Modeling for Haptics-based Surgical Simulation

By


A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada K1S 5B6

May 2006

© Copyright 2006, Weiyu Zheng
ABSTRACT

With advances in robotics, computer graphics, and modelling technologies, the Virtual reality and Haptics based Surgery Simulator is rapidly emerging and increasingly perceived as a valuable addition to, or replacement of, traditional training methods.

Cutting is an important part of a surgical procedure. In order to render immediately visual feedback when the cut proceeds, we use a progressive cutting method. A temporary particle is created to track the position of the haptic device in the virtual environment.

Simulating soft tissues is a significant part of surgical simulation. The mass spring system is applied to model the soft tissue. Based on two criteria - computation efficiency and precision - this thesis analyzes three different integration algorithms and employs the Leapfrog Verlet method. Solving differential equations for each particle requires expensive computation. This thesis proposes to limit the computation within a “zone of influence”.

This thesis also proposes a haptic force model to represent the force feedback to the user. The model includes a back-propagated neural network algorithm and a white-Gaussian noise model. We divide the cutting force into three parts: the indenting force, which is simulated by a back-propagated algorithm; the organ force, which is represents by a spring model; and other forces, which is described by a white-Gaussian noise model.
ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor Dr. Peter X. Liu, for his guidance, patience and encouragement for my research.

I would like to thank all the graduate students in my lab for their helpful discussions.

I would like to thank my mother, father and brother, for their support whenever I am in need.

Finally, I would like to thank my husband, Chengliang Li, for his constant support and encouragement during my study.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iii</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td>x</td>
</tr>
<tr>
<td>CHAPTER 1  INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Haptics</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Virtual Reality and Haptics-based Surgical Simulator</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Problem Statement</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Contributions</td>
<td>6</td>
</tr>
<tr>
<td>1.6 Thesis Outline</td>
<td>7</td>
</tr>
<tr>
<td>CHAPTER 2  BACKGROUND REVIEW</td>
<td>10</td>
</tr>
<tr>
<td>2.1 VHS Framework</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Deformation Model</td>
<td>13</td>
</tr>
<tr>
<td>2.2.1 Surface-based Models</td>
<td>14</td>
</tr>
<tr>
<td>2.2.2 Volumetric Models</td>
<td>15</td>
</tr>
<tr>
<td>2.2.3 Mass Spring System</td>
<td>17</td>
</tr>
<tr>
<td>2.2.4 Finite Element Method</td>
<td>20</td>
</tr>
<tr>
<td>2.3 Tissue Cutting</td>
<td>23</td>
</tr>
<tr>
<td>2.4 Haptic Force Feedback</td>
<td>24</td>
</tr>
<tr>
<td>CHAPTER 3  SYSTEM OVERVIEW</td>
<td>26</td>
</tr>
<tr>
<td>3.1 Framework</td>
<td>26</td>
</tr>
<tr>
<td>3.2 System Diagram</td>
<td>28</td>
</tr>
<tr>
<td>CHAPTER 4  PROGRESSIVE CUTTING</td>
<td>30</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER 7 CONCLUSIONS ...................................................................................... 84
  7.1 Contributions........................................................................................................ 84
  7.2 Future Work ........................................................................................................... 86
REFERENCES .................................................................................................................. 88
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Virtual reality and haptics-based surgical simulator</td>
<td>2</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>Basic architecture of a VHS application</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Tightly coupled framework</td>
<td>10</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Loosely coupled framework</td>
<td>13</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Mass spring method</td>
<td>17</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Framework of our simulator</td>
<td>27</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>System diagram</td>
<td>29</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Interaction loop of the indentation procedure</td>
<td>31</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Interaction loop of the moving procedure</td>
<td>32</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Example of cutting operation</td>
<td>33</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>Example of progressive cutting without temporary subdivision</td>
<td>35</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>Example of progressive cutting with temporary subdivision</td>
<td>36</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>Four different progressive cutting cases</td>
<td>38</td>
</tr>
<tr>
<td>Figure 4.7</td>
<td>Mapping from the world coordinates to workspace coordinates</td>
<td>39</td>
</tr>
<tr>
<td>Figure 4.8</td>
<td>Progressive cutting algorithm</td>
<td>41</td>
</tr>
<tr>
<td>Figure 4.9</td>
<td>Progressive cutting experiment</td>
<td>41</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>Response for the pulse signal in MSS.</td>
<td>46</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Surface mesh model</td>
<td>47</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>Mesh model</td>
<td>57</td>
</tr>
</tbody>
</table>
Figure 5.4  Applied perpendicular force (100 particles) ................................................. 58
Figure 5.5  Displacement of particles in different rings (100 particles) ......................... 58
Figure 5.6  Applied perpendicular force (400 particles) ................................................. 59
Figure 5.7  Displacement of particles in different rings (400 particles) ......................... 59
Figure 5.8  Wire frame of the heart ................................................................................... 60
Figure 5.9  Undeformed model of the heart ..................................................................... 61
Figure 5.10 Deformed model of the heart ........................................................................ 61
Figure 6.1  Architecture of neuron .................................................................................... 63
Figure 6.1  Architecture of a BP network ......................................................................... 64
Figure 6.3  Displacement in indenting procedure ............................................................ 67
Figure 6.4  Three cases in moving procedure .................................................................. 69
Figure 6.5  Liver indentation tests (adopted from [53]) ................................................... 71
Figure 6.6  Raw experimental data for liver cutting (adopted from [56]) ....................... 72
Figure 6.7  Topology of simulation BP neural network .................................................. 73
Figure 6.8  Experimental data vs. trained neural network data ....................................... 74
Figure 6.9  NN’s performance ........................................................................................... 74
Figure 6.10 Experimental data vs. trained NN data under both conditions .................... 76
Figure 6.11 Force vs. distance from constrained edge of specimen ............................. 78
Figure 6.12 Force vs. displacement during rabbit heart testing ...................................... 80
Figure 6.13 Force vs. displacement during porcine heart testing .................................... 82
LIST OF TABLES

Table 4.1  Enumeration of new elements added in different cases ..................... 37
Table 5-1  Precision of integration algorithms ...................................................... 54
Table 5-2  Time steps vs. integration algorithms (1) .............................................. 55
Table 5-3  Time steps vs. integration algorithms (2) .............................................. 55
Table 5-4  Time steps vs. integration algorithms (3) .............................................. 55
Table 6-1  Trained weights and biases ................................................................. 75
Table 6-2  Trained weights and biases ................................................................. 75
**LIST OF ABBREVIATIONS**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AABB</td>
<td>axis aligned bounding box</td>
</tr>
<tr>
<td>BP</td>
<td>back-propagated</td>
</tr>
<tr>
<td>BSP</td>
<td>binary space partition</td>
</tr>
<tr>
<td>CAM</td>
<td>control algorithm module</td>
</tr>
<tr>
<td>CDM</td>
<td>collision detection module</td>
</tr>
<tr>
<td>DCM</td>
<td>deformation computation module</td>
</tr>
<tr>
<td>DE</td>
<td>differential equation</td>
</tr>
<tr>
<td>DOF</td>
<td>degree of freedom</td>
</tr>
<tr>
<td>FEM</td>
<td>finite element method</td>
</tr>
<tr>
<td>FRM</td>
<td>force response module</td>
</tr>
<tr>
<td>FSM</td>
<td>finite sphere method</td>
</tr>
<tr>
<td>GD</td>
<td>graphic display</td>
</tr>
<tr>
<td>GDE</td>
<td>global discretization error</td>
</tr>
<tr>
<td>HI</td>
<td>haptic interface</td>
</tr>
<tr>
<td>HRM</td>
<td>haptic rendering module</td>
</tr>
<tr>
<td>I/O</td>
<td>input/output</td>
</tr>
<tr>
<td>JND</td>
<td>just-noticeable-difference</td>
</tr>
<tr>
<td>LDE</td>
<td>local discretization error</td>
</tr>
<tr>
<td>MIS</td>
<td>minimally invasive surgeries</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>MSE</td>
<td>mean of squared errors</td>
</tr>
<tr>
<td>MSS</td>
<td>mass spring system</td>
</tr>
<tr>
<td>NN</td>
<td>neural network</td>
</tr>
<tr>
<td>OBB</td>
<td>oriented bounding box</td>
</tr>
<tr>
<td>PCMFS</td>
<td>point collocation-based method of finite spheres</td>
</tr>
<tr>
<td>RK-4</td>
<td>the fourth-order Runge-Kutta</td>
</tr>
<tr>
<td>RTS</td>
<td>real-time simulation</td>
</tr>
<tr>
<td>SISO</td>
<td>single input single output</td>
</tr>
<tr>
<td>SNR</td>
<td>signal-to-noise ratio</td>
</tr>
<tr>
<td>VHS</td>
<td>virtual reality and haptics-based surgical simulator</td>
</tr>
<tr>
<td>VR</td>
<td>virtual reality</td>
</tr>
<tr>
<td>VRM</td>
<td>virtual rendering module</td>
</tr>
</tbody>
</table>
CHAPTER 1 INTRODUCTION

1.1 Motivation

Surgical training has undergone many changes in the last decade. Traditionally, surgical training is based on three methods: greenhorn/master setting, cadavers and animals. In the greenhorn/master setting, the expert operates on actual patients while the trainee is watching. Although it is effective, it has serious drawbacks. Because the novice trainee has no experience to perform procedures, patients who the novice trainee operates on will be subject to risks. Cadaveric training presents realistic anatomy. Nevertheless, tissue responses are affected by storing techniques and temperatures. Additionally, it may not reflect mechanical properties of living tissues. The limited supply of cadavers makes the training cost very high. Animals have different anatomy from human beings. Training using animals is very expensive as well. After dissecting, they have to be discarded. Moreover, there exist ethical problems.

The virtual reality and haptics-based surgical simulator (VHS) potentially offers an excellent solution to these problems. The VHS can be programmed to simulate various pathologies and surgical procedures. It offers a safe, learner-centered way for the trainee to learn surgical skills through repeated practice. The performance of trainees can be
stored and reviewed by experts later on. Since the force and trajectory profiles of experts are significantly different from novices, the training system can record the experts’ force and trajectory to train the novices’ motor-control skills. If trainees diverge from the trajectory, a correcting force can be used to pull them back [1]. Moreover, the virtual environment is reusable and even reconfigurable, so the training cost can be significantly cut down.

Figure 1.1 shows a typical VHS configuration. When the trainee manipulates the surgical instruments and interacts with virtual soft tissues, the screen displays the simulated operation field of the virtual patient, and haptic devices feed back the reaction force to the trainee.

