A Surface-based Volume Haptics Approach and Applications in Surgical Simulation

BY

Jie Jiang
B.S., Electronics Engineering, Beijing Institute of Technology, China, 2010

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Industrial Engineering and Operations Research
in the Graduate College of the
University of Illinois at Chicago, 2017

Chicago, Illinois

Defense Committee:

Prashant Banerjee,	Chair and Advisor
Michael Scott	
Cristian Luciano	
Ben Roitberg,	Case Western Reserve University
Silvio Rizzi,	Argonne National Laboratory

To my family,

for their unconditional love and support.

ACKNOWLEDGEMENTS

I want to thank my committee members. Prof. Prashant Banerjee supported my research for the past five years and granted me the freedom pursuing various opportunities without objection. Dr. Silvio Rizzi is a mentor as well as a sincere friend. He shared his wisdom of life and encouraged me to reach beyond my grasp. Prof. Cristian Luciano guided me through my first couple years in my graduate program, since then he has always been a great consultant. Without contributions and feedback from Dr. Ben Roitberg, I would never have finished this work to the degree as it is right now. Last but not least, Dr. Michael Scott, as the DGS at the Department of MIE, is supportive and considerate. I appreciate all the time you and the staff spent on granting my petitions so that I can participate in fun projects outside of my program. I am very lucky to have the privilege to work with many great colleagues. Dr. Jia Luo and I have shared 11 years of friendship and 5 years of collaboration. Many collaborators from ImmersiveTouch, Inc have contribution to this project. Special thanks to the community of Electronic Visualization Laboratory for the help they generously provided. Lastly, I appreciate the guidance from my external mentors during my summer internships, including Prof. Joseph Insley from Argonne National Laboratory, Dr. Thomas Fogal, Dr. Peter Messmer and Cliff Woolley from NVIDIA.

JJ

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF ABBREVIATIONS

3D Three Dimension.

API Application Programming Interface.

AR Augmented Reality.

CG Computer Graphics.

CT Computed Tomography.

DICOM Digital Imaging and Communications in Medicine.

DOF Degrees of Freedom.

FPS Frames per Second.

GPGPU General Purpose Graphics Processing Unit.

GPU Graphics Processing Unit.

GUI Graphical User Interface.

HIP Haptic Interface Point.

HMD Head Mounted Display.

MIS Minimally Invasive Surgery.

MRI Magnetic Resonance Imaging.

MSD Mass Spring Damper.

NSN Naive Surface Net.

OR Operating Room.

OS Operating System.

LIST OF FIGURES (Continued)

FIGURE **PAGE**

RMIS Robot-assisted Minimally Invasive Surgery.

SDK Software Development Kit.

STL Stereolithography.

VCS Volume Coordinate System.

VR Virtual Reality.

WCS World Coordinate System.

SUMMARY

Volume haptics algorithms are widely studied to provide multi-sensory feedback during data exploration. Efficient haptics rendering of consistent surface topology from a structured volumetric data set is desired by applications but has not been thoroughly explored. We presented a proxy-based volume haptics approach inspired by a fast voxel traversal algorithm that delivers efficient and robust surface representation. The technique has the flexibility to work either as an independent volume haptics rendering module or as an extension to incorporate volume haptics into polygonal-based haptics modules. The volume haptics approach is efficient at handling large volume data set and it scales decently with voxel density. This technique enables us to develop virtual reality based surgical simulations with comprehensive procedures. With the reliable surface-based volume haptics module, we developed cranial and spinal surgical simulations involving bone removal and extensive contact interactions. Our applications leverages the advantages of virtual reality based simulators over traditional training methods.

CHAPTER 1

INTRODUCTION

Existing volume haptics algorithms render the internal properties and the surface representation of a volumetric data set. Due to the intermediate surface presentation and the ray sampling scheme, they inevitably compromise on maintaining a consistent surface representation of the volume object.

For skull base and spinal surgical simulation, haptic interaction should be restricted on its surface and penetration should be avoided for realistic representation. Unfortunately, these desired features are not available through existing volume haptics algorithms[1–3].

The lacking of a robust surface-based volume haptics algorithm prevents applications to directly utilize the abundant patient data collected during clinical operations. It motivates the exploration of a new surface-based volume haptics algorithm.

We proposed an approach that combines a discrete space division technique and a fast ray traversal algorithm. The technique traverses all voxels on a line segment without extensive sampling and interpolation. It provides a precise and efficient way to interpret the geometric topology of the volume data. Based on this technique, we developed a volume haptics algorithm that provides robust and efficient surface-feature rendering.

1.1 Thesis Outline

The outline of this dissertation is briefly presented as follows. For the rest of this chapter we discuss the fundamental problems our research focuses on. In Chapter 2, simulation in surgical training and related research are discussed in detail. Chapter 3 presented surface-based volume haptics technique. Chapter 4 covers details about the implementation and integration of volume haptics module to frameworks with different haptic pipeline. Applications of the volume haptics module in collaboration projects are presented in Chapter 5 and Chapter 6. Lastly, Chapter 7 summarizes the contribution and our conclusion of the research.

1.2 Medical Imaging

Realistic anatomical models are very important for medical simulations. Medical imaging produces a visual representation of patient anatomies. It is a prevalent tool used during clinical diagnosis and surgical intervention.

Medical imaging produces volume data to recreate anatomical models. Its popularity in clinics provides abundant data for the construction of diverse human anatomy and various pathology to cover more training cases in simulations.

1.2.1 X-Ray and CT/MRI

Plain radiography and fluoroscopy create two dimensional images. These techniques relies on variances of attenuation factors of different structures. The intensity of radiation on the receiving end could reflect the accumulated attenuation along the path and represent the underlying anatomy. Fluoroscopy could monitor moving body structures by

Figure 1: Fluoroscopy machine

C-arm X-ray machine used for intraoperative fluoroscopic imaging. Intraoperative fluoroscopy provides high-resolution images to aid real-time decision making.

generating real-time X-Ray images. Fluoroscopy is used during diagnostic procedures, image-guided surgeries and minimal invasive procedures.

Three dimensional images could be retrieved through technologies like computed tomography (CT) or magnetic resonance imaging (MRI).

CT and MRI use different imaging techniques. CT use a series of angled X-Ray scan to reconstruct the cross-section, while MRI relies on a external magnetic field to excite the nuclear spine energy transition of hydrogen atoms. MRI monitors the magnetic gradient field to localize the signal and compute the internal distribution. Although MRI does not use X-Ray and hence it is free of hazards from excessive radiation. MRI does require longer

imaging process and it is prone to problems caused by magnetic sensitive particles. CT and MRI are complementary technology and each has its own application.

CT/MRI produce similar result as a stack of two dimensional images that can be treated as a structured volumetric data set. Many clinical diagnosis relies on examining the internal anatomies through the volume data.

1.2.2 Volumetric Data Set

Volumetric data set is a 3D data array accompanied with meta information about dimensions, size, spacing and data type that is necessary to reconstruct the object from its volume representation. Medical imaging is stored in different formats[4]. They differ in aspects like compatible data format and embedded communication protocol. Those features concerns the consistency and accessibility of data across facilities. The volumetric data set shares the same form as a plain data array even for different formats. Hence it is possible to apply general 3D reconstruction techniques to visualize and analyze the medical imaging.

Reconstruction techniques could be classified by the dimension of their results. Multiplanar reconstruction samples the volume data on an intersecting plane and presents it as a 2D image. Traditional cross-section slices have been widely used in clinics. It provides fine control and quantitative measurement used for surgical planning and assessment. Modern software allows reconstruction on non-orthogonal intersection to compensate curving vessels and surfaces, which could not be properly visualized on cross-section slices. 3D rendering represents the entire volume data instead of a single slice at a time.

(a)

(b)

Figure 2: CT and MRI machine

Medical imaging equipment: (a) CT machine reconstruct cross section imaging by calculating the observed attenuation X-ray passed through the object in the center; (b) Medical MRI scanner applies oscillating magnetic field to the patient and monitors signal emitted from excited hydrogen atoms to reconstruct internal structures.

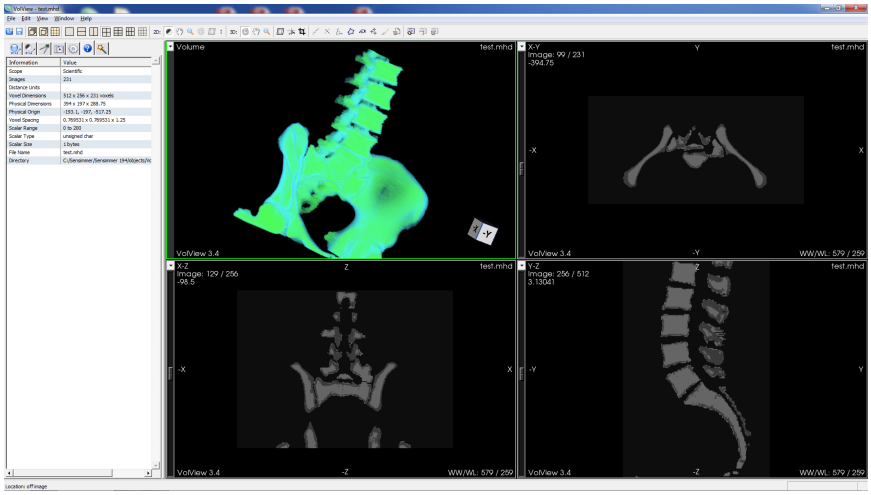


Figure 3: Volview

Open-source volume visualization system VolView provides direct volume rendering as well as multiplanar.

Surface rendering technique extracts isosurface from volume data based on preset threshold and then renders the 3D polygonal model. Direct volume rendering renders the volume data using transparency and color mapping. Unlike surface rendering, volume rendering is capable of depicting interior features inside the volume object.

In reality it is common for software to combine multiple reconstruction approaches as Figure 3. Domain expertise-based image segmentation is often used to identify anatomies that is hard to register solely by images. Research in semi-automated segmentation are used to aid this labor-intensive process[5]. Enhanced reconstruction has applications in navigation system and image guided surgery[6].

1.3 Haptics

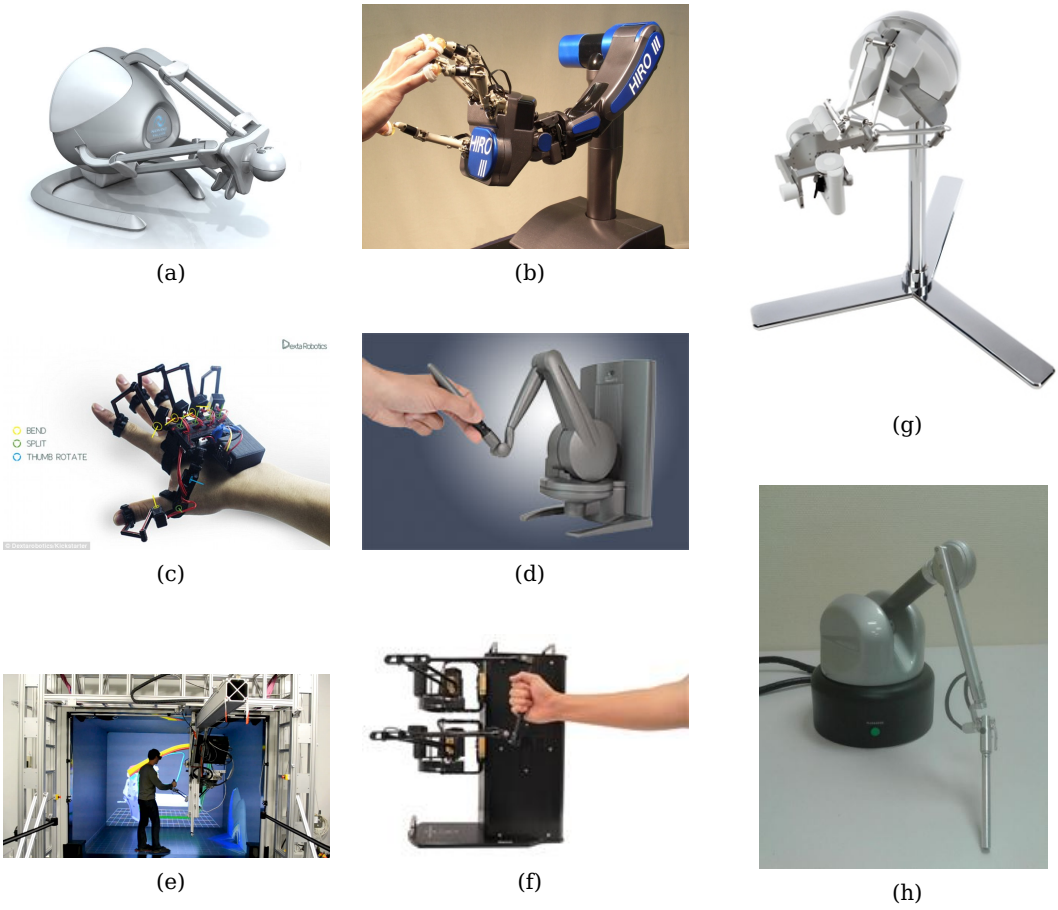
Haptic devices stimulate haptic perception to creates the sense of touch by applying vibration, force and motion. Haptic technology has applications in video games, medicine, robotics and manufacturing. Haptic cue is one of the sensory cues which contribute to perception. It is a subject of research in cognitive science and studies have explored the impact of haptic cue in knowledge acquisition.

1.3.1 Hardware

Many researchers have worked on systems with haptic feedback[7]. Earlier research is motivated by application in teleoperation systems seeking successor to remote handling system used by industrial and military projects operating in high risk environment. Through years of research, haptic devices have evolved from stationary manipulator arms and room-sized exoskeleton systems to wearable gloves and high-precision desktop devices (see Figure 4).

Virtual reality (VR) has extended the application of teleoperation systems. Study in communication[15] defines virtual reality independent from specific technology utilized to create the environment and it instead claims VR as a simulated environment that provides telepresence. The study discusses determinants of telepresent which are a combination of vividness and interactivity.

The addition of haptic feedback improves the fidelity of the simulated environment. Meanwhile, evolving haptic device provides ergonomic user interface to perform complex tasks and enhance interactivity.



(a) (b) (g) (c) (d) (e) (f) (h)

Figure 4: Haptic Devices

(a) Novint Falcon used in application of robot manipulator[8]. (b) Finger tip haptic interface built by Endo[9]. (c) Glove-like exoskeleton system developed by Gu[10]. (d) Desktop haptic stylus provided by Geomagic Touch X (formerly Sensable Phantom[11]). (g) sigma.7 by force dimension used for MiroSurge[12]. (f) HD^2 High Definition Haptic Device by QUANSER[13]. (h) Virtuose™ 3D desktop and (e) Scale™ 1 both from HAPTION[14]

1.3.2 Software

Haptics rendering is a complex and multidisciplinary field that has been studied in previous research[16,17].

A constraint-based proxy method for haptics rendering for polygonal models was proposed by Zilles[18]. The approach utilizes polygonal model that is also used for graphics rendering. The consistency between haptics and graphics combines two sensory cues without overhead for synchronization.

The method has two components illustrated in Figure 5. The constraint model handles the collision detection and restrict the proxy motion in the virtual environment. The proxy model handles the force calculation. The calculated force is sent to haptic device for force rendering.

The proxy method traces a proxy that follows haptic interface point (HIP). HIP is the mapped interaction point of the haptic device that moves freely in the virtual scene. While the proxy has the constraint that it cannot enter any virtual model. The proxy is connected to HIP with a spring and damper model that generates the force to pull HIP towards the proxy. Spring and damper model maintains a proxy that follows the trace of HIP and it generates stable force that could be parameterized to reflect various haptic characteristics from soft tissues to hard bones.

The constraint model allows proxy to approximate the HIP in the virtual scene while satisfying the constraints that proxy shall not entering any virtual object. Within each haptics frame, constraint model shoots a ray from the previous frame proxy to current HIP.

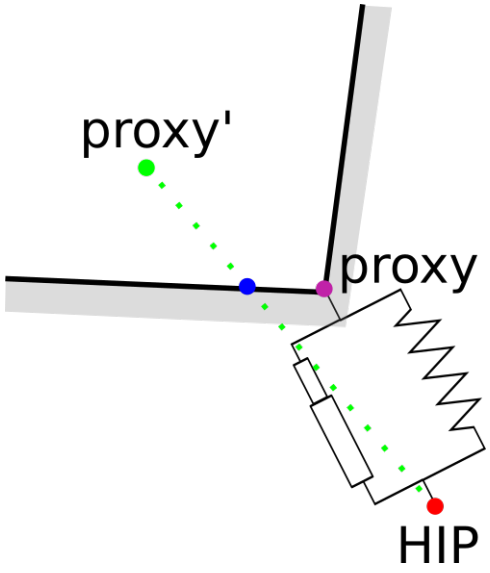


Figure 5: Proxy-based haptics

Constraint model update proxy location that it does not enter any geometry. Spring and damper model connecting proxy to haptic device could generate the force reflecting the collision detected in the virtual scene.

If no penetration or constraint condition detected, the algorithm simply updates current HIP as the current frame proxy. Otherwise, it uses iterative ray query to find relevant constraints and solves the candidate proxy following least energy model.

A constraint-based proxy method is currently the standard for haptics rendering approach and has inspired improvements and enhancements[2,19–21]. Geomagic provides OpenHaptics Toolkit[22] to enable developers to utilize Geomagic Touch and the toolkit uses proxy-based haptics rendering. The toolkit exposes low-level application programming interface (API) for application to send direct force parameters to the device. This en-

ables community software with their own haptics algorithm to integrate OpenHaptics and to provide compatibility to Geomagic hardware. Open source software has been developed by community and has supported the development of commercial products. H3DAPI[23] and CHAI3D[24] are two frameworks that have integrated haptics and computer graphics to enable fast development of haptics-enabled VR applications.

Polygonal model and constraint-based proxy method both focus on the surface feature of virtual model. Although some studies have put endeavors into expanding this to other data format, reliable surface-based representation is not yet available for volumetric data.

1.4 Scope of The Dissertation

The focus of this dissertation is to develop a robust surface-based volume haptics algorithm. Consistent haptic representation is important to achieve high fidelity in VR application. The absence of reliable surface-based volume haptics technique requires application to either compromise on haptics rendering or adopt preprocess on volume data. For medical applications, the drawbacks undermine the benefits of VR-based simulation for surgical training and pre-surgical planning. Our technique enables a reliable haptic representation in VR applications with patient specific model reconstructed from raw volumetric data set retrieved from commercial medical imaging equipments.

The proposed volume haptics algorithm is flexible to accommodate frameworks with diverse haptics pipeline. We integrate volume haptics into a legacy framework Sensimmer to improve volume haptics rendering quality. We build a craniotomy simulation mainly consists of volume objects for haptics rendering. Later we developed and wrapped volume

haptics into the Unity3D toolkit targeting VR-based surgical simulations. We implemented a procedure combining laminectomy and pedicle screw placement using the new toolkit.

Our technique efficiently provides a reliable surface representation directly from raw volumetric data sets. With reliable surface representation of volumetric data, we improved the user experience during sessions involving extensive interaction with complex surface on volumetric model. This research improves the user experience in VR-based surgical training. Moreover, it leads the research from VR-based simulation to surgical planning by eliminating labor-intensive preprocessing as well as the artifacts introduced by these procedures.

The algorithm is versatile and could adapt to applications with diverse pipeline. Our successful integration of volume haptics module validates the versatility of our approach. The flexibility assures the accessibility of this technique and strengthens the impact of our work.

CHAPTER 2

SIMULATIONS IN SURGICAL TRAINING

In surgical education efficient training has been receiving growing emphasis. Simulation offers the opportunity for residents to acquire technical skills and knowledge in a safe environment prior to operating on actual patient. Simulation eliminates pressure during training and reduces risks of patient cares. Research also indicates that simulation-based training outperforms problem-based learning in acquisition of some skills[25]. It makes simulation an indispensable part[26] for surgical training. Institutions have been providing curriculum involving considerable attention on simulations[27,28].

Surgical simulation has been used as influential means of medical education for a very long time and there are dedicated research on validating and improving the simulation-based medical programs[29]. Medical simulators generally falls into 4 basic types: physical, VR, web-based and hybrids. Physical simulator and VR simulator are mostly used in educational programs. Das[30] summarizes common simulation trainings used in teaching neurosurgery.

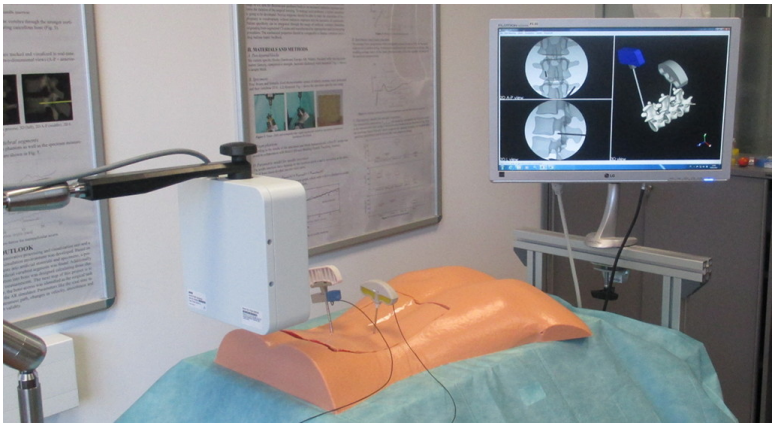
2.1 Physical Simulation

Physical simulators usually are cadaver models and manikins. The advantage of cadaver models is that they provide accurate tool manipulation and realistic tactile feedback. That makes it a preferred training method for a series of operations including skull

drilling, sawing and soft tissue manipulations. The disadvantages of cadaver training are their limited supplies and high expenses for procurement and maintenance. Moreover, cadavers can not replicate pathological conditions such as tumor or aneurysms. The other physical simulators are manikins. Manikins have advantages including realistic tactile feedback, allowing usage of real surgical instruments during training and capability of replicate pathological conditions. Manikins are commonly used for early medical students for trainings on cranial and spinal procedures[31,32]. The drawback of manikins are their limited cases that the surgeons cannot add new cases to their simulation unless provided different manikin models. Abe[33] combines plastic skull models fabricated by a stereolithography system from real patient skull data and hand-made tumors, major vessels and nerves for surgical planning and education of residents. However the time and cost of making each model greatly restricted its accessibility. Automatic assessment is lacking so that access to simulator is limited and dependent on the availability of instructors and operators. Sarle[34] designed training drills for surgical robotics[35,36] that rises with the minimally invasive surgery (MIS). Latest physical models also integrated computerized component for navigation and visualization while relies on physical models for tactile feedback and tool manipulation[37].

Overall, physical simulators still have the best fidelity of tool handling. Physical simulators have been used for neurosurgeries (see Figure 6) in the spine discipline[40–42]. However high-fidelity models that could be used for entire operation can only be reused a

(a)



(b)

Figure 6: Physical lumbar puncture simulators

Simulators: (a) commercially available physical lumbar puncture kit (Picture from Limbs & Things[38]); (b) augmented reality simulator used by Fuerst[39].

few times and it comes with high ongoing maintenance cost. Disadvantages listed above limit the accessibility of physical simulators.

2.2 Virtual Reality based Simulation

Previous computerized simulators focus on visualization of anatomy[43] and surgical procedures[44] due to the limited computational power of available hardware. With the improvement of computer graphics (CG) and the introduction of haptic devices[11,45,46], VR-based simulators have made huge improvement and have been adopted by training and educational programs. Established evolution of surgical simulators proves the potential of VR-based simulators for neurosurgical training[47]. Commercially available products[48,49] have been promoting their VR-based simulators for different surgical procedures.

