frontiers | Frontiers in Virtual Reality

TYPE Original Research
PUBLISHED 27 June 2023
DOI 10.3389/frvir.2023.1130156

OPEN ACCESS

EDITED BY
Marco Parente,
University of Porto, Portugal

REVIEWED BY
George Papagiannakis,
University of Crete, Greece
Ryan Beasley,
Private Practitioner, San Carlos, CA, United States

*CORRESPONDENCE
Rachel B. Clipp,
Rachel.clipp@kitware.com

RECEIVED 22 December 2022
ACCEPTED 31 May 2023
PUBLISHED 27 June 2023

CITATION
Moore J, Scheirich H, Jadhav S, Enquobahrie A, Paniagua B, Wilson A, Bray A, Sankaranarayanan G and Clipp RB (2023), The interactive medical simulation toolkit (iMSTK): an open source platform for surgical simulation. *Front. Virtual Real.* 4:1130156. doi: 10.3389/frvir.2023.1130156

The interactive medical simulation toolkit (iMSTK): an open source platform for surgical simulation

Jacob Moore[1], Harald Scheirich[1], Shreeraj Jadhav[1], Andinet Enquobahrie[1], Beatriz Paniagua[1], Andrew Wilson[1], Aaron Bray[1], Ganesh Sankaranarayanan[2] and Rachel B. Clipp[1]*

[1]Medical Computing Group, Kitware Inc.,Carrboro, NC, United States, [2]Department of Surgery and Department of Biomedical Engineering, University of Texas Southwestern Medical Center, Dallas, TX, United States

Introduction: Human error is one of the leading causes of medical error. It is estimated that human error leads to between 250,000 and 440,000 deaths each year. Medical simulation has been shown to improve the skills and confidence of clinicians and reduce medical errors. Surgical simulation is critical for training surgeons in complicated procedures and can be particularly effective in skill retention.

Methods: The interactive Medical Simulation Toolkit (iMSTK) is an open source platform with position-based dynamics, continuous collision detection, smooth particle hydrodynamics, integrated haptics, and compatibility with Unity and Unreal, among others. iMSTK provides a wide range of real-time simulation capabilities with a flexible open-source license (Apache 2.0) that encourages adoption across the research and commercial simulation communities. iMSTK uses extended position-based dynamics and an established collision and constraint implementations to model biological tissues and their interactions with medical tools and other tissues.

Results: The platform demonstrates performance, that is, compatible with real-time simulation that incorporates both visualization and haptics. iMSTK has been used in a variety of virtual simulations, including for laparoscopic hiatal hernia surgery, laparoscopic cholecystectomy, osteotomy procedures, and kidney biopsy procedures.

Discussion: iMSTK currently supports building simulations for a wide range of surgical scenarios. Future work includes expanding Unity support to make it easier to use and improving the speed of the computation to allow for larger scenes and finer meshes for larger surgical procedures.

KEYWORDS

virtual simulation, open source, surgical simulation, soft tissue dynamics, position-based dynamics, continuous collision detection, medical simulation

1 Introduction

Human error is one of the leading causes of medical error. It is estimated that human error leads to between 250,000 and 440,000 deaths each year (Gruen et al., 2006; Committee on Pediatric Emergency Medicine, 2007; James, 2013; Makary and Daniel, 2016). Medical simulation has been shown to improve the skills and confidence of clinicians and reduce

TABLE 1 Comparison of different libraries for medical simulation of tissues and tools.

Software	License/ Cost	Haptics support	Modeling	Supported languages	Supported game engines	Active community
iMSTK	Apache 2.0 Free to Use	Haply OpenHaptics	Particle-Based Dynamics Smoothed-Particle Hydrodynamics Level Set Method	C++ C#	Unity Unreal	Active
SOFA	Lesser GPL	OpenHaptics	Particle-Based Dynamics Smoothed-Particle Hydrodynamics	C++	3rd Party Unity Asset	Active
Obi	License Fee	-	Particle-Based Dynamics	C#	Unity	Active
OSS	Apache 2.0 Free to Use	OpenHaptics	Finite Element Methods Rigid Bodies Smoothed-Particle Hydrodynamics	C++	-	Inactive

medical errors (Nishisaki, Keren, and Nadkarni, 2007; Falcone et al., 2008; Nishisaki et al., 2011). Simulation is used across a broad range of areas with a variety of techniques (Foronda et al., 2020; E. M; Sims, 2007; E; Sims et al., 2016; J. L; Arnold, 2017; J; Arnold and Diaz, 2016), including live patient interactions (Fu et al., 2017), mannequins ("Caehealthcare, 2017.; "SynDaver Labs, 2017; "Laerdal, 2017a; "Laerdal, 2017b; Willie et al., 2016; Issenberg et al., 2005), virtual and augmented reality (Bauman, 2013; Sims, Ed and Powell, 2015; Brown, McIlwain, and Willson, 2016; Clipp et al., 2019; Taylor et al., 2019.; Couperus et al., 2020), and skill trainers (Frantzides, 2007; Hoopes et al., 2020). Surgical simulation is critical for training surgeons in complicated procedures and can be particularly effective in skill retention (Stefanidis et al., 2005; Dimitrios Stefanidis, Acker, and Heniford, 2008; Mannella et al., 2019). While physical simulators can be effective for training particular skills, including suturing, these do not provide the realistic visual and physical feel required for entire procedures. Virtual simulators are a growing part of the simulation marketplace (Schijven and Jakimowicz, 2003). The use of virtual reality to support medicine is a growing commercial and research endeavor (Kitware, 2022a; Kitware, 2022b; "EducationXR, 2022" n. d.; Zikas et al., 2023; Yang, Zhou, Chen, et al., 2022a; Yang, Zhou, Song, et al., 2022b; Bansal et al., 2022), including for surgical training (Katić et al., 2013; Patel and Patel, 2014; Parham et al., 2019). However, for surgical simulation in particular, it is critical to include accurate, physics-based soft-body mechanics and haptics to ensure the tissue responses are realistic both visually and physically (Yiannakopoulou et al., 2015).

There are several existing technologies to build surgical simulators, including Simulation Open Framework Architecture (SOFA) (SOFA, 2022), OpenSurgSim (OSS) (OpenSurgSim, 2022), Obi (Obi, 2022), and the Unity (Unity, 2022) and Unreal (Unrealengine, 2022) game engines. These applications and research offerings have some limitations. Unity and Unreal are commercial game engines used widely to develop virtual content and virtual reality applications. While these engines have built-in advantages, such as GPU-enabled shaders, that can be leveraged for simulation, they have significant limitations when modeling fluid behavior and soft-body mechanics. This is particularly true when realistic physics-based behavior is required, as in surgical procedure simulation. SOFA is an open-source library for prototyping and researching physics-based simulation. However, this framework has disparate licensing and limited support for different modules, which can limit adoption by commercial companies. OSS was developed by SimQuest; however, this company is no longer in business and OSS has not been supported in recent years. Obi is a licensed position-based dynamics framework that can only be used via Unity. It can be purchased via the Unity Asset Store. The interactive Medical Simulation Toolkit (iMSTK) is an open-source platform with position-based dynamics, continuous collision detection, smooth particle hydrodynamics, integrated haptics, and compatibility with Unity and Unreal, among others. iMSTK provides a wide range of real-time simulation capabilities with a flexible open-source license (Apache 2.0) that encourages adoption across the research and commercial simulation

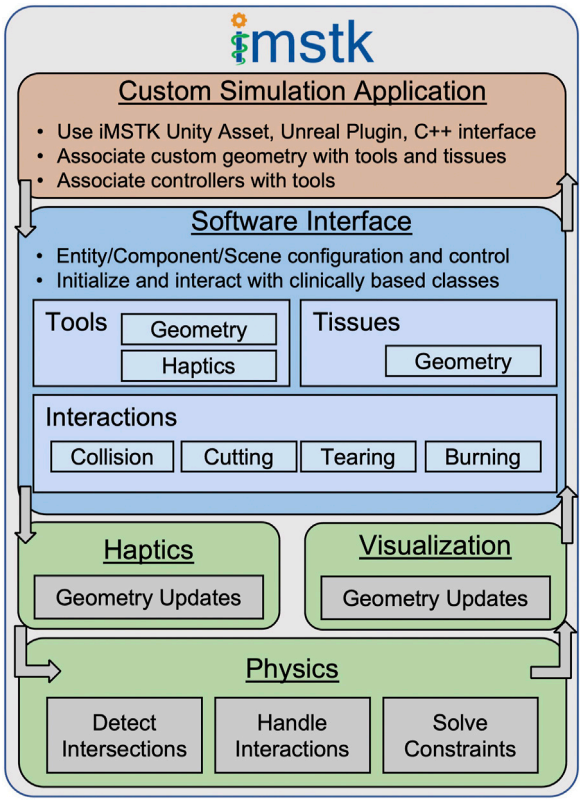

FIGURE 1
The iMSTK software architecture consists of an Entity Component System (ECS) for scene creation and modules to encapsulate physics, haptic, and visualization algorithms.

