NANYANG TECHNOLOGICAL UNIVERSITY

GEOMETRY MAP ALGORITHMS FOR INTERACTIVE DEFORMATION AND REAL-TIME VISUALIZATION

LIU QIANG

SCHOOL OF COMPUTER ENGINEERING

2008

Geometry Map Algorithms for Interactive Deformation and Real-time Visualization

Liu Qiang

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

2008

Abstract

Laparoscopic surgery is a popular technique that requires intensive training and practice before actual surgery. Computer simulation provides a virtual training environment for surgeons to practise. Surgery simulators are becoming practical with new frameworks that include core components such as enhanced medical imaging, rapid medical data modeling, realistic visualization procedures, and high fidelity animation techniques. The emphasis on cinematic quality rendering has gained importance with the widespread availability of affordable high end graphics capabilities on desktop computers. However, the challenge is the computation of man-machine interaction with the deformable tissues/organs.

Modeling organs for surgery simulation is a formidable challenge due to three reasons. First, the organs and body parts have not only very complex shapes, but are deformable too. The second challenge is to provide a mechanism for surgeons to interact with these complex anatomical objects. The third challenge is the expensive 3D deformation computation required for haptic feedback.

We propose a new technique that is extensively superior to the conventional rigid Geometry Images. In our approach, a parameterized representation of virtual organs with multi-layered 2D flat maps is used for the purpose of surgery simulation. An unstructured 3D input mesh is parameterized and resampled into a regular 2D parameterized model called Geometry Map. To accomplish real-time interaction with

deformable organs, instead of computing the deformation on the organ models in 3D space, we use a novel yet simple and fast free-form deformation on the 2D geometry map itself. Real-time haptic feedback can be provided with this deformation technique.

Our parameterized representation can efficiently perform both collision detection of tool tip with deformable object and contact detection between multiple deformable objects. We also show that the application of the geometry map can even be extended to the haptic sensing of virtual textiles. With this representation, we demonstrate that we can construct a practical and realistic environment with both visual and haptic rendering for interactive surgery simulation.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Dr. Edmond C. Prakash. I have gained intellectual guidance, inspiration and constant encouragement from him, as well as benefited from his enthusiasm and serious attitude to research work. He has provided me with guidance on the research as well as updating me on the latest technological trends in the areas of my work.

I am also thankful to Chen Chen, Evgeny Markin, Zheng Zhi, Liu Qi, Yu Xiaozhou, Jin Tao, Qiu Jie, and Chen Quan, my labmates in the Centre for Advanced Media Technology (CAMTech), for their warm help and encouragement. Deep thanks also expressed to my friends Sahren Ahmad, Xiang Tuowen, Zhu Lin, Guan Yunqiang, Xu Fei, and Zhou Jiong, for their support in my work and daily life as well.

Many thanks to Associate Prof Chan K. Y. Tony and Prof. Müller-Wittig for providing me a very good and harmonic research environment and advanced facilities at CAMTech. Special thanks also go to the staff at CAMTech, Gerrit, Jochen, and Marcus, for their help in facilitating access to the utilities in the lab.

Last but not the least, I wish to express my deepest gratitude to my parents and my brother, for their unconditional love, support and encouragement.

Contents

Abstract i

Acknowledgements iii

Contents iv

List of Figures xii

List of Tables xvi

1 Introduction 1

1.1 Motivation 1

1.2 Possible Interactions 2

1.3 Challenges 3

1.3.1 Static Model Representation 4

1.3.2 Deformable Organs 5

1.3.3 Cinematic Quality Rendering 6

1.4 Contributions 6

1.5 Thesis Outline 8

2 Literature Review **11**

2.1 Introduction: Challenges in Designing Surgery Simulators 11

2.2 Static Organ Shape Modeling 12

2.2.1 Geometric Modeling in Hepatic Simulation 13

2.2.2 Surface vs Volumetric Soft Tissue 14

2.2.3 Dynamic Organ Geometry Modeling 14

2.3 Dynamic Organ Deformation Modeling 15

2.3.1 Deformable Modeling with Surface Representations 16

2.3.2 Deformable Modeling with Volume Representations 17

2.3.3 Deformation Modeling in Hepatic Simulation 19

2.3.4 Dynamic Volumetric Simulation of Deformation 20

2.3.5 Mass-Spring Mesh for Vessel Deformation 21

2.3.6 Deformation with Sphere-Filled Organ Model 21

2.3.7 FEM with Parallel Computing 22

2.3.8 Real-Time FE for Virtual Suturing 22

2.4 Haptics and Interactive Deformation 23

2.4.1 Localized Deformation with Finite Points 23

2.4.2 Localized Deformation with Beam-Skeleton 25

2.4.3 Deformation with Force Transmittal Mechanism 25

2.4.4 LOD in Haptic Deformation 27

2.5 Surgery Applications 27
2.5.1 Bone Surgery Simulation 27
2.5.2 Virtual Endoscopy 28
2.5.3 Bronchoscope Simulation 29
2.5.4 Cleft Lip Simulation 29
2.5.5 Liver Surgery Simulation 30
2.5.6 Intestinal Surgery Simulation 30
2.5.7 Virtual Suturing and Knot-Tying Simulation 30
2.5.8 Microsurgery Simulation 31
2.6 Major Components of Surgery Simulators 32
2.6.1 Cakmak et al. [17] 32
2.6.2 Delingette and Ayache [36] 33
2.6.3 Chung et al. [22] 34
2.6.4 Raghupathi et al. [71] 35
2.6.5 Haptic Browser [69] 36
2.7 Cinematic Quality Rendering 36
2.7.1 Realistic Rendering of an Organ Surface in Real-Time 36
2.7.2 Glistening Effect in Anatomical Objects 38
2.7.3 Automatic Medical Illustration Techniques 39
2.8 Summary 41

3 Physically Based Object Deformation 43

3.1 Introduction 43
3.2 The Local Weak Form 45
3.3 Interpolation Scheme and Test Functions 47
3.4 Imposing Boundary Conditions 49
3.5 Addition/Deletion of Nodes 51
3.5.1 Large Deformation 51
3.5.2 Topological Change 52
3.6 Application in Surgery Simulation 52
3.6.1 Precomputation 52
3.6.2 Runtime Computations 53
3.6.3 Results 53
3.7 Summary 54

4 Physically Based Object-Object Interactions 57

4.1 Introduction 57
4.2 FE Analysis in Surgery Simulation 58
4.3 Method 60
4.3.1 Tissue-Tool Interactions 60
4.3.2 Tissue-Support Interactions 64
4.3.3 Tissue-Tissue Interactions 66
4.3.4 Calculation of the Penalty Force 67
4.4 Simulation Procedure 68

4.4.1 Pre-Computation 69
4.4.2 Runtime Simulation 69
4.5 Summary 71

5 Deformable Geometry Maps Representation 73

5.1 Introduction 73
5.2 Surface Parameterization 73
5.3 Interactive Deformation Methods 75
5.3.1 In-Place Deformation 75
5.3.2 Mapped Deformation 76
5.3.3 Motivation for Deformable Geometry Map 78
5.4 Geometry Map Representation 79
5.4.1 Parametric Geometry Map Representation 79
5.4.2 Resampled Geometry Map Representation 80
5.4.3 Reconstructed 3D Representation of Geometry Map 82
5.4.4 Representation of Parametric Deformation 85
5.5 Summary 87

6 Interactions with Deformable Geometry Map 89

6.1 Introduction 89
6.2 Input Data Preparation 89
6.2.1 Geometry Map Parameterization 90

6.2.2 Geometry Map Resampling 90
6.2.3 3D Reconstruction from Geometry Map 91
6.3 Interactions 91
6.3.1 Tool-Organ Interactions 91
6.3.2 Collision Detection in the Geometry Map 94
6.4 Geometry Map for Haptic Force Feedback 96
6.5 Contact Deformation Between Objects 97
6.5.1 Collision Detection 97
6.5.2 Displacement Computation 98
6.5.3 Volume Preserving Local Weighted Deformation 100
6.6 System Architecture 101
6.7 Summary 101

7 Experimental Results 103

7.1 Introduction 103
7.2 Experiment Setup for Surgery Simulation 103
7.3 Visualization of Organ Deformation 104
7.3.1 Initial Results 104
7.3.2 Cinematic Quality Rendering 104
7.3.3 High Quality Rendering 104
7.3.4 Organ-Plane Interaction 109
7.4 Analysis of Visualization Results 111

7.5 Time Results 112
7.6 Experiment Setup for Textile Simulation 113
7.7 Visualization Results of Textile Simulation 114
7.7.1 Draped on Irregular Objects 114
7.7.2 Draped on Planar Surfaces - Analysis 117
7.7.3 An Upholstery Example - Textile Draping on Chair 120
7.8 Summary 121

8 Conclusion and Future Work **123**

8.1 Contributions 123
8.2 Discussions 125
8.2.1 Imaging vs Deformable Object Modeling 125
8.2.2 Medical Visualization and Virtual Surgery 126
8.2.3 Physical Behavior of Tissues 127
8.2.4 Novelty of DGM 128
8.2.5 Qualitative Comparison of DGM with Related Work 128
8.2.6 Quantitative Comparison of DGM with Existing Techniques . . 128
8.2.7 Limitation of DGM 129
8.3 Future Work 130
8.3.1 Improvements in Geometry performance 130
8.3.2 Improvements in Physically Based Realism 131
8.3.3 Improvements in Haptics and Interaction 133

Acronyms **134**

Author's Publications **135**

Bibliography **137**

List of Figures

1.1 Foundations of virtual human modeling 3

2.1 Organ deformation modeling methods 18

2.2 The architecture of a general surgery simulation system 32

2.3 Architecture of the Haptic Browser 37

2.4 Visualization interface for ear assembly 37

2.5 Visualization of various organs during haptic interaction 37

2.6 The rendering of a liver model 38

2.7 Glistening liver rendered using a cube map 39

2.8 Different generations of surgery simulators 41

3.1 Global domain Ω, local domains Ω_s^I and Ω_s^J, and their boundaries . . . 45

3.2 Intersections of shape function in T_{sh}^J $(J = 1, 2, ..., M)$ with the test function in T_{te}^I 48

3.3 Point force exerted on the intersection of node N^C and node N^D 50

3.4 The flowchart for the deformation procedure 54

3.5 A simple example of the deformation of a cube under point forces . . . 55

3.6 Proposed effect of the simulator using MFEM 55

4.1 The displacement of the contact point 61

4.2 Decomposing the feed back force into a tangential component and a normal component 63

4.3 Determination of the new contact point Q 64

4.4 The interactions between tissue and support 65

4.5 A penalty force $\mathbf{F}_{pen}$ is applied to the node that has penetrated into the support 66

4.6 Penalty force are applied on the surface nodes of tissue A that has penetrated into tissue B. 67

4.7 The material depth and the penalty force 68

5.1 The categorization of interactive deformation methods 75

5.2 Free-Form Deformation 77

5.3 The parameterization of a sphere 81

5.4 The barycentric coordinate of a point lies inside the triangle 82

5.5 Reconstruction of the 3D sphere 84

5.6 The shape of the Gaussian distribution function in the i direction . . . 85

5.7 The deformation of the sphere with free form deformation 86

5.8 The deformation method with DGM in interactive deformation methods 87

6.1 The parameterization of a virtual stomach mesh model 90

6.2 Reconstruction of the 3D mesh 92

6.3 The deformation of the stomach model with free form deformation . . . 93
6.4 The collision map . . . 95
6.5 The collision detection between the organ and the tool . . . 96
6.6 The process of the contact between a deformable object and a plane . . 98
6.7 The flowchart to handle the contact between objects . . . 99
6.8 Architecture of our interactive system . . . 102

7.1 Surface deformation of the liver model and the stomach model . . . 105
7.2 Cinematic quality rendering of the liver . . . 106
7.3 Stomach model with texture and bump map . . . 106
7.4 The deformation of the stomach model with user interactions . . . 107
7.5 The deformation of the liver model with free form deformation . . . 108
7.6 Deformation of a heart model . . . 109
7.7 Stomach-plane contact and deformation . . . 110
7.8 The shape of the deformation in a local area . . . 111
7.9 The deformation of denim fabric wrapped on the 3D deformable model 116
7.10 The deformation of satin wrapped on the 3D deformable model . . . 117
7.11 The deformation of various textiles wrapped on the liver and stomach . 118
7.12 Variable kernel for deformation of a denim textile on a flat surface . . . 119
7.13 Variable kernel for deformation of plaid textile on a flat surface . . . 120
7.14 The deformation of textile wrapped on the deformable seat . . . 121

8.1 The comparison of the organ deformation results . . . 129

8.2 Multi-layer Deformable Geometry Maps 132

List of Tables

7.1 Virtual stomach deformed with different parameter values 113

7.2 Virtual liver deformed with different parameter values 114

7.3 Time Results 115

8.1 Quantitative comparison of DGM to existing techniques 130

Chapter 1

Introduction

1.1 Motivation

In recent years, laparoscopic surgery is becoming more and more common for many procedures. In laparoscopic surgery, abdominal operations such as hepatic (liver) resections are accomplished through small incisions. The abdomen of the patient is inflated with gas to create open space inside and a video camera is inserted into the abdomen through one of the small incisions. The video image is magnified and transmitted to a high-resolution monitor, allowing the surgeon to see the abdominal anatomy with great clarity. The surgery is performed using special instruments introduced through the incisions. With this technique, patients benefit from less pain and less strain for the organism, and faster recovery and shorter hospitalization time. The drawback of this technique is that it is more difficult than traditional surgery procedure, and the surgeons need to learn and adapt themselves to this new type of surgery. To master laparoscopic surgery, massive training and practice are required.

The use of plastic models or animal cadavers in conventional surgery training does not fully reflect the real-life situation in the operating room as there is not realistic tissue and physiological behavior in the models. Setting up and disposal of the animal

cadavers are also difficult and expensive. Most importantly, the use of such models and animal cadavers does not allow repeated training on the same scenario and does not allow for objective assessment of the training success. Due to above-mentioned reasons, surgery simulator becomes necessary for training surgeons in the advanced and latest surgical procedures.

Noninvasive imaging techniques such as Computed Tomography (CT) scans and Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET), Ultra Sound Radiography have been revolutionary in medical imaging visualization for diagnostic and surgical applications. With advancing technology, it has become viable to visualize the change in the geometry, appearance and texture of deformable tissues [86], [72], [49]. Visualization of dynamics in the medical data leads to enormous applications in the area of virtual reality based surgical simulations in assisting surgical training, surgical planning, pre-operative rehearsal, and intra-operative execution. With the latest medical imaging techniques and VR technology, surgery simulators can provide cheap and intensive training without the need of cadavers or animals [60].

We feel that there is a need to build a high quality surgery simulator. We have designed a framework and the components of our framework have been evaluated for both rigid as well as deformable organs. Immediate application of the virtual surgery system can be used for training medical students, with long term applications in everyday planning of pre-surgery, post-surgery and also during live-surgery.

1.2 Possible Interactions

In a surgery simulator, the possible interactions that have to be taken into account include:

Deformable Interaction: The physical modeling of a simulator includes the modeling

of contacts between virtual instruments and soft tissues as well as the biomechanical deformation of soft tissues. When collision between an anatomical structure and a surgical instrument is detected, the boundary constraints on the soft tissue models are updated and, depending on the nature of the instrument, a piece of tissue is removed. The soft tissue is then deformed according to a given biomechanical behavior.

Cutting Interaction: By restricting the regions where cutting is allowed, it is therefore possible to include in the simulator deformable models of large size.

Background Structures for Better Interaction Cues: For more realistic representation of the operative field, other elements are introduced.

1.3 Challenges

The difficulty of virtual human simulation comes from the fact that human bodies are extremely complex entities, and modeling of virtual human is closely related to subjects besides computer science, such as medicine and biomechanics (See Figure 1.1). Until now, the simulation of accurate human bodies still remains one of the greatest challenges in computer graphics.

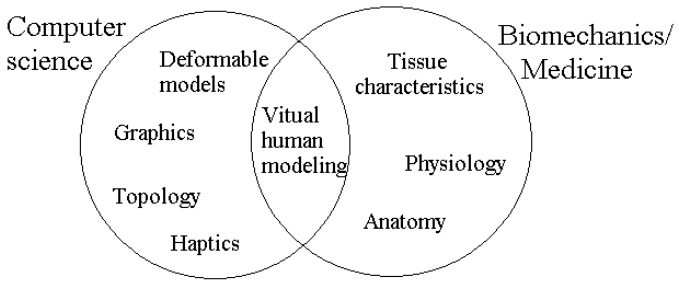


Figure 1.1: Foundations of virtual human modeling.

Simulators have been used in the area of training at medical institutions or in preoperative procedures for surgery planning [48], or to predict the outcome of an operation, for example, Computer-Aided Plastic Surgery used by Pieper et al. [68], and the cleft clip surgery simulation by Sheppard [76].

In such applications, visualization is tightly integrated with interaction, haptics, as well as volume deformation.

A major issue of surgery simulation is to model the virtual human organs in a realistic way, both visually and physically, and there must be an element of unpredictability. For realistic visual rendering, advanced graphics techniques like texture map, normal map, light map, and bump map can be utilized. For the physical behavior, information related to the elastic property of the tissues is needed so that the virtual organ can be deformed in the interactive simulation. The challenges we face for the realistic visualization of human organs are detailed below.

1.3.1 Static Model Representation

Static model representation refers to the geometry and appearance (normals and textures etc.) of the organ. The traditional model representations such as complex triangle meshes or NURBS surfaces pose a variety of problems when it comes to interactive deformation. Some of the problems encountered are: determining the contact point, application of textures, difficulty in deformation, and non-real-time response for interaction.

- *Mesh Primitives:* Traditionally, the input data for modeling 3D deformable objects have been 3D meshes and primitives which are represented as a collection of triangles. This representation seriously limits the picking of one or more triangles out of a large mesh. It is also extremely difficult to determine the neighborhood

points or triangles associated with a given contact point.

- *Contact Point:* Determining the contact point in 3D space is tedious. This requires ray intersection with the object of interest to find the closest intersection point.

- *Deformation Footprint:* Even though the footprint kernel of shape deformation in picking or pinching is quite uniform for most objects, representing the orientation of the shape of the kernel in 3D is difficult.

- *Interactive Deformation:* To visualize the deformation, we need to have an immediate feedback to facilitate responsive user interaction with the 3D deformable object.

1.3.2 Deformable Organs

There exist a number of approaches to model the deformation of the organ [40], including Free Form Deformation (FFD), Mass-Spring Models, and Finite Element Methods (FEM).

- **Human Soft Tissue Simulation:** The appearance of the soft tissue is obtained from real textures or images of the organs. This texture is mapped onto a polygonal mesh of the organ that is under examination. The deformation of the mesh is generally carried out using a range of techniques from free form deformation, mass-spring models or even realistic FEM models. To create flicker free movement and collision detection, we need to compute 30 frames/sec for which the computational requirements are high.

- **Tissue Tool Interaction:** The interaction between the tool and the organ is a challenge to compute. The position of the tool has to be estimated in relation to

the organ first. The behavior of the tissue has to be simulated in real-time when the tool interacts with it (e.g. cut, prodded or punctured).

1.3.3 Cinematic Quality Rendering

Rendering scenes that simulate realistic surgery depends on several factors. These include lighting, choice of tissue and organ material properties. Apart from this the system should create photorealistic renderings in real-time. These functions should not only be for static organs or models, but also include complete deformable shape modeling and animation. The renderer to achieve the best quality computes, (i) reflection, (ii) refraction, (iii) high quality shading effects, and (iv) anti-aliasing.

To attain a cinematic quality visualization of deformable tissue, one should achieve 30 frames/sec that will give a flicker free movement and will require very high computational power and memory. The challenge is to find optimized visualization algorithms to achieve the cinematic quality visualization of deformable tissue by exploiting maximum performance out of the available computing resources. This study will discuss various visualization algorithms for achieving cinematic quality visualization of body parts and deformable tissue.

1.4 Contributions

In addressing these challenges, this thesis makes the following new contributions.

- **Deformable Geometry Map Representation:** We have proposed a new Deformable Geometry Map (DGM) for laparoscopic surgery simulation. Arbitrary input virtual organ meshes are parameterized and resampled into regular high resolution models. The high resolution models increase the visual quality of the virtual organs. The parameterized mesh is helpful for the simulation in a few

aspects: collision detection, free-form deformation, simulation of the interaction between the tool and organ, and an update rate fast enough to provide haptic feedback. The DGM representation also helps to handle the contact between the organ and another object.

- **Deformation of Geometry Maps:** Secondly, we introduce a free-form deformation approach for the parameterized mesh to simulate the interaction between the tool and the organ to achieve an update rate fast enough to provide haptic feedback. The parametric deformable shape representation described in this work helps to approximate the local deformation. Advantage of the local deformation now includes the ability to interact with the 3D shape model using the parametric space instead of the 3D mesh. As a result, the deformation computation need not be performed on the entire mesh.

- **Interaction with the Deformable Mesh:** We also propose a procedure to handle the collision between the tool and the object surface as well as the contact between the object and another object.

- **Texture Deformation:** The appearance of the 3D mesh is accomplished using appropriate textures. Because of the parameterization, the texture mapping is implicit. In our examples, the texture mapping works well for the deformable surfaces.

- **DGM for Surgery Simulation:** We have demonstrated the parameterized representation of 3D organ meshes for the simulation of laparoscopic surgery. Unstructured input virtual organ meshes are parameterized and resampled into regular high resolution models. The high resolution models increase the visual quality of the virtual organ. The parameterized mesh is helpful for the simulation in many aspects.

- **DGM for Textile Deformation:** We have demonstrated the parameterized representation of 3D meshes for the simulation of deformation of textiles. The high resolution models increase the visual quality of the virtual textiles. The parameterized mesh is helpful for the simulation as shown in our examples.

1.5 Thesis Outline

The rest of the thesis is organized as follows.

- Chapter 2 gives a background and introduction to the challenges in designing surgery simulators. Section 2.2 gives an outline of static organ shape modeling where the focus is the geometry and appearance of the organ model. In Section 2.3 we present an extensive introduction to the modeling of the behavior of organs. The dynamic behavior model includes the specifications of soft tissue, deformation modeling for specific organ and volumetric behavior. Section 2.4 discusses organ shape deformation from two perspectives. First is visual feedback on deformation when a tool interacts with an organ, and second is the time to compute deformation that is necessary to provide an appropriate haptic feedback. Section 2.5 presents the various surgery simulation techniques that have been proposed recently. In Section 2.6 we analyze the computational process or architecture design in constructing surgery simulators. One of the important characteristics that we feel necessary in surgery simulation is cinematic quality rendering which is outlined in Section 2.7.

- In Chapter 3, we introduce the physically based object deformation. We discuss the formulation of the interpolation scheme, boundary conditions, specification and editing of nodes in the case of local and large deformation. Acceleration schemes such as precomputation, and runtime optimization are also discussed.

- In Chapter 4, we discuss about the object-object interactions in surgery simulation with the physics based object deformation.

- In Chapter 5, we introduce a novel deformable geometry map (DGM) representation. Unlike previous parametric representation, we show our representation allows control of deformation in real-time. Our method is novel as we do not perform the 3D deformation directly, in-place on the 3D mesh.

- In Chapter 6, we exploit the DGM representation for interactions (e.g. a tool-organ interaction). First we show how DGM representation can be used for collision detection of the tool with the organ. This interactive deformation on DGM can be done in real-time, which enables both visual feedback and haptic feedback. Finally we also show that DGM representation can be used for contact detection and collision detection between objects (e.g. an organ and a plane).

- In Chapter 7, we use our interactive DGM representation for two distinct case studies. First we show that real-time deformation results achieved are realistic for a range of organs. The results are also visually better than other deformation approaches known to us. Furthermore, we show results from textile simulation that validates the usefulness of our interactive DGM representation.

- Finally, Chapter 8 closes with a summary of the main topics, a discussion of commonly raise questions, and further issues that remain to be investigated in the future work.

The existing literature on surgery simulation, physically based deformation and parametric modeling is large and varied. Reviews of literature will consequently be spread throughout the thesis. Chapter 2 contains the relevant citations for surgery simulators, organ shape modeling, and deformation. Chapter 3 contains one section, a

report of relevant background work on physics based meshless deformation. Chapter 5 contains one section on related work on parametric modeling and free form deformation.

Chapter 2

Literature Review

2.1 Introduction: Challenges in Designing Surgery Simulators

One of the biggest challenges in a surgical simulator is real-time deformation of soft-tissues. The challenges on real-time deformable models for surgery simulation have been presented in 1993 by Cover et al. [26] and more recently in a survey by Meier et al. in 2005. [59]. The reconstruction of objects using these models has two conflicting characteristics: interactivity and motion realism. Meier et al. [59] also present the advantages and disadvantages of each of the various deformable models. The focus of comparison is mainly on three key requirements of simulating deformable tissue for surgical applications described as computational demand, topological flexibility and biomechanical realism.

Keeping the above parameters in mind, there is no outstanding model that comprises all the parameters for surgery simulation of deformable organs. The evaluation clearly shows that interactivity and biomechanical realism are two conflicting characteristics of deformable models. Any one of them is promoted to the detriment of the other. Interactivity is mostly chosen for real-time surgical simulators.

This implies that the choice of a model is purely dependent on the feature to be emphasized during the surgery simulation. The complex nature of certain models may pose difficulty in implementation. Mass-Spring models, being the easiest to create and modify, have found extensive use in real-time interactive and haptic surgical simulations. Limitations may also be posed by present computer hardware configurations in terms of computational power and memory.

Haptic rendering on the other hand requires much higher level of physical realism and much faster update rate than graphic rendering to achieve high quality and realistic rendering, which is essential for many applications that simulate manipulations in real physical world. However, high-level of physical realism and fast update rate of rendering are often conflicting requirements.

The rest of this chapter is organized as follows. Section 2.2 gives an overview of the challenges involved in organ shape modeling for surgery simulators. Section 2.3 presents a survey of the state-of-art techniques for modeling the deformation of organs. Section 2.4 presents a survey of related work on haptics and interactive deformation. Section 2.5 presents the various surgery simulation applications that have recently attracted the attention of researchers. Section 2.6 presents the major architectural components and computation steps in surgery simulations. Section 2.7 presents some of the early attempts on cinematic quality visualization, and Section 2.8 presents a summary on the three generations of surgery simulators.

2.2 Static Organ Shape Modeling

In this section we describe some of the recent techniques for shape modeling of organs used in surgery simulators. Data modeling for visualization has been a research issue for many years [63], and more recently modeling for virtual environments have been

studied by Zhong et al. [90].

2.2.1 Geometric Modeling in Hepatic Simulation

Geometric modeling in hepatic surgery simulation involves the 3D reconstruction of the principal hepatic structures of interest – the hepatic parenchyma, the main vessels, the Couinaud (or functional) segments, and the potential lesions. The processes involved in the creation of a 3D anatomical model of the liver from the preoperative CT images are as follows:

- Hepatic parenchyma extraction is achieved by applying a deformed simplex mesh geometrical model to fit the corresponding CT image.
- Main vessels extraction is based on the exploitation of mathematical morphology and digital topology techniques.
- Couinaud segments extraction of the eight segments is based on the computation of the first branches of the vein, which is similar to a generalized Voronoi diagram.

Marching cubes algorithm is commonly employed for surface rendering or extraction application. For example, the extraction of the external surface of the hepatic parenchyma could be achieved by applying a sub-voxel triangulation provided by the marching cubes algorithm.

However, such application is greatly compromised by the following two limiting factors: (1) the number of triangles generated is too large for further computer processing (i.e. high computational costs), and (2) a smoothing of the extracted surface is necessary to avoid staircase effects. Therefore, Delingette and Ayache [36] applied *simplex meshes* as opposed to marching cubes algorithm for liver surface extraction.

2.2.2 Surface vs Volumetric Soft Tissue

The geometric representation of deformable tissue may consist of surfaces or volumes. The choice between surface and volume based models is governed by two factors: computational efficiency and physical accuracy. In terms of computation, surface models are advantageous because they have fewer vertices than volumetric models for representing the same shapes.

However, most biomechanical models, naturally call for volumetric representations rather than surface models. Surface models tend to give physically invalid deformations especially in regions that are thin. Furthermore, the behavior of volumetric models may take into account physical inhomogeneities, for instance due to the presence of lesions. Volumetric models are better-suited to the simulation of cutting or suturing operations. The operations change the geometrical and the physical nature of the model. However, surface models may be relevant for modeling cavernous tissues such as vessels or the gallbladder. In this case, the physical model of deformation can incorporate a representation of a liquid or of a gaseous pressure combined with surface tensions.

2.2.3 Dynamic Organ Geometry Modeling

Surgery simulation in the near future will focus on intraoperative surgery planning. The main issue in intraoperative surgery is to model the geometry shape of moving anatomical structures in order to estimate the deformation of the structure. Fused visualizations of preoperative and intraoperative images have also been modeled to help the surgeon in surgical decision making. The main emphasis of Warfield et al. [89] is on capturing intraoperative deformations of anatomical structures so as to help in image-guided therapy.

1. Unstructured mesh creation and surface representation: a volumetric tetrahedral

mesh is constructed throughout the brain. The mesh generation should be rapid as well as accurate enough to match the patient's geometry. Warfield et al. [89] have developed a tetrahedral mesh generator, which is the volumetric counterpart of marching tetrahedral surface generation algorithm.

