IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

A Hierarchical Framework for Real-Time Path Planning of Microswarm in Dynamic Environments

Yamei Li, *Student Member, IEEE*, Ruijian Ge, Aoji Zhu, Jiachi Zhao, Danjing Shi, Yinghan Sun, Yangmin Li, *Senior Member, IEEE*, and Lidong Yang, *Member, IEEE*

***Abstract*—Autonomous navigation of magnetic microswarms in dynamic and unstructured environments is essential for biomedical applications, such as targeted therapy and minimally invasive interventions. However, existing path planning methods struggle to simultaneously achieve real-time adaptability and path smoothness in dynamic obstacle environments. To address this, we propose a hierarchical Dynamic Rapidly-exploring Random Tree Star (D-RRT*) path planning framework that integrates dynamic step size adjustment, local target selection, and local planning that considers microswarms' turning capabilities and energy optimization. Comparative simulations and experiments validate the effectiveness of the proposed planning framework, and results show that it can significantly improve the planning efficiency, path smoothness, and collision avoidance in complex dynamic scenarios.**

***Index Terms*—Automation at micro-/nanoscale, micro-/nanorobots, microswarm, dynamic path planning, navigation.**

I. Introduction

Micro- and nanorobots have demonstrated great potential in a wide range of applications, such as targeted delivery [1], minimally invasive therapy [2], micromanipulation [3], and others [4]–[6]. Among various actuation methods, magnetic actuation is particularly advantageous for these applications, as magnetic fields can safely penetrate deep tissues even at high intensities [7]. However, the small size of individual magnetic microrobots makes in vivo tracking difficult due to the limited resolution of medical imaging instruments [8]. To address this, reconfigurable microswarms inspired by natural collective behaviors have been proposed, forming dynamic structures such as vortex-like [9], ribbon-like [10], and elliptical [11]. To enable practical biomedical applications, achieving autonomous navigation of microswarms in complex and dynamic environments—such as blood vessels and body fluids [12]—is essential. This requires robust and adaptive path planning to overcome challenges posed by intricate anatomical structures, confined spaces, moving instruments, and unpredictable flows, thereby ensuring smooth and collision-free microswarm navigation in these scenarios.

Existing path planning methods for dynamic environments can generally be classified into learning-based and traditional approaches. Learning-based planning methods leverage offline training on dynamic environment datasets for real-time inference, achieving high performance in obstacle avoidance. For instance, Jiang et al. [13] and Wang et al. [14] employed deep Q-learning networks (DQN) and proximal policy optimization (PPO), respectively, for real-time obstacle avoidance in microrobots of varying sizes and helical microrobots. However, both approaches faced challenges with path smoothness and overall path quality. Traditional planning methods do not rely on extensive offline training or large datasets, making them more practical for immediate deployment when the environment model and constraints are well-defined. Liu et al. [15] proposed a hierarchical radar-based method achieving real-time obstacle avoidance at nearly 10 Hz in simple environments, but frequent direction changes in multi-obstacle scenarios reduced path smoothness and efficiency. Similarly, classical algorithms such as RRT* and A* suffer from limited path smoothness; although some of their variants have been developed to improve path quality specifically for microrobot applications, these studies are conducted primarily in static obstacle environments and still face potential issues with poor real-time performance in dense and dynamic scenarios [16]–[19]. Although uniform tree expansion techniques can improve the sampling efficiency of RRT* [20], they frequently generate paths with sharp turns, which can destabilize microswarm motion. These inherent limitations—such as suboptimal real-time performance and insufficient path smoothness in dynamic, obstacle-laden environments—stem from the lack of consideration for dynamic system models during sampling and the absence of comprehensive cost function optimization. Furthermore, the use of fixed extension steps in those methods reduces flexibility in complex environments, often resulting in infeasible or suboptimal solutions. Therefore, addressing these challenges requires the development of a planning approach that prioritizes real-time adaptability, smooth path generation, and robust handling of dynamic obstacles.

Motivated by these challenges, this work proposes a hierarchical D-RRT* path planning framework that integrates dynamic step size adjustment, autonomous local target selection, and local path planning that considers microswarm turning capabilities and energy optimization. By optimizing

Manuscript received Aug 16 2025; Revised Oct 25 2025; Accepted Dec 25 2025. This paper was recommended for publication by Editor Xinyu Liu upon evaluation of the Associate Editor and Reviewers comments. This work is financially supported by the Research Institute for Advanced Manufacturing of Hong Kong Polytechnic University (grant no. 1-CDK3); the Guangdong Basic and Applied Basic Research Foundation (grant no. 2025A1515010116 and 2023A1515110709), the Hong Kong Research Grant Council (grant nos. STG1/E-401/23-N, 15206223, and 25200424). Yamei Li gratefully acknowledges financial support from the Research Committee of The Hong Kong Polytechnic University (student account code RM9K). *(Corresponding author: Lidong Yang)*

Yamei Li, Aoji Zhu, Jiachi Zhao, Danjing Shi, Yinghan Sun, Yangmin Li and Lidong Yang are with the Research Institute for Advanced Manufacturing, Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University (PolyU), Kowloon, Hong Kong, China.

Ruijian Ge is with the department of Mechanical Engineering, Vanderbilt University, US.

