Smart Magnetic Microrobots Learn to Swim with Deep Reinforcement Learning

Michael R. Behrens and Warren C. Ruder*

Swimming microrobots are increasingly developed with complex materials and dynamic shapes and are expected to operate in complex environments in which the system dynamics are difficult to model and positional control of the microrobot is not straightforward to achieve. Deep reinforcement learning is a promising method of autonomously developing robust controllers for creating smart microrobots, which can adapt their behavior to operate in uncharacterized environments without the need to model the system dynamics. This article reports the development of a smart helical magnetic hydrogel microrobot that uses the soft actor critic reinforcement learning algorithm to autonomously derive a control policy which allows the microrobot to swim through an uncharacterized biomimetic fluidic environment under control of a time-varying magnetic field generated from a three-axis array of electromagnets. The reinforcement learning agent learns successful control policies from both state vector input and raw images, and the control policies learned by the agent recapitulate the behavior of rationally designed controllers based on physical models of helical swimming microrobots. Deep reinforcement learning applied to microrobot control is likely to significantly expand the capabilities of the next generation of microrobots.

1. Introduction

Untethered swimming microrobotic systems have received significant research attention for performing micromanipulation tasks and particularly for their potential therapeutic biomedical applications.[1,2] Microrobots operating remotely inside the human body have potential to enable minimally invasive medical procedures including targeted drug, cell, or other cargo delivery,[3–6] tissue biopsy,[7] thermotherapy,[8] and blood clot removal.[9] Controlling these miniature devices within complex and dynamic environments such as the human body can present a significant engineering challenge.[10] This challenge is in part because the design of microrobotic systems is trending toward the use of complex composite materials,[6,11–14] dynamic morphologies,[15–18] and integrated biological components.[19–23] These features add layers of functionality to microrobotic systems, but can create difficulties when constructing accurate dynamic and kinematic models of microrobotic behavior, making it especially complex and challenging to use classical feedback control systems to control microrobot behavior.[5,15,24] Additionally, the environmental dynamics encountered by a biomedical microrobot inside the human body may be variable, complex, and poorly characterized.[17,25,26]

As one potential pathway to overcome these challenges, we can observe and adopt the strategies of natural biological agents that have evolved to operate in complex, unpredictable environments. Many biological systems can adapt to learn new behaviors based on experience, allowing them to thrive in a wide range of complex and variable environmental conditions by tailoring their behavior to suit the environment. Systems capable of learning adaptive behavioral patterns based on past events are ubiquitous in nature and are found across all levels of biological hierarchy, including in biochemical networks,[27] bacteria,[27,28] nematode worms,[29] insects,[30] plants,[31] adaptive immune systems,[32] and animal behavior.[33] Inspired by the wide-ranging applicability of adaption and learning to the success of living organisms, engineered microrobotic systems that learn new behaviors from past experience could enable new capabilities for complex microrobotic systems.[14]

Reinforcement learning (RL) is a biomimetic machine learning optimization technique inspired by the adaptive behavior of real-world organisms[35] that can enable learning behaviors in artificial engineered systems.[36] In RL, an agent observes the state of an environment, and chooses actions to perform in the environment to achieve a task specified by a reward signal, which is typically predefined. The reward signal is used to teach the agent to perform actions to maximize the expected future rewards, which enables the agent to learn to perform the task...
better based on past experience. RL algorithms have achieved success in a range of complex robotic control applications. For example, RL for robotic control has been demonstrated to create robust control policies that achieve better performance than many humans at complex tasks such as grasping and accurate throwing of irregularly shaped objects into bins. RL algorithms have also been shown to exceed human-level performance in complex virtual tasks with large possible state spaces that cannot be tractably and exhaustively modeled, such as the game of Go. Machine learning techniques including RL have already demonstrated promise for developing policies to control micro-robot behavior. In simulation, RL agents have been trained to control microrobot behavior for solving navigation and swimming challenges in heterogeneous fluids. An early RL algorithm, Q-learning, has also been shown to be effective for controlling the behavior of laser-driven microparticles in a discretized grid environment. Other similar machine learning techniques such as Bayesian optimization have been demonstrated to learn walking gates for difficult-to-model magnetoelastomeric millirobots. However, control of swimming microrobots that use deep RL to operate in dynamic, biomimetic, and microfluidic environments with clinically relevant magnetic actuation has yet to be reported.

In this work, we demonstrate that deep RL based on the soft actor critic (SAC) algorithm can be used to create soft smart helical magnetic microrobots that autonomously learn optimized swimming behaviors when actuated with nonuniform, nonlinear, and time-varying magnetic fields in a physical fluid environment. Our RL microrobots learned successful actuation policies both from state variable input and directly from raw images without any a priori knowledge about the dynamics of the microrobot, the electromagnetic actuator, or the environment (Figure 1a). Our RL agent discovered multiple successful actuation strategies in separate learning experiments, and the control policies learned by the agent all recapitulated the behavior of theoretically optimal physics-based approaches for actuating helical magnetic microrobots. These results demonstrate the potential of RL for developing high-performance multiple input, multiple output (MIMO) controllers for microrobots without the need for explicit system modeling. This capability to autonomously learn model-free microrobot control algorithms could significantly reduce the time and resources required to develop high-performance microrobotic systems.

2. Results

In order to create an environment where we could test the hypothesized efficacy of RL control systems for microrobots, we first designed and built a physical, biomimetic, fluidic arena with multidimensional magnetic actuation, and deployed a magnetic microrobot in the arena whose design was inspired by the work of Kumar and co-workers. In our experimental setup (Figure 1b), a helical agar magnetic robot (HAMR) was tasked with swimming clockwise through a fluid-filled lumen in a PDMS arena under control of a nonuniform rotating magnetic field generated by a three-axis array of electromagnetic coils (Magneturret). An overhead camera was used to track the position of the HAMR in the channel. The camera was used to pass images to a control algorithm consisting of an image processing module and a neural network which generated commands for the Magneturret (Figure 1c). The goal of the control system for our remotely actuated microrobot was to manipulate the shape and magnitude of the actuating energy field in order to move the microrobot to achieve an intended dynamic behavior. The controller neural network was trained via a RL agent using the SAC algorithm.

The fundamental control problem was encapsulated by this question: how should the currents in the electromagnetic coils be modulated in order to create a magnetic field that places forces and torques on the HAMR sufficient to drive its locomotion toward a specific target? Achieving this task usually requires an accurate dynamic model of the complete system, including the dynamics of the robot, the environment, and the actuator. Significant work has been done developing dynamic and kinematic models for different microrobots and actuators. These models are often developed by making simplifying assumptions about the system such as uniform magnetization, ideal shape, and system linearity, which could lead to behavioral deviations between the physical system and the modeled system. The difficulty in accurately modeling the dynamics of microrobot behavior increases significantly for microrobots with complex magnetization profiles, soft material composition, or active shape-changing capabilities.

Instead of explicitly modeling the dynamics of the magnetic actuator and the HAMR within the environment and specifying a controller, we performed the much simpler task of specifying the desired behavior of the HAMR in the form of a reward signal. The agent observed the state of the environment along with a reward signal containing information about actions that lead toward the successful completion of the task. The RL agent started without any a priori information about the task and had to learn to perform the task by sampling actions from the space of all possible actions and learning which actions resulted in behavior that was rewarded.

