Shaping a Swarm With a Shared Control Input Using Boundary Walls and Wall Friction

Paper-ID 66

***Abstract*—Micro- and nano-robots hold promise for targeted drug delivery and micro-scale manufacturing. Due to their small size, large numbers of micro-robots are required to deliver sufficient payloads, but their small size makes it difficult to perform onboard computation or contain a power and propulsion source. For this reason these robots are usually powered and controlled by global inputs, such as a uniform external electric or magnetic field, and every robot receives exactly the same control inputs. Nevertheless, these applications require precision control of the shape and position of the robot swarm. Precision control requires breaking the symmetry caused by the global input.**

A promising technique uses collisions with boundary walls to shape the swarm, however, the range of configurations created by conforming a swarm to a boundary wall is limited. This paper describes the set of stable configurations of a swarm in two canonical workspaces, a circle and a square.

To increase the diversity of configurations, we add wall friction 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 robots at desired locations. We conclude with efficient techniques to control the covariance of a swarm not possible without wall-friction. Simulations and hardware implementations with 97 robots validate these results.

I. Introduction

Micro- and nano-robots can be manufactured in large numbers, see recent examples in Donald et al. [3], Ghosh and Fischer [5], Kim et al. [8], Qiu and Nelson [12], or Martel et al. [10]. Our vision is for large swarms of robots remotely guided 1) through the human body, to cure disease, heal tissue, and prevent infection and 2) ex vivo to assemble structures in parallel. For each application, 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. The biggest barrier to this vision is a lack of control techniques that can reliably exploit large populations despite high under-actuation.

Even without obstacles or boundaries, the mean position of a swarm 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. A limitation is that global control can only compress a swarm orthogonal to obstacles. One implication is that a swarm in an axis-aligned rectangular workspace can not generate a non-zero covariance. This limitation is detrimental to desired applications because the ability to orient the swarm is often useful for navigating narrow passages.

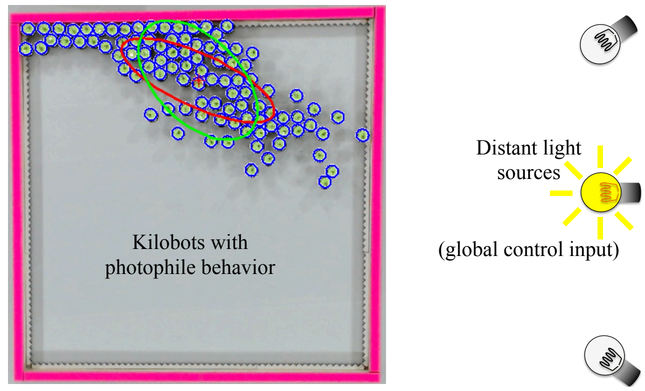

Fig. 1. Swarm of kilobots programmed to move toward the brightest light source. The current covariance ellipse and mean are shown in red, the desired covariance is shown in green. Navigating a swarm using global inputs is challenging because each member receives the same control inputs. This paper focuses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

After a review of recent related work (Section II), this paper provides analytical position control results in two canonical workspaces with frictionless walls (Section III-A). These results are limited in the set of shapes that can be generated. To extend the range of possible shapes, we introduce wall friction to the system model (Section III-B). We prove that two orthogonal boundaries with high friction are sufficient to arbitrarily position two robots (Section III-C), and extend this proof to prove that a rectangular workspace with high friction boundaries is sufficient to position a swarm of n robots arbitrarily within a subset of the workspace (Section III-D). (Section IV) describes implementations of both position control algorithms in simulation and (Section V) describes experiments with a hardware setup and up to 97 robots, as shown in Fig. 1. We end with directions for future research (Section VI).

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 Egerstedt and Hu [4], and the gradient-based decentralized controllers using control-Lyapunov functions in Hsieh et al. [6]. However, these approaches assume a level of intelligence and autonomy in the individual robots that exceeds the capabilities of current micro- and nano-robots. Current micro- and nano-robots, such as those in Martel [9], Yan et al. [19] and Chowdhury et al. [2], 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. The techniques in this paper are inspired by fluid-flow techniques and artificial force-fields.

a) Shear forces: are unaligned forces that push one part of a body in one direction, and another part of the body in the opposite direction. These shear forces are common in fluid flow along boundaries. Most introductory fluid dynamics textbooks provide models, for example, see Munson et al. [11]. 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. Physics-based swarm simulations have used these forces to disperse a swarm's spatial position for accomplishing coverage tasks as in Spears et al. [15].

b) Artificial Force-fields: Much research has focused on generating 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 Sudsang and Kavraki [16]. Vose et al. [17, 18] 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.

III. Theory

A. Controlling Covariance: Fluid Settling In a Tank

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.

a) Square workplace: This section examines the mean, variance, covariance and correlation of a very large swarm of robots as they move inside a square workplace under the influence of gravity pointing in the direction β. The swarm is large, but the robots are small in comparison, and together cover an area of constant volume A. Under a global input such as gravity, they flow like water, moving to a side of the workplace and forming a polygonal shape, as shown in Fig. 2.

