An Open Simulation Environment for Learning and Practice of Robot-Assisted Surgical Suturing

Adnan Munawar[1], Jie Ying Wu[2], Gregory S. Fischer[3], Russell H. Taylor[1], Peter Kazanzides[1]

Abstract—**Automation has the potential to improve the standard of care but is difficult to realize due to perceptual challenges, especially in soft-tissue surgery. Machine learning can provide solutions, but typically requires large amounts of training data, which is time-consuming to collect. Even with shared platforms, hardware differences can prevent effective sharing of data between institutions. This paper proposes a standardized simulation platform for training and testing algorithms to control surgical robotic systems, which is built upon an open-source simulator, the Asynchronous Multi-Body Framework (AMBF), to enable quick prototyping of different scenes. An illustrative example of a suturing task on a phantom is presented and has formed the basis of a challenge, released to the community. The top-level contribution is the open-source release of a dynamic simulation environment that enables realistic suturing on a phantom, but supporting contributions include its extendable architectural design and a series of algorithmic optimizations to achieve real-time control and collision detection, realistic behavior of the needle and suture, and generation of multi-modal ground-truth data, including labeled depth data. These capabilities enable simulation-based surgical training and support research in machine learning for surgical scene perception and autonomous action.**

I. INTRODUCTION

Automation in certain surgical fields, such as orthopedics, has improved precision to a superhuman extent [1]. Yet its adaptation to soft-tissue surgeries remains an open question due to the increased difficulty in understanding the environment. In a recent review of research done on one particular surgical robot research system, the da Vinci Research Kit (dVRK) [2], 68 of the 253 papers published between 2012-2021 focused on automation [3]. Related fields such as surgical gesture and skill recognition, and imaging and vision had 33 and 32 publications, respectively.

Despite the interest in these topics, it has been difficult to identify a common experiment setup to ensure that results are reproducible. There are currently over 40 sites in 11 countries using the dVRK. It is difficult to ensure that physical setups are consistent across these sites. This work proposes a modular simulation platform with dynamically loadable phantoms that can be designed and shared as a way to overcome the lack of standardized physical setups. Simulation platforms can also enable researchers without a dVRK to perform surgical robotics research. However, creating such a platform is challenging. Automation, such as for cutting, pick and place, irrigation, or endoscopic motion, requires precise actuation of the robot in the simulator and realistic response from the simulated environment. Vision-based methods to identify tool-tissue segmentation or phase of surgery require high-fidelity visual feedback. We present such a simulation environment in this manuscript that is built on top of the Asynchronous Multi-Body Framework (AMBF) [4].

We previously developed AMBF for simulating the dynamics of kinematically redundant robots and mechanisms, a construct employed frequently in surgical robots. As part of developing the surgical environment in AMBF, several new features were required, which were co-developed. These features can be readily used for many different applications. We briefly discuss some of these features alongside their use for the simulation environment in Sec. III.

The development of a surgical simulation environment, as presented in this manuscript, requires a significant amount of engineering work and system integration. In light of this, we aim to present the challenges faced, lessons learned, and our approach in various sections throughout the paper.

The primary contribution of this work is an open-source simulation environment for surgical suturing, integrated with shared research platforms such as the dVRK. This includes: (1) architectural contributions in the integration, using ROS, of software packages for defining object models (Blender AMBF Addon), dynamic simulation (AMBF Simulator) and interfaces to multiple devices (AMBF Client and Python scripts), and (2) algorithmic contributions in real-time collision checking, realistic needle grasping, suture simulation, visual rendering, generation of labeled depth images and a closed-form inverse kinematic solution for the dVRK PSM.

II. RELATED WORKS

There are existing technologies that implement simulated surgical procedures such as the da Vinci SimNow[1] and Mimic[2] simulation. These and other similar systems are all closed-source and are pre-configured to support a specific set of user-input devices. Due to their proprietary nature, and common

Manuscript received: September, 09, 2021; Revised December, 13, 2021; Accepted January, 06, 2022.

This paper was recommended for publication by Editor Jessica Burgner-Kahrs upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by NSF OISE-1927354 and OISE-1927275 and by an agreement between Johns Hopkins University and the Multi-Scale Medical Robotics Centre, Ltd.

Adnan Munawar, Russell H. Taylor, and Peter Kazanzides are with the Department of Computer Science at the Johns Hopkins University, Baltimore, MD, 21218 amunawar@jhu.edu

Jie Ying Wu is with the Department of Computer Science at Vanderbilt University, Nashville, TN, 37235

Gregory S. Fischer is with the Department of Robotics Engineering at Worcester Polytechnic Institute, Worcester, MA, 01609