The advantages of VR-based simulators are their realistic representation of living tissues and pathological conditions, unlimited practice and unbiased performance evaluation. VR-based simulators also allows very subtle and accurate measurement of trainee interaction than any physical simulator could offer.

Computerized simulators are used for visualization of anatomical structures and for training of intracranial procedures[50,51]. Spinal surgical simulators have very few devices available. There has been a paucity of commercial models aside from some basic physical simulators commonly used for educational curricula[40]. High fidelity simulators are mostly built for microsurgery and endoscopic surgery[52,53]. Other available simulators lacks essential fidelity for training purposes[54,55].

The feasibility and positive impact of neurosurgery simulations on residency training have been validated in Gasco[56]. The study suggests each simulation form has a different role in training and both should be considered in the development of an educational simulation program.

2.2.1 Haptic Feedback for Surgical Training

Haptic perception is crucial in traditional surgery. Surgeon relies on tactile sensory to differentiate anatomical structures and to evaluate the safety of their manipulation in order to avoid trauma to delicate patient organs.

Study suggests that inadequate tactile feedback is hypothesized to be one of the drawbacks for modern surgical techniques as laparoscopic surgery and robot-assisted minimally invasive surgery (RMIS)[57,58]. Research has been trying to introduce haptic sensory through creatively designed instruments for clinical usage[59]. Unfortunately for reasons of cost and feasibility, adequate haptic feedback for general interaction has not been established. Systems relies on sensory substitution. Visual and auditory cues are provided to enhance human perception where haptic cue is missing[60,61].

VR-based surgical simulation has a similar environment as RMIS. Although the risk caused by the lack of haptic feedback is not a real concern as it is during a procedure operating on real patient. Tactile feedback plays important role during interaction in VR environment.

VR environments have been used in training for medical education[62]. Navigation and interaction during complex surgical procedures in a VR environment using non-conventional

(a)

(b)

Figure 7: VR-based Simulators

(a) ImmersiveTouch open surgical simulator (Picture from[49]); (b) NeuroTouch cranial surgical simulator (Picture from[48]).

instruments are still challenging to medical personnel. The steep learning curve adds overhead to educational programs incorporating VR-based simulators. Meanwhile, the acquired skill on simulator operating has little contribution to teaching and surgical training. Research shows positive influence of haptic feedback in minimally invasive surgery (MIS) and suggests the consensus of tactile sensory on improving performance at early phase of simulation in VR environment[63]. A haptics-enabled VR-environment facilitates the utilization of computerized surgical simulations in educational programs. It improves the training efficiency through reducing the learning time spent on familiarizing trainees with VR environment.

2.3 Sensimmer-based VR Environment

Sensimmer is a framework for development of haptics-enabled VR-based surgical simulations. It integrates open source and commercially-available software to provide a complete software development kit (SDK) for augmented virtual reality applications. Sensimmer empowers the development of the ImmersiveTouch simulator[64] (see Figure 7a) and supports other research in augmented virtual reality and telerobotics[65].

2.3.1 Software modules

Sensimmer integrates OpenHaptics and Coin3D OpenInventor for haptics and graphics rendering respectively. The two library share polygonal models in order to achieve consistency between visual and tactile cues. Sensimmer provides stereoscopy rendering. It incorporates tracking devices and provides customized camera scheme to adjust head tracking to specific display setup. Sensimmer could be configured with dynamics engine

including PhysX and Vortex to simulate realistic physics in the virtual scene. Dynamics engine handles collision between primitives to polygonal mesh. This could be leveraged to enable primitive-based haptic interaction at the cost of stability of haptics rendering. FLTK is a cross-platform library for graphical user interface (GUI). Sensimmer also relies on its embedded GLUT emulation for windows management.

2.3.2 Hardware Component and Setup

Commercial hardwares are assembled to create and enhance augmented VR experience. The design of the augmented VR environment is discussed in details in Luciano[64].

The Geomagic Touch haptic device provides 6-DOF interaction. The workspace of the device maps the virtual scene to the physical world and provides intuitive way for user to interact in VR space.

A reflective mirror flips the stereoscopic display and projects the graphics rendering of the virtual scene to match the work space of the haptic device. It enables collocation during user interaction and enhances user experience of realistic VR.

A head tracking device detects the motion of user's head to compute the 3D perspective of virtual camera and improves the registration between the physical world and the virtual world.

2.3.3 Previous Simulations

Previous research endeavors explore approaches to construct patient specific model utilizing medical imaging data.

Rizzi[66] proposed a pipeline for automated segmentation and model extraction to convert medical imaging data into 3D polygonal models that could be used in haptics-enabled VR environment. Volume haptics introduced later into Sensimmer enables haptics rendering of volumetric data set. The integration of volume haptics expanded the compatibility and flexibility of the platform. It further reduces the human intervention involved with the model extraction and expedites the process of importing patient model into simulator.

Volume haptics algorithm enables the direct access to volume data set instead of polygonal models. It provides straightforward and computationally trivial model modification. The advantage of volume haptics approach enables the development of multiple surgical simulations on the ImmersiveTouch simulator. Benefits of the new pipeline have been validated by curricula accommodating simulations developed on this platform.

2.3.4 Volume Haptics

Previous research proposed various ways to interpret and represent haptic volume. A detailed discussion regarding existing approaches is presented later in Chapter 3.

The volume haptics module in Sensimmer adopts intermediate representation method[1] and implements it in OpenHaptics using custom shape. The custom shape from OpenHaptics allows client application to create a surface constraint for each custom shape. This is consistent with the collision plane generated by the intermediate representation method. The volume haptics implementation uses a Gaussian filter to alleviate the penetration problem that is common in intermediate presentation method.

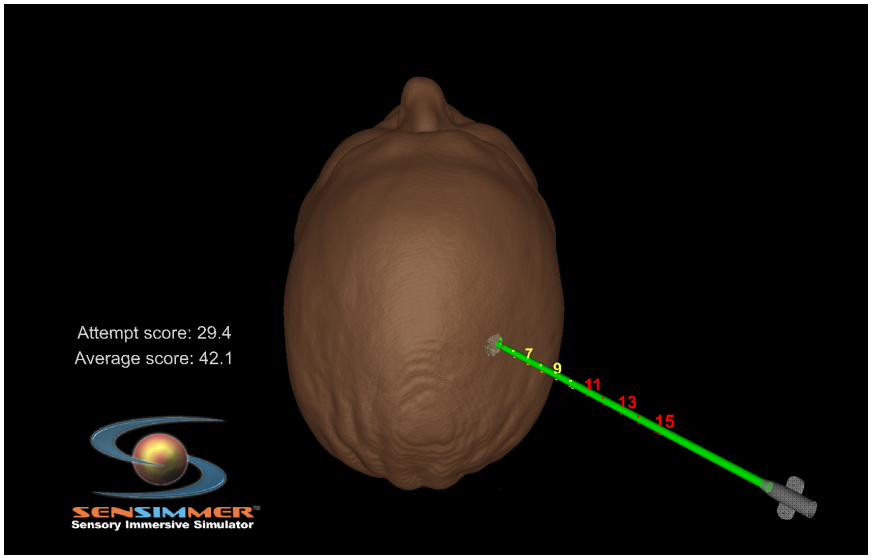


Figure 8: Ventriculostomy simulation

Ventriculostomy simulation with improvement of volume haptics module that makes it possible to drill burr hole during training[67].

Although there are many caveats with the legacy volume haptics including penetration of the intermediate representation and artifacts from the Gaussian filter. Multiple simulations developed using this tool shows the benefits of surface-based volume haptics and the potential of an pipeline that requires least human-in-the-loop interaction.

2.3.5 Ventriculostomy

Ventriculostomy is the procedure to access ventricle of the brain through catheter insertion. The procedure requires optimized trajectory which is planned through computed tomography (CT) study and appropriate insertion depth that is achieved by sensing the popping when catheter enters the ventricles.

Initial ventriculostomy simulation utilizes 3D isosurface extracted from imaging data for both haptics and graphics rendering[68]. Users practise catheter insertion on predefined entry points as illustrated in Figure 8. Later introduced volume haptics module enables user to create burr holes prior to catheter insertion[3]. Meanwhile it facilitates the expansion of patient cases by reducing the human intervention required to import medical imaging data into simulation.

The caveats of intermediate representation have little impact on user experience because of the interaction pattern in this simulation focuses on "popping" instead of a continuous surface contact.

2.3.6 Pedicle Screw Insertion and Craniotomy

Simulations for neurosurgeries involving extensive bone interaction include pedicle screw insertion and craniotomy.

Pedicle screw insertion is one of the steps during spinal fixation where surgeons place screws in the vertebrae. Our simulation involves three steps starts with drilling the surface of the vertebrae. A blunt is then inserted through the drilled surface to create a pedicle track. Finally a screw is placed following the pedicle track. Craniotomy removes a portion of skull to grant access to brain. The simulation requires the trainee to draw a bone flap on the skull and later isolates and removes it.

These procedures require both bone sculpture and continuous bone contact with the tool. Continuous bone contact eventually results in surface penetration. While bone sculpture breaks, the smoothed volume transitioning area required to sustain surface repre-

sentation and aggravates the problem. However, bone drilling is necessary for many neurosurgerical procedures. Meanwhile surface penetration damages the fidelity of the simulation and makes it difficult to interact with virtual models.

2.4 Simulation with Robust Surface-based Volume Haptics

Previous research works show great potential of haptics-enabled VR simulation platform for surgical education.

The robust surface-based volume haptics approach is developed to fill in the void with haptics rendering focussing on reliable representing of the isosurface extracted from volume data. The approach is all-inclusive to enable volume haptics applications without haptics pipeline. Meanwhile, it is versatile to expand haptics rendering to volume data for application with existing polygonal-based haptics pipeline.

The volume haptics module has been integrated with OpenHaptics pipeline in legacy Sensimmer SDK as well as haptics API from in-house Unity3D toolkit. Improved volume haptics allows us to develop cranial and spinal simulations with extensive bone interaction on those platform.

The software platform we created enables researchers and surgeons to efficiently prototype simulations and to study benefits of VR-based simulations in broader subjects. Moreover, surface-based volume haptics reduces labor-intensive preprocessing on medical imaging data and enables exploration of surgical planning on haptics-enabled VR environments.

We designed the next generation VR-based surgical simulators to teach principles and skills of surgeries using patient specific cases and pathologies. A VR-based simulator provides a diverse library of cases accessible to students. The diversity is difficult to be provided by traditional physical simulators which focus on teaching tool operation and soft tissues manipulation. Curriculum combining both simulators have advantages on accessibility and comprehensibility.

CHAPTER 3

SURFACE-BASED VOLUME HAPTICS

In this chapter we introduce the two surface-based volume haptics approaches. Our approaches are flexible to merge into pipelines with or without polygonal haptics module. The algorithm ensures a robust surface representation from volumetric data set, which is crucial to tactile rendering in scenarios like surgical simulations.

We start this chapter with a brief introduction to previous haptics rendering techniques. Then we discuss a fast ray casting algorithm used in the image rasterization and culling, followed by our implicit surface volume haptics algorithm that provides the desired tactile quality for our simulations. Later we discuss our explicit surface extraction technique that further expands the adaptability of our volume haptics to different pipelines.

3.1 Haptics Rendering

Proxy-based haptics is the prevalent technique used for haptics rendering of both polygonal models and volume models. The proxy-based haptics approach[18] maintains a god-object that follows the motion of HIP while being constrained outside of any virtual models. A spring and damper model connects the god-object to the HIP and generates the force rendered to the user (see Figure 5).

The approach is proposed initially just to handle polygonal model as discussed earlier in Subsection 1.3.2. Collision between HIP and geometries is detected. The constrained

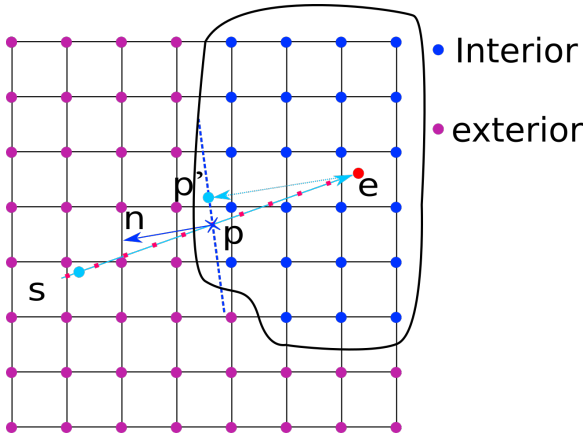


Figure 9: Previous volume haptics approach

Illustration of intermediate representation of volumetric data set and volume haptics approach. Each dot on the grid represents a voxel. Solid curve is the intermediate surface.

proxy is calculated by projecting HIP to the collision plane. The calculation could be accomplished through either solving a least energy model with multiple constraints or iterative collision detections.

3.1.1 Volume Haptics Extension

Researchers extend the idea to apply it on volumetric data sets[69]. Most concepts are borrowed from direct volume rendering in computer graphics. An intermediate surface representation is introduced to the continuous 3D scalar space reconstructed from volumetric data set. An intensity value is interpolated from the intensity values of neighboring voxels for a given grid. An intermediate surface is determined by the combination of two parameters: 1. a collision point where the interpolated intensity equals to a given threshold; 2. the estimated normal on that point.

The normal estimation uses the gradient calculated on the volume data as (Equation 3.2). It generates a vector from a scalar field through finite differentiation using one of the method noted in (Equation 3.1).

$$\begin{aligned} &\text{forward difference} && \Delta_h[f](x) = f(x+h) - f(x) \\ &\text{backward difference} && \nabla_h[f](x) = f(x) - f(x-h) \\ &\text{central difference} && \delta_h[f](x) = f(x+\frac{1}{2}h) - f(x-\frac{1}{2}h) \end{aligned} \tag{3.1}$$

$$\nabla f = \frac{\partial f}{\partial x}\vec{i} + \frac{\partial f}{\partial y}\vec{j} + \frac{\partial f}{\partial z}\vec{k} \tag{3.2}$$

Normal estimation in volume space has been used to generate smooth gradient transition while preserving sharp features[70,71].

3.1.2 Volume Penetration and Haptic Fall Through

Because of the nature of the intermediate surface representation, it is difficult to efficiently present a correct global surface topology without extensive ray casting and surface construction. As in Figure 9, the algorithm takes samples at fixed interval along ray from $\vec{s}$ to $\vec{e}$. $\vec{p}$ is the first sampler with an interpolated intensity value higher than the threshold. It calculates the estimated normal $\vec{n}$ at $\vec{p}$. $\vec{p}$ and $\vec{n}$ defines the constraint plane. Projecting the target $\vec{e}$ onto the constraint plane we get the new proxy point $\vec{p\prime}$. Firstly, it is easy to see that the collision point $\vec{p}$, representing intermediate surface, does not align with the global isosurface of the volume dataset. In order to detect a more precise collision point,

a very fine interval need to be used that could introduce huge computational expenses. Furthermore, new proxy location $\vec{p}\prime$ has penetrated the isosurface and enter the virtual object. The constraint plane could be lost as the penetration goes deeper and reaches the region with no defined normal estimation.

Efficient ray casting algorithms[72–75] reduces the computational time but requires preprocessing on volume data. Lost of constraint plane could be alleviated through using different techniques for haptics rendering that does not necessarily focus on representation the underlying surface feature. Preprocessing the volume data could also alleviate the problem as used by Rizzi[3]. Figure 10 shows a Gaussian filter creates the transition from exterior voxels to interior voxels and expands the region with defined normal estimation.

With these enhancements, previous volume haptics algorithms could deliver surface constraints on smooth convex shapes during temporary interaction. Unfortunately both techniques alleviate the issue of penetration without providing a solution to correctly reconstruct the isosurface or to maintain surface constrains. These approaches suffers from haptic fall through on regions with complex geometry topology and detailed texture (see Figure 11). Figure 12 shows how concave features are prone to penetration and haptic fall through.

The problem of fall through from previous volume haptics approaches limits its application involving models with complex shapes or prolonged voxel interactions. Surgical procedures like craniotomy and laminectomy require prolonged interactions on the skull and spine with rich texture and complex geometry. The penetration and haptics fall through

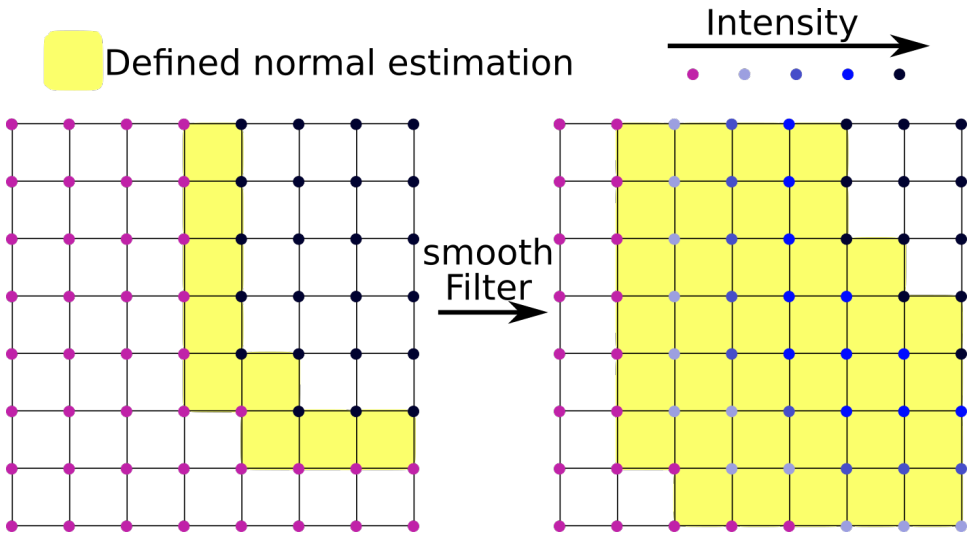
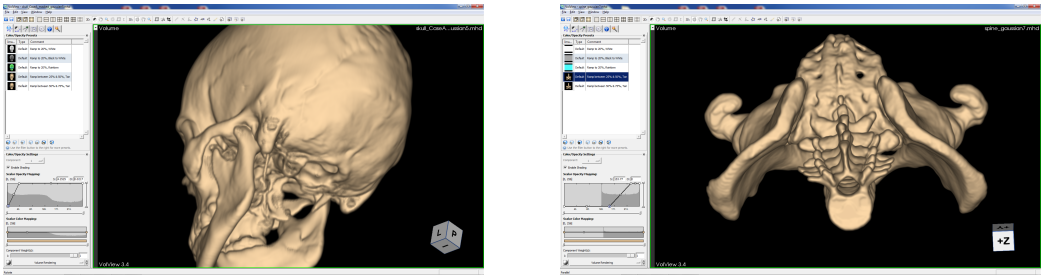


Figure 10: Smooth filter

Gaussian filter created a buffer that expands the thickness of regions with defined normal estimation. These region could sustain collision plane for penetrated proxy.



(a) (b)

Figure 11: Complex geometry in volume data set

Complex topology in human anatomy causes haptic fall through. (a) Textures and uneven surface from skull causes penetration that leads to haptic fall through. (b) Human spine has many concave shape making it impossible for existing volume haptics algorithm to maintain consistent haptic constraints.

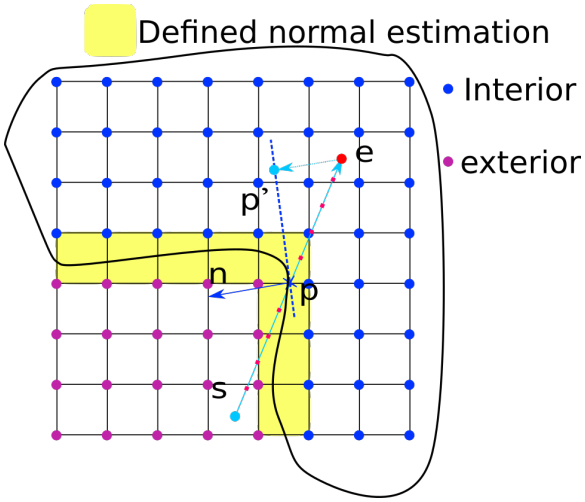


Figure 12: Volume penetration and haptic fall through

For collision at concave geometry, due to the inconsistency between the global geometry to the reconstructed collision plane, projected proxy $\vec{p}\prime$ penetrates the boundary and remains inside the geometry. In the following frame, it cannot create a collision plane to maintain the surface constraint. It results in the lost of haptic feedback perceived as haptic fall through.

undermine the surface representation of patient specific anatomical structures and impair the interactivity of the simulations.

Because of the discrepancies between haptics rendering and graphics rendering, we believe it is not suitable to directly employ some of the techniques from one to the other. Hence continuous intensity interpolation is not the suitable approach for reconstruction of the surface feature in direct volume haptics rendering. Instead, we explore techniques providing robust surface constraints for volume data set. With the improved surface representation, our volume haptics algorithm renders haptic feedback during interaction on

volume data when previous approaches fail. Our volume haptics module supports the development of surgical simulation on complex cranial and spinal procedures.

3.2 Voxel-resolution Collision Detection

We propose the new collision detection scheme that detects collision only on voxel-scale resolution. The idea is similar to spacial partition using octree, which is a popular technique for occlusion culling.

Traditional collision detection with intermediate surface representation for volume data requires a continuous scalar field to extract collision plane. It uses a sampling process to interpolate intensity values between voxel spacing. Subvoxel sampling breaks each cell (formed by 8 voxels) into an unstructured voxel cluster.

Our scheme is to avoid subvoxel collision detection, but instead use either individual voxel or cell as basic collision unit (see Figure 13).

High sampling rate and fine scalar field with interpolated intensity are critical to graphical direct volume rendering to reduce artifacts in rendered image.

These introduced artifacts are intensified given transfer function with steep gradient. High sampling rate (usually at an order of 10 samples per grid or higher) is required and introduces undesired computation complexity. Even for graphics rendering where the hardware is highly optimized at handling such computation, they still depend on alternatives like pre-integration, progressive rendering and filtering to achieve real-time frame rate[76–79].

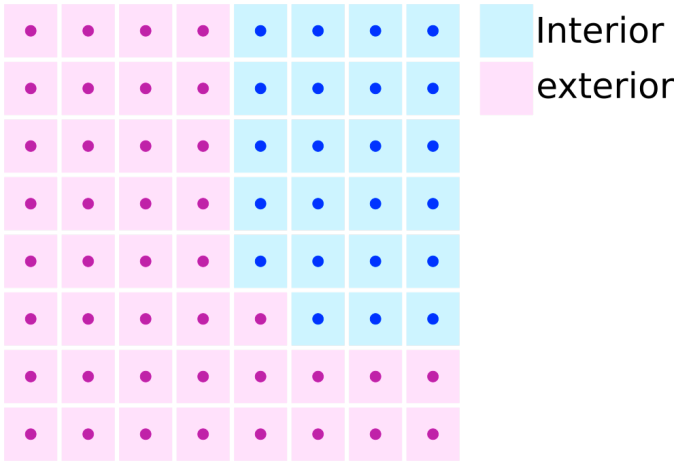


Figure 13: Voxel collision units

Instead of dividing the volume space into cells that formed by 8 voxels, this model divides and assigns neighboring space to each individual voxel.

Intermediate surface representation for volume haptics rendering defines a threshold for the intensity value on the boundary. This could be interpreted as a band-pass transfer function in graphical rendering that requires very high sampling rate. Proxy-based haptics collision mitigates the intensive computation through high frame rate and short ray length. But the influence is still observed during fast motion.

Voxel resolution collision eliminate the computational problem since there is no sampling or interpolation required. We focus our collision detection to an approximate block-shaped boundary. It simplifies the process of probing global surface into a localized problem. The step effects introduced by geometry model could be easily removed through an exponential filter on the output of rendered force.