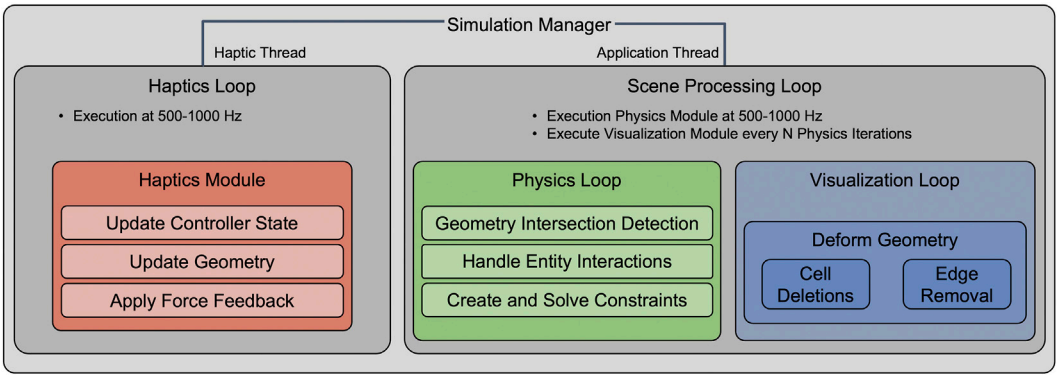

FIGURE 2
The iMSTK simulation loop executes the three modules, physics, visualization, and haptics. The haptic module must execute at 500–1000 Hz to maintain smooth motion for the users, the physics module must execute at a frequency at least that of the haptics module to provide updated forces for rendering. The visualization module can execute at a lower frequency to maintain performance and still provide immersive graphics.

communities. A comparison of these simulation frameworks is shown in Table 1.

2 Methods

The Interactive Medical Simulation Toolkit (iMSTK) is a C++ open-source platform for physics simulation specific to the medical field for the development of surgical training content. iMSTK provides a software framework for describing a simulation scene and a collection of modules that encapsulate various simulation methodologies. The architecture aims to reduce the nuanced understanding of physics-based models, that is, required to build medical simulations by increasing the usability. Simulation developers can more easily prototype and refine simulations by using a clinically based and extensible interface for defining human anatomy and medical tools. The iMSTK software architecture consists of an Entity Component System (ECS) for scene creation and modules to encapsulate physics, haptic, and visualization algorithms, as shown in Figure 1. The provided platform can be integrated with and provide physics to simulators developed in a variety of visualization platforms, including the Visualization Toolkit (VTK) (Schroeder, Martin, and Lorenson, 2006), Unity and Unreal. An optional VTK interface is provided within iMSTK. An optional C# interface is also available and is used extensively within the iMSTK Unity Asset. As the Unreal game engine is a C++ based architecture, iMSTK can also be used by Unreal developers.

2.1 iMSTK overview

iMSTK provides a class-based software framework to simplify development for simulation creators. The architecture was developed based on the ECS pattern. End users can create entities to represent various medical tools, tissues, and organ anatomy. Material properties, such as elasticity and stiffness, can be used to describe various physical attributes of the entity. Interactions, such as collisions, can then be added to entities as components. iMSTK provides a set of predefined components that can be used during a surgical scenario, including grasping, tearing, and burning. This framework was designed to support and encapsulate general computational and numerical methods and can easily be extended to include new components.

The simulation system is implemented in three distinct modules: physics, haptics, and visualization. The simulation loop can execute the three modules at different rates to ensure accurate physics, realistic smooth "touch" in the haptics, and immersive visualizations, Figure 2. The physics module encapsulates the implementation and execution of numerical methods for creating and solving constraints of the dynamic system within a simulation. Extended Position-Based Dynamics (xPBD) is the predominant method for simulating dynamic systems in iMSTK. However, smooth particle hydrodynamics have also been implemented in iMSTK (Feiger et al., 2020). This module also encapsulates logic associated with each of our provided entity interactions, such as grasping, suturing, and tearing. The visualization module tracks and updates the state of geometries based on the results of the physics simulation. The module runs serially with the physics module. For example, as the physics module calculates the constraints related to tissue removal by blunt dissection, the visualization module will use the calculated constraint values to remove cells/edges in the geometry when those values exceed a threshold. The haptic module renders the physics forces to the controller, such as (3D Systems, Rock Hill, SC), and renders the user input (movement) to the physics module. Controllers are associated with objects in the virtual scene, such as medical tools. The iMSTK simulation loop executes the three modules, physics, visualization, and haptics. The haptic module must execute at 500–1000 Hz to maintain smooth motion for the users, the physics module must execute at a frequency at least that of the haptics module to provide updated forces for rendering. The visualization module can execute at a lower frequency to maintain performance and still provide immersive graphics.

In the following sections, we highlight the physics-based models, collision detection algorithms, haptics integration, and the software processes used to ensure quality open source code for use by the medical modeling and simulation community.

2.2 XPBD—Extended Position Based Dynamics

iMSTK leverages components to define the dynamic behavior of entities in a simulation. The primary physics implementation in iMSTK is an xPBD solver that builds on Position Based Dynamics (PBD). PBD (Bousseau et al., 2017) is a numerical approach to simulating the behavior of dynamic systems where the solver depends solely on particle positions and their associated constraints. Constraints define the expected behavior of the interactions between a set of points in a simulation mesh. iMSTK supports multiple types of constraints found in the literature (Müller et al., 2007; Macklin, Müller, and Chentanez, 2016; Müller et al., 2020). The constraints most used in iMSTK are distance, volume, finite element method (FEM), and dihedral angle. The distance constraint constrains the distance between two adjacent points in a mesh to ensure a given distance is maintained. Similarly, the volume constraint maintains the volume of a tetrahedral cell. The FEM constraints mimic the finite element method by minimizing the strain energy of a given cell. The dihedral angle constraint maintains the angle between two adjacent triangles to be near a set initial configuration. Each constraint type is suited to different real-world bodies. Distance and volume constraints are often used together for soft body simulations, while FEM constraints are a higher fidelity approach to soft bodies. Dihedral angle constraints are often used for simulating thin tissues. This contrasts with traditional approaches to physics simulations that rely on Newtonian mechanics to drive simulations using force and impulse. PBD has become more popular due to its simplicity and computational efficiency in simulating visually plausible physics for computer games and virtual reality (VR) applications, such as surgical simulators. xPBD builds upon the concepts of PBD (Bousseau et al., 2017) with the addition of a total Lagrange multiplier to stabilize the solution with respect to time step and iteration count. xPBD also includes physics-based constraints derived from well-defined concepts of strain energy potentials (Bender et al., 2014; Macklin, Müller, and Chentanez, 2016). xPBD has been commonly used to simulate deformable materials, but it has recently been extended to the simulation of rigid bodies (Müller et al., 2020). By leveraging xPBD, iMSTK can simulate the behavior of complex deformable biological tissues and interactions between rigid surgical instruments and organs. iMSTK's unified xPBD solver works by collecting all the constraints currently active in a scene and solving them using an iterative Gauss-Seidel method (Müller et al., 2007). This approach allows for the internal constraints associated with the deformation of a body to be solved in the same system as the external constraints associated with collisions with other simulated objects. This method increases the overall stability of the system.

2.3 Collision detection and handling

iMSTK supports multiple types of collision interactions between objects in a surgical scenario enabling more complex collisions, such as tool to tissue, thread to tissue, and tissue to tissue. Collisions are resolved in two steps: detection and handling. For collision detection, a collision manifold is calculated. The collision manifold defines the geometry of the interface between two colliding entities. These geometries can be analytic, such as a capsule, cylinder, or sphere, implicit, such as a signed distance field, or level set, or a mesh, such as a line mesh, triangle mesh, tetrahedral mesh, or hexahedral mesh). iMSTK entities contain collider components that store the collision geometry and metadata, that is, used to define how the collision should be handled. The colliders are used to define the collision manifold that contains collision data that gets passed to the collision handler. The collision handler uses the collision data to generate a set of constraints. These constraints are added to the xPBD system to resolve the collisions so that the objects are no longer colliding. There are two main types of collision detection methods, static and dynamic. iMSTK primarily leverages static collision detection methods, but it also contains a set of dynamic collision detection methods for more complex interactions.

