2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Surgical Gym: A high-performance GPU-based platform for reinforcement learning with surgical robots

Samuel Schmidgall[1], Axel Krieger[2] and Jason Eshraghian[3]

***Abstract*— Recent advances in robot-assisted surgery have resulted in progressively more precise, efficient, and minimally invasive procedures, sparking a new era of robotic surgical intervention. This enables doctors, in collaborative interaction with robots, to perform traditional or minimally invasive surgeries with improved outcomes through smaller incisions. Recent efforts are working toward making robotic surgery more autonomous which has the potential to reduce variability of surgical outcomes and reduce complication rates. Deep reinforcement learning methodologies offer scalable solutions for surgical automation, but their effectiveness relies on extensive data acquisition due to the absence of prior knowledge in successfully accomplishing tasks. Due to the intensive nature of simulated data collection, previous works have focused on making existing algorithms more efficient. In this work, we focus on making the simulator more efficient, making training data much more accessible than previously possible. We introduce Surgical Gym, an *open-source* high performance platform for surgical robot learning where both the physics simulation and reinforcement learning occur directly on the GPU. We demonstrate between *100-5000*$\times$ faster training times compared with previous surgical learning platforms. The code is available at: https://github.com/SamuelSchmidgall/SurgicalGym.**

I. INTRODUCTION

The field of surgery has undergone a major transformation with the introduction of robots. These robots, controlled directly by surgeons, have enhanced the precision and efficiency of complex surgical procedures [1]. In practice, the surgeon operates the robot from a control station using tools like hand controllers and foot pedals. The robot, in response, carries out the surgeon's movements with great accuracy, eliminating even the smallest amount of shaking.

In modern medical practice, surgical robots play a pivotal role in supporting surgeons during minimally invasive procedures, with over a million surgeries per year carried out by human surgeons guiding these robots [2]. Despite their complexity, surgical robots in practice are not autonomous and rely on human control to function. The incorporation of autonomy has the potential to dramatically improve surgical outcomes since automation of these tasks can alleviate the burden of repetitive tasks and minimize surgeon fatigue [3]. The growing desire to improve surgical efficiency has led to a growth in efforts focused on automating various surgical tasks, ranging from suturing [4], [5], [6], endoscope control [7], [8], [9], and tissue manipulation [10], [11], [12], [13], to pattern cutting [14], [15], [16].

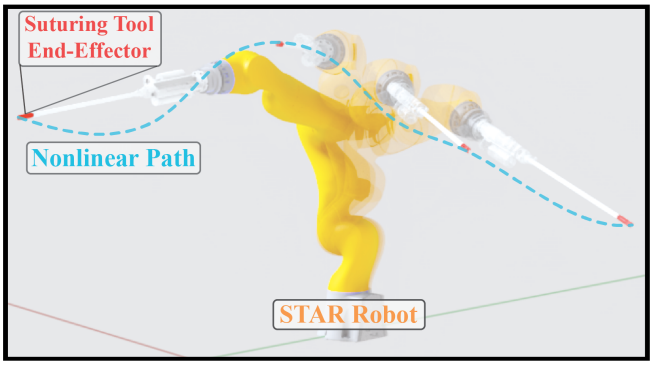

Fig. 1. Demonstration of Smart Tissue Autonomous Robot robot performing path following (path in blue dashed lines) going from right to left. Suturing tool end-effector (red) precisely follows randomly generated points along a nonlinear path.

Recently, deep reinforcement learning (RL) methods have shown they can produce various control strategies exceedingly well, which is promising for automated surgery [17], [16], [18], [19], [20]. However, this development does not come without a myriad of challenges. One of the most significant challenges results from the fact that RL-based methods often require a large amount of data to succeed, particularly because they start without any pre-existing domain knowledge [21], [22], [23]. While strategies using specialized domain knowledge provide significant value (*e.g. expert demonstration from surgeon*), there remains a substantial need for larger and more diverse training datasets to improve task generalization [24], [25]. Building on this, the field of robotics has recently experienced a surge of major advancements with learning-based control algorithms due to the development of high-performance GPU-based physics simulators, which provide *100-10000*$\times$ faster data collection rates than CPU based simulators [26], [27], [28], [29].

Despite these advancements in other areas of robotics, the potential benefits of these high-performance simulators for surgical robot learning are yet to be realized. We believe that providing easy access to surgical robotic training data is the next step toward building more capable and autonomous surgical robots. Toward this, we present **Surgical Gym**, an open-source GPU-based learning platform for surgical robotics.

