Supporting Information for "Hierarchical planning with deep reinforcement learning for three-dimensional navigation of microrobots in blood vessels"

Yuguang Yang[1], Michael A. Bevan[1], and Bo Li[1]

[1]Affiliation not available

July 31, 2022

Abstract

This supporting information includes supplemental figures, movies, additional results, and the key steps in the neural network training algorithm.

Corresponding author Email: yyang60@jhu.edu (Y.Y.) libome@tsinghua.edu (B.L.)

1 List of Supplemental Figures

- Fig. S1. The Actor-Critic architecture used to learn optimal control policies for the microrobot.
- Fig. S2. The approximate RBC shape we used to facilitate a fast collision check in the simulation.
- Fig. S3. Scheme of training and evaluation workflow. Efficient training is achieved by selecting increasingly challenging tasks as motivated by curriculum learning.
- Fig. S4. Navigation and localization trajectories of microrobots in free space with different propulsion rotation ratios.
- Fig. S5. Navigation and localization trajectories of microrobots in free space with different external flow fields.
- Fig. S6. Navigation of microrobots in blood vessels with different external flow fields.

2 Supplemental Movies

- Movie S1. Navigation and localization of microrobots in free space with different external flow fields.
- Movie S2. Navigation of microrobots in blood vessels with different vessel diameter D and RBC volume fraction f. Specifically, from left to right, $D = 12$um, f ˜ 10%; $D = 25$um, f˜10%; $D = 50$um, f˜5%; $D = 50$um, f˜10%.
- Movie S3. Navigation of microrobots in curved blood vessels with varying cross-section diameter.
- Movie S4. Exhaustive spatial survey in a blood vessel. From left to right, the blood vessel has zero RBCs, ˜5% RBCs, and ˜10% RBCs.

3 Supplemental Methods and Results

3.1 Actor-Critic deep convolution neural network architecture

3.1.1 Actor network

There are two inputs to the Actor network [Fig. S1]. The first input is the pixel-level binary sensory input of $30 \times 30 \times 30$ cubic neighborhood centering on the microrobot and aligned with its self-propulsion direction $\mathbf{p}$. The second input is a six-dimensional vector as the concatenation of the target position in the microrobot's local coordinate frame and the self-propulsion direction $\mathbf{p}$. The neighborhood sensory input first enters a 3D convolutional layer (LeCun et al., 1989, 2015; Maturana and Scherer, 2015) consisting of 32 filters with kernel size $2 \times 2 \times 2$, stride 1, and padding of 1, following a batch normalization layer (Loffe and Szegedy, 2015), a rectifier nonlinearity (Nair and Hinton, 2010) (i.e., max(0, x)) and a $2 \times 2 \times 2$ maximum pooling layer. The output then enters a second convolutional layer consisting of 64 filters and the same kernel, stride and padding as the previous layer, followed similarly by a batch normalization layer, a rectifier nonlinearly, and a maximum pooling layer. The local target coordinate first enters a fully connected layer consisting of 32 units, followed by rectifier nonlinearity. Then the output from the target coordinate input and the sensory input will merge and enter a fully connected layer of 64 units followed by rectifier nonlinearity. The output layer is a fully-connected linear layer with two outputs associated with the choice of w_1 and w_2. The tanh nonlinearity is applied to constrain the two outputs.

3.1.2 Critic network

There are three inputs to the critic network [Fig. S1]. The first is the binary cubic image of the neighborhood same as that in the Actor network. The neighborhood sensory input first enters a convolutional layer consisting of 32 filters with kernel size $2 \times 2 \times 2$, stride 1, and padding of 1, following a batch normalization layer, a rectifier nonlinearity (i.e., max(0, x)) and a $2 \times 2 \times 2$ maximum pooling layer. The output then enters a second convolutional layer consisting of 64 filters and the same kernel, stride, and padding as the previous layer, followed similarly by a batch normalization layer, a rectifier nonlinearly, and a maximum pooling layer. The local proxy target and the self-propulsion director p will first concatenate with the action output from the Actor network. The 8-dimensional concatenated vector then will enter a fully connected layer consisting of 32 units followed by rectifier nonlinearity. Then the output from the target coordinate input and the sensory input will merge and enter a fully connected layer of 64 units followed by rectifier nonlinearity. The output layer is a fully-connected linear layer with one output as the Q value given input of observation and action.

