Exploiting Non-Slip Wall Contacts to Position Two Particles Using a Shared Input

Shiva Shahrokhi, Jingang Shi, 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 individual 2D position of two particles in such a swarm is controllable in a rectangular workspace requiring non-slip wall contact. However, both in vivo and artificial environments are usually not rectangular. This work extends the analysis to convex workspaces and 3D positioning. This paper also implements the algorithms using a hardware setup inspired by intestine anatomy.**

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.

*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,atbecker}@uh.edu

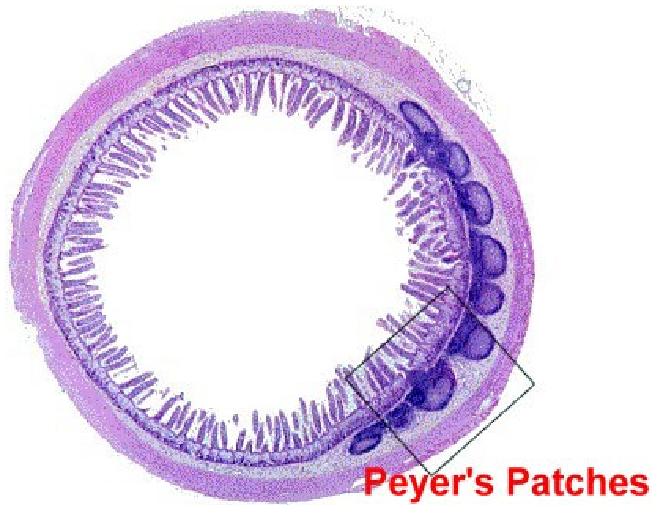

Fig. 1. Positioning particles that receive the same control inputs, but cannot move while a control input pushes them into a boundary.

The paper is arranged as follows. After a review of recent related work in Sec. II, Sec. III-A introduces a model for boundary interaction. We provide a shortest-path algorithm to arbitrarily position two robots in Sec. IV. Sec. V describes implementations of the algorithms in simulation and Sec. VI describes hardware experiments, as shown in Fig. 1. We end with directions for future research in Sec. VII.

This paper is elaboration of preliminary work in a conference paper [8] which consider only square workspaces. This work extends the analysis to convex workspaces and 3D positioning. This paper also implements the algorithms using a hardware setup inspired by intestine anatomy.

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 [9], and the gradient-based decentralized controllers using control-Lyapunov functions in [10]. 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], [11], [12] 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 [13], or engineered inhomogeneity [4], [14], [15]. This work assumes a uniform control with homogenous agents, as in [16]. 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 [17]. [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

A. Boundary Interaction Model

In the absence of obstacles uniform inputs move a swarm identically. Independent 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^{th} particle has position $\mathbf{x}_i(t)$ and velocity $\dot{\mathbf{x}}_i(t)$, we assume the following system model:

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

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

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.

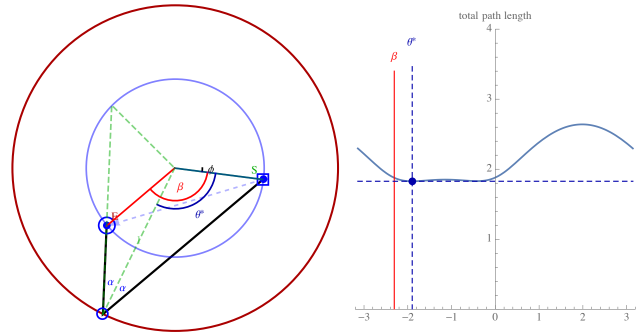

Fig. 2. The shortest path between two points (blue square) to (blue ellipse) in the unit disk that intersects the circumference. The path length as a function of intersection point, $(\cos\theta, \sin\theta)$ is shown at right.

