Control of Magnetic Surgical Robots With Model-Based Simulators and Reinforcement Learning

Yotam Barnoy, Onder Erin, *Member, IEEE*, Suraj Raval, *Graduate Student Member, IEEE*, Will Pryor, Lamar O. Mair, *Member, IEEE*, Irving N. Weinberg, *Life Member, IEEE*, Yancy Diaz-Mercado, *Member, IEEE*, Axel Krieger, *Member, IEEE*, and Gregory D. Hager, *Fellow, IEEE*

***Abstract*—Magnetically manipulated medical robots are a promising alternative to current robotic platforms, allowing for miniaturization and tetherless actuation. Controlling such systems autonomously may enable safe, accurate operation. However, classical control methods require rigorous models of magnetic fields, robot dynamics, and robot environments, which can be difficult to generate. Model-free reinforcement learning (RL) offers an alternative that can bypass these requirements. We apply RL to a robotic magnetic needle manipulation system. Reinforcement learning algorithms often require long runtimes, making them impractical for many surgical robotics applications, most of which require careful, constant monitoring. Our approach first constructs a model-based simulation (MBS) on guided real-world exploration, learning the dynamics of the environment. After intensive MBS environment training, we transfer the learned behavior from the MBS environment to the real-world. Our MBS method applies RL roughly 200 times faster than doing so in the real world, and achieves a 6 mm root-mean-square (RMS) error for a square reference trajectory. In comparison, pure simulation-based approaches fail to transfer, producing a 31 mm RMS error. These results demonstrate that MBS environments are a good solution for domains where running model-free RL is impractical, especially if an accurate simulation is not available.**

***Index Terms*—Magnetic robots, surgical robotics, autonomous control, reinforcement learning.**

Manuscript received 28 April 2022; revised 20 June 2022 and 22 August 2022; accepted 17 September 2022. Date of publication 12 October 2022; date of current version 21 November 2022. This article was recommended for publication by Associate Editor P. Poignet and Editor P. Dario upon evaluation of the reviewers' comments. This work was supported in part by the Office of the Assistant Secretary of Defense for Health Affairs under Award W81XWH-18-1-0769; in part by the University of Maryland Medical Device Development Fund (MDDF); and in part by the National Institute of Biomedical Imaging and Bioengineering (NIBIB) of the National Institutes of Health under Award R01EB020610. *(Corresponding author: Onder Erin.)*

Yotam Barnoy and Gregory D. Hager are with the Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21287 USA.

Onder Erin, Will Pryor, and Axel Krieger are with the Department of Mechanical Engineering, The Johns Hopkins University, Baltimore, MD 21287 USA (e-mail: oerin@jhu.edu).

Suraj Raval and Yancy Diaz-Mercado are with the Department of Mechanical Engineering, University of Maryland at College Park, College Park, MD 20742 USA.

Lamar O. Mair and Irving N. Weinberg are with the Division of Magnetic Manipulation and Particle Research, Weinberg Medical Physics, Inc., North Bethesda, MD 20852 USA.

This article has supplementary downloadable material available at https://doi.org/10.1109/TMRB.2022.3214426, provided by the authors.

Digital Object Identifier 10.1109/TMRB.2022.3214426

I. Introduction

STATE-OF-THE-ART surgical robots such as the da Vinci Surgical System [1] enable minimally invasive procedures by providing surgeons with enhanced tool control and increased instrument maneuverability while simultaneously decreasing the size and number of incisions in the patient. However, these robotic manipulators have large mechanical footprints in the patient compared to the surgical end-effectors (e.g., needles and grippers), making the robotic manipulator the most invasive component of the surgical apparatus. Surgical manipulators used for guiding needles may cause tissue damage, initiate infections, and induce scarring. Unlike conventional multi-link robotic platforms, magnetic robotics is a developing field currently in a research stage. By using magnetic fields, it is possible to exert forces and torques on rigid-bodies that would induce translational or rotational motion on the rigid body.

Using externally applied magnetic fields to wirelessly manipulate surgical end-effectors may enable accurate, safe, and ultra-minimally invasive surgical procedures. Magnetic guidance of end-effectors such as needles and grippers may pave the way for more complex, advanced manipulation of untethered robotic surgical tools. Such a system would enable ultra-minimally invasive procedures, like the closing of a hole in the heart or repairing a hernia, via surgical access ports no larger than the needle used in the surgery. Such a capability may lead to shorter recovery times, decreased tissue damage, and reduced scarring [2], [3], [4], [5]. In this work, we expand on our previous magnetic needle steering experiments [6], [7], [8], moving away from manual control and towards the implementation of machine learning techniques. By developing a model-based simulation (MBS) that informs a reinforcement learning algorithm, we enable needle control without requiring a detailed physics-heavy system model.

Magnetic actuation methods can be classified as (a) torque-based rotations and (b) force-based pulling. Current reinforcement learning-based approaches for magnetic robots are implemented for torque-based magnetic actuation methods, where the applied field aligns the robot in the field, similar to operation of a compass needle [9], [10], [11]. On the other hand, for the force-based pulling approaches, there is no such stable alignment. Applied magnetic forces exert highly non-linear translation-inducing forces on the rigid bodies, resulting in high accelerations and complicating efforts at fine control. Here we apply RL methods to gradient-based pulling magnetic actuation, which is inherently less stable due to the needle moving through a non-linear magnetic gradient field.

Prior work has demonstrated simulation-based reinforcement learning methods on the control of conventional robotic systems [12], [13]. These simulation platforms are created based on known physical interactions and motions. For miniature magnetic robots, such physical interactions are challenging to predict in a simulation setting since the physical forces involved (i.e., fluid drag, surface friction, adhesion forces) vary significantly based on micro-level topological differences. Even though magnetic actuation forces are well-characterized, near-proximity estimations are known to be highly inaccurate for the commonly used magnetic models [14], [15]. Moreover, the physical disturbance forces are in the same scale of magnetic actuation forces for the miniature robots, which makes the motion hard to predict. Therefore, simulation-based learning approaches with predefined physics emulators may not translate well in motion prediction for miniature magnetic robots. It is an alternative to train the system based on only experimental data but such an approach is time-wise costly. To overcome such challenges present in some physical robotic environments, as the initial step, we created the MBS platform by learning the dynamics of the needle through real-world experiments and reflecting these dynamics on the simulation platform. Then, by using this same simulation platform, we have refined our control capabilities on the robotic system for desired motions.

We recently described the MagnetoSuture^{TM} system, a robotic system of four electromagnets enabling the untethered guidance of needles for experimental surgical tasks such as tissue penetration, tissue ligation, and recreation of suture patterns *in vitro* [6]. Magnetic robot systems such as MagnetoSuture^{TM} are difficult to manipulate manually with high accuracy due to their non-linear magnetic dynamics. These dynamics make the relationship between needle velocity, coil current, and needle location in the sample volume highly dependent upon needle location, given a constant current in a coil. Human manipulation is generally not sensitive enough to limit the forces and torques applied by a magnetic system in a sufficiently safe manner, suggesting a role for autonomous control. Classical control techniques, on the other hand, are capable of demonstrating fine tuned control of suture needles, but require rigorous physics-based modelling of magnetic forces and rigid-body dynamics to perform accurately [16].

While our previous studies have implemented either handheld remote controller operation [6] or conventional physics-based control techniques [16] for guiding a magnetic needle in the MagnetoSuture^{TM} system, here we choose to use reinforcement learning (RL) in order to avoid overt or complete dependence upon rigorous modeling of the physical forces. RL-based techniques are particularly useful here, as these physical forces are nonlinear and highly varying depending on external conditions and the motion state of the magnetic needle. Additionally, physics simulations, which are often used to train RL algorithms (e.g., [17]), are unreliable for many domains where external forces on the robots are complex and challenging to simulate. In the case of magnetic surgical robotic systems, the highly non-linear magnetic forces close to the electromagnetic coils and complex robot-fluid-tissue interaction forces both present particularly challenging scenarios to accurately model the physics-based forces on the magnetic robots.

To ease the aforementioned challenges, we propose training our magnetic robotic system using an RL algorithm with the goal of accurate and precise magnetic needle control. With this approach, we provide an alternative for autonomously controlling magnetic agents which operate under highly nonlinear and hard-to-predict physical forces and torques.

