2026 IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob 2026)
August 1-4, 2026. Edmonton, Canada

Simulation of Surgical Suturing Using Position-Based Dynamics and the Material Point Method for Robot Reinforcement Learning

Tleukhan Mussin[1,*], Yafei Ou[1,*], and Mahdi Tavakoli[1,2]

***Abstract*— Recent advances in robotics research have created a strong demand for high-performance simulators. Surgical robotics simulation faces unique challenges due to the need to model diverse objects, such as rigid instruments, soft tissue, and fluids. While many studies simulate sutures or soft tissue independently, only a few have considered the complete soft-tissue suturing scenario, including the contact between sutures and deformable tissue during suture insertion. Building on previous work, this paper presents a novel suturing simulation environment using sutures modelled by position-based dynamics (PBD) and soft bodies modelled by the material point method (MPM) while considering two-way contact with frictional and drag forces. We introduce a contact coupling method between the PBD suture and the MPM soft tissue, enabling visually plausible suture-tissue interactions. The simulator is optimized for GPU execution with parallel scenes using multiple CUDA streams, and we present a Reinforcement Learning (RL) environment for autonomous suturing sub-tasks, including needle insertion, driving, and extraction. Using ML-Agents, RL agents trained in the simulator show stable learning and achieve 80% and 68% success rates in needle insertion and extraction, respectively, under the strictest distance threshold.**

I. INTRODUCTION

Recent advances in robotics research have created a strong demand for high-performance, high-fidelity simulators, which serve both as playgrounds for testing robotics control and automation algorithms and as sources for synthesizing virtual training data and experiences. However, despite the availability of many robotics simulation platforms, surgical robotics simulation faces unique challenges due to the need to model diverse objects, such as rigid instruments, soft tissue, and fluids.

The modelling of rope and rod-like objects in surgical simulators, particularly sutures, has been explored in existing studies. Surgical simulators and computer graphics literature typically represent such objects using rigid-body chains, mass-spring systems, position-based dynamics (PBD), or Cosserat rod formulations to achieve real-time performance [1]–[6]. For example, the Asynchronous Multi Body Framework (AMBF) implements a sequential impulse (SI) solver for simulating a chain of rigid bodies for representing sutures [4]. In [6], extended PBD (XPBD) is used for both suture and soft tissue simulation in a VR-based surgical simulator. Some studies have also explored using the finite-element method (FEM), although such implementations are expected to be more computationally expensive and less numerically stable [1]. In a more recent study [5], Isaac Sim is used for building surgical simulation scenes, where three different methods for modelling suture threads are compared, including using rigid-body chain, FEM deformable body, and PBD particle body. Recent studies have explored training autonomous suturing agents in simulators [7], [8].

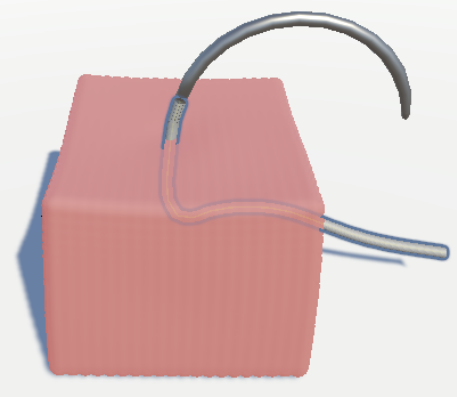

Fig. 1. Simulated suture in soft tissue.

While a large number of studies have explored the simulation of surgical sutures and biomedical soft tissue independently, only a few have considered the complete soft-tissue suturing scenario, including the contact between sutures and deformable tissue during suture insertion. Some frameworks primarily focus on rigid-body capabilities, and the suturing scenes do not consider soft-tissue [4], [7]. Simulators that consider more realistic scenarios often need to include different maths or solvers for both the suture and the tissue. However, modelling contact during suture insertion within tissue, as well as the coupling between two solvers, presents additional challenges, even when compared to regular rope-soft-body contact. In [2], the tissue is simulated with FEM, while the suture thread uses a 1D mass-spring model kinematically constrained to the needle path. This ignores the dynamic interaction between the suture and the tissue, and treats the suture as a "light" object that deforms entirely according to the needle path through the tissue. In [1], sutures are modelled as serially-linked beam elements using FEM and interact with hexahedral FEM tissue. Although both use FEM, insertion contact still requires separate modeling: bilateral constraints keep the needle shaft and suture thread

This research was supported by the Canada Foundation for Innovation (CFI), the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Canadian Institutes of Health Research (CIHR), and Alberta Innovates. *(Corresponding author: Yafei Ou.)*

[1]Tleukhan Mussin, Yafei Ou, and Mahdi Tavakoli are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada (e-mail: yafei.ou@ualberta.ca).

[2]Mahdi Tavakoli is also with the Department of Biomedical Engineering, University of Alberta, Edmonton, Alberta, Canada.

