Algorithms For Shaping a Particle Swarm With a Shared Control Input Using Boundary Interaction*

Shiva Shahrokhi, Arun Mahadev, and Aaron T. Becker

***Abstract*— Consider a swarm of particles controlled by global inputs. This paper presents algorithms for shaping such swarms in 2D using boundary walls. The range of configurations created by conforming a swarm to a boundary wall is limited. We describe the set of stable configurations of a particle swarm in two canonical workspaces, a circle and a square. To increase the diversity of configurations, we add boundary interaction to our model. We provide algorithms using friction with walls to place two robots at arbitrary locations in a rectangular workspace. Next, we extend this algorithm to place n agents at desired locations. We conclude with efficient techniques to generate correlations of a particle swarm not possible without wall-friction. Simulations and hardware implementations with 100 robots validate these results.**

These methods may have particular relevance for micro- and nano-robots controlled by global inputs.

I. Introduction

Particle swarms propelled by a global field are common in applied mathematics, biology, and computer graphics. If the ith particle has position $[x_i, y_i]$ and velocity $[\dot{x}_i, \dot{y}_i]$, then

$$[\dot{x}_i, \dot{y}_i]^\top = \mathbf{u}(t), \qquad i \in [1, n]. \tag{1}$$

These system dynamics represent particle swarms in low-Reynolds number environments, where viscosity dominates inertial forces and so velocity is proportional to input force [**?**]. In this regime, the input force command $\mathbf{u}(t)$ controls the velocity of the robots. The same model can be generalized to particles moved by fluid flow where the vector direction of fluid flow $\mathbf{u}(t)$ controls the velocity of particles, or for a swarm of robots that move at a constant speed in a direction specified by a global input $\mathbf{u}(t)$ [**?**]. Our control problem is to design the control inputs $\mathbf{u}(t)$ to make all n particles achieve a task. As a current example, micro- and nano-robots can be manufactured in large numbers, see [**? ? ? ? ? ? ? ?**]. Someday large swarms of robots will be remotely guided to assemble structures in parallel and through the human body to cure disease, heal tissue, and prevent infection. For each task, large numbers of micro robots are required to deliver sufficient payloads, but the small size of these robots makes it difficult to perform onboard computation. Instead, these robots are often controlled by a global, broadcast signal. They require techniques to shape the swarm that can reliably exploit large populations despite severe underactuation ($2 \ll 2n$).

*This work was supported by the National Science Foundation under Grant No. [IIS-1553063] and [IIS-1619278].

Authors are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204 USA {sshahrokhi2, aviswanathanmahadev, atbecker}@uh.edu

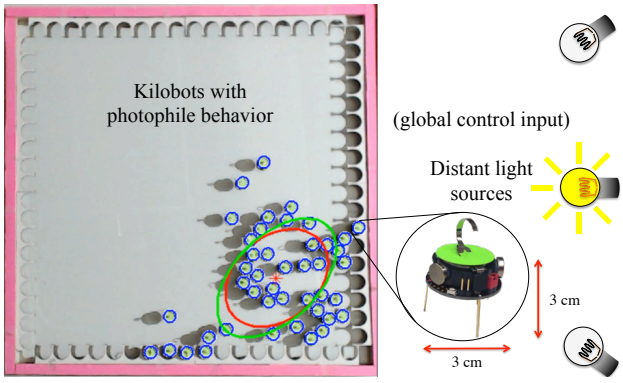

Fig. 1. Swarm of kilobots programmed to move toward the brightest light source as explained in Sec. **??**. The current covariance ellipse and mean are shown in red, the desired covariance is shown in green. Navigating a swarm using global inputs is challenging because each member receives the same control inputs. This paper focuses on using boundary walls and wall friction to break the symmetry caused by the global input and control the shape of a swarm. See multimedia attachment for overview of the experiments.

Even without obstacles or boundaries, the mean position of the swarm in (**??**) is controllable. By adding rectangular boundary walls, some higher-order moments, such as the swarm's position variance orthogonal to the boundary walls (σ_x and σ_y for a workspace with axis-aligned walls), are also controllable [**?**]. These techniques could only compress a swarm orthogonal to obstacles. Swarm shape control is necessary for navigation through narrow passages, for self-assembly, and for manipulating objects.

The paper is arranged as follows. After a review of recent related work in Sec. **??**, Sec. **??** provides analytical position control results of stable configurations in two canonical workspaces with frictionless walls. These results are limited in the set of shapes that can be generated. To extend the range of possible shapes, Sec. **??** introduces general models for boundary interaction. We prove that two orthogonal boundaries with high friction are sufficient to arbitrarily position two robots in Sec. **??**, and Sec. **??** extends this to prove a rectangular workspace with high-friction boundaries can position a swarm of n robots arbitrarily within a subset of the workspace. Both algorithms require robots with low process noise and require time proportional to the number of robots, so Sec. **??** introduces a robust and efficient technique to control swarm correlation. Sec. **??** describes implementations of the algorithms in simulation and Sec. **??** describes hardware experiments with up to 100 robots, as shown in Fig. **??**. We end with directions for future research in Sec. **??**.

II. Related Work

Controlling the *shape*, or relative positions, of a swarm of robots is a key ability for a range of applications. Correspondingly, it has been studied from a control-theoretic perspective in both centralized and decentralized approaches. For examples of each, see the centralized virtual leaders in [**?**], and the gradient-based decentralized controllers using control-Lyapunov functions in [**?**]. However, these approaches assume a level of intelligence and autonomy in individual robots that exceeds the capabilities of many systems, including current micro- and nano-robots. Current micro- and nano-robots, such as those in [**? ? ?**] lack onboard computation.