In addition to the magnetic agent control with RL-based algorithms, we further propose a hybrid training mechanism to bring the impractical training conditions (time and user supervision) into a reasonable level. Applying model-free RL agents to a real-world system such as MagnetoSuture^{TM} induces two related problems. First, training a model-free RL agent requires on the order of millions of samples, which takes a very long time to collect. Second, relatedly, not all real-world systems can be left to operate by themselves for sufficient time so as perform the required amount of state exploration to enable properly complete training data sets. Instead, some systems require constant human supervision for safety or other operational concerns. In the case of the MagnetoSuture^{TM} system, which is a custom-built electromagnet system under research lab conditions, running the system unsupervised for prolonged periods may cause overheating (and subsequent damage) or other malfunctions. These restrictions make such a system particularly unsuited to the approach of allowing an RL-based agent to run for a prolonged period of time. Other robotic environments suffer from similar constraints. For example, [18] needed to build a unique enclosed space and modify their drones to train drone robots to fly, and most grasping robots (e.g., [17]) are potentially extremely dangerous to humans or property, but are confined to a small safe space in a lab setting for the purposes of RL training. If we are to let these robots train using RL in realistic environments and realistic tasks, we need methods that enable them to operate safely for long periods of time.

We therefore introduce a method which we term model-based simulation (MBS). In MBS, we first learn the dynamics of the system using a neural network, and then use this model as the core of our simulator, inserting additional basic constraints as needed, in order to train a model-free RL algorithm. This approach does not require us to run all of the experiments in the physical world. After we acquire sufficient real world training data, we can transfer the magnetic agent motion primitives and run more repetitions in the simulation environment we created. Since the simulations are orders of magnitude faster to run experiments on, the overall time required for training is reduced. Moreover, the time required for supervision is also reduced, enabling us to run rapid training on the physical system with minimal heating issues or other time constraints. Our approach allows us to take advantage of the sample efficiency gap between supervised neural network training and the far less sample-efficient model-free RL algorithms. We compare MBS to learning on a 'pure' simulator (which we devised as a benchmark) and show that the latter approach fails to transfer to the real world, whereas MBS succeeds. We also show that the data used to create the MBS is insufficient to train an RL algorithm offline solely by using the real world training data.

The main contributions of this paper are as follows:

- We introduce MBS as a solution for systems where prolonged RL runtimes are unfeasible. MBS can accelerate traditional RL by more than two orders of magnitude and serves as a strong alternative approach when reliable simulations are unavailable.
- We demonstrate the first application of RL to magnetic needle control.

II. Related Work

Model-free RL has been applied successfully to Atari games [19], autonomous driving frameworks [20], [21], first person video games [22], and other environments. In the world of robotics, model-free RL has rapidly grown in prominence and capability, having been used to control robotic arms in the context of object grasping and movement [23], [24], [25]. While successful, RL in the real world poses challenges due to overly long training times. Model-based RL methods have been implemented for tuning human–robot interactions, teaching robots to assist humans in increasingly predictive and adaptive ways [26]. However, such model-based RL techniques have not been conveyed to advanced tetherless surgical robotic manipulators such as magnetic guidance systems.

Surgical robotics has been studied extensively in the non-RL setting [27], [28], [29]. Most of these works approach the problem using classical methods of path-planning and needle segmentation. Implementing RL-based methods may offer more generalizable and less model-specific approaches to surgical robotics.

Recently, RL has been applied to various surgical environments. Richter et al. developed an open-source simulation based on V-REP, named dVRL, which was used to teach an Intuitive Surgical da Vinci arm to reach a target and to move an object to a target [13]. Varier et al. applied Q-learning directly to a real world da Vinci robot [30]. These efforts encountered problems resulting from RL sample inefficiency and used multiple approaches to minimize those problems, including limiting the dimensionality of the space to accelerate learning. We previously used the da Vinci Skill Simulator as an RL environment, allowing for autonomous movement using either image or state data [31]. The sample inefficiency problem was handled by using simultaneous offline data playback.

Model-free RL was also successfully applied to magnetic surgery scenarios, where a magnetically actuated endoscope was controlled inside *ex vivo* stomachs [11]. However, training time was expensive, and thus the system was trained to follow one particular path only.

On the other side of the RL spectrum we find model-based RL, which is far more sample efficient than its model-free cousin, but requires some sort of reliable model for the dynamics of the system to search through, as previously demonstrated [32], [33]. Augmenting model-free RL with some model-based elements allows us to increase the sample efficiency, as compared with model-free RL, while maintaining some of the benefits of model-free RL, such as generalization and robustness. While some attempts at hybridization use non-neural network-based models [34], more recent attempts feature a neural network of some sort for the dynamics model.

Various approaches have been tried for combining model-free and model-based designs. Nagabandi et al. [35], for example, suggests an entirely model-based approach, the result of which is used to bootstrap a policy gradient model-free algorithm. Dreamer, on the other hand, redefines the concepts of model-free and model-based reinforcement learning, learning dynamics models and then back-propagating through them to maximize rewards [36], [37]. A similar approach is taken by Byravan et al. [38]. Wang et al. [39] adds a look-ahead model-based module to a model-free RL path. The final choice of which action to take is chosen between the model-free and model-based path. A similar strategy is followed by Weber et al. [40]. One element to note is that algorithms that work using images (such as [36]) require a reward-prediction model as part of their dynamics model. This cements the reward as part of the environment, which is a limiting factor—often one wishes to modify the reward to obtain better results. Becker-Ehmck et al. [18] use another approach whereby a probabilistic dynamics model is updated concurrently with a policy-gradient actor-critic model.

III. The MagnetoSuture^{TM} System Setup

We now describe the MagnetoSuture^{TM} system (Figure 1), which is similar to that shown in our previous work [41]. Using four individually addressable electromagnets oriented along the X and Y axes, we control a 22 gauge, 23.5 mm long, stainless steel hypodermic needle with 42 internally embedded NdFeB permanent magnets sealed inside with glue. We submerge the needle in a viscous medium mixture of 25 ml water to 5 ml glycerol, inside a Petri dish of diameter 85 mm. We obtain images using a FLIR Blackfly camera with a resolution of 1280 $\times$ 1024 pixels. The workspace is illuminated by a ring light mounted on a custom 3D-printed adapter. During activation, we reach maximum currents of 15 A per electromagnet. At high currents, the system heats up rapidly, requiring active water cooling to stabilize. The electromagnets are driven by motor controllers (RoboClaw, Basic Micro Inc.) powered by an AC/DC converter. The motor controllers are driven by an Arduino board, which is commanded by a Python program running on a connected computer.

Using the magnetic controls, we can move the needle around the Petri dish in two dimensions and control needle heading. However, the force and torque on the needle are location-specific due to non-linear variations in the magnetic fields, and this makes it difficult for a human to control the system accurately. Figure 2 outlines the general control zones of the

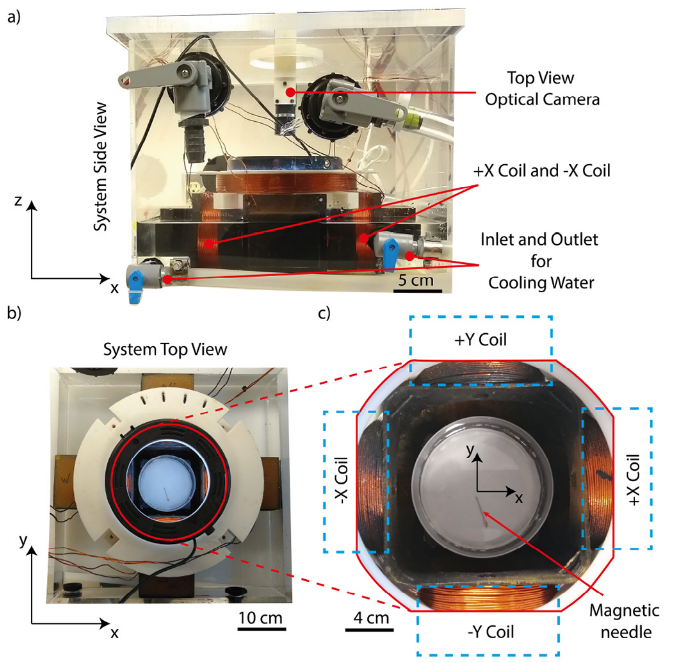


Fig. 1. Pictures of the MagnetoSuture^{TM} system, consisting of a 4 coil array surrounding a central Petri dish, with coil axes laying along the X and Y axes. The inside bottom surface of the Petri dish sits in the X-Y plane. a) A side view of the system, which is surrounded by a plexiglass container and partially submerged in water for cooling. The side view also shows a coil centered along the +Z axis (not implemented here). The camera at the top allows for needle localization and recording of experiments. b) The view from the camera at the top, looking through the +Z coil. The magnetic needle sits in a Petri dish, suspended in a viscous mix of water and glycerol. Four coils surround the dish along the four cardinal directions, allowing for moving the needle around the dish by supplying current to the coils. Bright masking tape was placed on the needle end to enable "fool-proof" localization for the purposes of data collection. c) Magnified inset detailing system axes and showing the Petri dish with a needle sitting in fluid in the Petri dish.