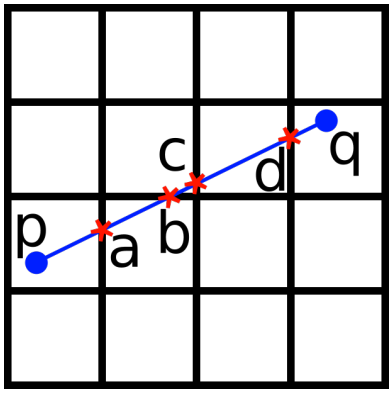


Figure 14: Fast ray traversal

Ray casting from $\vec{p}$ to $\vec{q}$, the approach returns detects all cubes visited by the ray, as well as all intersecting points a, b, c and d.

3.3 Fast Ray Casting

Fast continuous collision detection[80] is critical for proxy-based haptics approach. Voxel traversal algorithm is used for 3D space partition and rasterization[81]. Fast voxel traversal algorithm for ray tracing efficiently detects traversed voxels and could be modified to return all intersecting points along the ray.

We utilize the fast voxel traversal approach to detect voxel encounters along ray cast as the backbone of our continuous collision detection.

3.4 Robust Surface-based Volume Haptics

We proposed two methods for the volume haptics module. Implicit surface collision detection includes fast discrete collision detection and proxy-based force rendering model

using spring-and-damper. While explicit surface extraction enables efficient generation of isosurface, it provides a volume haptics extension to polygonal-based haptics technique.

3.4.1 Implicit Surface Collision Detection

We use a proxy model for our force rendering. The proxy follows the HIP and is constrained on the exterior of volume object. A spring-damper model connects the proxy and HIP to compute the force to be rendered each frame.

The implicit surface collision takes each voxel as a collision units. It takes the 3D scalar field and divides the whole space into a uniform grid of cubes. Each voxel occupies a unit cube space centered at the voxel location. We categorize every voxel as either exterior or interior according to their intensity (see Figure 13); We follow the constraint that proxy should never enter the space occupied by interior voxels.

For three-dimensional volume data, we define volume space coordinate system starting at the bottom-left-front corner with unit distance on each axis between neighboring voxels.

In order to have the fast ray casting to detect the encounter of each voxel cube, we shift the volume space coordinate system by half a unit distance on each axis to create the shifted volume space coordinate system.

In each collision frame, we identify the proxy position as the start point $\vec{s}$ and device position as the end point $\vec{e}$ for our collision detection ray (see Figure 15).

If there's no interior voxels along the ray, there is no collision detected and we simply update proxy position to the device position.

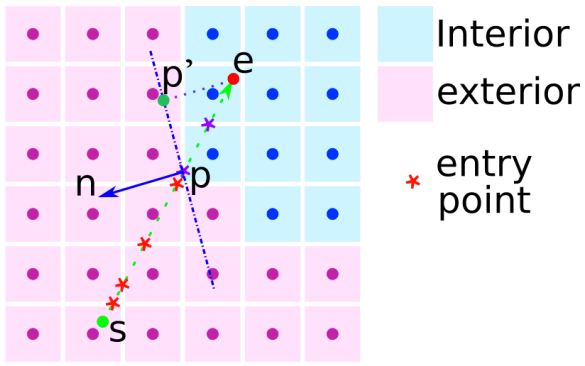


Figure 15: Voxel resolution collision

Crossing from exterior voxel to interior voxel has been detected along the ray from $\vec{s}$ to $\vec{e}$. Fast ray traversal finds all entry points marked as star in the figure. Implicit plane $\mathcal{P}$ generated using estimated normal $\vec{n}$ and collision point $\vec{p}$ at current iteration.

The entry point of first encounter from exterior voxel to interior voxel is identified as the collision point $\vec{p}$. A normal vector $\vec{n}$ is calculated at $\vec{p}$. Notice that although our collision unit is discrete, our collision point and normal is not necessarily discrete and could be precisely calculated/interpolated. The interpolated normal could improve the smoothness of our implicit collision plane.

A projection plane $\mathcal{P}$ is determined through the two vectors. We project the HIP onto the projection plane $\mathcal{P}$ as the projected HIP $\vec{p}\prime$. $\vec{p}\prime$ is the target proxy location on $\mathcal{P}$ where we want to place proxy. Because it is close to the collision point $\vec{p}$ and it abides the lowest energy model. However, we cannot simply move proxy to $\vec{p}\prime$ because it might violate the constrain and penetrate the interior voxel.

The objective is to move the proxy close to the HIP $\vec{e}$ without entering the interior voxels. We developed two methods for proxy movement. Iterative collision detection is a simplified feature-based constraint model from polygonal haptics. Path-finding algorithm is used to solve the undesired collision introduced by our cubic voxel geometry.

3.4.1.1 Iterative Collision Detection

This approach uses ray casting to move the proxy towards the target position at each iteration.

After first collision detection, we have the first pair of collision point $\vec{p_1}$ and projected HIP $\vec{p_1}\prime$. Now we take $\vec{p_1}$ as the new start point $\vec{s_2}$ and $\vec{p_1}\prime$ as the new end point $\vec{e_2}$. Follow the same principle we could perform iterative collision detections. The iterative collision detection is terminated when there is no collision detected or the total number of iterations reached a user defined threshold.

Self-occlusion for each iteration is a trivial issue. This happens due to errors during float computation. Similar to direct volume rendering we solve it by stepping our $\vec{p_i}$ along normal $\vec{n_i}$ by a small distance.

There are other problems concerning the convergence of our iterative approach. Firstly, for each iteration i, the algorithm tries to find the local minimum energy model for $\vec{p_i}\prime$. There is no guarantee that during each iteration the result would improve and would get closer to global minimum energy model of $\vec{s}$ and $\vec{e}$. Secondly, oscillation of proxy position between frames could happen in narrow concave volume topology, where interior voxels form a steep valley. This produces unstable force rendering and intense vibration. To solve

this problem, we keep track of all collision points through every iteration as proxy candidates. At the end of the iteration we perform candidate trimming by comparing their distance to the device position $\vec{e}$ and keep the lowest energy point. Lastly, even with iterative collision check, the artifacts introduced by our cubic voxel geometry still causes many undesired collision. Imagining a "flat" plane portrayed by a volume data set, if the plane passes through one axis and has a 45° angle with the other two axes (see Figure 16), the zigzag boundary causes undesired of collision while it represents a flat surface. With a large iteration threshold or a high collision frame rate, the collisions could be skipped through many iterations and the approach produces a smooth proxy trajectory. We observe dragging proxy under the combination of low iteration and low collision frame rate. However, at 1K collision framerate and an iteration threshold of 3, the dragging effect is no longer noticeable.

3.4.1.2 Path-Finding Technique

Path-finding uses the projected HIP $\vec{p}\prime$ as target position and tries to find the path from the collision point $\vec{p}$ to the exterior voxel point closest to the target position.

The idea behind path finding is to avoid undesired collisions observed in ray casting. We need the path finding algorithm to be able to pick up any straight path without going around too many obstacles. Otherwise it will ignore the geometry topology of the volume object that we do want to preserve.

We use a modified $A*$ algorithm. $A*$ algorithm runs on the discrete volume index space, we find out the voxel space where $\vec{p}$ and $\vec{p}\prime$ belongs to and set them as the start $\vec{v_s}$ and the

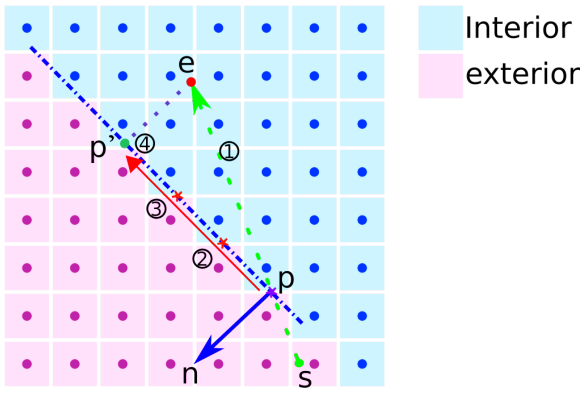


Figure 16: Non-axial aligned volume plane

Although the surface represented by the volume data should be a flat plane as $\mathcal{P}$. The space division from voxel collision model introduces artifacts that causes multiple collision detected marked as red star in the following red ray $\overrightarrow{pp\prime}$.

target $\vec{v_t}$. The normal at $\vec{p}$ and a predefined threshold t_{depth} could define a constraint plane to reduce the searching space. It helps us to pertain topology features in volume model.

After we marked the constraint on our voxel index space, we could use original $A*$ algorithm to find the exterior voxel $\vec{v_{p\prime}}$ closest to target $\vec{v_t}$. We connect the center of $\vec{v_{p\prime}}$ to device position $\vec{e}$ and take the intersection $\vec{p_c}\prime$ on the boundary of $\vec{v_{p\prime}}$ as the proxy position candidate. This step alleviates the step effect resulted from the discrete voxel grid.

Similar to the iterative collision detection approach, we solve the convergence problem by comparing the energy level of $\vec{p_c}\prime$ and $\vec{p}$ and pick the lower one to avoid oscillation.

The major advantage of the path finding approach comparing to ray casting is the capability to go around geometrical artifacts without violating global topology. This eliminates

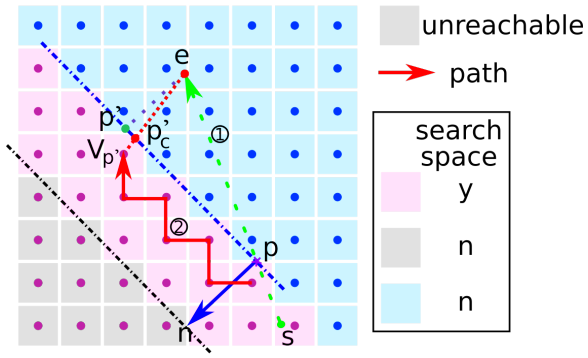


Figure 17: Proxy sliding using path finding

Using path finding in the second phase, we find the channel leading to the exterior voxel leading to $\vec{p}\prime$ from $\vec{p}$.

the false collision and dragging problem that requires our previous approach to utilize large number of collision iteration.

3.4.2 Explicit Surface Extraction

There are two advantages of using the explicit surface representation over the implicit surface representation for volume objects. One is the consistency between graphics and haptics rendering. If we use the same surface extraction algorithm we could generate identical polygons that represents a consistent volume object in both graphics and haptics. The other advantage is that the usage of polygonal models makes it easy to integrate volume haptics into pipelines with polygonal haptics modules. Details would be discussed later in Chapter 4.

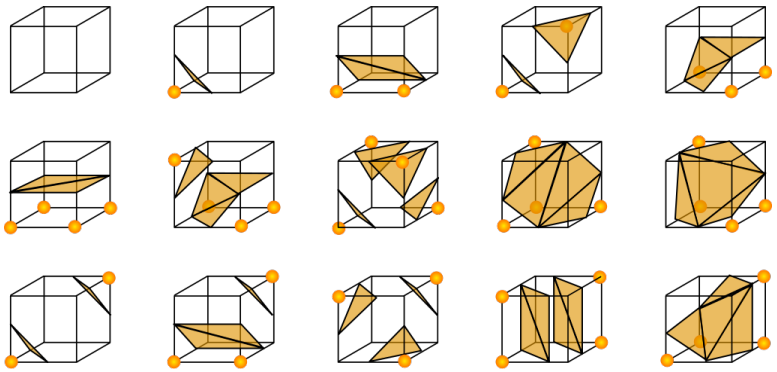


Figure 18: Marching cubes

Marching cubes reduces 256 cases of triangulation of a cube into 14 patterns in Lorensen[82].

3.4.2.1 Marching Cubes

Marching cubes extracts an isosurface from the volume dataset[82]. This technique looks at individual cell which is formed by 8 voxels. Comparing the voxel intensity value to a preset threshold, the algorithm interpolates the boundary crossing point on each of the 12 edges along the cell according to the intensity of the two vertices. It connects the crossings points according to 256 possible voxel intensity masks value and creates triangles that forms the isosurface (see Figure 18). Each edge is shared by 4 neighboring cell and the algorithm interpolation generates exact crossing points along the edge for each cell. The output of the procedure produces a manifold that resembles an isosurface of the volumetric object.

Later techniques use the dual scheme to generate surface topology with less computational complexity[83–85].

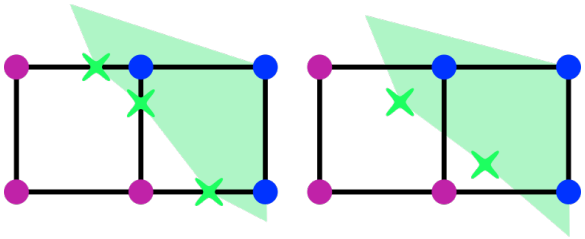


Figure 19: Primal and dual classification

Left figure demonstrates the primal scheme; Right figure demonstrates the dual scheme.

3.4.2.2 Primal and Dual Classification

The differences between primal and dual schemes could be illustrated in Figure 19. The primal scheme finds the crossing and generates triangle vertex per edge in volume grid. It then connects neighboring vertices to form triangles within each grid. Meanwhile the dual scheme generates triangle vertices per volume grid. It then finds the crossing and generates triangles per edge in volume grid.

From the figure we could easily see that dual scheme generates simpler topology comparing to primal scheme. For each volume grid there are at most one vertex generated in dual scheme. Primal scheme could generate four effective vertices (Each volumetric grid has 12 edges but each edge is shared by four grid).

The Dual scheme has another advantage that it avoids ambiguity in the topology from certain pattern mask. One drawback from dual scheme is the possibility of generating non-manifold vertices in the isosurface.

3.4.2.3 Naive Surface Net

Naive surface net (NSN) is a simple isosurface extraction algorithm using the dual scheme that was inspired from Surface Net[83,86].

Similar to marching cubes algorithm, NSN scans the value of all eight voxels and comparing them with a threshold value to generate 8-bit mask.

It then uses this mask to determine whether to generate a vertex for current cell and how to connect it to neighboring vertices to form triangles. Unlike Surface Net that calculates the position of vertex using trilinear interpolation, NSN only looks at zero-crossing and uses the center of weight to place the vertex. This greatly simplifies the implementation while preserving competitive results to the standard marching cubes. It also allows NSN to use quadrilaterals instead of triangles to reduce the total number of primitives.

Performance is the major obstacle that stops us from using polygonal models directly. Isosurface extraction algorithm generates enormous amount of triangles which require extensive computation for collision detection. Other attempts have been made to utilize preprocess including triangle reduction[66]. Naive surface net reduces the total number of primitives in the isosurface.

The problem here is that most of the triangles from the isosurface is irrelevant to the haptics rendering, because they do not contribute the constraints of the proxy update. We need an efficient trimming mechanism to remove the redundant collision test against far away triangles.

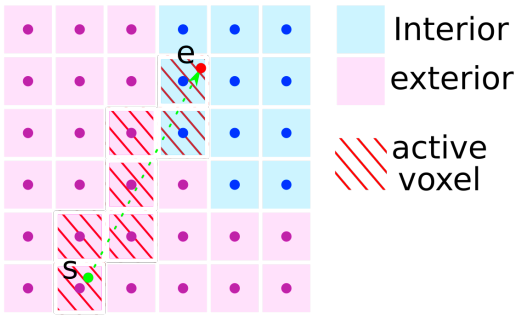


Figure 20: Explicit direct volume haptics

Fast ray traversal finds out all voxel that intersects with $\overrightarrow{se}$. Each voxel generates all triangles within neighboring grid.

We utilize fast ray casting to identify cells that intersect with HIP trajectory and then we generate primitives on the fly during each collision inquiry.

Although we only discuss 2 isosurface extraction algorithms here, the explicit surface representation is not limited to them. Applications could simply replace them with other algorithms as long as the new approach utilizes only local voxel intensities to generate triangles. We also leave the collision detection and haptics rendering to client application, assuming any application using this extension already has their implementation of polygonal haptics. More details on our implementation is discussed in Chapters 4.

3.5 Conclusion

Both the implicit surface collision detection and explicit surface extraction techniques provide robust surface representation. They solve the volume penetration and haptic fall through that previous volume haptics algorithms suffer from. Our approach maintains consistent surface constraint on volume data regardless of the complexity and topology of the underlying volume object. Moreover, our approach does not rely on data preprocessing like previous algorithms. It reduces the overhead on the pipeline of importing external volume data and it enables efficient real-time volume modification. Lastly, the presented approach has the flexibility to adapt to diverse pipelines. It could be integrated into existing applications to improve volume haptics rendering and to expand the haptic interaction models.

In Chapter 4, we discuss the integration of our volume haptics approach and experiments. Later in Chapter 5 and Chapter 6, we present surgical simulations we developed using the presented volume haptics approach, where previous volume haptics algorithms fails to deliver proper haptic feedback.

CHAPTER 4

VOLUME HAPTICS INTEGRATED SDK

We implemented the volume haptics algorithm and integrated it separately with Sensimmer and our in-house Unity3D toolkit. The integration of two frameworks with distinctive pipeline validates the flexibility of our approach. Moreover, the Unity3D toolkit shows advantages over the legacy SDK during our implementation. Its modularized design and commercial support shows great potential as a universal toolkit for development of VR-based surgical simulations.

This chapter covers in-depth discussion about our implementation and ideas behind our system design decisions.

4.1 Framework

Sensimmer is legacy SDK for haptics-enabled VR application development. The in-house Unity3D toolkit is an undergoing development. The Unity3D toolkit is being developed in an attempt to enhance and replace Sensimmer SDK in our production.

Haptics is one of the key components in both frameworks. Although the overall design and pipeline of Unity3D toolkit had significant influence and restrictions on the integration of our volume haptics module. It tests and validates the flexibility of our algorithm to adapt to an foreign pipeline. It also motivated our exploration of the extension of efficient explicit surface extraction in our algorithm. In the mean time, the integration of volume haptics

and certain surgical simulations demand functionalities that in return impact the design of many components in our toolkit.

4.1.1 Motivation

Sensimmer has been used to built many haptics-enabled VR applications. It facilitated the development by providing versatile application pipeline and exposing interfaces for customized modules[64]. Techniques and experiments are prototyped and developed using Sensimmer[65,66]. Surgical simulations built on Sensimmer has also been studied and leveraged in educational programs[68,87–92].

However, as Sensimmer evolves with its applications two limitation emerges and stumps many developments and projects. Firstly, with the advances of Computer Graphics and the ongoing bloom of VR, many tools and improvements has been made to the modules upon which Sensimmer is built. It takes enormous amount of work to keep Sensimmer core updated while compatible with legacy code. Secondly, Sensimmer community is relatively small comparing to prevalent VR-compatible framework. Compatibility between frameworks requires lots of development time in order to reuse modules implemented under different framework. This overhead impedes our research progress and discourages other researchers to experiment with our work.

With the above reasons, we started the development of a toolkit with loosely coupled modules built in commercial game engine Unity3D.

4.2 Sensimmer

Sensimmer integrated OpenHaptics library for haptics rendering. OpenHaptics accesses the graphics buffer and renders haptics for polygonal primitives using proxy-based algorithm.

Sensimmer exposes three run-time threads, graphics thread running at 30 frames per second (FPS), collision thread at 100 FPS and servo thread running at 1000 FPS.

4.2.1 Volume Visualization

Sensimmer uses GPU accelerated marching cubes implementation to extract isosurface from volume data. Direct volume rendering is also available natively through OpenInventor toolkit, but performance drops noticeably when larger volume data is in the scene.

4.2.2 Volume Haptics

Our volume haptics is enabled by inserting an volume collision module into OpenHapitcs pipeline. Our volume haptics module maintains its own haptics proxy point $\vec{p_v}$. Volume collision module is implemented based on our efficient implicit surface collision detection. Volume haptics module only interrupts OpenHaptics workflow when volume collision has been detected.

4.2.2.1 Volume Collision Integration

The volume haptics module initiated collision detection on servo thread at 1K fps. While OpenHaptics runs its own collision detection and produces device position $\vec{d_{OH}}$ and proxy position $\vec{p_{OH}}$, volume haptics module takes the end-of-frame proxy $\vec{p_{OH}}$ position from OpenHaptics as its target position $\vec{d_v}$ while maintaining its own volume haptics proxy $\vec{p_v}$.

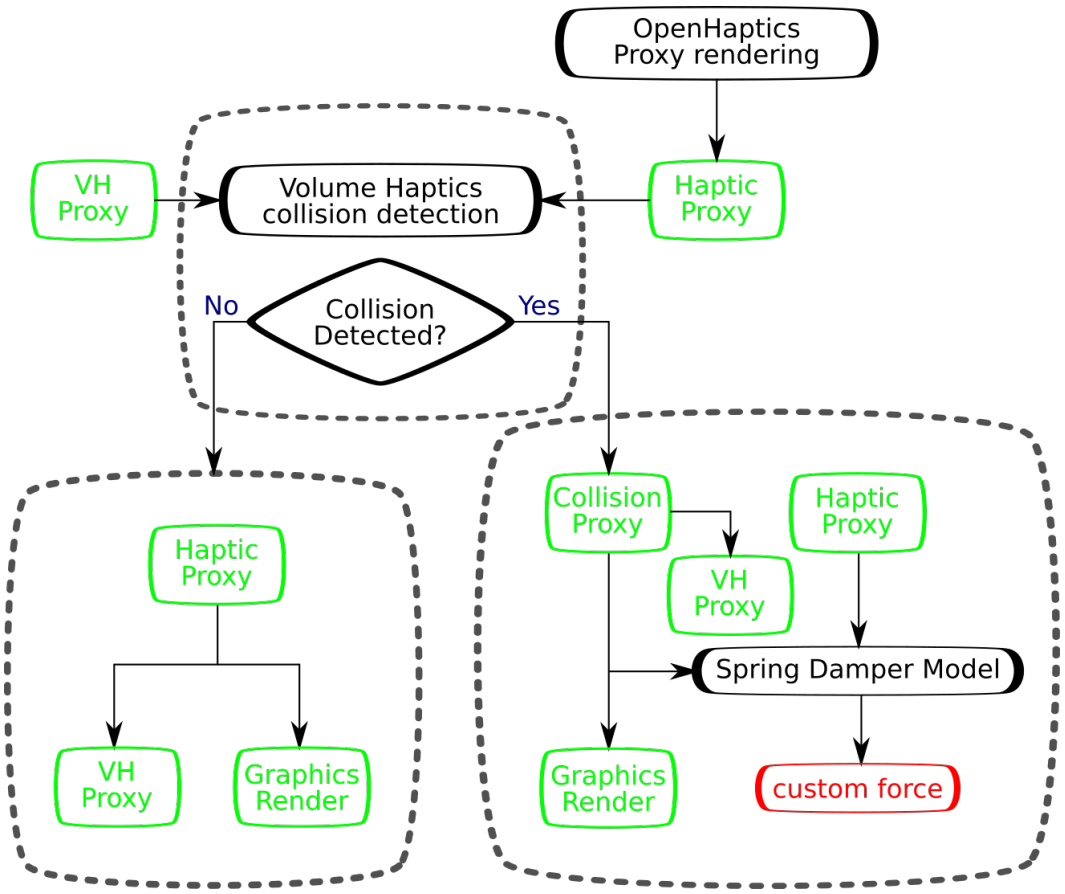


Figure 21: Volume Haptics Module Pipeline with Implicit Surface Collision Detectoin

Implicit surface volume haptics module takes OpenHapitcs proxy $p_{\vec{OH}}$ as its target position $\vec{d_v}$. Meanwhile it maintains its own volume haptics proxy $\vec{p_v}$. volume haptics module only intervenes when collision is detected.

