Aggregation and localization of simple robots in curved environments

Rachel A. Moan Victor M. Baez Aaron T. Becker Jason M. O'Kane

***Abstract*—This paper is about the closely-related problems of localization and aggregation for extremely simple robots, for which the only available action is to move in a given direction as far as the geometry of the environment allows. Such problems may arise, for example, in biomedical applications, wherein a large group of tiny robots moves in response to a shared external stimulus. Specifically, we extend the prior work on these kinds of problems presenting two algorithms for localization in environments with curved (rather than polygonal) boundaries and under low-friction models of interaction with the environment boundaries. We present both simulations and physical demonstrations to validate the approach.**

I. Introduction

The problem of localization —that is, the process of determining the location of a robot with respect to its current environment— is a fundamental problem in robotics. Traditional approaches to this problem [7], [10], [11], [26], [31] are suitable for robots of sufficiently large scale and sufficiently high complexity that they can carry range sensors, cameras, or other information-rich sensors for perceiving their own motions through their environments and relatively precise actuators for effecting those motions. The localization problem becomes more challenging when the robots are extremely small, when they lack strong sensing and actuation abilities, or both.

Future (and some current) biomedical robots have both challenges. Micro and nano devices have little room onboard for computation and little storage space for the energy required for propulsion. For overviews, see the excellent surveys in [5], [6], [22], [25], [30], [32], which outline both the diverse applications of tiny robots inside the body, and the challenges of sensing and control of tiny robots. Instead of internal computation and propulsion, these biomedical devices are propelled by an external source, by biological processes (such as blood flow) or by diffusion.

Many tasks for such robots including drug delivery, clotting, and targeted therapy can be characterized as aggregation tasks, in which devices spread through an environment are gathered in a single location. Motivated by the the potential for these kinds of applications —and noting that, at small scales, localizing a single robot and aggregating many robots become the same essential problem— we consider the localization problem for a very simple robot with only a single capability: that of moving in a commanded direction until it reaches the obstacle boundary. This behavior could be implemented, for example, using a traditional mobile robot equipped with a compass and a contact sensor, or by a swarm of medicine-bearing micro-robots suspended in a fluid and responding to an externally supplied magnetic field [17].

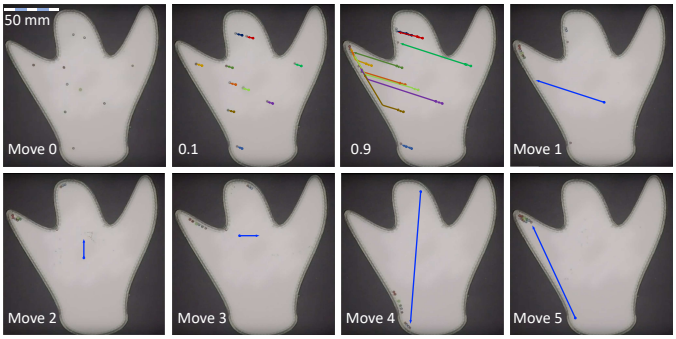

Fig. 1. A collection of particles spread through a known environment with a curved boundary. The particles are aggregated in response to carefully planned global translations of the underlying substrate. The frames marked 0.1 and 0.9 depict the particles in the process of executing the first move. This corresponds to Sequence 1 in the attached video, https://youtu.be/fVhFc41T88I.

Prior work by O'Kane and LaValle [23] described a family of localization algorithms for several types of extremely simple robots, including a model similar to the setting described above. However, the applicability of that work was strongly limited by a deep reliance on polygonal models of the robot's environment. Moreover, their algorithms depended in crucial ways on an assumption that the robot's motion stops immediately when it comes into contact with the environment boundary, without any 'sliding' behavior. Many biological systems are slippery; nearly all are non-polygonal [13], [18], [29]. Thus, in this paper, we show how to generalize that prior approach to a substantially more realistic setting that removes these two limitations. Specifically, we model the environment boundary as a composite cubic Bézier curve, and consider movement models and planning algorithms that can account for robots that may slip along the boundary after reaching it. See Figure 1.

The main contribution of this paper is a planning algorithm for this problem, along with demonstrations of the effective-

R. A. Moan is with the Department of Computer Science, Winthrop University, Rock Hill, South Carolina, USA. V. Montano Baez and A. T. Becker are with the Department of Electrical and Computer Engineering, University of Houston, Houston, Texas, USA. J. M. O'Kane is with the Department of Computer Science and Engineering, University of South Carolina, Columbia, South Carolina, USA. moanr2@mailbox.winthrop.edu, {vjmontano, atbecker}@uh.edu; jokane@cse.sc.edu This material is based upon work supported by the National Science Foundation under Grant [IIS-1659514], [IIS-1553063], and [IIS-1619278].

ness of this approach both in simulation and in laboratory experiments. Our results are applicable both for actively *localizing* a single robot from a collection of possible starting positions, and for *aggregating* a collection of simple robots moving in response to common control signals within the same environment.