2.3.1 Static collision detection

Static collision detection methods look at the geometric configuration of two colliders at a single instance in time. These are the primary methods that iMSTK uses to detect and handle collisions because they are more performant and require less memory than dynamic collision detection (Ericson, 2005). Static collision detection works well for solid geometries with a sufficient spatial footprint that a small simulation timestep is unlikely to miss any intersections. iMSTK does not currently support all possibilities for geometry collision. The collision detection types are all subclasses of a general collision detection algorithm, making it easy to extend the list. Table 2 shows the set of currently supported collision detection types in green. The collision types in grey could be easily added to iMSTK.

2.3.2 Dynamic collision - Continuous collision detection

Dynamic collision detection methods look at the position and orientation of the geometries at multiple time steps to find both where and when objects collide. Continuous collision detection (CCD) is a class of dynamic collision detection algorithms that are often necessary for robust operation in cases of fast-moving objects or thin geometries where static collision detection of intersections is insufficient. CCD algorithms are more suitable in the case of thin geometries, such as surfaces, lines, and points, and degenerate intersections, such as three-dimensional line-line intersections, as they compare continuity between two consecutive time steps for detecting intersections/collisions. iMSTK supports CCD between line meshes based on the method described by (Qi et al., 2017). This method, combined with our xPBD solver, enables iMSTK to simulate thin threads, such as those required for suturing simulations.

CCD between two line meshes reduces to a computation of pairwise line-line (cell-cell) intersections across two consecutive time steps. In the case of self-collision, the immediate neighbors of a line (cell) can be ignored as they are always attached to each other. Further optimization can be achieved through the early elimination of pairs through space partitioning. However, this is

TABLE 2 The collision detection types in green are currently supported in iMSTK. iMSTK can be easily extended to support the types in grey.

Geometry	Capsule	Cylinder	Implicit	LineMesh	Box	Plane	PointSet	Sphere	Surface mesh	Geometry
Capsule										Capsule
Cylinder										Cylinder
Implicit										Implicit
LineMesh										LineMesh
Box										Box
Plane										Plane
PointSet										PointSet
Sphere										Sphere
Surface Mesh										Surface Mesh

not a part of the current implementation and will be addressed in future releases of iMSTK. A constant thickness is set for the line meshes. Six distinct cases are considered when processing line-line intersections:

1. Collision between the two lines in the current time-step (due to thickness)
   a) edge-edge proximity, b) vertex-vertex proximity c) vertex-edge proximity
2. Crossing of edges between the two consecutive frames
   a) edge-edge crossing, b) vertex-vertex crossing c) vertex-edge crossing

In the cases of 1a, 1b, and 1c, only the current time step is required for a static proximity check within the tolerance defined by the thickness parameter. If the closest points on the edges (to each other) are within the interior of the edges, this results in a collision by case 1a. If the closest points are not in the interior of the line segments, a test is done to check if any pair of vertices are closer than the thickness, which can lead to collision by case 1b. Finally, if the closest point on at least one edge is in the interior, we test for vertex-edge proximity. Cases 2a, 2b, and 2c are handled similarly by comparing two consecutive time steps. For example, for case 2a to be true, the closest points on the segments should be in the interior for both time steps, and the vector connecting the two points is inverted between the time steps, indicating that the lines have crossed each other. This can be done using a simple dot product test for negative magnitude.

2.4 Haptics integration

Force feedback is important in many medical simulations to provide the sensation of touch to the user/trainee. The ability to incorporate a haptic device to ensure realism is important for the development of simulators. iMSTK includes support for haptic integration through the OpenHaptics ("OpenHaptics Developer Software" n. d.) and the Virtual Reality Peripheral Network (VRPN) (VRPN, 2022) software packages. OpenHaptics is a commercial library that provides support for the 3DSystems suite of haptic devices, including the Touch. This device is used in many of the projects discussed in the Results section. VRPN is an open source, device-independent, network-transparent system for accessing virtual reality peripherals, including haptic devices. VRPN introduces a layer for device-independent data input and output via Transmission Control Protocol (TCP). iMSTK's controller implementation allows for multiple haptic devices to be used simultaneously in a scene. This is important for medical procedures that require a tool in each hand, such as suturing.

The position and orientation of the haptic device are read by iMSTK and used to update models and constraints. The calculated force and torque information is then written to the device to provide the touch sensation to the user. iMSTK has been integrated with both three degrees of freedom (DOF) (no torque rendering) and five DOF (torque rendering) haptic devices. There are two methods available for force rendering, direct and virtual coupling. With direct coupling, the force is directly sampled and rendered to the device. However, when significant changes in force are rendered, the solution becomes unstable and leads to vibration and discontinuities in the sensation felt by the user. Dynamic virtual coupling was proposed as a solution to this significant limitation (Colgate, Stanley, and Brown, 1995). The position and orientation are connected via a spring damper system and the resulting force is rendered to the user. When moving the manipulator in open space, free of contact, the virtual pose is easily able to keep up with the real pose resulting in no haptic forces. Contact results in a smoothly increasing force, which eliminates the discontinuities present in direct coupling. However, the spring damper parameters must be tuned to achieve the desired haptic behavior for a given system.

2.5 Suturing implementation

Suturing is a task common to most surgical operations that involves the closing of wounds from either traumatic events or surgical operations (Lloyd, Marque, and Kacprowicz 2007). To support this task in iMSTK, the approach requires needle-tissue interactions, needle-thread interactions, and thread-thread interactions. The current approach implements a needle, thread, and interaction for both objects with the surface mesh of a soft body. The needle is

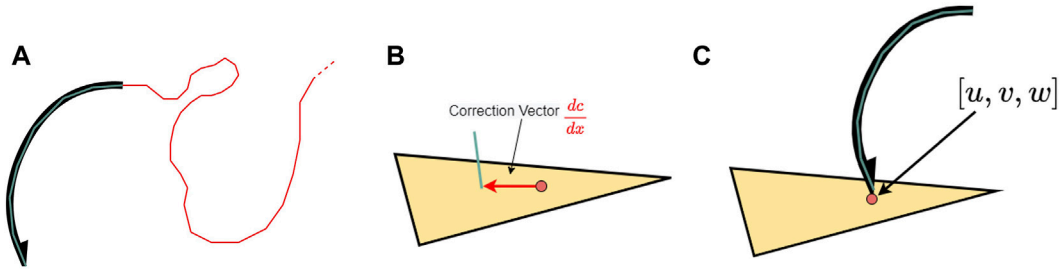

FIGURE 3
(A) The needle is represented by a visual mesh (black) and a line mesh (cyan) for collision and puncture interactions. The suture thread (red) is a PBD line mesh with distance and bend constraints applied as well as self-continuous collision detection. **(B)** The penetration point for the needle is represented using the barycentric coordinates on the punctured triangle. **(C)** The insertion constraint applied a distance constraint that only considers displacement between the line mesh and puncture point along the triangle's surface, which lets the needle slide through the punctured tissue.

represented by both a surface mesh to represent the visual geometry of the needle and a rigid line mesh to represent the physical geometry. This needle object is then connected to a line mesh thread object (with self CCD applied) via the use of the end of the needle object as a boundary condition as shown in Figure 3A. To begin the simulation, the needle state is set to unpunctured. In this state, collision is used to detect when the needle is in contact with the tissue, and to determine if the needle orientation has the tip pushing into the body. When these conditions are met, a puncture is detected and the barycentric coordinates of the puncture points and the index of the surface triangle are saved, Figure 3B. A constraint is then specified to bind the puncture point to follow the tangential motion of the needle. The associated correction vector, $\frac{dc}{dx}$, is defined as a vector pointing from the puncture point to the nearest point along the line mesh representation of the needle, Figure 3C.

2.6 Software process

iMSTK development focuses on creating interfaces that provide end users with intuitive access to accurate, validated physics. By providing a well-defined architecture, we can focus on implementing and encapsulating our numerical methods and providing them in a way that software developers with little knowledge of the medical and numerical methods domains can easily create a variety of medical simulation applications. iMSTK is a cross platform library (Windows and Linux); however, many of the haptic drivers are limited to Windows operation. Due to its integration with Unity and Unreal, applications built using iMSTK are compatible with a variety of VR headsets. Due to the computational requirements, we do not recommend the use of iMSTK on mobile platforms.

