Shaping a Swarm Using Wall Friction and a Shared Control Input

Paper-ID [add your ID here]

***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. Nevertheless, these applications require precision control of the shape and position of the robot swarm. There are different ways to control shape of the swarm, like using corner walls to make correlations in the axes of the swarm. However; with this method the correlation that can be maid is limited and we can not control the shape precisely. This paper focuses on using friction with boundary walls to break the symmetry caused by the global input. Wall friction is used to steer two robots to arbitrary positions, and this technique is then extended to control the position of n robots which are demonstrated with simulations. Finally, wall friction is used to efficiently control the covariance of a swarm of 97 hardware robots.**

I. Introduction

Micro- and nano-robots can be manufactured in large numbers. 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.

The mean position of a swarm is controllable and that, with an obstacle, the swarm's position variance orthogonal to rectangular boundary walls is also controllable (σ_x and σ_y for a workspace with axis-aligned walls). The usefulness of these techniques was demonstrated by several automatic controllers. One controller steered a swarm of robots to push a larger block through a 2D maze Shahrokhi and Becker [11]. A limitation is that variance control could only compress a swarm along the world x and y axes. This means the swarm could not navigate workspaces with narrow corridors with other orientations, such as those shown in Fig. 2. Challenges like these require a controller that regulates the swarm's position covariance, σ_{xy}. This paper proves that two orthogonal boundaries with high friction are sufficient to arbitrarily position two robots, extends this proof to prove that a rectangular workspace with high friction boundaries is sufficient to arbitrarily position a swarm of n robots, implements these position control algorithms in

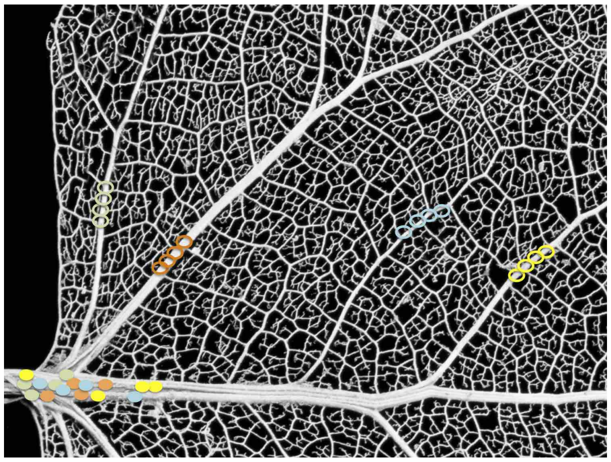
Fig. 1. Vascular networks are common in biology such as the circulatory system and cerebrospinal spaces, as well as in porous media including sponges and pumice stone. Navigating a swarm using global inputs, where each member receives the same control inputs, is challenging due to the many obstacles. This paper demonstrates how friction with walls can be used to change the shape of a swarm.

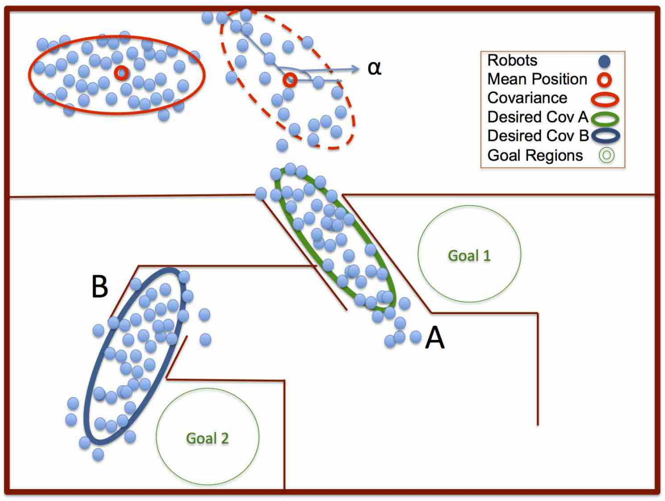

Fig. 2. Maintaining group cohesion while steering a swarm through an arbitrary maze requires covariance control.

simulation and on a hardware setup with up to 97 robots, and ends with directions for future research.

