# Gathering Physical Particles with a Global Magnetic Field Using Reinforcement Learning

Matthias Konitzny[1], Yitong Lu[2], Julien Leclerc[2], Sándor P. Fekete[1], and Aaron T. Becker[1,2*]

***Abstract*— For biomedical applications in targeted therapy delivery and interventions, a large swarm of micro-scale particles ("agents") has to be moved through a maze-like environment ("vascular system") to a target region ("tumor"). Due to limited on-board capabilities, these agents cannot move autonomously; instead, they are controlled by an external global force that acts uniformly on all particles.**

**In this work, we demonstrate how to use a time-varying magnetic field to gather particles to a desired location. We use reinforcement learning to train networks to efficiently gather particles. Methods to overcome the simulation-to-reality gap are explained, and the trained networks are deployed on a set of mazes and goal locations. The hardware experiments demonstrate fast convergence, and robustness to both sensor and actuation noise. To encourage extensions and to serve as a benchmark for the reinforcement learning community, the code is available at Github.**

## I. Introduction

Delivering active substances to a specific location in an organism is a crucial challenge for a wide range of important problems, such as the treatment of cancer, localized infections, or internal bleeding. Typically, this requires dealing with navigation through complex, maze-like environments, such as a vascular system. However, the size of particles necessary for passage through these vessels prohibits sufficient individual energy storage or computational power. A promising alternative is offered by employing a global external force, e.g., an electromagnetic field,

This paper investigates a method using magnetism to gather dispersed micro-particles at a desired target location. Biomedical applications of collecting magnetic particles include delivering chemotherapy to tumors [1], building localized embolisms (i.e., forming an artificial clot) [2], thrombolysis (delivering medication to remove a clot) [3], local hyperthermia [4], and enhanced imaging [5]. In these applications, external electromagnetic coils are used to create a time-varying magnetic field across a workspace. This magnetic field then generates forces and/or torques on any ferrous or magnetic components in the workspace. Because the actuation is external, these components do not require motors, internal power sources, or computation. Instead, the components are often particles whose shapes, composition, and coatings are selected for the particular application.

This work was supported by the Alexander von Humboldt Foundation and National Science Foundation Grants No. IIS-2130793 and IIS-1553063.

[1]Authors are with the Department of Computer Science, TU Braunschweig, Germany.

[2]Authors are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA.

*Corresponding author. Email: atbecker@uh.edu

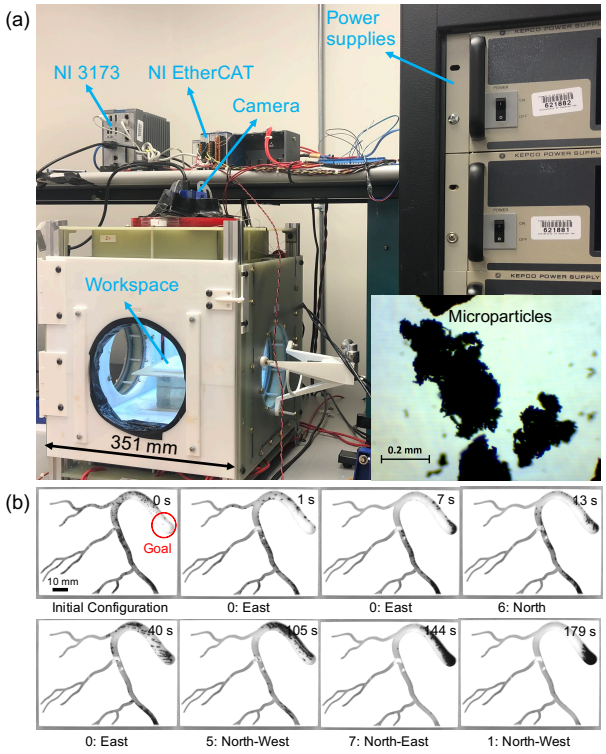


Fig. 1. **a**: Annotated photo of the magnetic manipulator used in this study. Inset: microscope photo of microparticles. **b**: Aggregating particles in the *Coronary* workspace. Underneath each picture, the next direction of the global force is shown. For further details, see the experimental section and the supplementary video.

Magnetic aggregation uses a small number of electromagnetic coils (usually 4 to 8) to control a much larger number of particles ($10^2$ to $10^9$), making this problem severely underactuated.

### A. Related Work

Current work on aggregation has focused on four methods.

*(1) Using large gradients to collect particles at a desired position:* Because magnetic monopoles do not exist, this approach is usually applied above or below a 2D workspace [6]. The same actuation is used in children's toys, in which the child uses a permanent magnet to drag iron particles into new configurations.

*(2) Pulling particles out of solution to form aggregates:* Luo et al. used spatially-varying magnetic fields to cause 1 µm diameter superparamagnetic particles in flowing fluid to aggregate [2]. They showed how this can generate embolisms in vascular phantoms and porcine tissue in areas with the magnetic field stronger than a critical value. Lowering the

magnetic field allowed the aggregates to fall apart and disperse. However, magnetic gradients decay more rapidly with distance than magnetic torque, much current work focuses on using the latter to aggregate and manipulate particles.