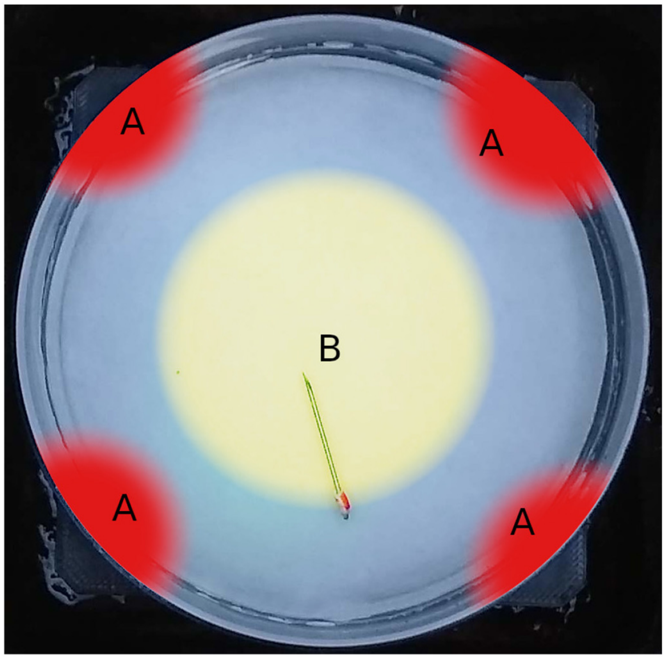


Fig. 2. Representation of control zones of the MagnetoSuture^{TM} state space. Area A represents dead zones, where the magnetic field doesn't allow free movement. Area B represents the zone of most stable control.

system. Zones A represent 'dead zones' which cannot be accessed by the needle due to the locations and dynamics of the magnets. Zone B represents the most stable zone in the middle of the Petri dish. Outside of this zone, movement is either overly fast due to proximity to a nearby coil, or too weak due to large distance from the opposite pulling coil. We present an analysis of the dynamics of the system for the simulator in Section IV-C.

IV. Method

A. Model-Free RL

We frame our problem in the familiar terms of a Markov Decision Process (*MDP*) involving an agent interacting with an environment. The agent obtains the state S_t of the environment, and in return chooses an action a_t out of the set of all possible actions A. In return, the agent receives a reward R_t and a new state S_{t+1}. The specific choices made by the agent given certain states are termed the policy π. Our goal is to find the optimal policy π^* by maximizing all future rewards over time. The value of each state is defined as the sum of all rewards including future rewards:

$$Q(S_t, a_t) = \sum_{i=t}^{\infty} \gamma R_{a_i}(S_i) \tag{1}$$

where Q is an expected rewards function given an action, as computed by our algorithm, S_t is a given state at time t, a_t is a selected action at time t, γ is the discount factor that describes the rewards function $(0 \leq \gamma \leq 1)$, R_{a_i} is the reward function for each action iteration i, and S_i is the i^{th} state. Here, the discount factor γ can be interpreted as the probability of succeeding for a given time step Δt. This leads us to the Bellman equation:

$$Q(S_t, a_t) = R_{a_t}(S_t) + \gamma Q\big(S_{t+1}, \text{argmax}_{\text{A}} Q\big(S_{t+1}, a'\big)\big) \tag{2}$$

where argmax_A is the action A that maximizes the rewards function $Q(S_{t+1}, a')$ for state S at time $t+1$, and a' is the proposed action. We optimize this equation using a neural network in an attempt to reach an optimal expected reward function Q^*. This equation is well defined over a discrete action space, as done by Deep Q-Networks (DQN) [19]. We choose to use a continuous model-free algorithm called Twin Delayed Deep Deterministic policy gradient (TD3) [42], which is more suitable for the current values used to control our system. For a continuous action space, TD3 utilizes two neural networks. The first is the actor network, which stores the current optimal policy: a mapping from states S to actions A. The second is called the critic, and it stores the Q-value of each state-action pair. The networks are updated in lockstep: First, we update the critic to the updated Q-value of the actor's chosen actions. We then use the gradient of the critic network to update the actor's behavior.

The model-free RL algorithm contains several weaknesses. Most severe from our perspective is its sample inefficiency, which forces very long training times.

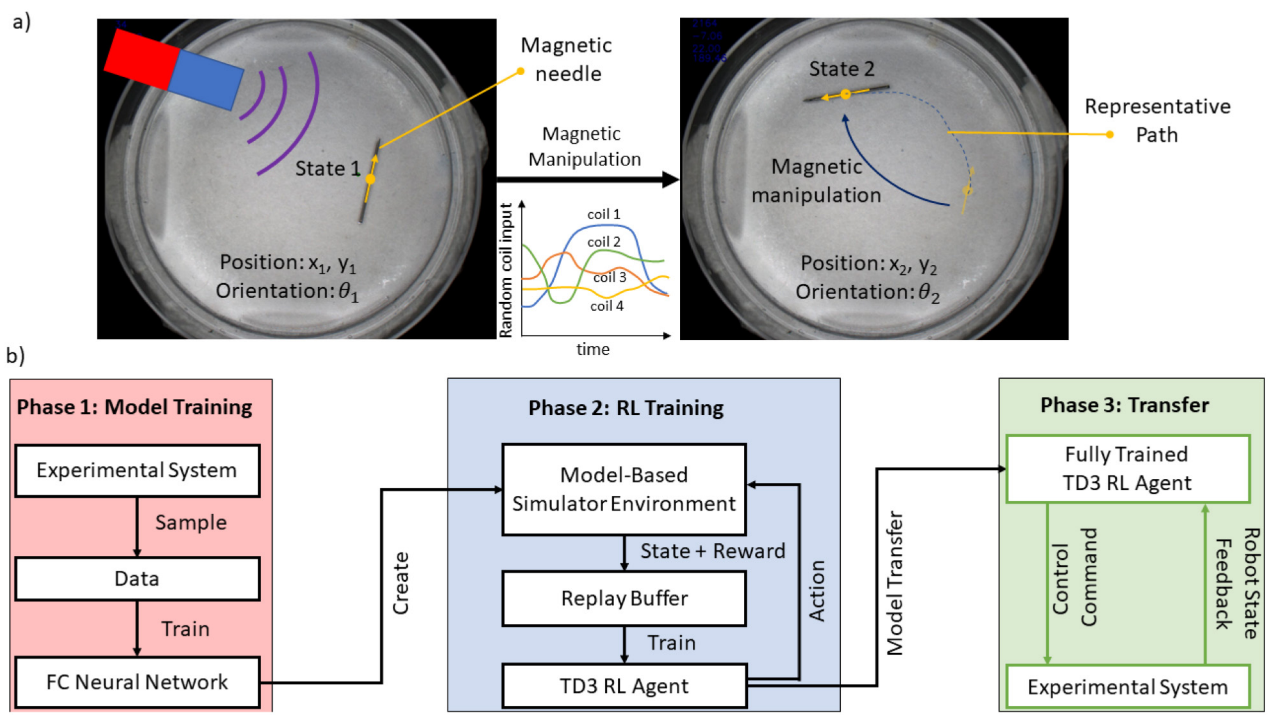


Fig. 3. Magnetic manipulation and workflow of our MBS Model-Free RL method. a) A magnetic needle in the workspace is shown with State 1, having position (x_1, y_1) and orientation θ_1. A random magnetic field generated by our algorithm induces magnetic pulling force and magnetic torque on the needle. After application of the random magnetic field, the magnetic needle arrives at State 2 with position (x_2, y_2) and orientation θ_2 after having traveled some representative path. The force and torque intensity and direction depend on both the coil current inputs and the location of the needle. Due to the many combined interactions with the environment such as fluidic and surface contact interactions, a precise, predictive mathematical model of the needle displacement may be challenging to obtain. b) Training the RL agent consists of 3 separate phases. In phase 1, we collect data from the real-world MagnetoSuture$^{\text{TM}}$ system and train the MBS using the collected data. In Phase 2 we use the MBS to train the TD3 RL Agent. TD3 works by collecting data in a replay buffer as it acts in the environment, and then sampling from it gradually. While the implementation of the TD3 agent is based on [42], our MBS is a novel component. Phase 3 involves domain adaptation to the real world, where we use the trained RL algorithm to receive the state of the MagnetoSuture$^{\text{TM}}$ system and act in response.