II. Related Work

Controlling the *shape*, or relative positions, of a swarm of robots is a key ability for a myriad of applications. Correspondingly, it has been studied from a control-theoretic perspective in both centralized, e.g. virtual leaders in EGERSTEDT [4], and decentralized approaches, e.g. control-Lyapunov functions gradient based decentralized controllers in Hsieh et al. [5]. Most approaches assume a level of intelligence and autonomy in the individual robots that exceeds the capabilities of current micro- and nano-robots Martel [7], Yan et al. [16].

Instead, this paper focuses on centralized techniques that apply the same control input to each member of the swarm, as in Becker et al. [1].

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, as described in introductory fluid dynamics textbooks Munson et al. [8]. 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 Spears et al. [12].

More research has focused on generating artificial force-fields. Applications have included techniques to design shear forces to a single object for sensorless manipulation Sudsang and Kavraki [13]. Vose et al. demonstrated a collection of 2D force fields generated by 6DOF vibration inputs to a rigid plate Vose et al. [14, 15]. This collection of force fields, including shear forces, could be used as a set of primitives for motion control for steering the formation of multiple objects.

III. Theory

A. Controlling Covariance: Fluid Settling In a Tank

One very simple idea to control the shape is using the corners and walls. Smashing the swarm to the corner walls will cause some correlation between x and y axes. Eq. **??** shows the relation between number of robots and area provided and the correlation we can reach with it. This equation shows that we are not able to reach high correlations bigger than $|0.5|$. Fig. **??** shows the different mean positions and correlations we are able to reach. There are ways and corridors that the swarm needs to pass and also to keep the majority of it. If we use the wall corners technique and the way needs a very high correlation to pass, a remarkable number of the swarm members will be lost, and we may lose track of them due to their small size. Meanwhile, the variance of the swarm would get bigger and bigger that it would not anymore be able to deliver sufficient payloads. For this reason we have thought another idea which uses *friction* to make correlations.

B. Friction

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 changing 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, as shown in Eq. (1). This causes robots along the boundary to slow down compared to robots in free-space. This enables covariance control using boundaries with arbitrary orientations.

Let the control input be a vector force $\vec{F}$ with magnitude F and orientation θ. The force of friction F_f is

$$\begin{aligned} N &= F\cos(\theta) \\ F_f &= \begin{cases} \mu_f N, & \mu_f N < F\sin(\theta) \\ F\sin(\theta), & \text{else} \end{cases} \\ F_{forward} &= F\sin(\theta) - F_f \end{aligned} \tag{1}$$

Fig. 3 shows the resultant forces on two robots when one is touching a wall. As illustrated, bot experiences different net forces although each receive 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 (1) and two walls are sufficient to arbitrarily position two robots.

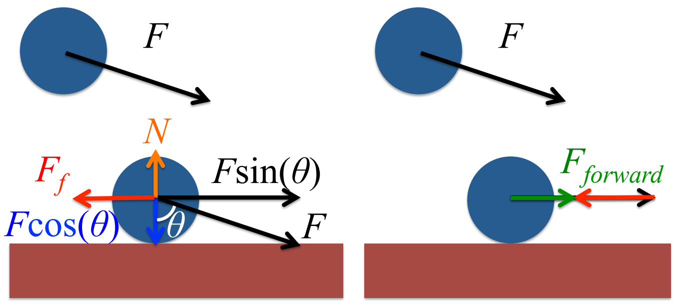

Fig. 3. Wall friction reduces the force for going forward $F_{forward}$ on a robot near a wall, but not for a free robot.