*(3) Exploiting fluidic vortices caused by rotating the magnetic field:* Wang et al. [4] investigate aggregating $Fe_3O_4$ microparticles for local hyperthermia treatment. They moved a clump of microparticles in water along the bottom of the container by using a 3D rotating magnetic field to induce fluidic vortices that clumped the particles, shrinking the swarm of particles to a third of the initial radius in 80 seconds. By changing the pitch angle of the rotation for movement, they generated movement speeds of up to $1.8\,\mu\mathrm{m/s}$. Similar approaches were used in [3], [5].

*(4) Using a rotating magnetic field to walk particles in a desired direction:* Yigit et al. [7] studied using a rotating global magnetic field (with a precessing axis) to move self-assembled chains of superparamagnetic microparticles (about $5\,\mu\mathrm{m}$ in diameter) in unison along the bottom of a container filled with water, with speeds of up to $10\,\mu\mathrm{m/s}$. They showed that obstacles could compress the swarm by around $30\,\%$.

Most of these experimental approaches emphasized either locally aggregating particles at a desired location, or moving a single aggregation along a desired path. This differs from our paper, which focuses on the *gathering* problem: bringing *all* particles in the workspace to the goal. Mahadev et al. [8] described an algorithm that delivers all particles in a grid environment with $n$ grid cells to a target in at most $O(n^3)$ actuator steps. This showed that delivery can always be achieved; however, a delivery time of this magnitude is usually impractical, and actually minimizing this time is an NP-hard problem, as shown by Becker et al. [9]. Heuristics based on RRTs were implemented in hardware to aggregate iron particles in [10], aggregating the majority of iron particles ($30\,\mu\mathrm{m}$ diameter) near a goal location in a $20\,\mathrm{mm}\times 20\,\mathrm{mm}$ maze in an average of $22.5\,\mathrm{min}$. The theory for aggregation in grid-environments was extended in [9] using an algorithmic strategy for gathering all particles with a worst-case guarantee of at most $O(kD^2)$ steps; here $k$ denotes the number of convex corners in the workspace and $D$ is the maximum distance between any two points in the workspace. Both $k$ and $D$ are usually much smaller than the number $n$ of grid locations, representing a large performance increase over [8]. In [9], the authors introduced a reinforcement learning technique for aggregating the particles, which outperformed the algorithmic approaches in simulation.

### B. Contributions

The primary contribution of this paper is to quickly and robustly gather particles that are spread in a 2D workspace at a desired location using camera feedback. This paper investigates methods using reinforcement learning (RL). Gathering has not been extensively studied in the RL community, so this work is also meant to serve as a benchmark for the field. To encourage reuse and expansion, the code is shared [11]. Previous reinforcement work [9] was performed only in simulation, using a grid world with no sensing or actuation noise, with particles that collapsed into a single particle whenever there was a collision. This paper uses a more accurate particle model. The training process is flexible and independent of the shape of the workspace. The resulting models can handle domain shift, including variations to particle motion, vision artifacts, and delays in timing.

Finally, the RL techniques were successfully tested on a hardware platform, demonstrating fast and robust performance under real-world conditions. There is a non-trivial simulation-to-reality gap that required augmenting our RL approach, and our paper documents the techniques used to overcome these challenges.

## II. Background

We use *Reinforcement Learning* (RL) to solve the gathering problem. This section provides a problem description and a definition of the RL terms used.

### A. Reinforcement Learning

RL is a field of unsupervised machine learning in which the goal is to train a neural network to learn specific behavior in an *environment* without any human interaction. Instead of learning from (labeled) training data, an RL agent learns by a sequence of interactions with an environment $\mathcal{E}$. In each *time step* $t$ of this interaction, the agent chooses an action $a_t$ from a set of valid *actions* $\mathcal{A}$, based on an observation $o_t$ of the environment and its policy $\pi(o_t)$. The agent then receives the next *observation* (e.g. an image), representing the state of the environment and a feedback signal called *reward* $r_t$ in return. This interaction between the agent and the environment is typically modeled as a large but finite Markov Decision Process (MDP), with each sequence of interaction defining a single state.

Using reward as feedback, the goal of the training process is to estimate an optimal policy $\pi^*$ to maximize the expected future return $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_t$, with future rewards discounted by a factor $\gamma$. Being in a state $s$ and selecting action $a$, we define the value of the next state as the action-value function $Q(s,a)$. If the value of $Q(s,a')$ for each action $a' \in \mathcal{A}$ was known in advance, we could always choose an action that maximizes the expected reward. In modern reinforcement learning, a neural network with weights $\theta$ is used as a function approximator. This allows using a policy that greedily selects the next action as $a = \max_a Q_\theta(s,a)$. Many modern RL architectures like A2C [12] additionally make direct estimates about the best action.