The RL controller required us to develop and formulate the task as well as the associated reward signal. At the beginning of each training episode, a target position was defined. 20° clockwise from the starting position of the HAMR in the circular channel. The objective of the RL agent was to develop an action policy, π, which maximized the total value of the rewards it would receive if it followed that policy in the future. When the environment was in state, s, the agent chose an action, a, from the policy according to $a \sim \pi(s)$, probabilistically selecting from a distribution of possible actions available in that state. The agent received a reward when it selected actions that moved the HAMR clockwise through the circular lumen toward the target, and it received a negative reward when it moved the HAMR counterclockwise. If the HAMR reached the target within the allotted time, the agent was given a large bonus reward, and the target position was immediately advanced 20° clockwise, starting a new episode. The reward function we selected was

$$r(s, a) = -\Delta \theta_s + (1000 \text{ if } \Delta \theta_s = \theta_g \pm 3^\circ)$$

where $\theta_s$ is the angular position of the HAMR in the channel in degrees, $\theta_g$ is the angular position of the goal, and $\Delta \theta_s$ is the change in angular position of the HAMR as a result taking of...
Within RL, proper reward selection can have a large effect on the performance of the system. While we did not attempt to extensively optimize our reward function to maximize the learning speed of the agent, we designed this reward to encourage the agent to reach the goal as quickly as possible. This two-part reward function encourages actions that move the HAMR a large distance in the correct direction with each action (optimizing velocity) and direct the agent to end on the target position to receive the bonus reward (providing a terminal condition to end an episode). It is important to note that the agent received a positive reward when the HAMR moved clockwise during an action (which corresponds to a negative change in $\theta_r$ using standard mathematical angle notation). The agent received the additional bonus reward for steps in which the position of the robot $\theta_r$ was within $3^\circ$ of the goal position $\theta_G$.

The RL problem was formalized as a Markov decision process, in which at time $t$, the state of the system $s_t$ is observed by the agent. The agent performs an action $a_t$ which changes the state of the environment to $s_{t+1}$, yielding a reward $r_t(s_t, a_t)$. This process continues for the duration of the task, yielding a trajectory of the form $(s_0, a_1, r_1, s_1, a_2, r_2, s_2, \ldots)$. The goal of the RL agent is to identify an optimal policy $\pi^*(a|s)$ for selecting actions, based on state observations, that maximize the rewards received for following the policy. Over the course of training, the agent autonomously learned a control policy by trying actions in the environment, observing the reward obtained by performing those actions, and modifying its future behavior in order to maximize the expected future return.

The control problem for our microrobotic system was formulated as an episodic, discrete time problem with a continuous action $a$ in state $s$. Within RL, proper reward selection can have a large effect on the performance of the system. While we did not attempt to extensively optimize our reward function to maximize the learning speed of the agent, we designed this reward to encourage the agent to reach the goal as quickly as possible. This two-part reward function encourages actions that move the HAMR a large distance in the correct direction with each action (optimizing velocity) and direct the agent to end on the target position to receive the bonus reward (providing a terminal condition to end an episode). It is important to note that the agent received a positive reward when the HAMR moved clockwise during an action (which corresponds to a negative change in $\theta_r$ using standard mathematical angle notation). The agent received the additional bonus reward for steps in which the position of the robot $\theta_r$ was within $3^\circ$ of the goal position $\theta_G$. The RL problem was formalized as a Markov decision process, in which at time $t$, the state of the system $s_t$ is observed by the agent. The agent performs an action $a_t$ which changes the state of the environment to $s_{t+1}$, yielding a reward $r_t(s_t, a_t)$. This process continues for the duration of the task, yielding a trajectory of the form $(s_0, a_1, r_1, s_1, a_2, r_2, s_2, \ldots)$. The goal of the RL agent is to identify an optimal policy $\pi^*(a|s)$ for selecting actions, based on state observations, that maximize the rewards received for following the policy. Over the course of training, the agent autonomously learned a control policy by trying actions in the environment, observing the reward obtained by performing those actions, and modifying its future behavior in order to maximize the expected future return.
action space and continuous state space. The state space consisted of all the possible states for the system: the position of the HAMR within the channel, the speed of the HAMR, the shape of the magnetic fields, the time remaining in the episode, and relative position of the robot to the target position in the channel. The action space consisted of four continuous actions, which controlled the magnitudes and phase angles for sinusoidal currents in the Magneturret. While the current waveforms could theoretically take on an infinite number of shapes, we chose to define the applied waveforms as sinusoids to bound the space of possible actions that the agent could take. Sinusoidal currents were chosen because these can be used to generate rotating magnetic fields in other three-axis electromagnetic actuators for microrobots, such as Helmholtz coils.\(^1\) We formulated the problem as episodic, with a time limit and a goal for the robot: to reach the goal position as quickly as possible. While RL tasks can be framed as episodic (with terminal states) or continuous,\(^5\) we chose to represent the task episodically to recreate the conditions we expect would be present in plausible use cases for microrobots. We chose to include a goal position to represent targets that a microrobot moving in the context of a biomedical application might attempt to reach, such as a blood clot, lesion, or tumor. In our current implementation, when the HAMR reached the goal, this represented the terminal state of the task and a new episode was started with a new goal position. By creating a virtual moving target for the HAMR during training, we were freed from the having to manually reset the state of the system each time the target was reached, which facilitated an automated training process.

### 2.1. Entropy-Regularized Deep RL Enabled Continuous Microrobot Control

We selected the SAC RL algorithm for this research. SAC is a maximum entropy RL algorithm that seeks to balance the expected future rewards with the information entropy of the policy.\(^1\) In other words, SAC learns a policy that successfully completes the task while acting as randomly as possible, which in practice often leads to robust policies that are tolerant of perturbations in environmental conditions.\(^3\) SAC had previously proven useful for real-world robotic tasks with high-dimensional, continuous state and action spaces,\(^2\) suggesting that it would be applicable to our microrobotic control problem. In previously reported applications of real-world RL with physical systems,\(^2\) SAC was demonstrated to be highly sample-efficient, requiring relatively few environmental interactions in order to develop a successful policy. Sample efficiency is critical when performing RL with real-world robotics (i.e., not simulated) in order to reduce wear and tear on the system, and in order to minimize the time needed to learn a policy.\(^1\)

The SAC algorithm seeks to develop an optimal stochastic policy \(\pi^*\)

\[
\pi^* = \arg\max_{\pi} \mathbb{E}_{(s, a) \sim \rho} [r(s, a) + \alpha \mathcal{H}(\pi(\cdot|s))] 
\]

where \(\mathcal{H}\) is the information entropy of the policy and \(\alpha\) is a temperature hyperparameter, which balances the relative impact of the policy entropy against the expected future rewards. Here, we used a version of the SAC algorithm in which the temperature is automatically tuned via gradient descent so that the entropy of the policy continually matches a target entropy, \(\overline{\mathcal{H}}\), which we selected to be -4 (-4 × Dim of the actions space) using a heuristic suggested by Haarnoja et al.\(^3\) A full derivation of the SAC algorithm is beyond the scope of this article, but interested readers are directed to Haarnoja et al.\(^3\) Briefly, the SAC algorithm uses an agent called the actor, denoted as \(\pi\), which is a deep neural network that takes the state of the system \(s\) as input, and returns action \(a\) as output. A value function is created to rate the value of taking actions when in particular states, and instantiated using two critic neural networks \(Q_1, Q_2\) which take states and actions as input, and return values corresponding to the relative value of taking action \(a\) in state \(s\). Two Q networks are trained in order to reduce overestimation in the value function. Environmental transitions in the form of \((s, a, r, s', d)\) sets are recorded in an experience replay buffer, \(D\), where \(d\) is a done flag denoting a terminal state, set either when the microrobot has reached the goal, or when the episode has timed out. The SAC algorithm learns off-policy by randomly sampling minibatches of past experiences from \(D\), and performing stochastic gradient descent over the minibatch in order to minimize loss functions for the actor network, \(\pi\), critic networks, \(Q_1\) and \(Q_2\), and temperature parameter, \(\alpha\). Over the course of learning, the parameters of the actor and critic neural networks are updated so that the behavior of the policy approaches the optimum policy, \(\pi^*\). A detailed version of the SAC algorithm for microrobot control that we used in this study is available in Algorithm 1. Neural network architectures and hyperparameters used are available in Table S1, Supporting Information.