2.6.1 Version control

iMSTK utilizes Git for version control and is hosted on Kitware's Gitlab site ("IMSTK," n. d.). Version control is essential for open source platforms to support distributed and cooperative development of the same code base. Having iMSTK in a public repository allows anyone to communicate, collaborate, and contribute to the development of the software. This approach also provides transparency and trust as anyone can view and comment on changes to the code base. Git provides the ability to tag a certain version of the code as a release. Git also provides pipelines for Continuous Integration that can run all tests for any specified branch and/or commit. As new features are completed, tested, and validated, the iMSTK team will add a version tag to the repository that will allow users to pull a completed version of the code that has passed the rigorous testing process.

2.6.2 Testing

iMSTK utilizes both unit tests and visual tests to ensure the quality of the code. The Google test library (Google, 2022) is used for unit testing, which is executed nightly for both Windows and Linux platforms. Unit tests verify accurate and consistent functionality of specific algorithms in iMSTK. Visual tests are used to fill a need for acceptance testing. These maintain testable code and examples for visual acceptance of functionality. They can be executed automatically, but results must be visually inspected for verification. This ensures code execution at a minimum and reasonable output. However, this can lead to minor changes incorrectly passing visual inspection. Future work will include migrating results to numeric evaluation for automated and more reliable testing.

3 Results

3.1 iMSTK model results

3.1.1 XPBD performance and results

xPBD was originally developed to allow for the simulation of deformable materials with comparable accuracy to finite element methods more commonly used in engineering analysis, but with the performance required to allow for real-time interactive simulations (Bender et al., 2014). Figure 4 shows the performance of four common cases implemented in iMSTK. These results were generated by simulating a meshed cube under gravitational loading with a varying number of constraints applied by iteratively refining the mesh. The red, green, and blue lines indicate two, five, and eight iterations for the Gauss-Seidel solver, respectively. The 4th plot includes the addition of contact constraints along the surface of the cube, which shows that adding collision constraints does not heavily affect the

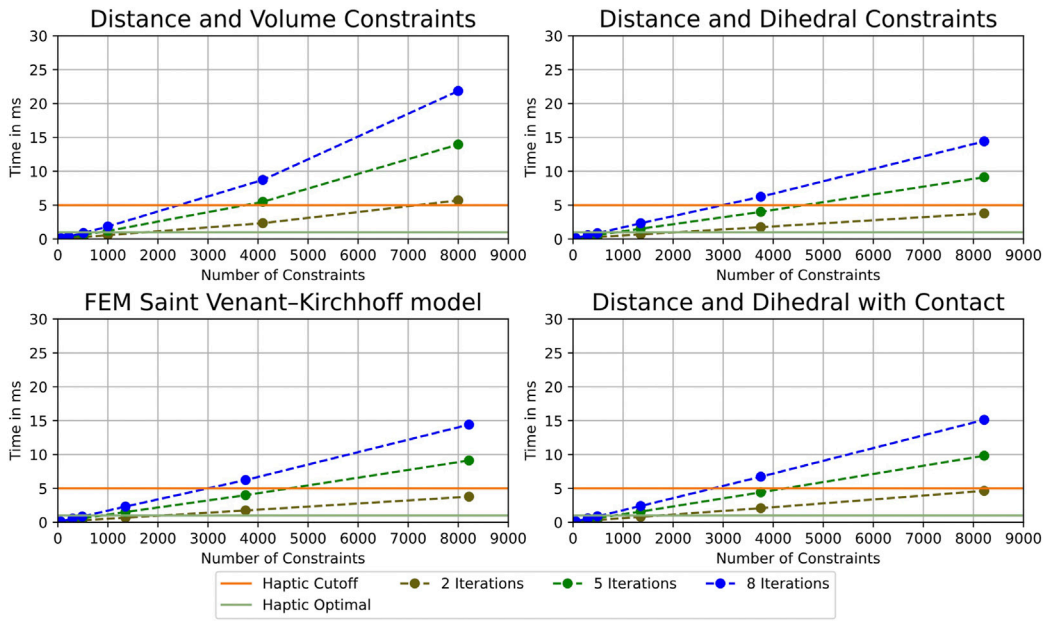

FIGURE 4
The performance of our xPBD solver using Distance and Volume Constraints (top left), distance and dihedral angle constraints (top right), the FEM Constraints using a Saint Venant-Kirchhoff model (bottom left), and Distance and dihedral with collision. These demonstrate the scene size that iMSTK is able to simulate and maintain realistic (smooth) haptics.

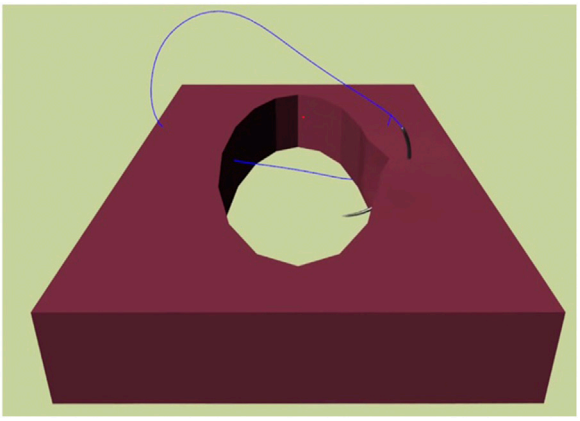

FIGURE 5
Suturing in iMSTK is achieved by simulating the interaction between the needle, tissue, and thread using constraints to represent the puncture point and path of the needle as well as the path of the thread through the tissue.

overall performance of the solver. The light green line represents the optimal performance for haptics (Colgate, Stanley, and Brown, 1995); however, our testing has shown simulations that can solve constraints in less than 5 milliseconds (ms) per frame are stable and feel smooth. Therefore, iMSTK uses 5 m per frame as a threshold for performant simulations. Our tests have also shown that a deformable organ can be simulated with ~500 constraints, any given scene can have upwards of ~2000 constraints for optimal performance, and up to ~7000 constraints before simulation instability becomes noticeable. All calculations were performed on a laptop with a 12th Generation Intel Core i9-12900 HK processor with 32 GB of RAM with the Windows operating system. The iMSTK results demonstrate that the system is perfomant enough to simulate a surgical simulation scene. However, future work will include performance optimization of the physics module to increase the complexity and size of the simulation scene iMSTK is able to compute.

3.1.2 Suturing example

An example of suturing with a tissue block, needle, and thread is shown in Figure 5. Future work in suturing will include adding more realistic coupled constraints between the thread and tissue, providing increased physical realism and haptic force feedback. We will also integrate a knot detection algorithm to detect what type of knots the user ties for training purposes.

3.2 Case studies–Surgical simulators

3.2.1 Virtual laparoscopic hiatal hernia simulator

A hiatal hernia is a condition involving herniation of the contents of the peritoneal cavity, most commonly the stomach, through the esophageal hiatus of the diaphragm, and into the mediastinum. A strong link has been established between gastroesophageal reflux disease (GERD) and the presence of hiatal hernia (Hyun and Bak, 2011). Laparoscopic hiatal hernia surgery is a minimally invasive procedure performed with specialized tools through five small incisions on the abdomen. In this procedure, the hernia sac is resected, and the hernia contents are

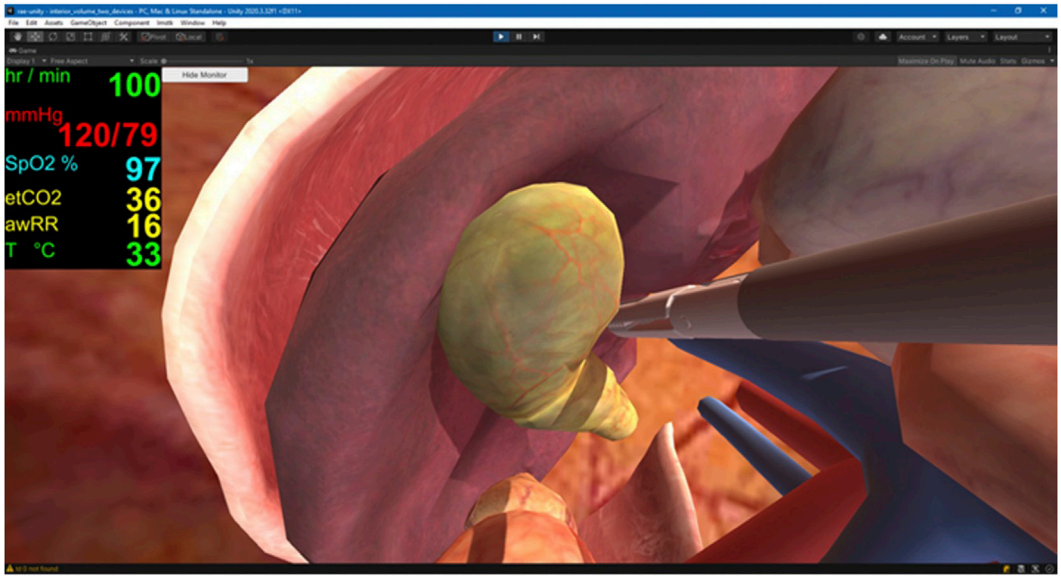

FIGURE 6
Virtual Laparoscopic Cholecystectomy. This simulator is built in Unity and leverages the iMSTK Unity Asset to simulate the interactions between objects in the scene. The Pulse Physiology Engine is also integrated with the simulator for patient vital sign feedback during the simulated procedure.