*Tleukhan Mussin and Yafei Ou contributed equally to this work.

within the path created by the needle tip.

A simpler approach that requires less contact modelling and coupling between different simulation solvers is to use PBD for both tissue and suture simulation [3], [6]. Suture-in-tissue can be modelled with constraints that directly fall under the regular PBD framework. For example, authors of [3] use PBD for modelling both tissue and suture and constrain suture particles so that they may slide along the needle path but cannot move away in orthogonal directions during suturing insertion. However, although PBD sutures are often acceptable, tissue realism in PBD relies on constraints and non-physical parameters rather than true physical modeling. This can introduce additional challenges for surgical robotics research, such as PBD parameter tuning to match the simulation with real-world tissue behavior.

To better simulate surgical suturing and allow robot learning with parallel simulations, this paper builds on previous work [9] and introduces a novel suturing simulation environment using sutures modelled by PBD and soft bodies modelled by the material point method (MPM) while considering dynamic contact, including frictional and drag forces. The usage of MPM soft tissue preserves physical realism compared to using PBD tissue. The main contributions of this work are:

- We extend our previous CRESSim-MPM framework with a PBD-based suture model to support flexible suture deformation in surgical suturing, as in Fig. 1.
- We introduce a novel contact coupling method between the PBD suture and the MPM soft tissue, enabling stable and realistic suture-tissue interactions.
- We optimize the simulator for GPU execution, build a parallel RL environment, and validate it by training agents for autonomous suturing sub-tasks.

It is worth noting that the RL task focuses only on validating that the simulator enables training vision-based policies while simulating soft tissue and suture contact, without considering the complete suturing task. This is important for developing agents that account for suture-tissue contact and for transferring to the real world in future work.

II. RELATED WORK

A. Surgical Robot Simulation

This work is related to the recent trend in developing surgical simulators and autonomous surgical robot agents using simulators. To build realistic surgical simulators, earlier studies include AMBF [4], [10], ATAR [11], and V-Rep Simulator for the dVRK [12]. More recent works, such as dVRL [13], UnityFlexML [14], AMBF-RL [15], LapGym [16], ORBIT-Surgical [8], Surgical Gym [17], FF-SRL [18], SurRoL [19] introduced platforms with specific focus on surgical robot learning. However, most of them consider rigid body dynamics or cannot simulate soft-tissue suturing easily and efficiently due to the use of FEM, as discussed in Section I. This work proposes an alternative approach by combining MPM soft tissue and PBD suture simulation, addressing some of the limitations in existing studies.

B. Autonomous Suturing

Suturing typically consists of multiple interdependent substeps, including needle localization and grasping, suture thread detection, needle insertion, needle driving, needle extraction, and knot tying. However, much of the existing research has focused on automating only two or three of these components.

Similarly, in this work we assume that the needle has already been grasped, and focus only on the needle insertion, driving, and extraction phases of suturing. Early approaches in this direction learn and replay open-loop trajectories from human demonstrations [20], [21], which lack feedback during execution and are sensitive to deviations from the demonstrated motion. Subsequent methods incorporate real-time geometric planning [22] or IL [23] to improve robustness, yet still rely on simplified planar or rigid tissue models that neglect internal physics of soft tissue, such as deformation forces and resistance. More recent frameworks employ RL to address longer-horizon suturing tasks and needle–suture manipulation [7], [8]. Although these approaches improve realism through physics-based simulation and task decomposition, they do not explicitly model internal force-based needle–tissue interactions. Moreover, [7] was not able to perform needle insertion using both PPO and SAC. In contrast, [24] incorporates direct needle–tissue interaction modeling via hybrid batch reinforcement learning, but is constrained by limited parallelism and CPU-based training in the da Vinci Skills Simulator. Finally, [25] demonstrates a compelete suturing using various IL approaches, but training was conducted in real-world settings with limited environmental variability.

III. ROPE AND TISSUE SIMULATION

A. Position-Based Dynamics

In position-based dynamics (PBD), a continuum body is discretized into particles. Consider each particle i with its mass m_i, position $\mathbf{x}_i$ and velocity $\mathbf{v}_i$. The equation of motion is derived from Newton's law $\dot{\mathbf{v}}_i = \mathbf{f}_i/m_i$, where $\mathbf{f}_i$ is the total force on the particle. We can apply symplectic Euler integration to solve the system at each step:

$$\begin{aligned} \mathbf{v}_i(t+\Delta t) &= \mathbf{v}_i(t) + \Delta t \frac{1}{m_i}\mathbf{f}_i(t), \\ \mathbf{x}_i(t+\Delta t) &= \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t+\Delta t) \end{aligned} \tag{1}$$

