2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

FF-SRL: High Performance GPU-Based Surgical Simulation For Robot Learning

Diego Dall'Alba[1,2,*], Michał Naskręt[1,*], Sabina Kamińska[1] and Przemysław Korzeniowski[1,†]

***Abstract*— Robotic surgery is a rapidly developing field that can greatly benefit from the automation of surgical tasks. However, training techniques such as Reinforcement Learning (RL) require a high number of task repetitions, which are generally unsafe and impractical to perform on real surgical systems. This stresses the need for simulated surgical environments, which are not only realistic, but also computationally efficient and scalable. We introduce FF-SRL (Fast and Flexible Surgical Reinforcement Learning), a high-performance learning environment for robotic surgery. In FF-SRL both physics simulation and RL policy training reside entirely on a single GPU. This avoids typical bottlenecks associated with data transfer between the CPU and GPU, leading to accelerated learning rates. Our results show that FF-SRL reduces the training time of a complex tissue manipulation task by an order of magnitude, down to a couple of minutes, compared to a common CPU/GPU simulator. Such speed-up may facilitate the experimentation with RL techniques and contribute to the development of new generation of surgical systems. To this end, we make our code publicly available to the community.**

I. INTRODUCTION

Robot-Assisted Surgical Systems (RASS) are becoming more popular and widely used, with a constant increase in their adoption in the last two decades [1]. Researchers are also exploring new possibilities and challenges of RASS, using platforms like da Vinci Research Kit (dVRK) [2].

One of the most interesting research topics is how to automate some aspects of RASS interventions, making these systems more autonomous and efficient [3]. A common approach is to use Reinforcement Learning (RL) techniques [4]–[8], which have proven to be effective in other robotic domains, such as legged locomotion and dexterous manipulation [9]. RL can support RASS in performing various surgical tasks that require adaptability and precision, such as manipulation of deformable tissues [7, 10] or suture needle [11, 12] and removing debris or liquids [4]–[6].

The publication was created within the project of the Minister of Science and Higher Education "Support for the activity of Centers of Excellence established in Poland under Horizon 2020" on the basis of the contract number MEiN/2023/DIR/3796. This project has received funding from the EU's Horizon 2020 research and innovation programme under grant agreement No 857533. This publication is supported by Sano project carried out within the International Research Agendas programme of the Foundation for Polish Science, co-financed by the EU under the European Regional Development Fund. We acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within the grant no. PLG/2020/013635

* These authors contributed equally to the paper. Ordered alphabetically.
[1] Sano Centre for Computational Medicine, Kraków, Poland https://sano.science/
[2] Department of Engineering for Innovation Medicine, University of Verona, Italy
† Corresponding author: Przemysław Korzeniowski (email: p.korzeniowski@sanoscience.org)

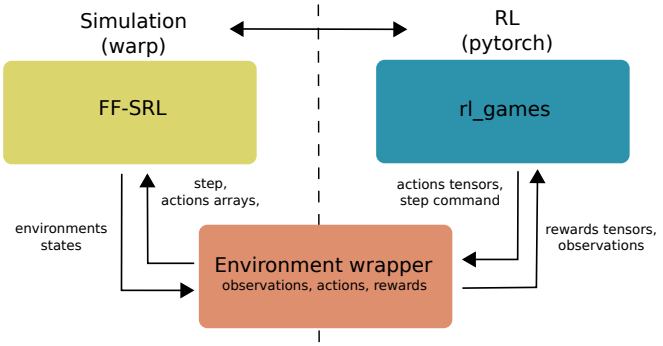

Fig. 1. Schematic representation of the integration between the proposed FF-SRL simulator and the rl_games RL framework to achieve a fully GPU training pipeline and support the development of autonomous RASS.

RL-based systems are usually trained in simulated environments, as many unsuccessful attempts are required which often exhibit unsafe behavior [9]. This is especially true for RASS, since performing real experiments faces significant economic and ethical constraints. However, to transfer the trained models in real settings, it is necessary to employ realistic, scalable, and robust simulations, supporting soft-body physics [7, 10]. Several simulators for RASS have been proposed, focusing on specific aspects, such as the kinematic model of the robot [4], more realistic physical [8] or visual [5] simulation or easier interfacing with real RASS [6]. All these simulators mainly exploit CPU calculations, relegating the use of the GPU to accelerate the training and inference part of the RL model only.

This is not optimal as it requires copying the data back and forth between CPU (simulation) and GPU (learning) memories . As such, it limits the complexity of the tasks that can be solved and, as a result, slows down research activities and community contributions. The solution is to compute the entire simulation and RL training process on the GPU. However, developing a massively-parallel GPU simulation is non-trivial. Its efficient implementation requires a careful handling of thousands of simultaneous threads and efficient global memory access patterns.

In this work we therefore propose:

- a high-performance, fully GPU-based simulator for surgical robotics, called FF-SRL, that supports soft tissue modeling via extended position based dynamics approach;
- the integration of FF-SRL into an entirely GPU-based RL training pipeline;
- the experimental validation of the proposed FF-SRL in

a fundamental surgical task (i.e. reaching a deformable tissue), comparing with simulator proposed in [8].

