

Virtual Environment and Interface with a Haptic Device for a Medical Simulator

Author: KONSTANTINOS KARGAS

Supervisor Professor: KONSTANTINOS VLACHOS

Thesis submitted in partial fulfilment of the
requirements for the degree of Master of Science in
Computer Science and Engineering

Ioannina, January, 2025



**ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA**

Ευχαριστίες

Θα ήθελα να εκφράσω την ειλικρινή μου ευγνωμοσύνη σε όλους όσοι συνέβαλαν στην ολοκλήρωση αυτής της διπλωματικής εργασίας. Ιδιαίτερα, θα ήθελα να ευχαριστήσω:

- Τον επιβλέποντα καθηγητή μου, κ. Κωνσταντίνο Βλάχο, για την καθοδήγηση, την υπομονή, την υποστήριξη και τις πολύτιμες συμβουλές του καθ' όλη τη διάρκεια αυτής της εργασίας.
- Τους συναδέλφους και συμφοιτητές μου για τη βοήθειά τους, την ανταλλαγή ιδεών και τη συνεργασία τους κατά τη διάρκεια των πειραμάτων.
- Την οικογένεια και τους φίλους μου για την αδιάκοπη υποστήριξη και κατανόησή τους καθ' όλη τη διάρκεια των σπουδών μου.
- Το προσωπικό του Εργαστηρίου Ρομποτικής για την τεχνική υποστήριξη και τις διευκολύνσεις που παρείχαν για την πραγματοποίηση των πειραμάτων.

Η συμβολή όλων όσοι συμμετείχαν ήταν καθοριστική για την επιτυχή ολοκλήρωση αυτής της προσπάθειας, και σας ευχαριστώ όλους από καρδιάς.

Ημερομηνία: 05/01/2025

Συγγραφέας: Κωνσταντίνος Κάργας

Acknowledgements

I would like to express my heartfelt gratitude to all those who contributed to the completion of this diploma thesis. In particular, I wish to thank:

- My supervisor professor, Mr. Konstantinos Vlachos, for his guidance, patience, support, and valuable advice throughout the course of this work.
- My colleagues and fellow students for their assistance, the exchange of ideas, and their collaboration during the experiments.
- My family and friends for their unwavering support and understanding throughout my studies.
- The staff of the Robotics Laboratory for their technical support and the facilities they provided to carry out the experiments.

The contribution of everyone involved was instrumental in the successful completion of this effort, and I thank you all from the bottom of my heart.

Date: 05/01/2025

Author: KONSTANTINOS KARGAS

Abstract

This diploma thesis presents the development of a simulation framework for the realistic modeling of skin surgical procedures, specifically focused on melanoma removal. The core of the simulation is built on a spring-mass-damper model, accurately replicating the mechanical properties of human skin, including elasticity, deformation, and tension. A dynamic 3D mesh serves as the foundation of the simulation, enabling interactive responses and realistic visual effects during the cutting process.

To achieve high mechanical accuracy and visual fidelity, the simulation employs advanced techniques such as vertex splitting, state transitions, and collision detection. A key feature of the system is its integration with a haptic mechanism, allowing users to interact with the virtual environment through force feedback. This connection enables tactile sensations that closely mimic real surgical procedures, significantly enhancing realism and interactivity.

The report also documents the challenges encountered during development, including mesh deformation handling and collision detection optimization. The final system is designed for use in a laboratory setting, featuring device-to-device communication to ensure smooth interaction between the simulation and robotic mechanisms.

Overall, this thesis contributes to medical simulation technology by combining physics-based modeling with haptic feedback, offering an immersive and effective tool for medical training and surgical procedure analysis.

Key Words: Spring-Mass-Damper Model, Human Skin Simulation, Melanoma Removal, Haptic Feedback, Vertex Splitting, Mesh Deformation, Medical Simulation, Surgical Training, Physics-Based Modeling, Interactive Simulation, Collision Detection, Epidermis Modeling, Texture Mapping, TCP-Connection, Real-Time Interaction, Robotic Mechanism

Contents

Chapter 1. INTRODUCTION	1
Chapter 2. BACKGROUND & RELATED WORK.....	3
2.1 Soft Tissue Simulation.....	3
2.2 Haptic Feedback in Medical Simulations.....	4
2.3 Cutting Mechanisms in Simulations	5
2.4 Project Contributions	5
Chapter 3. PROBLEM STATEMENT & OBJECTIVES	6
3.1 Problem Statement.....	6
3.2 Objectives	7
3.3 Initial Challenges and Development Decisions.....	8
Chapter 4. METHODOLOGY.....	9
4.1 General Overview of the Mass-Spring-Damper (MSD) Model.....	9
4.1.1 Equation of Motion	10
4.1.2 System Behavior Analysis.....	10
4.1.3 SMD Model Applications.....	10
4.2 Simulation Model for Skin Surgery	11
4.2.1 General Model Information.....	11
4.2.2 Purpose of A Surgical Simulation.....	11
4.2.3 Skin Structure and Mechanical Properties.....	12
4.2.4 Integration of Forces Using Euler's Method with Multi-Directional Springs ..	13
4.2.5 Euler's Simple Method for Visual Effect Force Calculation	14
4.3 User-perceived Force Calculations	15
4.3.1 Vertical Force F_y – Initial Compression and Penetration.....	15
4.3.2 Horizontal Force F_x – Cutting and Horizontal Drag.....	15
4.3.3 Diagonal Springs – Diagonal Force Components and Mixed Displacement	16
4.3.4 Breaking Point & Directional Mass Displacement	16
4.3.5 Total Force Calculation with Angle of Approach.....	17
4.3.6 Nonlinear Force Formulation with Multi-Directional Spring-Dampers.....	18
Chapter 5. DEVELOPMENT PLATFORM	19

5.1	General.....	19
5.2	Initial Exploration of Platforms.....	20
5.3	Challenges with Soft-Body Dynamics and Real-Time Interaction	22
5.4	Transition to Low-Level Programming.....	22
Chapter 6. IMPLEMENTATION		23
6.1	Installation and Setup of OpenGL in WSL Environment.....	23
6.2	Implementation Challenges	24
6.2.1	<i>Custom Soft Body Physics.....</i>	<i>24</i>
6.2.2	<i>Realistic Texture Mapping.....</i>	<i>24</i>
6.2.3	<i>Visual Clarity.....</i>	<i>24</i>
6.2.4	<i>Performance Optimization.....</i>	<i>24</i>
6.2.5	<i>Collision Detection.....</i>	<i>24</i>
6.2.6	<i>Dimensional Representation:.....</i>	<i>25</i>
6.2.7	<i>Haptic Device Connection:.....</i>	<i>25</i>
6.3	Bridging 2D and 3D Models	25
6.4	Summary.....	25
Chapter 7. 2D MODELING		26
7.1	Overview of the 2D Simulation Model.....	26
7.2	Advantages of the 2D Model	26
7.3	Limitations of the 2D Model	27
7.4	Challenges Encountered.....	27
7.5	Simple 2D Model.....	28
7.6	Textured 2D Model.....	30
7.7	Advanced 2D Model	32
Chapter 8. 3D Modeling.....		37
8.1	Overcoming the Limitations of 2D Simulation.....	37
8.2	Challenges in Transitioning to 3D	38
8.3	Early 3D Model Designs and Iterations.....	39
8.3.1	<i>Version 0: Basic Setup & Rendering</i>	<i>39</i>
8.3.2	<i>Version 1: 3D Lancet Tool & Texture Mapping.....</i>	<i>41</i>
8.3.3	<i>Version 2: Almost Complete</i>	<i>42</i>

Chapter 9. STUDYING THE HAPTIC MECHANISM.....	47
9.1 Introduction	47
9.2 Design.....	49
9.3 Kinematic Analysis	50
9.3.1 Forward Kinematics.....	50
9.4 Inverse Kinematics	52
9.5 Validation	55
9.6 Static Analysis.....	56
9.6.1 Static Equilibrium in Robotics.....	56
9.6.2 Formation of the Jacobian Matrix.....	56
9.6.3 Using Jacobian Matrix	56
9.6.4 Computation of Pantograph Jacobian Matrix.....	57
Chapter 10. HAPTIC CONNECTION.....	60
10.1 Architecture	60
10.1.1 Client Architecture.....	61
10.1.2 Server Architecture.....	62
10.1.3 Simulation's Role in Communication.....	63
10.1.4 Mechanism's Role in Communication.....	64
10.1.5 Data Flow Overview	65
10.2 Challenges and Optimizations	66
10.2.1 Latency and Timing.....	66
10.2.2 Synchronization.....	66
10.2.3 Performance.....	66
Chapter 11. VIRTUAL MECHANISM.....	67
11.1 Forward Kinematic Formulas.....	67
11.2 Client.....	68
11.3 Server.....	69
11.4 Window Interface	69
11.5 Testing The Connection	71
Chapter 12. FINAL PRODUCT & LAB TESTING	72
12.1 Key Features and Improvements	72

12.2	Results	76
12.3	Conclusion	77

Chapter 1. INTRODUCTION

The advancement of simulation technologies has revolutionized fields such as engineering and medical training by enabling precise modeling and interactive analysis of complex systems. This thesis focuses on developing a simulation framework for surgical operations, specifically skin melanoma removal, addressing the unique challenges posed by the intricate mechanical properties of human skin. By integrating a physics-based spring-mass-damper (SMD) model [KL20] with a robotic mechanism featuring haptic feedback, the system offers realistic deformation and cutting mechanics, creating a safe environment for surgical practice and valuable insights into tissue interactions.

Skin surgeries are particularly challenging due to the complex biomechanical properties of human skin. Unlike rigid materials, skin is a multi-layered, elastic structure that deforms, stretches, and separates during cutting. Accurately capturing these responses in a simulation requires an advanced physical model and an interactive system capable of replicating real-world surgical dynamics.

This thesis tackles these challenges by developing a comprehensive simulation framework that integrates a physics-based SMD model with a haptic feedback mechanism, allowing for tactile sensations akin to real surgical interactions. A robotic system connects the haptic device to the simulation, enabling precise control of a virtual cutting tool. This setup enhances realism by synchronizing deformation mechanics with detailed feedback, bridging the gap between virtual and physical surgical environments.

Developing this project involved overcoming significant hurdles, such as achieving realistic deformation and cutting effects, ensuring seamless communication between the simulation and the haptic device, and accurately visualizing subtle tissue responses. The complexity of simulating the multi-layered structure of human skin added considerable computational and modeling demands [ZM99]. Addressing these issues required innovative approaches, including advanced mesh deformation techniques, optimized

collision detection algorithms, and real-time synchronization of haptic feedback with visual and mechanical responses [DL95], [CS00], [BS02].

Beyond its technical contributions, this project offers a realistic, interactive platform for surgical training. It provides trainees with a risk-free environment to refine their skills and offers researchers a valuable tool for studying surgical dynamics, exploring new techniques, and evaluating the effects of interventions on human tissues.

This introductory chapter lays the foundation for the report, highlighting the significance of realistic skin simulations and the key challenges addressed. The subsequent chapters delve into the methodologies, technical solutions, results, and contributions of the thesis, concluding with a discussion on its potential impact on medical training, surgical education, and future research.

It is the author's hope that this thesis provides valuable insights into surgical simulation and its applications, inspiring further exploration and innovation in medical simulation technology to benefit both practitioners and researchers.

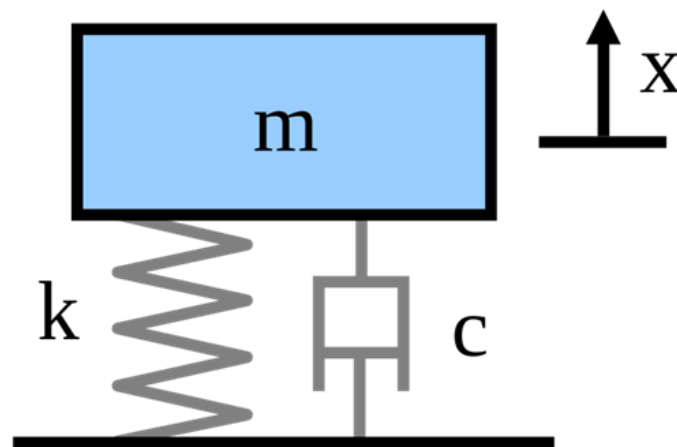


Figure 1: Spring-Mass-Damper Model

Chapter 2. BACKGROUND &

RELATED WORK

To understand the foundation and context of this project, it is essential to explore existing methods and technologies in soft tissue simulation, surgical training, and haptic-enabled systems. This chapter provides an overview of key advancements in these fields and highlights how this project builds upon and extends the current state of the art.

2.1 Soft Tissue Simulation

Simulating soft tissue involves replicating properties such as elasticity, anisotropy, and viscoelasticity. Particle-based models, particularly the spring-mass-damper (SMD) system, are widely adopted due to their computational efficiency in capturing these characteristics. Techniques like vertex splitting enable dynamic mesh separation during cutting, enhancing realism. While Finite Element Models (FEM) provide exceptional accuracy by calculating stress distribution and deformation at a granular level, their high computational demands make them unsuitable for real-time applications [KVG22].

In contrast, the SMD model approximates soft tissue as interconnected masses and springs, where springs represent the tissue's elastic and damping properties. This approach allows dynamic responses to deformation and cutting, maintaining computational performance suited for interactive and real-time simulations, such as surgical training systems [KL20], [KVG22].

Simulating layered structures, such as the epidermis, dermis, and hypodermis, is another critical aspect of soft tissue modeling. Each layer exhibits distinct mechanical properties and responds differently to external forces. Particle-based models can

accommodate these variations by assigning unique parameters (e.g., spring constants, damping coefficients) to each layer, accurately capturing composite tissue behavior [ZM99].

Recent advances, such as parallel processing and GPU acceleration, further enhance the performance of soft tissue simulations by enabling faster model updates without compromising detail or realism. By leveraging these developments, this project employs an SMD model to simulate the mechanical behavior of human skin, striking a balance between precision and computational efficiency for real-time surgical simulations.

2.2 Haptic Feedback in Medical Simulations

Haptic feedback systems have transformed surgical training by replicating tactile sensations and interactions in virtual environments. Devices such as robotic arms and handheld controllers recreate forces and textures experienced during procedures, enabling immersive, hands-on learning. Haptic feedback is especially valuable for teaching complex tasks like suturing, cutting, and excision, as it allows trainees to develop motor skills and refine techniques in a risk-free environment [CS00], [BS02].

To achieve realistic force rendering, haptic feedback must be computed and received at a minimum rate of 200 Hz [BS02]. This high update frequency ensures smooth, lag-free interactions, preventing instability and perceptible delays that could compromise precision. Lower frequencies can lead to incorrect force perception, reducing the effectiveness of training. A haptic mechanism typically consists of actuators, sensors, and control algorithms that process force interactions and translate them into tactile sensations. The system detects user input, computes the corresponding force response based on physical properties such as stiffness and damping, and applies feedback through actuators. By integrating a physics-based model with a robotic haptic interface, the system can simulate tissue resistance, elasticity, and cutting forces with high fidelity.

However, integrating haptic feedback with accurate soft tissue models presents significant challenges. Synchronizing the forces generated by haptic devices with the visual and mechanical responses of simulated tissue requires precise calibration. Dynamic interactions like deformation and cutting demand efficient computation to ensure alignment of tactile feedback with the virtual environment. Overcoming these challenges is critical for developing effective and realistic training systems, bridging the gap between virtual and physical surgical environments.

2.3 Cutting Mechanisms in Simulations

Simulating soft tissue cutting adds another layer of complexity to surgical simulations. Current methods often utilize vertex or edge splitting to simulate mechanical separation. These approaches must balance three key factors:

- **Visual Accuracy:** Ensuring that the cutting process appears realistic.
- **Computational Speed:** Maintaining real-time performance for seamless interaction.
- **Mechanical Realism:** Accurately replicating the forces and deformations involved in cutting.

Layered tissue simulation, such as human skin, introduces additional challenges. Each layer must deform and separate in a way that reflects its unique properties, requiring multi-layered cutting mechanics. Addressing these dynamics is essential to achieve the realism necessary for surgical training applications.

2.4 Project Contributions

This project advances surgical simulation by integrating a spring-mass-damper model with a haptic feedback mechanism that controls a virtual cutting tool in real time. Key contributions include:

- **Layered Tissue Dynamics:** Accurately simulates the mechanical behavior of the epidermis, dermis, and hypodermis during deformation and cutting, providing a comprehensive depiction of tissue responses.
- **Precise Tactile Feedback:** Delivering realistic sensations of force and texture, enhancing the simulation's fidelity.
- **Advanced Cutting Mechanics:** Implementing vertex splitting techniques with state management to simulate nuanced interactions between the cutting tool and layered tissues.

This framework captures intricate interactions between the cutting tool and soft tissue, offering a novel combination of realism and interactivity. By bridging critical gaps in current technology, this project establishes a robust platform for surgical training and procedural studies. By situating this work within the broader context of soft tissue modeling and haptic-enabled simulations, this chapter lays the groundwork for the methodologies and innovations presented in subsequent sections.

Chapter 3. PROBLEM STATEMENT

& OBJECTIVES

3.1 Problem Statement

Simulating human skin deformation and cutting (e.g. melanoma removal) presents significant challenges due to the multi-layered structure and complex mechanical properties of the skin. Accurately capturing dynamic responses, such as elasticity, tension and separation under the produced cutting forces, requires advanced modeling techniques.

The cutting process introduces additional complexity, involving dynamic and sometimes non-linear interactions between the virtual cutting tool and skin tissue. Accurately modeling separation mechanics, stress-induced deformations, and feedback forces are crucial for realism. To ensure effectiveness, the simulation must capture these dynamics in real time while maintaining computational efficiency.

Additionally, real-time haptic feedback integration demands precise synchronization between virtual and physical environments. This thesis addresses these challenges by developing a framework that combines realistic tissue modeling, interactive cutting mechanics, and efficient communication to integrate the simulation with the haptic robotic mechanism.

While haptic devices enable realistic tactile sensations, their integration poses challenges, including:

- Synchronizing physical and virtual movements.
- Maintaining interaction accuracy.
- Delivering tactile fidelity that closely mimics real-world sensations.

Visualizing subtle tissue deformations and cutting effects also requires high fidelity. The multi-layered nature of human skin—comprising the epidermis, dermis, and hypodermis demands the simulation account for distinct mechanical properties of each layer, further complicating the modeling process.

Another critical challenge is maintaining robust communication between the simulation and the robotic system controlling the haptic device. Efficient and precise data exchange is essential to synchronize user actions in the physical environment with the virtual environment and vice versa.

In summary, this project addresses these interconnected challenges by developing a simulation framework that integrates high-fidelity physics modeling, real-time cutting mechanics, intuitive user interaction, and seamless communication between hardware and software components.