### B. Problem Description

In this work, we consider gathering in a 2-dimensional grid environment (called the *workspace*) with free and blocked integer positions (called *pixels*). The workspace contains a set of particles which can be controlled by a global input which uniformly moves all particles in a specific direction (north, north-east, east, south-east, south, south-west, west, north-west). A particle will move if it is not blocked by a stationary obstacle or other particles. This makes it possible to change the shape of the particle swarm: Given a specific

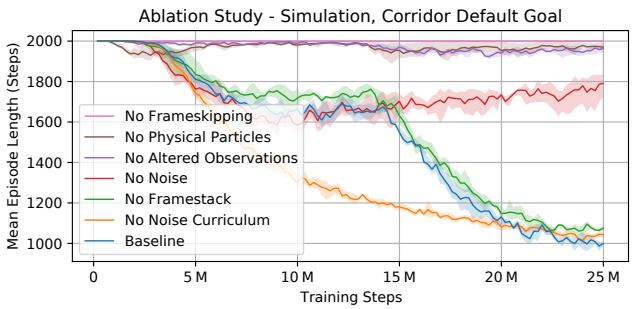


Fig. 2. Ablation study analysis on the *Corridor* environment showing performance during training. The performance is measured as the average episode length until all particles are gathered at the goal position. Measurements are done during training at an interval of 200k steps and use data from 64 episodes with randomly distributed particles. Standard deviation from different trials is indicated by lighter areas. The data is smoothed with an exponential moving average with $\alpha = 0.5$.

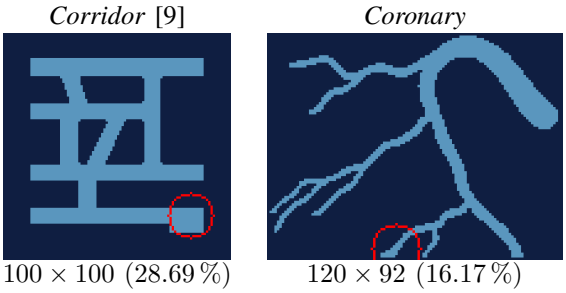


Fig. 3. Default test workspaces with size and, in parenthesis, percentage of non-blocked area. The target position is marked with a red circle.

*workspace*, a set of particle positions, and a target position, our objective is to find a control sequence that gathers in minimum time all particles at the target position.

To train agents without requiring large numbers of hardware experiments, we need a simulator which behaves similar to the real-world problem. Otherwise agents will not perform due to the large domain shift. The domain shift mainly occurs in two areas: particle behavior and observations. For the experiments we specifically augmented our simulation to reduce domain shift in both areas (see Section III-A and Section III-C for details).

## III. Implementation

To train the RL agent and test different configurations, we created a modular Python environment that is compatible with OpenAI Gym [13]. The code is on GitHub [11].

An agent trained with RL usually has trouble when transitioning from a simulated to a real-world environment. To reduce this effect, we specifically augment the simulation to mimic the behavior of physical particles and real-world observations.

To analyze the effect of the augmentations, we perform an ablation study. The results can be found in Fig. 3, which shows the performance of agents without certain components in the same simulated environment. To verify our assumptions, the impact of these changes is further analyzed in an ablation study performed on the hardware experiment, which shows similar effects (see Fig. 4 and Section V for details).

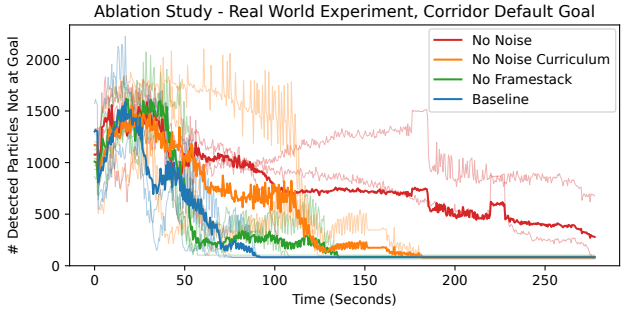


Fig. 4. Real-world experimental ablation study on the *Corridor* environment (default goal position) showing the performance of the final agent. The performance is measured as the average time to gather all particles at the goal position. Each experiment has been repeated three times; thick lines show the average value while the lighter lines show individual experiments.

### A. Particle Model

Previous work [8], [9] used a simplified algorithmic particle model where particles get directly moved. Also, particles collapsed into a single particle, if they entered the same integer position. Real particles behave differently from this simplified model. They have individual weights and exhibit second order dynamics that include momentum and gradual acceleration. Particles experience increased friction near walls. When particles collide with each other they do not fuse together and can split apart later. Particles also often build up in lumps which only slowly break up again.

To make the domain shift as small as possible, while also keeping a reasonable training speed, we implemented a basic physical properties model. Particles have a floating point position and an individual random weight. A uniform force is used to accelerate the particles instead of directly changing their positions. Particles also keep their velocity and particle-particle collision is modelled if more than three particles enter the same integer position. Since particles have different weights they may split up again after being merged into the same position.

### B. Rewards

In this work, we present a simple yet efficient reward system which gives the agent valuable feedback without the need to handcraft milestones for each individual workspace. The reward is directly calculated from the change in two basic metrics: the sum of distances of all particles to the target position and the maximum distance of any particle to the target position. Both metrics are calculated with respect to the integer grid position of the particles. To balance the reward produced by these metrics, we use a normalization which approximately scales both metrics to the interval $[0, 1]$. The normalization for the sum of distances is the average distance of any free pixel multiplied by the number of particles, and for the maximum distance we use the distance of the particle-containing free pixel that is the farthest away from the target.

