

Algorithms For Shaping a Particle Swarm With a Shared Control Input Using Boundary Interaction*

Shiva Shahrokhi, Arun Mahadev, and Aaron T. Becker

Abstract—There are driving applications for large populations of tiny robots in robotics, biology, and chemistry. These robots often lack onboard computation, actuation, and communication. Instead, these “robots” are particles carrying some payload and the particle swarm is controlled by a shared, global control input. In previous work, we showed that the 2D position of each particle in such a swarm is controllable if the workspace contains a single obstacle the size of one particle.

Requiring a small, rigid obstacle suspended in the middle of the workspace is a strong constraint, especially in 3D. This paper relaxes this constraint, and provides position control algorithms that only require interactions with the boundaries. Both in vivo and artificial environments often have boundaries. We assume that particles in contact with the boundaries have zero velocity if the global control input pushes the particle into the wall. This paper provides an optimal shortest-path algorithm for a swarm with two particles, and a generalization to n -particle swarms. Results are validated with simulations and a hardware demonstration.

I. INTRODUCTION

Particle swarms propelled by a global field are common in applied mathematics, biology, and computer graphics. As a current example, micro- and nano-robots can be manufactured in large numbers, see [1]–[2]. Someday large swarms of robots will be remotely guided to assemble structures in parallel and through the human body to cure disease, heal tissue, and prevent infection. For each task, 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. They require techniques to shape the swarm that can reliably exploit large populations despite severe underactuation ($2 \ll 2n$).

Swarm shape control is necessary for navigation through narrow passages, for self-assembly, and for manipulating objects.

The paper is arranged as follows. After a review of recent related work in Sec. II, Sec. III introduces general models for boundary interaction. We introduce algorithms with two orthogonal boundaries with high friction sufficient to arbitrarily position two robots in Sec. IV, and Sec. V extends this to prove a rectangular workspace with high-friction boundaries can position a swarm of n robots arbitrarily within a subset of the workspace. Sec. VI describes implementations of the algorithms in simulation and Sec. VII describes hardware

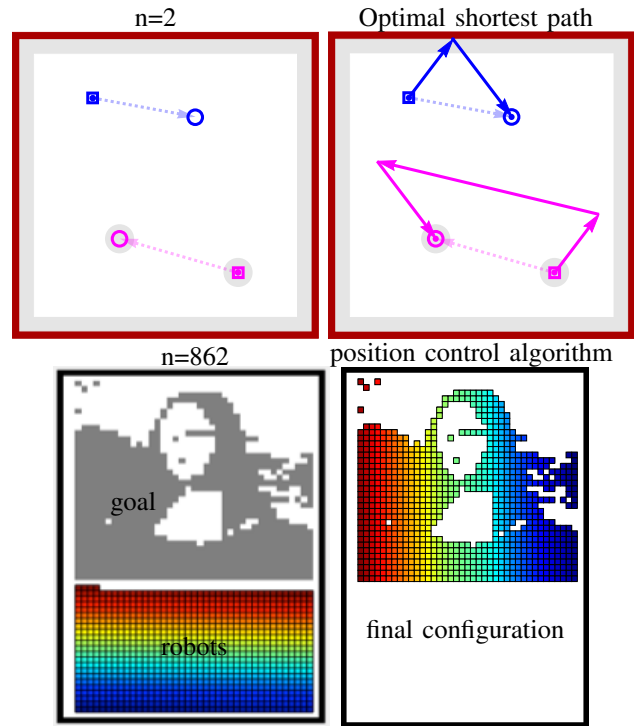


Fig. 1. Dynamic algorithm to find the shortest path for two robots is designed. The two top pictures show the initial position and goal position of the robots and the optimal shortest path by making use of wall friction when shared control input is applied. The two bottom pictures show the n robot positioning using shared control input and boundary walls.

experiments, as shown in Fig. 1. We end with directions for future research in Sec. VIII.

II. RELATED WORK

Controlling the *shape*, or relative positions, of a swarm of robots is a key ability for a range of applications. Correspondingly, it has been studied from a control-theoretic perspective in both centralized and decentralized approaches. For examples of each, see the centralized virtual leaders in [3], and the gradient-based decentralized controllers using control-Lyapunov functions in [4]. However, these approaches assume a level of intelligence and autonomy in individual robots that exceeds the capabilities of many systems, including current micro- and nano-robots. Current micro- and nano-robots, such as those in [1, 5, 6] lack onboard computation.