3.2 RBC approximation and collision dynamics

During the simulation process, we need to perform a collision check and compute sensory input on the fly. In every integration time step, we evolve the microrobot state $(\mathbf{r}(t), \mathbf{p}(t))$using the equation of motion [Eq. (1) in the main text] for an integration time step of Dt. Then we perform a collision check between the new position of microrobot $\mathbf{r}(t + \Delta t)$ and an approximate RBC shape [Fig. S2]. If the new position of microrobot $\mathbf{r}(t + \Delta t)$ collides with an RBC, we set its position back to the previous one $\mathbf{r}(t)$. However, we still need to update its orientation. This approximation is reasonable because the robot-RBC collision is not the dominant factor that affects the navigation process. In the new position, we construct the 3D binary sensory matrix, if the pixel center is inside an approximate RBC, that pixel will take values 1 and 0 otherwise.

3.3 Training algorithms and procedures

The algorithm we used to train the agent is the deep deterministic policy gradient algorithm (Lillicrap et al., 2015) plus the hindsight experience replay data augmentation (Andrychowicz et al., 2017) and scheduled

multi-stage learning following the idea of curriculum learning (Florensa et al., 2017). The whole training and evaluation pipeline is depicted in Fig. S3. At the beginning of each episode, the initial robot state and the target position are randomly generated in such a way that their distance gradually increases from a small value. More formally, let $D(k)$denote the maximum distance between the generated initial microrobot position and the target position at training episode k, which is given by

$$D(k) = S_m \times (T_\mathrm{e} + (T_\mathrm{e} - T_\mathrm{s}) \exp(-k/T_\mathrm{d})$$

where S_m is the maximum of width and height of the training environment (at free space we set $S_m = 20a=$), T_s is the initial threshold, T_e is the final threshold, and T_d is the threshold decay parameter. Then during the training process, the neural network gradually learns control strategies of increasing difficulties (in terms of initial distance to the target).

To alleviate the exploration-exploitation dilemma, during the training process, we added noises to the actions from Actor network to enhance the exploration in the state and policy space. The noise is sampled from an Ornstein–Uhlenbeck (OU) process (on each dimension) given by

$$\mathrm{d}\eta = -\alpha(m - \eta)\mathrm{d}t + \sigma_\mathrm{OU}\mathrm{d}B_t$$

where α is the reversion parameter, m is the mean level parameter, σ_{OU} is the volatility parameter, and B_t is the standard Brownian motion process.

In the Q function formulation, we set the discount factor γ to 0.99 to encourage the microrobot to seek rewards in the long run and R is the instant reward function, where R is set equal to 1 for all states that are within a threshold distance 1 to the target, and 0 otherwise.

The blood environments used for training include

- Cylindrical blood vessel (Radius 50, Height = 100), RBC volume fraction 5%;
- Cylindrical blood vessel (Radius 50, Height = 100), RBC volume fraction 10%;
- Cylindrical blood vessel (Radius 25, Height = 100), RBC volume fraction 10%;
- Cylindrical blood vessel (Radius 12, Height = 100), RBC volume fraction 10%;
- Cylindrical blood vessel (Radius 50, Height = 100), no RBC.

There are two loss functions we used to train the Actor network and the Critic network, respectively. By minimizing the loss function associated with the Critic network, the Critic network is optimized to approximate the optimal Q function; By minimizing the loss function associated with the Actor network, the Actor network optimizes the approximated π. The complete algorithm is given below.

Algorithm: deep deterministic policy gradient with hindsight experience replay

Initialize replay memory M to capacity N_M

Initialize Actor network μ with random weight θ^μ and critic network Q with random weights θ^Q

Initialize target Actor network μ' and critic network Q' with random weights $\theta^{\mu'}$ and $\theta^{Q'}$

For episode 1, . . . , NE **do**

Initialize particle state s0 and target position

Obtain initial observation $\phi(s_1)$

For n =1, . . . , maxStep **do**

Select an action a_n from Actor network plus an additional perturbation sample from an OU process.

