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

Shiva Shahrokhi and Aaron T. Becker

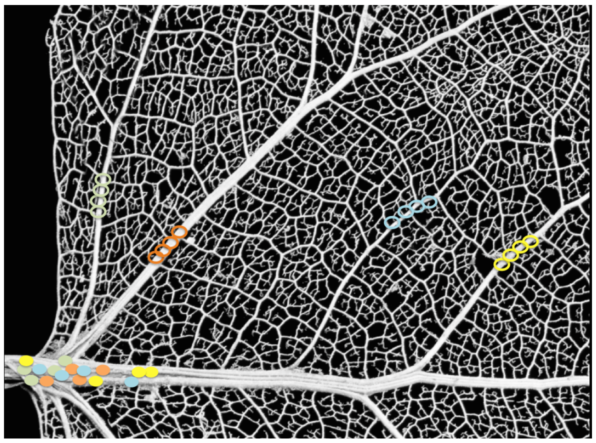


Fig. 1. Several real vascular networks exist in real world, which navigating a swarm with a global input would be a challenge because of the branches in the way.

***Abstract*— Micro- and nano-robots can be manufactured in large numbers. 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. In our previous work we focused on a block-pushing task, where a swarm of robots pushed a larger block through a 2D maze. One surprising result was that humans that only knew the swarm's mean and covariance completed the task faster that humans who knew the position of every robot [?]. Inspired by that work, we proved that we can control the mean position of a swarm and that with an obstacle we can control the swarm's position variance ($\sigma_x$ and $\sigma_y$). We then wrote automatic controllers which could complete a block pushing task, but these controllers had some limitations [?]. 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 $\sigma_{xy}$.**

## I. Introduction

## II. Theory

## III. Position Control of 2 robots using wall friction

In this section we focus on describing the algorithm we have developed for positioning the robots. Although the generalization of the proposed positioning algorithm in here is rather straightforward for multi robot systems, for the sake of simplicity we describe the algorithm designed for two robots. Therefore, we assume that we have two robots located at $s_1$ and $s_2$ and we aim to move them toward their corresponding destinations $e_1$ and $e_2$, respectively. We denote the current positions of the robots by $r_1$ and $r_2$. In addition, we use subscripts $x$ and $y$ to denote the $x$ and $y$ coordinates, i.e., $s_{1x}$ and $s_{1y}$ denote the $x$ and $y$ locations of $s_1$, respectively. As it is clear from swarm setting, our algorithm works with one global 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 our algorithm is employing the assumption we have made earlier about the walls' friction.

Our algorithm uses divide and conquer method to solve the positioning problem. It finds the final position of the robots in two steps: (i) First, our algorithm adjusts $\Delta r_x$ while it keeps $\Delta r_y$ constant. (ii) Having fixed $\Delta r_x$ to $\Delta e_x$ as desired, our algorithm keeps $\Delta r_x$ constant and adjusts $\Delta r_y$ to $\Delta e_y$, as desired. Although both of the steps (i) and (ii) are similar to each other from the algorithmic point of view, we give a detailed description of each of them in the following subsections.

### *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.

S. Shahrokhi and A. Becker are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204-4005 USA `{sshahrokhi2, atbecker}@uh.edu`

- 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 III-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 **??**.
- 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 $e_2$.

## IV. Position Control of $n$ Robots using Wall Friction

Algorithm 4 can be extended to control the position of $n$ robots using wall friction under several constraints. Assume an open workspace with four axis-aligned walls