Instead, this paper focuses on centralized techniques that apply the same control input to each member of the swarm. Precision control requires breaking the symmetry caused

*This work was supported by the National Science Foundation under Grant No. [IIS-1553063] and [IIS-1619278].

Authors are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204 USA {sshahrokhi2, aviswanathanmahadev, atbecker}@uh.edu

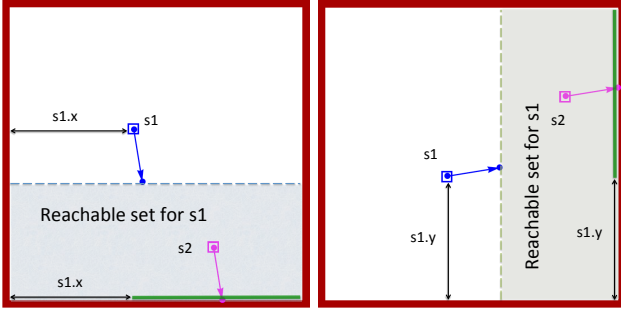


Fig. 2. Each robot gets the same control input. (left) If robot 2 hits the bottom wall before robot 1 reaches a wall, robot 2 can reach anywhere along the green line, and robot 1 can move to anywhere in the shaded area. (right) Similarly, if robot 2 hits the right wall before robot 1 reaches a wall, robot 2 can reach anywhere along the green line, and robot 1 can move to anywhere in the shaded area.

by the global input. Symmetry can be broken using agents that respond differently to the global control, either through agent-agent reactions, see work modeling biological swarms [7], or engineered inhomogeneity [8, 9, 10]. This work assumes a uniform control with homogenous agents, as in [11]. The techniques in this paper are inspired by artificial force-fields.

Artificial Force-fields: Much research has focused on generating non-uniform artificial force-fields that can be used to rearrange passive components. Applications have included techniques to design shear forces for sensorless manipulation of a single object by [12]. [13, 14] demonstrated a collection of 2D force fields generated by 6DOF vibration inputs to a rigid plate. These force fields, including shear forces, could be used as a set of primitives for motion control to steer the formation of multiple objects. However unlike the uniform control model in this paper, theirs was multi-modal and position-dependent.

III. THEORY

Using Boundaries: Friction and Boundary Layers

Global inputs move a swarm uniformly. Shape control requires breaking this uniform symmetry.

The following sections examine using boundaries that stop the particles if they are pushed into the wall. These forces are sufficient to break the symmetry caused by uniform inputs.

IV. POSITION CONTROL OF TWO ROBOTS USING WALL FRICTION

Alg. 1 uses wall-friction to arbitrarily position two robots in a rectangular workspace. This algorithm introduces concepts that will be used for multi-robot positioning. Fig. 5 shows a Mathematica implementation of the algorithm, and is useful as a visual reference for the following description.

Assume two robots are initialized at s_1 and s_2 with corresponding goal destinations g_1 and g_2 . We can exploit symmetry in the solution by labeling the leftmost (or, if they have the same x coordinate, the topmost) robot s_1 . If s_1 is not also the topmost robot, we mirror the coordinate frame by the right wall. To give an example to understand mirroring, consider the two starting position as $s_1 = (0.2, 0.2)$ and

$s_2 = (0.8, 0.8)$. Because the leftmost robot is not the topmost robot, we mirror the coordinate frame and by mirroring we will have new $s_1 = (0.2, 0.8)$ and new $s_2 = (0.8, 0.2)$. We do this to reuse the main algorithm's logic and after the path is found, we unmirror the coordinate frame and translate the path.

Denote the current positions of the robots r_1 and r_2 . Values $.x$ and $.y$ denote the x and y coordinates, i.e., $s_1.x$ and $s_1.y$ denote the x and y locations of s_1 . The algorithm assigns a global control input at every instance. The goal is to adjust the error in both coordinates, $\Delta e = (\Delta e_x, \Delta e_y) = (g_2 - g_1) - (s_2 - s_1)$ to $(0, 0)$ using a shared global control input. Our algorithm uses a dynamic search A* like method to find the shortest path in each move. Base case occurs when $\Delta e = (0, 0)$. In this base case, simply the shortest path is a straight line between current position of each robot to its goal position. If $\Delta e \neq (0, 0)$ then we consider the reachable set as shown in Fig. 3. If a robot is touching a wall, the other robot can go where ever in the reachable set while the first robot is not moving. Two reachable sets are possible, horizontal and vertical.