C. Position control of 2 robots using wall friction

This section describes an algorithm for positioning two robots and introduces concepts that will be used for multi-robot positioning. As we can see in Alg. 1, 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. Let the 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. As a result, our 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 similarly adjust $\Delta r_y = r_{2y} - r_{1y}$ from $\Delta s_y = s_{2y} - s_{1y}$ to $\Delta e_y = e_{2y} - e_{1y}$ with one global input at every instance. The key to the algorithm is the position-dependent friction model (1).

Our algorithm uses a divide and conquer method to solve the positioning problem. It finds the final position of the robots in two steps: (i) First, $|\Delta r_x - \Delta e_x|$ is reduced to zero while Δr_y is kept constant in Alg. 2. (ii) Having fixed Δr_x to Δe_x as desired, the algorithm next keeps Δr_x constant and adjusts Δr_y to Δe_y, as desired in Alg.. 3. Though steps (i) and (ii) are similar from an algorithmic point of view, the following subsections describe the process in detail.

Step (i): Fixing Δr_x

- Define $e'_1 = (e_{1x}, s_{1y})$ and $e'_2 = (e_{2x}, s_{2y})$. Our goal for defining e'_1 and e'_2 is to understand the direction to which robots should move in order to adjust Δr_x. Let $e'_{\text{top}} = \arg\max_i e'_{iy}$ and $e'_{\text{bottom}} = \arg\min_i e'_{iy}$. Now if $e'_{\text{top,x}} - e'_{\text{bottom,x}} > 0$, then the global input to both robots would be toward left direction and if $e'_{\text{top,x}} - e'_{\text{bottom,x}} < 0$, then the global input to both robots would be toward right direction. The two robots continue their horizontal path until one of them reaches the ϵ-neighborhood of one of the left or right walls.
- At this step, let $y_{\min} = \min_i r_{iy}$, i.e., $y_{\min}$ is the minimum height of the two robots. We move both robots downward by the amount of $y_{\min}$ such that one of the robots would touch the bottom wall and hence friction force will not let that robot to move left or right.
- The fact that the friction force of the bottom wall would not let the lower robot to move right or left will let the other robot to move to right and left freely to adjust Δr_x according to Δe_x.
- Finally, even if with the free move of the upper robot Δr_x is not set to the Δe_x, we can run the Step (i) (as described in the previous paragraphs) again to adjust the Δr_x. It is easy to show that it is guaranteed that we can adjust Δr_x to Δe_x in only two iterations.

Step (ii): Fixing Δr_y Now that we have adjusted the difference in robots' positions along one axis, we focus to do the same on the other axis as well. Therefore, similar to Section III-C, we employ the following steps:

- Let s'_1 and s'_2 be the points we derived at the end of the steps in Section III-C.
- Define $e''_1 = (s'_{1x}, e_{1y})$ and $e''_2 = (s'_{2x}, e_{2y})$. We define e''_1 and e''_2 to understand the direction to which robots should move in order to adjust Δr_y. Let $e''_{\text{right}} = \arg\max_i e''_{ix}$ and $e''_{\text{left}} = \arg\min_i e''_{ix}$. Now if $e''_{\text{right,y}} - e''_{\text{left,y}} > 0$, then the global input to both robots would be toward down direction and if $e''_{\text{right,y}} - e''_{\text{left,y}} < 0$, then the global input to both robots would be toward up direction. The two robots continue their vertical path until one of them reaches the ϵ-neighborhood of one of the top or bottom walls.
- At this step, let $x_{\min} = \min_i r_{ix}$, i.e., $x_{\min}$ is the minimum distance of the two robots from the origin along the x-axis. We move both robots to the left by the amount of $x_{\min}$ such that one of the robots would touch the left wall and hence friction force will not let that robot to move up or down.
- The fact that the friction force of the left wall would not let one of the robots to move up or down will let the other robot to move to up or down freely to adjust Δr_y according to Δe_y.
- Finally, even if with the free move of the robot which is not touching the wall Δr_y is not set to the Δe_y, we can run the Step (i) (as described in the previous paragraphs) again to adjust the Δr_y. It is easy to show that it is guaranteed that we can adjust Δr_y to Δe_y in only two iterations.

