

Path Planning and Aggregation for a Microrobot Swarm in Vascular Networks Using a Global Input

Li Huang¹, Sam Scheckman², Min Jun Kim², and Aaron T. Becker¹

Abstract—Microrobots have great potential for microassembly and non-invasive surgery. Motivated by studies proposing MRI-guided drug delivery to tumor cells using magnetic micro carriers, this paper studies two major challenges of this problem: (i) microrobot swarm trajectory generation, and (ii) swarm aggregation using a global input. We propose an augmented RRT for trajectory generation to reduce environment interference, and a divide-and-conquer algorithm for swarm aggregation to improve performance. Simulations demonstrate the utility of these approaches in comparison to alternate heuristics. Our trajectory generation and aggregation strategies are implemented on a swarm of ferromagnetic micro particles in oil using a 6-coil electromagnetic system with computer vision feedback.

I. INTRODUCTION

Microrobots have great potential to be used in micro-electro-mechanical systems (MEMS) for microassembly or micro manufacturing, and in non-invasive surgery for drug delivery. However, because each agent has limited capabilities for sensing and actuating, external sensors (e.g. MRI, camera) and actuators (e.g. an external electric or magnetic field) must be employed.

Once injected, chemotherapy circulates the human body, but chemotherapy kills healthy cells while killing tumors. To reduce toxic drug exposure to healthy cells, *targeted drug delivery* seeks to steer chemotherapy directly to diseased tissue. Many methods for drug delivery have been explored, including beaded delivery formulations, liposomal delivery systems, encapsulated chemotherapy in nanoparticles, and magnetic micro-carriers navigated by magnetic fields [1].

Recent works have investigated many strategies to manipulate a swarm of simple robots with limited computation and communication [2]–[6]. Fine et al. reported how to actively design environments to assist the process of controlling multiple agents using *shape grammars* [3]. This method addresses automatic generation of environments given specific swarm objective and a control model of agents. [5] showed particle computation methods to perform permutations between different swarm formations by aptly adding obstacles in grid workspace, where they used mobile particles with maximal motion and a global input. Another example of exploiting environment is [6], where a state space is partitioned into discrete transition systems, and gates are configured to guide a swarm of simple robots to achieve state transition,

¹ Li Huang and Aaron T. Becker are with Electrical and Computer Engineering, University of Houston, Houston, TX 77004, USA
lhuang21@uh.edu

² Sam Scheckman and Min Jun Kim are the Department of Mechanical Engineering, Southern Methodist University, Dallas, TX 75275, USA

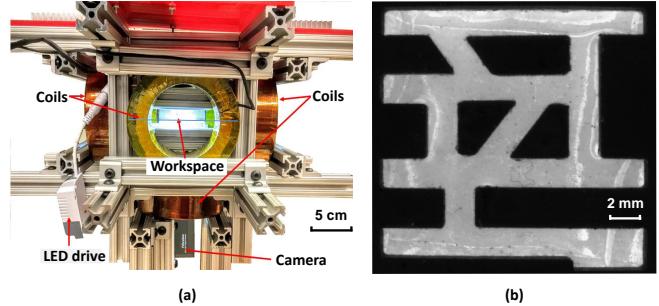


Fig. 1. (a) The six-coil electro-magnetic system. (b) The vascular network tested in the experiment

and thereby to accomplish high-level tasks. [2] proposed a control strategy that by introducing herders to drive a swarm of herding animals to a desired location with repelling potential fields. Experimentally, microrobot swarms such as paramagnetic microparticles, *Tetrahymena pyriformis*, and magnetotactic bacteria have attracted growing attention in many applications of micro-assembly, self-assembly, and targeted therapies, for example, [7]–[11]. These microrobots usually are physically simple agents, and are steered by global fields where every robot receives the same control signal. Many strategies and algorithms have been developed for navigation and motion control of multiple microrobots in free space [12]–[14]. [15] demonstrated control of single microrobot in a micro-fabricated maze. Scheggi et al. implemented and compared six path planning algorithms using magnetic microrobots [16].

Our previous work explored microrobot swarm aggregation in a planar grid environment [17], where we considered micro agents capable of overlapping, directed by a global input, and moving in discrete steps. The valid commands are limited to elements of the set $M = \{\uparrow, \downarrow, \rightarrow, \leftarrow\}$. We presented element-wise algorithms that worked iteratively by selecting two disjoint microrobots, moving the first microrobot until it was maneuvered to the same location as the second, and repeated until all microrobots were collected to the same location.

This paper addresses aggregating a microrobot swarm in vascular networks using only a global input. This is divided into three challenges: (i) generating swarm trajectories, (ii) realizing robust swarm transitions, and (iii) constructing swarm-level strategies to reduce task time complexity. To address (i) and (ii), we use an augmented Rapidly-exploring Random Tree (RRT) for path planning. A divide-and-conquer strategy is employed to address (iii) for swarm aggregation.

Problem formulation and modeling are elaborated in II. Section III and IV introduce trajectory generation and algorithms for aggregation. Section V compares performance with different maps, aggregation methods, and swarm populations. A hardware implementation is described in Section VI.

II. PROBLEM FORMULATION

Given a bounded 2D space $G \subset \mathbb{R}^2$, the plane is partitioned by obstacles into free space (G_{free}), obstacle space (G_{obs}) and contact space (∂G_{obs}), where ∂G_{obs} is the set of boundaries of all obstacles. G_{free} is continuous, and connected by paths of width at least w . A population of n simple microrobots, identified by $r_i, i \in \{1, \dots, n\}$, are randomly distributed in free space at time t_0 , with their positions represented by $\mathbf{p}_t^{r_i} \in G_{free}$. These microrobots are physically simple and have no on-board computation or communication. Their bodies are small compared to w , and interaction between agents is ignored. They are under the control of a global signal, that is, all units move in the same direction at the same speed until they hit an obstacle. We assume that microrobots in contact with the wall have zero velocity if the control input has a component directed into the wall. The goal is to quickly collect this swarm at a gathering point \mathbf{q}^g (Fig. 2). It can be inferred from [17] that there exists a solution to the problem. First, we focus on trajectory generation for each microrobot, and then present control strategies for swarm aggregation.

