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

Li Huang and Aaron T. Becker

***Abstract*— Microrobots have great potential for microassembly and non-invasive surgery applications. 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 microparticles in oil using a 6-coil electromagnetic system with image feedback.**

## I. INTRODUCTION

Microrobots have great potential to be used in non-invasive surgery for drug delivery. Traditional drug delivery circulates the human body indiscriminatingly, which is why chemotherapy kills healthy and tumor cells alike. 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]–[5]. [2] proposed a control strategy that by introducing herders to drive a swarm of herding animals to a desired location with repelling potential fields. 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. [4] 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 [5], 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, and thereby to accomplish high-level tasks.

However, many microrobots have limited capabilities for sensing and actuating, so external sensors (e.g. MRI, cameras) and actuators (e.g. external electric or magnetic fields) must be employed. 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, [6]–[10]. 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 microrobots in free space [11]–[13]. [14] demonstrated control of a single microrobot in a micro-fabricated maze. Scheggi et al. implemented and compared six path planning algorithms using magnetic microrobots [15].

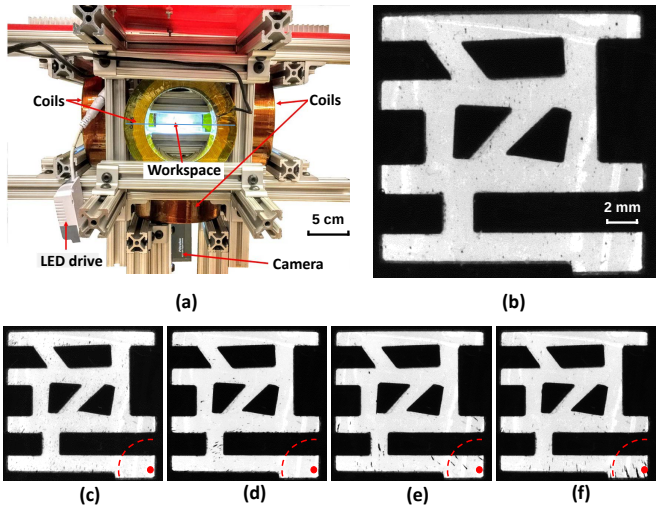


Fig. 1. (a) The six-coil electromagnetic system. (b) Microrobot swarms aggregation in a vascular network tested in experiments. (c)-(f) Video frames from a experiment. The goal location is marked with a red point. (c) $t = 0$ min, (d) $t = 4$ min, (e) $t = 19$ min, (f) $t = 36$ min

Our previous work explored microrobot swarm aggregation in a planar grid environment [16], where we considered microrobots 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 manuevered 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

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

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

Consider that a bounded 2D space $G \subset \mathbb{R}^2$ is consisted of free space ($G_{free}$) and obstacle space ($G_{obs}$). $G_{free}$ is connected by paths of width at least $w$. A swarm of simple microrobots are initialized at random locations in $G_{free}$. These simple microrobots have on-board computation or communication, and they are guided by a global field. Their bodies are less than $w/10$, and mutual interaction is ignored. An $n$-microrobot swarm has $2n$ degree-of-freedom in a plane, and applying a global field only adds two constraints ($x$ and $y$ components) to the system. To break the symmetry of swarm motion with a global field, the environment is employed to assist swarm tasks.

Our preliminary research explored path planning algorithms and aggregation strategies to collect these microrobots at a targeted location. We proposed an obstacle-weighted rapidly-exploring random tree (RRT) to plan near-medial-axis trajectories from any location to the goal, and benchmark heuristic and divide-and-conquer algorithms were developed for microrobot swarm aggregation in vascular networks. This paper presents an improved method for path planning with high efficiency and evaluate how well the environment can help with swarm aggregation.

The microrobots are modeled as dots without area. At time $t$, the position of the $i$-th microrobot is denoted as $(x_t^i, y_t^i)$ and a global control input $u_t = (u_x, u_y)$ moves the microrobot to

$$\left(x_{t+1}^i, y_{t+1}^i\right) = \left(x_t^i, y_t^i\right) + \left(u_x, u_y\right), \tag{1}$$

or the robot stops at the boundary of an obstacle.