Once Δr_x and Δr_y are set to Δe_x and Δe_y, we can use global input to easily move both robots from r_1 and r_2 toward e_1 and e_2.

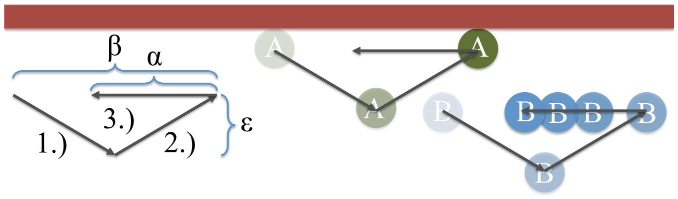

Fig. 4. A DriftMove$(\alpha, \beta, \epsilon)$ to the right consists of repeating a triangular movement sequence $\{(\beta/2, -\epsilon), (\beta/2, \epsilon), (-\alpha, 0)\}$. The robot A touching a top wall will move right β units, while robots not touching the top move right $\beta - \alpha$.

Algorithm 1 WallFrictionArrange2Robots(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 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

Algorithm 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.

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

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: $r_{1y} - r_{2y} \equiv s_{1y} - s_{2y}$
 1: $\epsilon \leftarrow$ small number
 2: $\Delta s_x \leftarrow s_{1x} - s_{2x}$
 3: $\Delta e_x \leftarrow e_{1x} - e_{2x}$
 4: $r_1 \leftarrow s_1$, $r_2 \leftarrow s_2$
 5: if $\Delta e_x < 0$ then
 6:     $m \leftarrow (L - \epsilon - \max(r_{1x}, r_{2x}), 0)$   ▷ Move to right wall
 7: else
 8:     $m \leftarrow (\epsilon - \min(r_{1x}, r_{2x}), 0)$   ▷ Move to left wall
 9: end if
10: $m \leftarrow m + (0, -\min(r_{1y}, r_{2y}))$   ▷ Move to bottom
11: $r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$   ▷ Apply move
12: if $\Delta e_x - (r_{1x} - r_{2x}) > 0$ then
13:     $m \leftarrow (\min(|\Delta e_x - \Delta s_x|, L - r_{1x}), 0)$   ▷ Move right
14: else
15:     $m \leftarrow (-\min(|\Delta e_x - \Delta s_x|, r_{1x}), 0)$   ▷ Move left
16: end if
17: $m \leftarrow m + (0, \epsilon)$   ▷ Move up
18: $r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$   ▷ Apply move
19: $\Delta r_x = r_{1x} - r_{2x}$
20: if $\Delta r_x \equiv \Delta e_x$ then
21:     return $(r_1, r_2)$
22: else
23:     return GenerateDesired$x$-spacing$(r_1, r_2, e_1, e_2, L)$
24: end if
```

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 $\text{DriftMove}(\alpha, \beta, \epsilon)$ control input, shown in Fig. 4. 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. 5.

Alg. 4 procedes as follows: First, the robots are moved left away from right wall, and down so robot k touches the bottom wall. Second, a set of $\text{DriftMove}()$s are executed that move robot k to the left wall with no net movement of the other robots. Third, a set of $\text{DriftMove}()$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 $k-1$.

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 position of the robots are $(r_1, r_2)$.
Ensure: $r_{1x} - r_{2x} \equiv s_{1x} - s_{2x}$
 1: $\Delta s_y \leftarrow s_{1y} - s_{2y}$
 2: $\Delta e_y \leftarrow e_{1y} - e_{2y}$
 3: $r_1 \leftarrow s_1$, $r_2 \leftarrow s_2$
 4: if $\Delta e_y < 0$ then
 5:     $m \leftarrow (L - \max(r_{1y}, r_{2y}), 0)$   ▷ Move to top wall
 6: else
 7:     $m \leftarrow (-\min(r_{1y}, r_{2y}), 0)$   ▷ Move to bottom wall
 8: end if
 9: $m \leftarrow m + (0, -\min(r_{1x}, r_{2x}))$   ▷ Move to left
10: $r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$   ▷ Apply move
11: if $\Delta e_y - (r_{1y} - r_{2y}) > 0$ then
12:     $m \leftarrow (\min(|\Delta e_y - \Delta s_y|, L - r_{1y}), 0)$   ▷ Move top
13: else
14:     $m \leftarrow (-\min(|\Delta e_y - \Delta s_y|, r_{1y}), 0)$   ▷ Move bottom
15: end if
16: $m \leftarrow m + (0, \epsilon)$   ▷ Move right
17: $r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$   ▷ Apply move
18: $\Delta r_y = r_{1y} - r_{2y}$
19: if $\Delta r_y \equiv \Delta e_y$ then
20:     $m \leftarrow (e_{1x} - r_{1x}, e_{1y} - r_{1y})$
21:     $r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$   ▷ Apply move
22:     return $(r_1, r_2)$
23: else
24:     return GenerateDesired$y$-spacing$(r_1, r_2, e_1, e_2, L)$
25: end if
```

