Western Graduate&PostdoctoralStudies

Western University
Scholarship@Western

Electronic Thesis and Dissertation Repository

12-13-2018 12:00 PM

Virtual Reality Simulation of Glenoid Reaming Procedure

Mohammadreza Faieghi, *The University of Western Ontario*

Supervisor: Tutunea-Fatan, Ovidiu-Remus, *The University of Western Ontario*
Co-Supervisor: Eagleson, Roy, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Biomedical Engineering
© Mohammadreza Faieghi 2018

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Biomechanical Engineering Commons, Biomedical Engineering and Bioengineering Commons, Computer-Aided Engineering and Design Commons, Medical Education Commons, and the Robotics Commons

Recommended Citation

Faieghi, Mohammadreza, "Virtual Reality Simulation of Glenoid Reaming Procedure" (2018). *Electronic Thesis and Dissertation Repository*. 5928.
https://ir.lib.uwo.ca/etd/5928

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.

Abstract

Glenoid reaming is a bone machining operation in Total Shoulder Arthroplasty (TSA) in which the glenoid bone is resurfaced to make intimate contact with implant undersurface. While this step is crucial for the longevity of TSA, many surgeons find it technically challenging. With the recent advances in Virtual Reality (VR) simulations, it has become possible to realistically replicate complicated operations without any need for patients or cadavers, and at the same time, provide quantitative feedback to improve surgeons' psycho-motor skills. In light of these advantages, the current thesis intends to develop tools and methods required for construction of a VR simulator for glenoid reaming, in an attempt to construct a reliable tool for preoperative training and planning for surgeons involved with TSA.

Towards the end, this thesis presents computational algorithms to appropriately represent surgery tool and bone in the VR environment, determine their intersection and compute realistic haptic feedback based on the intersections. The core of the computations is constituted by sampled geometrical representations of both objects. In particular, point cloud model of the tool and voxelized model of bone - that is derived from Computed Tomography (CT) images - are employed. The thesis shows how to efficiently construct these models and adequately represent them in memory. It also elucidates how to effectively use these models to rapidly determine tool-bone collisions and account for bone removal momentarily. Furthermore, the thesis applies cadaveric experimental data to study the mechanics of glenoid reaming and proposes a realistic model for haptic computations. The proposed model integrates well with the developed computational tools, enabling real-time haptic and graphic simulation of glenoid reaming.

Throughout the thesis, a particular emphasis is placed upon computational efficiency, especially on the use of parallel computing using Graphics Processing Units (GPUs). Extensive implementation results are also presented to verify the effectiveness of the developments. Not only do the results of this thesis advance the knowledge in the simulation of glenoid reaming, but they also rigorously contribute to the broader area of surgery simulation, and can serve as a step forward to the wider implementation of VR technology in surgeon training programs.

Keywords: Surgery Simulation, Total Shoulder Arthroplasty, Glenoid Reaming, Computer Assisted Orthopaedic Surgery, Voxelization, Collision Detection, Haptics, Finite Element Model, Parallel Computing

Co-Authorship Statement

Chapter 1: M Faieghi - sole author

Chapter 2: M Faieghi - study design, algorithm development, computer programming, data collection, manuscript writing
NK Knowles - study design, algorithm development, data collection, manuscript review
LM Ferreira - study design, manuscript review
OR Tutunea-Fatan - study design, manuscript review

Chapter 3: M Faieghi - study design, algorithm development, computer programming, data collection, manuscript writing
R Eagleson - study design, manuscript review
OR Tutunea-Fatan - study design, manuscript review

Chapter 4: M Faieghi - study design, algorithm development, computer programming, data collection, manuscript writing
R Eagleson - study design, manuscript review
OR Tutunea-Fatan - study design, manuscript review

Chapter 5: M Faieghi - study design, statistical analysis, algorithm development, computer programming, data collection, manuscript writing
M Sharma - study desing, experimental data collection, statistical analysis
V Popa - computer programming
R Eagleson - study design, manuscript review
OR Tutunea-Fatan - study design, manuscript review

Chapter 6: M Faieghi - sole author

Appendix A: M Faieghi - sole author

Acknowledgements

I would like to express my deepest appreciation to my supervisors, Dr. Ovidiu-Remus Tutunea-Fatan and Dr. Roy Eagleson. I am extremely grateful for their inexhaustible encouragement, unwavering guidance, and the liberty that they granted me to explore my ideas during my endeavour.

I would like to extend my deepest gratitude to the members of my advisory committee, Dr. Louis Ferreira and Dr. Ilia Polushin, for their constructive feedback during our committee meetings, and their ingenious suggestions that greatly enriched my research.

Special thanks to Dr. Xi, Dr. Johnson and Dr. Langohr for taking the effort reviewing my thesis and providing valuable comments that improved the quality of my work.

I wish to express my warmest thanks to my friends and colleagues, especially the amazing people of Digitally Enabled Manufacturing Technologies Laboratory and Hand and Upper-Limb Center, who extended a great amount of assistance to my research and made the graduate school life a magnificent experience for me. In my heart, I always remember them and their camaraderie.

I am deeply indebted to my parents, Rostam and Mahboubeh, and my sisters, Lida and Mina, who have supported and nurtured me with unconditional love throughout my academic pursuits.

I express my sincerest thanks to my wife, Sanaz, who made the difficult times of the PhD journey easy with her endless love, patience and sacrifices. Without her support, this thesis would have not been finished anytime soon.

To my beloved wife, Sanaz,
my parents, Rostam and Mahboubeh,
and my sisters, Lida and Mina

Contents

Abstract i

Co-Authorship Statement ii

Acknowledgements iii

Dedication iv

List of Figures viii

List of Tables x

List of Algorithms xi

List of Appendices xii

List of Abbreviations, Symbols, and Nomenclature xiii

1 Introduction **1**
- 1.1 Current Trends in Surgeon Residency Programs . . . 1
- 1.2 Virtual Reality Surgery Simulators . . . 2
- 1.3 Virtual Reality Surgery Simulators in Orthopaedics . . . 4
- 1.4 Glenoid Reaming as a Target for Surgery Simulation . . . 5
- 1.5 Research Objective and Challenges . . . 7
- 1.6 Review of Techniques Used in Surgery Simulation . . . 8
  - 1.6.1 Mechanics of Bone Machining . . . 8
  - 1.6.2 Simulator Software . . . 13
  - 1.6.3 Collision Detection . . . 17
  - 1.6.4 Point Cloud Construction . . . 21
  - 1.6.5 Processing Computed Tomography Data . . . 23
  - 1.6.6 Graphic Rendering Techniques . . . 24
- 1.7 Specific Aims and Thesis Outline . . . 26

2 Processing CT Data **28**
- 2.1 Algorithm Overview . . . 28
- 2.2 Voxel Grid Construction . . . 30
- 2.3 Generation of Geometry and Topology . . . 32

2.4 Material Model 34
2.5 Implementation Results and Discussion 35
  2.5.1 Running Time Breakdown - Fixed Voxel Size 35
  2.5.2 Running Time Break Down - Voxel Upsampling 36
  2.5.3 Time Complexity Analysis 37
2.6 Summary 39

3 Surface Voxelization 40
3.1 Algorithm Overview 40
3.2 Mesh Data Preparation 41
3.3 Voxel Data Representation 42
3.4 Triangle-Box Overlap Test 42
3.5 Algorithm Parallelization 44
3.6 Implementation Results and Discussion 46
  3.6.1 Overall Structure of the OpenCL Program 48
  3.6.2 GPU-Based Parallelization 50
  3.6.3 CPU-Based Parallelization 52
3.7 Summary 54

4 Collision Detection 55
4.1 Algorithm Overview 55
4.2 Voxmap Data Structure 56
4.3 PointShell Data Structure 58
4.4 Broad-Phase Collision Detection 58
4.5 Narrow-Phase Collision Detection 59
4.6 Implementation Results and Discussion 61
  4.6.1 Performance in Different Sampling Resolutions 61
  4.6.2 Running Time Break-Down 62
  4.6.3 Comparison with Zheng et al.'s Method 63
  4.6.4 Comparison with Yau et al.'s Method 65
4.7 Summary 66

5 Modeling and Simulation of Gleonid Reaming 68
5.1 Calibration Experiments 68
5.2 Thrust-Feedrate Relation 70
5.3 Vibration 76
5.4 Simulation of Glenoid Reaming 79
  5.4.1 Integration of all Simulator Components 79
  5.4.2 Simulation with a Haptic Device 81
5.5 Summary 83

6 Thesis Closure 85
6.1 Summary 85
6.2 Strengths and Limitations 88
6.3 Recommendations for Future Research 90

6.4 Significance 92

A Oblique Cutting Model for Glenoid Reaming 94

A.1 Reamer Geometry 94
    A.1.1 Web Angle 95
    A.1.2 Point Angle 96
    A.1.3 Inclination Angle 98
A.2 Coordinates Transformations in Oblique Cutting 98
A.3 Thrust and Torque in Glenoid Reaming 100

Bibliography 102

Curriculum Vitae 117

List of Figures

1.1 Joint replacement process in TSA 5
1.2 Glenoid reaming process 6
1.3 Main components of the intended VR surgery simulator 8
1.4 Mechanics of oblique cutting 10
1.5 Generic structure of OpenCL program 16
1.6 Generic structure of OpenCL device 16
1.7 Collision detection using Voxmap and PointShell 18
1.8 Flowchart of the VPS collision detection algorithm developed by Zheng et al. . 21
1.9 Main phases of voxelization 22

2.1 Flowchart of CT Data processing algorithm 30
2.2 Topology of the hexahedral element in Cartesian mesh 33
2.3 Generated FEMs from sample CT data 37
2.4 Log-log relation between model size and running time for the proposed algorithm 38

3.1 Data flow in our voxelization algorithm 41
3.2 Mesh data preparation prior to main voxelization computations 42
3.3 Benchmarked models 48
3.4 Core structure of the OpenCL program used to implement voxelization 49
3.5 Comparison between OpenCL device-host data transfer methods 50
3.6 Normalized voxelization running time obtained by GPUs 51
3.7 Voxelized representation of reamer in different resolutions 52
3.8 Normalized voxelization running time using CPUs 53

4.1 Data-flow in our collision detection algorithm 56
4.2 Construction of point cloud for reamer 58
4.3 A 2D representation of VOI computed in broad-phase 59
4.4 Comparative assessment of VOI on collision detection performance 63

5.1 Calibration experiments set-up 69
5.2 Average thrust-displacement characteristics of glenoid reaming in cortical bone 71
5.3 Average thrust-displacement characteristics of glenoid reaming in cancellous bone 72
5.4 Prediction of feedrate as a function of density in cortical bone 75
5.5 Prediction of feedrate as a function of density in cancellous bone 75
5.6 Periodogram of vibrations during reaming cortical bone in five different specimens 77

5.7 Periodogram of vibrations during reaming cancellous bone in five different specimens 77
5.8 Sample vibration signal prepared for haptic simulation of reaming cortical bone 78
5.9 Sample vibration signal prepared for haptic simulation of reaming cancellous bone 78
5.10 Data-flow in our simulator software 80
5.11 Computing time measured for haptic rendering frames 82
5.12 Computing time measured for graphic rendering frames 83
5.13 Graphic representation of glenoid reaming 83

A.1 A sample point on reamer cutting lip and its radial distance 95
A.2 Definition of web angle for reamer 96
A.3 Definition of point angle for reamer 97
A.4 Coordinates system defined for elementary forces in oblique cutting 99

List of Tables

2.1 Comparative assessment of hash mapping efficiency 34
2.2 Specifications of the models used for testing 35
2.3 Breakdown of the running time for different phases of the proposed algorithm . 36
2.4 Peak memory usage for the tested models 36
2.5 Breakdown of the algorithm running time for cancellous core with voxel upsampling 38

3.1 Comparison between triangle- and voxel-based parallelization schemes 45
3.2 Specifications of processors used in voxelization tests 47
3.3 Specifications of benchmarked models 48
3.4 Voxelization running time obtained for different models and resolutions using GPUs 51
3.5 Voxelization running time obtained for different models and resolutions using CPUs 53

4.1 Running time (ms) of our collision detection algorithm obtained for various sampling resolutions 61
4.2 Running time break-down of our collision detection algorithm in the presence of 103K contact points between PointShell and Voxmap of size 1024^3 62
4.3 Running time (ms) for Zheng et al.'s collision detection algorithm obtained for various sampling resolutions 64
4.4 Required memory for PointShell and Voxmap in our method compared with Zheng et al.'s method 65
4.5 Running time for Voxmap-based parallelization algorithm obtained for various resolutions 66

5.1 Bone density, reamer feedrate and appearant machining stiffness observed in calibration experiments 73
5.2 Time-domain metrics measured for vibrations during reaming different specimens 76

List of Algorithms

3.1 Voxelization kernel pseudocode 46

4.1 Narrow-phase kernel pseudocode 60

5.1 Pseudocode for scheduling haptics and graphics computations 81

List of Appendices

Appendix A : Oblique Cutting Model for Glenoid Reaming 94

List of Abbreviations, Symbols, and Nomenclature

a	Half-length of major axis of ellipse that describes reamer cutting lip
AABB	Axis-Aligned Bounding Box
API	Application Programming Interface
b	Half-length of minor axis of ellipse that describes reamer cutting lip
CNC	Computer Numerical Control
CPU	Central Processing Unit
CT	Computed Tomography
D	Density
$\mathcal{D}$	Density array to describe spatial occupancy of voxel grid
$\mathbf{d}$	Voxel grid dimension
df_n	Normal elemental force
df_t	Tangential element force
DICOM	Digital Imaging and Communications in Medicine
dm	Torque generated by an element
f	Feedrate
FEA	Finite Element Analysis
FEM	Finite Element Model
FPS	Frames per Second
g_{pk-pk}	Peak-to-peak amplitude of vibration signal
g_{rms}	Root mean square of vibration signal
$\mathcal{G}$	Voxel grid
GPGPU	General-Purpose Graphics Processing Unit
GPU	Graphics Processing Unit
k	Apparent machining stiffness
k_c	Specific cutting coefficient
k_f	Friction coefficient
l	Depth of cut
M	Global torque
$\mathcal{M}$	Triangular mesh
n_{max}	Total number of cells in voxel grid
OBB	Oriented Bounding Box
OpenCL	Open Computing Library

OpenGL	Open Graphics Library
p	Probability value
$\mathcal{P}$	PointShell data structure
$\mathbf{p}_0$	Voxel grid minimum corner
r	Radial distance for an element on reamer cutting lip
$\mathbf{R}$	Rotation matrix
R^2	Coefficient of determination
RAM	Random Access Memory
RMS	Root Mean Square
SAT	Separating Axis Theorem
T	Global thrust
$\mathcal{T}$	Triangle
$\mathbf{T}$	Transformation matrix from tool to bone coordinate systems
TSA	Total Shoulder Arthroplasty
(u, v, w)	Integer coordinates of voxel in voxel grid
V	Tool velocity
$\mathcal{V}$	Voxel
$\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$	Three vertices of triangle
VOI	Volume of Interest
VPS	Voxmap-PointShell
VR	Virtual Reality
w	Width of cut
α	Angle between cutting edge and radius of reamer circle
β	Web angle
γ_n	Normal rake angle
$\Delta\mathbf{p}$	Voxel diagonal
ε	Point angle
λ	Inclination angle
μ	Cutting angle
μCT	Micro Computed Tomography
μFEM	Micro Finite Element Model
ρ	Linear correlation coefficient
τ	Second Euler angle

Chapter 1

Introduction

1.1 Current Trends in Surgeon Residency Programs

The relation between the volume of practice and the outcome in surgery is a long-standing concern of surgeon training programs. It is widely reported that the more often a surgical procedure is performed, the lower its morbidity, and the better the outcome [1]–[10]. For example, in a systematic review of 135 volume-outcome studies spanning over 27 surgical procedures, it has been revealed that 71% of all studies of hospital volume and 69% of studies of physician volume reported statistically significant correlation between higher volume and better outcome while no study documented a statistically significant association between higher volume and worse outcomes [1]. Therefore, considering medical errors which are essentially caused by the lack of surgeon skills continue to remain one of the leading reasons of death [9], it is easy to infer that an increased emphasis on the practice of surgeons will translate into decreasing risks of surgeries. By the same token, it can be expected that the time and cost of surgeries can be diminished. However, it is important to realize that practice should not be confused with repetition; performance does not improve simply because a task is repeated. The key to consistent improvement is to engage surgeons in deliberate practice [10].

In clinical training, deliberate practice is interpreted as sustained practice with the intention of addressing the weaknesses that are identified by assessments and stimulated by feedback [11]. Deliberate practice plays a key role in improvement of psychomotor skills of surgeons [12]. Although skill acquisition rate varies significantly in practice of different surgeries, the well-known rule of 10,000 hours of practice to master a skill has become an ideal target in surgeon training programs [13]. However, a focus on patient safety, work hour restrictions, limited availability of cadavers and animals as well as their high cost and safety risks are evident enough that this amount of time cannot be attained during routine clinical programs. This brings about an interesting question: *how can it be ensured that the current medical programs provide enough practice options to enhance the overall competency of the trained surgeons?* Presently, it is believed that the answer that might be able to address this concern is constituted by a wider implementation of virtual surgical simulators, to be primarily used for training purposes.

1.2 Virtual Reality Surgery Simulators

A Virtual Reality (VR) surgical simulator can be regarded as nothing but a computer-based system developed to resemble surgical procedures for the purpose of training of medical professionals. As such, the latest generations of simulators are employing a combination of haptic technology and computer graphics to replicate surgical environments. On one hand, the haptic system is capable to enhance the surgical simulator with sense of touch otherwise known as haptic feedback. On the other hand, the computer graphics add-on provides visual guidance to the surgeon. Beyond eliminating the stringent need of a patient or cadaver, these simulators are best suited to provide structured feedback on the surgeons performance because they can effectively construct quantitative assessments of the performed operations. Consequently, simulation-based training is known to be an exceptional candidate in elevating the efficiency of residency programs. As a matter of fact, of fourteen studies reviewed in a meta-analysis, simulation-based training was found overall to be a superior method compared to the traditional apprenticeship model [14]. Many simulators have been designed for various surgeries. The Karlsruhe Endoscopic Surgery Trainer is an example of VR simulators in laparoscopy. It consists of a “phantom box” which provides a rough imitation of outward human abdomen by means of electromechanical systems as well as a “central unit” which is a high-performance graphics workstation. The system has been successfully installed at the minimally invasive surgery training center of University Hospital of Tuebingen since 1996 and also is commercially available [15]. As another paradigm of simulators in laparoscopy, Neyret et al. have developed a real-time rendering scheme towards high-end simulation of liver surgery [16]. In Rhinoplasty, Lee et al. have integrated image processing and computer graphics techniques to come up with a simulator that enables surgeons to better understand the segmentation of nose and face structures and in turn find the best way to perform the surgery [17]. Kusumoto et al. have investigated the application of simulation in oral implant surgery and have shown applicability of the simulator to train surgeons in basic tasks such as tooth drilling [18]. With a focus on stability and fidelity, Wu et al. developed a VR simulator for dental grinding process. A force model is adopted from machining theory to realistically replicate the grinding process during tool-tooth interaction in VR environment [19]. Among more recent work, Ho et al. have created a VR simulator for myringotomy and implemented metrics for measuring tangible indicators of skill, such as efficiency of blade movements and length and straightness of incisions. In addition, a group of otolaryngologists and otolaryngology residents had evaluated the simulator and the evaluations are further utilized to modify the simualator and improve its practicality [20]. Wang et al. have constructed a simulator for mandibular angle reduction procedure. It has been shown that the force feedback generated by the haptic device of the simulator is qualitatively identical to the cutting forces present in the actual procedure [21]. A phacoemulsification simulator has been designed by Lam et al. and a set of objective-based performance parameters are introduced to quantitatively assess the performance of surgeons [22].

Up to the present time, laparoscopic surgery is a leader in the field of surgery simulation.

Perhaps, the most compelling evidence is that as of 2008, Fundamentals of Laparoscopy Program - which mainly includes working with simulators - is a prerequisite for General Surgery Certification of the American Board of Surgeons [23]. However, other surgery fields - orthopaedics being among them - has not been able to adopt similar practices in their training programs.

1.3 Virtual Reality Surgery Simulators in Orthopaedics

Many orthopaedics training programs still rely on traditional apprenticeship model that has been largely unchanged for nearly 100 years. The fundamental barrier is particularly lack of validated surgical simulation training and assessment approaches [10]. Compared with other surgical specialties, orthopaedists are late in integration of virtual simulation in their training curriculum. In a 2010 review, only 23 articles found that dealt with specific simulators in orthopaedics, compared with 246 citations for laparoscopic simulators [24]. Indeed, there are several review papers that argue orthopaedics lags behind other fields in terms of surgery simulators [10], [24], [25]. The main reason is the limitations in realism and simulation of hard tissue (i.e. bone) interaction, whereas the recent growth of computational power in computers and theoretical developments of haptic rendering have created a great potential for VR to deal with the above obstacle.

Early work in orthopaedics simulation has mostly targeted minimal invasive procedures such as arthroscopy since their replications are generally simpler than open surgery procedures [26]–[28]. As fidelity of VR technology has grown in recent years, there has been more focus on simulation of complicated surgeries. In this regard, a number of simulators have been developed recently to replicate different bone machining operations, including drilling [29], [30], sawing [31], and burring [32]–[34]. One orthopaedic procedure that is currently lacking a VR simulator and can be a good target for future developments in the context of surgery simulation is glenoid reaming, which is known as one of the challenging tasks in Total Shoulder

Arthroplasty (TSA).

1.4 Glenoid Reaming as a Target for Surgery Simulation

TSA is a surgical procedure in which all or part of the glenohumeral joint is replaced by a prosthetic implant. As shown by Fig. 1.1, the main implantation steps in this surgery include placement of glenoid component on glenoid cavity, inserting a metal stem into humerus and installing the humeral head. Prior to positioning the glenoid component, the glenoid bone is often resurfaced through a machining process, otherwise known as glenoid reaming. This process is preformed manually by a surgeon using a power drill and a spherical cutting tool called reamer (Fig. 1.2). A standard glenoid reaming task involves removing the cartilage layer and gently reaming subchondral plate without violating the cancelluos bone to provide a convex bony surface that conforms with the glenoid implant undersurface [35].

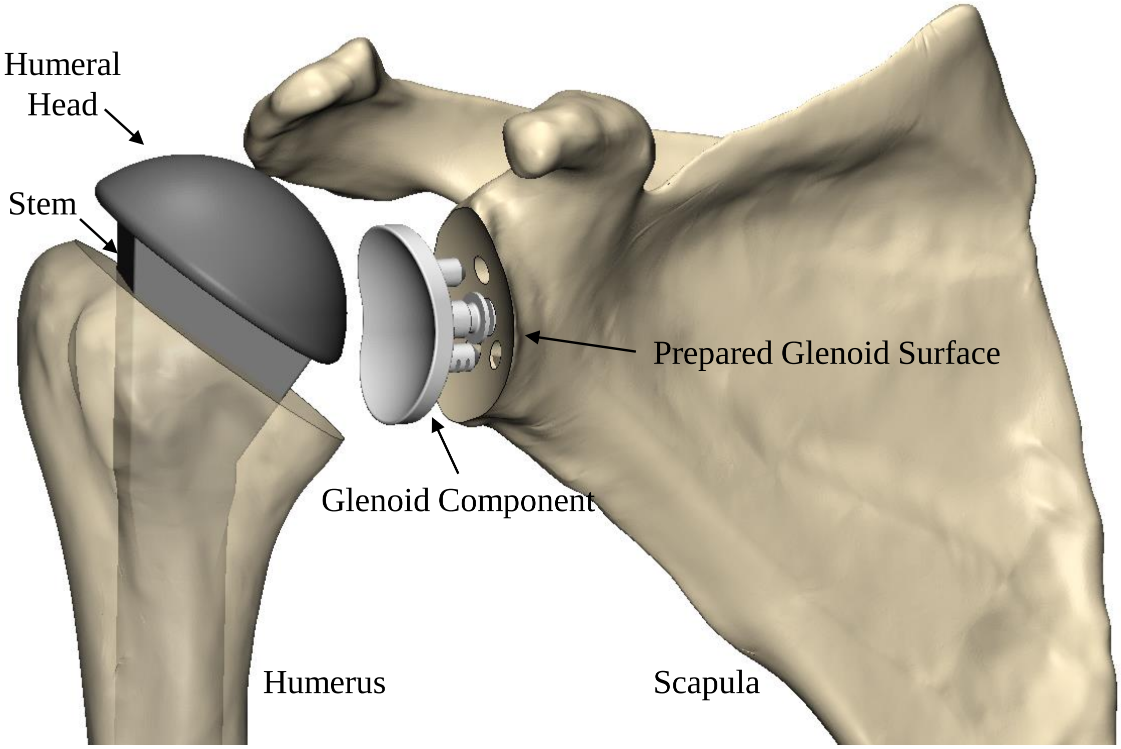
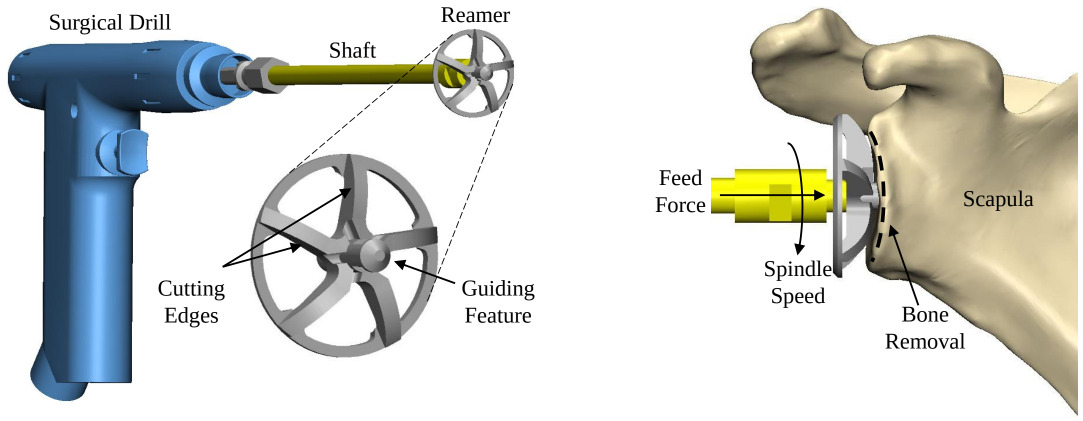


Figure 1.1: Joint replacement process in TSA



(a) Surgical tool (b) Bone removal

Figure 1.2: Glenoid reaming process

TSA can be associated by several complications, the most common of which is the glenoid component loosening [36]–[39]. Many factors may contribute to glenoid component loosening some of which can be directly related to excessive and inaccurate glenoid reaming [40], [41]. For instance, excessive reaming of subchondral bone reduces the glenoid bone strength and decrease the support for the prosthesis [42]. In addition, failure to restore the correct glenoid version will result in glenohumeral instability that eventually leads to implant loosening [43]–[46].

While glenoid reaming is crucial for extending TSA longevity, adequate completion of this task is challenging for surgeons due to the complex three-dimensional anatomy of glenoid surface [47] as well as scapula mobility which moves on the chest wall when reaming forces are applied. Furthermore, the sight to the reaming site is occluded by the reamer itself, leaving surgeons without any visual feedback during the operation. As a result, surgeons have to rely on other feedback mechanisms such as force, vibration or sound to perform glenoid reaming [48].

Learning to interpret subtle cues from the above information requires a high volume of practice, yet most surgeons perform a low number of TSA operations. In fact, while TSA is one of the most important orthopaedic surgeries in upper torso; it has lower incidence compared to hip and knee arthroplasties. About 53,000 people in the U.S. go through shoulder replacement surgery each year, compared to 600,000 and 300,000 of people a year who have knee and hip replacement surgeries, respectively [49]. This relatively low occurrence of TSA makes the majority of surgeons - irrespective of their overall level of surgical experience - less prepared for this type of surgery. A Virtual Reality (VR) simulator for glenoid reaming can provide sufficient volume of practice, without any need for patients or cadavers, and improve the overall level of competency of surgeons in performing not only TSA, but also other surgeries with similar operations, such as Reverse Total Shoulder Arthroplasty that is currently receiving a growing attention [50].

1.5 Research Objective and Challenges

Given the need for a VR simulator in TSA, this study intends to develop a simulator for glenoid reaming to help surgeons practice one of the most challenging tasks of TSA. Since force and vibration are important information that surgeons rely on during glenoid reaming, a suitable VR simulator for this procedure must provide haptic simulation. Moreover, despite the fact that the actual glenoid reaming is attributed by poor visibility, a graphical replication of the procedure can help trainees to become familiar with the characteristics of reamer progression and bone resurfacing.

The key to realistic simulation of glenoid reaming is to generate axial thrust, vibration and visual replication similar to the real procedure. This can be accomplished using a *model of the mechanics of the process* that is embedded in a computer program. The process can be simulated graphically at the same time using a *material removal routine* that eliminates a portion of bone and renders updated bone geometry. As a result, the knowledge of the intersection volume between reamer and bone is quintessential throughout the operation which can be determined using a *collision detection algorithm*. It is important to note that the collision detection running time must be less than 1 ms in order to fulfill the stringent 1 KHz refresh rate required for real-time haptic rendering. Many state-of-the-art orthopaedic simulators have partially alleviated this issue by using sampled geometrical representations, e.g. *point clouds* and *voxelized models*, to represent the virtual objects [29]–[33]. This simplification can, however, negatively affect the fidelity of simulation if the sampling resolution is too coarse. Consequently, the computational efficiency of collision detection becomes critically important since a more efficient algorithm can afford higher sampling resolutions and maintain the simulation realism. In light of the above points, the architecture of the intended VR simulator for glenoid reaming takes the block-diagram representation of Fig. 1.3.

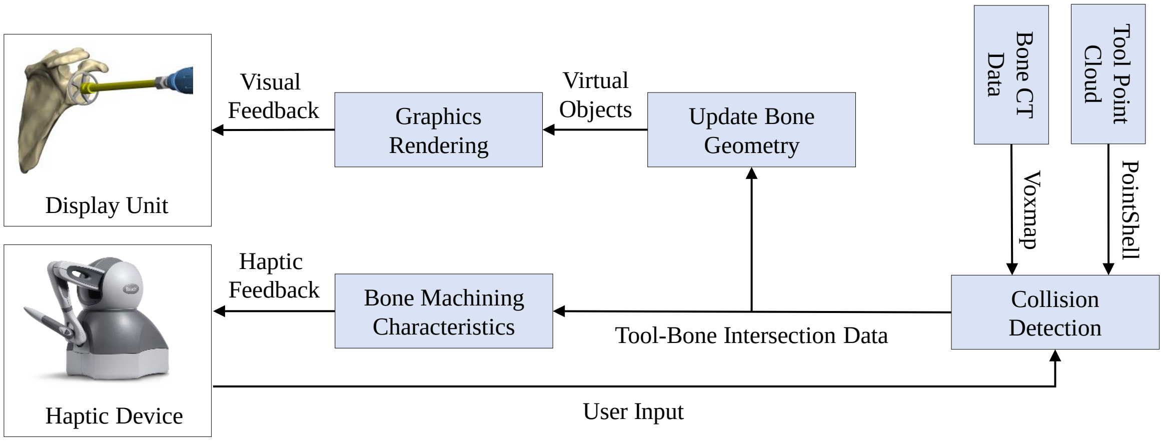


Figure 1.3: Main components of the intended VR surgery simulator

1.6 Review of Techniques Used in Surgery Simulation

The architecture of the intended simulator shown in Fig. 1.3 shares many similarities with numerous simulators that have been already developed in the literature. This section reviews the state-of-the-art methods for main components of an orthopaedic simulator, with the intention to seek the gaps in construction of a glenoid reaming simulator.

1.6.1 Mechanics of Bone Machining

High-fidelity haptic simulation of a bone machining process requires knowledge of mechanical characteristics of the process. This knowledge is additionally advantageous in designing appropriate cutting tools and selecting favorable surgical conditions to prevent problems such as tool breakage, bone temperature rise or damages to surrounding tissues [51]–[56]. For this reason, since the late 1950s, long before the present advent of surgical simulators, bone machining studies have been conducted [57]–[59]. It is notably accepted that the overall characteristics of bone cutting resemble - to a certain extent, of course - that of metal machining [60]. As such, the methods and theories that were specifically developed in metal machining domain can have at least a partial validity in the context of bone removal.

In metal machining, material removal can be modeled by plastic deformation due to shearing strain occurred at the intersection between cutting edge and workpiece [61]. As shown in Fig. 1.4, the force generated during the contact can be modeled by two vectors that are tangential df_t and normal df_n to the rake face and their amplitude can be related to the area of material sheared away as follows

$$df_n = k_c l w, \quad df_t = k_f k_c l w, \tag{1.1}$$

where w is the width of cut, l denotes the depth of cut, k_c is known as specific cutting coefficient and k_f is referred to as friction coefficient. The force acting on the relief face is insignificant and can be ignored.

The above principle is used to describe elementary machining forces for any infinitesimal element that lies on cutting edges of an arbitrary tool. The sum of all elementary forces can be used to predict axial thrust and torque in different machining operations. The key to accurate prediction of thrust and torque is to properly describe the coefficients k_c and k_f as functions of workpiece material properties, machine dynamics e.g. spindle speed, feed rate and tool geometry that is reflected by parameters such as inclination angle λ and normal rake angle γ_n [62]. The homogeneity of metal material simplifies modeling of k_c and k_f. As a result, analytic models can be derived for various metal machining operations such as drilling, reaming, milling and turning which can be found in [63]–[65].

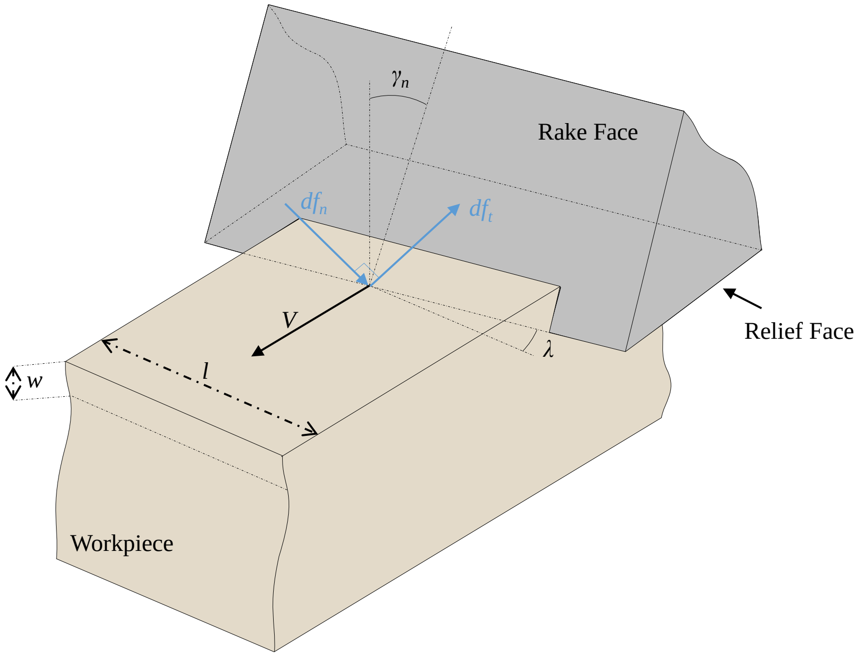


Figure 1.4: Mechanics of oblique cutting

The above methodology becomes complicated when the workpiece has non-uniform material properties. For example, in case of machining a fiber-reinforced composite, the chips are powder-like [66] and their formation mechanism is due to fractures rather than plastic deformation [67]. However, as outlined in [68], there is currently no theoretical model to describe fracture-based cutting process. Consequently, the existing studies on composite machining have developed *mechanistic* models which use the same formulation as metal machining models but require several calibration experiments to account for factors that have not been theoretically incorporated [69]–[74].

