IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Learning Nonprehensile Dynamic Manipulation: Sim2real Vision-based Policy with a Surgical Robot

Radian Gondokaryono[1,2], Mustafa Haiderbhai[1,2], Sai Aneesh Suryadevara[1,3], Lueder A. Kahrs[1,4]

***Abstract*—Surgical tasks such as tissue retraction, tissue exposure, and needle suturing remain challenging in autonomous surgical robotics. One challenge in these tasks is nonprehensile manipulation such as pushing tissue, pressing cloth, and needle threading. In this work, we isolate the problem of nonprehensile manipulation by implementing a vision-based reinforcement learning agent for rolling a block, a task that has complex dynamics interactions, small scale objects, and a narrow field of view. We train agents in simulation with a reward formulation that encourages efficient and safe learning, domain randomization that allows for robust sim2real transfer, and a recurrent memory layer that enables reasoning about randomized dynamics parameters. We successfully transfer our agents from simulation to real and show robust execution of our vision-based policy with a 96.3% success rate. We analyze and discuss the success rate, trajectories, and recovery behaviours for various models that are either using the recurrent memory layer or are trained with a difficult physics environment. Further project information is available at https://medcvr.utm.utoronto.ca/ral2023-rollblock.html.**

***Index Terms*—Surgical Robotics: Laparoscopy, Reinforcement Learning, Visual Servoing**

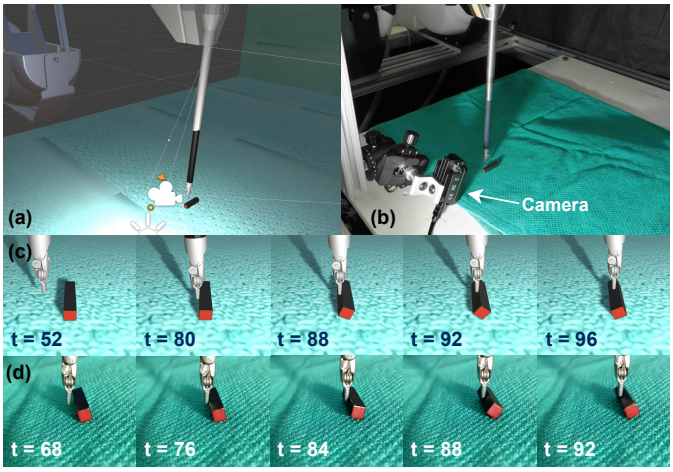

Fig. 1: Simulation (a) and real setup (b) of the da Vinci Research Kit and environment for the task of rolling a miniature block. Time series of the block rolling trajectory with a vision-based autonomous agent in simulation (c) and real (d).

I. INTRODUCTION

FINE manipulation skills are a critical aspect of surgical robotics tasks such as needle insertion, knot tying, and tissue retraction [1]. In recent years, there has been great success in applying deep learning methods such as reinforcement learning (RL) to learn these complex autonomous behaviors. These advances can be attributed to the development of simulations for surgical robots such as the da Vinci Research Kit (dVRK) [2]. One common challenge in RL for robotics is that agents trained in the simulation must be transferable to real robot scenarios. In our previous work [3], we created a novel simulation for the dVRK inside Unity3D and showed that a robust visuomotor policy could be trained efficiently in simulation and transferred to real through our *sim2real* training pipeline using Domain Randomization.

Another advancement in *sim2real* RL for automating a surgical task uses visual observations, Simulation Open Framework Architecture simulator [4], and domain adaptation to train a policy agent for tissue retraction which achieves a success rate of 50% [5]. While one issue to address is real-to-sim matching of deformable materials, we first investigate how an agent should reason about dynamics, such as friction and mass, which are principal components of surgical scenes. Furthermore, the typical surgical scene deals with small-scale objects such as needles, sutures, and tissue where small interactions of frictions, masses, and contacts have large effects on the task.

To experiment, we take inspiration from nonprehensile dynamic manipulation tasks [6] where the object being manipulated does not strictly follow the motion of the end-effector. Rather, it utilizes the object's dynamic interaction (e.g. force and friction) with the environment and the end-effector to accomplish the task. We extend our previous work of pushing a block to accomplish a task of rolling a miniature block (see Fig. 1). This task requires finer manipulation at the level of human expertise due to complex interactions of the ground and object during rolling and the large difference in size and inertia of the object relative to the robot manipulator. The training procedure is modified to include domain randomization on dynamics parameters which are unobservable by the input image, and the model is extended to include a recurrent layer

Manuscript received: April, 24, 2023; Revised July, 19, 2023; Accepted August, 14, 2023.

This paper was recommended for publication by Editor Jens Kober upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), RGPIN-2020-05833, and Mitacs.

[1]R. Gondokaryono, M. Haiderbhai, S. A. Suryadevara, and L. A. Kahrs are with the Medical Computer Vision and Robotics Lab, University of Toronto, Canada radian.gondokaryono@mail.utoronto.ca,

[2]R. Gondokaryono and M. Haiderbhai are with the Department of Computer Science, University of Toronto, Canada and also with The Wilfred and Joyce Posluns CIGITI, Sickkids Hospital, Toronto, Canada

[3]S. A. Suryadevara is with the Department of Mechanical Engineering, Indian Institute of Technology Bombay, India

[4]L. A. Kahrs is with the Department of Mathematical and Computational Sciences, University of Toronto Mississauga, Canada and the Institute of Biomedical Engineering, University of Toronto, Canada