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

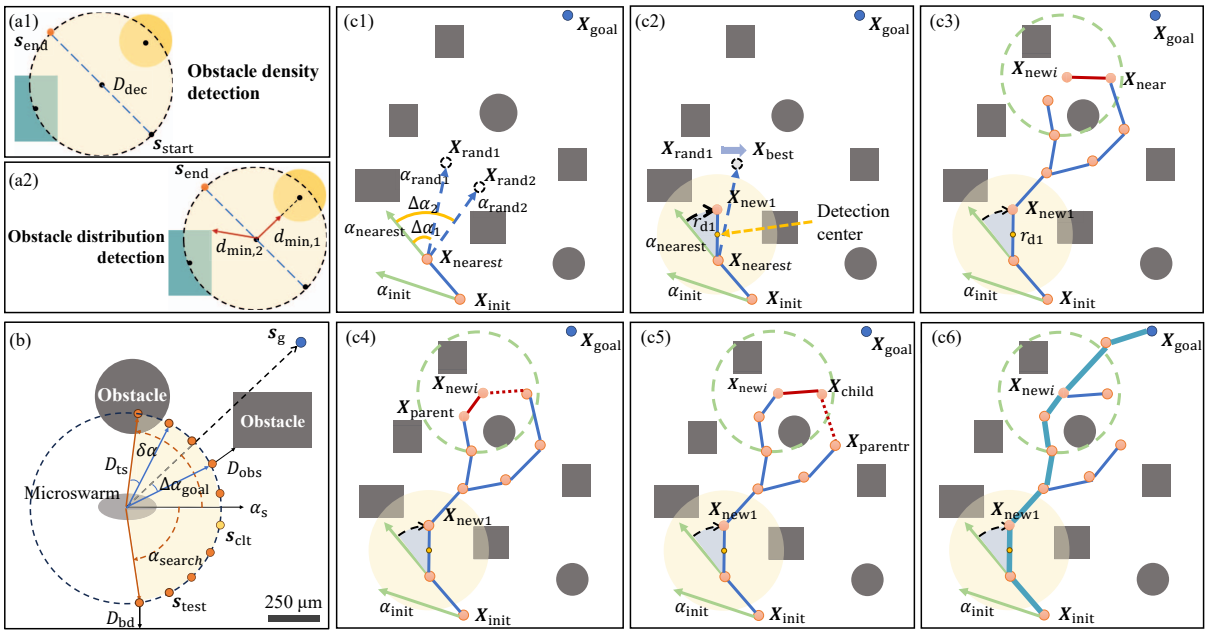

Fig. 1. The schematic illustration of the hierarchical D-RRT* algorithm. (a) Schematic illustration of detection range for dynamic step size adjustment. (a1) Obstacle density detection. (a2) Obstacle distribution detection. (b) Schematic illustration of local target selection. Orange points represent candidate test points, the yellow shaded region denotes the candidate direction scan range. The light gray area represents the swarm, the blue point indicates the global goal $\boldsymbol{s}_g$, and the selected target $\boldsymbol{s}_{\text{clt}}$ is marked in yellow. (c) The schematic illustration of the local path planning algorithm. (c1) Cost-based sampling strategy. Random points $\boldsymbol{X}_{\text{rand1}}$ and $\boldsymbol{X}_{\text{rand2}}$ are generated. The orientations of $\boldsymbol{X}_{\text{nearest}}$ and $\boldsymbol{X}_{\text{init}}$ are represented by orange arrows, while the orientations of $\boldsymbol{X}_{\text{rand1}}$ and $\boldsymbol{X}_{\text{rand2}}$ are shown by blue dashed lines. The turning angles $\Delta\alpha_1$ and $\Delta\alpha_2$ represent the angle differences between α_{nearest} and $\alpha_{\text{rand1}}, \alpha_{\text{rand2}}$, respectively. (c2) Generate $\boldsymbol{X}_{\text{new}}$. The best candidate point $\boldsymbol{X}_{\text{best}}$ is selected, and the steer procedure generates $\boldsymbol{X}_{\text{new1}}$. As $\Delta\alpha_1 < \Delta\alpha_2$, $\boldsymbol{X}_{\text{rand1}}$ is chosen as $\boldsymbol{X}_{\text{best}}$, and $\boldsymbol{X}_{\text{new1}}$ is extended with a dynamically adjusted step size determined by the yellow shaded detection circle. (c3) Node expansion. $\boldsymbol{X}_{\text{new}i}$ represents the i-th new node, and $\boldsymbol{X}_{\text{near}}$ indicates neighboring node within the range shown by the green dashed circle. (c4) Parent re-selection. $\boldsymbol{X}_{\text{parent}}$ is the new parent node of $\boldsymbol{X}_{\text{new}i}$. (c5) Rewiring process. $\boldsymbol{X}_{\text{child}}$ is the new child node, and $\boldsymbol{X}_{\text{parentr}}$ denotes its previous parent node. (c6) Final connection. The complete path is constructed and represented by the thick blue line.

the cost function and adaptively adjusting extension steps, the framework enhances real-time adaptability and facilitates the generation of smooth and efficient paths. The effectiveness of the framework is validated through comprehensive simulations and comparative experiments. The key contributions of this work are:

- A dynamic step size adjustment strategy is introduced, enabling efficient and flexible feasible path search according to obstacle density and distribution in dynamic environments.
- Autonomous local target selection and local planning that accounts for microswarm turning constraints jointly ensure path smoothness and real-time planning performance in dynamic obstacle environments, effectively preventing path discontinuities.

II. Hierarchical Path Planning Framework

Currently, path planning algorithms for microswarms are typically based on a fixed extension scale, or constant step size [18], [21], which defines the distance advanced toward the target at each planning step. However, this approach has notable limitations: It lacks adaptability to varying environments, as small step sizes reduce efficiency in open areas while large step sizes increase collision risk in cluttered regions. Additionally, it responds poorly to dynamic changes, making timely and precise path updates challenging in the presence of moving targets or sudden obstacles. Therefore, our planning framework incorporates dynamic step size adjustment, autonomous local target selection, and local path planning, as shown in Fig. 2. In particular, the dynamic step size adjustment strategy enables real-time adaptation to different obstacle densities and spatial distributions, thereby ensuring safe, efficient, and smooth navigation.

A. Dynamic Step Size Adjustment Strategy

Dynamic step size adjustment serves as the foundation of the framework, enabling adaptive modification of the local target selection distance D_{base} and extension step size r_{base} according to detection circles and environmental factors ρ. Each detection circle is centered at the midpoint between two characteristic points in the environment (e.g., the robot's current position $\boldsymbol{s}_{\text{start}}$ and a candidate target $\boldsymbol{s}_{\text{end}}$), as illustrated in Fig. 1(a). The detection circle diameter D_{dec} is set as a fixed parameter based on environmental and task requirements, balancing adaptability to different scenarios with computational efficiency.

1) Obstacle Density Detection: The local obstacle density factor w_1 quantifies the proportion of obstacles within the detection circle:

$$w_1 = \frac{S_{\text{obs}}}{S_{\text{det}}} \tag{1}$$

where S_{obs} denotes the total intersection area between obstacles and the detection circle, S_{det} denotes the area of the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

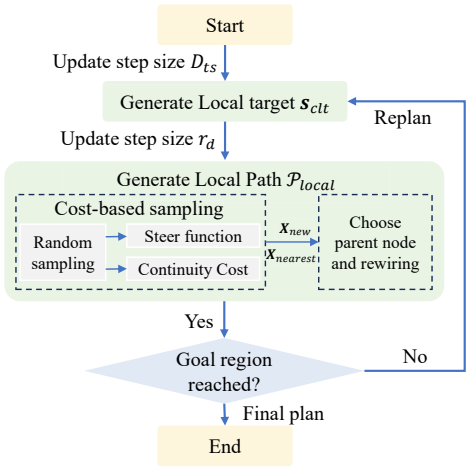

Fig. 2. Overview of the hierarchical D-RRT* planning process.

detection circle. Higher w_1 indicates a higher obstacle density, prompting the use of smaller step sizes.

2) Obstacle Distribution Detection: The obstacle distribution factor w_2 measures the concentration of obstacles around the detection center inside the detection circle:

$$w_2 = 1 - \frac{2 \cdot \sum_{i=1}^{N} d_{\min,i}}{N \cdot D_{\text{dec}}} \tag{2}$$

where $d_{\min,i}$ represents the minimum distance between the detection center and the i-th obstacle, and N is total number of obstacles. Lower w_2 indicates a more dispersed obstacle distribution, while a higher w_2 indicates a higher concentration of obstacles near the detection center.

3) Dynamic Adjust Factor: The dynamic adjust factor ρ combines w_1 and w_2:

$$\rho = 0.5 + \frac{1}{2}(1 - (w_1 + w_2)) \tag{3}$$

The adjusted extension step size and local target selection distance are $r_d = \rho \cdot r_{\text{base}}$ and $D_{ts} = \rho \cdot D_{\text{base}}$, respectively. Lower ρ enables fine-grained exploration in dense or clustered obstacle regions, while higher ρ allows larger strides in open spaces for faster search.

B. Autonomous Local Target Selection

In dynamic environments, direct global planning to the goal $\boldsymbol{s}_{\text{g}}$ often faces challenges such as limited real-time performance and discontinuous paths during replanning. To mitigate these issues, the local target selection strategy (***Algorithm*. 1**) iteratively guides the robot toward intermediate targets $\boldsymbol{s}_{\text{clt}}$, enhancing both path smoothness and adaptability.