Collision detection and proxy update are performed on each volume data set. We convert global vector $\vec{p_v}$ and $\vec{d_v}$ into a local coordinate for volume objects $\vec{p_v^{id}}$ and $\vec{d_v^{id}}$.

Collision detection produces four parameters for each volume object:

f^{id} Collision flag

d_p^{id} Penetration depth

$\vec{p_v}'^{id}$ Volume proxy location candidate

$\vec{c_v}^{id}$ Volume collision location

Volume haptics module only interferes the rendering pipeline when collision has been detected, during which it does two things. First, it calculates the contact force using spring model. Meanwhile, it takes the volume proxy location to update the graphics model, if proxy-rendering is enabled for haptic stylus.

Friction is implemented for volume object. If we consider the distance between $\vec{p_v}'^{id}$ and $\vec{c_v}'^{id}$ as the sliding distance, stateless friction could be introduced simply by adding a threshold for static friction and a constant step-back for dynamic friction.

4.2.2.2 Multiple Volume Objects

Multiple volume objects could be simultaneously present in our virtual scene as long as their isosurfaces are non-intersecting manifolds. This assumption is easily satisfied and sufficient for our target applications, where a single volumetric data set are labeled as different anatomical structures which are stored individual as masked volumetric data set.

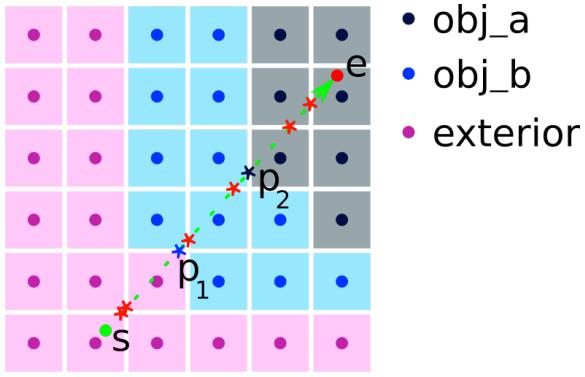


Figure 22: Multiple volume

Assuming object a and object b are both extracted from volume data set, collision ray $\overrightarrow{se}$ detects one collision point for each object as $\vec{p_1}$ and $\vec{p_2}$. Collision depth is defined as $|\overrightarrow{sp}|$.

This gives us the flexibility to put different textures on structures and distinguish them graphically in our virtual scene.

The collision for multiple volume objects is greatly simplified with this assumption. A complication only happens when volume collision ray casting penetrates multiple volume isosurface. We adapt a penetration depth check to detect the correct collision object identified as the volume with shortest penetration depth (see Figure 22).

4.2.2.3 Intersecting Polygonal Model

An intersection between polygonal model and volume objects could still generate reasonable haptic cue. Since our polygonal collision and volume collision are handled separately from each other, each module individually guarantees the surface constraint and together they prevent any penetration. Moreover, volume haptics module intervenes after

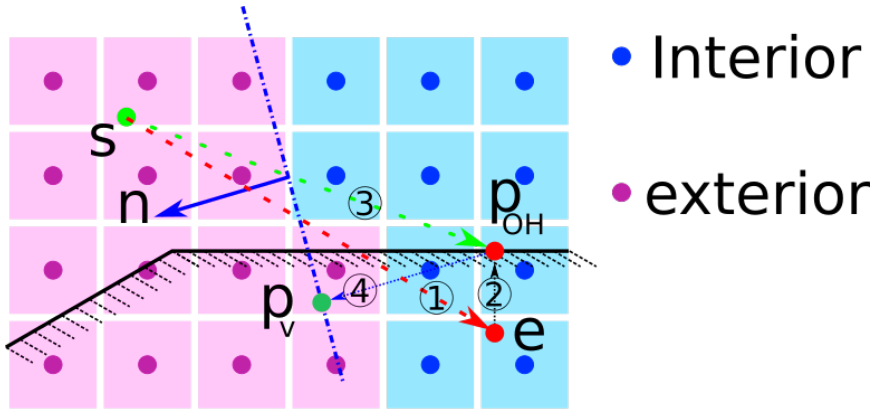


Figure 23: Volume intersecting polygonal model

Polygonal collision module use ray $\overrightarrow{se}$ and then project it to $\vec{p_{OH}}$. Volume haptics model use ray $\overrightarrow{sp_{OH}}$ and then project the volume haptics proxy to $\vec{p_v}$. Our pipeline uses $\vec{p_v}$ for graphics rendering of haptic instrument and creates a visual penetration.

OpenHaptics contraints and it takes OpenHaptics output proxy position as its target position instead of the real device position. This avoids the double counting of any penetration depth and corrects the final rendered force. Although graphically the proxy location does not correctly represent the combined surface constraints any more as in Figure 23.

4.3 Unity3D toolkit

Unity3D is a cross-platform engine with a huge community of developers in 3D games and VR/AR applications. Unity3D engine provides powerful scripting module, dynamic physics engine and versatile VR device support. Moreover, Unity3D enables user community to share and reuse independently developed modules as assets.

The features provided by Unity3D solves limitations we have been experiencing with the legacy Sensimmer framework. Unity3D developing team manages versions of software packages used by their modules and reduced large amount of maintenance work. Moreover, Asset store facilitates the accessibility of our work. Lastly, Unity3D provides efficient resource management scheme to deactivate modules and switch between scenes. It enables the capability to seamlessly merging sequential procedures.

The overall design of our toolkit follows the principle of Unity3D. Complete toolkit enables developers to build the baseline of haptics-enabled VR application. Meanwhile, we break down logically separate functionalities into lightweight stand-alone modules. We could easily swap each individual module to experiment with different implementations and algorithms. This also encourages people to merge the toolkit into their own application as we reduce the overhead to use individual modules.

4.3.1 Haptics API

The haptics module wraps on top of OpenHaptics low-level HD library. The module implemented polygonal model collision, force rendering and special haptic effects. The hapitcs API pipeline is shown in Figure 24.

4.3.1.1 Volume Data

The volume module loads raw volume data into an array. The volume instance inside SDK also stores meta information that is needed for reconstruction, including volume spacing and volume dimensions.

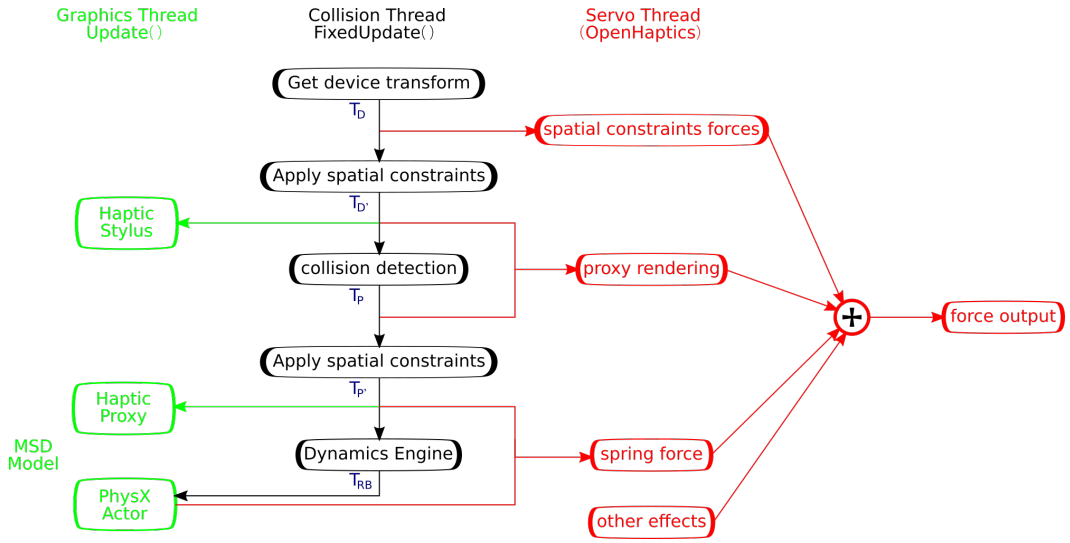


Figure 24: Hapitc API pipeline

Haptics API pipeline involves three threads running at different frequency. Graphics thread and Collision thread are managed by Unity. While Servo thread is managed by OpenHaptics.

4.3.1.2 Volume Coordinate System

Volume coordinate system (VCS) bridges the discrete volume index to the volume object in Unity world space as Figure 25.

Each object in Unity3D has a local coordinate system in the virtual scene as well as world coordinate system (WCS). Across the toolkit, we use WCS consistent manipulation and scale. However, We use the local coordinate system to map volume index directly. It allows us to leverage the transform provided by Unity3D engine to easily scale and shift our volume object in the virtual scene. While it is also easy to apply our volume haptics algorithm or any functionalities that requires direct access to raw voxel intensities.

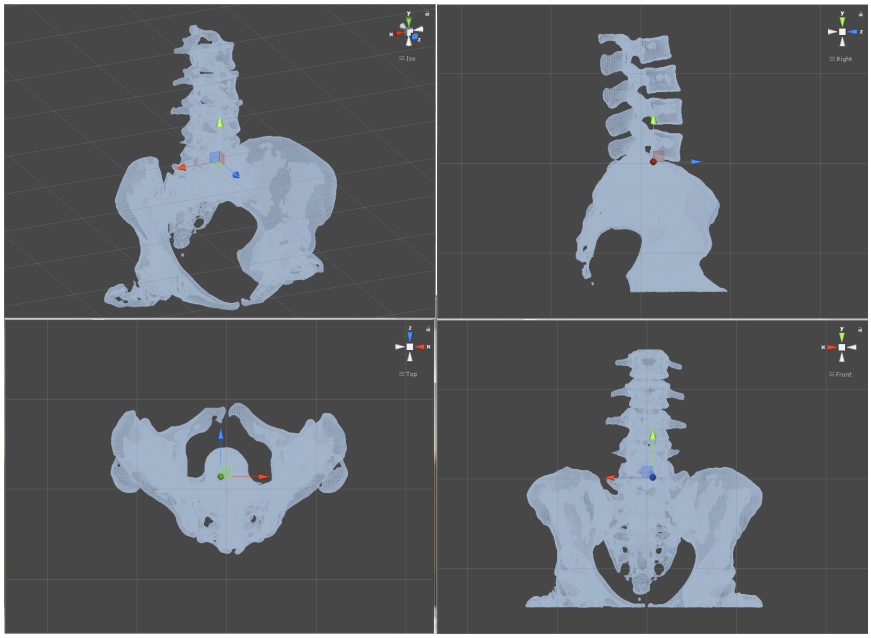


Figure 25: Volume Coordinate System

Volume coordinate system defines the center of volume space as the origin. It utilizes the transform matrix from Unity and connects volume index to the virtual scene graph.

4.3.1.3 Graphics Rendering

Toolkit provides two visualization modules for a volumetric object. Volume data could be visualized using either direct volume rendering module or isosurface module.

Our direct volume rendering stores volume as texture to utilize common hardware acceleration techniques[76,93] for performance improvement. However, current implementation cannot afford real-time volume modification for direct volume rendering.

For haptics-enabled volume object, isosurface approach for graphics rendering provides primitives that are consistent with the haptics rendering. Our Isosurface implementation breaks volume data into small chunks. Each chunk generates triangles on the isosurface inside its section. Our implementation use NSN algorithm to generate isosurface. We opt for concurrent CPU implementation for our isosurface extraction. Although the nature of NSN algorithm seems to be very suitable for parallel pipeline in graphics processing unit (GPU) architecture. It requires copying volume data into texture for GPU access. This will put restrictions on our pipeline. For example, any update to the volume data will require an update to the texture and could be expensive and infeasible to do in real time.

Volume chunks enables us to modify volume data and efficiently update the isosurface during run time. We could update the isosurface of chunks only for which the volume has been modified. We keep two copies of raw volume data separately on volume object and chunks for efficient data access. The volume object keeps the entire volume data as an array and each chunk keeps a copy of their data block. Application should interact only

with volume object. Any update would be sent to individual chunk through volume object. Updates mark related chunk as dirty and update of isosurface will be triggered by volume object.

From the perspective of performance, the smaller individual chunks means less workload during isosurface update. However, in Unity3D there is a performance penalty from the total number of active game objects. It also has a limitation on the maximum number of triangles each game object could have. Client application should determine chunk size for each application for optimized performance.

4.3.1.4 Haptics Rendering

Haptics module for volumetric data in our toolkit is implemented using explicit surface volume haptics technique mentioned in 3. The volume haptics module takes the ray query from the haptics API. It generates triangles along the ray and passes the triangles to a mesh collider so ray cast of the PhysX engine could detect any intersection (see Figure 26). This allows the haptics API to handle collisions and proxy updates of the volumetric data sets as polygonal models with precise and efficient triangle generation.

Explicit surface volume haptics exhibits great flexibility while we integrate it into our haptics API. It is straightforward to integrate it into the haptics collision pipeline. Meanwhile, the output triangles works with polygonal-based haptics seamlessly.

We also implemented implicit surface volume haptics approach. It utilizes customized force effect from haptics API to render force to the device. We utilized the `FixedUpdate()`

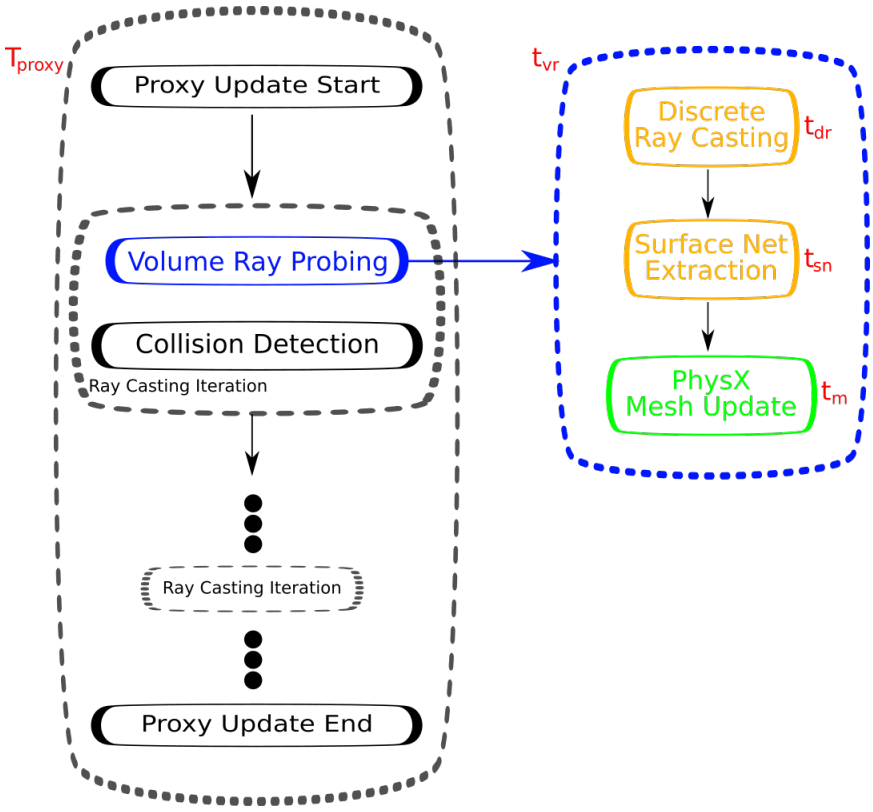


Figure 26: Collision Detection Frame with Volume Haptics Module

The pipeline for collision frame. T_{proxy} is the total time Haptics API takes to calculate its proxy location. We integrate explicit surface volume haptics by adding volume ray probing before each collision detection query to generate isosurface from volume data set in real-time. Volume ray probing consists of three modules and it could be called multiple times within a single collision frame.

from Unity3D as well as the 1K fps servo thread from haptic API as our volume haptics collision thread.

4.3.2 Head Tracked 3D Display

Researches have utilized head-tracked stereoscopic display for experiments of performance in "pointing" tasks in fish tank VR system[94]. Community claims head-tracked, egocentric camera control provided through head mounted display (HMD) enhance the sense of presence[95].

Our toolkit provide scripts to adjust camera position through tracking data updated at graphics thread. Currently we use commercial 3D guidance system with high precision.

Unity provides built-in HMD support through integrated libraries. Device compatibility is on per-platform basis. Because of SDKs with cross-compatibility, HMD-based applications are hardware independent.

4.4 Performance and Benchmark

In this section we discuss our experiments of the force rendering quality and performance of our volume haptics module.

4.4.1 Experiment Environment

The computer specs used for our experiment is listed in Table I. We run all experiments using Unity toolkit to leverage the profiling tools provided by Unity SDK. We used `CustomSampler` and `Recorder` from `UnityEngine.Profiling` because it leaves smaller footprint and provides better flexibility comparing to `Profiler`.

Hardware		Software	
Desktop	Dell Precision T7910XL	OS	Windows 7 64-bit
CPU	Intel(R) Xeon(R) E5-2640	Unity	2017.1.0f3
GPU	NVIDIA QUADRO 6000	GPU driver	NVIDIA 306.94
RAM	32 GB	PhysX	9.13.0725

TABLE I: Experiment Environment

4.4.2 Force Rendering Quality

Rizzi[96] compared performances among surface-based haptics rendering techniques using both intermediate representation of volumetric data sets and polygonal models. The research shows intermediate representation with great scalability regarding increasing data size. In terms of computation complexity, our implicit surface representation provides voxel-resolution collision detection without extensive sampling rate that intermediate representation requires. Hence, our approach has comparable performance with the potential to support more iterations of collision query.

Unlike intermediate representation where constraint surfaces between frames are irrelevant, force rendering quality of implicit surface representation could be affected by the artifacts introduced by voxel collision units as discussed earlier in Chapter 3.

Implicit surface collision detection running at custom effect thread at 1K Hz, which is much higher than the haptics API update thread running at FixedUpdate() in Unity. Implicit surface collision detection utilizes the proxy location from Haptics API as the target point

for ray casting. We observe a step wise pattern in rendered force due to the inconsistency between their updating rate.

We use non-recurrent non-seasonal exponential smoothing[97] to real-time force output from volume haptics module. The smoothing model is simple and effective.

$$f_n = \alpha x_n + (1 - \alpha) f_{n-1} \tag{4.1}$$

where: x_n is the calculated force from implicit surface collision detection, f_n is the output force from the volume haptics module at frame n, α is the smoothing factor that is used to adjust the smoothing effect.

In the first experiment, we engage a flat volume surface with the haptic device while monitoring the rendered force from volume haptics module. We record the original force calculated by implicit surface collision detection as well as the smoothed force output.

This experiment shows the transition from free motion to motion with contact constraint. Due to the implementation of implicit volume haptics in the Unity toolkit. Volume haptics module relies on the proxy output from the haptics API as the target for collision detection. While volume haptics runs at a much faster thread, the target update is slower than the collision detection and force calculation. The implicit collision surface from the experiment data here is a determined flat surface, hence the iterative ray casting converges and we observe the step-wise force output.

Figure 27 shows the smoothed force follows the trend of the raw output from volume collision detection. The enlarged session shows that the filtering removes the step effect and renders smooth force feedback.

The second experiment monitors a process of motion along a volume object. We use a human skull where the volume represented an uneven ellipsoid surface.

In this case, Figure 28 shows that filtered force can capture fine feature from the raw input, while the enlarged session demonstrates how filtering alleviates the overshooting of force rendering resulted from non-convergent iterative collision detection.

4.4.3 Benchmark

The haptic force rendering quality for explicit surface extraction based volume haptics approach depends on the polygonal haptics rendering technique.

We design our experiment to explore the efficiency and scalability of our voxel-resolution surface extraction scheme in combination with the Haptics API.

Our profiler measures the aggregated elapsed time of different code modules for each collision frame. Figure 30 illustrates the measurements for collision frame in the pipeline. Table II explains each measurements in details. We take a total of 3000 continuous contact and non-contact frames. We discard the first 100 frames to avoid the impact from the initialization process.

We explored the scalability of our volume haptics module using two experiments. The first experiment uses consistent voxel density with increasing volume size, where we keep the voxel spacing consistent among 5 volume data sets. As volume dataset doubles its

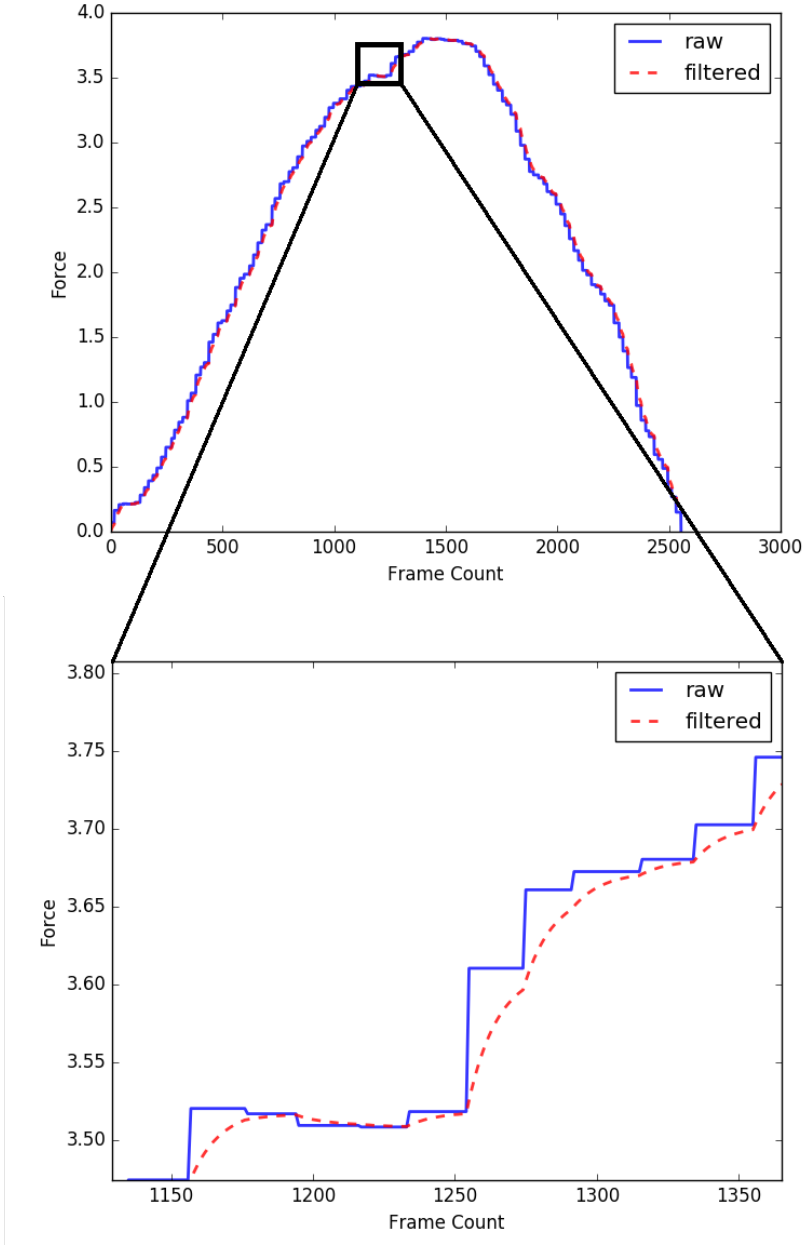


Figure 27: Z-direction Contact Force with Flat Surface

The simple geometry and slow graphics update resulted in the step-wise force rendering. Our exponential filter created a smooth output.