In the worse case, adjusting both Δr_x and Δr_y needs all four steps. If we consider the length of each side of the square workspace as L , the worst case path length is $(\sqrt{2} + 2)L$.

Algorithm 1 PoseControl2Robots(s_1, s_2, g_1, g_2, L)

Require: knowledge of starting (s_1, s_2) and goal (g_1, g_2) positions of two robots. $(0, 0)$ is bottom corner, L is length of the walls. Current robot positions are (r_1, r_2) . Assume $s_1.x < s_2.x$ and $s_1.y \geq s_2.y$. If not, rotate workspace coordinates counterclockwise 90° . ϵ is a small, nonzero, user-specified value.

Ensure: $(g_1, g_2), (s_1, s_2)$ all at least ϵ distance from walls.

- 1: $\Delta g_y = e_2.y - e_1.y$
- 2: $\Delta g_x = e_2.x - e_1.x$
- 3: Move $(-r_1.x, -\min(r_2.y, r_1.y + \Delta e_y) + \epsilon)$ \triangleright Touch left wall
- 4: Move $(r_2.x, |\text{sgn}(\Delta e_y)| \min(r_1.y - r_2.y + \Delta e_y - \epsilon, L - r_2.y - \epsilon) + \epsilon)$ \triangleright Adjust y
- 5: Move $(\max(\epsilon, -\text{sgn}(r_2.y - r_1.y)\Delta e_x), -\min(r_1.y, r_2.y))$ \triangleright Touch bottom wall
- 6: Move $(\text{sgn}(r_2.y - r_1.y)\Delta e_x, (|\text{sgn}(\Delta e_y)| - 1)|r_2.y - r_1.y|)$ \triangleright Adjust x
- 7: **if** $\Delta e_y \neq r_2.y - r_1.y$ **then**
- 8: **if** $\Delta e_x \neq 0$ **then**
- 9: Move $(\text{sgn}(r_2.y - r_1.y)\Delta e_x, |\text{sgn}(\Delta e_y) - 1||r_1.y - r_2.y|)$ \triangleright Touch right wall
- 10: **else**
- 11: Move $(|\text{sgn}(\Delta e_y - r_2.y + r_1.y)|\epsilon, |\text{sgn}(\Delta e_y) - 1||r_1.y - r_2.y|)$ \triangleright Touch right wall
- 12: **end if**
- 13: **if** $\Delta e_x < 0$ **then**
- 14: Move $(L - \max(r_1.x, r_2.x), \epsilon)$
- 15: **else**
- 16: Move $(L - \max(r_1.x, r_2.x), \Delta e_y - \max(r_1.y, r_2.y))$ \triangleright Adjust y
- 17: **end if**
- 18: **end if**
- 19: Move $(e_2.x - r_2.x, e_2.y - r_2.y)$ \triangleright Go to goal

V. POSITION CONTROL OF n ROBOTS USING WALL FRICTION

Alg. 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 a *staging zone* to the desired position in a *build zone*. All robots move according to the global input, but due to wall friction, at the end the k th loop, robots 1 through k are in their desired final configuration in

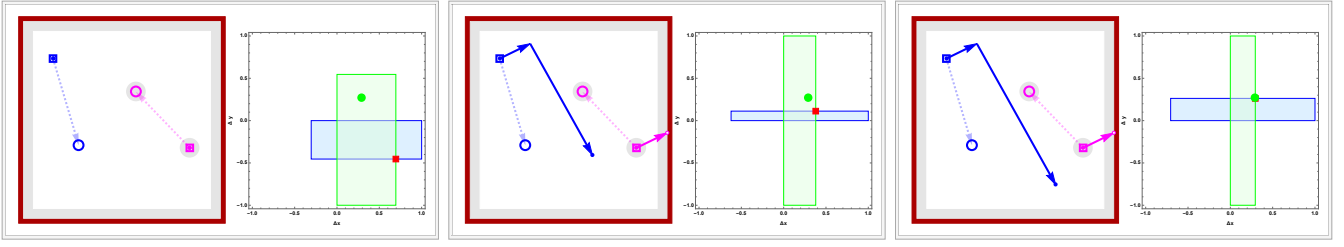


Fig. 3. Workspace and Δ configuration space for three sets of robot configurations with the same final goal. The red square represents the starting Δx and Δy and the green circle represents the goal Δx and Δy . The green rectangle illustrates one move reachable Δx and Δy by horizontal walls and the blue rectangle illustrates the vertical walls reachable region.

