world, the frequency of backward moving robots increases substantially. To examine more closely the effect that obstacles have, Figure 10 shows the velocity distributions themselves (where the wall velocity $v_{wall} = 4$). Even with no obstacles, the maximum robot velocity does not quite reach 4.0 (we would expect 3.75 in the sub-corridor nearest to the wall). This is caused by the backward moves. What is interesting is that the velocity distributions remain linear up to a reasonable obstacle density (30%), while the slope decreases as obstacles are added. Adding obstacles is roughly equivalent, therefore, to lowering the wall velocity v_{wall} !

To see if the correlation between obstacle density and wall velocity holds in the two-sided Couette flow, we re-ran the previous experiment, but with *both* walls having $v_{wall} = 4$. The results are shown in Figure 11. The theory predicts (see Appendix A) that for the two-sided Couette, $v_y = v_{wall}$ regardless of the value of x . Note that, as theory predicts, the

velocity distribution of the flow is independent of the distance from the walls – the large scale fluid motion is approximately that of the walls. Again, increasing obstacle density is very similar to decreasing wall speed.

In conclusion, without obstacles, the theory becomes highly predictive as the wall velocity increases. Furthermore, this very predictive theoretical formula can also be used to achieve a desired swarm velocity distribution, i.e., to *control* the swarm – simply set the value of v_{wall} , the virtual wall speed, to achieve the desired distribution, using the formula. On the other hand, with an increasing number of obstacles, the predictability of the theory is increasingly reduced. However, we have shown that (up to quite reasonable densities) the increase in the number of obstacles is roughly proportional to a reduction in wall velocity, v_{wall} . Thus there is a quite reasonable expectation that this effect can be captured theoretically. We intend to explore this in our future work.

X. SUMMARY OF THEORETICAL/EMPIRICAL COMPARISONS

With respect to the average speed and the spatial distribution, the empirical results are extremely close to the theoretically derived values. Although the theory was not derived with obstacles in mind, obstacles have almost no effect on the results. The errors are generally less than 10%. Surprisingly, this level of theoretical accuracy is achieved with only hundreds of robots, which is quite small from a kinetic theory perspective.

Our results for the velocity distribution are acceptable for no obstacles, generally giving errors less than 20%. As the obstacle density increases, so does the error. However, we have shown that an increase in obstacle density changes the slope of the linear velocity distribution. This is roughly equivalent to a commensurate reduction in wall speed.

In summary, we can conclude that when the actual scenario closely coincides with the theoretical assumptions (e.g., few obstacles), the theory is highly predictive. Also, we have provided an important insight into the nature of the effect that obstacle density has on the system. The most important conclusion to be drawn from these experiments is that in the future we can largely design KT systems using theory, rather than computationally intensive simulations, for the selection of optimal parameter settings. A subsidiary conclusion is that we have verified the correctness of our swarm code, which is something quite straightforward for a physicomimetic approach but much more difficult for alternative approaches.

XI. PERFORMANCE EVALUATION ALGORITHMS

The next several sections describe the performance evaluations of our KT controller on the full coverage task described in Section II. In order to provide a comparison, we have implemented three other algorithms. The “Random” controller is our baseline and represents the minimal behavior needed for the task. The finite-state machine (FSM) controller uses a genetic algorithm to learn the task over time. The final controller is “Ant.” Ant provides upper bound on spatial performance because it is known to provide complete spatial coverage. We now describe these alternative controllers.

A. Random Controller

The first algorithm we compare against is the Random baseline controller. Although we call it “Random,” it is actually smarter than pure random. It attempts to exploit known information about searching “shadow” areas – regions just downstream (i.e., toward the goal direction) of obstacles. The robots used by Random are nearly the same as those used by KT. The primary difference is that Random robots have only eight possible moves (for algorithm simplicity), in 45° increments, as shown in Fig. 12. The controller weights each move as more or less desirable based on the information known to the robot. The weights provide a bias to maintain the same direction, move toward the goal, and exploit shadow areas. We designed Random in this fashion because the shadow areas represent the hardest areas to cover.

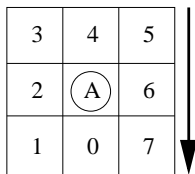


Fig. 12. The Random controller choices for moves. The arrow represents the heading of the robot.

The Random algorithm stochastically determines the next move by first sensing the environment. The robot assigns a weight to each available move, and then decides to continue along the same path 80% of the time if the move is available. It then divides the remaining weight into three parts. Two of the three parts go to shadow areas (see Figure 13) if they are available. The remaining one third is again split two ways. Half of this weight goes to the goal direction and the other half is divided evenly between all the open moves. The sum of these weights is equal to one. The robot’s next move is a weighted random choice.

B. Trained Finite-State Machine Controller

This work is loosely based on Spears’ earlier research on strategies for protecting resources [21]. A finite-state machine (FSM) is trained to maximize coverage when presented with random obstacle-laden corridors. An FSM is a model of computation composed of states, actions, and transitions. The FSM maintains two transition functions. The first function determines the next state of the FSM given the current state and the current input. The second function determines the

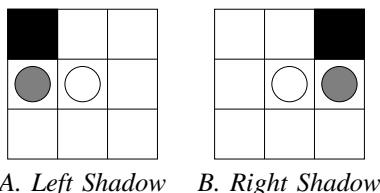


Fig. 13. Shadow preferences for the Random controller. The center open circle is the current position of the robot. The black squares represent walls or obstacles. The gray circle represents the preferred choice.

desired action of the FSM given the current state and current input. A genetic algorithm (GA) trains our FSMs. A genetic algorithm mimics natural evolution by maintaining a population of individuals. In our case, these individuals are finite-state machines. Each individual is given a fitness representing the degree of coverage. Those individuals with the highest fitness are selected to survive and reproduce, after which they are subject to recombination and mutation. The system evolves through many generations, thereby maximizing coverage.