Given the complex structure of bone and its anisotropic material properties [75], it is difficult to relate axial thrust and torque to particular mechanical properties and machining parameters. For this reason, several studies have attempted to derive empirical relations between machining force and various factors without solid theoretical justifications. For example, Yanping et al. have modeled machining forces as a function of feedrate, spindle speed and bone density using regression analysis [76]. While the obtained empirical model fits the experimental data well, no additional experiments is reported to evaluate how the obtained model predicts forces for new specimens. Nevertheless, in a validation study [77], the simulator established on this model has shown the ability to differentiate between novices and expert surgeons and the potential to train surgeons through repetitive practice. In another study, Pourkand et al. have modeled bone drilling thrust as a function of cutting depth and feedrate using polynomial fitting techniques [78]. Although this study reports low modeling errors, it only considers synthetic bone throughout experimentation which has limited applicability to model drilling of a real bone [79]. Similar regression analyses have been also reported for oral implant [18], mandibular angle reduction [21] and temporal bone [80] surgeries.

More theoretically profound bone machining models have been developed using the mechanistic approach employed for composite machining. Examples include several bone machining simulators in the context of orthopaedics [30]–[34], tooth drilling [81]–[84] and temporal bone surgery [85]. While all the above-mentioned studies have mentioned specific cutting coefficients must be determined throughout experiments, most of them do not report how the experiments have been carried out and what values are obtained for the coefficients. The only work that actually elaborates on the experimentation is the bone burring simulator developed by Arbabtafti et al. [33] in which constant cutting coefficients are assumed for every element on the cutting edges. A more sophisticated model is introduced by [60], [86] for bone drilling where the cutting coefficients are described as a function of cutting depth w, velocity of each element V and its normal rake angle γ_n, using a power-low equation as follows

$$\begin{pmatrix} \ln k_c \\ \ln k_f \end{pmatrix} = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 \end{pmatrix} \begin{pmatrix} 1 \\ \ln w \\ \ln V \\ \ln(1 - \sin\gamma_n) \end{pmatrix}, \tag{1.2}$$

where α_i and β_i are coefficients that can be determined via regression analysis using axial thrust and torque measured in real experiments. Once the model is calibrated, a set of new experiments have been conducted and compared with the model predictions. In both studies [60],

[86], significant inaccuracies in predicting dynamic axial thrust and torque are evident and the models can only describe the qualitative behaviour. The inaccuracies are primarily due to the complex structure of bone and the fact that bone properties vary from one specimen to the other. This observation raises a question about realisticity of haptic simulation using mechanistic models especially when there is no validation study found for the above-mentioned simulators. As mechanistic models can, however, predict qualitative behavior of a bone machining process, it appears that they can successfully simulate certain phases of an operation, such as transition from cortical bone to cancellous bone which is attributed by a change in the force and torque amplitudes [87].

It is worth noting that Finite Element Analysis (FEA) has been also utilized to model bone machining forces [88]–[92]. In this context, bone material behavior over the yield point is usually modeled using the Johnson-Cook model, similar to metal machining studies [93]–[95]. However, there are significant variations among the damage criteria and progression settings in these studies which has led to inconsistent results in the existing literature [96]. This is probably one of the reasons that has made FEA methods less appealing in the field of VR simulation.

The addition of vibration to the force feedback has proved extremely effective in promoting simulation realism [97]. Vibration in metal machining has been analyzed and modeled for different operations [98]–[102]. However, there is currently little work on understanding vibrations in bone machining. The common approach to generate vibration in haptic feedback is to collect vibration in real experiments using an accelerometer, filter the recorded data and add the outcome to the force feedback as a background signal [34], [80], [103]. In [104], a relation has been found to determine bone drilling vibration peak as a function of applied feed force and bone density. This relation is utilized to reconstruct a vibratory signal by identifying the dominant frequencies observed in the experiments. It is shown that the simulated vibration resembles the experimental signal and is useful to improve fidelity of simulation. Vibration in glenoid reaming is recently studied to develop an augmented reality simulator that features vibration feedback [105]. In this simulator, a haptic transducer is used to play a vibration signal measured during reaming porcine glenoids. Region-specific equations corresponding to cartilage layer, cortical and cancellous bone are developed to adjust the vibration peaks as a function of feed-force when machining different regions of bone. While the generated vibration feedback has proved effective to help surgeons identify the region of bone being resurfaced, it is expected that the fidelity of simulator can be improved using human cadaveric data.

The above points indicate that presently there is no theoretical method that can accurately model bone machining characteristics. All the existing studies have employed empirical or mechanistic approaches, both of which require model calibration against real experiments and can only predict qualitative behavior of an operation. While mechanistic models are theoretically more profound, there is no evidence that they provide superior accuracy compared to the empirical methods. Mechanistic models are also structurally more complicated and computationally expensive. There is currently no model developed to predict axial thrust and torque on glenoid reaming. The only relevant work found in the literature is the robot-driven glenoid reaming study [106] whose experimental data will be used in this thesis.

1.6.2 Simulator Software

An efficient computational platform is an integral part of any haptic-augmented VR surgery simulator. Although the use of game engines such as Unity [107] and Unreal Engine [108] is becoming more popular in developing VR applications, their use in haptic-augmented simulators remains minimal [109]. This is primarily due to the stringent timing requirement of haptics that mandates using a dedicated computational framework which is carefully crafted to perform haptic loop computations in less than 1 ms. This timing constraint is necessary to maintain stable haptic rendering especially when interfacing with a hard tissue [110]. In addition to the haptics, the graphic rendering loop must also update quickly in order to ensure a realistic simulation experience. The generally accepted baseline frame rate for smooth graphic rendering is 30 Frames per Second (FPS) which implies all graphics-related computations must be completed in less than 33.33 ms.

To reliably maintain the above timings, usually a multi-thread computer program is utilized where a separate high-priority thread, otherwise known as servo thread, is dedicated to handle haptic rendering computations. As haptic and graphic rendering loops require access to the same data concurrently, proper memory management and thread synchronization becomes another important aspect of the computational platform [111].

To address the above challenges, many studies have developed custom computational frameworks using C++ programming language. The two common Application Program Interfaces (APIs) that have been utilized to handle haptic interactions are OpenHaptics [112] and CHAI 3D [113]. Graphic rendering is usually handled using Open Graphics Library (OpenGL) [114] while a few studies [33], [84] have reported using Visualization Toolkit (VTK) [115], which is basically a set of classes built upon OpenGL.

In the recent years, the emergence of General-Purpose Graphics Processing Unit (GPGPU) technology have allowed massive parallelization of computing tasks and tremendous reduction of time and power required to solve complex scientific problems. Parallel computing using Graphics Processing Units (GPUs) has now become standard in different disciplines of biomedical engineering [116]. While development of GPU-based computing platforms is generally considered a relatively difficult programming task, its advantages are significant. For example, as reported in [82], a GPU-based surgery simulator program is more than 10 times faster than its conventional counterpart than runs on Central Processing Unit (CPU). Other examples of surgery simulators with GPU-based computing framework include [83], [84], [117]–[121].

In light of the above discussion, it appears that the use of a custom GPU-based framework is a promising approach in development of the glenoid reaming simulator. Presently, the two common APIs to develop a GPU-accelerated computing platforms are CUDA [122] and Open Computing Library (OpenCL) [123]. Contrasting with the vendor-specific nature of CUDA, OpenCL-based developments are compatible with a wide-range of graphics hardware from all major vendors. However, unlike CUDA, OpenCL tends to be more verbose in a sense that more low-level code is required to establish the parallel computing infrastructure. While previous works [82]–[84], [117]–[121] have mostly utilized CUDA, in this thesis, the portability of OpenCL is favored. Given the popularity of OpenHaptics and OpenGL, this study will use these APIs in connection with OpenCL. The remainder of this section is a brief review of OpenCL programming model and terminology. An in-depth explanation can be found in [124].

The OpenCL standard defines a set of data types, data structures, and functions that augment C and C++ to enable the use of both CPUs and GPUs with a single program. The major components of a typical OpenCL program are depicted in Fig. 1.5. *Kernels* are functions to perform parallel computing tasks and they are programmed in OpenCL C, a language established on C99 specifications [125], but further enriched to accommodate parallel programming. Kernels are executed by OpenCL devices that can be either multi-core CPUs or GPUs. It is possible that a computer contain several OpenCL *devices* in its architecture. The OpenCL *context* allows to choose from these devices and manage them for a specific computational task. Control of the kernels and devices in a context are initiated by a segment of regular C/C++ code, termed *host application*.

According to OpenCL terminology, control instructions are called *command queues*, while the CPU running the host application is called OpenCL *host*. In a typical computing task, the input data for kernels is prepared on the host side and is stored within the *host memory* (i.e., general computer Random Access Memory (RAM)). Following this, OpenCL *buffers* transfer the prepared data to the device memory and if the device is a GPU, the *device memory* becomes the memory of graphics card. Once the kernel is executed, the output can be either fetched by the host memory or stored in the device memory for subsequent computations.

The generic OpenCL device model is presented in Fig. 1.6. According to this model, the processing cores of a CPU or GPU are called *compute units*. After the kernel is invoked, the computing task is divided into several subtasks, called *work-groups*, each of them being executed by a compute unit. Furthermore, several processors are available inside of every

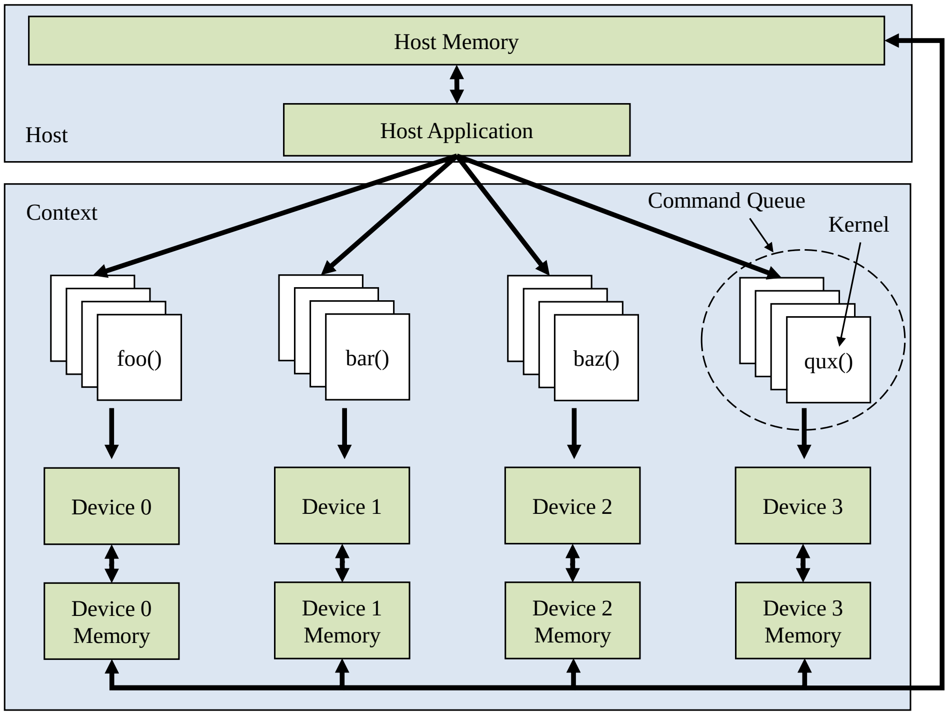


Figure 1.5: Generic structure of OpenCL program

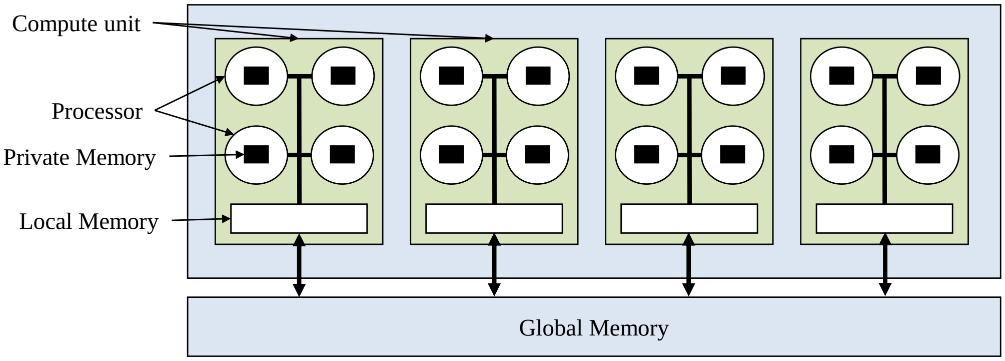


Figure 1.6: Generic structure of OpenCL device

compute unit, each of them being tasked to complete a certain *work-item*, i.e., a certain portion of the work-group.

To accommodate this complex type of hierarchical computing pattern, different layers of memory are embedded into the device. Every processor from the compute unit has a set of dedicated registers called *private memory*, while all processors belonging to the same compute unit share a segment of memory called *local memory*. All compute units can also access the *global memory* (i.e., device memory). A brief comparative analysis of all different types of memory available suggests that private memory remains the fastest, but also the most limited one. By contrast, global memory is the largest, but also the slowest option. Because of this, efficient OpenCL programming needs to limit the number of times of access to global memory to a maximum of two: one for reading kernel inputs and another one for writing kernel outputs.

It is also important to optimally parallelize a compute task by partitioning it into several work-groups. The number of work-items in a work-group is called *local size* and the total number of work-items is denoted by the term *global size*. While the best practice recommends the maximization of the local size, local memory tends to be limited such that the global memory has to be used in most cases and its extensive access will inevitably compromise the performance of the programming. As such, data partitioning often comes down to a challenging trade-off between the maximum use of the local size and the minimal access to the global memory.

1.6.3 Collision Detection

Collision detection computes the tool-tissue intersection and serves as the starting point for both haptic and graphic rendering loops. Therefore, it plays a critical role in performance of a surgery simulator.

In general context of computer graphics, collision detection is known as a performance-critical and integral part of many real-time applications. For this reason, numerous approaches have been proposed to accomplish this task in various applications [126]–[129]. In haptic applications, collision detection becomes a challenging task because its running time must be less than 1 ms. This becomes more challenging when an object undergoes material removal, since its geometry has to be updated in real-time.

To address the above challenge, one approach that has been widely used in bone machining simulation is the well-known Voxmap-PointShell (VPS) method. This method has been introduced by the pioneering work of McNeely et al. in 1999 [130] and has been the basis of many haptic applications ever since [131]–[134]. As depicted by Fig. 1.7, this method relies on sampled geometries of objects, namely a point cloud of surgery tool (PointShell) and a voxelized model of bone (Voxmap) that is originally derived from Computed Tomography (CT). The main advantage of VPS is that its performance is independent from geometrical complexities of objects.

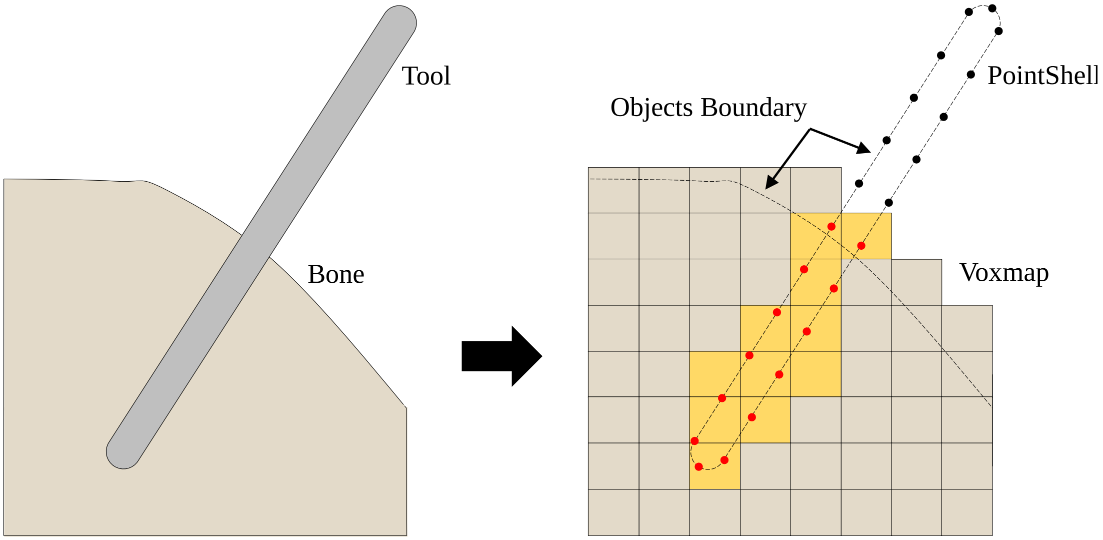


Figure 1.7: Collision detection using Voxmap and PointShell

The sampling resolution plays a determining role in quality of haptic feedback. In coarse resolutions, any changes in the intersection volume results in abrupt changes of force feedback and deteriorates haptic stability [83]. Moreover, bone consists of heterogeneous materials whose stiffness varies spatially [75]. Coarse voxels may miss modelling these variations and fail to incorporate them in force computations. Furthermore, low resolutions cannot capture sharp geometrical features of objects which leads to the so-called deep penetration issue. Given the above points, it is desirable to use finer resolutions to achieve high fidelity haptic simulation.

In practice, however, the maximum attainable sampling resolution is limited by the 1 KHz refresh rate required for haptics. Increasing the sampling resolution results in a drastically higher number of point-voxel collision queries, making collision detection a major bottleneck in the haptic rendering loop. Therefore, there is an imminent need to develop more efficient collision detection algorithms that can handle larger data sets in shorter periods of time.

In orthopedic surgery simulators, the common approach to implement VPS is to transform tool points to Voxmap coordinates and find voxels that enclose the transformed point [30]–[33]. To implement the mechanistic force model, the elemental forces are computed for intersecting point-voxel pairs which will be then aggregated to generate the resultant force feedback. Because seeking intersected voxels is performed in a serial manner, the collision detection performance is poor and sampling resolutions are limited. To address this, Vankipuram et al. [29] has utilized Bounding Volume Hierarchy to represent Voxmap. While this can reduce the number of collision checks, in high resolutions, the prolonged traverses over deep hierarchies become problematic.

In the broader field of haptics, the use of pre-computed distance fields for Voxmap has been proven extremely useful to increase the computational efficiency of VPS [135]–[137]. Further improvements have been achieved by integrating distance field with octree [138]. Nevertheless, such approaches are not suitable for bone machining simulation because the distance field must be reconstructed every time bone undergoes material removal. This constitutes an overhead in collision detection making it difficult to maintain the target 1 KHz frame rate, especially in case of severe intersections.

To address the collision detection between deformable objects, Heidelberger et al. [29, 30] have developed an algorithm similar to VPS but with the use of Layered Depth Image techniques [139], [140]. In this method, the portion of each object within the intersection volume are voxelized and the exact collision detection is queried by Boolean operations between the constructed voxels. To account for deformations, the voxelization step is required in every frame. However, its computing time is affected by the complexity of objects. As a result, the geometrical complexity of objects also affects the performance of this algorithm, as opposed to VPS which only depends on objects sampling resolutions.

Related works can be found in the area of Computed Numerical Control (CNC) simulation, as well [141]. In this regard, Hong et al. [142] have approximated tool geometry by implicit shapes and used analytic equations to describe tool surface. Then, collision detection has been attained by querying bone voxel coordinates against the tool surface. A serial implementation of this algorithm is sluggish and is only suitable for offline computations. However, it can be easily accelerated by checking the collision of voxels concurrently.

In the context of parallel processing, Zheng et al. have developed a GPU-based variant of VPS for a tooth drilling simulator [83]. In this method, each GPU thread queries collision of one point on tool against Voxmap. A 3D grid is utilized for implicit description of Voxmap. Access to the grid cells are performed using a look-up table. An octree-inspired top-down division is utilized to traverse the 3D grid and find intersecting voxels. Once elemental forces acting on each tool point are calculated, the resultant force feedback is computed using a parallel reduction algorithm. The flowchart of this algorithm is shown in Fig. 1.8. Although this method is shown useful to accelerate collision detection by up to 12 times compared to its CPU-based counterpart; the grid traversal makes it sensitive to Voxmap data size. As a result, the algorithm performance will degrade when applied for bone machining simulation, because the size of bone CT data are much larger than a tooth which will result in a much larger Voxmap that is beyond the capabilities of the algorithm.

The above literature review shows that the current works in orthopedic surgery simulation have only used basic variants of VPS which suffers from poor computational performance. This limits the sampling resolutions and in turn leads to low quality haptic feedback. In addition, the previous developments on VPS cannot be extended to bone machining domain effortlessly. Therefore, the need for a new VPS suitable for bone machining continues to remain valid. The use of GPU grid-based approaches appears to be promising as it has been successfully applied in tooth surgery recently [82]. However, bone machining requires a more efficient algorithm because it must be capable to handle much larger Voxmaps. In addition, in operations such as glenoid reaming, surgery tool is large and poses complex geometry, therefore it is necessary to sample tool geometry in high resolutions to capture its fine features. Consequently, the algorithm must respond well to increased resolutions of PointShell. Moreover, since the

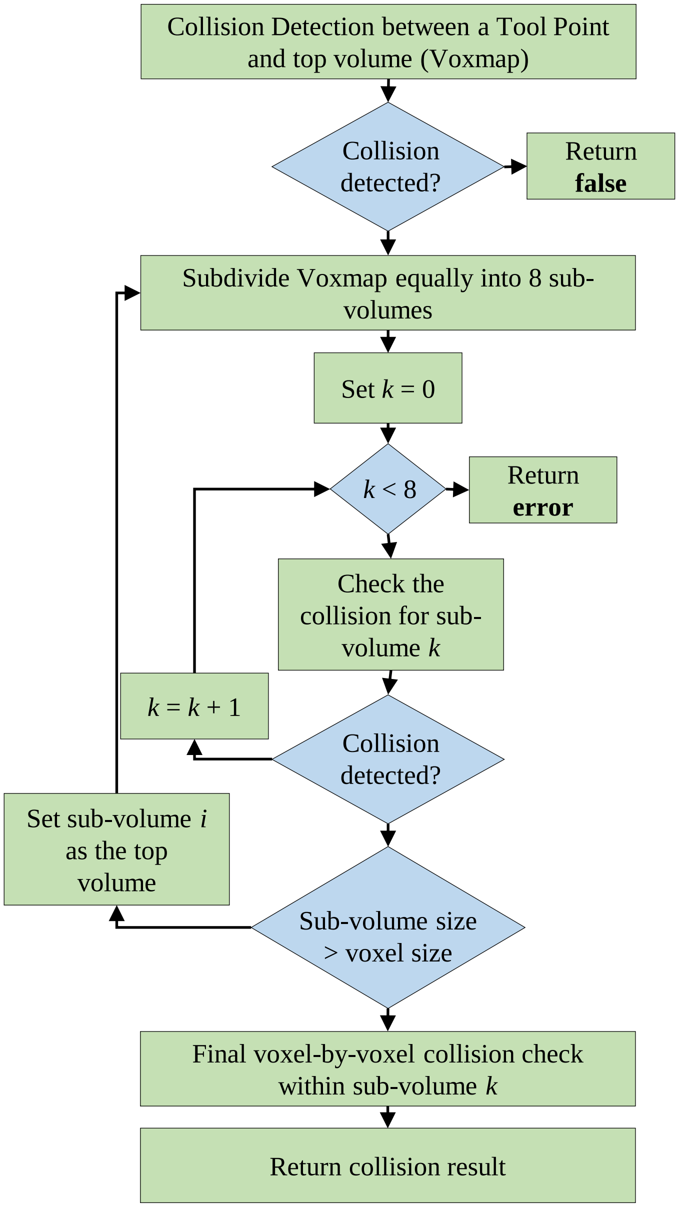

Collision Detection between a Tool Point and top volume (Voxmap)
Collision detected?
Return **false**
Subdivide Voxmap equally into 8 sub-volumes
Set *k* = 0
k < 8
Return **error**
Check the collision for sub-volume *k*
k = *k* + 1
Collision detected?
Set sub-volume *i* as the top volume
Sub-volume size > voxel size
Final voxel-by-voxel collision check within sub-volume *k*
Return collision result

Figure 1.8: Flowchart of the VPS collision detection algorithm developed by Zheng et al.

capacity of GPU memory is limited, low memory consumption becomes critical in order to store large data sets.

1.6.4 Point Cloud Construction

In order to implement the VPS method, the tool geometry must be represented in the form of a point cloud. One common way to construct point cloud representation of an object is to compute a voxelized model of the object surface and then extract the centroid of the voxels [130]. Since most surface representations are in triangular mesh format, the problem of point cloud construction boils down to voxelizing triangular meshes.

As implied by Fig. 1.9, the core of voxelization is represented by the computation of the intersection between triangular facets of a mesh and the voxel grid used to discretize the bounding box of the same mesh. To calculate the required intersections, conventional approaches relies on iterative serial algorithms that loop over all voxels of the grid by repeatedly testing for intersections with each of the mesh facets. In many cases, robust and reliable overlapping test techniques - such as the Separating Axis Theorem (SAT) [143] - can be used to identify and process the required voxel-facet intersections. However, the efficiency of these techniques is somewhat limited in case of fine meshes or dense voxel grids.

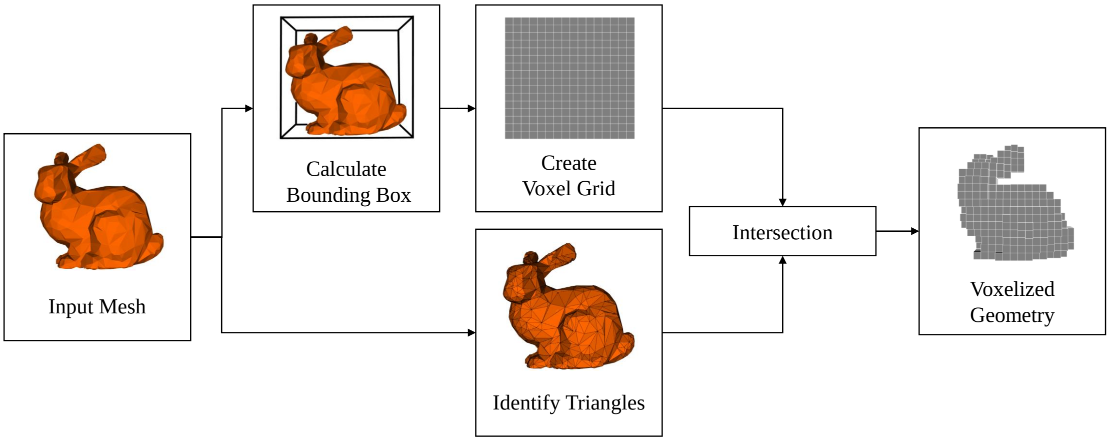


Figure 1.9: Main phases of voxelization

To accelerate voxelization, a number of recent studies have parallelized this task using the built-in graphics pipeline of GPUs [144]–[148]. However, the point sampling method used by the conventional rasterizers of the pipeline leads to inaccurate results for thin structures. To address this, Zhang et al. proposed a conservative voxelization technique [149] that in turn introduced redundant voxels, an issue that was later rectified in [150] by means of hardware-based tessellation and point-based rendering. Even though the latter technique is faster by two orders of magnitude compared to [149], the use of fixed-functions of the pipeline has limited flexibility and leads to inaccuracies for some models. The use of GPGPUs for voxleization is investigated in [151]. In this work, a GPU-based parallelization of the triangle-box overlapping test is presented which is faster than the techniques in [149] by one order of magnitude without comprising the accuracy.

As has been noted, voxelization is a computationally intensive task and can lead to a major overhead in initialization of the simulator software. Therefore, a fast and accurate voxelization technique can be advantageous for the overall performance of the simulator. Toward this end, the method presented in [151] appears to be promising; however, it is only developed in CUDA platform. As the present literature is currently lacking an OpenCL-based voxelization toolkit, it appears that an OpenCL implementation of voxelization algorithm can be valuable for development of glenoid reaming simulator.

1.6.5 Processing Computed Tomography Data

Voxelized models obtained from Computed Tomography (CT) are the standard for characterization of bone geometry and some of its material properties [152], [153]. While the existing orthopaedic simulators have unanimously relied upon clinical CT images, the use of high-resolution imaging techniques such as Micro Computed Tomography (μCT) can be useful to represent bone in higher details and improve simulation realism [154]. This option is, however, limited to cadavers and animals and cannot be used for patient-specific simulation. One way to achieve voxel resolutions beyond clinical CT images, is to upsample the native CT voxels to near micron resolutions. Implementation of this step in the initialization phase of the simulator software will reduce the number of manual steps in processing CT data and facilitate the simulation experience. Towards this end, efficient computing techniques are quintessential since upsampling voxels will generate a large number of voxels that can result in considerable computing time and memory overhead.

The voxelized model of bone can be viewed as a special type of hexahedral volume mesh that is solely constituted by equally shaped and sized eight-node brick elements [155]. This type of mesh is known as Cartesian mesh and has been widely utilized in Micro Finite Element Analysis (μFEA) of the internal porous structure of bone. Cartesian mesh provides the advantage of reduced time for both mesh generation phase and solving the finite element problem, but at the cost of jagged representation of the bone surface [156]. At the micron resolution levels, however, this type of discretizations represent a viable option, particularly since the small size of the elements tends to diminish the - otherwise prominent - surface appearance artifacts. To date, several studies have proved that Cartesian mesh can accurately predict the mechanical properties of the cancellous bone as measured through physical experiments [157]–[164].

Given the analogy between Cartesian mesh and the voxelized model of bone required for the simulator, it can be inferred that the CT processing routine in the initialization phase of the simulator software can be utilized to generate Cartesian meshes for μFEA purposes. This lends well to the research on micro-level structural analysis of bone because many of the available tools have limited capabilties in dealing with the sheer number of voxels in μCT data and are subject to extensive computing time as well as random crashes caused by excessive memory usage.

1.6.6 Graphic Rendering Techniques

Visual replication of glenoid reaming plays an important role to comprehend the characteristics of bone resurfacing since the actual operation is attributed by poor visualization of the reaming site. This entails computing the Boolean subtraction of reamer geometry from bone model in every graphic rendering frame. Faster completion of this task can result in higher graphic rendering frame rates which in turn leads to a smoother and more realistic simulation experience. However, a minimum frame rate of 30 FPS is generally accepted as the standard for smooth graphical replications. This implies that the bone model must be updated and rendered within 33.33 ms intervals.

Although triangle mesh models are considerably efficient in graphic rendering of complex objects, they are unsuitable for real-time Boolean operations. Most of the existing works in this regard report computing times of tens of seconds to perform simple Boolean operations between two surface meshes [115], [165]–[167]. A compelling evidence is the computing time reported for one of the latest algorithms developed in this regard that requires 32.04 s to compute union of two surface meshes with total of 1.5 M triangles even by using a high-end workstation with an Intel Xeon E5540 CPU and 70 GB RAM [168]. While it is expected that the above timing will reduce for coarse meshes, the above performance is by no means close to the requirements of real-time graphic rendering.

To accelerate Boolean operations between two objects, voxelized models serve as extremely effective alternatives. Since voxelized models are constructed using uniform grids, Boolean operations can be efficiently handled using grid-based operations. As a result, voxelized models have been commonly utilized by the existing simulators to deal with graphical representation of bone resurfacing. The methods that are presently developed to render voxelized models can be categorized into two groups. The first category refers to various volume rendering techniques that construct a 2D projection of the geometry that is constituted by 3D voxels. Examples of such techniques include voxel ray tracing [169]–[171], splatting [172]–[174] and texture-based volume rendering [175]–[177]. The second category includes the methods that extract the outer surface of an object using its voxel representation. Examples of these methods include marching cubes [178], surface nets [179] and dual contouring [180].

While all the above-mentioned methods have been commonly implemented in the existing literature, the marching cubes algorithm is the most common technique utilized in orthopaedic simulators. Although the serial implementation of this algorithm is sluggish, it can be effectively parallelized and can be successfully implemented for real-time isosurface computation [181], [182]. An OpenCL-based implementation of this algorithm is also freely available [183]. Therefore, considering the popularity of this algorithm and its satisfactory results, the code available in [183] will be directly integrated in our simulator software.

1.7 Specific Aims and Thesis Outline

The above literature review indicates several gaps in the knowledge and lack of efficient computational tools toward VR simulation of glenoid reaming operation. In particular, it reveals that:

- There is little known about mechanics of glenoid reaming and there is presently a need for a model derived from cadaveric experiments to quantify characteristics of this operation.
- Realistic simulation of glenoid reaming - and bone machining operations, in general - is a computationally demanding task and require development of custom computer programs that supports GPU computing.
- Sampled geometrical representations of tool and bone i.e. point cloud or voxelized models are quintessential to complete haptic and graphic rendering computations in real-time. However, construction of these models is currently challenging due to lack of effective software as well as complicated computations that require high computing power and memory.
- While the use of VPS collision detection algorithm appears to be well-suited for haptic simulation of bone machining operations, the currently available VPS algorithms cannot handle the large intersection volume between reamer and glenoid bone.
- Visual replication of surgical operations are well-addressed in the literature and there exists various methods such as the marching cubes algorithm that can be employed to effectively render updated bone geometry after material removal.

Consequently, it can be inferred that glenoid reaming is a unique bone machining operation with its own characteristics and many currently available techniques utilized in orthopaedic surgery simulation are unsuitable to realistically replicate this operation. Therefore, attaining the objective of this thesis requires

1. *To construct a computationally efficient tool for processing CT data* - This will be addressed in Chapter 2.
2. *To implement an OpenCL-based voxelization engine* - This will be completed in Chapter 3.
3. *To develop a fast VPS collision detection algorithm* - This will be attained in Chapter 4.
4. *To devise an effective material removal routine for updating bone geometry* - This will be also outlined in Chapter 4.
5. *To model mechanics of glenoid reaming* - This will be discussed in Chapter 5.
6. *To effectively consolidate all simulator components such that the target 1 KHz and 30 Hz frame rates for haptic and graphic rendering loops are attained* - This will be also tackled in Chapter 5.

Chapter 2

Processing CT Data

CT imaging of bone is the key to model geometry and material properties of bone. This chapter presents an algorithm to efficiently process CT data and construct the bone model. One of the main features of the algorithm is that it allows upsampling clinical CT voxels to micro-level resolutions. This can directly contribute to obtaining high realism in patient-specific simulations, as discussed in Section 1.6.5. In addition, as the algorithm is capable to effectively handle micro-level CT data, it can be utilized to construct Micro Finite Element Models (μFEMs). Therefore, the primary focus of the subsequent developments is placed upon finite element modeling, and the differences of the model utilized for the simulator is explained wherever necessary.

In order to test the viability of the algorithm, the developed approach has been intentionally tailored to the needs of a specific commercial package (Abaqus, Simulia, Rhode Island, USA) that is commonly used in FEA of biomechanical structures. Nevertheless, this does not significantly restrict the generality of the method.

2.1 Algorithm Overview

According to the specific format of Abaqus input files, the algorithm must generate four distinct blocks of data:

1. 3D coordinates of mesh nodes,
2. Indices of the nodes which are required for elements formation,
3. Element-sets; each set is represented by a group of elements characterized by identical material properties, and
4. Indices of the element-sets which are used to assign material properties to each of the element-sets

The first item refers to the geometry of the model, the second item corresponds to the topology of the model and the last two items describe the material characteristics of the model.

