# Algorithms for Shaping a Particle Swarm with a Shared Input by Exploiting Non-Slip Wall Contacts*

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 control input such as a uniform magnetic gradient or electric field. In previous works, 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 that constraint, and provides position control algorithms that only require non-slip wall contact in 2D. Both in vivo and artificial environments often have such boundaries. We assume that particles in contact with the boundaries have zero velocity if the shared control input pushes the particle into the wall. This paper provides a shortest-path algorithm for positioning a two-particle swarm, and a generalization to positioning an $n$-particle swarm. Results are validated with simulations and a hardware demonstration.**

## I. Introduction

Particle swarms propelled by a uniform field, where each particle receives the same control input, 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]–[7]. 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 broadcast signal. The tiny robots themselves are often just rigid bodies, and it may be more accurate to define the *system*, consisting of particles, a uniform control field, and sensing, as the robot. Such systems are severely underactuated, having 2 degrees of freedom in the shared control input, but $2n$ degrees of freedom for the particle swarm. Techniques are needed that can handle this underactuation. 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.

Positioning is a foundational capability for a robotic system, e.g. placement of brachytherapy seeds. However, requiring a single, small, rigid obstacle suspended in the middle of the workspace is often an unreasonable constraint, especially in 3D. This paper relaxes that constraint, and provides position control algorithms that only require non-slip wall contacts. We assume that particles in contact with the boundaries have zero velocity if the uniform control input pushes the particle into the wall.

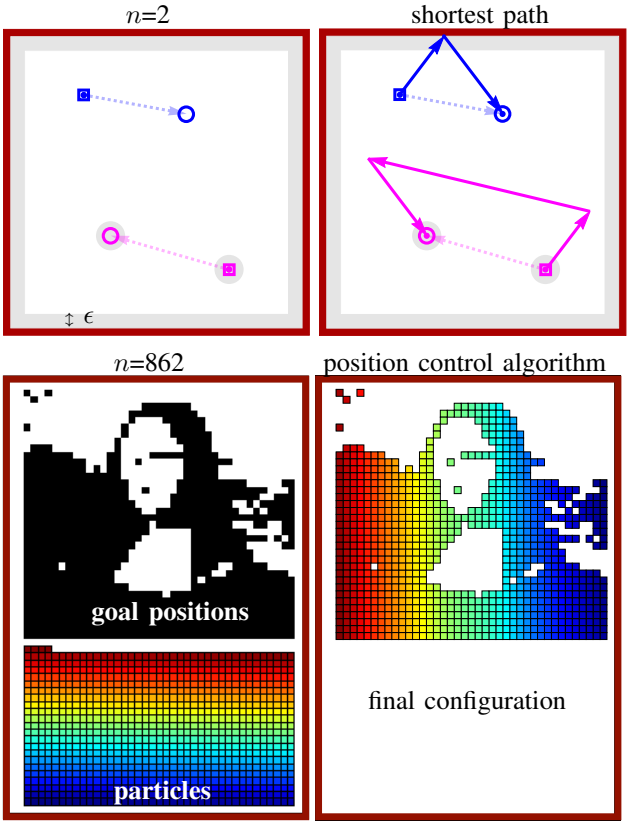


Fig. 1. Positioning particles that receive the same control inputs, but cannot move while a control input pushes them into a boundary. Top row shows the output of a best-first-search algorithm that finds the shortest path for two particles. Top left shows the initial position and goal position of the particles. The shortest path consists of moving at angle 50° until the blue robot contacts the top wall, then moving the magenta robot at angle 165° until the particles reach the desired relative spacing, then moving $-60°$ to the goal positions. The two bottom pictures show $n$-particle positioning using shared control inputs and boundary interaction.

The paper is arranged as follows. After a review of recent related work in Sec. II, Sec. III introduces a model for boundary interaction. We provide a shortest-path algorithm to arbitrarily position two robots in Sec. IV, and Sec. V extends this to prove a rectangular workspace with non-slip boundaries can position a swarm of $n$ robots arbitrarily within a subset of the workspace. Sec. VI describes im-

*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

plementations of the algorithms in simulation and Sec. VII describes hardware 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 [8], and the gradient-based decentralized controllers using control-Lyapunov functions in [9]. 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], [10], [11] 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 by the uniform input. Symmetry can be broken using agents that respond differently to the uniform control signal, either through agent-agent reactions, see work modeling biological swarms [12], or engineered inhomogeneity [4], [13], [14]. This work assumes a uniform control with homogenous agents, as in [15]. 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 [16]. [18] demonstrated a collection of 2D force fields generated by six degree-of-freedom 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, their control was multi-modal and position-dependent.

## III. Theory

*Using Boundaries: Friction and Boundary Layers*

In the absence of obstacles uniform inputs move a swarm identically. Shape control requires breaking this symmetry. The following sections examine using non-slip boundary contacts to break the symmetry caused by uniform inputs.