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

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

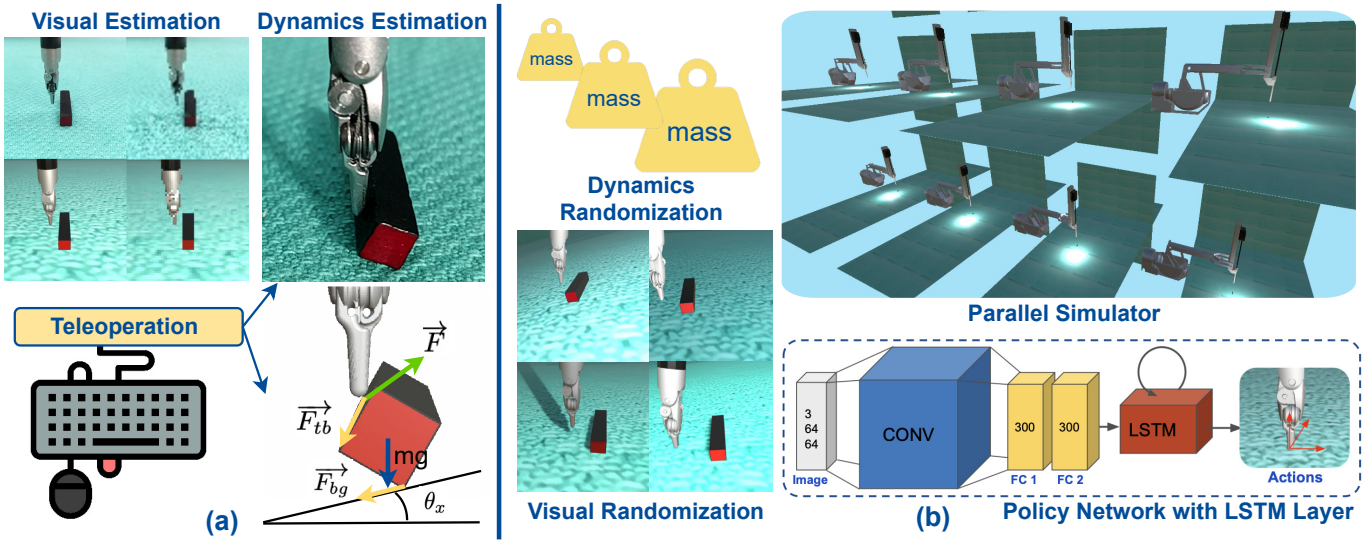

Fig. 2: Overview of the *sim2real* learning method for rolling a block. a) Estimation procedure to obtain simulation visual and dynamic environment parameters. b) Training methods including visual and dynamics randomization, a parallel simulator, and a policy network with a long-short term memory layer.

to reason about unobservable dynamics which are different in each episode. We show that our simulation, model, and training procedure can *sim2real* transfer substantially more dynamic tasks. This work presents the following contributions:

- The first visuomotor autonomous agent that executes, with a high success rate, a small scale nonprehensile task with the physical dVRK robot.
- Performance and behaviour analysis of the autonomous trajectory on a challenging ground inclination setup, which tests the dynamics understanding of the agent.
- A reward function for rolling a block that facilitates training times and creates a robust policy.

II. RELATED WORKS

A. Fine Manipulation Tasks

The task of rolling a block falls under the category of dynamic nonprehensile manipulation defined as manipulation of an object without a force closure grasp [6] and has been well studied from conventional motion planning and control perspective [7]. Recently, works on deep reinforcement learning for complex in-hand dexterous manipulation tasks have shown success in learning cube manipulation techniques such as rotating and sliding to achieve a desired object pose with a tri-finger robot [8]. Andrychowicz et al. [9] study similar dexterous in-hand manipulation with a robotic hand but propose a recurrent layer for memory in the policy network to reason about dynamic parameters. Both these works train in a visuallly and dynamically realistic simulation using Proximal Policy Optimization [10] and rely on Domain Randomization [11] to transfer policies to a physical setup. Unlike our task, their work chooses to first estimate the pose of the object, or keypoints of the object, using multiple cameras with data from the simulation and trains a policy using vector-based observations. Keypoint and pose estimation methods assume a precise robot to camera calibration method which is not feasible in most surgical applications. In contrast, our work relies on a single camera close-up view, and trains a policy end-to-end with visual observations. The reinforcement learning aspect of our work uses similar methods, employing Domain Randomization techniques for effective *sim2real* transfer.

In surgical robotics, there have been a few recent methods that deal with complex manipulation tasks. Wilcox et al. [12] apply perception techniques coupled with trajectory optimization to reposition a needle using two dVRK end-effectors. Similarly, Hwang et al. [13] create an optimized trajectory for peg transfer, a standard laparoscopic training task, and are able to complete robotic peg transfers with efficient movements surpassing human performance. Both works are great examples of dVRK tasks being solved in real scenarios. The limitation of these approaches is that, similar to [9], they rely on perception techniques and precise camera calibration methods without which the optimized trajectory would not be able to complete the task. Such methods do not generalize well to more complex tasks, such as the manipulation of soft materials, nonprehensile tasks, or other vision-based navigation in surgical applications where calibration might be difficult.

B. Surgical Robotics Simulation

Our work is closely related to concurrent work in advancing simulations to enable reinforcement learning and other techniques in autonomous surgical robotics. Most recently, novel simulators such as SurRoL [14] have been released, that build a simulation using PyBullet [15] and include manipulation tasks of small rigid objects such as needles. Their work is recently extended with more realistic rendering techniques, and learning from human demonstration, which allows for efficient policy learning. The limitation is that they do not explore *sim2real* transfer of learned policies to real scenarios. Other simulators, such as AMBF [16], also provide a simulation for the dVRK, but training vision-based policies using reinforcement learning is not explored. Unity3D [17] provides a realistic rendering pipeline and Unity ML-Agents [18]

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

provides sim2real techniques such as Domain Randomization and state-of-the art reinforcement learning training algorithms. These features facilitate our work which focuses heavily on *sim2real* transfer of learned policies. Furthermore, the open-community GUI-based editor in Unity allows quick development of the reinforcement learning tasks. Currently, there is no work that specifically deals with small-scale autonomous manipulation with the dVRK as seen in this paper.

III. METHODOLOGY

In this work, the goal is to autonomously execute a non-prehensile dynamic task which is to roll a miniature block with a small mass using a close-up camera view. The method starts by building a simulation of the task and aims to produce a robust policy to evaluate on the real dVRK surgical robot mounted with the Large Needle Driver tool.