3.2 Objectives

To overcome these challenges, the project aims to develop a physics-based simulation for skin surgery that integrates a virtual cutting tool controlled by a real-life haptic mechanism [OKB13]. The key objectives are as follows:

1. Realistic Skin Behavior

- Develop a spring-mass-damper model to replicate the mechanical properties of human skin, including deformation, tension, and cutting dynamics.

2. Layered Tissue Simulation

- Model the layered structure of skin, incorporating distinct responses for the epidermis and dermis to enhance realism during deformation and cutting.

3. Interactive Cutting Mechanics

- Implement robust vertex splitting and state transition mechanisms to simulate the cutting process with realistic visual and mechanical fidelity.

4. Haptic Feedback Integration

- Synchronize the simulation with the haptic mechanism to provide precise tactile feedback, enabling intuitive control of the cutting tool and realistic force sensations [HC04].

5. Real-Time Performance

- Optimize the simulation for real-time operation, ensuring smooth interaction between the cutting tool, virtual tissue, and haptic feedback without delays.

6. Lab-Ready Communication

- Establish robust socket-based communication between the simulation and the robotic system controlling the haptic device, ensuring seamless operation in a lab environment.

By achieving these objectives, this project aims to deliver an innovative solution for surgical training, bridging the gap between theoretical modeling and real-world applications.

3.3 Initial Challenges and Development Decisions

Developing a realistic simulation for skin surgery presented several challenges, particularly in achieving accurate soft-body dynamics and real-time interaction. Early evaluations of existing simulation environments revealed key limitations, including:

- **Soft-Body Physics Constraints:** Many frameworks either lacked support for realistic soft-body physics or were restricted to predefined animations, making them unsuitable for dynamic interactions.
- **Haptic Integration Difficulties:** Ensuring seamless communication between the simulation and the real-life haptic mechanism proved challenging due to limitations in real-time feedback and system flexibility.
- **Computational Performance:** The need for high-fidelity deformation and cutting mechanics introduced significant processing demands, requiring careful consideration of optimization strategies.

These challenges shaped the development approach and influenced critical design decisions. A more detailed discussion of the methodologies and solutions adopted to address these issues is presented in the following chapters.

Chapter 4. METHODOLOGY

The simulation employs a spring-mass-damper model, with springs configured in vertical, horizontal, and diagonal orientations to replicate tissue behavior under produced cutting forces. Vertical springs simulate compression during penetration, while horizontal springs model lateral resistance. Diagonal springs enhance stability and multi-directional deformation. Cutting is achieved through vertex splitting, where springs rupture upon exceeding force thresholds, creating realistic tissue separation and retraction. The system is optimized for real-time performance using parallel processing and caching techniques to ensure smooth interaction and accurate feedback.

4.1 General Overview of the Mass-Spring-Damper (MSD) Model

The Spring-Mass-Damper (SMD) Model is a fundamental mechanical framework used to study the dynamics of systems exhibiting oscillatory or damped behavior. The model comprises three main components: mass (inertia), spring (elasticity), and damper (resistance). This combination allows for simplified yet robust analyses of system responses under various forces, making the SMD model highly applicable in engineering, biomedical simulation, and control systems [KL20], [AH09].

1. **Mass (m):** Represents the object's inertia, resisting changes in motion according to Newton's Second Law: $F = m \times a$
2. **Spring (k):** Models the restoring force, proportional to displacement from equilibrium. Governed by Hooke's Law: $F_{spring} = -k \times x$ where x is displacement.
3. **Damper (b):** Models energy dissipation (e.g., friction or air resistance), with force proportional to velocity: $F_{damper} = -b \times u$

4.1.1 Equation of Motion

The dynamics of an SMD system follow a second-order differential equation derived

from Newton's laws: $m * \frac{d^2x}{dt^2} + b * \frac{dx}{dt} + k * x = F(t)$

where:

- $\frac{d^2x}{dt^2}$: Acceleration,
- $\frac{dx}{dt}$: Velocity,
- $F(t)$: The sum of the applied forces

4.1.2 System Behavior Analysis

1. **Undamped System (b=0)**: Exhibits simple harmonic motion with a natural frequency:

$$\omega_n = \sqrt{\frac{k}{m}}$$

2. **Damped System (b>0)**: Displays varying responses depending on the damping ratio ζ , defined as:

$$\zeta = \frac{b}{2\sqrt{mk}}$$

- **Underdamped ($\zeta < 1$)**: Oscillates with decreasing amplitude.
- **Critically Damped ($\zeta = 1$)**: Returns to equilibrium in the shortest time without oscillating.
- **Overdamped ($\zeta > 1$)**: Returns to equilibrium without oscillation, but more slowly.

4.1.3 SMD Model Applications

The SMD model is applied in diverse fields, including:

- **Mechanical Engineering**: Vehicle suspension systems for shock absorption.
- **Civil Engineering**: Building response modeling to seismic activity.
- **Control Systems**: Tuning responses for stability.
- **Vibration Analysis**: Predicting and mitigating oscillations in machinery.

4.2 Simulation Model for Skin Surgery

4.2.1 General Model Information

The surgical simulation replicates skin surgery (e.g. for melanoma removal) using a five-bar linkage mechanism called the "Pantograph," which provides precise control over the cutting tool and delivers haptic feedback. This feedback enhances the simulation by creating a realistic and immersive training experience.

Human skin is modeled as a 2D & respectively 3D spring-mass-damper (SMD) mesh, where mass points are interconnected by horizontal, vertical, and diagonal springs. This configuration replicates the mechanical properties of skin, allowing the model to deform under external forces while maintaining structural integrity until the tissue is severed.

4.2.2 Purpose of A Surgical Simulation

Surgical simulations play a critical role in modern medical training by providing a safe and controlled environment for practice and skill development. Unlike real-life surgical scenarios, simulations allow trainees to repeat procedures as often as needed, free from the pressures of working on living patients or animals. This not only improves their technical proficiency but also enhances overall patient safety in real operations.

The simulation offers objective and measurable ways to evaluate a trainee's performance, including:

- **Motion Analysis:** Tracks precision, speed, and dexterity during simulated surgical procedures, providing insights into the trainee's motor skills.
- **Task-Based Analysis:** Breaks down complex procedures into smaller, measurable steps, allowing for focused assessment of specific skills such as cutting, suturing, or incision-making.
- **VR Metrics:** Provides quantitative feedback, such as task completion times, error rates, and accuracy, which can be used to compare progress across sessions or benchmark against expert performance.

By integrating these evaluation methods, the simulation ensures trainees receive comprehensive feedback, fostering continuous improvement in surgical technique and decision-making.

4.2.3 Skin Structure and Mechanical Properties

Human skin is a complex, multi-layered structure with unique mechanical properties that must be accurately modeled for realistic surgical simulations. The skin comprises three main layers:

1. **Epidermis:** The outermost layer, acting as a protective barrier against environmental damage.
2. **Dermis:** The middle layer, composed of collagen and elastin fibers, giving skin its strength, elasticity, and flexibility.
3. **Hypodermis:** The deepest layer, consisting of subcutaneous fat and connective tissue, providing structural support and cushioning.

Each layer contributes distinct mechanical properties to the overall behavior of skin.

Key mechanical characteristics of skin include:

- **Nonlinear Behavior:** Skin exhibits highly nonlinear responses under large deformations, which are often modeled using hyperelastic equations to capture its behavior accurately.
- **Anisotropy:** Skin's response to stress varies depending on the direction of the applied force due to the alignment of collagen fibers along Langer's lines.
- **Viscoelasticity:** Skin demonstrates both elastic and viscous properties, meaning it can recover its shape after deformation but also exhibits time-dependent behavior under sustained forces.

These mechanical properties are integral to achieving a realistic simulation, as they influence how the skin deforms and reacts during surgical procedures like melanoma removal.

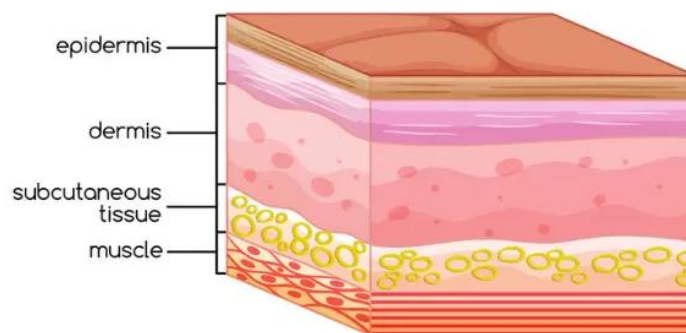


Figure 2: Human Skin Tissue Anatomy

4.2.4 Integration of Forces Using Euler's Method with Multi-Directional Springs

The Spring-Mass-Damper (SMD) model used in the simulation incorporates a network of horizontal, vertical, and diagonal springs to replicate the tension, deformation, and retraction characteristics of human skin. Each spring orientation contributes uniquely to the overall force distribution within the mesh, ensuring the model can handle complex deformations resulting from surgical interactions.

A key feature of the SMD model is the pre-stretched state of most of its springs (horizontal and diagonal, not vertical). This pre-tensioning creates baseline stress within the mesh, closely mimicking the inherent tension in real human skin. When a spring is severed during the simulated cutting process, the remaining interconnected springs redistribute the forces, causing the adjacent mass points to retract. This behavior replicates the retraction of tissue observed in real surgical scenarios, adding a layer of mechanical realism to the simulation.

The simulation employs Euler's Method, a numerical technique for solving ordinary differential equations, to compute both the dynamic responses of the mesh and the user-perceived forces transmitted through haptic feedback [AH09].

For mesh dynamics, Euler's Method iteratively updates the positions and velocities of mass points by integrating the forces exerted by springs and dampers over discrete time steps. This enables a real-time simulation of tissue deformation and retraction during cutting, allowing for interactive and visually accurate surgical procedures.

For user-perceived forces, the method accounts for the user's interaction with the mass points and the resulting deformation of the supporting spring-dampers. When the user stretches, presses, or manipulates the virtual tissue, one or more spring-dampers are displaced, generating forces proportional to the deformation direction and magnitude. These forces are then relayed to the user through haptic feedback, enhancing the realism of the simulation.

By combining the pre-stretched SMD model with advanced numerical integration, the simulation effectively replicates the complex mechanical behavior of human skin. This provides users with a realistic and immersive training environment where they can observe and feel the effects of their surgical actions in real time.

4.2.5 Euler's Simple Method for Visual Effect Force Calculation

To achieve real-time updates, we apply Euler's integration method to iteratively calculate forces, velocities, and positions of each mass point in the mesh. This integration method allows efficient approximation of displacements at each time step.

1. Equations of Motion:

The system is governed by the equation:

$$m * \frac{d^2x}{dt^2} + b * \frac{dx}{dt} + k * x = F(t)$$

2. Position and Velocity Update:

Using Euler's method:

$$x(t + \Delta t) = x(t) + u(t) * \Delta t$$

$$u(t + \Delta t) = u(t) + \frac{F(t)}{m} * \Delta t$$

Where:

F(t): is the sum of forces from vertical, horizontal, and diagonal spring-dampers,

Δt: is the discrete time step at which F(t) is recalculated.

As will be further examined in the subsequent chapters on project implementation, the forces contributing to the visual effects differ from those perceived by the user, though they are not entirely independent as they are produced under the same causes. The distinction arises because the visual effect forces are computed at a different rate than the physically simulated forces, ensuring both computational efficiency and a smooth visual experience.

Since visual effects are directly tied to frame rendering, their update rate is synchronized with the frame generation process, whereas other calculations may follow a shorter time step (Δt) to maintain numerical stability and physical accuracy. This decoupling allows for a balance between computational efficiency and visually satisfying results, preventing unnecessary computational load while preserving the fidelity of the simulation.

4.3 User-perceived Force Calculations

The cutting tool's speed and angle impact the forces in each direction, while the presence of vertical, horizontal, and diagonal springs ensures comprehensive coverage of force response in the mesh. Each spring type contributes to force calculations based on its orientation, affecting the displacement behavior when a spring of that type breaks.

4.3.1 Vertical Force F_y – Initial Compression and Penetration

Vertical spring-dampers primarily resist the penetrative motion of the cutting tool. The cumulative vertical force is given by:

$$F_{spring,y} = \sum_i k_{vertical,i} \times \Delta y_i$$

$$F_{damper,y} = \sum_i b_{vertical,i} \times u_{y,i}$$

where:

- Δy_i : is the total displacement, combining initial pre-stretch and tool displacement,
- $u_{y,i}$: is the mass's vertical speed component.

4.3.2 Horizontal Force F_x – Cutting and Horizontal Drag

Horizontal spring-dampers are responsible for resisting lateral (horizontal) tool movements. When moving the tool horizontally, the combined force from horizontal springs is calculated as:

$$F_{spring,x} = \sum_j k_{horizontal,i} \times \Delta x_j$$

$$F_{damper,x} = \sum_i b_{horizontal,j} \times u_{x,j}$$

where:

- Δx_j includes the initial stretch plus the tool's horizontal displacement,
- $u_{x,j}$ is the mass's horizontal speed component.

4.3.3 Diagonal Springs – Diagonal Force Components and Mixed Displacement

Diagonal spring-dampers provide additional structural support across the mesh, contributing to both vertical and horizontal forces depending on the direction of force application. When diagonal springs are engaged or ruptured, they affect mass displacements in both directions:

$$F_{spring,diag} = \sum_k k_{diagonal,k} \times \Delta d_k$$
$$F_{damper,diag} = \sum_k b_{diagonal,k} \times u_{diag,k}$$

where:

- Δd_k : is the combined displacement along the diagonal,
- $u_{diag,k}$: is the mass's diagonal velocity component.

4.3.4 Breaking Point & Directional Mass Displacement

Each type of spring-dampers (vertical, horizontal, or diagonal) has a maximum allowable displacement beyond which it ruptures. When a spring damper of any type breaks:

- **Vertical spring-dampers:** Breaking a vertical spring-damper causes a sudden drop in vertical resistance, with neighboring spring-dampers pulling the point masses in the remaining directions (horizontal and diagonal).
- **Horizontal spring-dampers:** Upon breaking, horizontal spring-dampers allow lateral movement, with vertical and diagonal spring-dampers pulling the mass points back toward the mesh.
- **Diagonal spring-dampers:** Breaking a diagonal spring-dampers leads to complex displacements in both vertical and horizontal directions, as the remaining spring-dampers redistribute forces along their respective orientations.

The combined effect of these multi-directional spring-dampers, especially under pre-stretched conditions, creates a directional pull on affected masses, simulating realistic tissue retraction. The produced force values range from 0.5N to 1.2N for penetration and 0.8N to 1.2N for lateral cutting.

4.3.5 Total Force Calculation with Angle of Approach

The total force exerted on the mesh combines contributions from vertical, horizontal, and diagonal spring-dampers, as well as the angle θ of the cutting tool. Additionally, frictional forces ($F_{friction}$) are included to account for material or skin texture properties. The complete force calculation is given by:

$$F_{total} = F_y \cos(\theta) + F_x \sin(\theta) + F_{diag} \cos(\theta + 15^\circ) + F_{friction}$$

where:

- F_y : Vertical force component from vertical springs,
- F_x : Horizontal force component from horizontal springs,
- F_{diag} : Diagonal force component from diagonal springs,
- $F_{friction}$: Frictional force, dependent on material/skin texture properties.

The frictional force $F_{friction}$ represents the resistance felt by the cutting tool due to the surface texture and material properties of the skin model. This component varies with the normal force and is influenced by the coefficient of friction, which is calibrated based on the tissue's realistic surface properties [ZM99].

The μ friction coefficient values range from 0.12 to 0.62.

The k spring values range from 10 to 20 N/m.

The b damper values range from 3 to 5 Ns/m.

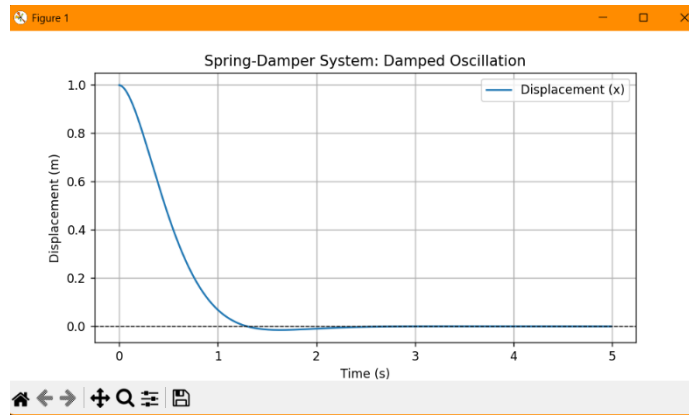


Figure 3: SMD model in time

This angle-adjusted calculation enables the model to dynamically capture the deformation and directional responses of the mesh. As the cutting tool moves and interacts with springs oriented in different directions, the resulting forces guide the motion and rupture effects, producing a realistic simulation of tissue cutting and retraction.

4.3.6 Nonlinear Force Formulation with Multi-Directional Spring-Dampers

To further enhance realism, our model incorporates nonlinear force formulation to capture the complex, realistic responses of biological tissue. This includes:

Elastic force variation: The spring-damper forces grow non-linearly with displacement, mimicking the natural, nonlinear elasticity of soft tissue. The elastic force is modeled as:

$$F_{elastic} = k * (x - x_0) + a * (x - x_0)^2$$

Here, k represents the linear spring constant, and a accounts for the nonlinear stiffness of the tissue, x is the current displacement, and x_0 is the rest length of the spring. This formulation ensures that the elastic response becomes increasingly nonlinear as the displacement grows, accurately replicating the behavior of soft biological tissues.

Damping force adaptation: Damping forces adjust based on the rate of deformation, allowing the model to simulate viscous tissue resistance that varies with speed. The damping force is expressed as:

$$F_{damping} = -c * \frac{dx}{dt} * \left(1 + j * \left|\frac{dx}{dt}\right|\right)$$

In this equation, c is the linear damping coefficient, and j adjusts the nonlinear damping effect based on the velocity $\frac{dx}{dt}$ of deformation. This allows the model to account for the increasing resistance of viscoelastic tissues at higher deformation speeds.

Since typical user interactions are unlikely to induce large deformations on the skin mesh, the visual effects of these nonlinear formulations may not always be prominent. However, these enhancements contribute to the underlying dynamics, ensuring that even small deformations respond in a biomechanically realistic manner.

By integrating Euler's method with these multi-directional, pre-stretched springs, the model simulates the forces and movements of the mesh in real time. The equations of motion are updated iteratively as follows:

$$x(t + \Delta t) = x(t) + \frac{dx}{dt} * \Delta t \quad \frac{dx}{dt} * (t + \Delta t) = \frac{dx}{dt} + \frac{F_{elastic} + F_{damping}}{m} * \Delta t$$

Here, m is the mass of the point, and Δt is the simulation time step. These updates ensure smooth, real-time deformation and cutting effects, achieving realistic deformation while maintaining the computational efficiency required for real-time surgical simulations.