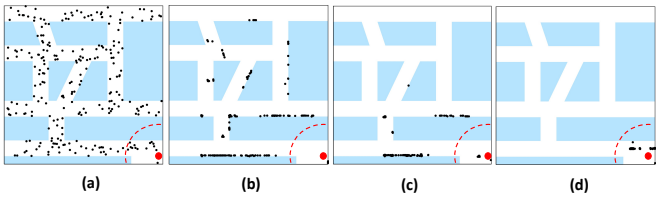


Fig. 2. Blue polygons represent obstacles, and white channels are free space. We place a red dot at the goal location. The dashed arc represents the region that corresponds to $F(n,t) < \sigma$. (a) Simulated aggregation process after 1 step, (b) 200 steps, (c) 500 steps, and (d) 800 steps.

## 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. Probabilistic roadmaps (PRMs) [17] and rapidly-exploring random trees (RRTs) [18] 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 [19] [20]. 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

$$\|\mathbf{p}^{r_i} - \mathbf{q}_v\|_2 \leq \varepsilon, \tag{2}$$

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. 2.

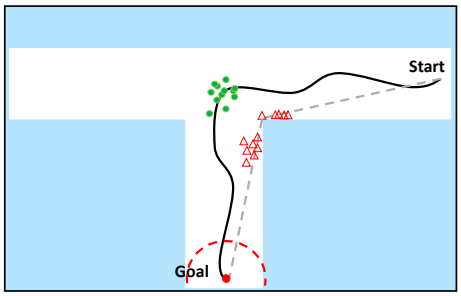


Fig. 3. Microrobot aggregation along different trajectories. One swarm (green circles) follows a black solid trajectory near the medial axes, and they keep moving together. Another swarm, represented by red triangles, follows the shortest path to the goal, which traps some microrobots at the corner and slows the aggregation process

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

| 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. |

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

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, a robot can reach the goal location from any tree nodes simply by following their predecessors iteratively.

Note that in Alg. 1, lines 10-12 are different from the original RRT [18]: 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 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,

$$w(\mathbf{q}_v) = e^{-a\|\mathbf{q}_v - \mathbf{q}_{obs}^{near}\|_2 + b}, \tag{3}$$

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 of configurations is constructed as:

$$V^* = \{\mathbf{q}_v \in V | w(\mathbf{q}_v) < \zeta\}, \tag{4}$$

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

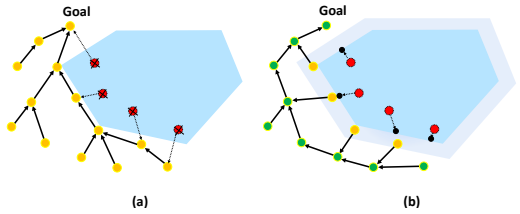


Fig. 4. (a) RRT (original): yellow dots are configurations of $\mathcal{T}$, and red dots are abandoned extensions within the blue obstacle. Trajectory generation relies on the shortest paths to the goal. (b) Obstacle-weighted RRT: green dots are near-medial-axis configurations $\in V^*$, yellow dots 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.

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.

## 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, [16] 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 the farthest microrobot to the goal.