Algorithm 4 PositionControl$n$RobotsUsingWallFriction(k)

```
 1: move( $-\epsilon, r - p_{k,y}$)
 2: while $p_{k,x} > r$ do
 3:     $\text{DriftMove}(0, \min(p_{k,x} - r, \epsilon), \epsilon)$ left
 4: end while
 5: $m \leftarrow \text{ceil}(\frac{f_{ky} - r}{\epsilon})$
 6: $\beta \leftarrow \frac{f_{ky} - r}{m}$
 7: $\alpha \leftarrow \beta - \frac{\epsilon}{m}$
 8: for $m$ iterations do
 9:     $\text{DriftMove}(\alpha, \beta, \epsilon)$ up
10: end for
11: move($r + \epsilon - f_{kx}, 0$)
12: move($f_{kx} - r, 0$)
```

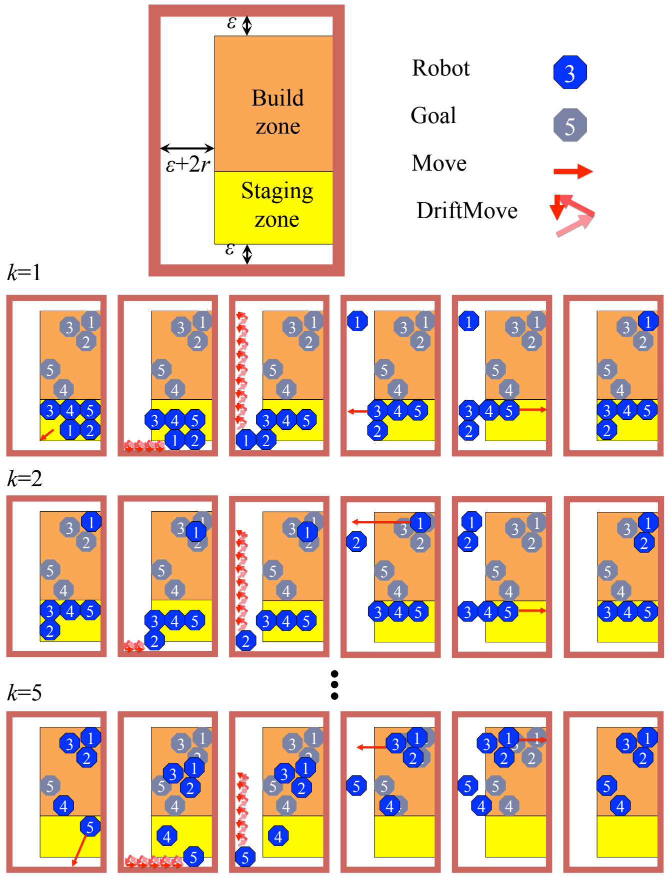

Fig. 5. Illustration of Alg. 4, n robot position control using wall friction.