Instead of relying on the physical relationship between deformation and internal forces, kinematic constraints are used in PBD. For instance, two neighboring particles can form a spring-like constraint. PBD constraints are solved by projecting the predicted particle positions from (1) on the constraint manifold to correct the prediction. The predicted particle positions are iteratively corrected by moving them toward satisfying all the constraints. Consider a total of M constraints:

$$C_1(\mathbf{x}) \succ 0, \quad C_2(\mathbf{x}) \succ 0, \quad \ldots \quad C_M(\mathbf{x}) \succ 0, \tag{2}$$

where $\mathbf{x}$ is the concentrated vector of all the particle positions. We can employ the non-linear Gauss–Seidel approach

to solve the positional corrections. Specifically, we want to find a correction $\Delta\mathbf{x}$ such that $C(\mathbf{x}+\Delta\mathbf{x}) \succ 0$. The constraints are linearized individually, approximated by

$$C(\mathbf{x}+\Delta\mathbf{x}) \approx C(\mathbf{x}) + \nabla C(\mathbf{x})\Delta\mathbf{x} \succ 0. \tag{3}$$

If $\Delta\mathbf{x}$ is restricted to be in the direction of ∇C, $\mathbf{x}$ will move toward converging to satisfying the constraints:

$$\Delta\mathbf{x} = -\lambda\mathbf{M}^{-1}\nabla C(\mathbf{x}), \tag{4}$$

where $\mathbf{M} = \text{diag}(m_1, m_2, \ldots, m_N)$ for all N particles. Substituting (4) into the equality version of (3) yields λ:

$$\lambda = \frac{C(\mathbf{x})}{\nabla C(\mathbf{x})^T\mathbf{M}^{-1}\nabla C(\mathbf{x})}. \tag{5}$$

This works for the equality constraints directly. Inequalities are handled by checking whether $C(\mathbf{x}) \geq 0$ to simply skip it in the iteration.

The extended PBD (XPBD) method additionally adds compliance to the constraints, achieving stiffness that is independent of the simulation time step, unlike in PBD. Each constraint is augmented with a compliance parameter $\alpha \geq 0$. The new Lagrange multiplier increment can be computed as

$$\Delta\lambda = \frac{-C(\mathbf{x}) - \alpha\lambda^{(k)}}{\nabla C(\mathbf{x})^T\mathbf{M}^{-1}\nabla C(\mathbf{x}) + \alpha}. \tag{6}$$

Here, $\lambda^{(k)}$ denotes the Lagrange multiplier value at the k-th iteration. The positional correction is then given by

$$\Delta\mathbf{x} = \mathbf{M}^{-1}\nabla C(\mathbf{x})\Delta\lambda, \tag{7}$$

and the multiplier is updated as $\lambda^{(k+1)} = \lambda^{(k)} + \Delta\lambda$. Setting the compliance parameter $\alpha = 0$ results in XPBD being equivalent to the PBD formulation in (4) and (5).

B. PBD Suture Simulation

A surgical suture is a rope-like object that can be discretized into connected particles and simulated using PBD. Ignoring particle orientation, we use only distance constraints between neighboring particles and bending constraints among triplets of adjacent particles.

Mathematically, for neighbouring particles i and $i+1$,

$$C(\mathbf{x}_i, \mathbf{x}_{i+1}) = \|\mathbf{x}_i - \mathbf{x}_{i+1}\| - d_0 = 0, \tag{8}$$

where d_0 is the constrained distance between the particles. For triplets of adjacent particles $i-1$, i and $i+1$, bending constraints are formed as

$$C(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = \theta(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) - \theta_0 = 0, \tag{9}$$

where

$$\theta(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = \arccos\left(\frac{\mathbf{x}_{i-1}-\mathbf{x}_i}{\|\mathbf{x}_{i-1}-\mathbf{x}_i\|}, \frac{\mathbf{x}_{i+1}-\mathbf{x}_i}{\|\mathbf{x}_{i+1}-\mathbf{x}_i\|}\right), \tag{10}$$

and θ_0 is the rest angle. Both types of constraints are formed as XPBD constraints with compliance parameters.

This approach allows simulating rope-like objects without the need to consider particle orientations. Alternative approaches that lead to more accurate simulations account for thread twisting constraints and add higher computational complexity. Furthermore, the current implementation does not consider self-collision, which can be added with additional constraints. Implementing these constraints and integrating them into our framework is straightforward if needed in future work, such as for simulating knot-tying.

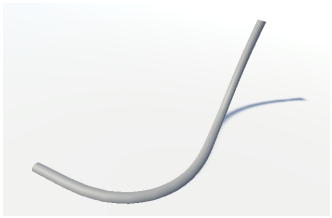

Fig. 2. PBD suture simulation aligned with a rigged cylinder for visual representation.

Visually, a narrow cylinder rigged with bones is used to represent and animate the suture. At each time step, the bones are aligned with the PBD particle positions to produce a visual simulation of the suture. An example of the simulated suture is shown in Fig. 2.