To the best of our knowledge, this is the first work that proposes a simulation environment for training autonomous RASS with RL techniques entirely on GPU, and the results confirm that our approach is able to significantly improve the performance compared to available RASS simulators, e.g. completing RL training in less than 2 minutes instead of over 1 hour. This enables training of complex learning tasks even on a commonly accessible consumer-grade hardware such as gaming laptops.

We make the simulator code publicly available[1] to encourage the community to adopt our approach and accelerate the progress of this promising research area.

II. RELATED WORKS

Research activities on RASS have been exploring the automation of various surgical tasks: these tasks range from simple ones, such as peg transfer [13]–[15], to more complex ones, such as manipulating suture needles [11, 12] or deformable tissues [16]–[19]. Among these tasks, we focus on tissue retraction, a preliminary phase of most surgical interventions. Tissue retraction involves lifting deformable tissue to expose an area of interest, such as an organ or lesion to be removed (see Fig. 2). This task is widely used by the RASS research community since it is simple to set up but, at the same time, with an adequate level of complexity for testing the developed automation approaches. Furthermore, this task can be learned in simulation and transferred to a real environment [7, 10, 20].

As task complexity has increased, there has been a growing interest in applying more complex automation approaches, such as RL techniques. RL allows the inclusion of multi-agent models [21], safety constraints [22], or demonstrations by expert users [12, 20, 23], also through interactive supervision [24]. To make RL training more effective, it is a common practice to carry it out in realistic simulation environments. Several simulation environments have been proposed for RASS. Some of them are based on rigid objects only, such as dVRL [4] and SurRoL [5]. dVRL provides an accurate kinematic model of the dVRK platform and simulates simple training or surgical tasks. SurRoL provides around ten surgical tasks supported by a more realistic visual rendering. Another simulator is AMBF-RL [6], which extends Articulated Multi-Body Framework (AMBF) [25] simulator specific for RL training and ensures simple interfacing with the real setup. However, none of these simulators exploit the simulation of deformable objects, which are essential for many surgical tasks. UnityFlexML [7] was the first simulator to address this issue by taking advantage of the Position-Based Dynamics (PBD) approach provided by NVIDIA Flex and integrating it within the Unity framework. LapGym [8] recently proposed a more advanced simulation environment, which relies on the accurate biomechanical finite element simulation provided by SOFA [26, 27]. Compared to other simulators, LapGym implements a rich catalog of surgical tasks, most based on interaction with deformable objects.

Thus, we propose a novel simulator that offers an entirely GPU-integrated RL simulation and training approach for RASS. Unlike described simulators that combine CPU calculations with a limited GPU-accelerated part, FF-SRL leverages the full power of GPU computing to optimize the overall process. To tackle the complexity of the simulation of highly non-linear behavior of tissues FF-SRL uses eXtended Postion-Based Dynamics (XPBD [28]). Several recent papers [29]–[33], turned XPBD into a serious competitor of more advanced simulation methods in terms of accuracy, stability, speed, and simplicity.

III. METHODS

In this section, we describe the proposed simulation environment, detailing the implementation choices regarding soft tissue simulation, RASS tools and collisions. Next, we describe how we included support for RL environments. We point out that some implementation choices have sacrificed pure computational performance in favor of a software architecture that simplifies maintenance and extensibility.

A. Simulation framework

FF-SRL implements an XPBD simulation technique [28], based on NVIDIA Warp [34], which is a Python framework for writing high-performance GPU-optimized simulation and graphics code. While Warp provides an XPBD implementation, we have implemented a version specifically optimized to handle the complexity of RASS simulation, including non-linear tissue behavior.

In particular, the proposed XPBD implementation improves the effectiveness of constraint solver iterations in simulating complex deformable objects. We split each time step into n smaller sub-steps and apply a single constraint iteration of PBD in each sub-step. This method, first suggested in [29], is more effective than performing a single large time-step with n constraint solver iterations. This approach has been further validated in [31], where a comprehensive rigid body solver was implemented to handle various joint types and the interaction with soft objects in a unified way. This is particularly useful in RASS, allowing easy interaction between rigid surgical tools and soft tissues. The authors of [28, 31] compared PBD with sub-stepping to more advanced solvers, both implicit and explicit, and found that it produces visually similar results while maintaining the simplicity of the original PBD method and reducing the sensitivity to matrix ill-conditioning.

1) Implementation of soft-body physics: Each object is represented by a set of V vertices with masses m_i, where $i = \{1, \cdots, V\}$, a position $\boldsymbol{x_i}$ and velocity $\boldsymbol{v_i}$. Vertices are embedded within a volumetric tetrahedral mesh and the objects' surface is divided into F triangle faces. Two adjacent vertices are combined into an edge. Two sets of constraints are used to provide object deformability with minimal computation required – conserving distance between points C^{d}

[1] https://github.com/SanoScience/FF-SRL

and volume of tetrahedrons C^{v}. Each constraint imposes corrections $\Delta \boldsymbol{x}_i$ on corresponding vertices' positions.