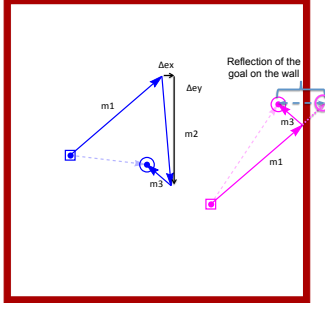


Fig. 4. Consider the reflection of the goal position to the wall that is aimed to be touched by one of the robots. In the right picture, it is shown that the shortest path with three moves is the straight line between the starting position of the robot and the reflection of the goal.

the build zone, and robots $k+1$ to n are in the staging zone. See Fig. 7 for a schematic of the build and staging zones.

The setup in ?? is an assembled practical setup that assumes that $\epsilon = 2r = 1$ cm. It is a 7×7 cm grid space. The wall friction is emulated by the jagged teeth design to keep rollers from moving out of place while implementing the drift move. The rollers are magnetic and can be controlled by magnetic steering. See video “refer the video which shows moving rollers” to see the algorithm at work. In the demonstration, the algorithm is a fixed step algorithm and all drift moves are an counterclockwise square move of size $1\text{cm} \times 1\text{cm}$. Once the k th roller gets to its designated location in every loop, a correction step is implemented. This makes the overall number of moves required for the fixed step algorithm more than that required for the Algorithm 2. ?? shows how there are only 6 stages per roller involved in Algorithm 2. The fixed step algorithm would go through 8 stages per roller as shown in ???. The fixed step algorithm can be found online. One significant difference between Algorithm 2 and the fixed move implementation of it is that, we can achieve higher accuracy in placing rollers in their precise location if Algorithm 2 is used.

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 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 $(\epsilon + 2r + \max(w_b, w_s), 2\epsilon + h_s, h_b)$.

The n robot position control algorithm relies on a $\text{DriftMove}(\alpha, \beta, \epsilon)$ control input, shown in Fig. 10. 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 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 bottom-to-top, and the f_k configurations top-to-bottom and right-to-left as shown in Fig. 7.

Algorithm 2 PositionControl n RobotsUsingWallFriction(k)

```

1: Move(  $-\epsilon, r - p_{ky}$  )
2: while  $p_{kx} > r$  do
3:   DriftMove( $\epsilon, \min(p_{kx} - r, \epsilon), \epsilon, left$ )
4: end while
5:  $m \leftarrow \text{ceil}(\frac{f_{ky} - r}{\epsilon})$ 
6:  $\beta \leftarrow \frac{f_{ky} - r}{m}$ 
7:  $\alpha \leftarrow \beta - \frac{r - p_{ky} - \epsilon}{m}$ 
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 3 DRIFTMOVE($\alpha, \beta, \epsilon, direction$)