The complexity of the FSM depends on the number of inputs. Previously, we assumed 24 sonar sensors. If each sonar has only 256 values, there would still be at least 256^{24} possible inputs, which is far too many. There are two solutions. The first is to lower the granularity of the sensors. The second is to “aggregate” adjacent sensors together. We do both.

Each FSM robot is equipped with eight sensors, one located every 45° . The sensors are very primitive, and only capable of seeing a distance equal to the speed of our robots (the distance traveled in one time step). Each sensor can only detect if an object is blocking the path or if the path is clear, i.e., each sensor has two states. Because the sensors can only see a limited distance in every direction, the world model is a discretized version of the real world. Therefore we have 2^8 possible inputs, which is tractable. Unfortunately, this degree of “aggregation” may very well place the FSM controller at a distinct disadvantage.

Representation: We represent the FSM as a table, as in [21]. For each state, s_i , and input, x_j , the table entry (i, j) contains the next state of the FSM, $\delta(s_i, x_j)$, and the action, a , to take. The number of states, S , can vary during learning, and the maximum is $S = 8$. As discussed above, the number of different possible inputs is 256. Hence, the size of our table is $S \times 256$.

Initialization: The objective is to allow the GA to generate a strategy that yields maximum coverage regardless of the number of obstacles and the size of the obstacles. Therefore, we introduce as few biases as possible. When initializing the “next states” in the table, a state is chosen randomly from $U(1, S)$. Care is taken when initializing actions – the only actions selected are those allowable for a given input. For example, actions that would result in a collision with a wall or obstacle are not allowed.

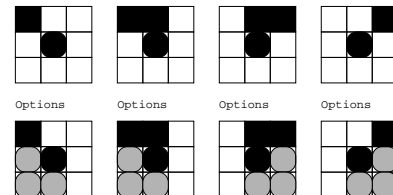


Fig. 14. Biases for the FSM. The center black circle is the current position of the robot. The black squares represent walls or obstacles. The options are shown as the gray circles. For each bias there are three preferred moves.

A bias was introduced in an attempt to ensure that the shadow areas are searched fully. Figure 14 shows these sensor readings and the preferred directions associated with them. When creating random FSMs or performing genetic operations

on existing FSMs, preference is given to choices that exploit shadow areas, when the sensors show them as available.

Operators: FSM table entries are modified by the genetic algorithm; this is accomplished via mutation and recombination. To keep the GA simple, we use uniform recombination with a probability of swapping alleles of $p = 0.5$. The probability of recombination is 0.6, as standard in the literature [6]. Mutation is performed as in [21]. Each allele is chosen for mutation with probability

$$p_m = \frac{1}{S \times 256}$$

Because each allele represents a (state/action) pair, we randomly decide whether the state or the action gets mutated. If the state is mutated, then a random state is chosen uniformly from $U(1, S)$. If the action is mutated, then a random action is chosen uniformly from the set of preferred or available actions for the given input.

Fitness Evaluation: To determine the fitness of each individual, the individual is run in our robotic simulator – to measure its performance. This simulator scales well to large numbers of robots and can represent an arbitrary-sized corridor. The corridor is divided into a 2D grid of cells in order to evaluate coverage. The evaluation consists of three parts. The first part measures the percent of complete coverage in the world, p_c . The second part of the fitness measures coverage of “shadow” regions, as discussed earlier. The percentage of shadow cells explored is the second measure of fitness, p_s . The robots are allotted only a certain number of time steps to explore the world. The percentage of the robots that get to the goal location (complete one sweep) in the allotted time is our third measure of fitness, p_t .

Each FSM is run through several different obstacle courses, with the values for p_c , p_s , and p_t averaged over these runs. To use a single fitness measure, we combine the three intermediate fitness values, weighted by their importance. The final fitness function is:

$$f = (c_1 \times p_c) + (c_2 \times p_s) + (c_3 \times p_t)$$

where

$$c_1 + c_2 + c_3 = 1.0$$

The results of successive runs of the genetic algorithm were used to tune the constants to $c_1 = 0.45$, $c_2 = 0.45$, and $c_3 = 0.1$.

Training, Selection and Termination: The FSMs were trained on random obstacle courses, up to an obstacle density of 30%. We used fitness proportional selection without elitism. For a termination criterion, we ran the GA a pre-defined number of generations (200). Selection was accomplished using Baker’s stochastic universal sampling [2]. The best controller found by the GA was selected as the one to compete with the other controllers.

C. Ant Controller

To experimentally compare our KT algorithm against what appeared to be the best performing and most appropriate alternative found in the literature, we duplicated the Ant

algorithm designed by Koenig, Szymanski, and Liu (2001). This algorithm employs graph theory to guarantee complete coverage. It assumes *stigmergy*, i.e., robots that can detect and leave *pheromones* (odors) in the environment. It is important to note that the Ant robot described here has an explicit advantage over our other algorithms, because it is able to interact with its environment.

The Ant algorithm determines its next move by investigating the areas surrounding itself in four directions: North, East, South, and West. After each direction has been examined, the direction with the minimum pheromone value is chosen as the direction to move. The robot then updates its current position before repeating the process [16].

XII. PERFORMANCE EVALUATION EXPERIMENT SETUP

To better understand the performance of our algorithms, we ran two sets of combinatorial experiments, varying the obstacle percentage in one set of experiments and varying the number of robots in the second set of experiments. For all of the combinatorial experiments, we compared all algorithms discussed above. To avoid unfairness in the comparison, the magnitude of the velocity of all robots was fixed to the same value, for all algorithms. First, let us formalize the problem for each of the experiments, and then define how the parameters were varied.