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

[1]https://www.davincisurgerycommunity.com/Systems_I_A/Skills_Simulator

[2]https://mimicsimulation.com/

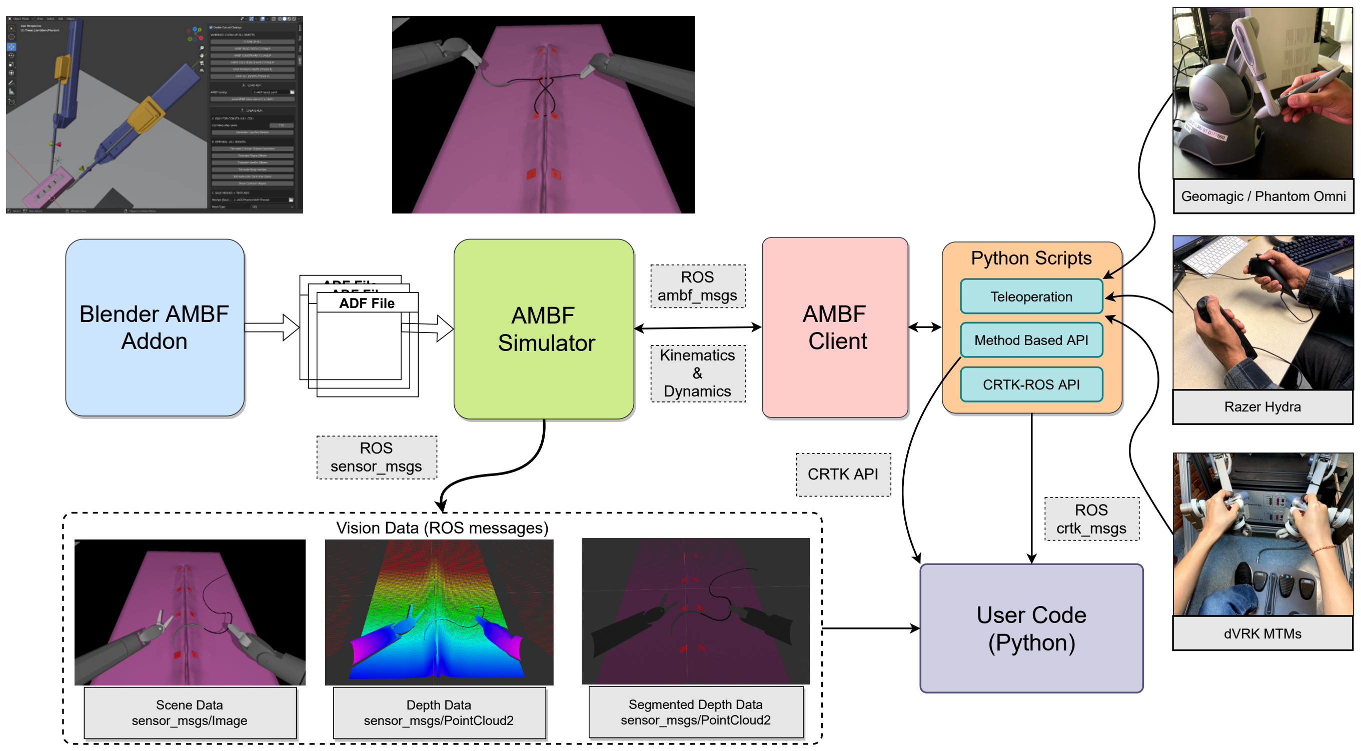


Fig. 1. Overview of the surgical simulation scene and its components. The Blender-AMBF addon generates ADF files which are loaded into the AMBF Simulator. The AMBF Simulator generates and streams kinematics, dynamics, and scene data in standardized formats (ROS msgs). The Python Client and the provided Python scripts convert the kinematics and dynamics data in CRTK compatible format. The Python scripts also implement teleoperation via input devices.

use in clinical settings, there is no evidence of these systems allowing the streaming of internal data for other applications such as machine learning.

Robot Operating System (ROS) [5] has revolutionized academic research in robotics, by providing a standard set of payloads combined with an easy-to-use and distributed middleware. Royalty-free and open-source robot simulators such as Gazebo [6] and V-REP [7] have also found great traction in robotics, in part due to their support for ROS as a middleware. These robot simulators rely on a dynamically loadable data format to define scene data (lights, cameras, robots, etc.). The Universal Robot Description Format (URDF) [8] and Simulation Description Format[3] (SDF) are such examples which are written using the XML language. Describing scene data in such dynamically loadable formats has many advantages over an application where these are pre-programmed. For example, SDF and URDF allow a model to be distributable, modular and supported over many different versions of the underlying simulator. At the same time, advanced features may quickly become cumbersome and complex in terms of description. Gazebo and V-REP support a rich set of plugins to simply these descriptions. However, plugin support across different versions of the simulator remains questionable as the API for open-source simulators is constantly evolving. Furthermore, relying heavily on plugins may defeat the purpose of the description format, and ultimately the use-case of these simulators themselves. Consequently, one may opt for developing the simulation environment in video game engines such as Unity[4] and Unreal engine[5], however, the open-source aspect, modularity, and support with ROS remain questionable.

These simulation environments can be used to prototype robot interactions. Fontanelli et al. proposed a specific surgical robotics environment implemented in V-REP [9]. Enayati et al. [10] demonstrated that training on even simple, simulated tasks could speed up surgeon skill acquisition. Other simulators, such as SOFA [11], focus on the patient side of the surgical scene. Nguyen et al. [12] note in their review that most simulators that focus on soft-tissue do not support manipulators so researchers must choose between high-fidelity robot movements or soft-tissue response.

To simulate a realistic surgical environment, there must be a simulator that combines the robot model and the patient model, while allowing for different combinations of surgeon and patient manipulators. Our previous work [13] detailed how we could construct a set of reference frames that allows for flexible pairings between different numbers of user input devices, camera motion, and patient manipulators.

OpenAI Gym [14] has demonstrated that an open-source environment can be a valuable tool for machine learning. While OpenAI Gym supports a broad range of applications, medical robotic environments often require more specific setups such as

[3]http://sdformat.org/spec

[4]https://unity.com/

[5]https://www.unrealengine.com/

the ability to manipulate fluids (e.g., blood [15]) or soft-bodies (e.g., tissue). Richter et al. proposed a simulator focused on driving surgical robotics, the dVRL [16]. This combines the physics environment for the da Vinci Surgical System, proposed by Fontanelli et al. [9], with the learning interface used in OpenAI Gym. This environment focuses on accurate robot movements but has limited support for tissue manipulation. Similarly, Xu et al. extends this environment in SurRoL, while still focusing on rigid phantom tasks such as peg transfer [17]. Tagliabue et al. [18] proposed an environment for soft tissue but is limited to sheet-like objects rather than fully 3D models. The proposed simulator's value is in bringing together accurate robot kinematics with tissue deformation while not limiting the simulation scenes to pre-optimized setups.

III. METHODS

The goal of this work is to provide a modular simulated suturing environment with realistic interactions, generation and streaming of structured data, and a standardized control interface. This is achieved by concurrent improvements to AMBF and its various components as described in this section. Sec. III-A gives a brief overview of the AMBF architecture. Sec. III-B describes the collision optimization of simulated bodies for realistic interactions. Sec. III-C describes the simulation of suture. Sec. III-D discusses the robot controls, how to teleoperate, and methods to achieve realistic grasping. Lastly, Sec. III-E details how the visualization works and how to stream out RGB and depth information.