### 2.2. Hardware to Control Magnetic Microrobots with RL

Magnetic fields created by electromagnetic coils are commonly used actuators for magnetic microrobots and have significant potential for clinical medical applications.\(^3\) Magnetic fields act by imparting forces and torques on the robot. For a microrobot with a magnetic moment, \(m\), in a magnetic field, \(B\), the robot experiences a force \(F\) according to

\[
F = \nabla (m \cdot B) 
\]

which acts to align the magnetic moment of the microrobot with the direction of the magnetic field. When the magnetic field is rotated so that the direction of \(B\) is constantly changing, it is possible to use this torque to impart spin to the microrobot at the frequency of the rotating magnetic field, up to the step out frequency of the robot.\(^4\) If the spinning microrobot is helically shaped, rotation can be transduced into forward motion so that the microrobot swims similar to how bacteria are propelled by flagella.\(^5\) This nonreciprocal helical swimming is efficient in low Reynolds number fluidic environments commonly...
encountered by microrobots. Because of the efficiency of this swimming mode, and because the magnetic torque available to a microrobot decreases more gradually with distance compared to the force, magnetic microrobots designed for long range magnetic operation are often helically shaped. For this reason, we selected a helical magnetic microrobot, the HAMR, as our model system.

The HAMR that we created for this study was composed of a 2% w/v iron agar hydrogel, which was uniformly diffused with 10% w/v iron oxide nanopowder to form a magnetically susceptible soft polymer. This magnetic agar solution was heated to melting temperature and a syringe was used to inject the liquid into a helical mold created using a stereolithography 3D printer. The agar in the mold solidified and the robots were removed with a metal needle and stored long-term in deionized (DI) water. The HAMRs molded for this study were 4.4 mm in length, 1 mm in diameter, and asymmetrical from head to tail, with a flat head and a pointed tail (Figure 2c,d).

Because the HAMRs swim with nonreciprocal, helical motion in the presence of a rotating magnetic field, a very common motif in microrobotic research, insights gained from this study could readily be extended to other microrobotic systems with similar characteristics.

Because the HAMRs were made of soft hydrogel, they were flexible and deformable. Soft-bodied microrobots have many favorable characteristics for in vivo use such as deformability to fit through irregular shaped channels and enhanced biocompatibility (e.g., by matching the elastic modulus of the robot to the biological environment). Previous studies investigating the material properties of agar hydrogels suggest that our HAMRs should have an elastic modulus in the range of ≈400 kPa which makes our HAMR softer than many of the biological tissues that a microrobot would interact with in vivo, such as artery tissue. However, the forces and torques experienced by the HAMR in this study did not lead to significant observable deformation, and the HAMR retained its helical shape throughout the training. Based on the relative rigidity of the HAMR, we expect that the swimming dynamics of the HAMR will behave analogously to a rigid helical swimmer with a similar geometry. Soft-bodied microrobots are appealing for biomedical applications, but it can be more difficult to create accurate dynamic models for soft-bodied microrobots. Our method of using RL to develop control systems without explicit modeling could be particularly useful for soft microrobots due to this modeling constraint, although the increased complexity of a highly deformable or kinematically complex microrobot would be likely to significantly increase the time required to train an RL control system. Finally, despite being soft-bodied, the hydrogel structure of the HAMR did not experience noticeable wear over the course of several months of continuous use, thus meeting a practical RL constraint that the system not be susceptible to significant wear and tear during extended use which would cause a distribution shift in the collected data as the dynamic properties of the system degraded.

As an actuator for our microrobot system, we developed a three-axis magnetic coil actuator—the Magneturret—which

Algorithm 1. Soft Actor Critic for Microrobot Control.

1: Initialize policy parameters $\sigma$, Q-function parameters $\omega_1, \omega_2$, and empty FIFO replay buffer $D$
2: Set target Q-function parameters equal to main parameters $\omega_{\text{target},n} \leftarrow \omega_n$
3: Initialize $\mathcal{P} = \text{number of actions (4), } n = 1$
4: Observe initial state $s_{\text{init}}$, and calculate $\theta_{\text{robot}} \in (0, 360^\circ)$
5: Set $\theta_{\text{goal}} = \theta_{\text{robot}} + 20^\circ, \theta_{\text{goal}} \in (0, 360^\circ)$
6: Data Collection Process: Repeat
7: \hspace{1em} If new $x_n$ is available: update
8: \hspace{2em} While t steps <33 or done = false
9: \hspace{3em} Select action $a_t \sim x_n$; \{t\}
10: \hspace{3em} Execute $a_t$ in the environment
11: \hspace{3em} For $j$ in range (3)
12: \hspace{4em} Wait 0.3 seconds
13: \hspace{4em} Observe next state $s_{t+1}$, reward $r_j(s_{t+1})$, and done
14: \hspace{4em} $d(1$ if $\theta_{\text{robot}} = \theta_{\text{goal}}$ or $t = 33$ \text{Else} 0$)
15: End For
16: Update Q-functions using
17: \hspace{2em} $\nabla_{\sigma} \mathcal{V}_{1}(s_{t}, a_{t}) = r(1 - d)Q(s_{t+1}, a_{t}) - \alpha \log x_n(s_{t})$, $\bar{\sigma} \sim x_n(\cdot|s_{t})$
18: Update Policy using
19: \hspace{2em} $\nabla_{\theta_{\text{robot}}} \mathcal{V}_{1}(s_{t}, a_{t}) = -\alpha \log x_n(a_{t}|s_{t})$, $\bar{\theta}_{\text{robot}} \sim x_n(\cdot|s_{t})$
20: Update target Q-functions using
21: \hspace{2em} $\omega_{\text{target},n} \leftarrow \omega_{\text{target},n} + (1 - \tau)\omega_n$ for $n = 1, 2$
22: End If
23: \hspace{1em} Send latest $x_n$ to data collection process every minute
24: Until convergence
contained six permalloy-core magnetic coils arranged on the faces of a 3D-printed acrylonitrile butadiene styrene (ABS) plastic cube (Figure 2e). The two coils on opposite sides of the central cube along each axis were wired together in series so that they both contribute to the generation of a magnetic field along their respective axis. Each of the three coils, hereafter referred to as the X, Y, and Z coils, were driven with a sinusoidal current generated by a pulse width modulated (PWM) signal created by a microcontroller and amplified in an H-bridge motor driver. The resulting magnetic field, produced by the superposition of the magnetic fields from the three coils, could be modulated by varying the frequency, amplitude, and phase angle of the sine current waves in each coil. To cool the coils and prevent thermal damage, the Magneturret was sealed with epoxy resin into a 3D-printed housing and coolant was continuously pumped through while the coil was operating (Figure 2f). The RL agent was given direct control over the magnitude ($M_x$, $M_y$, $M_z$) and phase angles ($\phi_x$, $\phi_y$, $\phi_z$) of the sinusoidal driving currents in the X-axis coils ($M_x$, $\phi_x$) and Y-axis coils ($M_y$, $\phi_y$) of the Magneturret, for a total of four continuously variable actions: $M_x$, $\phi_x$, $M_y$, and $\phi_y$ (Table 1). The Z-axis magnitude was calculated as the larger of the two magnitudes in $X$ and $Y$, and the Z-axis phase angle was fixed. The sinusoidal currents in each axis used a fixed angular frequency of 100 rad s$^{-1}$ (15.9 Hz).