Measurements: The simulated world was divided into 50×250 cells. We applied the following performance metrics:

- 1) w , the *sweep time*, which measures the number of simulation time steps for *all* robots to get to the goal location. These experiments assume only one sweep. Sweep time is a measure of temporal coverage.
- 2) p_c , the *total spatial coverage*, which is the fraction of all cells that have been visited at least once by at least one robot, over one sweep.
- 3) p_s , the *shadow coverage*. This is the same as p_c , except that we take this measure over a small region downstream (with respect to the goal location) of the obstacles (see Fig. 15). The shadow region is the hardest to cover.

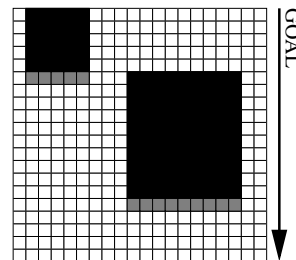


Fig. 15. Measurements taken by the simulation. Gray squares measure p_s . Obstacles are colored black. Free space is white.

Experiment 1:

- Ψ , a bounded 2D spatial corridor whose length is much greater than its width ($1000'' \times 5000''$).
- Γ , a swarm of 25 simulated robots.
- Λ , a set of randomly placed obstacles, varying from 0% to 40% coverage.

We varied the percentage of obstacles, Λ , to determine how this affects coverage. We generated five corridors for each obstacle percentage. Each individual experiment consisted of 10 trials through each of the five corridors, with random initial locations and orientations of the robots. The mean performance for each metric was graphed. Since each algorithm is not guaranteed to have all robots finish the obstacle course, we limited the total number of time steps to 150,000.

Experiment 2:

- Ψ , a bounded 2D spatial corridor whose length is much greater than its width ($1000'' \times 5000''$).
- Γ , a swarm of n simulated robots, varying from five to thirty robots.
- Λ , a set of randomly placed obstacles at 30% coverage.

In the second set of combinatorial experiments we varied the number of robots in order to see how coverage is affected when robots fail or are not available. Similarly to Experiment 1, we generated five corridors for each different number of robots, n . Each individual experiment consisted of 10 runs through each of the five corridors, with random initial locations and orientations of the robots. Again, we limited the total time to 150,000 steps.

XIII. PERFORMANCE EVALUATION EXPERIMENTAL RESULTS

For ease of presentation, we organize and present our results categorized according to the performance metric, instead of the combinatorial experiment. This allows us to look at different variations of parameter settings for each algorithm and how these settings affect specific metrics.

A. Temporal Coverage (w) Results

The first form of coverage the experiments measured was temporal coverage, w , also known as the sweep time. The results can be seen in Figure 16.

KT: KT was able to complete the task consistently, in a short time. As obstacles were added to the corridor, KT began to slow down. Although navigation became much tougher, KT's temporal coverage performance degraded gradually. This can be explained as an emergent behavior of the KT system. Areas of the corridor that require backtracking in order to successfully reach the goal have plagued prior solutions to the coverage task. These areas, referred to as obstacle "wells," or box canyons, are areas surrounded by three walls where the only direct path to the goal is blocked. KT has an inherent behavior (occasional stochastic backward moves) that allows it to escape these wells rather quickly.

For the second set of experiments, robots were added. As robots were added to the corridors, KT was able to continue to finish in roughly the same time. This is intuitively reasonable – with such a small number of robots their interactions are quite rare; the robots are almost independent of one another.

Ant: The Ant algorithm explored all areas. This caused Ant to take much more time than KT. As obstacles were added to the world, they did not appear to affect Ant. This is understandable, since obstacles just cover additional cells and Ant did not need to visit those cells.

On the second set of experiments, the number of robots, n , was increased. In this case, the time required for Ant to complete one sweep decreased. Ant is the only algorithm studied for which we observed this phenomenon. It is easily explained by the Ant algorithm. The algorithm visits those surrounding cells that have been visited the least. As more robots are added, the surrounding cells are given pheromones more quickly, causing the individual robots to be able to explore faster. Robots do interact with each other, through the environment.

FSM: Our trained FSM algorithm performed very poorly. With a longer learning time this problem may be overcome, but we can only comment on the current results. When any obstacles were added to the world, the FSM-controlled algorithm was unable to finish any runs. In retrospect, our constant $c_3 = 0.1$ was probably too low to emphasize this aspect of behavior when the GA was evolving the FSM. The second set of combinatorial experiments had obstacle coverage of 30%. The FSM controller was unable to finish in the allotted time for all runs, therefore achieving the worst possible temporal coverage.

Random: Our baseline controller actually performed the best on temporal coverage, up to a point. Once the obstacle percentage passed 30%, the Random controller quickly dropped in performance. At 40% obstacle coverage, the corridors have many obstacle wells and are difficult to navigate. By making purely stochastic decisions that emphasize exploration in the shadow areas, the Random algorithm begins to degrade.

The second set of experiments showed that as more robots were added to the Random controller, temporal performance was unaffected. This is intuitive since the robots controlled by Random are unaffected by what other robots do.

B. Total Spatial Coverage (p_c) Results

We have seen how the control algorithms perform with respect to temporal coverage. The next two subsections discuss how the algorithms fared with respect to two different measurements of spatial coverage. The first measurement is the total spatial coverage. 100% coverage may not be achieved, because the robots are shut down once they pass the goal line. This means that there are areas that may not be explored. These spaces could be explored on a second sweep, but for our experiments we are only concerned with a single sweep. The results for p_c can be seen in Figure 17.

KT: Inspecting Figure 17, it can be observed that as the obstacle percent increased, KT surprisingly performed better. Performance improved by several percentage points, and slowly tapered off as the corridors reached 40% obstacle coverage. As the obstacle percentage increased, the number of collisions with obstacles also increased. This slowed the robots' longitudinal movement (in the goal direction) and increased the lateral movement (orthogonal to the goal). This increase in the lateral movement increased spatial coverage. The slowing in the longitudinal direction can also be seen in Figure 10.