The remainder of the paper is organized as follows. Sections II and III review related work and formally define the problem, respectively. Then, in Section IV, we describe how to solve the passive problem of predicting the robot's (or robots') movements in response to given movement commands, under both 'sticking' and 'sliding' models of motion. In Section V, we present a planning algorithm that generates motion sequences to localize the robot. Results from both simulated and physical executions appear in Section VI, before concluding discussion in Section VII.

II. Related Work

Effective localization, which is widely viewed as essential for robot autonomy, has been intensely studied for systems with a variety of sensor systems [15], [16], [20], [24]. Representatives of the wide variety of well-known solution approaches include techniques based on the Kalman filter [7], [26], Markov approaches [10], [11], and Monte Carlo techniques [31].

An important distinction can be made between passive localization, which estimates the robot's position using an incoming stream of data and is based on the assumption that the motion of the robot cannot be controlled, and active localization, which assumes that the robot's movement and orientation can be partially or fully controlled [4].

Relative localization requires that the approximate position of the robot is known, whereas global localization can determine the position of a robot without any prior knowledge of its position [12], [28] This paper considers an active global localization problem.

Specifically, we are interested in solving such problems with robots whose sensing and movement capabilities are severely limited. Active, global localization algorithms have been studied from this perspective [9], [23], but only in polygonal environments, and only executed on traditionally-sized (dozens of centimeters) mobile robots.

Micro and nano-scale robots are often steered by a global field, such as a magnetic, gravitational, or electric field in such a way that all the robots get approximately the same input commands. See overviews in [3], [5], [6], [25], [30]. Dealing with swarms of tiny robots offers another approach to localization, by designing a control sequence that if executed would steer all particles to a common destination.

Huang et al. studied this problem using a magnetic field to aggregate microscale iron particles at a goal location in [13]. In a discretized world where particles move on a bounded polyomino grid, Mahadev et al. proved that iteratively moving one particle to another was sufficient to collect all the particles to a single point in $O(n^3)$ time, where n is the total number of free spaces in the polyomino [18].

Another related problem is that of using gravity to drain water out of a punctured polygon by rotating the shape [1]. For a polygon with n vertices, they provide an $O(n^2 \log n)$ time algorithm to determine the minimum number of holes required to drain the polygon.

III. Problem Statement

In this section, we give a formal definition of the problem addressed by our algorithm. The two central components are the environment (Section III-A) and the robot model (Section III-B).

We are interested in both localization of individual simple robots, and in aggregation of collections of multiple robots in a shared workspace. However, in this context, the problems share much of their essential structure: we can use the same approach to reason about a set of candidate locations for a robot in the process of localizing itself as for a set of actual robot locations during aggregation. Thus, to simplify the explanation, we describe our approach using language attuned to the localization interpretation.

A. The environment

A robot moves within a bounded, planar, simply-connected environment $E \subset \mathbb{R}^2$. We assume that the boundary of E, denoted ∂E, can be described as a *composite cubic Bézier curve*. That is, the boundary of E is described by a finite sequence of n curves $c^{(1)}, \ldots, c^{(n)}$. Each curve c_i is a parametric curve defined by four *control points* $P_0^{(i)}$, $P_1^{(i)}$, $P_2^{(i)}$, and $P_3^{(i)}$. A scalar parameter t varies across the range $[0, 1]$ in the equation

$$c^{(i)}(t) = (1-t)^3 P_0^{(i)} + 3(1-t)^2 t P_1^{(i)} + 3(1-t)t^2 P_2^{(i)} + t^3 P_3^{(i)}. \quad (1)$$

To ensure that these curves define a continuous boundary for E, we require that the endpoints of successive curves coincide. That is, for all $1 \leq i \leq n$, we assume that $P_3^{(i)} = P_0^{(i+1)}$. Similarly, to ensure that the curves describe a closed boundary for E, we require that $P_3^{(n)} = P_0^{(1)}$. Finally, to ensure that the boundary of E is differentiable, for all $1 \leq i \leq n$, we assume that $P_2^{(i)}$, $P_3^{(i)}$, and $P_1^{(i+1)}$ are collinear with $P_3^{(i)}$ in the middle of the three. We assume that the curve is oriented so that increasing values of t travel counterclockwise around the boundary. See Figure 2 for an example.

B. The robot model

For simplicity, we model the robot as a point in E. (If the robot has a non-zero radius, as real robots generally do, we can adjust E so that its boundary corresponds to the set of points that can be occupied by the robot's center point without causing a collision.) Time proceeds in a series of discrete *stages*, indexed $k = 1, 2, 3, \ldots$. At each stage k, the robot occupies a *state* $x_k \in E$.