If the $i^{\text{th}}$ particle has position $\mathbf{x}_i(t)$ and velocity $\dot{\mathbf{x}}_i(t)$, we assume the following system model:

$$\dot{\mathbf{x}}_i(t) = \mathbf{u}(t) + F\left(\mathbf{x}_i(t), \mathbf{u}(t)\right), \ i \in [1, n]. \tag{1}$$

$$F(\mathbf{x}_i(t), \mathbf{u}(t)) = \begin{cases} -\mathbf{u}(t) & \mathbf{x}_i(t) \in \text{boundary } \textbf{and} \\ & \mathbf{N}(\text{boundary}(\mathbf{x}_i(t))) \cdot \mathbf{u}(t) \leq 0 \\ 0 & \text{else} \end{cases}$$

Here $\mathbf{N}(\text{boundary}(\mathbf{x}_i(t)))$ is the normal to the boundary at position $\mathbf{x}_i(t)$, and $F(\mathbf{x}_i(t), \mathbf{u}(t))$ is the frictional force provided by the boundary.

These system dynamics represent particle swarms in low-Reynolds number environments, where viscosity dominates inertial forces and so velocity is proportional to input force [19]. In this regime, the input force command $\mathbf{u}(t)$ controls the velocity of the robots. The same model can be generalized to particles moved by fluid flow where the vector direction of fluid flow $\mathbf{u}(t)$ controls the velocity of particles, or for a swarm of robots that move at a constant speed in a direction specified by a uniform input $\mathbf{u}(t)$ [20]. As in our model, fluid flowing in a pipe has zero velocity along the boundary. Similar mechanical systems exist at larger scales, e.g. all tumblers of a combination look move uniformly unless obstructed by an obstacle. Our control problem is to design the control inputs $\mathbf{u}(t)$ to make all $n$ particles achieve a task.

## IV. Position Control of Two Robots Using Boundary Interaction

Alg. 1 uses non-slip contacts with walls to arbitrarily position two robots in a rectangular workspace. Fig. 2 shows a Mathematica implementation of the algorithm, and is useful as a visual reference for the following description.

Assume two robots are initialized at $r_1$ and $r_2$ with corresponding goal destinations $g_1$ and $g_2$. The solution is a best-first search algorithm that maintains a list of possible paths (*moves*), sorted according to an admissible heuristic on path length. The algorithm works by expanding the path with the shortest estimated length. Expanding a path means either moving directly to the goal, or pushing one robot to a wall and adjusting the relative position of the other robot. As soon as the goal is reached, the algorithm returns this shortest 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., $r_1.x$ and $r_1.y$ denote the $x$ and $y$ locations of $r_1$. The algorithm assigns a uniform control input at every instance. The goal is to zero the error in both coordinates, $\Delta e = (\Delta e.x, \Delta e.y) = \Delta g - \Delta r = (g_2 - g_1) - (r_2 - r_1)$ using a shared control input. The base case occurs when $\Delta e = (0, 0)$. In this base case, the shortest path is a straight line between the current position of each robot to its goal position, as shown in Fig. 2a. If $\Delta e \neq (0, 0)$, we compute the two-move reachable sets generated by first immobilizing one particle by contacting the wall and then repositioning the second particle. There are four reachable sets, as shown in Fig. 3. The horizontal (and vertical) reachable sets are equivalent in the $\Delta$ configuration space, so we can plan in this space and choose to immobilize the particle closest to a wall. Each non-slip move changes the $\Delta$ configuration space, as shown in Fig. 4.

If the goal configuration can be reached in three moves, then $m_1$ makes one particle hit a wall, $m_2$ adjusts the relative spacing error $\Delta e$ to zero, and $m_3$ takes the particles to their final positions, as shown in Fig. 2b. $m_2$ cannot be shortened, so optimization depends on choosing the location where the robot hits the wall. Since the shortest distance between two points is a straight line, reflecting the goal position across the boundary wall and plotting a straight line gives the optimal