A. System Architecture

The overall system architecture is shown in Fig. 1. The entry point of an AMBF simulation is via a launch file, which contains meta-data in the form of paths to different AMBF description format (ADF) files. This is shown in Fig. 2. There are three different types of ADF files that include world, input-devices, and models. A launch file must define a single world file, a single input-devices file, and as many model files as required.

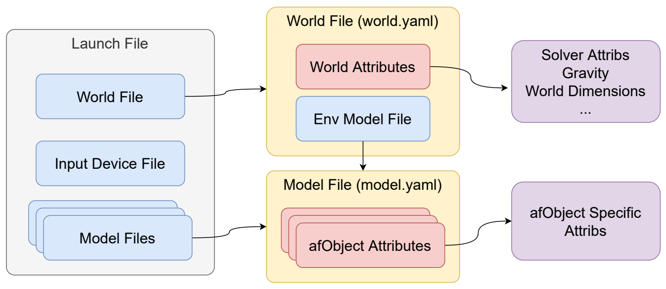


Fig. 2. Entry to AMBF simulation via a launch file

The world ADF file describes global attributes such as solver iterations, gravity, world's bounds, etc. It may also define lights and cameras, however, these could also be defined in model ADF files. A model ADF file describes afObjects, where an afObject refers to a simulation entity such as a rigid body or a joint. Different types of afObjects are shown in Fig. 3. Each afObject shares some common attributes such as naming, parenting, and communication. The surgical simulation scene consists of rigid bodies, ghost objects, joints, sensors, actuators, cameras, and lights. The provided launch file includes the paths to all these ADF files which can then be loaded in AMBF.

1) The Blender-AMBF Addon: The ADF files can be created by hand, however, this is not an easy task for complex scenes. We developed the Blender-AMBF addon that allows the creation of ADF files using a graphical interface. To create the surgical scene, several new features were added to the Blender-AMBF addon including: 1) advanced collision specification for a rigid body, 2) setting textures and normals maps for meshes, 3) setting 6 DOF constraints (joints), and 4) fine-tuning inertial attributes of a rigid body. To the best of our knowledge, the Blender-AMBF addon is the most feature-rich tool for creating robots and mechanisms for physics simulations, although it currently only supports the ADF specification.

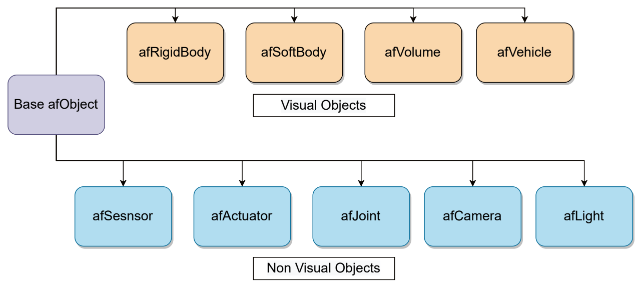


Fig. 3. Different Types of afObjects

2) Collaborative Robotics Toolkit Specification: Since our goal is to make an accessible simulation platform, we chose to use a standardized specification for interfacing with our simulator. We use the Collaborative Robotics Toolkit (CRTK) [19], which is a specification for describing and standardizing the task and joint space motion-related data for teleoperated and collaborative robots. CRTK is currently implemented in ROS but can be extended to any middleware. The software stack for the dVRK uses the CRTK API both for its internal method names pertaining to robot control and its ROS interfaces. Several other robots and devices that utilize the core software (*cisst* and SAW [2]) that runs the dVRK, are also CRTK compatible.

Keeping in line with the dVRK software, our simulation environment provides CRTK compatible interfaces (both in terms of ROS topics and method names) for controlling the simulated PSMs and ECM. This allows any algorithm developed in simulation to port easily to the real dVRK.

B. Realistic Interaction Between Environment Objects

Simulation environments that involve the fine manipulation of rigid bodies in and around other complex rigid bodies and narrow pathways, such as the one presented in this manuscript, require optimizations to the collision composition of each collidable object. This is because collision detection and contact resolution is commonly the most computationally expensive step. In general, convex shapes are always preferred over concave shapes as commonly used collision algorithms such as Gilbert-Jhonson-Kreethi (GJK) [20] offer almost linear time computation. Collision primitives that include spheres, capsules, cylinders, cones, and boxes are even more efficient than general convex hulls. Most of the collidable shapes in the surgical scene cannot be approximated with a single convex hull or primitive and instead only by a group (compound) of shapes.

Due to a large number of possibilities, optimizing collisions is a time-consuming task. The Blender-AMBF addon, the ADF specification, and AMBF have all been updated alongside the surgical scene to support such possibilities. Fig. 4 and Fig. 5 show the collision compositions of several critical shapes in the scene. Figure 5 also shows the simplicity of defining the convex-hulls and primitive based compound collision shapes using the Blender-AMBF addon.