2. Rigid registration of preoperative volumetric images to intraoperative volumetric images: a parallel registration algorithm has been used to correct the rotational and translational difference between the preoperative and intraoperative sets of data. This algorithm identifies an optimal set of transform parameters that maximizes the spatial overlap of segmented structures in the two data sets being aligned. The registration method used models the surface as a thin elastic sheet, and produces a mapping that registers the two surfaces in a way that is one to one and onto.

In summary, the shape modeling of organs for surgery simulation falls under three major groups. They are

- generic organ models for surgery simulation,
- multi-layered patient specific organ modeling, (or)
- intraoperative modeling of organs.

2.3 Dynamic Organ Deformation Modeling

In this section we will look at the organ deformation modeling in more details. Some examples of deformation modeling methods used in surgery simulation are investigated. A more comprehensive review of general interactive deformation methods will be presented in Chapter 5.

There exist a variety of approaches to model the deformation of 3D objects. Depending on the geometric representation, they can be generally classified into two categories:

deformable modeling with surface representations and deformable modeling with volume representations. These two categories of deformable modeling approaches will be discussed in general in Section 2.3.1 and Section 2.3.2 respectively. After that a few examples of deformable modeling methods that are used in surgery simulation will be studied from Section 2.3.3 to Section 2.3.8.

2.3.1 Deformable Modeling with Surface Representations

Gibson and Mirtich have surveyed various deformable modeling methods with surface representations introduced before 1997 [40]. Approaches including Free Form Deformation (FFD), mass-spring models, and Finite Element Methods (FEM) were studied with their advantages and limitations compared. Among them, FFD are classified as non-physical models, and mass-spring models and FEM are considered as physically based models.

While non-physical models has the advantage of computational efficiency, physically based models in general provide more accurate deformation, hence attract more research attention. More recent developments in physically based deformable models were reviewed by Nealen et al. [61], which included the method of finite differences, the finite volume method, the boundary element method, mesh-free methods, reduced deformation models, etc.

For applications in surgery simulation, the deformation must be simulated interactively. The real-time simulation of soft tissue deformation is still the major obstacle while developing simulator systems for soft tissue surgery. Meier et al. have surveyed various real-time deformable models that can be applied to surgery simulation [59]. These approaches are compared in terms of computational demand, biomechanical realism, and topological flexibility. Some examples of these techniques will be discussed later from Section 2.3.3 to Section 2.3.8.

The complex behavior of deformable tissue has posed many challenges to the physical modeling of human organs. The need for an accurate surgical simulation system has created much interest in researchers worldwide in developing new techniques for simulation of deformable models. The most successful deformable models in future would be those which can provide greater interactivity without compromising on the physical realism.

Mass-Spring Models: The biggest bottleneck for real-time simulation is the large global deformation of human organs during surgery. The ideal improvement for mass spring models may be to include volumetric techniques for representing large deformations. Hence the linked volume approach has the potential to offer this behavior.

Finite Element Models (FEM): The continuum-mechanical algorithms have to be improved for better performance in simulation applications. Efficient pre-processing techniques are required for real-time interaction of FEM models. Hence FEM models will continue to attract attention.

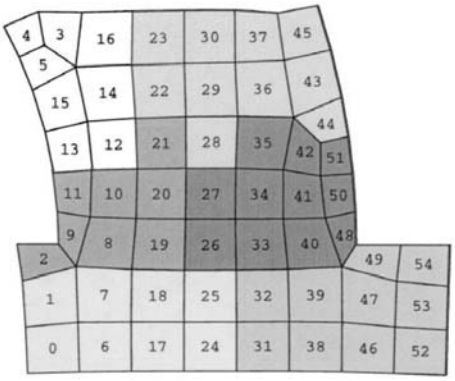
Performance: Collision detection is an important factor when considering organ-organ or organ-tool interactions. Fast and efficient algorithms are required to include these modeling forces.

Finally, with the advancement of computer hardware, accelerated computations will be available in future which may enhance the performance of computationally intensive models.

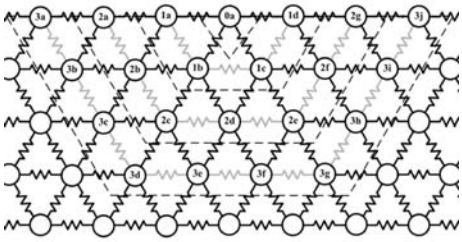
Some example of organ deformation modeling methods are shown in Figure 2.1.

2.3.2 Deformable Modeling with Volume Representations

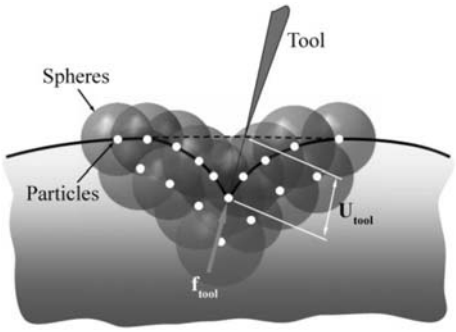
Unlike surface meshes, volume representations lack in geometrical topological and semantic information, which is much required for controlling deformation and animation.



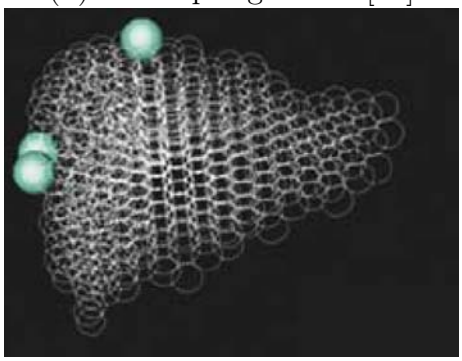
(a) Finite element model [83]



(b) Mass-spring model [20]



(c) Point-associated finite field (PAFF) model [34]



(d)Sphere-filled model [81]

Figure 2.1: Organ deformation modeling methods.

Chen et al. surveyed a wide-range of techniques that had developed for manipulating, deforming and animating volume date [19]. However, as pointed out by Chen et al., most of the deformation techniques for volume representations still relies heavily on those originally developed for surface representations. Especially, the physically-based models are mostly adopted directly from the deformation methods for surface representations.

Nevertheless, there are still a few deformation techniques which are unique to volume representations, including chain-mail algorithm and spatial transfer functions. In Section 2.3.4, an approach to simulate brain tissue deformation with volume data will be investigated.

2.3.3 Deformation Modeling in Hepatic Simulation

Delingette and Ayache [36] introduced a hybrid model for optimizing the computation of a realistic hepatic surgery simulation based on the FEM. The hybrid model combines two linear elastic models, namely the quasi-static pre-computed elastic model and the dynamic elastic model (or tensor-mass model).

The quasi-static pre-computed elastic model is suitable to model anatomical structures that only need to be visualized or deformed that greatly contribute to the visual realism of the simulation but where surgery does not occur. The pre-computation model assumes that the topology of the model mesh remains the same during the simulation. Such assumption obviously prevents the simulation of cutting, tearing or suturing gestures. Moreover, this method is also a quasi-static method, as it computes directly the equilibrium solution produced by the application of the instruments against the liver surface. Therefore, it does not take into account the visco-elastic nature of the soft tissues [36].

Inversely, the dynamic elastic model authorizes topology change on anatomical structures where tearing and cutting need to be simulated. This model implements a discrete approximation of a volumetric and continuous mechanical behavior that simulates the surgery gestures [36]. However, the iteration requires extensive computational time.

Therefore, the hybrid elastic model that combines the two models, which optimizes the trade-off between visual realism and computational time of the surgical simulation. The completed hybrid mesh of the liver model contains 1537 vertices and 7039 tetrahedra. About 18% (280 vertices and 1260 tetrahedra) of the liver hybrid mesh was modeled as tensor-mass system and the remaining as pre-computed linear elastic model.

2.3.4 Dynamic Volumetric Simulation of Deformation

In Warfield et al. [89], during the process of neurosurgery, the brain undergoes deformations due to many mechanical factors. Moreover, for continuous monitoring after the surgery, new volumetric MRI data is needed. From this data, the volumetric deformation field that warps the previous data into the new data set is estimated by solving a biomechanical model that simulates the brain tissue deformation. Since there is no access to good estimates of the true intraoperative pressures and forces acting upon and throughout the brain, imaging data is used to establish boundary conditions.

The approach is to estimate a volumetric deformation field using biomechanical model of the properties of the tissue throughout the brain. Previously, active surface algorithm has been used. It identifies the surface of the ventricles and brain explicitly in the data to be warped. Forces are then applied in an iterative fashion to drive the surfaces towards the target. When the surfaces match the edges of the target volume, they have to be minimized. Hence they are derived from a function of image intensity gradients. In the recent work they have used conformal mapping strategy to map the preoperative MRI of the prostate to intraoperative MRI of the same. The most straightforward model is to treat the brain tissue as a homogeneous linear elastic material. This model has also been extended to account for the inhomogeneous material characteristics of heterogeneous white matter of the brain by modeling the brain tissue as a transversely anisotropic linear elastic model.

Visualization of the fused data is done in the co-ordinate system of the intraoperative data. Once the above steps are carried out, the enhanced visualization is presented to the therapy team doing the operation. The displayed images are continuously updated with the most recent images obtained by intraoperative MRI. During the entire course of operation, this enables the surgeon to visualize the position in the anatomy and the

deformation caused by the tools at that point. The above procedure has also been carried out for capturing intraoperative deformations of MR-guided prostate biopsy.

2.3.5 Mass-Spring Mesh for Vessel Deformation

Brown et al. [14, 13, 15] represented the geometry of the deformable vessel by a 3D mesh of nodes (mass points) connected by spring links. The nodes and links on the object surface (vessel wall) are triangulated for graphical purpose, whereas the other points and links are unrestricted. The mechanical properties (viscoelastic, in most surgical simulation applications) of the object are described by data stored in the nodes and links.

To achieve real-time performance, the algorithms take advantage of several characteristics of surgical training [15]:

1. Visual realism is more important than accurate, patient-specific simulation.
2. Most tissue deformations are local.
3. Human-body tissues are well damped.
4. Surgical instruments have relatively slow motions.

2.3.6 Deformation with Sphere-Filled Organ Model

Suzuki et al. [81] introduced a deformable organ model using a sphere-filled method instead of the finite element method. The sphere-filled organ model consists of a group of small spheres inside and a polygonal surface mesh outside. The element spheres are structured in layers, with the outer most spheres defined as the first layer, all spheres neighboring the spheres of the first layer are defined as the second layer, and so on.

When an external force is applied to the model, the movement of each element sphere deforms the shape of the surface mesh. The spheres defined as layer one are

displaced from their positions in the direction that avoids the effect of an external force. Other element spheres are displaced in turn by measuring their distances from the neighboring element spheres, so that no spheres would overlap with each other. When an incision is applied, layer one spheres are displaced from their positions in the left and right directions of the incision plane. A new incision plane is created as the incision of the surface mesh. Thus, various physical operations that involve pressing, pinching, or incising an organ's surface can be performed. Furthermore, the model is equipped with a sense of touch by connecting it to a force feedback device.

2.3.7 FEM with Parallel Computing

One way to provide the necessary computational power for the real-time solution of complex finite element systems is to build a parallel computer which supports fully parallel algorithms for the explicit time integration scheme. Székely et al. [83] built a parallel computer with 64 processors ($4 \times 4 \times 4$) to simulate the deformation of a uterus model with about 2000 elements. The model is partitioned for parallel computation, with each element mapped to exactly one processor, and one processor computes the internal forces of several elements. Realistic deformation can be achieved with this method. However, the high requirement on the hardware is prohibitive. Moreover, to simulate a more complex model with larger number of elements, more processors are required for the parallel computer, and the parallel algorithm needs to be adapted as well. Therefore it is not practical for complex procedures.

2.3.8 Real-Time FE for Virtual Suturing

To improve the computational efficiency of FE analysis, novel preprocessing techniques and alternative real-time solving methodologies are utilized. Berkley et al. [10] presented a new real-time methodology based on linear finite element analysis that is

appropriate for a wide range of surgical simulation applications.

The principle of *calculate only what you need* has served as the basis for the real-time FE approaches. A condensation process is applied to reduce the problem greatly so that only the displacement of the nodes that can be seen or touched is calculated. After condensation, the reduced stiffness matrix is inverted to permit the calculation of visible nodal displacements. In addition, a constraint approach is adopted, so that contact can be accommodated not only for the nodal points, but anywhere on the surface of the model.

The methodology is characterized by high model resolution, low preprocessing time, unrestricted multi-point surface contact, and adjustable boundary conditions. These features make the method ideal for modeling suturing.

2.4 Haptics and Interactive Deformation

Interaction and haptic feedback are paramount in surgery simulators. The focus here is to redesign the algorithms that simulate physical behavior of organs in real-time.

2.4.1 Localized Deformation with Finite Points

In spite of the great success of the finite element method as effective numerical tools for the solution of boundary value problems on complex domains, there has been a growing interest in the so-called *meshless* methods over the past decade. The primary reason for the interest in meshless methods is that in the finite element method a mesh is required. De and Bathe [29] introduced a meshless technique, called the *method of finite spheres*, which is truly meshless in nature in the sense that the nodes are placed and the numerical integration is performed without a mesh.

De et al. [30, 32, 33, 34] and Basdogan et al. [8] extended this meshless method and

developed a novel point-associated finite field (PAFF) approach for real-time deformation of soft tissues. In PAFF, De et al. proposed two real-time acceleration techniques. One is fast re-analysis technique based on localized interaction assumption and pre-calculation of stiffness matrix. The other involves much less nodal points in the vicinity of tool-tip and can minimize the size of stiffness matrix. It is empirically accepted that minimum update rates of 30 Hz and 1000 Hz are required for a stable and consistent visual and haptic feedback [16, 20, 80], respectively. Lower update rates would cause visual or haptic artifacts. The two proposed techniques are suitable for a real-time application.

Both techniques are based on the vicinity interaction assumption which assumes the interaction of surgical tool-tip and soft tissue are local and the deformation dies off rapidly with increase of distance from tool-tip. Although this assumption follows the physical nature of elastic organ, the acceleration methods will produce computational errors which increase with distance from tool-tip. Moreover, it is inaccurate to treat every organ the same since they vary in elastic and viscoelastic behavior.

The advantage of fast re-analysis technique is benefited by the pre-calculation of stiffness matrix, which may cost large amount of computation if the number of involved nodal points is over thousands. The localized computation overcomes the problem of big-sized stiffness matrix while, of course, only acceptable in small vicinity of tool-tip. De et al. proved that if given the same allowable computation limit, the fast re-analysis method can solve the real-time calculation with more degrees of freedom (i.e. more involved nodal points) than the localized solution. It means the fast re-analysis technique is more competent, in relative small sized problem, than localized in real-time simulation.

2.4.2 Localized Deformation with Beam-Skeleton

By introducing a novel beam-skeleton model, Luo and Xiao [55] computed the stresses and strains of a deformed elastic object at certain extremal points as well as the stresses at multiple contact regions. Based on this model, the paper further introduces fast computation of global shape change through an interpolation method that achieves minimization of elastic energy. Moreover, the paper takes into account the different effects of different contact areas on shape change under the same force.

The current techniques provide the solution to deal with deformable objects under complex contact states involving multiple contacts and compliant motions with friction. It is new and different with the majority of previous approaches which assume single contact region and localized deformation.

By introducing a novel approach, modeling and rendering in real-time both the nonlinear contact force response and the shape deformation of a general elastic object caused by a rigid object contacting it and moving compliantly on it, are realized. This approach takes into account friction.

Both real-time efficiency and physical accuracy are achieved by taking advantage of nonlinear physics equations, elasticity principles, beam bending theory, and geometrical properties of general surfaces.

The implementation results show that the new approach is effective, which reached an update rate of over 1 kHz for the entire rendering process, including collision detection and rendering both haptic force and graphic shape change.

2.4.3 Deformation with Force Transmittal Mechanism

Choi et al. [21] use a Force Transmittal Mechanism (FTM) that employs the Breadth-First Search (BFS) algorithm to identify the nodes to be involved in the deformable

simulation. On top of that, an artificial intelligence technique using Simulated Annealing (SA) algorithm has also been developed to identify and optimize the deformable model parameters.

The idea of the Force Transmittal Mechanism is to model deformation as a process where forces are transmitted to individual mass points in a mass-spring network.

To realize the Force Transmittal Mechanism, Breadth-First Search algorithm determines the propagation order among the mass points. During the search process, a node that has been visited previously will not be re-visited. For example, Choi et al. [20] introduced a mass-spring system to model soft tissue. Tissue deformation is simulated as a process of force propagation among the mass points. The deformation is scalable simply by controlling the penetration depth. Stiffness matrix formulations and operations are not needed, and the number of nodes involved in the deformable simulation is relatively small. Therefore the update rate is enough for haptic rendering.

In actual implementation, the number of nodes increases from 2D to 3D. Thus, the author scales the simulation simply by controlling the penetration depth of the nodes. The connectivity information is also stored in the data structure of each node to reduce the computational time.

To further optimize the deformable model parameters, Simulated Annealing (SA) heuristic method is used. The main advantage of SA is that it offers a higher chance of convergence solution to reach global minimum based on the cost function definition as compared to other optimization techniques. The optimization process involves two stages: during mass-spring-based force transmittal mechanism (MSM-FTM) deformable simulation and this simulation is repeated for the same object under same conditions using the linear static FEM (as the benchmark).

On the other hand, force feedback is also integrated into the VR system to enhance

the realism of virtual environments to provide better sense of touch for diagnostic purposes. Force feedback is incorporated into the force transmittal model using the six degrees of freedom (6-DOF) haptic device PHANToM Desktop multi-threaded platform with 1 KHz refresh rate and the authors presented experiments that report reasonable and acceptable results. Also, the basic shape used is a hexahedron, containing 8 corner nodes with 3-DOF for each node in the author's work. The choice of hexahedron is to have the same topology as the mass-spring lattice that needs to be optimized.

2.4.4 LOD in Haptic Deformation

To reduce the demand in deformation computation, a Level-Of-Detail (LOD) in haptic rendering is introduced by Pyandeh et al. [65]. The interaction between the graphical mesh and the haptic mesh as a function of various levels of subdivision is done by adjusting model parameters such that the user feels the same reaction force for a given deformation, regardless of the level of local subdivision.

2.5 Surgery Applications

2.5.1 Bone Surgery Simulation

Petersik et al. [67, 66] presented a haptic rendering algorithm to simulate petrous bone surgery based on a multi-point collision detection approach which provides realistic tool interactions. With the simulated virtual drill, bony structure can be removed and paths to the middle ear can be studied. Volume model from CT scan data is directly used. Both haptics and graphics are rendered at sub-voxel resolution, which leads to a high level of detail and enables the exploration of the models at any scale.

Modeling of 3D shapes for orthopedic surgery simulation has gained lot of attention [77]. This is of interest for the design of biomedical implants and prosthetic devices.

Another area of interest is to model geometric implants and embed them in medical data, which will help in orthopedic surgery simulation. Sourin et al. [78, 79] developed a virtual reality system called Virtual Bone-setter which can provide training for improving skills and efficiency of orthopedic surgeons in internal fixation of bone fractures.

2.5.2 Virtual Endoscopy

Virtual endoscopy is a convenient alternative for minimally invasive procedures. In the past few years, virtual endoscopy modes have been widely used in virtually every commercial medical imaging software [7], for training, planning and diagnosis without an actual invasive intervention. Usually, minimally invasive procedures are performed using an endoscope, which is a fiber optic device that is moved to the target area. Beside the light source, an endoscope consists of the optic fiber for a camera to transport the acquired image to a monitor, and has one or more *working tubes* that are used to move tools such as pliers, to the target area.

Virtual endoscopy is based on a 3D scan of the respective body region. Based on the resulting volumetric data, the organs of interest are visualized and inspected from interior viewpoints. Bartz [7] investigated various concepts used in current virtual endoscopy systems in research and products, and how they might be applied to daily practice in health care.

One application of virtual endoscopy is virtual colonoscopy [48]. In virtual colonoscopy, physicians are mainly interested in visualizing the inner surface of the colon to detect polyps. A flight through the colon using a common endoscopic view shows only a small percentage of the inner surface. Vilanova and Gröller [87] proposed two methods to virtually unfold the colon for a more efficient way to look at the inner surface. One method unfolds the colon locally using local projections, and the other one obtains a

global unfolding of the colon.

2.5.3 Bronchoscope Simulation

Bronchoscopy is a special form of minimal-access surgical procedure for examining the trachea, bronchi, and air passages that lead to the lungs. Deligianni [35] emphasizes that due to the complexity of instrument control, restricted vision and mobility, and lack of tactile perception, bronchoscopy requires a high degree of manual dexterity and hand-eye coordination. As with other forms of minimal-access surgery, the required surgical skills are normally obtained through practice on inanimate plastic models and subsequently on patients. However, with plastic models, it is difficult to provide the high-fidelity physical responses that are necessary for advanced skills training and assessment, while practicing on real subjects prolongs the examination time, can involve considerable discomfort to the patient, and has certain risks of complications. With the maturity of augmented-reality systems, there has been an increasing demand for integration of computer simulation in certain aspects of this training, particularly for developing hand-eye coordination and instrument control.

2.5.4 Cleft Lip Simulation

Virtual surgery has been used to bring back smiles in cleft lips by Sheppard and Potel [76]. Three-dimensional (3D) computer animation is gaining popular to demonstrate surgical techniques. 3D visualization would be particularly useful as a teaching tool for cleft lip and palate surgery, which involves multiple flaps in a complex 3D environment. The paper presents the power of virtual technology and advanced 3D animation to articulate surgical and anatomical concepts.

2.5.5 Liver Surgery Simulation

Delingette and Ayache [36] outlined the creation of a hepatic surgery simulator for training the physicians. This is helpful in performing invasive surgical procedures thereby reduce surgeon's learning curve. Their primary motivations were the importance of liver pathologies and the inherent complexity of hepatic surgery. They realized that modeling the interaction with the deformable organs of the abdomen was a very challenging research problem which they wanted to study thoroughly.

2.5.6 Intestinal Surgery Simulation

Raghupathi et al. [71] presented the design of a VR-based trainer for laparoscopic colectomy, a surgery procedure to remove colon cancer. As a part of the current training procedure, surgeons perform the procedure on pigs to get a feel for the organ's behavior. However, this procedure is prohibitively expensive and also raises numerous ethical issues for conducting the training on live animals. A VR-based simulator platform can significantly help nonspecialist surgeons to acquire the necessary surgical skills in a cost effective way. Although the current scope of the research work does not include the simulation of the cancer removal itself, Raghupathi et al. [71] managed to simulate the behavior of the intestine.

2.5.7 Virtual Suturing and Knot-Tying Simulation

Suturing is a task fundamental to almost every surgical procedure. Surgery on the skin ranges from simple suturing of lacerations to complex tissue movements such as flaps. A suturing simulator requires the following capabilities [10, 51]:

- Accurate deformation and force-feedback.
- Contact at any point on the model surface.

- Adjustable boundary conditions.
- Multipoint contact.
- Rapid preprocessing.

Berkley et al. [10] built a real-time FE model with a constraint-based methodology for suturing simulation, which accommodates most of the objectives defined above.

Suturing inevitably involves knot-tying, which involves contact detection and management. Some practical knots can only be achieved by complicated crossings of the rope, yielding multiple simultaneous contacts, especially when the rope is pulled tight. Brown et al. [12] presented novel algorithms for the real-time graphical simulation of a rope like object, with focus on tying knots. The algorithms are based more on the geometry of the rope than on physics, nevertheless, they handle important physical properties of knot tying, such as allowing objects to slide along each other without passing through or penetrating each other.

2.5.8 Microsurgery Simulation

Microsurgery is a well-established surgical field which involves the repair of approximately 1mm vessels and nerves under an operating microscope. It is a necessity in many reconstructive procedures, including the successful reattachment of severed digits. In the procedure, the ends of two vessels are sutured together with several stitches. Microsurgeons typically acquire their initial skills through months of practice in an animal lab. Without practice, these skills can quickly degrade. Joel Brown et al. [14, 13, 15] built a system for microsurgical training. Goals of this system include a decrease in training time, objective evaluation of the training, and an alternative to using lab animals. Similarly Holbrey et al. [47] introduced a model based on FEM deformation for virtual suturing of vascular walls.

2.6 Major Components of Surgery Simulators

In this section we describe the major architectural components and the computational issues in some of the recent surgery simulators reported in the literature. The architecture of a general surgery simulation system is shown in Figure 2.2.

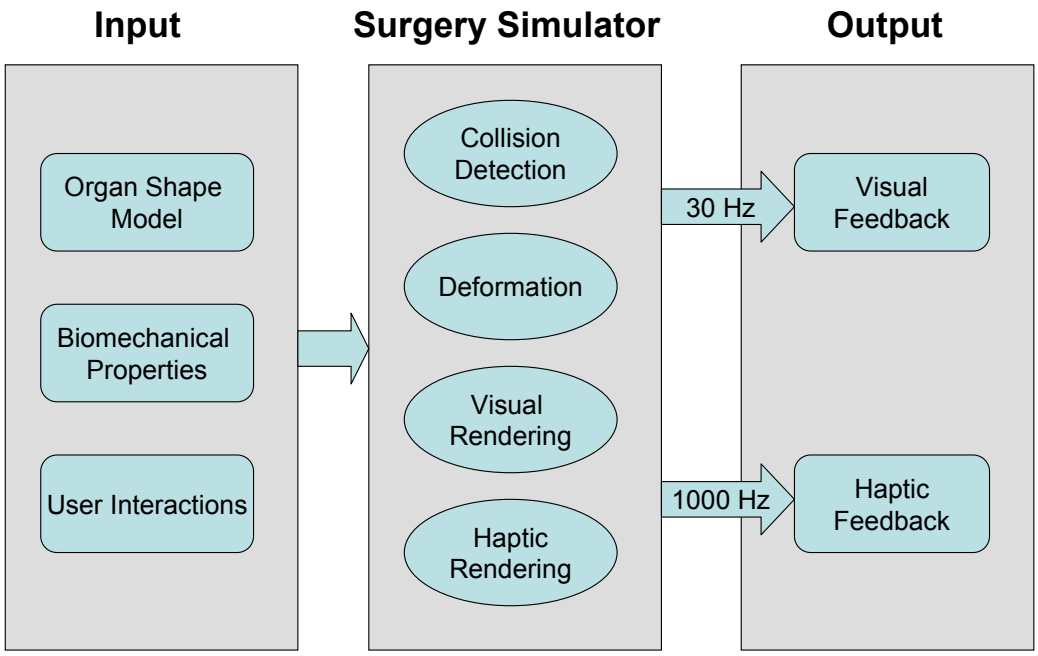


Figure 2.2: The architecture of a general surgery simulation system.

2.6.1 Cakmak et al. [17]

The architectural components and the procedure by Cakmak et al. [17] are outlined below.

1. Input: The process of forming the tissue models starts with the import of medical imaging data into the system. The medical imaging data can be from the CT or MRI scanners and the form of the data must comply with the established standardized image formats.

2. Interactive Organ Modeling: The built-in volume renderer in the KisMo software then enables the user to examine the dataset and pre-select anatomical structures required for the tissue modeling. Following the selection process is the slice-rendering mode. The outlines of the target structure are marked by the modeler with spline curves in a few medical image slices. Whenever there is a significant change of target shape during the training process, a new outline is constructed. Missing spline curves are interpolated by the software and a new 3D spline surface is generated. Image information in the tomography slice data is converted into a 3D object representation with deformation characteristics.

3. Interactive Tool Modeling: The instrument models are generated with CAD software and extended with an appropriate kinematic description.

4. Output Model Representation: The system then generates the final simulator model that contains the static and deformable organ model, the instrument models, the texture, the tissue connectivity information, the physiological organ description and the haptic interaction parameters. The simulation techniques presented allow the modeling of soft tissues based on a data-model which reflects the physical characteristics such as mass, stiffness and damping of real tissues. Virtual organ geometry is modeled as elastic polyhedron, NURBS or volume objects.

2.6.2 Delingette and Ayache [36]

The architectural components of the hepatic liver surgery system are outlined here.

1. The paper shows the design of a combination of image-processing techniques to extract the principal hepatic structures of interest from the clinical CT images taken before surgery (preoperative images).

2. Rib-Cage Modeling: Delingette and Ayache [36] show in-depth results explaining method for surgery simulation including a volumetric model built from medical images and an elastic modeling of the deformations. Ribs are added for more realistic representation of the operative field for more visual feedback.

3. Cutting: By restricting the regions where cutting is allowed, it is therefore possible to include in the simulator deformable models of a large size. In the hepatic surgery context, the authors decomposed their model into eight regions, corresponding to the detected segments, and restricted the cutting regions to a number of 3D bands at the interfaces between these segments. These bands are described by tensor-mass models, while the rest of the mesh is described by pre-computed models.

4. Interaction: The physical modeling of a simulator includes the modeling of contacts between virtual instruments and soft tissues as well as the bio-mechanical deformation of soft tissues. When a collision between an anatomical structure and a surgical instrument is detected, the boundary constraints on soft tissue models are updated and, depending on the nature of the instrument, a piece of tissue is removed. The soft tissue is then deformed according to a given bio-mechanical behavior.

