Shaping a Swarm Using Wall Friction and a Shared Control Input

Shiva Shahrokhi and Aaron T. Becker

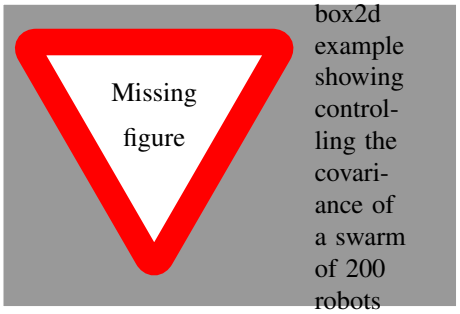
***Abstract*— Micro- and nano-robots can be manufactured in large numbers. Large numbers of micro robots are required in order to deliver sufficient payloads, but the small size of these robots makes it difficult to perform onboard computation. Instead, these robots are often controlled by a global, 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 (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}.**

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.

I. Introduction

II. Theory

III. Simulation



Simulation: box2d example showing controlling the covariance of a swarm of 200 robots

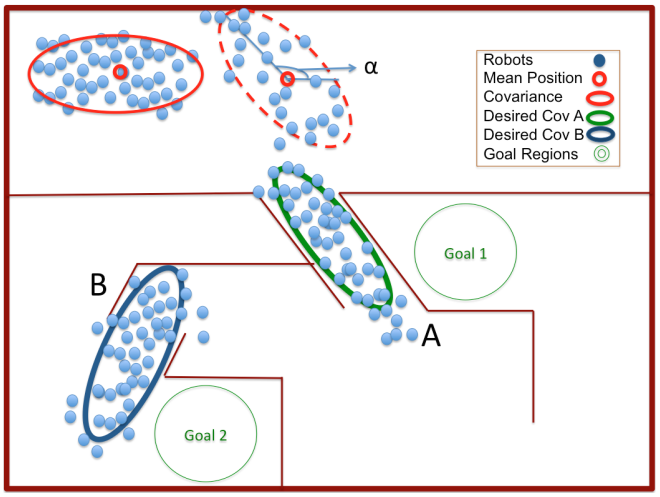


Fig. 2. We can control covariance of the swarm by using friction.

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

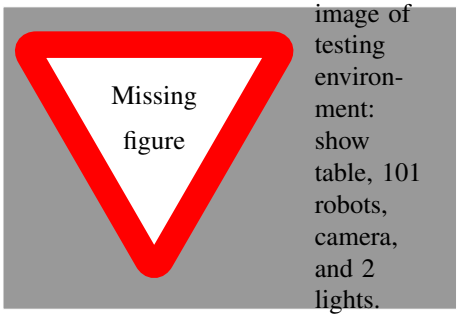
$\Delta s_x \leftarrow s_{1x} - s_{2x}$
$\Delta e_x \leftarrow e_{1x} - e_{2x}$
$r_1 \leftarrow s_1$, $r_2 \leftarrow s_2$
if $\Delta e_x < 0$ **then**
    $m \leftarrow (L - \max(r_{1x}, r_{2x}), 0)$ ▷ Move to right wall
else
    $m \leftarrow (-\min(r_{1x}, r_{2x}), 0)$ ▷ Move to left wall
end if
$m \leftarrow m + (0, -\min(r_{1y}, r_{2y}))$ ▷ 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**
    $m \leftarrow (e_{1x} - r_{1x}, e_{1y} - r_{1y})$
    $r_1 \leftarrow r_1 + m$, $r_2 \leftarrow r_2 + m$ ▷ Apply move
    return (r_1, r_2)
else
    return GenerateDesired$x$-spacing(r_1, r_2, e_1, e_2, L)
end if

Algorithm 2 Getting desired Y-space

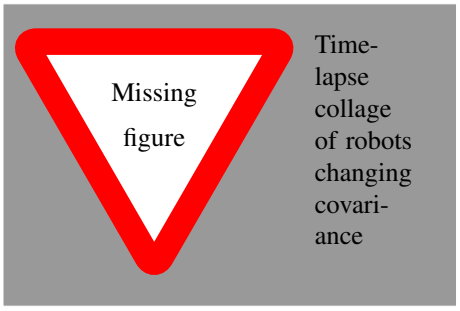
Require: Knowledge of the starting and ending positions of the two robots s_1 (right robot) & s_2 (left robot) & e_1 & e_2 and L length of the walls. Current position of the robot will be showed as r.