We also use a time penalty to encourage the agent to bring the particles to the target as fast as possible. We found that using the same time penalty for every workspace can be a problem, as larger workspaces also require more steps for solving. Because other metrics are bound to the interval $[0, 1]$, the proportion between time penalty and other metrics would change depending on the minimal episode length. We therefore estimate a time penalty to be also approximately in the interval of $[0, 1]$ independent of the workspace by

$$\text{episode length} \approx a \cdot d_{\text{max}} \cdot \log(d_{\text{avg}} \cdot k), \tag{1}$$

where $d_{\text{max}}$ is the maximum distance between any two free pixels of the workspace and $d_{\text{avg}}$ is the average distance between any pixel and the target position. The parameter $k$ is the number of convex corners of the current workspace and we set $a$ to $0.75$ for all experiments.

We also experimented with other reward components, including reward for decreasing the number of unique particles, curiosity reward [14], or goal-based rewards, as well as other normalization terms, but found that the simple continuous system yielded better results. For more details, see [15].

### C. Observations

Observations are generated as images $x_t \in \mathbb{R}^d$, with $d$ being the number of pixels in the workspace. This vector is binary and encodes the current integer positions of the particles (1 if at least one particle is present in the pixel, 0 otherwise). RL agents are able to learn the goal position from the reward signal and the shape of the environment from the absence of particles in blocked pixels.

During the hardware experiments, we track the position of the particles with a camera. Particles are extracted from the image using thresholding. The camera resolution exceeds the resolution of the simulated workspace, and real-world particles may be smaller than simulated particles. As the camera is imperfect, the particle extraction includes some false-positive and false-negative detections. The camera image is cropped to the region of the workspace, but this cropping process results in slight image shifts, or size changes.

To harden the agent to these effects, we specifically alter the observations from the simulated environment to mimic real-world errors during training. This includes the addition of static dirt (i.e. particles which will never move), Gaussian noise, slight image shifting and cropping, particle detection errors, and artifacts from image downscaling and thresholding.

Finally, since particles may have a certain velocity, it is harder to make predictions based on a single still image. We therefore apply *frame-stacking*, and provide the agent with a combination of the last two observations. This ensures the agent receives information about the current trajectory of the particles without requiring a recurrent network architecture.

### D. Agent Architecture and Training Details

All our agents are trained using the *Proximal Policy Optimization* algorithm (PPO) [16]. To ensure easy comparison, we used the implementation of *stable-baselines3* [17]. We train on 128 parallel environments to stabilize the training process. During training, the learning rate is scheduled to decrease linearly from $8.5 \times 10^{-5}$ to $5.0 \times 10^{-5}$ over the first $4\,\%$ of training. The learning rate decreases to $2.0 \times 10^{-5}$ after $20\,\%$ and to $5.0 \times 10^{-6}$ after $95\,\%$ of training. The discount factor $\gamma$ is set to $0.99$, the GAE parameter $\lambda$ to $0.95$ and the clipping $\epsilon$ to $0.1$. We train on large batches of size 2048 samples and perform 16 epochs of optimization.

The neural network consists of a small convolutional feature encoder with three layers with 32 ($8 \times 8$, $s = 2$), 64 ($4 \times 4$, $s = 2$) and 64 ($4 \times 4$, $s = 1$) filters respectively, followed by a single fully connected layer with $512$ neurons. Both feature extractor and the fully connected layer are shared between the actor and the critic. After each layer we apply the scaled exponential linear unit activation function.

To slightly reduce the impact of altered observations on the overall training time, we decided to make use of curriculum learning [20] and introduce noise gradually during training: The amount of noise is linearly increased from zero after half of the training is finished and reaches its maximum value at $90\,\%$ of the training time. This allows the agent to learn basic behavior, before learning how to deal with noisy inputs.

Observations are preprocessed before being fed into the network by a pipeline commonly used to train agents for Atari games [18]. With frame-skipping, agents experience the environment at lower frame-rate by repeating actions for $k$ consecutive steps. We use $k = 4$ for all simulated experiments and $k = 6$ for all agents trained for the hardware experiments. To reduce the computational complexity, observation are scaled down to $84 \times 84$ pixels. Finally, observations get normalized by $x_t \to x_t/255$ to the interval $[0, 1]$.

During training, collected rewards get normalized by a running mean. This prevents discouraging all actions in episodes which generate mostly negative reward. These situations often occur during earlier stages of the training process, when the negative time-penalty dominates the returned reward. Additionally, episodes longer than 2000 steps are truncated. This ensures that agents will not get stuck in suboptimal environment states. Each training episode generated 100 to 350 particles with uniformly random positions. For the *Corridor* environment we trained the agent for 25 M steps, while we used 140 M steps for the *Coronary* environment.

## IV. Hardware Setup

The experimental platform used ferrous particles actuated by a magnetic manipulator. Ferromagnetic microparticles were scattered into an air-filled plastic workspace. An external magnetic system generates a flux density of $3\,\mathrm{mT}$ that actuates the microparticles. When the magnetic field is applied, the microparticles self-assemble into small clumps. A rotating magnetic field is used to roll these assemblies to gather at the goal position.