The distance constraint C^{d} is realized by the following corrections for vertices a, b residing on a given edge:

$$\begin{aligned} \Delta \boldsymbol{x}_{\boldsymbol{a}}^{\mathrm{d}} &= -k_{\mathrm{s}} \frac{w_a}{w_a + w_b} \left(|\boldsymbol{x_{a,b}}| - d_{a,b} \right) \frac{\boldsymbol{x_{a,b}}}{|\boldsymbol{x_{a,b}}|}, \\ \Delta \boldsymbol{x}_{\boldsymbol{b}}^{\mathrm{d}} &= +k_{\mathrm{s}} \frac{w_b}{w_a + w_b} \left(|\boldsymbol{x_{a,b}}| - d_{a,b} \right) \frac{\boldsymbol{x_{a,b}}}{|\boldsymbol{x_{a,b}}|}, \end{aligned} \tag{1}$$

where $\boldsymbol{x_{a,b}} = \boldsymbol{x_a} - \boldsymbol{x_b}$, w_a, w_b are inverse masses of the vertices a and b, $d_{a,b}$ is the rest distance between vertices a, b calculated before the simulation loop, and $k_{\mathrm{s}} \in [0, 1]$ is the constraint stiffness.

The volume constraint $C^{\mathrm{v}} = \frac{1}{6}(\boldsymbol{x_{b,a}} \times \boldsymbol{x_{c,a}}) \cdot \boldsymbol{x_{d,a}} - V_0$ (where V_0 is the rest volume of the tetrahedron) is calculated by the following corrections for the vertices a, b, c, d residing on a given tetrahedron:

$$\begin{aligned} \Delta \boldsymbol{x}_{\boldsymbol{a}}^{\mathrm{v}} &= -\frac{1}{6} s k_{\mathrm{v}} \cdot \\ &\quad (\boldsymbol{x_{b,a}} \times \boldsymbol{x_{c,a}} + \boldsymbol{x_{c,a}} \times \boldsymbol{x_{d,a}} + \boldsymbol{x_{d,a}} \times \boldsymbol{x_{b,a}}), \\ \Delta \boldsymbol{x}_{\boldsymbol{b}}^{\mathrm{v}} &= -\frac{1}{6} s k_{\mathrm{v}} (\boldsymbol{x_{b,a}} \times \boldsymbol{x_{c,a}}), \\ \Delta \boldsymbol{x}_{\boldsymbol{c}}^{\mathrm{v}} &= -\frac{1}{6} s k_{\mathrm{v}} (\boldsymbol{x_{c,a}} \times \boldsymbol{x_{d,a}}), \\ \Delta \boldsymbol{x}_{\boldsymbol{d}}^{\mathrm{v}} &= -\frac{1}{6} s k_{\mathrm{v}} (\boldsymbol{x_{b,a}} \times \boldsymbol{x_{d,b}}), \end{aligned} \tag{2}$$

where k_{v} is the stiffness and s is scaling factor:

$$s = \frac{C^{\mathrm{v}}}{\sum_{i \in \{a,b,c,d\}} |\nabla_{\boldsymbol{x}_i} C^{\mathrm{v}}|^2}. \tag{3}$$

We parallelized the algorithm using Jacobi-style constraint solver and atomic add function. In this case all the constraints can be calculated in parallel and atomically added as a vertex' total correction. Subsequently, the correction is averaged by the number of constraints applied to the vertex to obtain the final position correction $\overline{\Delta \boldsymbol{x}} = \Delta \boldsymbol{x}/n$. The stiffness parameters k_{s} and k_{v} are adopted from previous studies [7, 16, 19].

Tissue connections between different objects are implemented as distance constraints between adjacent vertices of one mesh and geometrical centers of triangles of other mesh. The constraint is applied with position correction $\Delta \boldsymbol{x}^{\mathrm{c}}$.

An overview of the algorithm loop is presented in Algorithm 1 with $\boldsymbol{g}$ being the gravity constant.

2) Surgical tool model: The surgical tool model employs two layers: visual and computational. The former is only a graphical representation of instrument mesh and the latter is used for collision detection. The computational model consists of three capsules: one acting as a rod corresponding to instrument's shaft, and two acting as clamps. The clamps are nested at the end of the rod and can pivot around their fixed point with angle $0° < \alpha < 30°$ from their symmetry axis.

We considered a simplified model of the movement of the instruments, but still representative of those used in minimally invasive surgical procedures, including RASS ones [4, 8]. We did not consider the distal wrist joints in this initial implementation, since most RL work in RASS does not control the orientation of the tool, keeping it constant [7, 10, 20, 22]. The main constraint that must be satisfied is imposed by the access port (i.e. the trocar) necessary to pass through the patient's abdominal wall. Therefore, the tool can rotate freely with respect to the access point $\boldsymbol{p_{RCM}}$ and translate along the shaft axis [8]. However, the control of the instrument is commonly carried out in Cartesian space, so we have implemented a simple mapping of the instrument's movements.