A. Problem Formulation

We choose to formulate our problem as a visual servoing task where kinematic errors [19] produced by cable tension [20], friction hysteresis, and backlash from the tool in a cannula [21] are compensated by commanding tool-tip Cartesian increments to the current pose based on an input image from one RGB camera. This formulation is analogous to teleoperation, the original control scheme for the commercial system, where users operate the robotic arms from the Master Console that has a first-person camera display. In this work, we fix the rotation of the tool-tip to always be pointing down, and use inverse kinematics to solve the joint state positions. More details on the modular control architecture are explained in our previous works [3], [22].

B. Task Setup

The task consists of a prism-shaped wooden block of size 25.45, 5.11, and 5.14 mm, placed on top of a green cloth. The block color is black with one face painted red. Fig. 1a-b shows the task setup in both simulation and real. This setup is chosen because the size, mass, and friction create a complex scenario where the tool-tip interacting with the block could either push or roll the block. Rolling the block requires precise manipulation with the correct contact location, force direction and magnitude ($\overrightarrow{F}$ in Fig. 2a). Incorrect contact or force would easily lead to sliding or twisting (rotation along the shorter axes) of the block. The RGB camera (OAK-1, OpenCV.AI Corporation) observes the scene from a side perspective.

To simulate our task, we extend our previous work [3] where we build a dVRK simulation in Unity3D [17] which includes parallel training and system-identified robot dynamics [20]. The visual and dynamic environment parameters are determined through an estimation process. This process is an approximation where the goal is not to accurately identify the parameters, but to estimate a starting point for Domain Randomization (Section III-E). Fig. 2a illustrates the estimation process. We match the camera position by aligning the edges of the block when placed in the center of the field of view. The appearance of the cloth is replicated by creating a texture from a picture of the scene and the color of the tool and block are estimated. Environment dynamic properties such as friction coefficients of the block, tool, and cloth as well as block mass, are estimated by executing a trajectory by user teleoperation in the physical setup that causes a roll, and replaying this trajectory in simulation until a similar behavior is observed. These friction coefficients determine the resulting magnitudes of $\overrightarrow{F_{tb}}$ and $\overrightarrow{F_{bg}}$ in Fig. 2a. For this work, it is satisfactory to set static and dynamic friction coefficients as the same value. Similar executions are also collected for sliding and twisting behaviors.

C. Reward Function Design

The reward function is designed to encourage the agent to roll the block once per episode but allows for multiple rolls to occur in real where the episode does not end. Our reward function consists of dense and sparse components as motivated by [23] which allows the training to be stable, faster, and alleviates the need for user demonstrations [24]. Geometric and physical parameters considered for the reward calculations are visualized in Fig. 3a. The reward components are:

1. Roll reward: $r_r(s) = w_r \frac{\phi(s)}{90^\circ}$ subject to $0 \leq \phi(s) \leq 90^\circ$ where w_r is a weight, and $\phi(s)$ is the current roll angle in degrees between the ground normal vector (y_g) and the block up normal vector (y_b). y_b is changed to another surface normal vector facing gravity if the block has rotated 90° in the $\phi(s)$ negative direction, resulting in $\phi(s) = 0$.

2. Distance reward: $r_d(s) = w_d \min\left(e^{-a(d_t(s)-d_b)}, 1\right)$ uses an exponential distance function where w_d is a weight, $d_t(s)$ is the current distance between the tool-tip and the center of the block, and a is an exponential scaling factor. d_b is the spherical radius and is calculated by $d_b = \sqrt{(pb_y/2)^2 + (pb_z/2)^2}$ where b_y, b_z, and p are block size y, block size z, and padding respectively. This allows the distance reward term to have a maximum value of w_d when $d_t(s) = d_b$ which encourages the tool-tip to go near the block surface but, for higher rewards, needs to find the proper interaction which increases the roll angle, $\phi(s)$. All distances are in meters.

3. Reach goal reward: $r_g(s) = n_{rg}(s)$ where $n_{rg}(s)$ is 0, otherwise 1 when $\phi(s) > 80$ and the tool has reached a chosen distance, d_g, left of the block. The condition $n_{rg} = 1$ also triggers the end of an episode. This reward encourages multiple rolls when the episode does not end because the tool is forced to return to the start position after finishing a roll.

4. Penalties: $r_p(s) = -w_l \frac{t(s)}{T} - w_{ht} n_{ht}(s) - w_{hg} n_{hg}(s)$ where w_l, w_{ht}, and w_{hg} are weights, $t(s)$ is the current step count,

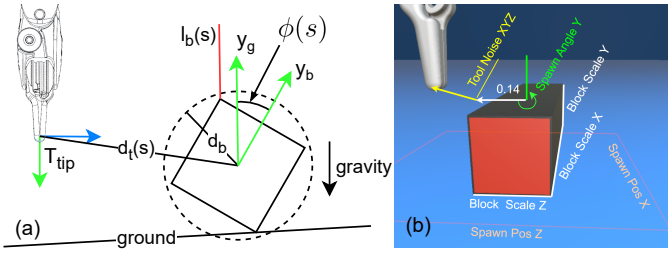

Fig. 3: a) Parameters of the reward function. b) Visualization of specific environment parameters.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

and T is the number of maximum steps in an episode. $n_{hg}(s)$ is the current number of times the tool mesh colliders hit the ground colliders and $n_{ht}(s)$ is the current number of times the tool colliders hit the block colliders when the tool-tip position is on the right side of the boundary $l_b(s)$. Colliding multiple times, accumulates $n_{hg}(s)$ or $n_{ht}(s)$ which results in an accumulative negative reward. These penalties discourage dangerous behaviours that might damage the system.