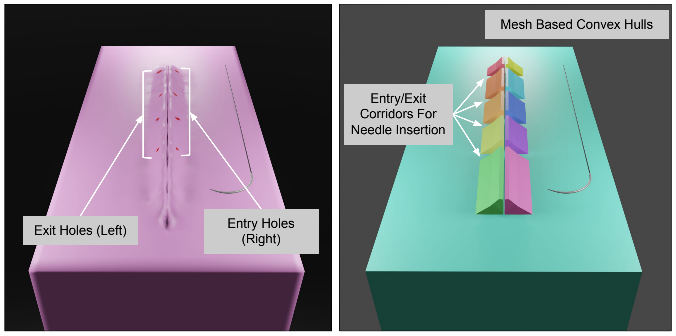


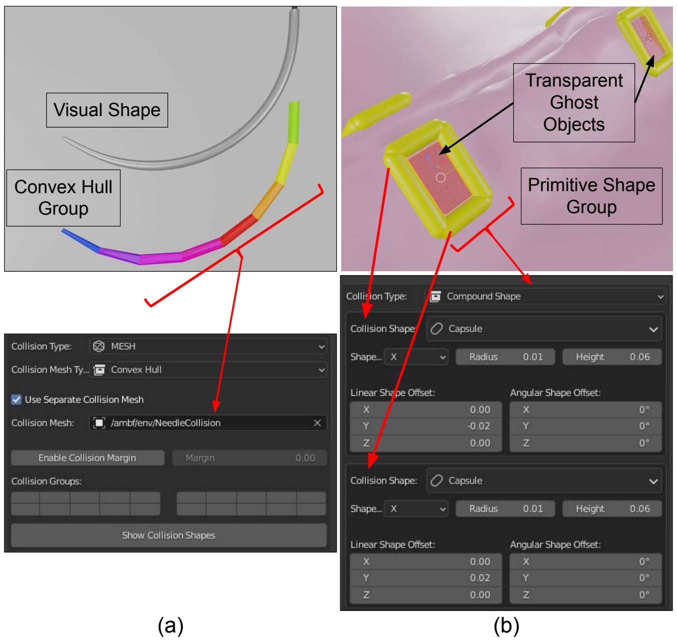
Fig. 4. The visual mesh for the phantom (left) with demarcated points for needle insertion shown in red. Optimized collision mesh for the phantom mesh shown on the right. Each color corresponds to a different convex hull, all assembled into a single mesh with different materials.

C. Simulation of Surgical Suture

Simulation of ropes and rods has been investigated in several works including [21], [22] and [23]. For rope simulation, the rope length is approximated by a system of particles interconnected via distance constraints (Jakobsen constraints [24]). After the formulation of the equations of motions and the constraint equations, the system state is updated using integration (e.g., Verlet integration [25]) to compute the particle's position. Each particle is defined as a point mass with no associated orientation. For the simulation of elastic rods [26] bending, twisting and elongation is modeled by additional constraint equations and occasionally with implicit integration techniques.

Position Based Dynamics (PBD) [27] is a class of solvers that offers stable and visually plausible soft body simulations, including the simulation of ropes and rods [22]. In our simulation framework, we use the Bullet Physics [28] library and its default velocity based solver (Sequential Impulse (SI) [29] solver) for simulating robots with revolute and prismatic joints.

The SI solver computes corrective impulses which indirectly alter the position of the bodies at subsequent states. Unlike SI, the PBD solver uses constraint projection to directly rectify the position of the particles (bodies). As a result, PBD offers better stability and convergence, albeit at the expense of solution



Collision Type: MESH
Collision Mesh Ty... Convex Hull
☑ Use Separate Collision Mesh
Collision Mesh: /ambf/env/NeedleCollision
Enable Collision Margin | Margin 0.00
Collision Groups:
Show Collision Shapes

Collision Type: Compound Shape
Collision Shape: Capsule
Shape... X | Radius 0.01 | Height 0.06
Linear Shape Offset: X 0.00, Y -0.02, Z 0.00
Angular Shape Offset: X 0°, Y 0°, Z 0°
Collision Shape: Capsule
Shape... X | Radius 0.01 | Height 0.06
Linear Shape Offset: X 0.00, Y 0.02, Z 0.00
Angular Shape Offset: X 0°, Y 0°, Z 0°

Fig. 5. The collision composition of the needle (top left). Each colored segment denotes a different convex hull. All the convex hulls together can be combined into a single mesh with a different material for each hull and set as the collision mesh using the Blender-AMBF addon (lower left). (Right) A boundary mesh (yellow cylinders) to constrain the orthogonal motion of the needle once it is inserted into the phantom's holes. The collision of the boundary mesh is composed of four capsule primitives (yellow) which can be set via the Blender-AMBF addon (lower right). The red transparent box denotes a ghost object to detect the overlap of simulated objects and perform necessary computation.

accuracy [27]. On the other hand, SI is widely used in fast physics simulations for video games and applications involving force feedback and haptics. While the SI solver is not ideal for the simulation of the surgical thread, it is already integrated with AMBF. The general equation for computing the impulses in SI is formulated as:

$$\vec{I_c} = -J^T M_e^{-1} \left(J \left(\vec{V_i} + M^{-1} \vec{F_e} \Delta t \right) + \vec{b} \right) \tag{1}$$

Where $\vec{I_c} \in \mathbb{R}^{12}$ is the computed impulse for a pair of constrained bodies, $\vec{F_e}$ is the external force acting on the two bodies, J is the constraint Jacobian, $\vec{V_i}$ is the stacked vector of the velocities of the two constrained bodies, M is the combined mass matrix, $M_e = JM^{-1}J^T$ is the effective mass matrix, and $\vec{b}$ is the bias velocity. The SI solver uses the symplectic Euler integration to update the state of the system after solving the constraints.

$$\vec{V}_{i+1} = \vec{V_i} + M^{-1} \left(F_e \Delta t + \vec{I_c} \right) \tag{2}$$

$$\vec{X}_{i+1} = \vec{X_i} + \vec{V}_{i+1} \Delta t \tag{3}$$

To simulate the surgical thread, we use a 6 Degree of Freedom (DOF) spring constraint with three linear and three angular DOF between two rigid bodies. We set the nominal length (equilibrium point), the stiffness, and damping of the 3 linear DOF to 0 to emulate a universal joint.

Since our simulation environment involves both objects with low masses and inertias (needle and thread elements), as well as objects with high masses (robot links), we end up with constraints that handle large mass ratios. The SI solver's solution convergence and stability are negatively affected by large mass ratios. Specifically for the surgical thread, which is manipulated by PSMs with much larger masses, the individual masses of the thread elements, their inertias, and correspondingly the angular spring parameters (stiffness and damping) require tuning by trial and error. In our testing, the thread was divided into 40 elements, each with a mass of 0.03 kg and the spring parameters were set as: angular limits = $[-\frac{\pi}{6}, \quad \frac{\pi}{6}]\ rad$, stiffness = 0.02 Nm/rad and damping 0.01 Ns/rad along the three axes.

Finally, as shown in Fig. 5, we use Ghost Objects to detect if the needle or thread overlaps with the entry or exit hole in the phantom. We apply viscous friction $F_v = -K\vec{V}_{relative}$ to the needle (or the suture) to emulate as if it were in the tissue. This stabilizes the needle and prevents it from moving on its own if a suture is not attached. With a suture attached, the needle tends to drift as a result of the forces imparted by the suture. An additional constraint may be required to stabilize it in such cases.