Instead, this paper focuses on centralized techniques that apply the same control input to each member of the swarm. Precision control requires breaking the symmetry caused by the global input. Symmetry can be broken using agents that respond differently to the global control, either through agent-agent reactions, see work modeling biological swarms [**?**], or engineered inhomogeneity [**? ? ?**]. This work assumes a uniform control (**??**) with homogenous agents, as in [**?**]. The techniques in this paper are inspired by fluid-flow techniques and artificial force-fields.

Fluid-flow: Fluid flow along boundaries generates a shear force that pushes different parts of a body in opposing directions. Most introductory fluid dynamics textbooks provide models [**?**]. Similarly, a swarm of robots under global control pushed along a boundary will experience shear forces. This is a position-dependent force, and so can be exploited to control the configuration or shape of the swarm. [**?**] used these forces to disperse a swarm's spatial position for coverage for physics-based swarm simulations.

Artificial Force-fields: Much research has focused on generating non-uniform artificial force-fields that can be used to rearrange passive components. Applications have included techniques to design shear forces for sensorless manipulation of a single object by [**?**]. [**? ?**] demonstrated a collection of 2D force fields generated by 6DOF vibration inputs to a rigid plate. These force fields, including shear forces, could be used as a set of primitives for motion control to steer the formation of multiple objects. However unlike the uniform control model in this paper, theirs was multi-modal and position-dependent.

III. Theory

A. Using Boundaries: stable configurations of a swarm

One method to control a swarm's shape in a bounded workspace is to simply push in a given direction until the swarm conforms to the boundary. Like fluid settling in a tank, the stable final configuration minimizes potential energy.

a) Square workspace: We first examine the mean $(\bar{x}, \bar{y})$, covariance $(\sigma_x^2, \sigma_y^2, \sigma_{xy})$, and correlation ρ_{xy} of a large swarm of robots as they move inside a square workspace under the influence of a force pulling in the direction β. The swarm is large, but the robots are small in comparison, and together occupy a constant area A, $A \in [0, 1]$. Under a global input, the swarm moves to a side of the workspace and forms a polygonal shape that minimizes potential energy, as shown in Fig. **??**, see also [**?**].

The range for the global input angle β is [0,2π). In this range, the swarm assumes eight different polygonal shapes. These shapes alternate between triangles and trapezoids when the area A<1/2, and between squares with one corner removed and trapezoids when A>1/2.

Computing means, variances, covariance, and correlation requires integrating over R, the region containing the swarm:

$$\bar{x} = \frac{\iint_R x\, dx\, dy}{A}, \qquad \bar{y} = \frac{\iint_R y\, dx\, dy}{A} \tag{2}$$

$$\sigma_x^2 = \frac{\iint_R (x-\bar{x})^2\, dx\, dy}{A},\ \sigma_y^2 = \frac{\iint_R (y-\bar{y})^2\, dx\, dy}{A} \tag{3}$$

$$\sigma_{xy} = \frac{\iint_R (x-\bar{x})\,(y-\bar{y})\, dx\, dy}{A},\ \rho_{xy} = \frac{\sigma_x^2}{\sigma_x \sigma_y} \tag{4}$$

The region of integration R is the polygon containing the swarm. For example, if $A < 1/2$ and the force angle is β, the mean when R is a triangular region in the lower-left corner is:

$$\begin{aligned}\bar{x}(A,\beta) &= \frac{\int_0^{\sqrt{2A\tan(\beta)}} \left(\int_0^{\cot(\beta)\left(\sqrt{2A\tan(\beta)}-x\right)} dy \right) x\, dx}{A} \\ &= \frac{\sqrt{2}}{3}\sqrt{A\tan(\beta)} \end{aligned} \tag{5}$$

$$\begin{aligned}\bar{y}(A,\beta) &= \frac{\int_0^{\sqrt{2A\tan(\beta)}} \left(\int_0^{\cot(\beta)\left(\sqrt{2A\tan(\beta)}-x\right)} y\, dy \right) dx}{A} \\ &= \frac{\sqrt{2}}{3}\sqrt{A\cot(\beta)} \end{aligned} \tag{6}$$

The full equations are included in the appendix [**?**], and are summarized in Fig. **??**. A few highlights are that the correlation is maximized $\pm$1/2 when the swarm is in a triangular shape. The covariance of a triangle is always $\pm(A/18)$. Variance is minimized in the direction of β and maximized orthogonal to β when the swarm is in a rectangular shape. The range of mean positions are maximized when A is small.

b) Circular workspace: Though rectangular boundaries are common in artificial workspaces, biological workspaces are usually rounded. Similar calculations can be made for a circular workspace. The workspace is a circle centered at (0,0) with radius 1 and thus area π. For notational simplicity, the swarm is parameterized by the global control input signal β and the fill-level h. Under a global input, the robot swarm fills the region under a chord with area

$$A(h) = \arccos(1-h) - (1-h)\sqrt{(2-h)h}. \tag{7}$$

For a circular workspace, the locus of mean positions are aligned with β and the mean position is at radius $r(h)$ from the center:

$$r(h) = \frac{2(-(h-2)h)^{3/2}}{3\left(\sqrt{-(h-2)h}(h-1) + \arccos(1-h)\right)} \tag{8}$$