The second graph shows that as more robots were added, KT's coverage increased. This is as expected and it is important to note that with 30 robots, KT was able to achieve

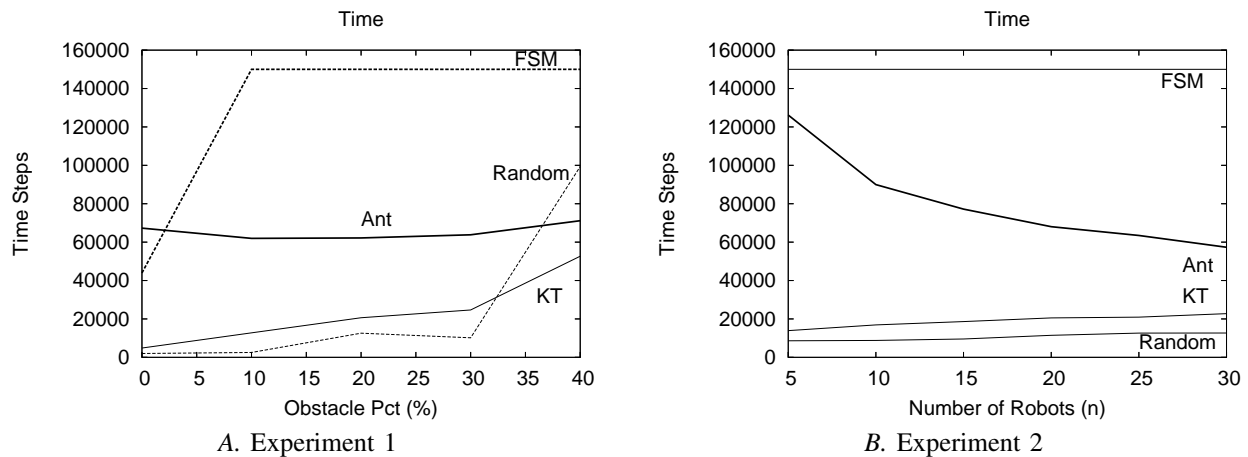


Fig. 16. Temporal Coverage.

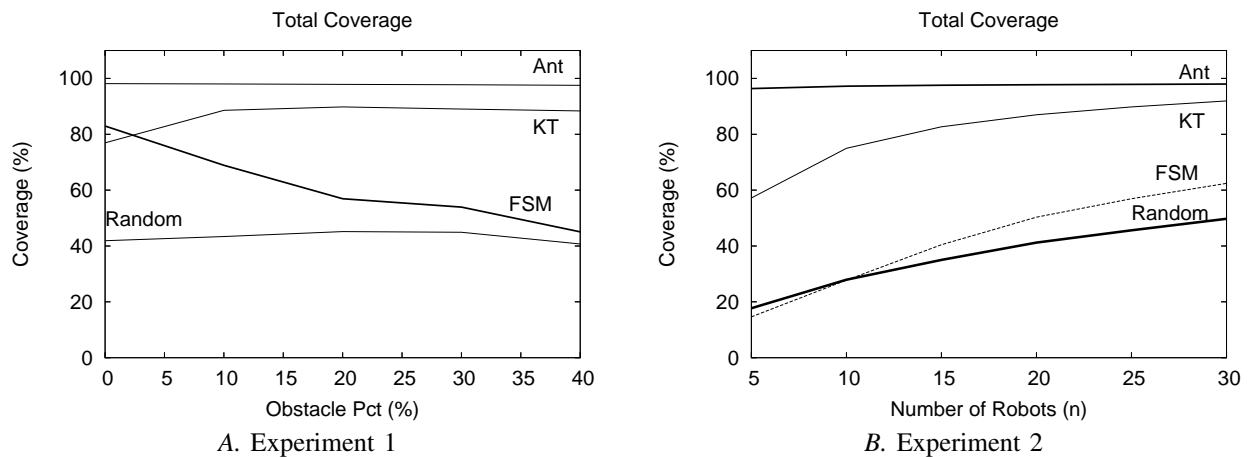


Fig. 17. Total Spatial Coverage.

roughly 90% coverage, which is close to the success of the Ant algorithm. This was even accomplished without leaving pheromones in the environment and is done in roughly 1/3 the time!

Ant: Ant is known to be a complete solution to the task problem. Therefore, it is no surprise to see that for both sets of experiments Ant set the maximum standard by achieving the best coverage.

FSM: The FSM performance was somewhat surprising. The most important results came from the corridors without obstacles. When watching the robots move through the corridor, it appeared that the GA learned a sweeping motion to increase coverage. We noticed that the FSM controller actually performed the second best when there were no obstacles, but KT quickly regained the advantage as soon as obstacles were added to the corridor. As more obstacles were added to the environment, the FSMs performance deteriorated until it performed roughly the same as Random. It is also important to consider temporal coverage along with spatial coverage. Robots were allowed 150,000 time steps to perform one sweep, yet the FSM controller never finished if obstacles were added to the environment. This means that it is likely that

there are cycles in the robots' behavior. These cycles would directly affect the amount of coverage that the FSM controller would be able to achieve. Such cycles were also observed by Spears and Gordon-Spears (2002). Future work will attempt to overcome the cycle problem by adopting a solution similar to that of Spears and Gordon-Spears (2002).

In the second set of experiments, it is clear that performance for the FSM improved as more robots were added to the environment. Again, this is not a surprising result. More robots means more independent exploration.

Random: Our final algorithm was supposed to show the baseline performance. The Random controller attempts to combine greedy exploitation of shadow regions with exploration, in a hand-crafted manner. This algorithm performed surprisingly well, covering roughly 45% of the world, regardless of the number of obstacles.

The second set of experiments provided no new insights on Random. The results improved monotonically as more robots were added. Since the robots' decision making ability is independent of other robots in Random, this is not surprising.