Our main contributions are as follows:

- We introduce an open-source high-performance plat-

[1]Samuel Schmidgall is with the Department of Electrical Engineering, Johns Hopkins University, Baltimore, MD, USA sschmi46@jhu.edu
[2]Axel Kriger is with the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD, USA axel@jhu.edu
[2]Jason Eshraghian is with the Department of Electrical Engineering, University of California, Santa Cruz, Santa Cruz, CA, USA jeshragh@ucsc.edu

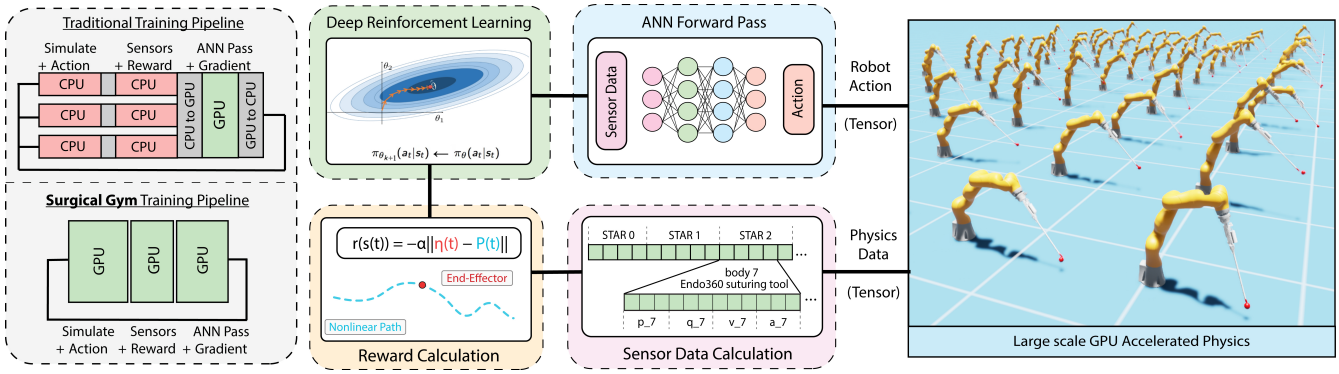

Fig. 2. **(Left) Data process flow comparison.** Comparison between the traditional reinforcement learning pipeline which use CPU based physics engines and Surgical Gym, which uses a GPU based physics engine. Surgical Gym avoids expensive data transfer between the CPU and GPU, with all operations tensorized on the GPU. **(Right) Detailed Surgical Gym process flow.** Large scale GPU accelerated physics are executed with thousands of robots in parallel. Sensor data is calculated from actor bodies and their corresponding positions, rotations and velocities which are stored directly in PyTorch tensors. Reward calculation takes relevant sensor data, calculates a per-agent reward, which is passed to the GPU-accelerated Proximal Policy Optimization implementation for gradient calculation. Finally, sensor data is forward propagated through an ANN to calculate an action, which is sent to the physics engine.

form for surgical robot learning that accelerates training times up to *5000*× by leveraging GPU-based physics simulation and reinforcement learning, thereby making surgical robot training data more accessible.

- We built *five* training environments which support learning for *six* different surgical robots & attachments (da Vinci Research Kit [30] and the Smart Tissue Autonomous Robot [31], [32]). GPU-based RL algorithms are incorporated into this library, making training on these environments simple and reproducible even without RL expertise.

We believe that providing easy access to surgical robot training data is the next step toward building surgical robots with greater autonomy, and hope Surgical Gym provides a step forward in this direction. All of the code and environments are *open-source* and can be found at the following link: https://github.com/SamuelSchmidgall/SurgicalGym.

II. Background

Reinforcement Learning for Robotics

RL is a machine learning paradigm where an agent learns from the environment by interacting with it and receiving rewards for performing actions [33]. The fundamental concept behind reinforcement learning is trial-and-error search, i.e., the agent selects an action a_t when given a particular state s_t, receives a reward r_t based on that state, and then transitions to a new state s_{t+1}. The goal of the agent is to maximize the expected cumulative reward over time. Mathematically, the environment is modeled as a Markov Decision Process (MDP), defined by a tuple (S, A, P, R), where S is the space of possible states, A is the space of possible actions, P is the state transition probability, i.e., $P(s'|s, a)$ is the probability of transitioning to state s' after taking action a in state s, and R is the reward function. $R(s, a)$ is the expected reward received after taking action a in state s.

The agent's behavior is defined by a policy (π), which is a mapping from states to probabilities of selecting each action, i.e., $\pi(a|s)$ is the probability of taking action a in state s. The value of a state, denoted by $V^\pi(s)$, is the expected cumulative reward when starting from state s and following policy π. The Bellman equations for $V^\pi(s)$ is defined as:

$$V^\pi(s) = \sum_a \pi(a|s) \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^\pi(s') \right]$$

where γ is the discount factor, s' is the next state, and a' is the next action.

The objective in reinforcement learning is to find the optimal policy π^* that maximizes the expected cumulative reward, i.e., $\pi^* = \arg\max_\pi V^\pi(s)$.

Surgical Robot Algorithms

Previous efforts in surgical robotics have focused on developing sophisticated controllers for particular subtasks [34], [35], [31], [32]. While these approaches are quite successful and reliable, they typically focus on solving surgical subproblems independently. Considering the wide diversity of scenarios that clinical surgeries may present, it is unlikely that isolated subtask solutions will lead to full surgical *automation*.