This strategy involves sampling candidate directions and dynamically adjusting the local target selection distance D_{ts} based on ρ, where the base distance D_{base} is empirically set to four times the microswarm's width, as shown in Fig. 1(b). Specifically, candidate directions α_{c} are sampled around the robot's current orientation α_{s} at regular angular intervals $\delta\alpha$ within a range of α_{search}. For each direction, a test point $\boldsymbol{s}_{\text{test}}$ is generated at a distance D_{ts}, which is dynamically adjusted according to the detection circle. The center of the detection

Algorithm 1 Autonomous Local Target Selection

Input: $\boldsymbol{s}_{\text{s}}$, $\boldsymbol{s}_{\text{g}}$, $\boldsymbol{s}_{\text{gpre}}$, D_{ts}, r_s, α_{search}, $\delta\alpha$
Output: Local target $\boldsymbol{s}_{\text{clt}}$

if $\boldsymbol{s}_{\text{gpre}}$ exists **then**
  $\alpha_{\text{gpre}} \leftarrow \arctan 2(y_{\text{gpre}} - y_s, x_{\text{gpre}} - x_s)$
else
  $\alpha_{\text{gpre}} \leftarrow \alpha_s$
end if
for $\alpha_c \leftarrow \alpha_s - \alpha_{\text{search}}$ to $\alpha_s + \alpha_{\text{search}}$ step $\delta\alpha$ **do**
  $\mathcal{P} \leftarrow \{(x_s + D_{\text{ts}} \cdot \cos(\alpha_c), y_s + D_{\text{ts}} \cdot \sin(\alpha_c))\}$
  $Cost_l \leftarrow \text{SelectionCostFunction}(\mathcal{P}, \alpha_c, \alpha_{\text{goal}}, \alpha_{\text{gpre}}, r_s)$
  Append $(\alpha, Cost_l)$ to $\mathcal{C}_{\text{total}}$
end for
Sort $\mathcal{C}_{\text{total}}$ by $Cost_l$ in ascending order ▷ Prioritize lower-cost directions
for $(\alpha_c, Cost_l) \in \mathcal{C}_{total}$ **do**
  $\boldsymbol{s}_{\text{clt}} \leftarrow (x_s + D_{\text{ts}} \cdot \cos(\alpha_c), y_s + D_{\text{ts}} \cdot \sin(\alpha_c))$
  $\alpha_{\text{opt}} \leftarrow \alpha_c$
  if CollisionFree($\boldsymbol{s}_{\text{s}}$, $\boldsymbol{s}_{\text{clt}}$) **then**:
    return $\boldsymbol{s}_{\text{clt}}$ ▷ Return the first valid target
  end if
end for

circle lies at the midpoint between robot's current position $\boldsymbol{s}_{\text{s}}$ and $\boldsymbol{s}_{\text{clt}}$, and its diameter D_{dec} is generally predefined as a fixed parameter. Notably, for the initial determination of D_{ts}, since the local candidate target $\boldsymbol{s}_{\text{clt}}$ has not been selected yet, the detection circle diameter is set to the distance between $\boldsymbol{s}_{\text{s}}$ and $\boldsymbol{s}_{\text{g}}$.

Candidate directions are evaluated using a multi-criteria cost function, **SelectionCostFunction** ($Cost_{\text{l}}$), to ensure safe, efficient, and smooth motion. $Cost_{\text{l}}$ integrates four factors: obstacle cost ($Cost_{\text{obs}}$), penalizing proximity to obstacles; boundary cost ($Cost_{\text{bd}}$), penalizing approaches to environment boundaries; goal deviation cost ($Cost_{\text{dir}}$), penalizing misalignment with the global goal; and direction consistency cost ($Cost_{\text{dc}}$), penalizing abrupt changes from the previous direction. The $Cost_{\text{l}}$ for a candidate direction is defined as:

$$Cost_{\text{l}} = \alpha_1 \cdot Cost_{\text{obs}} + \alpha_2 \cdot Cost_{\text{bd}} + \alpha_3 \cdot Cost_{\text{dir}} + \tag{4a}$$

$$\alpha_4 \cdot Cost_{\text{dc}} \tag{4b}$$

$$Cost_{\text{obs}} = \left(1 - \frac{D_{\text{obs}}}{r_s^2}\right)^2 \tag{4c}$$

$$Cost_{\text{bd}} = \left(1 - \frac{D_{\text{bd}}}{r_s^2}\right)^2 \tag{4d}$$

$$Cost_{\text{dir}} = \left(\frac{\Delta\alpha_{\text{g}}}{2\pi}\right)^2, \quad \Delta\alpha_{\text{g}} = |\alpha_c - \alpha_{\text{g}}| \tag{4e}$$

$$Cost_{dc} = \begin{cases} \left(\frac{\Delta\alpha_{\text{gpre}}}{2\pi}\right)^2, & \text{if } \boldsymbol{s}_{\text{gpre}} \text{ exists} \\ 0, & \text{otherwise} \end{cases}, \tag{4f}$$

$$\Delta\alpha_{\text{gpre}} = |\alpha_c - \alpha_{\text{gpre}}| \tag{4g}$$

Here, $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are weighting factors for each cost term. Increasing any of these weights makes the corresponding cost term play a more significant role in path selection, thus biasing the path towards greater avoidance of obstacles

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Algorithm 2 Local Path Planning

Input: $\boldsymbol{X}_{\text{start}}$, $\boldsymbol{X}_{\text{goal}}$, r_{search}, Goal tolerance ϵ_g
Output: Generated path $\mathcal{P}_{local}$ from $\boldsymbol{X}_{\text{start}}$ to $\boldsymbol{X}_{\text{goal}}$

$T \leftarrow \text{InitTree}(\boldsymbol{X}_{\text{start}})$
for $k = 1$ to K **do**
  $\boldsymbol{X}_{\text{new}}, \boldsymbol{X}_{\text{nearest}} \leftarrow$ ***Algorithm* 3** $(\boldsymbol{T}, \boldsymbol{X}_{\text{start}}, \boldsymbol{X}_{\text{goal}})$
  if $\text{CollisionFree}(\boldsymbol{X}_{\text{nearest}}, \boldsymbol{X}_{\text{new}})$ **then**
    $\mathcal{N} \leftarrow \text{NearArea}(\boldsymbol{X}_{\text{new}}, \boldsymbol{T}, r_{\text{search}})$
    $\boldsymbol{X}_{\text{parent}} \leftarrow \text{ChooseParent}(\mathcal{N}, \boldsymbol{X}_{\text{new}}, \boldsymbol{T})$
    $\boldsymbol{T} \leftarrow \text{InsertNode}(\boldsymbol{X}_{\text{parent}}, \boldsymbol{X}_{\text{new}}, \boldsymbol{T})$
    $\boldsymbol{T} \leftarrow \text{Rewire}(\mathcal{N}, \boldsymbol{X}_{\text{new}}, \boldsymbol{T})$
    if $\|\boldsymbol{X}_{\text{new}} - \boldsymbol{X}_{\text{goal}}\| < \epsilon_g$ **then**
      $\mathcal{P}_{local} \leftarrow \text{BacktrackPath}(\boldsymbol{X}_{\text{new}}, T)$
      return $\mathcal{P}_{local}$
    end if
  end if
end for

or boundaries, increased alignment with the goal, or higher directional consistency with previous direction, respectively, and the specific values are selected based on application requirements and empirical tuning to balance safety, efficiency, and smoothness. D_{obs} and D_{bd} denote the minimum distances from the test points $\mathcal{P}$ to the nearest obstacle and to the environment boundaries, respectively. $\Delta\alpha_{\text{g}}$ is the angular deviation between the candidate direction α_c and the global goal direction α_{g}, where $\alpha_{\text{g}} = \text{atan2}(y_g - y_s, x_g - x_s)$). $\Delta\alpha_{\text{gpre}}$ is the angular deviation between α_c and the direction of the previous target $\boldsymbol{s}_{\text{gpre}}$. The safety margin r_s accounts for the finite size of the microswarm in collision detection, inflating obstacle boundaries to reduce collision risk due to dynamic behavior.