Chapter 5. DEVELOPMENT

PLATFORM

5.1 General

The development of this simulation environment requires detailed exploration of methodologies and deliberate decision-making to meet the demanding requirements of a realistic melanoma removal simulation. Achieving high levels of precision, responsiveness, and interactivity demands innovative solutions to challenges such as modeling real-time soft-body dynamics, integrating haptic feedback, and simulating realistic cutting mechanics.

A key objective is to create a tailored simulation framework capable of mimicking the intricate mechanical behavior of human skin under surgical conditions. This involves accurately modeling layered tissue structures, simulating deformation and cutting in real time, and synchronizing tactile feedback with visual and mechanical responses. While existing platforms offer a foundation for simulation and rendering, they lack the flexibility to meet these specific demands.

This chapter presents a comprehensive account of the development process, beginning with the exploration of existing platforms and the limitations that led to their rejection. The decision to adopt a low-level programming approach was driven by the need for precise control over physics, rendering, and real-time performance. By leveraging OpenGL for rendering and building a custom physics engine, this project successfully creates a scalable, interactive, and high-fidelity simulation framework for surgical training.

5.2 Initial Exploration of Platforms

During the early stages of the project, simulation platforms such as Unity, ROS-Gazebo, and Blender were assessed for their potential to construct and simulate the skin surgery model. These platforms have proven effective in previous undergraduate projects involving graphical setups, 3D modeling, and robotic manipulation, highlighting their capabilities in rendering, simulation, and hardware integration.

Despite their strengths, these platforms were found unsuitable for the specific needs of this project. None of them provides adequate support for real-time soft-body dynamics, which are critical for accurately modeling the mechanical behavior of human skin. Additionally, their handling of cutting mechanics is limited to predefined animations, making them incapable of simulating dynamic and interactive cutting processes. Integrating a haptic feedback mechanism also presents challenges, as these platforms are not designed to handle the complexities of real-time interaction between physical and virtual environments.

1. Unity:

- **Introduction:** The author was first introduced during “Computer Graphics” mandatory course in the 7th academic semester. The objective of the project was to design our solar system using objects and textures and add to them movement and collision.
- **Primary programming language: C#**
- **Strengths:** Robust graphics capabilities, extensive libraries for 3D rendering, and support for physics-based interaction and force computation.
- **Limitations:** Can not handle soft-body dynamics in real time, with support limited to hardcoded animations that lack the flexibility needed for dynamic cutting and deformation.



Figure 4: Unity Platform

2. ROS-Gazebo:

- **Introduction:** The author was first introduced to this platform in the “Robotics” mandatory optional course during the 10th academic semester. The objective of the offered project was to calculate and simulate the trajectory of 2 virtual robots in space. ROS helped program the movement of those robots and gazebo to visualize their movement.
- **Strengths:** A powerful tool for robotics and mechanical simulation, offering integration with hardware components.
- **Limitations:** Designed primarily for rigid-body dynamics, with no built-in support for simulating soft-tissue behavior or cutting mechanics.

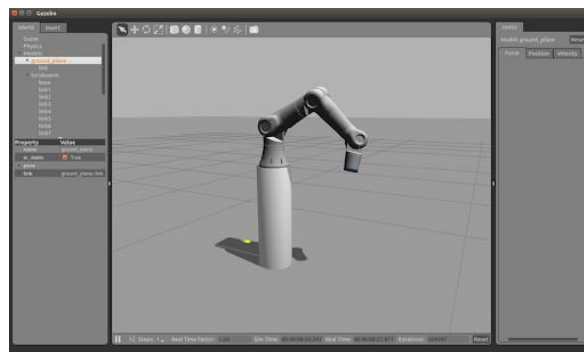


Figure 5: Ros-Gazebo Platform

3. Blender:

- **Introduction:** The author was first introduced to that tool while searching for a suitable platform to develop this project.
- **Strengths:** Excellent for creating visually detailed models and animations.
- **Limitations:** Geared toward pre-rendered animations rather than interactive simulations, with no support for real-time soft-body physics or integration with haptic feedback systems.

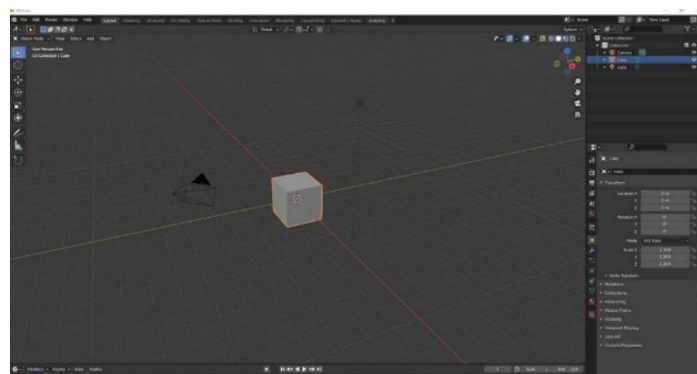


Figure 6: Blender Platform

5.3 Challenges with Soft-Body Dynamics and Real-Time Interaction

The primary challenges encountered included:

- **Soft-Tissue Simulation:** Existing known platforms either did not support soft-body physics or restricted deformation behavior to static, pre-defined animations.
- **Dynamic Cutting Mechanics:** Simulating realistic cutting required vertex splitting, mesh and texture manipulation, which these platforms could not perform dynamically.
- **Integration with Haptic Devices:** Ensuring real-time feedback and synchronization between the simulation and the haptic mechanism was challenging due to platform limitations.

5.4 Transition to Low-Level Programming

To overcome the challenges presented above and the limitations of the already explored platforms, a decision was made to develop a custom simulation environment using low-level C programming and OpenGL. This approach provided full control over the physics model, rendering, and haptic integration.

1. Advantages of OpenGL:

- Flexibility in rendering custom graphical elements for the simulation.
- Direct control over the mesh structure and real-time updates.
- Ability to optimize performance for real-time interaction.

2. Custom Physics Engine:

- Developed to handle the complex dynamics of a spring-mass-damper system.
- Integrated advanced techniques such as vertex splitting and state transitions for realistic cutting.

3. Haptic Integration:

- A custom socket-based communication system was implemented to synchronize the haptic device with the simulation, providing real-time tactile feedback.

Chapter 6. IMPLEMENTATION

6.1 Installation and Setup of OpenGL in WSL Environment

The initial step in the development process involved setting up the OpenGL environment in the Windows Subsystem for Linux (WSL). This decision leveraged WSL's capability to provide a Linux-like environment while utilizing the GPU for hardware acceleration. The following steps outline the installation and configuration process:

1. **WSL Installation:**

- Ensuring WSL was installed and updated to version 2, providing the necessary support for GPU-accelerated applications.

2. **Graphics Driver Configuration:**

- Installing the latest NVIDIA graphics drivers to enable GPU passthrough to WSL, allowing OpenGL applications to utilize the hardware effectively.

3. **Installing OpenGL Libraries:**

- Using package managers like apt to install OpenGL libraries (mesa-utils and libgl1-mesa-dev) and GLFW for window management.

4. **Development Tools Setup:**

- Installing development tools, including g++ for compiling code, make for build automation, and editors like vim or VS Code for coding.

5. **Testing the Environment:**

- Running sample OpenGL programs to verify that rendering and GPU acceleration were functioning correctly.

This setup ensured a robust and flexible development environment, ready for the implementation of the custom simulation models [BBC+99].

6.2 Implementation Challenges

The implementation of the 2D and 3D simulation models in OpenGL presented a range of technical and design challenges, each of which required innovative solutions to achieve the project's objectives.

6.2.1 Custom Soft Body Physics

Simulating the mechanical behavior of human skin demanded a custom soft-body physics engine capable of replicating elastic and viscoelastic properties. The model needed to balance accuracy with computational efficiency, ensuring realistic deformation and response under cutting and tension forces.

6.2.2 Realistic Texture Mapping

Applying textures to the soft-body mesh was a critical challenge. The textures needed to dynamically follow the deformation of the mesh while maintaining their visual integrity. This required precise UV mapping and texture coordination, ensuring the rendered skin appeared realistic even during complex interactions.

6.2.3 Visual Clarity

Ensuring that the cutting process was visually clear posed additional challenges. Subtle deformations, combined with the dynamic behavior of the mesh and its texture, required continuous adjustments to camera angles, lighting, and rendering settings. This was particularly important to provide the user with an intuitive and immersive view of the simulated surgical environment.

6.2.4 Performance Optimization

Real-time simulation imposed strict performance constraints. Achieving a balance between realism and computational efficiency necessitated optimization of the physics model, rendering pipeline, and collision detection algorithms. This included leveraging GPU acceleration and caching techniques to reduce computation times.

6.2.5 Collision Detection

Interactions between the cutting tool and the skin mesh demanded efficient collision detection algorithms. These algorithms needed to identify the closest vertex or element to the cutting tool, calculate the force exerted, and update the mesh state in real time to simulate realistic cutting effects.

6.2.6 Dimensional Representation:

Developing both 2D and 3D models presented unique challenges. The 2D simulation offered simplicity and a game-like interface for experimentation, while the 3D model provided a more detailed and realistic representation of skin behavior. Designing for both dimensions required careful consideration of the user experience and the technical demands of each approach.

6.2.7 Haptic Device Connection:

Establishing a reliable two-way connection between the simulation and the haptic mechanism was essential for providing a realistic surgical experience. The simulation needed to send force data to the haptic device while simultaneously receiving position updates in real time. Achieving this level of synchronization requires robust socket-based communication and efficient data handling.

6.3 Bridging 2D and 3D Models

The implementation process also involved transitioning between the 2D and 3D simulation models. While the 2D model provided a simplified environment for testing basic mechanics, the 3D model required additional complexities, such as multi-layered meshes, volumetric textures, and depth-based collision handling. Lessons learned from the 2D implementation informed the development of the 3D model, ensuring a smoother transition and minimizing redundant efforts.

6.4 Summary

The implementation of the 2D and 3D simulation models in OpenGL required overcoming significant technical challenges, from setting up the development environment in WSL to addressing issues related to physics modeling, texture mapping, and real-time interaction. These efforts established the foundation for a robust simulation environment capable of providing an immersive and realistic surgical training platform. By bridging the gap between 2D experimentation and 3D realism, the project successfully achieved a balance between simplicity, precision, and interactivity. In the following chapters the modeling of both projects is described.

Chapter 7. 2D MODELING

7.1 Overview of the 2D Simulation Model

The 2D simulation model serves as the initial framework for replicating the behavior of human skin during surgical procedures (e.g. melanoma removal). This model uses a spring-mass-damper system represented by a dynamic 2D mesh, where mass points are interconnected by springs and dampers. The simulation integrates the mechanics of deformation, tension, and cutting, providing a simplified yet functional platform for surgical interaction. The cutting tool, controlled by a haptic mechanism, interacts with this mesh, enabling real-time feedback and visualization of the cutting process.

7.2 Advantages of the 2D Model

1. Computational Efficiency:

- The 2D model requires fewer calculations compared to a 3D counterpart, allowing for real-time simulation even on standard hardware.

2. Simplified Visualization:

- It is easier to visualize and debug the interactions between the cutting tool and the mesh, making it suitable for testing core mechanics and physics.

3. Faster Development Cycle:

- Reducing complexity accelerates the prototyping and iteration process, allowing rapid testing of cutting dynamics and mesh deformation.

4. Educational Value:

- Provides a straightforward environment for demonstrating the fundamental principles of tissue deformation and cutting.

7.3 Limitations of the 2D Model

1. Lack of Depth and Realism:

- The 2D simulation cannot capture volumetric deformations, limiting its ability to replicate the full complexity of human skin.

2. Simplified Cutting Dynamics:

- Cutting in 2D does not account for depth variations, making it less suitable for simulating layered tissue structures.

3. Limited Haptic Feedback:

- Interaction in a 2D plane restricts the range of forces and tactile sensations that can be conveyed through the haptic device.

7.4 Challenges Encountered

1. Mesh Deformation Management:

- Ensuring that the 2D mesh deforms realistically under various forces while maintaining stability posed a significant challenge.

2. Real-Time Performance:

- Balancing the need for accurate deformation with the requirement for real-time interaction required careful optimization of the physics calculations.

3. Cutting Implementation:

- Simulating vertex splitting and tissue retraction in a way that feels natural and responsive to user input involved applying an additional fine-tuning to the cutting algorithms.

4. Haptic Integration:

- Providing meaningful feedback through the haptic device in a 2D context requires mapping forces accurately to the user's movements.

7.5 Simple 2D Model

The initial versions of this model feature a simple quad grid composed of point masses, visually represented as red dots, interconnected by spring-dampers rendered as white lines. This minimalistic mesh design served as a foundational framework for adjusting and fine-tuning the physical properties of the simulation, such as elasticity, damping, and tension.

The straightforward visual representation was intentionally chosen to focus on the core mechanics of the system without the complexity of detailed textures or advanced rendering. By using this simple grid structure, it is possible to iteratively test and calibrate the spring constants (k) and damping coefficients (b) to achieve the desired mechanical behavior of the mesh. This approach provides a controlled environment for experimenting with deformation and retraction dynamics, ensuring the system behaves as intended before progressing to more complex implementations.

This early-stage model played a crucial role in establishing the physics foundation of the simulation, enabling the creation of realistic and responsive interactions that would later be incorporated into more advanced versions with enhanced visual fidelity and functionality.

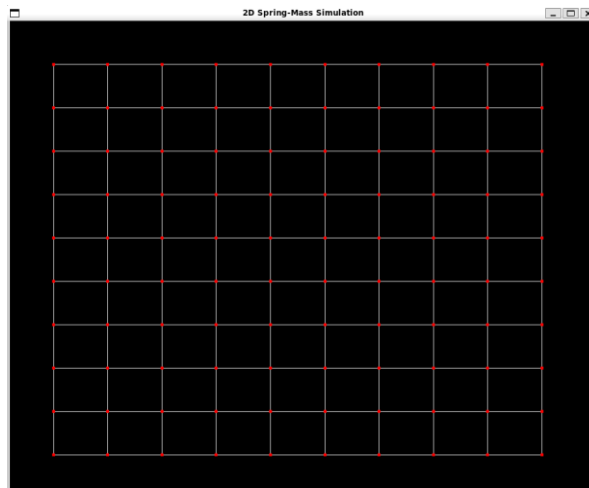


Figure 7: Simple 2D SMD Simulation

This approach provides an intuitive way to adjust the k values of the springs and the b values of the dampers, ensuring that the masses moved as intended. This behavior allows for the creation of an opening in the mesh whenever a spring-damper element is removed.

When a spring-damper is removed, the connected masses are influenced by the remaining spring-dampers in the mesh. The forces produced cause the affected masses to retract in opposite directions, as they are pulled by the other spring-dampers still in place. This process generates velocities and pulling forces, which persist until the masses settle into a new equilibrium position over time. The resulting dynamic movement accurately simulates the retraction of tissue in response to cutting, providing an important foundation for modeling realistic wound behavior.

Initially, the simulation employs the mouse cursor as the cutting tool. By positioning the cursor near a spring-damper (represented as a white line in the mesh) and clicking the left mouse button, the user can "cut"/remove the element. The affected point masses are then displaced over time using Euler's method (as described in the chapter of Methodology), a straightforward numerical integration technique.

This action creates a gap in the mesh, representing a simulated wound. This functionality is a critical step in enhancing the simulation's realism, as the gap serves as the basis for modeling realistic wound behavior in subsequent, more realistic iterations of the simulation. By incorporating this mechanism, the simulation effectively mimics the physical separation and deformation of soft tissue, paving the way for more complex and lifelike interactions in future developments.

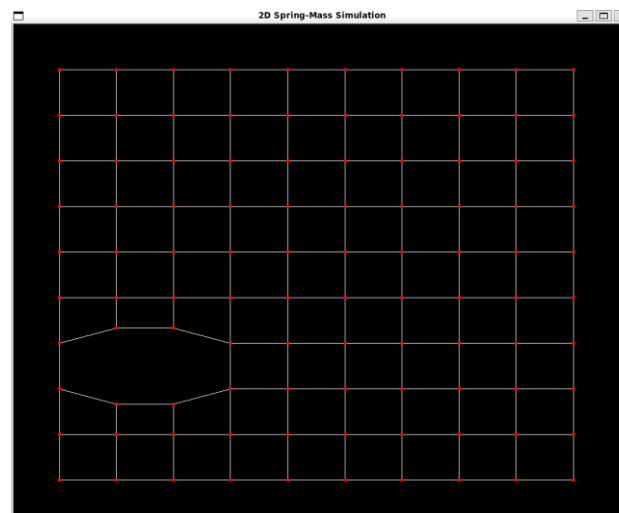


Figure 8: Simple 2D SMD Simulation with Removed Spring-Damper

With the foundational physics and mechanics of the model established, we are now ready to advance to the next level by incorporating new features. These include the addition of realistic textures and enhanced visual effects, which will significantly improve the model's appearance and realism. By integrating these elements, the simulation will more closely replicate the visual and mechanical properties of real human tissue, creating a more immersive and accurate experience for users.

7.6 Textured 2D Model

Building upon the previous progress, the next step is to enhance the 2D model to make it more closely resemble a real layer of skin tissue. This requires a series of improvements aimed at both visual accuracy and mechanical realism. Specifically, we need to:

1. **Load a Real Texture:** Apply a high-resolution image as the texture for the mesh to provide a visually accurate representation of human skin tissue.
2. **Determine Texture Coordinates:** Calculate effective texture coordinates to ensure the texture aligns correctly with the mesh, even as it deforms.
3. **Add a Melanoma Representation:** Incorporate a black, melanoma-like shape at the center of the mesh to simulate the target area (for surgical removal purpose).
4. **Simulate Splitting Effects:** Achieve a realistic splitting effect by deforming both the spring-mass-damper (SMD) mesh and the applied texture during cutting interactions.

To implement these features, we utilize OpenGL's SOIL (Simple OpenGL Image Library) for texture loading. After creating the mesh as before, we calculate the corresponding texture (UV) coordinates for each vertex and load the desired texture. This ensures that the texture seamlessly adheres to the mesh, even during deformation.

The result is a visually enhanced 2D model that features a realistic texture overlaid on the SMD mesh, accurately mimicking the appearance of human skin tissue. This advancement marks a significant step toward achieving a more immersive and lifelike simulation, paving the way for further enhancements in texture dynamics and interaction realism.