We present the heuristic aggregation in Alg. 3. with the following assumptions: (i) the graph $G$ is connected and bounded, and (ii) the goal location is inside a closed region (Def. 1) at a dead end (Def. 2). The second assumption is inspired by the concept of discrete system transitions in [5], 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: $\mathcal{T}_{obs} = \{V, V_{obs}, V^*\}$, initial positions $\{\mathbf{p}_{t_0}^{r_i}\}$ of all robots $r_i$. Output: $\mathbf{u}_t$

```
while F(n,t) > σ do
    r_i ← the farthest robot
    q_v^{r_i} ← the nearest vertex ∈ V to r_i
    u_t ← move r_i towards q^g via 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*

$$d\langle \mathbf{p}^{r_i}, \mathbf{q}^g \rangle < c\sigma, \tag{5}$$

**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 $\mathcal{D}$, if $\mathbf{p}^{r_i} \in \mathcal{D}$ at $t_0$, then $\mathbf{p}^{r_i} \in \mathcal{D}$ for all $t > t_0$.*

### 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. 8. In these maps, vessels are connected by junctions, and most of them are T-junctions.

**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 $\{\mathbf{q}_j^d\}$.

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

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

**Definition 3.** (Region $R_i$.) *We define a partition of a map $G_{free}$ as non-overlapping regions $\{R_i\}_{i=1,2,\cdots,N_R}$, such that $\{R_i \in G_{free} | \bigcup_{i=1}^{N_R} R_i = G_{free}, R_i \cap R_j = \emptyset, \forall i \neq j\}$.*

**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.*

As shown in Fig. 5(a), 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 $\{B_{j,k}\}_{k=1,2,\cdots,N_{B_j}}$ 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 $(\mathbf{q}_j^d, \mathbf{q}_v^*)$.

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

$V_J = \{\mathbf{q}^g\}$
**for** all $\mathbf{q}_v^* \subset V^*$ **do**
  **if** $\mathbf{q}_v^*$ is a junction node **then**
    $V_J = V_J \cup \{\mathbf{q}_v^*\}$
$\{\mathbf{q}_j^d\}_{j=1,2,\cdots,N_d} \leftarrow Clustering(V_J, \text{'distance'})$
$\{\psi(\mathbf{q}_v^*) = R_j\}_{j=1,2,\cdots,N_R} \leftarrow RegionSeg(V^*, \{\mathbf{q}_j^d\})$
**for** $(j = 1; j = j+1; j \le N_R)$ **do**
  $S \leftarrow \{\mathbf{q}_v^* \in V^* | \psi(\mathbf{q}_v^*) = R_j\}$
  $\{\mathbf{q}_{j,k}^o\}_{k=1,2,\cdots,N_{B_j}} \leftarrow Clustering(S, \mathbf{q}_j^d$ 'orientation')
  $\{\phi(\mathbf{q}_v^*) = B_{j,k}\}_{k=1,2,\cdots,N_{B_j}} \leftarrow BranchSeg(S, \{\mathbf{q}_{j,k}^o\})$

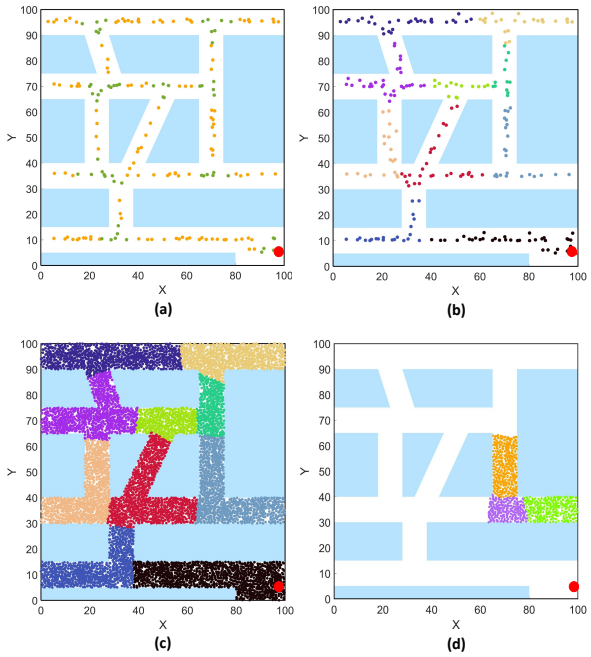


Fig. 5. Map segmentation illustration for Alg. 4. The goal (red dot) is located at (99,5). (a) represents step 1 and 2, where yellow points are nodes in $V^*$, and green points are junction nodes in $V_J$. (b) and (c) show step 3, where regions are marked as different colors. (d) illustrates step 4 in a region, where branches are marked as different colors.

**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.*

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. The map segmentation is essential for identifying relatively closed regions $\{R_i\}$ which we use to perform discrete region transitions on swarms.