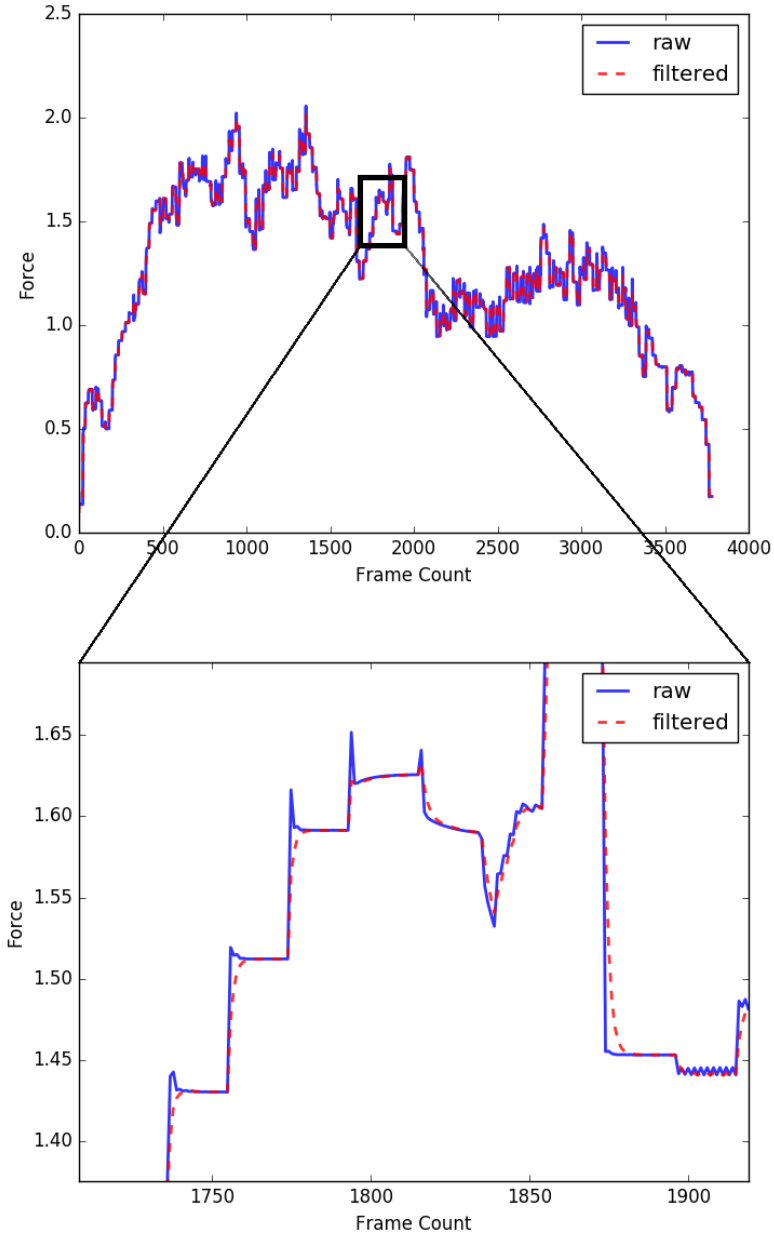


Figure 28: Contact Force with Uneven Surface

Divergent iterations in Haptics API collision lead to overshooting and oscillation for the force rendering. Again the exponential filtering alleviate the instability without eliminating subtle force feedback.

Measurement	Description
T_{proxy}	Total proxy update time for a Haptic API frame
t_{dr}	Accumulated time for discrete fast ray traversal
t_{sn}	Accumulated time for voxel-resolution surface net extraction
t_m	Accumulated time for triangles transferring to PhysX engine

TABLE II: Benchmark Code Segments

T_{proxy} contains multiple calls of volume probing. Each volume probing consists of sequential calls to three modules and its time complexity should be $t_{vrp} \approx t_{dr} + t_{sn} + t_m$

	T_{proxy} (ms)	
Data	No Touch	Touch
50^3	0.09 ± 0.08	0.34 ± 0.06
100^3	0.08 ± 0.06	0.34 ± 0.06
200^3	0.09 ± 0.07	0.35 ± 0.07
400^3	0.08 ± 0.06	0.35 ± 0.09
800^3	0.10 ± 0.06	0.37 ± 0.07

TABLE III: Scalability with Fixed Voxel Density

Increasing volume data size with fixed voxel density (spacing) does not affect the performance complexity.

length in each dimension, the extracted volume object also expands, while the number of voxels within a given space remains unchanged. We measure the total haptics frame time T_{proxy} during contact and non-contact events as listed in Table III.

T_{proxy} represents the overall performance for the volume haptics module. Performance during non-contact frames is trivial since the module only engages the fast ray traversal component for collision detection.

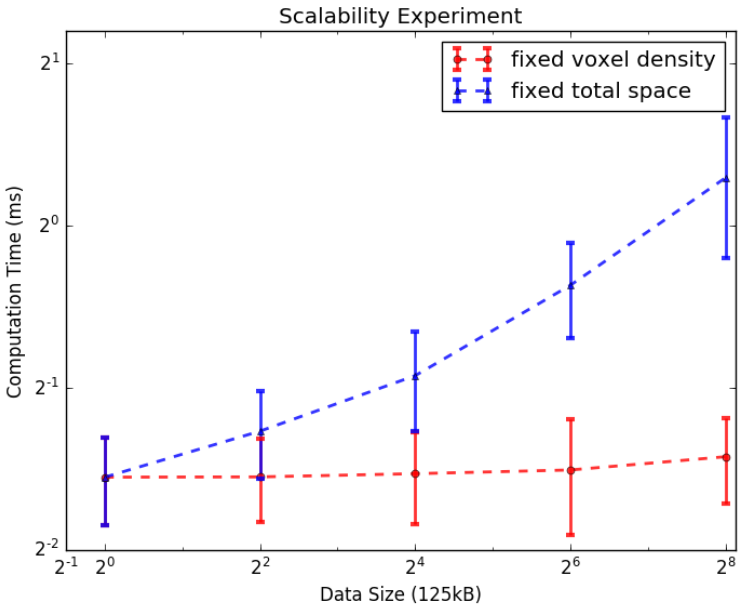


Figure 29: Performance and Voxel Density

Volume haptics module exhibits linear scalability to voxel density while remains unaffected by size of volume dataset.

The performance during contact frames measures fast ray traversal as well as explicit surface extraction. Because we use a localized approach that only access voxels within the interaction space, it is theoretically independent from the volume data size. The observation complies to our assumption that we did not see significant differences of T_{proxy} with increasing data size (see Figure 29).

In our second experiment, we explore the scalability over increasing voxel density. We kept the overall size of volume object in our virtual scene uniform over the 5 volume data

	T_{proxy} (ms)					
	Modest Contact		Continuous Contact		Rapid Contact	
Data	No Touch	Touch	No Touch	Touch	No Touch	Touch
50^3	0.09 ± 0.08	0.34 ± 0.06	0.09 ± 0.05	0.35 ± 0.07	0.11 ± 0.09	0.36 ± 0.05
100^3	0.11 ± 0.08	0.42 ± 0.08	0.11 ± 0.02	0.44 ± 0.09	0.11 ± 0.08	0.43 ± 0.08
200^3	0.15 ± 0.09	0.53 ± 0.11	0.13 ± 0.03	0.56 ± 0.11	0.13 ± 0.09	0.5 ± 0.1
400^3	0.22 ± 0.11	0.78 ± 0.16	0.2 ± 0.05	0.88 ± 0.21	0.18 ± 0.08	0.71 ± 0.12
800^3	0.36 ± 0.12	1.23 ± 0.36	0.34 ± 0.09	1.5 ± 0.36	0.3 ± 0.08	1.14 ± 0.21

TABLE IV: Scalability with Increasing Voxel Density

The increasing voxel density leads to longer computation time, while various interaction pattern does not cause noticeable impact.

sets, while we shrank the voxel spacing as we increased volume data size for each data set.

Following the same procedure we collected the results in Table IV. When we look at T_{proxy} across different data sets, we notice an increasing computational time as the voxel density increases. Figure 29 shows that the volume haptics scheme scales linearly to the voxel data density.

The performance during non-contact frames is still trivial comparing to the performance during contact frames and shows a similar linear scalability to voxel density.

In the next experiment we explored the characteristics of the volume haptics module regarding various interaction patterns.

For every data set we run 3 sets of experiments each involves a different interaction pattern: continuous contact, rapid contact and modest contact. These interaction patterns differ from each other by how frequently the user switches between contact and no contact.

Data	t_m (ms)	t_{dr} (ms)	t_{sn} (ms)	T_{proxy} (ms)	$\frac{t_m}{t_{vrp}}$
50^3	0.174 ± 0.032	0.023 ± 0.007	0.051 ± 0.018	0.341 ± 0.063	70.3%
100^3	0.200 ± 0.037	0.032 ± 0.009	0.095 ± 0.025	0.416 ± 0.077	61.3%
200^3	0.221 ± 0.057	0.051 ± 0.013	0.161 ± 0.038	0.527 ± 0.111	51.1%
400^3	0.317 ± 0.066	0.081 ± 0.021	0.280 ± 0.064	0.775 ± 0.156	46.7%
800^3	0.470 ± 0.178	0.148 ± 0.043	0.520 ± 0.155	1.228 ± 0.357	41.3%

TABLE V: Composite Performance

The detailed benchmark of volume haptics modules during each collision frame is shown here. The overhead of Haptics API on top of volume ray probing could be calculated by $T_o \approx T_{proxy} - t_{dr} - t_{sn} - t_m$.

The continuous contact pattern means the user does not switch the state during the length of the measurement, while the rapid contact pattern means user constantly switch between states. The experiment result is listed in Table IV. We do not observe any statistically significant differences among different interaction patterns. Benchmark data for each interaction pattern also shows similar scalability to increasing voxel density.

In the last experiment we looked into the performance of each component utilized by the volume haptics module. Because non-contact frames have trivial computational complexity and do not use every component, we do not look into non-contact frames. We also verified that different interaction patterns have little impact on performance from last experiment. Therefore, We focus only on data collected from contact frames with the modest interaction pattern list in Table V.

The results show that, t_{sn} and t_{dr} are more sensitive to the voxel density because it is the major factor determining their computational complexity. Although t_m increases with

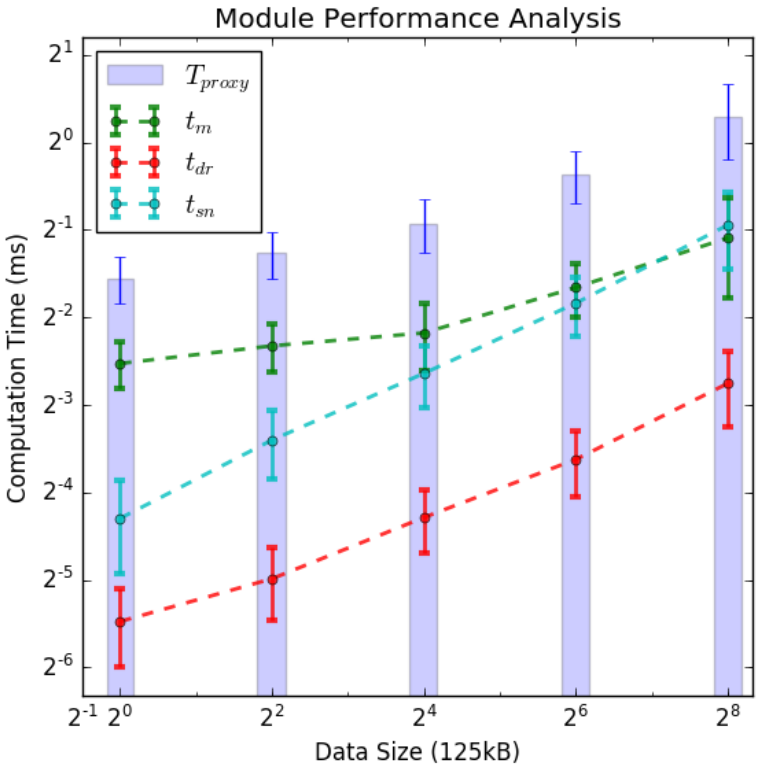


Figure 30: Detailed Benchmark during Explicit Surface Volume Haptics

the voxel density, it has smaller impact. We believe that the overhead of updating PhysX collision mesh has a bigger impact on the computational time comparing to copying triangles to PhysX at this data scale. T_{proxy} has a constant overhead on top of t_{vrp}. Meanwhile, the results show that through all scalability cases, t_m is the largest factor until the last case with volume data set of size 800^3.

Because Hapitcs API relies on PhysX ray casting for collision detection. During t_{sn} our volume haptics model already generates triangles for collision detection on CPU. The triangles need to be transferred to PhysX to access during collision detection.

Applications with haptics module that could directly utilize CPU memory could reuse the triangles without the overhead of copying it over to GPU. This could eliminate t_m from the timeline and improve the performance significantly (see $\frac{t_m}{t_{vrp}}$ in Table V).

4.5 Conclusion

The volume haptics algorithm renders robust and consistent surface feature of volume object regardless of its topology. It overcomes the drawbacks of previous volume haptics algorithms and extends the application of volume haptics to simulations with prolonged contact interaction.

Our volume haptics module accommodates different scenarios. It enables volume haptics to applications and SDKs with or without polygonal haptics module. The integration of the volume haptics into two different SDK validates its compatibility and flexibility.

The force rendering quality experiments show the smooth force feedback generated by our implicit surface volume haptics approach. Our algorithm combines exponential

smoothing and discrete fast voxel traversal approach in order to prevent surface penetration and to provide stable force feedback. The benchmark shows linear scalability of the explicit surface extraction approach. The capability to support real-time volume modification is critical to applications involving progressive volume sculpturing.

With reasons listed above, we believe the presented volume haptics approach can be easily integrated to other platforms. By enabling reliable and modifiable haptic surface representation from volumetric data set, the volume haptics module benefits applications of surgical simulations and pre-surgical rehearsal.

Last but not least, the Unity3D toolkit facilitates developments of haptics-enabled VR applications by providing universal modules. Its modularized design benefits prototyping and validating of new techniques. Loosely-coupled architecture makes the toolkit expandable and customizable to accommodate various development environments.

CHAPTER 5

FREE HAND CRANIOTOMY SIMULATION USING SENSIMMER

This simulation is built with Sensimmer SDK and uses volume haptics module to assist the interaction with skull model.

5.1 Clinical Procedures and Simulation

Craniotomy is the procedure which removes a portion from a skull to open the cranium (see Figure 31). It is one of the most commonly performed procedures for brain tumor removal. There are many other situations that would require this procedure to grant access to the brain, including hematomas, aneurysms, swelling of the brain, infection or skull fractures.

Craniotomy could be categorized by the area of the skull to be removed, as frontotemporal craniotomy or suboccipital craniotomy. It could also be categorized by the size and complexity of the operation. Procedures that drill a small hole to access the brain is called keyhole craniotomy, while large and complex craniotomy is referred to as skull base surgery.

Hospital stay length and risks associated with craniotomy depend on reasons for craniotomy. But overall it is considered to be a major operation associated with significant risks that could lead to severe consequences[99,100]. It should only be performed by experienced neurosurgeons with additional training in skull base surgery.

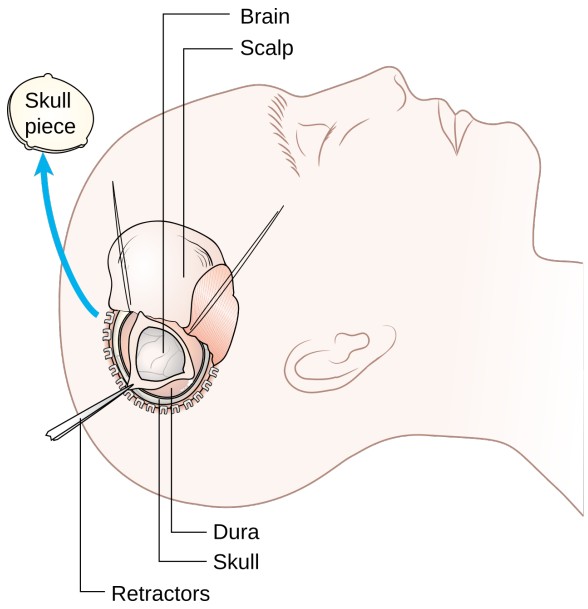


Figure 31: Craniotomy

Diagram showing the craniotomy procedure from Cancer Research UK[98]

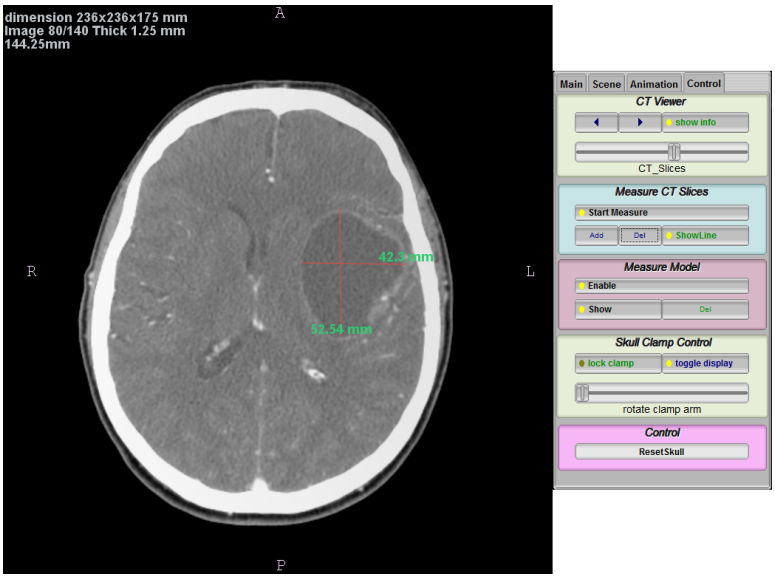
5.2 Technical Challenges

This simulation requires additional modules which are helpful to complete the simulation and are critical to teach certain knowledge for the procedure.

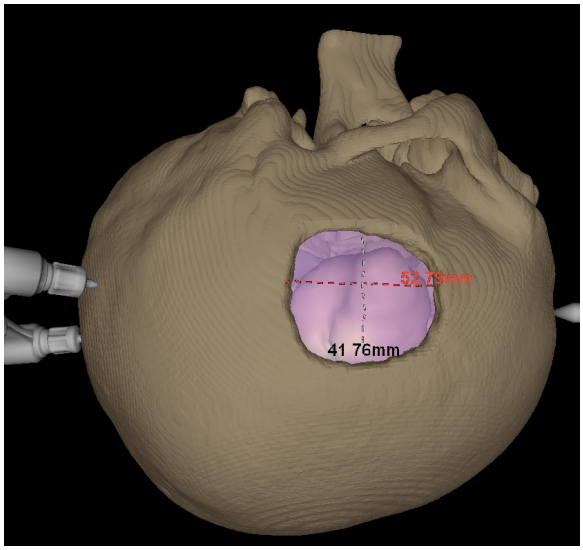
5.2.1 CT Viewer and Measurement

We provide a CT viewer module that allows users to scan through all CT slices to assess patient condition and design the bone flap (see Figure 32a). The CT viewer uses images from the original full resolution CT data for accuracy and fidelity.

Meta information is also shown in CT viewer including dimension, scale and slice thickness etc. Two measuring tools are provided to assist the planning phase. A simple ruler



(a)



(b)

Figure 32: Measurement module for craniotomy design

(a) CT viewer with controlling GUI, red line and green text are the measuring marks; (b) virtual skull model with user measurements. Notice how the CT measurement of the diameter of the tumor matches the virtual model.

tool could be used to measure distance between two points marked by the haptic stylus on a single CT slice.

Correspondingly, a soft tape tool is used to measure the length in the virtual scene. A model measurement module allows user to measure distance on the virtual model (see Figure 32b). A dashed line is drawn on the model while the accumulated distance is shown right next to the end of the line.

The CT viewer and the measuring tools enable trainees to register the CT to the virtual patient. This is a common clinical approach in planning and executing craniotomy. The obsession of this skill is critical in operating room (OR) when a navigation system is not available.

5.2.2 Skull Clamp Placement

A skull clamp is used to hold patient's head during an operation. The Mayfield skull clamp is the model we used in our simulation. The clamp uses 3 pins to lock the patient head and is connected to the operating bed through an adjustable arm.

During the simulation, the skull clamp is used to evaluate the choice of patient posture as well as the appropriate skull clamp positioning designed by the trainee (see Figure 33). Different craniotomy approaches require different patient posture for accessibility and specific clamp positions for stability. To achieve the optimal stability of the system, skull pins should be applied within an imaginary sweatband on the patient head and the dual skull pin on the rocker should have equal distance from the central line.

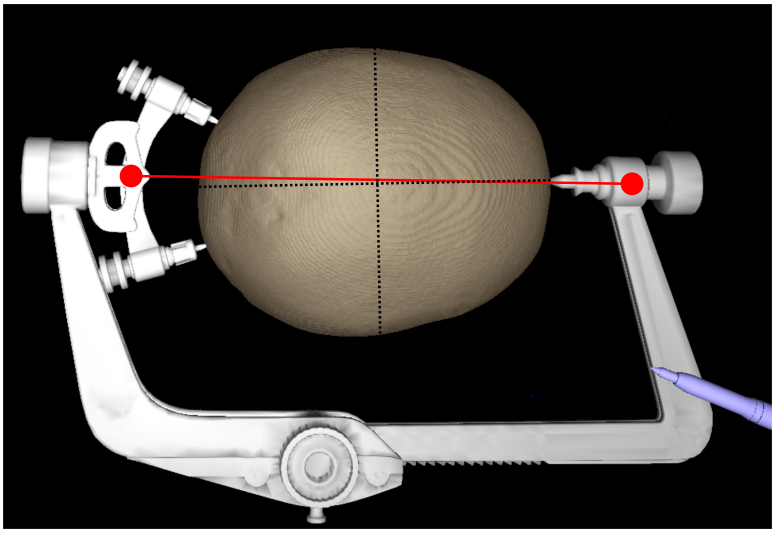


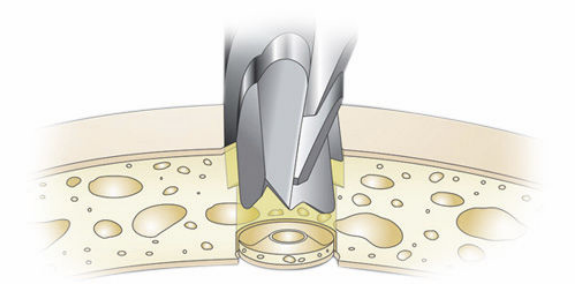
Figure 33: Skullclamp placement

Three pins of the Skull clamp should be placed on the opposite sides, with the line connecting the single pin to the 2-pin rocker arm crossing the center of the skull for best stability.

Our module simplifies the placement of the skull clamp. It requires trainees to assign the location of three pins on the virtual patient head. The module places the skull clamp to best match the user design under the mechanical constraints of the physical model. The trainee adjusts the patient posture and clamp arms before fixing the patient head in the virtual scene. The simplified model allows the assessment of trainees' knowledge of properly placing a skull clamp in clinical environments.

5.2.3 Bone Flap Removal

A bone flap is the piece of bone that is removed from the skull during a craniotomy procedure to grant access to intracranial anatomies. In operating room, surgeons creates



(a)

(b)

Figure 34: Cranial Perforator

(a) A Design of perforators that automatically turn off drilling after burr holes are created preventing potential damage to any delicate soft tissue[102]. (b) A commercially-available cranial perforator model from MERIDIAN.

small holes on the boundary of the bone flap using cranial perforators (see Figure 34). They then connect the burr holes using saws to isolate the bone flap from the rest of the skull. Jandial[101] explains this procedure in details.