B. Model-Based Simulation RL

Figure 3 shows the schematic and workflow of our MBS model-free RL method, which combines our model-free RL algorithm with a pretrained generative model supplemented by some additional common-sense constraints. Utilizing an MBS environment provides many advantages. MBS RL allows us to run the RL process safely on the computer, while also running orders of magnitude faster than in the real world. Further concerns can also be alleviated with MBS. For example, depending on the task at hand, resetting the experiment to advance to the next episode may be difficult in the real world. Furthermore, RL often needs adjustments and corrections between runs, such as modifications to reward shaping. This can require rerunning the entire algorithm from scratch.

1) Learning a Model-Based Simulation: In *Phase 1: Model Training* of our algorithm, we train a neural network to learn the dynamics of our system. The dynamics we are trying to capture are illustrated in Figure 3a. As the magnetic forces from the coils interact with the magnetic needle, they induce needle motion. We capture these dynamics by converting images of the workspace to needle locations using our localization algorithm, and then teach the extracted dynamics to the network. The network predicts the next state S_{t+1}, comprised of location P_{t+1} and angle θ_{t+1} of the needle, given the previous state S_t, which is comprised of location P_t, angle θ_t, and coil currents $\mathbf{I_t}$. Our generative dynamics model is deterministic and Markovian: given state S_t and action a_t, we train a neural network to predict state S_{t+1} by minimizing the L_2 loss between the ground truth and the state predicted by the neural network using stochastic gradient descent (SGD).

We use a fully connected neural network accepting an input layer of size 5 comprising of S_t and a_t The network contains 3 hidden layers of size 256, 128 and 32, respectively, and then an output layer of size 3 predicting S_{t+1}. The input and hidden layers each use ReLU activations and make use of batch normalization layers.

Our state space consists of the position tuple (x, y, θ) of the needle. We obtain this position via the improved neural network-based approach we introduced in [41] for localization. This algorithm uses a U-Net convolutional neural network to segment the camera image into needle and background pixels, then processes the needle pixels using clustering and RANSAC line detection to extract position and orientation. This allows us to avoid having to use image data, which would add a further burden on the algorithm of having to find an appropriate embedding. We augment the localization method, which is reliable as far as angle and coordinates are concerned, with a color-based method for determining needle orientation with 100% certainty. After this orientation correction the localization achieves a root-mean-square error of 0.61 mm in position and 0.81 degrees in orientation in the environment used in this work.

2) Training RL With MBS: most In *Phase 2: RL Training* of our algorithm, we've finished training the MBS, and we now use it as an OpenAI Gym [43]-compatible RL environment for trajectories. Since our RL algorithm can learn to navigate dynamically to any user-chosen location, we integrate the desired (x_{user}, y_{user}) points into our state space. The RL algorithm therefore needs to learn to differentiate states S_i not only based on the needle position, but also by the target position desired by the user.

The TD3 [42] algorithm utilizes a critic and actor network in tandem. The critic is a fully-connected network consisting of an input layer of size 7: we use 3 for the states (x, y, θ), 2 for target coordinates (x_{user}, y_{user}), and 2 for the desired action (x_v, y_v). Additionally, the critic is composed of 2 hidden layers of size 256 neurons each, and an output layer of size 1 for the Q value. The actor network is similarly fully-connected, consisting of an input layer of size 5 for the state and target, two fully-connected hidden layers of size 256, and an output layer of the action space dimensionality (2). All layers but the output layers use ReLU activation, while the actor's output layer is mapped to the $[-1, 1]$ range via the hyperbolic tangent (tanh). Batch normalization is avoided for TD3, as it tends to hinder learning.

3) Action Space: Our action space consists of the current value (in Amperes) of each of the four coils in the system (Figure 1b). We simplify this action space by limiting ourselves to nonholonomic constraints by removing simultaneous control of oppositely-oriented coil arrays. Four-dimensional control is difficult for a human to produce reliably, making the full space exploration needed for Section IV-B1 more difficult to obtain. We therefore reduce the control signal to a simple two dimensional current vector $\mathbf{I}$.

The system is driven using pulses applied for t_{pulse}, with an intermittent rest time of t_{rest}. We set t_{pulse} to 100 ms and t_{rest} to 200 ms, resulting in a 3 Hz rate of control. This serves multiple purposes. First, it helps our localization system avoid any mistakes due to blurry images by giving time for the needle to come to a full stop. Second, it enables us to enforce Markovian assumptions about the state space: since previous velocity is not a factor as the needle comes to a full stop, we do not need to worry about providing the neural networks with a history of states, and all next states S_{t+1} are directly dependent on S_t and the action a_t.

For RL to function, proper state-space exploration and an appropriate reward function is crucial. Rather than using a stateless epsilon-greedy approach for exploration, we make use of an Ornstein-Uhlenbeck process [44], which keeps some memory of previous random values and momentum. This process tends to provide a good random exploration signal in our experience.

4) Rewards: Reward functions ultimately drive the entire model-free RL process, and a bad reward function can make a problem intractable for the algorithm.

Our reward function rewards the agent for advancing towards the target and penalizes it for moving away from it. Additionally, we penalize the agent for taking too much time to encourage rapid convergence to the target. A large bonus awaits the agent once it reaches the target. Our reward structure is as follows:

$$R_t = \alpha \Delta d_t + \beta f_{close} + \tau \tag{3}$$

where R_t is the reward function at time t, α is the learning rate constant, Δd_t is the difference in distance to the target at time t, β is a constant to weight for proximity indication in the reward function, f_{close} is 1 when the target is within 3 mm and 0 otherwise, and τ is a time penalty per step. Experimentally, we set the values of the constants to be $\alpha = -10$, $\beta = 10$, and $\tau = -0.04$. Given the constraints of our system, Equation (3) sets the range of episodal rewards to be somewhere between -5 and 10 (that is, $-5 \leq R_t \leq 10$).

The MBS mostly uses the neural network for dynamics, but also contains its own additional rules and constraints: Any movement outside of the Petri dish (which would never happen in the real environment) causes an early episode reset (most episodes last between 5 and 30 steps), and we do not feed zero current values (which would result in no movement) into the neural network in order to reduce noise. Essentially, we recognize that the MBS may not perfectly capture the full dynamics of the system and try to account for errors.

5) Translating to the Real World: In *Phase 3: Transfer* of the algorithm, once the RL agent is fully trained, we can use it to apply RL to the real world, assuming the dynamics captured by the MBS are sufficiently accurate. We test transfer performance in Section V-D.