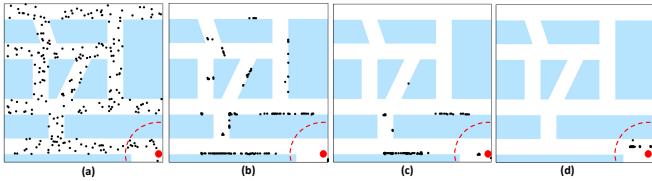


Fig. 2. Blue polygons represent obstacles, and white channels are free space. We place a red dot at each goal location. (a) Aggregation process after 1 step, (b) 200 steps, (c) 500 steps, and (d) 800 steps.

We define a distance metric $d : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}_{\geq 0}$ which denotes the cost to reach \mathbf{p}^j from \mathbf{p}^i via an accessible path in G_{free} . A microrobot is modeled by a discrete-time dynamic system, with a control input vector \mathbf{u}_t :

$$\mathbf{p}_{t+1}^{r_i} = \mathbf{p}_t^{r_i} + \delta \cdot \mathbf{u}_t, \quad (1)$$

where $\mathbf{p}_t^{r_i} \in \mathbb{R}^2$ represents the position of robot r_i at time t , $\|\mathbf{u}_t\| \ll w$, and δ is a scale factor that depends on \mathbf{u}_t and $\mathbf{p}_t^{r_i}$. Let $\bar{\mathbf{p}}_t^{r_i} = \mathbf{p}_t^{r_i} + \mathbf{u}_t$, and $(\mathbf{p}_t^{r_i}, \bar{\mathbf{p}}_t^{r_i})$ be an edge connecting vertices $\mathbf{p}_t^{r_i}$ and $\bar{\mathbf{p}}_t^{r_i}$, then

$$\delta = \begin{cases} \|\mathbf{p}_t^{r_i} - \mathbf{p}^x\|_2, & \text{if } (\mathbf{p}_t^{r_i}, \bar{\mathbf{p}}_t^{r_i}) \cap \partial G_{obs} = \mathbf{p}^x \\ 1, & \text{otherwise.} \end{cases} \quad (2)$$

The swarm has n microrobots $2n$ degree-of-freedom but a global input with only x and y components, so the system is under-actuated.

Let a cost function $F(n, t)$ at time t be the average distance of all robots to the gathering point \mathbf{q}^g :

$$F(n, t) = \frac{1}{n} \sum_{i=1}^n d(\mathbf{p}_t^{r_i}, \mathbf{q}^g). \quad (3)$$

We interpret the trajectory generation problem as determining the single-source shortest paths in a weighted graph using distance metric d . The aggregation problem finds a deterministic policy to decrease the cost function, such that as $t \rightarrow \infty, F(n, t) < \sigma$, for some small $\sigma \in \mathbb{R}^+$.

III. SWARM PATH PLANNING

We introduce an obstacle-weighted rapidly-exploring random tree (RRT) planner to explore the environment, and discover collision-free routes to the goal location \mathbf{q}^g .

Sampling-based motion planning algorithms have shown great success in exploring collision-free paths for many scenarios, and probabilistic roadmaps (PRMs) [19] and rapidly-exploring random trees (RRTs) [20] are two popular planners. These planners generate random configurations in free space, connect them to create a graph of feasible paths, and link start and goal locations. In this paper, we focus on multi-shot 2D path planning with RRT and its extensions. Many attempts have shown success in improving the performance of RRTs near obstacles, such as narrow passage and tight region problems using a retraction strategy [21] [22]. Typically, retraction-based planners bias a configuration towards a more desirable region, for example, sampling new configurations in narrow paths more densely, and growing the tree near medial axes of G_{free} . As a result, less iterations are required to generate a feasible path connecting start and goal locations.

Applying an RRT-based path to a microrobot swarm using a global input can lead to problems: moving one particular agent may cause all the others to drift away from their initial locations. These locations may not be near any existing configurations on the tree (\mathcal{T}). Hence our RRT planner should have the following feature: for any robot r_i in G_{free} , we have

$$\|\mathbf{p}^{r_i} - \mathbf{q}_v\|_2 \leq \varepsilon, \quad (4)$$

for some $\varepsilon > 0$, where \mathbf{q}_v is the nearest vertex in \mathcal{T} . We grow a tree in an unbiased manner, such that sampling configurations are distributed uniformly. We also require sufficient configurations to guarantee Eqn. 4.



Fig. 3. Microrobot aggregation along different trajectories. The blue polygons are obstacles, and the white region is free space. One swarm (green circles) follows a black solid trajectory near the medial axes, and they keep moving together. Other swarms, represented by red triangles, follow the shortest path to the goal, which traps some microrobots at the corner and slows the aggregation process

Another issue with microrobot swarms is the environment interference. For example, in Fig. 3 a gray dashed line

Variables, and functions used in Alg. 1 and 2

V — the set of vertices (configurations) of \mathcal{T} .

V_{obs} — the set of sampling nodes on the boundary of obstacles.

V^* — the set of configurations near medial axes of G_{free} .

$\pi(\mathbf{q}_v)$ — the predecessor of the configuration \mathbf{q}_v in \mathcal{T} .

$Adj(\mathbf{q}_v)$ — the set of adjacent vertices.

(trajectory 1) is the shortest path to the goal location, and a black solid line (trajectory 2) represents a near-medial-axis path. Although trajectory 1 is shorter than 2, such a path significantly slows the aggregation process near obstacles. We propose an approach in Alg. 2, which is similar to the Gaussian sampling method, rerouting existing paths towards medial axes of free space. Compared to retraction-based planners, this approach replans a new route with existing configurations in \mathcal{T} instead of generating biased sampling of tree nodes.

The basic RRT planner builds a connected tree rooted at the goal location, and samples tree nodes randomly in free space of G to explore the graph. This process (Alg. 1) proceeds as follows: to grow the tree, we generate a random point \mathbf{q}_{rand} in G_{free} , and perform the nearest neighbor query (lines 3-4). Next, \mathbf{q}_{near} is extended towards \mathbf{q}_{rand} with unit length, and ends with \mathbf{q}_{new} . If the edge $(\mathbf{q}_{near}, \mathbf{q}_{new})$ is collision-free, which corresponds to a control input steering robots from \mathbf{q}_{near} to \mathbf{q}_{new} , we add the new node to the tree and update the distance metric (lines 6-9). Since all sampling points are connected to the tree, an agent can reach the goal location from any tree nodes simply by following their predecessors iteratively.