The major steps to construct these blocks of data are outlined in Fig. 2.1. First, CT (or μCT) images are exported as 16-bit Digital Imaging and Communications in Medicine (DICOM) files which are then loaded into the commercial Mimics (Materiallize, Leuven, Belgium) software. The high-frequency noise of the raw images is removed by means of a discrete Gaussian filter. As recommended in [154], a specimen-specific gray-value threshold is used for cancellous bone in order to best preserve its architecture. The image segmentation is performed using region growing with embedded 6-connectivity. This approach ensures the face connectivity of hexahedral elements and avoids generation of nonmanifold geometries. Micro-scale finite element studies for bones are often performed within the elastic regime with Poisson's ratio equal to 0.3 and Young's modulus derived from gray-values [184]. Therefore, the gray-value of each voxel must be passed to the mesh generation algorithm, along with its spatial data. Mimics allows exporting this information into a text file in which every row contains the centroid coordinates $(x_i, y_i, z_i) \in \mathbb{R}^3$ and the gray-value $r_i \in \mathbb{R}$ of i-th voxel, where $1 \leq i \leq n$ and n is the number of post-processed CT voxels.

The choice of Mimics for data preparation is merely determined by its availability. Alternatively, it is also possible to read the voxel data directly from DICOM files and then feed it into the algorithm. However, since CT images often require manual segmentation in order

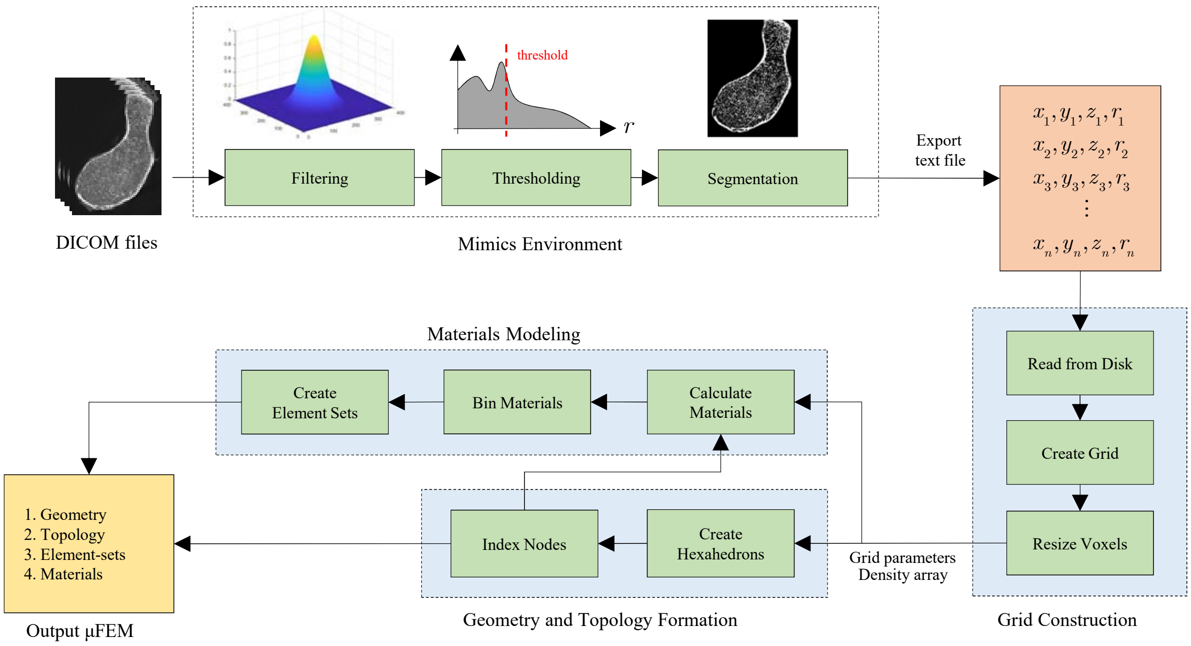


Figure 2.1: Flowchart of CT Data processing algorithm

to extract the region of interest, the need for a powerful image processing tool remains valid [185].

Once the input text file is created, the algorithm constructs a structured 3D grid that embeds all CT voxels. This allows implicit identification of each voxel using efficient integer operations and plays a key role in computational efficiency of the algorithm. The grid is then passed to the *Geometry and Topology Formation* block, in which an explicit representation of the mesh is generated. Then, the *Material Processing* block constructs distinct element-sets and assigns each element-set with a material property that is inferred from gray-values of each voxel. This completes the mesh generation process such that the resulting information can be transferred to Abaqus for the remainder of the FEA steps.

2.2 Voxel Grid Construction

A voxel grid $\mathcal{G}$ is a uniform 3D grid which embodies all voxels obtained from a CT image. The domain of $\mathcal{G}$ spans from the minimum coordinates to maximum coordinates of CT voxels observed during importing the text file. Generally, $\mathcal{G}$ can be uniquely identified using three parameters:

1. The minimum corner of the grid $\mathbf{p}_0 \in \mathbb{R}^3$,
2. Voxel diagonal $\Delta\mathbf{p} \in \mathbb{R}^3$, and
3. Grid dimension i.e., the number of voxels in each direction $\mathbf{d} \in \mathbb{N}^3$.

The total number of cells in $\mathcal{G}$ is then determined as $n_{max} = d_x d_y d_z$.

One effective way to represent $\mathcal{G}$ in memory is to map the grid to a 1D array of length n_{max}, hereafter referred to as the density array $\mathcal{D}$. Each element in $\mathcal{D}$ stores the gray-value of an individual voxel and the index to that element encodes the coordinates. As a result, each voxel can be easily identified by means of a few elementary integer operations. Specifically, a voxel $\mathcal{V}$ with integer grid coordinates $\mathbf{o}_3 \leq (u, v, w) < \mathbf{d}$, maps to the i-th element of $\mathcal{D}$ using

$$i = u + v\ d_x + w\ d_x\ d_y. \tag{2.1}$$

Conversely, for a given index i, the voxel coordinates is calculated by

$$w = \left\lfloor \frac{i}{d_x d_y} \right\rfloor,\ v = \left\lfloor \frac{i - w d_x d_y}{d_x} \right\rfloor,\ \text{and } u = i - w\ d_x\ d_y - v\ d_x \tag{2.2}$$

Therefore, the center coordinates of a voxel can be readily computed by

$$\mathbf{c} = \mathbf{p}_0 + \Delta\mathbf{p} \odot (u, v, w) + 0.5\Delta\mathbf{p}, \tag{2.3}$$

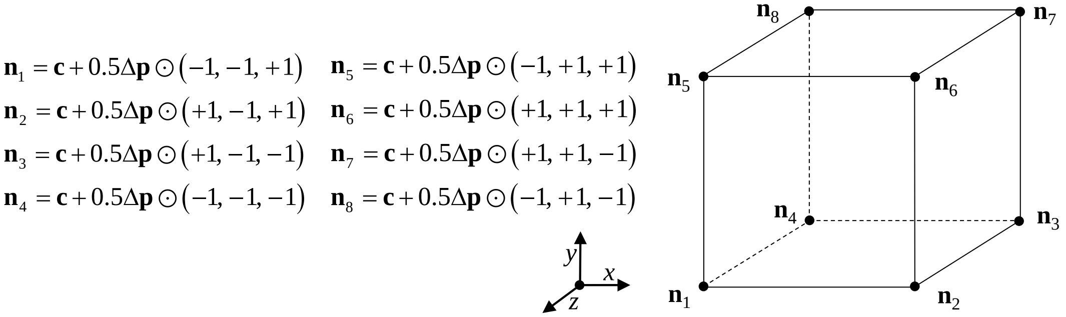
where $\mathbf{c} \in \mathbb{R}^3$ represents center coordinates of a voxel and $\odot$ denotes the component-wise product. As the above equations only involve integers, the use of the density array minimizes the need for floating-point operations. This results in higher numerical robustness because floating-point operations of data with micron-level resolutions are error-prone primarily due to the large number of decimal places associated with this data.

One important advantage of using voxel grids is the added flexibility in resizing voxels when upsampling them to arbitrary finer resolutions. In this regard, the new voxel size must be a divisor of the native CT voxel size to ensure preserving the model volume. In other words, if $\Delta\mathbf{p}_{new}$ is the new voxel diagonal, then $\Delta\mathbf{p}_{new} = \alpha^{-1}\Delta\mathbf{p}$ and $\alpha \in \mathbb{N}$. To upsample voxels, a new voxel grid $\mathcal{G}_{new}$ is constructed with a domain identical to that of $\mathcal{G}$. Subsequently, a new density array $\mathcal{D}_{new}$ is allocated with a length of $\alpha^3 n_{max}$. Iterating over the elements of $\mathcal{D}_{new}$, the gray-value of new voxels are computed using linear interpolation of gray-values associated with the adjacent native CT voxels.

It is noteworthy that $\mathcal{D}$ along with $\mathbf{p}_0$, $\Delta\mathbf{p}$ and $\mathbf{d}$ are sufficient to model bone geometry and its material characteristics required for surgery simulation purposes. As it will be shown in Chapter 4, this representation is advantageous to collision detection performance because voxels can be implicitly identified via Eqs. (2.1) and (2.2), without any need for time-consuming grid traversals.

2.3 Generation of Geometry and Topology

As a general rule, Cartesian mesh is comprised of eight-node hexahedral elements. Node coordinates can be computed by means of the centroid coordinates of the corresponding CT voxels. For this purpose, the algorithm loops over $\mathcal{D}$, calculates the centroid coordinates of occupied voxels via Eq. (2.3), and generates eight nodes in an order shown by Fig. 2.2. The resulting nodes are stored in an array with a length of $8n$ where n is the number of hexahedral elements. The elements spanning from $8i$ to $8i + 7$ correspond to the i-th hexahedral element; therefore, the array describes both the geometry and topology of the mesh. The main advantage of this approach resides in that the knowledge of adjacent nodes is not required since nodes are constructed independently for each elements. This is potentially beneficial for out-of-core as well as parallel extensions of the algorithm which are valuable tools in dealing with very large models. However, this approach will inevitably lead to duplicate nodes that are unacceptable for many FEA solvers. This can be solved by taking a node indexing step, where the uniqueness of each node is ensured while preserving their connectivity.



$$
\begin{aligned}
\mathbf{n}_1 &= \mathbf{c} + 0.5\Delta\mathbf{p} \odot (-1, -1, +1) & \mathbf{n}_5 &= \mathbf{c} + 0.5\Delta\mathbf{p} \odot (-1, +1, +1) \\
\mathbf{n}_2 &= \mathbf{c} + 0.5\Delta\mathbf{p} \odot (+1, -1, +1) & \mathbf{n}_6 &= \mathbf{c} + 0.5\Delta\mathbf{p} \odot (+1, +1, +1) \\
\mathbf{n}_3 &= \mathbf{c} + 0.5\Delta\mathbf{p} \odot (+1, -1, -1) & \mathbf{n}_7 &= \mathbf{c} + 0.5\Delta\mathbf{p} \odot (+1, +1, -1) \\
\mathbf{n}_4 &= \mathbf{c} + 0.5\Delta\mathbf{p} \odot (-1, -1, -1) & \mathbf{n}_8 &= \mathbf{c} + 0.5\Delta\mathbf{p} \odot (-1, +1, -1)
\end{aligned}
$$

Figure 2.2: Topology of the hexahedral element in Cartesian mesh

Evidently, one of the simplest ways to remove duplicate nodes would rely on nested loops that continuously iterate over an array in order to identify the identical elements. However, since the average time complexity for searching an array is linear, the worst-case time complexity of this method is quadratic which results in poor performance in case of large datasets. To accelerate this particular phase, our algorithm incorporates a more efficient technique using hash mapping.

In brief, a hash map is a data container that stores every node coordinate while pairing it with a key value. These key values are computed by a hash function assigned to the hash map. While a linear array will inevitably store its elements in a sequence, the hash map places the elements based on their keys. As a result, searching an element is a constant time complexity operation in a hash map. Once a hash map of all nodes is constructed, the algorithm iterates over each entry of the hash map, identifies all duplicates of the entry and removes them from the hash map. This removes the need for a nested loop set-up and leads to an overall linear time complexity that in turn translates into significant reduction of computing time compared to the conventional nested-loop approach (Tab. 2.1). The resulting nodes and indices will form the explicit representation of the final μFEM.

Table 2.1: Comparative assessment of hash mapping efficiency

Number of CT Voxels	Total Number of Nodes	Number of Duplicate Nodes	Running Time (ms)	
			Nested Loops	*Hash Map*
500	4,000	3,147	2.70	0.21
1,000	8,000	7,167	10.48	0.44
5,000	40,000	33,735	162.17	1.80
15,000	120,000	101.123	1,592.70	6.18
40,000	320,000	270,113	10,986.82	15.42

2.4 Material Model

Although gray-values can provide precise information about the density of an object, additional processing is required in order to convert them into meaningful material properties. While presently there is no generally accepted mapping between gray-values and bone elasticity, most conversion methods advocate for the need of an user-defined function that maps the gray-values into a particular material property [186]. For cancellous structures, this function might be defined as a linear mapping [187]. This function is implemented in the *Calculate Material* block and essentially converts the CT gray-value into a corresponding Young's modulus.

Next, in the optional *Material Binning* step, the computed moduli are categorized into bins of user-defined widths. Subsequently, materials belonging to the same bin are substituted by the center of their bin. This process decimates the number of materials derived from CT in order to reduce the complexity of the resulting FEM in an attempt to speed up the FEA computations. Once the material models are finalized, they must be linked to the mesh elements. To this end, another hash map is utilized to identify all elements characterized by identical material properties. These elements are grouped as distinct element sets and their elasticity values are assigned by indexing to the finalized moduli described above. This completes the material information of μFEM required by Abaqus.

2.5 Implementation Results and Discussion

Several sample CT images are used to test the performance of the proposed algorithm. C++ language was used as the programming platform and computing time was measured by means of the `chrono` timer that is available in the C++ standard library [188]. The hardware used for the tests included a standard Core-i7 6700K CPU equipped with 16 GB RAM. The models used for the tests include three CT datasets as well as a clinical CT image of scapula bone. Table 2.2 presents the details of these models.

2.5.1 Running Time Breakdown - Fixed Voxel Size

Initially, the resolution of the hexahedral mesh is set to match the native CT scan resolution. The material properties of the larger samples (e.g., cellular foam and cadaveric glenoid) are binned with a bucket size of 10. Table 2.3 shows the breakdown of the running time for different steps of the algorithm. To eliminate confounding errors, I/O times are not considered. Given the comparison results in Tab. 2.1 and the number of CT voxels in the studied samples, it is easy to infer that the use of hash tables is significantly advantageous for the overall performance of the algorithm, even though indexing operations continue to remain one of the major bottlenecks.

Peak memory usage for each model is reported in Tab. 2.4. While the algorithm does not run out of memory in none of the analyzed cases, it is expected that larger models will require excessive computing memory. Nevertheless, since the construction of each hexahedral

Table 2.2: Specifications of the models used for testing

Model	Voxel Size (μm)	Voxel Grid Dimension $d_x \times d_y \times d_z$	Number of Occupied Voxels	Text File Size on Disk (MB)
Cancellous Core	$32 \times 32 \times 32$	$281 \times 372 \times 353$	1.7 M	84.8
Cellular Foam	$32 \times 32 \times 32$	$422 \times 629 \times 652$	12.4 M	746
Glenoid	$64 \times 64 \times 64$	$1021 \times 548 \times 742$	36.2 M	1680
Scapula	$472 \times 472 \times 1000$	$311 \times 284 \times 169$	558 K	33.5

Table 2.3: Breakdown of the running time for different phases of the proposed algorithm

Phase	Running Time (ms)			
	Cancellous core	*Cellular foam*	*Glenoid*	*Scapula*
Create Voxel Grid	25.06	189.14	641.88	17.36
Create Hexahedral	196.09	1,383.37	4631.19	69.36
Nodes Indexing	1,354.80	12,888.20	37,013.26	343.33
Calculate Material	15.74	147.41	500.39	9.34
Material Binning	N/A	35.41	97.42	N/A
Material Indexing	387.31	3,301.62	12,279.43	109.24
Sum	1,979.00	17,945.14	55,163.56	548.79

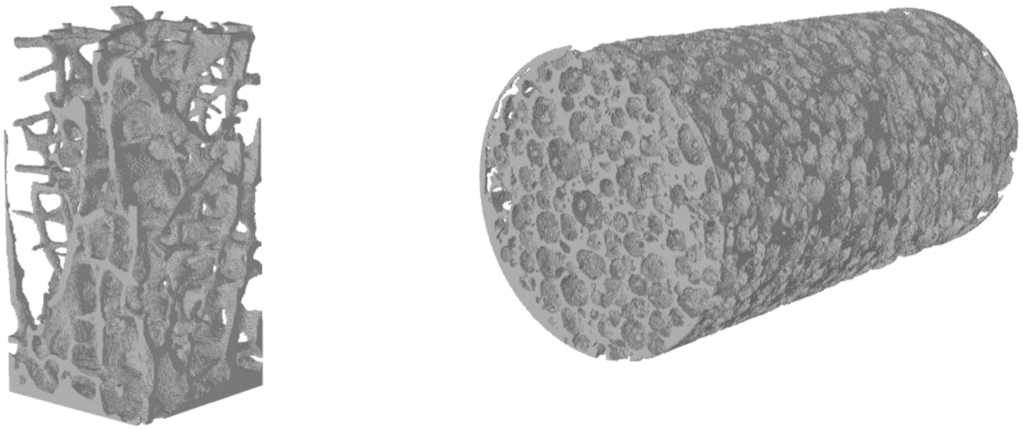
Table 2.4: Peak memory usage for the tested models

Model	Peak Memory Usage (MB)
Cancellous Core	359
Cellular Foam	3,650
Glenoid	10,830
Scapula	213

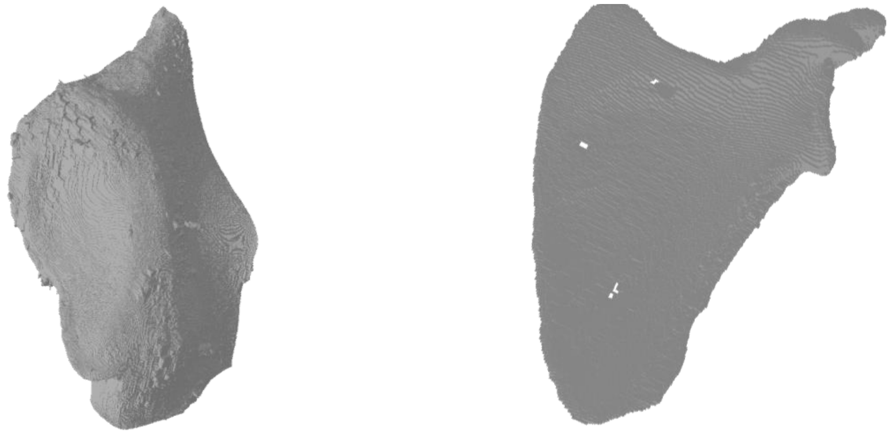
element is independent from the rest of the elements, it is practically possible to use out-of-core implementations in order to accommodate larger models. However, it is reasonable to expect that the slower access to disk will negatively impact the overall computing time. Figure 2.3 depicts the FEMs generated from the sample CT images.

2.5.2 Running Time Break Down - Voxel Upsampling

To investigate the effect of voxel resizing on the algorithm running time, a new μFEM is generated for the cancellous core by fragmenting each CT voxel into eight smaller voxels. This results in an isotropic voxel resolution of 16 μm. The gray-values of the new voxels were obtained through linear interpolation of the CT voxels. As shown in Tab. 2.5, the voxel resizing step - tested on the cancellous core - needs only an additional 4.54 s in order to upsample more than 1.7 M voxels. However, the overall computing time of the algorithm has experienced a significant increase due to the considerably finer resolution of mesh generated at this



(a) Cancellous core (b) Cellular Foam



(c) Glenoid (d) Scapula

Figure 2.3: Generated FEMs from sample CT data

time. Nevertheless, the total running time of 20.74 s remains remarkable, particularly when considering that the total size of the mesh is in excess of 13.6 M hexahedrons. In addition, the results suggest that the number of CT voxels present in an image has a significant impact on the algorithm running time.

2.5.3 Time Complexity Analysis

To further investigate the functional relationship between the number of CT voxels n and the algorithm running time t, various decimations of the glenoid model is tested and the results are

Table 2.5: Breakdown of the algorithm running time for cancellous core with voxel upsampling

Phase	Running Time (ms)
Create Voxel Grid	23.16
Resizing Voxels	4,542.85
Create Hexahedrals	1,470.70
Nodes Indexing	11,330.33
Calculate Material	122.16
Material Indexing	3,255.17
Sum	20,744.37

depicted on a log-log plot. Figure 2.4 reveals that for large datasets, the slope of the plot is approximately one, which implies that

$$\log\left(\frac{t_2}{t_1}\right) = \log\left(\frac{n_2}{n_1}\right) \rightarrow \frac{t_2}{t_1} = \frac{n_2}{n_1}. \tag{2.4}$$

This indicates that the algorithm running time increases linearly with the number of CT voxels which confirms the linear time complexity that was inferred previously. Furthermore, Fig. 2.4 seems to suggest that the algorithm is characterized by constant time complexity for small datasets. However, this is due to the initialization overhead of the algorithm that becomes dominant portion of the running time when the number of CT voxels are low.

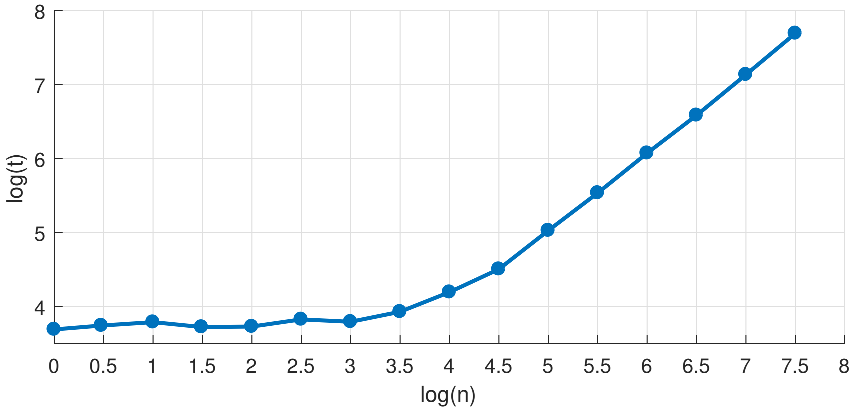


Figure 2.4: Log-log relation between model size and running time for the proposed algorithm

2.6 Summary

This chapter showed that the use of grid-based approaches can provide an effective implicit description of CT voxels. One of the main advantages of using a grid is that every voxel can be accessed by integer grid coordinates, and most voxel-related computations can be handled by fast and robust integer operations. This chapter also presented an efficient voxel storage method using a linear array. While the index to the elements of the array can be utilized to quickly derive spatial information of a voxel, the elements themselves store gray-values associated with each voxel. For CT voxels, a 32-bit float variable is often sufficient to store the gray-values; however, in case of binary voxels that will be described in the next chapter, only one bit of information is required per voxel just to determine whether a voxel is present in a particular grid cell or not. Furthermore, this chapter outlined how grid-based techniques can be utilized to conveniently change voxel resolutions without modifying the bone volume. The method presented in this chapter can be also employed to generate μFEMs. The use of grid-based approach combined with hashing techniques provided a fast and numerically robust algorithm with linear time complexity to generate Cartesian meshes. One important application of such algorithm is micro-level finite element studies of Cancellous bone.

Chapter 3

Surface Voxelization

Point cloud representation of the tool is important to speed up collision detection and simplify haptic rendering computations. As discussed in Section 1.6.4, one convenient way for constructing point cloud is to compute a voxelized model of the tool surface and extract the centroid of the generated voxels. Considering the fact that triangle meshes are widely utilized in computer-aided design and modeling of surgical tools, this chapter develops an algorithm to convert triangle meshes to voxelized representations, in an attempt to develop an effective toolkit for constructing voxelized - and ultimately, point cloud - representations of the reamer, or generally, any object that is modeled by a triangle mesh.

3.1 Algorithm Overview

Figure 3.1 illustrates the general block-diagram of the intended voxelization toolkit. The algorithm starts by importing the mesh data and configuring them in a format that is suitable for accelerating the main voxelization computations. During the mesh import, the algorithm determines the boundaries of the mesh domain and constructs an Axis-Aligned Bounding Box (AABB) for the mesh. This AABB is utilized to construct a voxel grid that will embody all the output voxels. Next, the mesh data as well as the voxel grid are passed to an OpenCL kernel where an exact triangle-box overlap test is invoked for numerous pairs of mesh facets and grid

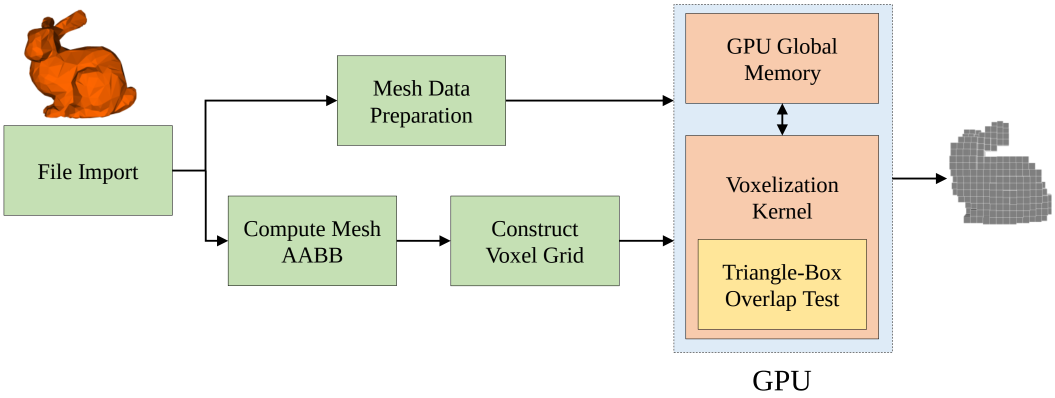


Figure 3.1: Data flow in our voxelization algorithm

cells. Completion of this step generates all the data necessary to construct a voxelized model of the input mesh. This data can be kept in GPU memory for further computations or can be transferred to RAM for exporting to a file.

3.2 Mesh Data Preparation

The vast majority of common triangular mesh formats store mesh data in two different arrays: a float array used for vertex coordinates (i.e., mesh geometry) and an integer array used to describe how the vertices are connected to form the triangles in the mesh (i.e., mesh topology). This particular type of data storage saves memory space but results in a cluttered memory access during kernel execution that in turns slows down the entire voxelization process. To address the issue, the *Mesh Data Preparation* step uses the above two arrays to construct a new float array in which vertex coordinates are sorted per triangle. Specifically, if $\mathcal{M}$ is a mesh with n triangles, then a float array of length $9n$ is generated to hold vertex coordinates such that the elements $9i$ to $9i + 8$ correspond to the vertices of i-th triangle. Therefore, the entire vertex information for each of the mesh triangles is orderly sorted in one locality as shown by Fig. 3.2. This will enable coalesced memory access for the voxelization kernel and can improve the algorithm running time.

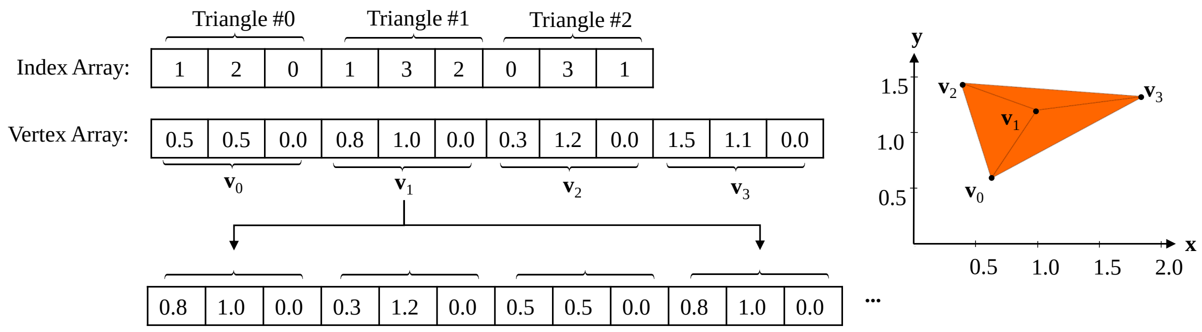


Figure 3.2: Mesh data preparation prior to main voxelization computations

3.3 Voxel Data Representation

The algorithm utilizes a voxel grid $\mathcal{G}$ similar to the one presented in Section 2.2 to represent the output voxelized model. The AABB of the mesh is utilized to construct $\mathcal{G}$. In particular, once the AABB is identified, it is divided into identical cubes whose size is specified by the user. In this manner, all resulting voxels will be aligned with the axes of the coordinate system; therefore, the calculation of normal vectors for all voxels becomes trivial. The voxel grid data is stored in a 1D array as described in Section 2.2 and access to each voxel is attained using Eqs. (2.1) and (2.2). The only information required to store in the density array of the grid is whether a voxel is occupied by a mesh triangle or not. This can be fulfilled by allocating only one bit for each voxel, unlike CT voxels that contain material properties and require larger data types to accommodate storing gray-values.

3.4 Triangle-Box Overlap Test

The algorithm implements the mathematical technique introduced in [151] which is essentially an enhanced version of the SAT method presented in [143] and requires a lower number of operations without comprising the accuracy. According to this method, the evaluation of intersection between a triangular mesh facet and a voxel is a four-step process centered on querying the intersection between the triangle's plane and the voxel.

Toward this end, let $\mathcal{T}$ with vertices $\mathbf{v}_0,\ \mathbf{v}_1,\ \mathbf{v}_2$ be a triangular mesh facet and $\mathcal{V}$ be a voxel characterized by the extreme corners $\mathbf{p}$ and $\mathbf{p} + \Delta\mathbf{p}$, respectively. Under these conditions, the facet-voxel overlap test comes down to the calculation of $\mathcal{T}$'s normal $\mathbf{n}$ and $\mathcal{T}$'s critical point

$$\mathbf{c} = \left(\begin{cases} \Delta\mathbf{p}_x, & \mathbf{n}_x > 0 \\ 0, & \mathbf{n}_x \leq 0 \end{cases}, \begin{cases} \Delta\mathbf{p}_y, & \mathbf{n}_y > 0 \\ 0, & \mathbf{n}_y \leq 0 \end{cases}, \begin{cases} \Delta\mathbf{p}_z, & \mathbf{n}_z > 0 \\ 0, & \mathbf{n}_z \leq 0 \end{cases} \right), \tag{3.1}$$

followed by the evaluation of

$$(\langle \mathbf{n}, \mathbf{p} \rangle + a_1)(\langle \mathbf{n}, \mathbf{p} \rangle + a_2) \leq 0, \tag{3.2}$$

where $a_1 = \langle \mathbf{n},\ \mathbf{c} - \mathbf{v}_0 \rangle$, $a_2 = \langle \mathbf{n},\ \Delta\mathbf{p} - \mathbf{c} - \mathbf{v}_0 \rangle$ and $\langle \cdot, \cdot \rangle$ denotes the dot product. The rest of the intersection test boils down to the assessment of the projections of $\mathcal{T}$ and $\mathcal{V}$ onto the principal planes of the coordinate system. For instance, the following expressions must be evaluated with respect to the xy plane:

$$\mathbf{n}^{xy}_{\mathbf{e}_i} = \left(-\mathbf{e}_{i,y}, \mathbf{e}_{i,x}\right)^T \cdot \begin{cases} 1, & \mathbf{n}_z \geq 0 \\ -1, & \mathbf{n}_z < 0 \end{cases}, \tag{3.3}$$

$$a^{xy}_{\mathbf{e}_i} = -\langle \mathbf{n}^{xy}_{\mathbf{e}_i},\ \mathbf{v}_{i,xy} \rangle + \max\left\{0,\ \Delta\mathbf{p}_x \mathbf{n}^{xy}_{\mathbf{e}_i,x}\right\} + \max\left\{0,\ \Delta\mathbf{p}_y \mathbf{n}^{xy}_{\mathbf{e}_i,y}\right\}, \tag{3.4}$$

for all three edges $\mathbf{e}_i = \mathbf{v}_{i+1 \bmod 3} - \mathbf{v}_i$. If the expression

$$\langle \mathbf{n}^{xy}_{e_i},\ \mathbf{p}_{xy} \rangle + a^{xy}_{e_i} \geq 0, \tag{3.5}$$

hold true $\forall i \in \{0, 1, 2\}$, then the projections of $\mathcal{T}$ and $\mathcal{V}$ on xy plane are intersecting.

One of the important advantages of this method is that if only one of the statements in Eqs. (3.2) or (3.5) are false, then it can be immediately concluded that $\mathcal{T}$ and $\mathcal{V}$ are separated. This can result in early termination of many unnecessary computations and significantly contributes to reducing the running time.

The above intersection test generates a 26-separating voxelized representation of the mesh.

26-separability is a topological property that means there is no path of 26 adjacent voxels that connects a voxel on one side of the surface and a voxel on the other side. This leads to a conservative voxelized representation of the mesh surface in that the output voxels constitute a *supercover* of the input mesh [189]. Alternatively, a *thinner* voxelized model can be constructed using 6-separability property. While 26-separability voxels share a common vertex, edge or face, 6-separability voxels have only faces in common. To compute a 6-separability voxelized representation of the input mesh, the above test must be slightly modified. To this end, the offsets in the plane test become

$$a_1 = \langle \mathbf{n}, \frac{1}{2}\Delta\mathbf{p} - \mathbf{v}_0\rangle + \frac{1}{2}\Delta\mathbf{p}_\diamond \left|\mathbf{n}_\diamond\right|, \tag{3.6}$$

$$a_2 = \langle \mathbf{n}, \frac{1}{2}\Delta\mathbf{p} - \mathbf{v}_0\rangle - \frac{1}{2}\Delta\mathbf{p}_\diamond \left|\mathbf{n}_\diamond\right|, \tag{3.7}$$

where $\diamond = \underset{\square = x,y,z}{\arg\max} \left|\mathbf{n}_\square\right|$. Similarly, Eq. (3.4) becomes

$$a^{xy}_{\mathbf{e}_i} = \langle \mathbf{n}^{xy}_{\mathbf{e}_i}, \frac{1}{2}\Delta\mathbf{p}_{xy} - \mathbf{v}_{i,xy}\rangle + \frac{1}{2}\Delta\mathbf{p}_\diamond \left|\mathbf{n}^{xy}_{e_i,\diamond}\right|, \tag{3.8}$$

where $\diamond = \underset{\square = x,y}{\arg\max} \left|\mathbf{n}^{xy}_{\mathbf{e}_i,\square}\right|$.

3.5 Algorithm Parallelization

Construction of a voxelized model of a mesh entails invoking the above triangle-box overlap test for every pair of mesh facets and voxels. This leads to tedious computations especially in the presence of high number of triangles or voxels. One effective method to circumvent this issue is to parallelize launching the overlap test for different triangle-voxel pairs. This parallelization can be accomplished by either dedicating a thread per voxel, or per triangle.

In the Voxel-Based (VB) parallelization, the number of threads required to accomplish voxelization is essentially equal to the total number of voxels present in $\mathcal{G}$. Each voxel is assigned to a thread where the relative position of the voxel is queried against all mesh facets, iteratively. As soon as an intersection is detected, the iterations terminate and the voxel acquires a boundary status. One of the disadvantages of this approach is that all threads require access to whole mesh data. Since mesh data are often large and cannot fit in GPU local memory, the threads require repetitive access to global memory. This can negatively affect the kernel execution time.

Alternatively, in the Triangle-Based (TB) parallelization, the vertices of each triangular facet $\mathcal{T}$ are assigned to an individual thread. Since voxels outside of $\mathcal{T}$'s AABB are definitely non-intersecting with $\mathcal{T}$ itself, the voxelization problem reduces to querying intersection of $\mathcal{T}$ against only the voxels that lie within $\mathcal{T}$'s AABB. This allows significant reduction of the number of triangle-box overlap tests required in the TB method, compared to its VB counterpart. For example, in voxelizing a sample mesh at different resolutions, the TB approach turns out to be more than four orders of magnitude more efficient than the VB method, as presented in Tab. 3.1.