C. MPM Soft Tissue and Needle Contact Coupling

Existing work, such as [4], has explored the use of PBD for suture simulation in surgical scenes. However, most of the existing studies are limited to contact models between PBD particles and rigid body solvers, and assume suturing through rigid apertures. This work aims to achieve more realistic surgical simulation by enabling interaction between sutures and deformable soft tissue modeled using MPM. This requires a novel contact modelling approach between the PBD suture and the MPM soft body.

In the MPM, a soft body is discretized into Lagrangian material points (moving particles) to store local material data, such as mass, velocity, stress tensor, and deformation gradient, while the deformation computation happens in an Eulerian (fixed) grid. The MPM formulation follows 4 steps: **(1) Particle to grid (P2G)**: Material data at the particles are distributed to the grid using an interpolation shape function. **(2) Grid update**: the momentum of each grid node is integrated. **(3) Grid to particle (G2P)**: The updated node momenta are interpolated back to the particles. Material properties such as the deformation gradient and stress are also updated. **(4) Reset grid**: reset grid node data to zeros.

Our previous work [9] introduced CRESSim-MPM, a CUDA-accelerated MPM simulation library designed for surgical soft-tissue simulation. We also proposed a novel coupling strategy between a rigid needle and an MPM-based deformable body. The needle is modeled as a 1D manifold (*e.g.,* an arc) when in contact with the soft tissue, allowing insertion and dragging, as indicated by the downward and leftward arrows in Fig. 3, respectively. Across all experiments, the tissue model was parameterized using Lamé elastic constants $\lambda = 714285.8$ and $\mu = 178571.4$, where λ represents the volumetric stiffness and μ represents the shear stiffness.

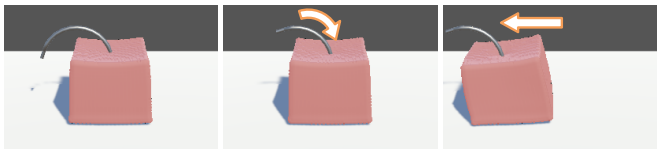

Fig. 3. Arc-shaped suture needle in contact with MPM soft body.

D. Coupling between MPM Soft Tissue and PBD Suture

We can formulate the contact between the PBD suture and the soft tissue in the following manner: we construct virtual line segments between neighboring PBD particles as 1D geometries, similar to the needle representation, and constrain their motion accordingly, as shown in Fig. 4.

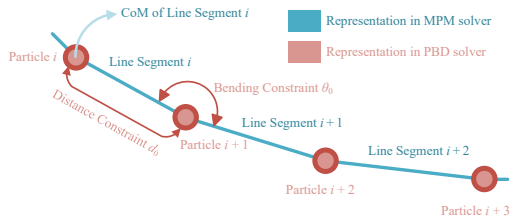

Fig. 4. Suture representations in both PBD and MPM solvers for contact coupling between PBD suture and MPM soft body.

Specifically, for particles i and $i+1$, we create a `ConnectedLineSegments` geometry in CRESSim-MPM to apply contact forces from the PBD suture to the MPM soft tissue. To achieve two-way coupling, momentum changes induced by contact are applied as impulses from the MPM tissue back to the virtual line segments.

As these line segments are virtual from the perspective of the PBD solver, we found it sufficient to apply the resulting impulse directly to particle i in most cases. This corresponds to assuming that the center of mass (CoM) of the virtual line segment coincides with particle i, as shown in Fig. 4.

On the MPM solver side, contact resolution is performed during the Eulerian grid update. The resulting impulses are recorded and applied to the PBD particles after the MPM solver step and before the next PBD solver step. Example snapshots of suturing in soft tissue are shown in Fig. 5.

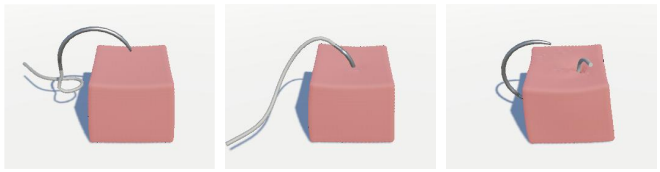

Fig. 5. Snapshots from a suturing demo scene

E. Parallel Simulation

The capability to simulate many instances of similar scenes is important for robot learning. PBD-based thread simulation is not computationally expensive, even without GPU acceleration, especially due to the relatively small number of particles used to model rope-like objects. As a result, a large number of threads can be simulated in parallel, making the MPM soft tissue simulation the primary bottleneck in large-scale parallel simulation.

To better utilize GPU resources, we further introduce improvements to CRESSim-MPM. In particular, `Scene`-related data transfers and solver computations are separated into multiple CUDA streams, leading to improved GPU utilization across solver stages by overlapping the stepping of different `Scenes` under certain settings. In practice, distributing `Scenes` to two CUDA streams on an NVIDIA RTX 4090 usually results in improved performance compared to using only one stream. More detailed simulation performance evaluations will be presented in Section V-A.