![Virtual reality and haptics-based surgical simulator](image)

Figure 1.1 Virtual reality and haptics-based surgical simulator
1.2 Haptics

Haptics originated from the Greek word “haptesthai”, which means “to touch”. Haptic feedback usually refers to force sensing. More precisely, it refers to the sense of differential forces (or displacements) on the skin in time and in space. “Haptic rendering is the process of computing and generating forces in response to user interactions with virtual objects.” [4]

Since the early 20th century, haptics has been used to label the subfield that focuses on human touch-based perception and manipulation. In 1970s and 1980s, research efforts in robotics began to focus on haptics. In the early 1990s, the confluence of several emerging technologies made virtualized haptics possible.

Haptic feedback can be categorized into tactile (cutaneous) feedback and kinesthetic (force) feedback. Tactile feedback provides information about surface features of the touched object such as geometry, texture, slippage, etc. Kinesthetic feedback provides information about the position and motion of the user’s hand and arm relative to the object. In general, tactile and kinesthetic sensations occur simultaneously. Unlike other I/O channels, human’s haptic interaction with the environment is bidirectional, with energy and information flowing both to and from the user [3], which is very different from the cases of unidirectional audio and visual channels. Using a haptic interface, a user can touch, feel and manipulate a simulated virtual object.
In this thesis, we will mainly focus on kinesthetic (force) feedback. Kinesthetic feedback is stimulated by bodily movements. The update frequency is approximately 30-50 Hz and the force just-noticeable-difference (JND) is approximately 7% [5].

1.3 Virtual Reality and Haptics-based Surgical Simulator

Human beings interact with the external world through all five senses: vision, auditory, haptics, smell and taste. Traditional virtual reality (VR) applications integrate only a smaller subset, typically vision and hearing. With the development of haptics, now VR can also rely on the sense of touch. Figure 1.2 shows the architecture of a VHS application which has visual, auditory and haptic feedback. It consists of the following elements [6]:

1) Simulation engine: Its main function is to compute behaviors of virtual objects.

2) Audio-visual and haptic rendering algorithms: They compute the virtual environment’s graphic, sound and force responses to the user.

3) Video, audio and haptic devices: Those devices covert visual, audio and haptic signals into the form that the user can perceive. The user feels the video and audio feedback by audio and visual displays, such as speaker, computer screen or head-mounted display. To perceive haptic feedback, the user typically holds or wears a haptic interface device. The force conveyed to the user by the haptic device gives
the user a sense of the natural interaction with the virtual environment and objects within it.

![Diagram of Basic architecture of a VHS application](image)

**Figure 1.2** Basic architecture of a VHS application

### 1.4 Problem Statement

Although VHS is very promising for training surgical students, constructing such a system is a challenging task.

A realistic deformable model should be developed to imitate behaviors and properties of living tissues or organs. The model needs to compute effectively in order to visualize deformations in a real-time fashion. In general, there are two methods for modeling deformations: Finite Element Method (FEM) and Mass Spring System (MSS), each of which has its own advantages and disadvantages.
Cutting is a critical part for most general surgical procedures. If cutting occurs, the geometry and topology of the virtual object (tissue or organ) is changed. Real-time visualization of these changes to the user (i.e., surgical resident) is always expected and a new updated graphic model of the virtual object is required. The object deformation is based on the topology modification. New elements of the deformable object are introduced when the scalpel is cutting the soft tissue. The number of new elements will increase very quickly as the object is cut. Finding an efficient technique to update the model is necessary.

VHS contains both graphic and haptic rendering algorithms. The haptic rendering algorithm calculates the force reflected to the user and applied to the visual soft tissue. Although measuring user's motion is easy, determining appropriate display force is a very complex process.

1.5 Contributions

The main contribution of this thesis is the development of a realistic model of haptic force interaction between the surgical tool and soft tissue. The force feedback to the user consists of three parts: the indenting force, which is modeled by a back-propagated (BP) neural network (NN) algorithm, the organ force, which is described by spring model, and other forces, which is simulated by the white-Gaussian noise function. The BP NN
algorithm and the introduction of the white-Gaussian noise is first proposed to model the force feedback in the VHS.

In order to model properties of the soft tissue more realistically, we modify the traditional used MSS, which only use link springs to connect particles in the VHS. Both link and home springs/dampers are applied in our system. In order to test three integration methods' performance in our VHS, we also analyze them and do some experiments in terms of two criteria: computation efficiency and precision.

Other contributions are developing a progressive cutting method with temporary subdivision, and confining the computation area of the deformable object. A temporary particle, which tracks the position of the haptic device in the virtual environment, is applied in the system. A snapping approach is used to minimize the number of adding elements. This method can display the visual feedback with little lag. Confining the computation area within a "zone of the influence" reduces the number of differential equations significantly. Those two methods guarantee good performance of the real-time visual feedback.

1.6 Thesis Outline

The rest of this thesis is organized as follows:
Chapter 2 describes the background and provides a literature survey in this field. There are two existing types of the framework: loosely binding and tightly binding of the haptic and the deformation models. For the deformation of virtual objects, two common physics-based models are usually used: MSS and FEM. The current soft tissue cutting techniques and haptic modeling are also reviewed.

Chapter 3 concentrates on the system framework. In this chapter, the loosely-coupled framework that is employed in this thesis is discussed in detail. The system diagram is illustrated and explained.

Chapter 4 focuses on the progressive cutting procedure. After overviewing the diagram of the cutting procedure, it is explained how to represent the surface of the soft tissue. The progressive cutting method is elaborated and possible topology modification cases are discussed.

Chapter 5 investigates the deformation model. Here, MSS is applied. Model equations are discussed in detail. Based on two criteria: computation efficiency and precision, Leapfrog Verlet, Euler and Runge-Kutta integration algorithms are analyzed and compared. In this chapter, the author also limits MSS within a "zone of influence". Only those particles in the zone of influence are computed.

Chapter 6 proposes a force model. The author divides the cutting procedure into two sub procedures. One is that the scalpel is indenting the virtual tissue, which is modeled by a BP neural network. The other is that the scalpel is moving within the virtual tissue. Here,
the white-Gaussian noise function is applied to model friction and other forces except organ and indenture force.

Chapter 7 draws conclusions from the work presented in this thesis as well as recommendations for future work.
CHAPTER 2 BACKGROUND REVIEW

In this chapter, the author reviews various approaches to the VHS. In terms of framework, research has been done on both loosely and tightly binding of haptic and graphic rendering. Regarding object deformation models, MSS and FEM are mainstays in the literature. Progressive cutting techniques and haptic modeling are reviewed in the remaining part of this chapter.

2.1 VHS Framework

![Tightly coupled framework](image)

**Figure 2.1** Tightly coupled framework
There are two mainstream frameworks of the VHS. One is that the deformation and haptic rendering module are tightly coupled. The other is that they are loosely coupled.

Figure 2.1 shows the tightly coupled framework. It has three basic components: haptic interface (HI), real-time simulation (RTS) and graphic display (GD). They are explained as follows:

1) HI. It is the hardware with which the user physically interacts. These devices are instrumented so that the position and orientation of the device can be measured or calculated, and the force feedback can be rendered to the user as the he or she is interacting with the virtual environment.

2) RTS. The real-time simulation is responsible for computing the interaction between virtual tools (e.g., the scalpel) and virtual objects as the user moves the haptic device. It consists of five modules: collision detection module (CDM), force response module (FRM), control algorithm module (CAM), deformation computation module (DCM) and visual rendering module (VRM).

Avatar is “the virtual representation of the haptic interface through which the user physically interacts with the virtual environment” [6]. The CDM checks collision between avatar and the virtual object and provides detailed information about collision, such as when, where and to what extent the collision (collision point and depth of penetration) has occurred. In the VHS, tools are usually rigid objects while organs are soft tissues, which can be deformed. The topology of the soft
tissue may be changed due to collision. It is thus very different from common collisions between rigid objects. There are three popular techniques to improve the performance of collision detection: axis aligned bounding box (AABB) [7], oriented bounding box (OBB) [8] and binary space partition (BSP) [9].

The FRM describes the force reflected to the user and applied to the visual soft tissue. Although measuring user’s motion is easy, determining appropriate display force is a complex process. In addition to accuracy, stability and realism (e.g. timeliness) are also very important.

The CAM presents force feedback to the haptic device. It approximates the ideal interface force based on the device’s capabilities.

The VRM mainly focuses on the design of advanced computational models for geometric and topological representation of virtual objects.

3) Graphic display (GD). It displays visual feedback during surgical simulation.

For the tightly coupled architecture, the FRM provides the information about the interaction force between avatar and virtual objects to the DCM. Then the DCM computes deformation of virtual objects. In the loosely coupled framework, the DCM and FRM are loosely coupled. They simulate the interactive force on deformable objects respectively. Since deformation models employed are usually linear, it is hard to reflect the nonlinear force-displacement characteristics of soft tissues. In this framework, The
FRM displays the nonlinear force feedback based on the experimental measurements while the DCM and VRM display images based on the characteristics of soft tissue. Figure 2.3 shows the loosely coupled framework [10].

The loosely coupled framework is used in this thesis. After obtaining the collision information, the HRM and DCM calculate the force independently. The force feedback from the HRM is applied to the haptic device while the DCM and VRM calculate the virtual feedback.

![Loosely coupled framework](image)

**Figure 2.2 Loosely coupled framework**

### 2.2 Deformation Model

VHS researchers are interested in so-called “soft” objects, such as human tissues, which deform under applied forces. The deformation model describes behaviors of the soft tissue and its mechanical properties, and visualizes it to the user in a realistic fashion. The update rate of an ideal deformation model should be fast so that the user feels the
animation is smooth. The model should also be realistic. Moreover, it should facilitate cutting and suturing. From the viewpoint of geometric representation of deformable objects, deformation models are divided into two categories: surface-based and volumetric model.

2.2.1 Surface-based Models

Surface-based models represent the exterior of the object. They are based on parametric or polygonal representations. Therefore, it is limited to cases where the complex interior structure needs not be presented.

Terzopolous et al. [11] first introduced the physical-based deformable object simulation based on elasticity theory. Elastically deformable models responds in a natural way to applied forces and constraints, and the description of shape and motion is unified. He and Waters [12] also propose a facial model to describe the detailed anatomical structure and dynamics of human face. Based on three anatomically distinct layers of facial tissue: the dermis, a layer of subcutaneous fatty tissue and the muscle layer, they construct a three-layer mesh of mass points. Their model consists of 6500 springs and is animated at interactive rates.

Wu et al. [13] presents a simplified mesh system to reduce the computation time while visual realism is still maintained.
Song and Reddy [14] demonstrates cutting surface-based models by using FEM in two dimensions. The force exerted by the user is regarded as a nodal force. Cutting occurs once the force exceeds the shear strength of the material.

Thompson and Toga [15] introduces a surface-based approach to warp anatomical images elastically. It can quantify variations in the shape and location of cortical and sub-cortical brain structures.