The bone flap removal module removes a bone flap based on the boundary drawn by the trainee on the skull in the VR simulator, while we leave the training of tool manipulation

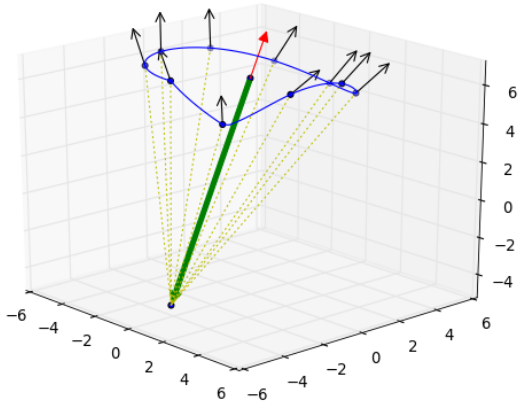


Figure 35: Pyramid generated from line segments

to physical simulators. We assume the boundary is the intersection of a pyramid and the skull, where the apex of the pyramid is inside the skull while the base is outside.

We take the length of each segment formed by two neighboring boundary anchor points as its weight. The weighted center of the boundary is calculated as the average of the coordinate of each anchor points multiplied by corresponding weight. Similarly, we could calculate the weighted average normal (see Figure 35).

Based on the center point and the weighted average normal we can define a line as the collection of all possible apexes for the pyramid. With a preset parameter as the distance between the center point to the apex, we can determine an apex. Using the fast voxel traversal between the apex and each point on the boundary we can identify all voxels intersecting the side of the pyramid. It is important to execute this ray casting on discrete

volume space in order to avoid skipping on the continuous WCS. Removing these voxels isolates the bone flap from the rest of the skull. We start a bi-directional fast voxel traversal at the center point along the normal. And we find the first "filled" point as the seed for our flood algorithm that removes the entire bone flap.

In practice, our bone flap removal produces a reliable and satisfying result. It greatly accelerates the simulation by eliminating the process of drilling and sawing. It serves the purpose of teaching core concepts of craniotomy design while leaving tool handling to physical simulators.

5.3 Simulation Steps

The free hand craniotomy simulation could be used to practice different types of craniotomy on patient-specific models. The application runs on ImmersiveTouch simulator and is designed to for medical educational programs on disciplines of craniotomy.

Trainees will be interacting with a virtual head model. The head model consists of skull, brain and other detailed anatomical structures. The skull model is the most important component for the craniotomy simulation. It is examined and manipulated by the operator to design a proper craniotomy. The brain and other anatomies in the simulation provides a more realistic experience as well as a reference to evaluate the bone flap design. The free hand craniotomy is performed through following steps listed in Table VI.

Step 1, 2, 3, 4 and 6 are performed by the trainee. Step 5 is automatically accomplished by the simulation.

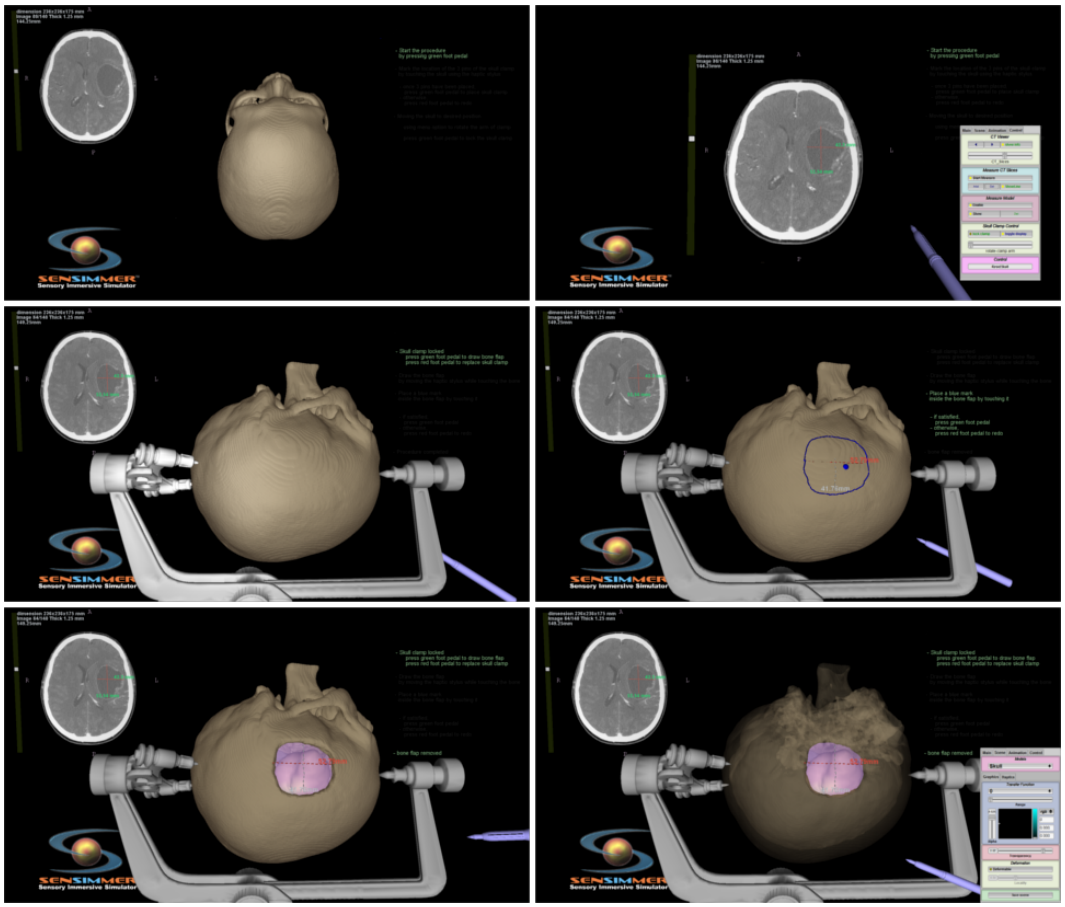


Figure 36: Craniotomy Simulation Procedure

The procedure of craniotomy simulation consists of: a) CT evaluation, b) CT measurement, c) skull clamp placement, d) skull measurement, e) bone flap removal, f) evaluation

Simulation Steps
1. Examine and evaluate the patient model
2. Draw measurements on CT slices
3. Place the skull clamp and adjust the patient posture
4. Measure the patient head and draw the bone flap design
5. Execute the bone flap removal
6. Validate and evaluate the bone flap design

TABLE VI: Craniotomy Simulation

Trainees first scan through all CT slices and visualize their bone flap designs. Then they determine the skull clamp placement and the patient head posture to facilitate their procedure. Trainees use the haptic device to draw the desired boundary and set the seed on the bone flap for flood removal. At step 5, based on the boundary drew by the trainee, the simulation attempts to isolate the bone flap through cutting the side of a pyramid geometry. The algorithm then use a flood fill algorithm to remove the whole isolated bone flap. We apply a Gaussian filter to the bone flap boundary to achieve a smooth surface on the cut. By this time the procedure is done and our trainee could use auxiliary tools to evaluate the outcome.

5.4 Performance Evaluation

After the completion of a procedure, the trainee and the instructor could review the performance in the simulator. The supervised evaluation inside the simulator assesses the skull clamp placement, the virtual patient posture and the bone flap design (see Figure 37). The performance evaluation is assisted through adjustments of rendering parameters and

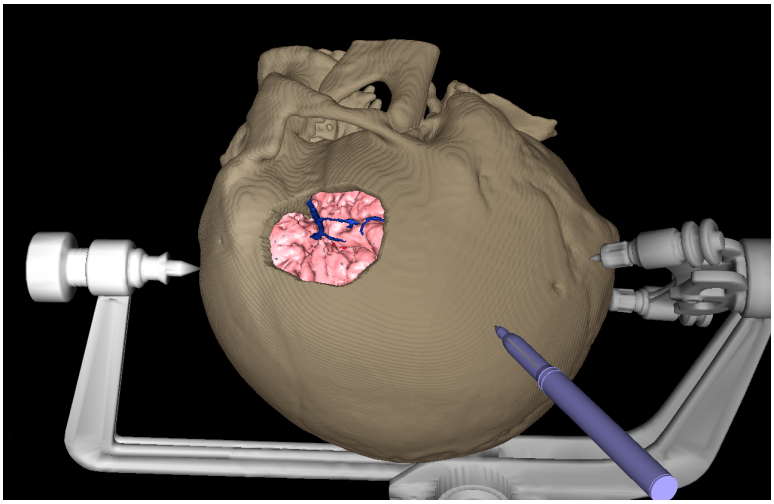


Figure 37: Anatomical Structures Inside Skull

Removing the bone flap exposes detailed anatomical structures underneath the skull. This information helps the evaluation of the bone flap design.

the cutting plane module. By excluding certain structures from computer graphics rendering, it is straightforward to show and to explain techniques relying on anatomical landmarks[103].

The simulation could export modified polygonal models into files in stereolithography (STL) format. The exported models could be opened in external software or 3D-printed as physical models.

Exported models facilitate the evaluation process. It mitigates the requirement of physical presence of the instructor during training sessions because the bone flap design could be evaluated asynchronously and remotely based on the exported models. This feature increases the accessibility and flexibility of the simulator.

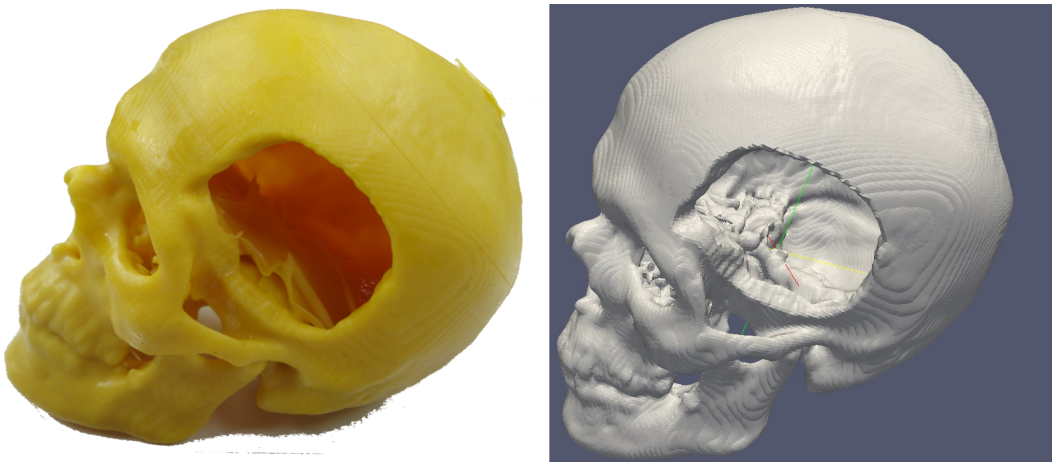


Figure 38: STL Model Export

Exported model in STL format viewed in Paraview on the right; 3D printed model on the left.

CHAPTER 6

LAMINECTOMY AND PEDICLE SCREW PLACEMENT SIMULATION

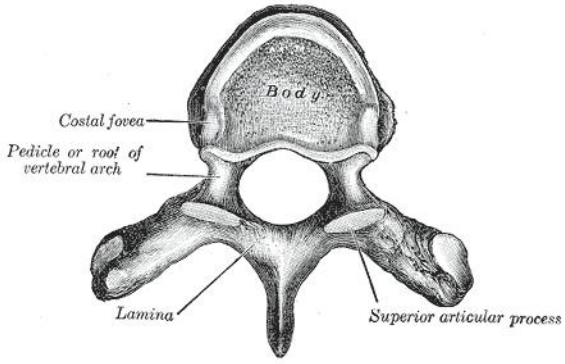
This simulation is developed using the Unity3D toolkit. The volume haptics module enables haptic feedback during interaction with spine models represented by volume data sets.

6.1 Clinical Procedures and Simulations

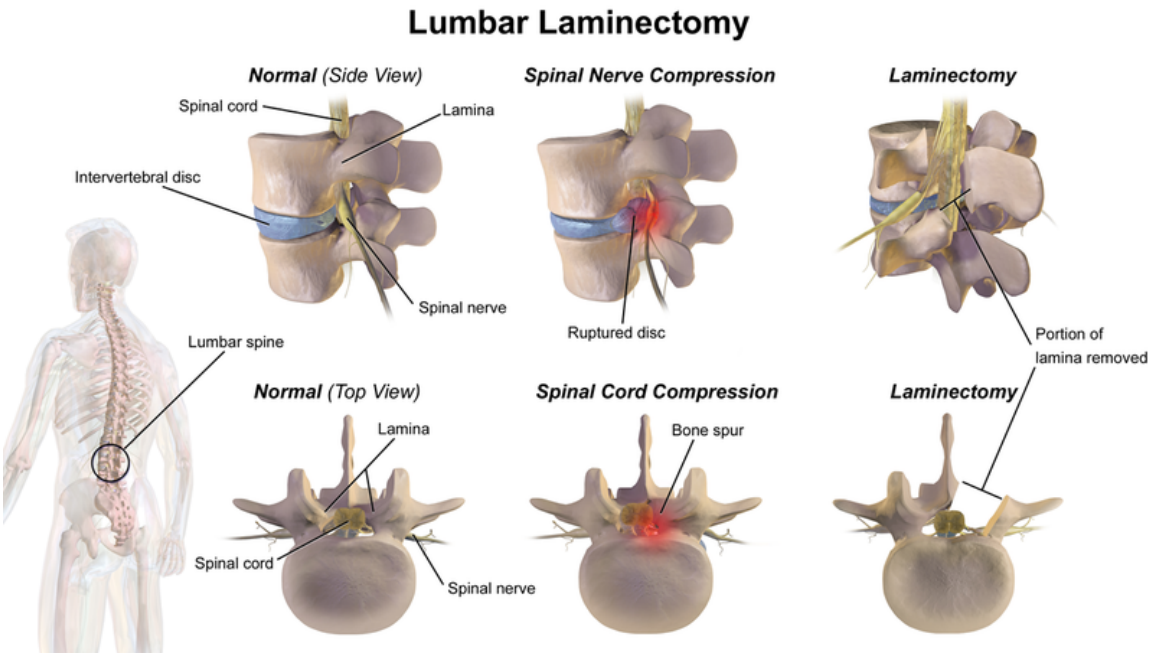
Laminectomy is a surgical procedure that removes a portion of vertebral bone named lamina (see Figure 39a). Lamina is removed to get rid of the constraint of the spinal canal in order to release the compression on the soft tissue and nerves (see Figure 39b). Laminectomy is also performed to allow access to deeper tissues inside the spinal canal.

An open technique laminectomy requires an incision on the back of the patient. The back muscles are pushed aside and the spinal vertebral bone is exposed. A conventional laminectomy removes the lamina and the spinous process using a variety of surgical tools including drills and rongeurs.

Complications during a laminectomy procedure are damages to spinal nerves, cerebrospinal fluid leaks, unsuccessful treatments and infections. Removal of substantial spinal bone often requires additional procedures like spinal fusions to reinforce the spine structure.



(a)



(b)

Figure 39: Laminectomy

(a) Upper view of human vertebra. Lamina is the posterior arch connecting the spinous process and the lateral pedicle. (b) Performing laminectomy to relieve stress on the spinal nerve and the spinal cord[104].

Open pedicle screw is one of the procedures to perform a spinal fusion. This procedure is to permanently join multiple vertebrae into one piece using screws and rods to avoid any relative motion.

The simulation uses volume models for vertebrae and polygon meshes for other anatomical structures including nerves, discs and muscles. It focuses on the correct angle and the depth for the screw insertion as well as appropriate vertebral bone removal. A line effect with friction simulates the insertion of the probe. The trainee confirms the insertion position and selects the screw length. An animation moves the screw and plants it into the designated track. The procedure is assisted with a fluoroscopy module which produces an X-ray image of the spine and the screws. By examining the fluoroscopy image at different angles, the trajectory of the screw could be revealed and evaluated. The final results could be assessed using an enhanced cross-section camera.

6.2 Technical Challenges

This simulation is an attempt to merge multiple procedures into a seamless simulation using the Unity3D toolkit. The integration of multiple procedures is difficult in the legacy Sensimmer SDK. We reimplemented the open pedicle screw insertion procedure and the laminectomy procedure into a single simulation using the Unity3D toolkit.

The spinal simulation requires some additional modules to provide functionalities needed during the procedures. We talk about the implementation of customized modules before we discuss the overall simulation.

6.2.1 Fluoroscopy Camera

Fluoroscopy camera uses X-ray imaging to guide surgeons during operations. As X-ray passes through tissues and bone its intensity is reduced by absorption and deflection. The 2D visualization of the receiving X-ray reflects the anatomical structures of the patient. The mathematical attenuation equation for transmitted intensity of X-ray beam can be expressed as:

$$dI(x) = -I(x) \cdot \mu \cdot dx \tag{6.1}$$

where: $I(x)$ is the radiation intensity at position x, μ is the linear attenuation coefficient, dx is the distance of traversed media and $dI(x)$ is the change in intensity.

A vertebra has different bone density across its structure. This results in a varying attenuation coefficient that is linearly to its volume intensity $\mu \propto \rho$.

The mechanism of fluoroscopy is similar to direct volume rendering where the camera shoots many beams and detects the accumulated attenuation at the receiving end. However, direct volume rendering for fluoroscopy suffers the limitations we discussed earlier for surgical simulation that it is expensive to update volume data set at run-time. It is also computationally expensive for graphics rendering.

We simplified the radiation attenuation model by ignoring the variances of scattering factors within the object. Once we assume μ to be constant within a single object we can integrate (Equation 6.1):

$$I = I_0 e^{-\mu x} \tag{6.2}$$

(Equation 6.2) shows the only factor that determines the attenuation is the travel distance inside the object.

We implemented a fluoroscopy camera using shaders. Shaders are extremely efficient for this scenario because the computation is lightweight for each ray. Moreover, it accommodates updates in volume by leveraging the isosurface that is updated during volume modification.

We implemented this technique using a two-pass shader scheme. At the first pass of rendering, the vertex shader outputs current depth to camera per vertex. The fragment shader calculates its weighted depth according to intensity factor of each object and changes the sign depending on whether it is facing towards or against the camera (see Figure 41). The depth is multiplicatively weighted by an intensity factor assigned to each object and additively blended to the destination texture. At the second pass of rendering, a graphics blit renders from a source texture to a destination texture. It reads the sum of weighted depth from the first pass and convert it to an illumination value to mimic an X-ray attenuation.

We noticed that on the attenuation coefficient from the bone density gives vertebrae a prominent edge on the fluoroscopic image. This is the results of the shell of a vertebra being much denser than its interior.

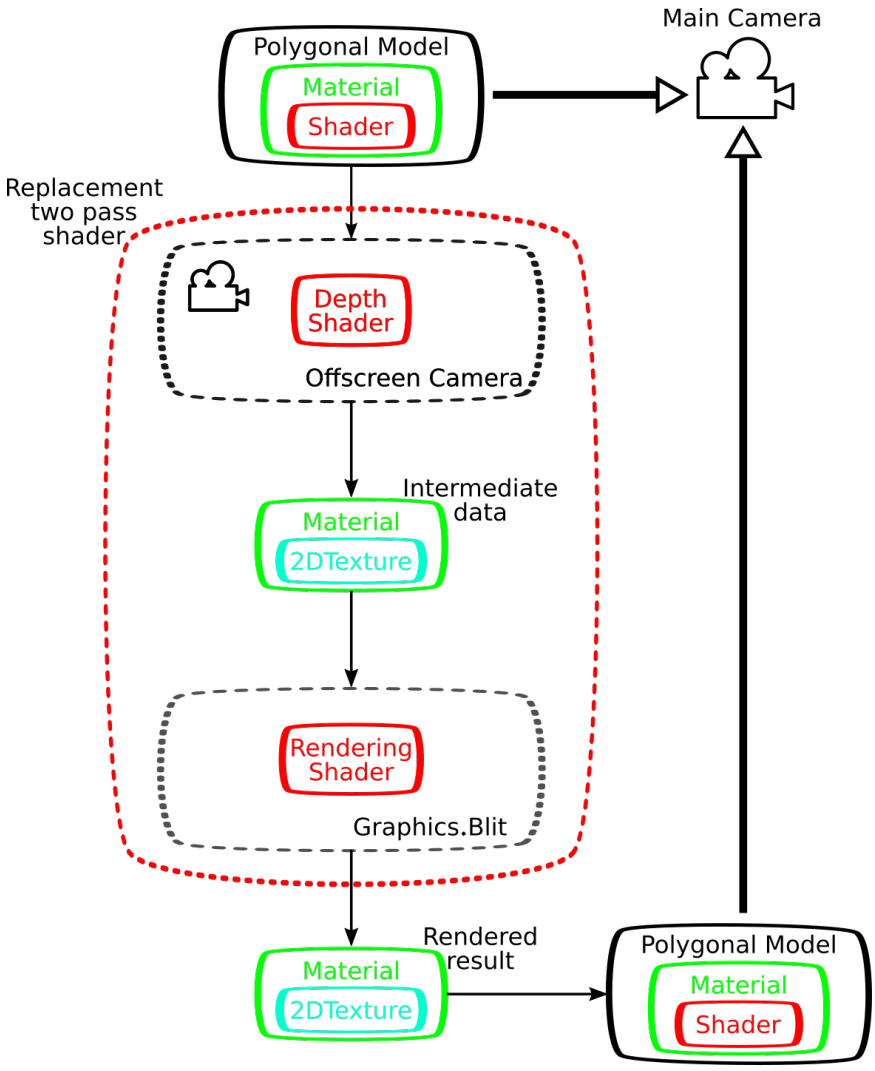


Figure 40: Replacement shader pipeline

Replacement shader attached to a secondary camera allows us to leverage shader functions without affecting graphics rendering of main camera. The texture rendering provides flexibility on post-processing as well as displaying.

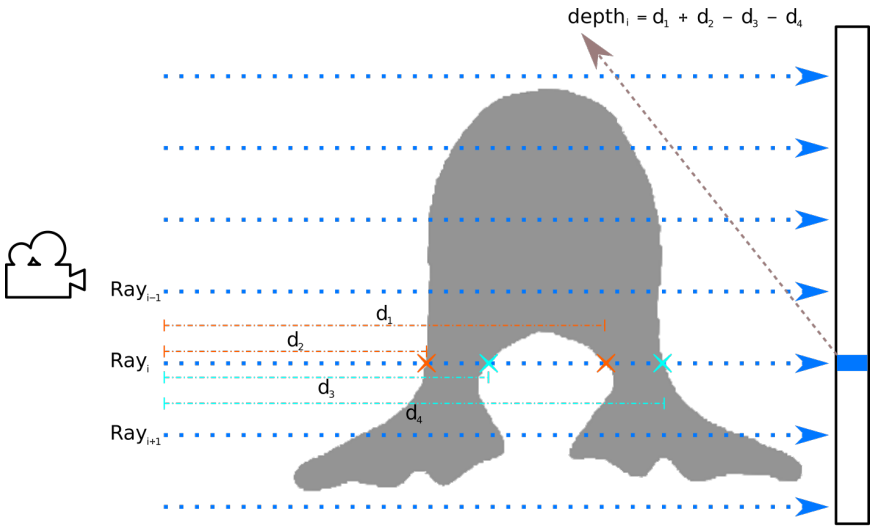


Figure 41: Depth shader

Each pixel stores the accumulated entering and exiting distance to the camera. The difference between these two gives us the accumulated distance traveled within an object. We introduce a individual multiplicative weight factor for each object in order to distinguish different objects in the fluoroscopy.

Because of the simplified radiation attenuation model, we ignore the variation of scalar intensity value from the original volume data set. It leads to the plain transition between air and bone. The absence of the defined boundary seen in a real X-ray imaging undermines the fidelity of this model.

We apply an illumination compensation for edge features to counteract the absence of volume intensity. Common edge enhancement techniques use image processing and geometry features. Image processing techniques use filtering or edge detection on the rendered image without leveraging information from the 3D model. The limitation is obvious that it can not detect edge features lost during transition from 3D models into 2D images.