Execute action a_n using simulation and observe new state s_{n+1} and reward $r(s_{n+1})$

Generate observation state $\phi(s_{n+1})$ at state s_{n+1}

Store transition $(\phi(s_n), a_n, r(s_{n+1}), \phi(s_{n+1})$ in replay memory M

Store extra hindsight experience in M every H step

Sample random mini-batch transitions $(\phi(s_j), a_j, r(s_{j+1}), \phi(s_{j+1}))$ of size B from M

Set target value y_j: If s_{j+1}arrives at the target, $y_j = r(s_j)$; else $y_j = r(s_j) + \gamma Q'(\phi(s_{j+1}, \arg\max_v(\phi(s_{j+1}), v)$

Perform a gradient descent step on $(y_j - Q(\phi(s_j)), a_j)^2$to update the critic network parameters θ^Q

Update the actor network using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a \mid \theta^Q)|_{s-s_i, a-\mu(s_i)} \nabla_{\theta^\mu} \mu(s \mid \theta^\mu)|_{s_i}$$

Update the target networks:

$$\theta^{Q'} = \beta\theta^Q + (1-\beta)\theta^{Q'}$$
$$\theta^{\mu'} = \beta\theta^\mu + (1-\beta)\theta^{\mu'}$$

End For

End For

Training Parameters

Training episode, N_E = ˜80000

Minibatch size, B=64

Replay memory size, N_M=500000

Target network update frequency C=100

Discount factor γ=0.99

Learning rate α=0.00001

Soft update parameter β=0.01

OU process m, s, a=0,0.5,0.15

Target generation T_s, T_e, T_d=0.3,1,10000

Max step in an episode, maxStep=100

Sensor window size W=11, pixel resolution U=2.5

3.4 Simulation setup and performance evaluation

3.4.1 Sensory input construction

The local neighborhood sensory input is obtained by first constructing a cube of width $W = 11$ centering on the microrobot and aligned with its orientation and then extracting a $W \times W \times W$ binary 3D image with a pixel resolution of $U = 2a$ (1 when there is an overlap with an RBC and 0 otherwise). Target positions are represented in local coordinate system of the motor. RBC is modeled by biconcave shape (Das et al., 2019) with random position and orientation. The RBCs generated in the simulation have diameters randomly sampled between 6um and 8um. We employed an approximate RBC shape to enable fast computation of sensory input and collision dynamics.

3.4.2 Local sensor design consideration

Both Actor and Critic neural networks employ 3D convolution neural layers to process local sensory information, represented by a $W \times W \times W$ binary 3D image ($W = 11$) with a pixel resolution of $U = 2a$.

The designed sensor for the local neighborhood has the following considerations. A large vision field allows the microrobot to detect obstacles early and take paths that avoid clashing with obstacles. A large vision field also captures the rich configuration that allows the learning of better and more robust navigation strategies. However, a significant large vision field contains information not essential for local path planning, which increases the learning computational cost and sensor hardware design difficulties. In terms of sensor resolution, high resolution will increase the computation cost while low resolution may disable robots to detect small trapping features of an RBC.

1.1.3 Curved vessel geometry

The central axis of the curved vessel is characterized by a 3D parameteric curve function (x_c, y_c, z_c) given by ,where L controls the length of the vessel (e.g., $L = 500a$). The section radius R_c of the vessel is varying around the axis line, which is given by ,where R_{avg} controls its average radius. We have considered two cases in Fig. 4 (H) and (I) in main text, where we use the same $k_1 = 0.05a^{-1}, k_2 = 0.02a^{-1}, R_0 = 10a, R_{avg} = 25a, L = 500a$, but with $R_1 = 5a$ for (H) and $R_1 = 15a$for (I).

3.5 Control policy mapping under perturbations

By exploiting symmetry existing in the system, we can reuse the control policy p obtained at one set of hyperparameters $(v^\star_{sp}, v^\star_{max})$ to another hyperparameter setting (v_{sp}, v_{max}) and save the re-training cost. We can write the control policy $\pi(\mathbf{r}^t, \mathbf{r}, \mathbf{p}, \phi(s); v_{SP}, v_{max})$as a function of observational variables $\mathbf{r}^t, \mathbf{r}, \mathbf{p}, \phi(s)$, which characterize the system state and the hyperparameters v_{SP} and w_{max}, which specify the physical parameters of the microrobot.