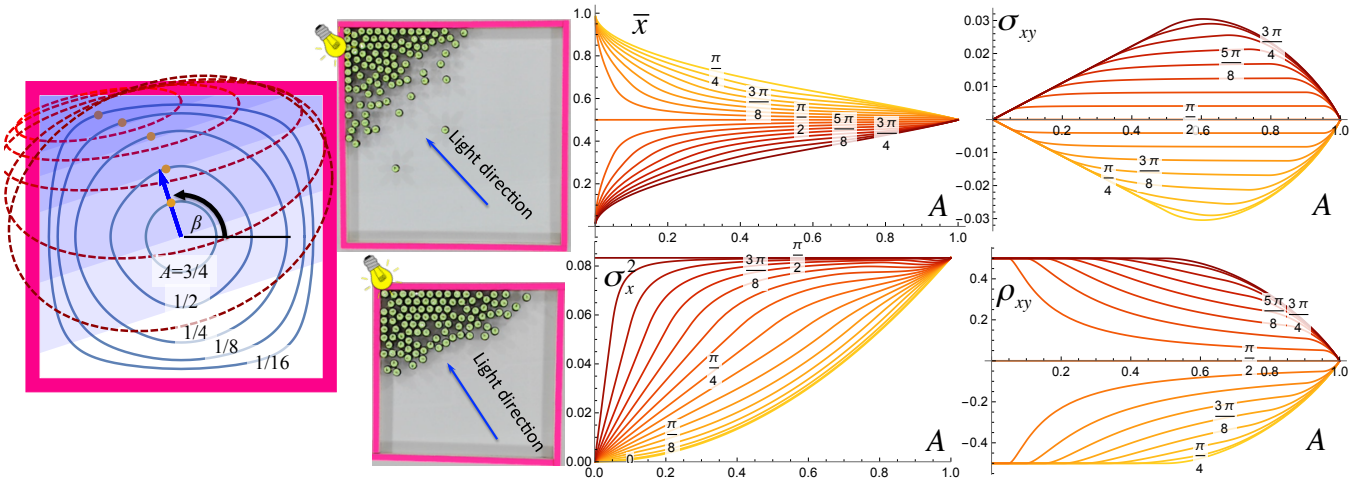

Fig. 2. Pushing the swarm against a square boundary wall allows limited control of the shape of the swarm, as a function of swarm area A and the commanded movement direction β. Left plot shows locus of possible mean positions for five values of A. The locus morphs from a square to a circle as A increases. The covariance ellipse for each A is shown with a dashed line. Center shows two corresponding arrangements of kilobots. At right is $\bar{x}(A), \sigma_{xy}(A), \sigma_x^2(A)$, and $\rho(A)$ for a range of β values. See online interactive demonstration at [**?**].

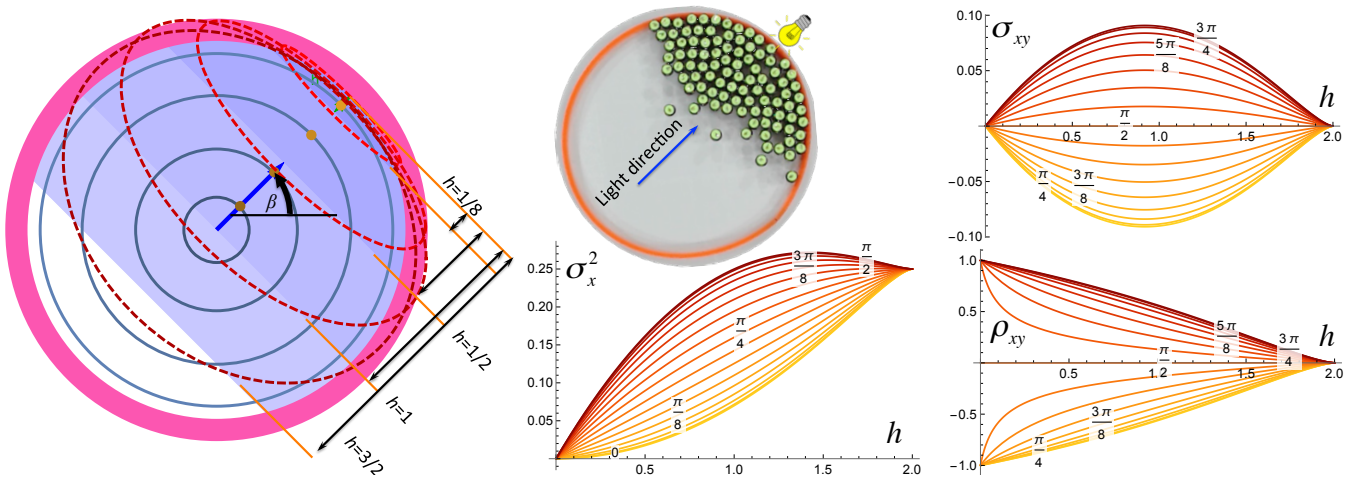

Fig. 3. Pushing the swarm against a circular boundary wall allows limited control of the shape of the swarm, as a function of the fill level h and the commanded movement direction β. Left plot shows locus of possible mean positions for four values of h. The locus of possible mean positions are concentric circles. See online interactive demonstration at [**?**].

Variance $\sigma_x^2(\beta, h)$ is maximized at $\beta = \pi/2 + n\pi$ and $h \approx 1.43$, while covariance is maximized at $\beta = \pi 3/4 + n\pi$ and $h \approx 0.92$. For small h values, correlation approaches ± 1. Results are displayed in Fig. **??**, see also [**?**].

B. Using Boundaries: Friction and Boundary Layers

Global inputs move a swarm uniformly. Shape control requires breaking this uniform symmetry. A swarm inside an axis-aligned rectangular workspace can reduce variance normal to a wall by simply pushing the swarm into the boundary. If the swarm can flow around each other, pushing the swarm into a boundary produces the limited set of configurations presented in Sec. **??**. Instead of pushing our robots directly into a wall, the following sections examine an oblique approach using boundaries that generate friction with the robots. These frictional forces are sufficient to break the symmetry caused by uniform inputs. Robots touching a wall have a friction force that opposes movement along the boundary. This causes robots along the boundary to move more slowly than robots in free-space.

