

Shaping a Swarm Using Wall Friction and a Shared Control Input

Shiva Shahrokhi and Aaron T. Becker

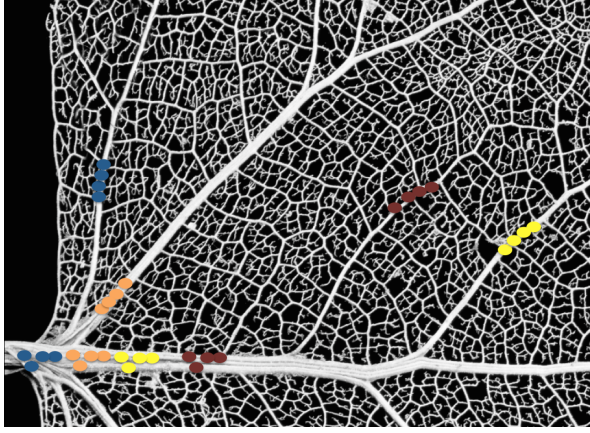


Fig. 1. Several real vascular networks exist in real world, which navigating a swarm with a global input would be a challenge because of the branches in the way.

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 than 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 (σ_x and σ_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, σ_{xy} . For controlling σ_{xy} , we prove that the swarm position covariance σ_{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 σ_{xy} .

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 position of the robots are (r_1, r_2) .

Ensure: $r_{1y} - r_{2y} \equiv s_{1y} - s_{2y}$

```

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

I. INTRODUCTION

II. THEORY

III. 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 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.

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

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:    $m \leftarrow (L - \max(r_{1y}, r_{2y}), 0)$   $\triangleright$  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 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

```

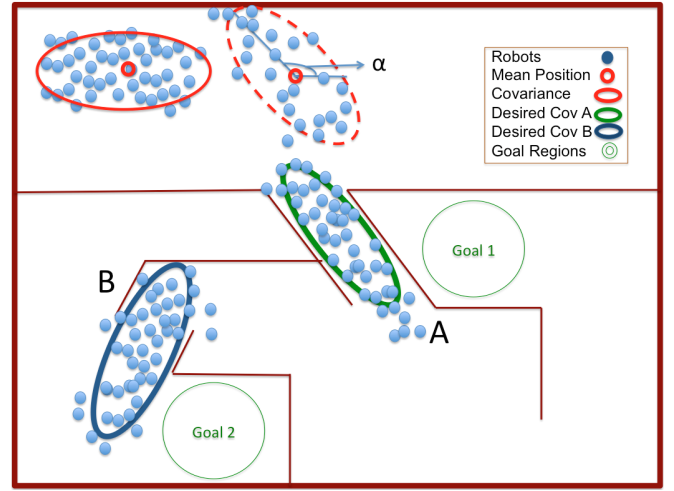


Fig. 2. We can control covariance of the swarm by using friction.

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

Without loss of generality, assume the final configuration is above the starting configuration. Furthermore, there must be at least ϵ space above the final configuration, ϵ 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)

```

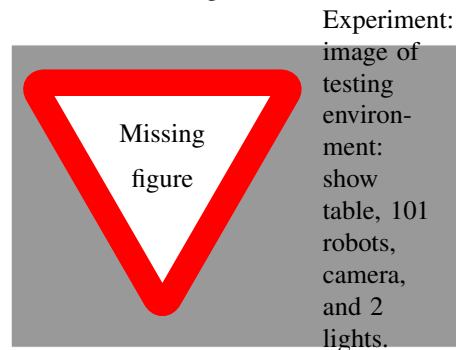
1: move(  $-\epsilon, -r_{k,y}$ )
2: drift move left until  $r_k \equiv (0, 0)$ 
3: drift move up until  $r_{ky} \equiv f_{ky}$ 
    $f_{kx} - f_{(k-1)x} \quad p_{kx} - p_{(k-1)x}$ 
4: move (  $, 0$ )

```

|||||| Updated upstream ===== <<<<<<< Stashed changes

IV. 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.

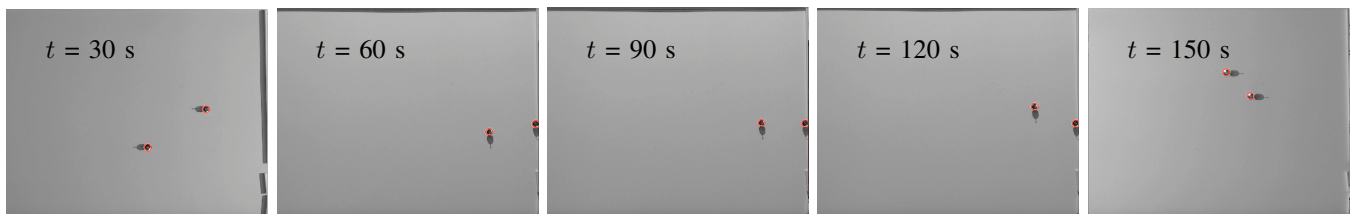


Fig. 4. Two robot positioning with using infinite friction for walls

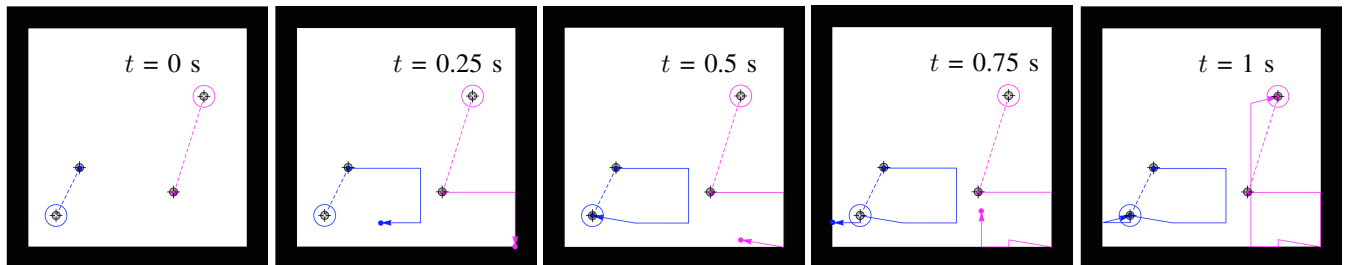


Fig. 5. Two robot positioning with using infinite friction for walls

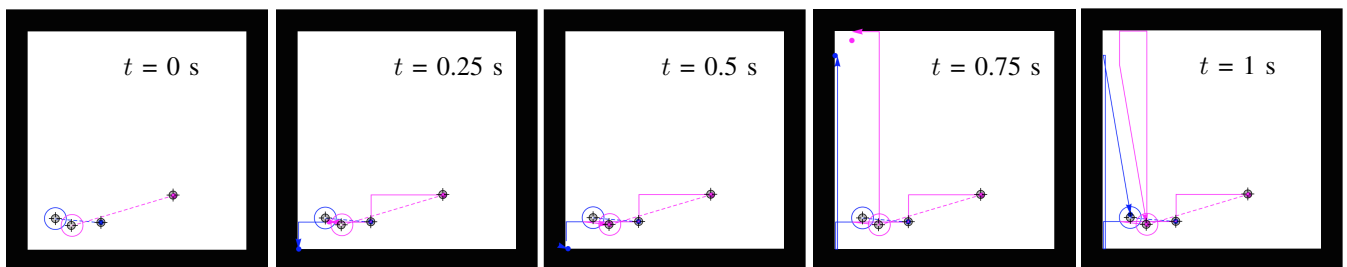


Fig. 6. Two robot positioning with using infinite friction for walls: switching positions. Code available at [github address](#)