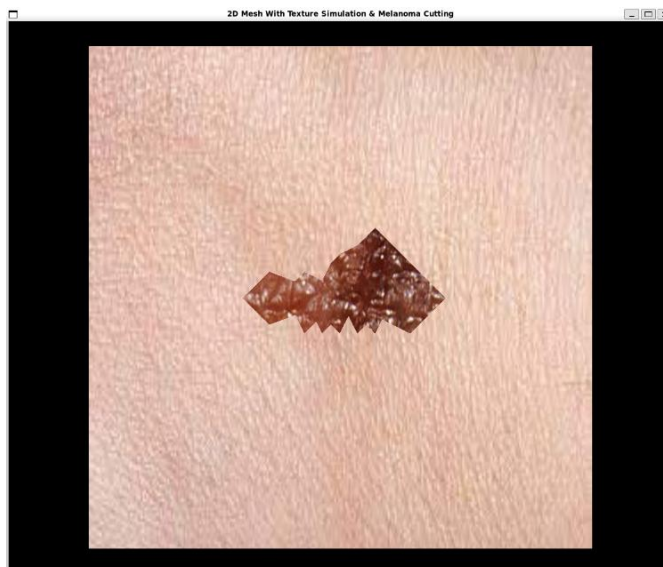


Figure 9: 2D MSD Simulation with Texture

Once again, the mouse cursor is utilized as the cutting tool within the simulation. By clicking and holding the left mouse button, the spring-dampers closest to the cursor are sequentially removed, effectively splitting the mesh. This interaction allows the user to simulate the cutting process dynamically, creating gaps in the mesh that mimic the separation of soft tissue.

The initial result of this process is as follows:

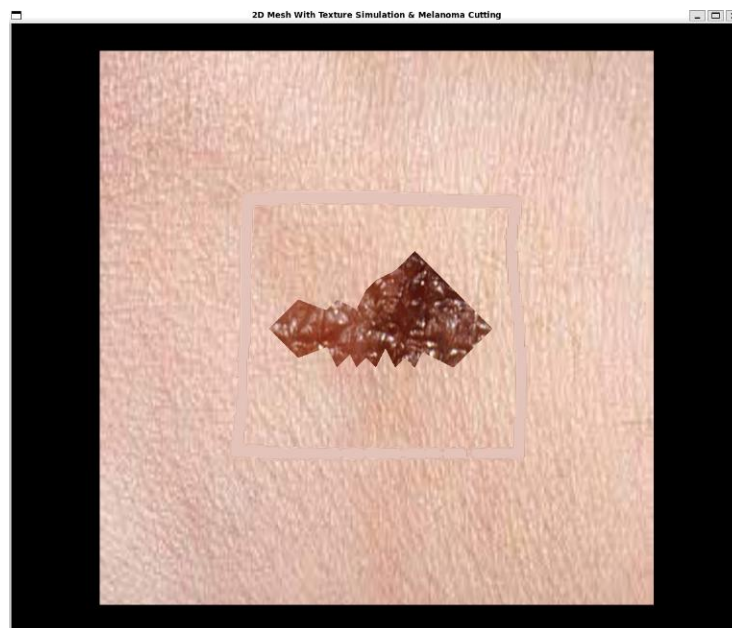


Figure 10: Result of 2D Mesh Cut Procedure

As a first attempt to simulate a realistic cut to human skin, the results are promising but far from perfect. By closely examining the deformed textured mesh, several shortcomings become apparent:

- 1. Lack of Visual Realism:**

The current skin texture lacks sufficient quality, and the melanoma representation appears artificial, with an unrealistic shape that detracts from the overall visual fidelity.

- 2. Inadequate Tissue Behavior During Cutting:**

The mesh fails to exhibit realistic opening behavior when the split effect is triggered. Instead of simulating a genuine cut, the interaction resembles a superficial graze, as though the tool merely draws on the surface of the mesh rather than cutting through it.

3. **Absence of Force Feedback:**

The model does not display any forces acting on or generated by the cutting tool, which is a critical component for achieving realism. This omission significantly reduces the simulation's ability to mimic real-world surgical interactions and haptic dynamics.

Given these limitations, the model requires further enhancement to achieve a higher degree of realism and interactivity. Addressing these issues will involve improving texture quality, refining cutting mechanics to simulate genuine tissue separation, and incorporating force feedback to provide a more immersive and realistic user experience. In the next chapter, we will explore potential solutions to these challenges and discuss strategies for improving the model, thereby advancing the simulation's fidelity and realism.

7.7 Advanced 2D Model

The advanced model introduces significant improvements, enabling it to more accurately represent human skin tissue. These enhancements include the following key attributes:

1. **High-Quality Texture:**

The model features a high-resolution skin texture with a clearly defined melanoma at its center, providing a more realistic and visually accurate representation of human tissue.

2. **Multi-Layered Representation:**

The addition of a second tissue layer, representing the dermis, enhances the model's realism. This layer becomes visible upon cutting through the epidermis (the first visible layer of the mesh), accurately mimicking the structure of human skin and the process of surgical exposure.

3. **Interactive Toolbar:**

A side toolbar has been added, featuring three buttons, each dedicated to a specific action. This interface improves user interaction, allowing for easy access to essential functionalities within the simulation.

4. Lancet-Like Tool:

The model includes a lancet tool that dynamically follows the mouse cursor, providing an intuitive interface for interacting with the mesh. This tool can actively engage with the tissue, simulating realistic cutting mechanics.

5. Force Calculation and Data Logging:

The simulation now calculates forces generated during interaction with the mesh and stores these data for analysis. This feature enables the plotting of forces over time, providing valuable insights into the cutting process and offering a way to measure progress and refine techniques.

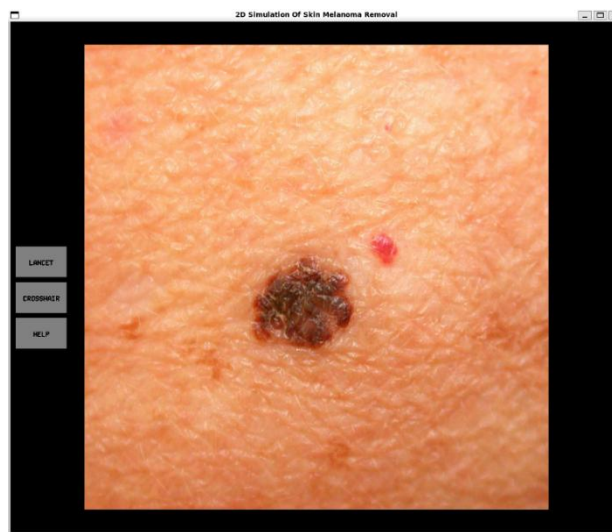


Figure 11: Advanced Textured 2D Model

Clicking the first button (LANCET) activates the lancet tool, enabling the user to interact with and deform the mesh. Clicking the button again deactivates the tool, allowing for toggling between active and inactive states.

When the lancet tool is activated, a white guidance box appears around the melanoma. This box serves as a visual aid, helping the user cut at the correct distance from the melanoma and ensuring a steady and precise cutting path. This feature enhances the simulation's educational value by encouraging accurate and controlled surgical techniques, replicating the precision required in real-life melanoma removal procedures.

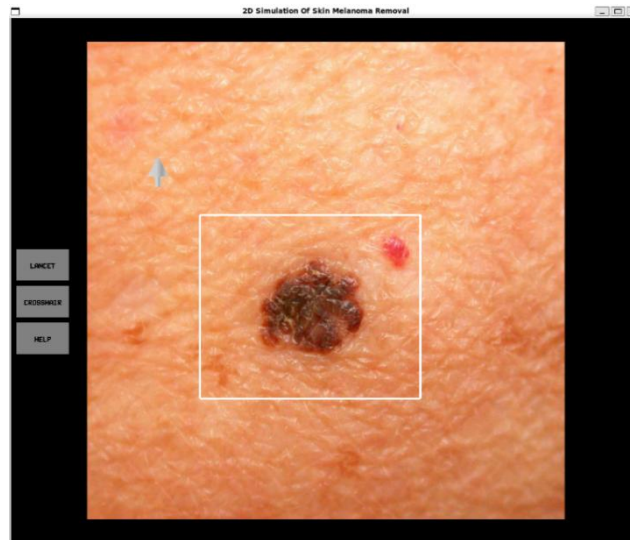


Figure 12: Lancet Tool Activated

The second button (CROSSHAIR) is designed as an extension for potential use in future projects. While it does not currently interact directly with the simulation, it lays the groundwork for additional features or functionalities that can be implemented later, offering flexibility for future development.

The third button (HELP) provides the user with access to a help box containing useful information about the simulation's controls and general usage instructions. This feature enhances user experience by offering immediate guidance, especially for those unfamiliar with the simulation. Clicking the button again hides the help box, ensuring the interface remains uncluttered during active use.

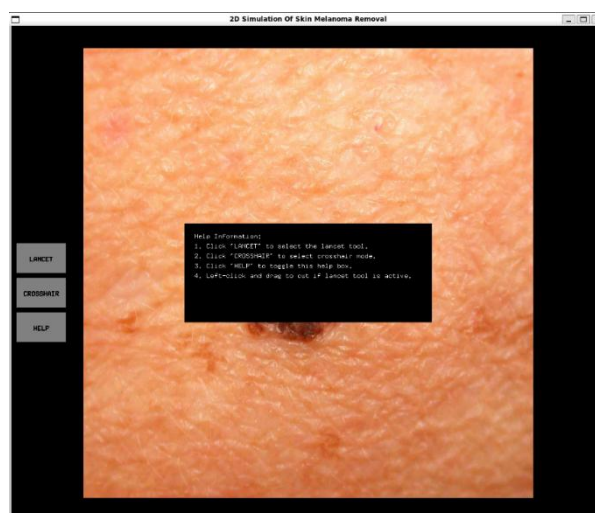


Figure 13: Display Help

Now that the user is familiar with the controls of the advanced simulation model, it is time to perform the cutting procedure. By activating the lancet tool and moving the mouse close to the designated cut box, the user can begin interacting with the mesh. Pressing and holding the left mouse button allows the user to deform the mesh dynamically by dragging the lancet across its surface. This action simulates the process of cutting, with the mesh responding in real time to the applied forces and movement.

Upon releasing the mouse button, the lancet tool is "lifted," ceasing the deformation and leaving behind a visible cut on the mesh. The resulting deformation reflects the accuracy and steadiness of the user's movements, as guided by the visual aid provided by the cut box. The cut box turns red if the user drags the tool too close to the melanoma region.

The outcomes of this cutting procedure can be observed as follows:

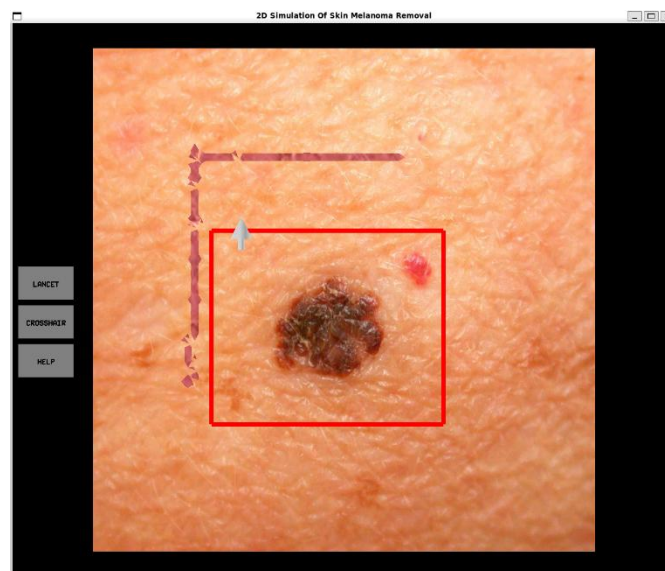


Figure 14: Result of 2D Cutting Procedure

During the cutting process, the horizontal and vertical forces generated by the mesh simulate the resistance encountered as if cutting real skin tissue. These forces dynamically oppose the movement of the lancet tool, providing a realistic representation of the effort required to perform the procedure.

After the program termination, the user has the option to plot a diagram illustrating the progression of these forces over time. This visualization provides valuable insights into the cutting process, enabling the user to analyze how the tool interacted with the mesh and how the forces varied during the procedure. Such data can be instrumental in refining techniques, evaluating performance, and enhancing the overall realism of the simulation.

A dedicated Python program generates the plot diagram below, illustrating the elapsed time on the horizontal axis and the cutting forces—horizontal F_x (blue) and vertical F_y (red).

Analysis of the diagram reveals the following issues:

1. Using a mouse as the input device introduces hand tremors, resulting in a zig-zag effect in the plotted forces. This underscores the need for a more stable and precise input mechanism.
2. The force calculation and direction detection algorithm require further optimization. Spurious force components appear, as evidenced by red spikes within the blue regions (horizontal cutting) and blue spikes within the red regions (vertical cutting).



Figure 15: Plot Diagram of Forces Felt Over Time

Chapter 8. 3D Modeling

8.1 Overcoming the Limitations of 2D Simulation

While the 2D simulation model plays a crucial role in establishing the foundational mechanics and principles of interaction, it has inherent limitations that hinder its ability to fully replicate the complexity of surgical procedures. The flat, two-dimensional representation is adequate for exploring basic dynamics like deformation and cutting mechanics, but it falls short in capturing the depth, spatial intricacy, and realistic tissue behavior required for a high-fidelity surgical simulation.

The transition to a 3D modeling framework addresses these shortcomings by introducing a multi-dimensional representation of the skin, melanoma and lancet. This advancement allows the simulation to replicate the volumetric behavior of human tissue, including its layered structure and depth-specific responses to external forces. Unlike the 2D model, where interactions are confined to basic, mouse-driven cutting, the 3D simulation can incorporate both virtual and real haptic mechanisms. This innovation enables users to interact with the mesh using a steady and reliable physical tool, enhancing immersion and offering a hands-on experience that closely mirrors real-world surgical procedures.

Additionally, the 3D model captures complex deformations that are impossible to simulate in the 2D plane. These include the natural retraction of tissue after a cut, as well as the dynamic response of deeper layers of skin to applied pressure and cutting forces. The inclusion of depth-based force feedback further improves realism, allowing users to feel resistance and tactile feedback as they manipulate the cutting tool.

The shift to 3D not only improves the visual and mechanical fidelity of the simulation but also expands its educational value. This evolution highlights the importance of dimensional representation in achieving a comprehensive and realistic surgical simulation.

8.2 Challenges in Transitioning to 3D

While the transition to a 3D simulation model offers significant advantages in terms of realism and interactivity, it also introduces a range of technical and computational challenges. The addition of a third dimension brings greater complexity to nearly every aspect of implementation, demanding innovative solutions to maintain both accuracy and performance.

One of the primary challenges is ensuring mesh stability during deformations. In the 3D model, the mesh represents a multi-layered structure with increased points of interaction and more intricate connections. Managing this complexity required precise handling to avoid instability during dynamic interactions, such as cutting or tissue manipulation. The simulation must account for realistic cutting dynamics, including the separation of vertices and the retraction of tissue layers, while maintaining mechanical fidelity across the entire mesh.

The expanded data structure of the 3D mesh also increases the computational demand. Force calculations and deformation updates now span a larger dataset, necessitating optimization strategies to maintain real-time responsiveness. Techniques such as parallel processing, GPU acceleration, and vertex caching are implemented to reduce computation times and ensure a complete user experience.

Another significant hurdle is the enhancement of custom physics algorithms capable of accurately simulating 3D interactions. These algorithms require careful calibration of parameters to ensure that the mechanical and visual responses of the mesh align with the tactile feedback provided by the haptic device. This synchronization is critical for creating an immersive experience, as discrepancies between visual and tactile feedback can disrupt the realism of the simulation.

Additionally, the transition to 3D demanded greater attention to force interactions and depth-based behaviors. The simulation needs to account for variations in tissue response based on the depth of the cut and the direction of applied forces. Implementing these features requires a detailed understanding of the mechanics of human tissue and the development of algorithms that can mimic these behaviors with high accuracy.

Overcoming these challenges necessitate a blend of creativity, observation, technical expertise, and rigorous testing. While the process is demanding, it may ultimately result in a robust and high-fidelity simulation that pushes the boundaries of realism in surgical training. The 3D model stands as a testament to the importance of innovation and persistence in addressing complex computational and mechanical problems.

8.3 Early 3D Model Designs and Iterations

The development of the 3D simulation model begins with a series of experimental designs aimed at establishing a robust foundation for rendering, interaction, and physics-based modeling in three-dimensional space. These initial prototypes, although simple in structure, play a crucial role in uncovering the complexities of working in 3D environments.

Through these early iterations, valuable insights emerge into several key aspects of 3D simulation development. Additional challenges such as managing depth perception, implementing accurate texture mapping, and maintaining real-time responsiveness become immediately apparent. The process also highlights the intricacies of coordinating interactions between the virtual mesh and external input tools, such as the mouse or haptic devices, within a 3D space.

These prototypes serve as a testing ground for integrating shaders and textures, which are essential for enhancing visual fidelity. Experimentation with object loading, texture mapping and vertex caching provides a deeper understanding of how to create a realistic and immersive representation of human skin tissue. This iterative approach allows for gradual refinement, addressing technical challenges and laying groundwork for more advanced versions.

By iterating on these experimental designs, the development process continuously identifies and solves problems, ensuring a strong foundation for achieving the desired level of realism and functionality in the final simulation product.

8.3.1 Version 0: Basic Setup & Rendering

The first design iteration, referred to as v0 (version 0), introduces a rudimentary 3D simulation window. This version successfully implements the loading of fragment and vertex shaders and renders a 3D mesh consisting of point masses interconnected with spring-dampers arranged horizontally, vertically, and diagonally. Users can zoom in and out of the center of the mesh using the mouse wheel and rotate the view using the W, A, S, D keys, providing better clarity and an improved perspective of the simulated environment.

The general principles of spring-damper removal and mass displacement remain largely the same as in the 2D version of the simulation. However, in this iteration, the algorithm must evolve to handle these processes in 3D space, introducing additional complexity.

Furthermore, the 3D mesh requires significantly more components—springs, masses, and dampers—to create depth and multiple layers that closely resemble real skin tissue. These additions are essential for accurately simulating the volumetric properties of human skin. Now, let us address the properties of this 3D prototype.

As explained in Chapter 4 (Methodology), the cutting process begins with the "Initial Compression and Penetration" phase, where the lancet is pushed vertically against the tissue to achieve penetration. During this phase, the cutting tool applies force along the z-axis to compress and ultimately break through the upper layers of the tissue. However, in this version of the simulation, the mouse cursor is constrained to movement along the x-y plane and lacks the ability to move along the z-axis. This restriction prevents the cursor from accurately interacting with the mesh in 3D space, making it impossible to simulate the deformation and penetration of the tissue during this critical phase. Without the ability to apply depth-based movement and forces, the simulation fails to replicate the natural behavior of tissue compression and penetration, limiting its realism and interactivity.