Figure 2. Hardware for real-world control of magnetic microrobots using RL. a) HAMR schematic. b) HAMRs were fabricated by molding molten magnetic hydrogel in 3D printed molds. Scale bar = 10 mm. c) HAMR on the tip of a gloved human finger. Scale bar = 1 cm. d) HAMRs were composed of agar hydrogel infused with iron oxide nanopowder. Scale bar = 1 mm. e) The Magneturret was composed of six coils of copper magnet wire wrapped around permalloy cores, positioned on the faces of a central 3D printed cube. Scale bar = 20 mm. f) The Magneturret was enclosed within a 3D printed housing and sealed with epoxy, and glycerol coolant was continuously pumped through the Magneturret housing. Scale bar = 20 mm. g) A circular PDMS track with a rectangular cross section used as an arena for the HAMR. A black, circular fiducial marker indicates the center of the arena. Scale bar = 10 mm. h) The PDMS arena was submerged in a water-filled petri dish placed on top of the Magneturret, with an acrylic LED light sheet as a backlight for uniform bottom-up illumination. Scale bar = 30 mm. i) The complete hardware system.
Table 1. Control inputs for electromagnet waveforms.

| Control variable | Symbol | Source | Range |
|------------------|--------|--------|-------|
| Frequency        | \(f\)  | Fixed  | 15.9 Hz (\(\omega = 2\pi f = 100 \text{ rad s}^{-1}\)) |
| Magnitude X      | \(M_x\) | RL agent | \([-1,1]\) unitless |
| Magnitude Y      | \(M_y\) | RL agent | \([-1,1]\) unitless |
| Magnitude Z      | \(M_z\) | max(\(|M_x|,|M_y|\)) | \([0,1]\) unitless |
| Phase angle X    | \(\phi_x\) | RL agent | \([0,2\pi]\) radians |
| Phase angle Y    | \(\phi_y\) | RL agent | \([0,2\pi]\) radians |

Control equations:

\[
\begin{align*}
\text{Current in X-axis coil} & = I_x = M_x \sin(f t + \phi_x) \\
\text{Current in Y-axis coil} & = I_y = M_y \sin(f t + \phi_y) \\
\text{Current in Z-axis coil} & = I_z = M_z \sin(f t)
\end{align*}
\]

We chose to operate our HAMR in a circular, fluid-filled track for this study. This arena served as a simple environment which mimics the tortuous in vivo luminal environments that microrobots operating in the body might encounter while providing a simple environment for us to establish a robust proof-of-concept RL control system (Figure 2g). The HAMR could swim in a complete circle within this arena, and no human intervention was required to reset the position of the robot in the environment during training, which facilitated automated learning. The arena was constructed by pouring polydimethylsiloxane (PDMS) over a polyvinyl chloride ring with an outer diameter of 34 mm and a 1.7 mm \(\times\) 3 mm rectangular cross section. The PDMS was then cured, and plasma bonded to a second flat sheet of cured PDMS to form a rectangular lumen for the HAMR to swim. PDMS is transparent, allowing us to see the robot in the arena and to visually track it with an overhead camera. During long-term learning experiments, the PDMS arena was submerged in a petri dish filled with DI water in order to prevent the formation of air bubbles in the channel due to evaporation over the course of an experiment. This petri dish was then placed on top of the Magneturret, with the center of the Z-axis coil aligned with the center of the circular track (Figure 2h). A black rubber wafer was placed into the center of the arena on top of the PDMS to act as a fiducial marker so that the center of the arena could easily be identified with image processing. A diffuse white LED backlight was positioned between the Magneturret and the PDMS arena for uniform bottom-up illumination which facilitated simple image processing by binary thresholding to identify the position of the microrobot in the channel.

We did not perform an extensive analysis to identify the shape or magnitude of the magnetic field created by the Magneturret, or to model the swimming dynamics of the HAMR in the PDMS arena. We hypothesized that we would be able to develop a high-performance control system using RL without going through the effort of developing a system model first.

2.3. RL Can Be Used to Learn Microrobot Control Policies

RL systems have been demonstrated that can learn to achieve tasks from a wide range of state information sources. For this study, we evaluated the ability of our RL agent to learn control policies from either state vector-based inputs (Figure 3a) or raw images augmented with the goal position (Figure 3b). In state-based input mode, state information of the microrobotic system was derived by using image processing to create a state vector-based input, which was passed to the RL agent. The angular position, \(\theta_x\), of the microrobot in the channel was calculated with image processing by binary thresholding and simple morphological operations. The camera was deliberately run with a slow shutter speed so that the images were intentionally washed out to remove noise. This simplified the task of using binary thresholding operations to identify the position of the HAMR and the center of the channel. The angular position of the HAMR in the channel was measured relative to the fiducial marker in the center of the circular arena. This information, as well as the position of the goal, \(\theta_y\), the last action taken by the agent (\(M_{x,i-1}, M_{y,i-1}, \phi_{x,i-1}, \phi_{y,i-1}\)), and the time, \(t\), remaining in the episode were used to create a state vector. In the second input mode, we gave the agent observations in the form of raw pixel data from the camera. The images from the overhead camera were scaled down to 64 \(\times\) 64 pixels, and the images were augmented with a marker indicating the position of the goal \(\theta_y\) as a line radiating outward from the center of the circular track to the goal position in front of the HAMR. These images were then passed into a convolutional neural network, which is a deep neural network architecture that has been demonstrated to effectively learn to identify features in images for classification tasks and has been used to control robots with raw image input using RL.

RL is based on the mathematics of Markov decision processes, which theoretically require the full state of the system to be available to the agent in order for convergence to be guaranteed. In our particular implementation, the velocity of the HAMR at any given time could not be determined from a single still-frame observation, so the total state of the system given to the agent at each time step was composed of three concatenated subobservations taken 0.3 s apart, for a total step time of 0.9 s (see Algorithm 1, steps 11–15). This allowed the agent to infer the velocity of the HAMR based on differences between the three subobservations. This technique of batching sequential observations for improving the observability of the system for RL has been used successfully in domains such as Atari video games, in which the agent learned from raw pixel data gathered from sequential screenshots of the game.