Among the evaluated candidate directions, the first collision-free candidate with the lowest total cost is selected as the next $\boldsymbol{s}_{\text{clt}}$, as illustrated in Fig. 1(b). The collision-free condition is evaluated using the **CollisionFree** function, which checks whether the straight-line path (vector) from $\boldsymbol{s}_{\text{s}}$ to $\boldsymbol{s}_{\text{clt}}$ intersects with any obstacles in the environment.

C. Local Path Planning

Once the local target $\boldsymbol{s}_{\text{clt}}$ is determined, the hierarchical D-RRT* planning framework performs local path planning between current $\boldsymbol{s}_{\text{s}}$ and $\boldsymbol{s}_{\text{clt}}$. Without loss of generality, the dynamic path planning process is described for a single tree $\boldsymbol{T}$ as an example. Unlike existing RRT*-based methods, this algorithm explicitly incorporates the swarm's orientation into the initial node state $\boldsymbol{X}_{\text{init}}(x_{\text{init}}, y_{\text{init}}, \alpha_{\text{init}})$ to ensure the initial path aligns with the swarm's dynamic constraints, as shown in Fig. 1(c). However, subsequent states (e.g., $\boldsymbol{X}_{\text{new}}$) are not constrained by a fixed orientation, allowing for greater flexibility in path planning.

1) Cost-based sampling strategy: As depicted in Fig. 1(c1), ***Algorithm* 2** begins at $\boldsymbol{X}_{\text{init}}(x_{\text{init}}, y_{\text{init}}, \alpha_{\text{init}})$, where α_{init} corresponds to the initial orientation of the swarm. Instead of relying solely on random sampling as in standard RRT*, D-RRT* employs a cost-based sampling strategy to promote smoother paths.

Algorithm 3 Cost-based sampling for D-RRT*

Input: T, $\boldsymbol{X}_{\text{start}}$, $\boldsymbol{X}_{\text{goal}}$
Output: $\boldsymbol{X}_{\text{new}}$, $\boldsymbol{X}_{\text{nearest}}$

Initialize: $\mathcal{M} \leftarrow \emptyset$, $\boldsymbol{X}_{\text{best}} \leftarrow \emptyset$, $C_{\text{best}} \leftarrow \infty$, Max attempts $N_{\text{attempts}} \leftarrow 15$
while $\text{len}(\mathcal{M}) < M$ **and** attempts $< N_{\text{attempts}}$ **do**
  $\boldsymbol{X}_{\text{rand}} \leftarrow \text{Sample}()$
  $\boldsymbol{X}_{\text{nearest}} \leftarrow \text{FindNearestNode}(T, \boldsymbol{X}_{\text{rand}})$
  $\alpha_{\text{rand}} \leftarrow \text{atan2}(y_{\text{rand}} - y_{\text{nearest}}, x_{\text{rand}} - x_{\text{nearest}})$
  $\boldsymbol{X}_{\text{rand}} \leftarrow [x_{\text{rand}}, y_{\text{rand}}, \alpha_{\text{rand}}]$
  if not $\text{CollisionFree}(\boldsymbol{X}_{\text{nearest}}, \boldsymbol{X}_{\text{rand}})$ **then**
    continue
  end if
  $\boldsymbol{X}_{\text{steer}} \leftarrow \text{SteerFunction}(\boldsymbol{X}_{\text{nearest}}, \boldsymbol{X}_{\text{rand}})$
  if not $\text{CollisionFree}(\boldsymbol{X}_{\text{nearest}}, \boldsymbol{X}_{\text{new}})$ **then**
    continue
  end if
  Add $\boldsymbol{X}_{\text{rand}}$ to $\mathcal{M}$
  $C_{\text{current}} \leftarrow \text{ContinuityCost}(\boldsymbol{X}_{\text{nearest}}, \boldsymbol{X}_{\text{rand}})$
  if $C_{\text{current}} < C_{\text{best}}$ **then**
    $\boldsymbol{X}_{\text{best}} \leftarrow \boldsymbol{X}_{\text{rand}}$, $\boldsymbol{X}_{\text{new}} \leftarrow \boldsymbol{X}_{\text{steer}}$, $C_{\text{best}} \leftarrow C_{\text{current}}$
  end if
end while
Add $\boldsymbol{X}_{\text{new}}$ to $\boldsymbol{T}$
return $\boldsymbol{X}_{\text{new}}$, $\boldsymbol{X}_{\text{nearest}}$

Algorithm 4 Hierarchical D-RRT* Planning Framework

Input: $\boldsymbol{s}_s$, $\boldsymbol{s}_g$, D_{ts}, Arrival threshold ϵ_g, r_s
Output: Local path $\mathcal{P}_{local}$ to the current target $\boldsymbol{s}_{\text{clt}}$

Initialize: $\boldsymbol{s}_{\text{clt}} \leftarrow \emptyset$, $\mathcal{P}_{local} \leftarrow \emptyset$
while $\|\boldsymbol{s}_s - \boldsymbol{s}_g\| > \epsilon_g$ **do**
  Check if Replanning is Needed:
  if not $\mathcal{P}_{local}$ **or** Path Safety Fails **or** P_{clt} Unsafe **then**
    Replan:
    if $\|\boldsymbol{s}_s - \boldsymbol{s}_g\| > D_{\text{ts}}$ **then**
      $\boldsymbol{s}_{\text{clt}} \leftarrow \text{LocalTargetSelection}(\boldsymbol{s}_s, \boldsymbol{s}_g, D_{\text{ts}})$
    else
      $\boldsymbol{s}_{\text{clt}} \leftarrow \boldsymbol{s}_g$
    end if
    $\mathcal{P}_{local} \leftarrow \text{D-RRT*}(\boldsymbol{s}_s, \boldsymbol{s}_{\text{clt}}, r_s)$
  end if
  Return $\mathcal{P}_{local}$
end while
return Success ▷ Goal reached

In each call to ***Algorithm* 3**, random points $\boldsymbol{X}_{\text{rand}i}$ are repeatedly sampled within the defined free space. Valid candidate points are collected into the set $\mathcal{M}$, up to a maximum of M candidates or until a preset maximum number of sampling attempts is reached. The goal sampling rate is used to guide the distribution of sampled points, ensuring a balance between exploration and exploitation. For each $\boldsymbol{X}_{\text{randi}}$ $(i = 1 \cdots M)$, the nearest node in the tree, $\boldsymbol{X}_{\text{nearest}}$, is identified based on the Euclidean distance. The orientation of $\boldsymbol{X}_{\text{randi}}$ is determined by the vector from $\boldsymbol{X}_{\text{nearest}}$ to $\boldsymbol{X}_{\text{randi}}$:

$$\alpha_{\text{randi}} = \text{atan2}(y_{\text{randi}} - y_{\text{nearest}}, x_{\text{randi}} - x_{\text{nearest}}) \tag{5}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