Surface-based models are easy to manipulate and computationally effective. They provide a simple way to modify the geometry when force is applied. However, they cannot simulate the interior change of the tissue. For example, in the simulation of brain surgery, it is hard to simulate the surgical scenarios that a cutting procedure is performed to remove a tumor from the brain tissue based on the surface-based model.

2.2.2 Volumetric Models

Volumetric models are made of voxels. They have more complex connectivity among geometric primitives.

Mazura and Siefert [16] placed a cutting surface by specifying the beginning and end points of the cutting edge to create a set of triangles, which intersect with the tetrahedral mesh to split the intersected elements. They were able to process a model with 15,000 elements in approximately 6 minutes.
Bielser et al. [17] proposed to cut through a mass spring object by tracking the tip and the direction of the cutting edge through the object. At each time step, the plane between the previous and the current edge positions was created. The cutting action always divided an element into seventeen smaller ones using the midpoint subdivision.

Mor and Kanade [18] generated a minimal set of new elements to replace intersected tetrahedral during progressive cutting.

Radetzky and Nurnberger [19] proposed indirect volume rendering. A segmentation method was introduced to render the surface of the volumetric data, which was visible from the viewpoint of the user. It drastically reduced the required computation time. Another advantage of this method was that it could add visual surface by using texture mapping.

Volumetric models are able to maintain the original volumetric information. It is a natural way to represent 3D images and maintain detailed information about the internal anatomical structure of tissues. However, it is only suitable if the number of volumetric elements is relatively low. The main drawback of volumetric models is that it requires high storage space and high computational power in order to visualize the cutting in real time [20].

As shown before, surface-based models describe only the exterior of an object. For cutting or suturing operations, the physical nature is less accurate but it is computationally effective. On the other hand, volumetric models can represent the
interior of an object, it is thus more accurate, but it requires expensive computation. The choice between surface-based and volumetric models is governed by two factors: computation efficiency and physical accuracy [21].

Deformable models can be also broken down into non-physics based and physics-based ones. Non-physics based models employ purely geometric techniques without considering the physical properties of materials. A typical example of this technique is the free-form deformation. Physics-base models integrate physical properties of the objects, such as pulling, cutting and breaking. For example, the model will exhibit deformation accordingly when force is applied. MSS and FEM are the two widely used physics-based techniques.

2.2.3 Mass Spring System

MSS is the most common model of soft tissues for real time VHS simulations. As Figure 2.3 shows, Deformable objects are discretized into a set of mass nodes. Nodes are

![Mass Spring System Diagram](image)
connected by springs and dampers; and deformation is simulated by dynamics of those mass nodes. Mass nodes move under the influence of internal and external forces. When the external force is imposed on the object, it reaches a new equilibrium of energy.

Based on the simulation software --- KISMET, Kuhnapfel et al. [22] developed a VHS to simulate an endoscope surgery. The soft tissue system was a volumetric MSS and the spring stiffness is a 3rd order polynomial. Their research illustrated that the non-linear shape of the stress-strain curve could be well approximated by the polynomial.

The model of Zhang et al. [24] was also based on the surface-based MSS. A relatively coarse mesh model represented the deformation object, so the number of nodes was reduced and the simulation could be run at a sufficiently high rate. Zhang [23] extended the work in paper [24]. The model simulated virtual dissection by progressive subdivision and re-meshing. He proposed a novel algorithm, the search and split algorithm, to generate interior structures, which showed the cutting result of the interaction between the instrument and virtual tissue.

Gibson et al. [36] [37] proposed the “chainmail” approach. The model governed the displacement of all nodes. Each node was connected with six neighbors. Only when the node displacement exceeded distance thresholds, does the node move in the desired direction. Otherwise, it was frozen. It is suitable for real-time deformation.

MSS is fast and simple to be understood. Moreover, it is easy to be implemented since it does not require continuous parameterization. It describes the virtual object realistically if
deformations are small. Another advantage of MSS is that it can model the cutting or suturing procedure by simply removing or adding vertices and connections. However, it has the following weakness [21]:

1) The topology must be designed carefully. Since vertices are linked by springs and dampers, the global behavior of the virtual object depends heavily on the state of vertices. If the object is under constrained, several rest positions of a vertex are possible and the system can fall into unwanted local minima. On the other hand, if the object is over constrained, the range of deformation is limited.

2) It is difficult to set spring and damper parameters. In addition, it is not easy to compare deformations calculated by MSS with those acquired by biomechanical studies. Nevertheless, Keeve et al. [25] showed that MSS behave similarly to the linear elastic FEM for small deformations. Zhang et al. [24] proposed a scheme to change the values of mass and spring constants to compute the force feedback for the subdivided surface. Deussen et al. [26] identified spring parameters based on the simulated annealing algorithm. Radetzky et al. [66] applied a neural network to adjust spring constraints so that deformations match those acquired from experiments on real tissue.

3) There is a relationship between critical stiffness $K_c$ and time step $\Delta t$ when the system is divergent. The spring stiffness is a function of time step as follows:

$$K_c \approx \frac{u}{\pi^2 (\Delta t)^2} \approx \frac{m_{\text{total}}}{n \pi^2 (\Delta t)^2}$$

(2-1)
Where $n$ is the number of nodes; $m_{\text{total}}$ is the mass of the virtual object. The mass of the node is $u = \frac{m_{\text{total}}}{n}$.

From equation (2-1), we can draw a conclusion that the smaller the time step, the stiffer the spring.

4) For large deformations of the virtual object, the accuracy of MSS drops significantly.

### 2.2.4 Finite Element Method

FEM is defined by shape elements, shape function and energy function. It models mechanical properties --- the stress-strain relationship --- of the virtual object by decomposing the object into a number of elements. The object deformation is then represented by the displacement of nodes in each element. Shape functions interpolate nodal displacements. For dynamic computation, the position, velocity and acceleration of nodes are governed by standard Newtonian mechanics. The basic physical law governing FEM models is follows:

$$M\ddot{U} + C\dot{U} + KU = F \quad (2-2)$$

Where $K$ is the stiffness matrix, $U$ is the vector of node displacement; $M$ is the mass matrix for the mesh; $C$ is the damping matrix; and $F$ is applied external forces.
Bro-Nielsen [27] first proposed the interactive deformation simulation using a FEM approach. He applied linear elasticity on a tetrahedral mesh and the simulation was static. Since internal nodes of the mesh were eliminated from equations, the size of the system was reduced and displacements were computed efficiently.

Cotin et al. [28] developed a tensor mass approach based on linear elastic continuum mechanics to model soft tissue. It was also based on FEM. The stiffness matrix was stored locally at each nodal point. The simulation achieved an update rate of 40Hz with a mesh made up of 760 vertices and approximately 4000 edges. The update rate is similar to the one obtained from a mass-spring system. However, the straightforward implementation is slow. Some work has been done on FEM models to make a trade-off between the accuracy and the updated rate.

Bro-Nielsen et al. [29] developed a FEM system that run at a real-time rate, which was achieved by the use of condensation, pre-calculation of the inversion and exploitation of the sparse structure of the force vector. They assumed that internal nodes, which were not visible or did not interact directly with the user, were not important. Then in most situations, the user was just touching a few of surface nodes at a time. The force vector was thus made up of mostly zeros. They achieved an update rate of 20Hz on models, which had 250 surface nodes, without condensation, and 20Hz with ones, which had 700 nodes, with condensation.
Cotin et al. [30] preprocessed “elementary deformations” derived from a FEM to obtain the real-time computation of the deformation. For a liver model with 1500 nodes, the simulation achieved an update rate of 50Hz for the visual feedback.

De et al. [31] used the Method of Finite Sphere (MFS), which was a meshless numerical scheme, avoiding problems associated with meshing and re-meshing. MFS represented the computational domain by a scattered set of nodes; and used the Galekin formulation to generate discretized versions of partial differential equations, which governed the deformable object’s behavior.

Lim et al. [35] enhanced the realism of virtual cutting by employing a local subdivision algorithm, which localized point collocation-based method of finite spheres (PCMFS). PCMFS computed deformation fields and interaction forces. Without the generation of new primitives, progressive cutting was achieved by a snapping algorithm.

Other methods, such as adaptive FEM [32], hybrid methods [33], modal analysis [34] and so on, were reported to make a trade-off between expensive computation and updated rate.

In summary, comparing with MSS, FEM based approaches are able to model the complex deformation of soft tissue more accurately. Since they incorporate non-linear properties of living tissues, object deformations are more realistic. However, it requires intensive computation. High computation requirement makes it impossible to achieve real-time deformation. The reduction of the number of nodes in turn degrades the realism of the model. Due to its intensive computation requirement, it is very hard to achieve
real-time performance. Calculating the haptic force feedback also increases the complexity of the system. For the procedure of cutting or suturing, the topology of virtual objects has to be re-meshed and the stiffness matrix has to be updated within each time step, making the computation more expensive. Since the size of matrices is large, matrices should be allocated carefully in order to save memory.

### 2.3 Tissue Cutting

One critical procedure for most VHS is the cutting phenomena. When avatar is moving through the soft deformable tissue, real-time and realistic visual and haptic feedback is very important.

Some algorithms process the cutting action only after the cutting procedure is completed. Bielser et al. [17] used five possible topological representations to describe the procedure of cutting a tetrahedron. Three of them represented partial cutting while the other two modeled a complete tetrahedron split. Only edges and faces that are part of the cutting face were considered. After applying the midpoint subdivision approach, an intersected tetrahedron is split into 17 smaller ones. If one of edges had been dissected, the current intersection point was replaced by the edge midpoint. Split edges were substituted by two edges, with two vertices at the new location, while unsplit edges keep one vertex. The algorithm could lead to cracks because new vertices had no link with neighbor tetrahedrons. Because splitting tetrahedrons was carried out after cutting, apparent
latency was introduced, leading to visual inaccuracies and discontinuities. Bielser and Gross [38] extended the work of [17] by reducing the number of elements generated for each cut element. Bruyns et al. [40] gave an excellent survey of interactive mesh-cutting techniques, which was not based on the progressive cutting.

Interactive mesh-cutting techniques are not realistic due to latency. To overcome this problem, Mor [18] [39] proposed a progressive minimal cutting approach. It generates a minimal set of new elements that followed the traces of avatar. Temporary subdivisions were created based on intersection points on faces and edges, which the cutting path had already passed through, and the current position of avatar. Small permanent elements were created right after the scalpel passes a boundary. The generated number of elements affected the update rate significantly, and could lead the system to instability. Bielser et al. [41] used a state machine to track the topology of each tetrahedron and control the progressive subdivision.

Zhang et al. [23] also employed progressive cutting to subdivide the surface that followed the path of the cutting tool. Different from previous methods, it was based on the surface model.

### 2.4 Haptic Force Feedback

How to provide realistic force feedback is also a technical bottleneck for VHS. For example, although an update rate of 20 Hz to 75 Hz is sufficient to visualize the changes
of the deformable virtual object in a real time fashion, faithful haptic feedback may require an update rate up to 1K Hz, which is extremely hard to achieve in practice. On the other hand, the high-frequency haptic feedback may cause an oscillatory behavior to the haptic device, which may thus become highly unstable and inflict harm on the user [48].