Algorithm 1 *RRT*. Input: configuration space G , goal location \mathbf{q}^g , total number of configurations $NumNode$. Output: an RRT \mathcal{T}

```

1:  $V = \{\mathbf{q}^g\}, V_{obs} = \emptyset, \mathcal{T} = \{V, V_{obs}\}$ 
2: while  $|V| \leq NumNode$  do
3:    $\mathbf{q}_{rand} \leftarrow$  a randomly generated point in  $G$ 
4:    $\mathbf{q}_{near} \leftarrow$  the nearest neighbor of  $\mathbf{q}_{rand}$  in  $V$ 
5:    $\mathbf{q}_{new} \leftarrow$  extend  $\mathbf{q}_{near}$  towards  $\mathbf{q}_{rand}$  for unit length
6:   if  $(\mathbf{q}_{near}, \mathbf{q}_{new}) \cap G_{obs} = \emptyset$  then
7:      $V = V \cup \{\mathbf{q}_{new}\}$ 
8:      $\pi(\mathbf{q}_{new}) = \mathbf{q}_{near}$ 
9:      $d\langle \mathbf{q}_{new}, \mathbf{q}^g \rangle = d\langle \mathbf{q}_{new}, \mathbf{q}_{near} \rangle + d\langle \mathbf{q}_{near}, \mathbf{q}^g \rangle$ 
10:   else
11:      $\mathbf{q}_{obs} \leftarrow (\mathbf{q}_{near}, \mathbf{q}_{new}) \cap \partial G_{obs}$ 
12:      $V_{obs} = V_{obs} \cup \{\mathbf{q}_{obs}\}$ 
13:   return  $\mathcal{T}$ 

```

Note that in Alg. 1, lines 10-12 are different from the original RRT [20]: if there is a collision along the path, we retract \mathbf{q}_{new} to the boundary of the obstacle \mathbf{q}_{obs} . \mathbf{q}_{obs} is not considered as a valid configuration in \mathcal{T} , instead, we add it to V_{obs} to grow an obstacle-weighted RRT. These obstacle nodes assist in steering steering paths away from obstacles.

We illustrate this process in Alg. 2, and compare it with

the original RRT in Fig. 4. The weight of a node \mathbf{q}_v in \mathcal{T} is calculated as follows,

$$w(\mathbf{q}_v) = e^{-a\|\mathbf{q}_v - \mathbf{q}_{obs}^{near}\|_2 + b}, \quad (5)$$

where $a, b \in \mathbb{R}^+$. Therefore, weight decreases with distance from nearby obstacles. Hence, if we identify a gradient descent path to the goal with minimum-weight nodes, the new path tends to proceed near medial axes of free space. A near-medial-axis set is constructed as:

$$V^* = \{\mathbf{q}_v \in V | w(\mathbf{q}_v) < \zeta\}, \quad (6)$$

for some $\zeta \in \mathbb{R}^+$. The trajectory generation (lines 7-16) is shown in Fig. 4(b). We trim the tree to remove edges not connecting to vertices in V^* , and perform adjacent neighbors query to regrow the tree towards near-medial-axis regions. Section IV shows that obstacle-weighted RRT decreases aggregation time.

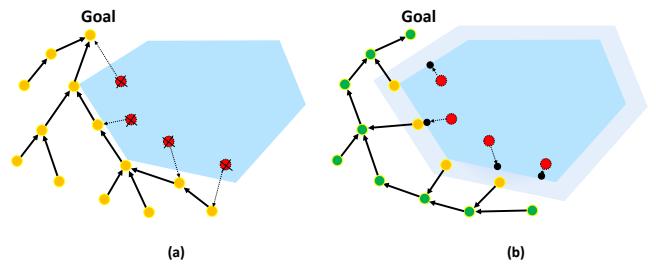


Fig. 4. (a) RRT (original): yellow dots are configurations of \mathcal{T} , and red dots are abandoned extensions within the blue obstacle. Trajectory generation relies on the shortest paths to the goal. (b) Obstacle-weighted RRT: green dots are near-medial-axis configurations in V^* , yellow dots in the shadow are elements in V affected by obstacles, and red dots inside the obstacle are retracted to the boundary (black dots) and added into V_{obs} . New paths tend to avoid near-obstacle regions, and approach near-medial-axis space.

Algorithm 2 *Obstacle-weighted RRT*. Input: the RRT \mathcal{T} of Alg. 1. Output: an obstacle-weighted RRT: \mathcal{T}_{obs}

```

1:  $V^* = \{\mathbf{q}^g\}, \mathcal{T}_{obs} = \{V, V_{obs}, V^*\}$ 
2: for all  $\mathbf{q}_v \in V$  do
3:    $\mathbf{q}_{obs}^{near} \leftarrow$  the nearest neighbor of  $\mathbf{q}_v$  in  $V_{obs}$ 
4:    $w(\mathbf{q}_v) = e^{-a\|\mathbf{q}_v - \mathbf{q}_{obs}^{near}\|_2 + b}$ 
5:   if  $w(\mathbf{q}_v) < \zeta$  then
6:      $V^* = V^* \cup \{\mathbf{q}_v\}$ 
7:   for all  $\mathbf{q}_v^* \in V^*$  do
8:      $d\langle \mathbf{q}_v^*, \mathbf{q}^g \rangle = \infty$ 
9:     for all  $\mathbf{q}_u^* \in Adj(\mathbf{q}_v^*) \cap V^*$  do
10:      if  $d\langle \mathbf{q}_v^*, \mathbf{q}^g \rangle > d\langle \mathbf{q}_v^*, \mathbf{q}_u^* \rangle + w(\mathbf{q}_u^*) + d\langle \mathbf{q}_u^*, \mathbf{q}^g \rangle$  then
11:         $\pi(\mathbf{q}_v^*) = \mathbf{q}_u^*$ 
12:         $d\langle \mathbf{q}_v^*, \mathbf{q}^g \rangle = d\langle \mathbf{q}_v^*, \mathbf{q}_u^* \rangle + w(\mathbf{q}_u^*) + d\langle \mathbf{q}_u^*, \mathbf{q}^g \rangle$ 
13:   for all  $\mathbf{q}_v \in V$  do
14:     if  $\pi(\mathbf{q}_v) \notin V^*$  then
15:        $\mathbf{q}_{near}^* \leftarrow$  the nearest neighbor of  $\mathbf{q}_v$  in  $V^*$ 
16:        $\pi(\mathbf{q}_v) = \mathbf{q}_{near}^*$ 
17:   return  $\mathcal{T}_{obs}$ 

```

IV. SWARM AGGREGATION

This section presents a divide-and-conquer aggregation method with heuristic strategies to improve performance. The motivation behind microrobot swarm aggregation is efficient control strategies for drug delivery in vascular networks. However, a global input with a highly under-actuated swarm system makes it difficult constructing an optimal controller. Pioneering research has proposed different strategies for the aggregation/gathering problem, but most are element-wise algorithms, that is, performing the task in terms of individuals. For example, [17] combines two agents each time and tests the efficiency of four heuristics. Our goal is to propose a swarm-level strategy to carry out swarm aggregation, and reduce time complexity compared to element-wise methods.