B. Shortest Path

The shortest path between two points in the unit disk that reflects off the circumference is composed of two straight line segments shown in Fig. 2. The problem can be simplified by choosing the coordinate system carefully. We define the x axis along the position of the starting point: $S = (s, 0)$, and define the point of intersection by the angle θ from the x axis $P = (\cos\theta, \sin\theta)$, and the final point by a radius e and angle β, $E = e(\cos\beta, \sin\beta)$. Then define a symmetry point about S of line OP named T. Then the length of the two line segments is

$$\sqrt{(s - \cos\theta)^2 + (-\sin\theta)^2} + \sqrt{(e\sin\beta - \cos\theta)^2 + (e\sin\beta - \sin\theta)^2} \tag{2}$$

which is minimized by choosing an appropriate θ value. This equation can be simplified to

$$\sqrt{1 + e^2 - 2e\cos(\beta - \theta)} + \sqrt{1 + s^2 - 2s\cos\theta}. \tag{3}$$

The length of the two line segments as a function of θ is drawn in the right plot. There are several simple solutions. If s is 1 or e is 0 or β is 0, the optimal angle θ^* is 0. If e is 1 or s is 0, the optimal angle is β. Label the origin O. The optimal solution shows that the angle $\angle OPS$ (from the origin to P to S) is the same as the angle $\angle OPE$ (from the origin to P to E). We name these angles α. This can be proved by drawing an ellipse whose foci are S and E. When the ellipse is tangent to the circle, the point of tangency is exactly P. Since the distance from the origin to P is always 1, we can set up three equalities using the law of sines: From triangle OSP: $\frac{\sin\alpha}{s} = \frac{\sin(\alpha+\theta)}{1} = \frac{\sin\theta}{SP}$, and from triangle OEP: $\frac{\sin\alpha}{e} = \frac{\sin(\beta-\theta)}{EP}$. If we mirror the point S about the θ axis and label this point C, from triangle CEO: $\frac{\sin(\alpha+\theta)}{e} = \frac{\sin(2\theta-\beta)}{CE}$.

Simplifying this system of equations results in: $s = e\csc\theta(s\sin(2\theta - \beta) + \sin(\beta - \theta))$. Solving this last equation results in a quartic solution that has a closed-form solution with four roots, each of which can be either a clockwise or a counterclockwise rotation θ, depending on the sign of β, with $-\pi <= \beta <= \pi$. We evaluate each and select the solution that results in the shortest length path. Note that the optimal path satisfies the law of reflection off the unit circle, with angle of incidence equal to angle of reflection.

C. Different Polygonal Workspaces

Fig. 3 shows different workspaces and their representative Δ configuration spaces. Consider one robot touching each vertex of the workspace. For each pair of vertices, compute where the other robot will be if the first robot goes to that vertex. If the final position of the second robot is still inside the workspace, then its position is one of the vertices of the reachable set. If the point is not inside the polygon, then the intersection of the line that point and the second robot's position with the polygon is one vertex of the reachable set. Compute the distance to all the vertices of the workspace from this point. By subtracting the relative distance of the robots, then all the vertices of one reachable set are found. Doing this for all the vertices of the workspace will give us all the reachable sets.

IV. Position Control of Two Robots Using Boundary Interaction

Alg. 1 uses non-slip contacts with walls to arbitrarily position two robots in a circular workspace. In our previous work we used a rectangular workspace. We use the same idea and algorithm here but we have to modify the algorithm to adjust it for a circular workspace.

Assume two robots are initialized at s_1 and s_2 with corresponding goal destinations g_1 and g_2. Denote the current positions of the robots p_1 and p_2 and the current distance between the robots is d. Values $.x$ and $.y$ denote the x and y coordinates, i.e., $p_1.x$ and $p_1.y$ denote the x and y locations of p_1. The algorithm assigns a uniform control input at every instance. The goal is to move the particles to the goal positions using a shared control input. We do this by first moving them to the correct relative position and then translating the particles to the goal. The first step minimizes $||\Delta g - \Delta p|| = ||(g_2 - g_1) - (p_2 - p_1)||$.

We define a Δ configuration space as a circular shape that considers all possible Δps. We also show the starting and ending relative distance as Δs and Δe in Δ configuration space in Fig. 5. Reachable set is the part of Δ configuration space where if one robot touches a wall in a specific location, the other robot can make the required relative distance without causing touching robot to move. To compute reachable set for circular workspace, first we considered all passible hitting point locations in the workspace. The set of boundary points that a robot can touch before the other robot touches is an arc of angle $2(\pi - \frac{\arcsin d}{r})$, where $d = |s_1 - s_2|$ and r is the radius of the circle. We define the angle between two particles as $\theta = \arctan(\frac{p_1.x - p_2.x}{p_1.y - p_2.y})$.