This limitation underscores the need for a more advanced cutting tool capable of operating in 3D space. Such a tool must allow users to interact with the mesh along all three axes, ensuring accurate deformation and penetration of the tissue. Addressing this challenge marks a pivotal step toward achieving a more realistic and immersive simulation, bridging the gap between theoretical modeling and practical surgical training.

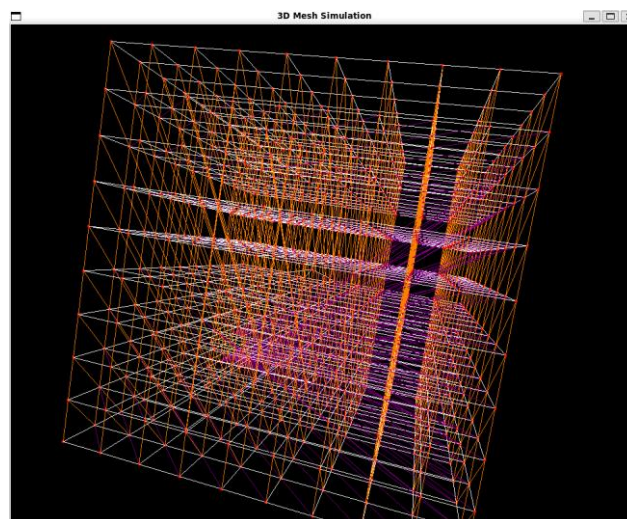


Figure 16: Early 3D SMD Model

8.3.2 Version 1: 3D Lancet Tool & Texture Mapping

This version referred to as v1, successfully implements the loading of fragment and vertex shaders, enabling the rendering of a 3D skin mesh with an applied texture. Additionally, a 3D object resembling a lancet is loaded and rendered. The lancet features a handle and a pointed edge, specifically designed for cutting the skin, with metallic textures applied to enhance its visual realism and give it the appearance of a surgical tool. The lancet can be moved freely through space using the W, A, S, D keys for directional movement and rotated using the arrow keys, allowing basic interactivity within the simulation environment.

However, several limitations become evident at an early stage. The simulation currently cannot detect the UV coordinates essential for accurate texture mapping—a critical component for creating a realistic representation of the skin mesh. In the absence of proper UV mapping, the applied texture appears misaligned and unrealistic, thereby diminishing the simulation's overall fidelity. Moreover, a more detailed, high-resolution texture is required for both the skin layers and the melanoma to achieve greater visual realism. Finally, the lancet's inability to interact with the skin mesh remains unresolved, preventing meaningful engagement between the tool and the virtual tissue.

Despite these setbacks, this version serves as a valuable testbed for exploring the fundamentals of shader functionality and understanding the challenges associated with texture mapping. These experiments highlight key areas for improvement, particularly the need for a sophisticated collision detection mechanism that would enable the lancet to interact dynamically with the mesh. Proper texture mapping also emerges as a critical requirement for achieving visual realism. While this version is far from complete, it establishes a foundation for addressing these challenges in subsequent iterations of the simulation.

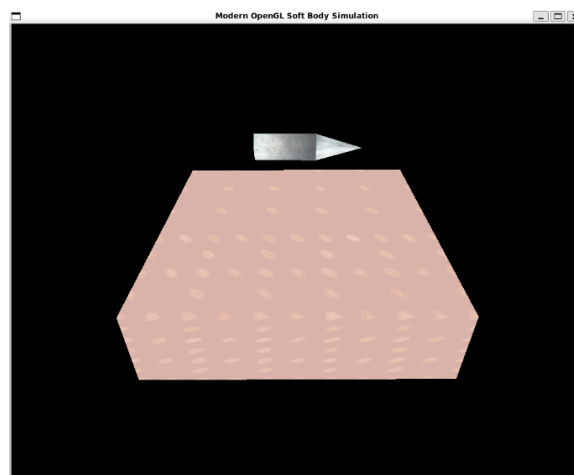


Figure 17: Early Textured 3D SMD Model & Lancet

8.3.3 Version 2: Almost Complete

8.3.3.1 Creating a 3D cutting tool

The lancet is custom-designed and created using Blender, a platform previously discussed in this report and deemed unsuitable for the simulation itself. However, Blender proves to be an invaluable tool for designing unique and intricate 3D objects, making it the ideal choice for creating the lancet. By utilizing its robust modeling features, Blender allows us to craft a highly detailed surgical tool tailored specifically to the needs of the simulation.

The design process begins with the careful placement of vertices to form a 2D frame that outlines the lancet's handle and blade. A reference image of a real surgical lancet is used as a guide to ensure accuracy and authenticity. Once the initial frame is constructed, the vertices are connected to create surfaces, which are then extruded to transform the flat 2D shape into a fully realized 3D object. This critical step establishes the physical dimensions and proportions of the lancet, giving it a realistic structure suitable for integration into the simulation.

To enhance both visual and functional realism, beveling and sharpening techniques are applied to refine the blade's edges. These adjustments create a sharp, precise blade while increasing the vertex density along its surface. This added detail is crucial, as the blade must be capable of colliding and interacting dynamically with the skin mesh. The dense vertex arrangement ensures that the blade can cut through the mesh effectively, simulating realistic tissue separation during the cutting process.

This meticulous design process results in a tool that not only adds visual authenticity to the simulation but also provides the mechanical precision required for effective collision detection and realistic deformation of the skin mesh. The custom-designed lancet is an essential component of the simulation, bridging the gap between virtual and physical interactions and significantly contributing to the overall realism of the surgical procedure.

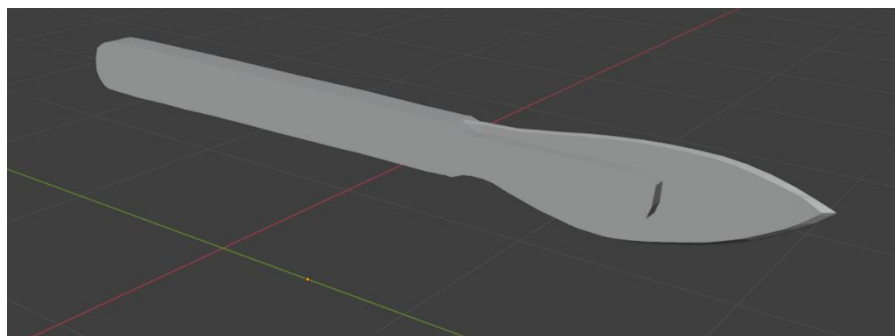


Figure 18: Creation of 3D Lancet

8.3.3.2 Exporting and loading to simulation

After creating and saving the Blender file, a natural question arises: how do we integrate this custom-designed object into the 3D simulation? The answer is not immediately obvious, and several experiments were necessary to arrive at a practical and efficient solution.

The first step involves exporting the object in a format that is both readable and easily parsable by the simulation. For this purpose, we use the Wavefront (.obj) format, which is widely supported and straightforward to work with. This can be achieved through Blender's Export -> Wavefront (.obj) functionality. The .obj file contains the essential geometric data of the object, including vertices, UV coordinates, and faces. These components are labeled with string identifiers: v for vertices, vt for texture coordinates, vn for vertex normals, and f for faces.

However, during the initial exploration, we observe that the faces in the exported file may consist of more than three numbers, representing polygons with more than three vertices. This irregularity complicates the parsing algorithm and introduces additional complexity in rendering the object. To simplify the process, we choose to export the mesh in a triangulated form, ensuring that every face consists of exactly three vertices. This adjustment guarantees uniformity in the file structure, making it easier to identify and parse each component.

The second step involves setting up the parsing algorithm. Using embedded libraries for file handling, we open and read the .obj file line by line. The algorithm systematically filters out unnecessary information, such as comments or unsupported features, and retains only the relevant mesh data. This data is then organized into a specialized object structure, where the vertices, UV coordinates, normals, and faces are stored in a format suitable for rendering within the simulation.

Once the parsing is complete, the data is used to render the lancet in the simulation. To enhance its realism, we apply a metallic texture to the lancet, ensuring it visually resembles a real surgical tool. The texture is mapped using the UV coordinates from the .obj file, creating a polished and lifelike appearance.

Through this workflow, the custom-designed lancet transitions accurately from Blender to the simulation environment, fully functional and visually realistic. This process not only highlights the importance of careful export and parsing strategies but also ensures the successful integration of complex 3D objects into the simulation.

8.3.3.3 Advanced 3D Meshes

Building on the lessons from version 0 & 1 shifts us towards additionally utilizing GLUT, an approach already proven effective in the 2D versions of the simulation. This one introduces several critical improvements and now includes the following attributes:

- 1. Simulation Window:**

The simulation opens a 1400×1200-pixel window, providing ample screen space to observe and analyze the deformation and cutting processes in detail.

- 2. Frame Rate Regulation:**

The system maintains a frame rate of 60 frames per second (fps), independently regulated from the user-perceived force calculation loop. This setup ensures fluid visual rendering without compromising the accuracy of visual effect-force computations.

- 3. Skin Layer Initialization:**

The point masses are replicated fourfold, forming quads of vertices (VertexQuad) to enable the desired splitting effect. The simulation then initializes an epidermis mesh, together with the corresponding internal VertexQuad, Spring, and Damper structures. These data entities track and manage geometric and topological attributes crucial for realistic tissue modeling.

- 4. Camera Configuration:**

The camera's view is centered on the epidermis mesh, granting a clear vantage point for both external visualization and observation of underlying deformations.

- 5. Mesh Rendering:**

The epidermis mesh is rendered from the front face as well as its internal and external side faces, offering a comprehensive depiction of tissue layers during the simulation.

- 6. Texture Lancet Mesh Initialization:**

A lancet mesh is introduced at a defined position and orientation relative to the epidermis, aligning with the end effector of the real haptic mechanism. This alignment ensures consistency between the virtual tool and its physical counterpart. A metallic texture is loaded and rendered on the lancet mesh, enhancing the visual fidelity of the surgical instrument.

7. Lancet Manipulation:

The lancet can be moved using the keyboard arrow keys. Left and right for lateral displacement, and up and down for forward and backward motion.

8. Camera Orbit Controls:

The camera can orbit around the scene horizontally (keys A and D) and vertically (keys W and S), enabling flexible viewing angles for more detailed analysis.

9. Side Toolbar Menu:

The simulation provides a toolbar menu with the following functionalities:

- **Lancet Control:** Activate or deactivate the lancet.
- **Camera Lock:** Lock or unlock camera orbiting.
- **Help Display:** Show or hide a help box containing usage instructions.
- **Cut Box Display:** Activate or deactivate a guiding square around the melanoma, delineating the optimal region for incision.

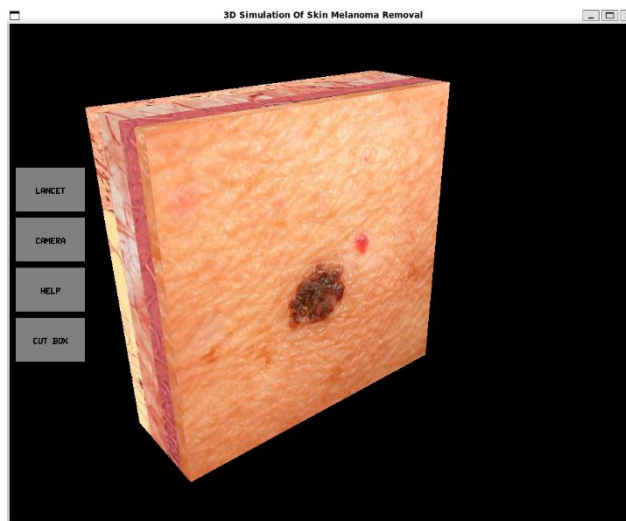


Figure 19: Layered Textured SMD Model with Side Toolbar

10. VertexQuad Detection and Cutting

The system identifies the closest VertexQuad to the blade of the lancet by calculating the distance. Upon detection, the following actions occur:

- **Deformation:**

A pressure proportional to the lancet's movement is applied along the z-axis, causing the epidermis mesh to deform in a manner analogous to physical tissue compression in real-life scenarios.

- **Splitting-Penetration:**

Once the *SPLIT_THRESHOLD* is exceeded, one or more spring-damper elements collapse, causing the affected quad of masses and vertices (VertexQuad) to separate. Subsequently, the lancet penetrates the epidermis, forming a small, wound-like gap.

- **Further Lateral Cutting:**

After penetration, the lancet may continue cutting horizontally by moving left or right or proceed deeper into the tissue.

11. Calculation of Vertical (Fz) & Horizontal (Fx) Forces:

The force Fz is calculated throughout the vertical pressure application on the epidermis, beginning with initial contact and continuing until the lancet is fully withdrawn from the epidermal layer, at which point both Fx and Fz return to zero. The force Fx is determined when the lancet performs horizontal cuts to the left or right following epidermal penetration.

12. Force and Displacement Update Rate:

Force calculations and vertex displacements are performed independently from the graphical rendering process. By employing this separation, the system maintains real-time responsiveness while preserving the accuracy of force modeling for both visual effects and haptic feedback for the user.

13. Data Logging and Plotting:

Both Fx and Fz are recorded continuously in a log file, which opens for writing at the start of the simulation. This data is then visualized using the specialized Plot_Diagram.py script, enabling a detailed analysis of force profiles over time in relation to the lancet's movement.

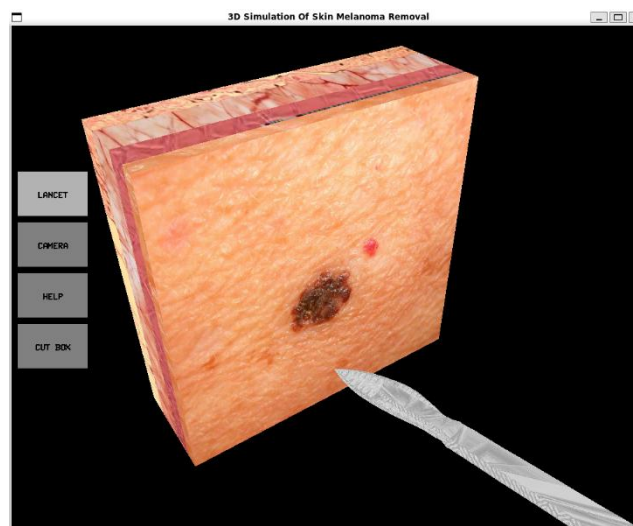


Figure 20: Layered Textured SMD Model with Lancet Activated

Chapter 9. STUDYING THE HAPTIC MECHANISM

Before establishing the connection with the haptic mechanism, a detailed examination of its properties is essential. This chapter explores the mechanical characteristics of the available five-bar linkage robotic haptic mechanism. Despite its seemingly simple design, achieving optimal performance necessitates complex calculations.

9.1 Introduction

A five-bar linkage haptic mechanism constitutes a specialized mechanical system employed in haptic feedback devices. Haptic feedback—often referred to as force feedback or tactile feedback—provides users with a sense of touch or force in virtual or remote environments. By enabling users to feel and interact with virtual or remote objects, it enhances overall user immersion in applications such as virtual reality (VR), simulators, and remote robotic control.

Five-bar linkage haptic mechanisms aim to offer a broad spectrum of feedback sensations, including force, pressure, texture, and vibration. They find applications in diverse areas, such as gaming, medical simulations, training platforms, and remote robotic operations. Moreover, the design and control algorithms governing these mechanisms can be highly complex, as they seek to create a seamless and immersive user experience in both virtual and remote environments.

In this mechanism, the “five-bar linkage” denotes an arrangement of mechanical linkages and joints that transmit forces and motions to the user’s hand or other body parts. It consists of five interconnected bars joined by pivot joints in a pentagon topology, designed to mimic the kinematics of the human hand and deliver a realistic sense of touch and force feedback.

The principal elements of a five-bar linkage haptic mechanism encompass the following:

1. **Actuators:** Typically implemented as motors or equivalent devices, actuators generate the forces and motions necessary to simulate tactile sensations. By driving the movement of the mechanical links, they enable the delivery of resistance or force feedback.
2. **Sensors:** Sensors measure the position and motion of the user's hand or the manipulated object. Their output is crucial for the control system, which uses this information to tailor haptic feedback to the user's real-time interactions.
3. **Control System:** This module processes data gathered by the sensors and issues control signals to the actuators. It computes the forces, resistances, or other tactile cues that faithfully simulate user interactions with virtual or remote objects.
4. **Links and Joints:** The five-bar linkage architecture comprises five interconnected links joined by pivot joints. This configuration emulates the kinematics and flexibility of the human hand. Certain haptic mechanisms may feature a greater or lesser number of bars, depending on design requirements.
5. **End-Effector:** Serving as the interface between the mechanism and the user, the end-effector is the component that physically transmits force and tactile sensations to the user's hand or other body parts.

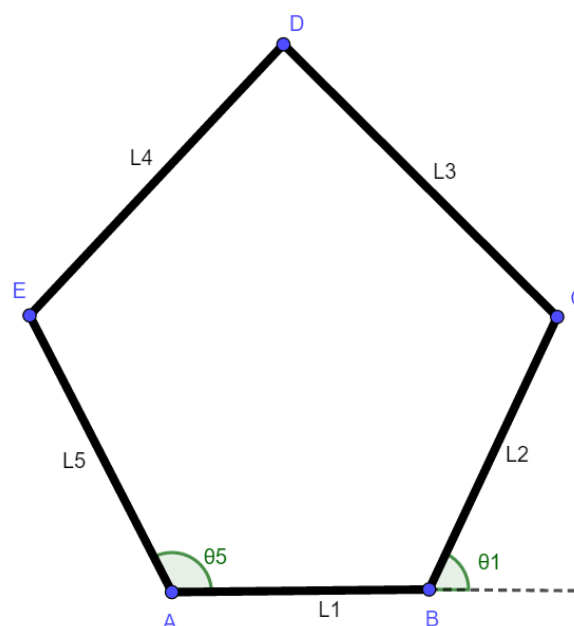


Figure 21: Simple Diagram of a Closed-Chain 5-Bar-Linkage Mechanism

9.2 Design

In the accompanying photo, a five-bar linkage mechanism, adapted from a traditional pantograph, consists of five interconnected metallic bars forming the fundamental structure. On the left side, one can see the end effector: a handle that the user grips to manipulate and interact with the system. Positioned on the right side are two electric actuators, which are crucial for generating the haptic forces the user experiences [CHB93].

These actuators are driven by voltage signals supplied through specialized driver chips. The driver chips, in turn, receive their instructions from a central processor. To provide the processor with real-time data, two dedicated I/O cards read the motor angles, voltages, and positions at high frequency. Importantly, the I/O cards do not carry out any heavy computations; rather, they act as intermediaries, forwarding sensor and motor data to the processor. Once the processor computes the required torque values based on this incoming information, it sends corresponding commands back to the driver chips. The driver chips then apply precise volt signals to the actuators, enabling them to generate accurate torques and forces and feedback sensations.