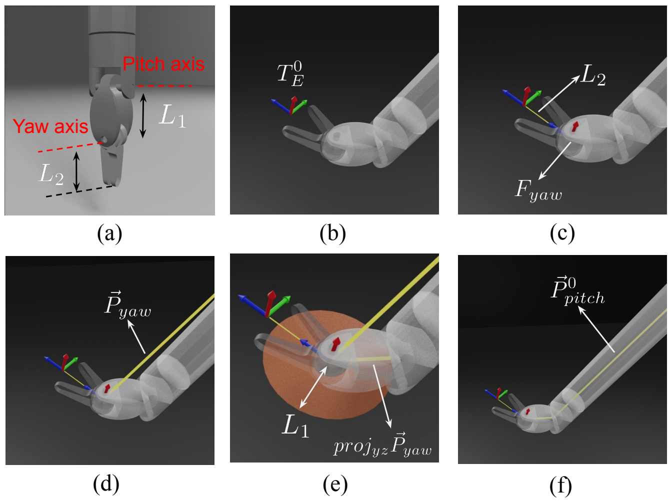


Fig. 6. Geometrical breakdown of PSM's IK. (a) The two parameters (L_1 and L_2) that differ between different tool types. (b) (c) Traversing back from end-effector pose to compute frame F_{yaw}. (d) Computing $\vec{P}_{yaw}$ as error between F_{yaw} and PSM origin F_0. (e) Projecting $\vec{P}_{yaw}$ onto F_{yz} to get $proj_{yz}\vec{P}_{yaw}$. (f) Computing $\vec{P}^0_{pitch}$

D. Controlling the Robot

1) Kinematics of Simulated Robots: The simulated robot end-effectors are modeled after the da Vinci PSMs. The dVRK models the inverse kinematics of the PSMs using the damped-least squares method [30] which is computationally more expensive as compared to analytical methods, however, analytical solutions do not exist for many robot manipulators.

The PSMs are 6 Degree of Freedom (DOF) manipulators (excluding the gripper) with a remote center of motion (RCM), and thus an analytical solution exists. The computational cost saving achieved by employing an analytical solution is valuable for us since our teleoperation code is Python based.

We provide a brief geometrical description of our analytical inverse kinematics solution in Fig. 6. The main challenge is to identify a set of planes and orthogonal vectors that relate the end-effector pose T^0_E to a point along the line passing between the base frame F_0 and the RCM point. The frames for kinematic computation are assigned identical to the dVRK software stack [2].

$$T^E_{yaw} = \left[I_{3\times 3}; \vec{P}_{L2}\right] \quad \in \mathbb{R}^{4\times 4} \tag{4}$$

Where $\vec{P}_{L2} = (0, 0, L2)^T$

$$T^0_{yaw} = T^0_E * T^E_{yaw} \quad ; \vec{P}_{yaw} = (R^0_{yaw})^{-1} * \vec{P}^0_{yaw} \tag{5}$$

$$\vec{P}^{yaw}_{pitch} = proj_{yz}\vec{P}_{yaw} = -\left|\left|\vec{P}_{yaw} \odot (0,1,1)\right|\right| * L1 \tag{6}$$

The operator $\odot$ represents element-wise multiplication.

$$T^{yaw}_{pitch} = \left[I_{3\times 3}; \vec{P}^{yaw}_{pitch}\right] \quad ; T^0_{pitch} = T^0_E * T^E_{yaw} * T^{yaw}_{pitch} \tag{7}$$

Now the inverse kinematic solution q_i where $i \in [1, 2...6]$ is calculated as:

$$q_1 = \text{atan2}\left(Px^0_{pitch}, -Pz^0_{pitch}\right) \tag{8}$$

Px , Py and Pz denote the x, y and z components of a vector $\vec{P}$, $\vec{P}^0_{pitch}$ in the above equation.

$$q_2 = -\text{atan2}\left(Py^0_{pitch}, \left|\left|\vec{P}^0_{pitch} \odot [1,0,1]\right|\right|\right) \tag{9}$$

$$q_3 = \left|\left|\vec{P}^0_{pitch}\right|\right| + L_{tool2rcm} \tag{10}$$

The length $L_{tool2rcm}$ is the offset between the tool shaft's end (intersection of joint 4 and 5) and the remote center point at the home position (i.e., the prismatic insertion joint q_3 fully retracted). The calculation of the last three joint angles is tricky and requires identifying the correct components of the assigned rotation matrices to the last three links. Moreover, to compute joint angles within the assigned joint limits, the function GetAngle $\left(\vec{a}, \vec{b}, \vec{c}\right)$ in Eq. 12, 13, and 14 returns the angle between $-\pi$ to π rather than the usual 0 to π. The returned value is the angle between $\vec{a}$ and $\vec{b}$, along $\vec{c}$.

$$\vec{P'} = \vec{Rx}^{\,0}_{E} \times \left(\vec{P}^0_{yaw} - \vec{P}^0_{pitch}\right) \tag{11}$$

$$q_4 = \text{GetAngle}\left(\vec{P'}, \vec{Ry}^{\,0}_{3}, -\vec{Rz}^{\,0}_{3}\right) \tag{12}$$

$\vec{Rx}$, $\vec{Ry}$ and $\vec{Rz}$ denote the 1st, 2nd and 3rd columns of a rotation matrix, respectively. R^0_3 is computed from the forward kinematics formulation and using the computed q_1, q_2 and q_3.

$$q_5 = \text{GetAngle}\left(\vec{P}^0_{yaw} - \vec{P}^0_{pitch}, \vec{Rz}^{\,0}_{4}, -\vec{Ry}^{\,0}_{4}\right) \tag{13}$$

$$q_6 = \text{GetAngle}\left(\vec{Rz}^{\,0}_{E}, \vec{Rx}^{\,0}_{5}, -\vec{Ry}^{\,0}_{5}\right) \tag{14}$$

This analytical solution can be used to compute the inverse kinematics of any PSM tool with the correct values of

$L_{tool2rcm}$, $L1$, and $L2$. A joint limit check is performed in practice after obtaining the inverse kinematics solution.