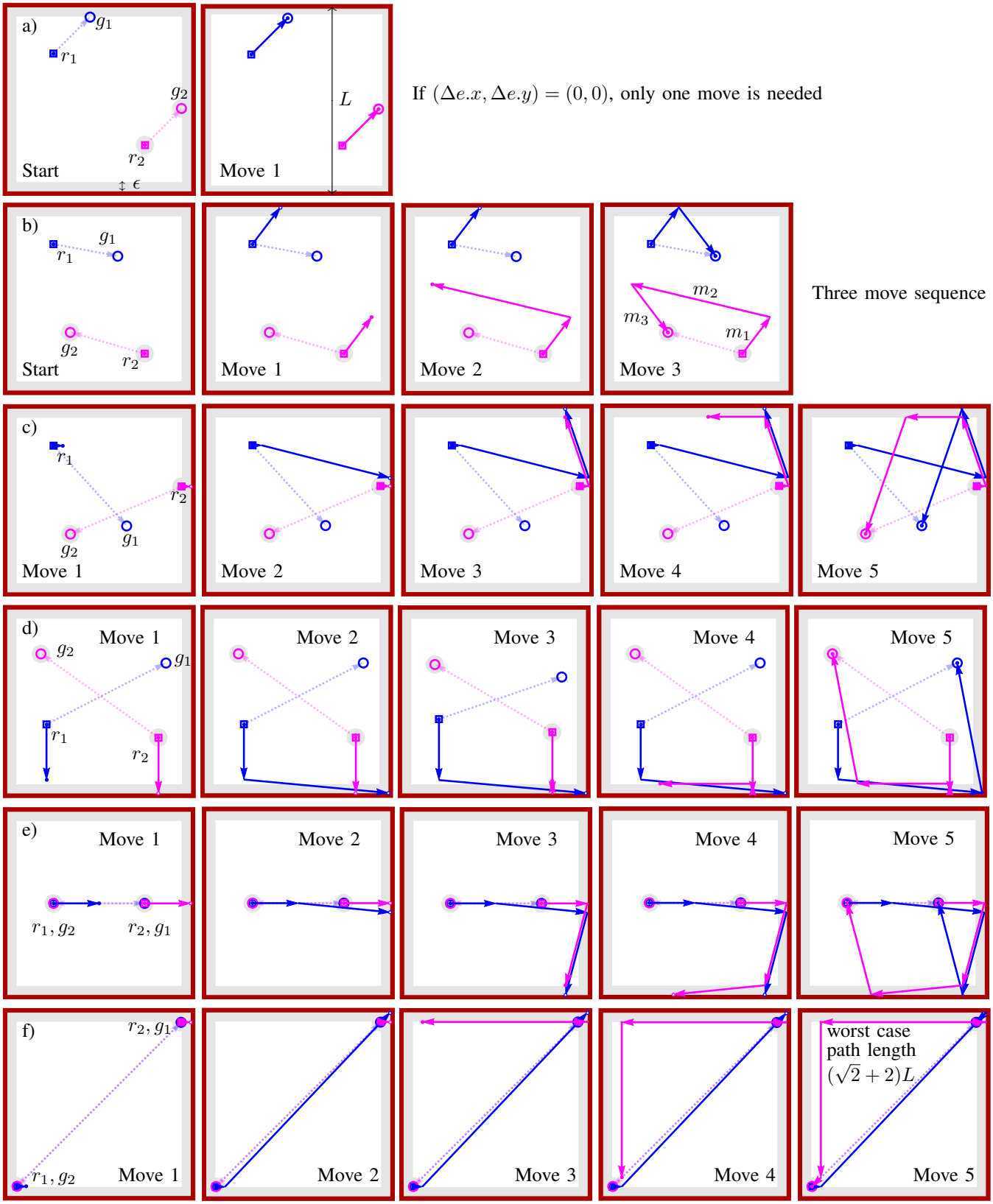


Fig. 2. Frames from an implementation of Alg. 1: two robot positioning using walls with non-slip contacts. Robot start positions are shown by a square, and goal positions by a circle. Dashed lines show the shortest route if robots could be controlled independently. Solid arrows show path given by Alg. 1. Online demonstration and source code at [21].

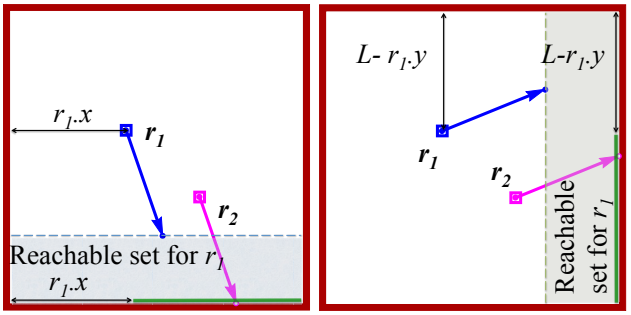


Fig. 3. Boundary interaction is used to change the relative positions of the particles. Each particle gets the same control input. (left) If particle 2 hits the bottom wall before particle 1 reaches a wall, particle 2 can reach anywhere along the green line, and particle 1 can move to anywhere in the shaded area. (right) Similarly, if particle 2 hits the right wall before particle 1 reaches a wall, particle 2 can reach anywhere along the green line, and particle 1 can move to anywhere in the shaded area.

hit location, as shown in Fig. 5. That point is selected when possible, but if this point would cause $m_2$ to push the moving robot out of the workspace, the hit point is translated until the moving robot will not leave the workspace. If $m_2$ causes the two particles to overlap, we add or subtract $\epsilon$ to $m_2.x$ to avoid collisions. This is shown in Fig. 6 with three different $\epsilon$ values.

If $\Delta g$ is not in the reachable set, we choose the nearest reachable $\Delta x$ and $\Delta y$ to $\Delta g$.

Alg. 1 uses an admissible heuristic that adds the current path length to the greatest distance from each robot to their goal. This heuristic directs exploration by expanding favorable routes first.

$$h(moves, r_1, r_2, g_1, g_2) = \sum_{i=1}^{|moves|} \|moves_i\| + \max(\|g_1 - r_1\|, \|g_2 - r_2\|) \quad (2)$$

We exploit symmetry in the solution by labeling the leftmost (or, if they have the same $x$ coordinate, the topmost) robot $r_1$. If $r_1$ is not also the topmost robot, we mirror the coordinate frame about the right boundary. As an example, consider the two starting positions, $r_1 = (0.2, 0.2)$ and $r_2 = (0.8, 0.8)$. Because the leftmost robot is not the topmost robot, we mirror the coordinate frame about the right boundary giving $r_1 = (0.2, 0.8)$ and $r_2 = (0.8, 0.2)$. After the path is found, we undo the mirroring to the output path. Similarly, we exploit rotational symmetry and assume the command pushes a robot to hit the top wall. If a different wall is selected, we rotate the coordinate frame by 90°, 180°, or 270° counterclockwise and then push the robot to hit the top wall. After the path is found, we undo the rotation. This symmetry allows us to use a single function, Alg. 2, for collisions with all four walls.