The total reward is calculated by $r(s) = r_r(s) + r_d(s) + r_g(s) + r_p(s)$. We recalculate the reward at each step. We choose $a = 2$, $p = 1.10$, $d_g = 0.14$, $T = 600$, $w_r = 2$, $w_d = 1$, $w_l = 1$, $w_{ht} = 0.5$, and $w_{hg} = 0.5$. Each simulation environment in Unity runs at a default 50 Hz, meaning that each time step is 0.02 seconds in terms of physics calculations. The policy network also takes image observations and actions at the same time step.

TABLE I: Environment parameters in Unity. Parameters with ranges are domain randomized using a uniform sampler. Metric units are scaled by 20 of the real setup value.

Parameter	Value/Range	Parameter	Value/Range
Cam Pos Y [m]	[0.09, 1.22]	Ground B	[0.90, 1.00]
Cam Pos X [m]	[-1.50, -1.15]	Ground Height [m]	[0.15, 0.25]
Cam Pos Z [m]	[-1.27, -1.07]	Ground Friction Coef.	0.33
Cam Rot X [°]	[26, 33]	Tool Friction Coef.	0.20
Cam Rot Y [°]	[86, 94]	Block Scale X [m]	[0.50, 0.53]
Cam Rot Z [°]	0	Block Scale Y, Z [m]	[0.10, 0.115]
Light Intensity	[10, 60]	Block Mass [gr]	[0.40, 0.90]
Light Pos Y [m]	[0.50, 3.50]	Block Color H, S	0
Light Pos X [m]	[-3.80, -1.80]	Block Color V	[0.05, 0.18]
Light Pos Z [m]	[-3.50, -0.80]	Block Side Color R	[0.42, 0.72]
Ground R	[0.65, 1.00]	Block Side Color G	[0, 0.20]
Ground G	[0.90, 1.00]	Block Side Color B	0.075

D. Policy Architecture and Training

We employ a policy network architecture provided by Unity ML-Agents [18] which uses a ResNet-50 [25] visual encoder (not pretrained) followed by two fully connected layers of 300 units (Fig. 2b). Block position and rotation are observable from images, but our training task includes randomized unobservable parameters (Section III-E), such as friction and mass, that effect the success of the task. To account for this, we experiment with a long short-term memory (LSTM) [26] layer of memory size of 256. This recurrent memory component allows the network to reason about unobservable dynamics in the scene by first interacting with the environment at the beginning of each episode. Our policy and value network take a 64x64 pixel image as input and outputs the xyz position increments through discrete action branches. Each discrete action branch has 9 bins, equally spaced from -1 to 1, and are multiplied by a speed modifier to produce Cartesian increments. Discrete actions were chosen following the recommendation and experiments [9] that performs better than continuous actions probably due to a more expressive Gaussian probability distribution.

We train using Proximal Policy Optimization with the hyperparameters $\epsilon = 0.2$, $\beta = 0.001$, $\lambda_d = 0.95$, number of epochs = 3, batch size = 400, buffer size = 14,000, learning rate = 0.0001, time horizon = 300, reward γ = 0.99, reward strength = 1, and max_steps = 25,000,000. max_steps takes into account all steps taken from each of the 12 simulation environments running in parallel. All models were trained using a cluster (Digital Research Alliance of Canada) which finishes the 25M steps in 24-28 hours.

E. Domain Randomization and Sim2real Transfer

The estimation procedure in Section III-B sets the mean values for Domain Randomization (DR). The ranges are chosen based on the specific parameter and estimation procedure, and the ability of the agent to train. For example, the block size can have a small range as a caliper is used to measure, however, the camera position might need larger ranges as it is difficult to ensure the exact position in simulation vs. real. These ranges ensure that the task is still solvable, such as ensuring the block is still observable in the camera view. A sample of different visual randomizations generated during DR is illustrated in Fig. 2b. Due to the large number of randomizations, agent training can fail or take extremely long. To combat this we utilize curriculum learning [27], and start with small ranges that incrementally increase as the next lesson is triggered.

Table I lists the randomizations of our environment and Table II lists the randomizations with a curriculum learning lesson plan. All randomizations use a uniform sampler and the next lesson is only triggered when the reward reaches a value of 2.7 for a minimal 300 episodes (600 for block friction coefficient). The reward value directly corresponds to the task success rate due to the defined reward terms in the previous section. After the block is spawned at position xyz, the tool-tip position is spawned 0.14 metric units z behind the block with added tool noise xyz (Fig. 3b).

We train different models using the same randomizations but with different amounts of ground inclination randomization, to create steepness in the direction of the roll. The three models we trained are FC_T0, FCM_T0, and FCM_T20. T0 denotes a model trained with no ground inclination randomization and T20 denotes a model trained with 0 minimum inclination randomization and as the maximum inclination with increasing lessons: 1, 3, 5, 7, 10, 13, 16, and 20 degrees. 20 degrees is the chosen maximum incline as at 22 degrees, the block almost always rolls down the incline after any interaction with the tool. FCM is the policy network with the LSTM layer (Fig. 2b) and FC is a policy network without the LSTM layer but replaced with a 300 unit fully connected layer.

F. Experiment Setup

Our experiment setup aims to test the *sim2real* performance of the different models on three different ground inclinations: 0, 10, and 15 degrees. We stop at 15 degrees as any larger inclination causes the block to have a high chance of rolling down due to its own mass and gravity. Fig. 4a displays the different inclinations as seen from the camera's perspective. The light source is placed behind the camera to the right, to ensure the red face of our block is clearly seen. A protractor is used to measure the inclination angle.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

TABLE II: Domain randomized parameters with curriculum learning. Metric units are scaled by 20 of the real setup value.