RL, on the other hand, demonstrates the ability to generalize well to novel circumstances given a sufficient amount of experience [36]. However, while control algorithms have been demonstrated to be reliable over decades of research, RL algorithms are relatively new and have yet to be deployed in patient-critical applications. This is particularly important for high-precision problems like automated surgery. Therefore, it is likely a combination of control-based and RL-based approaches will enable highly reliable and autonomous surgery.

Surgical Robot Platforms

The da Vinci: The da Vinci Surgical System is a robotic surgical system designed to facilitate complex surgery using a minimally invasive approach [37].

The system operates through an interface with the Master Tool Manipulator (MTM), which serves as the control

center for the surgeon to direct surgical actions. Through the MTM's handles or joysticks, the surgeon's movements are translated into corresponding motions of the **Patient Side Manipulators (PSMs)**, which are robotic arm attachments responsible for performing the surgery. The PSMs are flexible, multi-jointed instruments capable of holding and manipulating surgical tools, adjusting to the unique anatomy and requirements of each procedure.

A third component of the da Vinci system is the **Endoscopic Camera Manipulator (ECM)**. The ECM is another robotic arm attachment that holds and controls the movement of a stereo endoscope, a special camera that provides a high-definition, three-dimensional view of the surgical field. This allows the surgeon, from the control console, to have a detailed and magnified view of the area being operated on, improving precision during the surgical procedure.

The Smart Tissue Autonomous Robot: The **Smart Tissue Autonomous Robot (STAR)** [35], [31] is an autonomous robot designed to perform soft tissue surgery with minimal assistance from a surgeon. The STAR robot has been used for a variety of autonomous surgical procedures, most notably, the first autonomous laparoscopic surgery for intestinal anastomosis [32] (reconnection of two tubular structures such as blood vessels or intestines). The STAR performed the procedure in four different animals, producing better results than humans executing the same procedure.

The STAR uses a seven DoF light-weight arm mounted with an actuated suturing tool, which has actuators to drive a circular needle and a pitch axis (see *Robot Descriptions*). The STAR supports a manual mode and an automatic mode. In *manual mode*, a surgeon selects where each stitch is placed, whereas in *automatic mode*, an incision area is outlined and the STAR determines where each stitch is placed. Automatic mode chooses these placements based on the contour of the incision, which, once chosen, is transformed to the 3D camera frame and stitches are distributed evenly [35].

Software Platforms for Surgical Robot Learning

The first open-source RL-focused environment for surgical robotics is the da Vinci Reinforcement Learning (dVRL) suite [38], which aims to enable scientists without a surgical background to develop algorithms for autonomous surgery. dVRL introduces two straightforward training tasks: a robotic reach and a pick-and-place problem. Transfer to robotic hardware is demonstrated using the robotic target reaching policy for suctioning blood using a simulated abdomen (created by molding pig parts into gelatin and then filled with blood). Transfer to hardware was made possible since environments were designed around being compatible with the da Vinci Research Kit (dVRK) [30], a large-scale effort which enables institutions to share of a common hardware platform for the da Vinci system.

UnityFlexML [39] presents a framework that expands on autonomous surgical training, focusing on manipulating deformable tissues during a nephrectomy procedure. This framework was built on Unity3D, which naturally supports deformable objects. Like dVRL, the policy in UnityFlexML demonstrates successful transferability to the dVRK.

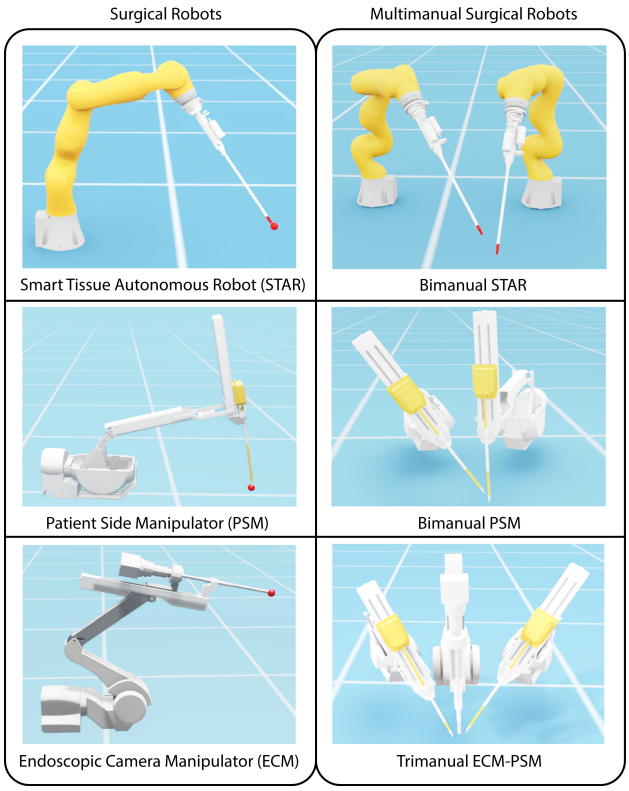

Fig. 3. Demonstration of surgical robots & attachments supported in Surgical Gym on a target reaching task.