Finally, all robots are moved right until robot k is in the desired target position.

E. Controlling Covariance Using Wall Friction

We can use friction to control covariance of the swarm.

Algorithm 5 Swarm Covariance Control With Wall Friction

Require: Swarm is Gaussian Distributed(in limit it is uniformly distributed)

Require: L is the length of the x of the workspace

```
1: while do $\bar{x} < L - \sigma_x(t)$
2:     Go Left
3: end while
4: Go Up for ...
```

IV. Simulation

Algorithms 1, 2, 3, were implemented in Mathematica using point robots (radius = 0). Figs 6 and 7 show the examples of the implementation of our algorithm. In both of these figures we have denoted the starting points and the destinations by small circles. However, destination points are surrounded by larger circles so as to be distinct from starting points.

In each of these figures we have five snapshots of the running of our algorithm taken every quarter second. For the sake of brevity we have replaced straight moves (e.g. upward, downward, etc) with oblique moves that shows a combination of two moves simultaneously (e.g. left and down together).

As we can see, in Fig. 6 Δr_x is adjusted to Δe_x in the second snapshot, i.e., at $t = t_1$ where $t_1 < 0.25$. The rest of the steps in this figure is dedicated to adjusting the Δr_y to Δe_y. As it is clear from Fig. 6, Δr_y is also adjusted at $t = t_2$ where $0.75 < t_2 < 1$. Finally, once Δr_x and Δr_y are adjusted, the algorithm gives a global input both of the robots so as to move them toward their corresponding destinations. This is happening in the time interval of $(t_2, 1]$.

Similarly, in Fig. 7 we can see that the Δr_x is adjusted in the third snapshot, i.e., at $t = t_3$ where $0.25 < t_3 < 0.5$ and Δr_y is adjusted in the last snapshot at $t = t_4$ where $0.75 < t_4 < 1$. The final positioning steps are happening in the time interval of $(t_4, 1]$.

As we pointed out earlier, adjusting each of Δr_x and Δr_y needs two iterations in the worst case. In other words, both of the Alg. 2 and Alg. 3 are executed two times in the worst case in positioning process of the robots. It is easy to see that we need two iterations of Alg. 2 only if $|\Delta e_x - \Delta s_x| > L$. Similarly we need two iterations of Alg. 3 only if $|\Delta e_y - \Delta s_y| > L$.

V. Experiment

A. Hardware system

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 Rubenstein et al. [9, 10] is a low-cost robot designed for testing collective algorithms with large numbers of robots. It is available commercially or as an open source platform K-Team [6]. 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. 8, we used n=64 kilobots, a 1.5 m$\times$1.2 m whiteboard as the workspace, and four 30W LED floodlights arranged 1.5 m above the plane of the table at the $\{N, E, S, W\}$ 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 board. At top of the table, an overhead machine vision system was added to track the position of the swarm.

The walls of the hardware platform have almost infinite friction, due to the three legged design of the kilobots. When a kilobot is steered into the wall, they pin themselves to the wall until the light changes direction and they begin turning in the other direction. This wall friction is sufficient to enable independent position control of two kilobots, as shown in Fig. 9.