2.6.3 Chung et al. [22]

For patient-specific bronchoscope simulation, the structure of the bronchial tree as well as the photo-realistic rendering of the bronchial lumen is essential. The method proposed by Chung et al. [22] has the following computational components.

1. Input: First the patient specific 3D CT scan data is used as an input to capture the geometry of the bronchial tree. The appearance of the inner wall is captured

as a video sequence from a real bronchoscope.

2. Processing: There are two major processing steps.

   (a) In the first stage, the video images are projected onto the surface geometry to obtain the appropriate textures.

   (b) Since the textures have illumination information, a BRDF procedure is used to eliminate the probe specific illumination information.

3. Output: A photo-realistic output for any novel view can be synthesized by mapping the appropriate texture with the addition of illumination information.

2.6.4 Raghupathi et al. [71]

The architectural components of the intestinal surgery simulator are outlined here.

1. Input: The input model consists of three different models:

   (a) *A geometric model used for rendering.* The geometry of the intestine is defined by creating a piecewise tubular surface of radius 2cm along its skeleton curve. The mesentery is defined as the surface generated by a set of nonintersecting line segments.

   (b) *A mechanical model for animation.* A mass-spring approach is applied to animate the intestine model.

   (c) *A collision detection model for evaluating the influence of the environment.* A stochastic approach which exploits temporal coherence is adopted for fast collision detection during animation.

2. Process: A new method is used for collision response which alters the displacements and velocities such that it instantaneously cancels the interpenetration

while keeping a resting contact between the two colliding bodies with no bouncing effects.

3. Rendering: A new method based on generalized cylinders is introduced for fast rendering of the intestine. In addition, hardware-based rendering using skinning is applied.

4. Output: The output is the real-time simulation of intestine behavior with user interactions.

2.6.5 Haptic Browser [69]

In haptic deformation, the sense of touch is added to accomplish realistic interaction with the organs or models. Figure 2.3 shows the architecture for a haptics simulation system, called Haptic Browser [69]. The concept is the same as the web browser (that can display a web page), but the *haptic browser* can display and enable interaction with *haptic enabled* organs. Figure 2.5 shows the haptic needle interacting with various organs. Figure 2.4 shows the same tool used to assemble/disassemble parts from a complex organ, the human ear.

2.7 Cinematic Quality Rendering

2.7.1 Realistic Rendering of an Organ Surface in Real-Time

In the scope of simulation for training professionals, the realism of the rendering is not a goal in and of itself. It can even be dangerous if it provides non-pertinent information to the trainee that he might use later in real situations. For instance, vessels may be used consciously or unconsciously to locate a site, while the vessel locations vary from one patient to another. The presence of realism addresses three purposes: quality of

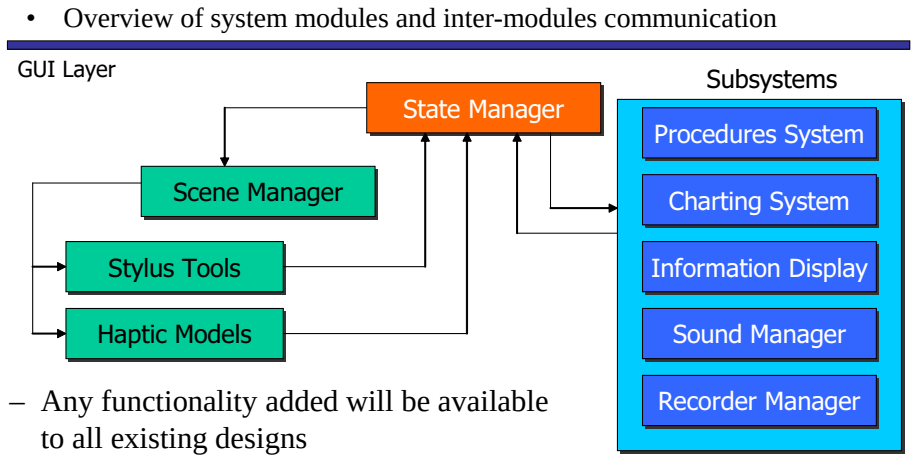


Figure 2.3: Architecture of the Haptic Browser.

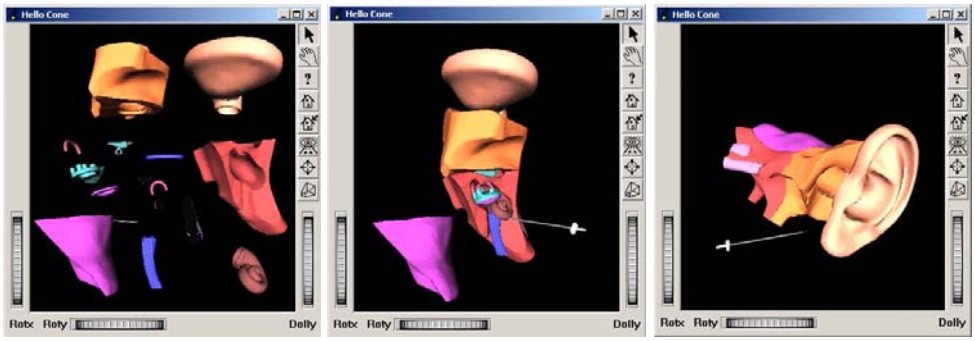


Figure 2.4: Visualization interface for ear assembly.

Figure 2.5: Visualization of various organs during haptic interaction.

immersion, carrying pertinent information, and providing 3D information that is present in the real situation [62]. Mayoral et al. [58] emphasized the importance and stress the integration of visual modalities with haptics for a total hip replacement planning system.

Neyret et al. [62] addressed aspects of the appearance of the organ surface: the organ skin texture, the specular highlights, and the reactions of the organ to the instruments and focused on the rendering of liver surface. The organ surface texture has three layers: skin map, effects map, and reflection map. For the reactions of the organ to the instruments, three types of effects are addressed: blood drops rolling on the surface, clear or deep cauterization, and whitening of the surface under local pressure. To meet the real-time constraint, advanced graphics features are used such as multi-pass rendering, OpenGL texture extensions and lookup-tables. The rendering effect of a liver is shown in Figure 2.6.

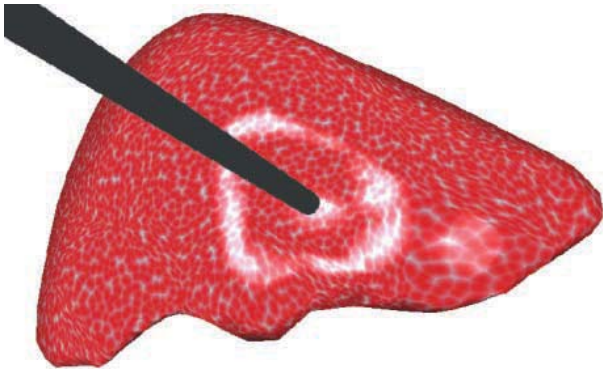


Figure 2.6: The rendering of a liver model [62].

2.7.2 Glistening Effect in Anatomical Objects

The smooth glistening appearance of the internal organs is partly due to the lining by the serous membrane (serosa). The complexity associated with modeling multi-layered thin films and the physical behavior of light by explicitly tracing the light rays

throughout a scene has led to the exploration of alternative techniques to simulate realistic reflections. Since the simulation of glistening effects by conventional ray casting requires considerable computational effort, it is difficult to simulate such effects at interactive rates. Prakash et al. [70] studied and implemented an environment mapping approach for texture mapping of anatomical objects with glistening using a cube environment map. The glistening effect can be rendered in real-time for surgery simulation, as shown in Figure 2.7.

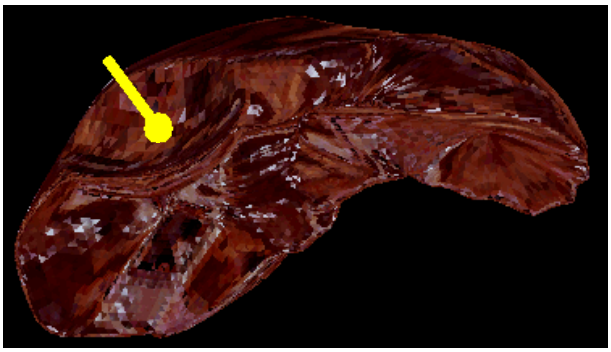


Figure 2.7: Glistening liver rendered using a cube map [70].

2.7.3 Automatic Medical Illustration Techniques

Medical illustration [82, 37] can help to improve the quality of rendering in a system for clinicians who teach and for medical students to provide training modules for medical education and surgery simulation for training surgeons. The need for such a system arises because:

1. Graphical rendering has its own limitations. Producing cinematic quality automatically is difficult and requires lots of manual intervention to create visual effects.

2. On the other hand, traditional medical illustrative styles can effectively be used in rendering medical data sets. The illustration approach mainly aims to enrich

the expressiveness of rendering by highlighting important features while reducing insignificant details, and rendering the result in a way that resembles an illustration.

To design appropriate illustrations, rendering parameters that define an illustrative approach are used to achieve high-level control of the visualization process. They can describe a wide variety of effects. That includes highlighting a selection to emphasize and focus the viewer's attention, removing layers of occluding material while exposing the important objects, and providing a chalk sketch of the outer shape of the body part to visualize only the contours, limiting visual confusion.

Volume rendering includes a high-level, more natural user interface and a domain-specific illustration specification framework [82]. In spatial focusing method, a simple ellipsoidal and rectangular focal area for general data sets is used. It allows focusing for segmented data sets which is useful for specification of the region of interest and for applying different rendering styles. A transfer function separates materials in the volume that corresponding to different value ranges in volume data values. The appropriate ranges in the transfer function domain are selected to define the material. Segmentation provides the process with an extra volumetric map, which identifies a material or part to which each voxel belongs.

The rendering of illustrations can be enhanced in three ways. First, volume rendering enhancements in the toolkit can be produced when field experts in medical visualization cooperate. Next, extensions to current illustration system should provide more flexibility for various applications. Finally, real-time interactive systems should be developed. This will enable dynamic adjustment of the region of interest and level of detail based on the user's level of expertise.

2.8 Summary

Surgery simulators can be classified into three generations [36], as shown in Figure 2.8.

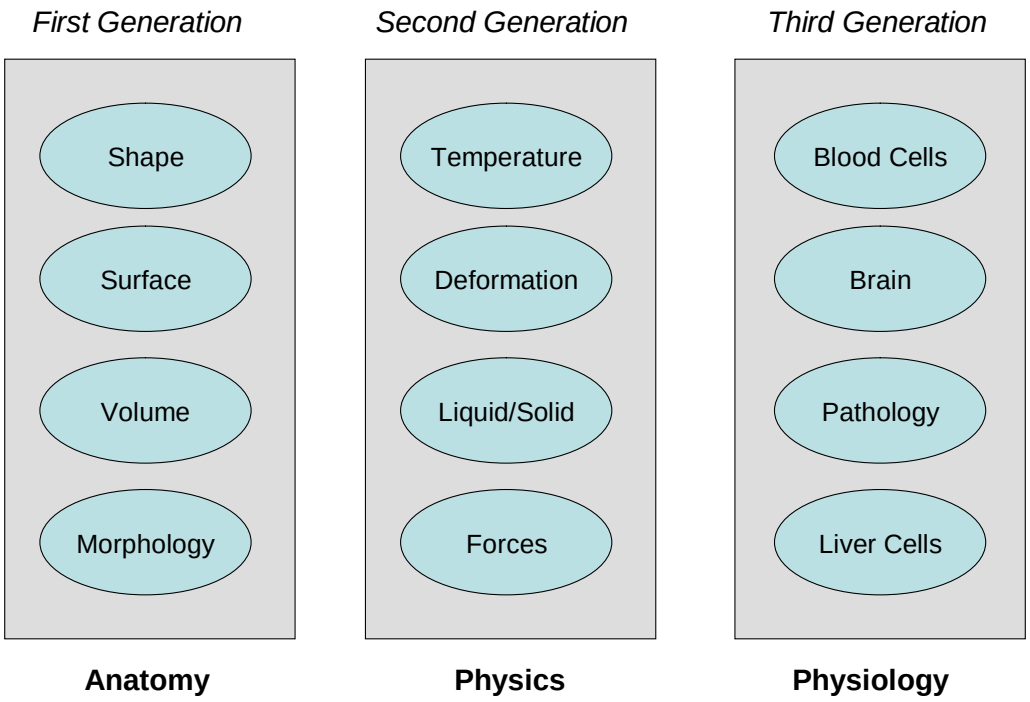


Figure 2.8: Different generations of surgery simulators [36].

First-generation simulators describe only the anatomy, in particular the geometry of the structures involved in a surgical intervention. In these simulators, the user can essentially navigate within a virtual representation of the patient, with a limited set of possible interactions. These simulators are mainly used as complementary diagnostic tools, and to assist with surgery planning, but are not well adapted to the simulation of surgical gestures. Examples of first generation simulators include bone surgery simulation [67, 66, 78, 79], virtual endoscopy [7, 87] and brochoscope simulation [35].

Second-generation simulators not only include the geometric modeling of the body

anatomy, but also the modeling of the physical properties of the soft tissues. The introduction of biomechanical properties is essential to allow realistic interactions between surgical instruments and soft tissues, including deformations and cutting.

Third-generation simulators combine anatomical, physical, and physiological modeling, that is, the modeling of the functions of some organic systems such as the cardiovascular, respiratory, or digestive systems. There is an additional degree of complexity due to the coupled nature of physiological and physical properties.

Although there are some efforts in the physiological modeling, for example, Santhanam et al. [73], the focus of the current research is on the second-generation simulators, especially the realistic deformations, cinematic quality rendering, real-time haptic interactions, and incorporation of surgical gestures and procedures. We have reviewed various research work in all of these aspects of surgery simulators in this chapter. In the next chapter we will present a physically based method for the deformation of soft tissues.

Chapter 3

Physically Based Object Deformation

3.1 Introduction

As computer graphics has been playing an important role in modeling and simulation, increasing levels of realism are demanded. In particular, the ability to model and transform deformable objects are becoming essential for a wide range of applications. For instance, deformable objects are used in computer animation, especially for realistic animation of human or animal characters and facial expressions. Also, surgical simulations and training systems require physically realistic deformable objects to model body tissues.

Approaches to model the deformation of objects can be categorized into non-physical methods and physically based methods. In non-physical methods, individual or groups of control points or shape parameters are manually adjusted for shape editing and design. Generally, these techniques are computationally efficient, and they rely on the skill of the designer rather than on physical principles. In contrast, physically based models are based on principles of mathematical physics and therefore are active: they respond in a natural way to applied forces, constraints, ambient media, and impen-

etrable obstacles. This category includes mass-spring models, finite element models, approximate continuum models, and low degree of freedom models. Other physically based models are [28], [6], and [43].

In all physically based approaches, finite element methods (FEM) are of particular interest, because they offer the greatest accuracy. However, the use of FEM in computer graphics has been limited because of the computational requirement. The primary limitation of FEM is that it requires a mesh. The automatic generation of good quality meshes is difficult. In addition, after the objects undergo large deformation or topological changes, a costly process of re-meshing is required. Therefore it is difficult to apply FEM in real-time systems.

These difficulties are circumvented when no mesh is needed. That is the reason for the increasing interests in meshless approaches in recent years. Based on different techniques of interpolation and integration, there are different types of meshless methods in the field of computational mechanics. However, most of them are only *pseudo meshless* since they need a background mesh for the numerical integration. As truly meshless techniques, the meshless local Petrov-Galerkin (MLPG) method [5, 3, 4] and local boundary integral equation (LBIE) method [3] seem to be the most promising. Based on MLPG, De et al. developed the method of finite spheres [29, 30] and applied it into the field of medical simulation [31, 32]. However, in their simulation, only a simple case with point interaction between the tool and the tissue was studied. In this chapter we experiment a meshless finite element method (MFEM) mainly based on MLPG by Atluri and Shen [4].

3.2 The Local Weak Form

In computer graphics, the object to be deformed is usually a domain Ω in R^3 or R^2 spaces. Consider a domain Ω of material of mass density $\rho(\mathbf{x})$ enclosed by a boundary Γ, in which linear elasticity is assumed. Conservation of linear momentum over the domain Ω results in the governing equilibrium equations

$$\nabla^2 u(\mathbf{x}) + b(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega \tag{3.2.1}$$

where ∇^2 is a second-order symmetric positive definite differential operator, $u(\mathbf{x})$ is the displacement of point $\mathbf{x}$, and b is the body force/unit volume (i.e. $b(\mathbf{x}) = \rho(\mathbf{x})\mathbf{g}$), and the boundary conditions are prescribed as:

$$u = \bar{u} \quad on \quad \Gamma_u, \tag{3.2.2}$$

$$\frac{\partial u}{\partial n} \equiv q = \bar{q} \quad on \quad \Gamma_q \tag{3.2.3}$$

where $\bar{u}$ and $\bar{q}$ are the prescribed displacement and normal flux, respectively, on the boundary Γ_u and Γ_q. The boundary $\Gamma = \Gamma_u \cup \Gamma_q$, and $\Gamma_u \cap \Gamma_q = \emptyset$.

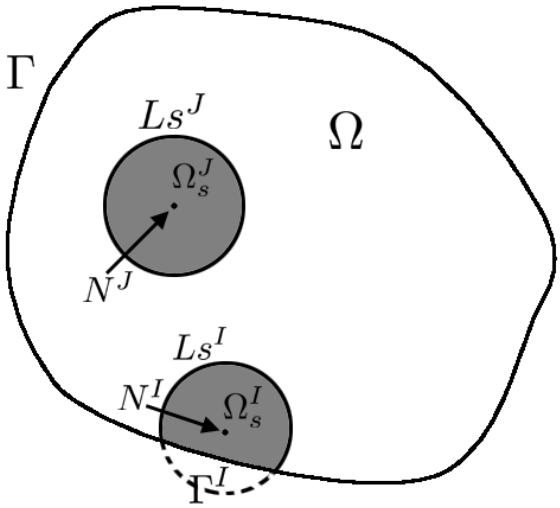


Figure 3.1: Global domain Ω, local domains Ω_s^I and Ω_s^J, and their boundaries.

A weak form is generated over the domain Ω_s, which is located entirely inside the global domain Ω, as illustrated in Figure 3.1. Note that Ω_s can be any shape. in our MFEM, 2D or 3D spheres, or the intersections of Ω and the spheres, are used as the local domain Ω_s.

The local weak form can be written as:

$$\int_{\Omega_s^I} (\nabla^2 u + b) v^I d\Omega = 0 \tag{3.2.4}$$

where Ω_s^I is the I^{th} local domain, u is the trial function, and v^I is the test function associated with this local domain. Using Green's Theorem, we obtain the following expression:

$$\int_{\partial\Omega_s^I} u_i' n_i v^I d\Gamma - \int_{\Omega_s^I} (u_i'(v^I)_i' - bv^I) d\Omega = 0, \quad i = 1, ..., d \tag{3.2.5}$$

where $\partial\Omega_s^I$ is the boundary of local domain Ω_s^I, u_i' and $(v^I)_i'$ are partial derivatives of trial function and test function respectively, n_i is the i^{th} component of the outward unit normal to the local boundary, and d is the dimensionality of the problem. Notice that

$$u_i' n_i = \frac{\partial u}{\partial n} \equiv q. \tag{3.2.6}$$

For local domains that are completely inside the global domain,

$$\partial\Omega_s^I = Ls^I \tag{3.2.7}$$

For local domains that intersect the boundary of the domain Γ,

$$\partial\Omega_s^I = Ls^I + \Gamma^I \tag{3.2.8}$$

where Γ^I is the intersection of the boundary of local domain $\partial\Omega_s^I$ and the boundary of global domain Γ, i.e. $\Gamma^I = \partial\Omega_s^I \bigcap \Gamma$.

Substitute Equation (3.2.6), Equation (3.2.7) and Equation (3.2.8) into Equation (3.2.5), we obtain:

$$\int_{\Omega_s^I} u_i'(v^I)_i' d\Omega - \int_{L_s^I} u_i' n_i v^I d\Gamma = \int_{\Omega_s^I} b v^I d\Omega + \int_{\Gamma^I} q v^I d\Gamma \tag{3.2.9}$$

The left side of Equation (3.2.9) is the interior force or elastic force. And the right side is external force, in which the first term is gravity force, and the second one is the external force imposed on the global boundary Γ. For local domains that are completely inside the global domain, the second term on the right side is zero.

We distribute a number of nodes $N^1, N^2, ..., N^N$ in the entire domain, with a local domain Ω_s^I associated with each node N^I. These local domains overlap with each other and the collection of all the local domains covers the global domain Ω, i.e.

$$\bigcup_{I=1}^{N} \Omega_s^I = \Omega$$

Over each local domain Ω_s^I, a local weak form as Equation (3.2.9) can be constructed.

3.3 Interpolation Scheme and Test Functions

The trial function u in Equation (3.2.9) is interpolated as:

$$u(\mathbf{x}) = \sum_{J=1}^{N} \phi^J(\mathbf{x}) \hat{u}^J \tag{3.3.1}$$

where ϕ^J is the shape function associated with node N^J, and $\hat{u}^J$ is the virtual displacement at this node. When all the virtual nodal displacements $\hat{u}^J$ are known, displacement at any point $\mathbf{x}$ can be approximated using Equation (3.3.1).

For a local domain Ω_s^I, the local approximation $u(\mathbf{x})$ is decided by the shape functions of M nodes $\{N^J\}$ $(J = 1, 2, ..., M)$, whose supported domains $\{T_{sh}^J\}$ intersect with Ω_s^I, as illustrated in Figure 3.2. To achieve local control of the approximation, the shape function should be compactly supported, i.e. ϕ^J is non-zero only in small domains T_{sh}^J.

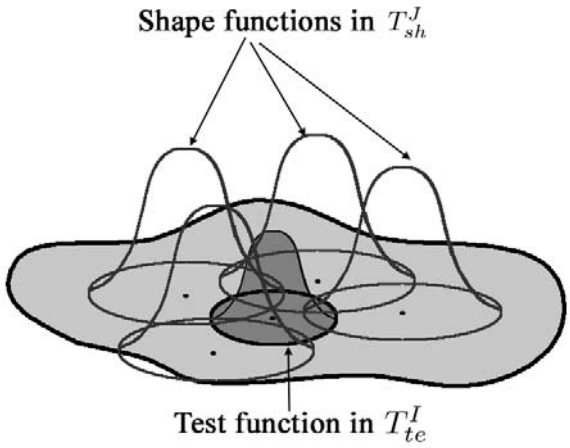


Figure 3.2: Intersections of shape function in T_{sh}^J $(J = 1, 2, ..., M)$ with the test function in T_{te}^I. T_{sh}^J and T_{te}^I are both sphere shapes, but with different radii.

Likewise, the test function $v^I(\mathbf{x})$ should also be compactly supported, but the supported domain of the test function T_{te}^I can be different from T_{sh}^I. In MFEM, we choose spheres centered at node N^I as T_{sh}^I and T_{te}^I. T_{te}^I is same size as Ω_s^I and T_{sh}^I is proportionally bigger.

The nodal shape and test functions can also be different, as far as they satisfy some minimal requirements. Discussion and comparison of different types of shape and test functions can be found in [4]. In our method, the partition of unity (PU) paradigm is used for shape function and the Heaviside step function is used as test function.

Substitute Equation (3.3.1) into Equation (3.2.9), we can obtain the local weak form over the local domain of I^{th} node in the form:

$$\sum_{J=1}^{N} \Big(\int_{\Omega_s^I} (\phi^J)'_i (v^I)'_i d\Omega - \int_{L_s^I} (\phi^J)'_i n_i v^I d\Gamma \Big) \hat{u}^J = \int_{\Omega_s^I} b v^I d\Omega + \int_{\Gamma^I} q v^I d\Gamma \tag{3.3.2}$$

Assemble N local weak form equations together, we obtain the system equation:

$$\mathbf{K}\hat{\mathbf{U}} = \mathbf{F}_b + \mathbf{F}_q \tag{3.3.3}$$

where $\mathbf{K}$ is the stiffness matrix, which is a $dN \times dN$ banded unsymmetrical matrix, $\hat{\mathbf{U}}$ is virtual nodal displacement vector, $\mathbf{F}_b$ is the body force vector and $\mathbf{F}_q$ is the boundary force vector.

3.4 Imposing Boundary Conditions

There are two types of boundary conditions: essential boundary conditions (Equation (3.2.2)) and natural boundary conditions (Equation (3.2.3)). The natural boundary conditions can be directly applied to the right side of Equation (3.3.3):

$$\mathbf{F}_q^I = \int_{\partial\Gamma^I} \bar{q} v^I d\Gamma. \tag{3.4.1}$$

In MFEM, the shape functions do not satisfy the Kronecker delta property at the nodes. Therefore, the essential boundary conditions can not be imposed directly. Various techniques of imposing essential boundary conditions have been employed in meshless methods, including collocation techniques [5], Lagrange multipliers, penalty formulations, use of finite elements along essential boundaries, and modified variational principles [29]. Here we introduce a simple procedure of imposing essential boundary conditions when only point forces are exerted on the boundary. In this method, the point forces can be exerted at any point on the global boundary, without requiring them to be applied at nodal points.

As illustrated in Figure 3.3, a point force is exerted on the point $\mathbf{x}_p \in \Gamma$. Assume point $\mathbf{x}_p$ is in the intersection of the boundary of two nodes N^C and N^D, i.e. $\mathbf{x}_p \in \Gamma^C \cap \Gamma^D$. In this case, the system equation Equation (3.3.3) is written as:

$$\mathbf{K}\hat{\mathbf{U}} = \mathbf{F}_b + \mathbf{F}_p \tag{3.4.2}$$

where F_p is point force vector, whose elements are all zero except C^{th} element and D^{th} element, which are unknown.

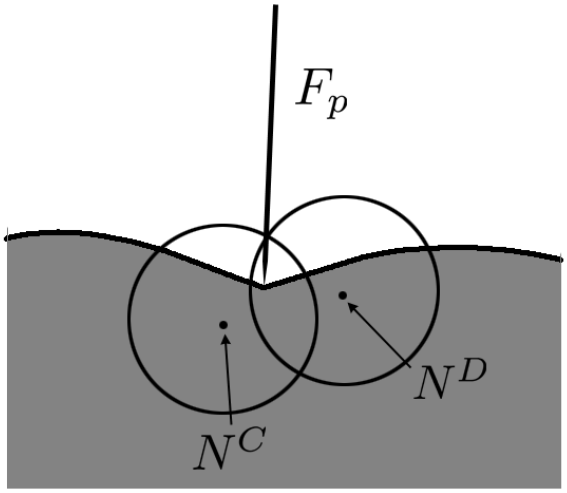


Figure 3.3: Point force exerted on the intersection of node N^C and node N^D.

The displacement at this point is known as $U(\mathbf{x}_p)$. It can be written as:

$$\sum_{J=1}^{N} \phi(\mathbf{x}_p)^J \hat{u}^J = U(\mathbf{x}_p). \tag{3.4.3}$$

The C^{th} and D^{th} equation of the system are:

$$\sum_{J=1}^{N} K_{CJ}\hat{u}^J = F_C + F_p \tag{3.4.4}$$

$$and \quad \sum_{J=1}^{N} K_{DJ}\hat{u}^J = F_D + F_p \tag{3.4.5}$$

respectively. Subtract Equation (3.4.4) from Equation (3.4.5), we obtain:

$$\sum_{J=1}^{N} (K_{DJ} - K_{CJ})\hat{u}^J = F_D - F_C \tag{3.4.6}$$

Substitute the C^{th} equation and D^{th} equation of the system with Equation (3.4.3) and Equation (3.4.6) respectively, the system equation Equation (3.4.2) becomes

$$\mathbf{K}'\hat{\mathbf{U}} = \mathbf{F}'_b \tag{3.4.7}$$

where $\mathbf{F}'_b$ is the body force vector with C^{th} element and D^{th} element substituted by the right side of Equation (3.4.3) and Equation (3.4.6), respectively. The bandwidth of stiffness matrix is normally unaffected. Elements of $\mathbf{K}'$ and $\mathbf{F}'_b$ are all known, and therefore the virtual nodal displacement vector $\hat{\mathbf{U}}$ can be solved.

This simple method of imposing essential boundary conditions can be extended to the cases that point forces are exerted on any number of points, with each point lies on the intersection of the boundaries of any number of nodes. In other words, it can be used in cases where only point forces are applied.

3.5 Addition/Deletion of Nodes

When the object undergoes large deformation or topological changes, the numerical approximation can be adjusted by addition or deletion of a few nodes.

3.5.1 Large Deformation

When large deformation occurs, the density of nodes in the objects is detected. In the region where nodes density has increased, some of the nodes can be dropped. In the region where nodes density has decreased, a few nodes should be added. To add a node, we only need to insert d rows and d columns of elements into stiffness matrix $\mathbf{K}$, and insert d elements into body force vector $\mathbf{F}_b$ (d denotes the dimensionality of the problem). Likewise, to drop a node, we only need to delete d rows and d columns of elements from stiffness matrix $\mathbf{K}$, and delete d elements from body force vector $\mathbf{F}_b$. The rest elements in $\mathbf{K}$ and $\mathbf{F}_b$ do not need to be integrated again, and therefore the computational cost is generally acceptable.