Algorithm 1 Simulation algorithm.

```
set initial positions x_0 and velocities v_0
h ← Δt/simSubSteps
k ← 0
while i < simSteps do
    evaluate instrument state
    while j < simSubSteps do
        predict velocity: ṽ ← v_k + hg
        predict position: x̃ ← x_k + hṽ
        initialize corrections: Δx̃ ← 0
        for all vertices do
            Δx̃ ← Δx̃ + Δx̃^d + Δx̃^g + Δx̃^c
        end for
        for all tetrahedra do
            Δx̃ ← Δx̃ + Δx̃^v
        end for
        x̃ ← x̃ + Δx̃/n
        v_{k+1} ← (x̃ − x_k)/h
        x_{k+1} ← x̃
        k ← k + 1
        j ← j + 1
    end while
    evaluate collisions
    i ← i + 1
end while
```

As represented in Fig. 2, given the instrument in two different poses to which the distal points (center of the grasper) $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ correspond. Given the position of the constraint point $\boldsymbol{p_{RCM}}$, we can calculate the vectors $\boldsymbol{v_1} = \boldsymbol{p_1} - \boldsymbol{p_{RCM}}$ and $\boldsymbol{v_2} = \boldsymbol{p_2} - \boldsymbol{p_{RCM}}$. Given these vectors it is possible to calculate the rotation angle θ and the respective axis $\boldsymbol{r_A}$ with the formulas:

$$\theta = \cos^{-1}\left(\frac{\boldsymbol{v_1} \cdot \boldsymbol{v_2}}{||\boldsymbol{v_1}|| \cdot ||\boldsymbol{v_2}||} \right), \quad \boldsymbol{r_A} = \frac{\boldsymbol{v_1} \times \boldsymbol{v_2}}{||\boldsymbol{v_1}|| \cdot ||\boldsymbol{v_2}||} \tag{4}$$

These rotations satisfy the $\boldsymbol{p_{RCM}}$ constraint, while the translation component can be obtained from $||\boldsymbol{v_1} - \boldsymbol{v_2}||$.

The surgical tool model has a predetermined dragging point positioned at the end of closed clamps (see points $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ in Fig. 2). When the angle between clamps is lower than 3° the algorithm checks for the closest vertex in radius r_l. If a vertex is found in the radius r_l then a dragging constraint is applied to the mesh and the dragging point. When the angle between clamps exceeds 3° all the dragging constraints are deactivated. Dragging simulation is carried

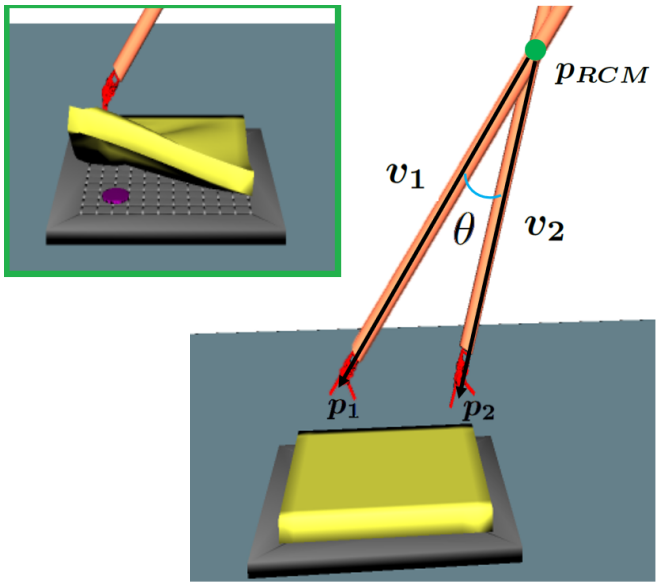

Fig. 2. Representation of the instrument movement model, required to ensure compliance with the remote center of motion; see the main text for further details. In the subfigure outlined in green, we show an example frame of tissue retraction procedure simulation created within our framework, including a model of the RASS instrument (orange shaft and red gripper), deformable fat tissue (yellow), and rigid supporting base (dark gray), including a target area of interest (purple).

out as creating an additional distance constraint with $k_s = 1$ between the dragging point and a mesh vertex. The constraint is applied with position correction $\Delta \boldsymbol{x}^g$. All vertices have an additional variable that stores the boolean value of the dragging constraint state.

3) Collisions: In order to provide a robust solution for collision detection between mesh triangles and the surgical tool model we employ Signed Distance Field (SDF) approach. SDFs are a popular choice for the collision detection shapes due to their performance [19]. We follow the per-element local optimisation scheme to find intersection points between SDFs and continuous surfaces proposed in [30]. Each capsule of the grasper model is treated as a SDF and its penetration is checked against the mesh' faces.

For each intersecting face an additional distance constraint is introduced, that is applied after deformation constraints are calculated. The constraint pushes the face out of the grasper SDFs in the direction of penetration vector and proportionally to the penetration depth with stiffness of $k_c = 1$.