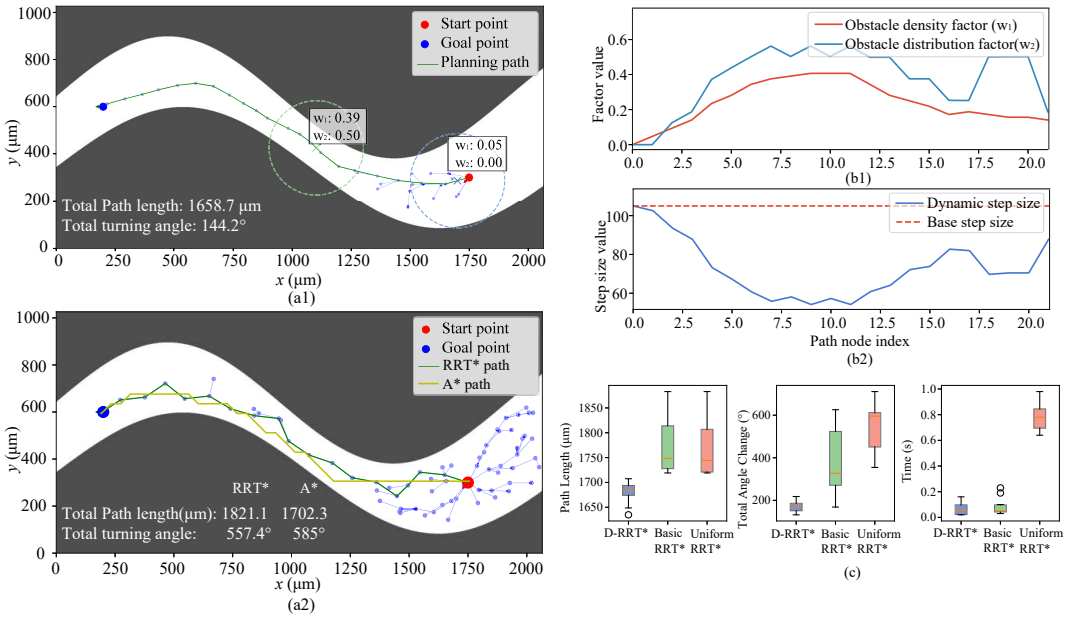

Fig. 3. Path planning simulation results in an 'S'-shaped channel. (a1) The proposed D-RRT* local planner. (a2) Standard RRT* and A*. (b) Dynamic step size adjustment in local path planning of D-RRT*. (b1) Evolution of w_1 and w_2 along the path. (b2) Comparison of dynamic step size with the fixed base step size. (c) Quantitative performance comparison between D-RRT*, standard RRT*, and uniform RRT*.

To ensure feasibility, ***Algorithm* 3** performs collision detection to verify that the path connecting $\boldsymbol{X}_{\text{nearest}}$ and $\boldsymbol{X}_{\text{randi}}$ does not intersect any obstacles. Candidates that fail this check are immediately discarded. For valid candidate, the **SteerFunction** is applied to compute a steer point, $\boldsymbol{X}_{\text{steeri}}$, based on the unicycle motion model as shown in Fig. 1(c2). While $\boldsymbol{X}_{\text{init}}$ has a fixed orientation, subsequent states like $\boldsymbol{X}_{\text{steeri}}$ allow α_{steeri} to vary freely, enabling smoother and more flexible path generation. This approach takes into account the swarm's turning capability, ensuring that the generated path is both realistic and feasible for the swarm's motion. $\boldsymbol{X}_{\text{steeri}}$ is computed as a cumulative result over N discrete time steps as shown in Eq. 6:

$$\begin{cases} x_{\text{steer}i} = x_{\text{nearest}} + \sum_{j=1}^{N} dt \cdot v \cos(\alpha_{\text{steer}i,j}) \\ y_{\text{steer}i} = y_{\text{nearest}} + \sum_{j=1}^{N} dt \cdot v \sin(\alpha_{\text{steer}i,j}) \\ \alpha_{\text{steer}i} = \alpha_{\text{nearest}} + \sum_{j=1}^{N} dt \cdot \omega_j \end{cases} \tag{6}$$

Here, N is determined by the actual extension time T_{d}, with $N = T_{\text{d}}/dt$. Based on the forward velocity v, the actual extension step size is $r_{\text{d}} = \rho(T_{\text{base}} \cdot v)$, where T_{base} is empirically set to 2. For each extension, the detection circle used to compute ρ is centered at the midpoint between the previous $\boldsymbol{X}_{\text{new}}$ and the corresponding $\boldsymbol{X}_{\text{nearest}}$, with the radius manually set based on the local environment. the previous ω_j is the angular velocity at the j-th time step, determined by proportional control:

$$\omega_j = k_p \cdot (\alpha_{\text{randi}} - \alpha_{\text{steeri},j}) \tag{7}$$

where k_p is the proportional gain, and $\alpha_{\text{steeri},j} = \alpha_{\text{steeri},j-1} + dt \cdot \omega_{j-1}$ is the orientation of steer point at step j. To ensure reliability and safety, the path from $\boldsymbol{X}_{\text{nearest}}$ to $\boldsymbol{X}_{\text{steer}i}$ through collision detection. is checked for collisions. If collision-free, the corresponding $\boldsymbol{X}_{\text{rand}i}$ is added to the candidate pool $\mathcal{M}$. Otherwise, the candidate is discarded.

During each sampling process, ***Algorithm* 3** evaluates the smoothness of each connection using the **ContinuityCost**:

$$C_{\text{continuity}} = \lambda_s \left|\alpha_{\text{randi}} - \alpha_{\text{nearest}}\right| \tag{8}$$

where λ_s is a user-defined weight for smoothness. A smaller $C_{\text{continuity}}$ value indicates a smoother and more desirable connection. ***Algorithm* 3** dynamically updates the candidate with the smallest **ContinuityCost** as the best candidate $\boldsymbol{X}_{\text{best}}$, and the corresponding $\boldsymbol{X}_{\text{steer}}$ is directly selected as the new node $\boldsymbol{X}_{\text{new}}$ to be added to the tree $\boldsymbol{T}$ for further expansion. A successful extension of tree $\boldsymbol{T}$ is shown in Fig. 1(c3).

2) Parent Node Selection and rewiring: To further optimize the path, ***Algorithm* 2** evaluates all neighboring nodes within a circular region (the green dash circle in Fig. 1(c3)) of radius r_{search} centered at the newly added node $\boldsymbol{X}_{\text{new}}$, to determine the best parent node $\boldsymbol{X}_{\text{parent}}$.