A few approaches have been presented to solve the above problem. Ellis et al. [49] introduced a numerical approach, which applied prediction and correction to extrapolate force feedback. However, human behavior is unknown. In addition, uncertainty in prediction introduced errors that would cause slight vibration. Cavasoglu et al. [50] used a linear approximation to represent the inter-sample behavior of the nonlinear full order model, and performed balanced model reduction on the linearized one. Unfortunately, numerical errors, which were coming from the linearization and the model reduction, could cause unexpected behaviors of non-heterogeneous compliant surfaces. Zilles et al. [51] and Ruspini et al. [52] applied god-object or proxy methods to render haptic forces.
CHAPTER 3 SYSTEM OVERVIEW

3.1 Framework

As discussed in chapter 2, a VHS simulator consists of three basic components: HI, RTS and GD. This thesis focuses on the RTS. We use a surface-based MSS model combined with a progressive cutting method to simulate the surgical scene. A new force model based on a BP NN and Gaussian noise function is applied to render force feedback.

As mentioned before, compared to FEM, MSS is easier to be created and requires lower computation. In addition, previous work shows that the MSS model is realistic if the deformation is small. In order to achieve real-time haptic and graphic responses, the computation of object deformation is confined within a zone of influence. Triangles are used as the basic shape elements, which make up of the virtual object surface. Each vertex is modeled as a mass particle. Adjacent particles are connected by springs and dampers.

During a palpating or cutting procedure, the data structure of the MSS model is accessed and modified in terms of the position and velocity of the avatar.
The loose-coupled framework is employed in this thesis. After receiving the collision information, the HRM and DCM calculate the external force respectively. The HRM renders force feedback to the haptic device. The DCM computes the updated deformation and renders the position of each particle to VRM. Figure 3.1 shows the framework of our simulator.

![Figure 3.1 Framework of our simulator](image)

When the user manipulates the haptic device, the CDM checks if the avatar is interacting with the virtual object. If collision occurs, the information about the avatar position \((X)\) and contacted triangle \((T)\) are provided to the DCM while the information about \(X\) and velocity \((V)\) are obtained by the HRM. Based on the values of \(T\) and \(X\), the DCM calculates the position, velocity and acceleration of each particle. Based on the values of \(V\) and \(X\), the HRM unit presents the force response \((F_r)\), which is approximated to the best of device’s capabilities, to the haptic device. The VRM handles transformations.
among different coordinate systems. The audio rendering module displays audio sound, if any, to the user.

3.2 System Diagram

Figure 3.2 shows the control flow of our simulator. It has two threads: one is the application thread with an update rate of approximately 30 Hz; the other is the haptic thread for which the update rate is around 1000 Hz.

For the haptic thread, after the haptic device is initialized, the servo control loop, whose update rate is about 1000 Hz, is activated. In every update cycle, the thread queries the current position of the haptic device and checks if a collision between the avatar and the deformable object occurs. The system is assumed to have two interactive scenarios: palpating or cutting.

For the application thread, the initialization routine creates the virtual environment as well as the deformable object; and establishes the mapping between MSS and the object geometry. The application thread receives updates from the haptic one and modifies the geometry of the virtual object accordingly.
Figure 3.2 System diagram
CHAPTER 4 PROGRESSIVE CUTTING

For a VHS simulator, the user (i.e., surgical student) expects to receive visual feedback in real time, as the avatar is moving through the deformable object surface. Therefore, the object topology has to be updated during the cutting procedure. For this purpose, new elements are usually added into the topology and old elements are removed. Updating of the object topology requires expensive computation. So, in order to achieve real-time performance, the less change the better. In this chapter, we first overview the general cutting procedure, then a progressive cutting method is discussed in detail.

4.1 Conceptual Overview

As mentioned before, the deformable object can be represented by either the surface-based or the volumetric model. Cutting through surface-based models was demonstrated by Song and Reddy [42], Tanaka et al. [43] and Zhang et al. [23]. Cutting through volumetric models was developed by Mor [39], Bielser et al. [17] and so on. Although volumetric models (such as the tetrahedral model) simulate the interior object structure, their topology modification is very complex due to the complex connectivity between geometric primitives. Here, the surface-based mesh model is applied to simulate the
virtual soft tissue. There are three states of the VHS: uncontacting, indenting and moving. Accordingly, the cutting operation can be divided into two main procedures. The first one is indentation, in which the surgical scalpel (i.e. avatar) indents the soft deformable tissue. It is then followed by the moving procedure, for which the scalpel moves within the soft tissue.

![Interaction loop of the indentation procedure](image)

**Figure 4.1** Interaction loop of the indentation procedure
Figure 4.1 shows the interaction loop of the indentation procedure. When collision occurs, the system enters into the indenting state, checking if the indenting threshold has been reached. The soft tissue is palpated if the applied force is not strong enough to puncture the surface; no new element is created. Otherwise, the scalpel penetrates the surface of the soft tissue; a new particle is generated at the penetrating point; and the system state is changed to the moving state.

Figure 4.2 Interaction loop of the moving procedure
Figure 4.2 shows interaction loop of the moving procedure. If the scalpel is still within the soft tissue, new elements have to be added, as the scalpel is sweeping the soft tissue. Otherwise, no new element is added; and the system enters into the uncontacting state.

4.2 Progressive Cutting

4.2.1 Surface Representation

The surface of soft tissues surface consists of triangles. Figure 4.3 shows an example that the scalpel sweeps the surface of the tissue.

In Figure 4.3, T1, T2, ..., T5 are triangles. P1, P2, ..., P6 are interaction points that the scalpel contacts with the triangles. Dash line is the scalpel path inside a triangle. We simplify any path within a triangle by a single straight line connecting starting and ending intersection points within the triangle (solid line). As Figure 4.3 shows, once the scalpel

33
indents into the deformable object, e.g. triangle T1, we deem it as the start of one cut. The beginning point, i.e. point P1, is determined. As the scalpel moves within the soft tissue, the underlying mesh that represents the object is modified. If the scalpel is lifted up and has no contact with the object surface, we can detect the last contacting point and triangle, i.e. ending point P6 in triangle T5.

In general, the scalpel path is not straight, even within a triangle. However, if the triangle is sufficiently small, the actual curve segment within a triangle can be approximated as a straight line. In this thesis, triangles like T1 are called beginning triangles; last interaction triangles like T5 are called ending triangles; and T2, T3 and T4 are referred to as middle triangles. Indented points like P1 are called beginning points; and last contacted points like P6 are ending points.

4.2.2 Progressive Cutting with Temporary Subdivisions

There are two methods to describe the progressive cutting procedure.

One method does not split the underlying triangle until the scalpel moves out of it. Figure 4.4 shows an example of this method. When the scalpel is moving into a triangle, the contacted point on the edge is recorded. The object topology is not changed when the scalpel is moving within the triangle. Only after the scalpel is moving out the triangle, the contacted triangle is split into small triangles, and the object topology is updated. This
method minimizes the computation during the cutting procedure. However, the noticeable delay is introduced if the contacted triangle is large and the velocity of the scalpel is slow.

Figure 4.4 Example of progressive cutting without temporary subdivision

The other method creates temporary subdivisions within the intersecting triangle. Figure 4.5 shows an example of this method. This method splits the triangle exactly following the scalpel motion. Once the scalpel is inside one triangle (T0), according to the scalpel's current position V, T0 is temporarily subdivided into smaller ones: T1...T4, which share the same vertex V. Therefore, temporary elements are updated using the scalpel's latest position. When the scalpel goes into another triangle, temporary triangles are removed and some new permanent smaller triangles substitute T0. The procedure then repeats. When the whole cut is completed, i.e. the scalpel is pulled out from the object, new permanent triangles replaced temporary triangles. This method needs more processing time but have little lag.
In order to render the cutting procedure realistically, we implement the second method. The following pseudo code illustrates the key part of the topology update algorithm.

If Triangle Intersected
   If it is not the last intersected triangle
      Create permanent triangles for the previous intersected triangles
      Remove temporary triangles in the previous intersected triangles
      Create temporary triangles (T1...T4) based on the position of scalpel and direction of the triangle
      Make the triangle invisible
      Update last intersected triangle
   Else
      Updated V
   Endif
Endif

Figure 4.5 Example of progressive cutting with temporary subdivision
4.2.3 Different Possible Cases for Progressive Cuts

Once the scalpel moves out from the triangle, the intersected triangle is subdivided. There are four different cases, which are shown in Figure 4.6. We define V as the vertex of the triangle and P as the intersected point on the edge. V-P, V-V, P-V and P-P represent those four cases. V-P means that the scalpel moves into the triangle from the vertex and moves out of the triangle from the edge. V-V refers that the scalpel moves into and out of the triangle both from vertices. P-V describes that the scalpel moves into the triangle from the edge and out of the triangle from the vertex. P-P means that the scalpel moves into and out of the triangle both from edges. Table 4.1 enumerates new elements added for those different cases.

<table>
<thead>
<tr>
<th>Case</th>
<th>New particles</th>
<th>New triangles</th>
</tr>
</thead>
<tbody>
<tr>
<td>V-P</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>V-V</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>P-V</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>P-P</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

In the case V-V, the system generates less particles and triangles than in the case V-P. We draw the same conclusion for the case P-V versus P-P. The less are new elements created, the faster does the system compute. We thus apply the snapping method to minimize the number of new elements. If the intersected point P is near to a vertex V and
the distance between them is shorter than the tolerant error $\varepsilon$, i.e. $\|P - V\| \leq \varepsilon$, then $P$ is snapped to $V$.

Figure 4.6 Four different progressive cutting cases

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
4.3 Experiments

When initializing the VHS, we have to establish a mapping between the workspace and the graphic scene. The relationship describes how to translate the movement between the physical device and the avatar. Our system has four coordinates [63]:

1) World coordinates are the global frame of reference for the graphic scene.

2) View coordinates are the local coordinates of the camera.

3) Touch coordinates are the parent coordinates system of the workspace. They are usually identical as the view coordinates.

4) Workspace coordinates are local coordinates of the haptic device.

![Diagram of coordinate transformation]

Figure 4.7 Mapping from the world coordinates to workspace coordinates (adopted from [63])

World-view matrix, view-touch matrix and touch-workspace matrix are defined to transform the vector/matrix among those different reference frames respectively. Figure
4.7 (adopted from the reference [63]) illustrates how to map the vector/matrix in the graphic scene to the haptic workspace.