## V. Position Control of $n$ Robots Using Boundary Interaction

The ideas from Alg. 1 can be extended to control the position of $n$ particles using walls with non-slip contact. The solution is complete, but not optimal, and requires the

**Algorithm 1** 2-ParticlePathFinder$(r_1, r_2, g_1, g_2, L)$

**Require:** knowledge of current $(r_1, r_2)$ and goal $(g_1, g_2)$ positions of two robots. $(0, 0)$ is bottom corner, $L$ is length of the walls. *PathList* contains all the paths sorted by their path length plus an admissible heuristic.

```
PathList ← {}
P ← {h({}, r1, r2, g1, g2), {}, r1, r2}                ▷ P contains the admissible heuristic, the move sequence, and the current robot positions
while P.r1 ≠ g1 and P.r2 ≠ g2 do
    for θ ∈ {0°, 90°, 180°, 270°} do
        (r1, r2, g1, g2) ← Rotate(P.r1, P.r2, g1, g2, θ)
        {d, moves, r1, r2} ←
            PlanMoveUp(r1, r2, g1, g2, L, P.moves)
        (moves, r1, r2) ← Rotate(moves, r1, r2, −θ)
        Push {d, moves, r1, r2} onto PathList
    end for
    Sort(PathList)                                     ▷ sort by admissible heuristic
    P ← Pop first element of PathList
end while
return moves
```

starting and final configurations of particles to be disjoint. The solution described here is an iterative procedure with $n$ loops. The $k^{\text{th}}$ loop moves the $k^{\text{th}}$ robot from a *staging zone* to the desired position in a *build zone*. All robots move according to the uniform input, but due to non-slip wall contacts, at the end of the $k^{\text{th}}$ 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. See Fig. 8 for a schematic of the build and staging zones.

Assume an open workspace with four axis-aligned walls with non-slip contact. 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. Let $d$ be the diameter of the particles. Furthermore, there must be at least $\epsilon$ space above the build zone, $\epsilon$ below the staging zone, and $\epsilon + d$ to the left of the build and staging zone. The minimum workspace is then $(\epsilon + d + \max(w_b, w_s), 2\epsilon + h_s, h_b)$.

The $n$ robots position control algorithm relies on a DriftMove$(\alpha, \beta, \epsilon, \theta)$ control input, described in Alg. 4 and shown in Fig. 9. For $\theta = 0°$, a drift move consists of repeating a triangular movement sequence $\{(\beta/2, -\epsilon), (\beta/2, \epsilon), (-\alpha, 0)\}$. Any particle touching a top wall moves right $\beta$ units, while every particle not touching the top moves right $\beta - \alpha$.

Let $(0, 0)$ be the lower left corner of the workspace, $p_k$ the $x, y$ position of the $k^{\text{th}}$ robot, and $f_k$ the final $x, y$ position of the $k^{\text{th}}$ 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. 10.

Alg. 3 proceeds as follows: First, the robots are moved left away from the right wall, and down so all robots in

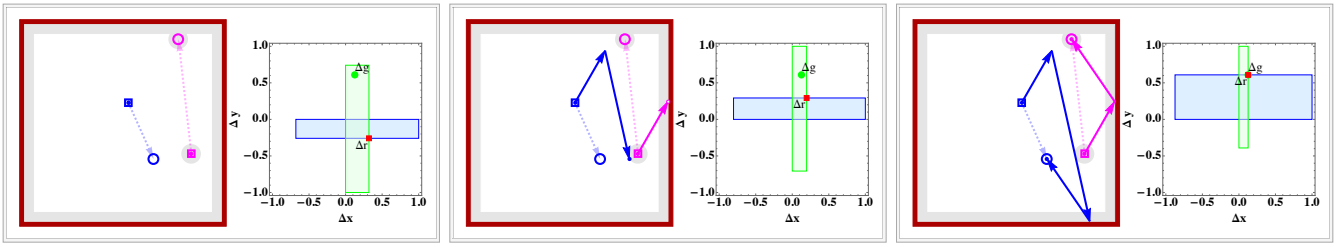

Fig. 4. Workspace and $\Delta$ configuration space for three sets of robot configurations with the same final goal. The red square represents the starting $\Delta x$ and $\Delta y$ and the green circle represents the goal $\Delta x$ and $\Delta y$. The green rectangle illustrates reachable $\Delta x$ and $\Delta y$ when one particle is in contact with a horizontal wall and the blue rectangle illustrates the reachable region when in contact with a vertical wall.

**Algorithm 2** PLANMOVEUP($r_1, r_2, g_1, g_2, L$,*moves*)

**Require:** knowledge of current $(r_1, r_2)$ and goal $(g_1, g_2)$ positions of two robots. $(0, 0)$ is bottom corner, $L$ is length of the walls. The array *moves* is the current sequence of moves up to the current position. Assume $r_1.x < r_2.x$ and $r_1.y \geq r_2.y$. If not, mirror the coordinate frame and swap the robots, then undo the mirroring before returning. $\epsilon$ is a small, nonzero, user-specified value.