Most shader-based edge techniques leverage multiple passes to extract depth information in order to produce consistent outlines drawn outside of the object. For our application these characteristics introduce error by expanding the boundary of the object and does not improve the fidelity of the visualization.

We added an illumination compensation based on Fresnel Effect. In computer graphics Fresnel Effect is the reflection seen on a surface viewing at certain angles. The implementation of Fresnel compensation uses the vertex shader and the fragment shader. The vertex shader outputs the surface normal in world coordinate system. The fragment shader uses the cross product of the interpolated normal and the camera direction as the compensation factor. To easily adjust the blending of the edge enhancement we use two channels to store weighted depth and compensation factor separately. The comparison of the spine rendered

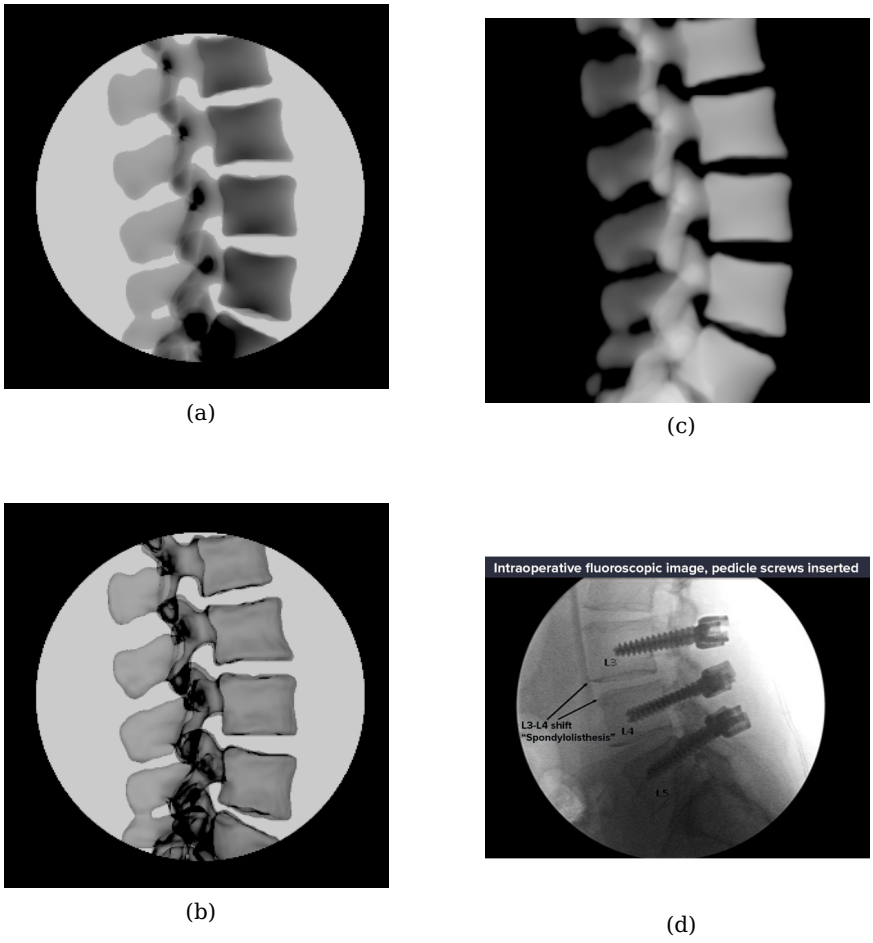


Figure 42: Fluoroscopy comparison

(a) Fluoroscopic image generated using the simplified attenuation model without edge enhancement; (b) Fluoroscopic image generated with edge enhancement. (c) Fluoroscopic image generated with direct volume rendering; (d) Intraoperative fluoroscopic image;

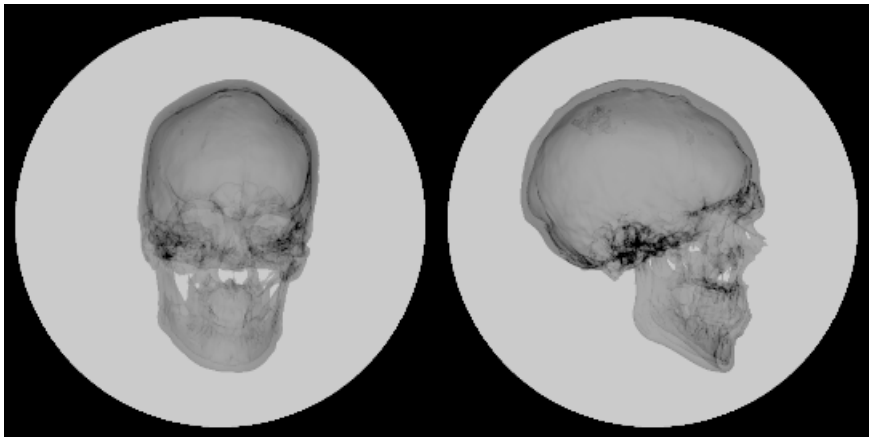


Figure 43: Edge Enhancement Fluoroscopy module on Skull

The shader implementation of the simplified attenuation model with the edge enhancement scheme renders satisfying images of skulls and spines.

with different fluoroscopy approach could be seen from Figure 42. Our edge enhancement scheme also works on anatomical structures other than spine (see Figure 43).

6.2.2 Cross-section Camera

Surgeons use CT for postoperative evaluations after spinal procedures. We provide a module that shows the cross-section topology of objects in the virtual scene to visualize an augmented CT imaging.

The cross-section camera is used to show the sculptured spine and the implanted screws. The goal is to highlight the inconsistency and potential error that the trainee might have made during the simulation. Because polygonal models consists of triangles only on the surface, showing cross-section of polygonal model would require generating new tri-

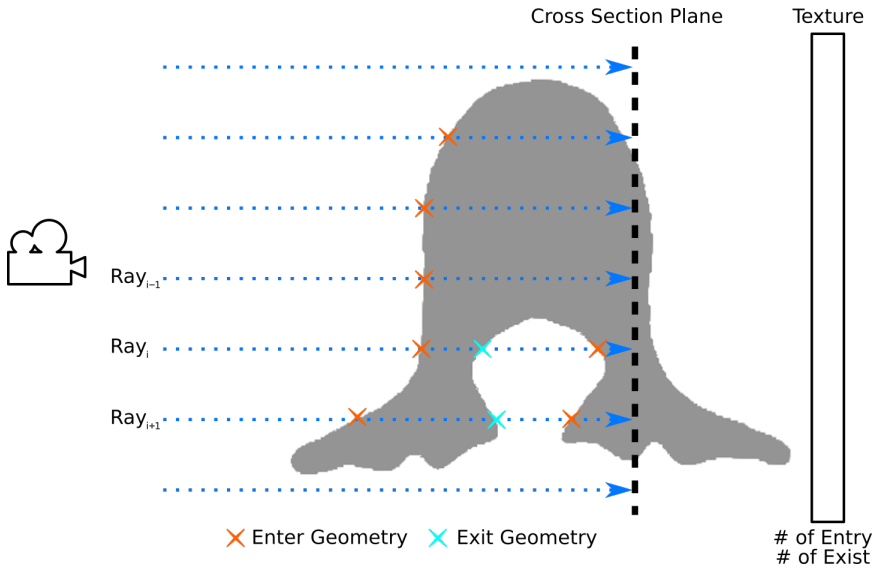


Figure 44: Cross section camera

The augmented cross-section shader use the cross-section plane to define its view frustum. Each pixel stores the total entry point and exit point to determine whether current pixel is enclosed by a manifold.

angles on the cross-section plane to fill the void interior, which is a quite complicated process. Instead, we used shaders to mask the interior of the object on a cross-section and we colored each pixel according to its layer mask.

The implementation is similar to the fluoroscopic camera module. But instead of accumulating the depth of each polygon, we count the number of the entrance and the exit along each ray using the facing in fragment shader (see Figure 44). Then we color different pixels accordingly to highlight the region of screws outside of bone. Figure 45 shows the result.

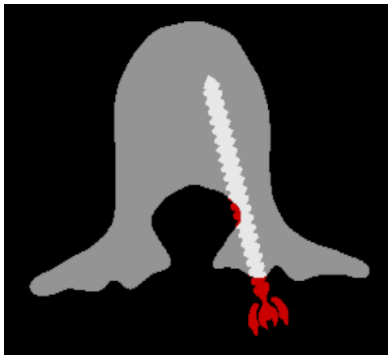


Figure 45: Cross section

Color coded cross section. Area with gray color is interior of a vertebra, while the white color indicates the interior of a screw. Notice how the red region highlights the penetration that would otherwise be hard to recognize.

6.2.3 Probe Insertion

We mimic the probe insertion effect using a constrained line effect provided by Haptics API.

During the insertion, consistent resistance is felt while the probe is being inserted to create the screw track[105]. A loss of resistance delivers a sudden plunge and indicates a penetration of the pedicle.

We use a line effect from the Haptics API to constrain the virtual instrument on a fixed line segment after insertion. By controlling the length of the line segment we generate the consistent resistance only when probe is lengthening the track inside pedicle.

Detecting pop-in and pop-out event of the pedicle probe allow us to temporarily disable the resistance when the tip of the probe went outside of the pedicle. Thus recreating a realistic sudden plunge when penetration happens.

6.2.4 Drilling

Multiple models for real-time volume modification have been proposed by Chen[106]. Our framework utilizes volume data set directly for haptics rendering. With progressive volume modification, there is no abrupt interruption to neither the collision detection nor the haptic force calculation. During drilling, we reduce voxel intensity on voxels close to the tip of the haptic stylus. The isosurface extraction of graphics module in our toolkit breaks the volume into small chunks. This allows the graphics module to only update isosurface of chunks which are being modified instead of updating the whole isosurface which is very expensive. For the worst case, 8 chunks are updated and needs to regenerate their isosurface. With an appropriate chunk size, this could easily be done at graphics thread running at 30 FPS.

6.3 Simulation Steps

The spinal simulation use a state machine design pattern, where trainees switch between different tools which enable them to perform a variety of tasks on the virtual patient model.

Trainees start with the laminectomy procedure where they use a burr drill and a rongeur to remove a portion of a vertebra that produces excessive pressure on spinal cord or nerves. The application then switch to the pedicle screw insertion, where a pedical

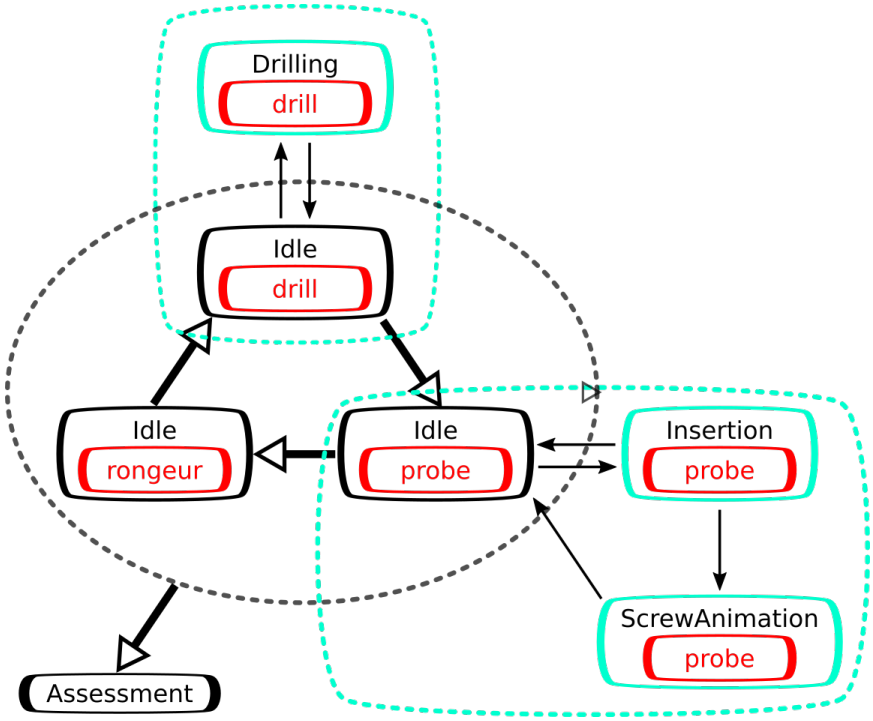


Figure 46: State Machine for Spinal Simulation

The state machine implementation allows maximum freedom to user interactions in a simulation. This programming pattern simplifies the control of a simulation combining multiple procedures.

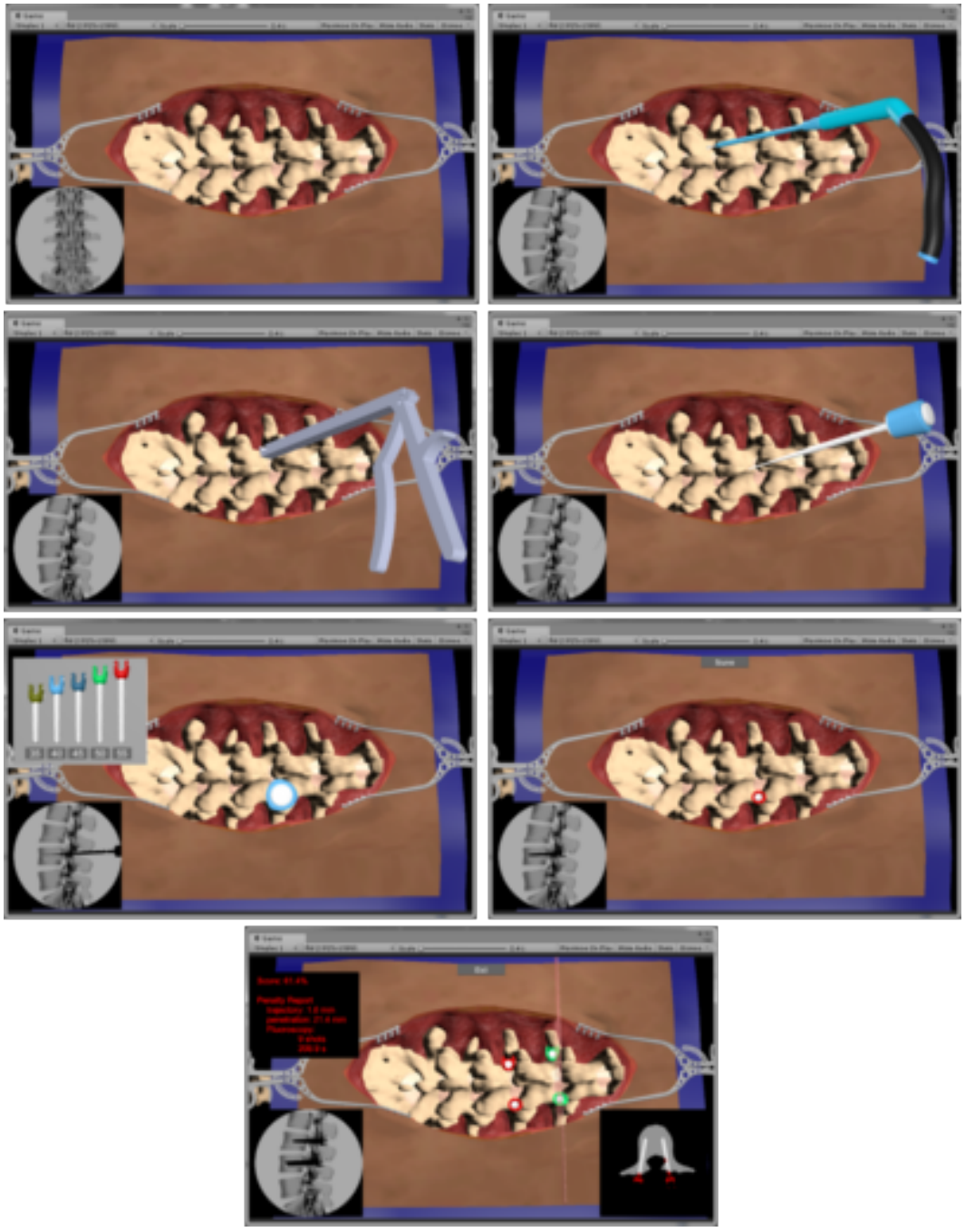


Figure 47: Laminectomy and Pedicle Screw Placement Simulation Procedure

The procedures for laminectomy and pedicle screw placement. The figures above shows three tools: a burr drill, a rongeur and a pedicle probe. Tools are used during the spinal bone removal, the pedicle probe insertion and the screw placement.

probe is used to create the pedicle track that guides the trajectory of a screw implant. Fluoroscopic guidance is commonly used at this time to avoid medial penetration of the spinal canal or anterior penetration of the vertebral body cortex[107–109].

We use animations for the screw placement. A screw with selected length is inserted through the trajectory created. The real-time fluoroscopic camera and cross-section view are used to evaluate the performance.

6.4 Performance Evaluation

The automatic evaluation assesses the screw insertion combining two factors, the trajectory of the implanted screw and consistency between the depth of the track created during probe penetration and the screw length.

We have two pairs of optimal trajectories for each vertebra. For a given screw placement we find the corresponding trajectory with the closest entry point. The insertion is quantitatively evaluated by adding the two distances, one between the optimal entry point and the entry point of the insertion, the other between the target points. The consistency between the probe penetration and screw selection is measured by the difference between the penetration depth and length of the selected screw.

A weighted total score reflects a general assessment of the procedure. While the cross-section tool and fluoroscopic image provide visualized hints during the evaluation of the trajectory.

CHAPTER 7

CONCLUSION

We summarize the contribution of this dissertation and discuss future work to continue and extend the scope of our research.

7.1 Summary of Contribution

We presented a versatile and robust surface-based volume haptics rendering algorithm. Our technique extracts reliable surface features from volumetric data sets. It runs efficiently on large volumetric data set and scales linearly with voxel density. The approach requires no preprocessing. The algorithm could be used with its own collision detection and force rendering as a full package of volume haptics module. It could also be used as a lightweight volume haptics extension for a polygonal-based haptics rendering scheme.

The robust surface topology of volume objects distinguishes our approach from existing volume haptics algorithms, which focus on providing tactile feedback as part of multisensory interaction[110]. Our endeavor to incorporate the volume haptics algorithm in heterogeneous virtual scene with mixed polygonal and volumetric models increases the flexibility for client applications. Applications could utilize available volumetric data and leverage the advantages of both model representations.

Our volume haptics approach adapts to various pipelines and could be easily integrated to existing frameworks and applications. The integration of our volume haptics scheme to

the legacy Sensimmer SDK and the Unity3D toolkit validates the adaptability to different use scenarios.

The volume haptics implementation in Unity3D toolkit utilizes a variety of native Unity3D functionalities, which reduce the integration effort. The volume-haptics-enabled Unity3D toolkit facilitates the development of VR-based surgical simulations. The toolkit provides a baseline for development of general-purpose surgical simulations. Its modularized design grants users freedom to customize existing functionalities and to plug in their own extensions. Moreover, universal modules could be reused among applications. The assets feature in Unity platform makes it easy to share and distribute modules among community. Our experience with the fluoroscopy module and the line effect are examples, where both modules are developed for spinal simulation initially and later adopted by other applications and integrated as part of the toolkit.

7.2 Future work

Due to the scope of our research, the potential of our approach and toolkit have not been fully explored in this dissertation.

The volume haptics algorithm we presented has the efficiency to handle large volumetric data set and many collision detection iterations. It is worth to explore multiple points collision detection and 6-DOF force rendering using our implicit surface volume haptics algorithm. Established research[111,112] exploring multi-point collision and 6-DOF force rendering shares common methodologies from the proxy-based haptics rendering. Our algorithm could be extended in similar way to enable 6-DOF force feedback.

Moreover, our current Unity3D toolkit stores volume in CPU memory for applications involving volume modification. This works well for our explicit surface volume haptics and volume visualization using isosurface. However, a break down of volume data with each chunk stored in GPU as texture could expand the capabilities of our framework. Partitioned texture enables us to update only a small portion of the entire volumetric data set allowing real-time modification. GPU allows parallel processing and direct volume rendering for graphics rendering. Last but not least, the multi-point collision detection that utilizes multiple ray casting could leverage the parallelism from the GPU architecture.

The cranial and spinal surgical simulations developed by us are validations of our work as well as practical tools for residents' training. However, an extensive evaluation and study of surgical simulation is necessary in order to exploit the merits of haptics-enabled VR-based simulations. Future study should involve a pilot study to explore an efficient curriculum combining various surgical simulators.

7.3 Conclusions

Volume haptics is one of the critical components in VR-based surgical simulations while the research community has not yet found a universal approach that is suitable for diverse scenarios. Our presented volume haptics approach focuses on features and characteristics desired for surgical simulations that cannot be found in existing techniques. The adoption of the fast ray traversal and the combination with smooth filtering technique distinguish our work from other techniques. Instead of preserving high precision and smooth collision through the whole process, we introduced artifacts at the first phase and alleviated the im-

pact on the output later. It allows our technique to improve the computational complexity significantly while maintaining a reliable surface representation.

The Unity3D toolkit with the volume haptics integration is a promising framework. It facilitates the development of our surgical simulation. The commercial platform reduces the overhead from maintenance of an in-house close source framework. Furthermore, Unity3D provides native modules as well as many community assets to support the deployment of commercial hardware like HMD and gesture recognition devices. Lastly, it simplifies the distribution of modules and encourages collaborations among the community. Admittedly, a commercial platform inevitably introduces restriction on the pipeline and other design decisions of our toolkit. Our experience during the development of simulations with the Unity3D toolkit shows the benefits are well worth these disadvantages.

With the continuous improvement of computational power and the development of user interaction hardware, the fidelity and capability of VR environments have come a long way in the last decade. Healthcare system utilizes computerized imaging for diagnosis and surgical planning in clinical environment. Exploring the advantages of augmented and immersed haptics-enabled VR environments over traditional 2D monitors is of great value to the improvements of the system. Surgical simulators are one of the first applications where the VR technology is introduced to the medical system to compensate the short comings of traditional tools like physical simulators. The simulations we developed for residents' training show promising value for the surgical training. Advantages of VR-based environments should be further explored to benefit clinical diagnosis and surgical rehearsals.

Research that links simulations to operating room should also get more attention. We have confidence in the positive influence on healthcare system from the advance of technology in VR and haptics.

CITED LITERATURE

1. Kenneth Salisbury and Christopher Tarr. Haptic rendering of surfaces defined by implicit functions. In *ASME Dynamic Systems and Control Division*, volume 61, pages 61–67, 1997.
2. Karljohan Lundin, Björn Gudmundsson, and Anders Ynnerman. General proxy-based haptics for volume visualization. In *Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint*, pages 557–560. IEEE, 2005.
3. Silvio H Rizzi, Cristian J Luciano, and P Pat Banerjee. Haptic interaction with volumetric datasets using surface-based haptic libraries. In *Haptics Symposium, 2010 IEEE*, pages 243–250. IEEE, 2010.
4. Michele Larobina and Loredana Murino. Medical image file formats. *Journal of digital imaging*, 27(2):200, 2014.
5. Pat Banerjee, Mengqi Hu, Rahul Kannan, and Srinivasan Krishnaswamy. A semi-automated approach to improve the efficiency of medical imaging segmentation for haptic rendering. *Journal of Digital Imaging*, pages 1–9, 2017.
6. WEL Grimson, RJFA Kikinis, Ferenc A Jolesz, and PM Black. Image-guided surgery. *Scientific American*, 280(6):54–61, 1999.
7. Robert Stone. Haptic feedback: A brief history from telepresence to virtual reality. *Haptic Human-Computer Interaction*, pages 1–16, 2001.
8. Steven Martin and Nick Hillier. Characterisation of the novint falcon haptic device for application as a robot manipulator. In *Australasian Conference on Robotics and Automation (ACRA)*, pages 291–292, 2009.
9. Takahiro Endo, Haruhisa Kawasaki, Tetsuya Mouri, Yasuhiko Ishigure, Hisayuki Shimomura, Masato Matsumura, and Kazumi Koketsu. Five-fingered haptic interface robot: Hiro iii. *IEEE Transactions on Haptics*, 4(1):14–27, 2011.
10. Xiaochi Gu, Yifei Zhang, Weize Sun, Yuanzhe Bian, Dao Zhou, and Per Ola Kristensson. Dexmo: An inexpensive and lightweight mechanical exoskeleton for motion capture and force feedback in vr. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 1991–1995. ACM, 2016.
11. Thomas H Massie and J Kenneth Salisbury. The phantom haptic interface: A device for probing virtual objects. In *Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems*, volume 55, pages 295–300. Chicago, IL, 1994.