3.5.2 Topological Change

In some applications, the topology of the objects needs to be changed. For instance, in surgical simulation, the tissue usually undergoes cutting and stitching. Here we only consider the splitting of deformable models.

When the force applied on the boundary exceeds a critical value, the model is cut. There are two phases in the cutting procedure. In the first phase, the model is cut but still stick together as one whole object. In the second phase, the model splits into two objects.

During the first phase, the nodes whose local domains intersect the cutting are examined and allocated to one side of the cutting according to their relative position. Their local domains are re-evaluated and their corresponding elements in the stiffness matrix are re-integrated. A few nodes need to be added near the cutting to ensure the collection of the local domains covers the entire global domain. In the second phase, the stiffness matrix are split into two matrices. No additional numerical evaluation is needed.

3.6 Application in Surgery Simulation

We designed the application of MFEM in surgery simulation, in which human tissues are modeled as deformable models. Complex interaction between tissues and tools are considered, including touching and clamping. The procedure is described as below.

3.6.1 Precomputation

Given a tissue model, a number of nodes are distributed inside the entire domain, with the collection of their local domains covering the global domain. The stiffness matrix

$\mathbf{K}$ and body force vector $\mathbf{F}_b$ are precomputed.

3.6.2 Runtime Computations

When the interaction between the tissue and a tool is detected, boundary conditions are applied. After each time step, the system equation are solved and the tissue deforms. The density of the nodes is controlled by adding or dropping nodes when large deformation occurs, and the sizes of $\mathbf{K}$ and $\mathbf{F}_b$ change accordingly.

If the tool is a cutting tool, the tissue is cut after the reacting force reaches some threshold value. After the tissue is cut into two parts, the system breaks into two, with deformation computed separately.

The flow chart for the deformation procedure with MFEM is shown in Figure 3.4.

3.6.3 Results

We have tested MFEM on a simple $0.1 \times 0.1 \times 0.1\ m^3$ cube model deformed by one point force. 27 nodes ($3 \times 3 \times 3$) are distributed inside the cube. The cube is assumed to be of isotropic material, which is an approximation of human tissues. An intermediate, convenient value is chosen for Young's modulus, E. While Poisson's ratio, ν, is set to approximate a volume preserving material. The density, ρ, is set to be the density of muscle which was obtained through experiment. These physical constants for the experiment are obtained from [27] and are listed below.

$E = 100Pa \quad \nu = 0.49 \quad \rho = 1040\ kg/m^3$

Figure 3.5 shows the results of the deformation of the cube model when it is deformed by a point force. The deformation can be calculated in real-time.

Our ultimate goal is to simulate a simple surgery with MFEM, with operations including touching and clamping, to compare with the result from the commercial

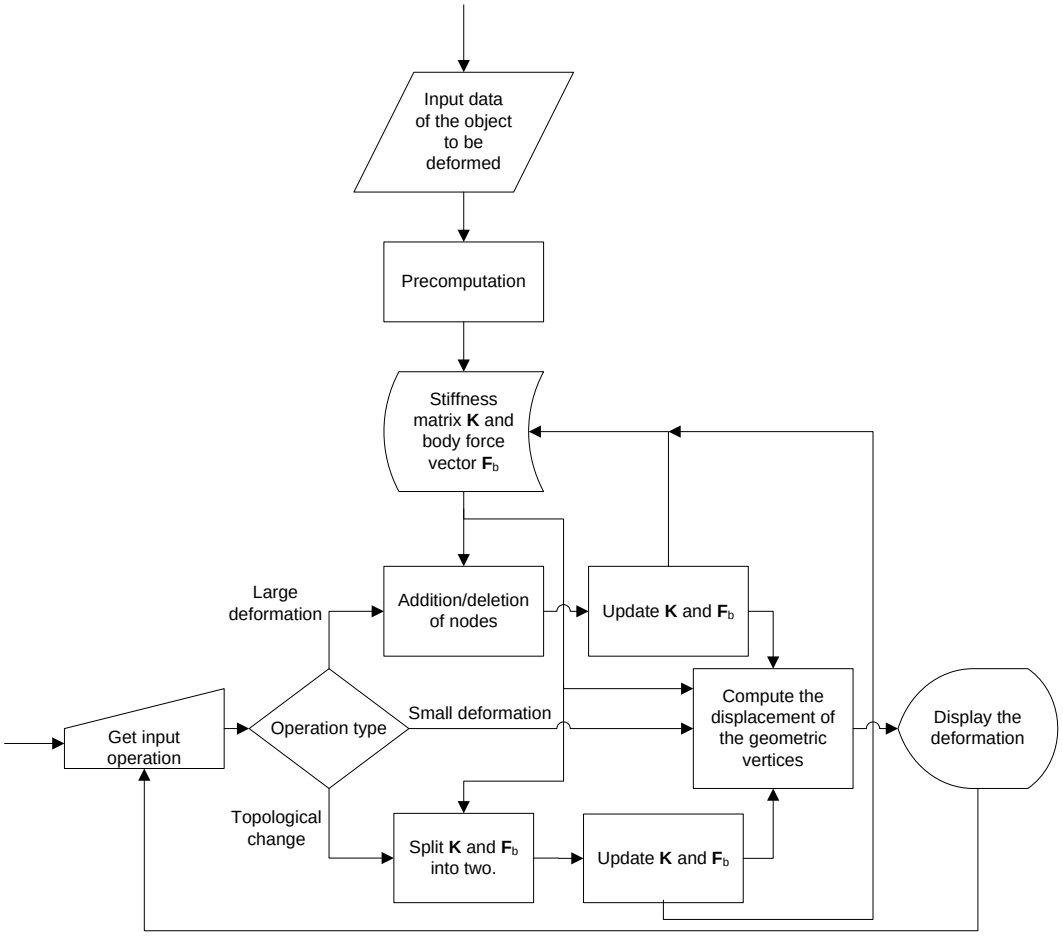


Figure 3.4: The flowchart for the deformation procedure with MFEM.

LapSim system [1] illustrated in Figure 3.6.

3.7 Summary

In this chapter, we have introduced the formulation of an accurate meshless finite element method for deformable models in computer graphics. In addition, a simple method to impose exact essential boundary conditions is suggested, and an application of MFEM to surgical simulation is proposed. In MFEM, no mesh is needed, nodes can be easily added and dropped, and therefore large deformation and topological change

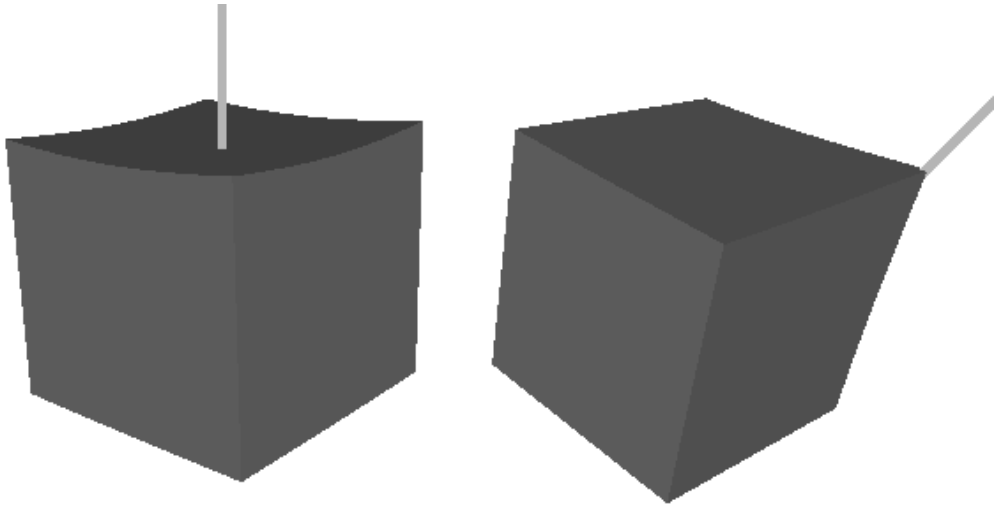


Figure 3.5: A simple example of the deformation of a cube under point forces. 27 nodes are distributed inside the cube model.

Figure 3.6: Proposed effect of the simulator using MFEM ([1]).

can be achieved. This method overcomes the main problems of FEM, and hence is a promising deformable modeling method in computer graphics. De et al. [34, 33] has successfully applied a similar meshless method in interactive surgery simulations with haptic feedback for very simple meshes, without much emphasis on realistic rendering.

In the next chapter we will discuss about the interactions between objects during surgery simulation with FE analysis. In Chapter 5 we will introduce a parameterized representation of static shape with a deformation technique, which is an alternative approach that leads to a simpler implementation.

Chapter 4

Physically Based Object-Object Interactions

4.1 Introduction

In surgery simulation with finite element (FE) analysis, boundary conditions are specified on the surface of the FE model, so that the displacement of the nodes can be calculated, and the shape of the tissue after deformation can then be obtained. In simulation, the boundary conditions change with the interactions between the tissue and other objects. Therefore, we must study the interactions of the tissue with all the other objects that are in contact with it.

Another criteria for surgery simulation is the interactivity. Real-time simulation must be achieved. Therefore, the calculation of the deformation must be fast enough. Researchers have explored some techniques to speed up FM analysis, so that it can be used in real-time simulation, such as the point collocation-based method of finite spheres (PCMFS) introduced by Basdogan et al. [8] and Lim and De. [53], and the precalculation and synthesis method developed by Mahvash and Hayward [56]. Since the boundary conditions must be updated after every time step, the process of updating boundary conditions must also be fast enough to ensure real-time simulation.

In the work of Basdogan et al. [8] and Lim and De. [53], the tissue deformation was computed locally, and only the interactions between tool tip and the tissue were studied. Berkley et al. [10] used FE modeling for virtual suturing. In their application, the tissue is attached on the skeleton. Those nodes in contact with the bone are assumed to be fixed. The interactions between the tissue and a virtual needle is studied. In addition to the normal tool-tissue interaction, Mahvash and Hayward [56] also discussed the sliding between the tool and tissue. Hirota et al. [46] introduced a penalty method for FE simulation based on the concept of *material depth* to solve the contact between soft tissues.

In addition to the contact model for surgery simulation, there are some general methods for the problem of contact. For example, acceleration, velocity and position correction are used by Volkov [88] to deal with the collision between cloth in cloth simulation. However, this method is limited for dynamical analysis.

In this chapter, we classify the interactions between objects in surgery simulation into three types. The method to update the boundary conditions for each type of interactions is also presented. Our algorithm covers almost all the possible interactions between objects in a simple simulation procedure. In addition, the boundary conditions are imposed in a point-collocation basis, and the update of boundary conditions is therefore fast.

4.2 FE Analysis in Surgery Simulation

In FE analysis, a tissue model is discretized into finite number of elements connected by nodes, or an FE model. Partial differential equations are solved in the FE model. After integration and assembling of the discretized set of equations, the system matrix

$\mathbf{K}$ is calculated, and the system equation for static analysis can be written as:

$$\mathbf{KU} = \mathbf{F}^b + \mathbf{F}^s, \tag{4.2.1}$$

in which $\mathbf{U}$ is the vector of nodal displacement, $\mathbf{F}^b$ is the vector of body force, or gravity force in most cases, and $\mathbf{F}^s$ is the vector of external forces applied on the surface of the FE model.

Our discussion is restrained in the content of static analysis, because surgery normally requires slow precise concentrated movements, and the dynamic contributions can be generally neglected [10].

For every point on the surface of the FE model, there is a boundary condition specified at it. The boundary condition can either be an essential boundary condition, which is specified as displacement, or a natural boundary condition, which is specified as stress. In normal FE analysis, imposing natural boundary condition requires integration over the model surface. To speed up the process, we impose the boundary conditions in a point-collocation basis, which means the boundary conditions are directly specified at nodal points:

$$\mathbf{u}(\mathbf{x}_I) = \mathbf{u}^s \qquad on\ \Gamma_u \tag{4.2.2}$$

and

$$\mathbf{f}^s(\mathbf{x}_I) = \mathbf{f}^s \qquad on\ \Gamma_f. \tag{4.2.3}$$

The vector of nodal displacement can be partitioned as $\mathbf{U} = [\mathbf{U}_n\ \mathbf{U}_u]^T$, in which $\mathbf{U}_n$ corresponds to those boundary nodes whose displacement is known, and $\mathbf{U}_u$ corresponds to the other nodes whose displacement is unknown. The corresponding system equation of this partition can be written as

$$\begin{bmatrix} \mathbf{K}_{nn} & \mathbf{K}_{nu} \\ \mathbf{K}_{un} & \mathbf{K}_{uu} \end{bmatrix} \begin{bmatrix} \mathbf{U}_n \\ \mathbf{U}_u \end{bmatrix} = \begin{bmatrix} \mathbf{F}_n^b \\ \mathbf{F}_u^b \end{bmatrix} + \begin{bmatrix} \mathbf{F}_n^s \\ \mathbf{F}_u^s \end{bmatrix}. \tag{4.2.4}$$

In simulation, the boundary conditions are imposed by specifying the value of the elements in vector $\mathbf{U}_n$ and $\mathbf{F}_u^s$.

4.3 Method

In our method, the object-object interactions in surgery simulation are classified into three types: tissue-tool interactions, tissue-support interactions, and tissue-tissue interactions. To study the interactions between objects, we need to know when and where the contact happens. The detection of collision between objects is a difficult problem in computer graphics. There exist a wide range of methodologies for collision detection, such as I-COLLIDE algorithm proposed by Cohen et al. [23], OBBTree algorithm proposed by Gottschalk et al. [41], and AABB tree algorithm proposed by Bergen [85]. Lombardo et al. [54] introduced a real-time collision detection method relying one the graphics hardware for testing the interpenetration between a rigid tool and a deformable organ in surgery simulation. Each of these collision detection methods possesses its own advantages and drawbacks, depending on the intended applications. In our procedure, we assume that the problem of collision detection is solved, and that after each time step, we know all the points on the tissue surface that collide with other objects.

4.3.1 Tissue-Tool Interactions

Tissue-tool interactions are the most important type of interactions in surgery simulation. In Lim and De [53], the contact between a tissue and a tool is considered as point contact. To avoid artificial defect, the contact should be allowed anywhere on the surface. Berkley et al. [10] introduced a constraint approach to accommodate general contact anywhere on the surface of the model. In the current discussion, we assume that the contact always occurs at the nodal point.

In simulation, the initial configuration of the tissue model is stored. Once the tool tip touches the surface of the tissue, unless slide occurs, the contact point P on the tissue surface is considered to be *attached* on the tool tip, and the displacement of this point is specified as the displacement of the tool tip, as illustrated in Figure 4.1.

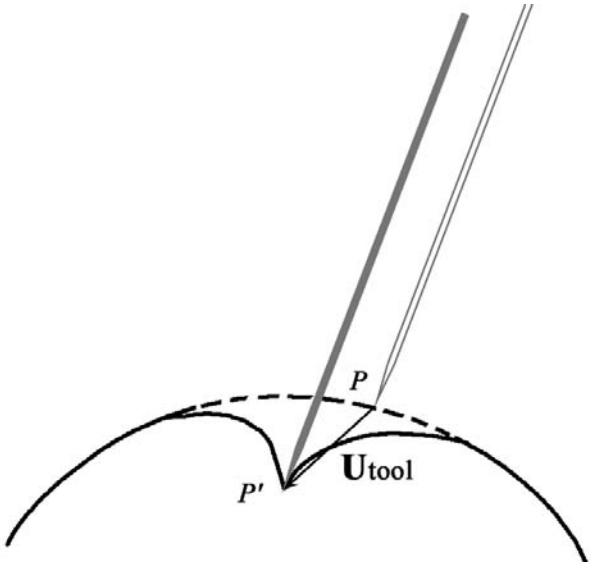


Figure 4.1: The displacement of the contact point.

If other types of interactions are not considered, the essential boundary condition is $\mathbf{u}(\mathbf{x}) = \mathbf{U}_{tool}$ at point P, and the natural boundary condition is $\mathbf{f}^s(\mathbf{x}) = 0$ for all the surface points except P. That means in Equation (4.2.4),

$$\mathbf{U}_n = \mathbf{U}_{tool} \tag{4.3.1}$$

and

$$\mathbf{F}_u^s = 0. \tag{4.3.2}$$

Substitute Equation (4.3.1) and Equation (4.3.2) into Equation (4.2.4), the vector of unknown displacement can be obtained from

$$\mathbf{U}_u = \mathbf{K}_{uu}^{-1}(\mathbf{F}_u^b - \mathbf{K}_{un}\mathbf{U}_{tool}). \tag{4.3.3}$$

The force vector delivered back to the tool tip is

$$\mathbf{F}_{feedback} = -\mathbf{F}_n^s = \mathbf{F}_n^b - \mathbf{K}_{nn}\mathbf{U}_{tool} - \mathbf{K}_{nu}\mathbf{U}_u \tag{4.3.4}$$

After each time step, we only need to track the position of the tool tip P' and update the boundary condition $\mathbf{U}_{tool}$. The haptic feedback and the deformation of the tissue can be calculated using Equation (4.3.4) and Equation (4.3.3).

If the tool is a clamp, once it clamps the tissue, the motion of the contact point always follows the tool tip, until the clamping action stops. If the clamp does not clamp the tissue, or the tool is not a clamp, sliding between the tool and the tissue might happen.

A local coordinate system is defined at the contact point P in the initial configuration of the tissue model, so that the feedback force $\mathbf{F}_{feedback}$ can be decomposed into a normal component $\mathbf{F}_n$ and a tangential component $\mathbf{F}_t$, as shown in Figure 4.2. The transform of $\mathbf{F}_{feedback}$ in the world coordinate system to the local coordinate system can be achieved by multiplying a rotation matrix $\mathbf{R}_P$:

$$\begin{bmatrix} \mathbf{F}_{x'} \\ \mathbf{F}_{y'} \\ \mathbf{F}_{z'} \end{bmatrix} = \mathbf{R}_P \begin{bmatrix} \mathbf{F}_x \\ \mathbf{F}_y \\ \mathbf{F}_z \end{bmatrix}. \tag{4.3.5}$$

We define a coefficient of friction μ for the tissue surface. As we know, when $\mathbf{F}_t > \mu\mathbf{F}_n$, the tool tip will slide on the surface of the tissue, and the contact point is no longer attached to the tool tip. To decide the tissue deformation after slide, we use two steps.

In the first step we assume that there is no sliding. The feedback force $\mathbf{F}_{feedback}$ is calculated using Equation (4.3.4). It is then decomposed into $\mathbf{F}_t$ and $\mathbf{F}_n$. If $\mathbf{F}_t \leq \mu\mathbf{F}_n$, it means there is no sliding. The tissue deformation is calculated directly and the second step is skipped. The second step is conducted only if $\mathbf{F}_t > \mu\mathbf{F}_n$, which means sliding

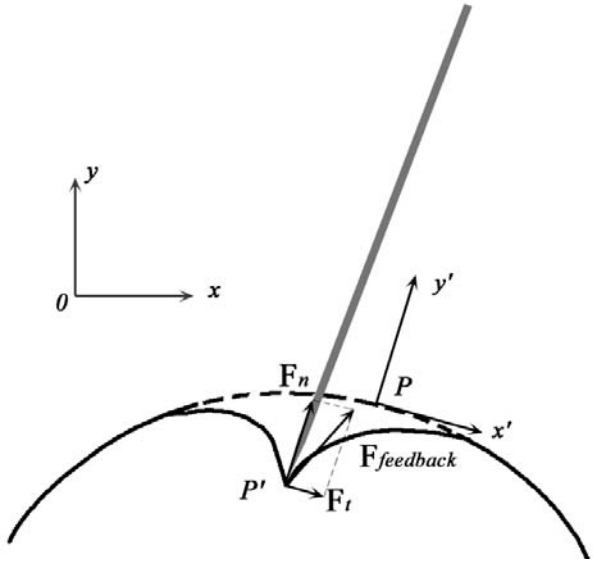


Figure 4.2: Decomposing the feed back force into a tangential component $\mathbf{F}_t$ and a normal component $\mathbf{F}_n$.

has happened. After sliding, the tool tip is in contact with a new point Q on the tissue surface, which corresponds to a new node in the FE model.

In the second step, the essential boundary condition $\mathbf{U}_n = \mathbf{U}_{tool}$ is specified at point Q, Equation (4.3.3) is used again to calculate the real deformation. Now the problem is how to determine the position of point Q in the initial configuration of the tissue model.

As illustrated in Figure 4.3, we search the position of Q on the tissue surface in its initial configuration. The displacement of the tool tip during this time step is transformed into the local coordinate system of point P, and the tangential component $\mathbf{U}_{tangential}$ is calculated. The corresponding point $\mathbf{Q}''$ on the tangential plane is found. From this point, a ray is cast which is normal to the tangential plane. The intersection point of the ray and the tissue surface is the new contact point Q. For this new contact point, a new local coordinate system is defined for the sake of next time step.

From Figure 4.3 we can see that the surface distance from P to Q is usually different

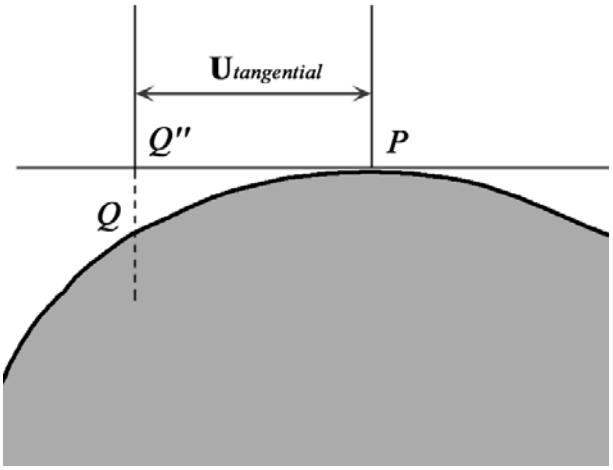


Figure 4.3: Determination of the new contact point Q.

from $\mathbf{U}_{tangential}$. However, the update rate of the simulation can be assumed to be fast enough, so that the displacement of the tool tip in one time step is relatively small. Therefore, this difference can be neglected.

When there is no collision between the tool and the original shape of the tissue, motion of the contact point no longer follows the tool tip, and the tissue regains its initial configuration.

4.3.2 Tissue-Support Interactions

During simulation, the tissue rests on other objects, which are called support. The support can be bones, muscles, and other tissues. Our primary interest is the tissue deformation during surgery. The surgery tool does not interact with the support directly, and the tool force has little influence on the shape of the support. Therefore, the support can be considered to be rigid.

The shape of the support surface is normally not a plane. Instead, the shapes of the support and the tissue are complementary. As shown In Figure 4.4, the tissue is in close contact with the support.

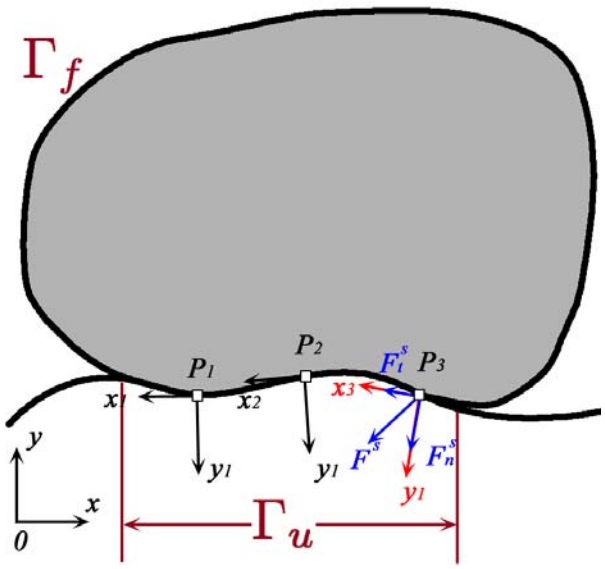


Figure 4.4: The interactions between tissue and support. The supporting force $\mathbf{F}^s$ at each node is decomposed into a normal component $\mathbf{F}_n^s$ and a tangential component $\mathbf{F}_t^s$.

Same as tissue-tool interactions, we study the interaction between tissue and support in two steps after each time step. The first step is a testing step, in which the essential boundary condition is set as $\mathbf{u}(\mathbf{x}) = 0$ on the contact surface Γ_u, and the natural boundary condition is set as $\mathbf{F}^s(\mathbf{x}) = 0$ on the non-contact surface Γ_f. After imposing the boundary conditions and solving Equation (4.2.4), the supporting force at each node on Γ_u and the position of each node on Γ_f can be obtained.

First we study the supporting forces at the nodes on Γ_u. For each such node, a local coordinate system is defined, same as the one defined in Section 4.3.1. The supporting force vector $\mathbf{F}^s$ at each node is decomposed into a normal component $\mathbf{F}_n^s$ and a tangential component $\mathbf{F}_t^s$, as shown in Figure 4.4. If $\mathbf{F}_n^s > 0$, it means the support does not provide supporting force to the tissue at this node. Instead, the support *drags* this node so that it is fixed on the support surface. For this case, in the second step, the boundary condition at this nodal point is set as $\mathbf{F}^s(x) = 0$. If $\mathbf{F}_n^s$ at the node is negative, the boundary condition does not change at this nodal point in the second

step.

Next we study the positions of the nodes on Γ_f. It is possible that these nodes might penetrate into the support after the first step. For such nodes, we apply a penalty force $\mathbf{F}_{pen}$ at each nodal point, as illustrated in Figure 4.5. The method to decide the magnitude and the direction of the force will be discussed in Section 4.3.4.

In the second step, the new boundary conditions are imposed, and the real deformation is calculated.

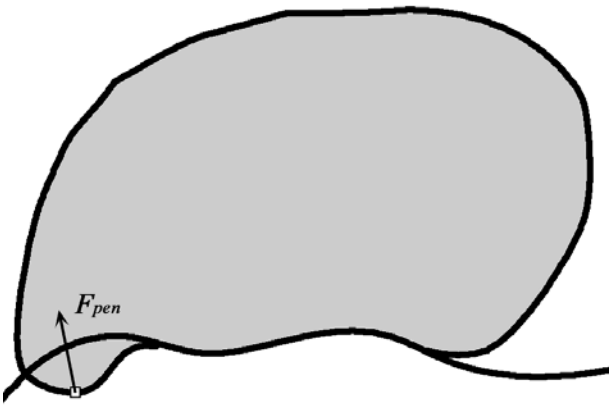


Figure 4.5: A penalty force $\mathbf{F}_{pen}$ is applied to the node that has penetrated into the support.

4.3.3 Tissue-Tissue Interactions

The contact between soft tissues is a difficult problem. The contact forces deform each other. As a result of the deformation, new parts of boundary surfaces may come into contact, or some parts of boundaries previously in contact may be separated. Therefore, it's difficult to impose boundary conditions for each tissue. In our work, we rely on a penalty method which is based on the work of Hirota et al. [46].

Here we only study the interactions between two tissues, which are called tissue A

and tissue B respectively for convenience. The method can be easily generalized into the situations with more number of tissues. Same as with other types of interactions, a testing step is performed before the calculation of real deformation. In the testing step, no contact is assumed for both tissues. After the displacements of the surface nodes are calculated, a process of detection of collision is performed. If the two tissues collide with each other, two more steps are carried out to compute the real deformation of each tissue. In the first step, we study one of the tissues, say, tissue A. For every nodes on the surface of tissue A which has penetrated into tissue B, a penalty force is applied, as illustrated in Figure 4.6. After updating these boundary conditions, the deformation of tissue A is calculated again. In the second step, the collision between tissue A and B is checked again. If there is still penetration, a penalty force is applied on the surface nodes of tissue B that has penetrated into tissue A, just as the previous step. The deformation of tissue B is then calculated.

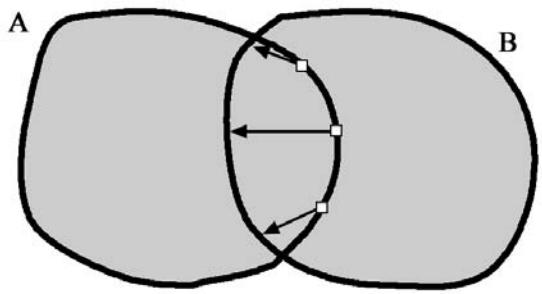


Figure 4.6: Penalty force are applied on the surface nodes of tissue A that has penetrated into tissue B.

4.3.4 Calculation of the Penalty Force

The magnitude and the direction of the penalty forces mentioned in Section 4.3.2 and Section 4.3.3 are determined based on a notion called *material depth*. The material

depth $\mathbf{g}(\mathbf{x})$ measures the distance from a point $\mathbf{x}$ which is inside an object, to the boundary of the object, as shown in Figure 4.7. $\mathbf{g}(\mathbf{x})$ is a vector whose direction points to the projection of point $\mathbf{x}$ on the boundary. If the object is deformable, the material depth is the distance from the point to its projection on the boundary in the initial undeformed configuration.

