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 [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**]. 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.

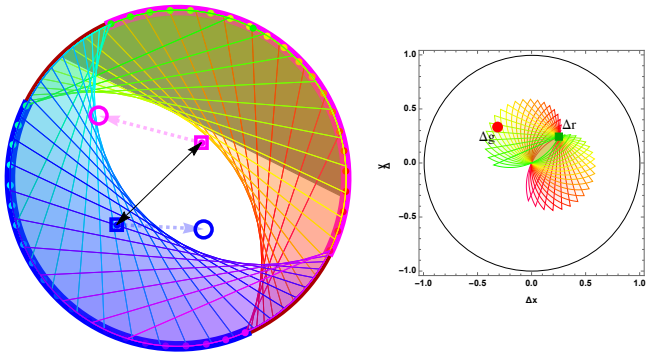

Fig. 1. Positioning particles that receive the same control inputs, but cannot move while a control input pushes them into a boundary. Top row shows the output of a best-first-search algorithm that finds the shortest path for two particles. Top left shows the initial position and goal position of the 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. **??** 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. **??** describes implementations of the algorithms in simulation and Sec. **??** describes hardware experiments, as shown in Fig. 1. We end with directions for future research in Sec. V.

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 [**?**], and the gradient-based decentralized controllers using control-Lyapunov functions in [**?**]. However, these approaches assume a level of intelligence and autonomy

*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

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 [**?**], [**?**], [**?**] 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 [**?**], or engineered inhomogeneity [**?**], [**?**], [**?**]. This work assumes a uniform control with homogenous agents, as in [**?**]. 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 [**?**]. [**?**] 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^{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 [**?**]. 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 $\mathbf{u}(t)$ [**?**]. 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. **??** 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. **??**a. 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. 2. The horizontal (and vertical) reachable sets are equivalent in the Δ 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 Δ configuration space, as shown in Fig. 3.

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. 4. 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. 5 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.

$$h(\textit{moves}, r_1, r_2, g_1, g_2) = \sum_{i=1}^{|\textit{moves}|} \|\textit{moves}_i\| \tag{2}$$

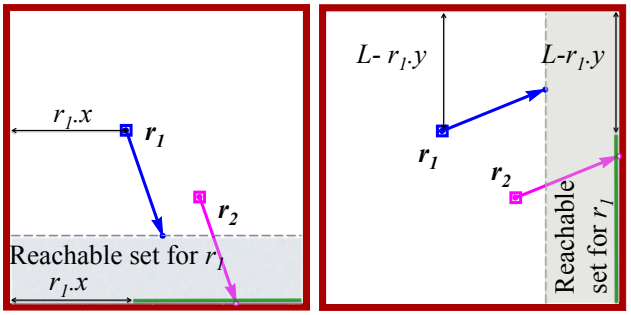

Fig. 2. Boundary interaction is used to change the relative positions of the robots. Each robot gets the same control input. (left) If robot 2 hits the bottom wall before robot 1 reaches a wall, robot 2 can reach anywhere along the green line, and robot 1 can move to anywhere in the shaded area. (right)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

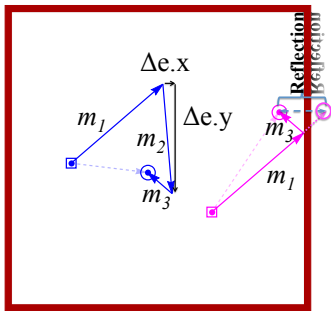

Fig. 4. If the goal configuration can be reached in three moves, the first move makes one particle hit a wall, the second move adjusts the relative spacing error Δe to zero, and the third move takes the particles to their final positions. The second move cannot be shortened, so optimization depends on 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.

$$+ \max(\|g_1 - r_1\|, \|g_2 - r_2\|)$$

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. 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 **??** 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.

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 the admissible heuristic, 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*

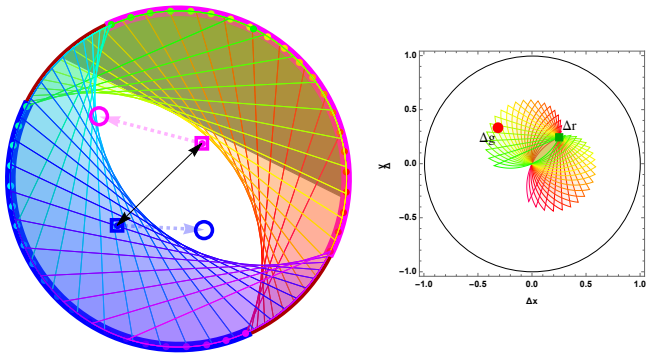

Fig. 3. Workspace and Δ configuration space for three sets of robot configurations with the same final goal. The red square represents the starting Δx and Δy and the green circle represents the goal Δx and Δy. The green rectangle illustrates reachable Δx and Δy when one particle is in contact with a horizontal wall and the blue rectangle illustrates the reachable region when in contact with a vertical wall.