The robot's motion at stage k is determined by its selection of an *action* $u_k \in U$, in which U is the set of unit vectors in the plane. The intuition is that each action u_k describes a motion direction for the robot in stage k.

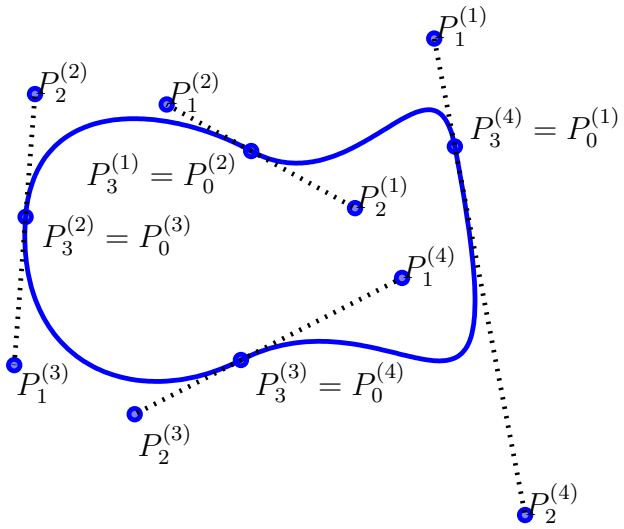

Fig. 2. An example environment with $n = 4$ curves. The 12 distinct control points are shown. Dashed segments illustrate the colinearity constraints.

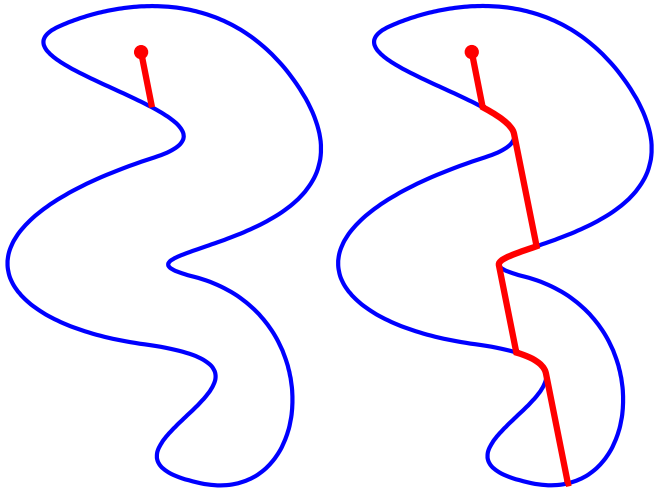

Fig. 3. [left] Motion of the robot under the sticky (f_{stk}) motion model. [right] Motion under the slipping (f_{slp}) motion model.

We consider two distinct models for how the robot's actions change its state.

1) Under the *sticky model*, the robot moves from position x_k in direction u_k as far as possible without leaving E. That is, the robot moves until it reaches the boundary, and then stops at that point. This behaviour might be realized, for example, either via direct sensing or via friction between the robot and the environment. We express this motion model as a *state transition function* $f_{\text{stk}} : E \times U \to E$, under which $x_{k+1} = f_{\text{stk}}(x_k, u_k)$. Details about how to compute f_{stk} appear in Section IV-A.
2) Under the *slipping model*, the robot moves in direction u_k, possibly sliding along ∂E, until it reaches a local maximum of ∂E in that direction. We write $f_{\text{slp}} : E \times U \to E$ and $x_{k+1} = f_{\text{slp}}(x_k, u_k)$ for the state transitions that occur under this motion model. Details are in Section IV-B.

Figure 3 illustrates these two models, which are intended to capture two extremal cases for how our robots may behave upon coming in contact with the environment. Which of them is most appropriate in a given setting depends on the physical characteristics of the robot and its interactions with the boundary of E. When the difference between f_{stk} and f_{slp} is not relevant, we instead write a generic f for whichever of the two is apropos. We also extend the notation to sequences of actions, writing $f(x_k, u_k, u_{k+1}, \ldots, u_K)$ instead of $f(\cdots f(f(x_k, u_k), u_{k+1}) \cdots), u_K)$.

C. The active localization problem

At the start of its execution, the robot does not know its own location. However, we assume that the robot does have access to a finite list of possible starting locations, which we denote $\eta_1 \subseteq E$.