reduced. This is done to the extent that proper abdominal esophageal length is achieved to restore the normal anatomical position of the stomach, and the crural defect is closed to restore the diaphragmatic hiatus to fit the esophagus. Finally, an anti-reflux operation is performed, traditionally a fundoplication procedure, to reinforce the lower esophageal sphincter. It is an advanced general surgery procedure that has a learning curve of about 50 cases to achieve an appropriate recurrence rate of the hiatal hernia (Neo et al., 2011). We are building the first simulator to address the training gap using iMSTK for realistic and accurate modeling of the tissue and tool interactions that occur during the procedure. We are in the process of developing both the crural repair and the fundoplication steps of this procedure by leveraging the Unity game engine and the iMSTK Unity Asset ("IMSTK | Physics | Unity Asset Store" n. d.). The deformable model of the stomach and esophagus are modeled using the xPBD method with a neo-Hookean material model. The stomach mesh is comprised of 2,485 cells. The boundary conditions of the stomach and esophagus have been modeled using realistic constraints from gastrosplenic, gastrohepatic, and gastrocolic ligaments, which are all modeled as thin tissues. Collision detection and response between the laparoscopic tools and the virtual models are implemented for realistic interaction in the simulator. Two Touch (3DSystems, 2022) haptic devices are integrated with the simulation for tool manipulation and scene interaction. Future work will continue development of the three modules: fundoplication, crural repair, and stomach dissection.

3.2.2 Virtual laparoscopic cholecystectomy

Cholecystectomy (gallbladder removal) remains one of the top 10 commonly performed types of surgeries in the United States, with ~500,000 operating room procedures per year (HCUP-US, 2022). Complications in laparoscopic cholecystectomy, including bile duct injury and obstruction, and blood vessel injury, occur in 0.5%–1% of cases (Shamiyeh et al., 2004; Sureka and Mukund, 2017). Many of these injuries go unrecognized during the surgery (approximately 3 out of 4 injuries in one study went unrecognized) (Hugh, 2002). Practicing surgeons in smaller facilities may not be exposed to enough variants to recognize them and make appropriate decisions to respond. Low-volume surgeons, such as those in smaller facilities, may perform only 40–100 laparoscopic cholecystectomies per year, compared to 200–250 per year for a high-volume surgeon at a large urban hospital (Sakpal, Bindra, and Chamberlain, 2010). A virtual laparoscopic cholecystectomy simulator is being developed to address this clinical problem. The simulator is being developed using the Unity Game Engine and iMSTK's Unity Asset. This allows access to Unity's easy features for including anatomy, lighting, and scene composition with iMSTK's physics-based models critical for accurate look and feel of human tissue and organ manipulation. The simulator uses iMSTK's xPBD constraints to control collisions between tools and organ models. The force is then calculated from the mesh displacement of the soft-body organ. The total number of cells representing the deformable tissues in the scene is 1,295. Like the laparoscopic hiatal hernia, two Touch haptic devices are integrated with the simulator and use virtual tool coupling to provide force feedback based on the force calculation of the solver. This simulator also includes integration of the open source Pulse Physiology Engine (Bray et al., 2019) via the Unity Asset (Girault, Bray, and Clipp, 2019) for patient vital sign feedback throughout the procedure. The first year of development is showcased in Figure 6. Future years will focus on adding connective tissue to the simulator, improving methods required for blunt and sharp dissection, and adding electrosurgery to iMSTK.

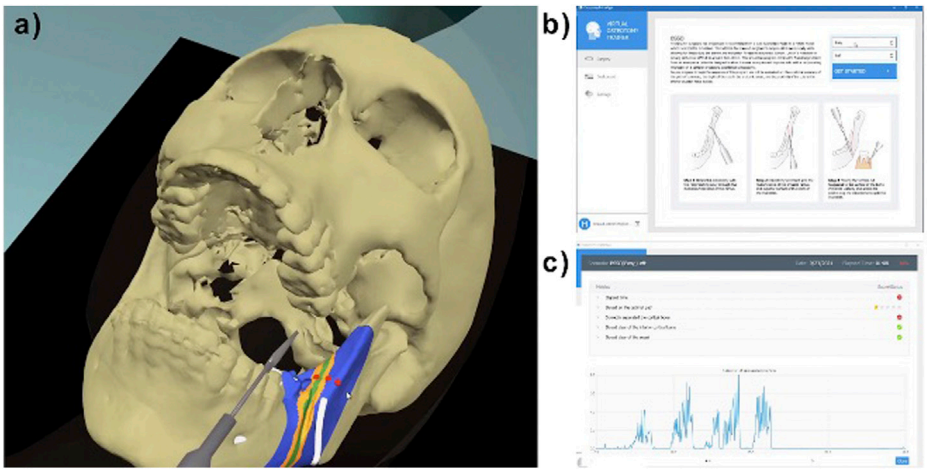

FIGURE 7
Osteotomy Trainer **(A)** BSSO virtual surgical scenario, with desired (green), acceptable (orange) and wrong (blue) osteotomy path. Red dots indicate the guidance mode that mimics the drilling holes surgeons do in the OR to guide their final cuts. White tubes show the retrofacial and mandibular nerve/vessel bundles. **(B)** Initial screen in the simulator graphical user interface. **(C)** Metrics screen in the graphical user interface.

3.2.3 Osteotomy simulator

Oral and Maxillofacial (OMF) surgery is considered a dental specialty, but both medicine and dentistry contribute to the unique scope, skills, and training needed to perform OMF surgery. Because of this duality, there has been a continued debate about how to train OMF surgeons (Punjabi and Haug 1990; Laskin, 2016). The Virtual Osteotomy Trainer, Figure 7, was designed to help improve procedural knowledge and surgical proficiency for performing osteotomies during orthognathic surgery. Specifically, one of the featured osteotomy surgical training scenarios is a bisagittal split osteotomy (BSSO). BSSO is when the lower jaw is separated from the face and repositioned to correct skeletal malocclusions. This type of jaw surgery requires significant precision, as it requires making shallow cuts in specific bone locations, while avoiding nerves and vessels. This simulator can increase the efficiency of surgical training in a risk-free training environment. iMSTK was used to support this simulator to ensure visual and haptic accuracy during bone cutting. iMSTK was further updated as part of this project to support volume-rendering via ray casting techniques. The geometry (anatomy) is rendered directly from volumetric data, that is, edited in real-time in the virtual surgical environment. As the user modified the geometry during bone cutting, the volume was updated via GPU-enabled ray casting. A piecewise update is conducted to re-render only the modified section of the geometry. VTK was used for the visualization of this project. This project also uses haptic-force feedback to provide sensation during the bone cutting process. The PhysX rigid body dynamics engine (NVIDIA, 2022) was integrated with iMSTK to allow estimation of the contact between the bone and the saw. This contact is directly sampled from the level set using the depth (signed distance) and the normal of the gradient of the level set field.

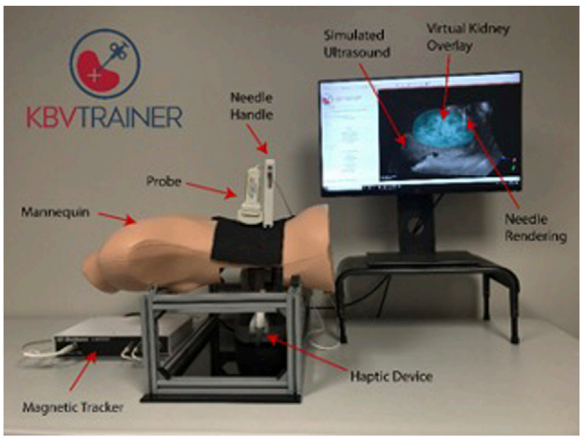

FIGURE 8
Kidney Biopsy Virtual Trainer. The KBVTrainer leverages a hardware setup integrated with iMSTK and the open source 3DSlicer platform to create the virtual software simulation.

