

# Shaping a Swarm Using Wall Friction and a Shared Control Input

Shiva Shahrokhi and Aaron T. Becker

**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. This paper uses 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. These techniques are demonstrated with simulations and hardware robots. Finally, wall friction is used to efficiently control the covariance of a swarm of 64 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. Large numbers of micro robots are required in order 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 incredible under-actuation.

Our previous work focused on a block-pushing task, where a swarm of robots pushed a larger block through a 2D maze. We proved the mean position of a swarm is controllable and that with an obstacle the swarm's position variance ( $\sigma_x$  and  $\sigma_y$ ) is also controllable. We then wrote automatic controllers which could complete a block pushing task, but these controllers had some limitations [2]. One of the limitations was that we could only compress our swarm along the world  $x$  and  $y$  axes, and could not navigate workspaces with narrow corridors with other orientations. One solution to these problems would be a controller that regulates the swarm's position covariance,  $\sigma_{xy}$ . For controlling  $\sigma_{xy}$ , we prove that the swarm position covariance  $\sigma_{xy}$  is controllable given boundaries with non-zero friction. We then prove that two orthogonal boundaries with high friction are sufficient to arbitrarily position a swarm of robots. We conclude by designing controllers that efficiently regulate  $\sigma_{xy}$ .

This paper (1) proves that the swarm position covariance  $\sigma_{xy}$  is controllable given boundaries with non-zero friction,

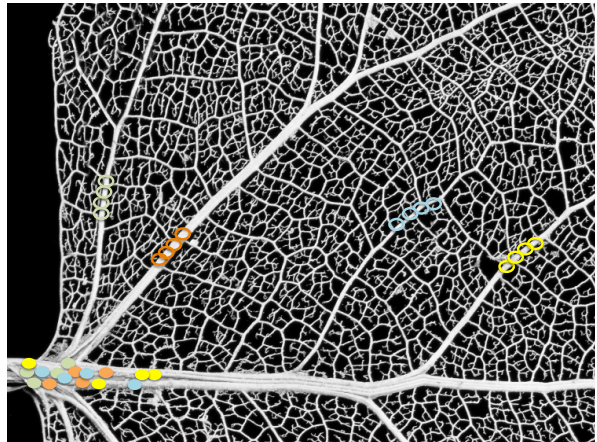


Fig. 1. Vascular networks are common in biology such as the circulatory system and cerebrospinal spaces, as well as in porous media including sponges and pumice stone. Navigating a swarm using global inputs, where each member receives the same control inputs, is challenging due to the many obstacles.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

(2) proves that two orthogonal boundaries with high friction are sufficient to arbitrarily position two robots, (3) proves that two orthogonal boundaries with high friction are sufficient to arbitrarily position a swarm of robots, (4) shows full-state position control with 2 or more robots using extensive simulations, and (5) demonstrate covariance control on our hardware platform with a large number of hardware robots.

## II. RELATED WORK

### A. Using shear forces to shape a set of particles

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 textbooks [?]. 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 the swarm. Physic's-based swarm simulations have used these forces to disperse a swarm's spatial position for accomplishing coverage tasks [?].

artificial forces have been used to apply shear forces to a single object [?]

For multiple objects, Vose et al. demonstrated a collection of 2D force fields generated by 6DOF vibration inputs to a rigid plate [?], [?]. This collection of force fields could be

used as a set of primitives for motion control, and included shear forces

Flow in a pipe or body lumen

### B. Global-control of micro- and nanorobots

We are particularly motivated by harsh constraints in micro- and nanorobotic systems. Small robots are often powered and steered by a global, broadcast control signal. Examples include *scratch-drive microrobots*, actuated and controlled by a DC voltage signal from a substrate [?], [?]; *light-driven nanocars*, synthetic molecules actuated by a specific wavelength of light [?], *MagMite* microrobots with different resonant frequencies controlled by a global magnetic field [?]; and magnetically controlled nanoscale helical screws [?], [?]. Large numbers of robots can be constructed, but the user interaction required to individually control each robot scales linearly with robot population. Instead, user interaction is often constrained to modifying a global input: while one robot is controlled, the rest are ignored. Making progress in targeted therapy and swarm manipulation requires the coordinated control of large robot populations.