Lesson	Spawn Pos X [m]	Spawn Pos Z [m]	Spawn Angle Y [°]	Tool Noise X [m]	Tool Noise Y [m]	Tool Noise Z [m]	Block Friction Coef.
0	[-0.06, 0.06]	[-0.03, 0.03]	[-3, 3]	[-0.08, 0.08]	[-0.02, 0.02]	[-0.04, 0.04]	[0.18, 0.20]
1	[-0.12, 0.12]	[-0.06, 0.06]	[-6, 6]	[-0.15, 0.15]	[-0.04, 0.04]	[-0.07, 0.07]	[0.14, 0.16]
2	[-0.25, 0.25]	[-0.12, 0.12]	[-9, 9]	[-0.258, 0.258]		[-0.07, 0.17]	[0.12, 0.14]
3		[-0.18, 0.18]	[-12, 12]				[0.10, 0.12]
4		[-0.225, 0.225]	[-15, 15]				
5			[-18, 18]				

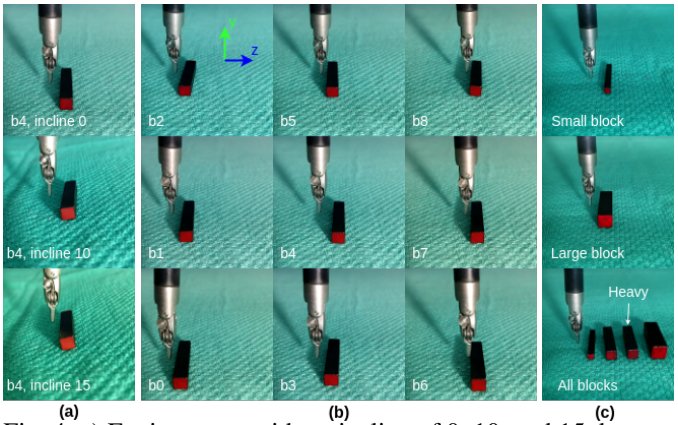

Fig. 4: a) Environment with an incline of 0, 10, and 15 degrees with the block at the central start position b4. b) 9 different tool/block start positions b0 - b8. c) Different block sizes.

Our testing protocol, for each incline, involves 9 different starting positions based on a 3x3 grid of x-z tool tip positions, with the block at a small distance in front of the tool. The positions are specified by the robots x-z coordinates and the height is adjusted, for each incline, to approximately the height of the block with a random noise of 1 mm. A homing script is used to ensure the robot starts at the same position for each run. Fig. 4b displays all 9 starting positions at 0° incline.

Each run at consists of 5 trials for each of the 9 starting positions, totaling 45 trials to evaluate each model at each incline. In conducting these 5 trials, a small rotation of the block along the y axis is added for randomness. The number of successful rolls is recorded for each trial. The policy network takes input images and gives actions to the real robot at 30 Hz due to the frequency limit of streaming images from the camera software implementation. Details about the *sim2real* execution of the agent can be found in our previous publications [3], [22].

IV. RESULTS

In this section, we report our findings that highlight the benefits of the LSTM memory layer and ground incline randomization that result in the best performance with a *sim2real* policy capable of multiple rolls on all tested inclines and starting positions relative to the camera.

A. Training

Fig. 5 shows the simulation training results of the three models FC_T0, FCM_T0, and FCM_T20 with final average rewards of 2.638, 2.813, and 2.744. The large variability in reward around 4M to 8M steps corresponds to when the lessons increase in difficulty and the final lesson is triggered at the latest 16M steps.

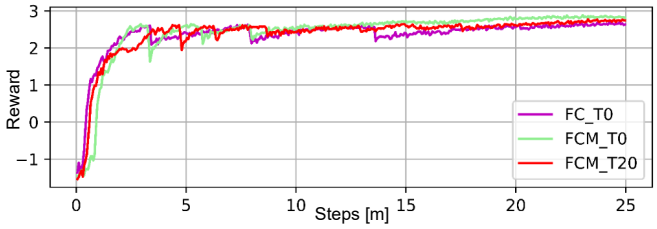

Fig. 5: Simulation training results of the three models FC_T0, FCM_T0, and FCM_T20.

TABLE III: Success rate (SR) to achieve at least 1 or 2 rolls in each trial out of 45 trials (9 starting positions and 5 trials each) of every model at varying inclinations of 0, 10, and 15 degrees.

Model	Minimal 1 Roll SR (%)				Minimal 2 Roll SR (%)			
	0°	10°	15°	Mean	0°	10°	15°	Mean
FC_T0	93.3	35.6	17.8	48.9	**84.4**	13.3	0.0	32.6
FCM_T0	**97.8**	73.3	22.2	64.4	**77.8**	20.0	0.0	32.6
FCM_T20	**100.0**	**100.0**	88.9	**96.3**	**73.3**	**73.3**	55.6	**67.4**

The model with a memory layer performs better (higher mean reward) than the model without the memory layer (FC_T0). During development until these final models are chosen, the training is stable (no complete failures) and all curriculum lessons have been triggered.

B. Sim2real Experiments

Table III shows the results of the experimental protocol evaluating the success rate (SR) of achieving a minimum of 1 and 2 rolls on the real setup in one trial on an average over 45 trials for every model on three different inclinations 0, 10, and 15 degrees. FCM_T20, the model with an LSTM layer and incline randomization training, achieves a minimal 1 roll with 100% SR on 0 and 10° inclines and 88.9% SR on the most difficult 15° incline. Taking the average of all inclines, this model achieves a 96.3% SR. This model outperforms both FC_T0 and FCM_T0 at all inclines. At 0° incline, the model with LSTM memory layer, FCM_T0, performs better (97.8% SR) than the model without memory FC_T0 (93.3% SR). Both models however performed poorly on 10 and 15° inclines. Similar trends are observed for a minimum of 2 rolls in a trial with the best average SR of 67.4% for FCM_T20.

Table IV shows the success rate of FCM_T20 for different sized blocks, same length but different cross sections, on the 0

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

TABLE IV: FCM_T20 tested with different, not observed in training, block sizes and masses on the 0 degree incline.