B. Reinforcement learning integration

The developed simulator was integrated with the rl_games framework, which provides highly optimized end-to-end GPU implementation of several RL algorithms [35]. These implementations vectorize observations and actions on the GPU allowing to fully exploit the parallelization provided by the developed simulator. As represented in Fig. 1, an efficient wrapping module has been implemented to support the simulator which takes care of transferring data between the simulator and the RL algorithm, taking advantage of the

TABLE I

HYPERPARAMETERS OF PPO USED FOR THE EXPERIMENTS. THE STEPS BEFORE PPO UPDATE ARE SUMMED OVER ALL ENVIRONMENTS AND $lin(x)$ IS A LINEAR SCHEDULE STARTING AT x AND ENDING AT 0

Hyperparameter	Value	Hyperparameter	Value
Total simulation steps	$5 \cdot 10^5$	Minibatch size	256
Steps before update	1024	Update epochs	4
Discount factor γ	0.995	λ_{GAE}	0.95
Clip range value func.	0.2	Clip range	$lin(0.1)$
Value function coeff.	0.5	Entropy coeff.	0.0
Max. gradient norm	0.5	Learning rate	$lin(2.5 \cdot 10^{-4})$

primitives provided by Warp for interfacing with PyTorch framework used by rl_games. The latter allow for CUDA interoperability between Warp arrays and PyTorch tensors without copying the underlying data.

The interface between the developed simulator and rl_games package is also used to calculate, transform and transfer the necessary data for the RL process. Namely, episode rewards are calculated and scaled, observations are extracted from the simulation, and actions are applied when calling the simulation's step function once per simulation step. The interface also checks for termination conditions, i.e. the limit on the number of steps and task fulfillment conditions.

For each variable required in the calculations (e.g. vertices, faces, tetrahedrons and tensors containing the information for the RL agent) one GPU array is created and data from each environment is concatenated and stored in the array.

IV. EXPERIMENTS

The experiments have the dual objective of evaluating the performance of the FF-SRL simulator alone and the integration into the RL training process. We are considering tissue retraction since it is a widely used task in validating autonomous RASS, as described in Section II. The environment, presented in Fig. 2, replicates the characteristics of the one proposed in [8, 10, 17, 19]. In addition to the RASS tool, there is a camera with the point of view shown in the figure, a rigid base in which an area of interest is presented and to which the soft tissue is fixed along the side furthest from the camera. The movement of the tool is implemented in Cartesian space, as described in Section III-A.2. The proposed method is compared with LapGym [8] since it provides a reference implementation of the tissue retraction task with a realistic CPU simulation of the deformable tissue based on a finite element model.

We initially evaluate only the simulation performance and scaling capabilities of the proposed simulator in terms of environments running in parallel and as the complexity of the biomechanical model of the simulated tissue increases (obtained by increasing the number of vertices of the tissue model). For each environment, we consider the fps generated from $n_{iter} = 500000$ simulation steps performed without any RL training process running. The measurements do not consider initialization times to ensure a fair comparison with the data obtained during RL training [9].

TABLE II

FRAMES PER SECOND AS THE NUMBER OF ENVIRONMENTS RUNNING IN PARALLEL (LEFT) OR THE COMPLEXITY OF THE BIO-MECHANICAL MODEL (RIGHT) INCREASES FOR THE SIMULATION ENVIRONMENTS CONSIDERED. THE REPORTED VALUES ARE IN THE FORM MEAN ± STANDARD DEVIATION CALCULATED OVER 5 RUNS INITIALIZED WITH RANDOM SEEDS.

Number of Enviroments	Simulation Only		RL Setup	
	FF-SRL	LapGym	FF-SRL	LapGym
1	2964.6 ± 40.6	77.7 ± 0.7	434.3 ± 8	68 ± 1.1
4	9093.2 ± 132.8	205.8 ± 7.2	1527.3 ± 34	173.8 ± 8.4
8	14021.3 ± 193.4	226.1 ± 11.5	2636.7 ± 40.7	192.3 ± 9.9
16	19032.9 ± 98.1	–	3955.4 ± 202.1	–
32	21662.2 ± 145.3	–	4992 ± 335.7	–
64	23728.7 ± 314	–	–	–
80	23932.7 ± 317.4	–	–	–

Number of Tetrahedral	Simulation Only	
	FF-SRL	LapGym
1170	2964.6 ± 40.6	77.7 ± 0.7
1431	2798.3 ± 118	73.6 ± 2.1
2880	2751.9 ± 119.3	48.2 ± 1.3
9729	2302.2 ± 44.5	37.7 ± 0.4
52359	905.5 ± 29	9.6 ± 0.1

We then evaluate the FF-SRL simulator applied to RL training, considering the first phase of the tissue retraction task, which consists of reaching a point on the surface of the soft tissue. This choice is consistent with the approach in [6] and replicates the task of reaching anatomical areas of interest which is fundamental for most surgical activities, as adopted by [4, 5]. In this task, the instrument starts from a fixed point above the tissue and reaches a fixed target point on the tissue surface. The training is based on the Proximal Policy Optimization (PPO) algorithm [36] since it is considered a suitable baseline, given its wide successful applications in heterogeneous fields [9], including RASS [7, 8, 10, 22]. However, the simulation conforms to OpenAI *gym* [37] standard and can therefore be applied to other RL algorithms and libraries in a straightforward way.