### C. Human user studies with large swarms

There is currently no comprehensive understanding of user interfaces for controlling multi-robot systems with massive populations [?]. Our previous work with over a hundred hardware robots and thousands of simulated robots [1] demonstrated that direct human control of large swarms is possible. Unfortunately, the logistical challenges of repeated experiments with over one hundred robots prevented large-scale tests. To gather better data, we designed *SwarmControl.net*, a large-scale online game to test how humans interact with large swarms [?]. Our goal was to test several scenarios involving large-scale human-swarm interaction, and to do so with a statistically-significant sample size. These experiments showed that numerous simple robots responding to global control inputs are directly controllable by a human operator without special training, and that the visual feedback of the swarm state should be very simple in order to increase task performance. All code [?], and experimental results were posted online. The current paper presents motion primitives and an automatic controller that solves one of the games from *SwarmControl.net*.

### D. Block-pushing and Compliant Manipulation

Unlike *caging* manipulation, where robots form a rigid arrangement around an object [?], [?], our swarm of robots is unable to grasp the blocks they push, and so our manipulation strategies are similar to *nonprehensile manipulation* techniques, e.g. [?], where forces must be applied along the center of mass of the moveable object. A key difference is that our robots are compliant and tend to flow around the object, making this similar to fluidic trapping [?], [?].

Our  $n$ -robot system with 2 control inputs and  $4n$  states is inherently under-actuated, and superficially bears resemblance to compliant, under-actuated manipulators [?], [?].

Like these manipulators, the swarm conforms to the object to be manipulated. However our swarm lacks the restoring force provided by flexures in [?] and the silicone in [?]. Our swarm tends to disperse itself, so we require artificial forces, such as the variance control primitives in Section ??, to regroup the swarm.

## III. THEORY

### A. Controlling covariance using friction

Controlling covariance needs the swarm to break symmetry when we are using global inputs. We can use an obstacle like a corner, but for having positive and negative covariances, one corner is not enough and for any covariances we should have a special obstacle which is very hard and make us very dependent to the environment. Instead, walls have friction that is sufficient to break the symmetry. The robots which are touching the wall will have negative friction force as shown in Eq. ??, causing them to slow down in comparison to the free robots. This is enough for breaking symmetry and enables us to control covariance with any horizontal and vertical wall.

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

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

Fig. 2 shows forces on two robots when one is touching a wall. As shown, there are different forces for different robots although they are receiving the same input. For ease of analysis, the following algorithms assume  $\mu$  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.

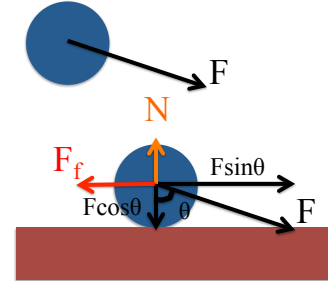


Fig. 2. Wall friction reduces the force for going forward  $F_{forward}$  on a robot near a wall, but not for a free robot.

## IV. POSITION CONTROL OF 2 ROBOTS USING WALL FRICTION

This section focuses on describing an algorithm for positioning two robots in introduces concepts that will be used

for multi-robot positioning. Therefore, 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 employing the assumption we have made earlier about the walls' friction.

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,  $\Delta r_x$  is reduced to zero while  $\Delta r_y$  is kept constant. (ii) Having fixed  $\Delta r_x$  to  $\Delta e_x$  as desired, the algorithm next keeps  $\Delta r_x$  constant and adjusts  $\Delta r_y$  to  $\Delta e_y$ , as desired. Though steps (i) and (ii) are similar from an algorithmic point of view, the following subsections describe the process in detail.