2) Teleoperation: Controlling the simulated tools using physical input devices is useful for both training surgeons and generating expert data for machine learning applications. Towards this end, we have paid special attention in developing a realistic teleoperation experience that is similar to the da Vinci system. The simulation package consists of scripts that allow the pairing of a variety of input devices including dVRK MTMs, Geomagic Touch (3D Systems, Rock Hill, South Carolina, USA) devices, and Razer Hydras (Razer, Irvine, California, USA) to the simulated robot end-effectors. The teleoperation scripts also allow a single input device to dynamically switch between multiple PSMs, or connect different input devices to different PSMs. The dynamic switching is useful not only for debugging purposes but also for the single-handed performance of otherwise bi-manual or multi-manual tasks.

3) Grasping and Manipulation: Friction-based grasping in real-time dynamic simulations using the SI solver is a challenging problem due to its iterative nature and the approximation of the friction model. Since grasping involves the pressing of an object between two or more fingers, in our experience, finding the optimal value of the friction coefficients to prevent slippage is not trivial. Modeling friction-based grasping is useful for tasks that involve within hand manipulation, in which the object is not firmly grasped. Approaches such as [31] [32] provide implementation techniques for this kind of grasping. In our specific application, the needle and the thread are often firmly grasped with little to no slippage. This allows us to "cheat" and implement a rigid grasping technique. In this technique, we place a ray-tracing sensor and a constraint actuator (general-purpose features available in ADF and AMBF) between the two fingers of the tool as shown in Fig. 7. Using the combination of the sensor and the actuator, and using the name of the sensed object to trigger the actuator, the grasping technique works reasonably well for our specific application.

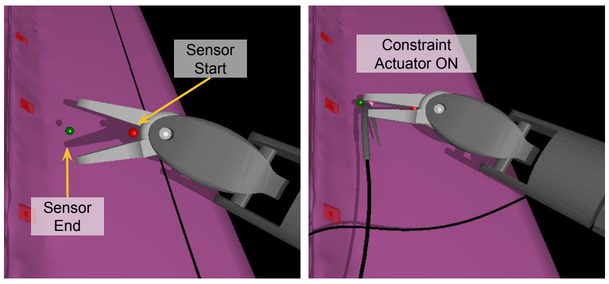


Fig. 7. Constraint based grasping using a combination of a ray-tracing sensor and constraint actuator parented to an object. The sensor detects an object and the actuator can be triggered to affix it to the parent object.

E. Rendering Scene Data

It is desirable to have a high degree of visual realism in the rendering quality of the surgical scene. The basic and simplest component for achieving better rendering output is to use textures and normal maps, though specular maps to control per pixel shininess can impart more realism. AMBF uses OpenGL's fixed rendering pipeline for outputting the scene to the display unit. The fixed rendering pipeline implements Gourard shading [33], which is a fairly simple model. Other models such as the Blinn-Phong [34], or more advanced models such as the Physically Based Rendering (PBR) [35], in combination with Image Based Lighting (IBL) [36], require access to custom vertex and fragment shaders. In this regard, AMBF has been updated to allow the specification of world specific, model specific, or object specific shaders using the ADF files. This opens the possibility for improving the scene rendering quality by dynamically loading different shading models.

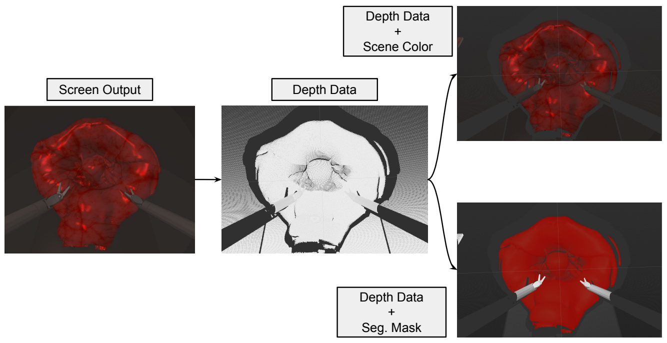


Fig. 8. Generating labeled depth maps by combining data from different rendering passes

For machine learning applications, access to scene data in the form of a video stream or images is required. This is possible by enabling a "publish image" flag for the camera in its ADF file. For depth perception, it is possible to launch two cameras with a parametric offset to emulate a stereo endoscope like that used on the clinical da Vinci robot, and then use the two scene images for stereo vision. However, a benefit of using a simulator is that ground truth depth data can be generated.

We have incorporated the support for generating and streaming the depth data for any camera in AMBF. Similar to enabling the streaming of scene data, setting a flag "publish depth" enables the generation of the depth data and its publishing. The depth data is defined with respect to the camera frame. Additionally, it is possible to generate segmented depth data by using a custom rendering pipeline of AMBF cameras, as shown in Fig. 8.

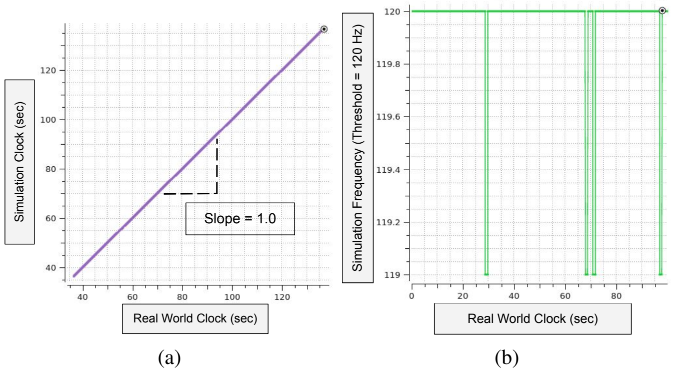


Fig. 9. (a) Comparison of simulation time vs real world time. (b) Frequency of the simulation during a suture task.