The parent node is selected using the **ChooseParent** function, which minimizes the total cost C_{total} for connecting $\boldsymbol{X}_{\text{parent}}$ to $\boldsymbol{X}_{\text{new}}$. This optimization aims to achieve a path with minimum energy consumption or minimum traveling distance. C_{total} is defined in Eq. 9a as the weighted sum of

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

the smoothness cost C_{smooth} and the path length cost C_{length}:

$$C_{\text{total}} = \lambda_s \cdot C_{\text{smooth}} + \lambda_l \cdot C_{\text{length}} \tag{9a}$$

$$C_{\text{smooth}} = \sum_{j=1}^{J_{\text{p}}-1} \left|\alpha_{\text{parent}}^{j} - \alpha_{\text{parent}}^{j+1}\right| + |\alpha_{\text{parent}} - \alpha_{\text{new}}| \tag{9b}$$

$$C_{\text{length}} = \sum_{j=1}^{J_{\text{p}}-1} \text{Dist}(\boldsymbol{X}_{\text{parent}}^{j}, \boldsymbol{X}_{\text{parent}}^{j+1}) + \text{Dist}(\boldsymbol{X}_{\text{parent}}, \boldsymbol{X}_{\text{new}}) \tag{9c}$$

Here, λ_s and λ_l are smoothness weight and path length weight, respectively, and are set to balance smoothness and efficiency according to application needs. C_{smooth} penalizes angular deviations along the path. The term $\sum_{j=1}^{J_{\text{p}}-1} \left|\alpha_{\text{parent}}^{j} - \alpha_{\text{parent}}^{j+1}\right|$ computes the cumulative angular change between consecutive nodes from $\boldsymbol{X}_{\text{init}}$ to the candidate parent node $\boldsymbol{X}_{\text{parent}}$, where J_{p} denotes the total number of nodes along the path from $\boldsymbol{X}_{\text{init}}$ to $\boldsymbol{X}_{\text{parent}}$, and j is the index of each node along this path. The term $|\alpha_{\text{parent}} - \alpha_{\text{new}}|$ accounts for the angular change introduced by connecting $\boldsymbol{X}_{\text{parent}}$ to $\boldsymbol{X}_{\text{new}}$, ensuring a smooth transition at the connection point. C_{length} evaluates the total distance along the path. The term $\sum_{j=1}^{J_{\text{p}}-1} \text{Dist}(\boldsymbol{X}_{\text{parent}}^{j}, \boldsymbol{X}_{\text{parent}}^{j+1})$ computes the cumulative path length from $\boldsymbol{X}_{\text{init}}$ to $\boldsymbol{X}_{\text{parent}}$, while $\text{Dist}(\boldsymbol{X}_{\text{parent}}, \boldsymbol{X}_{\text{new}})$ accounts for the Euclidean distance from $\boldsymbol{X}_{\text{parent}}$ to $\boldsymbol{X}_{\text{new}}$. This term promotes shorter paths, reducing the overall path length. As shown in Fig. 1(c4), the connection of node $\boldsymbol{X}_{\text{new}i}$ and node $\boldsymbol{X}_{\text{near}}$(red dotted line) will be discarded. Subsequently, node $\boldsymbol{X}_{\text{parent}}$ inside the circular boundary is chosen as the new parent node and the red line denotes the new connection.

Once the best parent node $\boldsymbol{X}_{\text{parent}}$ is determined, ***Algorithm* 2** performs a rewiring process to further optimize the tree structure by reducing the total cost for neighboring nodes, as demonstrated in Fig. 1(c5). For every neighboring node $\boldsymbol{X}_{\text{near}i}$ within r_{search}, the algorithm evaluates whether reconnecting $\boldsymbol{X}_{\text{near}i}$ to $\boldsymbol{X}_{\text{new}}$ results in a lower total cost from the root. The rewiring cost is expressed as:

$$\begin{aligned} C_{\text{neari,new}} &= C_{\text{new}} + \lambda_s \cdot |\alpha_{\text{new}} - \alpha_{\text{neari}}| + \lambda_l \cdot \text{Dist}(\boldsymbol{X}_{\text{new}}, \boldsymbol{X}_{\text{neari}}) \\ &= \sum_{j=1}^{J_{\text{new}}-1} \left(\lambda_s \cdot \left|\alpha_{\text{new}}^{j} - \alpha_{\text{new}}^{j+1}\right| + \lambda_l \cdot \text{Dist}(\boldsymbol{X}_{\text{new}}^{j}, \boldsymbol{X}_{\text{new}}^{j+1})\right) \\ &\quad + \lambda_s \cdot |\alpha_{\text{new}} - \alpha_{\text{neari}}| \\ &\quad + \lambda_l \cdot \text{Dist}(\boldsymbol{X}_{\text{new}}, \boldsymbol{X}_{\text{neari}}) \end{aligned} \tag{10}$$

where C_{new} represents the cumulative angular deviation and path length from $\boldsymbol{X}_{\text{init}}$ to $\boldsymbol{X}_{\text{new}}$. Let $C_{\text{near}i,\text{orig}}$ denote the original total cost from the root to $\boldsymbol{X}_{\text{near}i}$ via its current parent. If $C_{\text{near}i,\text{new}} < C_{\text{near}i,\text{orig}}$, then the parent of $\boldsymbol{X}_{\text{near}i}$ is updated to $\boldsymbol{X}_{\text{new}}$, and the tree structure is modified accordingly.

As illustrated in Fig. 1(c5), the rewiring process assigns the i-th new node $\boldsymbol{X}_{\text{new}i}$ as the new parent node of $\boldsymbol{X}_{\text{child}}$ (previously $\boldsymbol{X}_{\text{near}}$), thereby reducing the total cost from the root to $\boldsymbol{X}_{\text{child}}$. As a result, $\boldsymbol{X}_{\text{child}}$ is disconnected from its previous parent node $\boldsymbol{X}_{\text{parentr}}$, as depicted by the red dotted line. Subsequently, as shown in Fig. 1(c6), the final path is traced back from $\boldsymbol{X}_{\text{goal}}$ to $\boldsymbol{X}_{\text{init}}$, forming the optimal path $\mathcal{P}_{local}$ based on the cumulative cost criteria. This path $\mathcal{P}_{local}$ is then passed to the learning-based controller module, which generates corresponding control commands for the robot to execute the planned motion. Specifically, the controller is trained using imitation learning from expert demonstration trajectories, enabling it to generate smooth and robust control actions from the planned path in real time.

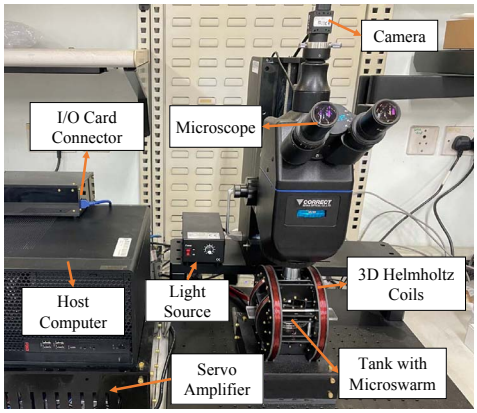

Fig. 4. Experimental setup.

3) Comparative studies: To evaluate the effectiveness of the proposed D-RRT* local path planner, we conducted comparative simulations against several mainstream algorithms, including standard RRT*, uniform RRT* [20], and the classic A* algorithm in an 'S'-shaped channel environment with two wider ends (see Fig. 3(a)). All planners started from $\boldsymbol{X}_{\text{start}} = (200, 700, 180°)$ and navigated to $\boldsymbol{X}_{\text{goal}} = (1750, 300)$, using identical parameters and a safety distance of 40 μm to ensure fair, collision-free comparisons.