**Ensure:** $(g_1, g_2), (r_1, r_2)$ all at least $\epsilon$ distance from walls the goals and starting points have at least $\epsilon$ distance from each other. $m_1$ is the first move toward the wall or goal. $m_2$ is the second move adjusting $\Delta e$.

$\Delta e \leftarrow (g_2 - g_1) - (r_2 - r_1)$
**if** $\Delta e = (0, 0)$ **then** ▷ base case
  $m_1 \leftarrow g_2 - r_2$
  *moves*$\leftarrow \{moves, m_1\}$
  $(r_1, r_2) \leftarrow$APPLYMOVE$(m_1, r_1, r_2)$
  **return** $\{h(moves, r_1, r_2, g_1, g_2), moves, r_1, r_2\}$
**end if**
**if** $r_2.x - r_1.x - 1 + 2\epsilon \leq \Delta g.x \leq 1$ **and** $r_2.y - r_1.y \leq \Delta g.y \leq 0$ **then** ▷ $\Delta g \in$ reachable region
  $m_1 \leftarrow \left(\frac{1-r_1.y}{2-g_1.y-r_1.y}(g_1.x - r_1.x), 1 - r_1.y\right)$
  **if** $r_2.x + m_1.x > L$ **then**
    $m_1.x \leftarrow 1 - r_2.x$
  **else if** $r_2.x + m_1.x < 0$ **then**
    $m_1.x \leftarrow -r_2.x$
  **end if**
**else**
  $m_1 = (0, 1 - r_1.y)$
  $\Delta g \leftarrow$ closest reachable $(\Delta x, \Delta y)$.
**end if**
*moves*$\leftarrow \{moves, m_1\}$
$(r_1, r_2) \leftarrow$APPLYMOVE$(m_1, r_1, r_2)$
$m_2 \leftarrow \Delta g - (r_2 - r_1)$
**if** robots on each other **or** on the wall **then**
  Add $\pm\epsilon$ to $m_2.x$ to avoid collision
**end if**
*moves* $\leftarrow \{moves, m_2\}$
$(r_1, r_2) \leftarrow$APPLYMOVE$(m_2, r_1, r_2)$
**return** $\{h(moves, r_1, r_2, g_1, g_2), moves, r_1, r_2\}$

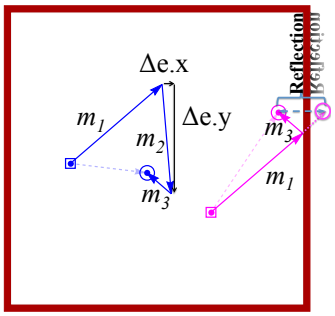


Fig. 5. If the goal configuration can be reached in three moves, the first move makes one particle hit a wall, the second move adjusts the relative spacing error $\Delta e$ to zero, and the third move takes the particles to their final positions. The second move cannot be shortened, so optimization depends on choosing the location where the robot hits the wall. Since the shortest distance between two points is a straight line, reflecting the goal position across the boundary wall and plotting a straight line gives the optimal hit location.

**Algorithm 3** PositionControl$n$Robots($k$)

Move( $-\epsilon, d/2 - p_{ky}$)
**while** $p_{kx} > d/2$ **do**
  DRIFTMOVE$(\epsilon, \min(p_{kx} - d/2, \epsilon), \epsilon, 180°)$
**end while**
$m \leftarrow \text{ceil}(\frac{f_{ky}-d/2}{\epsilon})$
$\beta \leftarrow \frac{f_{ky}-d/2}{m}$
$\alpha \leftarrow \beta - \frac{d/2-p_{ky}-\epsilon}{m}$
**for** $m$ iterations **do**
  DRIFTMOVE$(\alpha, \beta, \epsilon, 90°)$
**end for**
Move $(d/2 + \epsilon - f_{kx}, 0)$
Move $(f_{kx} - d/2, 0)$

**Algorithm 4** DRIFTMOVE$(\alpha, \beta, \epsilon, \theta)$

particles touching the wall move $\beta$ units, while particles not touching the wall move $\beta - \alpha$ units.

$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$
MOVE$(R \cdot [\beta/2, -\epsilon]^\top)$
MOVE$(R \cdot [\beta/2, \epsilon]^\top)$
MOVE$(R \cdot [-\alpha, 0]^\top)$

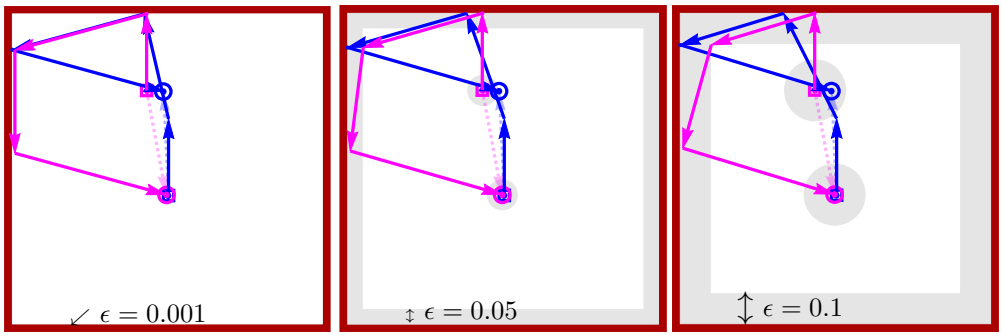


Fig. 6. Changing the minimum spacing $\epsilon$ changes the path. $\epsilon$ is the minimum spacing between two robots and the minimum separation from the boundaries.