#### A. Step (i): Fixing $\Delta r_x$

- 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  $\Delta r_x$ . 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  $\epsilon$ -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  $\Delta r_x$  according to  $\Delta e_x$ .
- Finally, even if with the free move of the upper robot  $\Delta r_x$  is not set to the  $\Delta e_x$ , we can run the Step (i) (as described in the previous paragraphs) again to adjust the  $\Delta r_x$ . It is easy to show that it is guaranteed that we can adjust  $\Delta r_x$  to  $\Delta e_x$  in only two iterations.

#### B. Step (ii): Fixing $\Delta r_y$

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 IV-A, we employ the following steps:

- Let  $s'_1$  and  $s'_2$  be the points we derived at the end of the steps in Section IV-A.
- 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  $\Delta r_y$ . 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  $\epsilon$ -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  $\Delta r_y$  according to  $\Delta e_y$ .
- Finally, even if with the free move of the robot which is not touching the wall  $\Delta r_y$  is not set to the  $\Delta e_y$ , we can run the Step (i) (as described in the previous paragraphs) again to adjust the  $\Delta r_y$ . It is easy to show that it is guaranteed that we can adjust  $\Delta r_y$  to  $\Delta e_y$  in only two iterations.

Once  $\Delta r_x$  and  $\Delta r_y$  are set to  $\Delta e_x$  and  $\Delta e_y$ , we can use global input to easily move both robots from  $r_1$  and  $r_2$  toward  $e_1$  and  $e_2$ .