Block Types	Minimal N Roll SR (%)			
	N=1	2	3	4
Small	**80.0**	40.0	17.8	11.1
Medium (Original Block, 0.50 gr)	100.0	73.3	22.2	0.0
Large	**77.8**	55.6	22.2	4.4
Medium Heavy (0.93 gr)	**68.9**	44.4	11.1	0.0

TABLE V: The slope angle of the y-z projection of the roll block trajectory for various models at different real setup inclinations 0, 10, and 15 degrees for an average over 9 trials.

Model	Mean Trajectory Slope (°)			
	0°	10°	15°	Mean
FC_T0	22.6	21.1	22.3	**22.0**
FCM_T0	18.4	17.2	17.0	**17.5**
FCM_T20	21.7	27.6	29.2	26.2

degree incline environment. The agent, which was not trained on these sizes, was able to achieve 80.0% SR on a small 3x3 mm block and 77.6% on a large 7x7 mm block. Additionally, the agent achieved a 68.9% SR on a heavier block with the same size. All blocks are shown in Fig. 5c.

Fig. 6 shows real tool-tip trajectories, projected onto the yz plane, all starting from a common relative origin (0, 0) and accumulating the discrete actions at each step. The y-direction points upwards and positive z is the forward direction during rolling from left to right (see Fig. 1). The trajectory first goes to a lower y until the RL model identifies, with the visual observation, that the tool makes contact with the block and proceeds to do an upward and forward motion to roll the block. The completion of the roll is observed when a maximum y is achieved and then moves in the z-negative direction.

Fig. 6a shows FCM_T0's trajectory has a constant slope angle when rolling the block over different inclinations 0, 10, and 15°. Fig. 6b shows how FCM_T20 is able to adapt and increase the trajectory slope for a steeper 15° incline (dark red line). In Fig. 7a, we compare how FCM_T20's trajectory slope (dark red line) is larger (steeper) than FCM_T0's trajectory slope (blue line) at 15° incline. A key observation is that FCM_T20's trajectory (red line) accomplishes a roll with only a 2.5 mm difference in height change (-1.5 to 1 mm in z). These findings are also summarized in Table V which is the slope of the y-z projection trajectories of the 3 different models on 3 different inclines for an average over 9 trials (Each trial at each starting position b0-b8). The T0 models which did not have inclination randomization do not change the slope angle to different inclines and have a constant slope. However, the value of this constant slope is different between the two models, 22.0° for FC_T0 and 17.5° for FCM_T0.

Fig. 7b shows the 3D trajectory of FCM_T20 at 15° incline.

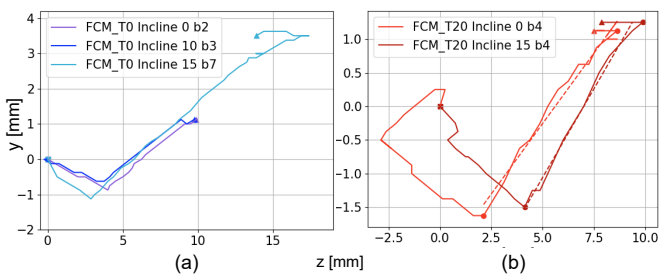

Fig. 6: y-z projection of trajectories from discrete actions accumulation for a) FCM_T0 at inclines 0, 10, and 15° b) FCM_T20 at incline 0 and 15°. Dotted lines depict the regressed slope using values between the circles. Start positions, squares, are superimposed to a common origin [0, 0].

All models at different trials/inclines have learned a similar shape trajectory. We further include other successful behaviors, in Fig. 8a-d, such as multiple rolls, fall and recover roll, a motion with two rolls, and rolling a small block. Fig. 8e also shows the main failure case, pushing with a y-axis rotation.

V. DISCUSSION

Our findings show how the proposed method can produce a robust agent to achieve a fine manipulation task where dynamics parameters play an important role.

Generalization, recovery behaviors, and failure cases. Table IV shows how our model is able to generalize to different sized blocks without previously being trained in that scenario. The drop in success rate is expected but still achieves an acceptable 80%. Fig. 4c shows the very small size of the 3x3 mm block compared to the tool. In Fig. 8b, we see a failure case and recovery behavior when the block is about to complete one roll, the block falls down the incline and the agent, not trained in this situation, can reason to follow the block and start rolling again from the left. This block falling, also seen during other trials, is due to either the stochastic nature of the ground-block friction, the tool not completing the entire roll trajectory, or the tool slightly interacts with the top of the block after finishing a trajectory. The main failure case, shown in Fig. 8d, is when the agent only pushes the block and rotates it about the y angle. Both of these failure cases are probably due to the combination of the 64x64 low resolution image, unseen edges from the black color, no depth understanding, and/or stochastic nature of the friction. Each of these limitations to improve the performance is a good starting point of exploration that we will consider in future work.

What challenges occur in a narrow field of view task? We experimented with a larger resolution of 128x128 in hopes of better understanding of the block edges and for rolling,

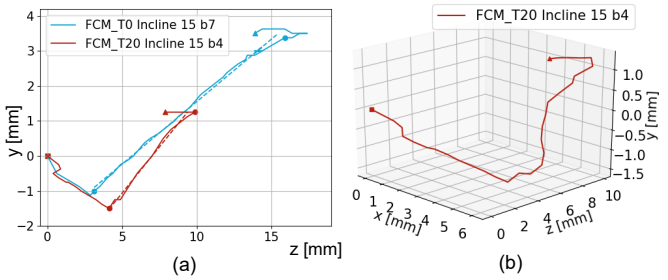

Fig. 7: a) y-z projection of trajectories from discrete actions accumulation for FCM_T0 vs. FCM_T20 at a 15° incline. Dotted lines depict the regressed slope using values between circles. b) 3D trajectory of FCM_T20 at 15°.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