Currently, the system is running on a Windows XP personal computer with a Pentium 4 3.60 GHz processor. A PHANTOM robotic arm from SensAble Technologies Inc. is used as the haptic interface. The PHANTOM robotic arm has three degrees of freedom (DOF) of force feedback, and six DOF input. The source code is developed in MS Visual C++ 6.0, using the OpenGL library for graphical rendering and GHOST SDK for haptic rendering. Figure 4.8 demonstrates the wire frame of the progressive cut algorithm. Figure 4.9 illustrates some snapshots of the progressive cutting on a three-dimensional surface.
Figure 4.8 Progressive cutting algorithm

Figure 4.9 Progressive cutting experiment
4.3 Summary

This chapter describes the progressive cutting algorithm. The mesh model of the soft tissue consists of triangles. Temporary particles are introduced to accommodate the scalpel motion when the scalpel is inside the soft tissue’s mesh. The triangle is small so that the track of the scalpel in the intersected triangle is approximated by a straight line. When the scalpel contacts with the triangle and the applied force is large enough to penetrate the soft tissue, a starting particle is introduced and the temporary particle is activated. In order to visualize the cutting with little lag, we implement a scheme to temporarily subdivide the intersecting triangle. The intersecting triangle is represented by temporary triangles, which share the same vertex with the intersecting triangles. The position of this vertex is identical with that of the scalpel. If the scalpel leaves the intersecting triangle from an edge, two permanent particles are added. If the scalpel leaves the triangle from a vertex, one permanent particle is added. The entering triangle is determined by the velocity of the scalpel and the intersected one is replaced by smaller triangles. This progressive cutting procedure is repeated as the scalpel is cutting the soft tissue; and the object geometry keeps to be updated. When no collision is detected, the ending particle is introduced in the temporary particle’s latest position; and the temporary particle is deactivated; the modification of the object topology is completed.
CHAPTER 5 TISSUE MODELING

In terms of modeling of soft tissues (e.g. human body or organ), MSS and FEM are two widely used physics-based techniques. The FEM method requires intensive computation, which makes it impossible to achieve real-time deformation. Compared to FEM-based models, the MSS method simulates the soft tissue in a more efficient way. In this thesis, we choose MSS for the development of our VHS simulator. In this chapter, we present the applied MSS in our simulator and discuss different integration methods in terms of two main criteria: computation efficiency and precision. In order to simulate a large deformable object in real time, we confine the computation area of the MSS model to achieve better performance.

5.1 Mass Spring System

5.1.1 The Model

For the MSS model, as mentioned in chapter 4, the mesh of the object surface is composed by a large amount of triangles. The vertices of each triangle are modeled as particles with mass and the edges of each triangle are modeled as springs and/or dampers.
X and V are the displacement and velocity of particles respectively. The differential equation of the particle movement is:

\[
\begin{bmatrix}
\dot{X} \\
\dot{V}
\end{bmatrix} = \begin{bmatrix}
V \\
F_{total} \\
M
\end{bmatrix}
\]

(5-1)

Where \( F_{total} \) is the total force applied to the MSS, \( M \) is the mass matrix. \( F_{total} \) consists of internal forces \( F_{int} \) and external forces \( F_{ext} \). Internal forces refer to damping forces \( F_{damping} \) and spring forces \( F_{spring} \). So the total force can be denoted as:

\[
F_{total} = F_{int} + F_{ext} = F_{spring} + F_{damping} + F_{ext}
\]

(5-2)

Combining equations (5-1) and (5-2), we can get

\[
M\ddot{X} = -KX - D\dot{X} + F_{ext}
\]

\[
M\ddot{X} + D\dot{X} + KX = F_{ext}
\]

(5-3)

Where \( D \) is the damping matrix and \( K \) is the stiffness matrix.

### 5.1.2 Modal Analysis

For a single input single output (SISO) system, the ordinary differential equation (DE) can be written as:

\[
m\frac{d^2x(t)}{dt^2} + d\frac{dx(t)}{dt} + kx(t) = f_{ext}
\]

(5-4)
where \( m \) is the particle mass, \( d \) is the damping coefficient, \( k \) is the stiffness coefficient, \( f_{\text{ext}} \) is the applied force, and \( x(t) \) is the displacement. We convert equation (5-4) to the s-domain by Laplace transform, the transfer function \( H \) is:

\[
H = \frac{1}{ms^2 + ds + k}
\]  

(5-5)

\[
H = \frac{1}{m} \frac{m}{s^2 + \frac{d}{m}s + \frac{k}{m}}
\]

The characteristic equation is:

\[
s^2 + \frac{d}{m}s + \frac{k}{m} = 0
\]  

(5-6)

The system poles are

\[
s_{1,2} = \frac{-d}{2m} \pm \frac{\sqrt{d^2 - 4mk}}{2m}
\]

Where \( d, m, k > 0 \). Let \( \omega_n^2 = \frac{k}{m} \), \( 2\zeta \omega_n = \frac{d}{m} \), equation (5-6) can be rewritten as:

\[
s^2 + 2\zeta \omega_n s + \omega_n^2 = 0
\]  

(5-7)

The location of poles dictates the nature of system stability and performance. For \( d > 0 \), the poles are on the left-hand complex plane, so the system is stable. For a second-order stable system, there are three scenarios: over damped, critically damped and under damped. In the case of over damped \( (d^2 - 4mk > 0, \text{i.e. } \xi > 1) \), \( s_{1,2} \) are negative real and
unequal, the response is sluggish and nonoscillatory. In the case of critically damped 
\( d^2 - 4mk = 0 \), i.e. \( \xi = 1 \), \( s_{1,2} \) are negative real and equal. The response is relatively fast 
and no oscillatory. It is the fastest nonoscillatory response. In the case of under damped 
\( d^2 - 4mk < 0 \), i.e. \( \xi < 1 \), \( s_{1,2} \) are complex number, the system is fast and oscillatory. If 
d = 0 , i.e. \( \xi = 0 \), poles are on the y-axis, it is critically stable.

![Graphs showing response for pulse signal in MSS](image)

1) Input signal

2) Over damped, \( w_n = 4, \xi = 2 \)

3) Critically damped, \( w_n = 4, \xi = 1 \)

4) Under damped, \( w_n = 4, \xi < 1 \)

5) Critically stable, \( w_n = 4, \xi = 0 \)

Figure 5.1 Response for the pulse signal in MSS
Figure 5.1 shows the displacement responses of the second order MSS model when the input signal is an impulse function under different scenarios. Here, $m = 0.01$. Increasing the value of $d$ can make the system more stable, but the response is slower. When we model the behaviors of the soft tissue, oscillation is not expected. Therefore, the critically and over damped system are suitable ones.

5.2 Physical Model

![Surface mesh model](image)

In our system, particles are linked together by springs and dampers as shown by Figure 5.2. The MSS lacks volume coherency: if it is put into a gravitational field, it collapses if the traditional model is not modified. Here, we use a rigid kernel, which represents the initial shape of the virtual object [44]. In the kernel, a home spring and damper is added.
to each particle. Therefore, there are two types spring in the system: home and link ones. The dampers also consist of home and link ones. A link spring or damper is defined along each triangle edge, and each particle is connected by the home spring and damping to its initial position.

For particle $i$, the DE is:

$$m_i a = -f_{\text{link-spring}} - f_{\text{home-spring}} - f_{\text{link-damping}} - f_{\text{home-damping}} + f_{\text{ext}} \quad (5-8)$$

While

$$f_{\text{link-spring}} = \sum_{j \in N(i)} k_{ij} \frac{(d_{ij} - l^0_{ij})d_{ij}}{d_{ij}} \quad (5-9)$$

$$f_{\text{home-spring}} = k_{ii} \frac{\|x_i - x_i^0\| x_i}{\|x_i\|} \quad (5-10)$$

The damping force operates in the same direction as the spring force, but its value depends on the relative velocity of particles. The equation of the damping term is given by

$$f_{\text{link-damping}} = \sum_{j \in N(i)} c_{ij} (v_i - v_j) \cdot \frac{d_{ij}}{d_{ij}} \frac{d_{ij}}{d_{ij}} \quad (5-11)$$

$$= \sum_{j \in N(i)} c_{ij} (\dot{x}_i - \dot{x}_j) \cdot \frac{d_{ij}}{d_{ij}} \frac{d_{ij}}{d_{ij}} \quad (5-12)$$

$$f_{\text{home-damping}} = c_{ii} v_i$$

Considering equations (5-9), (5-10), (5-11) and (5-12), equation (5-8) can be rewritten:
\[ m \ddot{x}_i + \sum_{j \in N(i)} (k_{ij} \left( \left\| \dot{d}_{ij} \right\| - l_{ij0} \right) + c_{ij} \left( \dot{x}_i - \dot{x}_j \right)) \frac{d_{ij}}{\left\| d_{ij} \right\|} + \frac{k_{ij} \left( \left\| x_i - x_j \right\| - x_{ij0} \right) x_i}{\left\| x_i \right\|} + c_{ii} v_i = f_{ext} \quad (5-13) \]

Where \( m_i \) represents the mass of particle; \( c_{ij} \) is the link damping coefficient of the particle \( i \) and its neighborhood particle \( j \); \( k_{ij} \) is the stiffness of the link spring, which connects particle \( i \) and \( j \); \( N(i) \) is the particle set of neighborhood of particle \( i \); \( k_{ii} \) is the stiffness of the home spring; \( d_{ij} \) is the distance between particle \( i \) and \( j \); \( l_{ij0} \) is the initial distance between particle \( i \) and \( j \); \( x_{ij0} \) is the initial position of particle \( i \); \( c_{ii} \) is the home damping coefficient of the particle \( i \). The displacement \( x \) and velocity \( v \) are chosen as state variables:

\[
\dot{x} = v
\]

\[
\dot{v} = \frac{f_{ext} - f_{home-spring} - f_{link-spring} - f_{home-damping} - f_{link-damping}}{m} \quad (5-14)
\]

### 5.3 Integration Method

Hairer et al. [45] pointed out that implicit integration methods were superior to explicit ones. To solve DEs, there are three main different implicit algorithms: the Euler, Runge-Kutta and Verlet ones.

“LDE (local discretization error) indicates how far the computed solution can be expected to deviate from the nearest true solution at each time step. GDE (global discretization error) indicates how far the computed solution can be expected to deviate from the nearest true solution at each time step. GDE (global discretization error) indicates how far the computed solution can be expected to deviate from the nearest true solution at each time step.
error) is the maximum error in the interval.” [46] When we choose the algorithm, the following two factors are considered:

1) The algorithm should be computationally efficient;

2) The algorithm should be accurate.