Although FF-SRL supports the use of generated endoscopic images, we consider state observations that contain the position of the instrument and the target point to be reached, combined with the reward $R = w_l l + w_d d + w_s s$; where l is the distance between the end-effector and the target, d is the change of l with respect to previous simulation step and s is a success flag normally at 0 and set to 1 when the distance $l < 3$ mm. The weights w_l, w_d, w_s are set to -1, -10 and 100, respectively.

The training is carried out for $n_{iter} = 500000$ steps using the parameters reported in Table I, replicating those used in [8]. The public code repository also provides further details on parameters and implementation choices. Performances are measured by considering the overall training time and analyzing the rewards obtained. All experiments are performed on a laptop with an Intel i7-9750HF CPU @ 2.60GHz, 16 GB RAM, and an NVIDIA RTX 2060 Mobile GPU with 6GB VRAM. At the time of writing, this configuration is low-range and easily accessible to most researchers.

V. RESULTS AND DISCUSSIONS

All the data reported refers to the statistics calculated on 5 different executions, considering different initialization seeds. In Table II (left), we report the performance of FF-SRL and the baseline considered (LapGym [8]), in terms of average and standard deviation of frames per second (fps) obtained as a function of the number of environments executed in parallel. The first two columns show the performance considering only the simulation, while the others also consider the RL training. On the system configuration considered, FF-SRL is able to run up to 80 environments in parallel, providing a constant increase in the throughput, starting from the 2964.6 ± 40.6 obtained with a single environment, up to 23728.7 ± 314 with 64 environments, and then saturating the resources available at 80 environments with a result of 23932.7 ± 317.4, slightly above the configuration with 64 environments. We also ran FF-SRL on an NVIDIA A100 GPU and managed to run 1125 environments in parallel. This demonstrates how FF-SRL environment can scale even on high-performance computing systems.

For reference, the LapGym's simulation alone with a single environment is capable of achieving 77.7 ± 0.7 fps, more than 38x slower than FF-SRL. A similar performance difference is also observed if we consider 4 and 8 environments running in parallel, where FF-SRL provides over 40x and 60x speedup, respectively. This is an expected result, since LapGym is based on the SOFA simulation framework, which relies on a bio-mechanical finite element model solved entirely on the CPU.

When evaluating pure simulation performance, it is interesting to analyze the fps of a single environment as the complexity of the biomechanical simulation varies, as reported in Table II (right). In the second row, we can see how an increase of approximately 20% in the number of vertices of the tissue model leads to a reduction in fps of about 5% for both simulators. The situation changes starting from the third row, in which a model with approximately 2.5 times the vertices of the initial one leads to a reduction in fps of around 8% for FF-SRL, while for Lapgym the reduction is around 40%. The last two rows of the tables confirm the capability of FF-SRL to manage models with an even higher number of vertices, i.e. 9729 and 52359, respectively, unlike Lapgym, which sees performance halved and even collapses below 10 fps in the case of the most complex model considered. This result demonstrates how the optimized XPBD implementation on which FF-SRL is based can optimally handle the complex simulations of deformable structures required in RASS applications

Considering the performance in terms of RL training, we report in the third and fourth columns of Table II (left), the statistics on the fps obtained from FF-SRL and Lapgym, respectively. Even in this context, we can observe how FF-SRL obtains a higher fps throughput than LapGym, provid-

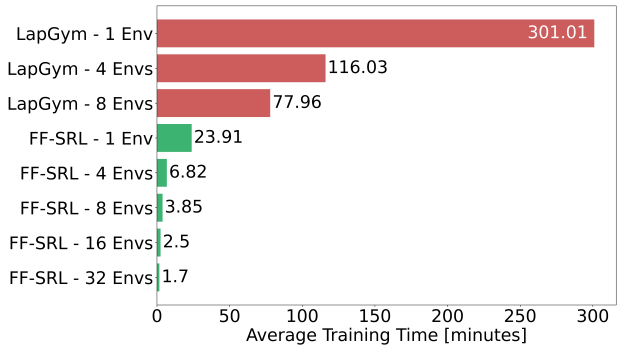

Fig. 3. Average training time for $n_{iter} = 500000$ steps, considering different number of environment running in parallel.

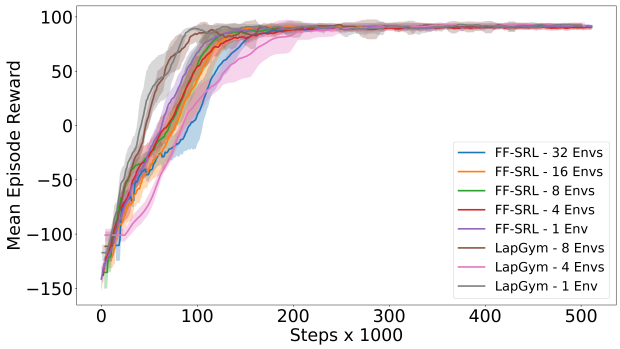

Fig. 4. Average episode reward with respect to steps.