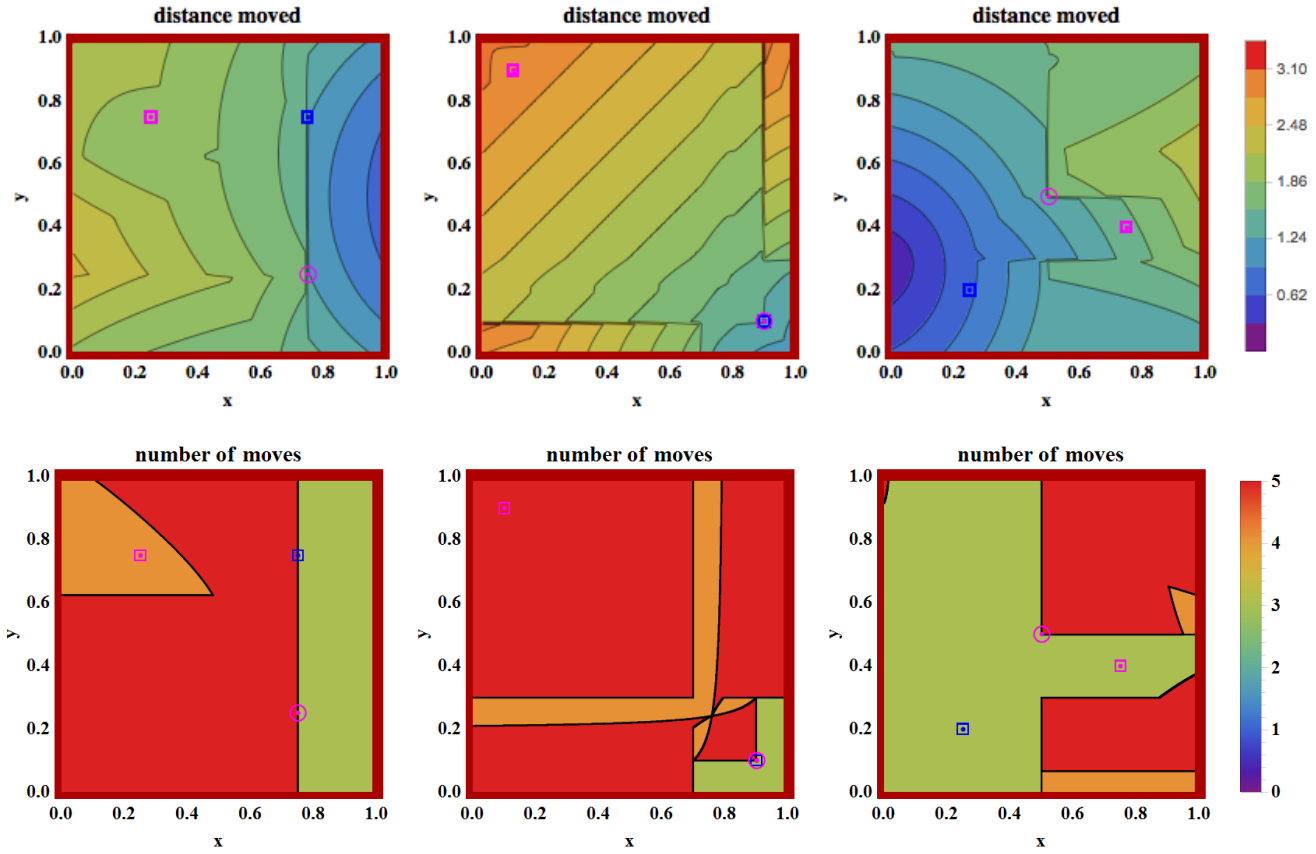


Fig. 7. Starting positions of robots 1 and 2 and goal position of robot 2 are fixed, and $\epsilon = 0.001$. The top row of contour plots show the distance if robot 1's goal position is varied in $x$ and $y$. The bottom row shows the number of moves required for the same configurations.

$k^{\text{th}}$ row touch the bottom wall. Second, a set of DriftMoves are executed that move all robots in $k^{\text{th}}$ row left until $k$ touches the left wall, with no net movement of the other robots. Third, a set of DriftMoves are executed that move only 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 $O(n(w+h))$.

The hardware platform depicted in Fig. 10 is an assembled practical setup that assumes that $\epsilon = 1$ cm. The workspace is a $7 \times 7$ cm grid space. All particles are 3D-printed plastic whose top is a 1cm diameter cylinder with a narrower base that encapsulates a steel bearing ball. Non-slip wall contact is generated by a toothed wall design to keep particles from moving out of place while implementing the drift move. The workspace boundary is mounted on top of a white sheet of cardboard. Underneath the cardboard, a grid of 3mm diameter magnets glued with 1 cm spacing to a thin board generates the uniform control input. A video attachment shows the algorithm at work. This discretized setup requires several modifications to Alg. 3. In this demonstration, all moves are 1 cm in length. All drift moves are a counterclockwise *square* move of size 1 cm$\times$1 cm. Once the $k^{\text{th}}$ particle gets to its designated location in each loop, a correction step is implemented. This correction step increases by two the total number of moves required per particle. Fig. 8 shows there are