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

Computing the means, variances, covariance, and correlation requires integrating over the area containing the swarm. This is simplified using an indicator function $\mathbf{1}_A(x,y)$ that returns 1 if inside the region containing the swarm, and 0 else. The formulas for means $(\bar{x},\bar{y})$, covariance $(\sigma_x^2,\sigma_y^2,\sigma_{xy})$, and correlation ρ_{xy} are as follows, integrated over the unit square with x and y from 0 to 1:

$$\bar{x} = \frac{\iint x\mathbf{1}_A(x,y)dxdy}{A},\ \bar{y} = \frac{\iint y\mathbf{1}_A(x,y)dxdy}{A} \tag{1}$$

$$\sigma_x^2 = \frac{\iint (x-\bar{x})^2\,\mathbf{1}_A(x,y)dxdy}{A}, \tag{2}$$

$$\sigma_y^2 = \frac{\iint (y-\bar{y})^2\,\mathbf{1}_A(x,y)dxdy}{A} \tag{3}$$

$$\sigma_{xy} = \frac{\iint (x-\bar{x}_x)(y-\bar{y})\,\mathbf{1}_A(x,y)dxdy}{A}, \tag{4}$$

$$\rho_{xy} = \frac{\sigma_x^2}{\sigma_x\sigma_y} \tag{5}$$

Rather than using an indicator function, the region of integration can be changed to only include the polygon containing the swarm. If the force angle is β, the mean when the swarm is in the lower-left corner is:

$$\begin{aligned}\bar{x}(A,\beta) &= \frac{\int_0^{\sqrt{2}\sqrt{-A\tan(\beta)}}\left(\int_0^{\sqrt{2}\sqrt{-A\cot(\beta)}+x\cot(\beta)} x\,dy\right)dx}{A}\\ &= \frac{1}{3}\sqrt{2}\sqrt{A\tan(\beta)}\end{aligned} \tag{6}$$

$$\begin{aligned}\bar{y}(A,\beta) &= \frac{\int_0^{\sqrt{2}\sqrt{-A\tan(\beta)}}\left(\int_0^{\sqrt{2}\sqrt{-A\cot(\beta)}+x\cot(\beta)} y\,dy\right)dx}{A}\\ &= \frac{1}{3}\sqrt{2}\sqrt{A\cot(\beta)}\end{aligned} \tag{7}$$

The full equations are included in the appendix, and are summarized in Fig. 2. A few highlights are that the correlation is maximized when the swarm is in a triangular shape, and is $\pm$1/2. 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 workplace: Though rectangular boundaries are common in artificial workspaces, biological workspaces are usually rounded. Similar calculations can be computed 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

$$A(h) = \arccos(1-h) - (1-h)\sqrt{(2-h)h}. \tag{8}$$

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:

$$r(h) = \frac{2(-(h-2)h)^{3/2}}{3\left(\sqrt{-(h-2)h}(h-1)+\arccos(1-h)\right)} \tag{9}$$

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 summarized in Fig. 3.

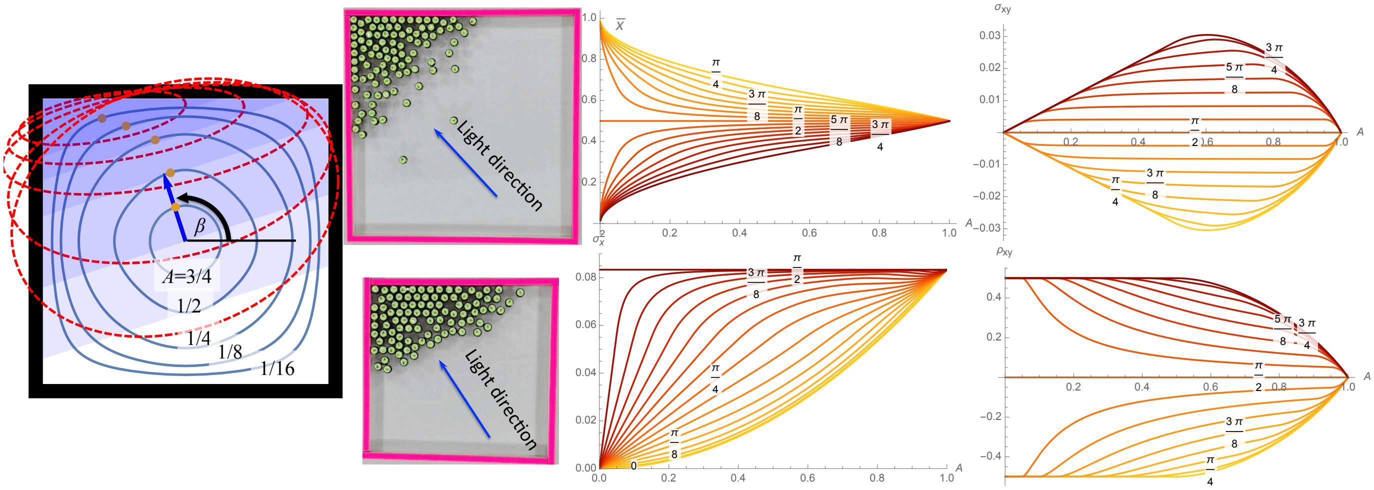

Fig. 2. Pushing the swarm against a square boundary wall allows limited control of the shape of the swarm, as a function of swarm area A and the commanded movement direction β. Left plot shows locus of possible mean positions for five values of A. The locus morphs from a square to a circle as A increases. 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 [withheld for double-blind review].