ing significant performance increases. If we consider a single environment, FF-SRL yields a performance increase of over 6x with respect to Lapgym. If we consider 4 environments, we have an increase of more than 8.5x, which becomes over 13.5x with 8 environments, the maximum number of environments supported by LapGym on the hardware configuration considered. However, FF-SRL can also run 16 and 32 environments in parallel, providing a further increase in performance compared to the configuration of 8 Lapgym environments, of 20 and 26 times respectively. We should note that we were not able to run more than 32 FF-SRL environments during RL training on the considered system. This reduction compared to the simulation alone, capable of reaching 80 environments, is linked to the limits of the graphics memory. The limit of 80 simulation environments in parallel is rather connected to the available system memory, since instantiating the simulation requires creating large arrays as described in Section III-B. It will certainly be an element that we will improve, optimizing the data types used and memory management by limiting the initial copies of data between Warp and Pytorch.

As seen in Table II (left), the FPS reduction from simulation only to full RL pipeline is significant in FF-SRL as the simulation steps take only small fraction of the time needed for full RL processing. For Lapgym the simulation steps time is a major contribution to the RL processing time and therefore the reduction is less pronounced.

Considering the overall RL training times shown in Fig. 3, the proposed simulator with a single environment completes training in less than 24 minutes compared to over 5 hours for the baseline. If we consider the maximum possible performances, we can complete the training in less than 2 minutes (32 environments configurations), guaranteeing a speedup of over 45 times compared to the 8 LapGym environment condition. Even comparing the 8 FF-SRL environments with the equivalent LapGym number, the speedup is still more than 20 times. It is important to observe how FF-SRL by going from 1 to 4 environments allows to reduce the training time by over 3.5 times, while LapGym achieves a reduction of approximately 2.6 times. Comparing the transition from 4 to 8 environments, FF-SRL obtains a time reduction of approximately 44%, compared to approximately 33% obtained by LapGym. These results confirm the scalability of FF-SRL with respect to the number of parallel environments.

Fig. 4 shows the curves of the average reward obtained per episode compared to the total number of steps. All the configurations considered are able to train an RL agent capable of solving the task considered (corresponding to a reward greater than 80). The convergence of RL training in FF-SRL and Lapgym is closely overlapping, guaranteeing a correct solution of the task already after 250,000 steps. This fact allows us to observe that the times reported in Fig. 3, which consider all 500,000 training steps, are overestimated and probably about half the time is sufficient to solve the considered task.

In this work, we did not optimize the training hyperparameters to ensure a fair comparison with the baseline. Optimization of these parameters could favor a further increase in performance and allow the management of even more complex tasks, such as complete tissue retraction or tasks that require the use of two instruments. These tasks have already been demonstrated to be able to be transferred effectively between simulator and reality [7, 10, 20]. Tests with the real robot will be carried out in the future to demonstrate the capabilities of FF-SRL to also interface with real robotic systems, exploiting ROS middleware. In the future, we will also focus on the optimization of some technical aspects, such as improving the implementation of inverse kinematics to support all degrees of freedom of the dVRK and extending the evaluation to visual RL methods.

VI. CONCLUSION

In this paper, we presented FF-SRL, a GPU-based simulation environment for robotic surgery that leverages an advanced XPBD simulation of deformable tissue. We showed that FF-SRL can significantly speed up the RL training process for surgical tasks, achieving higher frame-rates and faster training time than other available simulators. We also demonstrated the scalability and efficiency of our simulation environment, which can run on a single low-end GPU device. Our work opens up new possibilities for developing autonomous surgical systems using RL techniques, as well as for studying the interaction between robots and soft tissue. We hope that our code and simulator will be useful for the research community and foster further advances in this field.

ACKNOWLEDGMENT

The authors want to thank Paul Maria Scheikl from Friedrich-Alexander-Universität Erlangen-Nürnberg and Miles Macklin from Nvidia for valuable support.

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, S. P. DiMaio, *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," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 6434–6439.

[3] P. Fiorini, K. Y. Goldberg, Y. Liu, and R. H. Taylor, "Concepts and trends in autonomy for robot-assisted surgery," *Proceedings of the IEEE*, vol. 110, no. 7, pp. 993–1011, 2022.

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

[5] 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)*. IEEE, 2021, pp. 1821–1828.

[6] V. M. Varier, D. K. Rajamani, F. Tavakkolmoghaddam, A. Munawar, and G. S. Fischer, "Ambf-rl: A real-time simulation based reinforcement learning toolkit for medical robotics," in *2022 International Symposium on Medical Robotics (ISMR)*. IEEE, 2022, pp. 1–8.

[7] E. Tagliabue, A. Pore, D. Dall'Alba, E. Magnabosco, M. Piccinelli, and P. Fiorini, "Soft tissue simulation environment to learn manipulation tasks in autonomous robotic surgery," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3261–3266.

[8] 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.

[9] 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," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[10] 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, 2023.

[11] 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)*. IEEE, 2021, pp. 7737–7743.