Let the control input be a vector force $\vec{F}$ with magnitude F and orientation θ with respect to a line perpendicular to and into the nearest boundary. N is the normal or perpendicular force between the robot and the boundary. The force of friction F_f is nonzero if the robot is in contact with the boundary and $\sin(\theta) < 0$. The resulting net force on the robot, $F_{forward}$, is aligned with the wall and given by

$$\begin{aligned} F_{forward} &= F\sin(\theta) - F_f \\ \text{where } F_f &= \begin{cases} \mu_f N, & \mu_f N < F\sin(\theta) \\ F\sin(\theta), & \text{else} \end{cases} \\ \text{and } N &= F\cos(\theta) \end{aligned} \tag{9}$$

Fig. **??** shows the resultant forces on two robots when one is touching a wall. Though each receives the same inputs, they experience different net forces. For ease of analysis,

the following algorithms assume μ_f is infinite and robots touching the wall are prevented from sliding along the wall. This means that if one robot is touching the wall and another robot is free, the touching robot will not move when the control input is parallel or into the wall. There are many alternate models of friction that also break control symmetry. Fig. **??**c shows fluid flow along a boundary. Fluid in the free-flow region moves uniformly, but flow decreases to zero in the boundary layer [**?**].

$$F_{forward}(y) = F - F_f \begin{cases} \frac{h-y}{h}, & y < h \\ 0, & \text{else} \end{cases} \tag{10}$$

The next section shows how a system in a rectangularly bounded workspace with friction model (**??**) can arbitrarily position two robots.

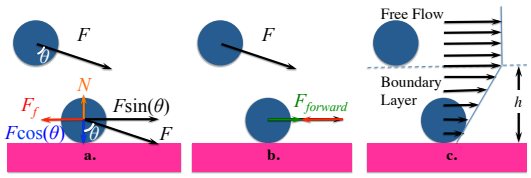

Fig. 4. (a,b) Wall friction reduces the force for going forward $F_{forward}$ on a robot near a wall, but not for a free robot. (c) velocity of a fluid reduces to zero at the boundary.

IV. Algorithms

A. Position Control of Two Robots Using Wall Friction

Alg. **??** uses wall-friction to arbitrarily position two robots in a rectangular workspace. This algorithm introduces concepts that will be used for multi-robot positioning. Fig. **??** shows a Mathematica implementation of the algorithm, and is useful as a visual reference for the following description.

Assume two robots are initialized at s_1 and s_2 with corresponding goal destinations e_1 and e_2. We can exploit symmetry in the solution by labeling the leftmost (or, if they have the same x coordinate, the topmost) robot s_1. If s_1 is not also the topmost robot, we rotate the coordinate frame by 90°. Denote the current positions of the robots r_1 and r_2. Values $.x$ and $.y$ denote the x and y coordinates, i.e., $s_1.x$ and $s_1.y$ denote the x and y locations of s_1. Define the sign function as:

$$\text{sgn}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} \tag{11}$$

The algorithm assigns a global control input at every instance. The goal is to adjust $\Delta r_x = r_2.x - r_1.x$ from $\Delta s_x = s_2.x - s_1.x$ to $\Delta e_x = e_2.x - e_1.x$ and adjust $\Delta r_y = r_2.y - r_1.y$ from $\Delta s_y = s_2.y - s_1.y$ to $\Delta e_y = e_2.y - e_1.y$ using a shared global control input. This algorithm exploits the position-dependent friction model (**??**).

Our algorithm solves the positioning problem in four steps: First, it adjusts $\Delta r_y, \Delta r_x$ as much as possible with the left wall. Second, $\Delta r_x - \Delta e_x$ is reduced to zero with the bottom wall. Third, if the robots were not correctly positioned relative to each other, $\Delta r_y - \Delta e_y$ is reduced to zero with the right wall. Lastly, the robots, now correctly positioned relative to each other, are moved to their goal locations.

In the worse case, adjusting both Δr_x and Δr_y needs all four steps. The worst case path length is $2(\sqrt{2}+1)L$.

Algorithm 1 GenerateDesiredSpacing(s_1, s_2, e_1, e_2, L)

Require: knowledge of starting (s_1, s_2) and ending (e_1, e_2) positions of two robots. $(0,0)$ is bottom corner, s_1 is leftmost robot, L is length of the walls. Current robot positions are (r_1, r_2). Assume $s_1.x < s_2.x$ and $s_1.y \geq s_2.y$. If not, rotate workspace coordinates 90°. ϵ is a small, nonzero, user-specified value.
Ensure: $(e_1, e_2), (s_1, s_2)$ all at least ϵ distance from walls.