5.3.1 Euler Method

The Euler integration method is the simplest and most obvious way to integrate a set of differential equations numerically. It assumes that computing rates at a given time are constant through the time interval (time step). Equations (5-13), the $2^{nd}$ DE, can be implemented as:

$$\begin{align*}
x(i, t + h) &= x(i, t) + h\dot{x}(i, t) \\
\dot{x}(i, t + h) &= \dot{x}(i, t) + h\ddot{x}(i, t) \\
\ddot{x}(i, t) &= \frac{f_{ext}(i, t) - f_{home-spring}(i, t) - f_{link-spring}(i, t) - f_{home-damping}(i, t) - f_{link-damping}(i, t)}{m_i}
\end{align*}$$

Where

$$f_{home-spring}(i, t) = k_{ii} \left( \|x_i(t) - x_i^0\| x_i(t) \right)$$

$$f_{link-spring}(i, t) = \sum_{j \in N(i)} k_{ij} \left( \|x(i, t) x(j, t)\| - l_i^0 \right) x(i, t) x(j, t) \left\|x(i, t) x(j, t)\right\|$$
\[ f_{\text{home-damping}}(i,t) = c_h v(i,t) \]

\[ f_{\text{link-damping}}(i,t) = \sum_{j \in N(i)} c_j \left\| v(i,t) - v(j,t) \right\| \frac{x(i,t) - x(j,t)}{\|x(i,t) - x(j,t)\|} \]

Let \( h \) denote the time step; \( x(i,t) \) represent the displacement of the particle \( i \) at time \( t \); and \( v(i,t) \) be the velocity of the particle \( i \) at time \( t \).

The LDE and GDE made by Euler integration are proportional to the square of time step \( O(h^2) \) \[46\]. We can decrease the time step to obtain more accuracy.

### 5.3.2 Fourth-order Runge-Kutta Method

The order of a Runge-Kutta method’s LDE is equal to the order of the GDE. For the fourth-order Runge-Kutta (RK-4), the LDE and GDE are \( O(h^4) \) \[46\]. The method is written as:

\[ x(t + h) = x(t) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \]

Where

\[ k_1 = f(t, x(t)) \]

\[ k_2 = f\left(t + \frac{h}{2}, x(t) + \frac{h}{2}k_1\right) \]

\[ k_3 = f\left(t + \frac{h}{2}, x(t) + \frac{h}{2}k_2\right) \]
\[ k_4 = f(t + h, x(t) + hk_3) \]

Here, \( f(t, x(t)) \) is equation (5-14).

The RK-4 requires as four times much computation per step as Euler's method. Although the computation is more expensive, the reduced discretization error makes the RK-4 method a more efficient way to achieve the desired accuracy.

### 5.3.3 Verlet Method

In molecular dynamics, the most widely used integration algorithm is the Verlet algorithm [47]. There are three main Verlet algorithms: the basic, leap-frog, and velocity ones.

The basic idea of the Verlet algorithm is to explore two 3rd Taylor expansions for the position \( x(t) \):

\[
x(t + h) = x(t) + v(t)h + \frac{f_{\text{total}}(t)}{2m} h^2 + \frac{\dddot{x}(t)}{6} h^3 + O(h^4) \tag{5-15}
\]

\[
x(t - h) = x(t) - v(t)h + \frac{f_{\text{total}}(t)}{2m} h^2 - \frac{\dddot{x}(t)}{6} h^3 + O(h^4) \tag{5-16}
\]

Adding equations (5-15) and (5-16), we get

\[
x(t + h) = 2x(t) - x(t - h) + \frac{f_{\text{total}}(t)}{m} h^2 + O(h^4)
\]
The basic Verlet algorithm has no explicit velocities. It is straightforward and the storage space is modest.

In the leap-frog Verlet algorithm, velocities are first calculated at time $t + \frac{1}{2}h$. Its advantage is the explicit calculation of velocities. However, velocities are not calculated at the same time as positions are done.

\begin{equation}
x(t + h) = x(t) + v(t + \frac{1}{2}h)h + O(h^4)
\end{equation}

\begin{equation}
v(t + \frac{1}{2}h) = v(t - \frac{1}{2}h) + \frac{f_{total}(t)}{m}h + O(h^3)
\end{equation}

\begin{equation}
v(t) = \frac{v(t + h) - v(t - h)}{2} + O(h^3)
\end{equation}

In the velocity Verlet algorithm, we can obtain the position and velocity explicitly at each time step. However, it is slightly slower since it needs more computation.

\begin{equation}
x(t + h) = x(t) + v(t)h + \frac{f_{total}(t)}{2m}h^2
\end{equation}

\begin{equation}
v(t + h) = v(t) + \frac{f_{total}(t) + f_{total}(t + h)}{2m}h
\end{equation}

5.3.4 Computational Efficiency and Precision Analysis

Mor [39] pointed out that among those three Verlet algorithms, the leap-frog Verlet method is the best one in terms of computational efficiency. The leap-frog Verlet method
is thus chosen to be analyzed among those three Verlet algorithms. In term of two criteria: computation efficiency and precision, we analyze and compare Leapfrog Verlet, Euler and Runge-Kutta algorithms.

Table 5-1 illustrates the precision of those three ones. From Table 5-1, we find that the Runge-Kutta method is the most accurate one while the Euler method is the least one.

Table 5-1  Precision of integration algorithms

<table>
<thead>
<tr>
<th>Methods</th>
<th>LDE</th>
<th>GDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euler</td>
<td>$O(h^2)$</td>
<td>$O(h^2)$</td>
</tr>
<tr>
<td>Runge-Kutta</td>
<td>$O(h^4)$</td>
<td>$O(h^4)$</td>
</tr>
<tr>
<td>Leapfrog Verlet</td>
<td>$O(h^3)$</td>
<td></td>
</tr>
</tbody>
</table>

In order to render the virtual object smoothly, the update frequency must be at least 33 Hz. The maximum acceptable time step is thus 0.033 second. Some experiments have been performed to evaluate this algorithms’ computation performance. Experiments are run on Windows XP with a 3.60GHz Pentium 4 processor. The source code is written in MS Visual C++ 6.0, using the OpenGL library for the graphical rendering. Table 5-2 lists the average time step for those three integration algorithms when the model has 675 elements (225 vertices and 450 edges); Table 5-3 shows their performances when the model has 1200 elements (400 vertices and 800 edges); Table 5-3 illustrates their
performance comparison when the model has 1875 elements (625 vertices and 1250 edges).

### Table 5-2 Time steps vs. integration algorithms (1)

<table>
<thead>
<tr>
<th>Methods</th>
<th>Average Time Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euler</td>
<td>0.016</td>
</tr>
<tr>
<td>Runge-Kutta</td>
<td>0.024</td>
</tr>
<tr>
<td>Leapfrog Verlet</td>
<td>0.016</td>
</tr>
</tbody>
</table>

### Table 5-3 Time steps vs. integration algorithms (2)

<table>
<thead>
<tr>
<th>Methods</th>
<th>Average Time Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euler</td>
<td>0.016</td>
</tr>
<tr>
<td>Runge-Kutta</td>
<td>0.042</td>
</tr>
<tr>
<td>Leapfrog Verlet</td>
<td>0.016</td>
</tr>
</tbody>
</table>

### Table 5-4 Time steps vs. integration algorithms (3)

<table>
<thead>
<tr>
<th>Methods</th>
<th>Average Time Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euler</td>
<td>0.022</td>
</tr>
<tr>
<td>Runge-Kutta</td>
<td>0.064</td>
</tr>
<tr>
<td>Leapfrog Verlet</td>
<td>0.022</td>
</tr>
</tbody>
</table>
From these comparisons, it is clear that Euler and Verlet algorithms compute faster than Runge-Kutta methods. However, the Euler’s LDE is $O(h^2)$ while the Verlet algorithm’s is $O(h^3)$. The Verlet method is more accurate than the Euler one. The Leapfrog Verlet algorithm is thus chosen in our simulator.

5.3 Localized Mass Spring Model

Usually, the surgical tool interacts with only one part of the soft tissue, especially when the size of the soft tissue is much larger than that of the scalpel’s tip. We thus assume that the deformation area of the soft tissue is confined within a “zone of influence”. Only the DEs of those particles, which are within the zone of influence, are computed. The system computation load is thus dramatically reduced. The method is useful to calculate massive complex topologies.

The external force can be divided as:

$$f_{\text{ext}} = f_{\parallel} + f_{\perp} \quad (5-17)$$

Where $f_{\parallel}$ is the force parallel to the deformable surface plane, and $f_{\perp}$ is the force perpendicular to the surface plane.

For equation (5-13), we assume $m = 1, k_{ij} = 1, c_{ij} = 1, k_{ii} = 2, c_{ii} = 1$. The particle mass value is: $m_i = \frac{m}{N}$ if the virtual object has $N$ particles. Let the virtual object be a 10*10...
square with 100 particles and particles on the boundary is nailed. The object mesh model is illustrated in Figure 5.3. We also suppose that the external force is not strong enough to cut the virtual object; and it is applied to the contact particle (the particle 1 in Figure 5.3), which is the closest to the scalpel’s tip. We define that particles in “the first ring” are those ones who connect directly to particle 1; and particles in “the second ring” are those who connect directly to particles in the first ring. Therefore, in Figure 5.3, particle 2 is in the first ring; and particle 3 is in the second ring. Figure 5.4 shows the relationship between $f_{\perp}$ and time; and Figure 5.5 illustrates the relationship between perpendicular displacements of particles and time when the scalpel interacts with the virtual object. Figure 5.6 and 5.7 shows the case of $m = 1$, $k_{ij} = 2$, $c_{ij} = 1$, $k_{ii} = 1$, $c_{ii} = 0.5$, $N = 400$. 

![Figure 5.3 Mesh model](image-url)
Figure 5.4 Applied perpendicular force (100 particles)

Figure 5.5 Displacement of particles in different rings (100 particles)
Figure 5.6 Applied perpendicular force (400 particles)

Figure 5.7 Displacement of particles in different rings (400 particles)
It is observed that when an external force is exerted on the virtual object, there is little influence on the particles in the second ring. So in our VHS, only particles within the second ring are calculated. Since the number of DE equations is dramatically reduced, the time for computation is significantly reduced as well.

5.3 Simulation of Palpating the Heart Model

In this thesis, a heart model is read in by the simulator from a data file. The basic wire frame of the heart [64] is shown in Figure 5.8: the model has 961 nodes, 1712 triangles, and 2687 edges. In our current implementation, the simulation runs at an update rate of approximately 50Hz, which is fast enough for our human perception, when the proposed localized MSS model is used.

![Wire frame of the heart](image)

**Figure 5.8** Wire frame of the heart

Figure 5.9 a) displays the flat shading of the heart model, in which the triangle normal is assigned to its three vertices; Figure 5.9 b) shows the shade of the model with the “true
normals" [65], in which the vertex is assigned by its own normal respectively. We find that the “true normals” method produces an image more realistic than the flat shading ones. Figure 5.10 shows the deformation of the virtual object (the virtual heart) when it is palpated by a virtual tool.