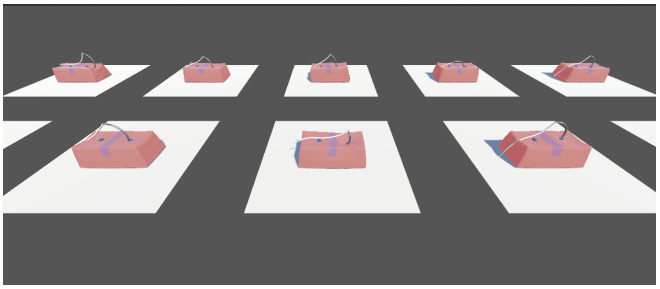

Fig. 6. Multiple training parallel environments.

IV. REINFORCEMENT LEARNING ENVIRONMENT FOR SURGICAL SUTURING

A. Task Definition

In this work, we develop a simplified surgical suturing RL task to demonstrate that autonomous agents can be trained using the simulation framework.

The objective is for a robotic manipulator to autonomously execute key sub-tasks of a suturing motion, including needle insertion at an entry marker, needle navigation inside the tissue, and needle extraction at a specified exit marker. In this study, we do not use a surgical robotic manipulator agents to grasp or manipulate the needle, but instead, the needle itself is treated as the agent. Therefore, we make several simplifying assumptions: the needle is already grasped prior to task execution, the suturing motion is confined to a planar workspace, and no re-grasping or hand-off of the needle is performed. The suture thread is attached to the needle end and therefore follows the needle path in the tissue.

The reward function is divided into three sequential phases, each corresponding to a separate sub-task of the suturing motion. Let scalars w_e, w_θ, w_x, w_T and $b_e, b_x, b_n > 0$ be the manually specified weights and reward scales for the three phases. In Phase 1, we guide the needle tip to the entry marker using a distance-based shaping term and a success bonus. Let t denote the time step. We define $d_{\text{entry}}(t)$ as the Euclidean distance between the needle tip and the entry marker. In addition, we introduce an indicator function $\mathbb{I}_{e(t)} \in \{0,1\}$, which equals to 1 when the needle tip reaches the entry marker at time t, and 0 otherwise. This reward is

$$r_t^{(1)} = -w_e\, d_{\text{entry}}(t) + b_e\, \mathbb{I}_{e(t)}. \tag{11}$$

Once the needle is in the tissue, Phase 2 encourages rotational needle driving while moving towards the exit marker. Let o denote the needle's arc intersection point (needle origin), and let e denote the entry marker. We define the angle $\theta(t)$ between vectors $\overrightarrow{oe}$ and $\overrightarrow{op_{\text{tip}}(t)}$, where $p_{\text{tip}}(t)$ is the needle-tip position (Fig. 7). Minimizing $\theta(t)$ encourages the needle to rotate about its origin and follow the desired arc through the tissue. Additionally, we panalize any translational motion inside the tissue to reduce needle sliding with $v_{\text{trans}}(t)$. The Phase 2 reward is:

$$r_t^{(2)} = w_\theta\,\theta(t) + w_x\,d_{\text{exit}}(t) + b_x\,\mathbb{I}_{\text{x}(t)} - w_T\,\left\|v_{\text{trans}}(t)\right\|. \quad (12)$$

Here, $d_{\text{exit}}(t)$ is the inverse Euclidean distance between the needle tip and the exit marker, while $\mathbb{I}_{x(t)} \in \{0,1\}$ is an indicator for when the needle tip reaches the exit marker.

Finally, Phase 3 focuses on accurately extracting the needle from the exit marker. In this phase, we provide a terminal reward only upon successful task completion. With an indicator function $\mathbb{I}_{\text{f}(t)} \in \{0,1\}$:

$$r_t^{(3)} = b_n\,\mathbb{I}_{\text{f}(t)}. \quad (13)$$

At each time step, the environment returns the reward corresponding to the active phase, *i.e.*, $r_t = r_t^{\{1,2,3\}}$.

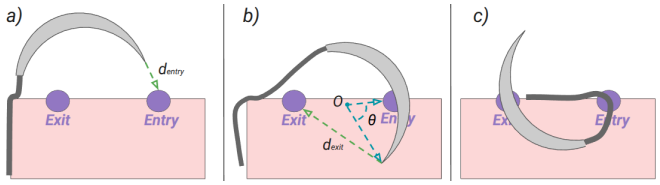

Fig. 7. Phases of the suturing sub-task in side view: (a) needle insertion, (b) needle driving, (c) needle extraction.