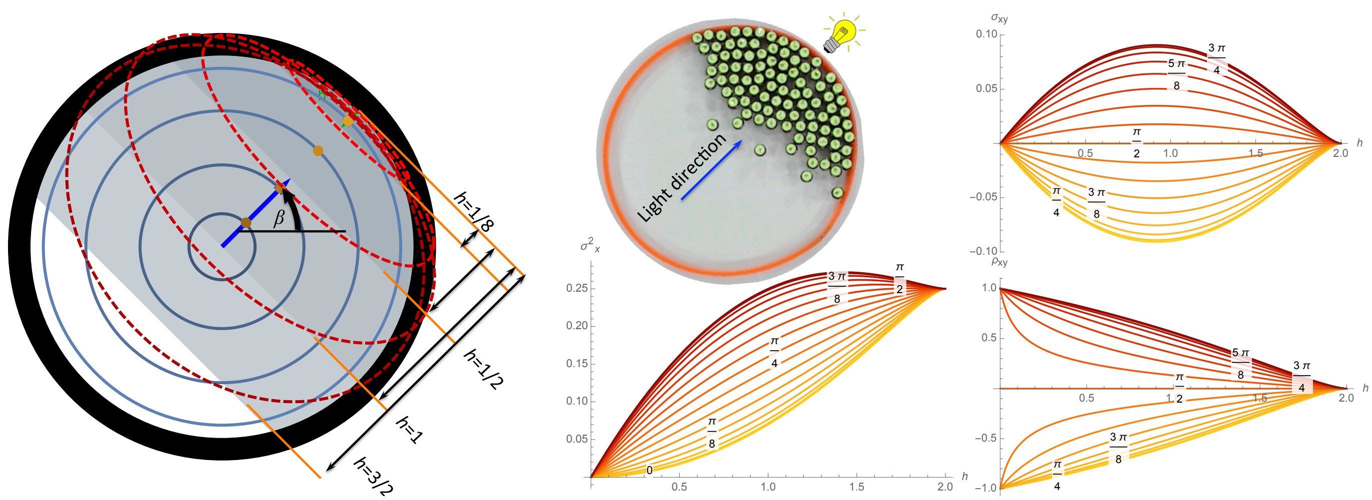

Fig. 3. Pushing the swarm against a circular boundary wall allows limited control of the shape of the swarm, as a function of the fill level h and the commanded movement direction β. Left plot shows locus of possible mean positions for four values of h. The locus of possible mean positions are concentric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

B. Friction with Boundary Wall

Global inputs move a swarm uniformly. Controlling covariance 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. Directly controlling covariance by pushing the swarm into a boundary requires changes to the boundary. An obstacle in the lower-right corner is enough to generate positive covariance. Generating both positive and negative covariance requires additional obstacles. Requiring special obstacle configuration also makes covariance control dependent on the local environment. Instead of pushing our robots directly into a wall, this paper examines an oblique approach, by 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 negative friction force that opposes movement along the boundary. This causes robots along the boundary to slow down compared to 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 $|\theta| < \pi/2$. The resulting net force on the robot, $F_{forward}$, is aligned with the wall and given by

$$
\begin{aligned}
&F_{forward} = F\sin(\theta) - F_f \\
&\text{where } F_f = \begin{cases} \mu_f N, & \mu_f N < F\sin(\theta) \\ F\sin(\theta), & \text{else} \end{cases} \\
&\text{and } N = F\cos(\theta)
\end{aligned} \tag{10}
$$

Fig. 4 shows the resultant forces on two robots when one is touching a wall. As illustrated, both experiences different net forces although each receives the same inputs. 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, if the control input is parallel or into the wall, the touching robot will not move. The next section shows how a system with friction model (10) and two orthogonal walls are sufficient to arbitrarily position two robots.

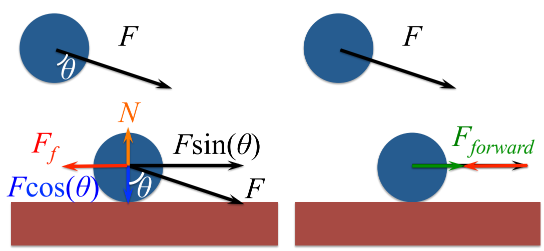

Fig. 4. Wall friction reduces the force for going forward $F_{forward}$ on a robot near a wall, but not for a free robot.

C. Position Control of 2 *Robots Using Wall Friction*

This section describes Alg. 1, an algorithm that uses wall-friction to arbitrarily position two robots in a rectangular workspace. This algorithm introduces concepts that will be used for multi-robot positioning. It only requires collisions with two orthogonal walls, in this case, the bottom and left walls. Fig. 5 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. Denote the current positions of the robots r_1 and r_2. Subscripts $_x$ and $_y$ denote the x and y coordinates, i.e., s_{1x} and s_{1y} denote the x and y locations of s_1. The algorithm assigns a global control input at every instance. The goal is to adjust $\Delta r_x = r_{2x} - r_{1x}$ from $\Delta s_x = s_{2x} - s_{1x}$ to $\Delta e_x = e_{2x} - e_{1x}$ and adjust $\Delta r_y = r_{2y} - r_{1y}$ from $\Delta s_y = s_{2y} - s_{1y}$ to $\Delta e_y = e_{2y} - e_{1y}$ using a shared global control input. This algorithm exploits the position-dependent friction model (10).

Our algorithm solves the positioning problem in two steps: First, $|\Delta r_x - \Delta e_x|$ is reduced to zero while Δr_y is kept constant in Alg. 2. Second $|\Delta r_y - \Delta e_y|$ is reduced to zero while Δr_x is kept constant in Alg. 3.

Step (i): Fixing Δr_x

1) If Δe_x is negative, the robots will be commanded to move toward the left wall and halt ϵ distance from the left wall. If $\Delta e_x \geq 0$, the robots will be commanded to move toward the right wall and halt ϵ distance from the right wall. The epsilon distance prevents the robots from experiencing friction along the vertical wall.
2) let $y_{\min} = \min_i r_{iy}$, i.e., $y_{\min}$ be the minimum height of the two robots. Move both robots downward $y_{\min}$ such that one of the robots touches the bottom wall and hence friction force will prevent this robot from moving left or right.
3) friction force holds the lower robot in place while the upper robot may move right and left freely to change Δr_x to Δe_x.
4) If after the free move of the upper robot Δr_x is not Δe_x, Step (i) will be repeated. No more than two iterations of step (i) are required.