Furthermore, each thread in the TB approach requires the knowledge of vertex coordinates for only a single triangle (nine floats) which can be transferred to local memory at the beginning phase of kernel execution. As a result, the threads do not require repetitive access to global memory. Instead, they only read input data once at the start of the computations and return the final results at the end. As such, the TB approach is our method of choice for the intended

Table 3.1: Comparison between triangle- and voxel-based parallelization schemes

Voxel Size (mm)	Number of Intersection Tests		Order of Magnitude Ratio (VB/TB)
	VB Approach	*TB Approach*	
3	1,767,431	93	4.2789
2	4,615,507	224	4.3140
1	44,406,021	1,107	4.6033
0.5	368,832,950	5,874	4.7979

voxelization toolkit. The pseudocode of the voxelization kernel using this approach is given by Alg. 3.1.

```
Input: Mesh Vertices Array, $n$, $\mathbf{p}_0$, $\mathbf{d}$, $\Delta\mathbf{p}$
Output: Voxel Array
$i \leftarrow$ work – item number
if $i < n$ then
    $\mathcal{T} \leftarrow$ Read $\mathbf{v}_0$, $\mathbf{v}_1$, $\mathbf{v}_2$ of $i$ – th triangle from Mesh Vertices Array
    Copy $\mathcal{T}$, $\mathbf{d}$, $\Delta\mathbf{p}$ to local memory
    $\mathcal{B} \leftarrow$ Calculate AABB of $\mathcal{T}$
    for every voxel $\mathcal{V} \in \mathcal{B}$ do
        if $\mathcal{T}'s$ plane overlaps $\mathcal{V}$ then
            if $\mathcal{T}$ and $\mathcal{V}$ projections on $xy$ – plane overlaps then
                if $\mathcal{T}$ and $\mathcal{V}$ projections on $xz$ – plane overlaps then
                    if $\mathcal{T}$ and $\mathcal{V}$ projections on $yz$ – plane overlaps then
                        Write proper material value to $\mathcal{V}$'s location in Voxel Array
                    else
                        Continue to next iteration
                    end
                else
                    Continue to next iteration
                end
            else
                Continue to next iteration
            end
        else
            Continue to next iteration
        end
    end
end
```

Algorithm 3.1: Voxelization kernel pseudocode

3.6 Implementation Results and Discussion

This section presents several implementation results to assess the performance of the developed voxelization toolkit. Given the cross-platform feature of OpenCL, the performance of the algorithm is evaluated on various processors. In particular, the hardware used for the tests include high- and mid-range desktop video cards (NVIDIA GeForce 970 GTX, AMD Radeon

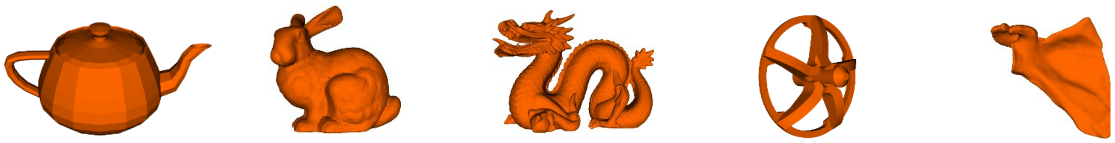
R7 240), a high-end video card for laptops (Nvidia GeForce 960 GTXM) as well as an integrated desktop video card (Intel HD Graphics 530). Since OpenCL also allows CPU-based parallelizations, additional tests are also conducted on desktop Intel Core i7 6700K and AMD FX 770K processors, as well as the mobile Intel Core i7 6700HQ. Table 3.2 summarizes the specifications of all the aforementioned processors. In this table, the number of compute units indicates the number of work-groups that can be concurrently executed in a device, the maximum local size represents the limit of the work-groups, while the local memory size constitutes the amount of dedicated memory that is available for each of the work-groups within a certain processing unit. As it can be inferred from the discussion above, these three parameters play a critical role on the parallelization capabilities of a certain hardware. The last two columns of the table, namely maximum clock frequency and global memory, denote the speed of the processor along with its associated video memory size (for GPU) or RAM (for CPU), respectively. As expected, the size of the global memory limits the maximum resolution of the voxel grid since at some point during the execution of the code the entire voxel data has to be stored in it.

Five different triangle mesh models are utilized for the tests. These models include teapot, bunny and dragon meshes since they are widely utilized as benchmark models in the literature. A reverse engineered triangle mesh model of the reamer used in glenoid reaming as well as a

Table 3.2: Specifications of processors used in voxelization tests

Hardware	Processor Type	Compute Unit Counts	Local Size (KB)	Local Memory (KB)	Clock Frequency (MHz)	Global Memory (GB)
NVIDIA GeForce 970 GTX	GPU	13	1,024	48	1,177	4
NVIDIA GeForce 960 GTXM	GPU	5	1,024	48	1,176	4
AMD Radeon R7 240	GPU	6	256	32	780	2
Intel HD Graphics 530	GPU	24	256	64	1,050	1
Intel Core i7 6700K	CPU	8	8,192	32	4,000	16
Intel Core i7 6700HQ	CPU	8	8,192	32	2,600	12
AMD FX 770K	CPU	4	1,024	32	3,493	8

scapula model obtained through iso-surface extraction of a patient-specific CT image are also included in the test models. These models are depicted in Fig. 3.3 and some of their principal characteristics are presented in Tab. 3.3.



(a) Teapot (b) Bunny (c) Dragon (d) Reamer (e) Scapula

Figure 3.3: Benchmarked models

Table 3.3: Specifications of benchmarked models

Model	Triangle Counts	AABB Size (X, Y, Z)	Voxel Grid Size		
			$\Delta\mathbf{p} = 0.5$	$\Delta\mathbf{p} = 0.5$	$\Delta\mathbf{p} = 0.1$
Teapot	894	$(63.17, 39.31, 29.48)$	$127 \times 79 \times 59$	$253 \times 158 \times 118$	$632 \times 394 \times 295$
Bunny	16,301	$(15.55, 15.33, 12.06)$	$32 \times 31 \times 25$	$63 \times 62 \times 49$	$156 \times 154 \times 121$
Dragon	100,000	$(56.37, 25.21, 39.76)$	$113 \times 51 \times 80$	$226 \times 101 \times 160$	$564 \times 253 \times 398$
Reamer	4,006	$(21, 36.47, 36.50)$	$42 \times 73 \times 73$	$84 \times 146 \times 146$	$211 \times 365 \times 366$
Scapula	470,340	$(144.76, 137.24, 73.28)$	$290 \times 275 \times 147$	$580 \times 549 \times 294$	$1488 \times 1373 \times 733$

The main metric used to evaluate the algorithm performance is the voxelization time, i.e. the time required to generate voxels at a preset size/resolution for each of the sample geometries. These timings are measured by means of the built-in profiling tools available in OpenCL [124]. The comparison baseline used for all the tests is generated by a serial C++ implementation of the algorithm using the Core-i7 6700K processor which is the fastest CPU among our test hardware.

3.6.1 Overall Structure of the OpenCL Program

An overview of the OpenCL code used to implement parallel voxelization is shown in Fig. 3.4. In brief, the program starts with the identification of the OpenCL platform that is available through the hardware followed by the selection of a device to perform the required computations. Next, the kernel code is compiled and brought to an executable form to run on the

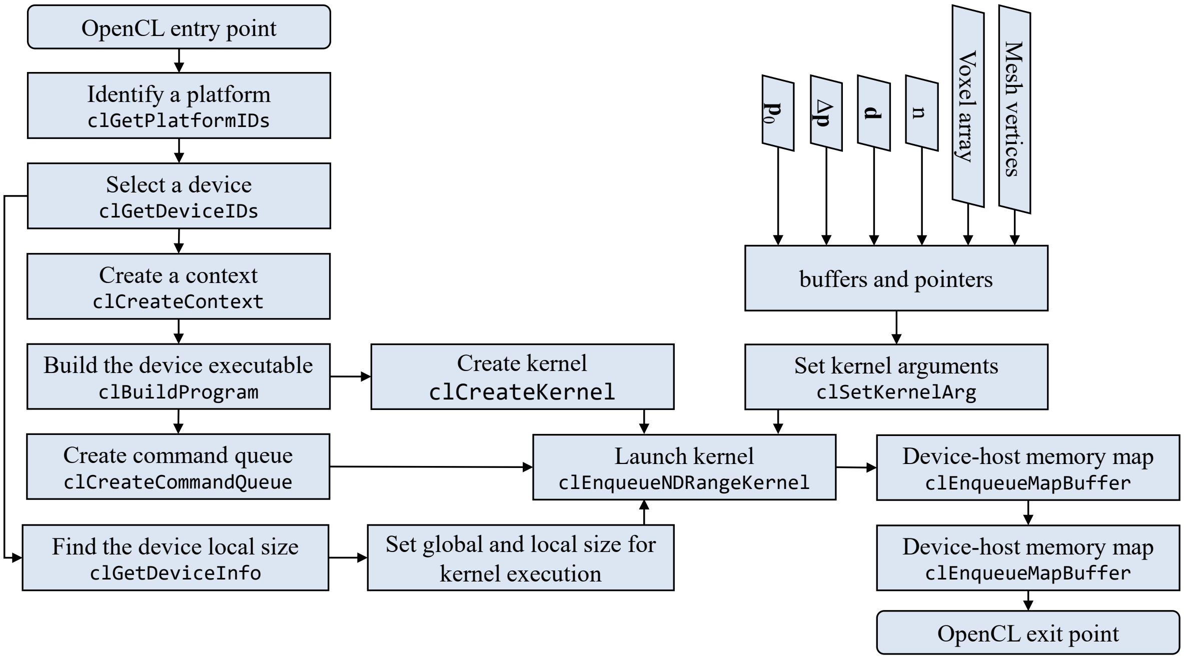


Figure 3.4: Core structure of the OpenCL program used to implement voxelization

selected device. As discussed above, the total number of threads (or global-size in OpenCL terminology) required to complete voxelization is equal to the number of mesh triangles n. However, according to OpenCL standard, the global size must be a multiple of local size. Therefore, the program first determines the maximum local size offered by the device and then sets the global size as a multiple of n. Evidently, OpenCL buffers and pointers pass all the variables and arrays required by the kernel. Once the entire data is passed to the device memory and kernel threads (or work-items in OpenCL terminology) are configured, the command queue launches the kernel. A write-only buffer is used to transfer the output voxel data from GPU global memory to RAM.

OpenCL allows two primary modes of device to host memory transfers: (1) reading from the buffer using `clEnqueueReadBuffer` command and (2) mapping device memory to host memory via `clEnqueueMapBuffer`. The direct comparison of the two options illustrated in Fig. 3.5 implies that memory mapping consistently outperforms the former approach. Interestingly, the advantage of using memory mapping becomes more prominent as the amount of transfer data increases. Given this clear superiority, memory mapping is our method of

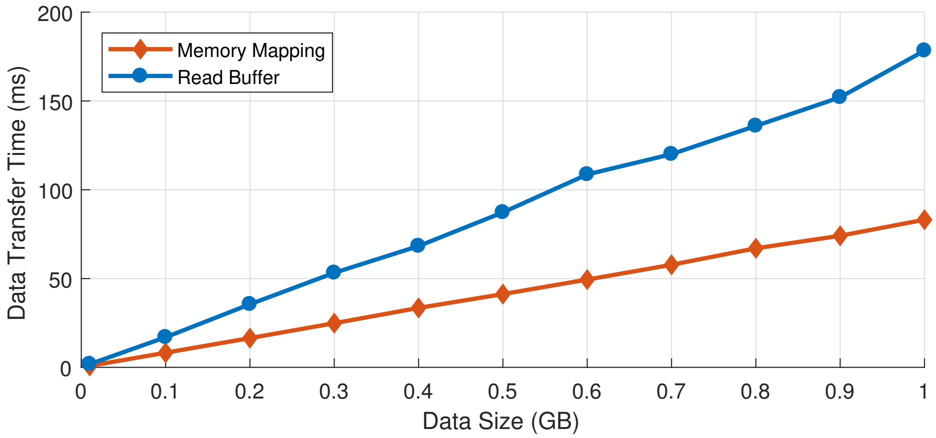


Figure 3.5: Comparison between OpenCL device-host data transfer methods

choice to transfer the output voxel data.

3.6.2 GPU-Based Parallelization

The timings for voxelization of the sample models in different voxel resolutions are measured on different GPUs and reported in Tab. 3.4. These timings are also charted in a normalized form in Fig. 3.6. As it can be noticed from the table, although the teapot geometry has the lowest number of facets, its larger domain has led longer timings compared to the bunny model. Interestingly, although the reamer mesh poses almost 25 times lower number of triangles compared to the dragon model, the running times for the reamer are sometimes more than six times higher than than those obtained for the dragon. This is probably due to topological complexities of the reamer that requires invoking higher number of triangle-box overlap test which significantly prolongs the voxelization time. Overall, while no definite conclusion can be drawn with respect to GPU performances, the data shown in Fig. 3.6 implies that GPU-based parallelization can speed up voxelization anywhere between 73.5% or 3.8 times (reamer on Radeon R7 240) and 99.6% or 260 times (bunny on GeForce 970 GTX) when compared to single-thread CPU-based voxelization. A sample of final voxelized geometry obtained in different voxel resolutions is presented in Fig. 3.7.

It is worth mentioning that the above results also provide a comparison between the perfor-

Table 3.4: Voxelization running time obtained for different models and resolutions using GPUs

Model	Voxel Size	Voxelization Time (ms)						
		GeForce 970 GTX		*GeForce 960 GTXM*		*Radeon R7 240*	*Intel HD Graphics 530*	*Single Thread*
		OpenCL	CUDA	OpenCL	CUDA	OpenCL	OpenCL	
Teapot	0.5	2.12	1.39	2.07	1.28	2.4	6.49	21
	0.25	13.57	8.41	13.1	7.78	14.66	25.9	108
	0.1	166.52	102.61	164.68	93.53	151.95	315.31	1081
Bunny	0.5	0.05	0.11	0.09	0.12	0.11	0.21	13
	0.25	0.08	0.14	0.16	0.15	0.19	0.22	18
	0.1	0.32	0.30	0.66	0.44	0.86	1.81	65
Dragon	0.5	0.31	0.30	0.72	0.53	0.76	1.77	85
	0.25	1.09	0.76	2.75	1.70	2.42	4.04	159
	0.1	10.17	5.57	25.55	12.96	19.44	34.52	757
Reamer	0.5	0.77	0.50	0.75	0.48	0.82	2.46	25
	0.25	5.32	3.45	5.13	2.88	5.56	8.51	61
	0.1	71.89	41.54	68.84	36.54	69.42	109.23	412
Scapula	0.5	2.31	1.34	5.44	2.72	3.8	5.47	420
	0.25	11.00	4.87	25.79	10.06	13.30	24.14	868
	0.1	112.66	43.68	283.97	88.80	82.10	256.34	4913

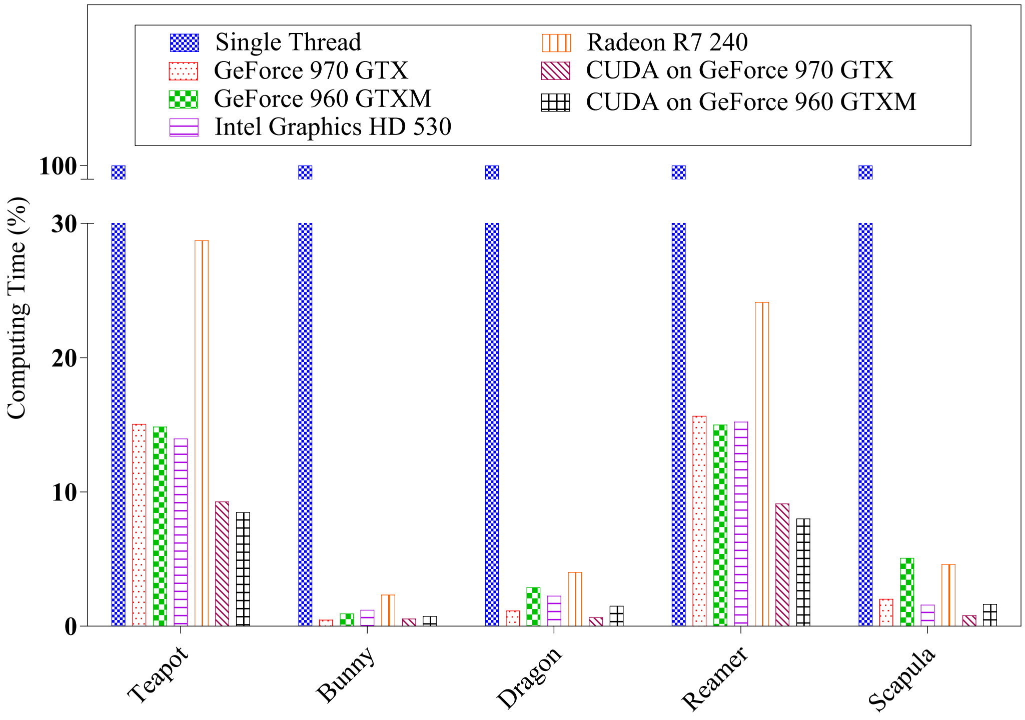
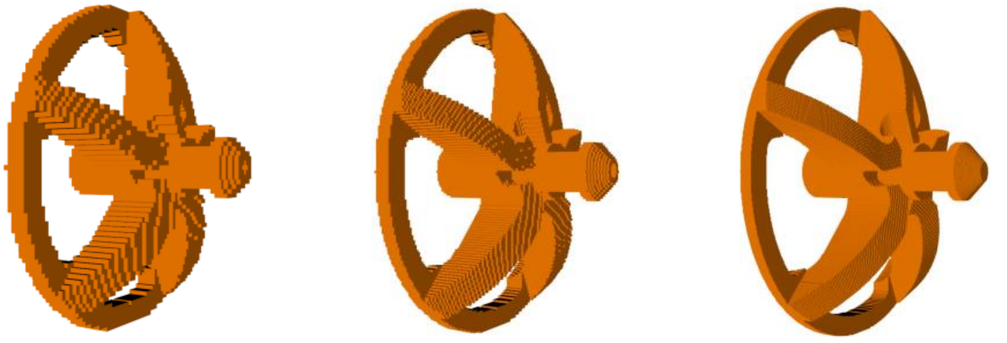


Figure 3.6: Normalized voxelization running time obtained by GPUs



(a) voxel size = 0.5 mm (b) voxel size = 0.25 mm (c) voxel size = 0.1 mm

Figure 3.7: Voxelized representation of reamer in different resolutions

mance of OpenCL and CUDA in the context of voxelization. As it is expected, CUDA appears to perform better for Nvidia GPUs; however, its major drawback remains its incompatibility with other processors produced by other vendors. However, the use of OpenCL still allows remarkable running time reductions compared to the single-thread implementation and its relatively weaker performance compared to CUDA can be neglected in favor of its cross-platform feature.

3.6.3 CPU-Based Parallelization

As OpenCL allows parallelization of a computing task using CPUs, the subsequent results are concerned with evaluating the algorithm performance when using CPU-based parallelization. Table 3.5 compares the running time of our algorithm on difference CPUs with the baseline single-thread results. A normalized representation of these results are also plotted in Fig. 3.8.

As expected, the parallel algorithm results in much lower running times compared to the single-thread implementations. The improvements in the performance are somewhere between 62.7% or 2.7 times (teapot on AMD FX 77K) and 97.5% or 40.1 times (dragon on Core-i7 6700K). It is worth noting that the worst parallel computing results are obtained for voxelizing the teapot using AMD FX 77K. However, this is still more than 60% faster than the running time obtained with the single-thread algorithm running on a relatively much faster CPU. This

Table 3.5: Voxelization running time obtained for different models and resolutions using CPUs

Model	**Voxel Size**	**Voxelization Time (ms)**			
		Intel Core-i7 6700K	*Intel Core-i7 6700HQ*	*AMD FX 77K*	*Single Thread*
Teapot	0.5	3.82	4.85	6.54	21
	0.25	20.84	27.5	36.5	108
	0.1	244.84	290.92	403.63	1081
Bunny	0.5	0.66	0.71	0.91	13
	0.25	1.24	1.91	1.9	18
	0.1	5.39	7.66	6.75	65
Dragon	0.5	2.12	3.94	6.17	85
	0.25	6.8	7.91	18.53	159
	0.1	48.64	55.81	106.87	757
Reamer	0.5	2.09	2.72	2.39	25
	0.25	9.18	11.79	7.37	61
	0.1	94.31	11.79	75.65	412
Scapula	0.5	7.59	9.13	32.5	420
	0.25	26.94	35.33	82.56	868
	0.1	244.71	292.21	610.57	4913

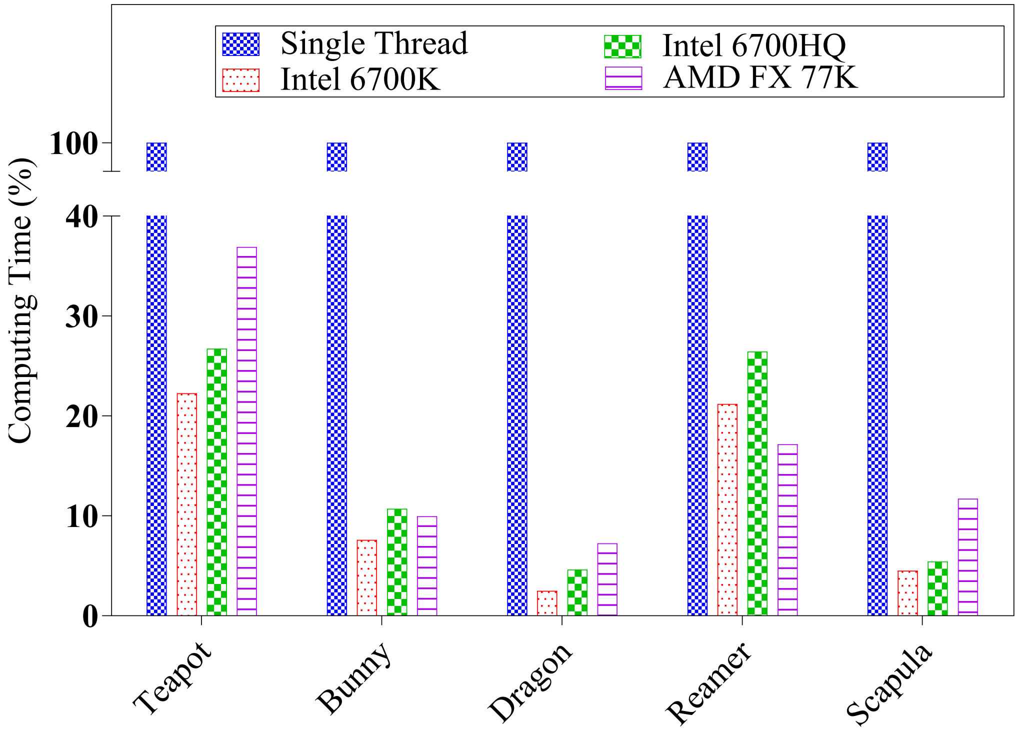


Figure 3.8: Normalized voxelization running time using CPUs

clearly indicate how parallel computing techniques can lead to better performances despite using much cheaper hardware.

Furthermore, a brief comparison of the results shown in Tabs. 3.4 and 3.5 shows that our algorithm runs faster on GPUs compared to CPUs. This confirms with the fact that the architecture of GPUs is generally more suitable for parallel computing compared to that of CPUs, and they result in faster running times for tasks that can be appropriately parallelized.

3.7 Summary

This chapter showed that concurrent execution of an accurate triangle-box overlap test results in fast construction of either 26-separability or 6-separability voxelized models. It also showed that the use of OpenCL-based GPU computing is extremely effective to speed up voxelization by up to 99.6% for some test models, compared to conventional serial implementations. The use of OpenCL also resulted promising results for parallelization on CPUs. Furthermore, this chapter presented results for various models that were voxelized at different resolutions using numerous CPUs and GPUs from all major vendors. These results proved that the developed algorithm is a fast and versatile toolkit that can be utilized to reliably construct voxelized representations from any soups of triangles.

Chapter 4

Collision Detection

Collision detection determines if surgery tool and tissue have come into contact in the virtual environment, and serves as a performance-critical and integral part of every surgery simulator. This chapter presents a new collision detection algorithm that employs the voxelized geometry of bone and tool outlined in Chapters 2 and 3 to rapidly determine the tool-bone intersection, and subsequently update bone geometry to account for bone resurfacing. The algorithm can be regarded as a new variant of the well-known VPS technique that is commonly utilized in the context of bone machining simulators. However, it features a set of new refinements in an attempt to provide higher performance that is capable to handle the large tool-bone intersection volume in glenoid reaming.

4.1 Algorithm Overview

Figure 4.1 illustrates the data-flow in our collision detection algorithm. Using the methods explained in Chapters 2 and 3, a voxelized model of bone (Voxmap) and a point cloud representation of tool (PointShell) are constructed and transferred to GPU global memory during an initialization step. Oriented Bounding Boxes (OBBs) of tool and bone are also computed in this step and will be utilized in the early phase of collision detection.

The running time of our collision detection algorithm involves two primary phases:

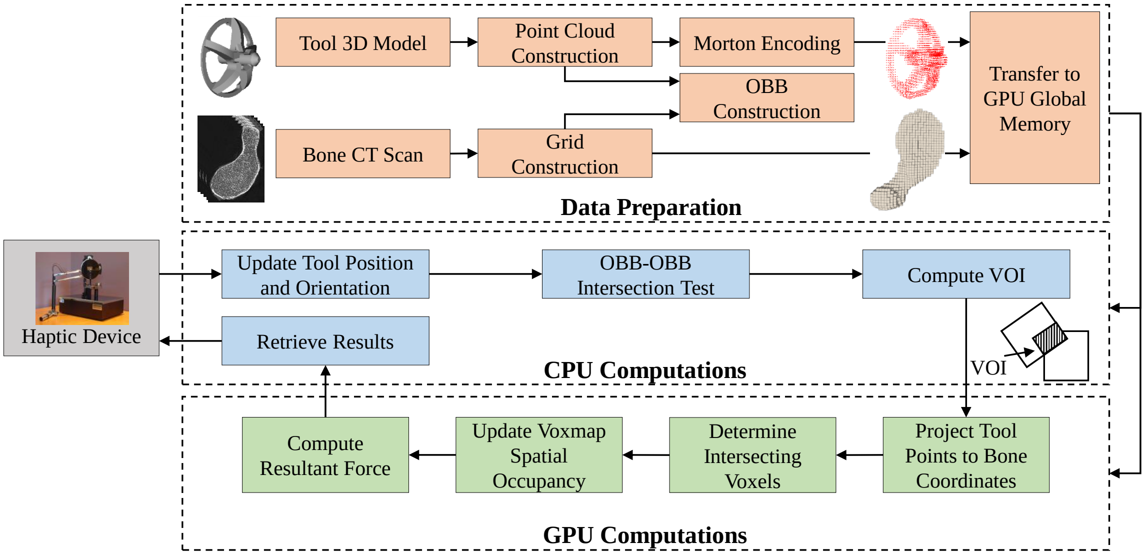


Figure 4.1: Data-flow in our collision detection algorithm

1. The broad-phase which performs a quick intersection test between the OBBs to verify whether objects are about to contact. It also determines a Volume of Interest (VOI) that identifies the possible intersection region between the objects. The advantage of using VOI is that any points and voxels outside of VOI will be culled from the rest of the computations because they are definitely non-intersecting. The broad-phase computations are performed in a single-thread manner using CPU.

2. The narrow-phase which completes an exact intersection test between all the elements lying inside the VOI, determines all intersecting points and voxels, and computes the resultant force feedback. It also updates Voxmap geometry in order to account for bone removal. The narrow-phase relies on GPU parallelization where each thread remains focused on collision of one PointShell element against Voxmap.

4.2 Voxmap Data Structure

The Voxmap data structure refers to the 1D density array $\mathcal{D}$ that is introduced in Chapter 2 to describe the spatial occupancy of the voxel grid $\mathcal{G}$ which is used to implicitly define post-processed CT voxels. Recalling Section 2.2, if the dimension of $\mathcal{G}$ is $d_x \times d_y \times d_z$, the voxel $\mathcal{V}(u, v, w)$ is mapped to the i-th element of $\mathcal{D}$, where

$$i = u + v\ d_x + w\ d_x\ d_y. \tag{4.1}$$

Conversely, given a particular i, the grid coordinates of $\mathcal{V}$ can be computed as

$$w = \left\lfloor \frac{i}{d_x d_y} \right\rfloor,\ v = \left\lfloor \frac{i - w d_x d_y}{d_x} \right\rfloor,\ \text{and } u = i - w\ d_x\ d_y - v\ d_x. \tag{4.2}$$

Since the coordinates of each $\mathcal{V}$ can be inferred from the index of its corresponding element in $\mathcal{D}$, it is sufficient to allocate only a 32-bit float variable for each $\mathcal{D}$'s element in order to store $\mathcal{V}$'s gray-value. In practice, however, the distribution of gray-values is usually much wider than the requirements of force computations. As a result, additional memory savings can be obtained by mapping the gray-values to the range 0-255, where 0 indicates void and 255 represents the highest gray-value present in the CT data. In this manner, only an 8-bit integer is required for each element which can further reduce the amount of memory required for Voxmap. In this set-up, as the gray-value of zero corresponds to void in bone structure, material removal can be replicated by decaying the values stored in $\mathcal{D}$ elements. In addition, the rate of material removal can be also controlled by the slope of the decay function. Therefore, material removal can be readily handled by simple write operations into the memory without any need for data reconstructions. Since arrays guarantee fast random access to their elements, access to the voxel data can be done quickly without any need for expensive grid traversals. Therefore, this way of data representation facilitates accessing to voxels and modifying them efficiently.

4.3 PointShell Data Structure

The voxelization toolkit developed in Chapter 3 is utilized to construct the PointShell. Figure 4.2 exemplifies a point cloud representation of the reamer that is obtained by extracting the centroids of all voxels that are present in its 6-separability voxelized model. While the density array presented in Section 2.2 can be utilized to implicitly describe the points, our algorithm employs bit interleaving techniques to construct a more compact representation. In particular, Morton encoding [190] is utilized to encode the three grid coordinates u, v, w of each voxel into a 32-bit integer. Using the method described in [191], 10 bits are required for each dimension of grid. Therefore, PointShells with resolutions as high as 1024^3 can be stored using 32 bits per point, resulting in 66% memory savings compared to storing all the floats. It is noteworthy that decoding Morton codes requires only a few bit operations; therefore, the added cost of deriving points coordinates from a Morton code is negligible.

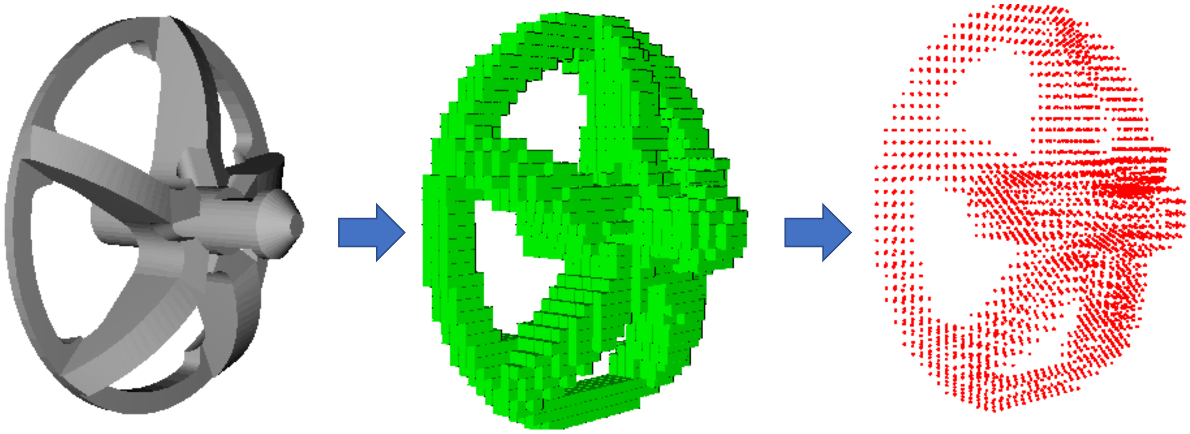


Figure 4.2: Construction of point cloud for reamer

4.4 Broad-Phase Collision Detection

The broad phase serves as an optimization step whose job is to determine a VOI that estimates the intersection volume between two objects and culls all the elements outside of this volume. To achieve this, the broad phase updates the position and orientation of the objects' OBBs based on user interactions. Then, it performs an exact Boolean intersection test using the SAT method presented in [192]. One effective optimization in implementing this test is to express bone OBB in tool coordinates [193]. In this manner, once an intersection occurs, computing the VOI boils down to comparing the boundaries of two boxes that are described in tool coordinates and extracting the overlapping region. As illustrated by Fig. 4.3, the minimum and maximum corners of the VOI are sufficient to describe its boundary in tool coordinates frame. This method is fast and always results in an overestimation of the intersection volume so that no intersecting element is missed.

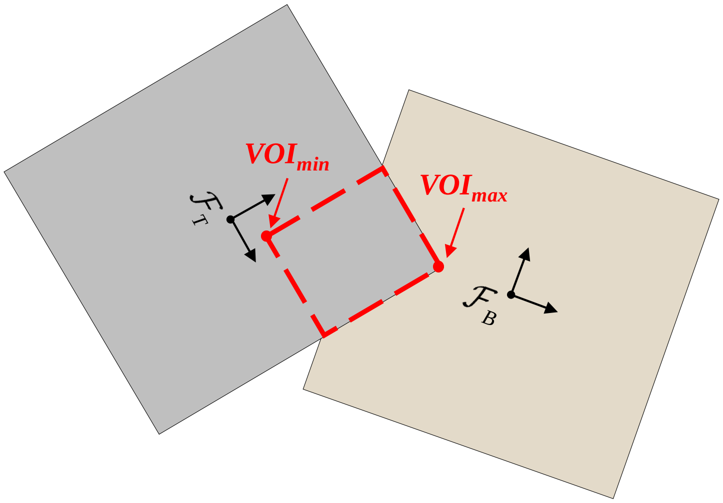


Figure 4.3: A 2D representation of VOI computed in broad-phase

4.5 Narrow-Phase Collision Detection

The narrow-phase queries the exact collision of points and voxels using a GPU kernel whose pseudocode is given by Alg. 4.1. Inputs to the kernel include the PointShell and Voxmap data structures which are denoted by $\mathcal{P}$ and $\mathcal{D}$, respectively and are transferred to GPU global memory once during initialization. The other inputs are the transformation matrix from tool to bone coordinate frames represented by $\mathbf{T}$ and also the VOI boundary. These parameters are required to be updated in runtime and delivered to GPU before every kernel launch.