As the robot moves, it can update this set in correspondence with the actions it selects. More precisely, we can define an *information transition function* $F : 2^X \times U \to 2^X$, in which

$$F(\eta_k, u_k) = \cup_{x_k \in \eta_k} \{f(x_k, u_k)\}. \tag{2}$$

In this way, we can define a sequence of *information states*, staring from η_1:

$$\eta_{k+1} = F(\eta_k, u_k). \tag{3}$$

Notice that, under this model, the robot does not have access to any direct feedback from its sensors. Thus, the passive view of the localization problem viewed by some as traditional — namely, that the localization algorithm should merely process sensor and action data, without any control over which actions the robot executes— is quite unsuitable here. Indeed, even the typical 'active localization' viewpoint is a poor fit, because the lack of direct sensor data in the model means that the robot cannot adjust its strategy based on its own observations. As a result, the robot's localization strategy can be expressed simply as a *finite sequence of actions*, under which any starting point in η_1 is driven to the same final point. That is, we want to select an action sequence $u_1, \ldots, u_K$ such that, for any $x, x' \in \eta_1$,

$$f(x, u_1, \ldots, u_k) = f(x', u_1, \ldots, u_k). \tag{4}$$

Equivalently, we seek an action sequence that reduces the size of the information state down to a single point:

Problem: Localization in a curved boundary

Input: An environment E described by its $3n$ distinct control points, a finite list η_1 of starting locations, and a motion model f (either f_{stk} or f_{slp}).

Output: A sequence of actions $u_1, \ldots, u_K$, for which $|\eta_K| = 1$.

IV. Computing action effects

Before describing how to select an action sequence that localizes the robot, we first consider in this question the passive problem of determining, under the models introduced in Section III, what the effects a given action will be. That is, if the robot is at position x_k and executes a motion in direction u_k, what location x_{k+1} will the robot reach? We begin with the sticking model (Section IV-A) and then extend that approach to the slipping model (Section IV-B).

A. Computing action effects under f_{stk}

Under the sticking model f_{stk}, at stage k, the robot starts at x_k and moves in direction u_k as far as possible while remaining within E. The resulting location is $f_{\text{stk}}(x_k, u_k)$, that is, x_{k+1}. Computing this x_{k+1} is, in essence, a form of ray shooting query in E.

We can parameterize the line along which the robot moves as $\ell(s) = x_k + su_k$ and find its intersections with a single curve $c^{(i)}$ along ∂E by setting $\ell(s) = c^{(i)}(t)$, yielding a vector equation in the two parameters s and t:

$$x_k + su_k = at^3 + bt^2 + ct + d, \tag{5}$$

in which the vector-valued constant coefficients on the right-hand side are

$$\begin{aligned} a &= -P_0^{(i)} + 3P_1^{(i)} - 3P_2^{(i)} + P_3^{(i)} \\ b &= 3P_0^{(i)} + 6P_1^{(i)} + 3P_2^{(i)} \\ c &= -3P_0^{(i)} + 3P_1^{(i)} \\ d &= P_0^{(i)}. \end{aligned}$$

Viewing Eq. 5 as a system of two scalar equations, we eliminate s by solving each of the scalar equations for s and equating the results. This yields a cubic equation in t,

$$At^3 + Bt^2 + Ct + D = 0,$$

in which the coefficients are

$$\begin{aligned} A &= a_x x_{ky} - a_y x_{kx} \\ B &= b_y u_{kx} + x_{kx} x_{ky} \\ C &= c_y u_{kx} + x_{kx} x_{ky} \\ D &= (u_{kx} - x_{kx})(d_y - x_{ky} - u_{ky}) \\ &\quad + x_{ky}(d_x + x_{kx} + u_{kx}). \end{aligned}$$

The real solutions of this equation within the interval $t \in [0, 1]$ provide the intersection points between the given curve and the line along which the robot is moving.

Iterating this process over all n curves gives a set of at most $3n$ candidates for x_{k+1}. For each, we can compute the parameter s that determines how far ahead the robot would have moved to reach that point. In general, we select the point with the smallest non-negative s, since negative values of s represent backward motion. However, if $x_k \in \partial E$ — an extremely common occurrence after the first stage, since our robot cannot stop except when it reaches the environment boundary— we will obtain a candidate point at x_k itself, that is, with $s = 0$, which must be handled specially.

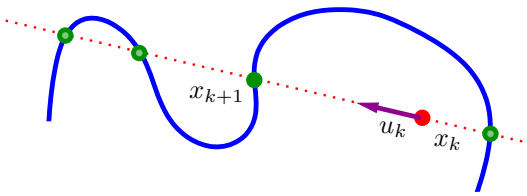

Fig. 4. Computing f_{stk}. The algorithm computes a collection of candidate locations by intersecting the boundary curves with a line, and then selects the nearest eligible point from amongst the candidates.

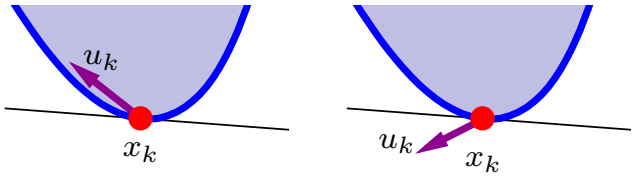

Fig. 5. Handling the case where $s = 0$ to determine whether a movement in direction u_k would collide immediately (right) with the environment or not (left).