```

1: Move(  $-\epsilon, r - p_{ky}$  )
2: Move(  $r + \epsilon - f_{kx}, 0$  )
3: Move(  $f_{kx} - r, 0$  )

```

Alg. 2 proceeds as follows: First, the robots are moved left away from the right wall, and down so robot k touches the bottom wall. Second, a set of DriftMoves are executed that move robot k to the left wall with no net movement of the other robots. Third, a set of DriftMoves 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 $k-1$. Finally, all robots are moved right until robot k is in the desired target position. Running time is $O(n(w + h))$.

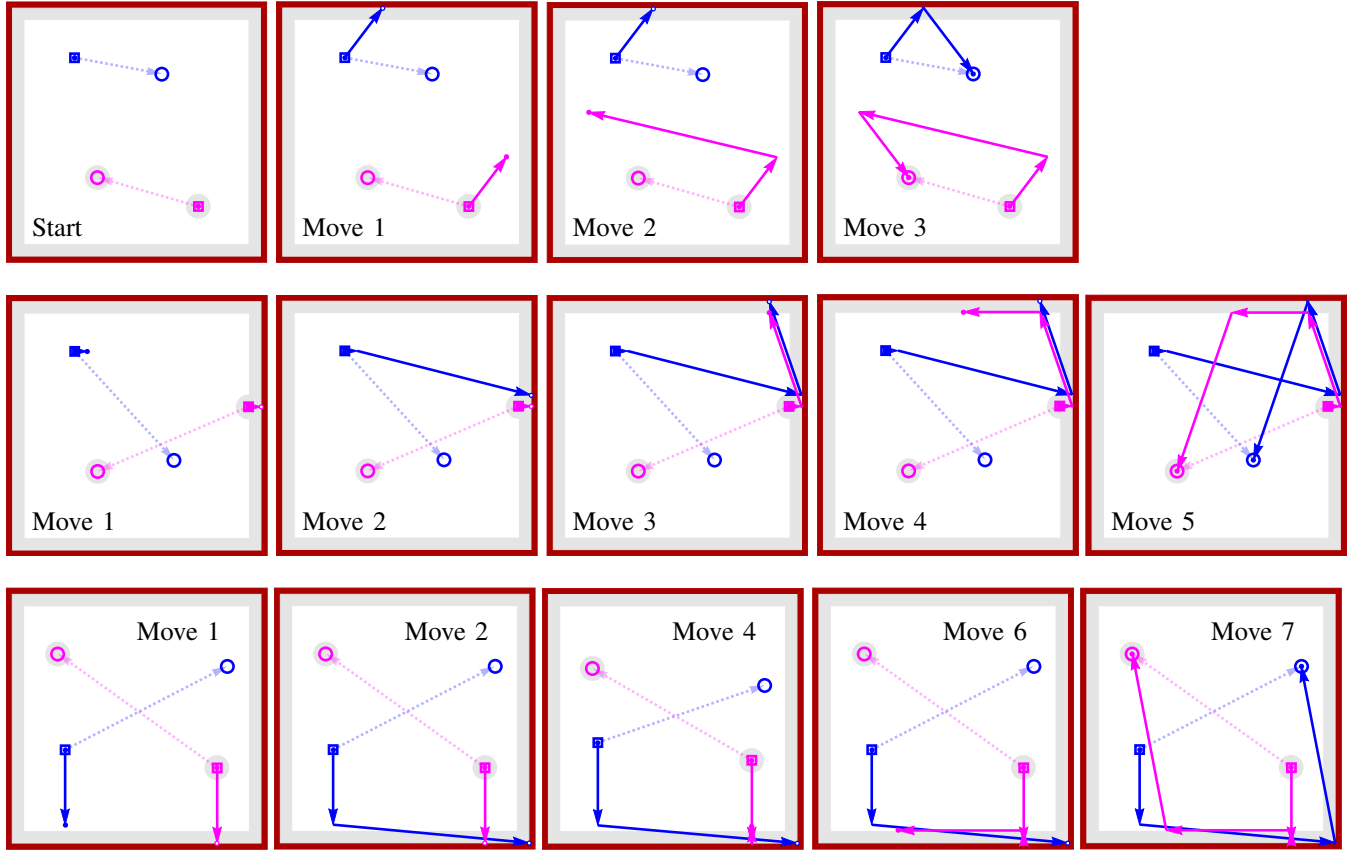


Fig. 5. Frames from an implementation of Alg. 1: two robot positioning using walls with infinite friction. Robot start positions are shown by a square, and goal positions by a circle. Dashed lines show the shortest route if robots could be controlled independently. Solid arrows show path given by Alg. 1. Online demonstration and source code at [15].

VI. SIMULATION

Two simulations were implemented using wall-friction for position control. The first controls the position of two robots, the second controls the position of n robots.

Two additional simulations were performed using wall-friction to control variance and covariance. The first is an open-loop algorithm that demonstrates the effect of varying friction levels. The second uses a closed-loop controller to achieve desired variance and covariance values.

A. Position Control of Two Robots

Algorithm 1 was implemented in Mathematica using point robots (radius = 0). An online interactive demonstration and source code of the algorithm are available at [15]. Fig. 5 shows an implementation of this algorithm with robot initial positions shown by hollow squares and final positions by circles. Dashed lines show the shortest route if robots could be controlled independently, solid lines the generated path.

B. Position Control of n Robots

Alg. 2 was simulated in MATLAB using square block robots with unity width. Code is available at [16]. Simulation results are shown in Fig. 11 for arrangements with an increasing number of robots, $n = [8, 46, 130, 390, 862]$. The distance moved grows quadratically with the number of