Figure 5.9 Undeformed model of the heart

Figure 5.10 Deformed model of the heart
CHAPTER 6 HAPTIC MODELING

In a surgical procedure, minimally invasive surgeries in particular, the surgeon rely mainly on haptic feedback to manipulate the surgical instruments in addition to visual feedback. For a VHS simulator, when the trainee interacts with the simulated virtual tissue through the avatar, he/she can not only observe the geometrical changes of the tissue but also perceive the interaction force. Although measuring the user motion is easy, it is a complex process to determine appropriate force feedback to the user. The function of the haptic model in the VHS is to calculate proper haptic (force) feedback to the user. Stable and realistic reflection of force is a fundamental requirement for a VHS simulator.

In this chapter, we propose a new method, which contains a BP NN and Gaussian noise function, to render haptic feedback.

6.1 Neural Network

Neural network (NN) is suitable for system modeling in the following cases [54]:

1) It is hard to formulate an analytic solution;

2) The example data can be obtained;

3) The system structure can be identified out from existing data.
In this thesis, a traditional feed-forward back-propagated (BP) algorithm is applied to represent some parts of force feedback model.

![Architecture of neuron](image)

**Figure 6.1 Architecture of neuron**

Figure 6.1 illustrates the architecture of a neuron for which the output is

\[ y = f(X^T W + b) \]

Here \( X \) is a \( R \times 1 \) length input vector, \( W \) is a \( R \times 1 \) vector, and \( b \) is a bias unit scalar.

Figure 6.2 is a BP NN architecture. A BP NN consists of three parts: input, hidden and output layers [55].

We choose a tan-sigmoid function as the transfer function of the hidden layer:

\[
y = f(x) = \frac{1 - e^{-x}}{1 + e^{-x}}
\]  

(6-1)

where \( x \) is the input variable of the transfer function and \( y \) is the output of the function.
We choose the following linear function as the transfer function for the output layer:

\[ y = f(x) = x \]  

(6-2)

A gradient descent algorithm is used to adjust weights and biases. For the input pattern \( p \):

Figure 6.1 Architecture of a BP network
\[ E_p = \frac{1}{2} \sum_{k=1}^{M} (y_{pk} - o_{pk})^2 \]  

(6-3)

Where \( y_{pk} \) is the desired output of the output node \( k \) when the input pattern \( p \) is presented; \( o_{pk} \) is the actual output of the output node \( k \); and \( M \) is the number of the output node.

For the output layer, weights are updated following the equation below:

\[
\begin{align*}
    w_{kj}^{0} &= w_{kj}^{0} + \eta \sum_{k=1}^{M} \frac{\partial E_p}{\partial w_{kj}} \\
    &= w_{kj}^{0} + \eta (o_{pk} - y_{pk})o_j
\end{align*}
\]  

(6-4)

Where \( w_{kj}^{0} \) is the weight between output node \( k \) and hidden node \( j \); \( o_j \) is the output of the previous layer \( j \); and \( \eta \) is the update rate.

For the hidden layer, weights are updated as:

\[
\begin{align*}
    \Delta w_{ij} &= \sum_{k=1}^{M} (o_{pk} - y_{pk}) * w_{jk} * f'_{n} * o_i \\
    w_{ij}(t+1) &= w_{ij}(t) + \eta \Delta w_{ij}
\end{align*}
\]  

(6-5)

(6-6)

where \( f'_{n} \) is the coefficient of equation (6-1).

Although the training procedure is computationally expensive, weights and biases are pre-computed.
6.3 Force Modeling

In order to create a haptic model which can realistically present force feedback to the user, we have to consider many factors, such as entering and moving angles between the scalpel and the soft tissue, the velocity of the scalpel movement, the sharpness of the scalpel blade, the indenting depth of the scalpel within the soft tissue and so on. However, it is very difficult to build a mathematical model incorporating all these factors. In this chapter, we introduce a new model to simulate the force response based on two factors: the scalpel velocity and indenting depth.

6.3.1 Force Modeling

The cutting procedure is divided into two main sub procedures: one is that the scalpel indents the soft tissue without obviously moving along other directions; the other is that the scalpel moves within the soft tissue. We assume that the depth to break the soft tissue is constant.

The model for force feedback can be written as:

\[ F_{\text{total}} = F_{\text{indent}} + F_{\text{organ}} + F_{\text{noise}} \]  \hspace{1cm} (6-7)

where \( F_{\text{indent}} \) is the indentation force, \( F_{\text{organ}} \) reflects the force related with blood vessels and muscles, \( F_{\text{noise}} \) is the other forces except \( F_{\text{indent}} \) and \( F_{\text{organ}} \), such as the friction and pressure force. \( F_{\text{total}} \) is the total force feedback to the user.
6.3.2 Force Modeling in the Indenting Procedure

During the indentation procedure, $F_{\text{organ}}$ and $F_{\text{noise}}$ can be regarded as 0. So

$$F_{\text{total}} = F_{\text{indent}}$$

In addition, the direction of the vertical displacement of the scalpel ($x_\perp$) is assumed to be perpendicular to the surface of the soft tissue, which is showed in figure 6.3.

![Figure 6.3 Displacement in indenting procedure](image)

The sub procedure of indentation is divided into two parts: one is before the scalpel puncturing the soft tissue, and the other is after the scalpel puncturing the soft tissue.

For the part of “before puncturing”, the force feedback is represented by a spring model, which is based on the following equation:

$$F_{\text{indent}} = k_{\text{indent}} \Delta x \quad \Delta x_\perp \leq \Delta x_{\perp \text{max}}$$ (6-8)
As mentioned many times before, the topology of soft tissue will be changed once the user exerts force on it. When the scalpel displacement ($\Delta x_\perp$) reaches the threshold $\Delta x_{\perp \text{max}}$, the soft tissue is split. The puncturing depth can be regarded as a constant.

After puncturing, the scalpel usually keeps moving more deeply. Since the encapsulated surface has been punctured, the force feedback is less than the puncturing force. A number of experiments have been performed by other researchers and in this thesis we simply adopt their experimental results directly. A BP NN is used to model these data during the indenting procedure.

### 6.3.3 Force Modeling in the Moving Procedure

There are three scenarios in the moving procedure, which are shown in figure 6.4. We assume $F_{\text{indent}}$ is an unloading force ($F_{\text{unload}}$) in the cases a) and b), and it is a loading force ($F_{\text{load}}$) in the case c). Therefore, $F_{\text{indent}}$ can be written as:

$$F_{\text{indent}} = \begin{cases} 
F_{\text{unload}} & \| (x(t) - x(0)) \cdot \vec{n} \| \leq \| (x(1) - x(0)) \cdot \vec{n} \| \\
F_{\text{load}} & \| (x(t) - x(0)) \cdot \vec{n} \| > \| (x(1) - x(0)) \cdot \vec{n} \| 
\end{cases}$$

(6-9)

Where the surface normal is $\vec{n}$, $x(t)$ is the scalpel position at time $t$, $x(0)$ is the initial scalpel position, where the scalpel punctures the soft tissue.
When the scalpel passes through blood vessels and muscles, it encounters the force mainly coming from blood vessel walls and muscles, which is represented by a linear spring model.

\[ F_{\text{organ}} = \alpha_{\text{depth}} f_{\text{wall}} = \alpha_{\text{depth}} k_{\text{organ}} \Delta x \]  

(6-10)
Where $k_{\text{organ}}$ is the average spring coefficient of blood vessel walls and muscles; and $\alpha_{\text{depth}}$ is the number of contact vessel walls and muscles. The deeper the scalpel indents the soft tissue, the larger $\alpha_{\text{depth}}$ is. The blood vessel wall’s normal is $\vec{n}_v$. There is no velocity change during a time step $\Delta t$. Hence,

$$\alpha_{\text{depth}} = \alpha \| (x(t) - x(0)) \cdot \vec{n} \|$$

(6-11)

Where $\alpha$ is the coefficient.

$$\Delta x = v(t) \Delta t \cdot \vec{n}_v$$

(6-12)

Combining equation (6-11) and equation (6-12), equation (6-10) can be rewritten as:

$$F_{\text{organ}} = -\alpha k_{\text{organ}} \| (x(t) - x(0)) \cdot \vec{n} \| v(t) \Delta t \cdot \vec{n}_v$$

(6-13)

We use $F_{\text{noise}}$ to represent all other forces other than the indenting and organ ones. This is supported by the result of [57]: “Under very general conditions, as the number of components becomes large, the distribution function of random variable approaches that of the Gaussian (normal) random variable” [57]. We thus assume $F_{\text{noise}}$ is a white-Gaussian function.

### 6.4 Experiment Data Analysis

Due to logistical and regulatory constraints, real-life data can not be directly obtained by our own experiments. Data used in this thesis are adopted from a few other papers.
From figure 6.5 [53], we observe that:

1) There is no obvious relationship between the indentation speed and the force feedback for the same probe diameter.

2) Figure 6.6 b) shows the plot of the force versus the displacement for the loading and unloading section. The loading section is regarded as the pushing operation while the unloading one can be considered as the pull operation.

3) It is clear that for the same probe displacement, the probe force is larger in loading section than the unload one.

For the indenting procedure, we assume the indenting force is a loading force, then use some part of experimental data to train the BP NN. For the moving procedure, we firstly
use constraints shown by equation (6-9) to determine the type of the indenting force, i.e. loading force or unloading force, and then the relevant experimental data are used to train the BP NN.

From the experimental data shown in Figure 6.6 (scalpel moved at a speed of 0.1 cm/sec) [56], we observe that the curve is oscillating around the mean of the output force.

![Figure 6.6 Raw experimental data for liver cutting (adopted from [56])](image)

The curve in Figure 6.6 matches our expectation. During the cutting procedure, the cutting depth still affects the force feedback. The soft tissue is cut when the scalpel is indenting about 1.8 cm and the applied force is around 3 N for the liver.
6.5 Simulation

We use a two-layer BP NN, which contains a hidden layer with five neurons and an output layer with one neuron, to model the indenting force. The experimental data set of Figure 6.5 a) is applied to train the BP NN. Figure 6.7 shows the topology of our BP NN. The input is the scalpel vertical displacement and the output is the force along the same direction. The training rate $\eta$ is 0.001. Weights and biases of each neuron are determined, which are presented in table 6-1. Figure 6.8 shows the comparison of experimental data and output of the testing data, which use the trained BP NN. Figure 6.9 displays the network’s performance according to the mean of squared errors (MSE). It illustrates that after being trained for 100 epochs, the BP NN’s mean of squared errors is around $10^{-5}$ N.