Specifically, when $s = 0$, we rely upon the counterclockwise presentation of ∂E assumed in Section III and test (using the standard clockwise test from computational geometry) whether the three points (i) x_k, (ii) $x_k + v$ where v is the counterclockwise tangent vector at x_k, and (iii) $x_k + u_k$ are arranged in clockwise order. If so, then the robot's motion is directly into the boundary, and the robot does not move. If those three points are counterclockwise, the robot can move freely away from x_k, and that candidate is ignored. See Figure 5.

B. Computing action effects under f_{slp}

Next, we turn to the slipping motion model f_{slp}. As with the f_{stk}, the robot starts at x_k and moves in direction u_k until reaching the environment boundary. From there, f_{slp} differs in that the robot may 'slide,' due to extremely low friction between itself and the environment boundary. It continues to move until it reaches a local maximum of ∂E in direction u_k.

The robot's motion within a single stage can thus be characterized as alternating, perhaps several times, between *jumping* motion within the interior of ∂E and *sliding* motion, along ∂E. Algorithm 1 summarizes this process. Changes in the robot's motion computed by this algorithm can be characterized as *jumps*, where the robot reaches a point at which the tangent to the environment boundary is parallel to the motion direction, and *extrema*, at which the tangent to the environment boundary is orthogonal to the motion direction.

To compute such points for a given boundary curve $c^{(i)}$, we need to find values of $t \in [0, 1]$ for which the tangent vector $c'(t)$ is parallel or antiparallel to u_k. This occurs when $u_k \cdot c'^{(i)}(t)^{\perp} = 0$. Equating these two implicit scalar equations gives a quadratic in t:

$$(a_x u_{ky} - a_y u_{kx})t^2 + (b_x u_{ky} - b_y u_{kx})t + (c_x u_{ky} - c_y u_{kx}) = 0,$$

Algorithm 1 COMPUTESLIPPINGMOTION(x_k, u_k, E)

```
jumps ← JUMPPOINTS(u_k, E)
extrema ← EXTREMEPOINTS(u_k, E)
events ← jumps ∪ extrema
x_curr ← x_k
while x_curr ∉ extrema do
    x_curr ← f_stk(x_curr, u_k)
    v ← counterclockwise tangent vector of ∂E at x_curr
    if cw(x_curr, x_curr + v, x_curr + u_k^⊥) then
        x_curr ← nearest point to x_curr from events, measured
                 in clockwise distance around ∂E.
    else
        x_curr ← nearest point to x_curr from events, measured
                 in counterclockwise distance around ∂E.
    end if
end while
x_{k+1} ← x_curr
return x_{k+1}
```

Algorithm 2 CHOOSEMOTIONDIRECTIONS(E, η_1, f)

```
k ← 0
while |η_k| > 1 do
    (p, q) ← pair of distinct points in η_k
    while p ≠ q do
        k ← k + 1
        if pq ⊂ E then
            u_k ← (q − p)/||q − p||
        else
            S(p, q) ← region in E hidden from p, containing q
            z ← anchor point of S(p, q)
            u_k ← (z − p)/||z − p||
        end if
        η_k ← F(η_k, u_k)
    end while
end while
return u_1, ..., u_k
```

in which a, b, c, and d are the vector-valued constant coefficients of $c'^{(i)}$:

$$\begin{aligned} a &= -3P_0^{(i)} + 9P_1^{(i)} - 9P_2^{(i)} + 3P_3^{(i)} \\ b &= 6P_0^{(i)} - 12P_1^{(i)} + 6P_2^{(i)} \\ c &= -3P_0^{(i)} + 3P_1^{(i)} \\ d &= P_0^{(i)}. \end{aligned}$$

The real solutions of this equation, for any t on the interval $[0, 1]$, provide the jump points for the path of the robot. Repeating this process for all curves provides a list of all the possible jump points for the robot. The process to compute the extrema nearly identical, but seeks values of t for which $u_k \cdot c'^{(i)}(t) = 0$.

Algorithm 1 uses these points to determine how far the robot slides before its motion stops (in the case of an extreme point) or moves into the interior of E (in the case of a jump point.) This process continues until the robot reaches a local maximum, at which point we have our $f_{\text{slp}}(x_k, u_k) = x_{k+1}$.

V. LOCALIZATION PLANNING

Based on these approaches to passively compute the outcomes of a given action, we can now attack the problem of generating, for a given environment E, a set of possible initial positions η_1, and a choice of motion model f; a sequence of actions $u_1, \ldots, u_k$ that localizes our robot.