At the beginning of each learning trial, the actor and critic neural network parameters were randomly initialized. We allowed the RL agent to train for a maximum of 100,000 steps, using a fixed ratio of one gradient update per environmental step, which has been shown to reduce training speed in exchange for higher training stability. At the beginning of training, the agent was typically observed to reach the target and receive the bonus reward within the allotted time approximately 25% of the time, based on random sequences of actions. 100,000 environment steps were adequate time for effective actuation policies to be learned that continuously moved the HAMR clockwise around the arena, both with state vector input and raw images (Figure 3c). It is commonly reported when using RL that several million environmental steps are necessary to derive a successful
policy, so this result shows the sample efficiency of SAC, which is critically important for RL tasks that are trained using physical systems instead of in simulation. A time-lapse movie of the HAMR recorded during the learning process is shown in Movie S1, Supporting Information. We tracked the net movement of the HAMR during the training process, and each training session ended with the microrobot going continuously around the track in a clockwise direction (Figure 3d). After 100,000 steps, the state vector-based policies achieved significantly higher overall level of performance, succeeding in approximately 90% of the episodes compared to the approximately 50% success rate for the image-based input. It is likely that the raw-image-based policies could have benefited from longer training periods.

Once the training sessions were complete, we evaluated the learned policies to test their performance. For evaluating policies, we used the highest performing policy parameters learned during a training session by monitoring a rolling average of the return over the last 100 episodes and saving the policy parameters each time the rolling average performance exceeded the last best performing model (Figure S1, Supporting Information). This was done because we sometimes observed a drop in performance after the peak performance was achieved in training, possibly due to overfitting. Early stopping, or selecting a policy before performance degradation has occurred, is a common technique used to prevent overfitting in neural networks. Successful policies were able to move the robot indefinitely around the complete circular track (Figure 3e, Movie S2, Supporting Information).

We recorded each action taken by the agent during training sessions and the resultant change in state of the microrobot.

Figure 3. RL yielded successful control policies for the HAMR within 100 k time steps. a) We used two input modes to train the RL agent. The first mode was to calculate a state vector consisting of the robot position, \( \theta_r \), the goal position, \( \theta_g \), the last action taken by the agent, \( M_x, M_y, \phi_x, \phi_y \), and the time remaining in the episode. Three sequential observations were taken 0.3 s apart, and combined as a single state, \( s \). b) The second input mode used images of the microrobot in the arena as the state. Three sequential images were then passed to a convolutional neural network as the state, \( s \). c) We trained the agent for a total of 100 k time steps per training session. The trace represents the average and standard deviation of the return from three successful training runs. d) Successful training resulted in policies which, after an initial learning period, achieved consistent clockwise motion, with the HAMR moving continuously around the circular arena. e) The RL agent learned policies that moved the microrobot around the full circular arena with helical swimming motion.

\[ \Delta \theta_r \]
in Figure 4 are indicated as positive for negative values of $\Delta\theta_r$, indicating travel in the clockwise direction. We separated the total 100 k steps into five bins of 20 k steps each. The distribution of actions over the first 20 k time steps (Figure 4a,i) is centered around a sharp peak of actions which result in no net movement, as we would expect from an agent with little experience randomly...
exploring the space of possible actions. By the second batch of 20 k steps (Figure 4a,ii), a pattern emerged in which the action distribution shifted to a bimodal distribution in which most actions still result in no net movement, but a second peak on the positive side indicates a trend toward selecting actions which result in clockwise movement. However, during this phase of training, the net motion of the robot remained close to zero (Figure 4b) because of flattening of the negative tail in the action distribution. As the learning process continued, the distribution continued to shift until the average movement was clockwise, with a second peak around 5° per time step, and a narrow tail representing few actions, which caused the robot to move in the counterclockwise direction (Figure 4a,v).

The SAC algorithm learns a continuous stochastic policy, \( \pi \), sampling actions from the policy according to \( a_t \sim \pi(\cdot|s_t) \), in which the actions selected during training are randomly sampled from a Gaussian distribution, and the agent learns the mean \( \mu \) and the variance of this distribution over the course of training. This is done in order to explore the space of possible actions during training. During training the agent seeks to balance the sum of future rewards with the information entropy of the policy by maximizing an entropy-regularized objective function, and the policy entropy corresponds to the explore/exploit tradeoff the agent makes during training. However, once the policies were trained, performance during policy evaluation could be increased by selecting actions from the mean of the distribution without further stochastic exploration according to \( a_t = \mu(s_t) \). This deterministic evaluation led to an increase in the proportion of actions taken by the agent which resulted in positive motion for both state-based (Figure 4c) and image-based agents (Figure 4d). We compared the total average velocity achieved by all the trained policies in both deterministic and stochastic action selection modes, which showed that deterministic action selection led to higher performance (Figure 4e). This is consistent with our expectation that the reward function we used would optimize the agent for velocity, and that removing the additional exploratory behavior of the stochastic agent would manifest as faster swimming.

We next examined the distribution of the action values chosen by the RL agent when evaluated deterministically according to \( a_t = \mu(s_t) \). For each of the four actions (\( M_x, M_y, \phi_x, \phi_y \)) taken by the policy over 3000 time steps, we plotted the value of the action against the position of the HAMR, \( \theta_r \) (Figure 4f). The plotted actions are color-coded according to \( \Delta \theta_r \), with red actions indicating clockwise forward motion and blue actions indicating retrograde motion. The majority of actions taken by each of the six policies during evaluation resulted in positive motion. Each of the three policies trained using state vector-based input followed similar patterns, in which the phase angle of the X coil was held constant, and the magnitude of the X coil varied according to the position of the motorrobot \( \theta_r \). The Y coil was controlled by actuating the phase angle as a function of position and holding the amplitude relatively constant. In contrast, the policies learned by the image-based agents were more heterogeneous, finding different possible ways to manipulate the four actions in order to produce forward motion. One consistent pattern across all learned policies is that the magnitudes tended to hold steady close to the maximum or minimum values of \(-1\) and positive \(1\), regardless of \( \theta_r \). This would result in the largest amplitude current sine waves, which we would expect because stronger magnetic fields would be able to create more powerful torques on the HAMR. The action distribution plotted in Figure 4f is plotted against the position of the HAMR, which was the component of the state vector that correlated most strongly with variations in the policy action distribution. For reference, the action distribution is plotted versus the remaining time in the episode in Figure S2, Supporting Information, plotted versus the previous action in Figure S3, Supporting Information, and plotted versus the velocity in Figure S4, Supporting Information.

2.4. Optimizing the RL-Trained Policies via Regression

We observed that the microrobot control policies learned by the RL agent sometimes performed actions that were obviously non-optimal (resulting in counterclockwise motion). We could likely further increase the performance of the learned policies by using more complex techniques like hyperparameter tuning and longer training times. However, by observing the behavior of the RL agent, we hypothesized that if we could distill the policies learned by the RL agent into mathematical functions of the state variables, we might achieve a higher level of performance (Figure 5a).

To test this, we chose one of the state-based policies and one of the image-based policies, and fit regression models to the data in order to create continuous control signals as a function of the robot position \( \theta_r \). First, we examined policy 1, learned by the state vector-based agent (Figure 4f). This policy was acting by modulating the magnitude in the X coil in what approximated a square wave pattern and modulating the phase angle in the Y coil in what appeared closer to a sine wave. The other two actions were held approximately constant regardless of the position of the robot. From all 3000 actions taken during policy evaluation, we selected the subset of actions taken by policy which had resulted in a positive movement of at least 3°, discarding the lower performing actions for this analysis. We then fit sinusoidal regression models to the \( M_x \) and \( \phi_x \) action distributions, and also fit a square wave to \( M_x \) (Figure 5b). The resulting policies are shown in Figure 5b as solid black lines superimposed over the action distribution. The sine wave policy (Figure 5c) and the mixed sine/square wave policy (Figure 5d) that we developed with the regression models were then used to control the HAMR. The image-based policy that we evaluated was significantly more complex than the state-based policy (Figure 5e). We modeled this policy mathematically by fitting a 20th order polynomial to the data, and used this polynomial policy to drive the HAMR (Figure 5f).