The kernel launches one thread per PointShell element. Each thread takes one element from the PointShell array, uses Morton decoding to derive the coordinates of a tool point, and checks the obtained coordinates against the VOI boundary. If the point lies outside the VOI, the thread terminates immediately. Otherwise, it continuous by transferring the tool point to bone coordinates system and computing its location within the Voxmap array. If the array returns a non-zero value, then the sample point has penetrated the bone. Therefore, the thread reduces the element's value to account for material removal. Moreover, the thread assigns an elemental force to the intersecting tool point. The resultant force can then be computed by aggregating these forces using atomic operations. Since atomic operations work with integers only, the elemental forces are multiplied by a large number N and casted to the integer type prior to the operation.

```
Input: $\mathcal{P}$, $\mathcal{D}$, $\mathbf{T}$, $VOI_{min}$, $VOI_{max}$
Output: $F$
$i \leftarrow$ thread number
if $i < n$ then
    $e \leftarrow \mathcal{P}(i)$
    $(x, y, z) \leftarrow$ MortonDecoding($e$)
    if $(x, y, z) \geq VOI_{\min}$ & $(x, y, z) \leq VOI_{\max}$ then
        $(x', y', z') = \mathbf{T} \times (x, y, z)$
        $j \leftarrow x' + d_x y' + d_x d_y z'$
        $v \leftarrow \mathcal{D}(j)$
        if $v! = 0$ then
            $\mathcal{D}(j) \leftarrow$Decay($v$)
            $f \leftarrow$ComputeElementalForce()
            $f' \leftarrow$CastToInteger($f$)
            $F \leftarrow$AtomicAdd($f'$)
        end
    end
end
```

Algorithm 4.1: Narrow-phase kernel pseudocode

4.6 Implementation Results and Discussion

This section evaluates the performance of the proposed algorithm in checking the intersection between the reamer and a patient-specific glenoid bone. The algorithm is implemented using OpenCL and tested on GeForce GTX 970 GPU with 4GB video memory and an Intel Core i7 6700K CPU with 16 GB RAM. The main metric to evaluate the algorithm is its running time which is measured in nanoseconds using the `chrono` clock in C++ standard library [188]. The running time is averaged over multiple runs to reduce the noise arising from the randomness of executing system instructions.

4.6.1 Performance in Different Sampling Resolutions

Since the fidelity of VPS-based methods depends upon the sampling resolution, it is necessary to examine the algorithm in different resolutions and find the maximum resolution that guarantees reaching the target 1 KHz refresh rate.

Table 4.1 presents the algorithm running time in different sampling resolutions. The presented timings are the average computing time measured in numerous positions and orientations of the reamer which mostly include severe collisions that may take longer running times. The results indicate that our algorithm manages to maintain < 1 ms running time for PointShells and Voxmaps with resolutions as fine as 1024^3. The running time of the algorithm increases as the PointShell resolution grows due to the added number of GPU threads that the

Table 4.1: Running time (ms) of our collision detection algorithm obtained for various sampling resolutions

Voxmap Resolution	PointShell Resolution				
	64^3	128^3	256^3	512^3	1024^3
128^3	0.19	0.20	0.23	0.34	0.76
256^3	0.19	0.21	0.24	0.34	0.79
512^3	0.20	0.21	0.24	0.34	0.79
1024^3	0.20	0.21	0.27	0.36	0.81

kernel must launch. Interestingly, the change in the Voxmap resolution has a little effect on the computing time. This is because the rise in the number of voxels only results in a slight increase of potential read/write operations that each thread must perform on the Voxmap array. These operations are inexpensive and are usually accelerated by coalesced memory access and the use of cache line; thus, their added overhead is insignificant. As a result, it is possible to reach significantly higher resolutions for bone representation as long as there is enough computing memory to store voxels.

4.6.2 Running Time Break-Down

It is important to break-down the running time of the algorithm and learn about the elapsed time in each step of the algorithm. For this purpose, the running time for different steps of the algorithm is measured for a severe collision scenario and are presented in Tab. 4.2. As expected, a significant portion of the algorithm running time is devoted to kernel execution. This step is the only step whose computing time is subject to change as the sampling resolutions vary. The remaining steps are insensitive to data size and their running time must remain about the same for other resolutions. The table also indicates that our implementation has managed to perform CPU-GPU data transfers in about 24% of the allowable 1 ms interval. This is achieved by wrapping the exchange data into one structure and passing them through a single buffer which is faster than passing the data through separate buffers. Interestingly, the broad-phase

Table 4.2: Running time break-down of our collision detection algorithm in the presence of 103K contact points between PointShell and Voxmap of size 1024^3

Step	Time (μs)
OBB-OBB Intersection Test	0.77
Computing VOI	0.26
Data Transfer from CPU to GPU	169.9
Kernel Execution	568.72
Data Transfer from GPU to CPU	71.03
Total	810.67

computations take only 1 μs which is negligible in the 1 ms time scale.

To assess the effect of the board-phase, especially the VOI, the algorithm running time is measured once by using the VOI and once without it. The results are plotted against the intersection volume in Fig. 4.4 which shows the effectiveness of the VOI in improving the collision detection performance especially when there is a small intersection volume between the objects. In practical machining scenarios, the tool-workpiece intersection often occurs at the boundary of the objects which implies the intersection volume remains small. Consequently, it can be inferred that the use of VOI can accelerate collision detection in glenoid reaming by almost 50% which further adds to the efficacy of our method.

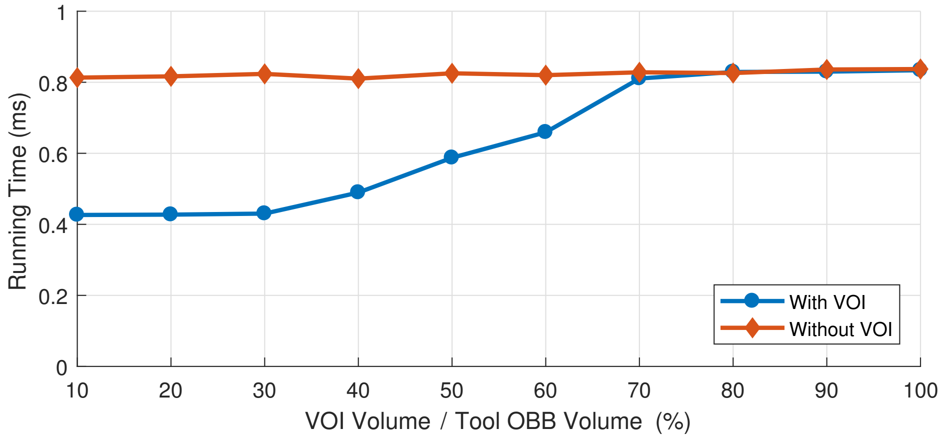


Figure 4.4: Comparative assessment of VOI on collision detection performance

4.6.3 Comparison with Zheng et al.'s Method

The most relevant method to our algorithm is perhaps the GPU-based VPS method developed by Zheng et al. for a tooth drilling simulator [82]. This section compares this method with our algorithm by presenting its implementation results for glenoid reaming simulation. Since the original implementation of this method is not available, it was not possible to reproduce all the details of the method; however, its main features such as the octree-inspired grid traversal, force computation using a reduction algorithm and data structures are well-explained in the paper [82] which helped us to implement them properly on our OpenCL platform.

Table 4.3 reveals the running time of Zheng et al.'s method for identical glenoid reaming scenarios that are used in Tab. 4.1. According to these results, this method is considerably slower than our algorithm and fails to reach <1 ms computing time for high resolutions. One reason for this difference is that, as was shown by Fig. 1.8, the method relies on an octree-like recursive subdivision of the grid to find intersecting voxels. As a result, the workload of each thread is sensitive to the grid resolution which leads to poor performance in case of fine grids. On the contrary, our algorithm directly computes the location of tool points in the Voxmap array. Seeking intersecting voxels in this way is faster and presents constant computational complexity which guarantees a better performance when dealing with large data sets. In addition, Zheng et al.'s method uses a reduction algorithm to compute resultant force. This entails launching unnecessary threads for points that have not immersed into bone, whereas our method relies on atomic operations that are invoked only for intersecting points and thereby requires lower number of operations.

Memory usage is also important in evaluating GPU-based algorithms given the relatively limited capacity of memory even in the state-of-the-art video cards. For this reason, Tab. 4.4 compares the amount of memory the two algorithms use to store PointShell and Voxmap. It is clear that our algorithm offers much lower memory usage compared to Zheng et al.'s method. In particular, as pointed out in section 4.2, our algorithm requires only an 8-bit integer for a voxel and a 32-bit integer for each tool point. By contrast, Zheng et al.'s method stores each voxel data in a structure constituted by a 32-bit float for its gray-value and a Boolean variable

Table 4.3: Running time (ms) for Zheng et al.'s collision detection algorithm obtained for various sampling resolutions

Voxmap Resolution	PointShell Resolution				
	64^3	128^3	256^3	512^3	1024^3
128^3	0.33	0.34	0.40	0.68	1.70
256^3	0.86	0.89	1.03	1.22	1.86
512^3	5.20	5.14	5.29	5.52	6.70

Table 4.4: Required memory for PointShell and Voxmap in our method compared with Zheng et al.'s method

Resolution	PointShell Required Memory (MB)		Voxmap Required Memory (MB)	
	Our Method	*Zheng et al.'s Method*	*Our Method*	*Zheng et al.'s Method*
64^3	0.02	0.05	0.26	1.31
128^3	0.07	0.22	2.10	10.49
256^3	0.30	0.89	16.78	83.89
512^3	1.19	3.56	134.22	671.09
1024^3	4.75	14.26	1073.74	5368.71

to flag whether it is collided or not. This method also requires three 32-bit floats for the 3D coordinates of each tool point. As a result, it is not possible to run Zheng et al.'s method for Voxmap at 1024^3 resolution because this method requires more than 5.3 GB computing memory which exceeds the amount of memory available in our test hardware.

Overall, it is clear that while Zheng et al.'s method has been successfully applied in the filed of tooth surgery, the complexity of glenoid reaming procedure and its need to reach fine resolutions reveals the limitations of this method. Our method outperforms this method by offering faster running time and lower memory usage, making it possible to reach resolutions that Zheng et al.'s method cannot handle.

4.6.4 Comparison with Yau et al.'s Method

Another method relevant to our algorithm is the method developed by Yau et al. in the context of CNC simulation [142]. As discussed in section 1.6.3, the original implementation of this method is sluggish and is only suitable for offline computations. This section develops a GPU-based variant of this method which can be comparable to our algorithm. Toward this end, the Voxmap structure and the material removal logic remains as the same as our algorithm. For a rotating reamer, the area swept by the cutting lips shape a spherical cap which can be described by an implicit equation. This equation replaces the PointShell in our algorithm and can be readily updated in every computing frame according to the position and orientation of the reamer. The GPU kernel dedicates one thread per Voxmap element where each thread queries the collision of a voxel against the reamer equation and modifies gray-values in case of collisions. A VOI is also constructed in Voxmap coordinates to cull threads that correspond to voxels out of the intersection volume of the objects' OBBs.

Table 4.5 presents the average running time of this method in scenarios similar to the ones used in Tab. 4.1. While the method exhibits acceptable performance in low resolutions, its running time grows significantly for higher resolutions due to the increased number of threads the kernel must launch. As a result, this method is also slower than our algorithm. Nevertheless, its memory usage is slightly better because it does not require storing a PointShell.

Table 4.5: Running time for Voxmap-based parallelization algorithm obtained for various resolutions

Voxmap Resolution	Average Running Time (ms)
128^3	0.23
256^3	0.50
512^3	2.19
1024^3	13.67

4.7 Summary

This chapter established that concurrent projection of PointShell elements to the coordinates of Voxmap grid leads to a fast tool-bone collision detection algorithm. It showed that parallelization of the projection tasks results in low sensitivity of the algorithm running time to Voxmap resolutions. As a result, the algorithm can easily deal with extremely fine bone voxel resolutions, as long as they can fit to the computing memory. The use of broad-phase computations was also shown to be notably useful. While these computations do not take more than a few microseconds, they allow to effectively cull unnecessary threads and improve the algorithm running time by up to 50%. Regarding the data structures that were used to describe the virtual objects, this chapter proved that not only did the implicit definition of voxels lend well to fast collision queries, but also it facilitated modifying voxel gray-values and replicate material removal in real-time. The chapter also showed that the use of Morton encoding can result in significant memory savings for PointShell representation, without compromising the algorithm performance. The developed algorithm was compared with some of the recent GPU-based variants of the VPS method where it was demonstrated that our algorithm presents exceptional performance and memory management advantages compared to the existing similar approaches. Overall, the proposed algorithm is presently one of the most efficient variants of the VPS method with the ability to update bone geometry in real-time.

Chapter 5

Modeling and Simulation of Gleonid Reaming

The knowledge of the mechanics of glenoid reaming is quintessential to compute realistic haptic and graphic feedback for simulating this procedure. As discussed in Section 1.6.1, the existing literature lacks an in-depth study regarding modeling of this unique bone machining operation. For this reason, this chapter outlines a statistical analysis of the experimental results obtained in a robot-driven glenoid reaming study [106] that was conducted in conjunction with the present thesis. This analysis provides a model to describe thrust-feedrate relation and vibrations observed during glenoid reaming experiments. Derivation of the model completes all the tools required to perform simulation of glenoid reaming. As a result, this chapter also outlines how to effectively integrate all the simulator components and perform simulations.

5.1 Calibration Experiments

Robot-driven glenoid reaming was used to perform calibration experiments since it allows to accurately repeat a procedure on different specimens. As shown in Fig. 5.1, the experiment set-up included a Kuka Light-Weight Robot (LWR) IV to move the reamer, a load cell (Nano 25E, ATI Industrial Automation, North Carolina, USA) mounted between the specimen pot

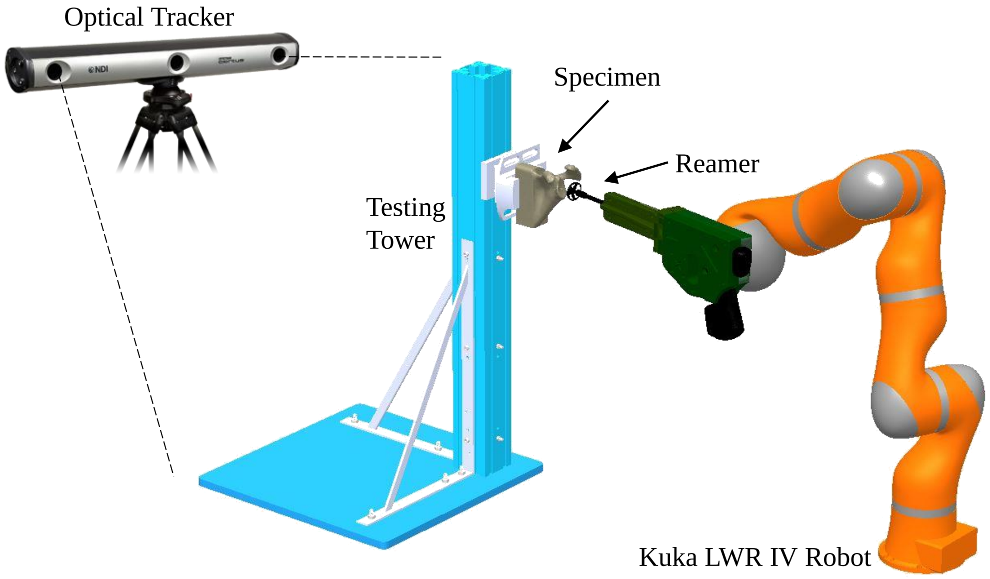


Figure 5.1: Calibration experiments set-up

and testing tower to measure reaming forces, optical trackers (Optotrak Certus, Northern Digital Inc., Ontario, Canada) to track the reamer motion, and also an accelerometer (Endevco 42A16, Meggitt Sensing Systems, Fribourg, Switzerland) to record vibrations. The reamer employed throughout the experiments was a nipple-guided spherical reamer (Zimmer Biomet, Indiana, USA) connected to a orthopaedic surgical drill (Synthes Small Battery Drive, DePuy Synthes, Massachusetts, USA) with 156 rpm spindle speed. Six freshly frozen human cadaveric scapulae with mean age of 66.2 ± 12.6 years were used. The specimens were kept at room temperature for 12 hours and soaked in saline for 2 hours prior to reaming to ensure their mechanical properties are well-maintained. Clinical CT images of the specimens were obtained before the experiments. In addition, 3D model of the post-reamed scapulae were acquired using a laser scanner (Space Spider, Artec 3D, Luxembourg).

To properly mimic the real glenoid reaming operation performed by surgeons, the robot was programmed in force-control mode. The command force trajectory was set to linearly increase from zero to a desired value and maintain at this value until the reamer reaches a predetermined reaming depth. The desired final force was set to 52 N which is the average feed-force applied by surgeons measured during porcine glenoid reaming in [105]. The prescribed force trajectory was kept identical for all the experiments; therefore, the reamer displacements and vibrations can be used to describe the characteristics of glenoid reaming.

The experiments were performed in two phases. In the first phase, only a potion of cortical layer was reamed without violating the cancellous bone. The reaming depth is determined by a surgeon through analyzing the cortical layer thickness of each specimen. Also, the position and orientation of the reamer was determined by the surgeon to ensure robot-driven glenoid reaming matches the clinical practice. It is worth mentioning that the cartilage layer on glenoid face was removed by the surgeon beforehand to ensure that only cortical bone is encountered during the experiment. Upon completion of the first phase, the rest of cortical layer was removed to perform the second phase which involved reaming the cancellous bone by a fixed depth of 2 mm. The above experimental scenario was inferred after several trials using artificial bones. A detailed explanation in this regard can be found in [106].

5.2 Thrust-Feedrate Relation

Figures. 5.2 and 5.3 illustrate the average and standard deviation of thrust and reamer displacement measured from reaming cortical and cancellous bone for all the specimens. The plots reveal low variation of force and displacement trajectories in both cortical and cancellous bone regions, especially when the command force of the robot is stabilized at 52 N. In particular, at this stage, the average value of thrust is 53.17 ± 4.26 N for cortical bone and 51.87 ± 4.45 N for cancellous bone. In addition, the average feedrate in these bone regions are 0.033 ± 0.007 mm/s and 0.24 ± 0.04 mm/s, respectively. These values clearly indicate a high level of repeatability in the experiments.

One way to employ the above data in modeling of glenoid reaming is to take a mechanistic approach similar to the one described for bone drilling [60], [86]. As discussed in section 1.6.1, although mechanistic models can theoretically predict instantaneous thrust as a function of displacement, in practice, the predictions are inaccurate and limited to qualitative behaviour. This

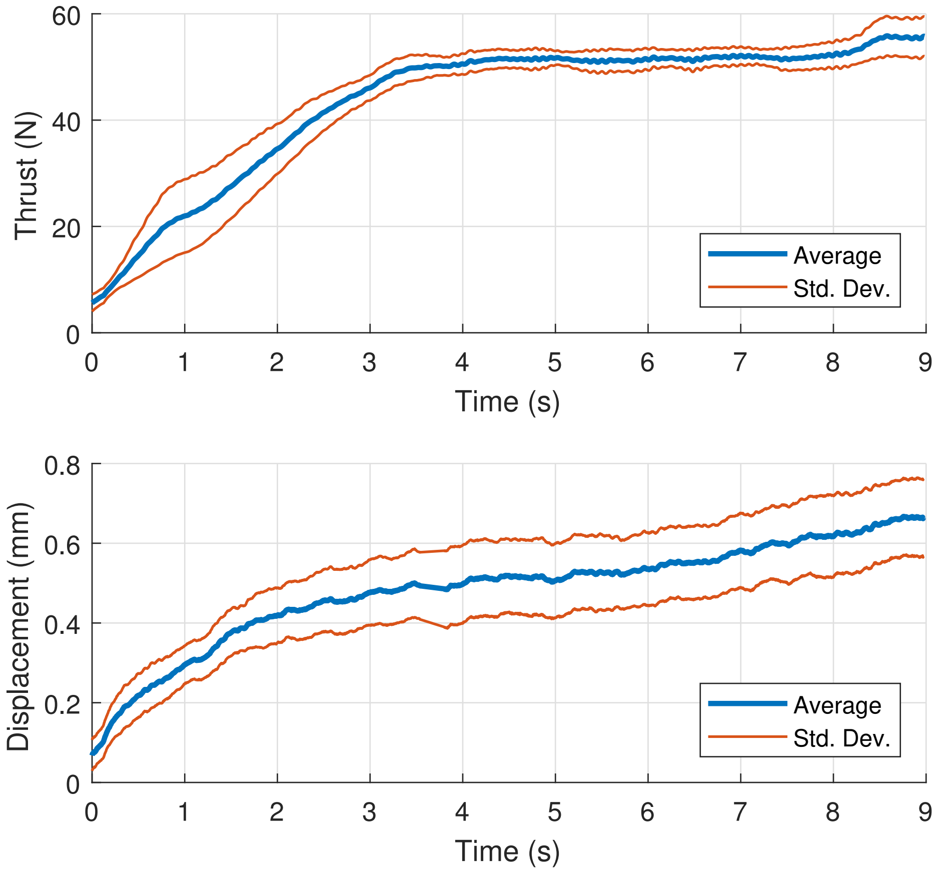


Figure 5.2: Average thrust-displacement characteristics of glenoid reaming in cortical bone

is because of the anisotropic material properties of bone and lack of well-founded theoretical methods in modeling of bone removal. This can become even more challenging in the case of glenoid reaming because the reamer spindle speed (156 rpm) is much lower than drilling and bone removal can be a result of material scraping. Therefore, theoretical methods in metal cutting may not be valid in this case.

Another approach is to assess the statistical relations between various factors monitored during the experiments and develop an empirical model to relate these factors to thrust-feedrate characteristics of glenoid reaming. As shown in the existing studies [76]–[78], it is expected that such approach will provide a fair prediction of the qualitative behaviour of glenoid reaming within the observation range accounted in the calibration experiments. Empirical approaches

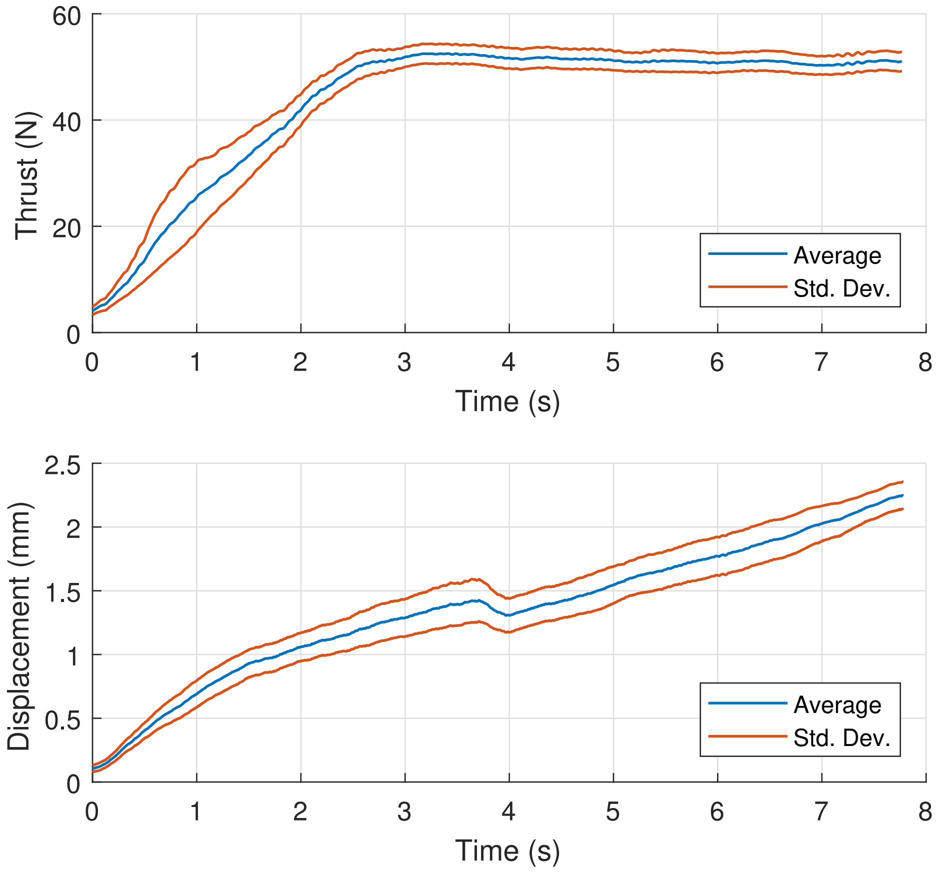


Figure 5.3: Average thrust-displacement characteristics of glenoid reaming in cancellous bone

can also result in computationally simpler models which is important for real-time haptic applications. In addition, considering the limitations of mechanistic modeling, there is no strong evidence that empirical approaches can lead to inferior results compared to their mechanistic counterparts. Based on these observations, an empirical approach is chosen as the method of choice for modeling of glenoid reaming. However, the mathematical derivations required for mechanistic modeling of glenoid reaming are additionally derived in Appendix A.

One factor that plays an important role in the modeling of bone machining operations is the tool-bone contact volume. In operations such as drilling, the contact area is small and can be estimated using the drill bit position and a piece-wise linear approximation of bone surface [60], [86]. However, this becomes a formidable challenge in glenoid reaming due to the irregular 3D shape of glenoid surface and large contact volume between reamer and bone. Moreover, the rate of reamer progression is very low especially in cortical bone (0.033 ± 0.007 mm/s) which needs ultra-precise measurements to capture the contact area variations in short time intervals. As a result, this information is not available in our calibration experiments. Since understanding the instantaneous thrust-feedrate characteristics of glenoid reaming is unfeasible without the knowledge of the contact volume, the above experimental results can only be used to model the average behaviour of the operation.

Although bone machining characteristics cannot be fully related to a single mechanical property, some studies have reported close relations between machining forces and bone density [194]–[196]. For this reason, 3D models of post-reamed scapulae were acquired using a laser scan and compared to the previously obtained CT scans of the specimens to measure the density of bone removed in each experiment.

Table 5.1 presents the density D information for each scapula as well as the average feedrate f and apparent machining stiffness k that are observed in each experiment. Although statistical analysis of this data shows no linear relation between the density and the apparent machining stiffness, it reveals a linear correlation between the density and the feedrate, both in cortical bone ($\rho = -0.7852, p = 0.0642$) and in cancellous bone ($\rho = -0.9384, p = 0.0056$). A related point to consider is that cortical bone is associated with a weaker linear correlation compared to the cancellous bone. This observation can be partially due to the robot compliance that causes

Table 5.1: Bone density, reamer feedrate and appearant machining stiffness observed in calibration experiments

Specimen No.	Cortical Bone			Cancellous Bone		
	$D\ (g/cc)$	$f\ (mm/s)$	$k\ (N/mm)$	$D\ (g/cc)$	$f\ (mm/s)$	$k\ (N/mm)$
1	1.52	0.037	58.24	0.55	0.25	44.06
2	1.54	0.035	83.28	0.41	0.28	40.70
3	1.61	0.021	84.34	0.72	0.16	50.42
4	1.40	0.038	81.06	0.45	0.26	48.42
5	1.58	0.026	53.66	0.62	0.23	27.28
6	1.30	0.039	83.10	0.48	0.25	39.45

larger retrogressions when interfacing with the stiffer tissue i.e. cortical bone.

Using linear regression analysis, the relation between feedrate and density in cortical bone takes the following form

$$f = -0.05D + 0.11, \tag{5.1}$$

where the sum of squared residuals (R^2) is 0.62 (Fig. 5.4). Similar equation can be derived for cancellous bone as follows

$$f = -0.34D + 0.42, \tag{5.2}$$

with $R^2 = 0.88$, as illustrated by Fig. 5.5. It is important to note that the above equations are derived based on experimental data that are collected using 52 N feed-force. Consequently, the above equations must be appropriately scaled to predict feedrate when a different feed-force is applied. Considering the reamer displacement trajectories in Figs. 5.2 and 5.3, it appears that there is an approximately linear trend between thrust and feedrate. Therefore, a linear scale should provide a fair modeling accuracy. However, better results can be obtained from additional experimental results performed with different feed-forces.

Overall, given the level of complexity and challenges in modeling of glenoid reaming, the above analyses appear to adequately describe the qualitative traits of this bone machining operation. Comparing the Eqs. (5.1) and (5.2), an interesting trend is observed that the reamer velocity in cancellous bone is on average 6.2 ± 0.08 times faster than that in cortical bone. This can be effectively replicated with a haptic device and can help trainees learn the different characteristics of reaming cortical and cancellous regions. In addition, the obtained equations are computationally simple and can be easily integrated with the collision detection algorithm developed in Chapter 4 without introducing any noticeable overhead in haptic rendering computations. Moreover, since bone density can be estimated from clinical CT images [152], [153], the need for high-resolution CT data is alleviated which in turn facilitates performing patient-specific simulations.

Since the Eqs. (5.1) and (5.2) predict feedrate as a function of feed-force, they can be

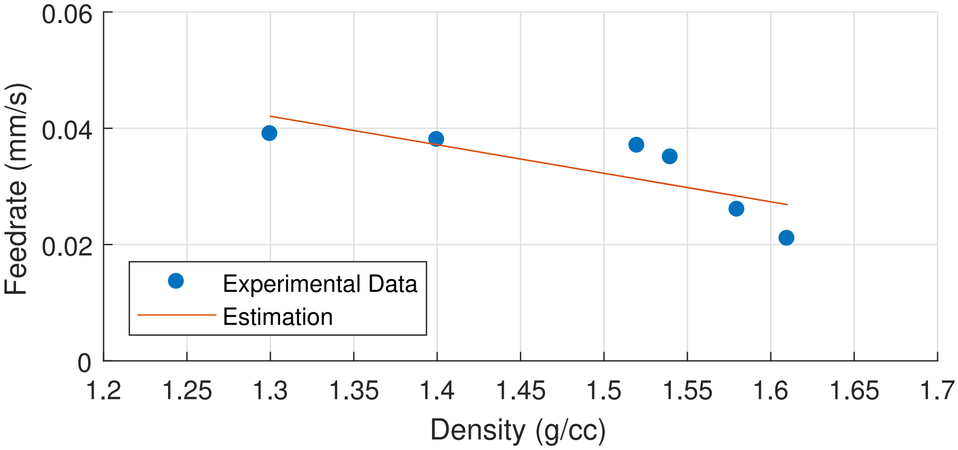


Figure 5.4: Prediction of feedrate as a function of density in cortical bone

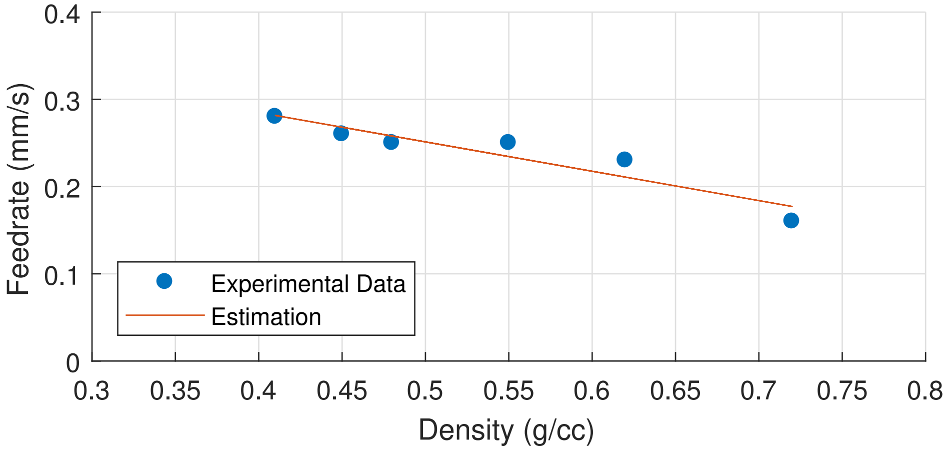


Figure 5.5: Prediction of feedrate as a function of density in cancellous bone

directly utilized for *admittance display* haptic devices. This type of haptic devices sense the force applied by the operator and constrain the operator's position to match the appropriate deflection of a simulated object or surface in a virtual world, whereas the *impedance display* devices sense the operator's position and generate a proper force feedback [110]. Therefore, a proper haptic rendering algorithm for an impedance display device must predict the feed-force as a function of feedrate. This can be addressed easily by interpreting the Eqs. (5.1) and (5.2) in the form of a damper where the damping coefficient varies by bone density.

5.3 Vibration

Vibration data was recorded for five specimens at 22050 Hz. A high-pass fourth-order Butterworth filter with a cut-off frequency of 0.2 Hz is employed to remove the gravity acceleration included within the data, as suggested in [197]. Since human haptic perception is limited to frequencies below 400 Hz [198], a low-pass filter with a cut-off frequency of 500 Hz is further utilized to eliminate imperceptible high-frequency vibrations.

In order to analyze time-domain characteristics of the vibrations, the Root Mean Square (g_{rms}) and Peak-to-Peak (g_{pk-pk}) values of the filtered signals are computed. These values are presented in Tab. 5.2 alongside the density of specimens that are repeated from Tab. 5.1. The results indicate a large variation of g_{rms} and g_{pk-pk} from one specimen to the other. Considering the average values, it can be concluded that the average g_{rms} and g_{pk-pk} for cortical bone are 0.3 ± 0.09 g and 4.93 ± 1.06 g, respectively, whereas the same values for cancellous bone are actually higher with $g_{rms} = 0.4 \pm 0.07$ g and $g_{pk-pk} = 5.28 \pm 0.95$ g. This unexpected observation can be a result of the discontinuty in material properties of bone while reaming cancellous bone because there is a layer of cortical bone remaining at the periphery of the glenoid face which can escalate the vibrations even though the material close to the reamer axis pose lower stiffness. Concerning with the effect of bone density on vibrations, no linear correlation is found between density with neither g_{rms} nor g_{pk-pk}.