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 path.

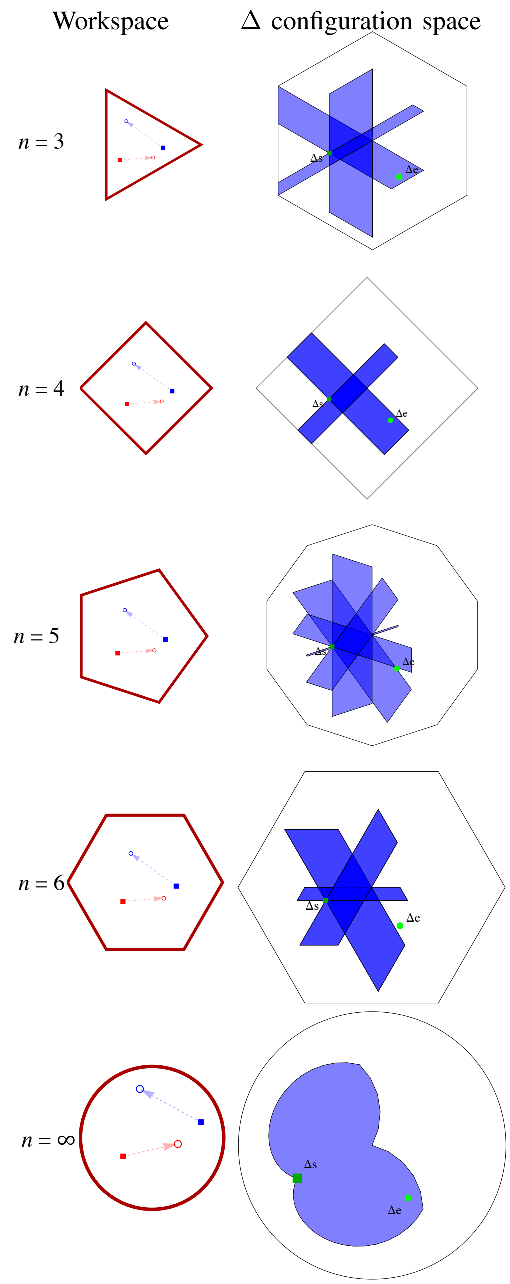

Fig. 3. Workspace and Δ configuration spaces for different polygonal workspaces and their representative Δ configuration spaces and reachable sets. As the number of sides in the polygon increases, the total area of the Δ configuration space is four times of the workspace.

There are infinite reachable sets, parameterized by first contact location ψ, as shown in Fig. **??**.

$$\psi \in [\theta + \frac{\sin^{-1} d}{2r} - \frac{\pi}{2}, \theta + \frac{\sin^{-1} d}{2r} - \frac{\pi}{2}] \tag{4}$$

γ is half the angle of the arc that the reachable set's chord has shown in Fig. 5 and is calculated by:

$$p_\psi = r[\cos(\psi), \sin(\psi)] \tag{5}$$

$$d_\perp = 2\,|(s_1.p_\psi - s_2.p_\psi)| \tag{6}$$

$$\gamma = \cos^{-1}(1 - \frac{d_\perp}{r}) \tag{7}$$

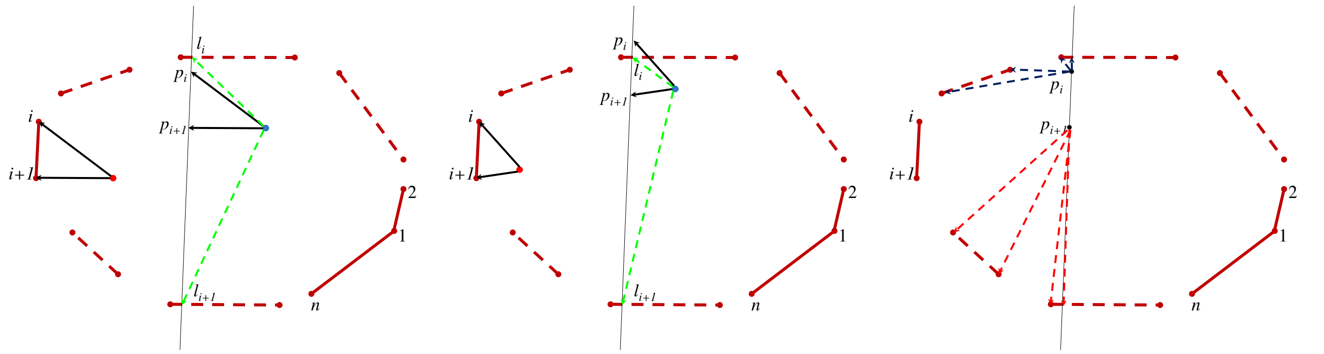

Fig. 4. blah blah blah

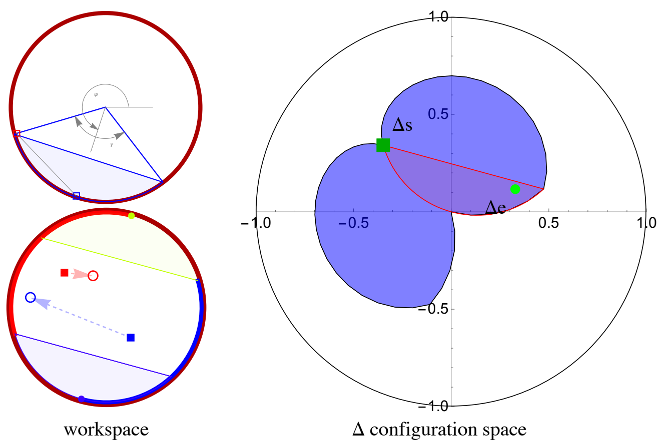

Fig. 5. Left: the set of points where the red robot is the first to contact the boundary are drawn with a red arc. The set of points where the blue robot is the first to contact the boundary are drawn with a blue arc. The possible points for the blue and pink particles to touch the boundary is shown in blue and pink arcs. Right: When the blue particle is touching a wall (blue square) the other particle (pink square) can go anywhere in the reachable set (blue region).

Reachable sets with π difference in ψ value are equivalent in the Δ configuration space, so we can plan in this space and choose to immobilize the particle closest to a wall.

The equation for the four lines outlining the reachable set can be found as follows:

Insert the equations

A. Square workspace

Our algorithm uses an A*-like method to find the shortest path in each move. If $(\Delta g.x, \Delta g.y)$ is in the reachable set, one robot touches a wall and the other robot zeros the error in one move. This is shown as m_2 in Fig. 8. To find the best place to minimize m_1 and m_3, the touching robot's goal is reflected on that wall. The minimum distance to get to the goal in two moves when the robot should touch the wall, is the straight line between the robot and the reflection of the goal position on that wall. 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 Δe to zero, and m_3 takes the particles to their final positions, as shown in Fig. **??**b. 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 hit location, as shown in Fig. 8. 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 ϵ to $m_2.x$ to avoid collisions. This is shown in Fig. **??** with three different ϵ values.

If Δg is not in the reachable set, we choose the nearest reachable Δx and Δy to Δg.

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.

$$h(moves, r_1, r_2, g_1, g_2) = \sum_{i=1}^{|moves|} \|moves_i\| + \max(\|g_1 - r_1\|, \|g_2 - r_2\|) \tag{8}$$

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.

B. 3D model

Extending to 3D is straight forward, but possible only if the two particles do not have the same x and y positions. Fig. 9 shows a cylinder. The blue particle is in the blue plate