Thanks to the original pantograph geometry, the mechanism can accurately track the user's hand movements by monitoring actuator angles instead of using sensors placed directly on the end effector. As a result, the user experiences vertical and horizontal forces that match those computed by the simulation. This setup ensures a realistic sense of touch and resistance, making the adapted five-bar linkage mechanism particularly suitable for a wide array of applications, including virtual reality, telemedicine, remote robotic operation, and training simulators. By separating data acquisition (via the I/O cards) from calculation (in the processor) and torque/force generation (through the actuators and driver chips), the system maintains a clear, modular workflow that supports precise control and immersive haptic feedback.

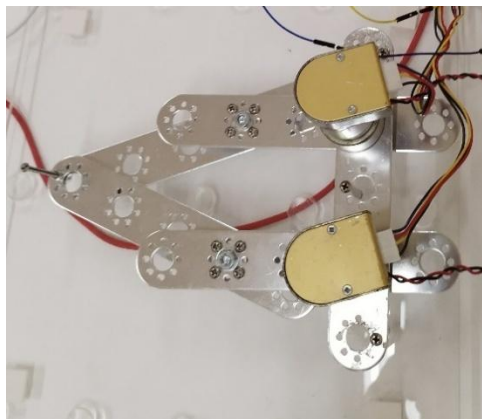


Figure 22: Pantograph structure

9.3 Kinematic Analysis

Kinematic analysis of a mechanism investigates the geometric and temporal relationships among interconnected links, joints, and moving parts. It aims to determine how specific points or end-effectors move as a function of given inputs or constraints, which may include joint angles, link lengths, or other parameters. By quantifying the positions, velocities, and accelerations of these components, kinematic analysis provides crucial insights into the motion transmission capabilities of the system, thereby facilitating the design and optimization of mechanical devices.

9.3.1 Forward Kinematics

Forward kinematics determines the coordinates of the end-effector given the characteristic link lengths and the angles of the driven links. The process of locating these coordinates is relatively straightforward. First, one computes the coordinates of points A and B through direct trigonometric calculations. Next, one applies the circle equation to identify the intersection points of two circles centered at A and B, respectively, with radii l_2 and l_3 (as illustrated in Figure 23).

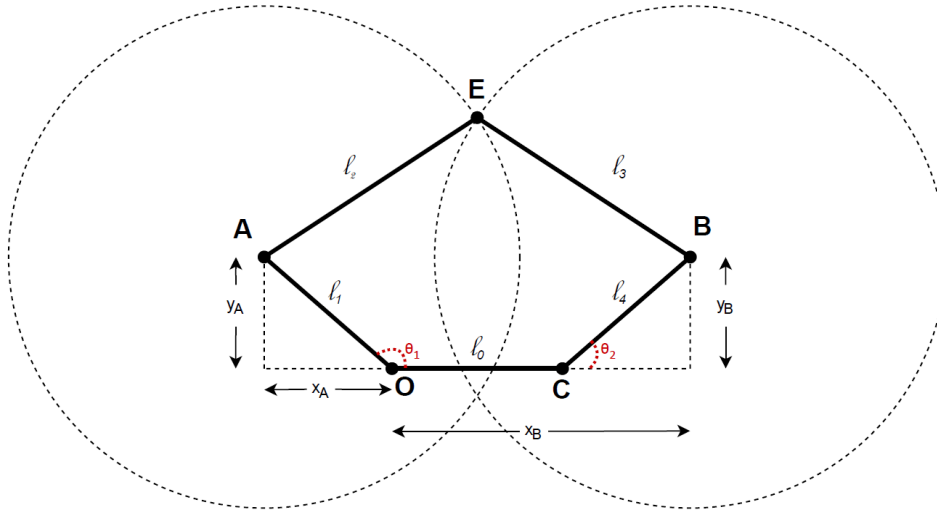


Figure 23: Forward Kinematics

The positive solution from the intersection of the two circles yields the end-effector coordinates. Point OO denotes the origin of the coordinate system, and point EE represents the end-effector.

By applying basic trigonometric relations, the coordinates of point A are derived as follows:

$$x_A = l_1 \cos(\theta_1) \quad (1)$$

$$y_A = l_1 \sin(\theta_1) \quad (2)$$

In a similar manner, the coordinates of point BB are derived as follows:

$$x_B = l_0 + l_4 \cos(\theta_2) \quad (3)$$

$$y_B = l_4 \sin(\theta_2) \quad (4)$$

Next, these derived equations are substituted into the equations of the two circles centered at points A & B.

For the circle centered at point A:

$$\begin{aligned} l_2^2 &= (x_E - x_A)^2 + (y_E - y_A)^2 \Leftrightarrow \\ l_2^2 &= ((x_E - l_1 \cos(\theta_1))^2 + ((y_E - l_1 \sin(\theta_1))^2 \Leftrightarrow \\ x_E^2 - 2x_E x_A + x_A^2 + y_E^2 - 2y_E y_A + y_A^2 - l_2^2 &= 0 \quad (5) \end{aligned}$$

For the circle centered at point B:

$$\begin{aligned} l_3^2 &= (x_E - x_B)^2 + (y_E - y_B)^2 \Leftrightarrow \\ l_3^2 &= (x_E - l_0 - l_4 \cos(\theta_2))^2 + (y_E - l_4 \sin(\theta_2))^2 \Leftrightarrow \\ x_E^2 - 2x_E x_B + x_B^2 + y_E^2 - 2y_E y_B + y_B^2 - l_3^2 &= 0 \quad (6) \end{aligned}$$

By equalizing Eq. (5) & Eq. (6) and solving for x_E :

$$\begin{aligned} x_E &= \frac{y_A - y_B}{x_B - x_A} y_E + \frac{y_B^2 + x_B^2 - l_1^2 - l_3^2}{2(x_B - x_A)} \Leftrightarrow \\ x_E &= a y_E + b \quad (7) \end{aligned}$$

Subsequently, substituting Eq. (7) into Eq. (5):

$$y_E = \frac{-d \pm \sqrt{d^2 - 4ce}}{2c}$$

Where:

$$c = a^2 + 1$$

$$d = 2(ab - 2ax_A - 2y_A)$$

$$e = b^2 - 2bx_A + l_1^2 - l_2^2$$

From the above expression for y_E , the positive solution is retained. This approach ensures physically meaningful results within the system, as negative solutions often correspond to non-physical or invalid operating parameters under the given constraints.

9.4 Inverse Kinematics

Inverse kinematics determines the angular positions of the driven links once the characteristic link lengths and the coordinates of the end-effector are established. This solution is obtained straightforwardly by applying the distance formula and the law of cosines, which exploit the geometric relationships between link lengths and end-effector position. With the effector coordinates defined, trigonometric methods and the cosine law are employed to calculate the rotational angles of the active joints (θ_1 and θ_4), as well as the passive joints (θ_2 and θ_3). Consequently, these computed joint angles ensure precise positioning and motion control in the five-bar linkage mechanism.

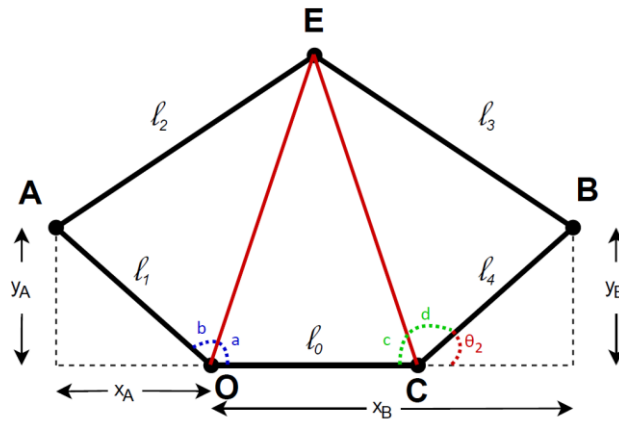


Figure 24: Inverse Kinematics

Using the distance formula, we calculate the lengths of the two newly added edges in the pantograph diagram (**red lines**), which connect the end effector to points O and C. These edges reflect the distance of the end effector from each reference point, thereby providing an accurate representation of the geometric relationships within the linkage mechanism. This step is critical for maintaining precise measurements, enabling more reliable analysis and design optimization for the overall system.

Left Edge **OE**:

$$OE = \sqrt{(x_E - x_O)^2 + (y_E - y_O)^2} \Leftrightarrow$$

$$OE^2 = x_E^2 + y_E^2$$

Right Edge **CE**:

$$CE = \sqrt{(x_E - x_C)^2 + (y_E - y_C)^2} \Leftrightarrow$$

$$CE^2 = (x_E - l_0)^2 + y_E^2 \Leftrightarrow$$

$$CE^2 = x_E^2 - 2x_E l_0 + l_0^2 + y_E^2$$

At this stage, the law of cosines is applied to $\triangle OEC$ to determine the measure of angles α & β . This mathematical approach leverages the known lengths of the triangle's sides to accurately calculate the angular relationship between them. By utilizing the law of cosines, we can derive the value of angle α & β , which are critical for understanding the spatial configuration and movement dynamics of the five-bar linkage mechanism. Accurate determination of this angle facilitates precise kinematic analysis, ensuring that the mechanism operates within the desired parameters and achieves optimal performance. Consequently, this application of the law of cosines is integral to the overall assessment and validation of the mechanical design.

1.The law of cosines will be applied to $\triangle OEC$ to find angle a :

$$CE^2 = l_0^2 + OE^2 - 2l_0OE\cos(a) \Leftrightarrow$$

$$\cos(a) = \frac{OE^2 - CE^2 + l_0^2}{2l_0OE} \Leftrightarrow$$

$$\mathbf{a} = \arccos\left(\frac{x_E}{\sqrt{x_E^2 + y_E^2}}\right)$$

2.The law of cosines will be applied to $\triangle OEA$ to find angle b .

$$l_2^2 = l_1^2 + OE^2 - 2l_1OE\cos(b) \Leftrightarrow$$

$$\cos(b) = \frac{l_1^2 - l_2^2 + x_E^2 + y_E^2}{2l_1\sqrt{x_E^2 + y_E^2}} \Leftrightarrow$$

$$\mathbf{b} = \arccos\left(\frac{l_1^2 - l_2^2 + x_E^2 + y_E^2}{2l_1\sqrt{x_E^2 + y_E^2}}\right)$$

3.The law of cosines will be applied to $\triangle OEC$ to find angle c .

$$OE^2 = CE^2 + l_0^2 - 2l_0CE\cos(c) \Leftrightarrow$$

$$\cos(c) = \frac{2l_0^2 - 2x_E}{2l_0\sqrt{(x_E - l_0)^2 + y_E^2}} \Leftrightarrow$$

$$\mathbf{c} = \arccos\left(\frac{l_0 - x_E}{\sqrt{(x_E - l_0)^2 + y_E^2}}\right)$$

4.The law of cosines will be applied to $\triangle CEB$ to find angle d .

$$l_3^2 = CE^2 + l_4^2 - 2l_4CE\cos(d) \Leftrightarrow$$

$$\cos(d) = \frac{x_E^2 + y_E^2 + l_0^2 + l_4^2 - l_3^2 - 2x_El_0}{2l_4\sqrt{(x_E - l_0)^2 + y_E^2}} \Leftrightarrow$$

$$\mathbf{d} = \arccos\left(\frac{x_E^2 + y_E^2 + l_0^2 + l_4^2 - l_3^2 - 2x_El_0}{2l_4\sqrt{(x_E - l_0)^2 + y_E^2}}\right)$$

Upon calculating the angles, a , b , c , and d , the motor angles θ_1 and θ_2 are subsequently derived using the established kinematic relationships. This derivation process employs the previously determined angles to accurately determine the necessary rotational positions of the motors controlling the active joints. Specifically, the motor angle θ_1 is calculated based on the geometric configuration and the desired movement trajectory of the first active joint, while θ_2 is derived similarly for the second active joint. By meticulously deriving θ_1 and θ_2 , the system achieves optimal synchronization and control, which are essential for the accurate and efficient operation of the robotic haptic mechanism.

$$\theta_1 = a + b$$

$$\theta_2 = \pi - (c + d)$$

9.5 Validation

Utilizing the specialized Python script `Pantograph_Forward_Kinematics.py`, which incorporates the previously discussed formulas, we generate a plotted diagram of the five-bar linkage mechanism's links and angles. This script enables the visualization of the geometric configuration, allowing for a comprehensive analysis of each link and its corresponding angles. By setting the motor angles to $\theta_1 = \theta_2 = 45^\circ$, we can accurately represent the mechanism's configuration and ensure that the theoretical calculations are correctly implemented. Furthermore, this plotting and validation process confirms the reliability of the forward kinematics model, providing confidence in the mechanism's design and functionality.

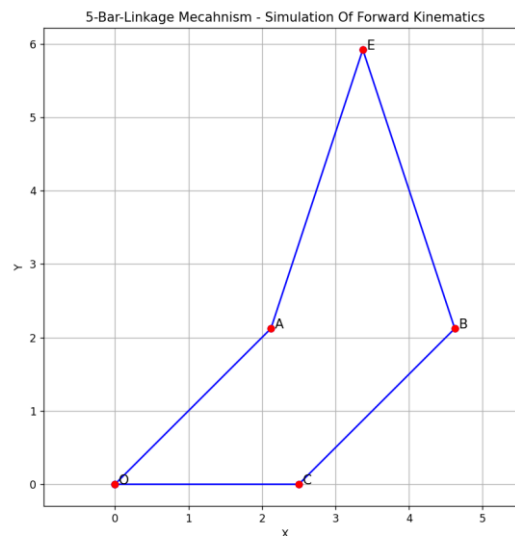


Figure 25: Simulation of Kinematics

9.6 Static Analysis

In robotics, static analysis involves examining a mechanism under equilibrium conditions, meaning all applied forces and torques balance out. This analysis helps in determining reaction forces, joint torques, and the force transmission capabilities of the robotic system.

9.6.1 Static Equilibrium in Robotics

For a robotic mechanism, static equilibrium requires that:

$$\sum F = 0, \quad \sum M = 0$$

where F represents forces and M represents moments (torques). The forces acting on a robotic manipulator include applied external forces, gravitational forces, and internal reaction forces at joints.

To compute these forces efficiently, we rely on the Jacobian matrix, which relates joint velocities to end-effector velocities in kinematic analysis but can also be used to relate forces and torques in static analysis.

9.6.2 Formation of the Jacobian Matrix

The Jacobian matrix J is a fundamental tool in robotic kinematics and statics. It provides a mapping between the joint space and the Cartesian space.

For a robotic mechanism with n degrees of freedom (DOF), the end-effector velocity v and angular velocity ω are related to the joint velocity $\dot{q}$ by:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = J(q)\dot{q}$$

where $J(q)$ is the geometric Jacobian matrix, which depends on the joint positions q .

9.6.3 Using Jacobian Matrix

By the principle of virtual work, the forces in the Cartesian space and the torques in the joint space are related by the transpose of the Jacobian matrix:

$$J^T(q)F_{ext} = \tau$$

where:

- $\mathbf{F}_{ext}$ is the force and moment applied at the end-effector in Cartesian coordinates.
- $\boldsymbol{\tau}$ represents the joint torques or forces in the actuator space.
- $\mathbf{J}^T$ acts as a force transformation matrix from the workspace (Cartesian space) to the joint space.

This equation is crucial in force control, where external forces applied to the robot (e.g., by a haptic mechanism) need to be mapped to actuator torques.

9.6.4 Computation of Pantograph Jacobian Matrix

Since the physical parts of the mechanism are moved by 2 actuators/motors, it is mandatory that we compute the Jacobian matrix and especially its transpose so we'll be able to know what torques the actuators should apply so that the force applied to the user's hand is the one anticipated.

One approach is to decompose the robotic system into two new separate systems, whose end-effector points will have the same velocity. For each system, the position equations of the end-effector are easily found through homogeneous transformations, and then they are differentiated to obtain the velocities, which will be equal.

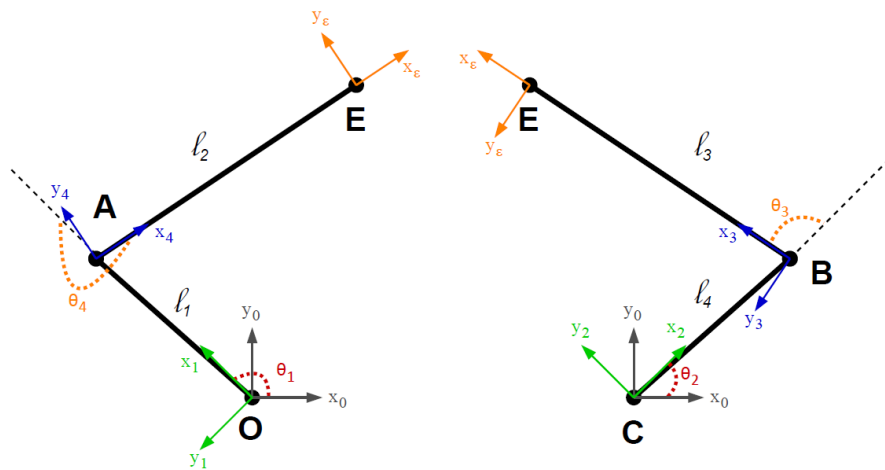


Figure 26: Two different robotic manipulators

Through the combination of successive transformations T from coordinate frame 0 to coordinate frame E, the position of the end-effector point is calculated

1. Applying matrix multiplication and differentiation on the result for the left system:

$${}^0T_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^1T_4 = \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & l_1 \\ \sin(\theta_4) & \cos(\theta_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^4T_E = \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0J_u = \begin{bmatrix} -l_1 \sin(\theta_1) - l_2(\sin(\theta_1) \cos(\theta_4) + \cos(\theta_1) \sin(\theta_4)) & -l_2(\sin(\theta_1) \cos(\theta_4) + \cos(\theta_1) \sin(\theta_4)) \\ l_1 \cos(\theta_1) + l_2(\cos(\theta_1) \cos(\theta_4) - \sin(\theta_1) \sin(\theta_4)) & l_2(\cos(\theta_1) \cos(\theta_4) - \sin(\theta_1) \sin(\theta_4)) \end{bmatrix}$$

For simplicity, the above matrix will be written as:

$${}^0J_u = \begin{bmatrix} -\alpha_{00} & -\alpha_{01} \\ \alpha_{10} & \alpha_{11} \end{bmatrix}$$

Thus, the velocities of the end-effector point can be written as follows:

$$\dot{x}_E = -\alpha_{00}\dot{\theta}_1 - \alpha_{01}\dot{\theta}_4 \quad \dot{y}_E = \alpha_{10}\dot{\theta}_1 + \alpha_{11}\dot{\theta}_4 \quad (1)$$

2. Subsequently, the transformations for the right system are calculated.

$${}^0T_2 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^2T_3 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & l_4 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^4T_E = \begin{bmatrix} 1 & 0 & 0 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0J_u = \begin{bmatrix} -l_4 \sin(\theta_2) - l_3(\sin(\theta_2) \cos(\theta_3) + \cos(\theta_2) \sin(\theta_3)) & -l_3(\sin(\theta_2) \cos(\theta_3) + \cos(\theta_2) \sin(\theta_3)) \\ l_4 \cos(\theta_2) + l_3(\cos(\theta_2) \cos(\theta_3) - \sin(\theta_2) \sin(\theta_3)) & l_3(\cos(\theta_2) \cos(\theta_3) - \sin(\theta_2) \sin(\theta_3)) \end{bmatrix}$$

For simplicity, the above matrix will be written as:

$${}^0J_u = \begin{bmatrix} -\beta_{00} & -\beta_{01} \\ \beta_{10} & \beta_{11} \end{bmatrix}$$

Thus, the velocities of the end-effector point can be written as follows:

$$\dot{x}_E = -\beta_{00}\dot{\theta}_2 - \beta_{01}\dot{\theta}_3 \quad \dot{y}_E = \beta_{10}\dot{\theta}_2 + \beta_{11}\dot{\theta}_3 \quad (2)$$

At this point, the velocities $\dot{\theta}_3$ and $\dot{\theta}_4$ must be eliminated from the velocity equations of the end-effector point, because, as previously mentioned, these angles are passive and cannot be controlled by the motors.