A. Heuristic Aggregation

A benchmark heuristic for aggregation is to move one microrobot to the goal, and then move the next agent. Repeat this till all robots gather near the goal location. In this paper, the benchmark heuristic moves farthest microrobot to the goal.

We present heuristic aggregation in Alg. 3. Alg. 3 is correct given the following assumptions: (i) the graph G is connected and bounded, and (ii) the goal location is inside a relatively closed region (Def. 1) at a dead end (Def. 2). The second assumption is inspired by the concept of discrete system transitions in [6], where gates are constructed to guide transitions from one region to another. In practice, the closed region indicates that once microrobots reach this area, it is hard for them to escape, given global inputs driving robots to \mathbf{q}^g . This is reasonable and essential, because an aggregated swarm may disperse in an open region given a global control signal. In fact, chemotherapy molecules are designed to release from carriers once they reach the region of tumor cells [1].

Algorithm 3 *Heuristic Aggregation.* Input: $\mathcal{T}_{obs} = \{V, V_{obs}\}$, initial positions $\{\mathbf{p}_{t_0}^{r_i}\}$ of all robots r_i . Output: \mathbf{u}_t

- 1: **while** $F(n, t) > \sigma$ **do**
- 2: $r_i \leftarrow$ the farthest robot
- 3: $\mathbf{q}_v^{r_i} \leftarrow$ the nearest vertex $\in V$ to r_i
- 4: $\mathbf{u}_t \leftarrow$ move r_i towards \mathbf{q}^g via $\mathbf{q}_v^{r_i}$

Definition 1. (closed region.) *Considering a global input \mathbf{u}_t that drives all robots to the goal \mathbf{q}^g , a closed region is a positive invariant set $\mathcal{M} \subset G_{free}$, $\mathbf{q}^g \in \mathcal{M}$. Given \mathbf{u}_t and a robot r_i , if $\mathbf{p}_{t_0}^{r_i} \in \mathcal{M}$ at t_0 , then $\mathbf{p}_t^{r_i} \in \mathcal{M}$ for all $t > t_0$. \mathcal{M} is bounded, so $\forall \mathbf{p}^{r_i} \in \mathcal{M}, \exists c > 0$, such that*

$$d\langle \mathbf{p}^{r_i}, \mathbf{q}^g \rangle < c\sigma, \quad (7)$$

Definition 2. (dead end.) *A dead end is a set $\mathcal{D} \subset \mathcal{M}$, $\mathbf{q}^g \in \mathcal{D}$. \mathcal{D} has the following properties:*

- 1) $\forall \mathbf{p}^{r_i} \in \mathcal{D}$, $d\langle \mathbf{p}^{r_i}, \mathbf{q}^g \rangle < \sigma$;
- 2) given that all robots $\{r_i\} \in G_{free}$ aggregate inside \mathcal{M} and a global input \mathbf{u}_t moves robots towards \mathcal{D} , such that if $\mathbf{p}^{r_i} \in \mathcal{D}$ at t_0 , then $\mathbf{p}^{r_i} \in \mathcal{D}$ for all $t > t_0$.

Variables and functions used in Alg. 4

Clustering(V_J , ‘distance’) — partition elements of V_J into junction clusters with the Euclidean distance metric, and returns the centroid of each junction.

RegionSeg(V^ , $\{\mathbf{q}_j^d\}$)* — partition elements of V^* into clusters by junctions, and returns the sub-region ID: $\psi(\cdot)$. First, assign a unique sub-region ID to the centroid of each junction, $\psi(\mathbf{q}_j^d) = R_j$. Next, $\psi(Adj(\mathbf{q}_j^d)) = R_j$. Then, assign all descendants of $Adj(\mathbf{q}_j^d)$ the same sub-region ID.

Clustering(S, \mathbf{q}_j^d ‘orientation’) — partition elements $s_i \in S$ into clusters by the orientation of a directed edge (s_i, \mathbf{q}_j^d) , and returns the mean orientation of each cluster, where \mathbf{q}_j^d is the centroid of the j -th junction cluster in V_J .

BranchSeg($S, \{\mathbf{q}_{j,k}^o\}$) — assign a branch ID $\phi(\cdot)$ to each element of S by orientation, where $\mathbf{q}_{j,k}^o$ is the orientation of branch $B_{j,k}$.

B. Divide-and-Conquer Aggregation

This method recursively aggregates microrobots into a smaller region that contains the goal. This transformation depends on how we define “a smaller region”. A proper definition of “region” reduces aggregation time. Interestingly, if we drive each microrobot all the way to the goal location, the algorithm is transformed into the heuristic aggregation.

The divide-and-conquer technique has two stages. We begin by splitting the aggregation problem into subproblems in smaller regions. Then we recursively perform discrete region transitions of microrobot swarms.

The first stage “divide” performs map segmentation of vascular systems like Fig. 7. In these maps, vessels are connected by junctions, and most of them are T-junctions.