[12] K. L. Schwaner, D. Dall'Alba, P. T. Jensen, P. Fiorini, and T. R. Savarimuthu, "Autonomous needle manipulation for robotic surgical suturing based on skills learned from demonstration," in *2021 IEEE 17th international conference on automation science and engineering (CASE)*. IEEE, 2021, pp. 235–241.

[13] Z. J. Hu, Z. Wang, Y. Huang, A. Sena, F. R. y Baena, and E. Burdet, "Towards human-robot collaborative surgery: Trajectory and strategy learning in bimanual peg transfer," *IEEE Robotics and Automation Letters*, 2023.

[14] D. Meli, P. Fiorini, and M. Sridharan, "Towards inductive learning of surgical task knowledge: A preliminary case study of the peg transfer task," *Procedia Computer Science*, vol. 176, pp. 440–449, 2020.

[15] M. Hwang, B. Thananjeyan, D. Seita, J. Ichnowski, S. Paradis, D. Fer, T. Low, and K. Goldberg, "Superhuman surgical peg transfer using depth-sensing and deep recurrent neural networks," *arXiv preprint arXiv:2012.12844*, 2020.

[16] D. Meli, E. Tagliabue, D. Dall'Alba, and P. Fiorini, "Autonomous tissue retraction with a biomechanically informed logic based framework," in *2021 International Symposium on Medical Robotics (ISMR)*. IEEE, 2021, pp. 1–7.

[17] E. Tagliabue, D. Meli, D. Dall'Alba, and P. Fiorini, "Deliberation in autonomous robotic surgery: a framework for handling anatomical uncertainty," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 11 080–11 086.

[18] C. D'Ettorre, S. Zirino, N. N. Dei, A. Stilli, E. De Momi, and D. Stoyanov, "Learning intraoperative organ manipulation with context-based reinforcement learning," *International Journal of Computer Assisted Radiology and Surgery*, vol. 17, no. 8, pp. 1419–1427, 2022.

[19] E. Tagliabue, D. Dall'Alba, M. Pfeiffer, M. Piccinelli, R. Marin, U. Castellani, S. Speidel, and P. Fiorini, "Data-driven intra-operative estimation of anatomical attachments for autonomous tissue dissection," *IEEE Robotics and Automation Letters*, pp. 1856–1863, 2021.

[20] A. Pore, E. Tagliabue, M. Piccinelli, D. Dall'Alba, A. Casals, and P. Fiorini, "Learning from demonstrations for autonomous soft-tissue retraction," in *2021 International Symposium on Medical Robotics (ISMR)*. IEEE, 2021, pp. 1–7.

[21] P. M. Scheikl, B. Gyenes, T. Davitashvili, R. Younis, A. Schulze, B. P. Müller-Stich, G. Neumann, M. Wagner, and F. Mathis-Ullrich, "Cooperative assistance in robotic surgery through multi-agent reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1859–1864.

[22] A. Pore, D. Corsi, E. Marchesini, D. Dall'Alba, A. Casals, A. Farinelli, and P. Fiorini, "Safe reinforcement learning using formal verification for tissue retraction in autonomous robotic-assisted surgery," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4025–4031.

[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] Y. Long, W. Wei, T. Huang, Y. Wang, and Q. Dou, "Human-in-the-loop embodied intelligence with interactive simulation environment for surgical robot learning," *IEEE Robotics and Automation Letters*, 2023.

[25] 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," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1875–1882.

[26] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, *SOFA: A Multi-Model Framework for Interactive Physical Simulation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 283–321.

[27] P. Schegg, E. Ménager, E. Khairallah, D. Marchal, J. Dequidt, P. Preux, and C. Duriez, "Sofagym: An open platform for reinforcement learning based on soft robot simulations," *Soft Robotics*, pp. 410–430, 2023.

[28] M. Macklin, M. Müller, and N. Chentanez, "Xpbd: Position-based simulation of compliant constrained dynamics," in *Proceedings of the 9th International Conference on Motion in Games*, 2016, p. 49–54.

[29] 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*, ser. SCA '19, 2019.

[30] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and Z. Corse, "Local optimization for robust signed distance field collision," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 3, no. 1, may 2020.

[31] M. Müller, M. Macklin, N. Chentanez, S. Jeschke, and T.-Y. Kim, "Detailed rigid body simulation with extended position based dynamics," *Computer Graphics Forum*, vol. 39, no. 8, pp. 101–112, 2020.

[32] M. Macklin and M. Muller, "A constraint-based formulation of stable neo-hookean materials," in *Motion, Interaction and Games*, 2021.

[33] M. Müller, M. Macklin, N. Chentanez, and S. Jeschke, "Physically based shape matching," *Computer Graphics Forum*, pp. 1–7, 2022.

[34] M. Macklin, "Warp: A high-performance python framework for gpu simulation and graphics," https://github.com/nvidia/warp, March 2022, nVIDIA GPU Technology Conference (GTC).

[35] D. Makoviichuk and V. Makoviychuk, "rl-games: A high-performance framework for reinforcement learning," https://github.com/Denys88/rl_games, May 2021.

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

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