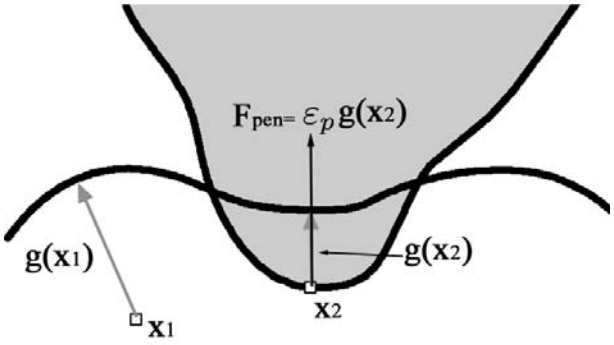


Figure 4.7: The material depth and the penalty force.

The penalty force applied on point $\mathbf{x}$ is defined as.

$$\mathbf{F}_{pen}(\mathbf{x}) = \varepsilon_p \mathbf{g}(\mathbf{x}), \tag{4.3.6}$$

where ε_p is the coefficient of penalty force.

We do not evaluate the material depth for every point. Only the material depth at some sample points are evaluated and stored before simulation. In run time, the material depth is interpolated between these sample points. For a tissue model which has already been discretized into FE model, these sample points are the nodal points.

4.4 Simulation Procedure

The procedure of surgery simulation is described in the following steps.

4.4.1 Pre-Computation

The pre-computation steps before simulation are carried out as follows.

1. Load the geometry of the tissues and support from files.
2. Discretize the virtual tissues into FE models. Evaluate the material depth at the nodal points. For the support model, evaluate the material depth at some sample points.
3. Compute the normal vector of each surface node.
4. Compute the system matrix $\mathbf{K}$ and the vector of body force $\mathbf{F}^b$.

4.4.2 Runtime Simulation

The runtime simulation is carried out as follows.

1. Detect the collision between the tool tip and the tissues. Once the collision happens, the point of contact P is detected, and the following steps are carried out after each time step, until there is no longer contact between the tool and the tissues. We assume that the tissue that is in contact with the tool tip is tissue A.
2. Define a local coordinate system at the contact point between tool tip and the tissue. For the nodes in contact with the support, a local coordinate system is defined at each node as well. The corresponding rotation matrices are calculated for the purpose of coordinate system transforming.
3. A testing step is carried out. In this step, the displacement of the contact point is set to be $\mathbf{U}_{tool}$, and the displacement of the nodes in contact with the support is set to be 0. The feedback force and the displacement of all the other surface nodes are computed.

4. Check whether the tool tip slides on the tissue surface. If it does, new contact point is determined, and the boundary condition $\mathbf{U}_{tool}$ is imposed on the new contact point.

5. Check the normal component of the supporting forces. If any of the normal component is positive, the boundary condition is changed at this point.

6. Check whether there is collision between tissue A and the support. If collision occurs, penalty forces are applied at the surface nodes that has penetrated into the support.

7. Check whether there is collision between tissue A and tissue B. If collision occurs, penalty forces are applied at the surface nodes of tissue A that has penetrated into tissue B.

8. Update all the boundary conditions of tissue A as described from Step 4 to Step 7. The displacement of all the nodal points are calculated again. The shape of tissue A deforms accordingly.

9. Check the collision between tissue A and tissue B. Penalty forces are applied at the surface nodes of tissue B that has penetrated into tissue A.

10. Update the boundary conditions of tissue B as described in Step 9. The displacement of all the nodal points are calculated, and the shape of tissue B deforms accordingly.

11. Go to next time step.

4.5 Summary

In this chapter, an algorithm has been introduced to deal with the object-object interactions in surgery simulation with FE modeling. Interactions between objects are classified into three types, and the methods to specify and update boundary conditions for each type of interactions are presented. Complex interactions such as sliding between tool and tissue, and the contact between soft tissues can be solved with this algorithm. Because the boundary conditions are imposed at nodal points only, no numerical integration is needed in runtime. Therefore, the update of boundary conditions is quite fast, which helps to come one step closer to real-time simulation.

In Chapter 3, the emphasis was on the formulation of a physically based deformation model. In this chapter, the focus was on physically based object-object interaction in surgery simulation. From our study, it is found that current computational limitation does not allow to incorporate both these approaches for real-time surgery simulation.

- *Computational complexity of MFEM*: To simulate tissue deformation with MFEM, a system of algebraic equations (Equation 3.3.3) need to be solved at each time step. The size of the stiffness matrix $\mathbf{K}$ increases when the number of nodes increases. The simple cube model we tested in Section 3.6.3 contains 27 nodes only, and the size of the stiffness matrix is 81 by 81. This can be simulated in real-time.

- *Computational limitation of MFEM for organs*: However, to simulate more complex models such as human organs, a much larger number of nodes are required. The time required to solve Equation 3.3.3 increases significantly with the increase in of the number of nodes. The real-time requirement can no longer be met. Therefore, MFEM is not suitable for real-time simulation. Similar results were also reported by Kim et al. [50], who stated that the global model increases linearly with the number of nodes and hence it is not possible to achieve real time

haptic updates.

More efficient object representations, object deformation, and realistic rendering need to be designed. In the next chapter a parameterized representation of shape modeling will be introduced, which helps to accomplish deformation in an effective and efficient manner.

Chapter 5

Deformable Geometry Maps Representation

5.1 Introduction

In this chapter we propose a new technique that is far superior to the conventional rigid Geometry Images approach by Gu et al. [44]. Our approach not only flattens the geometry, but also helps to accomplish deformation in an effective and efficient manner. Our approach is suitable for haptics computing, as it performs the deformation on the geometry map itself thereby avoiding the expensive 3D deformation computation. We demonstrate construction of the Deformable Geometry Map (DGM) representation in this chapter. Its application utilizing practical methods for interactive surgery simulation and interactive textile simulation are presented in Chapter 6 and Chapter 7.

5.2 Surface Parameterization

Certain types of surfaces, such as parametric surfaces and PDE surfaces, have natural parameterization. PDE surfaces are defined as solutions of Partial Differential Equations (PDEs). Du and Qin [38] presented an integrated approach that incorporates

PDE surfaces into the powerful physics-based modeling framework.

Unlike parametric surfaces or PDE surfaces, polygon meshes lack natural parameterization. Grimm [42] used manifolds for representing parameterization. Manifolds have the ability to handle arbitrary topology and represent smooth surfaces. Grimm presented specific manifolds for several genus types, including sphere, plane, n-holed torus, and cylinder. Grimm also introduced an algorithm for establishing a bijective function between an input mesh and the manifold of the appropriate genus. For most applications, the parameter space is preferably a 2D plane instead of a 3D genus. Lee et al. [52] introduced a piecewise linear parameterization of 3D surface that guarantees one-to-one mapping without foldovers. This technique requires solving a simple, sparse linear system with coefficients based on the geometry of the mesh. The non-negative coefficients are calculated with local straightest geodesics. The resulting parameterization guarantees visual smoothness of iso-parametric lines and preserves the conformal structure of the input mesh.

The parameterization of surface meshes provides the solution to various problems, such as remeshing and geometry images. It is most commonly used in computer graphics for texture mapping to enhance the visual quality of polygonal models [45].In texture mapping, the parameters are used to specify the coordinates of the 2D texture image for the corresponding vertices. Alliez et al. [2] introduced a flexible and efficient technique for interactive remeshing of irregular geometry, with the help of parameterization. Gu et al. [44] proposed to take advantage of parameterization and remesh an arbitrary surface onto a completely regular structure called *Geometry Image*, which captures rigid geometry as a simple 2D array of quantized points. Surface information like normals and colors is stored in 2D arrays too. The connectivity between sample vertices is implicit and therefore the data is more compact. In our approach, we introduce a parameterized representation of surface meshes called Deformable Geometry Map

(DGM), which extends the idea of geometry images, to represent deformable objects for interactive applications. We will first briefly review the various interactive deformation methods in Section 5.3 . The procedure to built a DGM representation from an arbitrary 3D mesh is described in Section 5.4.

5.3 Interactive Deformation Methods

There exist a variety of approaches to model the deformation of 3D objects. We classify the methods into two general categories: non-interactive deformation and interactive deformation. Our domain of interest is interactive deformation, which can be categorized into in-place deformations and mapped deformations, with subcategories in each group, as shown in Figure 5.1.

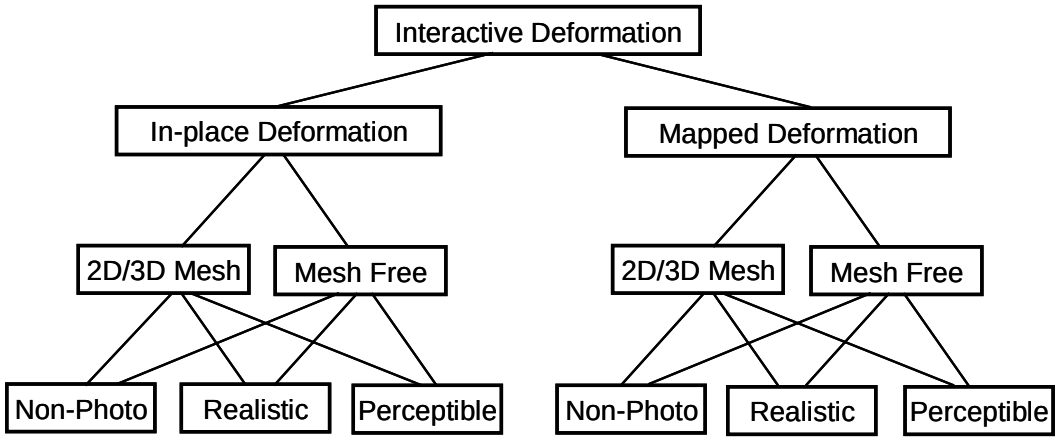


Figure 5.1: The categorization of interactive deformation methods.

5.3.1 In-Place Deformation

Onoue and Nishita [64] proposed a deformation algorithm for ground surfaces composed of granular material such as sand. In their proposed algorithm, objects and granular material on them are represented by a layered data structure called the Height Spans

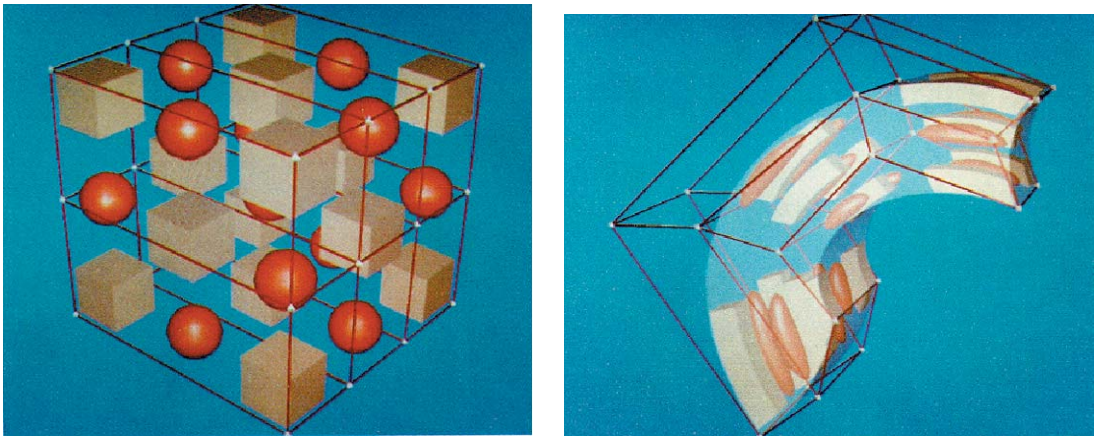
(HS) Map, which is a two-dimensional array of Height Spans. Each height span represents the span of an object and the height of the granular material at each grid point or column. When a column collides with solid objects, the granular material of this column is displaced to its neighboring columns, and thus the ground surface is deformed. The technique of texture sliding is used to render the motion of granular material. This method can simulate the contact between solid objects and the granular material interactively in a realistic way.

De and Bathe [29] introduced the meshless technique for deformation, called the method of finite spheres. De et al. [31, 32] and Basdogan et al. [8] extended this meshless method and applied it to surgical simulation for real time deformation of soft tissues. When the virtual surgery tool touches the organ model, a collection of sphere nodes are sprinkled around the tool tip, both on the surface of the organ model as well as inside. Because the computationally expensive process of defining the relative location of the nodes and the tool tip, and the process of computing the stiffness matrix are off-line, the runtime deformation can be performed in a fast update rate, i.e. 1kHz and higher, which is adequate to provide haptic feed back [8].

Choi et al. [20] introduced a mass-spring system to model soft tissue. Tissue deformation is simulated as a process of force propagation among the mass points. The deformation is scalable simply by controlling the penetration depth. Stiffness matrix formulations and operations are not needed, and the number of nodes involved in the deformable simulation is relatively small. Therefore the update rate is sufficient for haptic rendering.

5.3.2 Mapped Deformation

Free Form Deformation (FFD) methods are generally considered as mapped deformation. In FFD, objects to be deformed are embedded in a lattice of grid points of some



(a) (b)

Figure 5.2: Free-Form Deformation [75]. (a) Undeformed objects embedded in the control lattice. (b) Deformation mapped from the control lattice to the models.

standard geometry, such as cube or cylinder. By adjusting some of the control points of the lattice, the deformation is mapped from the control lattice to the target model [75], as shown in Figure 5.2.

FFD was initially used as a geometric modeling technique. For instance, Coquillart [24] used FFD to change the shape of a surface either by adding arbitrarily shaped bumps or by bending it along an arbitrarily shaped curve. Coquillart and Jancène [25] first proposed to use FFD for interactive animation of deformable objects, by animating the control lattice.

Simple FFD is a powerful tool that offers flexibility to manipulate three dimensional objects and is still widely used in computer graphics and animation. Schein and Elber [74] employed FFD to properly place deformable objects in arbitrary terrain. The locomotion of non-rigid objects, mostly animals, can be simulated realistically in their approach. In Blackett et al. [11], FFD is employed for model creation, customisation and visualization in biomedical applications.

Chen et al. [18] proposed a novel FFD method with a parametric surface instead

of a lattice-like structures. The method can be used to produce animation like a dolphin swimming along a parametric surface with shape deformation in real time. The parametric surface is flattened to a 2D plan first, by utilizing the technique of texture mapping with less distortion. The model to be deformed is then mapped onto the flattened surface. Based on the corresponding mapping between the flattened surface and the parametric surface, the model shape is deformed.

5.3.3 Motivation for Deformable Geometry Map

As mentioned in Section 2.1, interactivity and biomechanical realism are two conflicting characteristics of deformable models. Any one of them is promoted to the detriment of the other. Interactivity is mostly chosen over realism for real-time surgical simulators. Especially, haptic rendering entails much higher demand on the update rate, therefore requires much higher computational speed.

All the existing deformable models used for surgery simulation deal with the 3D model directly, and the problems are: (1) Deformation in the 3D space is expensive to compute. (2) Collision detection between the surgery tool and 3D organ models are computationally expensive.

Deformable Geometry Map (DGM) which is going to be introduced in this chapter reduces the size of the problem to 2D space, therefore significantly reduces the computational cost and greatly increases the interactivity. The strengths of DGM are:

- Deformation is calculated on the 2D geometry map and then mapped to the original mesh in the 3D space. Therefore the deformation process is more efficient and is suitable for interactive application with haptic feedback.

- The geometry map also includes the ability to interact with the 3D shape model using the 2D parameter space instead of the 3D mesh. Thus the collision between

tool and surface is handled efficiently.

- DGM also provides realistic deformation and visual rendering. The realism is not sacrificed for the interactivity.

5.4 Geometry Map Representation

A parameterized mesh is a regular 2D array of quantized points in 3D space of the following form.

$$V_{ij} = (x_{ij}, y_{ij}, z_{ij}), 0 \leq i \leq m, 0 \leq j \leq n \tag{5.4.1}$$

where

i and j are the surface parameters,

m and n are the dimensions of the parameterized mesh,

x_{ij}, y_{ij}, and z_{ij} are the x, y, z coordinates of the points.

The virtual 3D object meshes are usually arbitrary with low resolution. To be used for deformable simulation, they are first parameterized and then resampled into regular point arrays with high resolution.

5.4.1 Parametric Geometry Map Representation

Surface parameterization is the process of mapping each individual surface patch to an isomorphic triangulation on a 2D plane, so that there is one-to-one correspondence between each vertex (x, y, z) on the original 3D mesh and each vertex on the 2D parameter space (u, v), with $0 \leq u \leq 1$ and $0 \leq v \leq 1$.

Many different ways of surface parameterization have been proposed in the literature. Floater and Hormann [39] have surveyed various approaches up to 2001. Recently developments include the geodesics-based approach by Lee et al. [52]. These approaches

are mostly automatic and aim to minimize the distortions in angles and areas of triangles.

For the sake of easy implementation, we choose to conduct the parameterization of the 3D meshes semi-automatically with 3DS Max. The procedure is as follows:

1. *Cutting*: The original 3D mesh is usually of the topology of a sphere or cylinder. The triangle mesh is cut along a network of edge paths, so that the resulting triangle mesh is of the topology of a plane.

2. *Unfolding*: The triangle mesh is unfolded and mapped automatically onto a flat plane, which is the UV parameter space.

3. *Aligning*: The vertices on the margin of the triangle mesh are manually aligned onto the edges of a unit square on the parameter space.

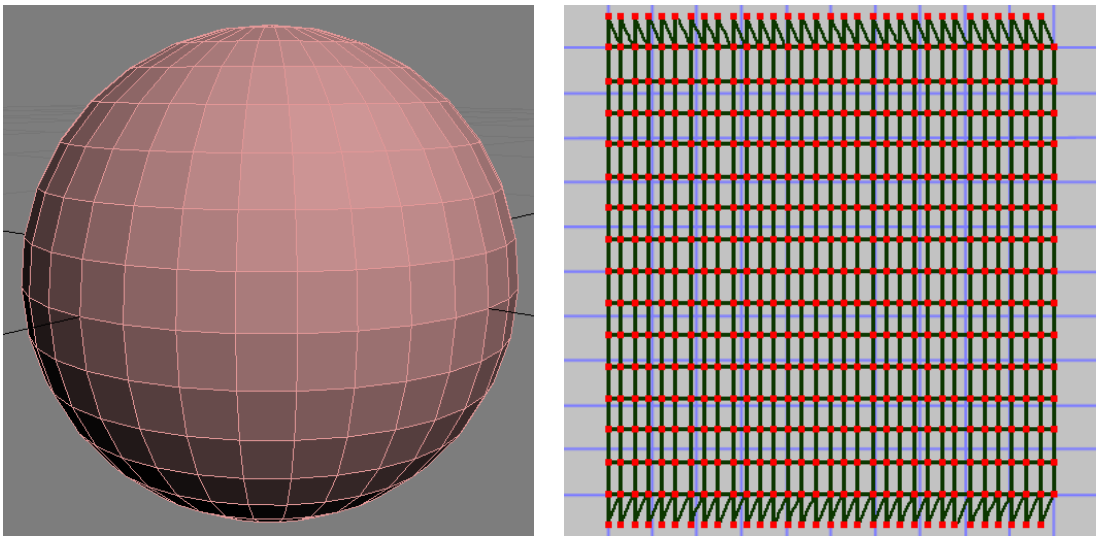
4. *Relaxation*: A number of relaxation loops are carried out automatically to minimize the distortions and to avoid overlapping of triangles.

After the parameterization process, we get a unite square on the parameter space, with triangles filled inside. Each triangle and each vertex have a corresponding triangle and vertex on the original 3D surface. Figure 5.3 shows the parameterization of a sphere.

5.4.2 Resampled Geometry Map Representation

After parameterization, the model is resampled into $m \times n$ points which are evenly distributed in the parameter space.

Each sampling point is indexed as $P_{ij} = (u_i, v_j)$, where $1 \leq i \leq m$, $1 \leq j \leq n$, and $0 \leq u_i \leq 1$, $0 \leq v_j \leq 1$. For each sampling point, we check for the triangle it resides.



(a) (b)

Figure 5.3: The parameterization of a sphere. (a) The input 3D mesh, and (b) The parameterization of the mesh.

The sampling point lies either on the vertex, edge or interior of a triangle. If it is on the vertex, we pick the first triangle that shares this vertex. If it is on the edge, we pick the first of the two triangles that share this edge. If a sampling point P_{ij} lies within a triangle $V_1'V_2'V_3'$, which corresponds to triangle $V_1V_2V_3$ on the 3D mesh, we calculate the barycentric coordinates (w_1, w_2, w_3) of P_{ij} as shown in Figure 5.4.

$$w_1 = \frac{u_i(v_2 - v_3) + v_j(u_3 - u_2) + u_2v_3 - u_3v_2}{A} \tag{5.4.2}$$

$$w_2 = \frac{u_i(v_3 - v_1) + v_j(u_1 - u_3) + u_3v_1 - u_1v_3}{A} \tag{5.4.3}$$

$$w_3 = \frac{u_i(v_1 - v_2) + v_j(u_2 - u_1) + u_1v_2 - u_2v_1}{A} \tag{5.4.4}$$

where A is the area of triangle $V_1'V_2'V_3'$ and

$$A = u_1v_2 + u_2v_3 + u_3v_1 - u_1v_3 - u_2v_1 - u_3v_2 \tag{5.4.5}$$

The barycentric coordinates are used to interpolate the coordinates of a 3D point

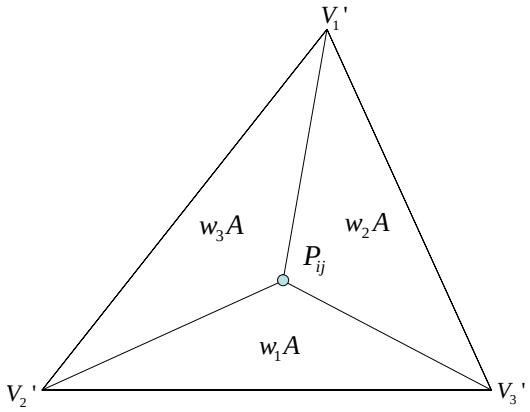


Figure 5.4: The barycentric coordinate of P_{ij} which lies inside the triangle $V_1'V_2'V_3'$.

V_{ij}, which corresponds to the vertex P_{ij} on the 3D mesh,

$$V_{ij} = w_1V_1 + w_2V_2 + w_3V_3 \tag{5.4.6}$$

Similarly, the normal of V_{ij} can be calculated via interpolation.

5.4.3 Reconstructed 3D Representation of Geometry Map

After sampling at each point, we get a $m \times n$ point array in the parameter space. Each point corresponds to a vertex on the 3D space, with (x, y, z) coordinates and normal information. With this point array, the 3D mesh of the virtual object can be reconstructed. The reconstruction of geometry map is the reverse process of surface parameterization. The procedure is as follows.

1. *Point Connectivity*: Points next to each other in the array are connected, as shown in Figure 5.5 (b).

2. *UV to 3D Mapping*: Each point P in the UV parameter space is mapped to a vertex V in the 3D space. The (x, y, z) coordinates of V are the coordinates information stored at P.

3. *Polygon Connectivity*: The connections between points are mapped to 3D space as well. For instance, if two points P_1 and P_2 are connected in the parameter space, their corresponding vertices V_1 and V_2 are connected in the 3D space too, as shown in Figure 5.5 (c).

4. *Constructing Triangles*: Based on the connections, triangles are constructed by using the connections as their edges.

5. *3D Rendering of Surface*: Each triangle is sent to the renderer for texture mapping and rendering. The surface mesh of the 3D model is thus reconstructed.

The algorithm of 3D reconstruction is outlined below:

for v

for u

get the coordinate information x_1, y_1, and z_1 of point $P_1 = (u_i, v_i)$

map P_1 to a vertex in the 3D space $V_1 = (x_1, y_1, z_1)$

get the coordinate information x_2, y_2, and z_2 of point $P_2 = (u_{i+1}, v_j)$

map P_2 to a vertex in the 3D space $V_2 = (x_2, y_2, z_2)$

get the coordinate information x_3, y_3, and z_3 of point $P_3 = (u_i, v_{j+1})$

map P_3 to a vertex in the 3D space $V_3 = (x_3, y_3, z_3)$

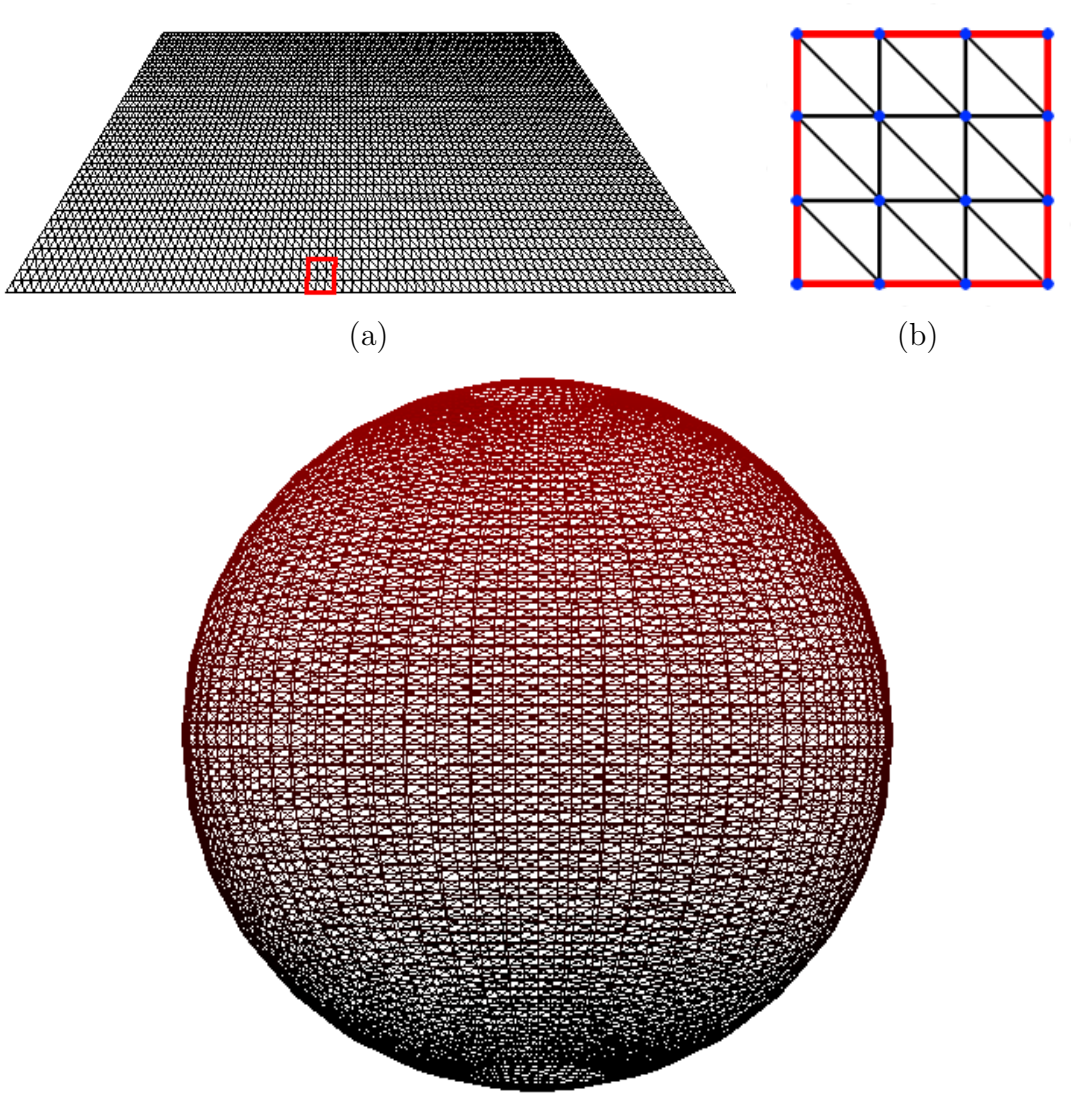
get the coordinate information x_4, y_4, and z_4 of point $P_4 = (u_{i+1}, v_{j+1})$

map P_4 to a vertex in the 3D space $V4 = (x4, y4, z4)$

render triangle $P_1P_2P_4$

render triangle $P_1P_4P_3$

In Figure 5.5 the resampled mesh is of 80 × 80 resolution. The connectivity of vertices is implicit, and requires no additional storage space.



(a) (b)

(c)

Figure 5.5: Reconstruction of the 3D sphere. (a) The 2D point array with neighboring points connected. (b) Closeup look of the connection of the points. (c) The reconstructed 3D sphere.

5.4.4 Representation of Parametric Deformation

We designed a fast and easy free form deformation method which is based on the parameterized representation of the organ model. Although the scheme is simple, realistic deformation can be achieved.