12. Andreas Tobergte, Patrick Helmer, Ulrich Hagn, Patrice Rouiller, Sophie Thielmann, Sébastien Grange, Alin Albu-Schäffer, François Conti, and Gerd Hirzinger. The sigma. 7 haptic interface for mirosurge: A new bi-manual surgical console. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3023–3030. IEEE, 2011.

13. QUANSER. Quanser. `http://www.quanser.com/`. [Online; accessed Aug 19, 2017].

14. haption. haption. `https://www.haption.com/site/index.php/en/`. [Online; accessed Aug 19, 2017].

15. Jonathan Steuer. Defining virtual reality: Dimensions determining telepresence. *Journal of communication*, 42(4):73–93, 1992.

16. Kenneth Salisbury, Francois Conti, and Federico Barbagli. Haptic rendering: introductory concepts. *Computer Graphics and Applications, IEEE*, 24(2):24–32, 2004.

17. Miguel A Otaduy and Ming C Lin. Introduction to haptic rendering. *ACM SIGGRAPH 2005 Courses*, page 3, 2005.

18. Zilles, Craig B and Salisbury, J Kenneth. A constraint-based god-object method for haptic display. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 146–151. IEEE, 1995.

19. Diego C Ruspini, Krasimir Kolarov, and Oussama Khatib. The haptic display of complex graphical environments. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 345–352. ACM Press/Addison-Wesley Publishing Co., 1997.

20. Chih-Hao Ho, Cagatay Basdogan, and Mandayam Srinivasan. Efficient point-based rendering techniques for haptic display of virtual objects. *Presence*, 8(5):477–491, 1999.

21. Laehyun Kim, Gaurav S Sukhatme, and Mathieu Desbrun. A haptic-rendering technique based on hybrid surface representation. *Computer Graphics and Applications, IEEE*, 24(2):66–75, 2004.

22. Brandon Itkowitz, Josh Handley, and Weihang Zhu. The openhapticsÂŹ toolkit: a library for adding 3d touchÂŹ navigation and haptics to graphics applications. In *Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint*, pages 590–591. IEEE, 2005.

23. SenseGraphics AB. H3d api-haptics software development platform, 2011.

24. Francois Conti, D Morris, F Barbagli, and C Sewell. Chai 3d. *Online: http://www.chai3d. org*, 2006.

25. Randolph H Steadman, Wendy C Coates, Yue Ming Huang, Rima Matevosian, Baxter R Larmon, Lynne McCullough, and Danit Ariel. Simulation-based training is superior to problem-based learning for the acquisition of critical assessment and management skills*. *Critical care medicine*, 34(1):151–157, 2006.

26. Malcolm Cox, David M Irby, Richard K Reznick, and Helen MacRae. Teaching surgical skillsÂŮchanges in the wind. *New England Journal of Medicine*, 355(25):2664–2669, 2006.

27. James D Michelson. Simulation in orthopaedic education: an overview of theory and practice. *The Journal of Bone & Joint Surgery*, 88(6):1405–1411, 2006.

28. Pamela Tuoriniemi and Darlene Schott-Baer. Implementing a high-fidelity simulation program in a community college setting. *Nursing Education Perspectives*, 29(2):105–109, 2008.

29. William C McGaghie, S Barry Issenberg, Emil R Petrusa, and Ross J Scalese. A critical review of simulation-based medical education research: 2003–2009. *Medical education*, 44(1):50–63, 2010.

30. P Das, T Goyal, A Xue, S Kalatoor, and D Guillaume. Simulation training in neurological surgeryid. *Austin J Neurosurg*, 1(1):1004, 2014.

31. Nathan R Selden, Thomas C Origitano, Costas Hadjipanayis, and Richard Byrne. Model-based simulation for early neurosurgical learners. *Neurosurgery*, 73:S15–S24, 2013.

32. Darlene A Lobel, J Bradley Elder, Clemens M Schirmer, Mark W Bowyer, and Ali R Rezai. A novel craniotomy simulator provides a validated method to enhance education in the management of traumatic brain injury. *Neurosurgery*, 73:S57–S65, 2013.

33. M Abe, K Tabuchi, M Goto, and A Uchino. Model-based surgical planning and simulation of cranial base surgery. *Neurologia medico-chirurgica*, 38(11):746–751, 1998.

34. Richard Sarle, Ashutosh Tewari, Alok Shrivastava, James Peabody, and Mani Menon. Surgical robotics and laparoscopic training drills. *Journal of Endourology*, 18(1):63–67, 2004.

35. Kevin Cleary and Charles Nguyen. State of the art in surgical robotics: clinical applications and technology challenges. *Computer Aided Surgery*, 6(6):312–328, 2001.

36. Richard M Satava. Surgical robotics: the early chronicles: a personal historical perspective. *Surgical Laparoscopy Endoscopy & Percutaneous Techniques*, 12(1):6–16, 2002.

37. Pascal Jabbour and Nohra Chalouhi. Simulation-based neurosurgical training for the presigmoid approach with a physical model. *Neurosurgery*, 73:S81–S84, 2013.

38. Limbs & Things Inc. Lumber puncture & epidural simulator mk2. `http://assets.limbsandthings.com//resized/b46fcc4fe6fee753524ef463d5d07e4006e44c5a.jpg`, year unspecified. [Online; accessed December 1, 2015].

39. David Fuerst, Marianne Hollensteiner, and Andreas Schrempf. A novel augmented reality simulator for minimally invasive spine surgery. In *Proceedings of the 2014 Summer Simulation Multiconference*, page 28. Society for Computer Simulation International, 2014.

40. James Harrop, Ali R Rezai, Daniel J Hoh, George M Ghobrial, and Ashwini Sharan. Neurosurgical training with a novel cervical spine simulator: posterior foraminotomy and laminectomy. *Neurosurgery*, 73:S94–S99, 2013.

41. James B Walker, Eddie Perkins, and H Louis Harkey. A novel simulation model for minimally invasive spine surgery. *Neurosurgery*, 65(6):ons188–ons195, 2009.

42. V Uppal, RJ Kearns, and EM McGrady. Evaluation of m43b lumbar puncture simulator-ii as a training tool for identification of the epidural space and lumbar puncture*. *Anaesthesia*, 66(6):493–496, 2011.

43. Ken Brodlie, Nuha El-Khalili, and Ying Li. Using web-based computer graphics to teach surgery. *Computers & Graphics*, 24(1):157–161, 2000.

44. Antonio Bernardo, Mark C Preul, Joseph M Zabramski, and Robert F Spetzler. A three-dimensional interactive virtual dissection model to simulate transpetrous surgical avenues. *Neurosurgery*, 52(3):499–505, 2003.

45. Richard Q Van der Linde, Piet Lammertse, Erwin Frederiksen, and B Ruiter. The hapticmaster, a new high-performance haptic interface. In *Proc. Eurohaptics*, pages 1–5, 2002.

46. Lionel Birglen, Clément Gosselin, Nicolas Pouliot, Bruno Monsarrat, and Thierry Laliberté. Shade, a new 3-dof haptic device. *Robotics and Automation, IEEE Transactions on*, 18(2):166–175, 2002.

47. Hani R Malone, Omar N Syed, Michael S Downes, Anthony L D'Ambrosio, Donald O Quest, and Michael G Kaiser. Simulation in neurosurgery: a review of computer-based simulation environments and their surgical applications. *Neurosurgery*, 67(4):1105–1116, 2010.

48. NeuroTouch. Neurotouchcranio. `http://www.neurotouch.ca/obj/images/simulators/cranio.jpg`, 2011. [Online; accessed December 1, 2015].

49. ImmersiveTouch. Immersivetouch. `http://www.immersivetouch.com/images/ImmersiveTouch_with-I-PAD.png`, ca. 2005. [Online; accessed November 11, 2015].

50. SPIROS Sgouros, Kalyan Natarajan, A Richard Walsh, Edward B Rolfe, and Anthony D Hockley. Computer simulation of a neurosurgical operation: craniotomy for hypothalamic hamartoma. *ChildÂŠs Nervous System*, 14(7):322–327, 1998.

51. George KC Wong, Canon XL Zhu, Anil T Ahuja, and Wai S Poon. Craniotomy and clipping of intracranial aneurysm in a stereoscopic virtual reality environment. *Neurosurgery*, 61(3):564–569, 2007.

52. Sébastien Delorme, Denis Laroche, Robert DiRaddo, and Rolando F Del Maestro. Neurotouch: a physics-based virtual simulator for cranial microneurosurgery training. *Neurosurgery*, 71:ons32–ons42, 2012.

53. Gail Rosseau, Julian Bailes, Rolando del Maestro, Anne Cabral, Nusrat Choudhury, Olivier Comas, Patricia Debergue, Gino De Luca, Jordan Hovdebo, Di Jiang, et al. The development of a virtual simulator for training neurosurgeons to perform and perfect endoscopic endonasal transsphenoidal surgery. *Neurosurgery*, 73:S85–S93, 2013.

54. Ming-Dar Tsai, Ming-Shium Hsieh, and Shyan-Bin Jou. Virtual reality orthopedic surgery simulator. *Computers in biology and medicine*, 31(5):333–351, 2001.

55. Chee-Kong Chui, Jackson SK Ong, Zheng-Yi Lian, Zhenlan Wang, Jeremy Teo, Jing Zhang, Chye-Hwang Yan, Sim-Heng Ong, Shih-Chang Wang, Hee-Kit Wong, et al. Haptics in computer-mediated simulation: training in vertebroplasty surgery. *Simulation & Gaming*, 37(4):438–451, 2006.

56. Jaime Gasco, Thomas J Holbrook, Achal Patel, Adrian Smith, David Paulson, Alan Muns, Sohum Desai, Marc Moisi, Yong-Fan Kuo, Bart Macdonald, et al. Neurosurgery simulation in residency training: feasibility, cost, and educational benefit. *Neurosurgery*, 73:S39–S45, 2013.

57. Allison M Okamura. Haptic feedback in robot-assisted minimally invasive surgery. *Current opinion in urology*, 19(1):102, 2009.

58. Brian T Bethea, Allison M Okamura, Masaya Kitagawa, Torin P Fitton, Stephen M Cattaneo, Vincent L Gott, William A Baumgartner, and David D Yuh. Application of haptic feedback to robotic surgery. *Journal of Laparoendoscopic & Advanced Surgical Techniques*, 14(3):191–195, 2004.

59. Pinyo Puangmali, Kaspar Althoefer, Lakmal D Seneviratne, Declan Murphy, and Prokar Dasgupta. State-of-the-art in force and tactile sensing for minimally invasive surgery. *IEEE Sensors Journal*, 8(4):371–381, 2008.

60. Allison M Okamura. Methods for haptic feedback in teleoperated robot-assisted surgery. *Industrial Robot: An International Journal*, 31(6):499–508, 2004.

61. Anthony R Lanfranco, Andres E Castellanos, Jaydev P Desai, and William C Meyers. Robotic surgery: a current perspective. *Annals of surgery*, 239(1):14, 2004.

62. Fabrizia Mantovani, Gianluca Castelnuovo, Andrea Gaggioli, and Giuseppe Riva. Virtual reality training for health-care professionals. *CyberPsychology & Behavior*, 6(4):389–395, 2003.

63. O AJ van der Meijden and MP Schijven. The value of haptic feedback in conventional and robot-assisted minimal invasive surgery and virtual reality training: a current review. *Surgical endoscopy*, 23(6):1180–1190, 2009.

64. Cristian Luciano, Pat Banerjee, Lucian Florea, and Greg Dawe. Design of the immersivetouchÂŹ: a high-performance haptic augmented virtual reality system. In *Proceedings of the 11th international conference on human-computer interaction, Las Vegas, Nevada*, 2005.

65. Young Soo Park, Xiaorui Zhao, and Scott Korthals. Simulation of augmented telerobotic operation. In *Optomechatronic Technologies (ISOT), 2014 International Symposium on*, pages 242–246. IEEE, 2014.

66. Silvio H Rizzi, P Pat Banerjee, and Cristian J Luciano. Automating the extraction of 3d models from medical images for virtual reality and haptic simulations. In *Automation Science and Engineering, 2007. CASE 2007. IEEE International Conference on*, pages 152–157. IEEE, 2007.

67. Silvio H Rizzi. *Volume-based graphics and haptics rendering algorithms for immersive surgical simulation*. PhD thesis, University of Illinois at Chicago, 2013.

68. Cristian Luciano, Pat Banerjee, G Michael Lemole, and Fady Charbel. Second generation haptic ventriculostomy simulator using the immersivetouchâĎć system. *Studies in health technology and informatics*, 119:343, 2005.

69. Karljohan Lundin, Anders Ynnerman, and Björn Gudmundsson. Proxy-based haptic feedback from volumetric density data. In *Proceedings of the Eurohaptic Conference*, pages 104–109, 2002.

70. Mark Edward Barry. Direct extraction of normal maps from volume data. 2007.

71. Roni Yagel, Daniel Cohen, and Arie Kaufman. Normal estimation in 3 d discrete space. *The visual computer*, 8(5-6):278–291, 1992.

72. Marc Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics (TOG)*, 9(3):245–261, 1990.

73. John Danskin and Pat Hanrahan. Fast algorithms for volume ray tracing. In *Proceedings of the 1992 workshop on Volume visualization*, pages 91–98. ACM, 1992.

74. Milos Sramek and Arie Kaufman. Fast ray-tracing of rectilinear volume data using distance transforms. *Visualization and Computer Graphics, IEEE Transactions on*, 6(3):236–252, 2000.

75. Steven Parker, Michael Parker, Yarden Livnat, Peter-Pike Sloan, Charles Hansen, and Peter Shirley. Interactive ray tracing for volume visualization. In *ACM SIGGRAPH 2005 Courses*, page 15. ACM, 2005.

76. Klaus Engel, Martin Kraus, and Thomas Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 9–16. ACM, 2001.

77. Robert A Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *ACM Siggraph Computer Graphics*, 22(4):65–74, 1988.

78. Philippe Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 451–458. ACM, 1994.

79. David Laur and Pat Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. *ACM SIGGRAPH Computer Graphics*, 25(4):285–288, 1991.

80. Stéphane Redon, Abderrahmane Kheddar, and Sabine Coquillart. Fast continuous collision detection between rigid bodies. In *Computer graphics forum*, volume 21, pages 279–287. Blackwell Publishing, Inc, 2002.

81. John Amanatides and Andrew Woo. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87, page 10, 1987.

82. William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

83. Sarah FF Gibson. Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 888–898. Springer, 1998.

84. Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *ACM transactions on graphics (TOG)*, volume 21, pages 339–346. ACM, 2002.

85. Humayun Irshad, Stéphane U Rigaud, Alexandre Gouaillard, et al. Primal/dual mesh with application to triangular/simplex mesh and delaunay/voronoi. *Insight Journal*, pages 1–14, 2012.

86. Sarah F Frisken Gibson. Using distance maps for accurate surface representation in sampled volumes. In *Volume Visualization, 1998. IEEE Symposium on*, pages 23–30. IEEE, 1998.

87. P P Banerjee, Charles S Bouchard, Philip Dray, Paul Bryar, Richard M Ahuja, and Deepak P Edward. Pgy 4 capsulorhexis performance metrics on the sensimmer phaco simulator and during phacoemulsification-a paired control study. *Investigative Ophthalmology & Visual Science*, 52(14):4715–4715, 2011.

88. Rachel Yudkowsky, C Luciano, P Banerjee, et al. Ventriculostomy practice on a library of virtual brains using a vr/haptic simulator improves simulator and surgical outcomes. In *12th Annual International Meeting on Simulation in Healthcare (IMSH)*, 2012.

89. Rachel Yudkowsky, Cristian Luciano, Pat Banerjee, Alan Schwartz, Ali Alaraj, G Michael Lemole Jr, Fady Charbel, Kelly Smith, Silvio Rizzi, Richard Byrne, et al. Practice on an augmented reality/haptic simulator and library of virtual brains improves residentsâĂŹ ability to perform a ventriculostomy. *Simulation in Healthcare*, 8(1):25–31, 2013.

90. P Banerjee, Silvio Rizzi, and Cristian Luciano. Virtual reality and haptic interface for cellular injection simulation. *Proceedings of 14th Medicine Meets Virtual Reality, JD Westwood et al, YOS Press*, pages 37–39, 2007.

91. P Pat Banerjee, Cristian J Luciano, G Michael Lemole Jr, Fady T Charbel, and Michael Y Oh. Accuracy of ventriculostomy catheter placement using a head-and hand-tracked high-resolution virtual reality simulator with haptic feedback. 2007.

92. Ali Alaraj, Fady T Charbel, Daniel Birk, Mathew Tobin, Cristian Luciano, Pat P Banerjee, Silvio Rizzi, Jeff Sorenson, Kevin Foley, Konstantin Slavin, et al. Role of cranial and spinal virtual and augmented reality simulation using immersive touch modules in neurosurgical training. *Neurosurgery*, 72(0 1):115, 2013.

93. Jens Kruger and Rüdiger Westermann. Acceleration techniques for gpu-based volume rendering. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 38. IEEE Computer Society, 2003.

94. Robert J Teather and Wolfgang Stuerzlinger. Pointing at 3d targets in a stereo head-tracked virtual environment. In *3D User Interfaces (3DUI), 2011 IEEE Symposium on*, pages 87–94. IEEE, 2011.

95. Randy Pausch, Dennis Proffitt, and George Williams. Quantifying immersion in virtual reality. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 13–18. ACM Press/Addison-Wesley Publishing Co., 1997.

96. Silvio H Rizzi, Cristian J Luciano, and P Pat Banerjee. Comparison of algorithms for haptic interaction with isosurfaces extracted from volumetric datasets. *Journal of computing and information science in engineering*, 12(2):021004, 2012.

97. Everette S Gardner. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1):1–28, 1985.

98. Cancer Research UK. Diagram showing a craniotomy. `https://en.wikipedia.org/wiki/Craniotomy`, 2014. [Online; accessed November 11, 2015].

99. JB Allibone, WJF Harkness, and RD Hayward. Craniotomy-related complications in a paediatric neurosurgery unita prospective study. *British journal of neurosurgery*, 13(2):148–153, 1999.

100. Daniel G Deschler, Philip H Gutin, Adam N Mamelak, Michael W McDermott, and Michael J Kaplan. Complications of anterior skull base surgery. *Skull base surgery*, 6(2):113, 1996.

101. Rahul Jandial, Paul McCormick, and Peter M Black. *Core Techniques in Operative Neurosurgery*. Elsevier Health Sciences, 2011.

102. John W Baker. Cranial perforator, October 13 1987. US Patent 4,699,550.

103. Tadashi Hamasaki, Motohiro Morioka, Hideo Nakamura, Shigetoshi Yano, Toshinori Hirai, and Jun-ichi Kuratsu. A 3-dimensional computed tomographic procedure for planning retrosigmoid craniotomy. *Operative Neurosurgery*, 64(suppl_5):ons241–ons246, 2009.

104. Wikipedia. Laminectomy — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Laminectomy`, 2015. [Online; accessed December 1, 2015].

105. Deepak Awasthi and Najeeb Thomas. Pedicle screw placement. `https://www.medschool.lsuhsc.edu/neurosurgery/nervecenter/tlscrew.html`. [Online; accessed July 1, 2017].

106. Hui Chen and Hanqiu Sun. Real-time haptic sculpting in virtual volume space. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 81–88. ACM, 2002.

107. Ioannis D Gelalis, Nikolaos K Paschos, Emilios E Pakos, Angelos N Politis, Christina M Arnaoutoglou, Athanasios C Karageorgos, Avraam Ploumis, and Theodoros A Xenakis. Accuracy of pedicle screw placement: a systematic review of prospective in vivo studies comparing free hand, fluoroscopy guidance and navigation techniques. *European Spine Journal*, 21(2):247–255, 2012.

108. Langston T Holly and Kevin T Foley. Three-dimensional fluoroscopy-guided percutaneous thoracolumbar pedicle screw placement. *Journal of Neurosurgery: Spine*, 99(3):324–329, 2003.

109. Frank L Acosta, Timothy L Thompson, Stacey Campbell, Philip R Weinstein, and Christopher P Ames. Use of intraoperative isocentric c-arm 3d fluoroscopy for sextant percutaneous pedicle screw placement: case report and review of the literature. *The Spine Journal*, 5(3):339–343, 2005.

110. Erik Vidholm. *Visualization and haptics for interactive medical image analysis*. PhD thesis, Acta Universitatis Upsaliensis, 2008.

111. William A McNeely, Kevin D Puterbaugh, and James J Troy. Six degree-of-freedom haptic rendering using voxel sampling. *ACM SIGGRAPH 2005 Courses*, page 42, 2005.

112. Sonny Chan, François Conti, Nikolas H Blevins, and Kenneth Salisbury. Constraint-based six degree-of-freedom haptic rendering of volume-embedded isosurfaces. In *World Haptics Conference (WHC), 2011 IEEE*, pages 89–94. IEEE, 2011.

VITA

Jie Jiang

Education

Ph.D., Industrial Engineering and Operations Research
University of Illinois at Chicago, Chicago, IL
Sep 2017

B.S., Electronics Engineering
Beijing Institute of Technology, Beijing, China
July 2010

Experience

Research Assistant
Spring 2017 to Summer 2017
Industrial Virtual Reality Institute, UIC, IL

Teaching Assistant
Fall 2012 to December 2016
Department of Mechanical and Industrial Engineering, UIC, IL

DevTech Intern
Summer 2016
NVIDIA, Santa Clara, CA

Graduate Research Aide
Summer 2014, Summer 2016
Argonne Leadership Computing Facility, Argonne National Laboratory, IL

Software Engineer System Engineer
Summer 2010 to Spring 2012
Telvent Schneider-electric, Beijing, China

VITA (Continued)

Publication

Jiang, Jie, Mark Hereld, Joseph Insley, Michael E. Papka, Silvio Rizzi, Thomas Uram, and Venkatram Vishwanath. "Streaming ultra high resolution images to large tiled display at nearly interactive frame rate with vl3." In Large Data Analysis and Visualization (LDAV), 2015 IEEE 5th Symposium on, pp. 133-134. IEEE, 2015.

Jiang, Jie, Thomas Fogal, Cliff Woolley, and Peter Messmer. "A lightweight H. 264-based hardware accelerated image compression library." In Large Data Analysis and Visualization (LDAV), 2016 IEEE 6th Symposium on, pp. 99-100. IEEE, 2016.

Patent

Banerjee, P. Pat, Cristian J. Luciano, Ali Alaraj, Fady T. Charbel, and Jie Jiang. "Haptic augmented and virtual reality system for simulation of surgical procedures." (2016).

Related Events

Peer-reviewed Workshop Talk at VISTech15.

Beste poster awards at LDAV 2015.

Guest editor for SIBGRAPI 2016.

Graduated from Honor Program at BIT 2010.