With map segmentation, we are able to process the second stage "conquer". This process is presented in Alg. 5 and illustrated in Fig. 6. 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, $d\langle \mathbf{q}_{j.next}^d, \mathbf{q}^g \rangle < d\langle \mathbf{q}_j^d, \mathbf{q}^g \rangle$. A local planner assigns priorities to microrobots at different branches of region $R_j$, and leads them to the closer region $R_{j.next}$.

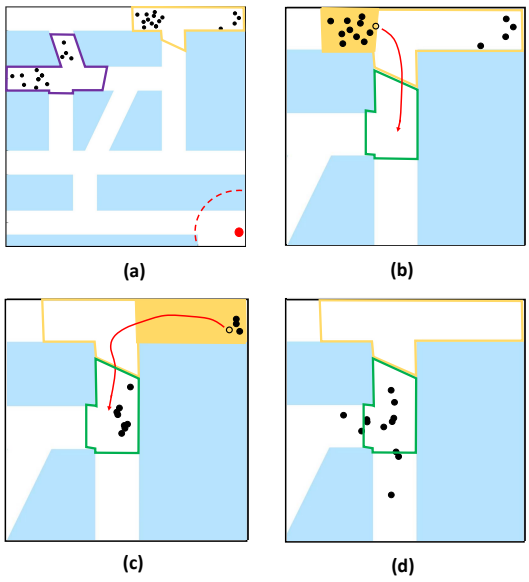


Fig. 6. Black circles are microrobots, and regions are marked by colored outline. (a) Robots exist in the purple region $R_i$ and the orange region $R_j$. $R_j$ is the farthest region from the goal (red dot). (b) and (c) The branch filled with orange has higher priority. $R_{j.next}$ is marked as green. A control input drives the robot $r_{k_m,i_m}$ (hollow circle) to $R_{j.next}$. (d) Complete a discrete region transition for a swarm from $R_j$ to $R_{j.next}$.

This divide-and-conquer algorithm consists of three *while* loops: (i) identify the farthest region $R_{j_m}$ that microrobots exist (Fig. 6(a)), and $d\langle \mathbf{q}_j^d, \mathbf{q}^g \rangle$ is the cost from the $j$-th junction centroid to goal; (ii) pick a branch with the highest priority, i.e., with more robots closer to the junction centroid $\mathbf{q}_{j_m}^d$ (Fig. 6(b)), computed by

$$\upsilon(B_{j_m,k}) = \sum_{i=1}^{\#\text{robots in } B_{j_m,k}} \gamma^{s_d(i)}, \tag{6}$$

with $0 < \gamma < 1, s_d(i) = d\langle \mathbf{p}^{r^{k_m,i}}, \mathbf{q}_{j_m}^d \rangle, r^{k_m,i}$ the $i$-th robot inside branch $B_{j_m,k_m}$; (iii) identify the closest robot $r_{k_m,i_m}$ to $\mathbf{q}_{j_m}^d$, and drive it to the nearest $\mathbf{q}_v^r \in V$, and then move it towards some $\mathbf{q}_{R_{jm.next}} \in V$ in $R_{jm.next}$ (Fig. 6(b) and (c)). After moving all robots in $R_{j_m}$ to $R_{j_m.next}$, we complete a discrete region transition for a swarm (Fig. 6(d)).

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

$$T(mn) = T(\xi mn) + f\big((1-\xi)\rho m \cdot (1-\xi)m\big), \tag{7}$$

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

**while** $F(n,t) > \sigma$ **do**
  $R_{j_m} \leftarrow \text{argmax}\, d\langle \mathbf{q}_j^d, \mathbf{q}^g \rangle, j = 1,2,\cdots,N_R$
  **while** there exists any robot in $R_{j_m}$ **do**
    $B_{j_m,k_m} \leftarrow \underset{B_{j_m,k} \in R_{j_m}}{\text{argmax}}\ \upsilon(B_{j_m,k}), k = 1,2,\cdots,N_{B_j}$
    **while** there exist any robot in $B_{j_m,k_m}$ **do**
      $r_{k_m,i_m} \leftarrow \text{argmin}\, d\langle \mathbf{p}_t^{r_{k_m,i}}, \mathbf{q}_{j_m}^d \rangle, \forall r_{k_m,i}$ in $B_{j_m,k_m}$
      $\mathbf{q}_v^r \leftarrow$ the nearest vertex $\in V$ to $r_{km,im}$
      $\mathbf{u}_t \leftarrow$ move $r_{km,im}$ towards $\mathbf{q}_{R_{jm.next}}$ via $\mathbf{q}_v^r$

where $f((1-\xi)^2\rho m^2)$ denotes the aggregation time in $R_{j_m}$, with $(1-\xi)\rho m$ the population and $(1-\xi)m$ the area. After we move out all robots in $R_{j_m}$, the aggregation map shrinks from $G$ to $G'(0)$.

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

$$T(\xi^i mn) = T(\xi^{i+1}mn) + f\big(\xi^i(1-\xi)\rho m \cdot \xi^i(1-\xi)m\big), \quad (8)$$