Now we discuss two scenarios where we can reuse a control policy p obtained at one set of hyperparameters $(v^\star_{sp}, v^\star_{max})$. Since the microrobot is constantly propelling and the rotation decision w ultimately affect the trajectory's minimum radius of curvature v_{SP}/v_{max}, without loss of generality, we consider the policy mapping when $R_{\mathrm{xw}} \neq R^*_{\mathrm{sw}}$, where $R^\star_{\mathrm{xw}} = v^\star_{\mathrm{SP}}/w^\star_{max}$. As the rotational decision on w aims to proactively adjust directions, a mimicking strategy is that a microrobot with hyperparameter R_{vw} mimics the trajectory of the baseline microrobot with hyperparameter $R^\star_{vw}$ as much as feasible, until the rotation reaches its limitation w_{max}. The policy mapping in terms of magnitude of (w_1, w_2) under hyperparameter R can be expressed as

$$w_i = \min\left(w_{\max}\frac{w\left(R^*_{\mathrm{w}}\right)}{v^*_{\mathrm{SP}}}v_{\mathrm{SP}}\right), i = 1, 2,$$

where $w_i(R^\star_{vw})$ is the magnitude of in-plane rotation ($i = 1$) and out-of-plane rotation ($i = 2$) from control policy learned under hyperparameter $R^\star_{vw}$.

Now consider there is an ambient flow field (or any other external force causing a constant drift of microrobot), whose velocity is characterized by $\mathbf{v}_f$, that modifies the velocity of the microrobot. The deterministic velocity of the microrobot now is the sum of original propulsion velocity $\mathbf{v}_{SP} \times \mathbf{p}$ and the dirft velocity $\mathbf{v}_f$. Equivalently, we can define a modified propulsion direction $\mathbf{p}_f$ and the corresponding modified self-propulsion speed $\mathbf{v}_{SP,f}$ via

$$\mathbf{v}_{\mathrm{SP,f}} = v_{\mathrm{SP}} \cdot \mathbf{p} + \mathbf{v}_{\mathrm{f}}, v_{\mathrm{SP},f} = ||\mathbf{v}_{\mathrm{SP}}||, \mathbf{p}_{\mathrm{f}} = \frac{\mathbf{v}_{\mathrm{SP,f}}}{v_{\mathrm{SP},f}}$$

We can treat a microrobot under external flow field as if a microrobot without external flow field but with a modified hyperparameter $(\mathbf{v}_{SP,f}, w_{max})$. Accordingly, the new control policy with flow field is now given b y $\pi(\mathbf{r}^t, \mathbf{r}, \mathbf{p}, \phi(s); v_{SP,f}, v_{max})$, where we can employ Eq. to reuse the policy.

3.6 Estimate shortest path distance

We estimate the shortest path distance between arbitrary two points in the blood environment [Fig. 6 in main text] using an approximate graph algorithm. We first created a set of 3D lattice points, with a step size of a in x, y, and z directions, respectively, to span the space of the test environments. We then remove lattice points that are outside the vessel or are overlapping with RBCs (we assume lattice point has a radius of a, same size as the microrobot). We then construct a weighted K-nearest neighbor graph (K=26), where each lattice point is a graph node, nodes are connected by edges if they are within the 26 nearest neighbors, and the edge weight is the distance between the connected nodes. Given a start point and a target, we associate them with the nearest lattice points in the graph and then use the Dijkstra algorithm to compute their shortest distance in the graph. The computed shortest path distance in the graph is used as the approximate the shortest path distance between the query points.

3.7 Additional results

3.7.1 Free space navigation under different hyperparameters

In Fig. S4, we tested the policy mapping formula [Eq. (S3)] under different hyperparameter settings. The neural network is trained under one hyperparameter setting $R^{\star}_{vw} = 1$). The mapped policies are still effective under other hyperparameters $R_{vw} = 2$ and $R_{vw} = 4$. Note that here we only need to consider the case $R_{vw} > R^{\star}_{vw}$, since Eq. (3) says that rotational decisions remain unchanged when $R_{vw} < R^{\star}_{vw}$.

3.7.2 Free space navigation under flow fields

In Fig. S5, we tested the policy mapping formula [Eq. (S3) and (S4)] under different hyperparameter settings. The neural network is trained under one hyperparameter setting ($R_{vw} = 1$) and no external flow field. The mapped policies are still effective under external flow fields $v_f = 0.5v_{SP}$ and $v_f = 0.8v_{SP}$.