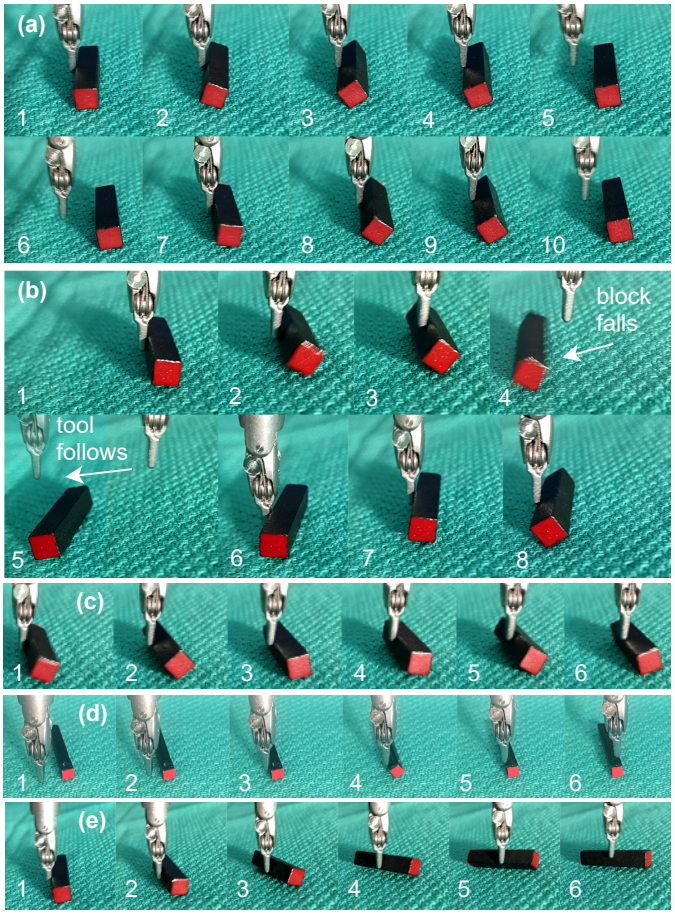

Fig. 8: Real setup trajectories seen from camera view top to bottom: a) multiple rolls, b) fall and recover roll, c) one motion double roll, d) small block roll, and e) failure push.

but failed to achieve robust *sim2real* as noise and lighting conditions highly affect the agent's domain adaptation to the real scene. This is accentuated by the metallic texture of the tool that was difficult to match by changing parameters in the simulator. The real setup works when a bright light is illuminating the tool to achieve a shiny texture.

What causes the decrease in success rate for more than one roll? After the policy agent rolls the block once on the real setup, there is a possibility the resulting y angle is large where the red face is not perpendicular to the camera. This situation does not occur in simulation because the real setup ground friction is stochastic due to the uneven surfaces and/or the combination of sim2real gaps which cause the trajectory to twist the y angle of the block. To alleviate this, we suggest to employ, at each time step, stochastic noise on the forces as in [9] and increase the spawn angle y in Table II to include this case in the training data distribution.

Why do we choose to output the Cartesian increment for the policy network? By giving an increment instead of absolute position, the agent does not need feedback of the current robot position, and can solely rely on image for spatial reasoning. This assumption simplifies the task as feedback can be noisy and is affected by underlying dynamics of the robot. Our results show longer training times for learning high fidelity dynamics as compared to previous tasks [3] and suggests learning joint level control, could potentially make the training longer with no benefit on success rate. Another recent publication states having the agent control a high-level policy is better for *sim2real* transfer [28].

Were there variations of the reward function? The roll term reward originally accounted for multiple rolls with the formulation: $r_r(s) = w_r \frac{\phi(s)}{90^\circ} + n_r(s)$ subject to $n_r(s) \leq n_{max}$ where $n_r(s)$ is the number of rolls, and n_{max} is the maximum number of rolls. $n_r(s)$ is incremented every time $\phi(s) > 80$ where $\phi(s)$ is then reset to 0. We experimented with $n_{max} = 3$ but resulted in worse performance probably due to an unbalanced data distribution because the resulting y angle is dependent on how the agent previously rolls the block.

How does this method scale to real scenarios? The general trend of simulators seem to provide rendering improvements such that, in the future, complex visual scenarios such as specularities, blood, and smoke can be reproduced. We experienced the difficulties in using domain randomization (DR) because it needs to adjust each individual parameter while also keeping a valid scene based on the combination of parameters. More realistic rendering techniques might have many parameters. While [9] has tried to address this by doing an automated curriculum for DR, we prefer to explore other methods of sim2real techniques that rely on matching the visual rendering of the simulation with real videos while also providing noise parameters (smaller range DR) to adjust the simulation style that encompasses any further sim2real gap. An alternative to simulation rendering is augmenting the 2D image for blood and smoke [29].

What are the limitations and future work? One of the main limitations of our work is that our initial task setup involves user estimation of the real scene to match the simulation. We observed moments during our work where a small error that is unaddressed can cause task failure. Recent works have shown that a method can optimize hidden parameters by minimizing the difference between simulated and real trajectories [30]. Bridging between rendered images in simulation and real images, recent works in image-to-image translation could be added [31]. One benefit of such an approach is that the resulting domain randomization ranges can be much smaller, as our parameters are an estimate of the real values. Another limitation is how the 3D trajectory (Fig 7b) tends to go toward the red face. It seems the agent learns the contrast between the red and black features. For our case, this did not affect the performance, however, it will be an interesting investigation to use 3D input observations. Additionally, our camera is running at 30 Hz while the agent was trained in simulation with 50 Hz physics step and observation. In the future, we will test whether performance of the model has been affected by a time-dilation between the sequence of input images.

VI. CONCLUSION

In this work, we explore autonomy of a fine manipulation task: Rolling a miniature block with a surgical robot. We train a robust vision-based policy in simulation and *sim2real* transfer our agents to a real setup. Our results show that we can achieve 96.3% (up to 100%) *sim2real* success, robustly manipulating the block to roll. Our work showcases the importance

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

of Domain Randomization for successful *sim2real* transfer, highlighting the special importance of simulating the physical behavior of rolling as well as inclination randomization and how it can be used to increase the robustness of the roll policy. We experiment between fully connected networks and LSTM layers, and find that LSTMs perform consistently better both in simulation and the real setup due to the ability to reason about randomized dynamic parameters which are unobservable to the input image. In the future, separate analysis of perception and control seems necessary, to isolate if certain failures are related to visual or dynamic parameter differences. One of the main limitations of our work is the need to manually estimate the visual and dynamical properties of the scene, which we hope to find alternatives for in the future to automate the process. The block rolling task proves that the dVRK surgical robot is capable of fine manipulation tasks with visuomotor policies that reasons about environment dynamics. Furthermore, this task is also a stepping stone for further autonomous surgical tasks and not intended for use in a surgical intervention.