Solving for $\dot{\theta}_4$ in Eq. (1) and substituting it in $\dot{y}_E$:

$$\dot{y}_E = -\frac{a_{11}}{a_{01}}\dot{x}_E + \frac{a_{10}a_{01} - a_{11}a_{00}}{a_{01}}\dot{\theta}_1 \quad (3)$$

We follow the same process for $\dot{\theta}_3$ in Eq. (2):

$$\dot{y}_E = -\frac{\beta_{11}}{\beta_{01}}\dot{x}_E + \frac{\beta_{10}\beta_{01} - \beta_{11}\beta_{00}}{\beta_{01}}\dot{\theta}_2 \quad (4)$$

At this point, Eq. (3) and Eq. (4) are set equal because, as previously mentioned, the end-effector points of the two systems have the same velocities.

$$\dot{x}_E = -\frac{\beta_{01}(a_{10}a_{01} - a_{11}a_{00})}{\beta_{01}a_{01} - a_{11}\beta_{01}}\dot{\theta}_1 + \frac{a_{01}(\beta_{10}\beta_{01} - \beta_{11}\beta_{00})}{\beta_{11}a_{01} - a_{11}\beta_{01}}\dot{\theta}_2 \quad (5)$$

$$\dot{y}_E = \left(\frac{a_{10}a_{01} - a_{11}a_{00}}{a_{01}} + \frac{a_{11}\beta_{01}(a_{10}a_{01} - a_{11}a_{00})}{a_{01}(\beta_{11}a_{01} - a_{11}\beta_{01})} \right) \dot{\theta}_1 - \frac{a_{11}(\beta_{10}\beta_{01} - \beta_{11}\beta_{00})}{\beta_{11}a_{01} - a_{11}\beta_{01}} \dot{\theta}_2 \quad (6)$$

Eq. (5) and Eq. (6) represent the symbolic solutions of Differential Kinematics.

Thus, the resulting Jacobian matrix is:

$${}^0J_u = \begin{bmatrix} \gamma_{00} & \gamma_{01} \\ \gamma_{10} & \gamma_{11} \end{bmatrix} \quad {}^0J_u^T = \begin{bmatrix} \gamma_{00} & \gamma_{10} \\ \gamma_{01} & \gamma_{11} \end{bmatrix}$$

Where:

$$\begin{aligned} \gamma_{00} &= -\frac{\beta_{01}(a_{10}a_{01} - a_{11}a_{00})}{\beta_{01}a_{01} - a_{11}\beta_{01}} \\ \gamma_{01} &= \frac{a_{01}(\beta_{10}\beta_{01} - \beta_{11}\beta_{00})}{\beta_{11}a_{01} - a_{11}\beta_{01}} \\ \gamma_{10} &= \left(\frac{a_{10}a_{01} - a_{11}a_{00}}{a_{01}} + \frac{a_{11}\beta_{01}(a_{10}a_{01} - a_{11}a_{00})}{a_{01}(\beta_{11}a_{01} - a_{11}\beta_{01})} \right) \\ \gamma_{11} &= -\frac{a_{11}(\beta_{10}\beta_{01} - \beta_{11}\beta_{00})}{\beta_{11}a_{01} - a_{11}\beta_{01}} \end{aligned}$$

Chapter 10. HAPTIC CONNECTION

The connection between the simulation and the mechanism employs a dual client-server architecture, ensuring efficient and synchronized data exchange throughout operation. This configuration enables the simulation and mechanism to transmit critical parameters, such as position and force, in real time, thereby facilitating accurate feedback and refined control. By maintaining continuous bidirectional communication, the architecture allows both ends to communicate and exchange the data needed in real time. Consequently, this robust setup ensures that both the simulation and the mechanism remain tightly integrated, enhancing overall responsiveness and operational precision [BCR94].

10.1 Architecture

The overall architecture is designed based on network communication principles, utilizing a TCP connection to establish a link between a client and a server program through a specified port. For data exchange to occur seamlessly, both programs must operate on the same ports, with TCP traffic explicitly allowed on both ends. Additionally, any antivirus software or firewall configurations that might block the connection must be disabled or adjusted to permit this critical communication. This setup ensures reliable and uninterrupted data transmission between the client and server, forming the backbone of the system's functionality.

10.1.1

Client Architecture

The client component within the network architecture is a critical module responsible for establishing, maintaining, and managing communication with the server. It is implemented as a structured C++ program and is designed to perform the following key functions:

1. Initialization and Connection:

The client initializes by creating a socket utilizing the AF_INET protocol family for IPv4 communication and the SOCK_STREAM type for TCP-based data transmission. It establishes a connection to the server by specifying its IP address and port. Upon successful connection, the client sets its status to active, ensuring readiness for subsequent data exchange. Robust error-handling mechanisms are integrated to identify and report any issues encountered during the connection process, such as invalid addresses or connection failures.

2. Data Transmission:

The client facilitates data exchange through its sendData() method, enabling the transmission of information to the server. Prior to data transmission, the client verifies the active status of the connection to prevent communication errors. This method ensures reliable and consistent data delivery within the TCP-based communication framework.

3. Connection Termination:

The client incorporates a closeConnection() method to securely terminate the connection with the server. This process includes releasing allocated resources, closing the socket, and updating the connection status to inactive. These measures are essential for maintaining system stability and avoiding resource leaks.

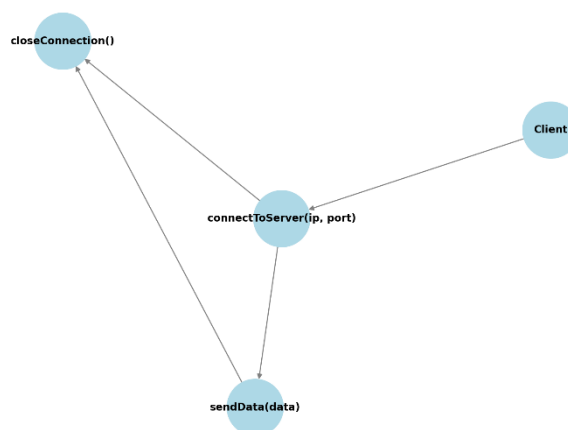


Figure 27: Functionality Graph of Client Component

10.1.2

Server Architecture

The server component in the network architecture serves as the central module responsible for managing client connections and facilitating data exchange and performs the following essential functions:

1. Initialization and Startup:

The server initializes by creating a socket using the AF_INET protocol family for IPv4 communication and the SOCK_STREAM type for TCP-based communication. It binds this socket to a specified port and sets up a listening state to accept incoming client connections. Upon successful initialization, the server transitions to an active state, ready to handle client requests.

2. Client Connection Management:

The server includes an `acceptClient()` method that facilitates the acceptance of incoming connections. This function waits for a client to connect, establishing a dedicated communication link once the connection is accepted.

3. Data Reception:

The server is equipped with a `receiveData()` method to handle incoming data transmissions from the connected client. This method reads data into a buffer, processes it, and handles disconnection scenarios gracefully. By implementing error detection and handling data reception, the server ensures reliable communication and continuous operation.

4. Shutdown and Resource Management:

The server includes a `stopServer()` method to terminate client connections, release resources, and deactivate the server. This controlled shutdown process ensures system stability and prevents resource leaks, maintaining the integrity of the network architecture.

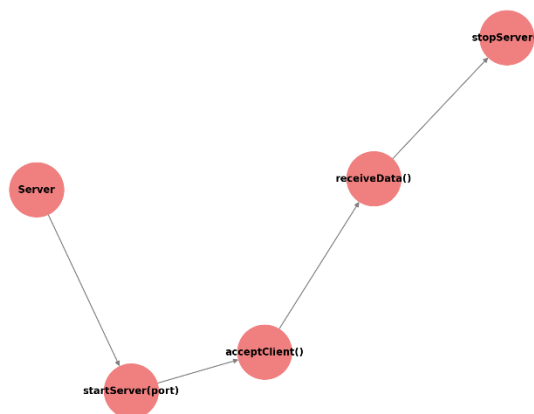


Figure 28: Functionality Graph of Server Component

10.1.3

Simulation's Role in Communication

The simulation integrates both server and client functionalities to facilitate bidirectional data exchange, ensuring uninterrupted interaction between the virtual environment and the physical haptic mechanism. The components are described as follows:

1. Simulation Server:

- **Listening for Data:** The simulation server continuously monitors incoming data from the haptic mechanism's client.
- **Receiving Position Updates:** It processes real-time updates regarding the end-effector's coordinates, enabling precise tracking of the mechanism's movement.
- **Synchronizing Positions:** The server ensures that the position of the virtual lancet in the simulation corresponds accurately to the physical motion of the haptic mechanism, providing an immersive and responsive user experience.

2. Simulation Client:

- **Force Data Transmission:** The simulation client transmits force data components (F_x , F_z), derived from the cutting interactions to the mechanism's server for feedback purposes.
- **High-Resolution Feedback:** Operating at a rate of 200 Hz (one update every 5 ms), the client provides high-frequency force feedback, ensuring a realistic and responsive haptic experience for the user.

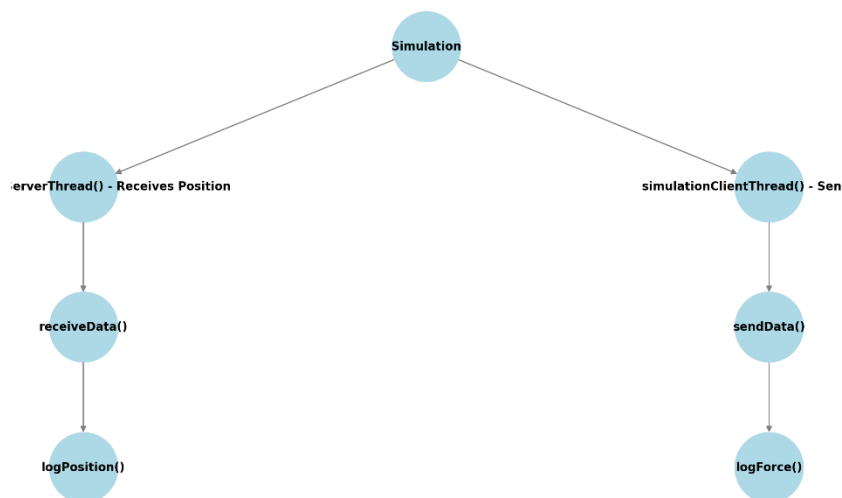


Figure 29: Network Functionality Graph of Simulation

10.1.4

Mechanism's Role in Communication

The haptic mechanism employs a dual client-server setup too to enable interaction with the simulation, ensuring real-time synchronization and accurate feedback during operations. The components of this setup are outlined as follows:

1. Mechanism Client:

- **Transmission of Position Data:** The mechanism client continuously transmits the current position data, such as the x, y, and z coordinates of the end-effector, to the simulation server.
- **Real-Time Synchronization:** By providing real-time updates, the client ensures precise alignment between the physical movements of the mechanism and the virtual environment, enabling accurate replication of the end-effector's trajectory.

2. Mechanism Server:

- **Reception of Force Data:** The mechanism server receives force data (F_x , F_z) from the simulation client, which represents the interaction forces calculated during the cutting process in the virtual environment.
- **Processing for Tactile Feedback:** The server processes the received force data and uses it to generate tactile feedback through the haptic mechanism. This feedback simulates the physical interaction of the virtual lancet with the simulated skin, providing the user with a realistic and immersive experience.

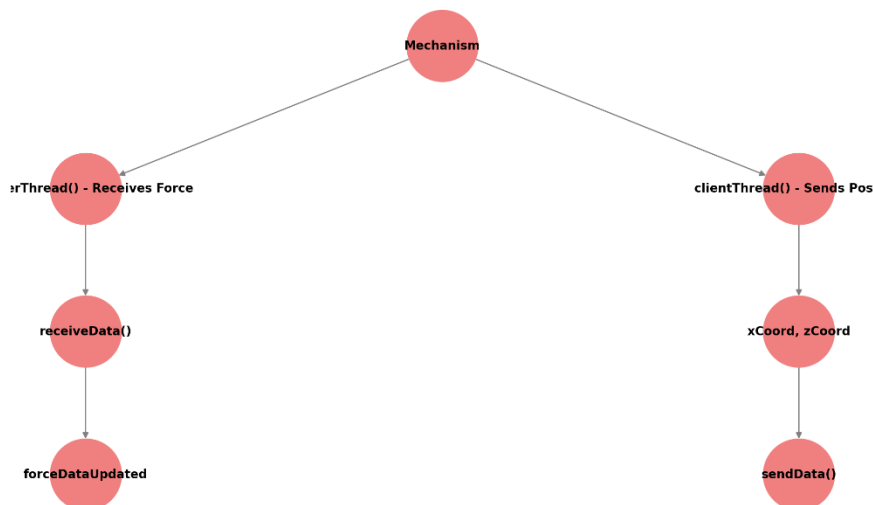


Figure 30: Network Functionality Graph of Mechanism

10.1.5

Data Flow Overview

The system's data flow is structured around a robust client-server architecture to ensure seamless interaction between the haptic mechanism and the virtual simulation. This architecture supports two primary data flows:

1. Position Data Flow (Mechanism → Simulation):

- The mechanism's client transmits real-time position data of the end-effector (x, y, z coordinates) to the simulation's server.
- The simulation synchronizes the virtual lancet's position with this input, maintaining alignment between the physical device and the virtual environment.

2. Force Data Flow (Simulation → Mechanism):

- The simulation's client sends calculated force data (F_x , F_z) to the mechanism's server.
- These forces are processed by the mechanism to generate tactile feedback, allowing the user to experience realistic interactions during cutting simulations. This is achieved through the Jacobian Matrix we computed in previous chapter which converts the forces produced from the simulation to torques which are then applied to the haptic mechanism's actuators. Subsequently, the links pose a resistance to the user's hand equal to the generated force

By decoupling position updates and force feedback into independent threads, the system ensures smooth, real-time performance with minimal latency. This separation supports a communication frequency of 200 Hz, critical for high-fidelity haptic feedback.

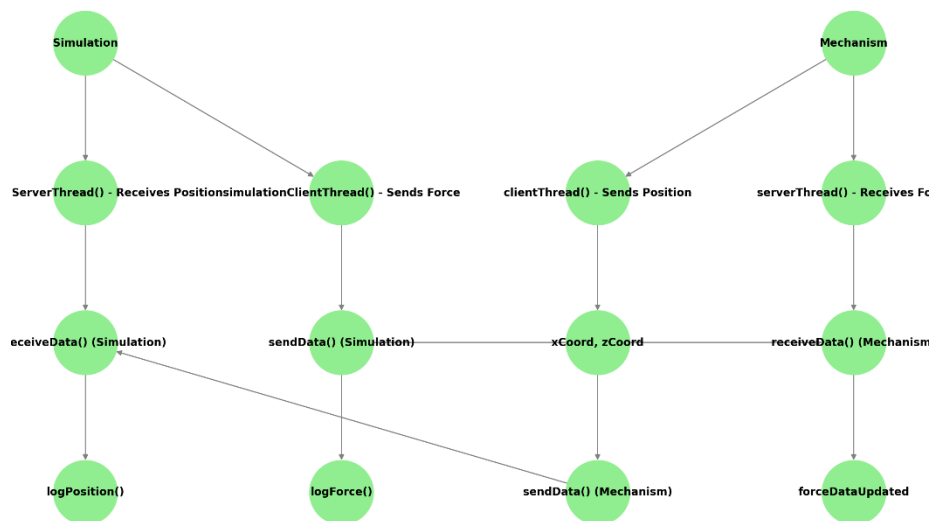


Figure 31: Overall Network Connection Diagram

10.2 Challenges and Optimizations

In preparation for the upcoming tests, several challenges emerged that highlighted critical issues in the design of each component. These challenges underscored the necessity for improved implementation strategies and the incorporation of advanced optimization techniques.

10.2.1 Latency and Timing

The implementation of a high-speed and reliable network is paramount to maintaining synchronization between the simulation and the physical mechanism. Any latency, interruptions, or lag could result in misalignment, compromising the fidelity of the system. This risk is especially critical in scenarios such as real-life surgical operations, where the surgeon relies on precise haptic feedback while interacting with the patient through a robotic mechanism.

Due to the potential unreliability and lower speeds of Wi-Fi connections, the use of Ethernet cables is strongly recommended to achieve consistent and high-bandwidth communication.

10.2.2 Synchronization

Message counters and debugging tools are utilized to monitor and ensure that the rate of data sent by the simulation matches the rate of data received by the mechanism, and vice versa. These measures prevent potential desynchronization, which could degrade system performance or lead to inaccurate interactions between the virtual and physical components.

10.2.3 Performance

To minimize latency and optimize system responsiveness, the processes for transmitting position data and force feedback are executed in separate threads. This decoupled architecture ensures that both data streams operate independently, reducing delays and maintaining real-time updates at a consistent rate of 200 Hz. This threading approach contributes to the system's ability to provide smooth and uninterrupted communication, ensuring precise control and realistic haptic feedback.

These measures collectively enhance the reliability, synchronization, and overall performance of the system, ensuring robust and responsive operation in high-stakes applications.