### A. Experiment Setup

*1) Magnetic manipulator:* These experiments used the lab-built magnetic manipulator system shown in Fig. 1. Figure 5a shows the block diagram of the hardware. The

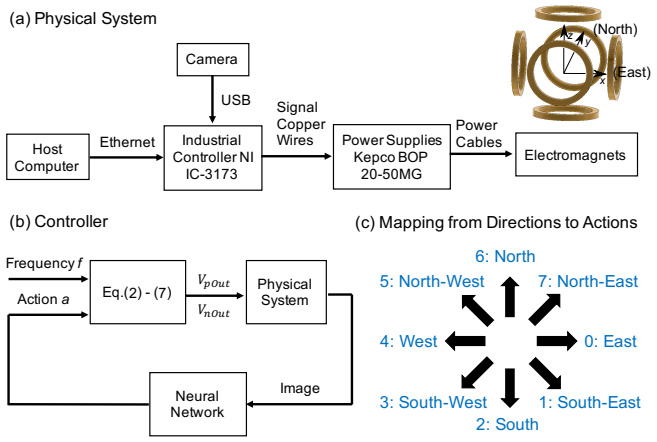


Fig. 5. Block diagram of the overall system. **a**: Physical system. **b**: Controller module. **c**: Mapping from directions to actions. Inset: Reference frame linked to the electromagnets.

experimental apparatus was extensively presented in [19], so it is only described briefly in the present paper. The system contains six electromagnets (EMs) with internal radii of $180\,\mathrm{mm}$ and external radii of $215\,\mathrm{mm}$. The EMs are arranged in a cube shape and separated by a distance of $300\,\mathrm{mm}$. A total of twelve Kepco BOP 20-50MG power supplies are used to power the electromagnets. Each power supply can generate $50\,\mathrm{V}$ and $20\,\mathrm{A}$. Each electromagnet is powered by a set of two power supplies connected in series. The magnetic manipulator system's large working distance and significant magnetic field strength enable the use of a larger workspace and provide enough torque to actuate a larger amount of microparticles than in [10].

The system's hardware also includes a host computer, an industrial controller NI IC-3173, and a Basler acA800 camera (see Fig. 5a). The industrial controller executes a LabVIEW program to actuate the microparticles with an external rotating magnetic field. The power supplies are controlled via an external analog signal generated by the industrial controller and are connected to the EMs with power cables. The camera views the workspace from the top and allows monitoring the microparticles.

Figure 5b shows the controller module. The industrial controller acquires images and shares them every $0.5\,\mathrm{s}$ with the host computer. A Python program executed on the host computer loads the images and uses them as input for the neural network. This program calculates the action $a$ to take and sends this information back to LabVIEW via TCP/IP. The calculation of the rotating magnetic field is discussed in Section IV-B. The mapping from direction to actions is shown in Fig. 5c.

*2) Workspaces design:* Two workspaces were used to experimentally validate the simulation results: the *Corridor* workspace and the *Coronary* workspace. Each workspace is made of two layers of acrylic cut using a Universal Laser Cutter. The base layer is $3\,\mathrm{mm}$ thick and the upper layer is $6\,\mathrm{mm}$ thick, with obstacle-free areas cut out by the laser cutter. These layers are bonded by WELD-ON 4 Acrylic Adhesive. The *Corridor* workspace is a $100\times100\,\mathrm{mm}^2$ square workspace with $8\,\mathrm{mm}$ wide channels. This maze is a $5\times$ scaled-up version of the experimental platform demonstrated in [10]. The *Coronary* workspace is a $100\times76\,\mathrm{mm}^2$ workspace. The smallest channel width is $1.5\,\mathrm{mm}$ and the largest channel width is $7\,\mathrm{mm}$. The microparticles used in this study are $Fe_3O_4$ iron filings. The size of microparticles is not consistent and ranges from $150$ to $700\,\mu\mathrm{m}$ in length (see Fig. 1a). For the experiments we used 148 mg of microparticles. If we assume the particles are spheres with an average radius of 0.1 mm (see Fig. 1), there are approximately 4484 particles.

### B. Calculation of the Rotating Magnetic Field

This subsection describes the calculation of the voltage to apply to the EMs to produce a rotating magnetic field in the commanded direction (see Fig. 5c) The voltage applied on an EM is proportional to the time derivative of the flux density minus the voltage drop created by Joule effect. This information is used to calculate the voltage that will produce a field rotating around the $y$ axis (action 0: East) using the following equation:

$$\begin{bmatrix} V_{xn} \\ V_{zp} \\ V_{xp} \\ V_{zn} \\ V_{yp} \\ V_{yn} \end{bmatrix} = V_0 \cdot \begin{bmatrix} \sin(\omega t) \\ \sin(\omega t - \pi/2) \\ \sin(\omega t - \pi) \\ \sin(\omega t - 3\pi/2) \\ 0 \\ 0 \end{bmatrix}, \tag{2}$$

$$\boldsymbol{V_p} = \begin{bmatrix} V_{xp} & V_{yp} & V_{zp} \end{bmatrix}^T, \tag{3}$$

$$\boldsymbol{V_n} = \begin{bmatrix} V_{xn} & V_{yn} & V_{zn} \end{bmatrix}^T. \tag{4}$$