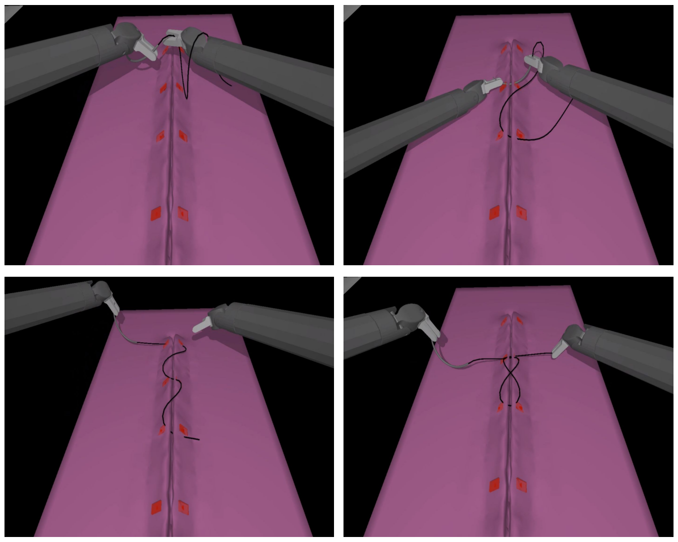


Fig. 10. Example Suture Task.

IV. RESULTS

A real-time simulation is required for user training such that the progression of the simulation time matches that of the real world. Most of the optimizations, discussed in Sec. III, were targeted towards achieving this goal. Fig. 9 shows the comparison between the simulation time and the real world time, and the simulation frequency for a suturing task. The test was performed on a system comprising of an AMD Ryzen 3600 CPU, 32 GB of RAM, and an Nvidia 1080 GPU running Ubuntu 20.04. Fig. 10 shows different snapshots of the surgical suture task performed via the input devices shown in Fig. 1 (see accompanying video). A more general evaluation of AMBF is presented in [37].

We obtained qualitative feedback from an OBGYN surgeon and a fellow who interacted with the simulation using the dVRK MTMs. Their critical feedback is summarized as: 1) the suture behavior, the overall teleoperation experience, and the scene, in general, look convincing, 2) the needle insertion dynamics and feedback into the tissue should be improved, 3) two-handed needle grasp felt more rigid compared to a real needle interaction via the da Vinci Xi instruments, and 4) the simulated PSMs move at a different speed compared to the da Vinci Xi PSMs.

V. DISCUSSION AND FUTURE WORK

In this work, we present a real-time simulation that models a robot-assisted suturing task. The simulation mimics the controls of both the camera motion and the PSMs of a real da Vinci Surgical System to ease the transfer of algorithms to physical setups. The environment is designed to be modular and easily modifiable to model different scenes with different sets of input devices and robot arms. The scene files and the source code are available at https://github.com/collaborative-robotics/surgical_robotics_challenge.

Despite its many features, it has several obvious limitations including the rigidity of the phantom, fixed needle penetration holes, and the firm needle grasp. We plan on addressing these limitations and making improvements to several different components as discussed below.

The foremost improvement is to replace the rigid body phantom with a softbody. In this regard, AMBF already provides the basics of softbody simulation [38]. However, tuning the elasticity of the softbody so that it behaves in a visually plausible way requires tuning of the underlying softbody parameters mostly by trial and error. Furthermore, softbody simulations are significantly more computationally expensive and thus require a considerable amount of optimizations. These optimizations include the simplification of the collision geometry as well as the visual geometry of the phantom mesh. We are considering integrating our previous works in using deep learning to learn more accurate softbody behavior [39] and to speed up simulation [40].

Currently, the needle is only able to penetrate the phantom at highlighted areas, which was a reasonable result for the intended challenge, but will be improved upon to increase the flexibility of the system. With the addition of a softbody phantom, we aim to allow the needle to penetrate anywhere on the tissue. Softbody simulations usually model the surface of a mesh and thus realistic needle penetration and dynamics have to be modeled explicitly. The ghost objects and velocity constraints as discussed in Sec. III-B may be utilized for this purpose. In terms of the dynamics of the needle insertion, we plan on dynamically creating 6 DOF spring constraints, with a low value of stiffness and damping along the needle curvature, and high values for all other axes. Similarly, for the needle grasping methodology as discussed in Sec. III-D3, future work may involve allowing for needle slippage.

Another possible improvement is the visualization of the suture thread. In our current implementation, the rigid body finite elements that make up the thread are directly visualized, which results in an unrealistic-looking thread with hard edges. This becomes more apparent as the camera moves closer to the thread. A better approach is to use a combination of curve fitting and extruded rendering (tubing). For curve fitting, one may use Bezier curves or Chaikin's corner-cutting algorithm [41] and the center of finite elements of the thread as control points. Once a reasonably smooth curve is obtained, a 2-dimensional shape can be extruded along the curve to generate a visually smooth suture thread.

Our current work was only qualitatively evaluated by an OBGYN surgeon and a fellow. For the future, we plan on recruiting several surgeons and residents to more formally evaluate the simulation, both quantitatively and qualitatively.

Finally, to motivate the use of our environment, we propose a challenge based on automating the suturing task. The task involves several sub-challenges including needle/tool detection and tracking, pick and place, needle-driving through the tissue (which results in interaction dynamics and occlusion), coordination between robot end-effectors (for needle handover), and path planning (for manipulating the thread and knot-tying). As such, it requires algorithms that span vision, phase recognition, and automation.

VI. ACKNOWLEDGEMENT

We thank Grace Chen M.D. and Anna Pancheshnikov M.D. for trying out the simulation and providing valuable feedback. We also thank Haoying Zhou for contributing to the code and Anton Deguet for providing support with the dVRK.

REFERENCES

[1] P. Kazanzides, B. D. Mittelstadt, B. L. Musits, W. L. Bargar, J. F. Zuhars, B. Williamson, P. W. Cain, and E. J. Carbone, "An integrated system for cementless hip replacement," *IEEE Engineering in Medicine and Biology Magazine, Special Issue on Robots in Surgery*, vol. 14, no. 3, May/June 1995.

[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 *IEEE Intl. Conf. on Robotics and Auto. (ICRA)*, Hong Kong, China, Jun 2014, pp. 6434–6439.

[3] C. D'Ettorre, A. Mariani, A. Stilli, F. Rodriguez y Baena, P. Valdastri, A. Deguet, P. Kazanzides, R. H. Taylor, G. S. Fischer, S. P. DiMaio, A. Menciassi, and D. Stoyanov, "Accelerating surgical robotics research: A review of 10 years with the da Vinci Research Kit," *IEEE Robotics and Automation Magazine*, vol. 28, no. 4, pp. 56–78, Dec. 2021.