---

#### 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:  $\Delta s_x \leftarrow s_{1x} - s_{2x}$ 
2:  $\Delta s_y \leftarrow s_{1y} - s_{2y}$ 
3:  $\Delta e_x \leftarrow e_{1x} - e_{2x}$ 
4:  $\Delta e_y \leftarrow e_{1y} - e_{2y}$ 
5: if  $\Delta s_x < \Delta s_y$  then
6:   GenerateDesired $x$ -spacing( $s_1, s_2, e_1, e_2, L$ )
7:   GenerateDesired $y$ -spacing( $s_1, s_2, e_1, e_2, L$ )
8: else
9:   GenerateDesired $y$ -spacing( $s_1, s_2, e_1, e_2, L$ )
10:  GenerateDesired $x$ -spacing( $s_1, s_2, e_1, e_2, L$ )
11: end if
```

---

## V. 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  $k$ th loop moves the  $k$ th robot from staging zone to the desired position in a build zone. At the end the  $k$ th 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.

---

**Algorithm 2** GenerateDesiredx-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:    $m \leftarrow (L - \epsilon - \max(r_{1x}, r_{2x}), 0)$     $\triangleright$  Move to right wall
- 7: **else**
- 8:    $m \leftarrow (\epsilon - \min(r_{1x}, r_{2x}), 0)$     $\triangleright$  Move to left wall
- 9: **end if**
- 10:  $m \leftarrow m + (0, -\min(r_{1y}, r_{2y}))$     $\triangleright$  Move to bottom
- 11:  $r_1 \leftarrow r_1 + m, r_2 \leftarrow r_2 + m$     $\triangleright$  Apply move
- 12: **if**  $\Delta e_x - (r_{1x} - r_{2x}) > 0$  **then**
- 13:    $m \leftarrow (\min(\|\Delta e_x - \Delta s_x\|, L - r_{1x}), 0)$     $\triangleright$  Move right
- 14: **else**
- 15:    $m \leftarrow (-\min(\|\Delta e_x - \Delta s_x\|, r_{1x}), 0)$     $\triangleright$  Move left
- 16: **end if**
- 17:  $m \leftarrow m + (0, \epsilon)$     $\triangleright$  Move up
- 18:  $r_1 \leftarrow r_1 + m, r_2 \leftarrow r_2 + m$     $\triangleright$  Apply move
- 19:  $\Delta r_x = r_{1x} - r_{2x}$
- 20: **if**  $\Delta r_x \equiv \Delta e_x$  **then**
- 21:    $m \leftarrow (e_{1x} - r_{1x}, e_{1y} - r_{1y})$
- 22:    $r_1 \leftarrow r_1 + m, r_2 \leftarrow r_2 + m$     $\triangleright$  Apply move
- 23: **return** ( $r_1, r_2$ )
- 24: **else**
- 25:   **return** GenerateDesiredx-spacing( $r_1, r_2, e_1, e_2, L$ )
- 26: **end if**

---

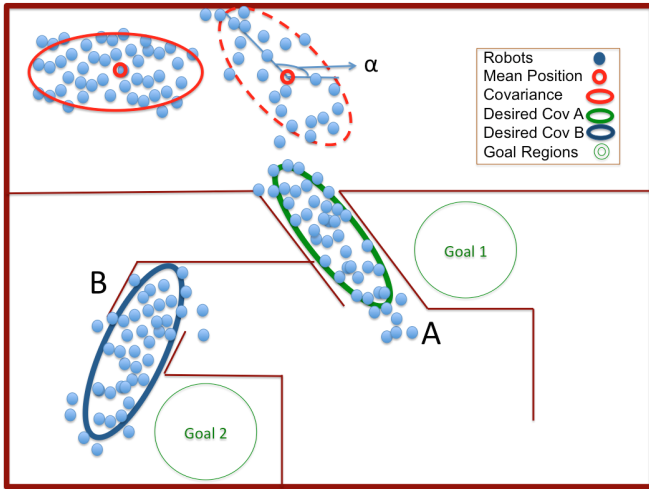


Fig. 3. Maintaining group cohesion while steering a swarm through an arbitrary maze requires covariance control.

---

**Algorithm 3** GenerateDesiredy-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:    $m \leftarrow (L - \max(r_{1y}, r_{2y}), 0)$     $\triangleright$  Move to top wall
- 6: **else**
- 7:    $m \leftarrow (-\min(r_{1y}, r_{2y}), 0)$     $\triangleright$  Move to bottom wall
- 8: **end if**
- 9:  $m \leftarrow m + (0, -\min(r_{1x}, r_{2x}))$     $\triangleright$  Move to left
- 10:  $r_1 \leftarrow r_1 + m, r_2 \leftarrow r_2 + m$     $\triangleright$  Apply move
- 11: **if**  $\Delta e_y - (r_{1y} - r_{2y}) > 0$  **then**
- 12:    $m \leftarrow (\min(\|\Delta e_y - \Delta s_y\|, L - r_{1y}), 0)$     $\triangleright$  Move top
- 13: **else**
- 14:    $m \leftarrow (-\min(\|\Delta e_y - \Delta s_y\|, r_{1y}), 0)$     $\triangleright$  Move bottom
- 15: **end if**
- 16:  $m \leftarrow m + (0, \epsilon)$     $\triangleright$  Move right
- 17:  $r_1 \leftarrow r_1 + m, r_2 \leftarrow r_2 + m$     $\triangleright$  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:    $r_1 \leftarrow r_1 + m, r_2 \leftarrow r_2 + m$     $\triangleright$  Apply move
- 22: **return** ( $r_1, r_2$ )
- 23: **else**
- 24:   **return** GenerateDesiredy-spacing( $r_1, r_2, e_1, e_2, L$ )
- 25: **end if**

---

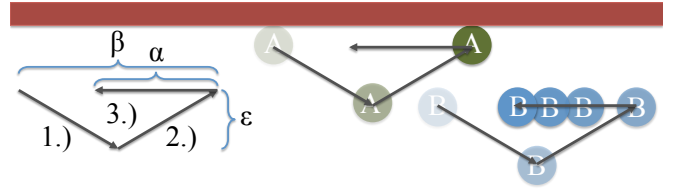


Fig. 4. A DriftMove( $\alpha, \beta, \epsilon$ ) consists of repeating a triangular movement sequence  $\{(\beta/2, -\epsilon), (\beta/2, \epsilon), (-\alpha, 0)\}$ . The robot A touching a top wall will move right  $\beta$  units, while robots not touching the top wall move right  $\beta - \alpha$ .

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  $\epsilon$  space above the build zone,  $\epsilon$  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. 4. A drift move consists of repeating a triangular movement sequence



Fig. 5. Illustration of algorithm for position control of  $n$  robots using wall friction.

$\{(\beta/2, -\epsilon), (\beta/2, \epsilon), (-\alpha, 0)\}$ . The robot  $A$  touching a top wall moves right  $\beta$  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  $k$ th robot, and  $f_k$  the final  $x, y$  position of the  $k$ th 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.

---

**Algorithm 4** PositionControl $n$ RobotsUsingWallFriction( $k$ )

---