Here $V_0$ is the maximum voltage applied to the EMs (100 V in our case) and $\omega = 2\pi f$, where $f$ is the frequency of the rotation of the field which can be adjusted manually. A rotation matrix $\boldsymbol{R}$ is then used to calculate the voltages that will produce a field rotating in the commanded direction:

$$\boldsymbol{R} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5}$$

$$\boldsymbol{V_{pOut}} = \boldsymbol{R} \cdot \boldsymbol{V_p}, \text{ for } [x^+, y^+, z^+] \text{ coils}, \tag{6}$$

$$\boldsymbol{V_{nOut}} = \boldsymbol{R} \cdot \boldsymbol{V_n}, \text{ for } [x^-, y^-, z^-] \text{ coils}. \tag{7}$$

The angle $\alpha$ is calculated using $45° \cdot a$, where $a$ is the action computed by the neural network. The reference frame linked to the electromagnets is shown in Fig. 5.

The current inside each electromagnet can be calculated using $I = V/Z$, where $Z = R + j\omega L$ is the complex impedance of the load. In this equation, $j$ represents the complex operator ($j^2 = -1$), $R$ is the electrical resistance and $L$ is the inductance. The experiment was performed at a relatively low frequency ($4\,\mathrm{Hz}$). The impedance of the circuit is low at these frequencies and, as a result, the current is large enough to cause the electromagnets to overheat after a few minutes of use. Thermocouple temperature sensors glued on the EMs allow monitoring the temperature of the electromagnets. To prevent hardware damage, the experiments were

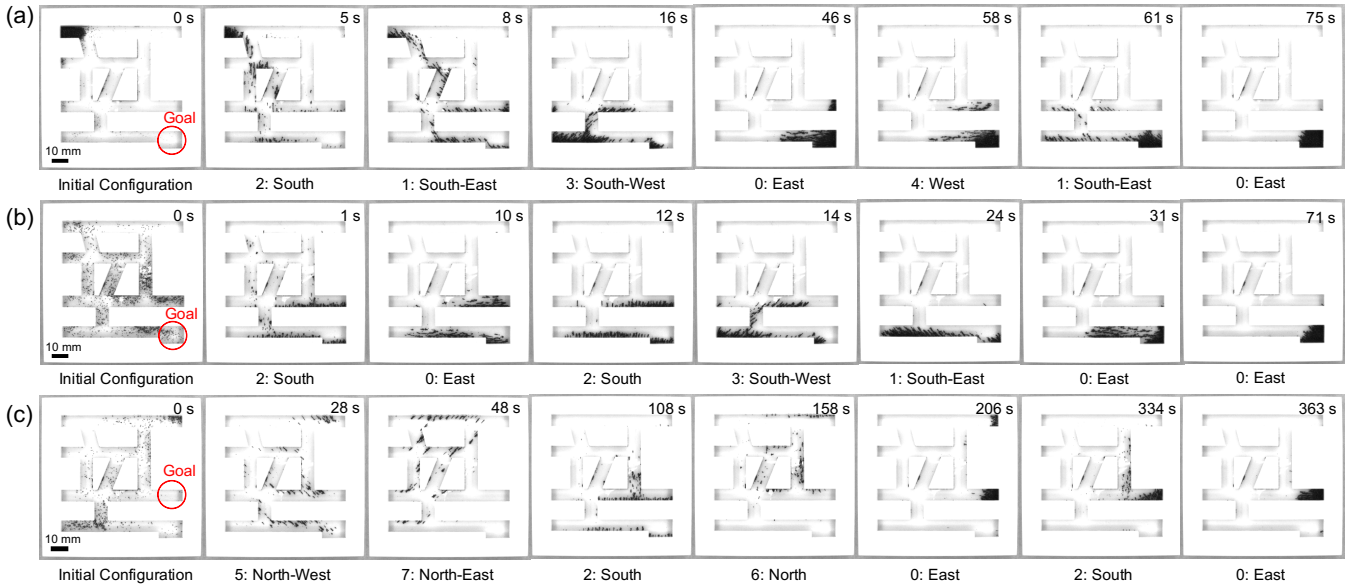


Fig. 6. Experimental demonstration of moving particles to goal positions in the *Corridor* workspace. **a** - **b**: Moving a clump of particles from the top left corner and aggregating a spread apart swarm to a goal position. **c**: Aggregating a spread apart swarm to another goal position. Underneath each picture, the next direction of the global force calculated from the neural network is shown. See https://youtu.be/CyVGlIrOFEA for videos of these demonstrations.

paused when the temperature of an EM reached $80\,^{\circ}\mathrm{C}$, and then resumed when the coils cooled down to $40\,^{\circ}\mathrm{C}$.