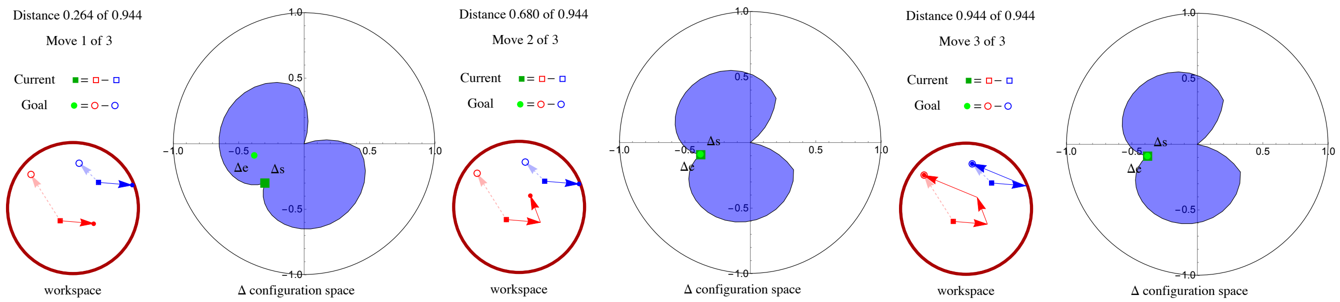

Fig. 6. Left circle shows the workspace. Right shows the Δ configuration space and the reachable set that is shown in red is representative of the point we need to go to get to the goal relative distance in one move.

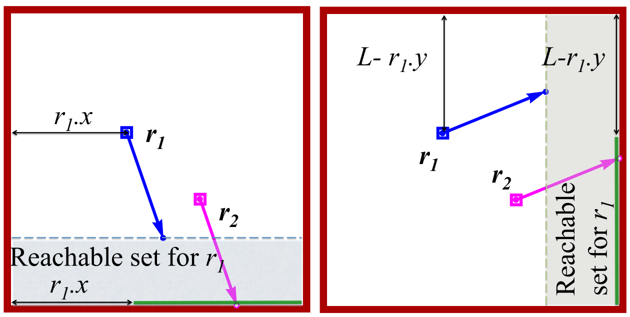

Fig. 7. Boundary interaction is used to change the relative positions of the robots. Each robot gets the same control input. (left) If robot 2 hits the bottom wall before robot 1 reaches a wall, robot 2 can reach anywhere along the green line, and robot 1 can move to anywhere in the shaded area. (right) 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.

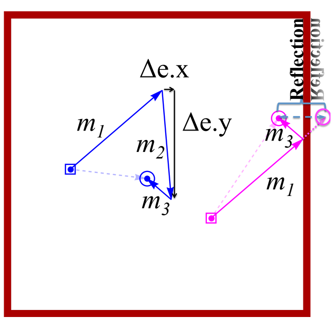

Fig. 8. If the goal configuration can be reached in three moves, the first move makes one particle hit a wall, the second move adjusts the relative spacing error Δe to zero, and the third move takes the particles to their final positions. The second move cannot be shortened, so optimization depends on 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.

and the red robot is in the red plate. There are two possible optimal paths shown with parallel arrows. Each arrow will cause one of the particles to touch the wall, enabling the other robot to move freely in z axis to make the required relative position.

V. Simulation

A. Position Control of Two Robots

Algorithm 1 was implemented in Mathematica using point robots (radius = 0).

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 $\leftarrow \{\}$
$P \leftarrow \{h(\{\}, r_1, r_2, g_1, g_2), \{\}, r_1, r_2\}$ ▷ P contains h, the admissible heuristic (8), the move sequence, and the current robot positions
while $P.r_1 \neq g_1$ **and** $P.r_2 \neq g_2$ **do**
  for $\theta \in \{0°, 90°, 180°, 270°\}$ **do**
    $(r_1, r_2, g_1, g_2) \leftarrow$ ROTATE($P.r_1, P.r_2, g_1, g_2, \theta$)
    $\{d,$ *moves*$, r_1, r_2\} \leftarrow$ PLANMOVEUP($r_1, r_2, g_1, g_2, L, P.$*moves*)
    $($*moves*$, r_1, r_2) \leftarrow$ ROTATE(*moves*$, r_1, r_2, -\theta$)
    PUSH $\{d,$ *moves*$, r_1, r_2\}$ onto *PathList*
  end for
  SORT(*PathList*) ▷ sort by admissible heuristic
  $P \leftarrow$ POP first element of *PathList*
end while
return *moves*

The contour plots in Fig. 10 left shows the length of the path for given s_1, s_2, g_1 with g_2 ranging over all the workspace. Fig. 10 right shows the total distance of the path. This plot clearly shows the nonlinear nature of the path planning. The hardest point to achieve is the when the goals have π difference and are very close to the boundary.