Concerning with frequency-domain analysis of vibrations, Figs. 5.6 and 5.7 illustrate the

Table 5.2: Time-domain metrics measured for vibrations during reaming different specimens

Specimen No.	Cortical Bone			Cancellous Bone		
	D (g/cc)	g_{rms}	g_{pk-pk}	D (g/cc)	g_{rms}	g_{pk-pk}
1	1.52	0.45	6.27	0.55	0.45	5.92
2	1.54	0.25	3.98	0.41	0.34	4.72
3	1.61	0.31	5.98	0.72	0.37	5.09
4	1.40	0.17	3.57	0.45	0.51	3.96
5	1.58	0.34	4.83	0.62	0.35	6.70

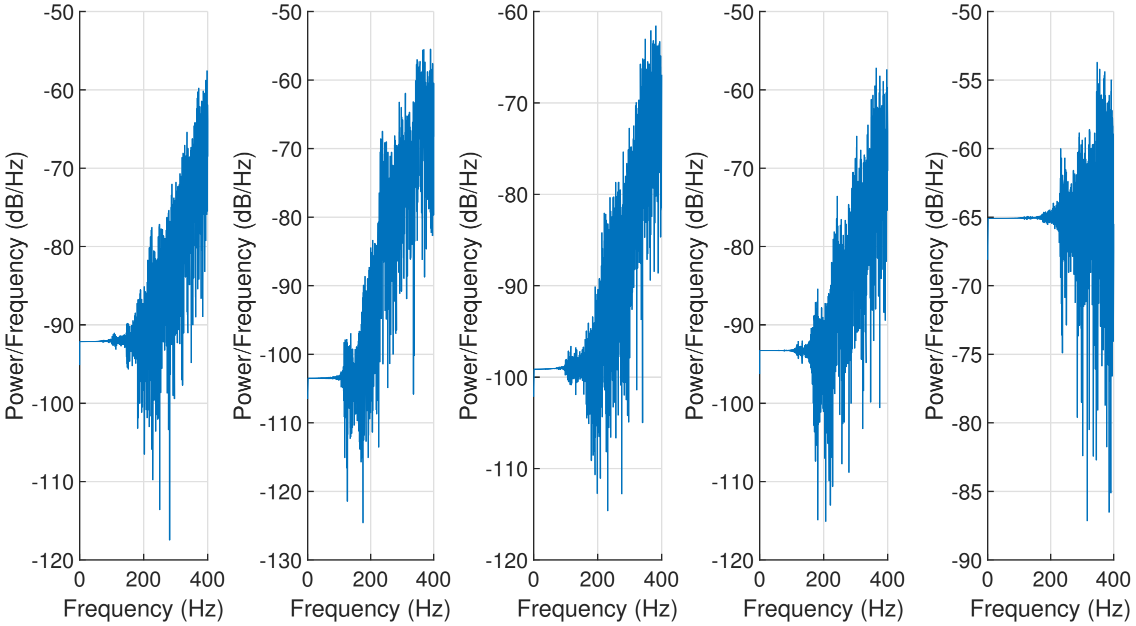


Figure 5.6: Periodogram of vibrations during reaming cortical bone in five different specimens

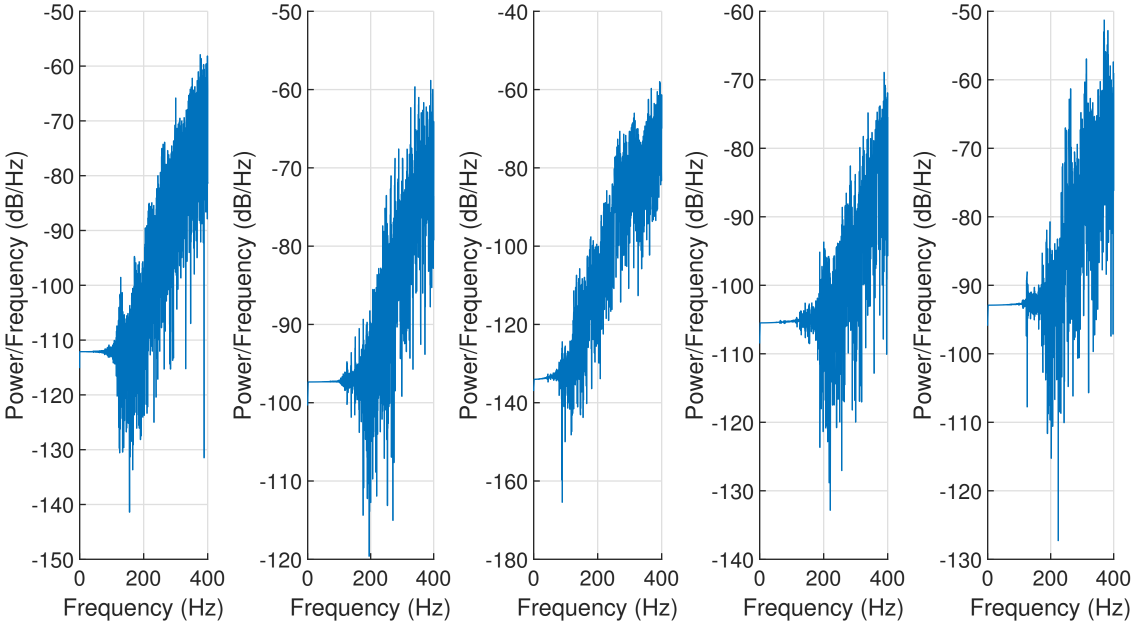


Figure 5.7: Periodogram of vibrations during reaming cancellous bone in five different specimens

periodogram of the filtered signals. Although these plots indicate different power spectrum from one specimen to the other, they unanimously indicate that the highest power of vibrations has occurred at the high spectrum of frequencies. Nevertheless, no distinct dominant frequency can be observed which conforms with the frequency-domain analysis presented for reaming porcine glenoid in [105].

Since there is no distinct peaks in frequency spectrum of the data, analytic replication of a realistic vibration feedback is unfeasible. Therefore, for the purpose of adding vibrations to haptic feedback, a portion of recorded vibration signals for both cortical and cancellous bone are selected and downsampled to 1 KHz to match the target haptic device refresh rate. The signals should be also scaled to represent the average g_{rms} in each region and can be played during haptic simulation. Figures 5.8 and 5.9 illustrate two sample vibration signals that can be applied in simulation of glenoid reaming for cortical and cancellous regions, respectively.

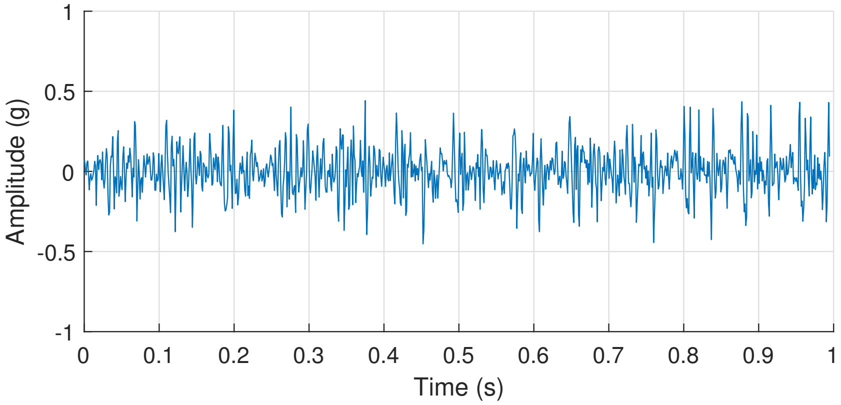


Figure 5.8: Sample vibration signal prepared for haptic simulation of reaming cortical bone

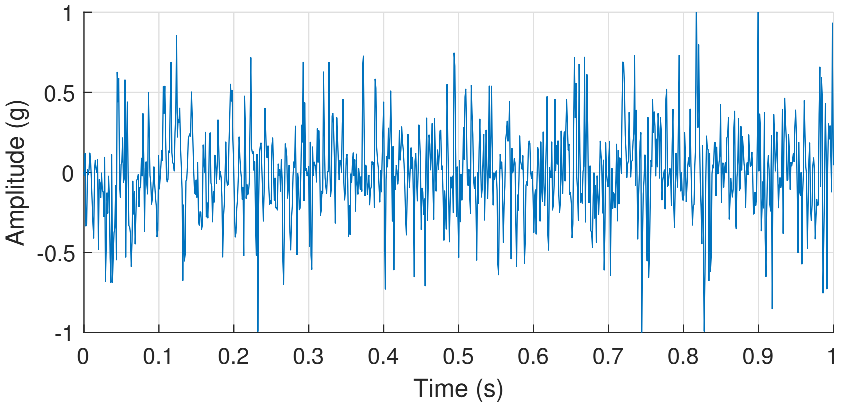


Figure 5.9: Sample vibration signal prepared for haptic simulation of reaming cancellous bone

5.4 Simulation of Glenoid Reaming

The results presented in Sections 5.2 and 5.3 can be combined with the computational tools developed in Chapters 2 – 4 to simulate glenoid reaming. As discussed in Section 1.6.2, the simulator software employs OpenCL, OpenGL and OpenHaptics APIs to manage GPU-based computations, graphics rendering and haptic interactions, respectively. The latter serves as merely an example haptic API and can be easily replaced by different APIs such as CHAI 3D. As the devices that are presently supported by OpenHaptics are limited to impedance displays, the subsequent developments is focused on this type of haptic devices. However, they can be readily extended to admittance displays, as well.

5.4.1 Integration of all Simulator Components

As shown by Fig. 5.10, the software starts by importing a previously segmented CT image of bone as well as a 3D mesh model of reamer to construct the Voxmap and PointShell data structures in user-specified resolutions. These data structures are then transferred to GPU global memory and will be used in every computing frames. Once the data initialization is completed, the user can start the simulation by interacting with the haptic device. The haptic rendering frames consist of updating the reamer positions based on the displacements of the haptic device end-effector and running collision detection followed by computing the force feedback. The graphic rendering frames contain running marching cubes for the bone Voxmap that is updated during collision detection, data exchange between OpenCL and OpenGL buffers and drawing the reamer and bone surface meshes. As pointed out in Section 1.6.6, the marching cubes algorithm is adopted from the online code available in [183].

The software has the duty to manage the above operations in a way that the target frame rates for haptic and graphic rendering loops are reliably maintained. Since time-consuming operations such as collision detection and marching cubes are accelerated using GPU computing, there is no major bottleneck to attain the target frame rates. However, the software must

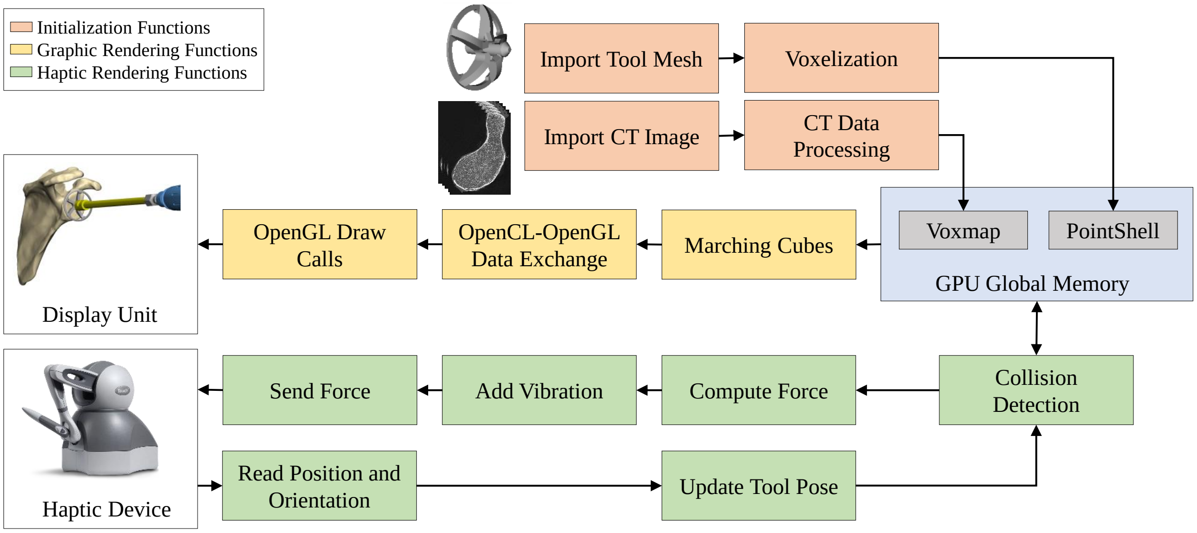


Figure 5.10: Data-flow in our simulator software

schedule these tasks appropriately such that collision detection and marching cubes are completed at least once within 1 ms and 33.3 ms time intervals, respectively. To address this, the OpenCL option `CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE` is utilized to allow GPU to enqueue kernels without waiting for their completion. In this manner, several kernels can be launched concurrently and the timings of the haptic and graphic rendering loops can be controlled by the OpenCL host application. This scheduling can be easily handled in OpenHaptics because this API generates at least two threads in host application: one high-priority thread dedicated for haptic rendering that can enqueue the collision detection kernel, and another thread to handle the rest of the computations including launching the marching cubes kernel. This can also be addressed using a single-thread host application and a timer using the algorithm given by Alg. 5.1.

It is worth mentioning that the OpenCL-OpenGL interoperability is also utilized in this software. This feature allows OpenCL to pass the ownership of marching cubes output to OpenGL without a need for neither CPU-GPU data transfers nor copying data within GPU which can save significant computing time for the graphic rendering loop.

```
$F_{haptics} \leftarrow 1000$ , $F_{graphics} \leftarrow 30$
$T_{haptics} \leftarrow \frac{1000}{F_{haptics}}$ , $T_{graphics} \leftarrow \frac{1000}{F_{graphics}}$
while Simulator engaged do
    $t^{current} \leftarrow$ GetCurrentTime()
    $\Delta t_{haptics} \leftarrow t^{current} - t^{previous}_{haptics}$
    if $\Delta t_{haptics} > T_{haptics}$ then
        PerfromHapticsComputations()
        $t^{previous}_{haptics} \leftarrow t^{previous}_{haptics} + T_{haptics}$
    end
    $\Delta t_{graphics} \leftarrow t^{current} - t^{previous}_{graphics}$
    if $\Delta t_{graphics} > T_{graphics}$ then
        PerfromGraphicsComputations()
        $t^{previous}_{graphics} \leftarrow t^{previous}_{graphics} + T_{graphics}$
    end
end
```

Algorithm 5.1: Pseudocode for scheduling haptics and graphics computations

5.4.2 Simulation with a Haptic Device

In order to evaluate the effectiveness of the simulator software, experiments are performed using a haptic device. Generally, the use of admittance display devices is preferable for a glenoid reaming simulator due to relatively higher range of force and stiffness supported by this type of haptic devices. However, our experiments use the PHANToM OMNI haptic device which is known as a simple and cost-effective impedance display [199]. This device can exert only 3.3 N force feedback in its nominal position and its maximum stiffness is limited to 2.31 N/mm. Therefore, considering the level of force and stiffness in glenoid reaming, this device is not suitable to provide a realistic haptic feedback for this operation. Nevertheless, it is still a viable option to evaluate the computational efficiency of our methods.

Towards the end, the Voxmap is constructed using clinical CT image of one of the specimens that was utilized in the calibration experiments. The size of CT image voxels were $0.668 \times 0.668 \times 0.625$ mm^3; however, each voxel is subdivided by a factor of eight in each direction in order to construct a fine Voxmap. As a result, the Voxmap grid size is $600 \times 384 \times 464$ with more than 12.7 M filled cells. The average feedrate measured in cortical and cancellous layers of this specimen are 0.026 mm/s and 0.23 mm/s, respectively. Therefore, the damping coefficient for these layers are set to 2000 Ns/mm for cortical bone and 267 Ns/mm for cancellous bone. The vibration signals depicted in Figs. 5.8 and 5.9 are also applied; however, the amplitude of force feedback is scaled down by a factor of 1/25 to ensure the force feedback amplitude does not exceed the limits of the device. Furthermore, a PointShell with almost 213 K sample points is constructed using a 6-separability voxelized model of the reamer obtained with 0.1 mm voxel size. The hardware utilized for the test include a GeForce 970 GTX GPU with 4 GB video memory and Core-i7 6700K CPU and 16 GB RAM. The computing time is measured in microseconds using the high-resolution `chrono` clock available in C++ standard library. The operating system used for the tests is Windows 10.

Figure 5.11 shows the computing time recorded for each haptic rendering frame during five seconds of glenoid reaming simulation. It is evident that for majority of the simulation time, the computing time is below 1 ms, with an average of 314.89 ± 158.61 μs. It should be noted that the computing time exceeds the 1 ms threshold on a few occasions. This can be partially due to the randomness of executing system instructions in Windows which can be fixed by using precise timers such as Windows Multimedia Timers [200]. Furthermore, the average computing time recorded for graphic rendering loop is 21.74 ± 0.46 ms which is considerably lower than the target 33.33 ms. Figure 5.12 depicts the computing time for each frame throughout

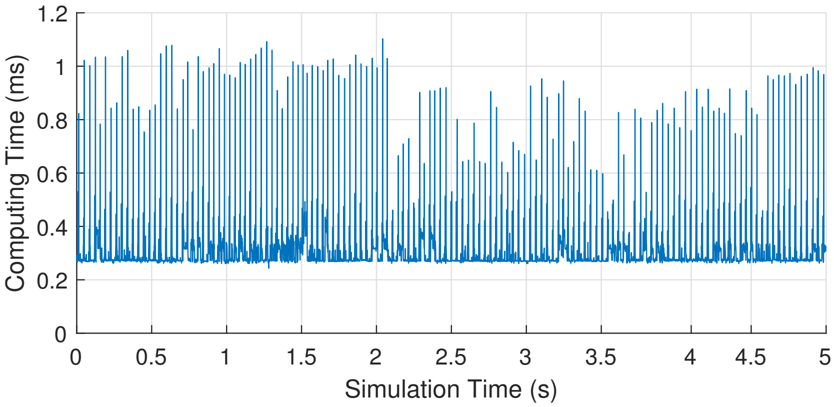


Figure 5.11: Computing time measured for haptic rendering frames

the simulation which clearly indicates the software is capable to consistently maintain 30 FPS frame rate. While the haptic feedback realism is negatively affected by the PHANToM OMNI limitations, the simulator software allows smooth graphical simulation of glenoid reaming and provides high-fidelity visual feedback of the operation. Figure 5.13 illustrates snapshots of the graphical environment and glenoid which has gone through a reaming operation.

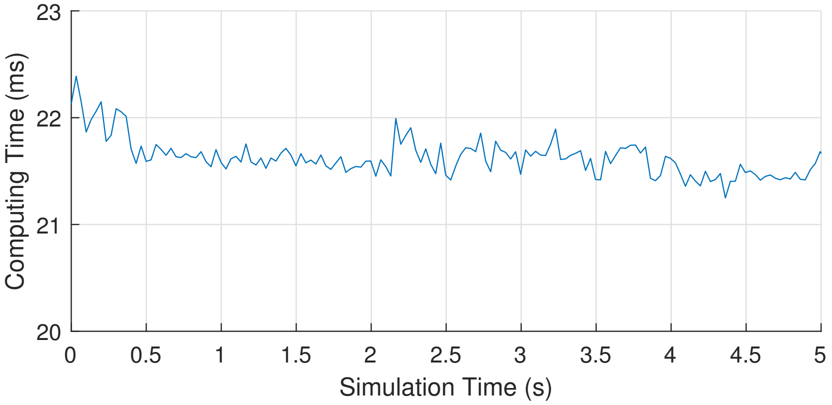
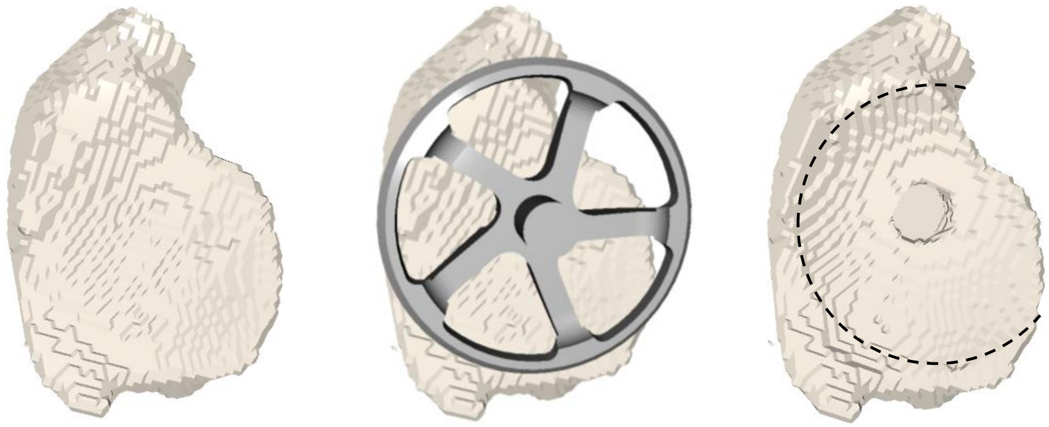


Figure 5.12: Computing time measured for graphic rendering frames



(a) Sample glenoid bone (b) Glenoid reaming (c) Resurfaced bone

Figure 5.13: Graphic representation of glenoid reaming

5.5 Summary

This chapter presented an empirical approach to model mechanics of glenoid reaming. It established linear equations to predict reamer displacement as a function of bone density and feed-force in cortical and cancellous regions, which can be further utilized to effectively compute haptic feedback during surgery simulation. The chapter also analyzed vibrations in glenoid reaming and presented a method to replicate vibrations during haptic simulations. The developed methods can appropriately demonstrate the differences in characteristics of reaming cortical and cancellous bone and are suitable for patient-specific simulations. This chapter also explained how to consolidate all simulator components that were developed throughout this thesis in one unified framework and how integrate this framework with a haptic device to perform simulations. The implementation results verified that the developed methods are capable to reliably maintain target frame rates for realistic haptic and graphic simulations.

Chapter 6

Thesis Closure

This chapter reviews the thesis objectives, summarizes the work that has been undertaken to address these objectives, discusses the strengths and limitations of this research, and outlines current and future research projects that emanate from this research.

6.1 Summary

While TSA is a well-established surgery in the upper torso, many surgeons find this surgery technically challenging primarily due to low practice volume. One of the most challenging tasks in TSA is glenoid reaming which plays a determining role on the long-term outcome of this surgery. The primary objective of this thesis was to develop a VR surgery simulator for this task in order to provide surgeons with a high-volume of practice for this procedure without the need for patients or cadavers. The development of VR surgery simulator for glenoid reaming was, however, a complicated problem. As revealed by the literature review in Section 1.6, there were gaps in the knowledge of the mechanics of glenoid reaming as well as paucity of effective computational tools to effectively replicate this unique bone machining process. As a result, various steps had to be undertaken to advance the methods in VR surgery simulation and also gain an understanding of the mechanical aspects of glenoid reaming to realistically replicate this procedure. Toward this end, throughout Chapters 2 – 5, several concrete contributions are introduced for different problems regarding processing CT data, voxelization, collision detection, modeling of glenoid reaming and simulator software development.

In particular, Chapter 2 developed an algorithm to efficiently process CT data and describe the geometry and material properties of bone for either simulation or FEA purposes. It showed that the use of grid-based approaches and hashing techniques leads to a number of desirable characteristics such as memory efficiency, robust numerical computations, implicit definitions of CT voxels and facile voxel upsampling. As a result, it is possible to generate Cartesian μFEMs with more than 36 M voxels within less than a minute using a commodity hardware. Importantly, the algorithm demonstrated linear time complexity which ensures reasonable increments in the computing time when dealing with larger data-sets.

Next, Chapter 3 developed a fast and flexible voxelization toolkit to convert triangle mesh models into their voxelized counterparts. The developed toolkit features parallel execution of an exact triangle-box overlap test which allows to rapidly convert any soup of triangle to an accurate voxelized model, with either 26-separability or 6-separability topological properties. As the toolkit is written in OpenCL, it is compatible with a wide-range of hardware from all major vendors, making it possible to promptly yet reliably construct voxelized - and ultimately, point cloud - representations of arbitrary objects that are modeled modeled by a triangle mesh.

Another important contribution of this thesis is a new VPS collision detection algorithm that was developed in Chapter 4. This algorithm leverages several simple yet effective data structures and methods, primarily in the form of GPU acceleration, VOI calculation, atomic operations and grid computations to quickly identify tool-bone intersection and modify bone geometry. The algorithm was proved to be superior, in terms of running time as well as memory efficiency, compared to the latest variants of the VPS method. Interestingly, it's running time exhibits remarkably low sensitivity to the number of bone voxels which enables fast collision detection in the presence of massively large data-sets. Such feature is not offered by any of other variants of the VPS method, making our algorithm an exceptional candidate for high-fidelity simulation of glenoid reaming and further, in a broad range of VR applications that involve collision detection of complex geometries.

It is worth mentioning that the geometry altering mechanism in our collision detection algorithm was shown to integrate effectively with the marching cubes algorithm to provide visual feedback of bone resurfacing. In particular, the implementation results in Section 5.4.2 showed that it is possible to reliably maintain 30 FPS frame rate and replicate material removal on micro-level voxel data without a need for a top-of-the-line GPU. The results can be further improved using faster implementations of marching cubes or other rendering algorithms. Beyond the context of surgery simulation, this lends well to the computationally demanding problem of computing Boolean operation between triangular meshes that is often required in many computer-aided design and modeling applications.

In addition, this thesis presented one of the first attempts in modeling the mechanics of glenoid reaming and its haptic simulation. In this regard, Chapter 5 utilized experimental data obtained by robot-driven glenoid reaming and derived simple equations that predicts reamer displacement as a function of bone density and feed-force. This chapter also studied vibrations in glenoid reaming and inferred that there is no dominant vibration frequency within the range of frequencies that is perceptible to humans. Interestingly, the findings of this chapter allowed to successfully distinguish the characteristics of reaming cortical bone from cancellous bone. Specifically, it became clear that reaming cancellous bone is attributed by 6.2 ± 0.08 times faster reamer feedrate and 1.52 ± 0.75 times higher vibration energy, compared to reaming cortical bone with the same feed-force.

Chapter 5 also elucidated how to implement the above results to replicate glenoid reaming using a haptic device. It established that the derived thrust-feedrate equations can be directly implemented in an admittance display haptic device and discussed why this type of haptic devices are more suitable to simulate glenoid reaming. It also outlined how to reinterpret the equations for implementation in an impedance display haptic device, primarily by inferring damping coefficients for cortical and cancellous bone and using them to predict thrust amplitude as a function of reamer velocity. Besides, the chapter showed how to properly prepare vibrations and add them to the haptic feedback which plays an important role in increasing simulation realism. The derived haptic rendering method was shown to integrate effectively with the rest of computational tools developed for the simulator. As a result, the simulator was able to reliably maintain the target 1000 Hz refresh rate for haptic rendering throughout the simulation tests.

6.2 Strengths and Limitations

Most of the computational problems that were addressed in this thesis, such as CT data processing, micro-level finite element modeling, voxelization, collision detection or haptic rendering, have been widely studied previously in various engineering problems. These problems are often characterized as computationally demanding due to presence of large size of processing data, severe timing constraints and the need for high computational power. Taking this into account, this thesis placed a particular emphasis on computational efficiency of methods throughout the development of simulator components. As a result, the algorithms and computer programs that were developed in this thesis offer high-level of performance compared to the existing methods.

Another strength of the work presented in this thesis is that it widely employed OpenCL-based GPU computing as the basis of many computational tasks. Presently, GPUs offer higher number-crunching performance and power efficiency compared to CPUs and are more cost-effective options for high-performance computing. In addition, there is no programming language that can target as wide range of devices as OpenCL [124]. Therefore, the developments in this thesis conform with the state-of-the-art trends in super-computing and the resulting computational tools deliver an exceptional level of performance and versatility. As an evidence, the running times reported for various computing tasks throughout this thesis were impressive while they were obtained using a mid-range gaming desktop computer.

Although there is no generic rule to set the resolution for Voxmap and PointShell, it is widely accepted that the higher the sampling resolutions, the better the simulation experience. While this matter has been overlooked by the majority of previous works on orthopaedic simulators, the focus on computational efficiency of the algorithms have enabled this thesis to extend the maximum attainable resolutions and potentially open the way for improving the level of realism in virtually every hard-tissue interaction simulator. In this regard, the developed CT processing algorithm allows to conveniently upsample clinical CT voxels to desired resolutions which serves as a step forward toward high-fidelity patient-specific simulations.

It is also important to note that this thesis utilized force-controlled glenoid reaming experiments to model the mechanics of the operation in contrast with the previous studies on different bone machining operations that have unanimously employed position-controlled experiments. While it is true that mechanics of bone and metal machining are basically similar, one key difference between them is that bone machining is operated by a surgeon who relies on force feedback whereas metal machining is often performed by a machine that uses a predetermined tool path without considering the cutting forces. Therefore, it appears that using a robot that is programmed in force-control mode is a more effective way to mimic the real bone machining operation and is more suitable for modeling purposes, rather than a CNC machine. Therefore, it is believed that this new experimentation approach can lead to better understanding of various bone machining operations in near future.

Despite the above strengths, the work presented in this thesis contains a number shortcomings, especially regarding modeling of glenoid reaming. In particular, the thrust-feedrate equations derived in Chapter 5 were based on experimental data obtained with identical feed-force trajectories. Linear scaling was suggested to predict the feedrate in case a different feed-force is applied. However, the linear scaling is inferred merely through observation of the trend in instantaneous thrust-displacements curves and is not strongly supported by experiments. In addition, the instantaneous reamer-bone intersection volume was not measured during the experiments and did not play a role in model derivation. Moreover, the vibration analysis did not determine the effect of tool rotation and tool-bone intersection distinctly which could be especially useful in simulation of the time of contact between reamer and bone. It is important to note that the experiments were performed using a serial robot manipulator with high compliance. This resulted in undesirable reamer retrogression throughout the experiments, especially during reaming cortical bone. Besides, this thesis did not discuss the torques which could be a valuable feedback mode to increase the level of realism in simulation.

It should also be pointed out that the methods developed in this thesis depend heavily on voxels. One of the disadvantages of voxels is the jagged representation of an object geometry which leads to loss of accuracy in different computational tasks. In addition, the VPS collision detection method suffers from the *tunneling effect* which refers to missing thin structures when the tool movement is larger than the voxel size. This thesis also did not present any results using a haptic device with a range of force and stiffness similar to the ones observed in glenoid reaming.

6.3 Recommendations for Future Research

The results presented in this thesis open new doors to future research on surgery simulation and biomechanics research in various directions. In the context of modeling of bone machining operations, this thesis suggests performing calibration experiments using robots with low compliance and high stiffness. As bone machining operations are performed manually by surgeons, the use of robots for experiments appear to be a more effective approach to replicate these operations, compared to CNC machines. It is worth mentioning that the robot should be programmed in force-control mode and different levels of force should be applied during the experiments. Besides, acquiring μCT image of specimens prior to experiments can be useful for accurate measurement of bone material properties and also measurement of instantaneous tool-bone intersection volume. It should be pointed out that the latter also requires precise measurement of tool and bone position and orientation throughout the experiments.

Concerning with analytic modeling of glenoid reaming, this thesis suggests considering theoretical approaches in material scraping as the reamer spindle speed is significantly low and material removal can be the result of rubbing bone surface instead of shearing. In addition, a valuable extension of the current work is to model scapula mobility during glenoid reaming and replicate its effects during haptic simulations. For haptic simulations of this operation, the use of an admittance display device is recommended even though these devices are generally more expensive than impedance displays. However, a future research to construct a custom and affordable admittance display that is capable to handle the range of force and stiffness observed in glenoid reaming will be extremely advantageous to replicate this operation realistically. Such haptic device can generally beneficial for simulation of any stiff objects, as well.

In terms of the computational tools for the simulator software, the collision detection algorithm can be improved in different ways. For example, although the proposed method for VOI calculation is fast, it is conservative in a sense that it overestimates the intersection volume and results in culling only a portion of unnecessary threads. Therefore, a more accurate VOI calculation method can further reduce the algorithm running time. In addition, the algorithm does not update bone OBB as bone undergoes material removal. A method to rapidly reconstruct bone OBB can lead to a tighter fit and can subsequently improve the accuracy of broad-phase computations. Furthermore, the algorithm can be integrated with continuous collision detection techniques such as internal bisection [193] in order to determine the exact time of tool-bone contact and eliminate the unwanted tunneling effect.

It is difficult to expect that a VR simulator - irrespective of the level of fidelity it provides - become employed in a surgeon training program without validation studies. In fact, as discussed in Chapter 1, one fundamental barrier for the use of simulation-based training in orthopaedics is the lack of validated simulators [10]. Therefore, internal and face validity of the developments of this thesis can be the focus an important future work. In addition, the use of feedback from surgeons during using the simulator can be valuable to modify the simulator design and enhance its practicality, as previously performed in [20], [29].

Regarding finite element modeling, the algorithm presented in Chapter 2 can be enhanced in several ways. Since the algorithm constructs hexahedral elements per CT voxels independently, it can benefit from out-of-core methods for more efficient memory management. This can be effectively combined by GPU parallelization to achieve higher computing performance. Moreover, although automatic segmentation of CT images continues to remain a formidable challenge, the algorithm can be integrated with image processing techniques to reduce manual steps in converting CT data to FEMs. It is also noteworthy that the algorithm generates equally shaped and sized elements which can lead to loss of accuracy in geometrical modeling. Therefore, another possible extension of the current work can be focused on the generation of adaptively-sized hexahedral elements that are possibly dimensioned through curvature tracking algorithms. However, this task is not trivial due to the occurrence of *hanging nodes* in the generated mesh [201]. Evidently, all these future extensions would lead to the generation of true hexahedral meshes as opposed to their present Cartesian aspect.

6.4 Significance

This thesis presented one of the first attempts to construct a VR surgery simulator for glenoid reaming to be primarily used as a convenient tool for practicing one of the most challenging tasks of TSA. While TSA continues to remain as a challenging orthopaedic surgery, the need for this surgery is continuously growing [50]. Given the overall aging of the American population, the demand for TSA is predicted to increase by 333.3% in patients younger than 55 years and by 755.4% in older patients, from 2011 to 2030 [202]. Even though such projections have not been reported for Canadians, it can be safely assumed that similar trends exist. It is presently believed that using a VR surgery simulator can effectively increase surgeons' volume of practice for this surgery and will ultimately translate into reduced procedure time and revision rates of TSA. The work presented in this thesis also contributes advances in methods and knowledge in the general area of surgery simulation and can be regarded as a step forward toward wider implementation of this technology in surgeon training programs.

Beyond the context of surgery simulation, the algorithms developed in this thesis are advantageous to various engineering problems. With the continuous grow in computing power, micro-level structural analysis of bone is becoming more popular. The Cartesian mesh generator presented in Chapter 2 serves as a fast and robust computational tool for construction of hexahedral μFEMs and has been successfully applied in several micro-level finite element studies of cancellous bone thus far [203]–[205]. This algorithm is also presented in [206]–[208] and is made available online for public use [209].

The use of voxels in geometrical modeling is currently receiving a surge of interest. Over the past few years, voxels have been utilized in many problems including skeleton extraction from mesh [210], composite model representations of functionally gradient materials [211], 3D printing [212], generation of porous surfaces [213], mesh repair [214], thickness analysis [215], etc. The voxelization toolkit presented in Chapter 3 is one of the fastest and most versatile tools that is currently available and can be effectively utilized to construct voxelized models. This toolkit is also presented in [216], [217] and is made available online for public use [218].

Appendix A

Oblique Cutting Model for Glenoid Reaming

The oblique cutting model introduced in Section 1.6.1 serves as the basis for many mechanistic models that have been developed for various bone machining operations. This approach uses infinitesimal elements on tool cutting lips and computes elementary cutting forces which are then aggregated to compute thrust and torque. The resulting model is constituted by a set of equations that are specific to geometry of the tool. Since these equations have not been derived for glenoid reaming in the present literature, this appendix focuses on geometrical treatments and mathematical derivations required for mechanistic modeling of this bone machining operation. While mechanistic modeling is not utilized for modeling of glenoid reaming in this thesis, the developments of this appendix can be used for future research toward theoretically more profound modeling of this operation.

A.1 Reamer Geometry

The spherical glenoid reamer considered in this thesis contains five cutting lips. For an arbitrary point A on a cutting lip, the radius r is defined as the distance from A to reamer axis, as shown in Fig. A.1. For this reamer, the normal rake angle γ_n that was introduced in Fig. 1.4 is zero

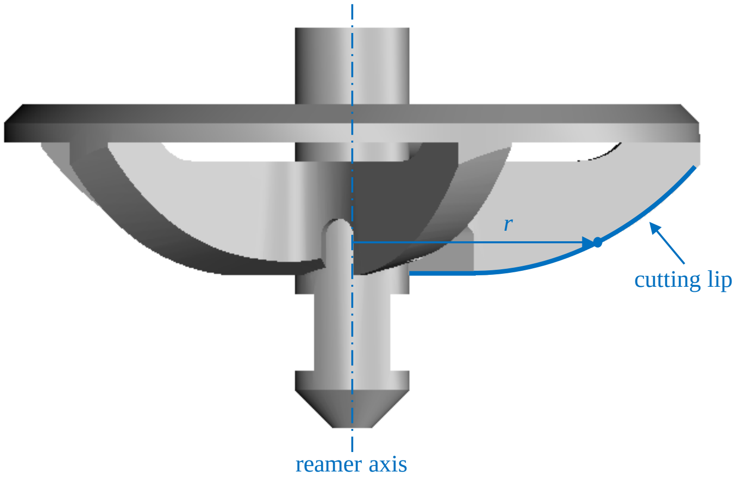


Figure A.1: A sample point on reamer cutting lip and its radial distance

for all points on the cutting lips. However, the clearance angle, which is defined between the relief face and workpeice, is varying as a function of r. The clearance angle has an insignificant effect on cutting forces and can be neglected in subsequent developments [60], [86]. There are three other angles that play important roles in mechanics of oblique cutting and are described in the remainder of this section. The subsequent developments are only presented for the curved section of the cutting lips and can be easily extended to the flat sections, as well.