Deformation is calculated based on 2D parameter space. When one vertex C is touched by the tool and moved, the amount of displacement $d_c = [d_{xc}\ d_{yc}\ d_{zc}]$ of this vertex is transferred to the corresponding point $P_{i_c j_c}$ on the 2D parameter space. A Gaussian distribution function is evaluated at each surrounding point, and the displacements of these points are calculated with this distribution function as

$$d(P_{ij}) = d_c e^{-\frac{(i-i_c)^2+(j-j_c)^2}{\sigma}} \tag{5.4.7}$$

In the equation above, σ is the standard deviation of the distribution, and is a parameter that reflects the material property of the deformable object. Figure 5.6 shows the shape of the Gaussian distribution function in the i direction. The amount of displacement that vertex V undergoes, depends on the distance between its corresponding

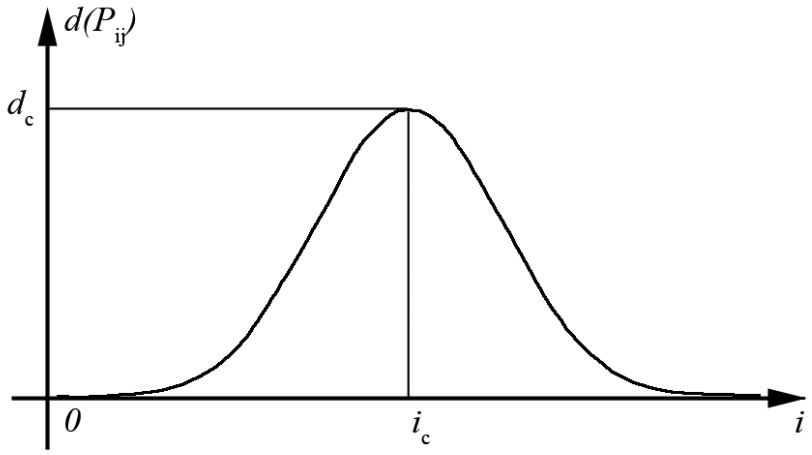


Figure 5.6: The shape of the Gaussian distribution function in the i direction.

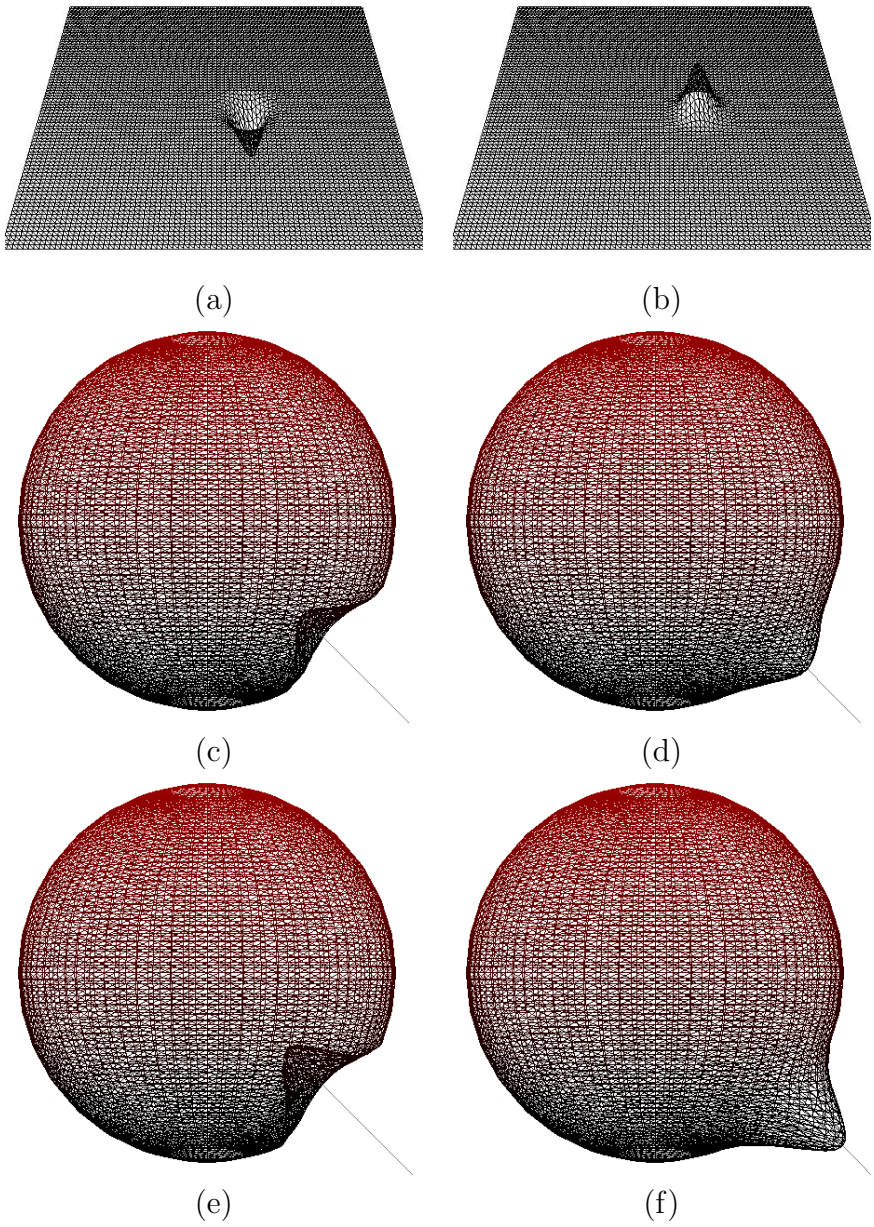


Figure 5.7: The deformation of the sphere with free form deformation: (a) (b) the deformation calculated on the 2D plane; (c), (d), (e), and (f) the deformation mapped onto 3D mesh.

vertex V' on the parameter space and $P_{i_cj_c}$. The further V' is from $P_{i_cj_c}$, the less displacement V undergoes. The displacements of the vertices are evaluated locally, i.e., in Equation 5.4.7, $(i_c - b) < i < (i_c + b), (j_c - b) < j < (j_c + b)$, where b is a constant. Therefore, the resulting deformation is a local deformation.

The 3D coordinates of the corresponding vertex on the mesh are updated based on the calculated displacements of each point. The shape of the 3D deformable object is transformed accordingly, and hence deformed, as shown in Figure 5.7. The amount of displacement d_c of the vertex in contact with the tool is restricted to a limit d_{max}. The magnitude of the displacement $\|d_c\|$ is constrained to make sure that $\|d_c\| \leq d_{max}$, so that unrealistic deformation does not occur.

5.5 Summary

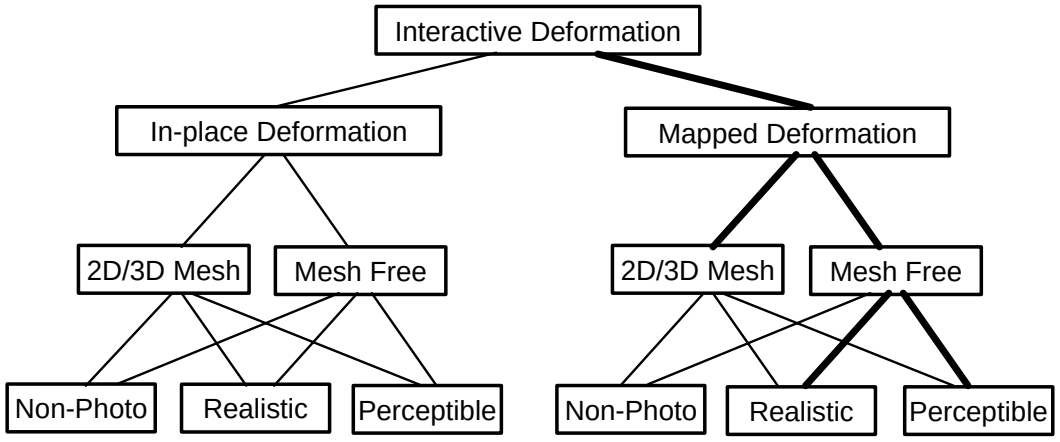


Figure 5.8: The deformation method with DGM in interactive deformation methods.

In this chapter, the Deformable Geometry Map have been presented, which is one of the major contributions of this thesis. A parameterized representation of 3D meshes was introduced which represents the shape of an object. An arbitrary 3D input mesh can be parameterized and resampled into a regular 2D parameterized model. It has also been

shown how deformation can be accomplished using this representation. The parametric deformable shape representation described in this chapter helps to approximate the local deformation.

This deformation method falls into the categories with thick lines in Figure 5.8. The deformation is calculated on the 2D geometry map and then mapped to the original mesh in the 3D space. Therefore the deformation process is more efficient and is suitable for interactive applications with haptic feedback.

In the next chapter it will be demostrated how interactions can be added to the geometry map: for example, to interact with representation of human organs for the purpose of virtual surgery simulation.

Chapter 6

Interactions with Deformable Geometry Map

6.1 Introduction

In this chapter we develop the geometry maps of human organs for surgery simulation, which can be deformed interactively with haptic feedback. An input organ mesh is processed offline to get the deformable geometry maps representation, with the procedure described in the previous chapter. Interaction and collision detection techniques are presented in Section 6.3, the haptic force feedback is presented in Section 6.4, and the method to handle contact between objects is described in Section 6.5.

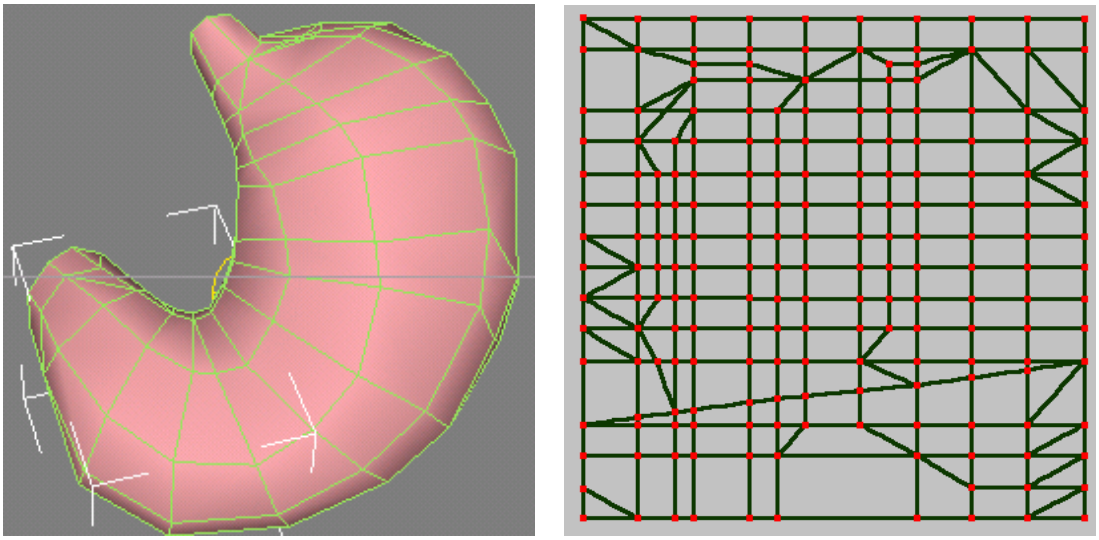
6.2 Input Data Preparation

In this section, we develop the procedure for the deformable geometry maps representation for a more complex object shown in Figure 6.1 (a). The procedure consists of the following three offline steps.

1. Geometry Map Parameterization
2. Geometry Map Resampling
3. 3D Reconstruction from Geometry Map

6.2.1 Geometry Map Parameterization

The procedure starts with an object that is a surface representation. The 3D model input to our system can be in any of the 3D model formats such as 3ds, obj, etc., which represents the object as a collection of triangles. If the vertex color and vertex normals are available, they are also used in the representation of the object in parametric form.



(a) (b)

Figure 6.1: The parameterization of a virtual stomach mesh model. (a) The original 3D stomach model. (b) The parameterized mesh of the stomach model.

Each vertex of the 3D object is projected onto the parameter space, using the procedure described in Section 5.4.1. Figure 6.1 shows the parameterization of a stomach mesh.

6.2.2 Geometry Map Resampling

To facilitate the reconstruction and deformation, we transform the parametric representation to an appropriate resolution to store the geometry data in a uniform data structure. The resolution of the resampled mesh has been selected in such a way, so that the object can be reconstructed in real-time.

For our experiments, we used a resolution of 81×81. However, with faster CPUs and geometry rendering GPUs, we can use a higher resolution for the resampled mesh. During the resampling phase, the higher resolution mesh can store not only the geometry, but also the color and normal information.

The two steps described so far, viz., geometry map parameterization and geometry map resampling are one-time operations that can be performed once offline. Figure 6.2 (a) shows the resampled mesh. Each node in the mesh contains the vertex information. This vertex information can be used to reconstruct the 3D shape during the dynamic deformation stage.

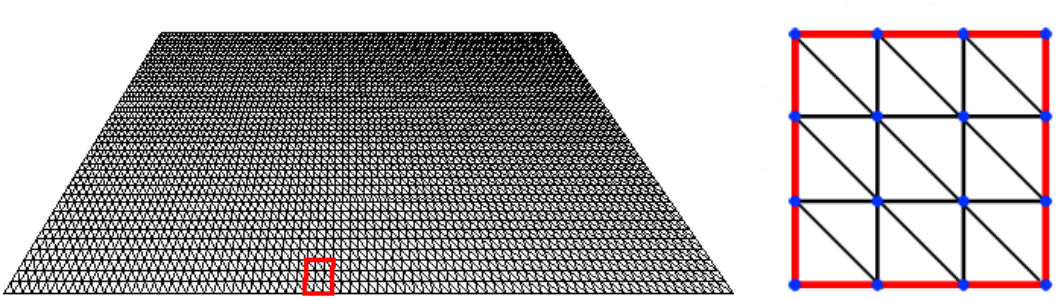
6.2.3 3D Reconstruction from Geometry Map

Figure 6.2 (b) shows a possible map of the parametric mesh onto surface triangles. Such an approach can efficiently render the shape as a triangle strip to reconstruct and to visually render the 3D mesh. Figure 6.2 (c) shows the wireframe rendering of the reconstructed object from the resampled parametric mesh.

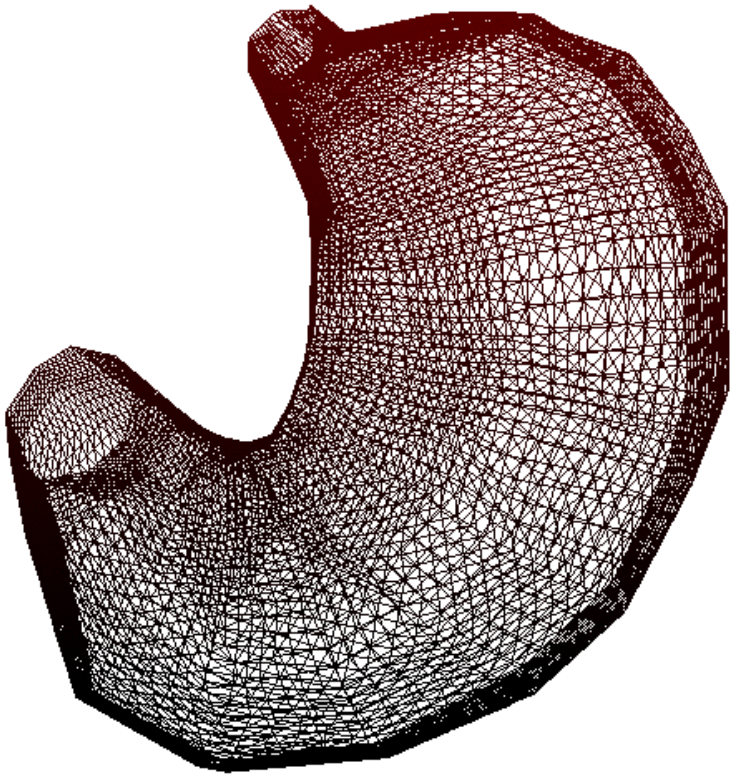
6.3 Interactions

6.3.1 Tool-Organ Interactions

To perform an interactive deformation, users can select a tool and touch the deformable object. Parameters for deformation include the contact point, the depth or height of the deformation and the radius of influence. The deformation representation is explained in Section 5.4.4. To join or blend the deformation onto the actual object, the deformation is positioned with the center of deformation to correspond with the contact point. Figure 6.3 (a) and (d) show the deformation template shown on a 2D plane. Figure 6.3 (b) shows the start phase of the inside deformation of the 3D mesh and Figure 6.3 (c)



(a) (b)



(c)

Figure 6.2: Reconstruction of the 3D mesh. (a) The 2D point array with neighboring points connected. (b) Closeup look of the connection of the points. (c) The reconstructed 3D stomach model mesh.

shows a snapshot of a larger deformation on the same object. Similarly Figure 6.3 (e) and (f) show the deformation when the object surface is pulled outward.

To accelerate the computation, since the distances between points are evaluated on a discrete 2D mesh, the Gaussian distribution value for each point can be pre-

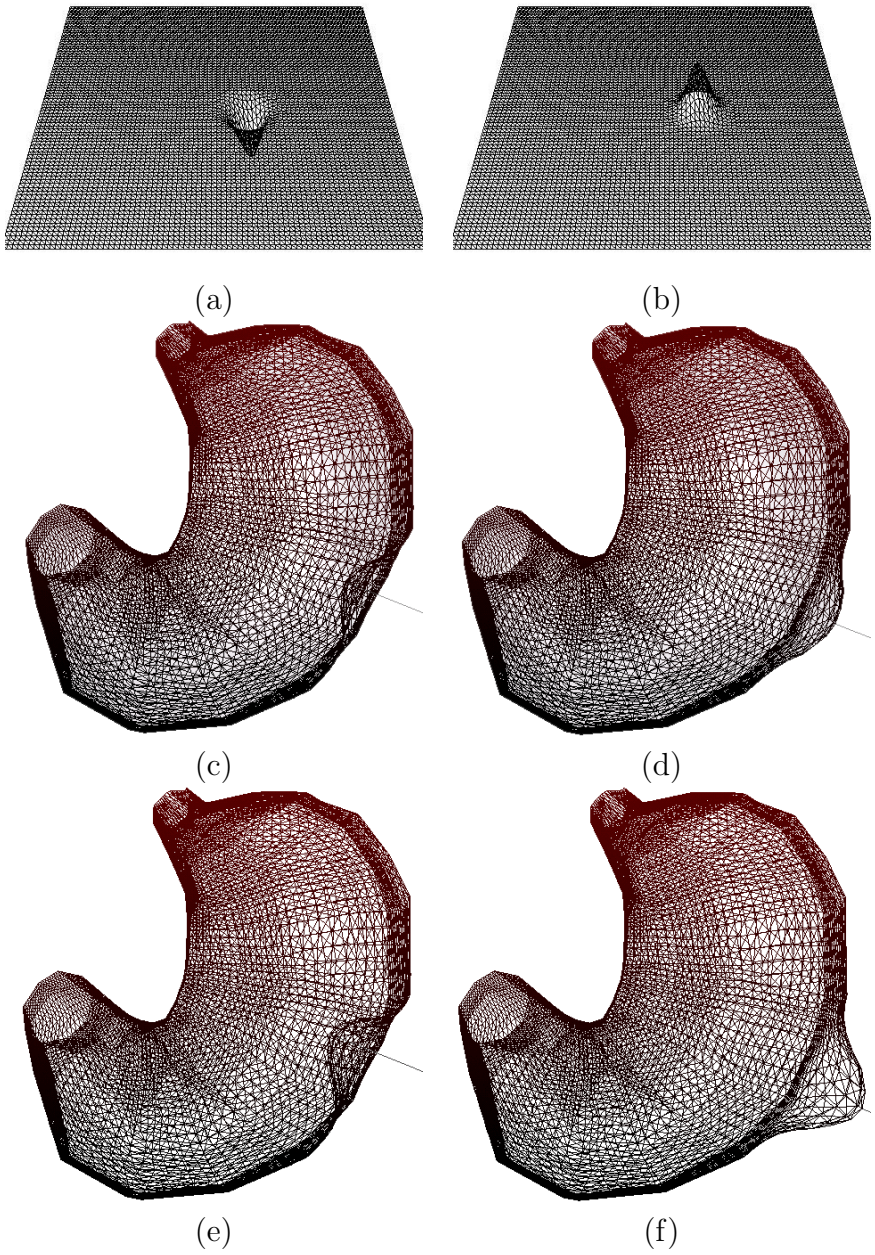


Figure 6.3: The deformation of the stomach model with free form deformation. (a) (b) the deformation calculated on the 2D plane; (c), (d), (e), and (f) the deformation mapped onto 3D mesh.

calculated. Moreover, the Gaussian distribution function has the shape of a *bell-shaped* curve. Therefore, before the simulation, we can evaluate the Gaussian distribution values locally, and store the values in a look-up table. During the simulation, we only need to cross-refer the pre-computed look-up table. By adopting this approach, very little calculation is needed, and the deformation is very fast.

6.3.2 Collision Detection in the Geometry Map

We implement one pick tool wherein the user can control the tool interactively to touch, poke, and grasp the virtual object. Like most deformation simulators, we assume that only the tool tip touches the object, and only the collision between the tool tip and the object is checked. Since the deformable mesh is resampled at high density, we can assume that the collision happens only at the vertices of the deformable mesh. This reduces the computation to a point-point collision detection.

The collision detection is performed on a 2D *Collision Map*, which is a $(2a+1) \times (2a+1)$ point array on the parameter space, where a is a constant.

Before the simulation, the distance between the tool tip and each vertex is calculated. We find the vertex V_n that has the shortest distance to the tool tip, whose corresponding point on the parameter space $V_n^{'} = (u_{i_n}, v_{j_n})$ is called *collision center*. The collision map includes the points from (u_{i_n-a}, v_{j_n-a}) to (u_{i_n+a}, v_{j_n+a}). During the simulation, with the movement of the tool tip in the 3D space, collision center and the collision map are constantly traced and updated. After each time step, the distance between the tool tip and each vertex in the collision map is calculated and compared. The new collision center is the point whose corresponding vertex has the shortest distance. The new collision map is updated accordingly, as shown in Figure 6.4. Since the movement of the tool in one time step is limited, only the points in the collision map need to be checked. The update of the collision map only involves a small number of point

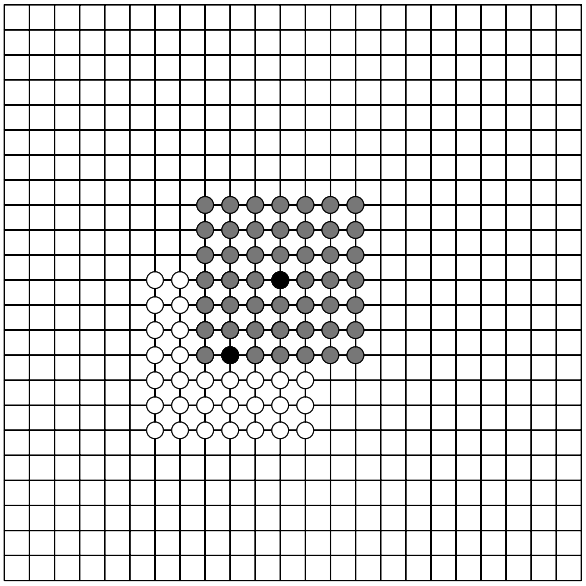


Figure 6.4: The collision map. The grey circles represent points of the collision map at current time step, and the white ones represent points of the collision map at previous time step. The black ones are the collision center at respective time step.

distances calculation, hence fast and efficient.

From the collision center V_n, two vectors are evaluated. The first vector is from this vertex to the tool tip, noted as N_t. The second is the surface normal vector at that vertex, noted as N_n as shown in Figure 6.5. If $N_t \cdot N_n < 0$, a collision is detected. The vector for V_n^t to the tool tip is the displacement of the vertex V_n^t. This displacement value is transferred to the parameter space to calculate the displacements of the neighboring vertices. The virtual object is thus deformed.

As the movement of the tool in one time step is limited, only vertices in a small area need to be checked. The collision detection procedure involves only a small number of point distance calculations and one dot product calculation. Therefore, the collision detection is fast and efficient.

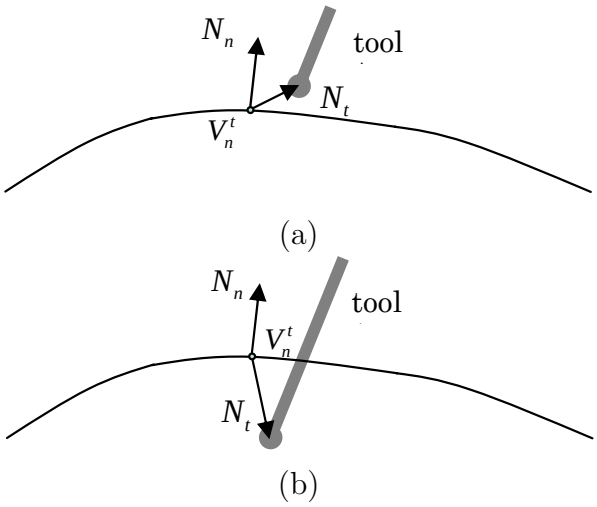


Figure 6.5: The collision detection between the organ and the tool. (a) $N_t \cdot N_n > 0$, no collision detected. (b) $N_t \cdot N_n < 0$, collision detected.

6.4 Geometry Map for Haptic Force Feedback

For rapid and reasonably accurate haptic feedback, we adopt the model used by Mahvash et al. [57, 56]. The feedback force F is computed directly based on the displacement d_c of the vertex in contact with the tool tip. The displacement d_c is decomposed into two components: d_n which is the component in the normal direction, and d_t which is the component in the tangential direction. Suppose the tool tip is of a hemisphere shape with radius r. The force feedback in the normal direction is computed analytically as

$$F_n = \frac{4}{3}\sqrt{r}E^*(d_n)^{3/2} \tag{6.4.1}$$

where E^* is defined in terms of the Young's modulus, and Poisson's ratio of the material. The force in the tangential direction is modeled as a mass-spring behavior:

$$F_t = kd_t, \tag{6.4.2}$$

where k is the spring constant. As we can see, it requires little calculation. Therefore, fast update rate can be guaranteed for the haptic device.

6.5 Contact Deformation Between Objects

Hirota et al. [46] introduced a novel penalty method to simulate mechanical contact between elastic objects based on the concept of material depth. The penalty method is used for finite-element simulation. This method results in a reliable and smooth simulation. However, it is impractical for interactive simulation due to the exorbitant computational cost.

We propose a method to handle the contact between the deformable object and a plane which can deform the object in a natural way and preserve the volume of the deformable object approximately. This method can be extended to handle the contact between the deformable object and another object of any shape. The other object can either be rigid, such as some kind of surgery tool, or be deformable, such as an organ.

The process of our proposed method is shown in Figure 6.6. First, the collision between the deformable object and the plane is detected. The vertices that penetrate the plane are projected to the plane. The areas around these vertices are then deformed so as to conserve the volume of the object.

The flowchart of the method is shown in Figure 6.7. At each time step, the following computations are performed.

6.5.1 Collision Detection

The collision between the plane and the bounding box of the virtual 3D object is checked first. If collision is detected, we further check whether the object collides with the plane. We just need to check every vertex P_{ij} in Figure 6.6 (d) to see if it is on the other side of the plane. The corresponding points (i, j) of these points on the parameter space are in a set Φ, i.e. $(i, j) \in \Phi$.

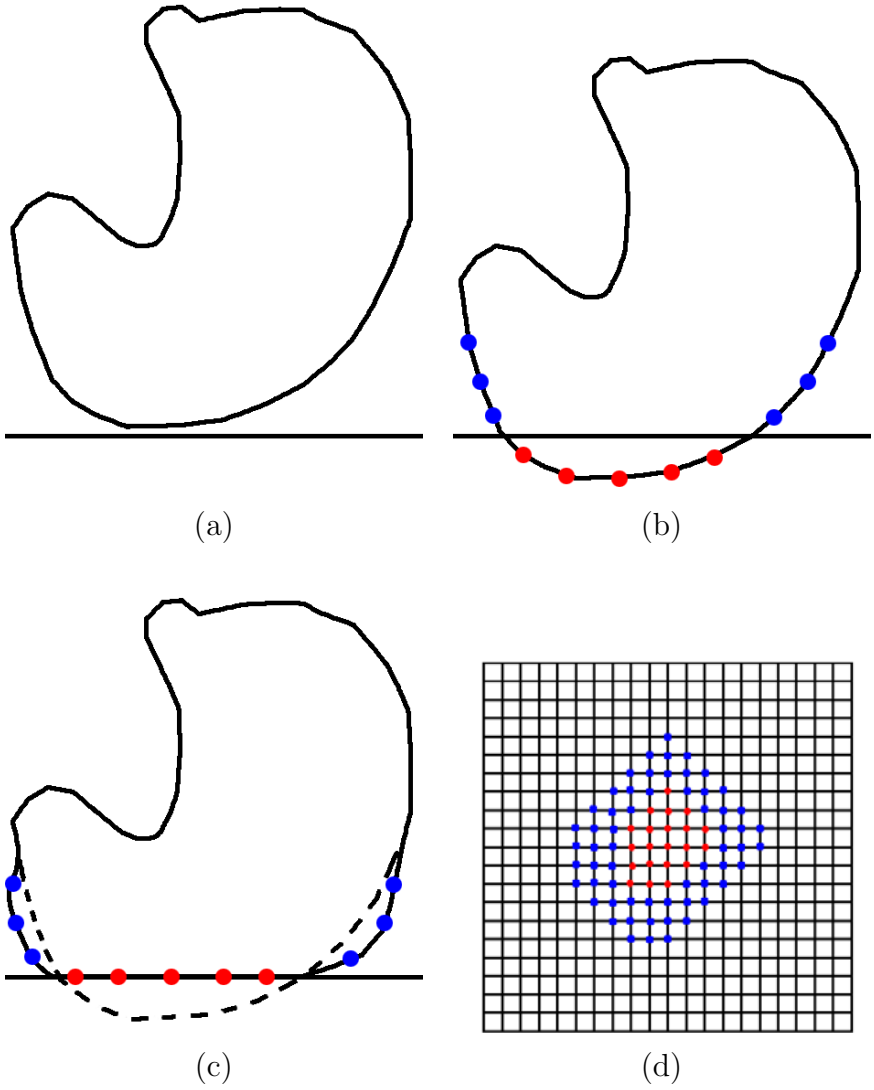


Figure 6.6: The process of the contact between a deformable object and a plane. (a) The collision between the object and the plane. (b) The vertices that penetrate the plane (in red color) are projected to the plane. (c) The vertices (in blue color) around the contact area is deformed. (d) The corresponding points in the parameter space.