C. The MagnetoSuture^{TM} System Simulator

We compare our MBS approach to a more traditional approach of accelerating RL by training on a physics-based simulator, as in [17]. This involves training an RL agent first on a simulator, and then transferring to a real-world environment. The MagnetoSuture^{TM} simulator implements the following physics models and assumptions. A magnetic needle moving on a planar Petri dish under magnetic fields experiences various forces due to the magnetic interactions and interactions with the physical environment. The magnetic forces and torques are the key for inducing a wireless motion and can be represented as follows: let the magnetic field induced by a current-carrying coil be given by $\mathbf{B}$, derived as the gradient of the magnetic potential

$$\mathbf{B} = -\mu_0 (\nabla_r \phi(\mathbf{r})) = -\mu_0 \frac{\partial \phi}{\partial \mathbf{r}}^T \tag{4}$$

where μ_0 is the permeability of vacuum, ϕ is the scalar electric potential, and $\mathbf{r}$ is the position vector of the point of interest with respect to the coil. If a magnet with magnetic moment $\mathbf{m}_p$ is present in the external magnetic field generated by this coil, the magnet experiences a force $\mathbf{F}_m$ proportional to the gradient of the magnetic potential $\mathbf{B}$, given by

$$\mathbf{F}_m = \nabla(\mathbf{m}_p \cdot \mathbf{B}). \tag{5}$$

The magnet also experiences a torque τ_m given by

$$\tau_m = \mathbf{m}_p \times \mathbf{B}. \tag{6}$$

The magnetic fields generated by multiple coils superimpose, i.e., for M coils generating individual magnetic fields, the total magnetic field vector at a point $\mathbf{r}$ is $\mathbf{B}(\mathbf{r}) = \sum_{k=1}^{M} \mathbf{B}_k(\mathbf{r})$, where $\mathbf{B}_k$ is the magnetic potential generated by the k^{th} coil in the system. When the current-carrying coils are relatively small or we are interested in the magnetic field for a point that is sufficiently far away from the coil, a dipole model can provide a suitable approximation of the magnetic field strength. The estimated magnetic field with the dipole moment can be represented as follows: let the center of a coil be given by a point $\mathbf{r}_k$ and let $\mathbf{d}_k = \mathbf{r} - \mathbf{r}_k$ be the displacement vector to the coil from the point of interest. The magnetic field estimation with the dipole moment $\mathbf{B}_k^{dipole}$ at a distance $\mathbf{r}$ from the magnet can be represented as

$$\mathbf{B}_k^{dipole}(\mathbf{r}) = -\frac{\mu_0}{4\pi \|\mathbf{d}_k\|^3}\left(\mathbf{I}_k - 3\frac{\mathbf{d}_k \mathbf{d}_k^T}{\|\mathbf{d}_k\|^2}\right)\mathbf{m}_k \tag{7}$$

where μ_0 is the permeability of vacuum, $\mathbf{m}_k = \mathbf{I}_k N_k A_{k,area}$ is the magnetic moment vector of the k^{th} coil having current $\mathbf{I}_k$, with number of turns N_k, pointing in the direction orthogonal to the coil loop area $A_{k,area}$. Although the dipole model gives a suitable approximation of the magnetic field $\mathbf{B}$ in the far-field, it does not accurately capture the nonlinearities of the magnetic field $\mathbf{B}$ at locations near to the coil.

In addition to the inaccuracies in modelling magnetic forces and torques, physical interactions also induce inaccuracies in estimating the net force and torque on the magnetic needle. For a needle immersed in the fluid and sliding along the surface of the Petri dish, the main forces and torques acting on the needle can be classified as the magnetic forces and torques ($\mathbf{F}_m$, τ_m), gravitational force ($\mathbf{F}_g$), the buoyancy force ($\mathbf{F}_b$), surface contact forces ($\mathbf{F}_n$), friction forces and torques ($\mathbf{F}_f$, τ_f), and drag forces and torques ($\mathbf{F}_d$, τ_d) [4], [5]. The equations of dynamics can be represented as follows:

$$\mathbf{F}_{net} = \text{m}\mathbf{a}_{acc} = \mathbf{F}_m + \mathbf{F}_d + \mathbf{F}_g + \mathbf{F}_b + \mathbf{F}_n + \mathbf{F}_f, \tag{8}$$

and

$$\tau_{net} = \gamma_{rot}\mathbf{I}_{moment} = \tau_m + \tau_d + \tau_f, \tag{9}$$

where m is the mass of the needle, $\mathbf{a}_{acc}$ is the linear acceleration of the needle, γ_{rot} is the rotational acceleration, and $\mathbf{I}_{moment}$ is the moment of inertia at the center of the needle. Since the physical environment is not perfect (i.e., asymmetric needle properties, surface roughness, temperature dependent fluid viscosity, non-quiescent fluid), capturing the second-order dynamics by modeling these forces yields significant inaccuracies. All of these factors make accurate simulation of the MagnetoSuture$^{\text{TM}}$ environment difficult.

V. Experiments

A. MBS Training Results

In order to create the MBS, we collect data while exploring the full area of the Petri dish at different current levels. Exploration is done manually using a controller, allowing us to explore the space both uniformly and rapidly. We notice while exploring the space that the magnetic field causes the formation of 'dead areas' close to the corners near the magnetic coils as seen in Figure 2. We collect 34,742 data points, which amounts to about 3.2 hours of recordings altogether. We then normalize the data and train the neural network, achieving an L_2 validation loss of *0.033* and a test loss of *0.045*. It is worth noting that while this data is sufficient for training a neural network to learn the dynamics, it does not appear to be enough for training an off-policy RL algorithm directly (see Section V-C).

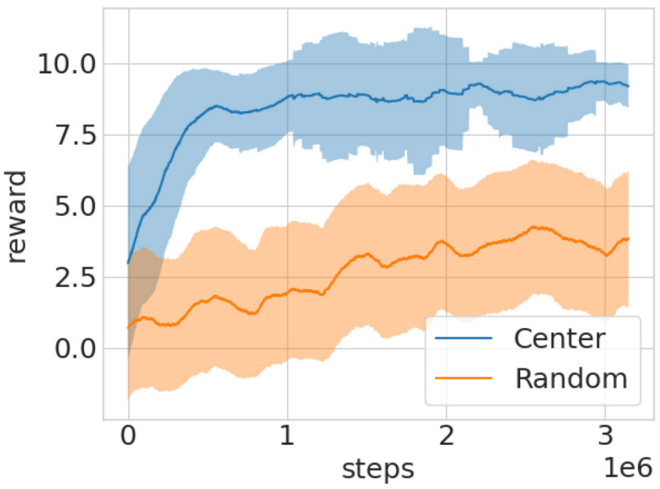


Fig. 4. Training result for the MBS-trained RL agents. One agent is trained to reach the center of the Petri dish, and one to reach a user-specified target which is sampled randomly. The graph shows averages over time and 2 standard deviations above and below the mean.

B. RL Training Results

We train and test the TD3 RL algorithm within the MBS environment. Given our desire to run environments in parallel, our bottleneck is the CPU. Despite making use of a fairly modern CPU (Ryzen 3900X), parallel execution tends to be limited by memory bandwidth and synchronization overhead. We found that running more than 3 environments in parallel gave us no additional benefit. We therefore run 3 environments in parallel, setting our learning rate to 0.0003 and use the Adam optimizer. Utilizing 3 parallel environments, the RL algorithm is able to take around 700 steps per second using the MBS. This is about 233 times faster than the 3 Hz limitation of the real-world MagnetoSuture$^{\text{TM}}$ system. More importantly, the MBS environment can run as long as we want it to, unlike the real-world system which we cannot run safely for the required periods of time.

We train two different policies: one trained to move to the center of the dish from any starting location, and one trained to move from a random location to any given location in the dish. For the latter policy, we append the desired target location to our state as described in Section IV-B2. The results can be observed in Figure 4. Additionally, we train similar RL policies on our simulator.

We observe that the MBS-based RL center policy (Figure 4) was able to reach its maximum reward within around 500k steps or 1.5 hours (Table I), whereas the much harder random-target policy needed 2.5 million steps to reach its maximum point. Assuming no issues during training and a similar convergence time, running only the center-based policy on the real system given the 3 Hz rate would take around 300 hours

TABLE I
COMPARISON OF THE DIFFERENT RL MODELS AND THEIR TRAINING TIMES TO ACHIEVE MAXIMUM REWARD. *RUNNING ON A REAL-WORLD SYSTEM WAS ATTEMPTED BUT FOUND TO BE UNFEASIBLE FOR THE DURATIONS REQUIRED

RL method	task	training time (hrs)
MBS	center	1.5
MBS	random	3.5
sim	center	5.5
sim	random	27
real	center	NA*

with constant supervision due to safety issues, which is simply not feasible. The random-target policy cannot attain as high of an average reward due to the fact that the Petri dish contains dead zones (see Figure 2) and we do not filter those out when selecting random target locations.