3.7.3 Blood vessel navigation under flow fields

In Fig. S6, we tested the policy mapping formula [Eq. (S3) and (S4)] under the external flow field when a microrobot is navigating inside a blood vessel with RBCs. When external flow fields are small ($v_f \leq 0.5v_{SP}$) and RBCs are dilute (e.g., 5%), the mapping formula can enable the microrobot to achieve targets without getting frequent traps. When external flow fields are large, microrobots can get trapped easily. This is because the existence of RBCs breaks space symmetry and therefore the correct policy in a crowded RBC environment is beyond the simple formula in Eq. (S3) and (S4).

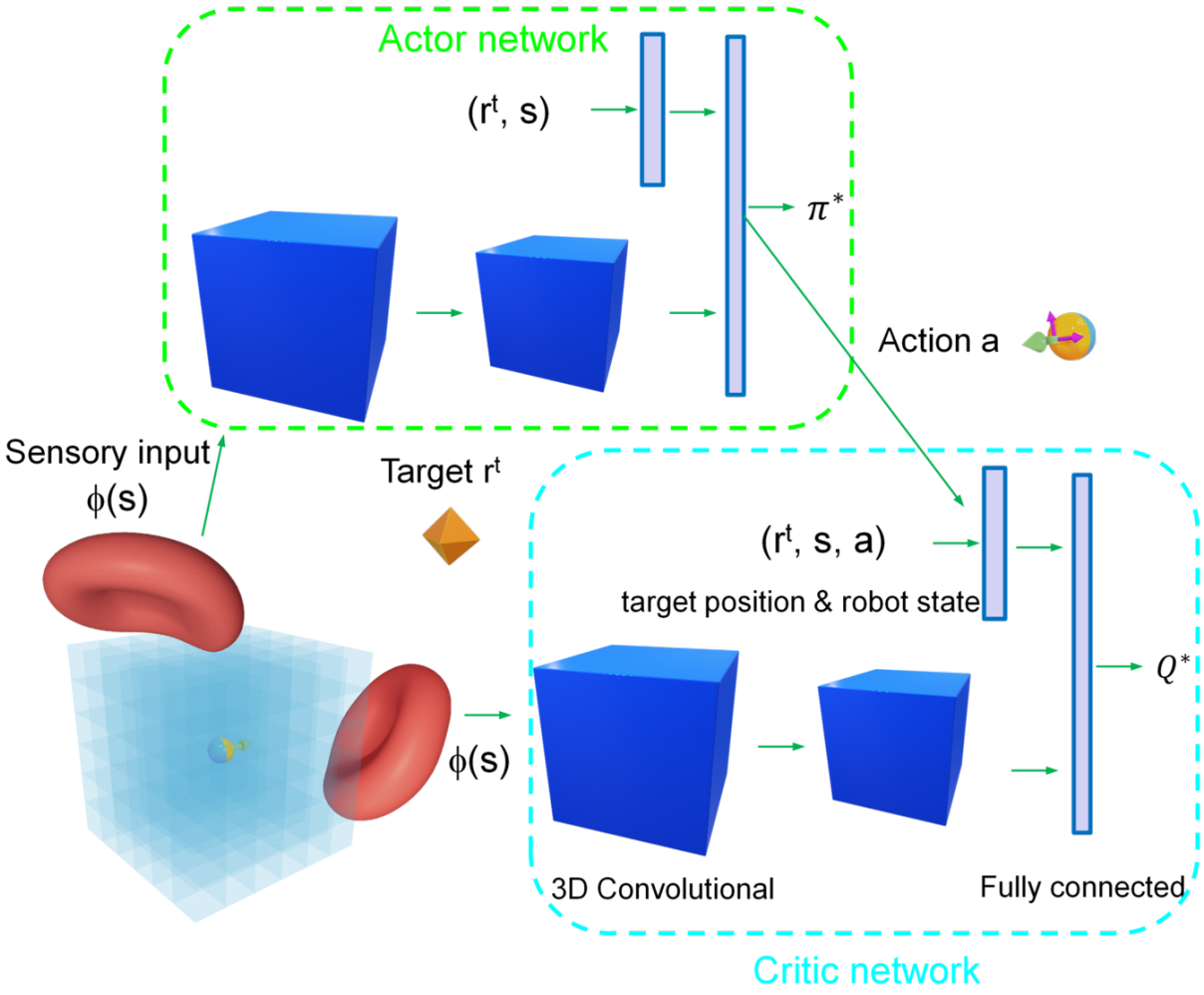

Figure 1: Fig. S1. The Actor-Critic architecture used to learn optimal control policies for the microrobot.