6.5.2 Displacement Computation

The vertices that penetrate the plane are displaced by simply projecting onto the plane. The amount of displacement d_{ij} is calculated for each vertex P_{ij}. There are two permissible limits for amount of displacements. One is the maximum amount of displacement

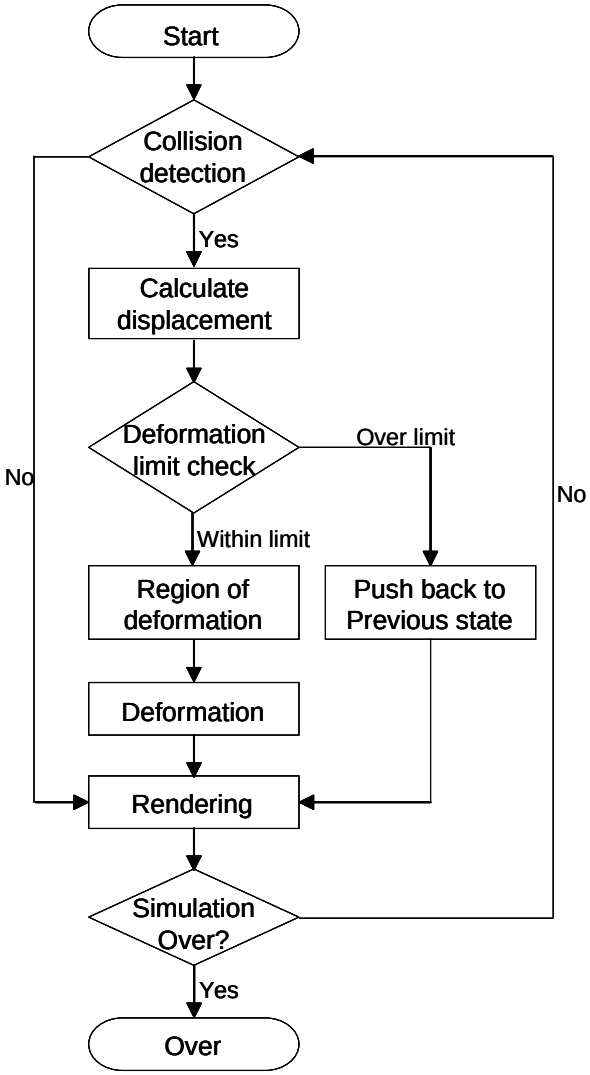


Figure 6.7: The flowchart to handle the contact between objects.

allowable for a single vertex d_{Smax}, and the other one is the maximum amount of displacement allowable for the total sum of the displacements of all the vertices that are in contact with the plane d_{Tmax}. If $|d_{ij}| > d_{Smax}$ for $(i,j) \in \Phi$, or $\sum_{(i,j)\in\Phi} |d_{i,j}| > d_{Tmax}$, the positions of all the vertices are set back to the state at the previous time step. The virtual organ is rendered, and we move on to the next time step.

If $|d_{ij}| \leq d_{Smax}$ for $(i,j) \in \Phi$, and $\sum_{(i,j)\in\Phi} |d_{ij}| \leq d_{Tmax}$, the following steps are performed.

6.5.3 Volume Preserving Local Weighted Deformation

After the contact vertices are displaced, the vertices that are around them are deformed, so as to preserve the volume of the organ. We construct an explicit deformation to the surface over a local region that is in contact with the plane. The region of the deformation Ψ is decided by the sum of the displacements of the contact vertices $D = \sum_{(i,j)\in\Phi} |d_{ij}|$. The higher the value of D, the larger is the region.

In the parameter space, we check each point (i, j) and find its nearest contact vertex and their distance s_{ij}. If the value of s_{ij} is lower than a threshold R, the vertex P_{ij} corresponding to (i, j) is in the region of deformation, i.e. $(i, j) \in \Psi$. R is defined as a function of D. The higher the value of D is, the higher is the value of R. In practice, we can pre-define a set of discrete threshold values R_k and its corresponding D_k, where $k = 1, 2, ..., p$.

If the vertex P_{ij} is in the region of deformation, the direction of its displacement d_{ij} is in its normal direction, so that the volume of the organ is increased to compensate the decrease of volume due to the contact area. The amount of displacement is defined as

$$|d_{ij}|_{(i,j)\in\Psi} = e^{-\frac{s_{ij}^2}{\sigma_k}} \sum_{(i',j')\in\Phi} |d_{i'j'}| \tag{6.5.1}$$

where σ_k is the standard deviation of Gaussian distribution that corresponds to the threshold value R_k.

Similar to the case where the object is deformed by the tool, the Gaussian distribution value for each discrete value of s_{ij} can be pre-calculated and stored. During simulation, the distribution values are looked up in the pre-calculated data so as to increase the speed of simulation.

6.6 System Architecture

We designed our surgery simulator based on the DGM representation of the organ models. The approach is simple, and realistic deformation can be achieved in real-time. The architecture of our surgery simulator is shown in Figure 6.8. The computational steps are as follows.

1. Input: The virtual 3D object meshes available are usually arbitrary with low resolution. To be used for deformable simulation, they are first parameterized and then resampled into regular point arrays with high resolution. First the specific 3D organ data is used as an input to capture the geometry map representation. The appearance of the organ surface is captured from a texture image.

2. Processing: There are two major processing steps.

   (a) In the first stage, the deformation is done on the parametric representation.

   (b) The deformations are then transformed into the actual mesh in the geometry map.

   (c) The textures are then mapped onto the 3D surface. Surface normals and bumps are added for realism.

3. Output: A photo-realistic output for any novel view can be synthesized by mapping the appropriate texture and with the addition of normal and bump mapping.

6.7 Summary

In this chapter a new simple interactive deformation procedure was introduced for achieving interactive deformation of the geometry maps, which is another major contribution of this thesis. With a parameterized representation, a high resolution 3D

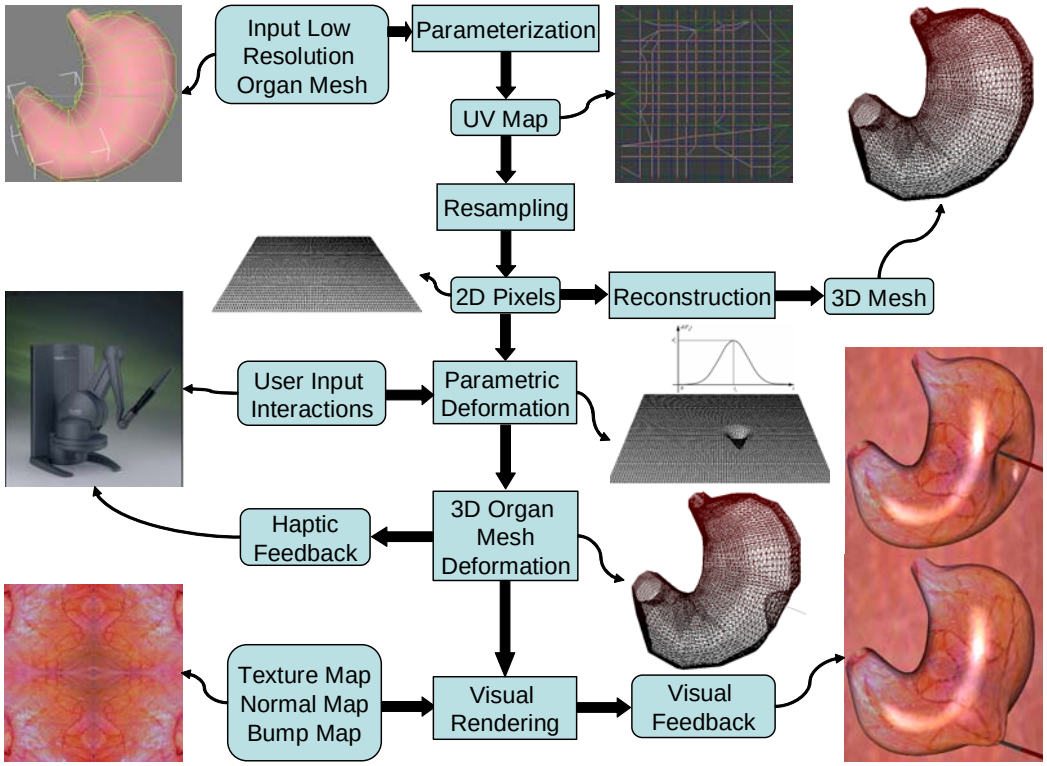


Figure 6.8: Architecture of our interactive surgery simulator.

mesh can be reconstructed and deformed interactively with a simple and fast free-form deformation method. The amount of deformation and force feedback can be calculated quickly [80, 9]. Therefore, fast haptic rendering can be achieved.

In addition, the parameterized mesh can be used to handle collision detection as well as the determination of surface contact between multi-objects in an efficient manner. In this manner, realistic visual and haptic rendering environment can be provided for interactive deformation.

Chapter 7

Experimental Results

7.1 Introduction

An important issue in surgery simulation is to model the virtual human organs in a realistic way, not only visually, but also haptically. Visually, the virtual organs should have adequate number of polygons so that the surfaces can be rendered smoothly. Textures are applied to add colors and details. Deformation methods should be applied to allow interaction between the user and the virtual organ.

7.2 Experiment Setup for Surgery Simulation

The interaction between the tool and organ was implemented. Two organ models were parameterized and resampled : stomach model and liver model, with relatively low resolution. Each is resampled by a 81×81 point array. Both resampled meshes can be simulated in real-time with user interaction. Texture and bump maps are used to add realism to the model. The textures of the organs are generated from the snapshots of real laparoscopic surgery. The bump maps are generated with Adobe PhotoShop Normal Map and DDS Authoring Plug-ins developed by NVIDIA in 2004.

7.3 Visualization of Organ Deformation

7.3.1 Initial Results

Interaction of a tool with parameterized and resampled organ models: stomach model and liver model are shown in Figure 7.1. The results show the rendered organ, the tool interaction and the deformation of the organ.

7.3.2 Cinematic Quality Rendering

The results shown in Figure 7.1 are not realistic. Hence we conducted experiments to achieve cinematic quality rendering of human liver for the use in surgery simulation. 3D mesh model of liver was used in 3D Studio Max 8 software to render high quality liver images shown in Figure 7.2.

Multi-layer textures are applied to obtain realistic effects. The 3DS tool has advanced ray tracing mechanism that has been utilized for cinematic quality visualization. Blinn shading and refraction maps are introduced to achieve a realistic liver surface. The main drawback is the enormous time required to render the object. Another major issue is interactivity. We cannot interact with the organ in real-time.

7.3.3 High Quality Rendering

In this experiment, we attempt to increase the quality of rendering and accomplish it in real-time. Bump maps are used for the vessels on the organ surface. Advanced shading techniques are employed for the glossy effect. In addition, a mesh smoothing process is conducted before parameterization to smooth the organ surface. Figure 7.3 shows two snapshots of the stomach model with more realistic rendering. The one in Figure 7.3 (a) is parameterized without surface smoothing, while the one in Figure 7.3 (b) is

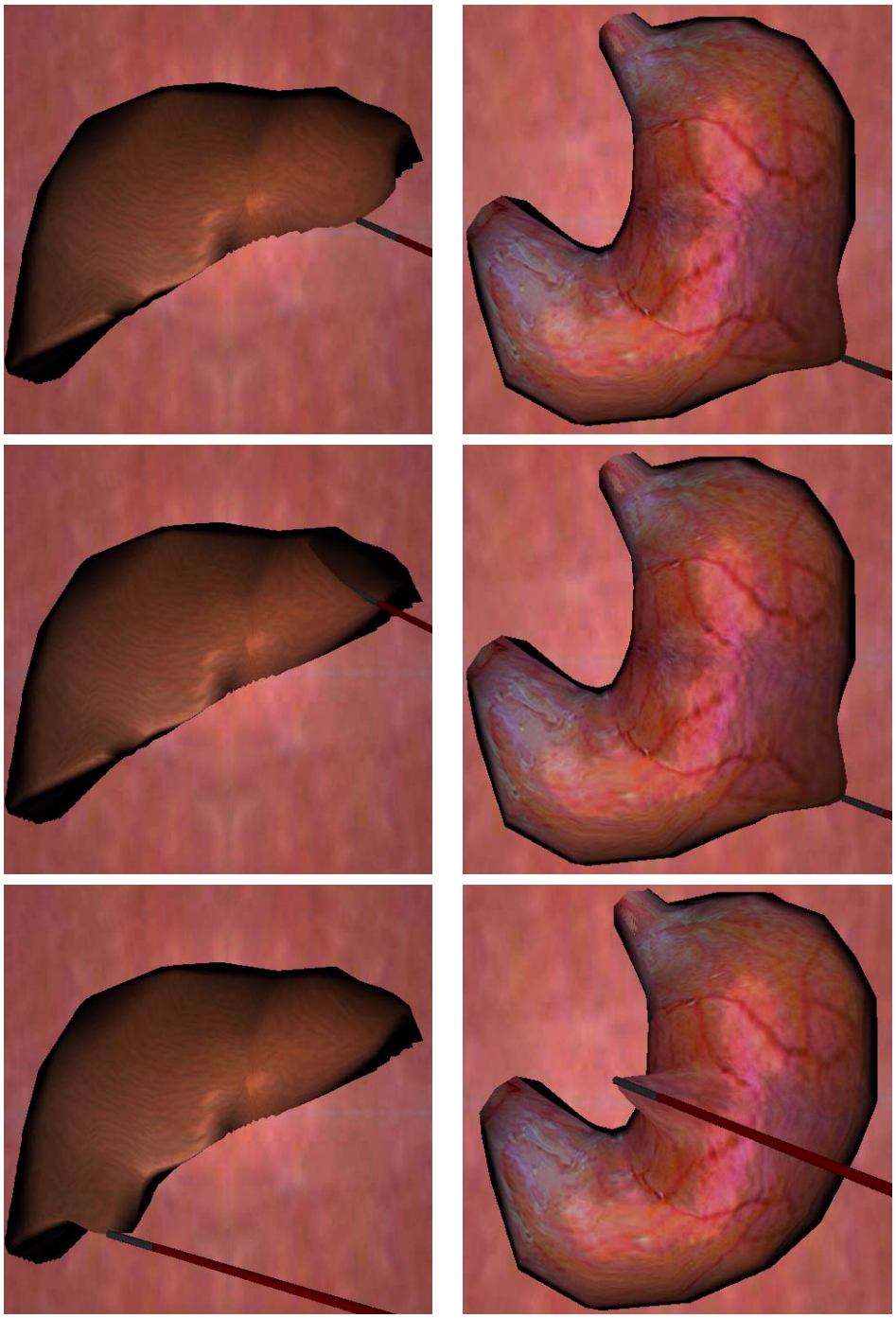


Figure 7.1: Surface deformation of the liver (left) model and the stomach model (right).

parameterized after surface smoothing.

To support interactions, we use the mouse input to the system to control the move-

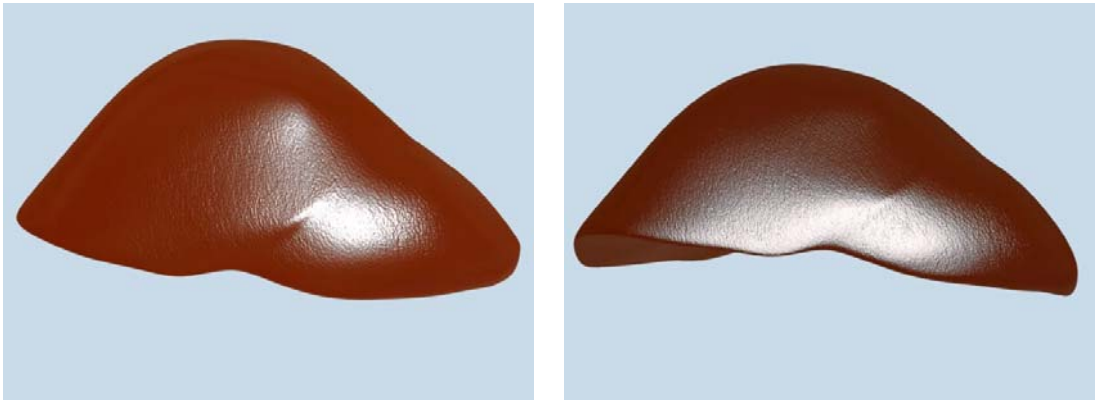
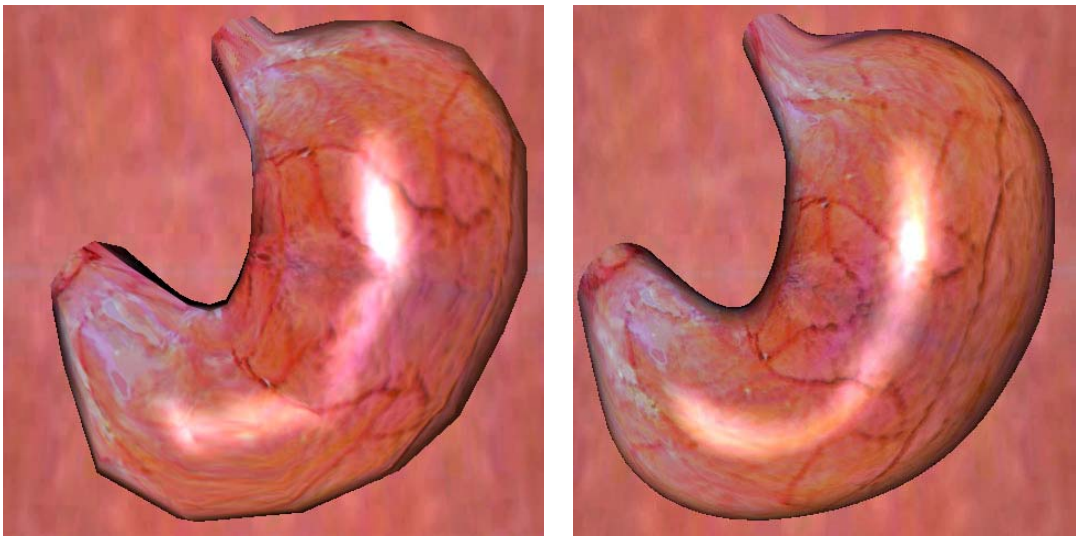


Figure 7.2: Cinematic quality rendering of the liver.



(a) (b)

Figure 7.3: Stomach model with texture and bump map. (a) Stomach model parameterized without surface smoothing. (b) Stomach model parameterized after surface smoothing.

ment of the virtual tool. Figure 7.4 and Figure 7.5 show a few snapshots of the surgery simulation with the interaction between the tool and the two organ models. Both resampled meshes can be simulated in real-time with user interaction. Texture and bump map are used to add realism to the model. The textures of the organs are obtained from real laparoscopic surgery.

to demonstrate the simplicity and usefulness of this methodology another organ, a

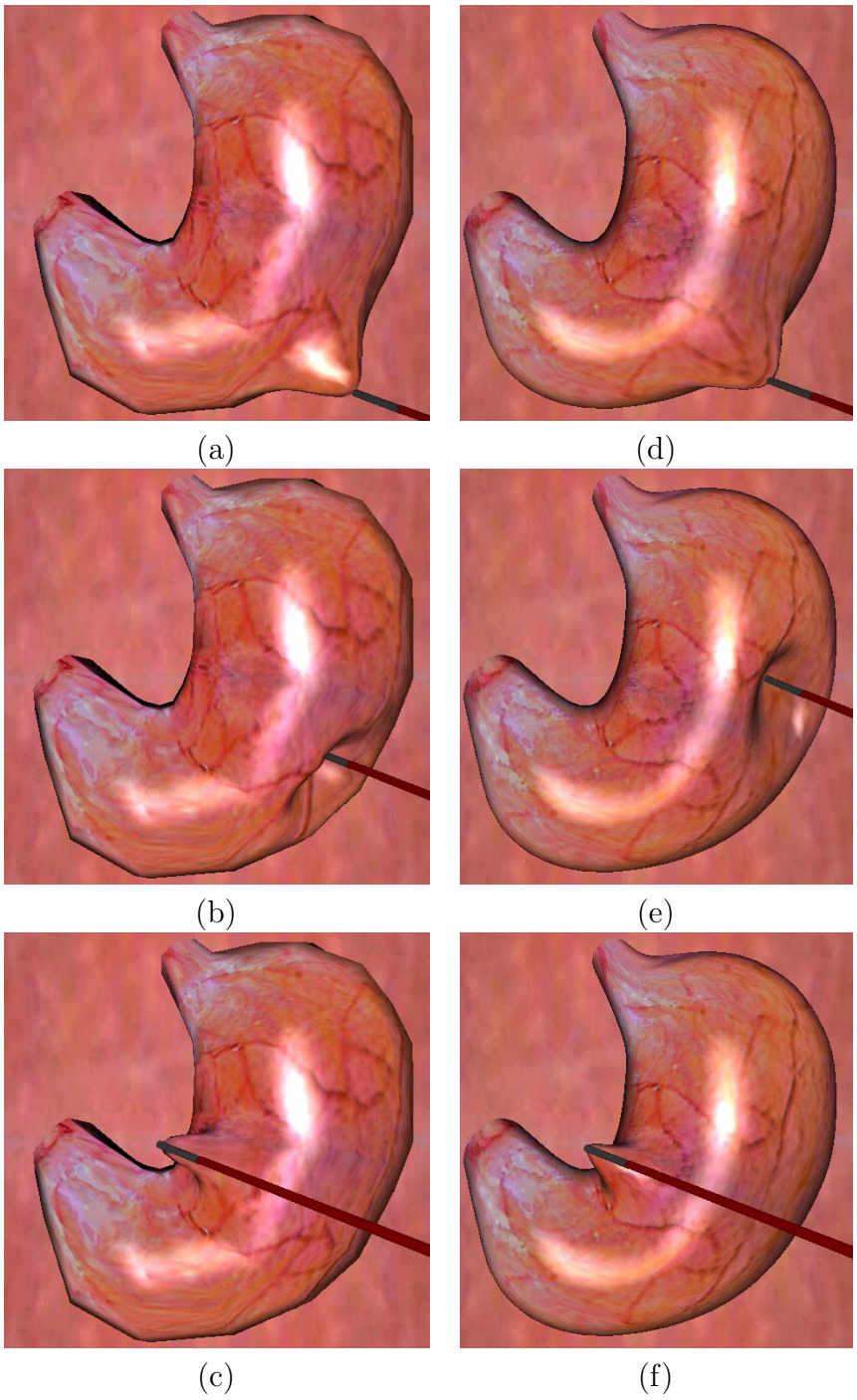


Figure 7.4: The deformation of the stomach model with free form deformation. (a) (b) (c) Stomach model parameterized without surface smoothing. (d) (e) (f) Stomach model parameterized after surface smoothing.

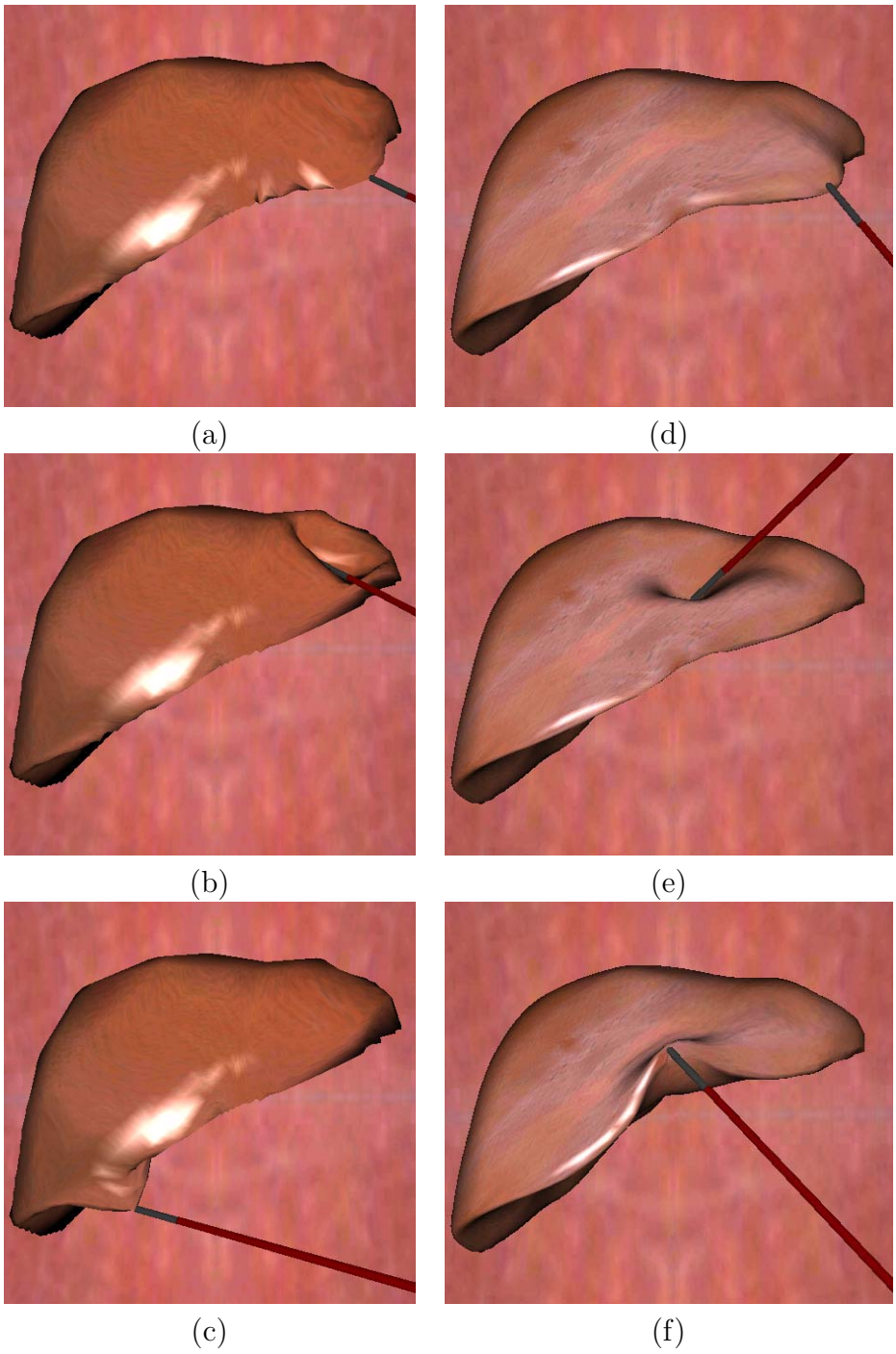


Figure 7.5: The deformation of the liver model with free form deformation. (a)(b)(c) Liver model parameterized without surface smoothing. (d)(e)(f) Liver model parameterized after surface smoothing.

heart, is modeled as deformable geometry map as an additional example. The interactions between the tool and the heart model are shown in Figure 7.6.

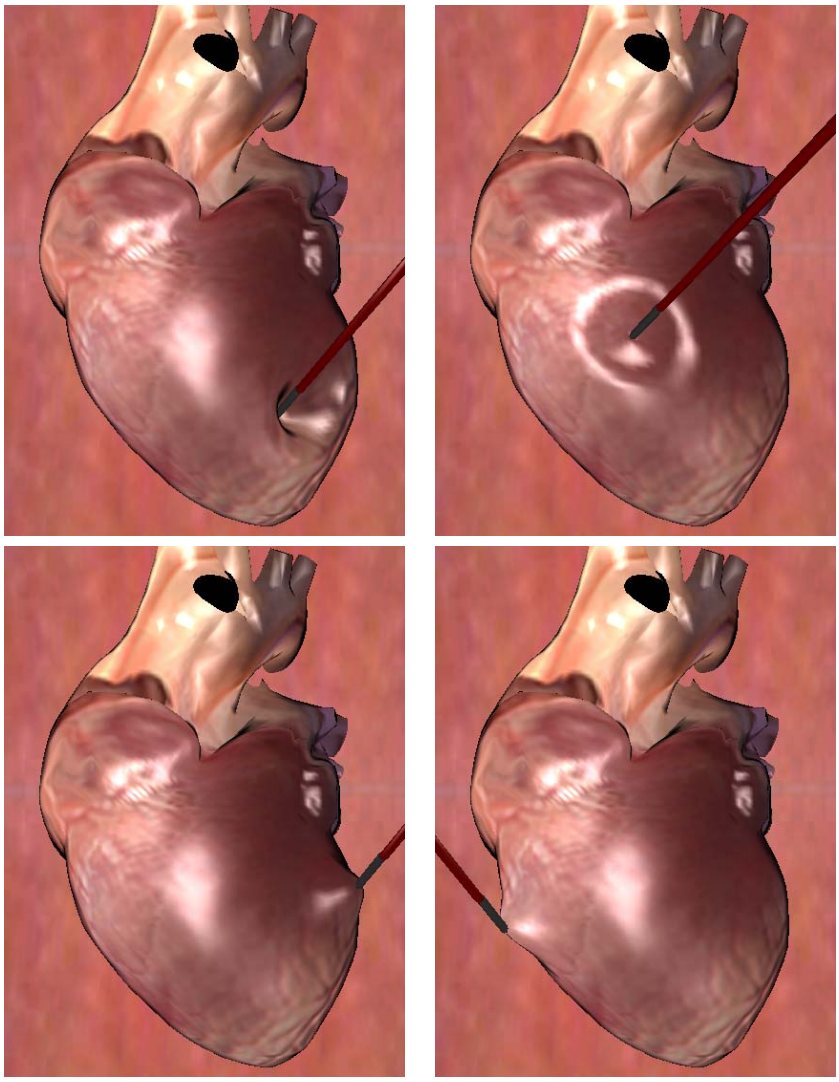


Figure 7.6: The deformation of a heart model with user interactions.