B. Training Settings

We utilize Unity's ML-Agents Toolkit, which enables training with both proximal policy optimization (PPO) and Soft Actor-Critic (SAC). The simulation runs with a fixed time step of 0.004 s, while the decision requester is configured to issue actions every 20 simulation steps. At each decision step, the agent observes a 250×250 RGB image, along with the needle's position and orientation. Observations are stacked over the current and two previous time steps and concatenated into a single vector. The inclusion of both visual and proprioceptive observations is important, since once the needle is inserted into the tissue, visual information alone is insufficient to infer its interaction with the tissue.

At the start of each episode, the tissue is initialized on a planar surface with a size of $2 \times 1 \times 1$ ($W \times H \times D$) expressed in **Unity units (UU)**. In our simulation, 1 UU is interpreted as 1 meter; however, this unit-to-meter mapping can be rescaled when matching the simulation to real-world measurements. The needle is spawned above it (Fig. 9a) and its initial position and orientation are randomized within bounded limits. To detect collisions between the needle tip and marker points, we used Unity's collision detection system, which continuously checks whether a collision is triggered by the tip. Each episode is limited to a maximum of 4,000 environment steps. We use a learning rate of 0.0003, a batch size of 1024, a replay buffer size of 10240, a constant entropy regularization coefficient of 0.01, and a discount factor of 0.95. To accelerate training, we run 10 parallel simulation environments concurrently. Both training and inference are performed on an NVIDIA RTX 4090 GPU.

V. RESULTS

A. Simulation Performance

To quantitatively evaluate the simulation performance, an example scene is built in Unity for profiling. This scene contains 10 parallel areas, each including a PBD suture with 20 particles and an MPM soft body with 20,000 material points. The 10 parallel areas correspond to 10 `Scenes` in the CRESSim-MPM library. The simulation step size is 0.004 seconds and the MPM integration step is 0.002 seconds, there are exactly two integration steps in the MPM solver for each `FixedUpdate` in Unity.

Usage of CUDA streams: We show that using separate CUDA streams in CRESSim-MPM increases the computational efficiency of MPM simulation. The example scene is run with a different number of CUDA streams used, and the MPM physics computation time is recorded. For better results, trials are conducted without rendering, which is also computationally heavy and competes with the MPM solver for GPU resources. The results over a 20-step window for each trial are shown in Table I. There is a significant performance improvement when increasing the number of CUDA streams from one to two. However, using more than two streams yields limited additional gains, likely due to bottlenecks in other GPU resources, such as register pressure.

TABLE I
MPM SOLVER STEP TIMES WHEN USING DIFFERENT NUMBERS OF CUDA STREAMS.

No. of CUDA streams	MPM physics step times (mean ± std, ms)
1	5.85 ± 0.15
2	2.10 ± 0.13
3	1.97 ± 0.17
4	1.70 ± 0.20

Computation time of individual components: We are also interested in the computational cost of each major component within a simulation step under standard simulation settings with soft-body rendering. To evaluate this, we collect computation times for MPM physics, PBD rope physics and visual alignment, and MPM soft-body rendering data computation at each step over a 20-step window. Two CUDA streams are used. The results are shown in Table II. It is worth noting that the rendering of the MPM soft body treats material points as individual points with color information and follows the screen-space rendering technique described in [26]. This is also computationally heavy and can compete

with the MPM solver for GPU resources, which may explain why MPM physics computation is significantly slower than that reported in Table I, although two CUDA streams are used in both cases. PBD-related computation is fast despite its CPU implementation and is not a bottleneck in the overall simulation.

TABLE II
COMPUTATION TIMES OF MAJOR COMPONENTS.

Component	Computation times (mean ± std, ms)
MPM physics	4.96 ± 0.35
PBD physics and visual	0.63 ± 0.07
MPM rendering	0.60 ± 0.70

B. Training Results

Fig. 8 shows the learning curve for agents trained with both SAC and PPO. These results indicate consistent policy improvement and stable convergence during training.

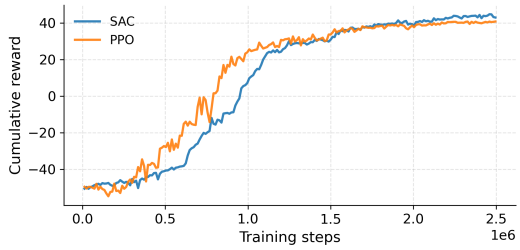

Fig. 8. Learning curves using PPO and SAC trained on the suturing task.

We further evaluate the trained agents in the simulated environment by measuring task success rates across two phases: (1) needle insertion, defined as the needle entering the designated entry point within a specified distance threshold, and (2) needle extraction, defined as the needle exiting the tissue through the designated exit point (Fig. 9). Results from 100 different episodes are recorded and summarized in Table III.

TABLE III
SUCCESS RATES FOR NEEDLE ENTRY AND EXIT UNDER DIFFERENT DISTANCE THRESHOLDS.