The plots in Fig. 11 show the exponentially increasing number of moves and distance when the accuracy of reaching to the goal (δ) is getting to zero when the goal positions have π difference with each on the boundaries.

VI. Experimental Results

VII. 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 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

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. ϵ is a small, nonzero, user-specified value.

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

$\Delta e \leftarrow (g_2 - g_1) - (r_2 - r_1)$
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 $\{h(moves, r_1, r_2, g_1, g_2), moves, r_1, r_2\}$

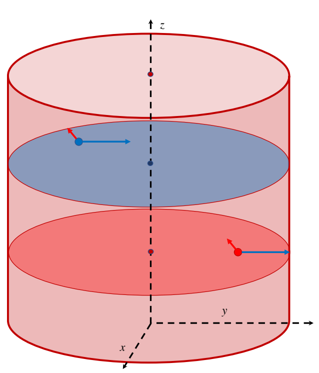

Fig. 9. Extending the algorithm to position the particles in 3D.

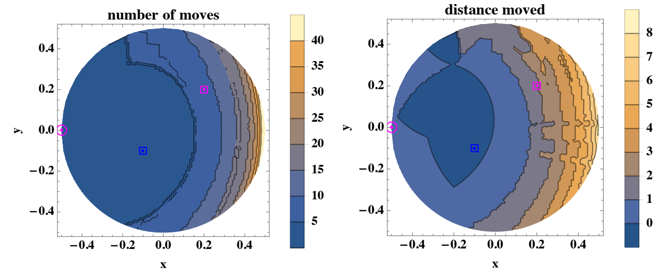

Fig. 10. Plots showing the algorithm with one goal on the boundary.

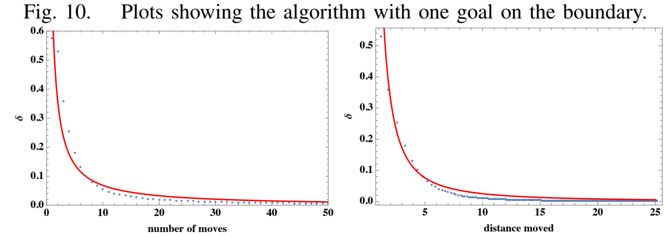

Fig. 11. Plots showing decreasing error when the number of moves grows.

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] S. Shahrokhi, A. Mahadev, and A. T. Becker, “Algorithms for shaping a particle swarm with a shared input by exploiting non-slip wall contacts,” in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, 2017.

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

[10] 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.

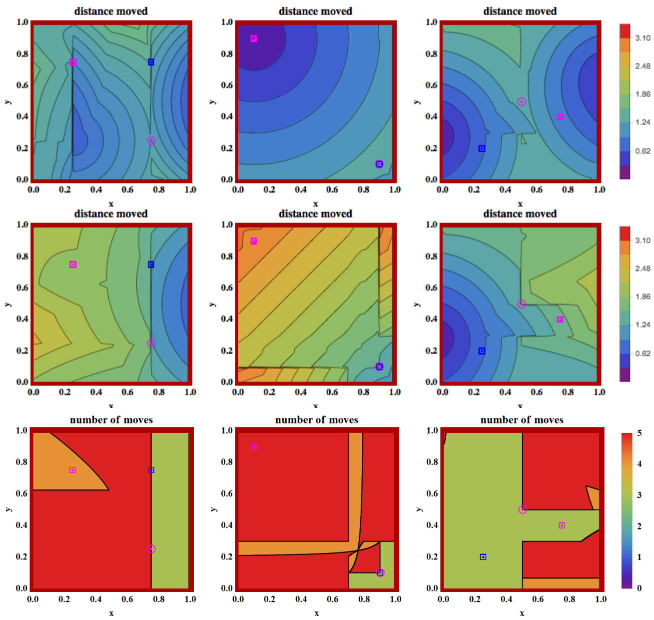

Fig. 12. Starting positions of robots 1 and 2 and goal position of robot 2 are fixed, and $\epsilon = 0.001$. The top row of contour plots show the distance if robot 1's goal position is varied in x and y. The bottom row shows the number of moves required for the same configurations.

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[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.