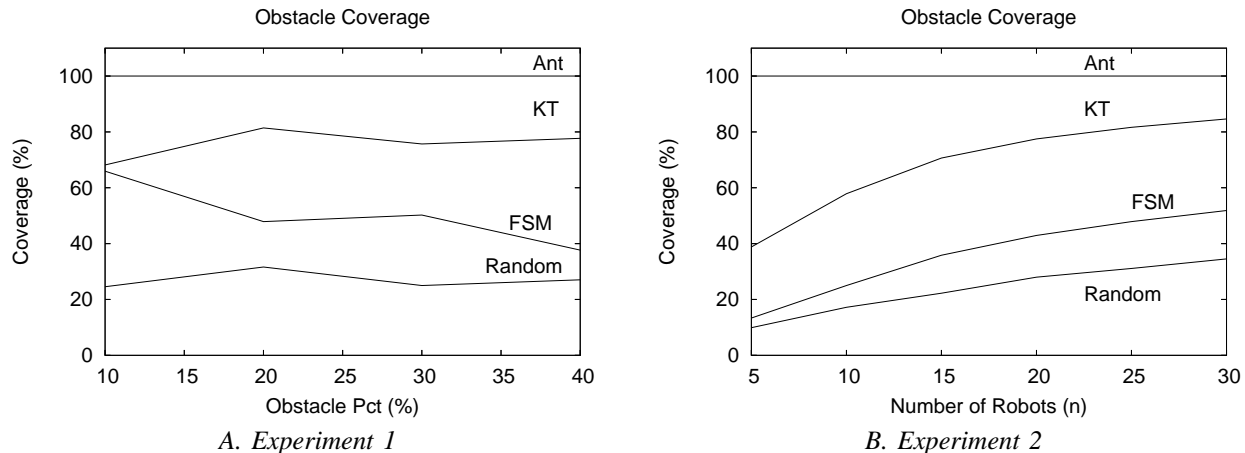


Fig. 18. Shadow Coverage.

C. Shadow Coverage (p_s) Results

The final measurement is shadow coverage. This is also a measure of how well the controller is able to explore. If the controller always goes directly to the goal, it will achieve a 0% shadow coverage. If the controller is optimal at exploration, then it will achieve a 100% shadow coverage. Note that greater shadow exploration affects not only shadow coverage, but it also affects total corridor coverage.

KT: *KT* performed very well on shadow coverage, which is especially important for effective task achievement, as can be seen in Figure 18. Regardless of the number of obstacles, *KT* was able to obtain roughly 80% shadow coverage. This implies that *KT* has a very good balance between exploitation and exploration. Apparently *KT* is adept at shadow coverage, despite this objective not having been explicitly designed into *KT*.

As robots were added to the system, the performance of *KT* gradually improved. An implication of this result is that if robots begin to fail, then the performance of *KT* will also degrade gradually, which is a desirable form of system robustness.

Ant: The *Ant* controller performed exactly as designed and achieved 100% coverage behind obstacles, regardless of the number of robots or the number of obstacles in the environment.

FSM: The *FSM*'s performance degraded as more obstacles were added to the corridor. It is likely that the cyclic behaviors mentioned earlier are contributing to this difficulty also.

The results of the second set of experiments shows that as more robots were added, the *FSM* controller was able to explore more shadow areas. This was expected.

Random: *Random* provided us with a usable baseline. With random search, we were able to achieve roughly 20% shadow coverage. Clearly, the other algorithms outperform random search.

Our baseline algorithm's performance was improved when extra robots were added to the environment. This was another expected result.

XIV. PERFORMANCE EVALUATION CONCLUSIONS

Several important results have been presented. First, the experimental results showed that *Ant* is superior in terms of spatial coverage. This superiority comes with numerous tradeoffs, however. First, *Ant* is much slower in terms of sweep time. Second, pheromones are required. Finally, the movements of the individual *Ant* robots are quite predictable. The latter two features imply that the *Ant* robots are susceptible to adversarial deception, lack of stealth, and interception.

Second, we showed the applicability of *KT* to the coverage task. Other than the *Ant* algorithm, no other algorithm was able to compete with *KT*. First, *KT* was able to provide excellent coverage *without* leaving pheromones in the environment. Second, if the coverage task requires multiple sweeps, *KT* clearly would have the advantage, since the sweep time is much less. Finally, the lack of predictability of individual robots using *KT* makes it hard for an adversary to anticipate future movements.

All of the results for *KT* shown above are statistically significant using a Wilcoxon rank sum test with a p -value of 0.05, except for the initial total spatial coverage at 0% and 10% corridor coverage.

XV. ANALYSIS OF SENSITIVITY TO NOISE

Since our simulation modeled an ideal world without noise, we conclude with results showing the relative insensitivity of *KT* to increasing amounts of noise. We focused on four sources of sensor noise. The first two stem from the trilaterative localization technique, which provides the range and bearing to a nearby robot. Both range and bearing can be noisy. Third, our sonars can provide noisy range readings. Finally, our goal direction sensor can exhibit noise.

Once again we focus on our three performance metrics; the sweep time w , the total spatial coverage p_c , and the shadow coverage p_s . The number of robots is 20 and the obstacle density is 20%. Noise is modeled as an additive Gaussian $N(0, \sigma)$, where σ ranges from 0.0 to 3.5. This translates into several inches of range noise and several degrees of

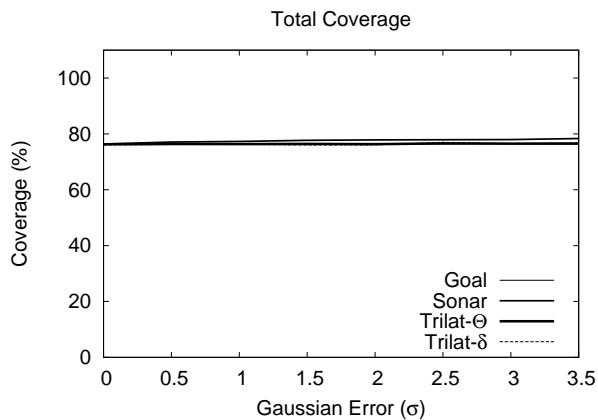


Fig. 19. Total spatial coverage, p_c , versus noise. *Goal* refers to goal noise. *Sonar* refers to sonar noise. *Trilat- θ* and *trilat- δ* refer to localization error in bearing and range, respectively.