Step (ii): Fixing Δr_y Now that Δr_x is corrected, Δr_y must be corrected:

1) If Δe_y is negative, the robots will be commanded to move toward the top wall and halt ϵ distance from the top wall. If $\Delta e_y \geq 0$, the robots will be commanded to move toward the bottom wall and halt ϵ distance from the bottom wall. The epsilon distance prevents the robots from experiencing friction along the horizontal wall.
2) let $x_{\min} = \min_i r_{ix}$, i.e., $x_{\min}$ be the minimum distance of the two robots from the origin along the x-axis. Move both robots to the left $x_{\min}$ such that one of the robots touches the left wall and hence friction force will prevent this robot from moving up or down.
3) friction force holds the left robot in place while the right robot may move up or down freely to change Δr_y to Δe_y.
4) If after the free move of the right robot Δr_y is not Δe_y, Step (ii) will be repeated. No more than two iterations of step (ii) are required.

Once Δr_x and Δr_y are set to Δe_x and Δe_y, a straight-line global input moves both robots from r_1 and r_2 to e_1 and e_2.

Algorithm 1 WallFrictionArrange2Robots(s_1, s_2, e_1, e_2, L)

Require: starting (s_1, s_2) and ending (e_1, e_2) positions of two robots. $(0, 0)$ is bottom corner, s_1 is rightmost robot, L is length of the walls. Current position of the robots are (r_1, r_2).

1: (r_1, r_2) = GenerateDesired$x$-spacing(s_1, s_2, e_1, e_2, L)
2: GenerateDesired$y$-spacing(r_1, r_2, e_1, e_2, L)

D. Position Control of n *Robots Using Wall Friction*

Alg. 1 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*. 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. 7 for a schematic of the build and staging zones.

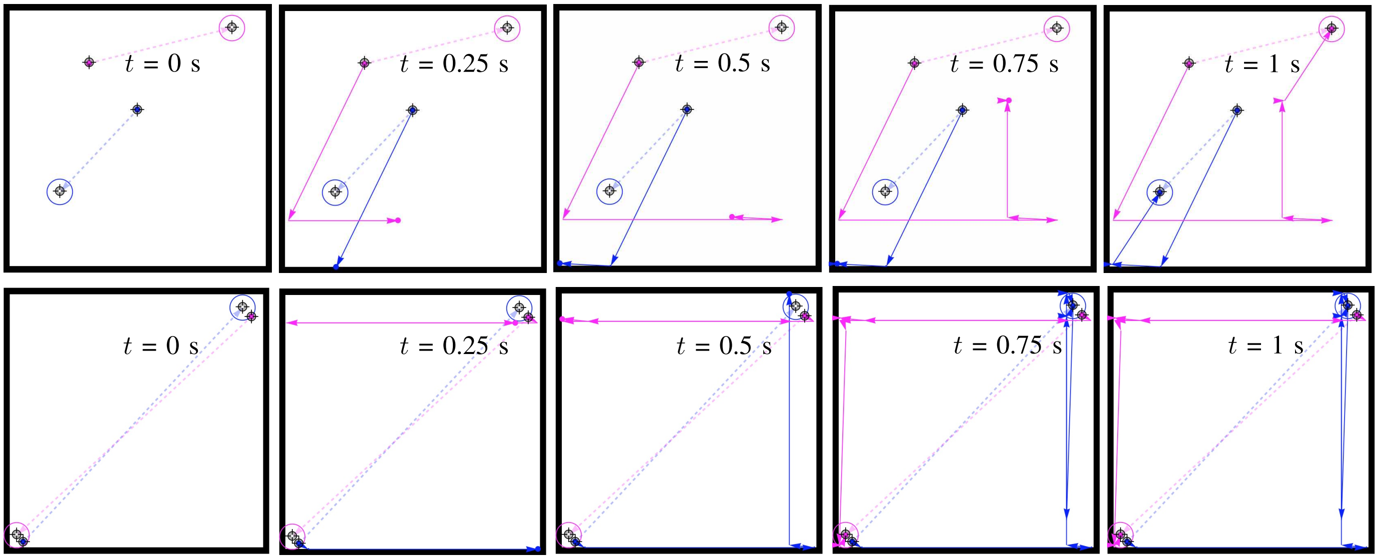

Fig. 5. Frames from an implementation of Alg. 1: two robot positioning using walls with infinite friction. The algorithm only requires friction along the bottom and left walls. Robot initial positions are shown by a crosshair, and final positions by a circled crosshair. Dashed lines show the shortest route if robots could be controlled independently. The path given by Alg. 1 is shown with solid arrows. The bottom row shows an extreme case where the robots must switch position.

Algorithm 2 GenerateDesired$x$-spacing(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 topmost robot, L is length of the walls. Current robot positions are (r_1, r_2).
Ensure: $r_{1y} - r_{2y} \equiv s_{1y} - s_{2y}$
$\epsilon \leftarrow$ small number
$\Delta s_x \leftarrow s_{1x} - s_{2x}$
$\Delta e_x \leftarrow e_{1x} - e_{2x}$
$r_1 \leftarrow s_1$, $r_2 \leftarrow s_2$
if $\Delta e_x < 0$ **then**
  $m \leftarrow (L - \epsilon - \max(r_{1x}, r_{2x}), 0)$ ▷ Move to right wall
else
  $m \leftarrow (\epsilon - \min(r_{1x}, r_{2x}), 0)$ ▷ Move to left wall
end if
$m \leftarrow m + (0, -\min(r_{1y}, r_{2y}))$ ▷ Move to bottom
$r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$ ▷ Apply move
if $\Delta e_x - (r_{1x} - r_{2x}) > 0$ **then**
  $m \leftarrow (\min(|\Delta e_x - \Delta s_x|, L - r_{1x}), 0)$ ▷ Move right
else
  $m \leftarrow (-\min(|\Delta e_x - \Delta s_x|, r_{1x}), 0)$ ▷ Move left
end if
$m \leftarrow m + (0, \epsilon)$ ▷ Move up
$r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$ ▷ Apply move
$\Delta r_x = r_{1x} - r_{2x}$
if $\Delta r_x \equiv \Delta e_x$ **then**
  return (r_1, r_2)
else
  return GenerateDesired$x$-spacing(r_1, r_2, e_1, e_2, L)
end if