LapGym [40] provides an RL environment for developing and testing automation algorithms in laparoscopic surgery, covering four main skills: spatial reasoning, deformable object manipulation & grasping, dissection, and thread manipulation. This includes a rich set of tasks from rope threading to tissue dissection. Unlike previous simulators, it presents a variety of image-based learning problems from which it supports RGB, depth, point clouds and semantic segmentation.

Finally, SurRoL [22] is an open-source RL focused simulation platform, like previous work, designed to be compatible with the dVRK. SurRoL develops environments for *both* da Vinci attachments (patient side and endoscopic camera manipulators, see subsection *The da Vinci System*) as well as for multiple arms.

SurRoL and LapGym have many environments, where SurRoL implements *ten* unique training tasks and LapGym implements a total of *twelve*. Other platforms only supported a limited number of training tasks, with UnityFlexML having *one* and dVRL having *two*. Applications of RL on these environments focus on data-efficiency, aiming to develop algorithms which use few environment interactions, e.g. with SurRoL only 100,000 [22], [23] interactions in their experiments. This is because data collection in these environments is ***expensive*** (see subsection *Simulator Performance Comparison*) and thus experimentation is much more limited.

III. METHODS

Simulation Setup

We build Surgical Gym on Isaac Gym [26], which is a tensorized physics platform utilizing PhysX. The physics simulation as well as the observation, reward, and action calculation takes place on the GPU, enabling direct data transfer from the physics buffers to GPU tensors in PyTorch [41]. This process bypasses any potential slowdowns that could occur if the data were to pass through the CPU.

In Surgical Gym, an actor is composed of rigid bodies connected by joints. Rigid bodies are the primitive shapes or meshes that comprise of the actor. Different joint types link rigid bodies, each having a specific number of DoFs: fixed joints have 0, revolute and prismatic joints have 1, and spherical joints have 3. Joints can take different types of simulation input, including forces, torques, and PD controls (position or velocity targets). The action space for Surgical Gym environments are currently configured for position-based (PD) commands. However, each environment also supports torque and velocity-based commands.

Like experiments in Isaac Gym, we use the Temporal Gauss Seidel (TGS) solver [42] to calculate future object states. It uses sub-stepping to accelerate convergence better than solvers using larger steps with multiple iterations. The solver computes and accumulates scaled velocities into a delta buffer per iteration, which is then projected onto constraint Jacobians and added to the biases in constraints. This process has a similar computational cost as the traditional Gauss-Seidel solver [43], but without the expense of sub-stepping. For positional joint constraints, an extra rotational term is calculated for joint anchors to prevent linearization artifacts.

Robot Descriptions

To support dVRK compatibility for the PSM and ECM attachments, URDF descriptor models were derived from the dVRK-ros[1] speficiations [30] and converted to Universal Scene Descriptor (USD) format to be compatible with the GPU-based physics engine. For the STAR robot, we used descriptors for the Kuka LWR (7 DoF) arm and the suturing tool from the Endo360 specification, as was used in suturing demonstrations with the STAR robot [32].

The PSM has seven DoFs, where the arm position is determined by the first six (denoted as θ_i), and the seventh DoF corresponds to the PSM jaw angle, which, from an action perspective, is either open or closed ($a_{jaw} \in \{0, 1\}$) but has an intermediate angle ($\theta_{jaw} \in \mathbb{R}$) during the process of opening or closing. The PSM arm includes both revolute (R) and prismatic (P) actuated joints, arranged in the following sequence: RRPRRR.

The ECM tool follows a similar structure to the PSM, however it does not have a jaw, rather, a camera attached to the tool tip. The ECM arm also consists of revolute and prismatic joints arranged in the following sequence: RRPRRR.

[1]https://github.com/jhu-dvrk/dvrk-ros

The STAR robot has 8 DoFs with a suturing tool attachment extending from the end of the robot arm. The STAR arm consists entirely of revolute joints arranged in the following sequence: RRRRRRRR.

Action Space: Although the dVRK supports 6-DoF motion (and the STAR 7-DoF), previous works have defined the action space as positional coordinates of the end-effector with a dynamics controller actuating the joints [22], [40]. While this accelerates learning where data is scarce, (1) it can restrict the utilization of the robot's complete motion capabilities and (2) it tends to be computationally expensive. However, in practice, end-effector control is likely a much better option from a safety perspective. We opted to define the action space in a way that enables the full range of robot motions via torque or PD control, capitalizing on the extensive data available in Surgical Gym, with the opportunity for custom controllers to be wrapped on top of this action space for all of the supported robots.

Observation Space: While the observation space differs slightly between tasks, most of it remains consistent. The base observation space for the robot is defined by the concatenation of DoF positions ($\boldsymbol{\theta} \in \mathbb{R}^n$ where $n =$ number of DoFs), DoF velocities ($\dot{\boldsymbol{\theta}} \in \mathbb{R}^n$), the tool link tip position ($\boldsymbol{p}_{tip} \in \mathbb{R}^3$), and the current PD-targets for the DoF ($\boldsymbol{\theta}_{target} \in \mathbb{R}^n$). With tasks involving a goal position (e.g. Target Reaching and Active Tracking) the goal position (g or $g_t \in \mathbb{R}^3$) is included in the observation vector. Similarly, with tasks involving image input (e.g. Endoscopic Image Matching) the target image ($p_{x,y,target}$) and the current end-effector image ($p_{x,y}$) is provided.