$\Delta e_y = e_2.y - e_1.y$
$\Delta e_x = e_2.x - e_1.x$
Move $(-r_1.x, -\min(r_2.y, r_1.y + \Delta e_y) + \epsilon)$ ▷ Touch left wall
Move $(r_2.x, |\text{sgn}(\Delta e_y)| \min(r_1.y - r_2.y + \Delta e_y - \epsilon, L - r_2.y - \epsilon) + \epsilon)$ ▷ Adjust y
Move $(\max(\epsilon, -\text{sgn}(r_2.y - r_1.y)\Delta e_x), -\min(r_1.y, r_2.y))$ ▷ Touch bottom wall
Move $(\text{sgn}(r_2.y - r_1.y)\Delta e_x, (|\text{sgn}(\Delta e_y)|-1)|r_2.y - r_1.y|)$ ▷ Adjust x
if $\Delta e_y \neq r_2.y - r_1.y$ **then**
  if $\Delta e_x \neq 0$ **then**
    Move $(\text{sgn}(r_2.y - r_1.y)\Delta e_x, |\text{sgn}(\Delta e_y) - 1||r_1.y - r_2.y|)$ ▷ Touch right wall
  else
    Move $(|\text{sgn}(\Delta e_y - r_2.y + r_1.y)|\epsilon, |\text{sgn}(\Delta e_y) - 1||r_1.y - r_2.y|)$ ▷ Touch right wall
  end if
  if $\Delta e_x < 0$ **then**
    Move $(L - \max(r_1.x, r_2.x), \epsilon)$
  else
    Move $(L - \max(r_1.x, r_2.x), \Delta e_y - \max(r_1.y, r_2.y))$ ▷ Adjust y
  end if
end if
Move $(e_2.x - r_2.x, e_2.y - r_2.y)$ ▷ Go to goal

B. Position Control of n Robots Using Wall Friction

Alg. **??** can be extended to control the position of n robots using wall friction under several constraints. The solution described here is an iterative procedure with n loops. The kth loop moves the kth robot from a *staging zone* to the desired position in a *build zone*. All robots move according to the global input, but due to wall friction, at the end the kth loop, robots 1 through k are in their desired final configuration in the build zone, and robots $k+1$ to n are in the staging zone. See Fig. **??** for a schematic of the build and staging zones.

Assume an open workspace with four axis-aligned walls with infinite friction. The axis-aligned build zone of dimension (w_b, h_b) containing the final configuration of n robots must be disjoint from the axis-aligned staging zone of dimension (w_s, h_s) containing the starting configuration of n robots. Without loss of generality, assume the build zone is above the staging zone. Furthermore, there must be at least ϵ space above the build zone, ϵ below the staging zone, and $\epsilon + 2r$ to the left of the build and staging zone, where r is the radius of a robot. The minimum workspace is then $(\epsilon + 2r + \max(w_b, w_s), 2\epsilon + h_s, h_b)$.

The n robot position control algorithm relies on a DriftMove$(\alpha, \beta, \epsilon)$ control input, shown in Fig. **??**. A drift move consists of repeating a triangular movement sequence $\{(\beta/2, -\epsilon), (\beta/2, \epsilon), (-\alpha, 0)\}$. The robot touching a top wall moves right β units, while robots not touching the top move right $\beta - \alpha$.

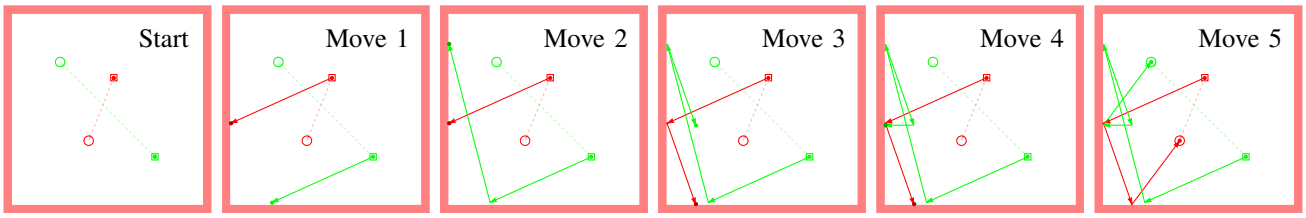

Fig. 5. Frames from an implementation of Alg. **??**: two robot positioning using walls with infinite friction. The algorithm only requires friction along the bottom and right walls. Robot initial positions are shown by a square, and final positions by a circle. Dashed lines show the shortest route if robots could be controlled independently. Solid arrows show path given by Alg. **??**. Online demonstration and source code at [**?**].

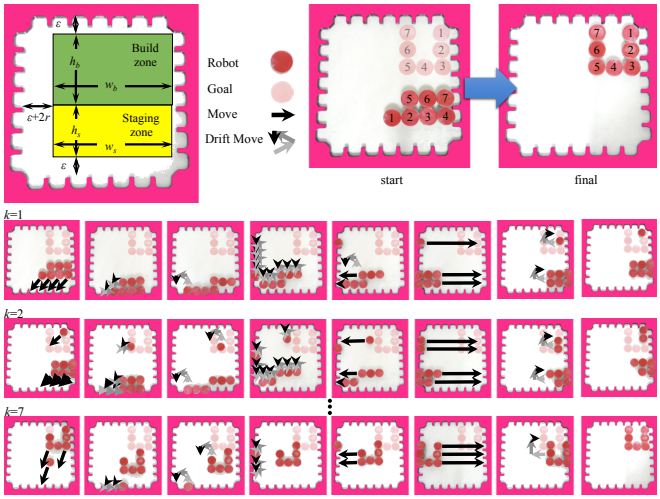

Fig. 6. Illustration of Alg. **??**, n robot position control using wall friction.

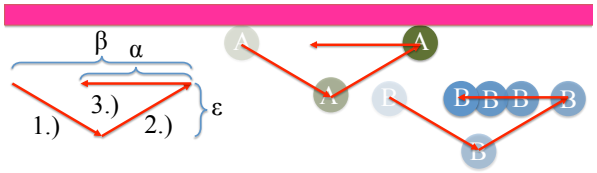

Fig. 7. A DriftMove$(\alpha, \beta, \epsilon)$ to the right repeats a triangular movement sequence $\{(\beta/2, -\epsilon), (\beta/2, \epsilon), (-\alpha, 0)\}$. Robot A touching a top wall moves right β units, while robots not touching the top move right $\beta - \alpha$.