where the density is assumed to be a constant in $f(\cdot)$. In fact, the density decreases with aggregation since microrobots overlap. So this assumption does not reduce the difficulty of the subproblem. The base case is $T(n) = f(n)$ with $m = 1$, and we simplify $f(x)$ with a linear model $f(x) = kx$, then

$$\begin{aligned} T(mn) &= \sum_{i=0}^{\log_{1/\xi} m} f\big(\xi^{2i}(1-\xi)^2\rho m^2\big) \\ &= k\rho m^2(1-\xi)^2 \sum_{i=0}^{\log_{1/\xi} m} \xi^{2i}. \end{aligned} \quad (9)$$

Assuming $\log_{1/\xi} m$ is an integer, and $m \gg \xi$, Eqn. 9 can be simplified to

$$T(mn) = k\rho m^2\big(\frac{2}{1+\xi} - 1\big), \quad (10)$$

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

$$\begin{aligned} T\big((1-i(1-\xi))mn\big) = T\big((1-(i+1)(1-\xi))mn\big) \\ + f\big((1-\xi)^2\rho m^2\big). \end{aligned} \quad (11)$$

Hence, we have

$$T(mn) = k\rho m^2 \sum_{i=0}^{1/(1-\xi)} (1-\xi)^2 \quad (12)$$

Assuming $\frac{1}{1-\xi}$ is an integer, we can write Eqn. 12 as

$$T(mn) = k\rho m^2(1-\xi)(2-\xi) \quad (13)$$

The performance of different discount factors $\xi$ is shown in Fig. 7. As $\xi$ 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 $\xi \to 0$, 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. 10 and 13), as $\xi \to \frac{1}{m}$, $T(mn) \to O(m^2)$; as $\xi \to 1 - \frac{k^*}{m}$, for some $k^* \in \mathbb{R}^+$, $k^* \ll m$, $T(mn) \to T(m)$. 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.

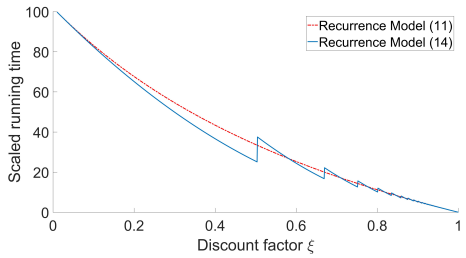


Fig. 7. Running time estimation of the first recurrence model in Eqn. 9 and the second recurrence model in Eqn. 12

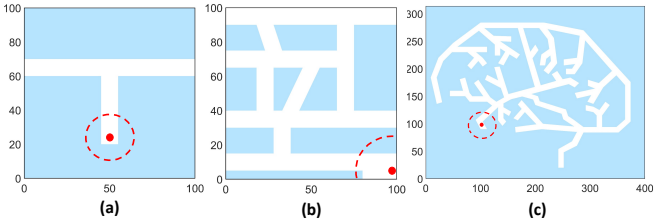


Fig. 8. Blue polygons represent obstacles, and white channels are free space. We place a red dot at each goal location. These maps increase in size and complexity: (a) T-junction map, (b) a vascular network, and (c) a larger vascular network.

## 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.

Path planning and aggregation are carried out in three simulated maps (Fig. 8), 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 $n \in \{2^1, 2^2, 2^3, \cdots, 2^{10}\}$, 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. **??** and **??**). 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. 8). 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.

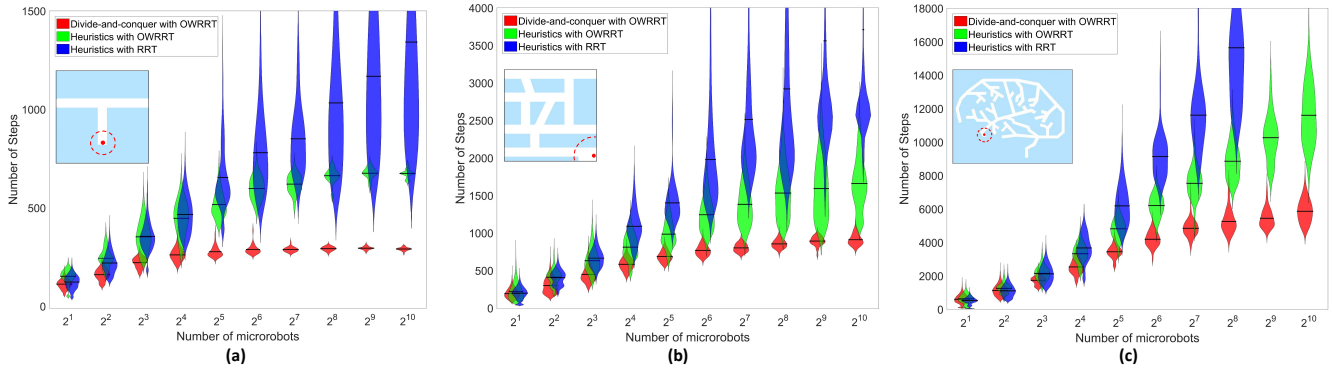


Fig. 9. Particle aggregation in maps (a,b,c). The violin plot shows the probability density of the simulation data and the black line indicates the mean value. We performed 30 simulations for each combination of methods and swarm populations.

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. 9.

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 has small standard deviation. 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 custom-made 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 (50 fps) with the region of interest approximate 20 mm$^2$. Each image has $379 \times 366$ pixels, and each pixel represents an area of 40 $\mu$m$^2$ of the workspace. We process microrobot detection and tracking in MATLAB using blob analysis and Kalman filters, 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 magnetic fields along any direction in the 20 mm$^3$ workspace center.