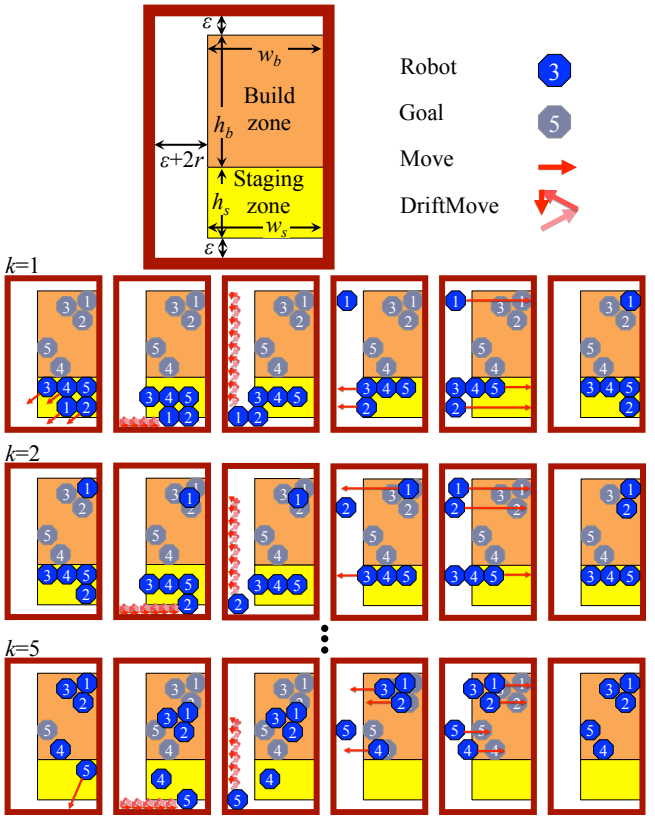


Fig. 8. Illustration of Alg. 3, $n$ robot position control using walls with non-slip contact.

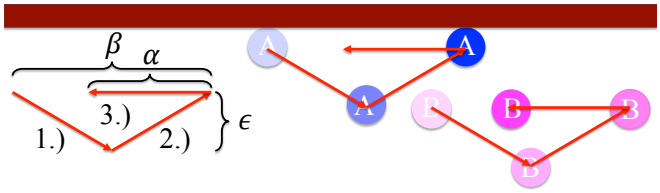


Fig. 9. A DRIFTMOVE$(\alpha, \beta, \epsilon, 0°)$ repeats a triangular movement sequence $\{(\beta/2, -\epsilon), (\beta/2, \epsilon), (-\alpha, 0)\}$. At the sequence end, robot $A$ has moved $\beta$ units right, and robot $B$ has moved $\beta - \alpha$ units right.

only 6 stages per particle involved in Alg. 3. The fixed step algorithm requires 8 stages per particle as shown in Fig. 10.

A significant difference between Alg. 3 and the fixed move implementation of it is that Alg. 3 enables placing particles at arbitrary, non-overlapping locations, while the fixed move implementation requires goal locations at the center of grid cells.

## VI. SIMULATION

Two simulations were implemented using non-slip contact walls for position control. The first controls the position of two robots, the second controls the position of $n$ robots.

### A. Position Control of Two Robots

Algorithm 1 was implemented in Mathematica using point robots (radius = 0). Fig. 2 shows an implementation of this algorithm with robot initial positions represented by hollow squares and final positions by circles. Dashed lines show the shortest route if robots could be controlled independently,

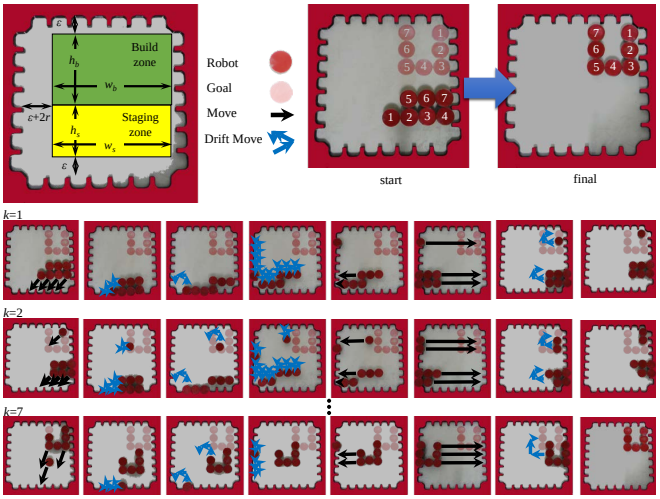


Fig. 10. Illustration of Alg. 3, discretized $n$ robot position control using walls that enforce non-slip contact.

while solid lines show the optimal shortest path using uniform inputs.

The contour plots in Fig. 7 top row show the length of the shortest path for given $s_1, s_2, g_1$ with $g_2$ ranging over all the workspace. This plot clearly shows the nonlinear nature of the path planning, with multiple isolated islands showing regions that are difficult to reach. If the length of each side of the square workspace is $L$, the worst case path length is $(\sqrt{2}+2)L$.