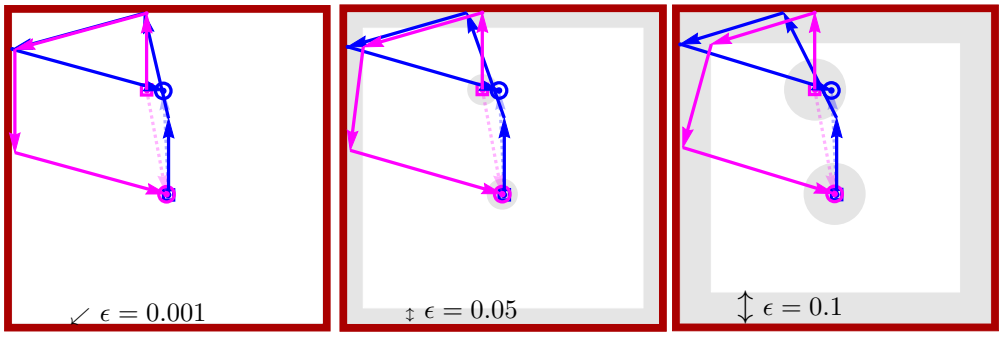

Fig. 5. Changing the minimum spacing ϵ changes the path. ϵ is the minimum spacing between two robots and the minimum separation from the boundaries.

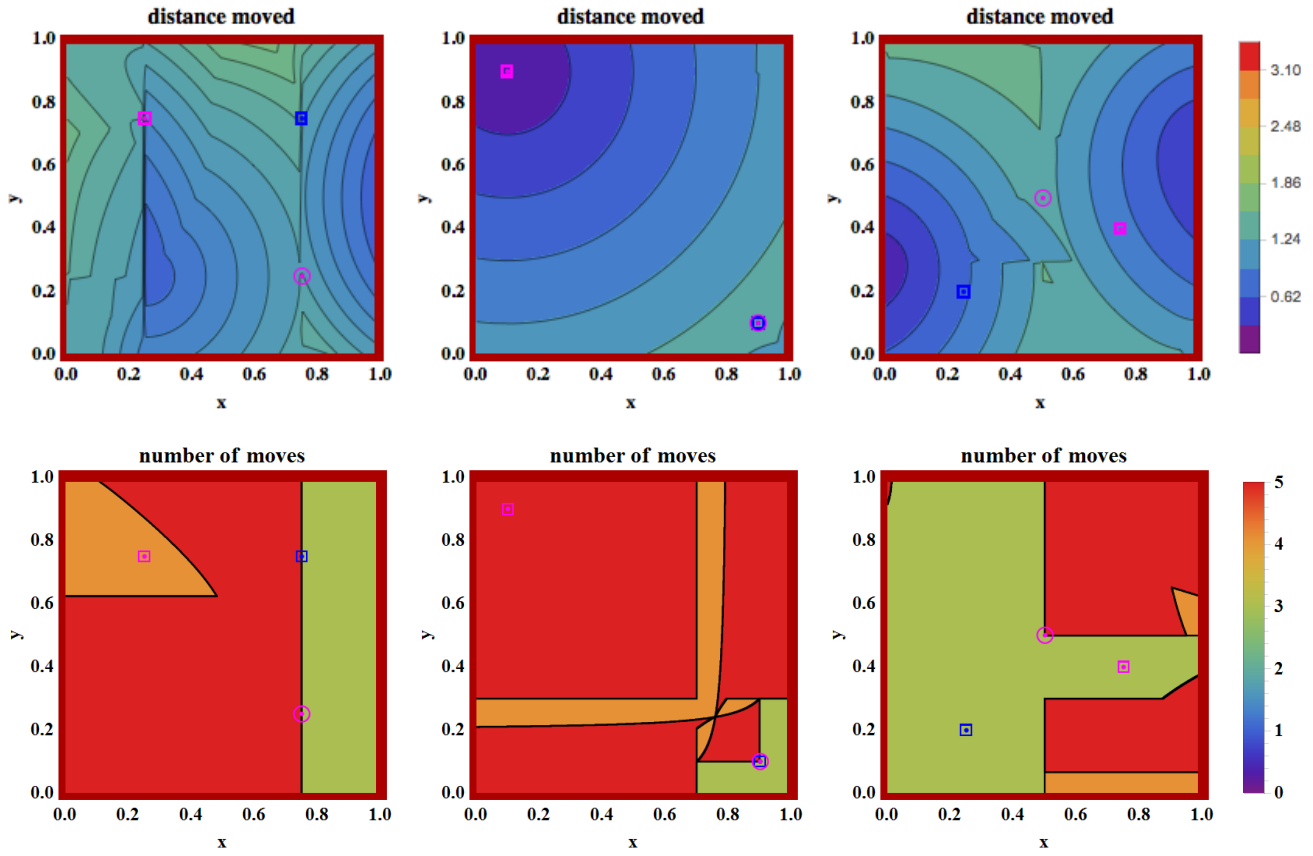

Fig. 6. Starting positions of robots 1 and 2 and goal position of robot 2 are fixed, and $\epsilon = 0.001$. The top row of contour plots show the distance if robot 1's goal position is varied in x and y. The bottom row shows the number of moves required for the same configurations.

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*$, r_1, r_2\}$