For comparison, we trained similar RL policies based on the simulator. We also attempted to train RL on MagnetoSuture^{TM} in the real world, but the time requirements combined with safety limitations made it impractical, as mentioned above. The time to reach maximum reward can be seen in Table I. It is worth noting that the pure simulation appears to take longer primarily because it can keep improving its behavior. However, the pure simulation fails to capture essential dynamics of the real system, as we shall see below.

C. Off-Policy RL Comparison

Off-policy algorithms like TD3 can potentially learn rapidly from off-policy data, i.e., data that has been recorded previously, but does not match the current policy π. Since we collect offline data for developing the MBS in Section IV-B1, we wish to test whether this data is sufficient to train an off-policy RL algorithm directly, as done in [31], [45], rather than requiring an MBS approach.

We split up the MBS training data from Section IV-B1 into pseudo-episodes of length 10-15 steps, and proceed to play back the off-policy data, evaluating performance on the MBS environment. In an attempt to make the RL agent learn as effectively as possible, we try different mixes of off-policy and on-policy environments (using data from the MBS environment). On-policy environments allow the agents to explore actively and collect data. Off-policy environments feed the agent the pre-recorded data we collected in Section IV-B1 as if they were exploring it, but the agents have no active control (the data is prerecorded). We use 3 parallel MBS environments, and test 4 mixes of off-policy stored data and fresh exploration:

a. 0 off-policy environments and 3 on-policy environments,
b. 1 off-policy environment and 2 on-policy environments,
c. 2 off-policy environments and 1 on-policy environment,
d. 3 off-policy environments and 0 on-policy environments.

These mixes provide different ratios of fresh on-policy data and off-policy data. All RL agents were trained to reach the center point of the Petri dish. The results are shown in Figure 5.

We notice that using some off-policy data allows the RL algorithm to learn a little faster, at the cost of some stability.

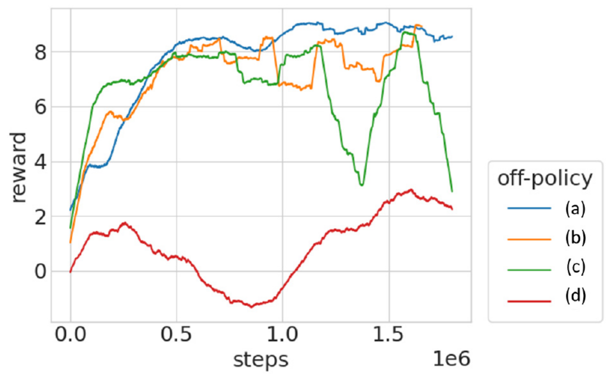


Fig. 5. We attempt to use the same data collected for MBS training and apply it to off-policy learning, mixing off-policy learning with on-policy exploration in the MBS pseudo-simulation environment. We show the results of running multiple agents in parallel. The graph shows the learning performance of (a) learning without an off-policy data environment, (b) with 1 off-policy environment and 2 on-policy environments, (c) with 2 off-policy and 1 on-policy, (d) with all 3 environments off-policy.

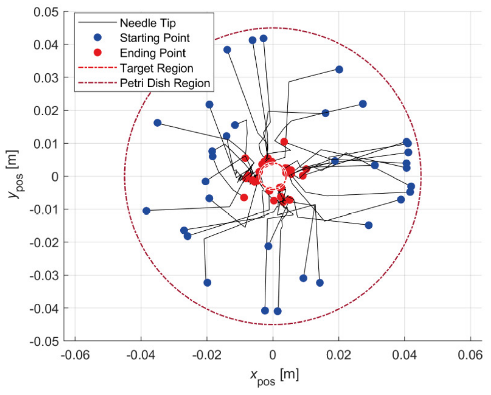
The pure off-policy RL algorithm collapses before it can learn sufficiently: it appears there simply is not enough collected data available for the RL algorithm to reach full success. This suggests that while our offline data is enough for the MBS neural network to pick up on dynamics, it is not enough to teach a full RL algorithm, which is less sample-efficient than supervised learning with the same data. In future work, we may attempt to further compare a fully off-policy RL approach to our own approach in terms of run time and mean error. However, here we observe that our collected data was insufficient for a purely off-policy RL approach. Off-policy RL is fast but sample-inefficient, while our MBS-based approach is both fast and sample-efficient.

D. Real-World Transfer

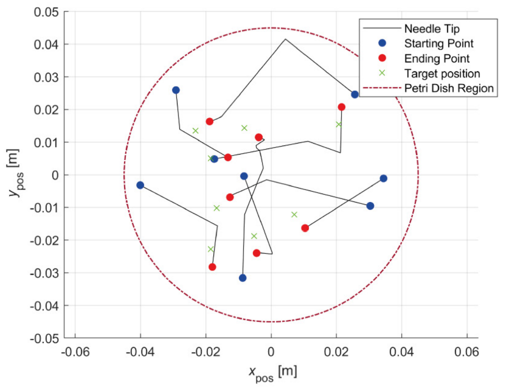
We now apply our learned RL policies, using both MBS and the physics-based simulation, to the real world by testing them on the MagnetoSuture^{TM} system. At each point we retrieve the current state of the world, append any requested target (if the policy requires it), and then have the TD3 policy choose actions for us, which we then carry out. We test 3 different real-world actions:

1) **Move to the center** This action consists of moving only to the center of the dish after we first move to a random location. We execute the policy (in this case, the second policy in Section V-B) and record the results, which can be observed in Figure 6a.
2) **Move to a random point** We generate random targets in the dish and tell the policy to move to that location. The results can be observed in Figure 6b.
3) **Follow a square path** We generate a set path of a square for the policy to follow, and feed the policy each point along that path one by one, until the path is done. The results can be observed in Figure 6c.

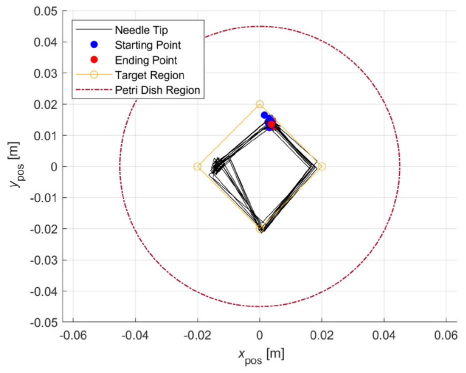
We measure the real-world performance of the different policies in terms of their deviations from the targets. The results can be observed in Table II. Additionally, a supplemental video contains demonstrations of the needle moving



(a) Center Point Results



(b) Random Point Results



(c) Square Path Results

Fig. 6. Results for the real-world center point, random point and square tracking tests. The center-point test is performed using an RL agent trained only for that purpose. The two latter tests are performed with an RL agent trained to go to a user-specified target. Both agents are trained using the same model-based environment.

under RL guidance. Note that the sim-based RL policy, which appeared to train well on our simulator, completely fails to transfer to the real world, resulting in errors of 31 mm, or roughly the size of the radius of the dish. The sim-based agent is unable to react intelligently to the state input and overshoots every time. The MBS-based agents, on the other hand, transfer successfully, performing quite well with root-mean-square (RMS) error of 5.3 mm to 7.4 mm, depending on the task. We attribute these errors to inaccuracies in our setup which we hope to improve on in future work. For example, our MagnetoSuture$^{\text{TM}}$ camera is not fixed in place, requiring a manual and imprecise calibration routine. Such small sources of noise can confuse the MBS training routine, and can be fixed by fixing the camera or using a robust automatic calibration routine, which we hope to add in future. Another possible source of error is the MBS model itself. Further work on improving the MBS neural network model could increase future accuracy. Additionally, further work on establishing precise control of needle orientation and angle will improve the system's ability to provide increasingly useful manipulations in future, more complex scenarios.

TABLE II
COMPARISON OF THE DIFFERENT RL MODELS AND THEIR REAL-WORLD PERFORMANCE ERRORS

RL agent	test	mean error (mm)	stdev (mm)	transfer
Sim-based center	center	31	29	no
MBS center	center	6.3	2.1	yes
MBS target	point	5.3	0.8	yes
MBS target	box	7.4	0.7	yes

VI. Conclusion

We successfully use model-based simulation (MBS) to train a model-free RL agent to control a magnetic robot system in an MBS and then in transfer to the real world. Our solution is applicable to domains where model-free RL is impractical due to its long running time and free-form exploration, and especially to domains where simulation is difficult or impossible due to hard-to-model forces. We show that our method can capture the dynamics of the system far better than a simulation of the magnetic and robot-environment interaction forces given their complexity. Our approach accelerates real-world model-free RL techniques significantly, allowing for a controlled data collection phase followed by offline RL training. This allows us to still enjoy the benefits of model-free RL, such as the lack of a need for detailed system models. Controlling the orientation of the needle may be implemented via multipoint trajectory tracking algorithms or assigning orientation as a separate parameter for the RL algorithm. Improving our algorithm for simultaneous position and orientation tracking will be investigated in the future studies.