Algorithm 3 GenerateDesired$y$-spacing(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 rightmost robot, L is length of the walls. Current robot positions are (r_1, r_2).
Ensure: $r_{1x} - r_{2x} \equiv s_{1x} - s_{2x}$
$\Delta s_y \leftarrow s_{1y} - s_{2y}$
$\Delta e_y \leftarrow e_{1y} - e_{2y}$
$r_1 \leftarrow s_1$, $r_2 \leftarrow s_2$
if $\Delta e_y < 0$ **then**
  $m \leftarrow (L - \max(r_{1y}, r_{2y}), 0)$ ▷ Move to top wall
else
  $m \leftarrow (-\min(r_{1y}, r_{2y}), 0)$ ▷ Move to bottom wall
end if
$m \leftarrow m + (0, -\min(r_{1x}, r_{2x}))$ ▷ Move to left
$r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$ ▷ Apply move
if $\Delta e_y - (r_{1y} - r_{2y}) > 0$ **then**
  $m \leftarrow (\min(|\Delta e_y - \Delta s_y|, L - r_{1y}), 0)$ ▷ Move top
else
  $m \leftarrow (-\min(|\Delta e_y - \Delta s_y|, r_{1y}), 0)$ ▷ Move bottom
end if
$m \leftarrow m + (0, \epsilon)$ ▷ Move right
$r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$ ▷ Apply move
$\Delta r_y = r_{1y} - r_{2y}$
if $\Delta r_y \equiv \Delta e_y$ **then**
  $m \leftarrow (e_{1x} - r_{1x}, e_{1y} - r_{1y})$
  $r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$ ▷ Apply move
  return (r_1, r_2)
else
  return GenerateDesired$y$-spacing(r_1, r_2, e_1, e_2, L)
end if

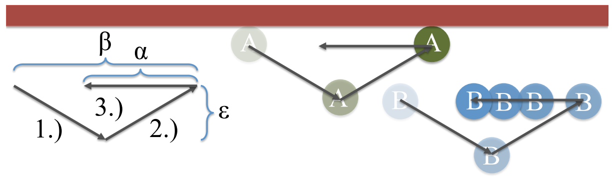

Fig. 6. A DriftMove$(\alpha, \beta, \epsilon)$ to the right consists of repeating a triangular movement sequence $\{(\beta/2, -\epsilon), (\beta/2, \epsilon), (-\alpha, 0)\}$. The robot A touching a top wall will move right β units, while robots not touching the top move right $\beta - \alpha$.

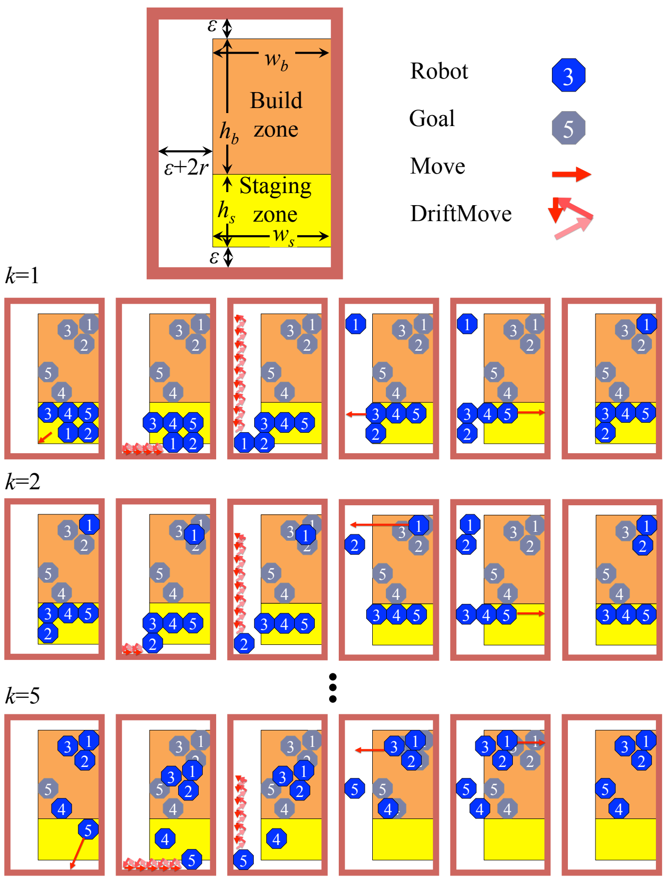

Fig. 7. Illustration of Alg. 4, n robot position control using wall friction.

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 $(\epsilon + 2r + \max(w_f, w_s), 2\epsilon + h_s, h_f)$.

The n robot position control algorithm relies on a DriftMove$(\alpha, \beta, \epsilon)$ control input, shown in Fig. 6. 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 $\beta - \alpha$.

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 top-to-bottom, and the f_k configurations right-to-left and top-to-bottom as shown in Fig. 7.

Algorithm 4 PositionControl$n$RobotsUsingWallFriction(k)

```
move( −ε, r − p_{k,y})
while  p_{k,x} > r do
    DriftMove(ε, min(p_{k,x} − r, ε), ε) left
end while
m ← ceil((f_{k,y} − r)/ε)
β ← (f_{k,y} − r)/m
α ← β − (r − p_{k,y} − ε)/m
for  m iterations do
    DriftMove(α, β, ε) up
end for
move(r + ε − f_{k,x}, 0)
move(f_{k,x} − r, 0)
```

Alg. 4 procedes 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 DriftMove()s are executed that move robot k to the left wall with no net movement of the other robots. Third, a set of DriftMove()s 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.