Ensure: $\Delta x(t) = \Delta x(0)$

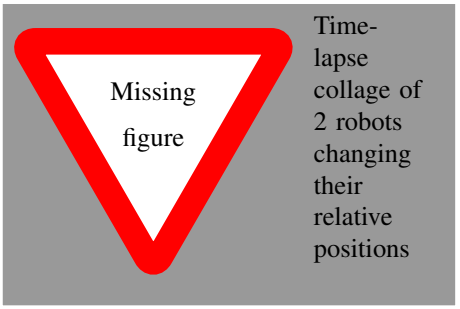
$e_{1y} - e_{2y} = \Delta e_y$
$s_{1y} - s_{2y} = \Delta s_y$
if $\Delta e < 0$ **then**
    Move almost to up wall
else
Move almost to down wall
end if
Move to right
if $\Delta e_y - \Delta s_y > 0$ **then**
    Update current position of the robots: r_1 , r_2
    Go right for $min(\|\Delta e_y - \Delta s_y\|, L - r_{1y})$
else
Go left for $-min(\|\Delta e - \Delta s_y\|, r_{1y})$
end if
Move ϵ left
Update current position of the robots: r_1 , r_2
$\Delta r_y = r_{1y} - r_{2y}$
if $\Delta r_y = \Delta e_y$ **then**
    Go to e_1, e_2
    Return
else
Do the algorithm again with the inputs of r_1, r_2, e_1, e_2
end if



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



Time-lapse collage of robots changing covariance



Time-lapse collage of 2 robots changing their relative positions

Algorithm 3 Getting desired Space

Require: Knowledge of the starting and ending positions of the two robots s_1 & s_2 & e_1 & e_2 and L length of the walls. Current position of the robot will be showed as r.

$e_{1x} - e_{2x} = \Delta e_x$
$e_{1y} - e_{2y} = \Delta e_y$
$s_{1x} - s_{2x} = \Delta s_x$
$s_{1y} - s_{2y} = \Delta s_y$
if $\Delta s_x < \Delta s_y$ **then**
    Do XSpace
    Do YSpace
else
    Do YSpace
    Do XSpace
end if

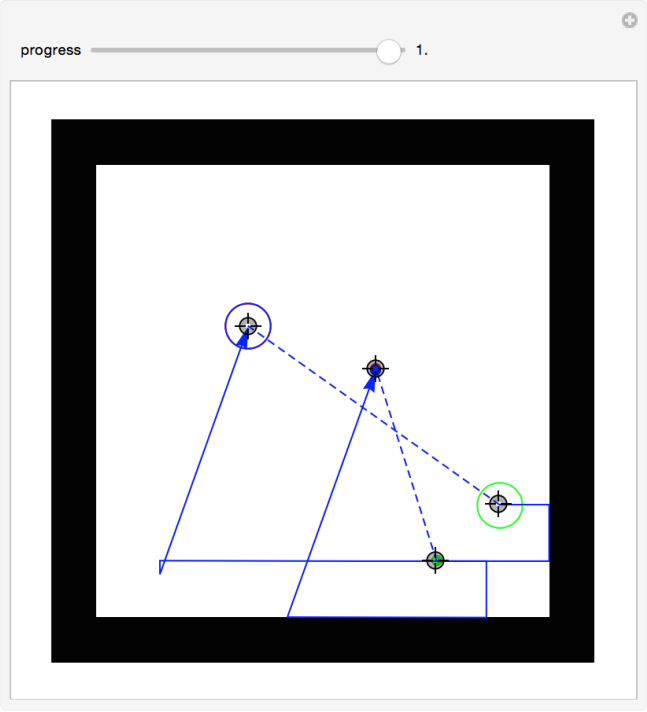


Fig. 3. Using friction to move two robots from the starting point to their end point.