D-RRT* demonstrated the ability to dynamically adjust its expansion step size by utilizing a detection circle for each extension. Within detection circles, w_1 and w_2 are evaluated, as shown in Fig.3(a1). In open areas (e.g., near the start in the purple dashed detection circle), low obstacle density and distribution factors (w_1, w_2) allowed larger steps for efficiency. In narrow regions (e.g., the green dashed detection circle), higher w_1 and w_2 reduced step size to enhance safety. The evolution of w_1 and w_2 along the path is depicted in Fig. 3(b), where both factors dynamically adapt to the environment and influence the step size to ensure smooth and safe navigation. By incorporating smoothness cost and adaptive step size, D-RRT* local planner achieved a path length of 1658.7 μm and a total turning angle of 144.2°, which are 8.9% shorter and 68.2% lower than those of standard RRT* (1821.1μm, 453.7°), and 2.6% shorter and 75.4% lower than those of A* (1702.3 μm, 585°), respectively. The adaptive step size in D-RRT* enabled efficient navigation through narrow sections, minimizing detours.

Across 10 trials (Fig. 3(c)), D-RRT* consistently outperformed standard and uniform RRT* in path quality and efficiency. The median path length for D-RRT* was 1679.8 μm (5.5% shorter), and the median turning angle was 170.8° (55.1% lower) compared to standard RRT*. Compared to A*, which produced a fixed result (1702.3 μm, 585°, and 0.01 s), D-RRT* achieved a shorter and much smoother path, while

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

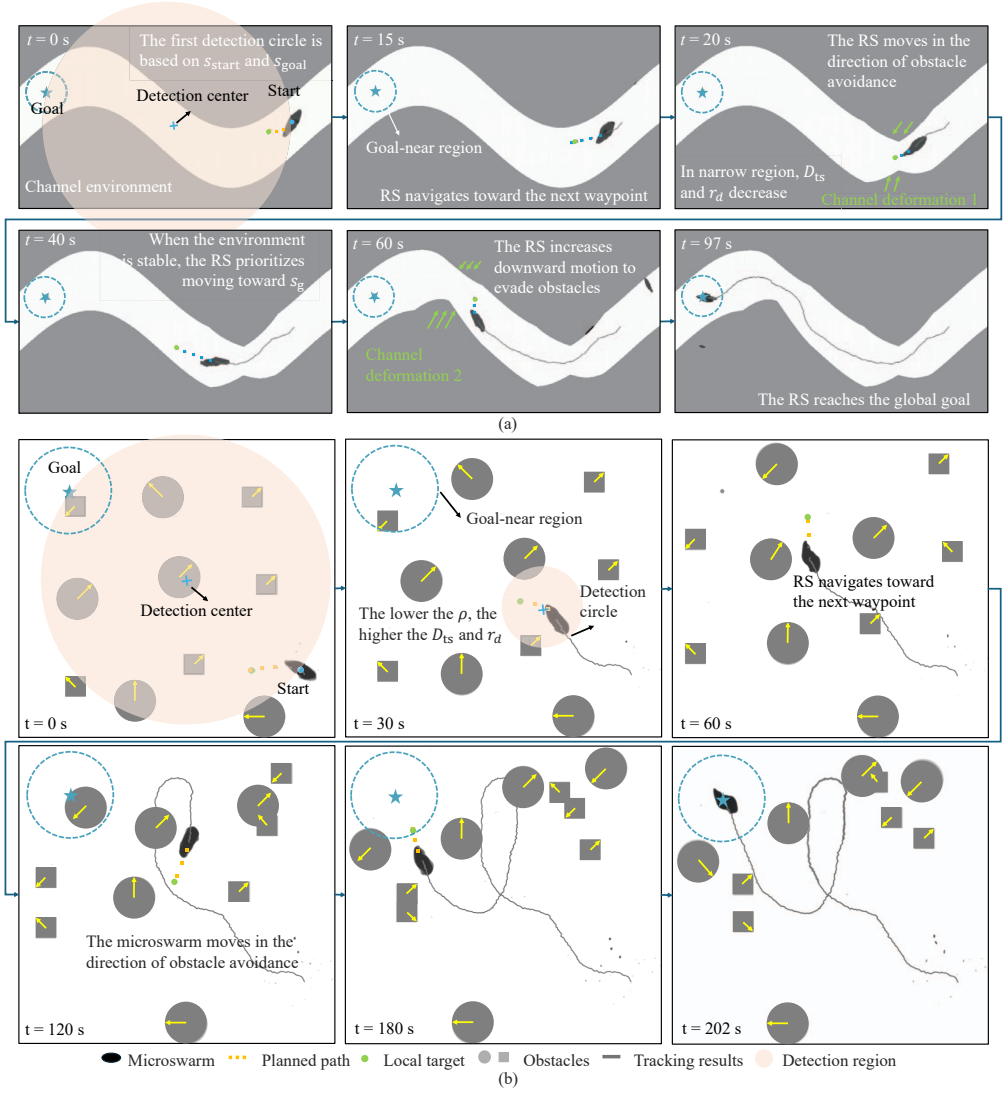

Fig. 5. Real-world navigation results. (a) Experiment in dynamically deforming channel. (b) Experimental in a multiple dynamic obstacle environment.

maintaining planning times suitable for real-time applications. Integration with efficient local target selection can further reduce overall planning time.

The overall hierarchical planning framework combines the above components, as summarized in ***Algorithm* 4**. The workflow alternates between local target selection and local path planning until the global goal is reached.

III. Experimental Results

To validate the proposed framework under real-world conditions, we conducted multiple navigation experiments using a reconfigurable microswarm of paramagnetic Fe_3O_4 nanoparticles [10] in an acrylic tank with a silicon substrate and 0.5% SDS solution. The addition of SDS, a commonly used surfactant, effectively reduces friction between the microswarm and the substrate, ensuring smooth and responsive movement. The experimental setup is shown in Fig. 4. Real-time image processing tracked the microswarm's position and distribution, while the D-RRT* planner provided target positions for closed-loop control. Actuation was achieved via a 3D Helmholtz coil system generating oscillating magnetic fields (16 Hz, 10 mT). The control algorithm, based on imitation learning, was implemented in LabVIEW and MATLAB at 10 Hz, ensuring stable and precise motion in low Reynolds number environments.

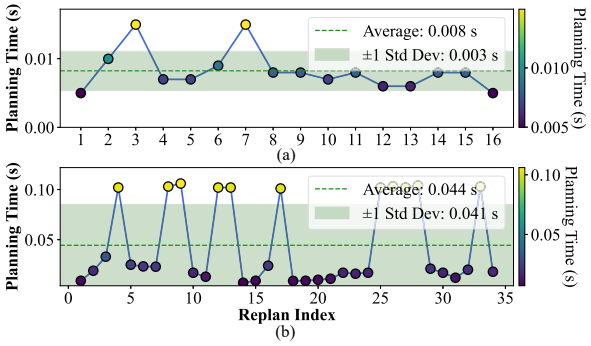

Fig. 6. Planning time statistics during experiments. (a) Dynamically deforming channel. (b) Multiple dynamic obstacle environment.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Practical biomedical environments often involve features such as dynamically deforming vessel-like channels and multiple moving obstacles, which can affect microswarm navigation. To evaluate the robustness and adaptability of the proposed framework, we conducted two representative experiments: (1) navigation through a dynamically deforming channel ((Fig. 5)(a)) from $(1982, 552, 111^\circ)$ to $(240, 880)$, and (2) navigation among several moving obstacles with different velocities ((Fig. 5)(b)) from $(1900, 1500, 90^\circ)$ to $(350, 350)$, both with a safety margin of 90 μm.