**Algorithm 1** 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:     $m \leftarrow (e_{1x} - r_{1x}, e_{1y} - r_{1y})$
22:     $r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$    ▷ Apply move
23:     return $(r_1, r_2)$
24: else
25:     return GenerateDesired$x$-spacing$(r_1, r_2, e_1, e_2, L)$
26: end if
```

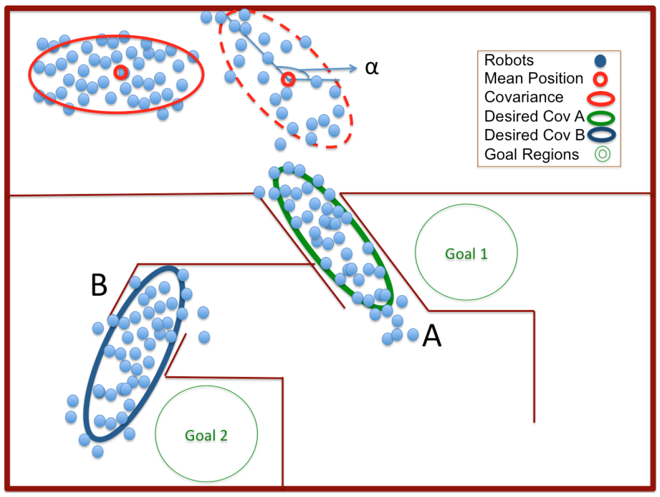


Fig. 2. We can control covariance of the swarm by using friction.

**Algorithm 2** 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 3** 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
```

Fig. 3. Illustration of algorithm for position control of $n$ robots using wall friction.

with infinite friction. The axis-aligned rectangle of dimension $(w_f, h_f)$ containing the final configuration of $n$ robots must be disjoint from the axis-aligned rectangle of dimension $(w_s, h_s)$ containing the starting configuration of $n$ robots. Without loss of generality, assume the final configuration is above the starting configuration. Furthermore, there must be at least $\epsilon$ space above the final configuration, $\epsilon$ below the starting configuration, and $\epsilon + w_r$ to the left of the final and start configurations, where $w_r$ is the width of a robot. The workspace is at least $(\epsilon + w_r + \max(w_f, w_s), 2\epsilon + h_s, h_f)$.

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.

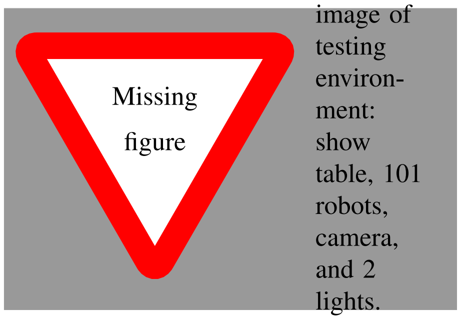
**Algorithm 4** PositionControl$n$RobotsUsingWallFriction$(k)$

```
1: move( $-\epsilon, -r_{k,y}$)
2: drift move left until $r_k \equiv (0, 0)$
3: drift move up until $r_{ky} \equiv f_{ky}$
        $f_{kx} - f_{(k-1)x}$ $p_{kx} - p_{(k-1)x}$
4: move ( ,0)
```

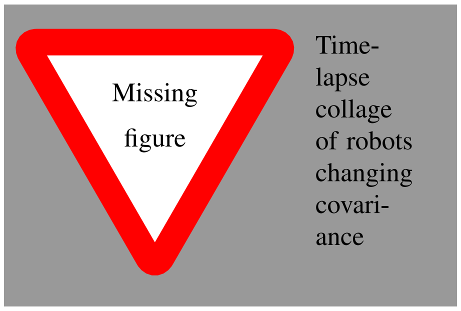
¡¡¡¡¡¡¡ Updated upstream ======= ¿¿¿¿¿¿¿ Stashed changes

## V. Simulation

ALgorithms NUM NUM were implemented in Mathematica using point robots (radius = 0). All code is available at (INSERT). Two figure show...[describe].



Experiment: image of testing environment: show table, 101 robots, camera, and 2 lights.