robots n . A best-fit line $210n^2 + 1200n - 10,000$ is overlaid by the data.

In Fig. 11, the amount of clearance is $\epsilon = 1$. Control performance is sensitive to the desired clearance. As ϵ increases, the total distance decreases asymptotically, as shown in Fig. 12, because the robots have more room to maneuver and fewer DriftMoves are required.

VII. EXPERIMENT

A. Hardware Experiment: Position Control of n Robots

A hardware setup with a bounded platform, magnetic sliders, and a magnetic guide board was used to implement Alg. 2. Designs for each are available at [17]. The pink boundary is toothed with a white free space, as shown in Fig 7. Only discrete, 1 cm moves in the x and y directions are used. The goal configuration highlighted in the top right corner represents a ‘U’ made of seven sliders. The dark red configuration is the current position of the sliders. Due to the discretized movements allowed by the boundary, drift moves follow a 1 cm square. Free robots return to their start positions but robots on the boundary to move laterally, generating a net sliding motion.

Fig. 7 follows the motion of the sliders through iterations $k=1, 2$, and 7. All robots receive the same control inputs, but boundary interactions break the control symmetry. Robots

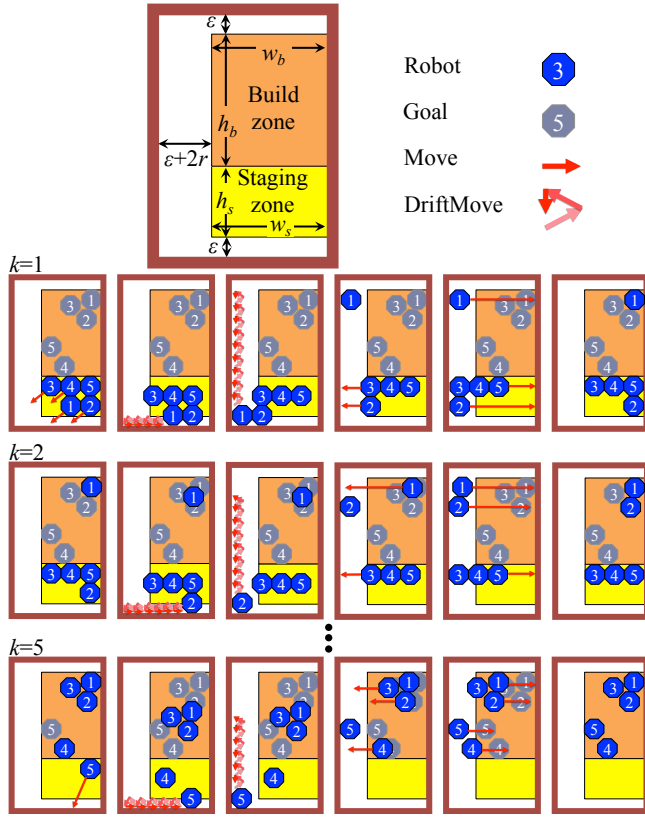


Fig. 6. Illustration of Alg. 2, n robot position control using wall friction.

reach their goal positions in a first-in, first-out arrangement beginning with the bottom-left robot from the staging zone occupying the top-right position of the build zone.

VIII. CONCLUSION AND FUTURE WORK

This paper presented techniques for controlling the shape of a swarm of robots using uniform global inputs and interaction with boundary friction forces. The paper provided algorithms for precise position control, as well as robust and efficient covariance control. Extending algorithms 1 and 2 to 3D is straightforward but increases the complexity. 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 [18]. Additionally, this paper assumed nearly infinite wall friction. The algorithms require retooling to handle small μ_f friction coefficients.

REFERENCES

- [1] S. Chowdhury, W. Jing, and D. J. Cappelleri, "Controlling multiple microrobots: recent progress and future challenges," *Journal of Micro-Bio Robotics*, vol. 10, no. 1-4, pp. 1–11, 2015.