## V. Experiment Results

A series of experiments were conducted to validate the simulation results. Figure 6 shows the experiment results in the *Corridor* workspace. In addition, refer to Fig. 1b for aggregating particles in the *Coronary* workspace. The frequencies used in these experiments are $4\,\mathrm{Hz}$. While other frequencies can be used as well, using a frequency of $4\,\mathrm{Hz}$ closely matches our simulation which has a fixed force applied to the particles at each step. We therefore chose to use a frequency of $4\,\mathrm{Hz}$ for all workspaces and goal positions. Figure 6a shows an experimental result of moving a clump of microparticles from the top left corner to the bottom right goal position (default goal position) in 75 seconds. Figure 6b shows collecting a dispersed group of microparticles to the default goal position in 71 seconds. Plots showing the number of detected particles gathered and not gathered over time for these experiments is shown in Fig. 7a and b. Particle positions are directly extracted from the camera image by thresholding. Because the particles clump and disperse, the number of detected particles is not constant. We demonstrated the robustness of our system by repeating experiments of aggregating a spread apart swarm to the default goal position 10 times, as shown in Fig. $7\mathrm{b}_{10}$.

Figure 6c shows collecting a dispersed group of microparticles into a different goal corridor on the right side. This goal configuration is considerably harder than the previous goals, and requires 363 seconds of control. The RL agents for this experiment also required more training time and used $140\,\mathrm{M}$ training steps. Figure 7c plots the gathering process. Figure 7d shows gathering on the *Coronary* workspace shown in Fig. 1, which requires 179 seconds of control. Flat areas of the plots in Fig. 7c and d occur when the magnetic manipulator overheated and was turned off. Removing these cooling periods provides the control times shown in Fig. 6.

Figure 4 shows the real world experimental ablation study analysis on the *Corridor* workspace (default goal position). We ran the experiments three times for each model. The baseline model performs the fastest, and can reach the goal position in the *Corridor* environment in $97\,\mathrm{s}$ on average. The other models each remove an element from the baseline. Under the No Framestack and No Noise Curriculum models, the particles can reach the goal position but require more time. However, the No Noise model sometimes cannot move all particles to the goal position, which matches the simulation results shown in Fig. 3.

## VI. Conclusion

This paper advanced methods for gathering particles in a 2D maze using reinforcement learning agents to plan gathering sequences and an external time-varying magnetic field for particle control. We presented a new approach that robustly and efficiently gathers particles in a hardware platform. The gathering time required was reduced from tens of minutes to tens of seconds.

The RL approach might allow calculating the difficulty of gathering particles at each position in the workspace. Such information could be useful for higher-level motion planners that use gathering as a primitive. Biological systems are complex, and most would require larger workspaces than presented here. Since training time increases super-linearly with workspace size, there is room for improvement. There are many exciting avenues for future work, including applying these methods to 3D workspaces, using bio-compatible sensing methods such as CT or ultrasound, handling disturbances such as blood flow, and improving training time for new mazes and new goals.