While our system environment here is fairly simple, conceptually the proposed methodology is applicable to surgical environments. In such settings, dynamics may make model-based systems cumbersome and, possibly, ineffective due to the number of variables needed for accurate prediction. Although the studied scenario demonstrates the capabilities of RL approaches in magnetic needle steering, variations in surgical conditions such as the presence of flow or variations in tissue mechanics will require significantly more training data and, possibly, entirely different training protocols. Learned and practiced human surgeons are able to adjust to complex surgical scenarios having numerous processes and physical variables without computationally precise understanding of every variable on hand. Similarly, it is hoped that machine learning may enable robotic surgeons to control instruments using similar experiential knowledge bases, without relying on completely physics-based models for planning subsequent actions. Here, we test drive a Reinforcement Learning approach in a simplified magnetic manipulation scenario involving a magnetic suture needle. While a limited setting, we feel the nascent nature of RL for magnetic manipulation may benefit from such constrained experiments. We look forward to refining this technique further and demonstrating its applicability in domains with similar constraints, such as robots in an open factory setting.

REFERENCES

[1] G. S. Guthart and J. Salisbury, "The intuitive telesurgery system: Overview and application," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, pp. 618–621.

[2] O. Onaizah and E. Diller, "Tetherless mobile micro-surgical scissors using magnetic actuation," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 894–899.

[3] C. Forbrigger et al., "Cable-less, magnetically driven forceps for minimally invasive surgery," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1202–1207, Apr. 2019.

[4] O. Erin, D. Antonelli, M. E. Tiryaki, and M. Sitti, "Towards 5-DOF control of an untethered magnetic millirobot via MRI gradient coils," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020, pp. 6551–6557.

[5] O. Erin, H. B. Gilbert, A. F. Tabak, and M. Sitti, "Elevation and azimuth rotational actuation of an untethered millirobot by MRI gradient coils," *IEEE Trans. Robot.*, vol. 35, no. 6, pp. 1323–1337, Dec. 2019.

[6] L. O. Mair et al., "MagnetoSuture: Tetherless manipulation of suture needles," *IEEE Trans. Med. Robot. Bion.*, vol. 2, no. 2, pp. 206–215, May 2020.

[7] L. O. Mair et al., "Going hands-free: MagnetoSuture for untethered guided needle penetration of human tissue ex vivo," *Robotics*, vol. 4, no. 10, p. 129, Dec. 2021. [Online]. Available: https://www.mdpi.com/2218-6581/10/4/129

[8] O. Erin et al., "Overcoming the force limitations of magnetic robotic surgery: Magnetic pulse actuated collisions for tissue-penetrating-needle for Tetherless interventions," *Adv. Intell. Syst.*, vol. 4, no. 6, Apr. 2022, Art. no. 2200072. [Online]. Available: https://doi.org/10.1002/aisy.202200072

[9] U. Culha, S. O. Demir, S. Trimpe, and M. Sitti, "Learning of sub-optimal gait controllers for magnetic walking soft millirobots," in *Robot. Sci. Syst. Online Proc.*, Jul. 2020, pp. 1–9.

[10] A. Marino, B. Scaglioni, and P. Valdastri, "Reinforcement learning based control for a magnetic flexible endoscope," in *Proc. Hamlyn Symp. Med. Robot*, 2021, p. 1.

[11] M. Turan et al., "Learning to navigate endoscopic capsule robots," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 3075–3082, Jul. 2019.

[12] A. A. Shahid, D. Piga, F. Braghin, and L. Roveda, "Continuous control actions learning and adaptation for robotic manipulation through reinforcement learning," *Auton. Robots*, vol. 46, no. 3, pp. 483–498, 2022.

[13] F. Richter, R. K. Orosco, and M. C. Yip, "Open-sourced reinforcement learning environments for surgical robotics," Mar. 2019, *arXiv:1903.02090*.

[14] O. Erin et al., "Enhanced accuracy in magnetic actuation: Closed-loop control of a magnetic agent with low-error numerical magnetic model estimation," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9429–9436, Oct. 2022.

[15] S. Raval et al., "Magnetic model calibration for tetherless surgical needle manipulation using Zernike polynomial fitting," in *Proc. IEEE 21st Int. Conf. Bioinf. Bioeng. (BIBE)*, 2021, pp. 1–6.

[16] M. Fan et al., "Towards autonomous control of magnetic suture needles," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2020, pp. 2935–2942.

[17] A. Hundt et al., "'Good robot!' Efficient reinforcement learning for multi-step visual tasks with Sim to real transfer," 2019, *arXiv:1909.11730*.

[18] P. Becker-Ehmck, M. Karl, J. Peters, and P. van der Smagt, "Learning to fly via deep model-based reinforcement learning," 2020, *arxiv:2003.08876*.

[19] V. Mnih et al., "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[20] A. El Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," 2017, *arXiv:1704.02532*.

[21] A. Kendall et al., "Learning to drive in a day," 2018, *arXiv:1807.00412*.

[22] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," 2016, *arXiv:1609.05521*.

[23] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2018, pp. 4238–4245.

[24] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-end robotic reinforcement learning without reward engineering," Apr. 2019, *arXiv:1904.07854*.

[25] D. Kalashnikov et al., "QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation," 2018, *arXiv:1806.10293*.

[26] L. Roveda et al., "Model-based reinforcement learning variable impedance control for human-robot collaboration," *J. Intell. Robot. Syst.*, vol. 100, no. 2, pp. 417–433, 2020.

[27] C. D'Ettorre et al., "Automated pick-up of suturing needles for robotic surgical assistance," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 1370–1377.

[28] S. Sen, A. Garg, D. V. Gealy, S. McKinley, Y. Jen, and K. Goldberg, "Automating multi-throw multilateral surgical suturing with a mechanical needle guide and sequential convex optimization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016, pp. 4178–4185.

[29] F. Zhong, Y. Wang, Z. Wang, and Y. Liu, "Dual-arm robotic needle insertion with active tissue deformation for autonomous suturing," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2669–2676, Jul. 2019.

[30] V. M. Varier, D. K. Rajamani, N. Goldfarb, F. Tavakkolmoghaddam, A. Munawar, and G. S. Fischer, "Collaborative suturing: A reinforcement learning approach to automate hand-off task in suturing for surgical robots," in *Proc. 29th IEEE Int. Conf. Robot Human Interactive Commun. (RO-MAN)*, 2020, pp. 1380–1386.

[31] Y. Barnoy, M. O'Brien, W. Wang, and G. Hager, "Robotic surgery with lean reinforcement learning," May 2021, *arXiv:2105.01006*.

[32] L. Buesing et al., "Learning and querying fast generative models for reinforcement learning," 2018, *arXiv:1802.03006*.

[33] D. Silver et al., "Mastering chess and Shogi by self-play with a general reinforcement learning algorithm," 2017, *arXiv:1712.01815*.

[34] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 2, 2017, pp. 1173–1185.

[35] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," *J. Jpn. Inst. Electron. Packag.*, vol. 13, no. 5, pp. 413–417, Aug. 2017.

[36] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," 2019, *arXiv:1912.01603*.

[37] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, "Mastering Atari with discrete world models," 2020, *arXiv:2010.02193*.

[38] A. Byravan et al., "Imagined value gradients: Model-based policy optimization with transferable latent dynamics models," 2019, *arXiv:1910.04142*.

[39] X. Wang, W. Xiong, H. Wang, and W. Y. Wang, "Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation," Mar. 2018, *arXiv:1803.07729*.

[40] T. Weber et al., "Imagination-augmented agents for deep reinforcement learning," 2017, *arXiv:1707.06203*.