A.1.1 Web Angle

Web angle β is defined as the angle between radial direction and cutting lip, measured in xy plane, as shown in Fig. A.2. Considering the triangle ABC and applying the law of sines, it follows that

$$\frac{\sin(\pi-\beta)}{a} = \frac{\sin\alpha}{r}. \tag{A.1}$$

Therefore, the web angle for this reamer can be expressed as

$$\beta = \sin^{-1}\left(\frac{a}{r}\sin\alpha\right). \tag{A.2}$$

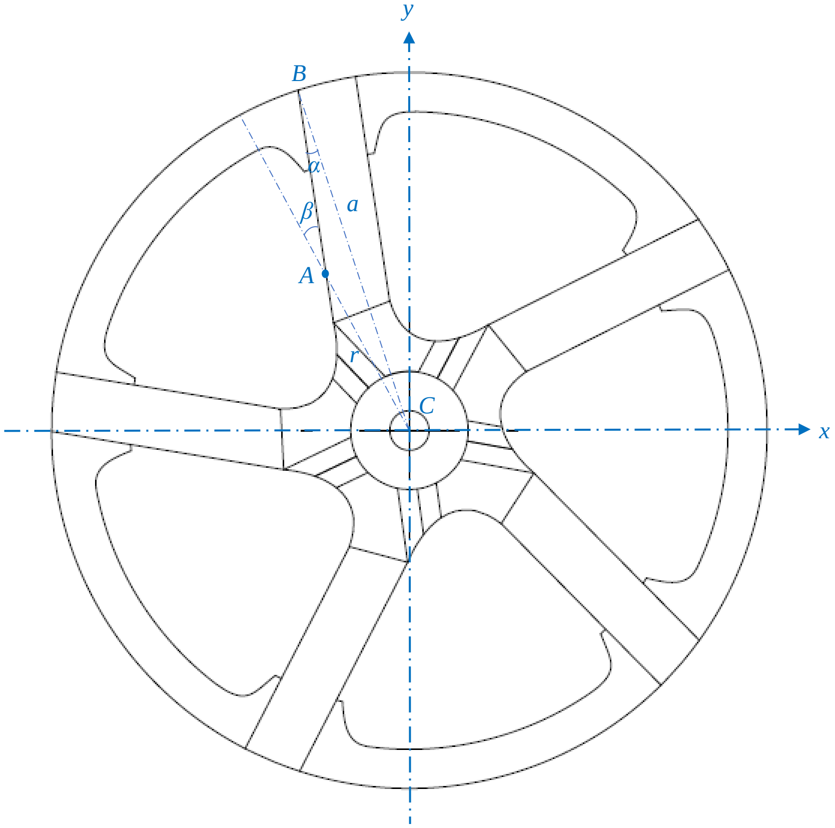


Figure A.2: Definition of web angle for reamer

It is noteworthy that α is specific to the reamer geometry and is equal to 9.46° for the reamer considered in this thesis.

A.1.2 Point Angle

The point angle ε for an arbitrary point A on a reamer cutting lip is defined as the tangent to the point, on yz plane. As it can be implied by Fig. A.3, this angle varies along the cutting lip and can be calculated as a function of r. Each reamer cutting lip can be regarded as an arc of an ellipse whose major and minor axes lengths are $2a$ and $2b$, respectively. Therefore, the following expression defines a cutting lip on yz plane

$$\frac{z^2}{b^2} + \frac{y^2}{a^2} = 1, \quad 0 \leq z \leq b, \quad -a \leq y \leq a. \tag{A.3}$$

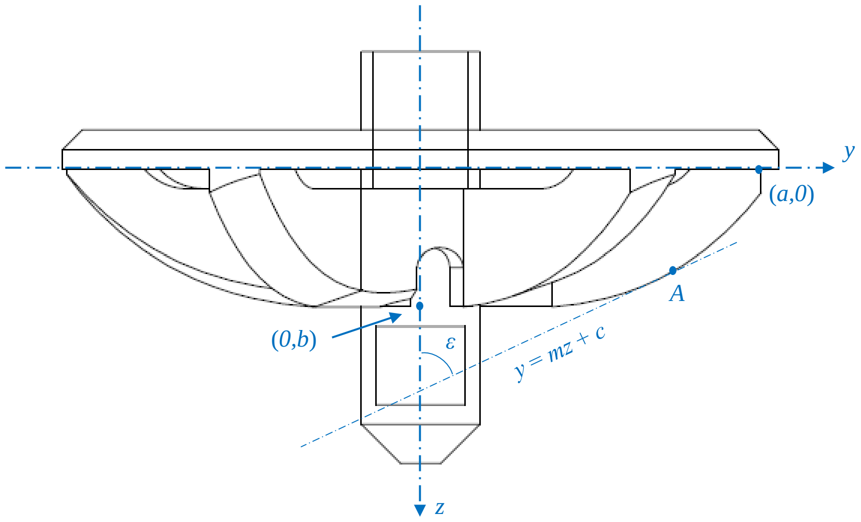


Figure A.3: Definition of point angle for reamer

It is worth mentioning that for the reamer under consideration, $a = 18.25$ mm and $b = 9$mm. Moving forward, the point A can be identified by $\left(-\frac{b}{a}\sqrt{a^2 - r^2}, r\right)$ on the yz plane. In addition, the tangent to the point A takes the form $y = mz + c$ which must fulfill

$$\frac{z^2}{b^2} + \frac{(mz + c)^2}{a^2} = 1, \quad -b \leq z \leq 0, \quad a \leq y \leq 0, \tag{A.4}$$

whose roots are given by

$$z = \frac{-b^2 mc \pm \sqrt{b^2 m^2 + a^2 - c^2}}{a^2 m^2 + b^2}. \tag{A.5}$$

According to the tangency condition, the expression $b^2 m^2 + a^2 - c^2 = 0$ must hold true. As a result, the intersection point can be found as $\left(-\frac{b^2 m}{c}, \frac{a^2}{c}\right)$, which yields

$$\frac{y_A}{z_A} = -\frac{a^2}{b^2 m} \rightarrow m = -\frac{a^2 z_A}{b^2 y_A}. \tag{A.6}$$

From the ellipse equation (A.3), it can be found $z_A = -\frac{b}{a}\sqrt{a^2 - r^2}$ and $y_A = r$. Therefore, the slope of the tangent at A is given by $m = \frac{a}{r}\sqrt{a^2 - r^2}$, which ultimately concludes the following expression for the point angle

$$\varepsilon = \tan^{-1}\left(\frac{a}{r}\sqrt{a^2 - r^2}\right). \tag{A.7}$$

A.1.3 Inclination Angle

Since the cutting lips do not intersect the reamer axis, there exists a non-zero inclination angle on the rake face. Using ε and β, this angle can be defined as follows

$$\lambda = \sin^{-1}\left(\sin\beta\cos\mu\sin\varepsilon + \sin\mu\cos\varepsilon\right), \tag{A.8}$$

where μ is called cutting angle and will be defined later. The derivation of the above expression can be found in [68], [219].

A.2 Coordinates Transformations in Oblique Cutting

In oblique cutting, there exists an acute angle λ between cutting edge and normal to tool velocity vector V in a plane containing both V and cutting edge. This angle is the angle between the cutting edge and x' in Fig. A.4. To study forces at an arbitrary point A on cutting edge, a Cartesian coordinate system $x'y'z'$ is considered at this point. In this coordinate system, y' is aligned with V and x' is perpendicular to y' in a plane defined by y' and cutting edge i.e. the plane on which λ is defined. Finally, z' is made perpendicular to $x'y'$ plane. According to [61], when the rake face comes into contact with the workpiece, two elementary forces, one tangential to the rake face df_t and one normal df_n to the face, are generated as a result of shearing strain. These elementary forces can be described in $x'y'z'$ coordinate system as follows

$$\begin{pmatrix} df_{x'} \\ df_{y'} \\ df_{z'} \end{pmatrix} = \begin{pmatrix} -\cos\gamma_n\sin\lambda & \sin\eta_c\cos\lambda - \cos\eta_c\sin\gamma_n\sin\lambda \\ -\cos\gamma_n\cos\lambda & \cos\eta_c\sin\gamma_n\cos i + \sin\eta_c\sin\lambda \\ -\sin\gamma_n & \cos\eta_c\cos\gamma_n \end{pmatrix} \begin{pmatrix} df_n \\ df_t \end{pmatrix}, \tag{A.9}$$

where η_c denotes the chip flow direction which is approximately equal to λ, according to the chip flow law of Stabler [220]. The forces obtained in $x'y'z'$ coordinates must be transformed to the machining coordinates system and aggregated to compute global thrust and torque.

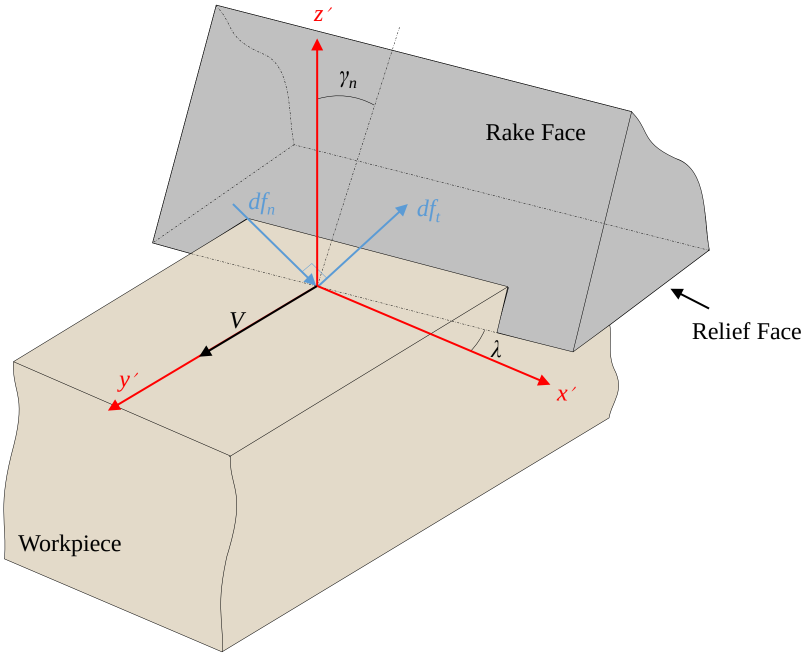


Figure A.4: Coordinates system defined for elementary forces in oblique cutting

The machining coordinates system xyz is shown in Figs. A.2 and A.3 where x is along radial direction and z is aligned with tool axis and global thrust. The axis y is made perpendicular to the xz plane. When there is no feed-force applied, V is aligned with y which results in coincidence of y' and y. However, in presence of a feed-force, V is inclined downward and can be decomposed into two components: V_n which is aligned with the z-axis and V_t which corresponds to the tangential velocity and is aligned with y.

The angle between V and V_t or y and y' measured in a plane perpendicular to the radial direction and passing through the point A is called cutting angle μ which can be calculated as follows

$$\mu = \tan^{-1}\left(\frac{f}{2\pi r}\right), \tag{A.10}$$

where f denotes feed rate. One of the advantages of introducing this angle is facilitating finding the rotation matrix between xyz and $x'y'z'$ coordinate systems which is denoted by $\mathbf{R}$. Another angle that helps finding such transformation matrix is the second Euler angle τ that is used to align x' with y'. This angle can be calculated using the following expression

$$\tau = \cos^{-1}\left(\frac{\sin\varepsilon\cos\beta}{\cos\lambda}\right). \tag{A.11}$$

Using these angles, **R** becomes

$$\mathbf{R} = \begin{pmatrix} \cos\tau & 0 & -\sin\tau \\ \sin\tau\sin\mu & \cos\mu & \cos\tau\sin\mu \\ \sin\tau\cos\mu & -\sin\mu & \cos\tau\cos\mu \end{pmatrix}, \tag{A.12}$$

which can utilized to transform the elementary forces from oblique cutting coordinates to the machining coordinates system.

A.3 Thrust and Torque in Glenoid Reaming

Once the elementary forces are transformed to the xyz coordinates systems, it is straightforward to compute global thrust T and torque M as follows

$$T = \sum_{k=1}^{n} N df_{z,k}, \tag{A.13}$$

$$M = \sum_{k=1}^{n} N dm_k, \tag{A.14}$$

where $dm_k = r_k df_{y,k}$ represents the torque generated by an element, $N = 5$ indicates the number of cutting lips, n is the number of elements imposed on each cutting lip, and the subscript k denotes one element. Considering rotation matrix described by (A.12), the following expression can be written for each element

$$\begin{pmatrix} df_Z \\ dm \end{pmatrix} = \begin{pmatrix} \sin\tau\cos\mu & -\sin\mu & \cos\tau\cos\mu \\ 0 & r & 0 \end{pmatrix} \begin{pmatrix} df_{x'} \\ df_{y'} \\ df_{z'} \end{pmatrix}. \tag{A.15}$$

Next, using (A.9) and substituting $\gamma_n = 0$, $\eta_c = \lambda$, it follows that

$$\begin{aligned} df_z = {} & (-\sin\tau\cos\mu\sin\lambda + \sin\mu\cos\lambda)df_n \\ & +(\sin\tau\cos\mu\sin\lambda\cos\lambda - \sin\mu\sin^2\lambda + \cos\tau\cos\mu\cos\lambda)df_t, \end{aligned} \tag{A.16}$$

and

$$dm = (-r\cos\lambda)df_n + (r\sin^2\lambda)df_t. \tag{A.17}$$

The above expressions combined with (1.1) are sufficient to determine thrust and torque in glenoid reaming.

Bibliography

[1] E. A. Halm, C. Lee, and M. R. Chassin, “Is volume related to outcome in health care? A systematic review and methodologic critique of the literature,” *Annals of internal medicine*, vol. 137, no. 6, pp. 511–520, 2002.

[2] A. B. Flood, W. R. Scott, and W. Ewy, “Does practice make perfect? Part I: The relation between hospital volume and outcomes for selected diagnostic categories,” *Medical care*, pp. 98–114, 1984.

[3] ——, “Does practice make perfect? Part II: The relation between volume and and outcomes and other hospital characteristics,” *Medical care*, pp. 115–125, 1984.

[4] H. S. Luft, “The relation between surgical volume and mortality: an exploration of causal factors and alternative models,” *Medical care*, pp. 940–959, 1980.

[5] H. S. Luft, J. P. Bunker, and A. C. Enthoven, “Should operations be regionalized? The empirical relation between surgical volume and mortality,” *New England Journal of Medicine*, vol. 301, no. 25, pp. 1364–1369, 1979.

[6] H. S. Luft, S. S. Hunt, and S. C. Maerki, “The volume-outcome relationship: practice-makes-perfect or selective-referral patterns?” *Health services research*, vol. 22, no. 2, p. 157, 1987.

[7] H. D. Taylor, D. A. Dennis, and H. S. Crane, “Relationship between mortality rates and hospital patient volume for Medicare patients undergoing major orthopaedic surgery of the hip, knee, spine, and femur,” *The Journal of arthroplasty*, vol. 12, no. 3, pp. 235–242, 1997.

[8] J. D. Birkmeyer, A. E. Siewers, E. V. Finlayson, T. A. Stukel, F. L. Lucas, I. Batista, H. G. Welch, and D. E. Wennberg, “Hospital volume and surgical mortality in the United States,” *New England Journal of Medicine*, vol. 346, no. 15, pp. 1128–1137, 2002.

[9] *Deaths and Mortality*, `https://www.cdc.gov/nchs/fastats/deaths.htm`, Accessed: 2018-11-07.

[10] G. W. Thomas, B. D. Johns, J. L. Marsh, and D. D. Anderson, “A review of the role of simulation in developing and assessing orthopaedic surgical skills,” *The Iowa Orthopaedic Journal*, vol. 34, p. 181, 2014.

[11] R. J. Duvivier, J. van Dalen, A. M. Muijtjens, V. R. Moulaert, C. P. van der Vleuten, and A. J. Scherpbier, “The role of deliberate practice in the acquisition of clinical skills,” *BMC Medical Education*, vol. 11, no. 1, p. 101, 2011.

[12] K. A. Ericsson, "Deliberate practice and the acquisition and maintenance of expert performance in medicine and related domains," *Academic Medicine*, vol. 79, no. 10, S70–S81, 2004.

[13] ——, "Deliberate practice and acquisition of expert performance: a general overview," *Academic Emergency Medicine*, vol. 15, no. 11, pp. 988–994, 2008.

[14] W. C. McGaghie, S. B. Issenberg, M. E. R. Cohen, J. H. Barsuk, and D. B. Wayne, "Does simulation-based medical education with deliberate practice yield better results than traditional clinical education? A meta-analytic comparative review of the evidence," *Academic Medicine*, vol. 86, no. 6, p. 706, 2011.

[15] U. Kühnapfel, H. K. Cakmak, and H. Maaß, "Endoscopic surgery training using virtual reality and deformable tissue simulation," *Computers & Graphics*, vol. 24, no. 5, pp. 671–682, 2000.

[16] F. Neyret, R. Heiss, and F. Sénégas, "Realistic rendering of an organ surface in real-time for laparoscopic surgery simulation," *The Visual Computer*, vol. 18, no. 3, pp. 135–149, 2002.

[17] T.-Y. Lee, C.-H. Lin, and H.-Y. Lin, "Computer-aided prototype system for nose surgery," *IEEE Transactions on Information Technology in Biomedicine*, vol. 5, no. 4, pp. 271–278, 2001.

[18] N. Kusumoto, T. Sohmura, S. Yamada, K. Wakabayashi, T. Nakamura, and H. Yatani, "Application of virtual reality force feedback haptic device for oral implant surgery," *Clinical Oral Implants Research*, vol. 17, no. 6, pp. 708–713, 2006.

[19] J. Wu, D. Wang, C. C. Wang, and Y. Zhang, "Toward stable and realistic haptic interaction for tooth preparation simulation," *Journal of Computing and Information Science in Engineering*, vol. 10, no. 2, p. 021 007, 2010.

[20] A. K. Ho, H. Alsaffar, P. C. Doyle, H. M. Ladak, and S. K. Agrawal, "Virtual reality myringotomy simulation with real-time deformation: Development and validity testing," *The Laryngoscope*, vol. 122, no. 8, pp. 1844–1851, 2012.

[21] Q. Wang, H. Chen, W. Wu, H.-Y. Jin, and P.-A. Heng, "Real-time mandibular angle reduction surgical simulation with haptic rendering," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 6, pp. 1105–1114, 2012.

[22] C. K. Lam, K. Sundaraj, and M. N. Sulaiman, "Computer-based virtual reality simulator for phacoemulsification cataract surgery training," *Virtual Reality*, vol. 18, no. 4, pp. 281–293, 2014.

[23] *ABS to Require ACLS, ATLS and FLS for General Surgery Certification*, `https://www.cdc.gov/nchs/fastats/deaths.htm`, Accessed: 2018-11-07.

[24] J. D. Mabrey, K. D. Reinig, and W. D. Cannon, "Virtual reality in orthopaedics: is it a reality?" *Clinical Orthopaedics and Related Research®*, vol. 468, no. 10, pp. 2586–2591, 2010.

[25] J. A. S. Shantz, J. R. Leiter, T. Gottschalk, and P. B. MacDonald, "The internal validity of arthroscopic simulators and their effectiveness in arthroscopic education," *Knee Surgery, Sports Traumatology, Arthroscopy*, vol. 22, no. 1, pp. 33–40, 2014.

[26] J. D. Mabrey, S. D. Gillogly, J. R. Kasser, H. J. Sweeney, B. Zarins, H. Mevis, W. E. Garrett Jr, R. Poss, and W. D. Cannon, "Virtual reality simulation of arthroscopy of the knee," *Arthroscopy*, vol. 18, no. 6, pp. 1–7, 2002.

[27] G. Megali, O. Tonet, M. Mazzoni, P. Dario, A. Vascellari, and M. Marcacci, "A new tool for surgical training in knee arthroscopy," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2002, pp. 170–177.

[28] P.-A. Heng, C.-Y. Cheng, T.-T. Wong, W. Wu, Y. Xu, Y. Xie, Y.-P. Chui, K.-M. Chan, and K.-S. Leung, "Application to anatomic visualization and orthopaedics training," *Clinical Orthopaedics and Related Research®*, vol. 442, pp. 5–12, 2006.

[29] M. Vankipuram, K. Kahol, A. McLaren, and S. Panchanathan, "A virtual reality simulator for orthopedic basic skills: a design and validation study," *Journal of Biomedical Informatics*, vol. 43, no. 5, pp. 661–668, 2010.

[30] M.-D. Tsai, M.-S. Hsieh, and C.-H. Tsai, "Bone drilling haptic interaction for orthopedic surgical simulator," *Computers in Biology and Medicine*, vol. 37, no. 12, pp. 1709–1718, 2007.

[31] M.-S. Hsieh, M.-D. Tsai, and Y.-D. Yeh, "An amputation simulator with bone sawing haptic interaction," *Biomedical Engineering: Applications, Basis and Communications*, vol. 18, no. 05, pp. 229–236, 2006.

[32] M.-D. Tsai and M.-S. Hsieh, "Accurate visual and haptic burring surgery simulation based on a volumetric model," *Journal of X-ray Science and Technology*, vol. 18, no. 1, pp. 69–85, 2010.

[33] M. Arbabtafti, M. Moghaddam, A. Nahvi, M. Mahvash, B. Richardson, and B. Shirinzadeh, "Physics-based haptic simulation of bone machining," *IEEE Transactions on Haptics*, vol. 4, no. 1, pp. 39–50, 2011.

[34] A. Danda, M. A. Kuttolamadom, and B. L. Tai, "A mechanistic force model for simulating haptics of hand-held bone burring operations," *Medical Engineering & Physics*, vol. 49, pp. 7–13, 2017.

[35] A. Karelse, S. Leuridan, A. Van Tongel, I. M. Piepers, P. Debeer, and L. F. De Wilde, "A glenoid reaming study: how accurate are current reaming techniques?" *Journal of Shoulder and Elbow Surgery*, vol. 23, no. 8, pp. 1120–1127, 2014.

[36] M. E. Torchia, R. H. Cofield, and C. R. Settergren, "Total shoulder arthroplasty with the Neer prosthesis: long-term results," *Journal of Shoulder and Elbow Surgery*, vol. 6, no. 6, pp. 495–505, 1997.

[37] M. A. Wirth and C. A. Rockwood Jr, "Complications of total shoulder-replacement arthroplasty," *The Journal of Bone and Joint Surgery*, vol. 78, no. 4, pp. 603–616, 1996.

[38] P. Y. Chin, J. W. Sperling, R. H. Cofield, and C. Schleck, "Complications of total shoulder arthroplasty: are they fewer or different?" *Journal of Shoulder and Elbow Surgery*, vol. 15, no. 1, pp. 19–22, 2006.

[39] K. I. Bohsali, M. A. Wirth, and C. A. Rockwood Jr, "Complications of total shoulder arthroplasty," *The Journal of Bone and Joint Surgery*, vol. 88, no. 10, pp. 2279–2292, 2006.

[40] I. Szabo, G. Walch, *et al.*, "Factors affecting cemented glenoid component loosening in total shoulder arthroplasty," *International Journal of Shoulder Surgery*, vol. 1, no. 1, p. 23, 2007.

[41] F. A. Matsen III, J. Clinton, J. Lynch, A. Bertelsen, and M. L. Richardson, "Glenoid component failure in total shoulder arthroplasty," *The Journal of Bone and Joint Surgery*, vol. 90, no. 4, pp. 885–896, 2008.

[42] G. Walch, A. A. Young, P. Boileau, M. Loew, D. Gazielly, and D. Molé, "Patterns of loosening of polyethylene keeled glenoid components after shoulder arthroplasty for primary osteoarthritis: results of a multicenter study with more than five years of follow-up," *The Journal of Bone and Joint Surgery*, vol. 94, no. 2, pp. 145–150, 2012.

[43] B. J. Brewer, R. Wubben, and G. Carrera, "Excessive retroversion of the glenoid cavity. A cause of non-traumatic posterior instability of the shoulder.," *The Journal of Bone and Joint Surgery*, vol. 68, no. 5, pp. 724–731, 1986.

[44] D. Weishaupt, M. Zanetti, R. W. Nyffeler, C. Gerber, and J. Hodler, "Posterior glenoid rim deficiency in recurrent (atraumatic) posterior shoulder instability," *Skeletal Radiology*, vol. 29, no. 4, pp. 204–210, 2000.

[45] G. R. Williams Jr, K. L. Wong, M. D. Pepe, V. Tan, D. Silverberg, M. L. Ramsey, A. Karduna, and J. P. Iannotti, "The effect of articular malposition after total shoulder arthroplasty on glenohumeral translations, range of motion, and subacromial impingement," *Journal of Shoulder and Elbow Surgery*, vol. 10, no. 5, pp. 399–409, 2001.

[46] C. S. Radnay, K. J. Setter, L. Chambers, W. N. Levine, L. U. Bigliani, and C. S. Ahmad, "Total shoulder replacement compared with humeral head replacement for the treatment of primary glenohumeral osteoarthritis: a systematic review," *Journal of Shoulder and Elbow Surgery*, vol. 16, no. 4, pp. 396–402, 2007.

[47] G. S. Lewis and A. D. Armstrong, "Glenoid spherical orientation and version," *Journal of Shoulder and Elbow Surgery*, vol. 20, no. 1, pp. 3–11, 2011.

[48] D. Nguyen, L. M. Ferreira, J. R. Brownhill, G. J. King, D. S. Drosdowech, K. J. Faber, and J. A. Johnson, "Improved accuracy of computer assisted glenoid implantation in total shoulder arthroplasty: an in-vitro randomized controlled trial," *Journal of Shoulder and Elbow Surgery*, vol. 18, no. 6, pp. 907–914, 2009.

[49] *Shoulder Joint Replacement*, `https://orthoinfo.aaos.org/en/treatment/shoulder-joint-replacement`, Accessed: 2018-11-07.

[50] S. H. Kim, B. L. Wise, Y. Zhang, and R. M. Szabo, "Increasing incidence of shoulder arthroplasty in the United States," *The Journal of Bone and Joint Surgery*, vol. 93, no. 24, pp. 2249–2254, 2011.

[51] P. Brett, C. Fraser, M. Hennigan, M. Griffiths, and Y. Kamel, "Automatic surgical tools for penetrating flexible tissues," *IEEE Engineering in Medicine and Biology Magazine*, vol. 14, no. 3, pp. 264–270, 1995.

[52] F. Ong and K. Bouazza-Marouf, “The detection of drill bit break-through for the enhancement of safety in mechatronic assisted orthopaedic drilling,” *Mechatronics*, vol. 9, no. 6, pp. 565–588, 1999.

[53] E. Jantunen, “A summary of methods applied to tool condition monitoring in drilling,” *International Journal of Machine Tools and Manufacture*, vol. 42, no. 9, pp. 997–1010, 2002.

[54] G. Augustin, S. Davila, K. Mihoci, T. Udiljak, D. S. Vedrina, and A. Antabak, “Thermal osteonecrosis and bone drilling parameters revisited,” *Archives of Orthopaedic and Trauma Surgery*, vol. 128, no. 1, pp. 71–77, 2008.

[55] S. Kasiri, G. Reilly, and D. Taylor, “Wedge indentation fracture of cortical bone: experimental data and predictions,” *Journal of Biomechanical Engineering*, vol. 132, no. 8, p. 081 009, 2010.

[56] D. Kendoff, M. Citak, M. J. Gardner, T. Stübig, C. Krettek, and T. Hüfner, “Improved accuracy of navigated drilling using a drill alignment device,” *Journal of Orthopaedic Research*, vol. 25, no. 7, pp. 951–957, 2007.

[57] H. Thompson, “Effect of drilling into bone.,” *Journal of Oral Surgery*, vol. 16, no. 1, pp. 22–30, 1958.

[58] C. Jacob, J. Berry, M. Pope, and F. Hoaglund, “A study of the bone machining process-drilling,” *Journal of Biomechanics*, vol. 9, no. 5, pp. 345–349, 1976.

[59] M. B. Abouzgia and D. F. James, “Temperature rise during drilling through bone.,” *International Journal of Oral & Maxillofacial Implants*, vol. 12, no. 3, 1997.

[60] J. Lee, B. A. Gozen, and O. B. Ozdoganlar, “Modeling and experimentation of bone drilling forces,” *Journal of Biomechanics*, vol. 45, no. 6, pp. 1076–1083, 2012.

[61] M. E. Merchant, “Mechanics of the metal cutting process. I. Orthogonal cutting and a type 2 chip,” *Journal of Applied Physics*, vol. 16, no. 5, pp. 267–275, 1945.

[62] M.-B. Lazar and P. Xirouchakis, “Mechanical load distribution along the main cutting edges in drilling,” *Journal of Materials Processing Technology*, vol. 213, no. 2, pp. 245–260, 2013.

[63] T. Childs, K. Maekawa, T. Obikawa, and Y. Yamane, *Metal machining: theory and applications*. Butterworth-Heinemann, 2000.

[64] V. P. Astakhov, *Geometry of single-point turning tools and drills: fundamentals and practical applications*. Springer Science & Business Media, 2010.

[65] Y. Altintas, *Manufacturing automation: metal cutting mechanics, machine tool vibrations, and CNC design*. Cambridge University Press, 2012.

[66] M. Ucar and Y. Wang, “End-milling machinability of a carbon fiber reinforced laminated composite,” *Journal of Advanced Materials*, vol. 37, no. 4, pp. 46–52, 2005.

[67] R. Zitoune, F. Collombet, F. Lachaud, R. Piquet, and P. Pasquet, “Experiment-calculation comparison of the cutting conditions representative of the long fiber composite drilling phase,” *Composites Science and Technology*, vol. 65, no. 3-4, pp. 455–466, 2005.

[68] M.-B. Lazar, “Cutting force modelling for drilling of fiber-reinforced composites,” PhD thesis, École Polytechnique Fédérale de Lausanne, 2012.

[69] V. Chandrasekharan, S. Kapoor, and R. DeVor, “A mechanistic approach to predicting the cutting forces in drilling: with application to fiber-reinforced composite materials,” *Journal of Engineering for Industry*, vol. 117, no. 4, pp. 559–570, 1995.

[70] G. Caprino and L. Nele, “Cutting forces in orthogonal cutting of unidirectional GFRP composites,” *Journal of Engineering Materials and Technology*, vol. 118, no. 3, pp. 419–425, 1996.

[71] V. Chandrasekharan, S. Kapoor, and R. DeVor, “A mechanistic model to predict the cutting force system for arbitrary drill point geometry,” *Journal of Manufacturing Science and Engineering*, vol. 120, no. 3, pp. 563–570, 1998.

[72] A. Langella, L. Nele, and A. Maio, “A torque and thrust prediction model for drilling of composite materials,” *Composites Part A: Applied Science and Manufacturing*, vol. 36, no. 1, pp. 83–93, 2005.

[73] M.-B. Lazar and P. Xirouchakis, “Experimental analysis of drilling fiber reinforced composites,” *International Journal of Machine Tools and Manufacture*, vol. 51, no. 12, pp. 937–946, 2011.

[74] D. Liu, Y. Tang, and W. Cong, “A review of mechanical drilling for composite laminates,” *Composite Structures*, vol. 94, no. 4, pp. 1265–1279, 2012.

[75] M. Marco, M. Rodríguez-Millán, C. Santiuste, E. Giner, and M. H. Miguélez, “A review on recent advances in numerical modelling of bone cutting,” *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 44, pp. 179–201, 2015.

[76] L. Yanping, Y. Dedong, C. Xiaojun, W. Xudong, S. Guofang, and W. Chengtao, “Simulation and evaluation of a bone sawing procedure for orthognathic surgery based on an experimental force model,” *Journal of Biomechanical Engineering*, vol. 136, no. 3, p. 034 501, 2014.

[77] ——, “Development and validation of a surgical training simulator with haptic feedback for learning bone-sawing skill,” *Journal of Biomedical Informatics*, vol. 48, pp. 122–129, 2014.

[78] A. Pourkand, N. Zamani, and D. Grow, “Mechanical model of orthopaedic drilling for augmented-haptics-based training,” *Computers in Biology and Medicine*, 2017.

[79] T. MacAvelia, M. Salahi, M. Olsen, M. Crookshank, E. H. Schemitsch, A. Ghasempoor, F. Janabi-Sharifi, and R. Zdero, “Biomechanical measurements of surgical drilling force and torque in human versus artificial femurs,” *Journal of Biomechanical Engineering*, vol. 134, no. 12, p. 124 503, 2012.

[80] D. Morris, C. Sewell, F. Barbagli, K. Salisbury, N. H. Blevins, and S. Girod, “Visuohaptic simulation of bone surgery for training and evaluation,” *IEEE Computer Graphics and Applications*, vol. 26, no. 6, 2006.

[81] J. Wu, G. Yu, D. Wang, Y. Zhang, and C. C. Wang, “Voxel-based interactive haptic simulation of dental drilling,” in *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, 2009, pp. 39–48.

[82] F. Zheng, W. F. Lu, Y. San Wong, and K. W. C. Foong, “An analytical drilling force model and GPU-accelerated haptics-based simulation framework of the pilot drilling procedure for micro-implants surgery training,” *Computer Methods and Programs in Biomedicine*, vol. 108, no. 3, pp. 1170–1184, 2012.

[83] ——, “Graphic processing units (GPUs)-based haptic simulator for dental implant surgery,” *Journal of Computing and Information Science in Engineering*, vol. 13, no. 4, p. 041 005, 2013.

[84] M. Razavi, H. Talebi, M. Zareinejad, and M. Dehghan, “A GPU-implemented physics-based haptic simulator of tooth drilling,” *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 11, no. 4, pp. 476–485, 2015.

[85] S. Chan, P. Li, G. Locketz, K. Salisbury, and N. H. Blevins, “High-fidelity haptic and visual rendering for patient-specific simulation of temporal bone surgery,” *Computer Assisted Surgery*, vol. 21, no. 1, pp. 85–101, 2016.

[86] J. Sui, N. Sugita, K. Ishii, K. Harada, and M. Mitsuishi, “Mechanistic modeling of bone-drilling process with experimental validation,” *Journal of Materials Processing Technology*, vol. 214, no. 4, pp. 1018–1026, 2014.

[87] N. P. Dillon, L. B. Kratchman, M. S. Dietrich, R. F. Labadie, R. J. Webster III, and T. J. Withrow, “An experimental evaluation of the force requirements for robotic mastoidectomy,” *Otology & Neurotology*, vol. 34, no. 7, e93, 2013.

[88] K. Alam, A. Mitrofanov, and V. V. Silberschmidt, “Finite element analysis of forces of plane cutting of cortical bone,” *Computational Materials Science*, vol. 46, no. 3, pp. 738–743, 2009.

[89] T. Childs and D. Arola, “Machining of cortical bone: Simulations of chip formation mechanics using metal machining models,” *Machining Science and Technology*, vol. 15, no. 2, pp. 206–230, 2011.

[90] W. A. Lughmani, K. Bouazza-Marouf, and I. Ashcroft, “Finite element modeling and experimentation of bone drilling forces,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 451, 2013, p. 012 034.

[91] C. Santiuste, M. Rodriguez-Millan, E. Giner, and H. Miguelez, “The influence of anisotropy in numerical modeling of orthogonal cutting of cortical bone,” *Composite Structures*, vol. 116, pp. 423–431, 2014.

[92] S. Li, A. Abdel-Wahab, E. Demirci, and V. V. Silberschmidt, “Penetration of cutting tool into cortical bone: experimental and numerical investigation of anisotropic mechanical behaviour,” *Journal of Biomechanics*, vol. 47, no. 5, pp. 1117–1126, 2014.

[93] D. Umbrello, R. M’Saoubi, and J. Outeiro, “The influence of Johnson–Cook material constants on finite element simulation of machining of AISI 316L steel,” *International Journal of Machine Tools and Manufacture*, vol. 47, no. 3-4, pp. 462–470, 2007.

[94] C. Duan, T. Dou, Y. Cai, and Y. Li, “Finite element simulation and experiment of chip formation process during high speed machining of AISI 1045 hardened steel,” *International Journal on Production and Industrial Engineering*, vol. 2, no. 1, p. 28, 2011.