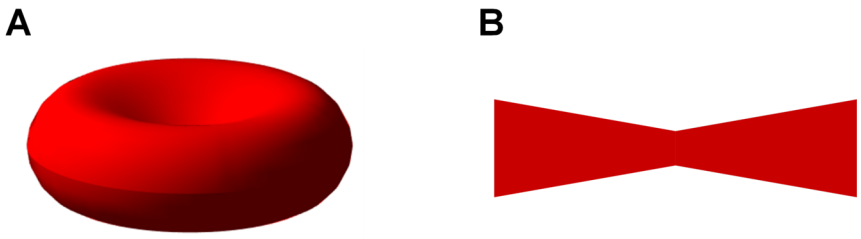

Figure 2: Fig. S2. (A) Biconcave shape of RBC. (B) To enable fast collision check in the simulation, we approximate a biconcave shaped RBC by a geometric shape whose cross-section is the two trapezoids glued together by their top. The approximate shape has the same diameter and the same thickness on both the thickest and thinnest part .

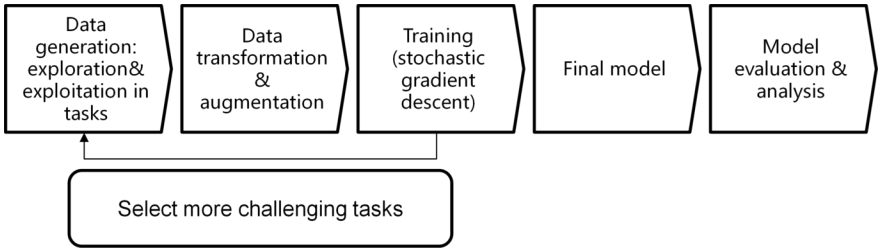

Figure 3: Fig. S3. Scheme of training and evaluation workflow. Efficient training is achieved by selecting increasingly challenging tasks as motivated by curriculum learning.