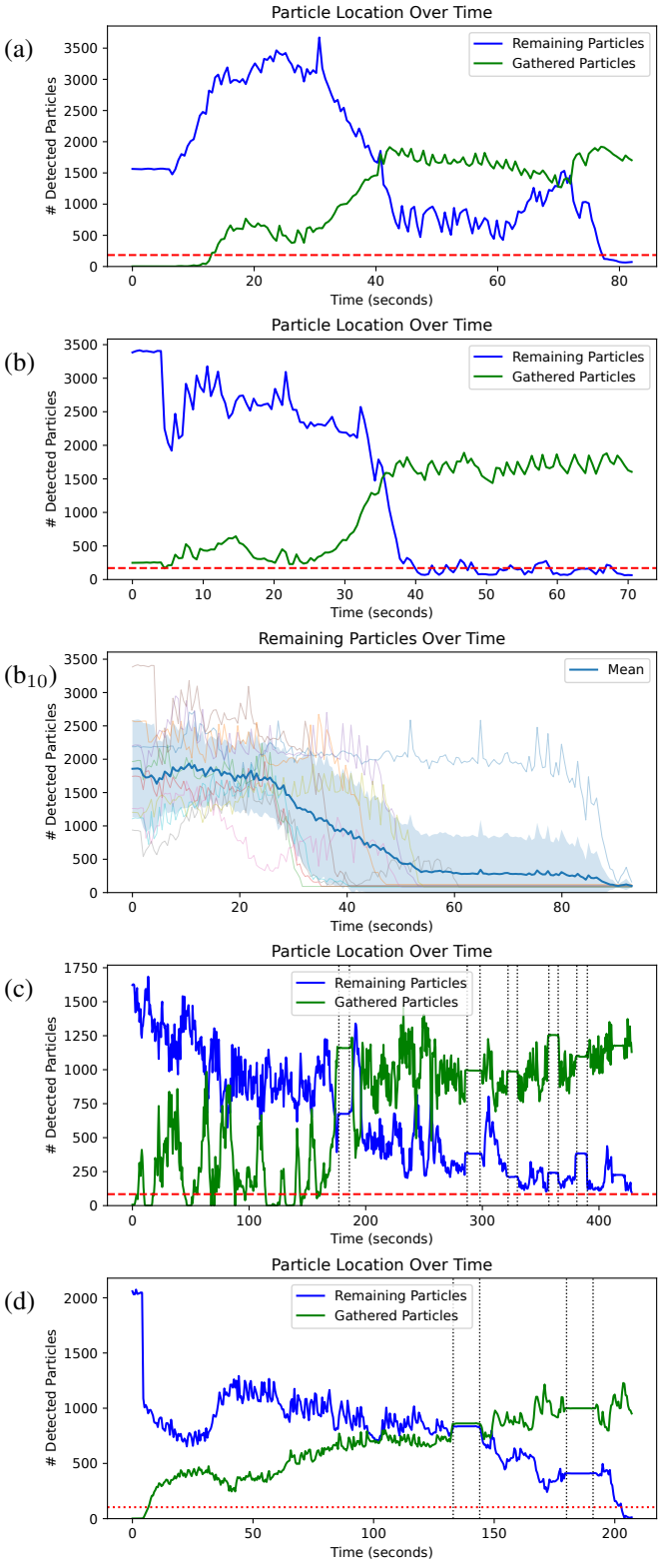


Fig. 7. Number of particles gathered over time for 13 hardware experiments. Red dashed lines are at 5 % of the maximum number of particles detected. (a), (b), and (c) plot data from the experiments shown in Fig. 6 on the *Corridor* workspace. (d) shows gathering on the *Coronary* workspace shown in Fig. 1. ($b_{10}$) shows 10 applications of (b). For ($b_{10}$) experiments were stopped if less than 5 % of the maximum particles were remaining. The shaded region is $\pm 1$ standard deviation. Dotted vertical lines show when control was paused.

## References


[1] 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," *Int. J. Rob. Res.*, vol. 33, no. 3, pp. 359–374, 2014.

[2] M. Luo, J. Law, X. Wang, L. Xin, G. Shan, M. Badiwala, X. Huang, and Y. Sun, "Robotic swarm control for precise and on-demand embolization," in *Int. Conf. Rob. Autom. (ICRA)*. IEEE, 2020, pp. 4470–4476.

[3] Q. Wang, D. Jin, B. Wang, N. Xia, H. Ko, B. Y. M. Ip, T. W. H. Leung, S. C. H. Yu, and L. Zhang, "Reconfigurable magnetic microswarm for accelerating tpa-mediated thrombolysis under ultrasound imaging," *IEEE/ASME Transactions on Mechatronics*, pp. 1–1, 2021.

[4] B. Wang, K. F. Chan, J. Yu, Q. Wang, L. Yang, P. W. Y. Chiu, and L. Zhang, "Reconfigurable swarms of ferromagnetic colloids for enhanced local hyperthermia," *Advanced Functional Materials*, vol. 28, no. 25, p. 1705701, 2018.

[5] J. Yu, Q. Wang, M. Li, C. Liu, L. Wang, T. Xu, and L. Zhang, "Characterizing nanoparticle swarms with tuneable concentrations for enhanced imaging contrast," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2942–2949, 2019.

[6] S. Martel and M. Mohammadi, "A robotic micro-assembly process inspired by the construction of the ancient pyramids and relying on several thousand flagellated bacteria acting as micro-workers," in *Int. Conf. Intell. Rob. Syst. (IROS)*, 2009, pp. 426–427.

[7] B. Yigit, Y. Alapan, and M. Sitti, "Programmable collective behavior in dynamically self-assembled mobile microrobotic swarms," *Advanced Science*, vol. 6, no. 6, p. 1801837, 2019.

[8] 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 *Int. Conf. Autom. Sci. Engin. (CASE)*. IEEE, 2016, pp. 1231–1236.

[9] A. T. Becker, S. P. Fekete, L. Huang, P. Keldenich, L. Kleist, D. Krupke, C. Rieck, and A. Schmidt, "Targeted drug delivery: algorithmic methods for collecting a swarm of particles with uniform, external forces," in *Int. Conf. Rob. Autom. (ICRA)*. IEEE, 2020, pp. 2508–2514.

[10] L. Huang, L. Rogowski, M. J. Kim, and A. T. Becker, "Path planning and aggregation for a microrobot swarm in vascular networks using a global input," in *Int. Conf. Intell. Rob. Syst. (IROS)*. IEEE, 2017, pp. 414–420.

[11] M. Konitzny, "Gathering environment GitHub repository," https://github.com/NeoExtended/gym-gathering, 2022.

[12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[13] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016.

[14] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, "Large-scale study of curiosity-driven learning," *arXiv preprint arXiv:1808.04355*, 2018.

[15] M. Konitzny, "Reinforcement learning for navigating particle swarms by global force," Master's thesis, TU Braunschweig, 07 2020. [Online]. Available: https://github.com/NeoExtended/rl-gathering-thesis/blob/master/masterthesis.pdf

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[17] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[18] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, and M. Bowling, "Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents," *J Artif Intell Res*, vol. 61, pp. 523–562, 2018.

[19] J. Leclerc, B. Isichei, and A. T. Becker, "A magnetic manipulator cooled with liquid nitrogen," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4367–4374, 2018.

[20] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.