### B. Experiment Setup

The vascular network we used to validate path planning and aggregation algorithms is shown in Fig. 1 (b) and Fig. 8 (b). This maze 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. The frame is a $20\times20$ mm$^2$ square, and the channel width is 2 mm. In each experiment, the maze is filled with a mixture of microrobots and vegetable oil (0.45 mL) at the same concentration, and placed in the workspace center. The microrobots are composed of ferromagnetic particles (30 microns $Fe_3O_4$, Alpha Chemicals). These microparticles aggregate into microrobots that vary in sizes and shapes, with initial population over 300. 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 uniform field in the vertical plane at 5 Hz causes microrobots to move at an average velocity of 80 $\mu$m/s, and maximum velocity is over 350 $\mu$m/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. 10. The running time is approximated by number of processed image frames for each experiment ($\approx$ 45 fps). The swarm population is estimated by averaging the number of pixels classified as robots in last 13500 frames ($\approx$ 5 min). With similar swarm populations, the average running time for the benchmark is 93,063 frames ($\approx$ 34.5 min), and 60,628 frames ($\approx$ 22.5 min) for the divide-and-conquer algorithm, which reduces by 34.9%. Hence the divide-and-conquer aggregation outperforms the benchmark.

## 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

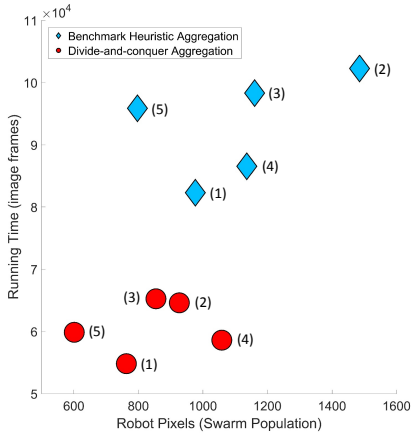


Fig. 10. Blue diamonds are benchmark data, and red circles are results for divide-and-conquer aggregation, with an experiment number next to each marker.

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.

## ACKNOWLEDGMENT

This work was supported by the National Science Foundation ([IIS-1553063] and [IIS-1619278] to A.T.B and and [IIS-1734732] and [CMMI-1737682] to M.J.K.).

## REFERENCES

[1] G. Tiwari, R. Tiwari, B. Sriwastawa, 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] 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.

[5] L. Bobadilla, O. Sanchez, J. Czarnowski, K. Gossman, and S. M. LaValle, "Controlling wild bodies using discrete transition systems," *Advanced Robots*, 2011.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] D. H. Kim, S. Brigandi, A. A. Julius, and M. J. Kim, "Real-time feedback control using artificial magnetotaxis with rapidly-exploring random tree (rrt) for tetrahymena pyriformis as a microbiorobot," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3183–3188.

[14] I. S. Khalil, M. P. Pichel, B. A. Reefman, O. S. Sukas, L. Abelmann, and S. Misra, "Control of magnetotactic bacterium in a microfabricated maze," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5508–5513.

[15] 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.

[16] 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.

[17] L. E. Kavraki and J.-C. Latombe, "Probabilistic roadmaps for robot path planning," 1998.

[18] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[19] 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.

[20] 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.