The contour plots in Fig. 7 bottom row show the same configurations, but plot the required number of moves. There is never more than one $g_2$ position reachable in one move. If $g_2$ results in a contraction of $(\delta x, \delta y)$, there are many three move sequences. Four and five moves are sometimes required.

### B. Position Control of $n$ Robots

Alg. 3 was simulated in MATLAB using square block robots with unity width. 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 $\epsilon$ 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. DEMONSTRATION

*Hardware Demonstration: Position Control of n Robots*
A hardware setup with a bounded platform, magnetic sliders, and a magnetic guide board was used to implement Alg. 3. The pink boundary is toothed with a white free space, as shown in Fig 10. 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 particles. The dark red configuration is the current position of the particles.

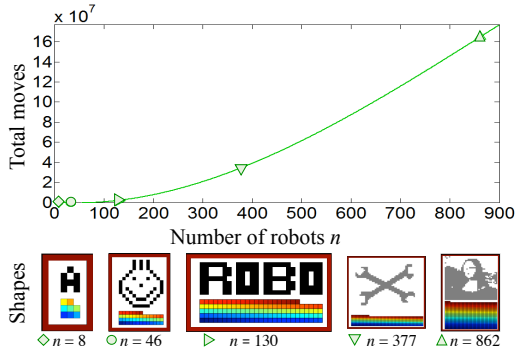


Fig. 11. The required number of moves under Alg. 3 using non-slip contact walls to rearrange $n$ square-shaped robots grows quadratically with $n$. See code at [22].

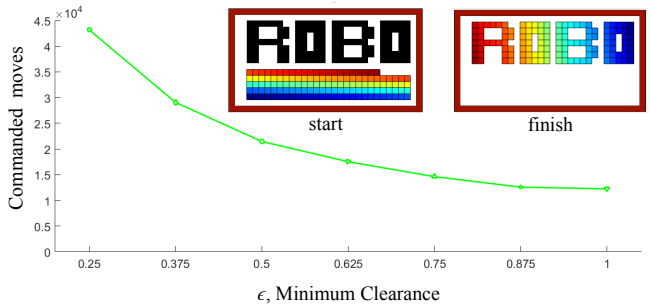


Fig. 12. Control performance is sensitive to the desired clearance $\epsilon$. As $\epsilon$ increases, the total distance decreases asymptotically.

Due to the discretized movements allowed by the boundary, drift moves follow a 1 cm square. Free particles return to their start positions but particles on the boundary to move laterally, generating a net sliding motion.

Fig. 10 follows the motion of the particles through iterations $k$=1, 2, and 7. All particles receive the same control inputs, but boundary interactions break the control symmetry. Particles reach their goal positions in a first-in, first-out arrangement beginning with the bottom-left particle 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 position of a swarm of robots using uniform 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 3 to 3D is straightforward but increases the complexity. Additionally, this paper assumed friction was sufficient to completely stop particles in contact with the boundary. The algorithms require retooling to handle small friction coefficients. The algorithms assumed a rectangular workspace. This is a reasonable assumption for artificial environments, but in vivo environments are curved. A best-first-search program could still work, but it cannot take advantage of the 4-fold rotational symmetry as in a rectangular environment. Future efforts should be directed toward improving the technology and tailoring it to specific robot applications.

## 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] S. Martel, S. Taherkhani, M. Tabrizian, M. Mohammadi, D. de Lanauze, and O. Felfoul, "Computer 3d controlled bacterial transports and aggregations of microbial adhered nano-components," *Journal of Micro-Bio Robotics*, vol. 9, no. 1-2, pp. 23–28, 2014.

[3] 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.

[4] 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.

[5] A. Ghosh and P. Fischer, "Controlled propulsion of artificial magnetic nanostructured propellers," *Nano Letters*, vol. 9, no. 6, pp. 2243–2245, 2009.

[6] Y. Ou, D. H. Kim, P. Kim, M. J. Kim, and A. A. Julius, "Motion control of magnetized tetrahymena pyriformis cells by magnetic field with model predictive control," *Int. J. Rob. Res.*, vol. 32, no. 1, pp. 129–139, Jan. 2013.

[7] F. Qiu and B. J. Nelson, "Magnetic helical micro-and nanorobots: Toward their biomedical applications," *Engineering*, vol. 1, no. 1, pp. 21–26, 2015.

[8] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE Trans. Robotics Automat.*, vol. 17, pp. 947–951, 2001.

[9] 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.

[10] S. Martel, "Magnetotactic bacteria for the manipulation and transport of micro-and nanometer-sized objects," *Micro-and Nanomanipulation Tools*, pp. 308–317, 2015.

[11] 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.

[12] 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.

[13] T. Bretl, "Control of many agents using few instructions," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007, pp. 1–8.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] E. M. Purcell, "Life at low Reynolds number," *American Journal of Physics*, vol. 45, no. 1, pp. 3–11, 1977. [Online]. Available: http://dx.doi.org/10.1119/1.10903

[20] 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.

[21] 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/

[22] A. V. Mahadev and A. T. Becker, “Arranging a robot swarm with global inputs and wall friction [discrete]. matlab central file exchange,” Feb. 2017. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/54526

[23] A. Mahadev, A. Nguyen, and A. T. Becker, “Position control using boundary interaction,” Feb. 2017. [Online]. Available: http://www.thingiverse.com/thing:1761909