Chapter 11. VIRTUAL MECHANISM

Before conducting laboratory tests on the components, it is essential to ensure that a reliable connection is established between the simulation and the mechanism using the client and server structures. Given that the physical haptic mechanism can only be utilized in the lab environment, an alternative approach is required to test the communication setup beforehand. This preliminary test ensures that the simulation can successfully connect, receive, and transmit data to any mechanism it interfaces with, thereby verifying the robustness of the communication architecture [BCR94].

To achieve this, we develop a virtual mechanism, implemented as a C++ script. This script is based on the Python code previously employed to validate the accurate adjustment of the end-effector's position in response to changes in the motor angles. By maintaining consistency in programming language and architecture, the virtual mechanism mirrors the design of the simulation's client and server modules. The virtual mechanism is equipped with specific attributes, enabling it to simulate the behavior of a real mechanism effectively and to facilitate comprehensive testing of the communication framework under controlled conditions.

11.1 Forward Kinematic Formulas

The formulas discussed and analyzed in **Chapter 9** are utilized to control and manipulate the movement of the end-effector by adjusting the rotational angles θ_1 and θ_4 of the motors. This process is facilitated through keyboard input, allowing precise and intuitive control over the end-effector's position within the simulation.

Specifically, pressing the up or down arrow keys increments or decrements the y-coordinate, while the right and left arrow keys control the increment or decrement of the x-coordinate. These controlled adjustments translate into effective movement of the

end-effector in the three-dimensional space, enabling the user to navigate and interact with the simulation environment.

This method ensures a high level of user control and responsiveness, allowing for accurate interaction with the simulation's skin mesh. By dynamically updating the end-effector's position in real time, the system provides a reliable and practical approach for testing and demonstrating the functionality of the simulation, including its ability to simulate realistic cutting and deformation scenarios.

11.2 Client

The client component is initialized and configured at the start of the program to establish a connection with the simulation server, which actively listens on port 8080 at a specific IP address. To achieve the required frequency of data exchange, the client operates on a dedicated thread, ensuring that communication remains efficient and uninterrupted.

If the client is unable to connect to the simulation server after a predefined maximum number of attempts, the program displays an error message and terminates the connection process to prevent indefinite retry loops. Upon successfully establishing the connection, the client transmits position data of the end effector, formatted as a UTF-8 encoded string. This position data is calculated using the forward kinematics formulas to ensure precise and accurate representation of the end effector's spatial coordinates.

The client is designed to transmit data at a fixed rate of 200 Hz, aligning with the simulation's force calculation frequency. This synchronization ensures integration between the physical mechanism and the virtual environment, maintaining real-time responsiveness and accuracy in data exchange. By operating within this high-frequency framework, the client contributes to the overall system's reliability and performance, supporting precise haptic feedback and interaction capabilities.

11.3 Server

The server component is also initialized at the start of the program and configured to listen for incoming connections on port 9090 at a specific IP address. In the event of a failure during initialization, such as an inability to bind to the specified port, the program generates a clear and detailed error message to inform the user of the issue. This ensures that connection problems are promptly identified and addressed.

Upon successful initialization, the simulation's client establishes a connection with the server and begins transmitting force data. The server is designed to continuously listen to this incoming data stream, ensuring reliable reception and minimal latency. The force data, calculated in the simulation based on interaction dynamics, is critical for subsequent processing in the laboratory environment.

At the lab, the received force data is used to control the motors of the haptic mechanism. By converting the force values into corresponding torque outputs, the server enables the haptic mechanism to apply precise forces to the user's hand, simulating the physical interaction with the virtual environment. This functionality is integral to providing realistic haptic feedback and enhancing the overall system's effectiveness in replicating tactile sensations during cutting and manipulation tasks.

11.4 Window Interface

The program features a graphical window interface designed to monitor and display critical data exchanged between the simulation and the mechanism in real time. The monitored data includes the following parameters:

1. Received Fx (Rcvd Fx):

- Displays the value of the Fx force component generated by the simulation and transmitted to the mechanism's server. This force corresponds to the horizontal interaction dynamics during cutting or manipulation tasks.

2. Received Fz (Rcvd Fz):

- Displays the value of the Fz force component generated by the simulation and transmitted to the mechanism's server. Although it corresponds to the y-coordinate of the mechanism's end effector, it is labeled as Fz to align with the simulation's coordinate system, ensuring consistency and clarity in data representation.

3. Received Message Count (Rcvd Msg Count):

- Tracks the total number of force data messages received from the simulation. This metric is essential for ensuring reliable communication and identifying potential data loss or transmission issues.

4. Sent Message Count (Sent Msg Count):

- Tracks the total number of position data messages sent by the mechanism's client to the simulation's server. This count provides a measure of the data flow consistency and confirms the mechanism's real-time communication with the simulation.

This interface serves as a vital tool for monitoring and validating the integrity of the data exchange processes, ensuring synchronized operation between the virtual simulation and the physical haptic mechanism. It also facilitates debugging and performance evaluation during both development and laboratory testing phases.

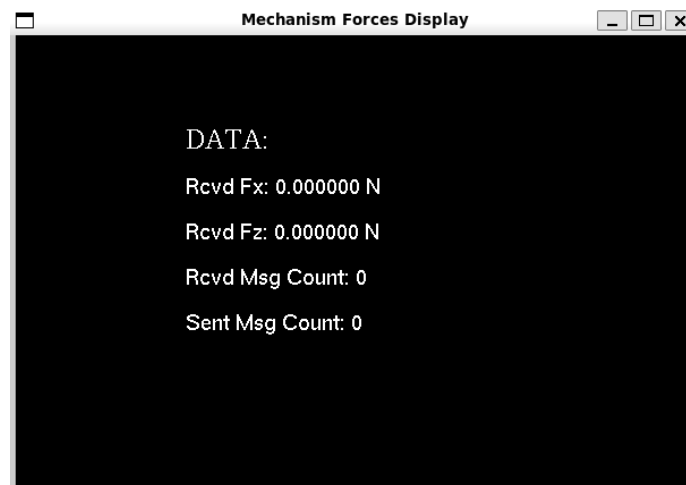


Figure 32: Window Interface of Virtual Mechanism

11.5

Testing The Connection

With the completion of the virtual mechanism, the connection between it and the simulation can now be thoroughly tested. The testing process begins by opening two WSL (Windows Subsystem for Linux) command-line terminals on the same or on different computers. In the first terminal, the simulation file is compiled, while in the second terminal, the mechanism file is compiled. Subsequently, the executable files are launched: the `./mechanism` executable is initiated first, entering a waiting state until the simulation is started by running the `./simulation` executable.

For the connection to be established, the ports and IP addresses of both programs must align. If both the virtual mechanism and the simulation operate on the same machine during testing, the same local inet IP address must be used for both. If there is a mismatch in port or IP configuration, one or both programs will terminate, displaying an error message to indicate the issue.

Once the connection is successfully established, the end effector of the virtual mechanism, represented as the cutting tool within the simulation, can be moved. The interaction between the two components is observed to function as expected: the virtual lancet precisely interacts with the simulated skin mesh, deforming it as intended. This satisfying outcome is achieved after fine-tuning the relationship between the keyboard-controlled movement of the lancet and the corresponding behavior within the simulation.

The results are particularly compelling when the virtual lancet penetrates the epidermis below the melanoma. This demonstrates the effectiveness of the virtual mechanism and simulation integration, highlighting its potential for realistic haptic feedback and surgical training applications. Of course, further adjustments need to be made in order to enhance the virtual effect of the skin cut.



Figure 33: Skin cut inflicted by pressing the lancet against it

Chapter 12. FINAL PRODUCT & LAB TESTING

The final design iteration, referred to as v4, represents a significant advancement in the development of the 3D simulation, addressing limitations identified in earlier versions and specifically tailored for laboratory testing and integration with a real-life haptic mechanism. This version incorporates numerous enhancements aimed at improving functionality, performance, and realism, establishing it as the most robust iteration to date.

12.1 Key Features and Improvements

1. User Interface Enhancements:

- A newly introduced control window appears at the simulation's startup, consisting of input fields and a start button. This panel allows users to configure the desired port numbers and IP addresses for establishing a connection with the haptic mechanism.

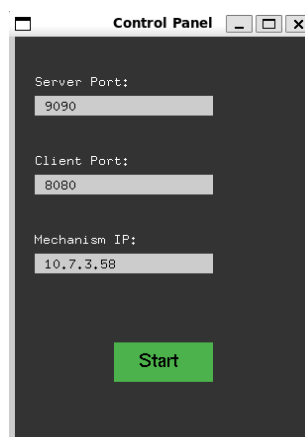


Figure 34: Control Window-Panel

2. Simulation Window and Rendering Performance:

- A regulated 1400x1200 simulation window runs at a consistent 60 frames per second (FPS), ensuring smooth and responsive visuals.
- User perceived force calculations have been decoupled from the rendering loop, resulting in improved computational performance and more efficient data processing.

3. Skin Mesh and Camera Configuration:

- The simulation initializes the skin mesh layers, including the internal VertexQuad structures of the epidermis, ensuring accurate modeling of tissue deformation and cutting.
- The camera is automatically centered on the mesh for optimal visualization, providing users with a clear view of the cutting process.

4. Lancet and Haptic Integration:

- The lancet mesh, featuring a metallic texture, is aligned with the skin mesh to match the real-life haptic mechanism's end-effector position (x: -0.04, y:0.19 cm).
- The lancet can move in 3D space using haptic inputs by parsing the position data messages, supporting both horizontal and vertical movements.
- Advanced functionality includes detecting the closest VertexQuad to the lancet tip & blade, applying pressure along the z-axis to deform the epidermis, and splitting the mesh when a defined threshold is exceeded, creating realistic cutting effects.

5. Force Calculation and Logging:

- User perceived forces (F_x and F_z) are calculated at a rate of 200 Hz during both vertical and horizontal cutting motions.
- These forces along with lancet velocity and position are logged for later analysis and visualization using a Python-based plotting tool, enabling detailed performance assessments.
- Visual effect forces are calculated at a steady rate, the same as the FPS rate.

6. Toolbars and Monitoring Features:

- A side toolbar menu includes button options to activate or deactivate the lancet, lock or unlock the camera, and toggle a help box and a guiding cutting box.
- A top toolbar menu monitors critical parameters such as elapsed time, messages sent and received, and the FPS rate, providing real-time insights into system performance.

7. Client-Server Communication:

- The client-server structure facilitates bi-directional communication. The simulation's client transmits force data every 5 ms, while the server logs received messages, ensuring real-time synchronization of position and force data.

8. Realism and Performance Enhancements:

- A static blood trail simulation activates when the lancet penetrates the epidermis, visually rendering the trail across the cutting path for increased realism.
- The VertexQuad splitting mechanism is fine-tuned to closely replicate the behavior of real-life epidermis cutting.

9. Optimization Techniques:

- Parallelization of blade vertex calculations, caching results, and filtering of VertexQuad meshes reduce computational overhead, lowering the update cycle time to under 5 ms.
- An indexed grid streamlines epidermis mesh partitioning and improves the efficiency of parallel closest VertexQuad to the blade searches.

10. Lancet Direction Computation:

- A custom integrated tool computes the direction of the lancet's movement to ensure accurate cutting based on the real end-effector's trajectory.
- Direction computation includes two filters:
 - A smoothing movement filter minimizes abrupt lancet flicks within the simulation.
 - A position stabilization filter addresses critical issues, such as:

1. Stabilizing the mechanism during directional changes (e.g., switching from cutting right to cutting left).
2. Preventing unintended force application when the lancet remains inside the epidermis after the user halts movement, ensuring only frictional forces are exerted until cutting resumes.

Smoothing movement filter:

$$P_{smoothed} = \alpha P_{current} + (1 - \alpha)P_{previous}$$

where:

- $P_{smoothed}$ is the new filtered position.
- $P_{current}$ is the current position input.
- $P_{previous}$ is the previous filtered position.
- α is the smoothing factor, controlling how much weight is given to the new position versus the historical position. In our case, $\alpha = 0.055$.

Position stabilization filter:

1. Compute Displacement Components

The change in position along the X and Z axes:

$$\Delta X = x_{current} - x_{prev}$$

$$\Delta Z = z_{current} - z_{prev}$$

2. Compute Absolute Displacements

$$|\Delta X| = abs(\Delta X)$$

$$|\Delta Z| = abs(\Delta Z)$$

3. Define Movement Threshold

A small threshold ϵ is used to filter out insignificant movements:

$$\epsilon = 0.00005$$

4. Check for Movement and Time Duration

If both displacements are below the threshold, the movement is considered negligible.

The time since the last detected movement is computed:

$$T_{duration} = T_{current} - T_{last\ movement}$$

If $T_{duration} > 100$ ms, the direction is set to: $direction = NONE$

5. Determine the Primary Direction of Movement

If the Z-axis displacement is dominant: $|\Delta Z| > |\Delta X|$ then:

$$direction = \begin{cases} BACKWARD, & \text{if } \Delta Z > 0 \\ FORWARD, & \text{if } \Delta Z < 0 \end{cases}$$

If the X-axis displacement is dominant but within a valid range:

$$|\Delta X| > |\Delta Z| \text{ and } 0.005 \leq |\Delta X| < 0.1$$

Then:

$$direction = \begin{cases} RIGHT, & \text{if } \Delta X > 0 \\ LEFT, & \text{if } \Delta X < 0 \end{cases}$$

6. Update the Previous Position

$$x_{prev} = x_{current} \quad z_{prev} = z_{current}$$

12.2 Results

By the end of this iteration, the simulation successfully achieves its intended functionality, incorporating realistic tissue cutting, seamless haptic feedback integration, and an optimized, efficient system that meets both visual and mechanical demands. These enhancements establish the system as a robust platform for testing, experimentation, and practical applications, particularly in surgical training and simulation scenarios.

The promising results of a lab experiment are illustrated below. The diagram depicts the position and velocity of the cutting tool, along with the force experienced by the user while interacting with the virtual tissue.

Initially, the user encounters vertical resistance as the virtual cutting tool penetrates the tissue. Once the tool has cut through, the user moves their hand left and right to perform a horizontal cut, experiencing resistance along the x-axis. Finally, the tool is withdrawn, and the simulation is successfully completed.



Figure 35: Results of Final Product Experiment

Below is a screenshot depicting the wound created by pressing the lancet against the skin mesh and penetrating it. The resulting wound accurately reflects the dimensions of the lancet and the depth of penetration. Additionally, small static blood stains are visible, formed as a consequence of cutting the virtual skin and subsequently removing the lancet.

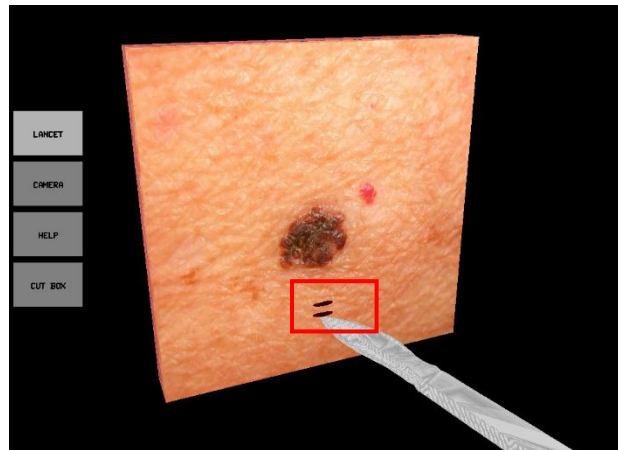


Figure 36: Wound Inflicted by Penetrating the Skin Followed by Static Blood Stains

12.3 Conclusion

This study has demonstrated the development and implementation of a physics-based simulation for skin surgery, integrating a haptic mechanism for realistic interaction. Through the modeling of soft tissue deformation, cutting mechanics, and visual feedback, the system provides an immersive and scientifically grounded framework for surgical training and research.

The findings highlight the challenges and solutions in simulating complex biomechanical properties, emphasizing the importance of precise force rendering, real-time feedback, and accurate wound formation. Future work may focus on enhancing the realism of tissue responses, incorporating dynamic bleeding effects, and refining the interaction model to further bridge the gap between virtual and physical surgical environments.

This research contributes to the advancement of medical simulation technology, offering a valuable tool for both training and scientific exploration.

Βιβλιογραφία

[BBC+99]

P.A. Bernstein, T. Bergstraesser, J. Carlson, S. Pal, P. Sanders, D. Shutt. Microsoft Repository Version 2 and the Open Information Model. *Information Systems*, 24(2), pp. 71-98, 1999.

[BCR94]

V.R. Basili, G. Caldiera, H.D. Rombach. The Goal Question Metric Approach. *Encyclopedia of Software Engineering*, pp. 528-532, John Wiley & Sons, Inc., 1994.

[DL95]

D. Lee. *Integration of Haptic Feedback in Soft-Tissue Simulations*. *IEEE Transactions on Robotics and Automation*, 11(3), pp. 390-400, 1995.

[KVG22]

K. Vlachos, G. Tsakiris. Real-Time Soft Tissue Modeling in Surgical Simulations. *Journal of Medical Robotics*, 18(4), pp. 125-138, 2022.

[KL20]

K. Lampert. Advances in Real-Time Spring-Mass-Damper Models for Surgical Training. *International Conference on Medical Physics*, pp. 55-67, 2020.

[SLR19]

S. Liu, L. Roberts. Enhancing Haptic Feedback Systems for Virtual Environments. *Virtual Reality Science*, 14(2), pp. 98-115, 2019.

[AH09]

K. E. Atkinson, W. Han. *Theoretical Numerical Analysis: A Functional Analysis Framework* (3rd ed.). Springer, 2009.

[ZM99]

M. Zhang, A. F. Mak. *In vivo friction properties of human skin*. *Prosthetics and Orthotics International*, 23(2), pp. 135-141, 1999.

[CS00]

G. C. Burdea. Haptic feedback for virtual reality. *Annual Reviews in Control*, 24(1), pp. 225-236, 2000.

[BS02]

R. D. Howe, D. A. Kontarinis, W. J. Peine. Haptic interfaces for teleoperated and virtual

environments. *Journal of Dynamic Systems, Measurement, and Control*, 124(4), pp. 468-478, 2002.

[HC04]

V. Hayward, O. R. Astley, M. Cruz-Hernandez, D. Grant, and G. Robles-De-La-Torre.

Haptic interfaces and devices.

Sensor Review, 24(1), pp. 16–29, 2004.

[OKB13]

H. O'Malley, S. K. Gupta, and M. K. Benali-Khoudja.

Haptic interfaces: theory and applications.

IEEE Robotics & Automation Magazine, 20(2), pp. 36–46, 2013.

[CHB93]

G. Campion, Q. Wang, and V. Hayward.

The Pantograph Mk-II: A Haptic Instrument.

Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 193–198, 1993.