Policy learning and control

For policy learning in Surgical Gym, we utilize a GPU-optimized implementation of Proximal Policy Optimization (PPO) algorithm [44] which has been adapted for efficient parallel learning with numerous robots [45].

Previous work found that the hyper-parameter, *batch size* $B = n_{\text{robots}} n_{\text{steps}}$, plays a significant role in learning efficiency. Here, n_{steps} signifies the steps each robot takes per update, and n_{robots} denotes parallel-simulated robots. By increasing n_{robots}, we decrease n_{steps} to optimize training times. However, to ensure algorithm convergence and effective Generalized Advantage Estimation (GAE) [46], n_{steps} cannot be reduced below a threshold, which has been shown to be fewer than 25 steps or 0.5s of simulated time [45]. For backpropagation, the batch is split into large mini-batches, which improves the learning stability without increasing training time.

During training, robot resets, triggered either by falls or after specific intervals to facilitate new trajectory exploration, disrupt the infinite reward horizon assumed by the PPO algorithm's critic function, potentially impairing performance. To rectify this, we differentiate between terminations due to falls or goal attainment, which the critic can anticipate, and time-outs, which are unpredictable. We address time-out cases by augmenting the reward with the critic's prediction of the infinite sum of future discounted rewards, a technique

known as "bootstrapping." [45] This strategy, necessitating modification of the standard Gym interface to detect time-outs, effectively maintains the infinite horizon assumption.

IV. Results

Environments

Surgical Gym makes RL easier to use by including a common Gym-like interface [47], which helps make the process of developing and evaluating algorithms more straightforward. Included in this interface, we have developed a range of learning-based tasks that are useful for surgical automation, covering varying degrees of task complexity. We build five tasks that support five different surgical robots & attachments. We chose to incorporate tasks that develop low-level control capabilities (e.g. end-effector target reaching, image-guided navigation) to build a foundation for more advanced problems that would require these skills. These five tasks are described in more detail below.

- **Target Reaching:** The Target Reaching task involves aligning the end-effector, η_t, of the robot with a red target sphere with a randomly sampled position, g, for each environment. The task's reward structure is devised based on the distance between the end-effector and the sphere.

  $$r(s_t) = \rho||\eta_t - g||$$

  This environment supports the PSM and ECM attachments for the da Vinci, and the STAR robot with the suturing tool end-effector. The goal point for this task has a randomization range of $g_{x,y,z} \sim \mathcal{N}(0, 0.05)$ in front of the robot for the ECM & PSM and $g_{x,y,z} \sim \mathbb{N}(0, 0.15)$ for the STAR, providing a wide reaching range relative to the robot.
- **Active Tracking:** In the Active Tracking task, similar to the Target Reaching task, the goal is to align the end-effector with a target position, g_t. Unlike the previous task, the goal location in this environment is time-varying and changes its position in all directions (x, y, z) as follows:

  $$g_t = g_t + \dot{g}_t$$

  $$\dot{g}_t = \dot{g}_t + \mathbb{N}(0, 1^{-4})$$

  This environment supports the PSM and ECM attachments for the da Vinci, and the STAR robot. The task reward system is based on the distance between the moving sphere and the end-effector. The tracking problem samples the goal coordinates from $g_{x,y,z} \sim \mathcal{N}(0, 0.05)$ for ECM & PSM and $g_{x,y,z} \sim \mathcal{N}(0, 0.15)$ for STAR with a max positional offset of -0.2 and 0.2.
- **Endoscopic Image Matching:** In the Endoscopic Image Matching task, the sensory input is a specific target image ($p_{x,y,target}$), and the goal is to attain the same image through the endoscopic camera of the ECM robot on its end-effector ($p_{x,y}$). The task is about creating an accurate match between the given image and the current viewpoint of the end-effector. The reward is defined as follows:

  $$r(s_t) = \frac{1}{w \cdot h} \sum_{x,y} (p_{x,y} - p_{x,y,target})$$

  This task tests the robot's ability to precisely position and orient itself to match a specific visual scene, simulating the surgical requirement of aligning tools to match reference images.
- **Path following:** In the Path Following task, the robot's objective is to navigate its end-effector, η_t, along a predefined trajectory, $\mathcal{P}$, which is represented as a sequence of waypoints in the environment. The main challenge lies in maintaining a consistent alignment with the path while adapting to any minor perturbations in the environment. The path waypoints are sampled using a parametric representation of the path, which is application configurable. By default, a cubic spline $\mathcal{S}(t)$ is used to represent the path $\mathcal{P}$, which interpolates the given waypoints. The cubic spline $\mathcal{S}(t)$ can be represented as:

  $$\mathcal{S}(t) = a(t - t_i)^3 + b(t - t_i)^2 + c(t - t_i) + d,$$

  where $[a, b, c, d]$ are the coefficients of the spline, which are randomly sampled within a set of values at the beginning of the task. Once the path is represented by the spline, the waypoints are sampled at regular intervals of distance along the spline. The resulting samples are then used as reference positions for the end-effector, η_t, at each timestep.
  The reward function for this task is calculated based on how closely the end-effector follows the trajectory:

  $$r(s_t) = -\alpha||\eta_t - \mathcal{P}(t)||$$

  where α is a scaling factor determining the penalty for deviating from the path and $\mathcal{P}(t)$ denotes the desired position on the path at time t.
  Supported environments for this task include the PSM & ECM attachments for the da Vinci, and the STAR robot.
- **Multi-Tool Target Reaching:** In the Multi-Tool Target Reaching task, the robot is required to coordinate the movement and positioning of several end-effectors, (e.g. with $n = 2$ there is η_1 and η_2) to align with multiple target spheres (e.g. g_1 and g_2). Each target sphere has its own distinct position, and the challenge lies in simultaneously positioning multiple tools and handling self-collisions. The reward function takes into account the distance between each end-effectors and their targets:

  $$r(s_t) = \rho \sum_{i=1}^{n} ||\eta_i - g_{\eta_i}||$$

  The environment setup is designed to simulate complex surgical scenarios where multiple tools need to be

coordinated simultaneously. This task is an extension of the Target Reaching task and represents a higher level of complexity. Supported environments for this task include position two independent PSMs (Bimanual PSM), two indepdendent STARs (Bimanual STAR), and two PSMs together with one ECM (Trimanual PSM-ECM).

Simulator Performance Comparison

To properly assess the computational efficiency of various surgical robot learning platforms, we conducted performance profiling of each simulation environment. Time profiling begins after the first simulation step and ends after *one million* simulation steps were reached. Profiling was done for each environment on their respective *PSM Reach* task, which was a shared environment among all simulators. Simulation times were collected for each environment by running the simulator 30 times in separate instances[2].

The performance of each simulator was first tested *without* incorporating learning dynamics in order to evaluate the speed of the simulator by itself. In this condition, the physics simulator was given random action input ($a_i \sim U(-1, 1)$) and simulator feedback was ignored. Results are shown below in *Table I*.

TABLE I
SIMULATION SPEED OF SURGICAL SIMULATORS

Simulator	Seconds per 1M Timesteps	Frames per second
UnityFlexML	17857 ± 1123.1	56 ± 4.2
UnityFlexML WD[3]	6410 ± 124	156 ± 6.4
LapGym	2166 ± 52.1	461 ± 11
dVRL	2113 ± 12.2	473 ± 3.1
SurRoL	242 ± 3.2	4124 ± 54
Surgical Gym	$\mathbf{2.9 \pm 0.2}$	$\mathbf{344828 \pm 22247}$

As can be seen, Surgical Gym takes an average of 2.9 seconds to collect $1M$ timesteps (344828 frames per second), whereas UnityFlexML takes as much as 17857 seconds ($\sim$ 300 minutes at 56 frames per second). In terms of simulation time, Surgical Gym is $\sim 7000\times$ faster than the slowest simulator and $\sim 80\times$ faster than the fastest simulator.

Next, the performance was tested *with* the incorporation of learning dynamics. In this condition, the simulator is given action input forward propagated through a neural network, reward is accumulated, and policy gradient updates are calculated. The neural network is a 3 hidden-layer network with hidden dimensions 256, 128, and 64 respectively using ELU nonlinear activations[48]. Rather than testing the pure simulation speed, this test benchmarks how long it takes for the policy to be *optimized* on one million simulation steps, validating the time efficiency of the *learning dynamics*. Results are shown below in *Table II*.

[2]Simulation speed times for LapGym and UnityFlexML were self-reported by the authors of the respective libraries.

TABLE II
SIMULATION SPEED OF SIMULATOR LEARNING DYNAMICS

Simulator	Seconds per 1M Timesteps	Frames per second
UnityFlexML	37037 ± 1796	27 ± 2.9
UnityFlexML WD[4]	19231 ± 1342	52 ± 3.9
LapGym (8 core)	833 ± 13.5	1200 ± 19.1
dVRL	7024 ± 26.5	142 ± 0.5
SurRoL	612 ± 35.2	1634 ± 94.3
Surgical Gym	$\mathbf{6.8 \pm 0.4}$	$\mathbf{147059 \pm 8170}$

Incorporating the learning dynamics, Surgical Gym takes an average of 6.8 seconds to train on $1M$ timesteps (147059 frames per second), whereas UnityFlexML takes as much as 37037 seconds ($\sim$ 600 minutes at 27 frames per second). With respect to training times Surgical Gym is $\sim 5500\times$ faster than the slowest simulator and $\sim 90\times$ faster than the fastest simulator.

For each of the libraries, default configurations were used for evaluation (e.g. LapGym using 8 cores). For Surgical Gym, we ran $20,000$ environments in parallel directly on a *single* GPU (8GB NVIDIA Quadro RTX 4000).