Definition 3. (Region R_i .) *We define a partition of a map G_{free} as non-overlapping regions $\{R_i\}_{i=1,2,\dots,N_R}$, such that*

$$\{R_i \in G_{free} \mid \bigcup_{i=1}^{N_R} R_i = G_{free}, R_i \cap R_j = \emptyset, \forall i \neq j\}.$$

Definition 4. (Vessel and junction.) *Let $Q = \{\mathbf{q}_v^* \cup adj(\mathbf{q}_v^*)\}$, where $\mathbf{q}_v^* \in V^*$ and $adj(\mathbf{q}_v^*)$ are adjacent neighbors of \mathbf{q}_v^* within a range. Fitting an ellipse to elements $\in Q$ in the X-Y plane, if the eccentricity $< \delta_e$, for some $\delta_e > 0$, then \mathbf{q}_v^* is a junction node; otherwise, \mathbf{q}_v^* is a vessel node.*

Algorithm 4 *Map Segmentation.* Input: The obstacle-weighted RRT $\mathcal{T}_{obs} = \{V, V_{obs}, V^*\}$. Output: Map segmentation: $M = \{V_J, \psi(V^*), \phi(V^*)\}$

- 1: $V_J = \{\mathbf{q}^g\}$
- 2: **for** all $\mathbf{q}_v^* \subset V^*$ **do**
- 3: **if** \mathbf{q}_v^* is a junction node **then**
- 4: $V_J = V_J \cup \{\mathbf{q}_v^*\}$
- 5: $\{\mathbf{q}_j^d\}_{j=1,2,\dots,N_d} \leftarrow Clustering(V_J, \text{‘distance’})$
- 6: $\{\psi(\mathbf{q}_v^*) = R_j\}_{j=1,2,\dots,N_R} \leftarrow RegionSeg(V^*, \{\mathbf{q}_j^d\})$
- 7: **for** ($j = 1; j++ ; j \leq N_R$) **do**
- 8: $S \leftarrow \{\mathbf{q}_v^* \in V^* | \psi(\mathbf{q}_v^*) = R_j\}$
- 9: $\{\mathbf{q}_{j,k}^o\}_{k=1,2,\dots} \leftarrow Clustering(S, \mathbf{q}_j^d \text{ ‘orientation’})$
- 10: $\{\phi(\mathbf{q}_v^*) = B_{j,k}\}_{k=1,2,\dots,N_{B_j}} \leftarrow BranchSeg(S, \{\mathbf{q}_{j,k}^o\})$

As shown in Fig. 5(b), we can separate junction nodes (green dots) from other nodes in straight vessels (orange dots) by their spatial distributions as specified in Def. 4. Generally speaking, a junction is the zone where different branches are joined, so the fitting ellipse of a junction node and its adjacent neighbors is similar to a circle. These junction nodes can determine boundaries between regions. In our implementation, we take the maximal width of local channels as the range of adjacent neighbors. With these definitions, we perform map segmentation using results from obstacle-weighted RRT. This process is presented in Alg. 4, and illustrated in Fig. 5:

- 1) use near-medial-axis configurations $\mathbf{q}_v^* \in V^*$ to identify junction nodes (lines 1-4);
- 2) partition the set of junction nodes (V_J) using Euclidean distance (line 5), and yield N_R junction clusters;
- 3) split free space into N_R regions corresponding to the N_R junction clusters (line 6);
- 4) partition each region into branches $\{B_{j,k}\}_{k=1,2,\dots,N_{B_j}}$ by their orientations (lines 7-10), where $B_{j,k}$ is the k -th of N_{B_j} branches in R_j , $N_{B_j} \leq 3$.

In step 3, region segmentation proceeds in three phases. First we select a cluster of V_J and set the junction nodes as seeds. Then we grow the region with these seeds by adding their descendants generation by generation in \mathcal{T}_{obs} . The region expansion stops at the next junction. In step 4, branch segmentation is a result of clustering. Considering all \mathbf{q}_v^* in the j -th region, we connect \mathbf{q}_j^d to \mathbf{q}_v^* , where \mathbf{q}_j^d is the centroid of the j -th junction cluster. Branches can be obtained by clustering all these directed edges $(\mathbf{q}_j^d, \mathbf{q}_v^*)$.

Definition 5. (aggregation robustness.) *Considering a multi-robot system with a global input, we define robustness as the ability of an aggregation procedure to keep aggregated swarms in their regions despite influences of a global controller.*

A proper map segmentation is essential for our aggregation, because it employs the environment to increase robustness of the procedure. For example, if microrobots keep escaping from the goal region, or the procedure gets trapped in a local minimum (e.g. moving a swarm of microrobots back and forth without progress), we say it is not robust.

With map segmentation, we are able to process the second stage “conquer”. This process is presented in Alg. 5, and the assumptions of Alg. 3 hold. A global planner moves a swarm of microrobots from region R_j to $R_{j,next}$, where region R_j and $R_{j,next}$ share an edge, and region $R_{j,next}$ is closer to the goal, $d(\mathbf{q}_j^d, \mathbf{q}^g) < d(\mathbf{q}_{j,next}^d, \mathbf{q}^g)$. A local planner assigns priorities to microrobots at different branches of region R_j , and leads them to the closer region $R_{j,next}$.

This divide-and-conquer algorithm consists of three *while* loops: (i) identify the farthest region R_{jm} that microrobots exist, and $d(\mathbf{q}_j^d, \mathbf{q}^g)$ is the cost from the j -th junction centroid to goal; (ii) pick a branch with the highest priority, computed

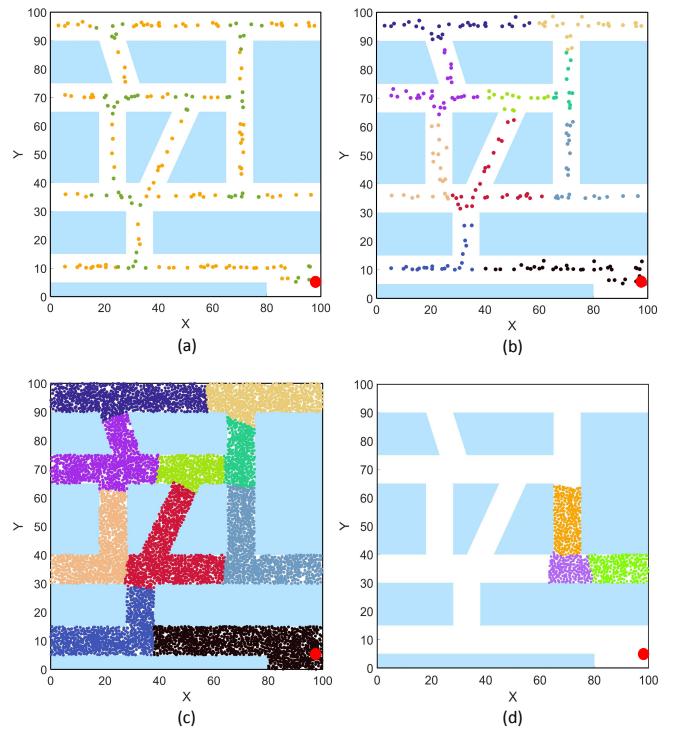


Fig. 5. Map segmentation illustration for Alg. 4. The goal (red dot) is located at (99,5). (a) represents step 1 and 2, (b) and (c) shows step 3, and (d) illustrates step 4 in a region.