```

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

```

---

Algorithm 4 proceeds as follows: First, the robots are moved away from right wall and down so robot  $k$  touches bottom. Second, a set of DriftMove()s are executed that move the  $k$  robot to the left wall with no net movement on the other robots. Third, a set of DriftMove()s are executed that move the  $k$  robot to it's target height and return the other robots to their initial heights. Fourth, all robots except the  $k$  robot are pushed left until the  $k$  robot 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.

## VI. SIMULATION

Algorithms 1, 2, 3, were implemented in Mathematica using point robots (radius = 0). All code is available at (INSERT). Two figure show...[describe].

## VII. EXPERIMENT

### A. Hardware system

Our experiments are on centimeter-scale hardware systems. It 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 [3], [4] 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 [5]. 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*,

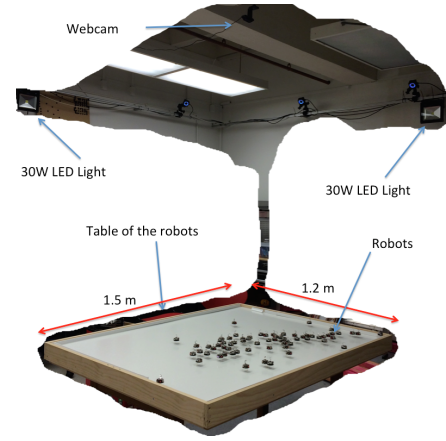
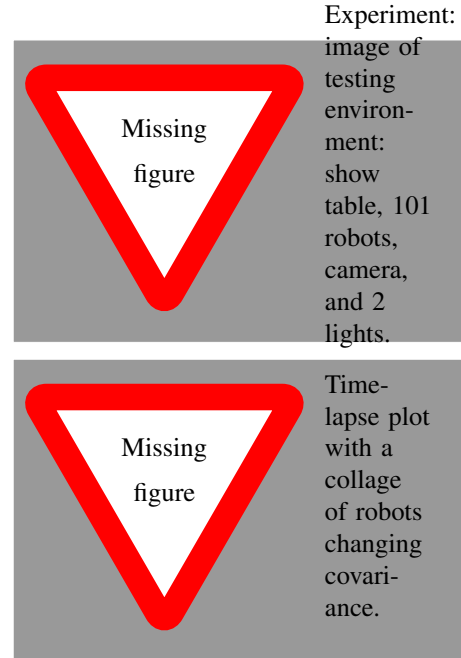


Fig. 9. Our workspace with a table of  $1.5 \times 1.2$  m and four 30W LED floodlights and an overhead machine vision system.