Metric	**0.075 UU**	**0.12 UU**	**0.18 UU**
Entry Point Success Rate (%)	80	85	91
Exit Point Success Rate (%)	68	74	85

In Table III, the reported thresholds are expressed in UU. As mentioned earlier, Unity Units (UU) refer to the engine's native, dimensionless units. We use the marker centers as ground-truth reference points, and upon needle tip extraction from a marker, we record the Euclidean distance between them. A distance of 0.075 UU corresponds to the diameter of the entry and exit markers; therefore, successful insertion or extraction within this threshold is considered the most accurate. The larger thresholds (0.12 UU and 0.18 UU) are used to evaluate how close the needle tip approaches the

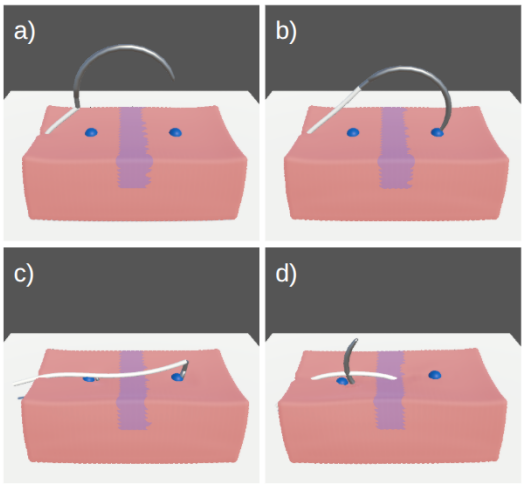

Fig. 9. Result of the trained RL policy performing suturing: (a) random initial needle position, (b) needle tip penetrating the tissue at the entry point, (c) needle tip reaching the exit point, and (d) completing suturing.

markers, with 0.12 UU indicating near-target performance and 0.18 UU representing a looser but still acceptable region. Across 100 episodes, the agent successfully inserts the needle within the most accurate region (0.075 UU) in 80 cases. As expected, the success rate increases with larger thresholds, reaching 91% within the 0.18 UU region, indicating that in only 9 out of 100 episodes the needle deviates significantly from the entry point at the beginning of the task. Exit performance is comparatively lower than entry performance. This behavior is primarily attributed to the limited visual feedback of the needle once it is inside the tissue. Therefore, after insertion, the needle mainly follows the trajectory determined prior to entering the tissue. Despite this limitation, the agent achieves highly accurate exit performance (within 0.075 UU) in 68% of the episodes and remains within the acceptable region of 0.18 UU in 85% of the cases.

VI. LIMITATIONS AND FUTURE WORK

The RL environment is built mainly for validating the proposed simulation framework and demonstrate that autonomous agents can be trained using it. As mentioned in Section IV-A, we focus only on the needle insertion, driving, and extraction phases of suturing, while simplifying the action space and constraining the needle motion to a single plane. However, practical robotic suturing would require explicit modeling of needle re-grasping prior to extraction, needle hand-off, and knot-tying, which is not supported in the current setup. Incorporating physical robotic manipulators capable of performing these sub-tasks is therefore necessary for real-world applicability. Future work will extend the simulation to a dual-arm robotic setup, such as the da Vinci Research Kit (dVRK) with two Patient Side Manipulators (PSMs), enabling coordinated control for learning complete suturing procedures. Additionally, self-collision for the PBD suture should also be implemented.

Additionally, as an alternative to PPO or SAC, IL could also be considered. However, as discussed in Section II-B,

the performance of IL is bounded by the quality and coverage of the available demonstrations. Consequently, deviations from the demonstrated distribution can lead to significant performance drops, while RL presents greater adaptability. In our current framework, we introduced variability only in the initial needle position. In future work, however, we plan to further randomize parameters including tissue appearance and mechanical properties, needle geometry, and entry/exit marker locations, settings in which the advantages of RL are expected to become more prevalent. However, we acknowledge that extending the framework to full suturing would require algorithms capable of handling long-horizon tasks, including imitation learning–based approaches.

VII. CONCLUSIONS

This work introduced a novel simulation environment for surgical suturing with dynamic suture-tissue contact, tailored for parallel robot learning with GPU acceleration. Building on CRESSim-MPM, a PBD-based suture was integrated, and a new coupling strategy between PBD sutures and MPM soft tissue was developed to achieve plausible contact dynamics. The simulator was optimized for GPU execution, and a preliminary parallel RL environment was created to demonstrate its suitability for robotic learning. Using this framework, autonomous suturing agents were successfully trained using RL. Overall, this work demonstrates the feasibility and promise of achieving autonomous surgical suturing through simulation-only training, providing a scalable foundation for future advances in surgical robot learning.

REFERENCES

[1] C. Guébert, C. Duriez, S. Cotin, J. Allard, and L. Grisoni, "Suturing simulation based on complementarity constraints," in *Symposium on Computer Animation*, 2009.