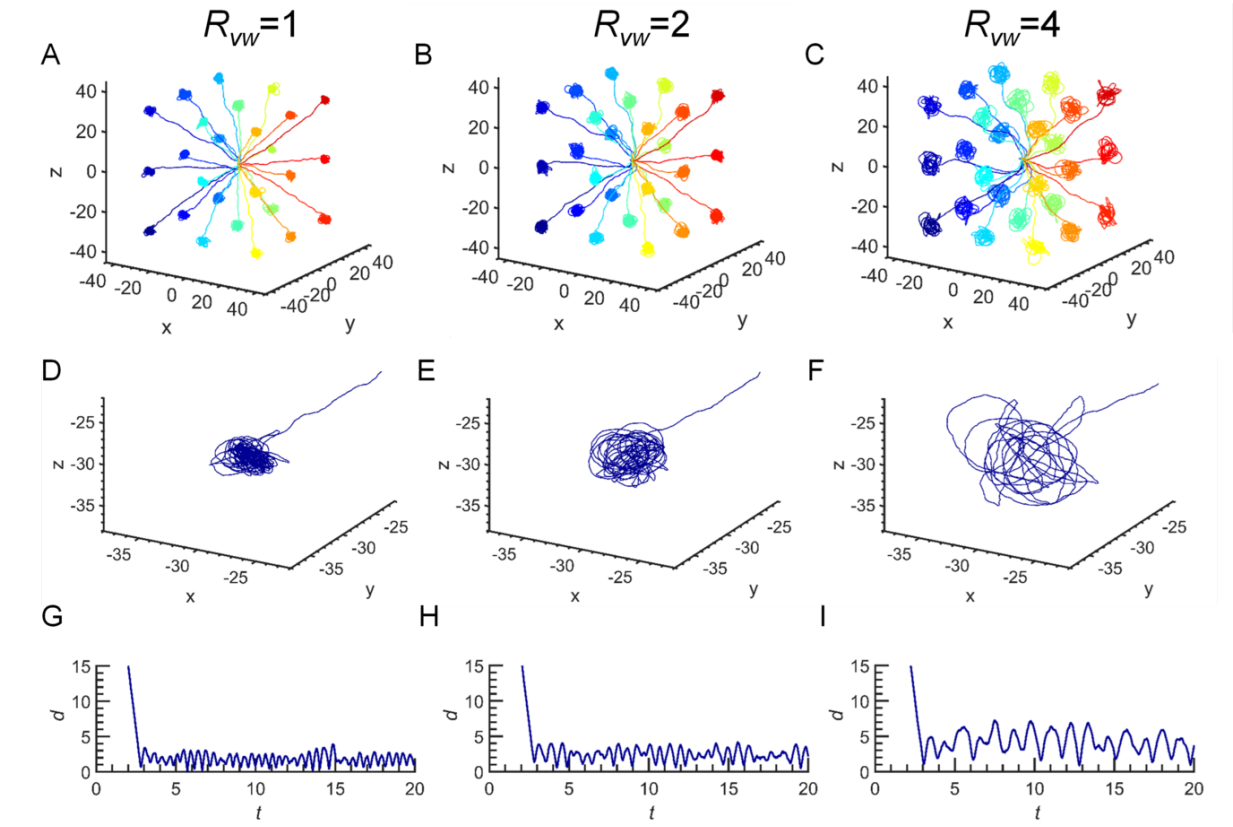

Figure 4: Fig. S4. Navigation and localization trajectories (200 control steps or 20 t) of microrobots with different propulsion rotation ratio $R_{vw} = v_{SP}/v_{max}$ with $R_{vw} = 1$ (A, D, G), $R_{vw} = 2$ (B, E, H), and $R_{vw} = 4$ (C, F, I). The targets are arranged like Fig. 3 in the main text. (A–C) Representative trajectories of robots navigating to different targets. (D–F) Representative trajectories of the microrobot around the target located at (-30, -30, -30); (G–I) The distance vs. time between the microrobot and the target located at (-30, -30, -30).