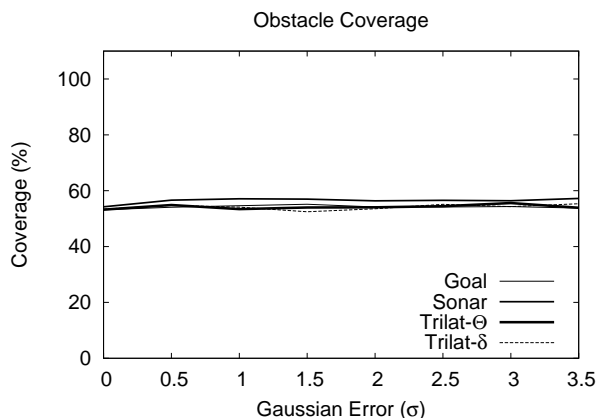


Fig. 20. Total shadow coverage, p_s , versus noise. *Goal* refers to goal noise. *Sonar* refers to sonar noise. *Trilat- θ* and *trilat- δ* refer to localization error in bearing and range, respectively.

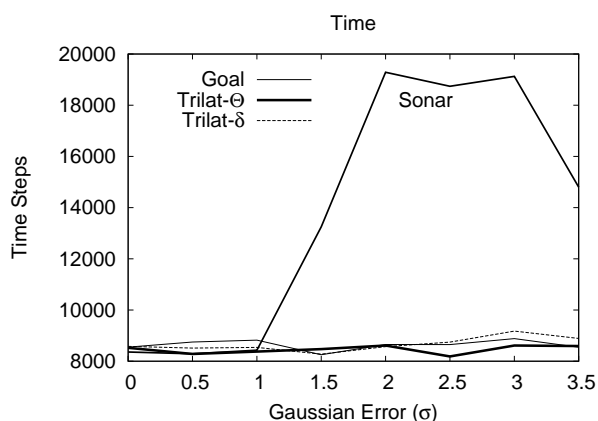


Fig. 21. Sweep time, w , versus noise. *Goal* refers to goal noise. *Sonar* refers to sonar noise. *Trilat- θ* and *trilat- δ* refer to localization error in bearing and range, respectively.

angular noise. The sensors on our own robots have noise approximately at this level [22], [23].

Figures 19 and 20 show the impact of noise on spatial coverage. The results are very impressive. Sensor noise has

essentially no effect on spatial coverage! We believe this is due to the already quite stochastic nature of KT. In essence, KT without sensor noise is already a “noisy” algorithm. It is precisely this feature that allows one to predict aggregate performance while maintaining the unpredictability of the individual robots.

Figure 21 show the results of noise on temporal coverage. Noise in the goal direction and localization noise have almost no effect on temporal coverage. The only noise that has an effect is due to sonar error. It appears as if increasing sensor noise at first causes false positives in terms of obstacle collisions (i.e., a robot responds to obstacles that it normally would not). This is analogous to increasing the obstacle density, which tends to increase the sweep time. At a certain point, however, further increases in sensor noise cause false negatives, and the robot starts to miss obstacles. This corresponds to decreasing obstacle density, which tends to decrease the sweep time. In the worse case the sweep time is doubled. However, it is important to point out that sonar error can easily be reduced by averaging over multiple acoustic “pings,” so we can easily mitigate this effect.

XVI. SUMMARY AND FUTURE WORK

We successfully created a new physics-based algorithm for controlling swarms of autonomous robots to achieve a challenging coverage task. Our algorithm requires limited sensors and communication between robots, and only a small amount of global knowledge. The algorithm was inspired by particle motion in a gas, which is a highly effective mechanism for expanding to cover complex environments in the natural world. Like the natural gases from which it was inspired, our KT algorithm is efficient, effective, and robust at maximizing coverage. It also has the advantages of stealth, very good noise tolerance, and predictability in the aggregate. Furthermore, because our KT controller is physics-based, behavioral assurances about the swarm are easily derived from well-known theoretical laws, e.g., [8]. These theoretical laws allow system designers to avoid costly trial-and-error – system parameters for optimal performance can be set using the theory alone.

When experimentally compared against three alternative algorithms designed for coverage tasks, KT has proven to be competitive. It provides excellent coverage in little time and is exceptionally robust. It does not provide quite as much spatial coverage as the Ant algorithm, which is the best known competitor to date, but it provides much better temporal coverage and also does not have the explicit pheromone trail requirements of Ant. Therefore, KT is better-suited than Ant for situations where inexpensive robot platforms, stealth, and/or speed of coverage are required.

Our highest priority for future work will be to further develop the theoretical analysis and to implement KT on actual robotic platforms. A physicomimetics algorithm for robot formations has already been successfully demonstrated on the UW Distributed Laboratory robots [23]. Further parallel theoretical and simulation developments will lead to more efficient, effective, and predictable physicomimetic swarms for practical tasks.