by

$$v(B_{jm,k}) = \sum_{i=1}^{\#\text{robots in } B_{jm,k}} \gamma^{s_d(i)}, \quad (8)$$

with $0 < \gamma < 1$, $s_d(i) = d(\mathbf{p}^{r_{km,i}}, \mathbf{q}_{jm}^d)$, $r_{km,i}$ the i -th robot inside branch $B_{jm,km}$; (iii) generate a global control input to drive robot $r_{km,im}$ to the nearest configuration (\mathbf{q}_v^r) in \mathcal{T}_{obs} , and then move it towards some configuration $(\mathbf{q}_{R_{jm,next}})$ in $R_{jm,next}$. After moving all robots in R_{jm} to $R_{j,next}$, we complete a discrete region transition for a swarm.

Algorithm 5 Divide-and-conquer Aggregation. Input: $\mathcal{T}_{obs} = \{V, V_{obs}, V^*\}$, $M = \{V_J, \psi(V^*), \phi(V^*)\}$, initial positions $\{\mathbf{p}_{t_0}^i\}$ of all robots $\{r_i\}$. Output: \mathbf{u}_t

```

1: while  $F(n,t) > \sigma$  do
2:    $R_{jm} \leftarrow \text{argmax } d(\mathbf{q}_j^d, \mathbf{q}^g), j = 1, 2, \dots$ 
3:   while there exists any robot in  $R_{jm}$  do
4:      $B_{jm,km} \leftarrow \text{argmax } v(B_{jm,k}), k = 1, 2, \dots$ 
       $B_{jm,k} \in R_{jm}$ 
5:     while there exist any robot in  $B_{jm,km}$  do
6:        $r_{km,im} \leftarrow \text{argmin } d(\mathbf{p}_t^{r_{km,i}}, \mathbf{q}_{jm}^d), \forall r_{km,i} \text{ in } B_{jm,km}$ 
7:        $\mathbf{q}_v^r \leftarrow \text{the nearest vertex } \in V \text{ to } r_{km,im}$ 
8:        $\mathbf{u}_t \leftarrow \text{move } r_{km,im} \text{ towards } \mathbf{q}_{R_{jm,next}} \text{ via } \mathbf{q}_v^r$ 

```

To analyze time complexity of the divide-and-conquer recurrence, we need the following assumptions: (i) the map is connected and bounded, (ii) “closed region” and “dead end” definitions, (iii) microrobots are uniformly distributed

in the map initially, and (iv) aggregation time is proportional to map area and population. Let G denote a map with $\text{Area}(G) = m$, $\text{Population}(G) = n$, $\text{Density}(G) = \rho = n/m$, and $G'(0)$ be a smaller map $G'(0) \subset G$ with $\text{Area}(G'(0)) = \xi m$, $\text{Population}(G') = \xi \rho m$, where ξ is a discount factor. If $T(mn)$ is the running time for map G , we start from region R_{j_m} , $G'(0) = G - R_{j_m}$, and level 0 of recurrence is in the form:

$$T(mn) = T(\xi mn) + f((1-\xi)\rho m \cdot (1-\xi)m), \quad (9)$$

where $f((1-\xi)^2 \rho m^2)$ denotes the aggregation time in the farthest region R_{j_m} , with $(1-\xi)\rho m$ the population and $(1-\xi)m$ the area.

We can easily derive the recursive form for level i using two models. In the first model, we assume that the aggregation map shrinks with a constant discount factor ξ each time, then $\text{Area}(G'(i)) = \xi \cdot \text{Area}(G'(i-1)) = \xi^{i+1}m$,

$$T(\xi^i mn) = T(\xi^{i+1} mn) + f(\xi^i(1-\xi)\rho m \cdot \xi^i(1-\xi)m), \quad (10)$$

where the density is regarded as a constant in $f(\cdot)$. The density decreases with aggregation since microrobots overlap. So this assumption does not reduce the difficulty of the subproblem. The base case is $T(n) = f(n)$, and we simplify $f(x)$ with a linear model $f(x) = kx$, then

$$\begin{aligned} T(mn) &= \sum_{i=0}^{\log_{1/\xi} m} f(\xi^{2i}(1-\xi)^2 \rho m^2) \\ &= k\rho m^2 (1-\xi)^2 \sum_{i=0}^{\log_{1/\xi} m} \xi^{2i}. \end{aligned} \quad (11)$$

Assuming $\log_{1/\xi} m$ is an integer, and $m \gg \xi$, Eqn. 11 can be simplified to

$$T(mn) = k\rho m^2 \left(\frac{2}{1+\xi} - 1 \right), \quad (12)$$

In the second model, we reduce the map by a constant area $(1-\xi)m$ each time, then level i has the form

$$T((1-i(1-\xi))mn) = T((1-(i+1)(1-\xi))mn) + f((1-\xi)^2 \rho m^2). \quad (13)$$

Hence, we have

$$T(mn) = k\rho m^2 \sum_{i=0}^{1/(1-\xi)} (1-\xi)^2 \quad (14)$$

Assuming $\frac{1}{1-\xi}$ is an integer, we can write Eqn. 14 as

$$T(mn) = k\rho m^2 (1-\xi)(2-\xi) \quad (15)$$

The performance of different discount factors ξ is shown in Fig. 6. As ξ increases, the scaled running time decreases fast, despite some fluctuations in the second model. This means that the more we reduce the map size each time, the less efficient divide-and-conquer aggregation becomes. As $\xi \rightarrow 0$, we actually have the heuristic aggregation instead. This is equivalent to decreasing the map size from m to 1 ($\xi = \frac{1}{m}$) with one recurrence. For both models (Eqn. 12 and

15), as $\xi \rightarrow \frac{1}{m}$, $T(mn) \rightarrow O(m^2)$; as $\xi \rightarrow 1 - \frac{k^*}{m}$, for some $k^* \in \mathbb{R}^+$, $k^* \ll m$, $T(mn) \rightarrow T(m)$. Hence, the divide-and-conquer strategy makes it possible to reduce time complexity from $T(m^2)$ to $T(m)$. Note that k^* is dependent on junctions in a map: the finer we can split the map, the smaller k^* is.