[4] 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 *IEEE/RSJ Intl. Conf. on Intell. Robots and Systems (IROS)*, 2019, pp. 1875–1882.

[5] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[6] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2149–2154.

[7] E. Rohmer, S. P. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 1321–1326.

[8] L. Kunze, T. Roehm, and M. Beetz, "Towards semantic robot description languages," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 5589–5595.

[9] 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 *IEEE Intl. Conf. on Biomedical Robotics and Biomechatronics (Biorob)*, 2018, pp. 1056–1061.

[10] N. Enayati, A. M. Okamura, A. Mariani, E. Pellegrini, M. M. Coad, G. Ferrigno, and E. De Momi, "Robotic assistance-as-needed for enhanced visuomotor learning in surgical robotics training: An experimental study," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2018, pp. 6631–6636.

[11] H. Talbot, N. Haouchine, I. Peterlik, J. Dequidt, C. Duriez, H. Delingette, and S. Cotin, "Surgery training, planning and guidance using the SOFA framework," in *Eurographics*, 2015.

[12] T.-N. Nguyen, M.-C. Ho Ba Tho, and T.-T. Dao, "A systematic review of real-time medical simulations with soft-tissue deformation: Computational approaches, interaction devices, system architectures, and clinical validations," *Applied Bionics and Biomechanics*, vol. 2020, 2020.

[13] A. Munawar, J. Y. Wu, R. H. Taylor, P. Kazanzides, and G. S. Fischer, "A framework for customizable multi-user teleoperated control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3256–3263, 2021.

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

[15] F. Richter, S. Shen, F. Liu, J. Huang, E. K. Funk, R. K. Orosco, and M. C. Yip, "Autonomous robotic suction to clear the surgical field for hemostasis using image-based blood flow detection," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1383–1390, 2021.

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

[17] J. Xu, B. Li, B. Lu, Y.-H. Liu, Q. Dou, and P. A. Heng, "SurRoL: An open-source RL centered and dVRK compatible platform for surgical robot learning," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021, pp. 1821–1828.

[18] 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 *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020, pp. 3261–3266.

[19] Y.-H. Su, A. Munawar, A. Deguet, A. Lewis, K. Lindgren, Y. Li, R. H. Taylor, G. S. Fischer, B. Hannaford, and P. Kazanzides, "Collaborative Robotics Toolkit (CRTK): Open software framework for surgical robotics research," in *IEEE Intl. Conf. on Robotic Computing (IRC)*, 2020, pp. 48–55.

[20] C. J. Ong and E. G. Gilbert, "The Gilbert-Johnson-Keerthi distance algorithm: A fast version for incremental motions," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 2, 1997, pp. 1183–1189.

[21] D. K. Pai, "Strands: Interactive simulation of thin solids using Cosserat models," in *Computer Graphics Forum*, vol. 21, no. 3. Wiley Online Library, 2002, pp. 347–352.

[22] N. Umetani, R. Schmidt, and J. Stam, "Position-based elastic rods," in *ACM SIGGRAPH 2014 Talks*, 2014, pp. 1–1.

[23] T. Kugelstadt and E. Schömer, "Position and orientation based Cosserat rods," in *Symposium on Computer Animation*, 2016, pp. 169–178.

[24] T. Jakobsen, "Advanced character physics," in *Game Developers Conf.*, vol. 3. IO Interactive, Copenhagen Denmark, 2001, pp. 383–401.

[25] R. Baltman and R. Radeztsky Jr, "Verlet integration and constraints in a six degree of freedom rigid body physics simulation," in *Game Developers Conference*, 2004.

[26] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun, "Discrete elastic rods," in *ACM SIGGRAPH 2008 papers*, 2008, pp. 1–12.

[27] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109–118, 2007.

[28] E. Coumans, "Bullet physics simulation," in *ACM SIGGRAPH 2015 Courses*, 2015, p. 1.

[29] E. Catto, "Modeling and solving constraints," in *Game Developers Conference*, 2009, p. 16.

[30] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *IEEE Transactions on Control Systems Technology*, vol. 2, no. 2, pp. 123–134, 1994.

[31] M. Ciocarlie, C. Lackner, and P. Allen, "Soft finger model with adaptive contact geometry for grasping and manipulation tasks," in *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07)*. IEEE, 2007, pp. 219–224.

[32] A. Munawar, N. Srishankar, L. Fichera, and G. S. Fischer, "A parametric grasping methodology for multi-manual interactions in real-time dynamic simulations," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 8712–8718.

[33] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Transactions on Computers*, vol. 100, no. 6, pp. 623–629, 1971.

[34] J. F. Blinn, "Models of light reflection for computer synthesized pictures," in *4th Annual Conf. on Computer Graphics and Interactive Techniques*, 1977, pp. 192–198.

[35] C. Schlick, "An inexpensive BRDF model for physically-based rendering," in *Computer Graphics Forum*, vol. 13, no. 3. Wiley Online Library, 1994, pp. 233–246.

[36] P. Debevec, "Image-based lighting," in *ACM SIGGRAPH 2006 Courses*, 2006, pp. 4–es.

[37] A. Munawar and G. S. Fischer, "An asynchronous multi-body simulation framework for real-time dynamics, haptics and learning with application to surgical robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6268–6275.

[38] A. Munawar, N. Srishankar, and G. S. Fischer, "An open-source framework for rapid development of interactive soft-body simulations for real-time training," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 6544–6550.

[39] J. Y. Wu, A. Tamhane, P. Kazanzides, and M. Unberath, "Cross-modal self-supervised representation learning for gesture and skill recognition in robotic surgery," *International Journal of Computer Assisted Radiology and Surgery*, vol. 16, no. 5, pp. 779–787, 2021.

[40] J. Y. Wu, A. Munawar, M. Unberath, and P. Kazanzides, "Learning soft-tissue simulation from models and observation," in *International Symposium on Medical Robotics (ISMR)*. IEEE, 2021.

[41] G. M. Chaikin, "An algorithm for high-speed curve generation," *Computer Graphics and Image Processing*, vol. 3, no. 4, pp. 346–349, 1974.