In both experiments, the D-RRT* planner adaptively replanned the navigation path as the environment changed, ensuring continuous and collision-free guidance of the swarm toward each waypoint. The local target and detection region were dynamically adjusted in response to either channel deformation or obstacle movement. In the deforming channel, a total of 16 planning events occurred (Fig. 6(a)), with a maximum planning time of 0.015 s, an average of 0.008 s, and a standard deviation of 0.003 s. In the dynamic obstacle scenario, 34 planning events were recorded (Fig. 6(b)), with a maximum planning time of 0.10 s, an average of 0.044 s, and a standard deviation of 0.041 s. Most of the planning times in both cases fell within one standard deviation, demonstrating the computational efficiency and stability of the framework. Notably, the average planning time in each case represents a significant reduction compared to the global planning time median (0.055 s, Fig. 3(c)), with reductions of 85.5% and 20.0%, respectively, highlighting the efficiency and adaptability of the D-RRT* algorithm for real-time path adjustment and obstacle avoidance in complex environments.

Throughout both experiments, the microswarm maintained the required safety margin, effectively avoided collisions, and reached the global goal with high accuracy. Supplementary videos further validate the robustness, efficiency, and practical applicability of the proposed framework.

IV. CONCLUSION

In this work, we propose a hierarchical D-RRT* planning framework. The planner incorporates dynamic step size adjustment, local target selection and local path planning that account for path smoothness and the turning capabilities of the microswarm, minimizing smoothness costs and travel distance while significantly enhancing real-time performance in dynamic obstacle environments. Comparative simulations and navigation experiments in environments with various dynamic obstacles validated the framework's path smoothness and planning efficiency. The proposed framework advances microswarm navigation in dynamic and complex environments, and can be extended to biomedical applications such as targeted delivery, minimally invasive therapy, and micromanipulation, as well as other scenarios requiring precise navigation in constrained spaces.

Despite promising results, several limitations remain. The proposed framework assumes that dynamic obstacles do not move faster than the microswarm. If obstacles are significantly faster, the planner may not be able to generate a safe path in time, reducing the effectiveness of dynamic avoidance. Future work will explore more advanced adaptive strategies to address these challenges.

REFERENCES

[1] J. Choi, J. Hwang, J.-y. Kim, and H. Choi, "Recent progress in magnetically actuated microrobots for targeted delivery of therapeutic agents," *Advanced Healthcare Materials*, vol. 10, no. 6, p. 2001596, 2021.

[2] B. J. Nelson, S. Gervasoni, P. W. Chiu, L. Zhang, and A. Zemmar, "Magnetically actuated medical robots: An in vivo perspective," *Proceedings of the IEEE*, vol. 110, no. 7, pp. 1028–1037, 2022.

[3] Z. Zhang, X. Wang, J. Liu, C. Dai, and Y. Sun, "Robotic micromanipulation: Fundamentals and applications," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 181–203, 2019.

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

[5] J. Li, B. Esteban-Fernández de Ávila, W. Gao, L. Zhang, and J. Wang, "Micro/nanorobots for biomedicine: Delivery, surgery, sensing, and detoxification," *Science robotics*, vol. 2, no. 4, p. eaam6431, 2017.

[6] R. Nauber, S. R. Goudu, M. Goeckenjan, M. Bornhäuser, C. Ribeiro, and M. Medina-Sánchez, "Medical microrobots in reproductive medicine from the bench to the clinic," *Nature Communications*, vol. 14, no. 1, p. 728, 2023.

[7] Y. Hou, H. Wang, R. Fu, X. Wang, J. Yu, S. Zhang, Q. Huang, Y. Sun, and T. Fukuda, "A review on microrobots driven by optical and magnetic fields," *Lab on a Chip*, vol. 23, no. 5, pp. 848–868, 2023.

[8] A. Aziz, S. Pane, V. Iacovacci, N. Koukourakis, J. Czarske, A. Menciassi, M. Medina-Sánchez, and O. G. Schmidt, "Medical imaging of microrobots: Toward in vivo applications," *ACS nano*, vol. 14, no. 9, pp. 10 865–10 893, 2020.

[9] J. Yu, L. Yang, and L. Zhang, "Pattern generation and motion control of a vortex-like paramagnetic nanoparticle swarm," *The International Journal of Robotics Research*, vol. 37, no. 8, pp. 912–930, 2018.

[10] J. Yu, B. Wang, X. Du, Q. Wang, and L. Zhang, "Ultra-extensible ribbon-like magnetic microswarm," *Nature communications*, vol. 9, no. 1, p. 3260, 2018.

[11] J. Yu, L. Yang, X. Du, H. Chen, T. Xu, and L. Zhang, "Adaptive pattern and motion control of magnetic microrobotic swarms," *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1552–1570, 2021.

[12] Y. Li, Y. Huo, X. Chu, and L. Yang, "Automated magnetic microrobot control: From mathematical modeling to machine learning," *Mathematics*, vol. 12, no. 14, p. 2180, 2024.

[13] J. Jiang, L. Yang, S. Yang, and L. Zhang, "A deep learning-based framework for environment-adaptive navigation of size-adaptable microswarms," *Engineering*, 2024.

[14] H. Wang, Y. Qiu, Y. Hou, Q. Shi, H.-W. Huang, Q. Huang, and T. Fukuda, "Deep reinforcement learning-based collision-free navigation for magnetic helical microrobots in dynamic environments," *IEEE Transactions on Automation Science and Engineering*, 2024.

[15] Y. Liu, H. Chen, Q. Zou, X. Du, Y. Wang, and J. Yu, "Automatic navigation of microswarms for dynamic obstacle avoidance," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2770–2785, 2023.

[16] Y. Liu, L. Zhang, X. Liu, and Q. Fan, "Safety-enhanced navigation planning for magnetic microrobots," *IEEE Transactions on Automation Science and Engineering*, 2025.

[17] Y. Liu, Z. Hou, J. Qu, X. Liu, and Q. Fan, "Optimized rrt planning with cma-es for autonomous navigation of magnetic microrobots in complex environments," *IEEE/ASME Transactions on Mechatronics*, 2024.

[18] Y. Liu, H. Wang, X. Wu, J. Qu, X. Liu, and Q. Fan, "Autonomous navigation of magnetic microrobots with improved planning and control in complex environments," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 2421–2432, 2024.

[19] Y. Liu, H. Wang, and Q. Fan, "Adaptive learning and sliding mode control for a magnetic microrobot precision tracking with uncertainties," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7767–7774, 2023.

[20] L. Yang, J. Jiang, X. Gao, Q. Wang, Q. Dou, and L. Zhang, "Autonomous environment-adaptive microrobot swarm navigation enabled by deep learning-based real-time distribution planning," *Nature Machine Intelligence*, vol. 4, no. 5, pp. 480–493, 2022.

[21] Q. Zou, X. Du, Y. Liu, H. Chen, Y. Wang, and J. Yu, "Dynamic path planning and motion control of microrobotic swarms for mobile target tracking," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 4, pp. 2454–2468, 2022.