REFERENCES

- [1] J. Anderson, *Computational Fluid Dynamics*. McGraw-Hill, 1995.
- [2] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proceedings of the Second International Conference on Genetic Algorithms and their Application*, 1987, pp. 14–21.
- [3] T. Balch and M. Hybinette, "Social potentials for scalable multi-robot formations," in *IEEE Int. Conf. on Robotics and Automation*, vol. 1, 2000, pp. 73–80.
- [4] M. Batalin and G. Sukhatme, "Spreading out: A local approach to multi-robot coverage," in *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems*, 2002, pp. 373–382.
- [5] O. Bayazit, J. Lien, and N. Amato, "Better group behaviors in complex environments with global roadmaps," in *Proceedings Int. Conf. on the Simulation and Synthesis of Living Systems*, 2002, pp. 362–370.
- [6] K. DeJong, "Analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, Ann Arbor, 1975.
- [7] R. Feynman, R. Leighton, and M. Sands, *The Feynman Lectures on Physics*. Addison-Wesley Publishing Company, 1963.
- [8] A. Garcia, *Numerical Methods for Physics*, 2nd ed. Prentice Hall, 2000.
- [9] R. Heil, "A trilateral localization system for small robots in swarms," Master's thesis, University of Wyoming, 2004.
- [10] S. Hettiarachchi and W. Spears, "Moving swarm formations through obstacle fields," in *International Conference on Artificial Intelligence*, 2005.
- [11] S. Jantz and K. Doty, "Kinetics of robotics: The development of universal metrics in robotic swarms," Dept of Electrical Engineering, University of Florida, Tech. Rep., 1997.
- [12] W. Kerr, "Physics-based multiagent swarms and their application to coverage problems," Master's thesis, University of Wyoming, 2005.
- [13] W. Kerr and D. Spears, "Robotic simulation of gases for a surveillance task," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, 2005.
- [14] W. Kerr, D. Spears, W. Spears, and D. Thayer, "Two formal fluids models for multiagent sweeping and obstacle avoidance," *Lecture Notes in Artificial Intelligence*, vol. 3228, 2004.
- [15] S. Koenig and Y. Liu, "Terrain coverage with ant robots: A simulation study," in *Agents'01*, 2001, pp. 600–607.
- [16] S. Koenig, B. Szymanski, and Y. Liu, "Efficient and inefficient ant coverage methods," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 41–76, 2001.
- [17] 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*, vol. 48, 2003, pp. 76–95.
- [18] S. Megerian, F. Koushanfar, M. Potkonjak, and M. Srivastava, "Worst and best-case coverage in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 1, pp. 84–92, 2005.
- [19] F. Reif, *Fundamentals of Statistical and Thermal Physics*. McGraw-Hill, 1965.
- [20] I. Rekleitis, V. Lee-Shue, A. P. New, and H. Choset, "Limited communication, multi-robot team coverage," in *2004 IEEE International Conference on Robotics and Automation*, 2004.
- [21] W. Spears and D. Gordon-Spears, *Theory and Applications of Evolutionary Computation: Recent Trends*. Springer-Verlag, 2002, ch. Evolution of Strategies for Resource Protection Problems, pp. 367–392.
- [22] W. Spears, J. Hamann, P. Maxim, T. Kunkel, R. Heil, D. Zarzhitsky, D. Spears, and C. Karlsson, "Where are you?" in *Second International Workshop on Swarm Robotics*, E. Sahin, W. Spears, and A. Winfield, Eds., to appear.
- [23] W. Spears, D. Spears, J. Hamann, and R. Heil, "Distributed, physics-based control of swarms of vehicles," *Autonomous Robots*, vol. 17, pp. 137–162, August 2004.



Diana Spears received the M.S. degree in 1986 and the Ph.D. degree in 1990, both from the University of Maryland College Park (UMCP) in Computer Science. Prior to coming to the University of Wyoming (UW) in 2001, she worked at NASA Goddard Space Flight Center, National Institute of Standards and Technology, and the Naval Research Laboratory (NRL). She is currently an Associate Professor at UW and Co-Director of the UW Distributed Robotics Laboratory. Dr. Spears was a recipient of the NRL Alan Berman Research Publication Award

in 1994 and in 2001.

Her primary research interests are in the areas of multi-robot applications (e.g., chemical plume tracing, surveillance), swarm engineering, and the verification of multi-agent system behavior.

Wesley Kerr received his B.S. degree in computer science from Kansas State University, in 2001, and M.S. degree in computer science from University of Wyoming at Laramie in 2005. Currently, he is a Ph.D. student in the Department of Computer Science, University of Southern California, Los Angeles.

His research interests are swarms, genetic algorithms, cognitive science, artificial intelligence.



William Spears received a Ph.D. in C.S. from George Mason University in 1998. Prior to coming to the University of Wyoming (UW) in 2001, he worked at the Naval Research Laboratory. He is currently an Associate Professor at UW and Co-Director of the UW Distributed Robotics Laboratory.

His current research includes swarm robotics, the epidemiology of computer virus spread, evolutionary algorithms, parallel computation, complex adaptive systems, and learning and adaptation.



APPENDIX

Appendix A: Derivations of Behavioral Laws**Derivation of KT Law for Velocity Distribution**

For KT, a traditional one-sided Couette drives the bulk swarm movement. The complete derivation for the velocity distribution of a Couette flow can be found in [1] (pages 417–420), but here we present a more concise version.

For steady, 2D flow with no external forces, there is a classical ‘‘Governing Equation’’ that predicts the y -direction momentum of the fluid. This Governing Equation is:

$$\rho v_y \frac{\partial v_y}{\partial y} + \rho v_x \frac{\partial v_y}{\partial x} = -\frac{\partial P}{\partial y} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} \quad (1)$$

where ρ is the fluid density, v_x and v_y are the x - and y -components of velocity, P is the fluid pressure, and τ_{yy} and τ_{xy} are the normal and shear stresses, respectively. We can use this momentum equation to derive the velocity. However, first we need to specialize the equation for our particular situation. We assume that we have a Newtonian fluid, parallel flow, and zero pressure gradient. This implies the stresses are:

$$\begin{aligned} \tau_{yy} &= \lambda \left(\frac{\partial v_y}{\partial y} + \frac{\partial v_x}{\partial x} \right) + 2\mu \frac{\partial v_y}{\partial y} = 0 \\ \tau_{xy} &= \mu \left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \right) = \mu \frac{\partial v_y}{\partial x} \end{aligned}$$