Time-lapse collage of robots changing covariance

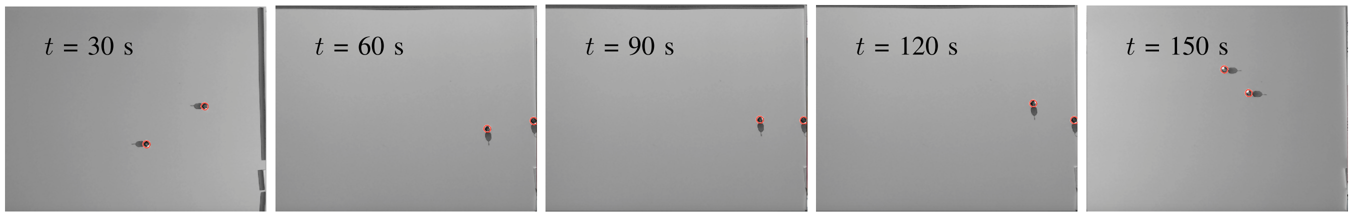


Fig. 4. Two robot positioning with using infinite friction for walls

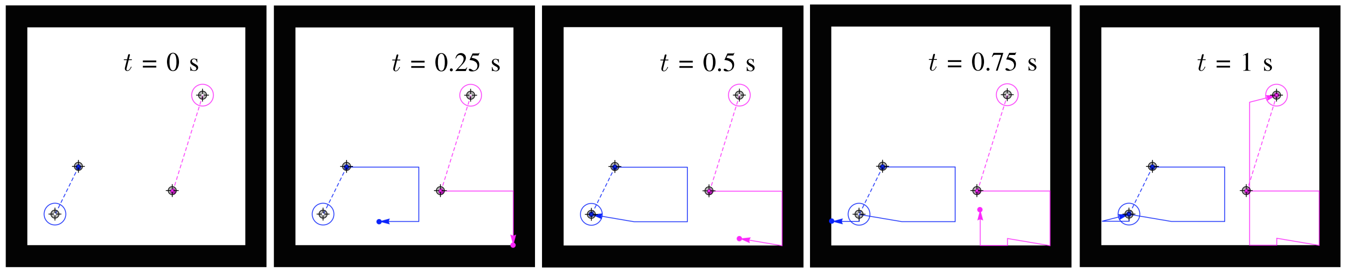


Fig. 5. Two robot positioning with using infinite friction for walls

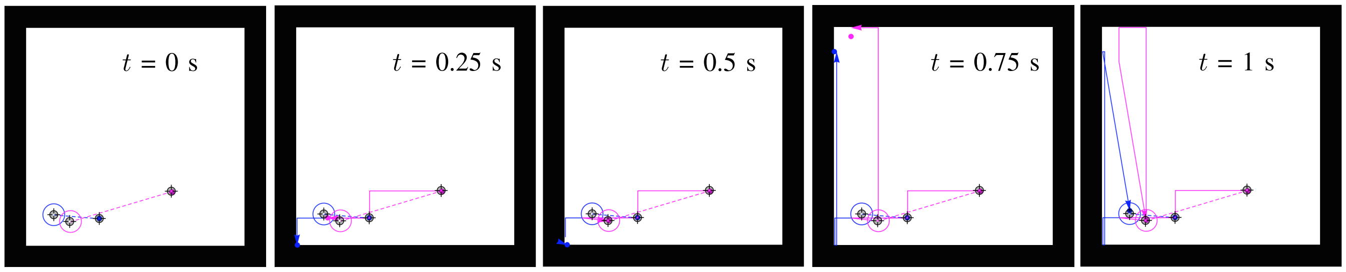
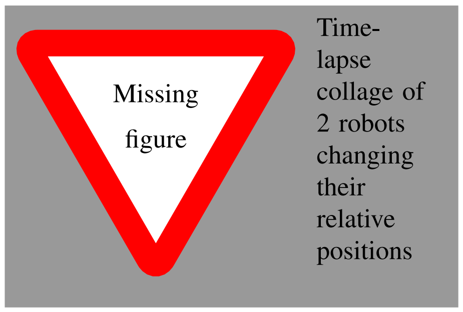


Fig. 6. Two robot positioning with using infinite friction for walls: switching positions. Code available at github address


Missing
figure
Time-lapse collage of 2 robots changing their relative positions