moving towards a light source. In these experiments as shown in Fig. 9, we used  $n=68$  kilobots, a  $1.5 \text{ m} \times 1.2 \text{ m}$  whiteboard as the workspace, and four 30W LED floodlights arranged at the  $\{N, E, S, W\}$  vertices of a 6 m square centered on the workspace. The lights were controlled by using an Arduino Uno board connected to an 8 relay shield board. Also at top of the table, an overhead machine vision system was added to track the position of the swarm.



## REFERENCES

- [1] A. Becker, G. Habibi, J. Werfel, M. 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)*, Nov. 2013, pp. 520–527.
- [2] S. Shahrokhi and A. T. Becker, "Stochastic swarm control with global inputs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, p. tbd.

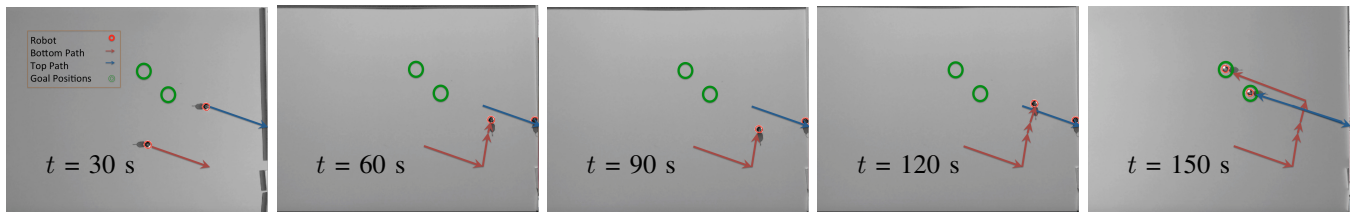


Fig. 6. Two robot positioning with using infinite friction for walls

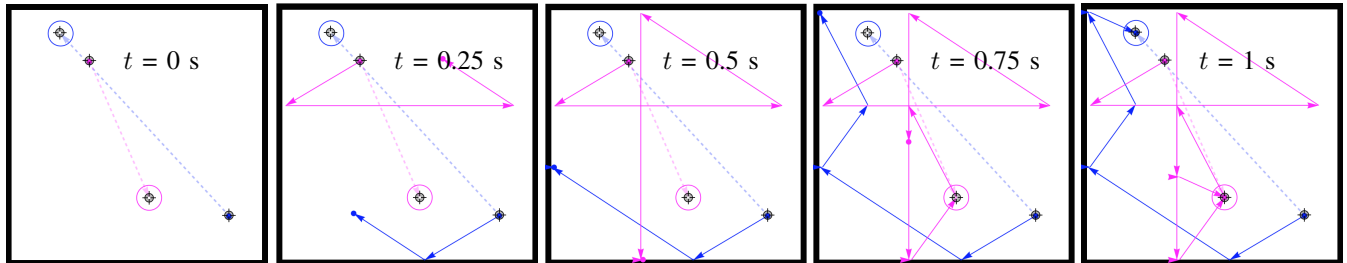


Fig. 7. Two robot positioning with using walls with infinite friction

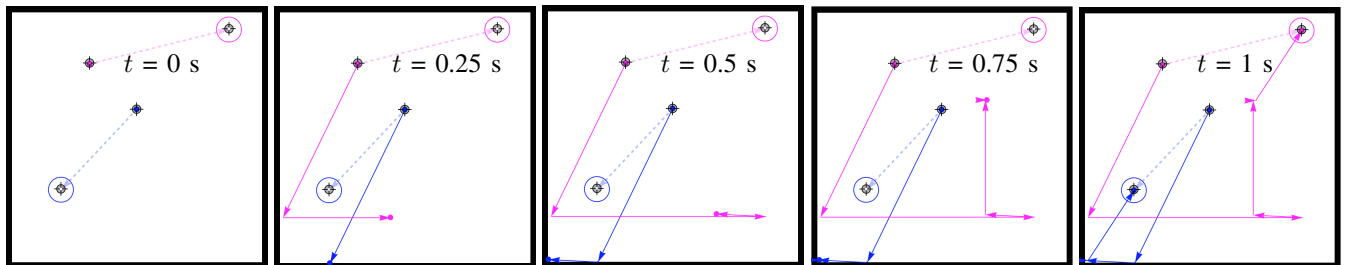


Fig. 8. Two robot positioning: switching positions. with using walls with infinite friction Code available at github address

- [3] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *IEEE Int. Conf. Rob. Aut.*, May 2012, pp. 3293–3298.
- [4] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.
- [5] K-Team, "Kilobot, [www.k-team.com/mobile-robotics-products/kilobot](http://www.k-team.com/mobile-robotics-products/kilobot)," 2015.