The overall approach appears as Algorithm 2. The algorithm constructs the plan sequentially, maintaining η_k, selecting u_k, and then moving to stage $k + 1$ by computing η_{k+1}. The underlying idea is to consider just a single pair (p, q) of distinct possible locations for the robot, and to generate motions that drive those two points from their distinct current locations to the same final location. In our implementation, we select p and q at random. The locations of the other points are updated along the way. After p and q are merged, the algorithm selects a new pair of distinct possible points and repeats the process. When only a single possible location remains, the localization plan is complete.

How, then, can we select actions that bring p and q together? One approach is based on the following idea.

Observation 1 (actions for visible points): If the line segment connecting p to q is contained entirely within E, then

1) $f_{\text{stk}}(p, p - q) = f_{\text{stk}}(q, p - q)$, and
2) $f_{\text{slp}}(p, p - q) = f_{\text{slp}}(q, p - q)$.

This provides a mechanism for the case in which p and q can 'see' each other in E: We simply move in a direction parallel to the line segment between those two points.

In the general case, however, it is possible that $\overline{pq} \not\subset E$. In such a case, we let $S(p, q) \subset E$ denote the maximal connected portion of E not visible to p but containing q. This region has an 'anchor point' z at which its bounding ray is tangent to E. See Figure 6. Under f_{stk}, we can make progress toward merging p with q by moving toward this point z:

Observation 2: Let $p = f_{\text{stk}}(p, z - p)$ and $q' = f_{\text{stk}}(q, z - p)$. Then either $\overline{p'q'}$ or $S(p', q') \subset S(p, q)$.

Thus, the robot chooses direction $z - p$. Under f_{slp}, the process is the same, though we cannot make as strong a guarantee that the S regions will decrease monotonically at each step. This overall process repeats until p and q are merged, after which we select a new p and a new q. When the candidate points have all been merged with each other, the localization plan is complete and the algorithm terminates.

VI. EXPERIMENTAL RESULTS

This section presents the results of our experimental evaluation of the approach. We show some computed examples, describe the results of a quantitative comparison against a baseline, and show a physical demonstration.

A. Simulated examples

Figure 7 shows the simulated execution of paths computed for both f_{stk} and f_{slp}. Each plan was computed based on a set

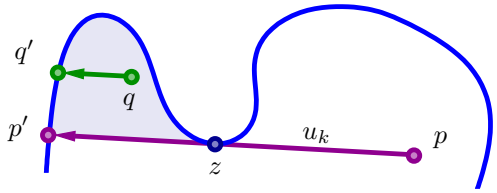

Fig. 6. Selecting motions that merge p and q. If p cannot see q, it moves toward the tangent point of ∂E that hides q from p.

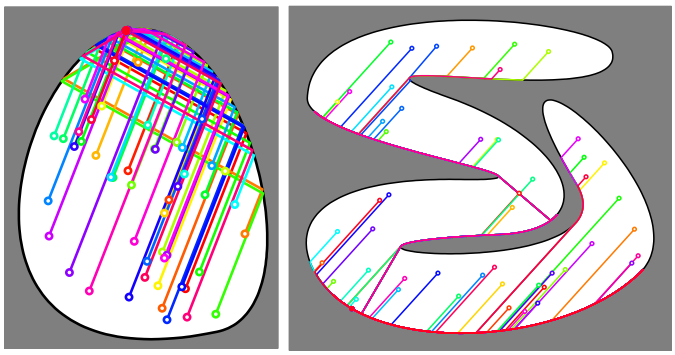

Fig. 7. Simulated paths generated by our algorithm. Starting locations are shown with open circles; the single final location is shown with a filled circle. [left] A sequence of 11 actions generated under f_{stk} for a simple environment. [right] A sequence of 4 actions generated under f_{slp} for a more complex environment. See video for animation [21].

η_1 of 50 randomly selected points. From these starting points, our algorithm generated plans of length 11 and 4 respectively. In both cases, from each starting point, executing the same action sequence, the simulated robots all reached the same final point.

B. Quantitative evaluation

We evaluated the success of our algorithm quantitatively by measuring the average distance between the particles after each action was executed. Using the environment in Figure 1, we selected 25 random starting positions, and executed Algorithm 2, using both f_{stk} and f_{slp}. We then simulated those plans and periodically calculated the mean distance between the particles, as a measure of the progress toward localization; when the distance reaches 0, the robot is localized. For comparison purposes, we also implemented a simple algorithm that selects motion directions uniformly at random. (There is some evidence that simple random plans can be surprisingly successful in contexts like this [19].) The results, which appear in Figure 8, show that our approach achieves a meaningful improvement in the efficiency of localization compared to this baseline.