- [2] F. Qiu and B. J. Nelson, "Magnetic helical micro-and nanorobots: Toward their biomedical applications," *Engineering*, vol. 1, no. 1, pp. 21–26, 2015.
- [3] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE Trans. Robotics Automat.*, vol. 17, pp. 947–951, 2001.
- [4] M. A. Hsieh, V. Kumar, and L. Chaimowicz, "Decentralized controllers for shape generation with robotic swarms," *Robotica*, vol. 26, no. 05, pp. 691–701, 2008.

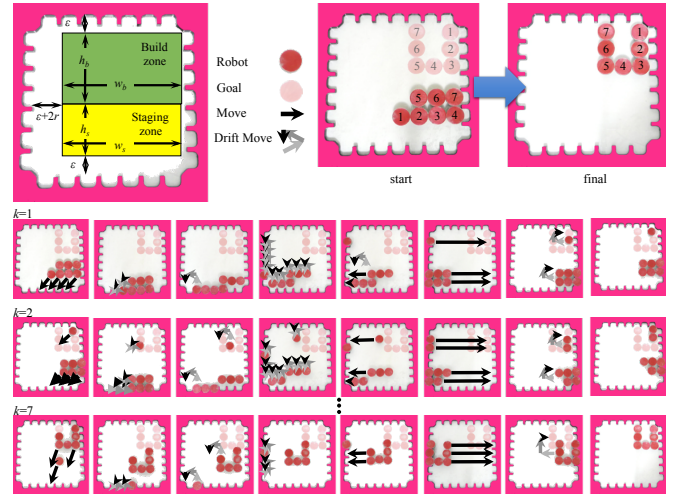


Fig. 7. Illustration of Alg. 2, n robot position control using wall friction.

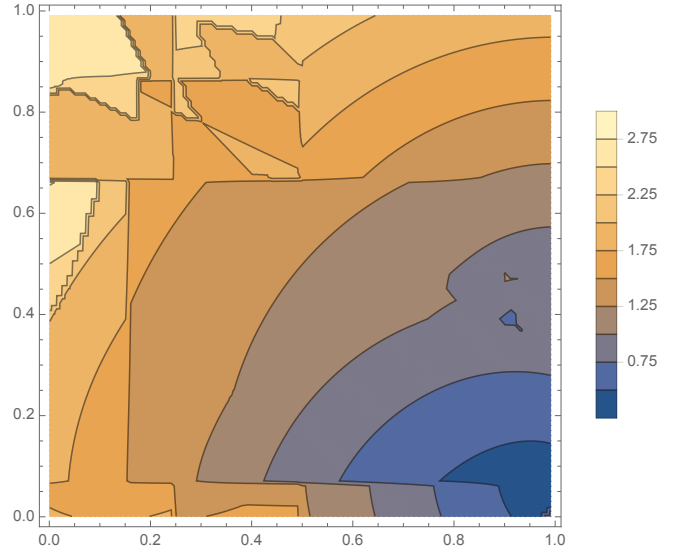


Fig. 8. The contour plot keeps 3 points steady.

- [5] S. Martel, "Magnetotactic bacteria for the manipulation and transport of micro-and nanometer-sized objects," *Micro-and Nanomanipulation Tools*, pp. 308–317, 2015.
- [6] X. Yan, Q. Zhou, J. Yu, T. Xu, Y. Deng, T. Tang, Q. Feng, L. Bian, Y. Zhang, A. Ferreira, and L. Zhang, "Magnetite nanostructured porous hollow helical microswimmers for targeted delivery," *Advanced Functional Materials*, vol. 25, no. 33, pp. 5333–5342, 2015.
- [7] A. L. Bertozzi, T. Kolokolnikov, H. Sun, D. Uminsky, and J. Von Brecht, "Ring patterns and their bifurcations in a nonlocal model of biological swarms," *Communications in Mathematical Sciences*, vol. 13, no. 4, pp. 955–985, 2015.
- [8] B. R. Donald, C. G. Levey, I. Paprotny, and D. Rus, "Planning and control for microassembly of structures composed of stress-engineered mems microrobots," *The International Journal of Robotics Research*, vol. 32, no. 2, pp. 218–246, 2013.
- [9] T. Bretl, "Control of many agents using few instructions," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007, pp. 1–8.
- [10] A. Becker, C. Onyuksel, T. Bretl, and J. McLurkin, "Controlling many differential-drive robots with uniform control inputs," *Int. J. Robot. Res.*, vol. 33, no. 13, pp. 1626–1644, 2014.
- [11] 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*

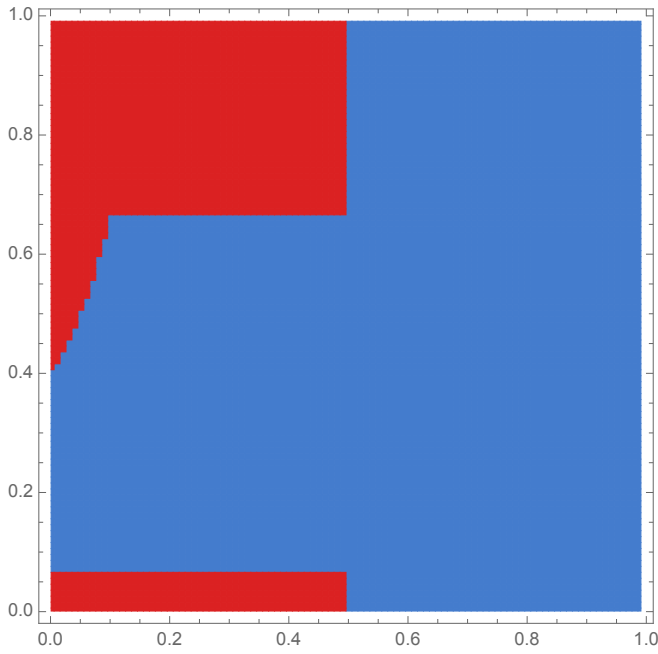


Fig. 9. Number of moves needed.

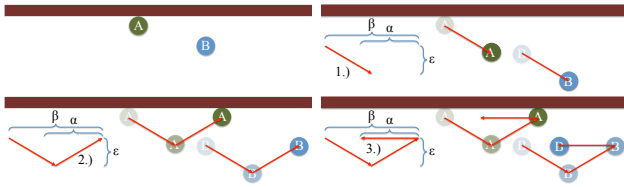