E. Controlling Covariance Using Wall Friction

Assume an open workspace with infinite boundary friction. Goal variances and covariance are $(\sigma^2_{goalx}, \sigma^2_{goaly}, \sigma_{goalxy}$ and mean, variances and covariance of the swarm are $(\bar{x}, \bar{y}, \sigma^2_x, \sigma^2_y, \sigma_{xy})$. For our experiments, $c_1 = 0.1$.

1) swarm is pushed into the left wall until $\sigma^2_x < c_1 \sigma^2_{goalx}$.
2) swarm's mean position is moved to the center of the workspace
3) swarm is pushed into the bottom wall until $\sigma^2_y \leq \sigma^2_{goalt}$.
4) if $\sigma_{goalxy} > 0$ swarm slides right until $\sigma_{xy} \geq \sigma_{goalxy}$ else swarm slides left until $\sigma_{xy} \leq \sigma_{goalxy}$
5) swarm's mean position is moved to the center of the workspace

IV. 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. All code is available online at [link withheld for review].

Two additional simulations were performed using wall-friction to control variance and covariance. The first is an

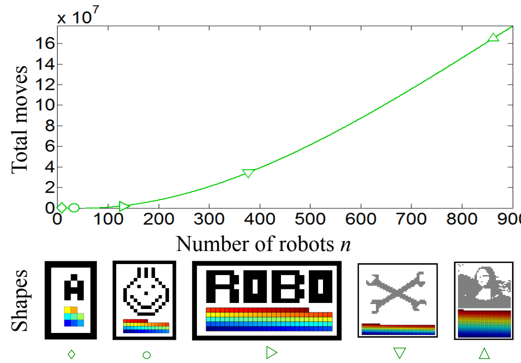

Fig. 8. The required number of moves under Alg. 4 using wall-friction to rearrange n square-shaped robots. See hardware implementation and simulation at [link withheld for review].

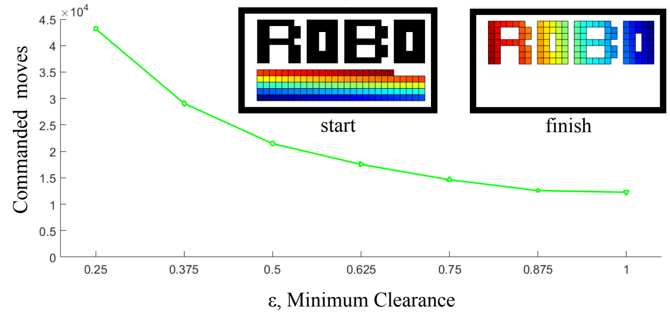

Fig. 9. Control performance is sensitive to the desired clearance ϵ. As ϵ increases, the total distance decreases asymptotically.

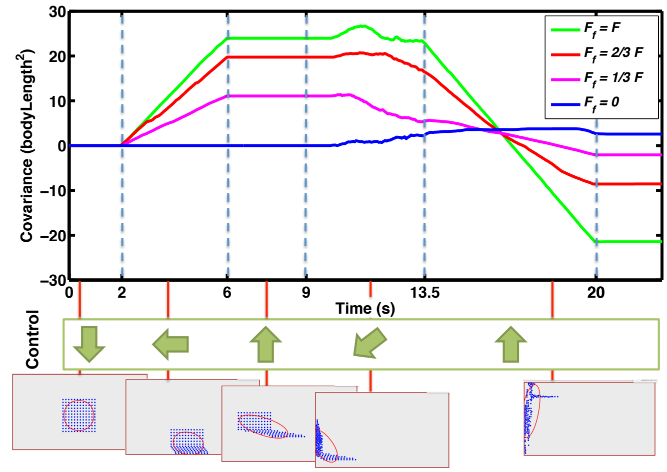

Fig. 10. Open-loop simulation with 144 disc robots and varying levels of boundary friction under the same initial conditions. Without friction, covariance is unchangeable. As friction increases, the covariance can be manipulated to greater degrees.

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

Algorithms 1, 2, 3, were implemented in Mathematica using point robots (radius = 0). Fig. 5 shows this algorithm for two configurations. Robot initial positions are shown by a crosshair, and final positions by a circled crosshair. Dashed lines show the shortest route if robots could be controlled independently. The path given by Alg. 1 is shown with solid arrows. Each row has five snapshots taken every quarter second. For the sake of brevity axis-aligned moves were replaced with oblique moves that combine two moves simultaneously. Δr_x is adjusted to Δe_x in the second snapshot at $t = 0.25$. The following frames adjust Δr_y to Δe_y. Δr_y is corrected by $t = 0.75$. Finally, the algorithm moves the robots to their corresponding destinations.

In the worse case, adjusting both Δr_x and Δr_y requires two iterations. Two iterations of Alg. 2 are only required if $|\Delta e_x - \Delta s_x| > L$. Similarly, two iterations of Alg. 3 are only required if $|\Delta e_y - \Delta s_y| > L$.

B. Position Control of n Robots

Alg. 4 was simulated in MATLAB using square block robots with unity width. Simulation results are shown in Fig. 8 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..

In Fig. 8, the amount of clearance $\epsilon = 1$. Control performance is sensitive to the desired clearance. As ϵ increases, the total distance decreases asymptotically, as shown in Fig. 9, because the robots have more room to maneuver and less DriftMoves are required.