Let $(0, 0)$ be the lower left corner of the workspace, p_k the x, y position of the kth robot, and f_k the final x, y position of the kth robot. Label the robots in the staging zone from left-to-right and bottom-to-top, and the f_k configurations top-to-bottom and right-to-left as shown in Fig. **??**.

Alg. **??** proceeds as follows: First, the robots are moved left away from the right wall, and down so robot k touches the bottom wall. Second, a set of DriftMoves are executed that move robot k to the left wall with no net movement of the other robots. Third, a set of DriftMoves are executed that move robot k to its target height and return the other robots to their initial heights. Fourth, all robots except robot k are pushed left until robot k is in the correct relative x position compared to robots 1 to $k-1$. Finally, all robots are moved right until robot k is in the desired target position. Running time is $O(n(w + h))$.

Algorithm 2 PositionControl$n$RobotsUsingWallFriction(k)

move($-\epsilon, r - p_{ky}$)
while $p_{kx} > r$ **do**
    DriftMove$(\epsilon, \min(p_{kx} - r, \epsilon), \epsilon)$ left
end while
$m \leftarrow \text{ceil}(\frac{f_{ky}-r}{\epsilon})$
$\beta \leftarrow \frac{f_{ky}-r}{m}$
$\alpha \leftarrow \beta - \frac{r-p_{ky}-\epsilon}{m}$
for m iterations **do**
    DriftMove$(\alpha, \beta, \epsilon)$ up
end for
Move $(r + \epsilon - f_{kx}, 0)$
Move $(f_{kx} - r, 0)$

C. Maximizing Correlation Using Wall Friction

Using the environment to individually move n particles to goal positions in Alg. **??** requires $O(n^2)$ time, while Alg. **??** can only control two particles. The controllers in Section **??** are efficient because they require only a single move but the range of possible configurations are limited. This section presents an efficient technique to generate desired correlations.

Assume an obstacle-free, bounded, unit-size, square workspace. As shown in Fig. **??**, the maximum correlation occurs when the swarm is pushed in the direction $\beta = 3\pi/4$. This correlation as a function of swarm area A is never larger than 1/2, and the maximum correlation decays to 0 as A grows to 1. By (**??**), this maximum correlation is:

$$\rho_{xy} = \begin{cases} \frac{1}{2}, & 0 \le A \le \frac{1}{2} \\ \frac{3A(2(A-2)A+1)}{4A^3-24A+\sqrt{2}(12A-12)\sqrt{1-A}+17} - 1, & \frac{1}{2} \le A \le 1 \end{cases} \quad (12)$$

If friction obeys the linear boundary layer model of (**??**) with boundary layer thickness h and maximum friction F_f equal to the maximum applied force F, we can generate larger correlations. If the swarm size is smaller than ≈ 0.43 and the boundary layer is sufficiently thick we can generate correlations larger than 1/2 using boundary friction.

Assume that the swarm is initialized in the lower-left corner, in a rectangle of width w and height A/w. Such a rectangular configuration can be accomplished using the variance controllers from [**?**]. If the swarm is then commanded to move a distance L to the right, components of the swarm outside the boundary friction layer of height h move

further than components near the boundary. The swarm is contained in a region R composed of no more than three stacked components: at bottom a parallelogram inclined to the right top, at middle a rectangle, and at top a parallelogram inclined to the left top. These regions can be defined by the rectangle's left side, bottom, and top:

$$\begin{aligned} r_{\text{left}} &= \min(L, 1-w) \\ r_{\text{bottom}} &= \min\left(\frac{A}{w}, h\frac{r_{left}}{L}\right) \\ r_{\text{top}} &= \min\left(\frac{A}{w}, 1-h\frac{r_{left}}{L}\right) \end{aligned} \tag{13}$$

If $\frac{A}{w} \leq r_{\text{top}}$ the top parallelogram has no area. Similarly, if $r_{\text{top}} \leq r_{\text{bottom}}$ the rectangle has no area. The mean, variance, and correlation are calculated using (**??**), (**??**), and (**??**) over the region R:

$$\begin{aligned} \iint_R f(x,y)\,dx\,dy = &\int_0^{r_{\text{bottom}}} \int_{\frac{L}{h}y}^{\frac{L}{h}y+w} f(x,y)dx\,dy \\ &+ \int_{r_{\text{bottom}}}^{r_{\text{top}}} \int_{r_{\text{left}}}^{r_{\text{left}}+w} f(x,y)dx\,dy \\ &+ \int_{r_{\text{top}}}^{\frac{A}{w}} \int_{-\frac{L(y-r_{\text{top}})}{h}+r_{\text{left}}}^{r_{\text{left}}+w-\frac{L(y-r_{\text{top}})}{h}} f(x,y)dx\,dy \end{aligned} \tag{14}$$

Given an environment parameterized by A and h, efficient correlation control consists of choosing the w, L pair that generates the desired positive correlation. Negative correlations can be generated by initializing the swarm in the upper left, or lower right.

V. Simulation

Two simulations were implemented using wall-friction for position control. The first controls the position of two robots, the second controls the position of n robots.

Two additional simulations were performed using wall-friction to control variance and covariance. The first is an open-loop algorithm that demonstrates the effect of varying friction levels. The second uses a closed-loop controller to achieve desired variance and covariance values.

A. Position Control of Two Robots

Algorithm **??** was implemented in Mathematica using point robots (radius = 0). An online interactive demonstration and source code of the algorithm are available at [**?**]. Fig. **??** shows an implementation of this algorithm wiht robot initial positions shown by hollow squares and final positions by circles. Dashed lines show the shortest route if robots could be controlled independently, solid lines the generated path.