C. Physical proofs-of-concept

There are many methods to generate global inputs on a 2D set of particles, ranging from gravity-based tilting [8], [19], using light to steer kilobots [27], or magnetic fields on cells [2] or particles [14]. In this work, we use a tabletop made of white tile hardboard (often used to make markerboards). The boundaries are laser cut from 6 mm thick acrylic and the particles are 2 mm diameter glass seed beads. When the acrylic

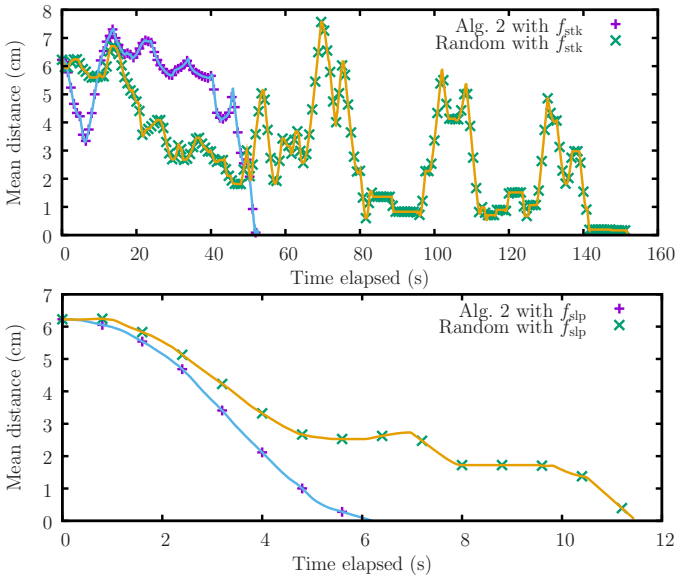

Fig. 8. Comparison of localization progress as a function of time. [top] Under f_{stk} [bottom] Under f_{slp}

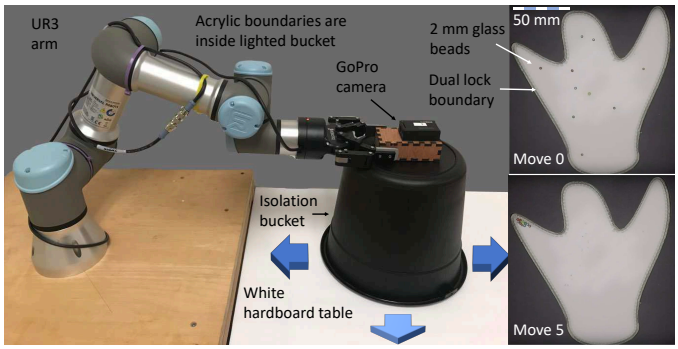

Fig. 9. A robot arm executes a plan generated by Algorithm 2 using a high-friction scenario, solved using f_{stk}. [right] two snapshots from the sequence. See video https://youtu.be/fVhFc41T88I for full experiment [21].

boundaries are translated above the table, the seed beads stay in place until a boundary wall touches them. We then place an inverted plastic bucket over the acrylic tray and attach them. The bucket is filled with LED lights and a camera is affixed above the assembly, looking down. The bucket is then translated by a UR3 robot arm along a precomputed trajectory. In this work we used two different walls. The unadorned acrylic walls reproduce the slipping (f_{slp}) motion model, while attaching a thin strip of Dual Lock (3M Reclosable Fastener) reproduce the sticky (f_{stk}) motion model.

VII. Conclusion and Future Work

This paper presented a localization/aggregation technique for extremely simple robots within a Bézier curve boundary. A number of interesting questions remain for future work. Chief among them is the question of optimality; rather than arbitrarily choosing a pair of points to merge, one might attempt plans that take a more global view, with an eye toward minimizing the plan's execution time. Interesting questions also remain about motion models situated between the two extremes considered here.

REFERENCES

[1] G. Aloupis, J. Cardinal, S. Collette, F. Hurtado, S. Langerman, and J. ORourke, "Draining a polygon —or— rolling a ball out of a polygon," *Computational geometry*, vol. 47, no. 2, pp. 316–328, 2014.

[2] D. Arbuckle and A. A. Requicha, "Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: algorithms and simulations," *Autonomous Robots*, vol. 28, no. 2, pp. 197–211, 2010.

[3] A. T. Becker, "Controlling swarms of robots with global inputs: Breaking symmetry," in *Microbiorobotics*. Elsevier, 2017, pp. 3–20.

[4] W. Burgard, D. Fox, and S. Thrun, "Active mobile robot localization," in *IJCAI*, 1997, pp. 1346–1352.

[5] X.-Z. Chen, M. Hoop, F. Mushtaq, E. Siringil, C. Hu, B. J. Nelson, and S. Pane, "Recent developments in magnetically driven micro-and nanorobots," *Applied Materials Today*, vol. 9, pp. 37–48, 2017.

[6] 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.