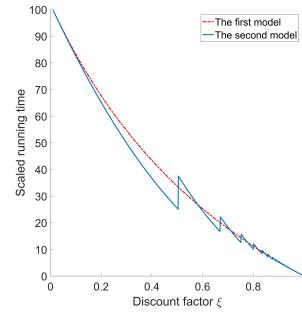


Fig. 6. Running time estimation. (a) refers to the first recurrence model in Eqn. 11. (b) refers to the second recurrence mode in Eqn. 14

V. SIMULATION

We report the simulation results to evaluate our path planning approaches, RRT and obstacle-weighted RRT (OWRRT), compare the divide-and-conquer aggregation (DCA) with the heuristic aggregation (HRA), and study the impact of map and swarm population. We perform three sets of simulations, and in each set, we present two algorithms for aggregation and two methods for path planning.

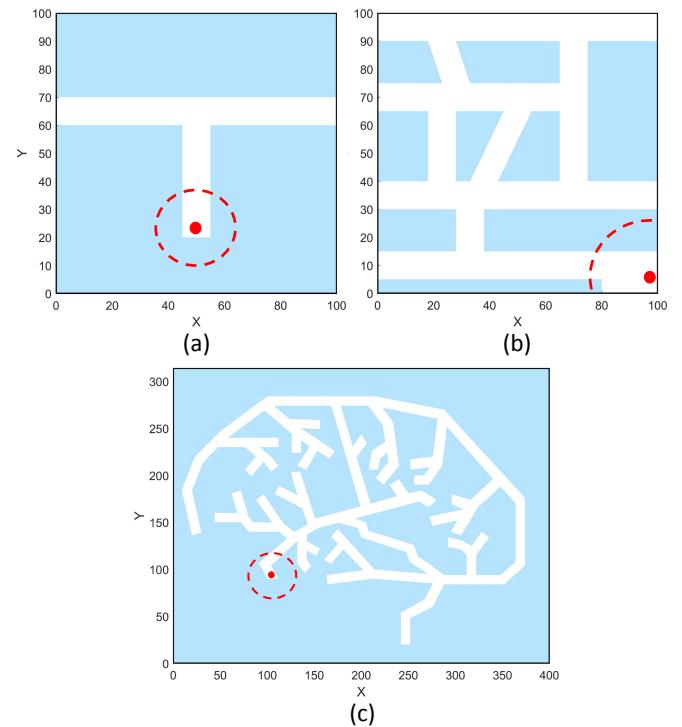


Fig. 7. Blue polygons represent obstacles, and white channels are free space. We place a red dot at each goal location. Three maps are from small to large, and simple to complex : (a) T-junction map, (b) a vascular network, and (c) a larger vascular network.

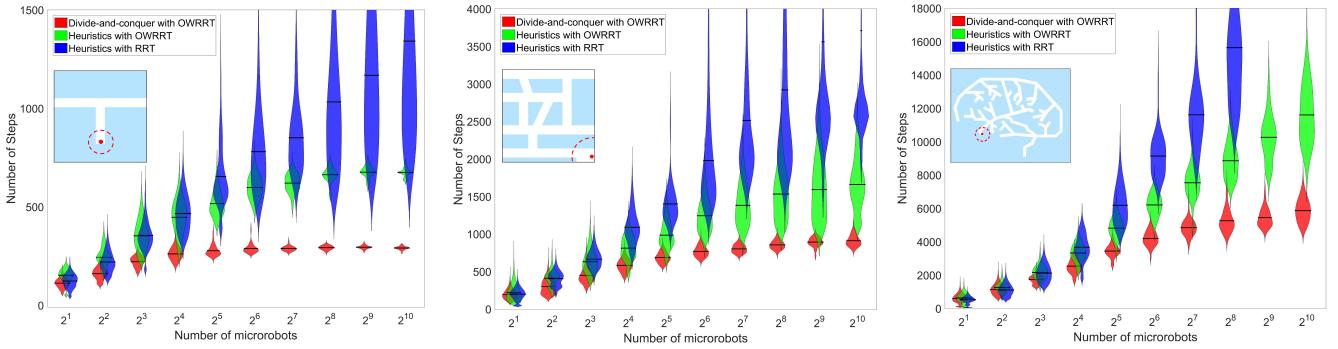


Fig. 8. Particle aggregation in maps (a,b,c). The violin plot shows the probability density of the simulation data and the black line indicates the mean value. We performed 30 simulations for each combination of methods and swarm populations.

Path planning and aggregation are carried out in three simulated maps (Fig. 7), including a T-junction map, and two vascular networks. The obstacles are marked as blue polygons, and free space is white. To initialize, we place n microrobots randomly in free space, where $n \in \{2^1, 2^2, 2^3, \dots, 2^{10}\}$, and each microrobot is represented by a point with no area. Given a global input \mathbf{u}_t at time t , all microrobots will move towards the assigned direction for one discrete step of unit length (Eqn. 1 and 2). The goal is to gather microrobots to the goal location. In practice, we define the task is accomplished if the average position of the swarm is near the goal, or $F(n,t) < \sigma$ (dashed circle in Fig. 7). We count the total number of steps to approximate the running time for swarm aggregation in a map. In each map, ten different microrobot populations are used. For each population, we perform 30 simulations for three combinations of aggregation algorithms and path planning methods respectively: DCA+OWRRT, HRA+OWRRT, and HRA+RRT. The results of simulations are compared using violin plots in Fig. 8.

We evaluate the performance of these algorithms by their average running time (number of steps) and data distributions. DCA+OWRRT outperforms any other combinations in all these simulation when the swarm population is large enough ($n \geq 2^3$). The average aggregation time of DCA+OWRRT does not grow as fast as others, and it tends to approach an upper bound asymptotically in each vascular network. Also, this combination shows reliability and efficiency with different environments and swarm populations. For each independent trial, the aggregation time is within a small range around the mean value with high probability. Neither HRA+OWRRT nor HRA+RRT can compete with DCA+OWRRT in average aggregation time when the swarm size is greater than 2^3 . The average running time of HRA+OWRRT and HRA+RRT increases with $\log n$ in most cases with large standard deviation, and the worst case can be extremely inefficient.