3.2.4 Kidney Biopsy Virtual Trainer

The overall prevalence of chronic kidney disease (CKD) in the general population is approximately 14 percent, with more than 661,000 Americans having kidney failure (NIDDK, 2022). Ultrasound-guided renal biopsy is a critically important tool in the evaluation and management of renal pathologies, and it requires competent biopsy technique and skill to consistently obtain high-yield biopsy samples. The KBVTrainer (Kidney Biopsy Virtual Trainer) was developed using iMSTK for ultrasound-guided kidney biopsy training. KBVTrainer is intended to train radiologists, nephrologists, and interventional radiologists to improve procedural skill competence in ultrasound-guided renal biopsy. This virtual simulator provides several advantages, including acceleration of the training of ultrasound-guided renal biopsy in a risk-free environment to improve the safety of kidney biopsy and ensuring

that the biopsy procedure yields high-quality specimen. The virtual simulator includes 1) a mannequin, 2) a real ultrasound probe, 4) a real biopsy needle, 4) an electromagnetic tracker (Ascension 3D Guidance trakSTAR) to track the position and orientation of the ultrasound probe, and e) a Touch haptic device for force feedback at the needle. This setup is shown in Figure 8. Software visualization and hardware interfacing for the trainer were provided by the open source software packages 3D Slicer (3D Slicer, 2022) and PLUS Toolkit (Plus Toolkit, 2022), respectively. The physics of the tissue deformation, needle-tissue interaction, 3D needle tracking and haptics are simulated using iMSTK.

Workflow of the application is as follows. The Slicer application receives tracking data of the US probe from a magnetically tracked sensor from PLUS over a local network connection. Needle tracking data is received directly through iMSTK. Using the probe's tracking information, the pre-acquired US volume is resliced along the plane of the probe to generate an image that reflects the current location and orientation of the probe with respect to the mannequin. A synthetic US image of the needle is generated using the needle's tracking data. This image is fused with the ultrasound image to simulate the real needle inside the tissue. The needle position and orientation data are used in the iMSTK submodule to drive the needle-tissue interaction model. Deformation data from the needle-tissue interaction will then be used to deform the displayed ultrasound volume, showing real-time "interaction" with the image. The computed force data from the needle-tissue interaction model is transmitted back to the haptic device through iMSTK allowing the user to experience realistic tactile feedback. A face validation study of the trainer was conducted at a research hospital (Enquobahrie et al., 2019). The face validation conducted provided an understanding of the quantitative challenges and opportunities in virtual simulator for kidney biopsy.

4 Discussion

Surgical simulation is an important aspect of clinical training with a proven history of improving surgeon skill proficiency (Kanumuri et al., 2008) and patient outcomes (Zendejas et al., 2011). Recent progress has seen more virtual simulators incorporated into the training paradigm. These virtual simulators can provide increased training opportunities, particularly for low-resource communities that cannot support large simulation centers or see a low volume of specific procedures. However, development of this content can be slow and cost prohibitive. iMSTK provides a mechanism for easy development of this content. Particularly when coupled with game engines, such as Unity and Unreal Engine (Epic Games, Cary, NC), it can be used to prototype surgical scenes and procedures quickly and inexpensively.

iMSTK currently supports building simulations for a wide range of surgical scenarios. However, future work includes extending Unity support by building a library of drag and drop surgical tools to simplify the scene building process. We are reviewing alternative solver methods to improve performance during the constraint projection step, which will allow for larger more complex surgical scenes. These new solvers may also increase the overall stability and robustness of the physics simulations. In addition to this, we plan to add more strain energy-based constraints to simulate a wider range of materials with both isotropic and anisotropic material properties, which will increase the realism of simulations involving muscles and aligned fascia. Also, while iMSTK has preliminary support for cutting, we are reviewing novel methods to improve the fidelity of our cutting simulation (He et al., 2022; Kamarianakis et al., 2022).

The iMSTK community includes medical simulation builders, software developers, and haptic users. As an open source toolkit, iMSTK is free to use for all teams, including academic and government researchers and commercial users. To encourage adoption and use, documentation is included via the repository and website and testing is in place to ensure quality. Multiple examples are included in the code repository to demonstrate basic use and serve as building blocks for more complex simulations. A Unity Asset and supporting tutorial were developed to enable easy integration of iMSTK into simulation applications developed in the Unity game engine ("IMSTK | Physics | Unity Asset Store" n. d.).

It is also important to maintain active communication channels with the user community. There is a discourse (Kitware, 2022c), blog, wiki, issue tracker, and email address for communication with the iMSTK team. Outside users have contributed to iMSTK through the submission of pull requests. The code was reviewed by the iMSTK team, feedback was provided, and ultimately the pull request was accepted. For help with using and integrating iMSTK into surgical simulation applications, submitting pull requests, and developing models for iMSTK, please contact the iMSTK team on discourse or via email (kitware@kitware.com).

All software is freely available for download and can be used free of charge for commercial and research applications under the Apache 2.0 license. More information can be found at https://www.imstk.org/ and the source code is available at https://gitlab.kitware.com/iMSTK/iMSTK. The iMSTK Unity Asset source code is also available for download and can be used free of charge for commercial and research applications. The source code is available at https://gitlab.kitware.com/iMSTK/imstk-unity and the asset can be downloaded from the Unity Asset store (IMSTK, 2022).

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author. Source code—https://gitlab.kitware.com/iMSTK.

Author contributions

JM, HS, SJ, AB, and RC—These authors are contributors to the iMSTK platform and contributed to the paper. GS, AE, and BP—These authors contributed case study writing to the paper and work on the development of the individual simulators cited in this paper. All authors contributed to the article and approved the submitted version.

Funding

We would like to acknowledge our funding sources for the development of iMSTK: National Institutes of Health Grant Numbers R44EB019802, R43DK115332, R44DK115332, R43DE027595, R44DE027595, R44AR075481-03, R01EB02547, and R01EB031808.

Acknowledgments

We would like to thank the many contributors to the iMSTK software platform, including Sreekanth Arikatla, Johan Andruejol, Alexis Girault, and Samantha Horvath. We would also like to thank our collaborators on our case studies, including Lora Carvuoto at the University of Buffalo, Kevin Cleary at the Children's National Medical Center.

Conflict of interest

Authors JM, HS, SJ, AE, BP, AW, AB, and RC were employed by the company Kitware Inc.

The remaining author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

3dsystems, 2022 3dsystems (2022). OpenHaptics developer software. https://www.3dsystems.com/haptics-devices/openhaptics.

Arnold, J., and Diaz, M. C. G. (2016). Simulation training for primary caregivers in the neonatal intensive care unit. *Seminars Perinatology* 40 (7), 466–472. doi:10.1053/J.SEMPERI.2016.08.007

Arnold, Jennifer L. (2017). Simulation for pediatric radiology. *J. Radiology Nurs.* 36 (1), 1–2. doi:10.1016/J.JRADNU.2016.12.001

Bansal, Gaurang, Rajgopal, Karthik, Chamola, Vinay, Xiong, Zehui, and Niyato, Dusit (2022). Healthcare in metaverse: A survey on current metaverse applications in healthcare. *IEEE Access* 10, 119914–119946. doi:10.1109/ACCESS.2022.3219845

Bauman, E. B. (2013). *Game-based teaching and simulation in nursing and Health care - eric B. Bauman, PhD*. New York, NY, USA: Springer Publishing Company, LLC.

Bender, Jan, Koschier, Dan, Charrier, Patrick, and Weber, Daniel (2014). Position-based simulation of continuous materials. https://animation.rwth-aachen.de/media/papers/2014-CAG-PBER.pdf.

Bousseau, A., Gutierrez Tutorial, D., Bender, J., Müller, M., Macklin, M., Bender, J., et al. (2017). A survey on position based dynamics. www.interactive-graphics.de.

Bray, Aaron, Webb, Jeffrey B., Enquobahrie, Andinet, Vicory, Jared, Heneghan, Jerry, Hubal, Robert, et al. (2019). Pulse Physiology engine: An open-source software platform for computational modeling of human medical simulation. SN Compr. Clin. Med. *March* 1, 1–16. doi:10.1007/s42399-019-00053-w

Brown, R., McIlwain, S., and Willson, B. "Enhancing combat medic training with 3D virtual environments," in Proceedings of the 2016 IEEE International Conference on Serious Games and Applications for Health (SeGAH), Orlando, FL, USA, May 2016.