B. Position Control of n Robots

Alg. **??** was simulated in MATLAB using square block robots with unity width. Code is available at [**?**]. Simulation results are shown in Fig. **??** for arrangements with an increasing number of robots, n= [8, 46, 130, 390, 862]. The distance moved grows quadratically with the number of robots n. A best-fit line $210n^2 + 1200n - 10,000$ is overlaid by the data.

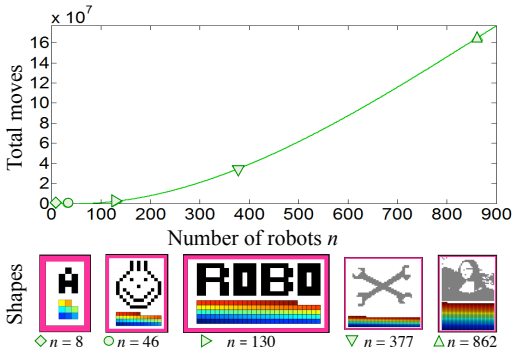

Fig. 8. The required number of moves under Alg. **??** using wall-friction to rearrange n square-shaped robots. See hardware implementation and simulation at [**?**].

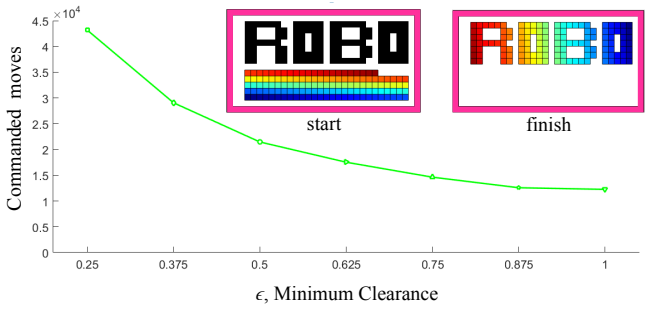

Fig. 9. Control performance is sensitive to the desired clearance ϵ. As ϵ increases, the total distance decreases asymptotically.

In Fig. **??**, the amount of clearance is $\epsilon = 1$. Control performance is sensitive to the desired clearance. As ϵ increases, the total distance decreases asymptotically, as shown in Fig. **??**, because the robots have more room to maneuver and fewer DriftMoves are required.

C. Efficient Control of Correlation

This section examines maximum correlation values as a function of w, L using (**??**) from Section **??**. The maximum correlation using boundary layer friction $\max_{w,L}(\rho(A,h,w,L))$ can be found by gradient descent, as shown in Fig. **??**. For swarms with small area, this method enables generating the full range of correlations ± 1, while the stable configuration method from Section **??** could only generate correlations $\pm 1/2$. As the swarm area A increases above ≈ 0.43, the stable configuration method is more effective. Larger boundary layers h enable more control of correlation. The multimedia attachment shows efficient control of correlation with simulations using the 2D physics engine Box2D [**?**] and 144 disc-shaped robots with boundary layer model (**??**).

VI. Experiment

Our experiments are on centimeter-scale hardware systems called *kilobots*. These allows us to emulate a variety of dynamics, while enabling a high degree of control over robot function, the environment, and data collection. The kilobot, from [**? ?**] is a low-cost robot designed for testing collective

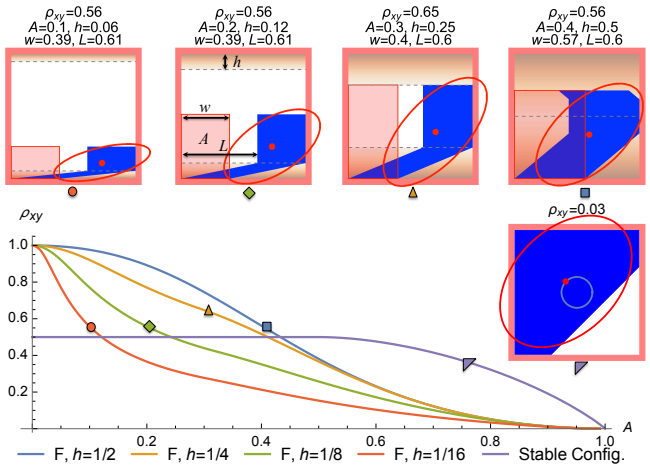

Fig. 10. Analytical results comparing maximum correlation ρ_{xy} under the boundary layer friction model of (**??**) with four boundary layer thicknesses h and the stable triangular configuration (**??**).

algorithms with large numbers of robots. It is available as an open source platform or commercially from [**?**]. Each robot is approximately 3 cm in diameter, 3 cm tall, and uses two vibration motors to move on a flat surface at speeds up to 1 cm/s. Each robot has one ambient light sensor that is used to implement *phototaxis*, moving towards a light source. In these experiments as shown in Fig. **??**, we used n=100 kilobots, a 1 m×1 m whiteboard as the workspace, four 30W and four 50W LED floodlights arranged 1.5 m above the plane of the table at the midpoints and vertices of a 6 m square centered on the workspace. An Arduino Uno connected to an 8-relay shield controlled the lights. Above the table, an overhead machine vision system tracked the position of the swarm.

A. Hardware Experiment: Position Control of Two Robots

The hardware platform walls have almost infinite friction, due to a laser-cut, scalloped border. When a kilobot is steered into the scalloped border, they are pinned to the wall unless the global input directs them away from the wall. Even though the kilobots are stochastic, this wall friction is sufficient to enable independent control of two kilobots, as shown in Fig. **??**.