VI. EXPERIMENT

A. Electromagnetic Platform

We use a customized electromagnetic platform which consists of three orthogonal pairs of coils with separation

distance equivalent to the outer diameter of a coil. The coils (18 AWG, Custom Coils, Inc) are powered by six SyRen10-25 motor drivers with Tekpower HY3020E DC power supply. An Arduino Mega 2560 provides PWM signal to control motor drives, and images are acquired using an IEEE 1394 camera with the region of interest approximate 20 mm². Each image has 379 × 366 pixels, and each pixel represents an area of 40 μm² of the workspace. We process microrobots detection and tracking in MATLAB using blob analysis and Kalman filter, and send control input \mathbf{u}_t (i.e., the orientation of the magnetic field) to the Arduino Mega via USB serial port communication. In experiments, the electromagnetic platform (with iron cores) can provide over 300 Gauss uniform magnetic fields along any direction in the horizontal plane.

B. Experiment Setup

The vascular network we choose to validate path planning and aggregation algorithms is shown in Fig. 1 (b), which is similar to Fig. 7 (b). This map is made of two layers of acrylic cut using a Universal Laser Cutter, one layer as the base, and the other as the polygonal obstacles, with the layers joined by double-sided adhesive ??. The base layer is 5 mm thick, and the obstacle layer is 2 mm thick. The frame is a 20×20 mm² square, and the channel width is 2 mm. We fill channels with vegetable oil, and place it in the center of our electromagnetic platform. The microrobots are ferromagnetic particles. These microrobots vary in sizes and shapes, with initial population over 200. Microrobots align with magnetic fields when the magnitude is larger than 100 Gauss. Because the density of ferromagnetic particles is over seven times larger than water, gradient fields provided by our electromagnetic platform are not able to drag microrobots around due to friction. Hence we create rotational fields to make the microrobots roll along the base. Rolling a ?? field at ?? Hz causes the microrobots to roll at ?? mm/s.

C. Validation of Aggregation Algorithms

The results of the divide-and-conquer aggregation are compared with those of the benchmark heuristic aggregation as shown in Fig. ?.

VII. CONCLUSIONS

This paper compared two path-planning methods and two control strategies applied to the problem of aggregating microrobot swarms in vascular networks using a global input. Although RRT creates an obstacle-free path from initial locations to a goal for a single robot, this path is not ideal for swarms. We propose an obstacle-weighted RRT that steers microrobots towards near-medial-axis regions to reduce environment interference. A divide-and-conquer strategy is employed to perform swarm-level aggregation via discrete region transitions. Compared to the benchmark strategy, the divide-and-conquer aggregation reduces the task time complexity. Future work should prove the convergence of our aggregation algorithms and explore a wider variety of maps.

APPENDIX

ACKNOWLEDGMENT

REFERENCES

- [1] G. Tiwari, R. Tiwari, B. Sriwastava, L. Bhati, S. Pandey, P. Pandey, S. K. Bannerjee, *et al.*, “Drug delivery systems: An updated review,” *International journal of pharmaceutical investigation*, vol. 2, no. 1, p. 2, 2012.
- [2] A. Pierson and M. Schwager, “Bio-inspired non-cooperative multi-robot herding,” in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1843–1849.
- [3] B. T. Fine and D. A. Shell, “Eliciting collective behaviors through automatically generated environments,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3303–3308.
- [4] S. Shahrokhi and A. T. Becker, “Stochastic swarm control with global inputs,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 421–427.
- [5] A. Becker, E. D. Demaine, S. P. Fekete, G. Habibi, and J. McLurkin, “Reconfiguring massive particle swarms with limited, global control,” in *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*. Springer, 2013, pp. 51–66.
- [6] L. Bobadilla, O. Sanchez, J. Czarnowski, K. Gossman, and S. M. LaValle, “Controlling wild bodies using discrete transition systems,” *Advanced Robots*, 2011.
- [7] I. S. Khalil, F. van den Brink, O. S. Sukas, and S. Misra, “Microassembly using a cluster of paramagnetic microparticles,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5527–5532.
- [8] P. S. S. Kim, A. Becker, Y. Ou, A. A. Julius, and M. J. Kim, “Swarm control of cell-based microrobots using a single global magnetic field,” in *Ubiquitous Robots and Ambient Intelligence (URAI), 2013 10th International Conference on*. IEEE, 2013, pp. 21–26.
- [9] U. K. Cheang and M. J. Kim, “Self-assembly of robotic micro-and nanoswimmers using magnetic nanoparticles,” *Journal of Nanoparticle Research*, vol. 17, no. 3, p. 145, 2015.
- [10] S. Martel and M. Mohammadi, “Using a swarm of self-propelled natural microrobots in the form of flagellated bacteria to perform complex micro-assembly tasks,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 500–505.
- [11] D. De Lanauze, O. Felfoul, J.-P. Turcot, M. Mohammadi, and S. Martel, “Three-dimensional remote aggregation and steering of magnetotactic bacteria microrobots for drug delivery applications,” *The International Journal of Robotics Research*, vol. 33, no. 3, pp. 359–374, 2014.
- [12] U. Kei Cheang, K. Lee, A. A. Julius, and M. J. Kim, “Multiple-robot drug delivery strategy through coordinated teams of microswimmers,” *Applied physics letters*, vol. 105, no. 8, p. 083705, 2014.
- [13] Y. Ou, P. Kang, M. J. Kim, and A. A. Julius, “Algorithms for simultaneous motion control of multiple t. pyriformis cells: Model predictive control and particle swarm optimization,” in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3507–3512.
- [14] D. Wong, E. B. Steager, and V. Kumar, “Independent control of identical magnetic robots in a plane,” *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 554–561, 2016.
- [15] I. S. Khalil, M. P. Pichel, B. A. Reefman, O. S. Sukas, L. Abelmann, and S. Misra, “Control of magnetotactic bacterium in a micro-fabricated maze,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5508–5513.
- [16] S. Scheggi and S. Misra, “An experimental comparison of path planning techniques applied to micro-sized magnetic agents,” in *Manipulation, Automation and Robotics at Small Scales (MARSS), International Conference on*. IEEE, 2016, pp. 1–6.
- [17] 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 *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1231–1236.
- [18] 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 *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 520–527.
- [19] L. E. Kavraki and J.-C. Latombe, “Probabilistic roadmaps for robot path planning,” 1998.
- [20] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” 1998.
- [21] L. Zhang and D. Manocha, “An efficient retraction-based rrt planner,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 3743–3750.
- [22] X. Tang, J.-M. Lien, N. Amato, *et al.*, “An obstacle-based rapidly-exploring random tree,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. IEEE, 2006, pp. 895–900.