Figure 6.7 Topology of simulation BP neural network
Figure 6.8 Experimental data vs. trained neural network data

Figure 6.9 NN’s performance
Table 6-1  Trained weights and biases

<table>
<thead>
<tr>
<th>$w$ (input layer $\rightarrow$ hidden layer)</th>
<th>$w$ (hidden layer $\rightarrow$ output layer)</th>
<th>$b$ (hidden layer)</th>
<th>$b$ (output layer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1959</td>
<td>0.0656</td>
<td>-13.8628</td>
<td>0.7025</td>
</tr>
<tr>
<td>-0.8624</td>
<td>-0.3429</td>
<td>18.5440</td>
<td></td>
</tr>
<tr>
<td>-0.5384</td>
<td>-0.1393</td>
<td>10.0367</td>
<td></td>
</tr>
<tr>
<td>-0.2533</td>
<td>-0.1248</td>
<td>3.7805</td>
<td></td>
</tr>
<tr>
<td>-0.1185</td>
<td>-0.0331</td>
<td>1.0538</td>
<td></td>
</tr>
</tbody>
</table>

When some data set of Figure 6.5 b) is applied to train the BP NN, the NN with a hidden layer (with five neurons) and an output layer (with one neuron) is implemented. Due to the number of input parameters increases, the input layer contains two neurons: one presents the scalpel vertical displacement; the other accepts the type of the indenting force: load or unload. Weights and biases of each neuron are presented in table 6-2. Figure 6.10 shows curves of the experimental data and output of the trained BP network. The network performance is $1.232 \times 10^{-4}$ N.

Table 6-2  Trained weights and biases

<table>
<thead>
<tr>
<th>$w$ (input layer $\rightarrow$ hidden layer)</th>
<th>$w$ (hidden layer $\rightarrow$ output layer)</th>
<th>$b$ (hidden layer)</th>
<th>$b$ (output layer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1211</td>
<td>-8.8553</td>
<td>1.4730</td>
<td>-3.3122</td>
</tr>
<tr>
<td>0.3226</td>
<td>0.4131</td>
<td>17.1267</td>
<td>-9.3809</td>
</tr>
<tr>
<td>-1.2783</td>
<td>-6.0967</td>
<td>-0.0032</td>
<td>3.7464</td>
</tr>
<tr>
<td>-0.6281</td>
<td>-0.4962</td>
<td>-0.0200</td>
<td>8.6331</td>
</tr>
<tr>
<td>-2.4796</td>
<td>-4.0361</td>
<td>0.0002</td>
<td>1.6728</td>
</tr>
</tbody>
</table>
In the moving procedure, to simplify the simulation of the force $F_{\text{organ}}$, we suppose $\bar{n}$ and $\bar{n}_v$ are orthogonal. Here, we define

$$\bar{n} = [0 \ 1 \ 0], \quad \bar{n}_v = [1 \ 0 \ 0],$$

$$x(0) = [0 \ 0 \ 0], \quad v(t) = [v_x \ v_y \ v_z], \quad x(t) = [p_x(t) \ p_y(t) \ p_z(t)],$$

$$F_{\text{organ}} = [f_x \ f_y \ f_z].$$

Equation (6-13) can be rewrite as:

$$\begin{cases} 
  f_x(t) = -\alpha k_{\text{organ}} \|v_x(t)\|v_x \Delta t = \beta \|p_y(t)\|(p_x(t) - p_x(t-1)) \\
  f_y(t) = 0 \\
  f_z(t) = 0 
\end{cases} \quad (6-14)$$

Figure 6.10 Experimental data vs. trained NN data under both conditions
Where $\beta = -\alpha k_{\text{organ}}$.

Since the experiment device is moving under a constant velocity ($v = 0.1\text{cm/sec}$) in the reference [56], the distance from the constraint edge has a linear relationship with time, Suppose $\beta = -18000 N/cm^2$. After introducing the $F_{\text{noise}}$, which modeled by the function of white-Gaussian noise and its signal to noise ratio (SNR) is 30 dB, we can get the relationship between the force magnitude and the distance from the constrained edge, which is shown in Figure 6.11 b). Figure 6.11 a) shows the relationship between the device depth and the distance from the constrained edge when the device moves through the liver. It is worth to point out that the output of our force model is very close to the corresponding experimental data. Therefore, the model we created can be applied to the VHS.
Figure 6.11 Force vs. distance from constrained edge of specimen

6.6 Extension

The proposed NN algorithm can be applied to simulate other procedures, especially the insertion procedure. Maurin et al. [58] used a Nano-17 6 DOF force sensor and a special handling tool, which can either be mounted as a robotic arm end-effector or held by hand, to acquire in vivo data with 18 gauges 15.24 cm biopsy needle in a living pig. Simone et al. [59] used the Johns Hopkins Steady-Hand Robot to obtain data with a needle at a constant velocity of 3 mm/s inserting into a whole bovine liver. Okamura et al. [60] used a 1 DOF robot equipped with a load cell and 1.27-mm diameter needle attachment to collect data. Heverly et al. [61] performed a needle insertion test on in-vivo porcine and rabbit heart samples. The applied force is one dimension and we can obtain the
experimental data of the whole insertion procedure. Therefore, the NN algorithm can be used to model the force of the whole procedure.

Figure 6.12 a) displays the experimental data adopted from the reference [61], and testing data vs. displacement when the needle is inserted into a rabbit heart. The NN is trained for 500 epochs, which has two hidden layers, for which the first one has five neurons and the second one has three neurons. The input layer has two neurons, which represent the needle displacement and the velocity respectively. After training for 500 epochs, the performance (MSE) is $5 \times 10^{-4}$. When one hidden layer with five neurons is applied, we can get the result in Figure 6.12 b), where the MSE is $7 \times 10^{-4}$. One hidden layer is thus good enough to simulate the force during the needle insertion procedure.
Figure 6.12 Force vs. displacement during rabbit heart testing

80
Figure 6.12 c) illustrates the figure of Yang’s model [62] compared with real data. Yang combined the spring model, Karnopp friction model, and still blood model to simulate the whole procedure.

Figure 6.13 a) shows the experimental data adopted from the reference [61] and testing data versus the displacement when the needle is inserted into a porcine heart. The NN has one hidden layer with five neurons, where the MSE is $6 \times 10^{-4}$. Figure 6.13 b) displays the result of Yang’s model [62] versus the real data. Our model can simulate the relationship between the force and the depth when the scalpel moves in different velocities, while Yang’s model is just suitable for the fixed velocity. Moreover, from Figure 6.13, we observe our model is more accurate than Yang’s.
Figure 6.13 Force vs. displacement during porcine heart testing

6.7 Discussion

The developed NN has been shown to be an approximate approach to modeling indenting force. It is also easy to get the corresponding force for a given displacement if we use the interpolating method. However, with increase of the number of input variables, the method becomes more complex. In the NN, on the other hand, it is simple to add more input parameters while little storage space is needed. It is not difficult to retrain and rebuild the NN-based model. Other models are not suitable if more input parameters are added. The framework of the model may not reusable.
During experiments, we notice that the more neurons and hidden layers are applied, the more accurate output of the testing data. When we train the NN, the MSE can not decreased significantly after being trained for some epochs. From Figure 6.12, we observe that although more neurons and hidden layers are added, the MSE does not decrease obviously.

Computation load is another factor we have to consider. In order to store more information, the NN has to contain more neurons and hidden layers. If we want to obtain the expected result for multiple input parameters by a fixed NN architecture, the NN is more complex and need more time to calculate. However, the complex computation may lead to the failure in force rendering within 0.001 second.

Therefore, there is a trade off among the complex, precision and efficiency. In our case, for the same soft tissue, the identical NN architecture is applied to compute the force, which is based on the depth and velocities of the scalpel. A hidden layer with five neurons can store enough information to achieve the acceptable performance, which is around $O(10^{-4})$.

When we model the force feedback, white-Gaussian noise is introduced to represent other possible forces such as the friction and pressure forces. If the scalpel blade is sharp, those forces are relatively small although they do exist. White-Gaussian noise is applied to simulate them for the sake of their simplicity.
CHAPTER 7 CONCLUSIONS

This thesis addresses several issues in VHS. A progressive cutting method is proposed to display the virtual interaction realistically as the avatar is moving through the object surface. The virtual object (e.g. soft tissue or organ) is a surface-based model, which consists of triangles. A modified MSS is used to model the behaviors of the soft tissue. A new neural network based approach is developed to render force. A prototype to show progressive cutting on a simple surface and palpating with the "visual" heart are developed as well in this thesis.

7.1 Contributions

Main contributions of this thesis are as follows:

1) The development of a new force model to simulate haptic force feedback to the user. In terms of the scalpel’s movement, we divide the cutting procedure into two sub-procedures: the indenting procedure and the moving one. A BP NN based algorithm is introduced to approximate the indenting force, which includes both the loading and unloading one. The simulation results show that the method matches the experimental data very well. Another advantage of the NN-based approach exists in the fact that it needs relatively little memory space compared to
other methods. Although training a NN is a time-consuming task, it can be performed offline in advance, which does not affect the computation in real time.

2) The introduction of white Gaussian noise to model friction and pressure forces. For a sharp scalpel, indenting and organ forces are the two main resistances which are modeled by a BP NN and a spring system respectively. The simulation results show that white Gaussian noise matches the experimental data very well.

3) The employment of a temporary particle to track the position of the haptic device in the virtual environment when the progressive cutting method is applied. The progressive cutting method renders the geometrical modification with little delay. In order to minimize the number of adding elements, those new points, which are very close to vertices of the intersected triangle, is snapped to the nearest vertices. In this case, the current interacting triangle is determined by the velocity of the haptic device. The intersected triangle is subdivided by new particles and the geometry of the virtual object is updated.

4) An improved MSS model for soft tissues. Both link and home springs/dampers are applied to model the properties of the soft tissue. In terms of the two key criteria: computation efficiency and precision, we compare the Leapfrog Verlet, Euler and Runge-Kutta algorithms and choose the Leapfrog Verlet as the integration method.
5) Development of a new scheme to localize the computation area within a “zone of influence”. It is very difficult to satisfy the real-time requirement for deformation if the number of particles is too large. Solving the differential equations of each particle in each updating time step requires expensive computation. In our approach, only those particles in the zone of influence are considered. Therefore, the time required to solving those equations is significantly reduced, which is especially obvious for the massive and complex topologies of soft tissues.

7.2 Future Work

Some potential future work is summarized and listed as follows:

1) Besides the scalpel velocity and indenting depth, the force feedback also depends on the sharpness, size and shape of the scalpel blade as well as its entering angles to the soft tissue. The factors should be incorporated if we want to achieve a higher degree of realism. Extending the NN-based approach to integrate these factors needs further investigation.

2) Some other complex interactions between tool and soft tissues occur during a surgical operation. For example, visualization of tissue bleeding and coagulation should also be taken into account.

3) In order to achieve more realistic visualization, a multiple-layered surface or volumetric model for the soft tissue might be used to describe the inherent nonlinearities and anisotropy of soft tissues. Complex tissue models, on the other
hand, are computationally expensive. There is always a trade-off between visual realism and computation efficiency.

4) It is still an open topic to set proper coefficients for a MSS soft tissue model.
REFERENCES


88


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


93

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


95