To demonstrate covariance control $n = 64$ robots were placed on the workspace and manually steered with a single light source, using friction with the boundary walls to vary the

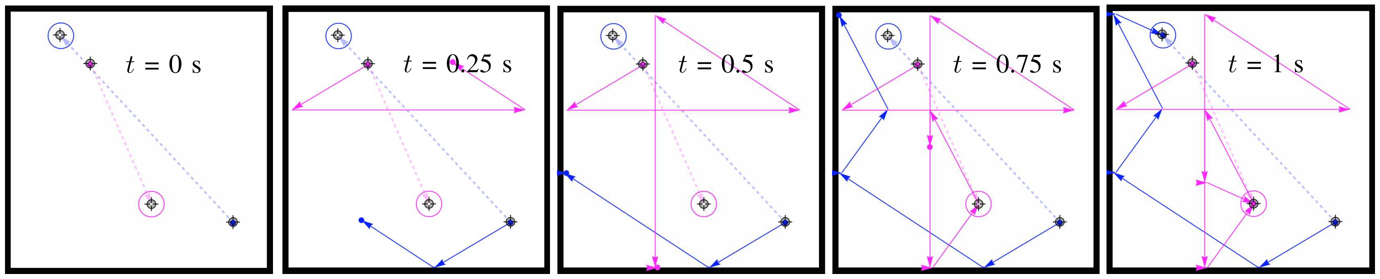

Fig. 6. Frames from an implementation of Alg. 1: two robot positioning using walls with infinite friction. Robot initial positions are shown by a crosshair, and final positions by a circled crosshair. Dashed lines show the shortest route if robots could be controlled independently. The path given by Alg. 1 is shown with solid arrows.

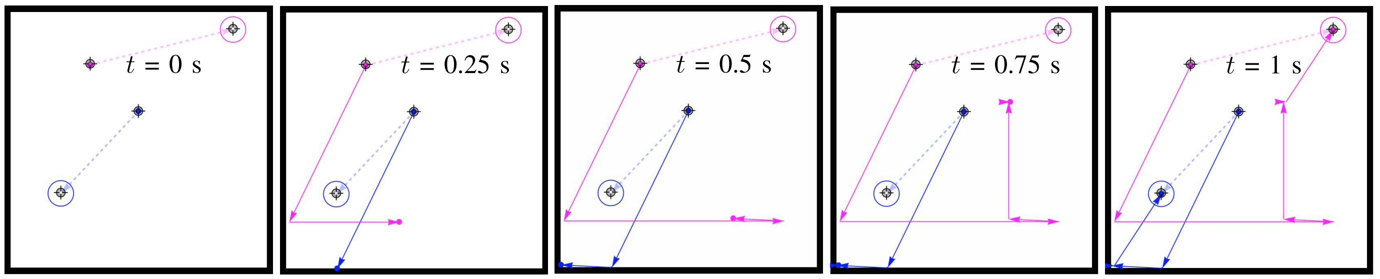

Fig. 7. Two robot positioning: switching positions using walls with infinite friction.

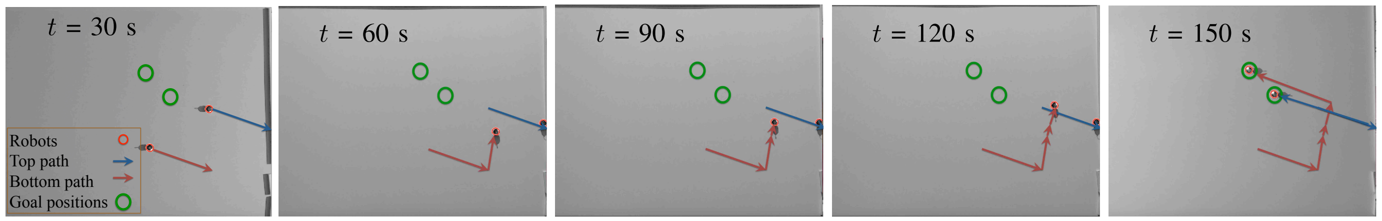

Fig. 9. Two robot positioning using the hardware setup and two kilobot robots. The walls have nearly infinite friction, as illustrated by this fig, the robot with the blue path that is stopped by the wall until the light changes orientation, while the orange robot in free-space is unhindered.