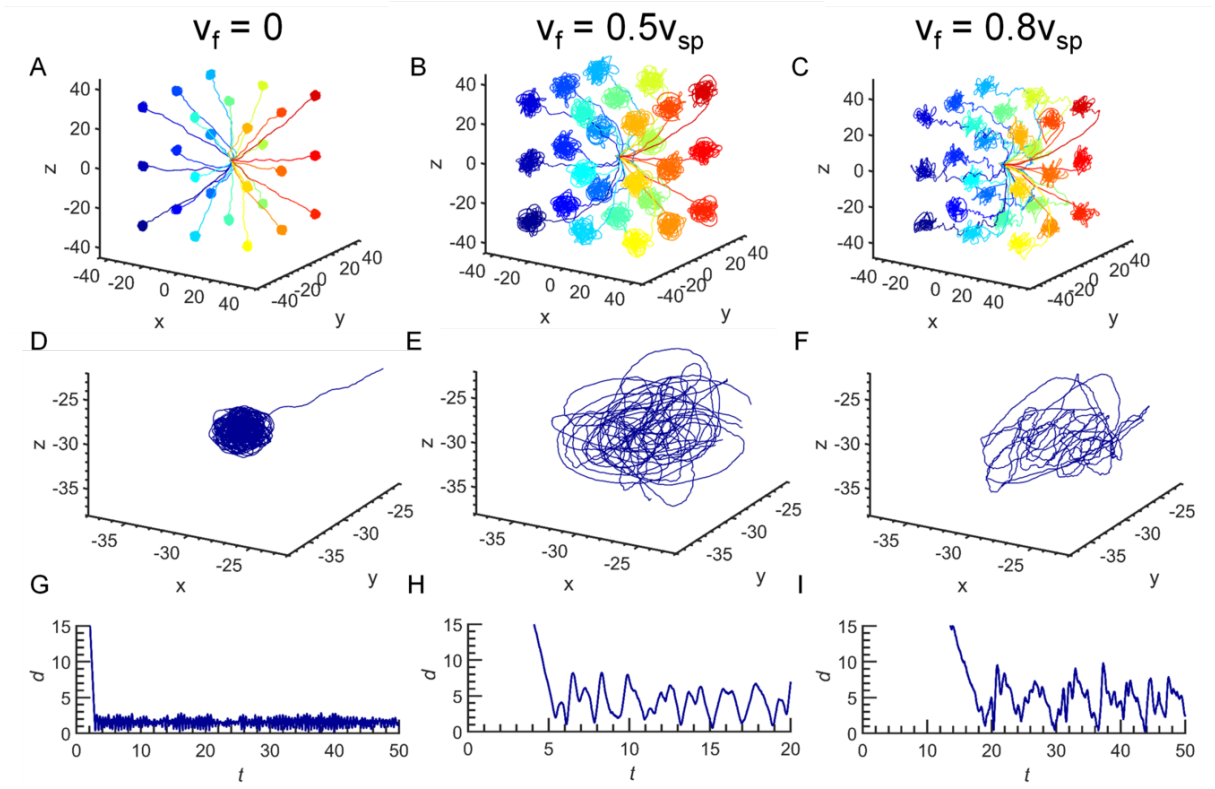

Figure 5: Fig. S5. Navigation and localization trajectories (500 control steps or 50 t) of microrobots under different external flow fields with $v_f = 0$ (A, D, G), $v_f = 0.5v_{SP}$ (B, E, H), and $v_f = 0.8v_{SP}$ (C, F, I). The targets are arranged like Fig. 3 in the main text. (A–C) Representative trajectories of robots navigating to different targets. (D–F) Representative trajectories of the microrobot around the target located at (-30, -30, -30); (G–I) the distance vs. time between the microrobot and the target located at (-30, -30, -30).

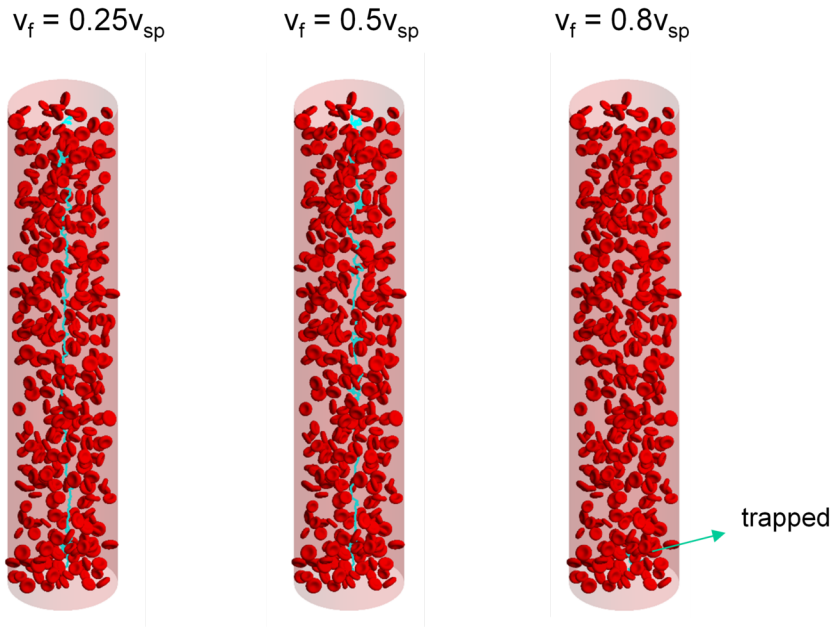

Figure 6: Fig. S6. Navigation trajectories of microrobots in a blood vessel with external flow field. At $v_f = 0.8v_{SP}$, microrobots frequently get trapped.

Supplementary Movie S1

Figure 7: Movie S1. Navigation and localization of microrobots in free space with different external flow fields.

Supplementary Movie S2

Figure 8: Movie S2. Navigation of microrobots in blood vessels with different vessel diameter D and RBC volume fraction f. Specifically, from left to right, D = 12um, f ˜ 10%; D = 25um, f˜10%; D = 50um, f˜5%; D = 50um, f˜10%.

Supplementary Movie S3

Figure 9: Movie S3. Navigation of microrobots in curved blood vessels with varying cross-section diameter.

Supplementary Movie S4

Figure 10: Movie S4. Exhaustive spatial survey in a blood vessel. From left to right, the blood vessel has zero RBCs, ˜5% RBCs, and ˜10% RBCs.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Proceedings of the Advances in Neural Information Processing Systems*, 2017.

Sudip Das, Shivraj D Deshmukh, and Rochish M Thaokar. Deformation of a biconcave-discoid capsule in extensional flow and electric field. *Journal of Fluid Mechanics*, 860:115–144, 2019.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning*, pages 482–495. PMLR, 2017.

Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems*, 2, 1989.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Sergey Loffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR, 2015.

Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the Proceedings of the 27th International Conference on Machine Learning*, 2010.