Caehealthcare (2017). CAE healthcare. https://caehealthcare.com/patient-simulation.

Clipp, Rachel B., Bray, Aaron, Webb, Jeffrey B., Kondrats, Janis, and Couperus, Kyle (2019). Combining computational Physiology with virtual reality for an immersive and effective trauma simulator. *Mil. Health Res. Symposium*.

Colgate, J. E., Stanley, M. C., and Brown, J. M. "Issues in the haptic display of tool use," in Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots, Pittsburgh, PA, USA, August 1995, 140–145. doi:10.1109/IROS.1995.5258753

Committee on Pediatric Emergency Medicine (2007). Patient safety in the pediatric emergency care setting. *Pediatrics* 120 (6).

Couperus, Kyle, Scott, Young, Walsh, Ryan, Kang, Christopher, Skinner, Carl, Essendrop, Robyn, et al. (2020). Immersive virtual reality medical simulation: Autonomous trauma training simulator. *Cureus* 12 (5). doi:10.7759/CUREUS.8062

Educationxr (2022). EducationXR. https://educationxr.com/.

Enquobahrie, Andinet, Horvath, Sam, Arikatla, Sreekanth, Rosenberg, Avi, Cleary, Kevin, and Sharma, Karun (2019). Development and face validation of ultrasound-guided renal biopsy virtual trainer. *Healthc. Technol. Lett.* 6 (6), 210. doi:10.1049/HTL.2019.0081

Ericson, Christer (2005). *Real-time collision detection*. Boca Raton, FL, USA: Taylor & Francis Group.

Falcone, Richard A., Daugherty, Margot, Lynn, Schweer, Patterson, Mary, Brown, Rebeccah L., and Garcia, Victor F. (2008). Multidisciplinary pediatric trauma team training using high-fidelity trauma simulation. *J. Pediatr. Surg.* 43 (6), 1065–1071. doi:10.1016/j.jpedsurg.2008.02.033

Feiger, B., Wilson, A., Arikatla, S., and Rachel, B. (2020). Simulation of arterial hemorrhage: Two-way coupling of a local high fidelity SPH model with a full body lumped parameter Physiology model - NASA/ADS. *APS Div. Fluid Dyn.* 65.

Foronda, Cynthia L., Fernandez-Burgos, Margo, Nadeau, Catherine, Kelley, Courtney N., and Henry, Myrthle N. (2020). Virtual simulation in nursing education: A systematic review spanning 1996 to 2018. *Simul. Healthc. J. Soc. Simul. Healthc.* 15 (1), 46–54. doi:10.1097/SIH.0000000000000411

Frantzidesand Constantine, T. (2007). Prospective randomized controlled trial of laparoscopic trainers for basic laparoscopic skills acquisition. *Randomized Control. Trial* 21. doi:10.1007/s00464-006-0149-6

Fu, Chung-Pei, Yeh, Jiann-Horng, Su, Chia-Ting, Liu, Chien-Hsiou, Chang, Wan-Ying, Chen, Yu-Lan, et al. (2017). Using children as standardized patients in OSCE in pediatric occupational therapy. *Med. Teach.* 39 (8), 851–858. doi:10.1080/0142159X.2017.1320540

Girault, Alexis, Bray, Aaron, and Clipp, Rachel (2019). Pulse is now available on the unity asset store - kitware blog. https://blog.kitware.com/pulse-is-now-available-on-the-unity-asset-store/.

Google (2022). GoogleTest user's guide GoogleTest. https://google.github.io/googletest/.

Gruen, Russell L., Jurkovich, Gregory J., Mcintyre, Lisa K., HughFoy, M., and Maier, Ronald V. (2006). Patterns of errors contributing to trauma mortality lessons learned from 2594 deaths. *Ann. Surg.* 244, 371–380. doi:10.1097/01.sla.0000234655.83517.56

Hcup-Us (2022). HCUP-US NIS overview. https://www.hcup-us.ahrq.gov/nisoverview.jsp.

He, Sirui, Qian, Yinling, Zhu, Xin, Liao, Xiangyun, Heng, Pheng-Ann, Feng, Ziliang, et al. (2022). Versatile cutting fracture evolution modeling for deformable object cutting simulation. *Comput. Methods Programs Biomed.* 219, 106749. doi:10.1016/j.cmpb.2022.106749

Hoopes, Sarah, Pham, Truce, Lindo, Fiona M., and Antosh, Danielle D. (2020). Home surgical skill training resources for obstetrics and gynecology trainees during a pandemic. *Obstetrics Gynecol.* 136 (1), 56–64. doi:10.1097/AOG.0000000000003931

Hugh, Thomas B. (2002). New strategies to prevent laparoscopic bile duct injury—surgeons can learn from pilots. *Surgery* 132 (5), 826–835. doi:10.1067/MSY.2002.127681

Hyun, Jong Jin, and YoungBak, Tae (2011). Clinical significance of hiatal hernia. *Gut Liver* 5 (3), 267–277. doi:10.5009/gnl.2011.5.3.267

Imstk "IMSTK physics unity asset store."2022. https://assetstore.unity.com/packages/tools/physics/imstk-225525.

Issenberg, S. B., Mcgaghie, W. C., EmilPetrusa, R., Lee Gordon, D., and RossScalese, J. (2005). Features and uses of high-fidelity medical simulations that lead to effective learning: A beme systematic review. *Med. Teach.* 27 (1), 10–28. doi:10.1080/01421590500046924

James, John T. (2013). A new, evidence-based estimate of patient harms associated with hospital care. *J. Patient Saf.* 9 (3), 122–128. doi:10.1097/PTS.0b013e3182948a69

Kamarianakis, Manos, Protopsaltis, Antonis, Angelis, Dimitris, Tamiolakis, Michail, and George, Papagiannakis (2022). Progressive tearing and cutting of soft-bodies in high-performance virtual reality. https://arxiv.org/abs/2209.08531.

Kanumuri, P., Ganai, S., EyadWohaibi, M., Bush, R. W., Grow, D. R., and E Seymour, N. (2008). Virtual reality and computer-enhanced training devices equally improve laparoscopic surgical skill in novices. *JSLS* 12.

Katić, Darko, Wekerle, Anna Laura, Görtler, Jochen, Patrick SpenglerBodenstedt, Sebastian, Röhl, Sebastian, et al. (2013). Context-aware augmented reality in laparoscopic surgery. *Comput. Med. Imaging Graph.* 37 (2), 174–182. doi:10.1016/J.COMPMEDIMAG.2013.03.003

Kitware (2022a). Kitware supports InciteVR with their first product release featuring immersive learning. https://www.kitware.com/kitware-supports-incitevr-with-their-first-product-release-featuring-immersive-learning/.

Kitware (2022b). Latest interactive medical simulation toolkit (imstk) topics - kitware. https://discourse.kitware.com/c/imstk/9.

Kitware (2022c). Lumeto uses Kitware's Pulse Physiology engine to incorporate real-time patient feedback into InvolveXR healthcare. https://www.kitware.com/lumeto-uses-kitwares-pulse-physiology-engine-to-incorporate-real-time-patient-feedback-into-involvexr-healthcare/.

Laerdal (2017a). SimMan® products & pricing. http://www.laerdal.com/us/doc/86/SimMan.

Laerdal (2017b). Welcome to laerdal medical – helping save lives. http://www.laerdal.com/us/.

Laskin, Daniel M. (2016). Oral and maxillofacial surgery: The mystery behind the history. *J. Oral Maxillofac. Surg. Med. Pathology* 28 (2), 101–104. doi:10.1016/J.AJOMS.2015.11.001

Macklin, Miles, Müller, Matthias, and Chentanez, Nuttapong "Xpbd: Position-Based simulation of compliant constrained dynamics," in Proceedings of the Motion in Games 2016: 9th International Conference on Motion in Games MIG, Burlingame California, October 2016. doi:10.1145/2994258.2994272

MakaryMartin, A., and Daniel, M. (2016). Medical error—The third leading cause of death in the US. *BMJ* 353. doi:10.1136/bmj.i2139

Mannella, Paolo, Malacarne, Elisa, Giannini, Andrea, Russo, Eleonora, Caretto, Marta, Papini, Francesca, et al. (2019). Simulation as tool for evaluating and improving technical skills in laparoscopic gynecological surgery. *BMC Surg.* 19 (1). doi:10.1186/S12893-019-0610-9

Müller, Matthias, Bruno, Heidelberger, Marcus, Hennix, and Ratcliff, John (2007). Position based dynamics. *J. Vis. Commun. Image Represent.* 18 (2), 109–118. doi:10.1016/J.JVCIR.2007.01.005

Müller, Matthias, Macklin, Miles, Chentanez, Nuttapong, Jeschke, Stefan, and Kim, Tae-Yong (2020). Detailed rigid body simulation with extended position based dynamics. *Comput. Graph. Form.* 39. doi:10.1111/cgf.14105

Neo, Eu Ling, Zingg, Urs, Devitt, Peter G., Jamieson, Glyn G., and Watson, David I. (2011). Learning curve for laparoscopic repair of very large hiatal hernia. *Surg. Endosc.* 25, 1775–1782.