We also know two other facts about Couette flow. First, there is no component of velocity going between the Couette walls, i.e., $v_x = 0$. Second, because the Couette walls, as well as the fluid flowing between them, extend infinitely in the $\pm y$ -directions, $\frac{\partial}{\partial y} = 0$ for all quantities. Substituting all of the above into Equation 1 yields:

$$0 = \frac{\partial}{\partial x} \left(\mu \frac{\partial v_y}{\partial x} \right)$$

where μ is the fluid viscosity. Assuming an incompressible, constant temperature flow with constant viscosity, this becomes:

$$\frac{\partial^2 v_y}{\partial x^2} = 0 \quad (2)$$

Equation 2 is the Governing Equation for steady, 2D, incompressible, constant temperature Couette flow. Integrating twice with respect to x to find v_y , we get:

$$v_y = c_1 x + c_2 \quad (3)$$

We can solve for c_1 and c_2 from the boundary conditions for the one-sided Couette. In particular, at the stationary Couette wall ($x = 0$), $v_y = 0$, which implies that $c_2 = 0$ from Equation 2. At the moving wall ($x = D$), $v_y = v_{wall}$, where D is the Couette width and v_{wall} is the velocity of the moving wall, which is in the y -direction (toward the goal). Then $c_1 = v_{wall}/D$ from Equation 2.

Substituting these values for c_1 and c_2 back into Equation 2, we get:

$$\begin{aligned} \frac{v_y}{v_{wall}} &= \frac{x}{D} \\ v_y &= \frac{x}{D} v_{wall} \end{aligned}$$

This is a linear velocity distribution.

Note that we can alternatively solve for c_1 and c_2 from the boundary conditions for the *two*-sided Couette. In particular, at *both* ($x = 0$) and ($x = D$), $v_y = v_{wall}$, where D is the Couette width and v_{wall} is the velocity of the moving walls. This implies that $c_1 = 0$ and $c_2 = v_{wall}$. Then $v_y = v_{wall}$ regardless of the value of x . In other words, all of the fluid between the walls eventually moves at the speed of the walls, once a steady state is reached. This is again a linear velocity distribution, but in this case the line is perpendicular to the walls.

Derivation of KT Law for Average Speed

We next show how we derive a KT formula for average speed by modifying the derivation for 3D $\langle v \rangle$ in [8] to a 2D formula for $\langle v \rangle$ (so it applies to our simulation). Assuming a system in thermodynamic equilibrium (since there is no bulk transport), with velocity components within the ranges $v_x + dv_x$ and $v_y + dv_y$, and k is Boltzmann’s constant, m is the robot mass, v is the magnitude of the robot velocity (i.e., the robot speed), and T is the initial system temperature (a simple, settable system parameter), then the probability, $f(v_x, v_y) dv_x dv_y$, that a robot has velocity components in these ranges is proportional to $e^{(-mv^2/2kT)} dv_x dv_y$. In particular, we have:

$$\begin{aligned} f(v_x, v_y) dv_x dv_y &= A e^{(-mv^2/2kT)} dv_x dv_y \\ &= A e^{(-mv_x^2/2kT)} e^{(-mv_y^2/2kT)} dv_x dv_y \end{aligned}$$

because $v^2 = v_x^2 + v_y^2$, and A is a normalization constant that is fixed by the requirement that the integral of the probability over all possible states must be equal to 1, i.e.,

$$\int_0^\infty f(v_x, v_y) dv_x dv_y = 1$$

Therefore,

$$A = \frac{1}{\int_0^\infty e^{(-mv_x^2/2kT)} dv_x \int_0^\infty e^{(-mv_y^2/2kT)} dv_y}$$

To simplify the expression for A , we can use the fact (from pages 40-46 of [7]) that:

$$\int_0^\infty e^{(-mv_x^2/2kT)} dv_x = \sqrt{\frac{2\pi kT}{m}}$$

and then do likewise for v_y . Therefore:

$$f(v_x, v_y) dv_x dv_y = \left(\frac{m}{2\pi kT} \right) (e^{(-m(v_x^2 + v_y^2)/2kT)}) dv_x dv_y$$

where $\frac{m}{2\pi kT}$ is A .

Note, however, that $f(v_x, v_y) dv_x dv_y$ is a probability for a velocity *vector*, but we want average *speed*. To get average

speed, the math is easier if we go from Cartesian to polar coordinates. In particular, to go from velocity to speed, we integrate over all angles.

In polar coordinates, $2\pi v dv$ is the area of extension (annulus) due to Δv . In other words, the area of an annulus whose inner radius is v and outer radius is $v + dv$ is $2\pi v dv$. Then the Maxwell-Boltzmann distribution of speeds, $f(v)dv$, is obtained by integrating the velocity distribution, $f(v_x, v_y)dv_x dv_y$, over all angles from 0 to 2π . This integration yields:

$$f(v)dv = 2\pi v \left(\frac{m}{2\pi kT} \right) (e^{-mv^2/2kT})dv$$

Canceling terms, the right-hand side becomes:

$$= v \left(\frac{m}{kT} \right) (e^{-mv^2/2kT})dv$$

Because $\langle v \rangle$ is an expected value,

$$\langle v \rangle = \int_0^\infty v f(v)dv = \frac{m}{kT} \int_0^\infty v^2 (e^{-mv^2/2kT})dv$$

From [19] (page 609), we know that $\int_0^\infty e^{-ax^2} x^2 dx = \frac{1}{4} \sqrt{\pi} a^{-\frac{3}{2}}$. Substituting v for x and $\frac{m}{2kT}$ for a , we get:

$$\langle v \rangle = \frac{1}{4} \sqrt{\frac{8\pi kT}{m}}$$