The sinusoidal policy achieved the highest level of performance (Figure 5g), achieving the highest average HAMR velocity of all policies tested in this study, while the square/sine policy performed slightly worse than the neural network policy on which it was based. Despite the complexity of the polynomial policy compared to the sine and square wave-based policies, the performance of this policy was approximately equal to that of the sine/square wave policy, and was superior to the neural network-based policy upon which it was based. As these mathematical policies only use \( \theta_r \) as the input, it is possible that we could further increase the performance of mathematically inferred policies by taking other parts of the state vector into account.
2.5. Control Policies Learned by the RL Agent Recapitulate the Behavior of Optimal Policies Based on Physical Models

Finally, we wanted to know whether the policies learned by the RL agent were modulating the magnetic field in a way that matched the control systems that human researchers have developed based on physical models. When using a uniform rotating magnetic field to steer a helical microrobot, the rotating magnetic field is usually made to rotate about the helical axis of the microrobot, which is also the direction in which the microrobot will swim (Figure 6a). Therefore, to drive a helical microrobot around a circular track like the one we used, we would expect that the direction of the rotating magnetic field would be tangent to the circular track at all points along the circle for the optimal policy.

Although we did not record the magnetic field at all points along the track during policy evaluation, we can approximate the behavior of the magnetic fields based on the actions taken by the agent. The recorded actions selected by the policy while driving the HAMR around the track were used by us to estimate the magnetic fields produced during the action. To do this, we constructed a 3D vector \( B = [B_x, B_y, B_z] \), where \( B_n = M_n \sin(\theta t + \phi_n) \), where the magnitude and phase angle were selected by the policy. The actions taken by the agent are run for a total of 0.9 s during each time step, during which time the actions are held constant and \( B \) rotates as a function of time with an angular frequency of 100 rad s\(^{-1}\). Taking the cross product \( B_{\perp} = B(t) \times B(t+1) \) results in a vector \( B_{\perp} \) which points in the direction perpendicular to the plane of the rotating magnetic field described by \( B \). By calculating the azimuthal angle \( \theta_{B_{\perp}} = \arctan(B_{2y}/B_{1x}) \), we can approximate the direction of the rotating magnetic field during an action taken by the policy. The results of this analysis are shown by plotting an arrow with direction \( \theta_{B_{\perp}} \) at the point \( \theta_t \) along the circular track for each action taken by the policy. Results are shown for the inferred mathematical policies (Figure 6b), the image-trained policies (Figure 6c), and the state trained policies (Figure 6d). Based on this mathematical approximation of the magnetic field.

Figure 5. Control policies learned by the RL agent could be translated into continuous functions in order to increase performance. a) The policies learned by the RL agent were used as a basis to derive higher-performance policies via regression of the positive action distribution. b) For one of the policies that the RL agent learned, the actions were plotted as a function of \( \theta_t \) and mathematical functions were fit to the subset of actions, which yielded HAMR velocities greater than 3° per step. Sine waves and square waves were fit to the data via regression (shown in black), and those mathematical function were used to control the HAMR for 1000 time steps. c) The results of running the sinusoidal policy and d) the sine/square policy. e) The policy learned by the image-based agent was fit with a 20th degree polynomial function. f) The polynomial policy was used to drive the HAMR for 1000 time steps. g) Comparing the average velocity of the HAMR when controlled by each policy.
field direction, each policy learned by the RL agent, regardless of input type, tended to create a rotating magnetic field nearly perpendicular to the direction of travel of the microrobot. This recapitulates the behavior of the theoretical optimal policy that we would expect based on a physical analysis of helical swimming magnetic microrobots. This data also sheds light on the reason that the sine function was the highest performing policy that we tested because the sine function most closely approximates the theoretical optimal policy in this analysis.
3. Discussion

Here, we have reported the development of a closed-loop control system for magnetic helical microrobots, which was implemented using RL to discover control policies without the need for any dynamic system modeling. Continuous control policies for high-dimensional action spaces were represented by deep neural networks for effective control of magnetic fields to actuate a helical microrobot within a fluid-filled lumen. Multiple high-dimensional state representations including state vector and raw images were sufficient to represent the state of the microrobot and yield successfully trained policies. Compared with previously reported control systems for magnetic microrobots, we believe that the system we have presented possesses a number of key advantages. Electromagnetic actuation systems for microrobots are either air core, such as Helmholtz coils and Maxwell coils, or contain soft magnetic materials in the core which enhance the strength of the generated magnetic field, but can lead to nonlinearities when summing the combined effect of fields from multiple coils with saturated magnetic cores. Such nonlinearities make modeling the behavior of the system more difficult, particularly when the coils are run with high enough power to magnetically saturate the core material. Additionally, when controlling microrobots with permanent magnets, those magnets are often modeled as dipole sources for simplicity, and the actual behavior of the physical system may not match the idealized model behavior. Neural network-based controllers trained with RL learn control policies from observing the actual behavior of the physical system, and deep neural networks can accurately model nonlinear functions.

Control policies learned with RL will automatically take into account the real system dynamics, and this model-free control approach can greatly simplify the job of the microrobotic engineer. Significant work has been done to create accurate dynamic models of rigid helical microrobots, and many microrobots are relatively straightforward to control using these models and classical control systems. However, many recently developed microrobotic systems are composed of soft, shape-changing materials, which are inherently harder to model than rigid bodies. Soft microrobots such as helical grippers, which can kinematically change their configuration during operation, might also be difficult to accurately model.

Here, we have shown that our algorithm was able to control a soft helical microrobot without any dynamic modeling on the part of the control system designers. RL-based microrobot control could enhance both the capabilities of novel microrobot designs and increase the efficiency of researchers by allowing the RL agent to do the work of developing a high-performance controller. RL-based controllers may be able to exceed the performance of classical control systems based on simplified models (e.g., linearized models) because the RL agent is able to learn based on the observed physical behavior of the system, and deep neural networks are capable of accurately modeling any observed nonlinearities that the microrobotic system might exhibit.

Our choice to use a model-free RL algorithm trained from scratch on a physical robotic system does entail potential downsides that are worth noting. In many robotic applications, training physical robots with RL is impractical due to the constraints imposed by the physical system or the task, particularly when safety is critical and exploration is costly. This could make it difficult to amass a sufficient quantity of training data to train a high-performance system. One potential approach to scale up real-world learning time is to multiplex robot training with many concurrently learning robots performing the same task.

Highly complex microrobots, which exhibit significant kinematic complexity and deformability, could be infeasible to train purely with model-free approaches because of the additional time it would require to train the system to fully explore the state space. For many complex robotic systems, transfer learning has been shown to be effective, in which a simulation of the physical system is used to amass a large quantity of training data in silico, and then the final control system is fine-tuned with training on the physical system. There will likely continue to be a tradeoff between the difficulty and utility of constructing a useful system model for in silico training, and the practicality of collecting large amounts of real-world training data.