C. Efficient Control of Covariance

A set of simulations were conducted to demonstrate the importance of boundary friction. These simulations use the 2D physics engine Box2D, by Catto [1]. 144 disc-shaped robots were controlled by an open-loop control input as illustrated in Fig. 10. All robots had the same initial conditions, but in four tests the boundary friction was $F_f = \{0, 1/3F, 2/3F, F\}$. Without friction, covariance has minimal variation. As friction increases, the covariance can be manipulated to greater degrees.

144 disc-shaped robots were also controlled by a closed-loop controller using the procedure in Section III-E. Fig. 11 illustrates that covariance and variances in x and y axis were controlled from a set of initial conditions.

V. 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 Rubenstein et al. [13, 14] is a low-cost robot designed for testing collective algorithms with large numbers of robots. It is available as an open source platform or commercially from K-Team [7]. 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

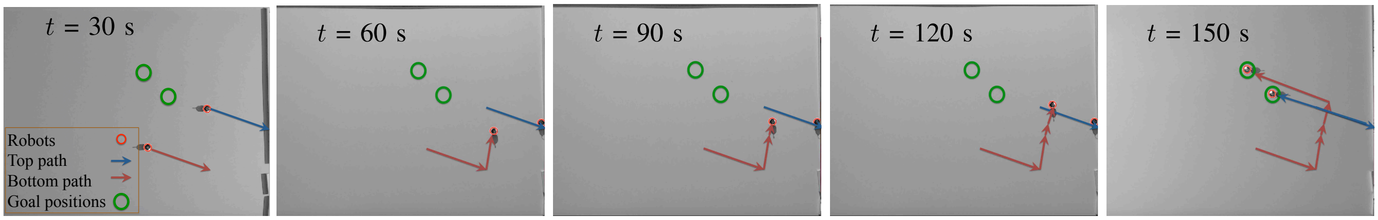

Fig. 12. Two robot positioning of two kilobot robots. The boundary walls have nearly infinite friction, so the blue robot is stopped by the wall from $t = 30$s until the commanded input is directed away form the wall at $t = 120$s, while the orange robot in free-space is unhindered.

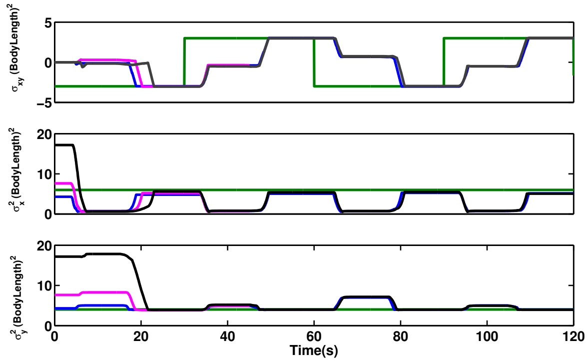

Fig. 11. Closed-loop simulation with 144 disc robots and three sets of initial conditions. The algorithm tracks goal variance and covariance values (green). The goal covariance switches sign every 30 s.

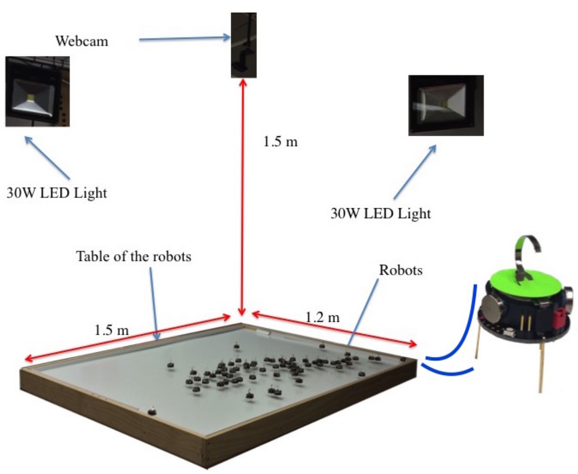

Fig. 13. Hardware platform: table with 1.5×1.2 m workspace, surrounded by eight remotely triggered 30W LED floodlights, with an overhead machine vision system.

source. In these experiments as shown in Fig. 13, we used n=97 kilobots, a 1.5 m×1.2 m whiteboard as the workspace, and eight 30W LED floodlights arranged 1.5 m above the plane of the table at the $\{N, NE, E, SE, S, SW, W, NW\}$ vertices of a 6 m square centered on the workspace. The lights were controlled using an Arduino Uno board connected to an 8-relay shield. Above the table, an overhead machine vision system tracks the position of the swarm.

The walls of the hardware platform have almost infinite friction, due to a laser-cut, zigzag border and the three-legged design of the kilobots. When a kilobot is steered into the zigzag border, they pin themselves to the wall unless the global

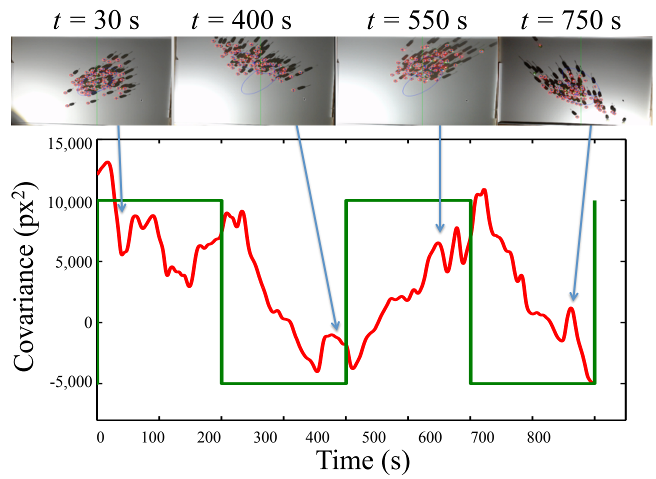

Fig. 14. Hardware demonstration steering 64 kilobot robots to desired covariance. The goal covariance is shown in green, the actual covariance in red. Frames above the plot show output from machine vision system and an overlaid covariance ellipse.

input directs them away from the wall. This wall friction is sufficient to enable independent control of two kilobots, as shown in Fig. 12.