[7] A. Curran and K. J. Kyriakopoulos, "Sensor-based self-localization for wheeled mobile robots," *Journal of Robotic Systems*, vol. 12, no. 3, pp. 163–176, 1995.

[8] M. A. Erdmann and M. T. Mason, "An exploration of sensorless manipulation," *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 369–379, 1988.

[9] L. H. Erickson, J. Knuth, J. M. O'Kane, and S. M. LaValle, "Probabilistic localization with a blind robot," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1821–1827.

[10] D. Fox, W. Burgard, and S. Thrun, "Active markov localization for mobile robots," *Robotics and Autonomous Systems*, vol. 25, no. 3-4, pp. 195–207, 1998.

[11] ——, "Markov localization for mobile robots in dynamic environments," *Journal of artificial intelligence research*, vol. 11, pp. 391–427, 1999.

[12] P. Goel, S. I. Roumeliotis, and G. S. Sukhatme, "Robust localization using relative and absolute position estimates," in *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, vol. 2. IEEE, 1999, pp. 1134–1140.

[13] L. Huang, L. Rogowski, M. J. Kim, and A. T. Becker, "Path planning and aggregation for a microrobot swarm in vascular networks using a global input," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 414–420.

[14] P. S. S. Kim, A. T. 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.

[15] P. Koch, S. May, M. Schmidpeter, M. Kühn, C. Pfitzner, C. Merkl, R. Koch, M. Fees, J. Martin, D. Ammon *et al.*, "Multi-robot localization and mapping based on signed distance functions," *Journal of Intelligent & Robotic Systems*, vol. 83, no. 3-4, pp. 409–428, 2016.

[16] A. Ledergerber, M. Hamer, and R. D'Andrea, "A robot self-localization system using one-way ultra-wideband communication," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3131–3137.

[17] A. S. Lübbe, C. Alexiou, and C. Bergemann, "Clinical applications of magnetic drug targeting," *Journal of Surgical Research*, vol. 95, no. 2, pp. 200–206, 2001.

[18] A. V. Mahadev, D. Krupke, J.-M. Reinhardt, S. P. Fekete, and A. T. Becker, "Collecting a swarm in a grid environment using shared, global inputs," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2016, pp. 1231–1236.

[19] P. Mannam, A. V. Volkov, R. Paolini, G. Chirikjian, and M. T. Mason, "Sensorless pose determination using randomized action sequences," *Entropy*, vol. 21, no. 2, p. 154, 2019.

[20] F. Martinelli, "A robot localization system combining rssi and phase shift in uhf-rfid signals," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1782–1796, 2015.

[21] R. A. Moan, V. M. Baez, A. T. Becker, and J. M. OKane, "Aggregating simple robots in curved environments," March 2020. [Online]. Available: https://youtu.be/fVhFc41T88I

[22] B. J. Nelson, I. K. Kaliakatsos, and J. J. Abbott, "Microrobots for minimally invasive medicine," *Annual review of biomedical engineering*, vol. 12, pp. 55–85, 2010.

[23] J. M. O'Kane and S. M. LaValle, "Localization with limited sensing," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 704–716, 2007.

[24] T. Pire, T. Fischer, J. Civera, P. De Cristóforis, and J. J. Berlles, "Stereo parallel tracking and mapping for robot localization," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1373–1378.

[25] F. Qiu and B. J. Nelson, "Magnetic helical micro-and nanorobots: Toward their biomedical applications," *Engineering*, vol. 1, no. 1, pp. 021–026, 2015.

[26] S. Rezaei and R. Sengupta, "Kalman filter-based integration of dgps and vehicle sensors for localization," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 6, pp. 1080–1088, 2007.

[27] M. Rubenstein, C. Ahler, N. Hoff, A. Cabrera, and R. Nagpal, "Kilobot: A low cost robot with scalable operations designed for collective behaviors," *Robotics and Autonomous Systems*, vol. 62, no. 7, pp. 966–975, 2014.

[28] S. Se, D. Lowe, and J. Little, "Local and global localization for mobile robots using visual landmarks," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 1. IEEE, 2001, pp. 414–420.

[29] S. Shahrokhi, H. Zhao, and A. T. Becker, "Reshaping particle configurations by collisions with rigid objects," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4436–4443.

[30] M. Sitti, H. Ceylan, W. Hu, J. Giltinan, M. Turan, S. Yim, and E. D. Diller, "Biomedical applications of untethered mobile milli/microrobots." *Proceedings of the IEEE*, vol. 103, no. 2, pp. 205–224, 2015.

[31] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[32] B. Wang, Y. Zhang, and L. Zhang, "Recent progress on micro-and nanorobots: Towards in vivo tracking and localization," *Quantitative imaging in medicine and surgery*, vol. 8, no. 5, p. 461, 2018.