Other control strategies that have been successfully applied for microrobot control could be enhanced with RL. Algorithms which have been used to control soft microrobots such as force–current mapping with PID control and path planning algorithms could potentially be combined with RL in order to optimize the gains in the PID controllers, and adapt to changes in environmental conditions by a process of continuous learning, or to optimize for multiple variables. Force–current mapping algorithms used to control microrobots are also often created with assumptions of linearity in magnetic field superposition, which could be violated with soft magnetic cores in the driving coils, a limitation that could be potentially overcome by the nonlinear function approximation capabilities of deep neural networks. Another potential combination of RL with classical control has been demonstrated by Zeng et al., who used a residual physics model in order to fine tune a physics-based model of a grasping and throwing problem for a robotic arm. Using such methods, imprecise models of microrobot dynamics could be improved by fine-tuning their parameters based on a data-driven RL approach, leading to increased performance.

We have demonstrated that it is possible to learn microrobot control policies with RL based on no prior knowledge, and then fine-tune the performance of the policy by fitting continuous mathematical functions to the learned policy behaviors. While the sine and sine/square policies that we created based on analyzing the learned policies might have been deduced by a first-principles analysis of the problem, this is certainly not the case for the polynomial policy we derived from the image-based input policy, which does not map to our intuitions of how to actuate a magnetic helical microrobot. We believe that this result provides strong support for the idea that a deep neural network trained to control microrobots with RL is likely to arrive at policies that are unintuitive, and could potentially uncover useful behaviors that would not be suspected or created by human engineers. Furthermore, our analysis of the direction of the rotating magnetic fields created by the learned policies strongly supports the idea that the RL agent reliably developed near-optimal behavior, which matches the behavior of a rationally designed controller. This suggests that if RL were applied to a more complex
microrobotic system for which no good models of optimal behavior were available, the RL agent might be able to autonomously identify the best way to control the system. This ability to detect subtle patterns from high-dimensional data in a model-free RL approach could ultimately lead to state-of-the-art control policies that exceed the performance of human-designed policies, as has been seen with RL algorithms in tasks like playing Go\cite{43} and classic Atari games.\cite{72}

In our experimentation, we found that the RL agent could learn successful polices from both state vector input and from raw camera images. This input flexibility demonstrates that RL could be applicable for a broad class of biomedical imaging modes in which the state of the system might be represented by MRI, X-Ray, ultrasound, or other biomedical imaging methods.\cite{25} Our results are consistent with the findings of Haarnoja et al.,\cite{61} in that the use of raw images as input requires more training time in order to develop high-quality policies compared to state vector input. Using higher dimension inputs like images has the potential to encode richer policies which respond to objects in the field of view such as obstacles which could impede the forward progress of the microrobot but would not be observable from lower dimensional feedback available in a state vector representation. In complex environments in which environmental factors such as lumen shape, fluid flow profiles, surface interactions, and biological interactions are likely to be a significant factor,\cite{39} the ability to use machine vision for state representation could significantly improve microrobot performance. While the HAMR in this study was not observed to exhibit significant deformability, an image-based input to an RL control system could also help to observe and control more kinematically complex microrobots by encoding the configuration of the robot in the state representation. All these points strongly favor the use of RL for developing the next generation of microrobot control systems.

4. Methods

4.1. HAMR

The HAMR was constructed based on a method published by Hunter et al.\cite{4} The structure of the robot was formed from a 2% w/v agar-based hydrogel (Fisher Cat. No. BP1423-500). The agar was melted to above 80 °C and mixed with iron oxide nanopowder (Sigma-Aldrich Cat. No. 637 106) to a total concentration of 10% w/v. This mix was injected into a helical 3D printed mold printed on the Elegoo Mars stereolithography 3D printer to form a helical microrobot 4.4 mm in length. The microrobot was manually removed from the mold after cooling and solidifying, and stored in deionized water until use. Because the yield of this batch fabrication technique was not 100%, robots used for subsequent experiments were chosen based on their morphology and responsiveness to magnetic fields.

4.2. Circular Swimming Arena

The PDMS swimming arena was created by molding Sylgard 184 elastomer (Sigma-Aldrich Cat. No. 761 036) over a thin 3 mm tall section of polyvinyl chloride pipe (31 mm inner diameter, 34 mm outer diameter). Access holes for the microrobot were cut, and then the molded PDMS was plasma bonded to a thin uniform sheet of PDMS to close the channel, and cured overnight at 65 °C.

4.3. Magneturret

The Magneturret was constructed by winding six identical coils with 400 turns each of 30 G magnet wire (Remington Industries Cat. No. 30H1200P) around a 0.26 in. diameter permalloy core (National Electronic Alloys Cat. No. HY-MU 80 Rod .260 AS DRAWN) cut to a length of 20 mm. These coils were fixed to the sides of an Acrylonitrile butadiene styrene (ABS) 3D printed cube with quick set epoxy. The coil was enclosed in a 3D printed housing printed in Zortrax Z-glass filament with a Zortrax M200 printer and sealed with epoxy. Glycerol coolant was pumped through the housing with a liquid CPU cooling system (Thermaltake Cat. No. CL-W253-CU12SW-A). The coils were energized by creating sinusoidal currents with an Arduino STEMtera breadboard, which took serial commands from the RL agent over USB and turned them into PWM signals which were sent to two Pololu Dual G2 High-Power Motor Driver 24v14 Shields. The power supply used to power the coils and was a Madewell 24 V DC power supply.

4.4. Overhead Camera

The overhead camera was an Alvium 1800 U-500c with a 6 mm fixed focal length lens from Edmund Optics. The camera used to take images for the state was set at a long exposure so that the HAMR and the center mark were the only visible objects in the image. A second identical camera placed above the arena at a slight angle was used to simultaneously record normal exposure video of the HAMR in the arena during operation, so that the features in the image were not washed out.

4.5. RL Algorithm

The SAC RL agent was developed in Python, using Tensorflow 2.0 for creating the neural network models. This was run on a desktop workstation from Lambda Labs. Separate processes were used for data collection and updating the neural networks so that the two operations could run in parallel. The full algorithm details are available in Algorithm 1.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

The authors acknowledge funding from federal agencies of the USA, including the National Institutes of Health through the Director’s New Innovator Award, DP2-GM132934 (WCR), Air Force Office of Scientific Research, FA9550-18-1-0262 (WCR), National Science Foundation, DMR 1709238 (WCR), Office of Naval Research, N00014-17-1-2306 (WCR), Army Research Laboratory, W911NF2120208 (WCR), National...
Institutes of Health Cellular Approaches to Tissue Engineering and Regeneration Training Program, T32-EB001026 (MRB), and the William Kepler Whiteford Faculty Fellowship from the Swanson School of Engineering at the University of Pittsburgh (WCR). We also gratefully thank Haley C. Fuller and Ting-Yen Wei for scientific discussions and manuscript suggestions.

**Conflict of Interest**

The authors declare no conflict of interest.

**Author Contributions**

M.R.B. and W.C.R.: conceptualization; M.R.B. and W.C.R.: methodology; M.R.B.: investigation; M.R.B.: visualization; W.C.R. and M.R.B.: funding acquisition; M.R.B. and W.C.R.: writing—original draft; M.R.B. and W.C.R.: writing—review and editing.

**Data Availability Statement**

The data that support the findings of this study are available in the supplementary material of this article. The code used in this work is available at https://github.com/Synthetic-Automated-Systems/RUDER_MBOT_RL.

**Keywords**

artificial intelligence, control systems, machine learning, magnets, microrobots, reinforcement learning, robotics

Received: January 24, 2022  
Revised: April 2, 2022  
Published online: July 6, 2022

[1] B. Wang, K. Kostarelos, B. J. Nelson, L. Zhang, *Adv. Mater.* 2021, 33, 2000247.