B. Hardware Experiment: Position Control of n Robots

Kilobots have too much stochasticity to implement Alg. **??**. Instead, a hardware setup with a bounded platform, magnetic sliders, and a magnetic guide board was used. Designs for each are available at [**?**]. The pink boundary is toothed with a white free space, as shown in Fig **??**. Only discrete, 1 cm moves in the x and y directions are used. The goal configuration highlighted in the top right corner represents a 'U' made of seven sliders. The dark red configuration is the current position of the sliders. Due to the discretized movements allowed by the boundary, drift moves follow a 1 cm square. Free robots return to their start positions but robots on the boundary to move laterally, generating a net sliding motion.

Fig. **??** follows the motion of the sliders through iterations k=1, 2, and 7. All robots receive the same control inputs, but boundary interactions break the control symmetry. Robots reach their goal positions in a first-in, first-out arrangement beginning with the bottom-left robot from the staging zone occupying the top-right position of the build zone.

C. Hardware Experiment: Control of Covariance

To demonstrate covariance control, up to 100 kilobots were placed on the workspace and manually steered with lights, using friction with the boundary walls to vary the covariance from -4000 to 3000 cm^2. The resulting covariance is plotted in Fig. **??**, along with snapshots of the swarm.

VII. Conclusion and Future Work

This paper presented techniques for controlling the shape of a swarm of robots using uniform global inputs and interaction with boundary friction forces. The paper provided algorithms for precise position control, as well as robust and efficient covariance control. Extending algorithms **??** and **??** to 3D is straightforward but increases the complexity. Future efforts should be directed toward improving the technology and tailoring it to specific robot applications.

With regard to technological advances, this includes designing controllers that efficiently regulate σ_{xy}, perhaps using Lyapunov-inspired controllers as in [**?**]. Additionally, this paper assumed nearly infinite wall friction. The algorithms require retooling to handle small μ_f friction coefficients.

Acknowledgments

We thank Haoran Zhao, Jarrett Lonsford, An Nguyen, and Lillian Lin for help in making structures for the experiments.

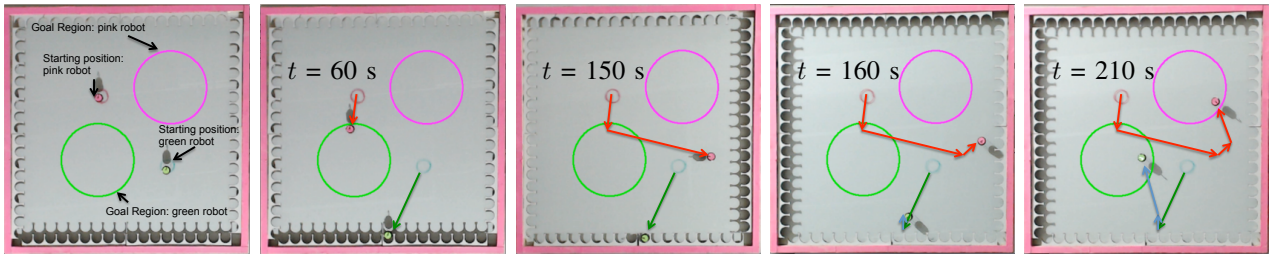

Fig. 11. Position control of two kilobots (Alg. **??**) steered to corresponding colored circle. Boundary walls have nearly infinite friction, so the green robot is stopped by the wall from $t = 60$s until the commanded input is directed away form the wall at $t = 150$s, while the pink robot in free-space is unhindered.

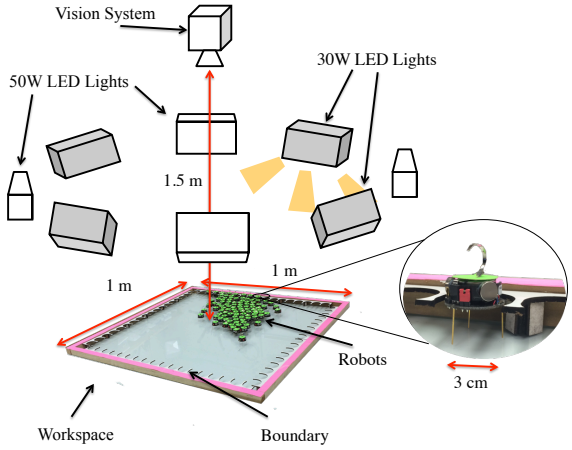

Fig. 12. Hardware platform: table with 1×1 m workspace, surrounded by eight triggered LED floodlights, with an overhead machine vision system.

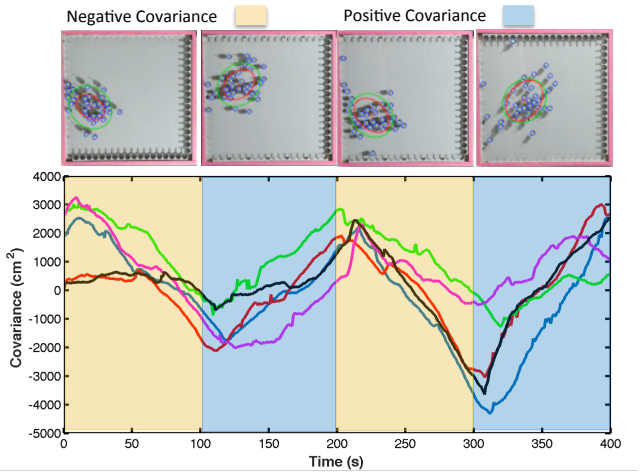

Fig. 13. Hardware demonstration steering ≈ 50 kilobot robots to desired covariance. The goal covariance is negative in first 100 seconds and is positive in the next 100 seconds. The actual covariance is shown in different trials. Frames above the plot show output from machine vision system and an overlaid covariance ellipse.