V. CONCLUSION

We see Surgical Gym expanding beyond the environments discussed in this paper. Future work could advance this library with more complex problems, such as in SurRoL [22] and LapGym [40]. We also see the incorporation of tasks involving soft body models which more closely resemble surgical tasks, such as tissue manipulation [10], which IsaacGym supports [49]. Transferring from simulation to hardware is more challenging with RL-trained torque and PD controllers than inverse dynamics (which was used in previous works). Thus, the further integration of V-REP kinematic models [50] redesigned with GPU-compatibility may make the transfer from simulation to hardware simpler.

While Surgical Gym provides significant improvements in simulation speed, there is still much work to be done before this can be applied to autonomous surgery. We hope by making this work open-source, researchers can build on it to make sophisticated autonomous controllers for a wide variety of surgical applications (e.g. suturing, tissue manipulation, pattern cutting). Here, we introduce an open-source platform that accelerates surgical robot training with GPU-based simulations and RL. This platform features five training environments, with six surgical robots. We believe easier access to simulated training data will enable the next generation of autonomous surgery and hope Surgical Gym provides a step in that direction.

ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship for Comp/IS/Eng-Robotics under Grant No. DGE2139757 and NSF/FRR 2144348.

REFERENCES

[1] A. Attanasio, B. Scaglioni, E. De Momi, P. Fiorini, and P. Valdastri, "Autonomy in surgical robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 651–679, 2021.

[2] M. Tindera, "Robot wars: $60b intuitive surgical dominated its market for 20 years. now rivals like alphabet are moving in. forbes," 2019.

[3] T. Haidegger, "Autonomy for surgical robots: Concepts and paradigms," *IEEE Transactions on Medical Robotics and Bionics*, vol. 1, no. 2, pp. 65–76, 2019.

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

[5] Z.-Y. Chiu, F. Richter, E. K. Funk, R. K. Orosco, and M. C. Yip, "Bimanual regrasping for suture needles using reinforcement learning for rapid motion planning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7737–7743, IEEE, 2021.

[6] 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 *2020 29th IEEE international conference on robot and human interactive communication (RO-MAN)*, pp. 1380–1386, IEEE, 2020.

[7] A. Pore, M. Finocchiaro, D. Dall'Alba, A. Hernansanz, G. Ciuti, A. Arezzo, A. Menciassi, A. Casals, and P. Fiorini, "Colonoscopy navigation using end-to-end deep visuomotor control: A user study," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9582–9588, IEEE, 2022.

[8] M. Turan, Y. Almalioglu, H. B. Gilbert, F. Mahmood, N. J. Durr, H. Araujo, A. E. Sarı, A. Ajay, and M. Sitti, "Learning to navigate endoscopic capsule robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3075–3082, 2019.

[9] G. Trovato, M. Shikanai, G. Ukawa, J. Kinoshita, N. Murai, J. Lee, H. Ishii, A. Takanishi, K. Tanoue, S. Ieiri, *et al.*, "Development of a colon endoscope robot that adjusts its locomotion through the use of reinforcement learning," *International journal of computer assisted radiology and surgery*, vol. 5, pp. 317–325, 2010.

[10] Y. Li, F. Richter, J. Lu, E. K. Funk, R. K. Orosco, J. Zhu, and M. C. Yip, "Super: A surgical perception framework for endoscopic tissue manipulation with surgical robotics," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2294–2301, 2020.

[11] C. Shin, P. W. Ferguson, S. A. Pedram, J. Ma, E. P. Dutson, and J. Rosen, "Autonomous tissue manipulation via surgical robot using learning based model predictive control," in *2019 International conference on robotics and automation (ICRA)*, pp. 3875–3881, IEEE, 2019.

[12] S. A. Pedram, P. W. Ferguson, C. Shin, A. Mehta, E. P. Dutson, F. Alambeigi, and J. Rosen, "Toward synergic learning for autonomous manipulation of deformable tissues via surgical robots: An approximate q-learning approach," in *2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*, pp. 878–884, IEEE, 2020.

[13] J. Lu, A. Jayakumari, F. Richter, Y. Li, and M. C. Yip, "Super deep: A surgical perception framework for robotic tissue manipulation using deep learning for feature extraction," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4783–4789, IEEE, 2021.

[14] T. Nguyen, N. D. Nguyen, F. Bello, and S. Nahavandi, "A new tensioning method using deep reinforcement learning for surgical pattern cutting," in *2019 IEEE international conference on industrial technology (ICIT)*, pp. 1339–1344, IEEE, 2019.

[15] B. Thananjeyan, A. Garg, S. Krishnan, C. Chen, L. Miller, and K. Goldberg, "Multilateral surgical pattern cutting in 2d orthotropic gauze with deep reinforcement learning policies for tensioning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2371–2378, IEEE, 2017.

[16] N. D. Nguyen, T. Nguyen, S. Nahavandi, A. Bhatti, and G. Guest, "Manipulating soft tissues by deep reinforcement learning for autonomous robotic surgery," in *2019 IEEE International Systems Conference (SysCon)*, pp. 1–7, IEEE, 2019.