[2] H. Ceylan, I. C. Yasa, U. Kilic, W. Hu, M. Sitti, *Prog. Biomed. Eng.* 2019, 1, 012002.

[3] S. Jeon, S. Kim, S. Ha, S. Lee, E. Kim, S. Y. Kim, S. H. Park, J. H. Jeon, S. W. Kim, C. Moon, B. J. Nelson, J. Y. Kim, S. W. Yu, H. Choi, *Sci. Robot.* 2019, 4, eaa4317.

[4] E. E. Hunter, E. W. Brink, E. B. Steager, V. Kumar, *IEEE Rob. Autom. Lett.* 2018, 3, 1592.

[5] S. Scheggi, K. K. T. Chandrasekar, C. Yoon, B. Sawarayn, G. van de Steeg, D. H. Gracias, S. Misra, in *Magnetic Motion Control and Planning of Untethered Soft Grippers Using Ultrasound Image Feedback*, IEEE, Piscataway, NJ.

[6] M. Dong, X. Wang, X. Z. Chen, F. Mushqat, S. Deng, C. Zhu, H. Torlakci, A. Terzopoulou, X. H. Qin, X. Xiao, J. Puigmarti-Luis, H. Choi, A. P. Pégo, Q. D. Shen, B. J. Nelson, S. Pané, *Adv. Funct. Mater.* 2020, 30, 1910323.

[7] Q. Jin, Y. Yang, J. A. Jackson, C. K. Yoon, D. H. Gracias, *Nano Lett.* 2020, 20, 5383.

[8] J. Park, C. Jin, S. Lee, J. Y. Kim, H. Choi, *Adv. Healthcare Mater.* 2019, 8, 1900213.

[9] A. Hosney, J. Abdalla, I. S. Amin, N. Hamdi, I. S. M. Khalil, in *In Vitro Validation of Cleaning Clogged Vessels Using Microrobots*, IEEE, Piscataway, NJ.

[10] M. Medina-Sánchez, O. G. Schmidt, *Nature* 2017, 545, 406.

[11] X. Wang, X. H. Qin, C. Hu, A. Terzopoulou, X. Z. Chen, T. Y. Huang, K. Maniura-Weber, S. Pané, B. J. Nelson, *Adv. Funct. Mater.* 2018, 28, 1804107.

[12] Y. Alapan, U. Bozyuyuk, P. Erkoc, A. C. Karacakol, M. Sitti, *Sci. Robot.* 2020, 5, eaba5726.

[13] J. Zhang, Z. Ren, W. Hu, R. H. Soon, I. C. Yasa, Z. Liu, M. Sitti, *Sci. Robot.* 2021, 6, eabf0112.

[14] S. Palagi, P. Fischer, *Nat. Rev. Mater.* 2018, 3, 113.

[15] W. Hu, G. Z. Limm, M. Mastrangeli, M. Sitti, *Nature* 2018, 554, 81.

[16] J. C. Breger, C. K. Yoon, R. Xiao, H. R. Kwag, M. O. Wang, J. P. Fisher, T. D. Nguyen, D. H. Gracias, *ACS Appl. Mater. Interfaces* 2015, 7, 3398.

[17] M. Medina-Sánchez, V. Magdanz, M. Guix, V. M. Fomin, O. G. Schmidt, *Adv. Funct. Mater.* 2018, 28, 1707228.

[18] T. Xu, J. Zhang, M. Salehizadeh, O. Onaizah, E. Diller, *Sci. Robot.* 2019, 4, eaa4494.

[19] L. Ricotti, B. Trimmer, A. W. Feinberg, R. Raman, K. K. Parker, R. Bashir, M. Sitti, S. Martel, P. Dario, A. Menciassi, *Sci. Robot.* 2017, 2, eaaq0495.

[20] H. Zhang, Z. Li, C. Gao, X. Fan, Y. Pang, T. Li, Z. Wu, H. Xie, Q. He, *Sci. Robot.* 2021, 6, eaa29519.

[21] H. Xu, M. Medina-Sánchez, M. F. Maitz, C. Werner, O. G. Schmidt, *ACS Nano* 2020, 14, 2982.

[22] Y. Alapan, O. Yasa, O. Schauer, J. Giltinan, A. F. Tabak, V. Sourjik, M. Sitti, *Sci. Robot.* 2018, 3, eaa4423.

[23] T. Y. Wei, W. C. Ruder, *Apl Mater.* 2020, 8, 101104.

[24] T. Xu, J. Yu, X. Yan, H. Choi, L. Zhang, *Micromachines* 2015, 6, 1346.

[25] A. Aziz, S. Pane, V. Iacovacci, N. Koukorakis, J. Czarnecki, A. Menciassi, M. Medina-Sánchez, O. G. Schmidt, *ACS Nano* 2020, 14, 10865.

[26] J. C. Yasa, H. Ceylan, U. Bozyuyuk, A. M. Wild, M. Sitti, *Sci. Robot.* 2020, 5, eaa3867.

[27] J. B. Stock, A. J. Ninfa, A. M. Stock, *Microbiol. Rev.* 1989, 53, 450.

[28] I. Tagkopoulos, C. H. Rankin, *Learn. Mem.* 2010, 17, 191.

[29] E. L. Ardiel, C. H. Rankin, *Learn. Mem.* 2020, 28, 012002.

[30] M. Giurfa, *WIREs Cogn. Sci.* 2015, 6, 383.

[31] P. Calvo Garzón, F. Keijzer, *Adapt. Behav.* 2011, 19, 155.

[32] F. A. Bonilla, H. C. Oettgen, *J. Allergy Clin. Immunol.* 2010, 125, S33.

[33] A. Dickinson, *Contemporary Animal Learning Theory*, Vol. 1, CUP Archive 1980.

[34] A. C. H. Tsang, E. Demir, Y. Ding, O. S. Pak, *Adv. Intell. Sys.* 2020, 2, 1900132.

[35] R. S. Sutton, A. G. Barto, *Introduction to Reinforcement Learning*, MIT Press Cambridge, Massachusetts London, England 1998.

[36] J. Kober, J. A. Bagnell, J. Peters, *Int. J. Rob. Res.* 2013, 32, 1238.

[37] T. Haarmanocha, S. Ha, A. Zhou, J. Tan, G. Tucker, S. Levine, *Learning to Walk Via Deep Reinforcement Learning*, arXiv preprint arXiv:1812.11103, 2018.

[38] H. Zhui, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, S. Levine, *The Ingredients of Real-World Robotic Reinforcement Learning*, arXiv preprint arXiv:2004.12570, 2020.

[39] M. Taran, U. Almaligolu, H. B. Gilbert, F. Mahmood, N. J. Durr, H. Araujo, A. E. Sari, A. Ajay, M. Sitti, *IEEE Rob. Autom. Lett.* 2019, 4, 3075.

[40] A. Zeng, S. Song, J. Lee, A. Rodriguez, T. Funkhouser, *IEEE Trans. Rob.* 2020, 36, 1307.

[41] J. Tan, T. Zhang, E. Coumans, A. Iscen, B. Dafihner, S. Bohez, V. Vanhoucke, *Sim-to-Real: Learning Agile Locomotion for Quadruped Robots*, arXiv preprint arXiv:1804.10332, 2018.

[42] A.R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, J. Bergstra, *Benchmarking Reinforcement Learning Algorithms on Real-World Robots*, Proceedings of The 2nd Conference on Robot Learning, PMLR 2018, 87, pp. 561–591.