[95] A. Molinari, R. Cheriguene, and H. Miguelez, “Contact variables and thermal effects at the tool–chip interface in orthogonal cutting,” *International Journal of Solids and Structures*, vol. 49, no. 26, pp. 3774–3796, 2012.

[96] B. Takabi and B. L. Tai, “A review of cutting mechanics and modeling techniques for biological materials,” *Medical Engineering & Physics*, vol. 45, pp. 1–14, 2017.

[97] W. McMahan and K. J. Kuchenbecker, “Haptic Displayof Realistic Tool Contact via Dynamically Compensated Control of a Dedicated Actuator,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3170–3177.

[98] M. Elbestawi, F. Ismail, R. Du, and B. Ullagaddi, “Modelling machining dynamics including damping in the tool-workpiece interface,” *Journal of Engineering for Industry*, vol. 116, no. 4, pp. 435–439, 1994.

[99] J. A. Yang, V. Jaganathan, and R. Du, “A new dynamic model for drilling and reaming processes,” *International Journal of Machine Tools and Manufacture*, vol. 42, no. 2, pp. 299–311, 2002.

[100] H. Paris, S. Tichkiewitch, and G. Peigne, “Modelling the vibratory drilling process to foresee cutting parameters,” *CIRP Annals-Manufacturing Technology*, vol. 54, no. 1, pp. 367–370, 2005.

[101] N. Guibert, H. Paris, and J. Rech, “A numerical simulator to predict the dynamical behavior of the self-vibratory drilling head,” *International Journal of Machine Tools and Manufacture*, vol. 48, no. 6, pp. 644–655, 2008.

[102] A. Jiménez, M. Arizmendi, and W. E. Cumbicus, “Model for the prediction of low-frequency lateral vibrations in drilling process with pilot hole,” *The International Journal of Advanced Manufacturing Technology*, vol. 96, no. 5-8, pp. 1971–1990, 2018.

[103] A. Ghasemloonia, S. Baxandall, K. Zareinia, J. T. Lui, J. C. Dort, G. R. Sutherland, and S. Chan, “Evaluation of haptic interfaces for simulation of drill vibration in virtual temporal bone surgery,” *Computers in Biology and Medicine*, vol. 78, pp. 9–17, 2016.

[104] K. Yu, S. Iwata, K. Ohnishi, S. Usuda, T. Nakagawa, and H. Kawana, “Modeling and experimentation of drilling vibration for implant cutting force presenting system,” in *Advanced Motion Control (AMC), 2014 IEEE 13th International Workshop on*, IEEE, 2014, pp. 711–716.

[105] J. R. Kusins, J. A. Strelzow, M.-E. LeBel, and L. M. Ferreira, “Development of a vibration haptic simulator for shoulder arthroplasty,” *International Journal of Computer Assisted Radiology and Surgery*, pp. 1–14, 2018.

[106] M. Sharma, “Experimental Determination of Motion Paramters and Path Forces of Robot-Driven Glenoid Reaming,” Master’s thesis, The University of Western Ontario, 2018.

[107] *Unity*, `https://unity3d.com/`, Accessed: 2018-11-07.

[108] *What is Unreal Engine 4*, `https://www.unrealengine.com/en-US/what-is-unreal-engine-4`, Accessed: 2018-11-07.

[109] N. Ghrairi, S. Kpodjedo, A. Barrak, F. Petrillo, and F. Khomh, "The State of Practice on Virtual Reality (VR) Applications: An Exploratory Study on Github and Stack Overflow," in *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, 2018, pp. 356–366.

[110] B. Siciliano and O. Khatib, *Handbook of robotics*. Springer, 2016.

[111] A. Williams, *C++ concurrency in action*. Manning, 2012.

[112] *Open Haptics Developer Software*, `https://www.3dsystems.com/haptics-devices/openhaptics`, Accessed: 2018-11-07.

[113] *CHAI3D*, `http://www.chai3d.org/`, Accessed: 2018-11-07.

[114] *OpenGL - The Industry Standard for High Performance Graphics*, `https://www.opengl.org/`, Accessed: 2018-11-07.

[115] *VTK - The Visualization Toolkit*, `https://www.vtk.org/`, Accessed: 2018-11-07.

[116] G. Pratx and L. Xing, "GPU computing in medical physics: A review," *Medical Physics*, vol. 38, no. 5, pp. 2685–2697, 2011.

[117] H. Courtecuisse, H. Jung, J. Allard, C. Duriez, D. Y. Lee, and S. Cotin, "GPU-based real-time soft tissue deformation with cutting and haptic feedback," *Progress in Biophysics and Molecular Biology*, vol. 103, no. 2-3, pp. 159–168, 2010.

[118] Z. A. Taylor, M. Cheng, and S. Ourselin, "High-speed nonlinear finite element analysis for surgical simulation using graphics processing units," *IEEE Transactions on Medical Imaging*, vol. 27, no. 5, pp. 650–663, 2008.

[119] O. Comas, Z. A. Taylor, J. Allard, S. Ourselin, S. Cotin, and J. Passenger, "Efficient nonlinear FEM for soft tissue modelling and its GPU implementation within the open source framework SOFA," in *International Symposium on Biomedical Simulation*, Springer, 2008, pp. 28–39.

[120] C. Dick, J. Georgii, and R. Westermann, "A real-time multigrid finite hexahedra method for elasticity simulation using CUDA," *Simulation Modelling Practice and Theory*, vol. 19, no. 2, pp. 801–816, 2011.

[121] G. Echegaray, I. Herrera, I. Aguinaga, C. Buchart, and D. Borro, "A brain surgery simulator," *IEEE computer Graphics and Applications*, vol. 34, no. 3, pp. 12–18, 2014.

[122] *CUDA Toolkit Documentation*, `https://docs.nvidia.com/cuda/`, Accessed: 2018-11-07.

[123] *The open standard for parallel programming of heterogeneous systems*, `https://www.khronos.org/opencl/`, Accessed: 2018-11-07.

[124] M. Scarpino, *OpenCL in action*. Manning, 2011.

[125] *The OpenCL C Specification*, `https://www.khronos.org/registry/OpenCL/specs/opencl-1.2.pdf`, Accessed: 2018-11-07.

[126] M. Lin and S. Gottschalk, “Collision detection between geometric models: A survey,” in *Proc. of IMA Conference on Mathematics of Surfaces*, vol. 1, 1998, pp. 602–608.

[127] P. Jiménez, F. Thomas, and C. Torras, “3D collision detection: a survey,” *Computers & Graphics*, vol. 25, no. 2, pp. 269–285, 2001.

[128] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, *et al.*, “Collision detection for deformable objects,” in *Computer Graphics Forum*, Wiley Online Library, vol. 24, 2005, pp. 61–81.

[129] S. Kockara, T. Halic, K. Iqbal, C. Bayrak, and R. Rowe, “Collision detection: A survey,” in *International Conference on Systems, Man and Cybernetics*, IEEE, 2007, pp. 4046–4051.

[130] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, “Six degree-of-freedom haptic rendering using voxel sampling,” in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., 1999, pp. 401–408.

[131] M. Renz, C. Preusche, M. Pötke, H.-P. Kriegel, and G. Hirzinger, “Stable haptic interaction with virtual environments using an adapted voxmap-pointshell algorithm,” in *In Proc. Eurohaptics*, 2001.

[132] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, “Voxel-based 6-DOF haptic rendering improvements,” *Haptics-e*, 2006.

[133] M. Sagardia, T. Hulin, C. Preusche, and G. Hirzinger, “Improvements of the voxmap-pointshell algorithm-fast generation of haptic data-structures,” in *53rd IWK-Internationales Wissenschaftliches Kolloquium, Ilmenau, Germany*, 2008.

[134] H. Xu and J. Barbic, “Adaptive 6-DOF haptic contact stiffness using the gauss map,” *IEEE Transactions on Haptics*, no. 3, pp. 323–332, 2016.

[135] J. Barbič and D. L. James, “Six-DOF haptic rendering of contact between geometrically complex reduced deformable models,” *IEEE Transactions on Haptics*, vol. 1, no. 1, 2008.

[136] K. Moustakas, “6DOF haptic rendering using distance maps over implicit representations,” *Multimedia Tools and Applications*, vol. 75, no. 8, pp. 4543–4557, 2016.

[137] H. Xu, Y. Zhao, and J. Barbic, “Implicit multibody penalty-baseddistributed contact,” *IEEE Transactions on Visualization & Computer Graphics*, no. 9, pp. 1266–1279, 2014.

[138] H. Xu and J. Barbič, “6-DOF Haptic Rendering Using Continuous Collision Detection between Points and Signed Distance Fields,” *IEEE Transactions on Haptics*, vol. 10, no. 2, pp. 151–161, 2017.

[139] B. H. M. T. M. Gross, “Real-time volumetric intersections of deforming objects,” in *Vision, Modeling, and Visualization: Proceedings*, 2003, p. 461.

[140] B. Heidelberger, M. Teschner, and M. Gross, “Detection of collisions and self-collisions using image-space techniques,” *Journal of WSCG*, vol. 12, no. 3, pp. 145–152, 2004.

[141] T. D. Tang, “Algorithms for collision detection and avoidance for five-axis NC machining: a state of the art review,” *Computer-Aided Design*, vol. 51, pp. 1–17, 2014.

[142] H. T. Yau, L. S. Tsou, and Y. C. Tong, “Adaptive NC simulation for multi-axis solid machining,” *Computer-Aided Design and Applications*, vol. 2, no. 1-4, pp. 95–104, 2005.

[143] T. Akenine-Möllser, “Fast 3D triangle-box overlap testing,” *Journal of Graphics Tools*, vol. 6, no. 1, pp. 29–33, 2001.

[144] Z. Dong, W. Chen, H. Bao, H. Zhang, and Q. Peng, “Real-time voxelization for complex polygonal models,” in *12th Pacific Conference on Computer Graphics and Applications*, IEEE, 2004, pp. 43–50.

[145] E. Eisemann and X. Décoret, “Fast scene voxelization and applications,” in *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM, 2006, pp. 71–78.

[146] ——, “Single-pass GPU solid voxelization for real-time applications,” in *Proceedings of graphics interface 2008*, Canadian Information Processing Society, 2008, pp. 73–80.

[147] S. Fang and H. Chen, “Hardware accelerated voxelisation,” in *Volume Graphics*, Springer, 2000, pp. 301–315.

[148] W. Li, Z. Fan, X. Wei, and A. Kaufman, “GPU-Based flow simulation with complex boundaries,” in *GPU Gems 2*, Addison Wesley, 2003, pp. 747–764.

[149] L. Zhang, W. Chen, D. S. Ebert, and Q. Peng, “Conservative voxelization,” *The Visual Computer*, vol. 23, no. 9-11, pp. 783–792, 2007.

[150] Y. Fei, B. Wang, and J. Chen, “Point-tessellated voxelization,” in *Proceedings of Graphics Interface 2012*, Canadian Information Processing Society, 2012, pp. 9–18.

[151] M. Schwarz and H.-P. Seidel, “Fast parallel surface and solid voxelization on GPUs,” *ACM Transactions on Graphics (TOG)*, vol. 29, no. 6, p. 179, 2010.

[152] J.-Y. Rho, M. Hobatho, and R. Ashman, “Relations of mechanical properties to density and CT numbers in human bone,” *Medical Engineering & Physics*, vol. 17, no. 5, pp. 347–355, 1995.

[153] N. K. Knowles, J. M. Reeves, and L. M. Ferreira, “Quantitative Computed Tomography (QCT) derived Bone Mineral Density (BMD) in finite element studies: a review of the literature,” *Journal of Experimental Orthopaedics*, vol. 3, no. 1, p. 36, 2016.

[154] M. L. Bouxsein, S. K. Boyd, B. A. Christiansen, R. E. Guldberg, K. J. Jepsen, and R. Müller, “Guidelines for assessment of bone microstructure in rodents using micro–computed tomography,” *Journal of Bone and Mineral Research*, vol. 25, no. 7, pp. 1468–1486, 2010.

[155] B. van Rietbergen, “Micro-FE analyses of bone: state of the art,” in *Noninvasive assessment of trabecular bone architecture and the competence of bone*, Springer, 2001, pp. 21–30.

[156] T. J. Baker, “Mesh generation: Art or science?” *Progress in Aerospace Sciences*, vol. 41, no. 1, pp. 29–63, 2005.

[157] Y. N. Yeni and D. P. Fyhrie, "Finite element calculated uniaxial apparent stiffness is a consistent predictor of uniaxial apparent strength in human vertebral cancellous bone tested with different boundary conditions," *Journal of Biomechanics*, vol. 34, no. 12, pp. 1649–1654, 2001.

[158] W. Pistoia, B. Van Rietbergen, E.-M. Lochmüller, C. Lill, F. Eckstein, and P. Rüegsegger, "Estimation of distal radius failure load with micro-finite element analysis models based on three-dimensional peripheral quantitative computed tomography images," *Bone*, vol. 30, no. 6, pp. 842–848, 2002.

[159] S. J. Shefelbine, U. Simon, L. Claes, A. Gold, Y. Gabet, I. Bab, R. Müller, and P. Augat, "Prediction of fracture callus mechanical properties using micro-CT images and voxel-based finite element analysis," *Bone*, vol. 36, no. 3, pp. 480–488, 2005.

[160] U. Wolfram, H.-J. Wilke, and P. K. Zysset, "Valid μ finite element models of vertebral trabecular bone can be obtained using tissue properties measured with nanoindentation under wet conditions," *Journal of Biomechanics*, vol. 43, no. 9, pp. 1731–1737, 2010.

[161] A. Torcasio, X. Zhang, H. Van Oosterwyck, J. Duyck, and G. H. van Lenthe, "Use of micro-CT-based finite element analysis to accurately quantify peri-implant bone strains: a validation in rat tibiae," *Biomechanics and Modeling in Mechanobiology*, vol. 11, no. 5, pp. 743–750, 2012.

[162] D. Christen, L. J. Melton III, A. Zwahlen, S. Amin, S. Khosla, and R. Müller, "Improved Fracture Risk Assessment Based on Nonlinear Micro-Finite Element Simulations From HRpQCT Images at the Distal Radius," *Journal of Bone and Mineral Research*, vol. 28, no. 12, pp. 2601–2608, 2013.

[163] M. Ramezanzadehkoldeh and B. H. Skallerud, "MicroCT-based finite element models as a tool for virtual testing of cortical bone," *Medical Engineering & Physics*, vol. 46, pp. 12–20, 2017.

[164] M. C. Costa, G. Tozzi, L. Cristofolini, V. Danesi, M. Viceconti, and E. Dall'Ara, "Micro Finite Element models of the vertebral body: Validation of local displacement predictions," *PloS one*, vol. 12, no. 7, e0180151, 2017.

[165] C. C. Wang, "Approximate Boolean operations on large polyhedral solids with partial mesh reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 6, pp. 836–849, 2011.

[166] S. Landier, "Boolean operations on arbitrary polyhedral meshes," *Procedia Engineering*, vol. 124, pp. 200–212, 2015.

[167] *The Computational Geometry Algorithms Library*, `https://www.cgal.org/`, Accessed: 2018-11-07.

[168] ——, "Boolean operations on arbitrary polygonal and polyhedral meshes," *Computer-Aided Design*, vol. 85, pp. 138–153, 2017.

[169] J. Amanatides, A. Woo, *et al.*, "A fast voxel traversal algorithm for ray tracing," in *Eurographics*, vol. 87, 1987, pp. 3–10.

[170] M. Levoy, "Efficient ray tracing of volume data," *ACM Transactions on Graphics (TOG)*, vol. 9, no. 3, pp. 245–261, 1990.

[171] T. J. Purcell, I. Buck, W. R. Mark, and P. Hanrahan, “Ray tracing on programmable graphics hardware,” in *ACM SIGGRAPH 2005 Courses*, ACM, 2005, p. 268.

[172] D. Laur and P. Hanrahan, “Hierarchical splatting: A progressive refinement algorithm for volume rendering,” in *ACM SIGGRAPH Computer Graphics*, ACM, vol. 25, 1991, pp. 285–288.

[173] K. Mueller and R. Yagel, “Fast perspective volume rendering with splatting by utilizing a ray-driven approach,” in *Proceedings of the 7th conference on Visualization’96*, IEEE Computer Society Press, 1996, 65–ff.

[174] P. P. Li, S. Whitman, R. Mendoza, and J. Tsiao, “ParVox-a parallel splatting volume rendering system for distributed visualization,” in *Parallel Rendering, 1997. PRS 97. Proceedings. IEEE Symposium on*, IEEE, 1997, pp. 7–14.

[175] W. Li, K. Mueller, and A. Kaufman, “Empty space skipping and occlusion clipping for texture-based volume rendering,” in *Proceedings of the 14th IEEE Visualization 2003 (VIS’03)*, IEEE Computer Society, 2003, p. 42.

[176] J. Kniss, P. McCormick, A. McPherson, J. Ahrens, J. Painter, A. Keahey, and C. Hansen, “Interactive texture-based volume rendering for large data sets,” *IEEE Computer Graphics and Applications*, no. 4, pp. 52–61, 2001.

[177] I. Boada, I. Navazo, and R. Scopigno, “Multiresolution volume visualization with a texture-based octree,” *The Visual Computer*, vol. 17, no. 3, pp. 185–197, 2001.

[178] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” in *ACM SIGGRAPH Computer Graphics*, ACM, vol. 21, 1987, pp. 163–169.

[179] S. F. Gibson, “Constrained elastic surface nets: Generating smooth surfaces from binary segmented data,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 1998, pp. 888–898.

[180] T. Ju, F. Losasso, S. Schaefer, and J. Warren, “Dual contouring of hermite data,” in *ACM Transactions on Graphics (TOG)*, ACM, vol. 21, 2002, pp. 339–346.

[181] F. Goetz, T. Junklewitz, and G. Domik, “Real-Time Marching Cubes on the Vertex Shader.,” in *Eurographics (Short Presentations)*, 2005, pp. 5–8.

[182] E. Smistad, A. C. Elster, and F. Lindseth, “Real-time surface extraction and visualization of medical images using OpenCL and GPUs,” *Norsk informatikkonferanse*, pp. 141–152, 2012.

[183] *NVIDIA OpenCL SDK Code Samples*, `https://developer.nvidia.com/opencl`, Accessed: 2018-11-07.

[184] Y. Chevalier, D. Pahr, H. Allmer, M. Charlebois, and P. Zysset, “Validation of a voxel-based FE method for prediction of the uniaxial apparent modulus of human trabecular bone using macroscopic mechanical tests and nanoindentation,” *Journal of Biomechanics*, vol. 40, no. 15, pp. 3333–3340, 2007.

[185] N. M. Zaitoun and M. J. Aqel, “Survey on image segmentation techniques,” *Procedia Computer Science*, vol. 65, pp. 797–806, 2015.

[186] D. W. Wagner, D. P. Lindsey, and G. S. Beaupre, “Deriving tissue density and elastic modulus from microCT bone scans,” *Bone*, vol. 49, no. 5, pp. 931–938, 2011.

[187] B. C. Bourne and M. C. van der Meulen, “Finite element models predict cancellous apparent modulus when tissue modulus is scaled from specimen CT-attenuation,” *Journal of Biomechanics*, vol. 37, no. 5, pp. 613–621, 2004.

[188] N. M. Josuttis, *The C++ standard library: a tutorial and reference*. Addison-Wesley Professional, 2012.

[189] D. Cohen-Or and A. Kaufman, “Fundamentals of surface voxelization,” *Graphical Models and Image Processing*, vol. 57, no. 6, pp. 453–461, 1995.

[190] G. M. Morton, “A computer oriented geodetic data base and a new technique in file sequencing,” IBM Ltd., Tech. Rep., 1966.

[191] J. Baert, A. Lagae, and P. Dutré, “Out-of-core construction of sparse voxel octrees,” in *Proceedings of the 5th High-Performance Graphics Conference*, ACM, 2013, pp. 27–32.

[192] S. Gottschalk, M. C. Lin, and D. Manocha, “OBBTree: A hierarchical structure for rapid interference detection,” in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ACM, 1996, pp. 171–180.

[193] C. Ericson, *Real-time collision detection*. CRC Press, 2004.

[194] C. Jacobs, M. Pope, J. Berry, and F. Hoaglund, “A study of the bone machining process - orthogonal cutting,” *Journal of Biomechanics*, vol. 7, no. 2, pp. 131–136, 1974.

[195] C. Plaskos, A. J. Hodgson, and P. Cinquin, “Modelling and optimization of bone-cutting forces in orthopaedic surgery,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2003, pp. 254–261.

[196] M. Mitsuishi, S. Warisawa, N. Sugita, M. Suzuki, H. Moriya, H. Hashizume, K. Fujiwara, N. Abe, H. Inoue, K. Kuramoto, *et al.*, “A study of bone micro-cutting characteristics using a newly developed advanced bone cutting machine tool for total knee arthroplasty,” *CIRP Annals-Manufacturing Technology*, vol. 54, no. 1, pp. 41–46, 2005.

[197] V. T. Van Hees, L. Gorzelniak, E. C. D. Leon, M. Eder, M. Pias, S. Taherian, U. Ekelund, F. Renström, P. W. Franks, A. Horsch, *et al.*, “Separating movement and gravity components in an acceleration signal and implications for the assessment of human daily physical activity,” *PloS one*, vol. 8, no. 4, e61691, 2013.

[198] A. M. Okamura, J. T. Dennerlein, and R. D. Howe, “Vibration feedback models for virtual environments,” in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, IEEE, vol. 1, 1998, pp. 674–679.

[199] A. J. Silva, O. A. D. Ramirez, V. P. Vega, and J. P. O. Oliver, “PHANToM OMNI haptic device: Kinematic and manipulability,” in *Electronics, Robotics and Automotive Mechanics Conference, 2009. CERMA’09.*, IEEE, 2009, pp. 193–198.

[200] *Multimedia Timers*, `https://docs.microsoft.com/en-ca/windows/desktop/Multimedia/multimedia-timers`, Accessed: 2018-11-07.

[201] Z. Wang, “A Quadtree-based adaptive Cartesian/Quad grid flow solver for Navier-Stokes equations,” *Computers & Fluids*, vol. 27, no. 4, pp. 529–549, 1998.

[202] E. M. Padegimas, M. Maltenfort, M. D. Lazarus, M. L. Ramsey, G. R. Williams, and S. Namdari, “Future patient demand for shoulder arthroplasty by younger patients: national projections,” *Clinical Orthopaedics and Related Research®*, vol. 473, no. 6, pp. 1860–1867, 2015.

[203] N. K. Knowles, G. D. G. Langohr, M. Faieghi, A. Nelson, and L. M. Ferreira, “Development and Cross-Validation of a CT-Compatible Loading Device for Mechanical Testing of Trabecular Bone Specimens,” in *Annual Meeting of the Orthopaedic Research Society*, 2018.

[204] J. R. Kusins, N. K. Knowles, M. Faieghi, N. K. Knowles, A. Nelson, and L. M. Ferreira, “Accuracy of Density-Modulus Relationships Used in Finite Element Modeling of the Shoulder,” in *World Congress on Biomechanics*, 2018.

[205] N. K. Knowles, G. D. G. Langohr, M. Faieghi, A. Nelson, and L. M. Ferreira, “Development of a validated glenoid trabecular density-modulus relationship,” *Journal of the mechanical behavior of biomedical materials*, vol. 90, pp. 140–145, 2019.

[206] M. Faieghi, N. K. Knowles, O. R. Tutunea-Fatan, and L. M. Ferreira, “Efficient Voxelization-Based Construction of Finite Element Meshes,” in *Computer Aided Design*, 2018.

[207] ——, “An efficient hexahedral mesh generation algorithm for micro-level trabecular bone modeling,” in *World Congress on Biomechanics*, 2018.

[208] ——, “Fast Generation of Cartesian Meshes from Micro-Computed Tomography Data,” *Computer Aided Design and Application*, vol. 16, no. 1, pp. 161–171, 2019.

[209] *Cartesian Mesh Generator*, `https://github.com/mfaieghi/HexahedralFiniteElementModelGenerator`, Accessed: 2018-11-07.

[210] C. Song, Z. Pang, X. Jing, and C. Xiao, “Distance field guided L_1-median skeleton extraction,” *The Visual Computer*, vol. 34, no. 2, pp. 243–255, 2018.

[211] F. Wang, “Composite model representation for computer aided design of functionally gradient materials,” PhD thesis, Missouri University of Science and Technology, 2016.

[212] K. Vidimče, S.-P. Wang, J. Ragan-Kelley, and W. Matusik, “OpenFab: a programmable pipeline for multi-material fabrication,” *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 136, 2013.

[213] J. C. Dinis, T. F. Moraes, P. H. Amorim, M. R. Moreno, A. A. Nunes, and J. V. Silva, “POMES: An Open-Source Software Tool to Generate Porous/Roughness on Surfaces,” *Procedia CIRP*, vol. 49, pp. 178–182, 2016.

[214] F. S. Nooruddin and G. Turk, “Simplification and repair of polygonal models using volumetric techniques,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 191–205, 2003.

[215] S. Patil and B. Ravi, “Voxel-based representation, display and thickness analysis of intricate shapes,” in *Computer Aided Design and Computer Graphics, 2005. Ninth International Conference on*, IEEE, 2005, 6–pp.

[216] M. Faieghi, O. R. Tutunea-Fatan, and R. Eagleson, “Fast and Cross-vendor OpenCL-based Implementation for Voxelization of Triangular Mesh Models,” in *Computer Aided Design*, 2017.

[217] ——, “Fast and cross-vendor OpenCL-based implementation for voxelization of triangular mesh models,” *Computer-Aided Design and Applications*, vol. 15, no. 6, pp. 852–862, 2018.

[218] *OpenCL Voxelizer*, `https://github.com/mfaieghi/OpenCLVoxelizer`, Accessed: 2018-11-07.

[219] M. Shaw and C. Oxford, “On the drilling of metals 1: basic mechanics of the process,” *Trans. ASME*, vol. 77, no. 2, pp. 103–111, 1955.

[220] G. Stabler, “The chip flow law and its consequences,” *Advances in Machine Tool Design and Research*, vol. 5, pp. 243–251, 1964.

Curriculum Vitae

Name:	Mohammadreza Faieghi
Post-Secondary Education and Degrees:	Hamedan University of Technology Hamedan, Iran 2006 – 2011 B.Sc. in Electrical Engineering
	Iran University of Science and Technology Tehran, Iran 2011 – 2013 M.Sc. in Electrical Engineering
Honours and Awards:	PSAC 610 Academic Scholarship The University of Western Ontario 2017
	Ontario Graduate Scholarship The University of Western Ontario 2016 – 2017
	Ontario Graduate Scholarship The University of Western Ontario 2015 – 2016
	Outstanding Student Award Iran University of Science and Technology 2011 – 2012
	Best Poster Award 5th Symposium on Fractional Differentiation and its Applications, Hohai University, China 2012
Related Experience:	Teaching Assistant The University of Western Ontario 2014 – 2018

Journal Papers:

1. N. K. Knowles, G. D. G. Langohr, **M. Faieghi**, A. Nelson and L. M. Ferreira, “Development of a Validated Glenoid Trabecular Density-Modulus Relationship”, *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 90, pp. 140–145, 2019.
2. **M. Faieghi**, N. K. Knowles, O. R. Tutunea-Fatan and L. M. Ferreira, “Fast Generation of Cartesian Meshes from Micro-Computed Tomography Data”, *Computer Aided Design and Application*, vol. 16, no. 1, pp. 161–171, 2019.
3. **M. Faieghi**, O. R. Tutunea-Fatan and R. Eagleson, “OpenCL-Based Voxelization of Triangle Mesh Models”, *Computer Aided Design and Applications*, vol. 15, no. 9, pp. 852–862, 2018.
4. **M. Faieghi**, A. A. Jalali, S. K. Mousavi Mashhadi and D. Baleanu, “Passivity-Based Cruise Control of High Speed Trains”, *Journal of Vibration and Control*, vol. 24, no. 13, pp. 492-504, 2018.
5. **M. Faieghi**, S. K. Mousavi Mashhadi and D. Baleanu, “Sampled-Data Nonlinear Observer Design for Chaos Synchronization: A Lyapunov-Based Approach”, *Communications in Nonlinear Science and Numerical Simulations*, vol. 19, no. 7, pp. 2444–2453, 2014.
6. **M. Faieghi**, S. Kuntanapreeda, H. Delavari and D. Baleanu, “Robust Stabilization of Fractional Order Chaotic Systems with Linear Controllers: LMI-Based Sufficient Conditions”, *Journal of Vibration and Control*, vol. 20, no. 7, pp. 1042–1052, 2014.
7. **M. Faieghi**, A. A. Jalali, S. K. Mousavi Mashhadi, “Robust Adaptive Cruise Control of High Speed Trains”, *ISA Transactions*, vol. 53, no. 2, pp. 533–541, 2014.
8. **M. Faieghi**, H. Delavari and D. Baleanu, “A note on stability of sliding mode dynamics in suppression of fractional-order chaotic systems”, *Computers and Mathematics with Application*, vol. 66, no. 5, pp. 832–837, 2013.
9. **M. Faieghi**, A. A. Jalali, S. K. Mousavi Mashhadi and S. A. Zahiripour, “Adaptive Sliding Mode Controller Design for Cruise Control of High Speed Trains”, *Journal of Control*, vol. 7, no. 1, pp. 1–12, 2013.
10. **M. Faieghi**, S. Kuntanapreeda, H. Delavari and D. Baleanu, “LMI-based Stabilization of a Class of Fractional-Order Chaotic Systems”, *Nonlinear Dynamics*, vol. 72, no. 1–2, pp. 301–309, 2013.
11. M. Naderi and **M. Faieghi**, “Comments on “Second-order sliding mode control with experimental application””, *ISA Transactions*, vol. 51, no. 6, pp. 861–862, 2012.
12. **M. Faieghi**, H. Delavari and D. Baleanu, “Control of an Uncertain Fractional-Order Liu System via Fuzzy Fractional-Order Sliding Mode Control”, *Journal of Vibration and Control*, vol. 18, no. 9, pp. 1366–1374, 2012.

13. **M. Faieghi**, H. Delavari and D. Baleanu, “A novel adaptive controller for two-degree of freedom polar robot with unknown perturbations”, *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 2, pp. 1021–1030, 2012.

14. **M. Faieghi** and H. Delavari, “Chaos in fractional-order Genesio-Tesi system and its synchronization”, *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 2, pp. 731–741, 2012.

Book Chapters:

1. D. M. Senejohnny, **M. Faieghi** and H. Delavari, “Adaptive Second-Order Fractional Sliding Mode Control with Application to Water Tanks Level Control”, in *Fractional calculus: History, Theory and Application*, Nova Publisher, 2015, pp. 149–164,

2. A. Nemati and **M. Faieghi**, “The Performance Comparison of ANFIS and Hammerstein-Wiener Models for BLDC Motors”, in *Electronics and Signal Processing*, Springer, 2011, pp. 29–37.

3. **M. Faieghi** and A. Nemati, “On Fractional-Order PID Design”, in *Applications of MATLAB in Science and Engineering*, InTech, 2011, pp. 273–292.

Conference Papers and Abstracts:

1. **M. Faieghi**, N. K. Knowles, O. R. Tutunea-Fatan and L. M. Ferreira, “An efficient hexahedral mesh generation algorithm for micro-level trabecular bone modeling”, presented at the *World Congress on Biomechanics*, Dublin, Ireland, 2018.

2. N. K. Knowles, G. D.G. Langohr, **M. Faieghi**, A. Nelson and L. M. Ferreira, “Accuracy of Density-Modulus Relationships Used in Finite Element Modeling of the Shoulder”, presented at the *World Congress on Biomechanics*, Dublin, Ireland, 2018.

3. **M. Faieghi**, N. K. Knowles, O. R. Tutunea-Fatan and L. M. Ferreira, “Voxelization-based Fast Construction of Finite Element Models from Micro-Computed Tomography”, in *Proceedings of CAD18*, Paris, France, 2018, pp. 21–25.

4. J. R. Kusins, N. K. Knowles, **M. Faieghi**, A. Nelson and L. M. Ferreira, “Development and Cross-Validation of a CT-Compatible Loading Device for Mechanical Testing of Trabecular Bone Specimens”, presented at the *2018 Annual Meeting of the Orthopaedic Research Society*, New Orleans, LA, US, 2018.

5. N. K. Knowles, G. D.G. Langohr, **M. Faieghi**, A. Nelson and L. M. Ferreira, “Development of a Validated Glenoid Trabecular Density-Modulus Relationship”, presented at the *Annual Meeting of the Orthopaedic Research Society*, New Orleans, LA, USA, 2018.

6. **M. Faieghi**, O. R. Tutunea-Fatan and R. Eagleson, “Fast and Cross-Vendor OpenCL-Based Implementation for Voxelization of Triangular Mesh Models”, in *Proceedings of CAD17*, Okayama, Japan, 2017, pp. 410–414.

7. Y. Jalalabadi, M. Naderi and **M. Faieghi**, “Improvement of Hard Disk Drive Positioning Using Fractional-Order Controller”, in *proceedings of the 1st National Symposium on Electrical and Computer Engineering of South of Iran*, Khormoj, Iran, 2013.
8. **M. Faieghi** and A. Jalali, “Robust Velocity Tracking of High Speed Train via Sliding Mode Control”, in *Proceedings of the 3rd International Conference on Recent Advances in Railway Engineering*, Tehran, Iran, 2013.
9. **M. Faieghi**, S. Kuntanapreeda, H. Delavari and D. Baleanu, “Stabilization of a Class Fractional Chaotic System with a Simple Controller”, in *Proceedings of 5th IFAC Symposium on Fractional Differentiation and its Applications*, Nanjing, China, 2012.
10. **M. Faieghi**, H. Delavari and D. Baleanu, “A Note on Stability of Sliding Mode Dynamics in Suppression of Fractional-Order Chaotic Systems”, in *proceedings of 5th IFAC Symposium on Fractional Differentiation and its Applications*, Nanjing, China, 2012.
11. D. M. Senejohnny, **M. Faieghi** and H. Delavari, “Adaptive Synchronization of Fractional Order Chaotic Systems with Unknown Perturbation”, in *proceedings of 5th IFAC Symposium on Fractional Differentiation and its Applications*, Nanjing, China, 2012.
12. **M. Faieghi**, H. Delavari and A. Jalali, “Control of Lorenz System with a Fractional Controller: A Caputos Differintegration Based Approach”, in *proceedings of 2nd International Conference on Control, Instrumentation, and Automation*, Shiraz, Iran, 2011, pp. 616–620.
13. **M. Faieghi**, M. Naderi and A. A. Jalali, “Design of Fractional-Order PID for Ship Roll Motion Control using Chaos Embedded PSO Algorithm”, in *proceedings of 2nd International Conference on Control, Instrumentation, and Automation*, Shiraz, Iran, 2011, pp. 606–610.
14. H. Delavari, **M. Faieghi** and A. Ranjbar, “Fractional-Order Sliding Mode Controller for Inverted Pendulum”, in *proceedings of 4th IFAC Workshop Fractional Differentiation and its Applications*, Badajoz, Spain, 2010.
15. **M. Faieghi**, H. Delavari and D. Baleanu, “Adaptive Active Sliding Mode Control for Two-Degree of Freedom Polar Robot”, in *proceedings of 3rd Conference on Nonlinear Science and Complexity*, Ankara, Turkey, 2010.
16. **M. Faieghi** and H. Delavari, “Control of an Uncertain Fractional-Order Chaotic System via Fuzzy Fractional-Order Sliding Mode Control”, in *proceedings of 13th Iranian Student Conference on Electrical Engineering*, Tehran, Iran, 2010.
17. **M. Faieghi** and S. M. Azimi, “Design an Optimized PID Controller for Brushless DC Motor by Using PSO and Based on NARMAX Identified Model with ANFIS”, in *proceedings of 12th International Conference on Computer Modelling and Simulation*, Cambridge, UK, 2010, pp. 16–21.