[2] A. Maciel, G. Sankaranarayanan, T. Halic, V. S. Arikatla, Z. Lu, and S. De, "Surgical model-view-controller simulation software framework for local and collaborative applications," *International journal of computer assisted radiology and surgery*, vol. 6, no. 4, pp. 457–471, Jul. 2011.

[3] P. Yu, J. Pan, H. Qin, A. Hao, and H. Wang, "Real-time suturing simulation for virtual reality medical training," *Computer Animation and Virtual Worlds*, vol. 31, no. 4-5, p. e1940, 2020.

[4] A. Munawar, J. Y. Wu, G. S. Fischer, R. H. Taylor, and P. Kazanzides, "Open Simulation Environment for Learning and Practice of Robot-Assisted Surgical Suturing," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3843–3850, Apr. 2022.

[5] T. W. Kim, H. Zhou, J. A. Barragan, H. Ishida, P. Kazanzides, and A. Munawar, "Surgical Robotics Environment in NVIDIA Isaac Sim for Robot-Assisted Suturing," in *2025 International Symposium on Medical Robotics (ISMR)*, May 2025, pp. 192–198.

[6] G. Westergaard *et al.*, "Real-time haptic-based soft body suturing in virtual open surgery simulations," *Computers & Graphics*, vol. 134, p. 104507, Feb. 2026.

[7] J. Wu, H. Zhou, P. Kazanzides, A. Munawar, and A. Liu, "SurgicAI: A Hierarchical Platform for Fine-Grained Surgical Policy Learning and Benchmarking," *Advances in Neural Information Processing Systems*, vol. 37, pp. 63 771–63 789, 2024.

[8] Q. Yu *et al.*, "Orbit-Surgical: An Open-Simulation Framework for Learning Surgical Augmented Dexterity," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 15 509–15 516.

[9] Y. Ou and M. Tavakoli, "CRESSim–MPM: A Material Point Method Library for Surgical Soft Body Simulation with Cutting and Suturing," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11 247–11 254. [Online]. Available: https://ieeexplore.ieee.org/document/11247388/

[10] 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)*, Nov. 2019, pp. 1875–1882.

[11] N. Enayati *et al.*, "Robotic Assistance-as-Needed for Enhanced Visuomotor Learning in Surgical Robotics Training: An Experimental Study," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 6631–6636.

[12] 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)*, Aug. 2018, pp. 1056–1061.

[13] F. Richter, R. K. Orosco, and M. C. Yip, "Open-Sourced Reinforcement Learning Environments for Surgical Robotics," Jan. 2020.

[14] 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)*, Oct. 2020, pp. 3261–3266.

[15] 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)*, Apr. 2022, pp. 1–8.

[16] P. M. Scheikl *et al.*, "LapGym - An Open Source Framework for Reinforcement Learning in Robot-Assisted Laparoscopic Surgery," *Journal of Machine Learning Research*, vol. 24, no. 368, pp. 1–42, 2023.

[17] S. Schmidgall, A. Krieger, and J. Eshraghian, "Surgical Gym: A high-performance GPU-based platform for reinforcement learning with surgical robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 13 354–13 361.

[18] D. Dall'Alba, M. Naskręt, S. Kamińska, and P. Korzeniowski, "FF-SRL: High Performance GPU-Based Surgical Simulation For Robot Learning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2024, pp. 8378–8384.

[19] 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)*. Prague, Czech Republic: IEEE, Sep. 2021, pp. 1821–1828.

[20] J. Schulman, A. Gupta, S. Venkatesan, M. Tayson-Frederick, and P. Abbeel, "A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4111–4117.

[21] H. Zhou, Y. Jiang, S. Gao, S. Wang, P. Kazanzides, and G. S. Fischer, "Suturing tasks automation based on skills learned from demonstrations: A simulation study," in *2024 International Symposium on Medical Robotics (ISMR)*. IEEE, 2024, pp. 1–8.

[22] O. Özgüner, T. Shkurti, S. Lu, W. Newman, and M. C. Çavuşoğlu, "Visually guided needle driving and pull for autonomous suturing," in *2021 IEEE 17th international conference on automation science and engineering (CASE)*. IEEE, 2021, pp. 242–248.

[23] J. Huang *et al.*, "Visuomotor policy learning for task automation of surgical robot," *IEEE Transactions on Medical Robotics and Bionics*, 2024.

[24] Y. Barnoy *et al.*, "Applying neural networks to robots in complex environments with a focus on robotic surgery," Ph.D. dissertation, Johns Hopkins University, 2021.

[25] J. Haworth *et al.*, "Suturebot: A precision framework & benchmark for autonomous end-to-end suturing," *arXiv preprint arXiv:2510.20965*, 2025.

[26] Y. Ou and M. Tavakoli, "Learning autonomous surgical irrigation and suction with the da vinci research kit using reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 16 753–16 767, 2025.