To demonstrate covariance control n=97 robots were placed on the workspace and manually steered with a single light source, using friction with the boundary walls to vary the covariance from -5000 to 10,000 pixel^2. The resulting covariance is plotted in Fig. 14, along with snapshots of the swarm.

VI. Conclusion and Future Work

This paper presented techniques for controlling the shape of a swarm of robots using global inputs and interaction with boundary friction forces. The paper provided algorithms for precise position control, as well as demonstrations of efficient covariance control. 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 Kim et al. [8]. Additionally, this paper assumed that wall friction was nearly infinite. The algorithms require retooling to handle small μ_f friction coefficients. It may be possible to rank controllability as a function of friction. In hardware, the wall friction can be varied by laser-cutting boundary walls with different of profiles.

References

[1] Erin Catto. User manual, Box2D: A 2D physics engine for games, http://www.box2d.org, 2010. URL http://www.box2d.org.

[2] Sagar Chowdhury, Wuming Jing, and David J. Cappelleri. Controlling multiple microrobots: recent progress and future challenges. *Journal of Micro-Bio Robotics*, 10(1-4):1–11, 2015. doi: 10.1007/s12213-015-0083-6. URL http://dx.doi.org/10.1007/s12213-015-0083-6.

[3] Bruce R Donald, Christopher G Levey, Igor Paprotny, and Daniela Rus. Planning and control for microassembly of structures composed of stress-engineered mems microrobots. *The International Journal of Robotics Research*, 32(2):218–246, 2013. URL http://ijr.sagepub.com/content/32/2/218.abstract.

[4] Magnus Egerstedt and Xiaoming Hu. Formation constrained multi-agent control. *IEEE Trans. Robotics Automat.*, 17:947–951, 2001.

[5] Ambarish Ghosh and Peer Fischer. Controlled propulsion of artificial magnetic nanostructured propellers. *Nano Letters*, 9(6):2243–2245, 2011/10/23 2009. doi: 10.1021/nl900186w. URL http://dx.doi.org/10.1021/nl900186w.

[6] M Ani Hsieh, Vijay Kumar, and Luiz Chaimowicz. Decentralized controllers for shape generation with robotic swarms. *Robotica*, 26(05):691–701, 2008.

[7] K-Team. Kilobot, www.k-team.com/mobile-robotics-products/kilobot, 2015.

[8] Paul Seung Soo Kim, Aaron Becker, Yan Ou, Anak Agung Julius, and Min Jun Kim. Imparting magnetic dipole heterogeneity to internalized iron oxide nanoparticles for microorganism swarm control. *Journal of Nanoparticle Research*, 17(3):1–15, 2015.

[9] Sylvain Martel. Magnetotactic bacteria for the manipulation and transport of micro-and nanometer-sized objects. *Micro-and Nanomanipulation Tools*, 2015.

[10] Sylvain Martel, Samira Taherkhani, Maryam Tabrizian, Mahmood Mohammadi, Dominic de Lanauze, and Ouajdi Felfoul. Computer 3d controlled bacterial transports and aggregations of microbial adhered nano-components. *Journal of Micro-Bio Robotics*, 9(1-2):23–28, 2014.

[11] Bruce R. Munson, Alric P. Rothmayer, Theodore H. Okiishi, and Wade W. Huebsch. *Fundamentals of Fluid Mechanics*. Wiley, 7th edition, 2012.

[12] Famin Qiu and Bradley J Nelson. Magnetic helical micro-and nanorobots: Toward their biomedical applications. *Engineering*, 1(1):21–26, 2015.

[13] M. Rubenstein, C. Ahler, and R. Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *IEEE Int. Conf. Rob. Aut.*, pages 3293–3298, May 2012.

[14] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[15] Diana Spears, Wesley Kerr, and William Spears. Physics-based robot swarms for coverage problems. *The international journal of intelligent control and systems*, 11(3), 2006.

[16] A. Sudsang and L. E. Kavraki. A geometric approach to designing a programmable force field with a unique stable equilibrium for parts in the plane. In *Proceedings of The 2001 IEEE International Conference on Robotics and Automation (ICRA 2001)*, volume 2, pages 1079–1085, Seoul, Korea, May 2001. IEEE Press, IEEE Press. doi: 10.1109/ROBOT.2001.932737. This paper was a finalist for best conference paper award.

[17] T.H. Vose, P. Umbanhowar, and K.M. Lynch. Friction-induced velocity fields for point parts sliding on a rigid oscillated plate. *The International Journal of Robotics Research*, 28(8):1020–1039, 2009. doi: 10.1177/0278364909340279. URL http://ijr.sagepub.com/content/28/8/1020.abstract.

[18] Thomas H Vose, Paul Umbanhowar, and Kevin M Lynch. Sliding manipulation of rigid bodies on a controlled 6-dof plate. *The International Journal of Robotics Research*, 31(7):819–838, 2012.

[19] Xiaohui Yan, Qi Zhou, Jiangfan Yu, Tiantian Xu, Yan Deng, Tao Tang, Qian Feng, Liming Bian, Yan Zhang, Antoine Ferreira, and Li Zhang. Magnetite nanostructured porous hollow helical microswimmers for targeted delivery. *Advanced Functional Materials*, 25(33):5333–5342, 2015. ISSN 1616-3028. doi: 10.1002/adfm.201502248. URL http://dx.doi.org/10.1002/adfm.201502248.