REFERENCES

[1] C. D'Ettorre, A. Mariani, A. Stilli, F. R. y Baena, P. Valdastri, A. Deguet, P. Kazanzides, R. H. Taylor, G. S. Fischer, *et al.*, "Accelerating surgical robotics research: A review of 10 years with the da vinci research kit," *IEEE Robotics & Automation Magazine*, vol. 28, no. 4, pp. 56–78, 2021.

[2] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An open-source research kit for the da vinci® surgical system," *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 6434–6439.

[3] M. Haiderbhai, R. Gondokaryono, T. Looi, J. M. Drake, and L. A. Kahrs, "Robust sim2real transfer with the da vinci research kit: A study on camera, lighting, and physics domain randomization," *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3429–3435.

[4] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, *et al.*, "Sofa: A multi-model framework for interactive physical simulation," *Soft tissue biomechanical modeling for computer assisted surgery*, pp. 283–321, 2012.

[5] P. M. Scheikl, E. Tagliabue, B. Gyenes, M. Wagner, D. Dall'Alba, P. Fiorini, and F. Mathis-Ullrich, "Sim-to-real transfer for visual reinforcement learning of deformable object manipulation for robot-assisted surgery," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 560–567, 2022.

[6] K. M. Lynch and M. T. Mason, "Dynamic nonprehensile manipulation: Controllability, planning, and experiments," *The International Journal of Robotics Research*, vol. 18, no. 1, pp. 64–92, 1999.

[7] J. Z. Woodruff and K. M. Lynch, "Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks," *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4066–4073.

[8] A. Allshire, M. MittaI, V. Lodaya, V. Makoviychuk, D. Makoviichuk, F. Widmaier, M. Wüthrich, S. Bauer, A. Handa, and A. Garg, "Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger," *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 802–11 809.

[9] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

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

[11] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[12] A. Wilcox, J. Kerr, B. Thananjeyan, J. Ichnowski, M. Hwang, S. Paradis, D. Fer, and K. Goldberg, "Learning to localize, grasp, and hand over unmodified surgical needles," *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 9637–9643.

[13] M. Hwang, J. Ichnowski, B. Thananjeyan, D. Seita, S. Paradis, D. Fer, T. Low, and K. Goldberg, "Automating surgical peg transfer: Calibration with deep learning can exceed speed, accuracy, and consistency of humans," *IEEE Transactions on Automation Science and Engineering*, 2022.

[14] J. Xu, B. Li, B. Lu, Y.-H. Liu, Q. Dou, and P.-A. Heng, "Surrol: An open-source reinforcement learning centered and dvrk compatible platform for surgical robot learning," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1821–1828.

[15] "Bullet real-time physics simulation," https://pybullet.org/wordpress/, accessed: 2023-07-15.

[16] A. Munawar, Y. Wang, R. Gondokaryono, and G. S. Fischer, "A real-time dynamic simulator and an associated front-end representation format for simulating complex robots and environments," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019, pp. 1875–1882.

[17] "Unity physics simulation," https://unity.com/, accessed: 2023-07-15.

[18] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," *arXiv preprint arXiv:1809.02627*, 2020. [Online]. Available: https://arxiv.org/pdf/1809.02627.pdf

[19] Z. Cui, J. Cartucho, S. Giannarou, *et al.*, "Caveats on the first-generation da vinci research kit: latent technical constraints and essential calibrations," *arXiv preprint arXiv:2210.13598*, 2022.

[20] Y. Wang, R. Gondokaryono, A. Munawar, and G. S. Fischer, "A convex optimization-based dynamic model identification package for the da vinci research kit," *IEEE Robotics and Automation Letters*, pp. 3657–3664, 2019.

[21] G. Chrysilla, N. Eusman, A. Deguet, and P. Kazanzides, "A compliance model to improve the accuracy of the da vinci research kit (dvrk)," *Acta Polytechnica Hungarica*, vol. 16, no. 8, 2019.

[22] R. Gondokaryono, M. Haiderbhai, A. Munawar, T. Looi, J. Drake, and L. A. Kahrs, "A modular ros-based dvrk teleoperation controller architecture," *Hamlyn Symposium on Medical Robotics*, 2022, pp. 139–140.

[23] M. J. Mataric, "Reward functions for accelerated learning," *Machine learning proceedings 1994*. Elsevier, 1994, pp. 181–189.

[24] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[28] J. Truong, M. Rudolph, N. H. Yokoyama, S. Chernova, D. Batra, and A. Rai, "Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation," *Conference on Robot Learning*. PMLR, 2023, pp. 859–870.

[29] E. Colleoni, P. Edwards, and D. Stoyanov, "Synthetic and real inputs for tool segmentation in robotic surgery," *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2020, pp. 700–710.

[30] Y. Jiang, T. Zhang, D. Ho, Y. Bai, C. K. Liu, S. Levine, and J. Tan, "Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning," *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2884–2890.

[31] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "Rl-cyclegan: Reinforcement learning aware simulation-to-real," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 157–11 166.

ACKNOWLEDGMENT

The authors acknowledge The Wilfred and Joyce Posluns Centre for Image Guided Innovation & Therapeutic Intervention (PCIGITI) at The Hospital for Sick Children, Toronto, Canada. Our research would not have been possible without access to laboratory space and equipment. We thank Florian Shkurti, Thomas Looi, and James Drake for their support and discussions.