Niddk (2022). Kidney disease statistics for the United States NIDDK. https://www.niddk.nih.gov/health-information/health-statistics/kidney-disease.

Nishisaki, Akira, Keren, Ron, and Nadkarni, Vinay (2007). Does simulation improve patient safety?: Self-efficacy, competence, operational performance, and patient safety. *Anesthesiol. Clin.* 25 (2), 225–236. doi:10.1016/j.anclin.2007.03.009

Nishisaki, Akira, Nguyen, Joan, Colborn, Shawn, Watson, Christine, Dana, Niles, Hales, Roberta, et al. (2011). Evaluation of multidisciplinary simulation training on clinical performance and team behavior during tracheal intubation procedures in a pediatric intensive care unit. *Pediatr. Crit. Care Med. A J. Soc. Crit. Care Med. World Fed. Pediatr. Intensive Crit. Care Soc.* 12 (5), 406–414. doi:10.1097/PCC.0b013e3181f52b2f

Nvidia (2022). NVIDIA PhysX SDK 4.1 documentation — NVIDIA PhysX SDK 4.1 documentation. https://gameworksdocs.nvidia.com/PhysX/4.1/documentation/physxguide/Index.html.

Obi (2022). Obi - unified particle physics for unity 3D. http://obi.virtualmethodstudio.com/.

Opensurgsim (2022). OpenSurgSim. http://www.opensurgsim.org/.

Parham, Groesbeck, Bing, Eric G., Cuevas, Anthony, Fisher, Boris, Skinner, Jonathan, Mwanahamuntu, Mulindi, et al. (2019). Creating a low-cost virtual reality surgical simulation to increase surgical oncology capacity and capability. *Ecancermedicalscience* 13. doi:10.3332/ECANCER.2019.910

Patel, Hiten R. H., and Patel, Bijen P. (2014). Virtual reality surgical simulation in training. *Expert Rev. Anticancer Ther.* 12 (4), 417–420. doi:10.1586/ERA.12.23

Plustoolkit (2022). Plus toolkit. https://plustoolkit.github.io/."

Punjabi, Anil P., and Haug, Richard H. (1990). The development of the dual-degree controversy in oral and maxillofacial surgery. *J. Oral Maxillofac. Surg.* 48, 612–616.

Qi, Di, Panneerselvam, Karthikeyan, Ahn, Woojin, Arikatla, Venkata, Enquobahrie, Andinet, and De, Suvranu (2017). Virtual interactive suturing for the fundamentals of laparoscopic surgery (FLS). *J. Biomed. Inf.* 75, 48–62. doi:10.1016/J.JBI.2017.09.010

Sakpal, Sujit Vijay, Singh Bindra, Supreet, and Chamberlain, Ronald S. (2010). Laparoscopic cholecystectomy conversion rates two decades later. *JSLS J. Soc. Laparoendosc. Surg.* 14 (4), 476. doi:10.4293/108680810X12924466007926

Schijven, M., and Jakimowicz, J. (2003). Virtual reality surgical laparoscopic simulators how to choose. *Surg. Endosc.* 17. doi:10.1007/s00464-003-9052-6

Schroeder, Will, Martin, Ken, and Lorenson, Bill (2006). *The visualization toolkit*. New York, NY, USA: Kitware.

Shamiyeh, A., Shamiyeh, A., Wayand, W., and Wayand, W. (2004). Laparoscopic cholecystectomy: Early and late complications and their treatment. *Langenbecks Arch. Surg.* 389, 164–171. doi:10.1007/s00423-004-0470-2

Sims, Edward M. (2007). Reusable, lifelike virtual humans for mentoring and role-playing. *Comput. Educ.* 49 (1), 75–92. doi:10.1016/J.COMPEDU.2005.06.006

Sims, E., Silverglate, D., Hanahel, D., Sweet, R., Reihsen, T., and Norfleet, J. "A modular architecture for blending virtual and manikin-based medical simulations," in Proceedings of the Interservice/Industry Training Simulation and Education Conference (I/ITSEC), Orlando, FL, USA, November 2016.

Sims (2015). Stat! ICU - a serious game for teaching critical care skills. *Soc. Simul. Healthc. Serious Games Arcade*.

Slicer (2021). 3D slicer. https://www.slicer.org/.

Sofa (2022). Sofa. https://www.sofa-framework.org/.

Stefanidis, D., Ryan, B., Stefanidis, D., JamesKorndorffer, R., Sierra, R., Touchard, C., et al. (2005). Skill retention following proficiency-based laparoscopic simulator training. *Surgery*, 165–170. doi:10.1016/j.surg.2005.06.002

Stefanidis, Dimitrios, Acker, Christina, and Heniford, Todd B. (2008). Proficiency-based laparoscopic simulator training leads to improved operating room skill that is resistant to decay. *Surg. Innov.* 15 (1), 69–73. doi:10.1177/1553350608316683

Sureka, Binit, and Mukund, Amar (2017). Review of imaging in post-laparoscopy cholecystectomy complications. *Indian J. Radiol. Imaging* 27. doi:10.4103/ijri.IJRI_489_16

Syndaver (2017). SynDaver Labs. http://syndaver.com/.

Taylor, Glenn, Deschamps, Anthony, Tanaka, Alyssa, Nicholson, Denise, Bruder, Gerd, Welch, Gregory, et al. "Augmented reality for tactical combat casualty care training," in Proceedings of the 12th International Conference, AC 2018, Held as Part of HCI International, Las Vegas, NV, USA, July 2019. doi:10.1007/978-3-319-91467-1_19

Unity (2022). Unity real-time development platform | 3D, 2D VR & AR engine. https://unity.com/.

Unrealengine (2022). The most powerful real-time 3D creation tool - unreal engine. https://www.unrealengine.com/en-US/.

Vrpn (2022). Vrpn - virtual reality peripheral network. https://vrpn.github.io/.

Willie, Chelsea, Chen, Fei, Joyner, Benny L., and Blasius, Kimberly (2016). "Using high-fidelity simulation for critical event training," in *Medical education* (Hoboken, New Jersey, United States: Blackwell Publishing Ltd). doi:10.1111/medu.13179

Yang, Dawei, Zhou, Jian, Chen, Rongchang, Song, Yuanlin, Song, Zhenju, Zhang, Xiaoju, et al. (2022a). Expert consensus on the metaverse in medicine. *Clin. EHealth* 5, 1–9. doi:10.1016/j.ceh.2022.02.001

Yang, Dawei, Zhou, Jian, Song, Yuanlin, Sun, Mengting, and Bai, Chunxue (2022b). Metaverse in medicine. *Clin. EHealth* 5, 39–43. doi:10.1016/j.ceh.2022.04.002

Yiannakopoulou, Eugenia, Nikiteas, Nikolaos, Perrea, Despina, and Tsigris, Christos (2015). Virtual reality simulators and training in laparoscopic surgery. *Int. J. Surg.* 13, 60–64. doi:10.1016/J.IJSU.2014.11.014

Zendejas, Benjamin, Cook, David A., Bingener, Juliane, Huebner, Marianne, Dunn, William F., Sarr, Michael G., et al. (2011). Simulation-based mastery learning improves patient outcomes in laparoscopic inguinal hernia repair: A randomized controlled trial. *Ann. Surg.* 254 (3), 502–511. doi:10.1097/SLA.0B013E31822C6994

Zikas, Paul, Protopsaltis, Antonis, Lydatakis, Nick, Kentros, Mike, Geronikolakis, Stratos, Kateros, Steve, et al. (2023). Mages 4.0: Accelerating the world's transition to VR training and democratizing the authoring of the medical metaverse. *IEEE Comput. Graph. Appl.* 43 (2), 43–56. doi:10.1109/MCG.2023.3242686