Fig. 10. A $\text{DriftMove}(\alpha, \beta, \epsilon)$ to the right repeats a triangular movement sequence $\{(\beta/2, -\epsilon), (\beta/2, \epsilon), (-\alpha, 0)\}$. Robot A touching a top wall moves right β units, while robots not touching the top move right $\beta - \alpha$.

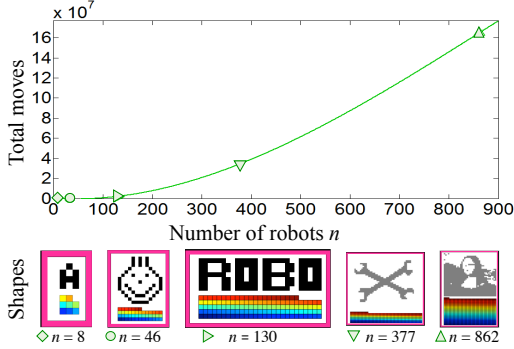


Fig. 11. The required number of moves under Alg. 2 using wall-friction to rearrange n square-shaped robots. See hardware implementation and simulation at [16].

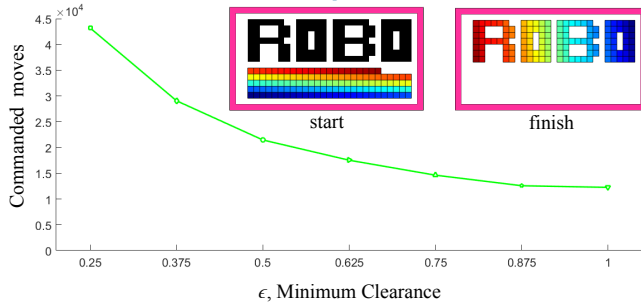


Fig. 12. Control performance is sensitive to the desired clearance ϵ . As ϵ increases, the total distance decreases asymptotically.

- Conference on Intelligent Robots and Systems (IROS)*, Nov. 2013, pp. 520–527.
- [12] F. Lamiraux and L. E. Kavraki, “Positioning of symmetric and non-symmetric parts using radial and constant fields: Computation of all equilibrium configurations,” *International Journal of Robotics Research*, vol. 20, no. 8, pp. 635–659, 2001.
 - [13] T. Vose, P. Umbanhowar, and K. Lynch, “Friction-induced velocity fields for point parts sliding on a rigid oscillated plate,” *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 1020–1039, 2009.
 - [14] T. H. Vose, P. Umbanhowar, and K. M. Lynch, “Sliding manipulation of rigid bodies on a controlled 6-dof plate,” *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 819–838, 2012.
 - [15] S. Shahrokhi and A. T. Becker, “Moving Two Particles with Shared Control Inputs Using Wall Friction, Wolfram Demonstrations Project,” Nov. 2015. [Online]. Available: <http://demonstrations.wolfram.com/MovingTwoParticlesWithSharedControlInputsUsingWallFriction/>
 - [16] A. V. Mahadev and A. T. Becker, “Arranging a robot swarm with global inputs and wall friction [discrete]. matlab central file exchange,” Dec. 2015. [Online]. Available: <https://www.mathworks.com/matlabcentral/fileexchange/54526>
 - [17] A. Mahadev, A. Nguyen, and A. T. Becker, “Position control using boundary interaction,” Sep. 2016. [Online]. Available: <http://www.thingiverse.com/thing:1761909>
 - [18] P. S. S. Kim, A. Becker, Y. Ou, A. A. Julius, and M. J. Kim, “Imparting magnetic dipole heterogeneity to internalized iron oxide nanoparticles for microorganism swarm control,” *Journal of Nanoparticle Research*, vol. 17, no. 3, pp. 1–15, 2015.