[17] M. Yip, S. Salcudean, K. Goldberg, K. Althoefer, A. Menciassi, J. D. Opfermann, A. Krieger, K. Swaminathan, C. J. Walsh, H. Huang, *et al.*, "Artificial intelligence meets medical robotics," *Science*, vol. 381, no. 6654, pp. 141–146, 2023.

[18] 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.

[19] Y. Barnoy, O. Erin, S. Raval, W. Pryor, L. O. Mair, I. N. Weinberg, Y. Diaz-Mercado, A. Krieger, and G. D. Hager, "Control of magnetic surgical robots with model-based simulators and reinforcement learning," *IEEE Transactions on Medical Robotics and Bionics*, vol. 4, no. 4, pp. 945–956, 2022.

[20] Y. Ou and M. Tavakoli, "Sim-to-real surgical robot learning and autonomous planning for internal tissue points manipulation using reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2502–2509, 2023.

[21] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim, "Guided reinforcement learning with learned skills," *arXiv preprint arXiv:2107.10253*, 2021.

[22] 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," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1821–1828, IEEE, 2021.

[23] T. Huang, K. Chen, B. Li, Y.-H. Liu, and Q. Dou, "Guided reinforcement learning with efficient exploration for task automation of surgical robot," *arXiv preprint arXiv:2302.09772*, 2023.

[24] N. Heess, D. Tb, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.

[25] J. Hilton, J. Tang, and J. Schulman, "Scaling laws for single-agent reinforcement learning," *arXiv preprint arXiv:2301.13442*, 2023.

[26] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[27] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell, "Real-world robot learning with masked visual pre-training," in *Conference on Robot Learning*, pp. 416–426, PMLR, 2023.

[28] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," in *Conference on Robot Learning*, pp. 297–307, PMLR, 2022.

[29] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik, "In-hand object rotation via rapid motor adaptation," in *Conference on Robot Learning*, pp. 1722–1732, PMLR, 2023.

[30] 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," in *2014 IEEE international conference on robotics and automation (ICRA)*, pp. 6434–6439, IEEE, 2014.

[31] A. Shademan, R. S. Decker, J. D. Opfermann, S. Leonard, A. Krieger, and P. C. Kim, "Supervised autonomous robotic soft tissue surgery," *Science translational medicine*, vol. 8, no. 337, pp. 337ra64–337ra64, 2016.

[32] H. Saeidi, J. D. Opfermann, M. Kam, S. Wei, S. Léonard, M. H. Hsieh, J. U. Kang, and A. Krieger, "Autonomous robotic laparoscopic surgery for intestinal anastomosis," *Science robotics*, vol. 7, no. 62, p. eabj2908, 2022.

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[34] J. H. Palep, "Robotic assisted minimally invasive surgery," *Journal of minimal access surgery*, vol. 5, no. 1, p. 1, 2009.

[35] S. Leonard, K. L. Wu, Y. Kim, A. Krieger, and P. C. Kim, "Smart tissue anastomosis robot (star): A vision-guided robotics system for laparoscopic suturing," *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 4, pp. 1305–1317, 2014.

[36] C. Packer, K. Gao, J. Kos, P. Krähenbühl, V. Koltun, and D. Song, "Assessing generalization in deep reinforcement learning," *arXiv preprint arXiv:1810.12282*, 2018.

[37] S. DiMaio, M. Hanuschik, and U. Kreaden, "The da vinci surgical system," *Surgical robotics: systems applications and visions*, pp. 199–217, 2011.

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

[39] E. Tagliabue, A. Pore, D. Dall'Alba, M. Piccinelli, and P. Fiorini, "Unityflexml: Training reinforcement learning agents in a simulated surgical environment," in *I-RIM Conf.*, 2020.

[40] P. M. Scheikl, B. Gyenes, R. Younis, C. Haas, G. Neumann, M. Wagner, and F. Mathis-Ullrich, "Lapgym–an open source framework for reinforcement learning in robot-assisted laparoscopic surgery," *arXiv preprint arXiv:2302.09606*, 2023.

[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[42] M. Macklin, K. Storey, M. Lu, P. Terdiman, N. Chentanez, S. Jeschke, and M. Müller, "Small steps in physics simulation," in *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 1–7, 2019.

[43] R. Bagnara, "A unified proof for the convergence of jacobi and gauss–seidel methods," *SIAM review*, vol. 37, no. 1, pp. 93–97, 1995.

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

[45] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*, pp. 91–100, PMLR, 2022.

[46] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[47] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[48] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[49] V. Lim, H. Huang, L. Y. Chen, J. Wang, J. Ichnowski, D. Seita, M. Laskey, and K. Goldberg, "Real2sim2real: Self-supervised learning of physical single-step dynamic actions for planar robot casting," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 8282–8289, IEEE, 2022.

[50] G. A. Fontanelli, M. Selvaggio, M. Ferro, F. Ficuciello, M. Vendittelli, and B. Siciliano, "A v-rep simulator for the da vinci research kit robotic platform," in *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, pp. 1056–1061, IEEE, 2018.