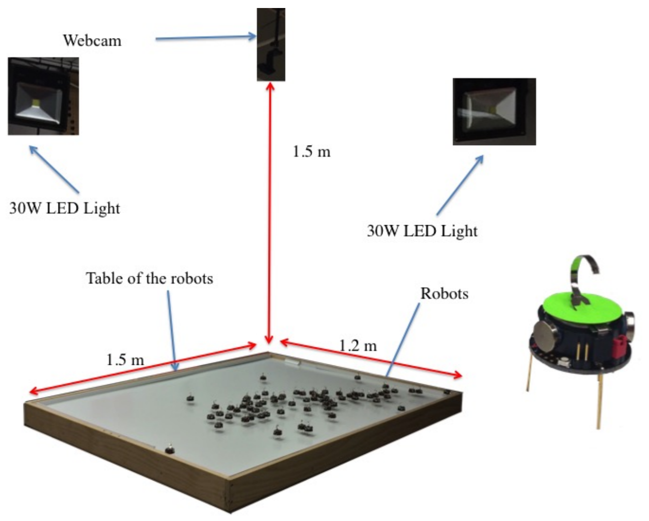

Fig. 8. Hardware platform: table with 1.5×1.2 m workspace, surrounded by four remotely triggered 30W LED floodlights, with an overhead machine vision system.

covariance from -5000 to 10,000. The resulting covariance is plotted in Fig. 10, 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 Becker et al. [2, 3]. 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, friction could be modified by outfitting the kilobots with a round skirt to avoid the almost infinite friction due to the triangular leg arrangement. The wall friction can be varied by laser-cutting boundary walls

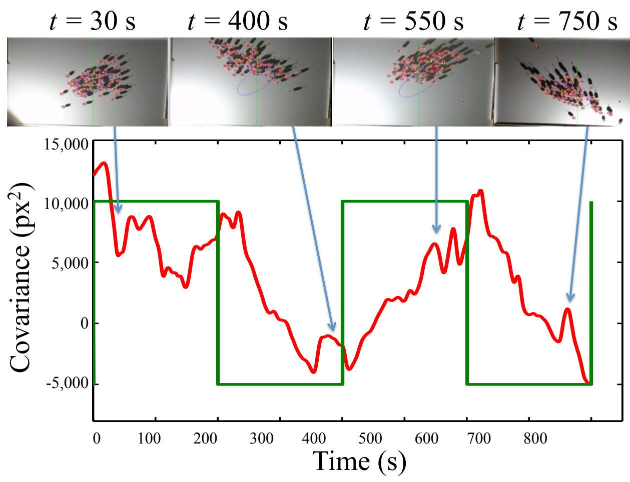

Fig. 10. Hardware demonstration steering 64 kilobot robots to desired covariance. Frames above the plot show output from machine vision system and an overlaid covariance ellipse.

with different of profiles.

Acknowledgments

References

[1] Aaron Becker, Golnaz Habibi, Justin Werfel, Michael Rubenstein, and J. McLurkin. Massive uniform manipulation: Controlling large populations of simple robots with a common input signal. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 520–527, November 2013.

[2] Aaron Becker, Yan Ou, Paul Kim, MinJun Kim, and Agung Julius. Feedback control of many magnetized tetrahymena pyriformis cells by exploiting phase inhomogeneity. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3317–3323, November 2013.

[3] Aaron Becker, Ouajdi Felfoul, and Pierre E Dupont. Simultaneously powering and controlling many actuators with a clinical MRI scanner. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2017–2023. IEEE, 2014.

[4] M EGERSTEDT. Formation constrained multi-agent control. *IEEE Trans. Robotics Automat.*, 17:947–951, 2001.

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

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

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

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

[9] 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.

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

[11] Shiva Shahrokhi and Aaron T. Becker. Stochastic swarm control with global inputs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page tbd, September 2015.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.