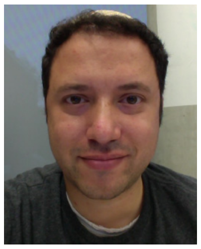
[41] W. Pryor et al., "Localization and control of magnetic suture needles in cluttered surgical site with blood and tissue," 2021, *arXiv:2105.09481*.

[42] S. Fujimoto, H. V. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[43] G. Brockman et al., "OpenAI gym," Jun. 2016, *arXiv:1606.01540*.

[44] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the Brownian motion," *Phys. Rev.*, vol. 36, no. 5, p. 823, 1930.

[45] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3599–3609.



Yotam Barnoy received the B.S. degree in computer engineering from the University of California at Irvine, Irvine, in 2001, the M.B.A. degree in finance from Bar-Ilan University in 2009, and the Ph.D. degree in computer science from Johns Hopkins in 2021. He is a Senior Algorithms Engineer with Cipia, working on driver monitoring systems using deep learning. Previously, he worked as a Realtime Engineer for Marvell Technology and ECI Telecom. His research interests include reinforcement learning, deep learning, vision, and robotics.

Onder Erin (Member, IEEE) received the B.S. degree in mechatronics engineering from Sabanci University, Istanbul, Turkey, in 2014, as Valedictorian and the Ph.D. degree in mechanical engineering from Carnegie Mellon University, PA, USA, in 2020. He is a Postdoctoral Fellow with Mechanical Engineering Department, Johns Hopkins University, MD, USA. His research interests include medical robotics, mobile and soft robotics, and medical applications of robotic systems. His doctoral studies focus on manipulation, imaging, and control of magnetic robots with a single MRI device. He was the recipient of Max-Planck Society Ph.D. Fellowship in 2017, and he conducted some of his research activities at Max-Planck-Institute for Intelligent Systems, Stuttgart, Germany.

Suraj Raval (Graduate Student Member, IEEE) received the B.Tech. degree with (first class Distinction) in mechanical engineering from Nirma University, India, in 2017, and the M.S. degree from University of Maryland at College Park, USA, in 2019, where he is currently pursuing the Ph.D. degree with the Mechanical Engineering Department. Before joining the University of Maryland at College Park as a Ph.D. student in 2020, he was employed Full-Time as a Research and Development Engineer with Actasys, Brooklyn, NY, USA, where he worked on enhancing sensor capabilities for ADAS applications. His research interests include control and estimation of robotic systems by leveraging concepts from control theory and data-driven science.

Will Pryor received the B.S. and M.S. degrees in robotics from Worcester Polytechnic Institute, Worcester, MA, USA. He is currently pursuing the Ph.D. degree with Johns Hopkins University, MD, USA. He has also worked with Verb Surgical in the areas of robot control and motion planning. His research interests include semiautonomous teleoperation for space robotics, 2D and 3D perception, and motion planning.

Lamar O. Mair (Member, IEEE) is a Materials Engineer with Weinberg Medical Physics, Inc., working on synthesis and manipulation of magnetic particles and devices ranging in size from tens of nanometers to a few centimeters.

Irving N. Weinberg (Life Member, IEEE) received the Ph.D. degree from the University of California at Irvine and the M.D. degree (clinical training) from the University of Miami and Johns Hopkins Medical School for Residency, where he practiced as a Radiologist. He was a Experimental Plasma Physicist with the University of California at Irvine. His diverse background in Technical Academics and in Medicine have provided him with the tools and inclination to design and construct medical devices. He has launched four FDA-approved products that have been used by over one million Americans with suspected or confirmed cancer. The diagnostic accuracy of these products has been documented in multiple peer-reviewed publications. Ten years ago, he transitioned to the young field of image-guided therapy and diagnosis with pulsed magnetic fields and magnetic particles. His team has accumulated over 30 patents and numerous publications. He is the President of Weinberg Medical Physics and an Advisor of many companies, including Brain Bioscience Inc., Promaxo, and Otomagnetics. He also has served as the President of Naviscan PET Systems Inc. which he founded. He is a member of advisory boards at UCLA and George Mason University.

Yancy Diaz-Mercado (Member, IEEE) received the B.S. degree *(magna cum laude)* from the University of Puerto Rico at Mayaguez in 2011, and the M.S. and Ph.D. degrees from the Georgia Institute of Technology in 2014 and 2016, respectively. He is an Assistant Professor with the Department of Mechanical Engineering, University of Maryland at College Park, College Park. He is the Director of the Collaborative Controls and Robotics Laboratory, whose research focus is on developing collaborative autonomy for multi-agent systems, robotics, and enabling human-swarm interactions. Before joining the Faculty with the University of Maryland College Park in 2018, he was a Senior Professional Engineer with the Johns Hopkins University Applied Physics Laboratory, where he was a member of the Advanced Concepts section within the Guidance, Navigation, and Controls group in the Air and Missile Defense Sector. He holds multiple patents and patent applications for his work.

Axel Krieger (Member, IEEE) received the undergraduate and master's degrees from the University of Karlsruhe, Germany, and the Doctoral degree from Johns Hopkins University, where he pioneered an MRI guided prostate biopsy robot used in more than 50 patient procedures at three hospitals. He is an Assistant Professor of Mechanical Engineering with Johns Hopkins University, focusing on the development of novel tools, image guidance, and robot-control techniques for medical robotics. Before joining Johns Hopkins in 2020, he was an Assistant Professor with the University of Maryland and a Research Professor and a Program Lead for Smart Tools with the Sheikh Zayed Institute for Pediatric Surgical Innovation, Children's National Hospital. He leads a team of students, scientists, and engineers in the research and development of robotic tools and laparoscopic devices. Projects include the development of a surgical robot called smart tissue autonomous robot and the use of 3D printing for surgical planning and patient-specific implants. He is an Inventor of more than 20 patents and patent applications. Licensees of his patents include medical device start-ups Activ Surgical and PeriCor, as well as industry leaders such as Siemens, Philips, and Intuitive Surgical. He spent several years as a Product Leader with Sentinelle Medical Inc., and Hologic Inc., developing devices and software systems from concept to FDA approval and market introduction. He is a member of the Laboratory for Computational Sensing and Robotics.

Gregory D. Hager (Fellow, IEEE) received the B.A. degree *(summa cum laude)* in mathematics and computer science from Luther College in 1983, and the M.S. and Ph.D. degrees from the University of Pennsylvania in 1986 and 1988, respectively.

He was a Fulbright Fellow with the University of Karlsruhe, and was on the Faculty of Yale University prior to joining Johns Hopkins in 1999. He is the Mandell Bellmore Professor of Computer Science with Johns Hopkins University, and holds joint appointments with the Department of Electrical and Computer Engineering and the Department of Mechanical Engineering. He is the Founding Director of the Johns Hopkins Malone Center for Engineering in Healthcare, an interdisciplinary research center aimed at developing innovative healthcare technology and systems. He is known for his research on collaborative and vision-based robotics, time-series analysis of image data, and medical applications of image analysis and robotics. He has published more than 300 articles and books on these topics. In 2014, he was awarded a Hans Fischer Fellowship at the Technical University of Munich's Institute of Advanced Study, where he also holds an appointment in computer science. He has been a member of numerous prominent review committees and panels on Artificial Intelligence (AI), including Stanford University's inaugural "100 Year Study on Artificial Intelligence;" a roundtable on AI and foreign policy held by the National Academies of Science, Engineering, and Medicine; and a panel at the 2018 American Association for the Advancement of Science (AAAS) annual meeting on "Artificial Intelligence: Augmenting Not Replacing People." He co-chaired the 2015 review of the Networking and Information Technology Research and Development Program for the President's Council of Advisors on Science and Technology. He has served on the editorial boards of IEEE TRANSACTIONS ON ROBOTICS, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, and *International Journal of Computer Vision*. He has served as the Deputy Director of the NSF Engineering Research Center for Surgical Systems and Technology and as the Chair of the Department of Computer Science from 2010 to 2015. His many contributions to the field of vision-based robotics has earned him status as an IEEE Fellow. Additionally, he has been named a Fellow of the MICCAI Society, the Association of Computing Machinery, the American Institute for Medical and Biological Engineering, and AAAS. He is a member of the National Science Foundation's Computer and Information Science and Engineering Advisory Committee, a member of the Governing Board of the International Federation of Robotics Research, and is the Chair Emeritus of the Computing Community Consortium, as well as a Board Member of the Computing Research Association.