7.3.4 Organ-Plane Interaction

We also conducted experiments to simulate the organ interacting with an object such as a plane. The organ deforms when it is in contact with the plane. We used the geometry map deformation introduced in Section 6.5 to accomplish the deformation of the organ

when it rests on the plane.

The contact between the virtual stomach and a plane is shown in Figure 7.7.

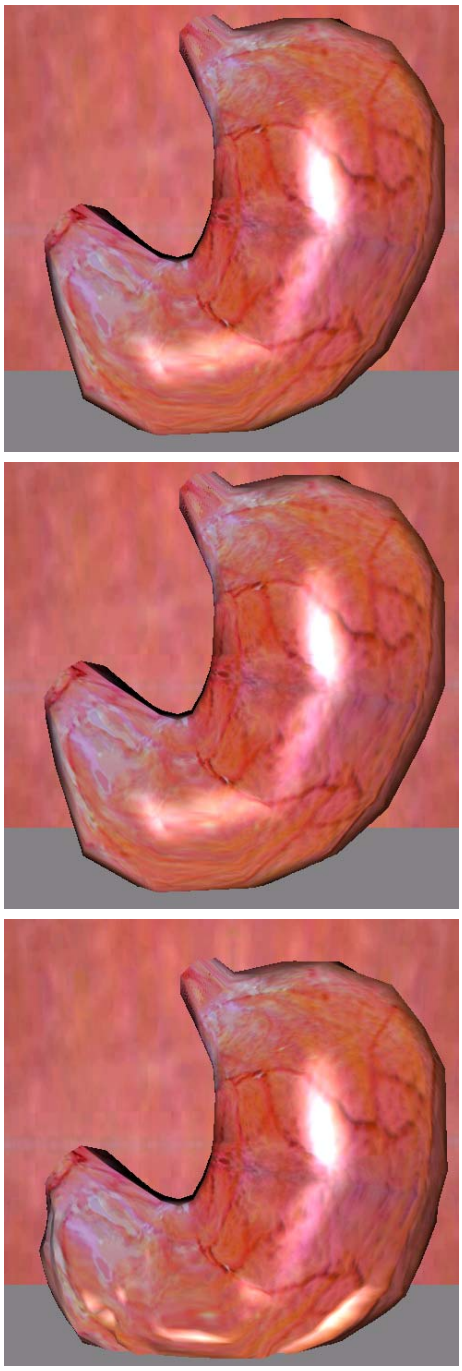


Figure 7.7: Stomach-plane contact and deformation.

7.4 Analysis of Visualization Results

Two parameters are utilized to control the shape of the deformation of the organ with tool interaction. The first parameter is the maximum allowed deformation d_{max}, which is represented as a value relative to the size of the virtual organ:

$$d_{max} = \beta \frac{(x_{max} - x_{min}) + (y_{max} - y_{min}) + (z_{max} - z_{min})}{3}$$

where β is a scalar value, and x_{max}, x_{min}, y_{max}, y_{min}, z_{max}, and z_{min} are the maximum and minimum coordinates in the x, y, z directions respectively.

The second parameter is the standard deviation of the Gaussian distribution σ. As shown in Figure 7.8, at a point E whose distance to the deformation center is l, the maximum length of the displacement is $d_E = \alpha d_{max}$, where α is the Gaussian distribution value at E. If value of α at this point is small enough, the displacement of point E is almost unnoticeable (in our implementation, $\alpha = 0.01$), and l can be considered as the radius of the local deformation area. σ can be represented as a function of l as $\sigma = -\frac{l^2}{ln\alpha}$. Now the shape of the deformation is controlled by two parameters, l and β. In Table 7.1, we compare the shape of the deformation of the stomach model with different l values and β values. Similarly, in Table 7.2, we compare the shape of the deformation of the liver model with different l values and β values.

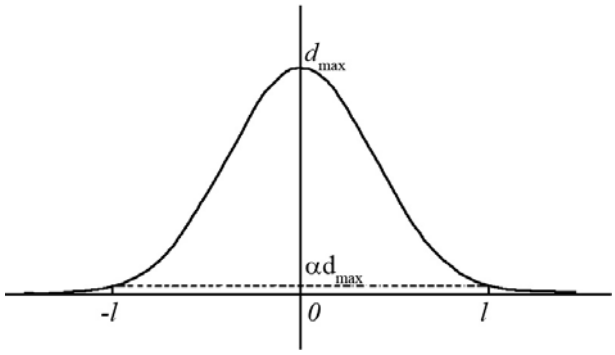


Figure 7.8: The shape of the deformation in a local area.

In our experiments shown in Table 7.1 and Table 7.2, we find that for $\beta = 0.25$, the deformations are not realistic. Similarly, when $l = 20$, the deformations are not realistic. So for simulation, we recommend the values along the diagonals to produce acceptable deformation. We can conclude that, to avoid unrealistic free-form deformation,

1. the value of β should increase with the value of l, and
2. the value of l and β should not to be too large, i.e., the deformation area and the maximum allowed deformation should not be too large.

In practice, β is set to increase with l, as a discrete function, e.g. $\beta = \beta_1$ if $0 < l < l_1$; $\beta = \beta_2$ if $l_1 < l < l_2$; $\beta = \beta_3$ if $l_2 < l < l_3$; and so on.

7.5 Time Results

The liver model is resampled with different resolutions. The simulation time results of DGM with different models and different deformation areas are listed in Table 7.3. The visual frame rate means the number of simulation iterations per second with the scene rendered in every iteration. While the haptic frame rate is the number of simulation iterations per second when the visual rendering rate is constrained at 30, i.e. the scene is rendered in every 0.033 second. The time results are collected from a commodity desktop PC with Intel Pentium D 3 GHz CPU and ATI RADEON X600 graphic card with 256 MB memory. Please note that the time results listed are simulation time, which includes not only the deformation time, but also the time for interaction and collision detection.

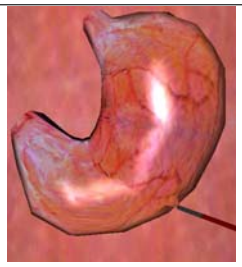
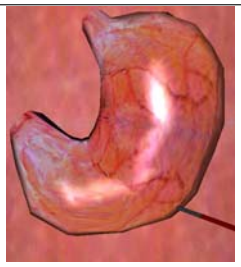
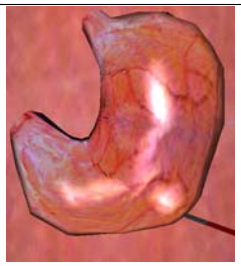
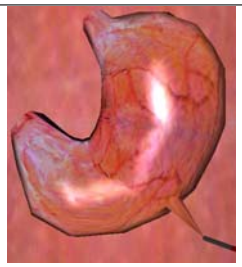
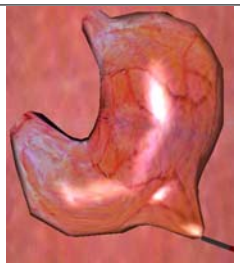
β	$l = 1$	$l = 5$	$l = 10$	$l = 20$
0.05				
0.1				
0.15				
0.25				

Table 7.1: The shape of the virtual stomach deformed with different parameter values.

7.6 Experiment Setup for Textile Simulation

We further demonstrate that our deformable geometry map can be extended to some other applications, such as the haptic deformation of textiles (Thalmann and Wolter [84]). A practical use of our approach is to apply the deformation computation for fabric wrapped onto objects. In our first experiment, we wrap textiles onto 3D deformable models. Different textile textures are used to simulate the different appearances of the

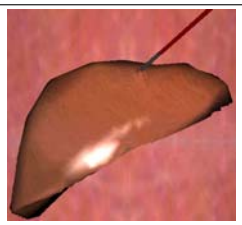
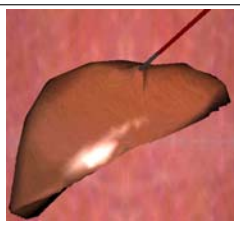
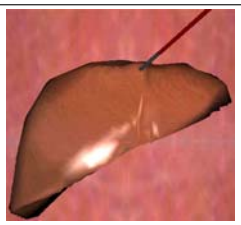
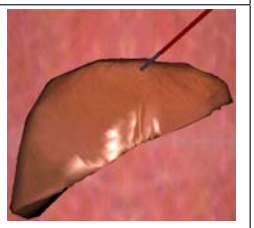
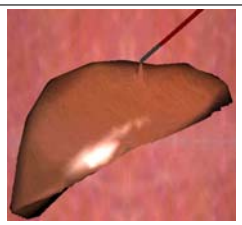
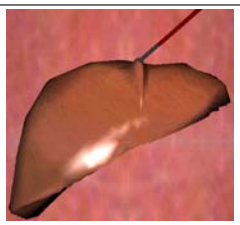
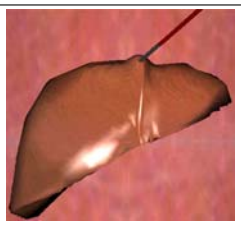
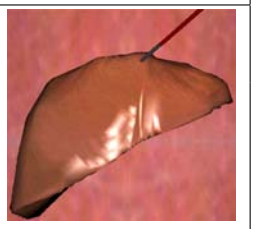
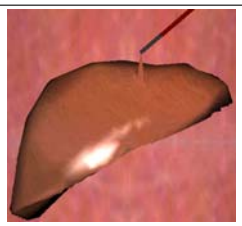
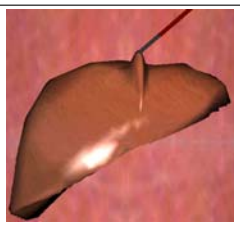
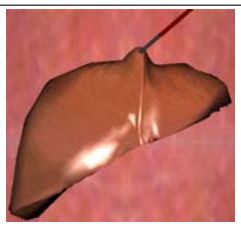
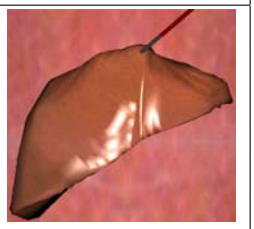
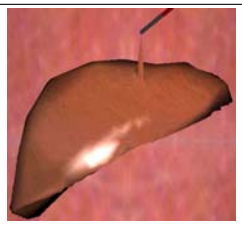
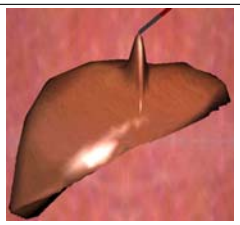
β	$l = 1$	$l = 5$	$l = 10$	$l = 20$
0.05				
0.1				
0.15				
0.25				

Table 7.2: The shape of the virtual liver deformed with different parameter values.

textiles. In the second set of experiments, we use a variable kernel for deformation of textiles. In the third experiment, we simulate the deformation of textiles wrapped on a deformable seat.

7.7 Visualization Results of Textile Simulation

7.7.1 Draped on Irregular Objects

A prototypical example of draping a textile on a 3D deformable object is considered. A typical scenario is a cover for a pillow. The pillow is deformable and hence the textile

Model	Size	Total No. of Triangles	Deformed Triangles	Visual Frame Rate (Hz)	Haptic Frame Rate (Hz)
Liver_1×	40 × 40 mesh	3,200	8	237	10,638
			200	237	4,728
			800	224	1,934
			1,800	224	990
Liver_4×	80 × 80 mesh	12,800	8	129	1,553
			200	129	1,155
			800	129	932
			1,800	129	665
Liver_6×	100 × 100 mesh	20,000	8	97	582
			200	97	518
			800	97	466
			1,800	97	358
Liver_25×	200 × 200 mesh	80,000	8	28	—
			200	28	—
			800	28	—
			1,800	27	—

Table 7.3: The simulation time results of DGM with different models and different deformation areas.

that is used to cover the pillow deforms along with it. The shape of the pillow cover is represented using DGM described in Chapter 5.

The appearance of the cover is represented as the texture. When we touch the cover, it deforms and the texture maps onto the deformed cover. This problem setting is described in Figure 7.9. For convenience, we use the deformable organ mesh instead of a real pillow. The left half of Figure 7.9 shows that the surface is pressed. The right half shows that the surface is pulled outward. The textile appearance in this example is obtained by mapping a denim fabric texture. This particular illustration describes the required behavior of a textile with the property of the *denim fabric*.

Similarly, to simulate a different types of fabric, we replace the denim texture, with the texture of a different fabric and use the same deformation approach described earlier. The surface appearance of a satin textile is rendered in Figure 7.10.

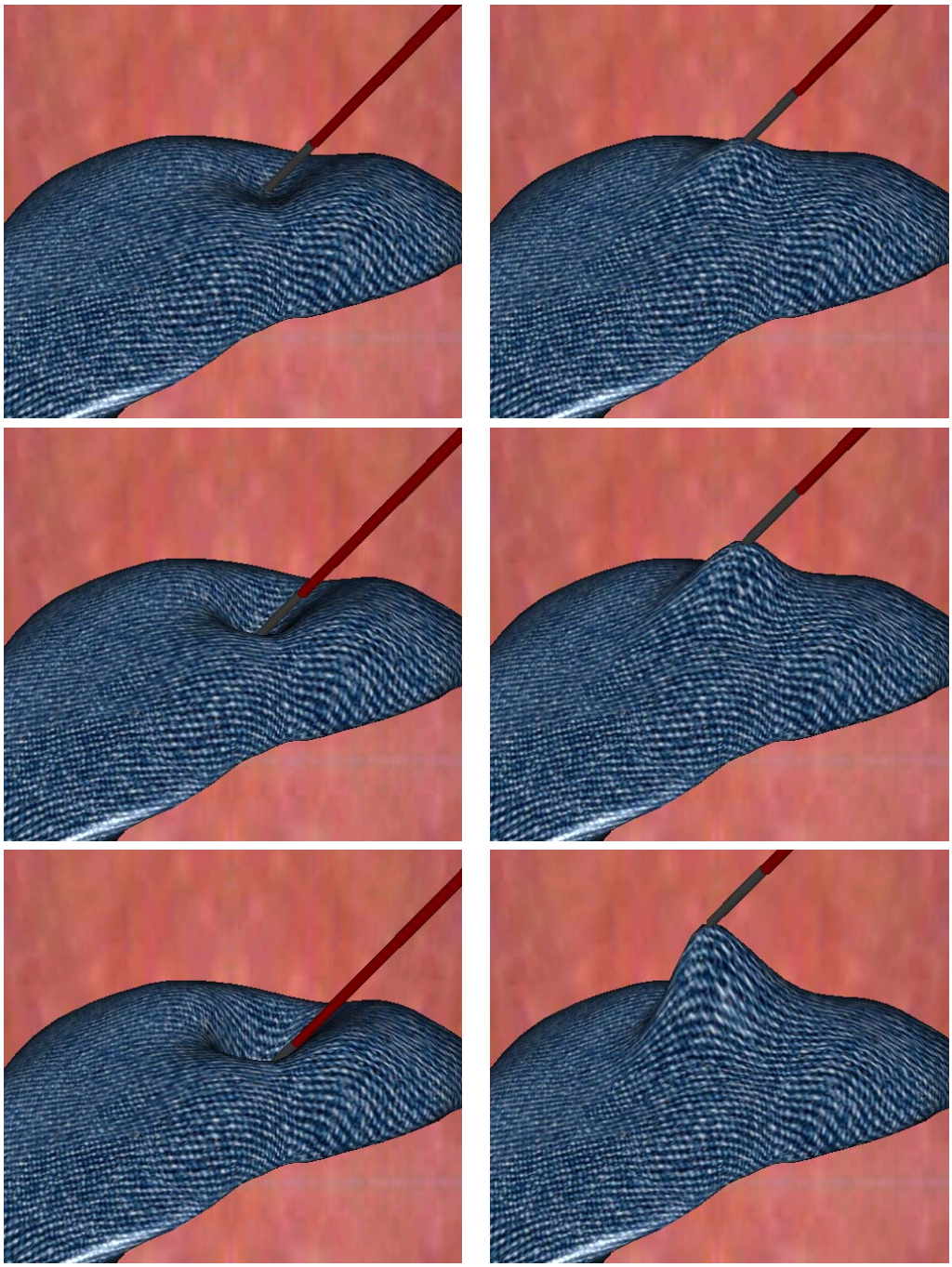


Figure 7.9: The deformation of denim fabric wrapped on the 3D deformable model.

We further experimented with some other type of textiles. The interactive deformations of the textiles wrapped on the deformable objects are shown in Figure 7.11.

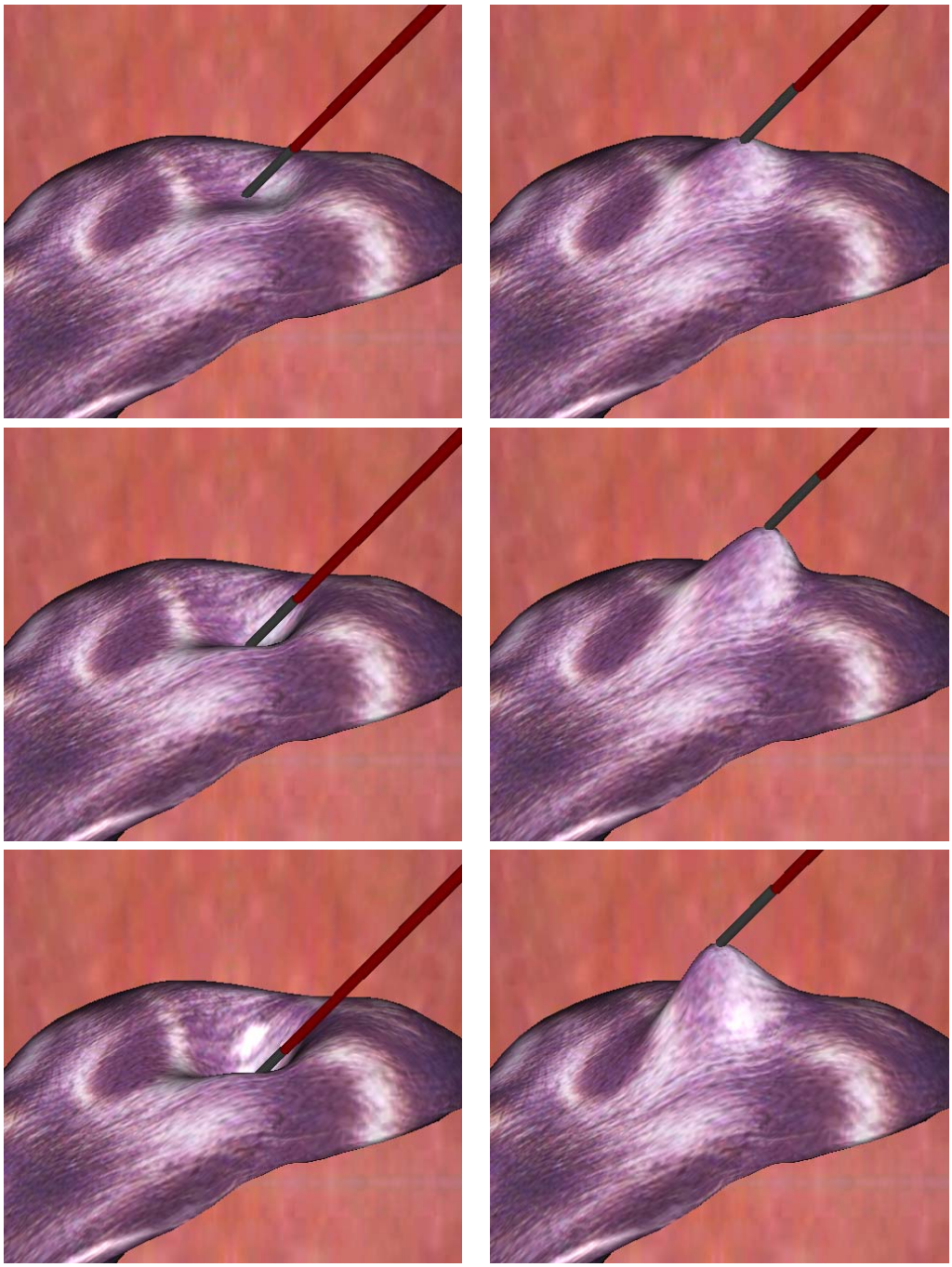


Figure 7.10: The deformation of satin wrapped on the 3D deformable model.

7.7.2 Draped on Planar Surfaces - Analysis

More generally, we can extend this by mapping a single textile texture onto different underlying shapes for simulating the deformation of different shapes. To illustrate this

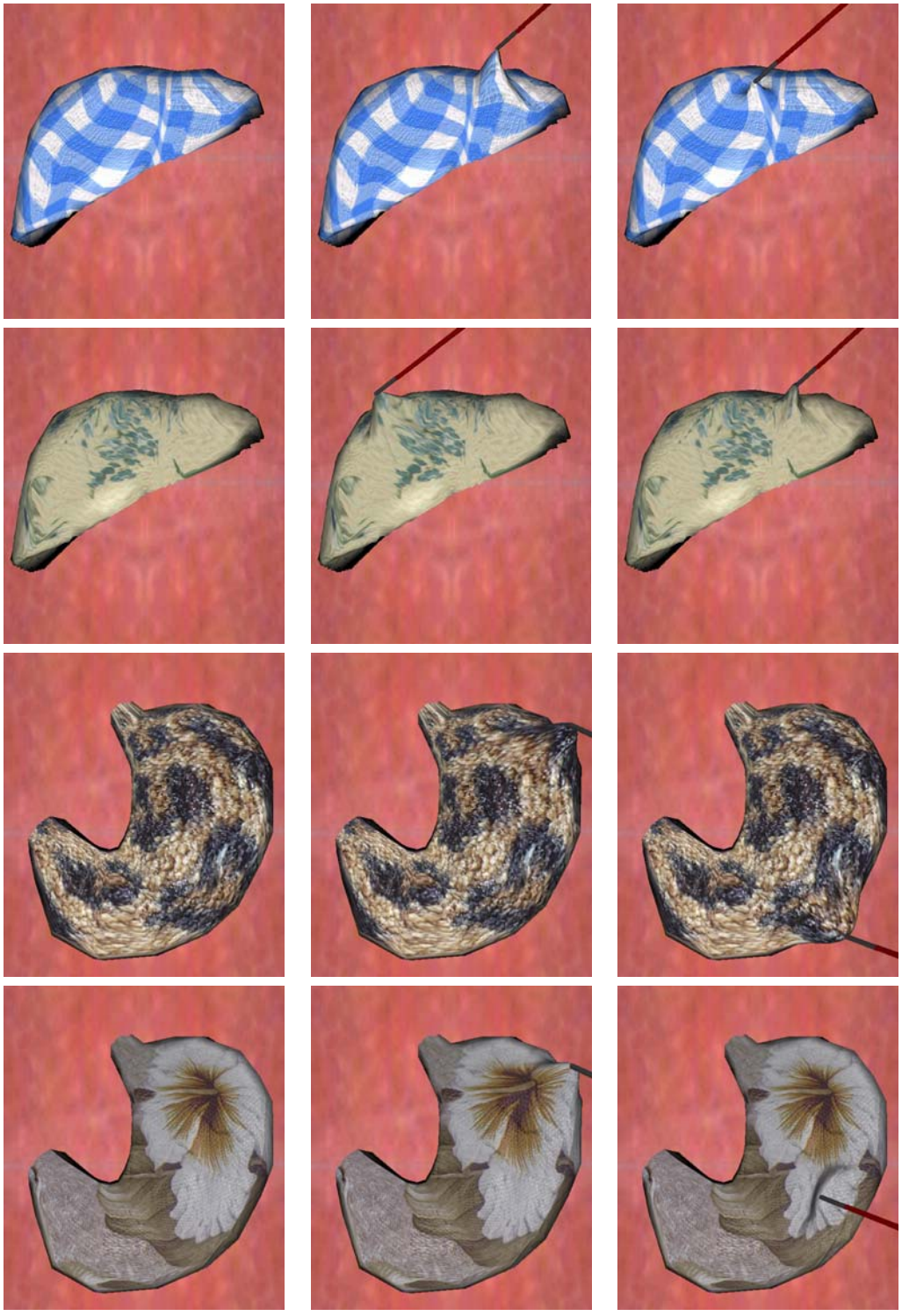


Figure 7.11: The deformation of various textiles wrapped on the liver and stomach.

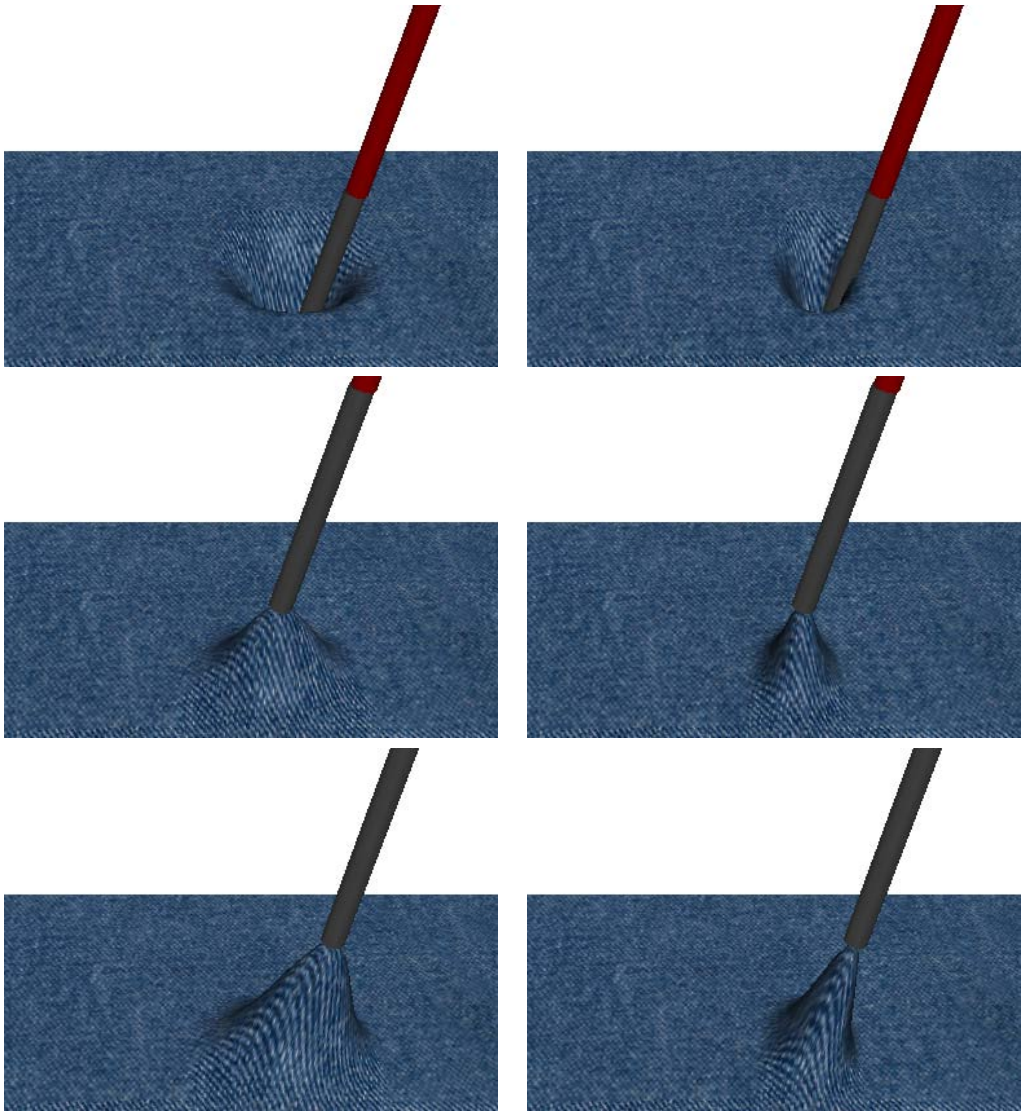


Figure 7.12: Variable kernel for deformation of a denim textile on a flat surface.

further, in Figure 7.12, we map the denim texture on a flat surface. To interact with the fabric, we select a pre-defined deformation kernel to simulate the deformation. The deformation kernel has a circular base as shown in the left half of Figure 7.12. Instead of using a circular base, we can also use an elliptic kernel as shown in the right half of Figure 7.12. Figure 7.13 shows the same pinch effect on a different fabric.

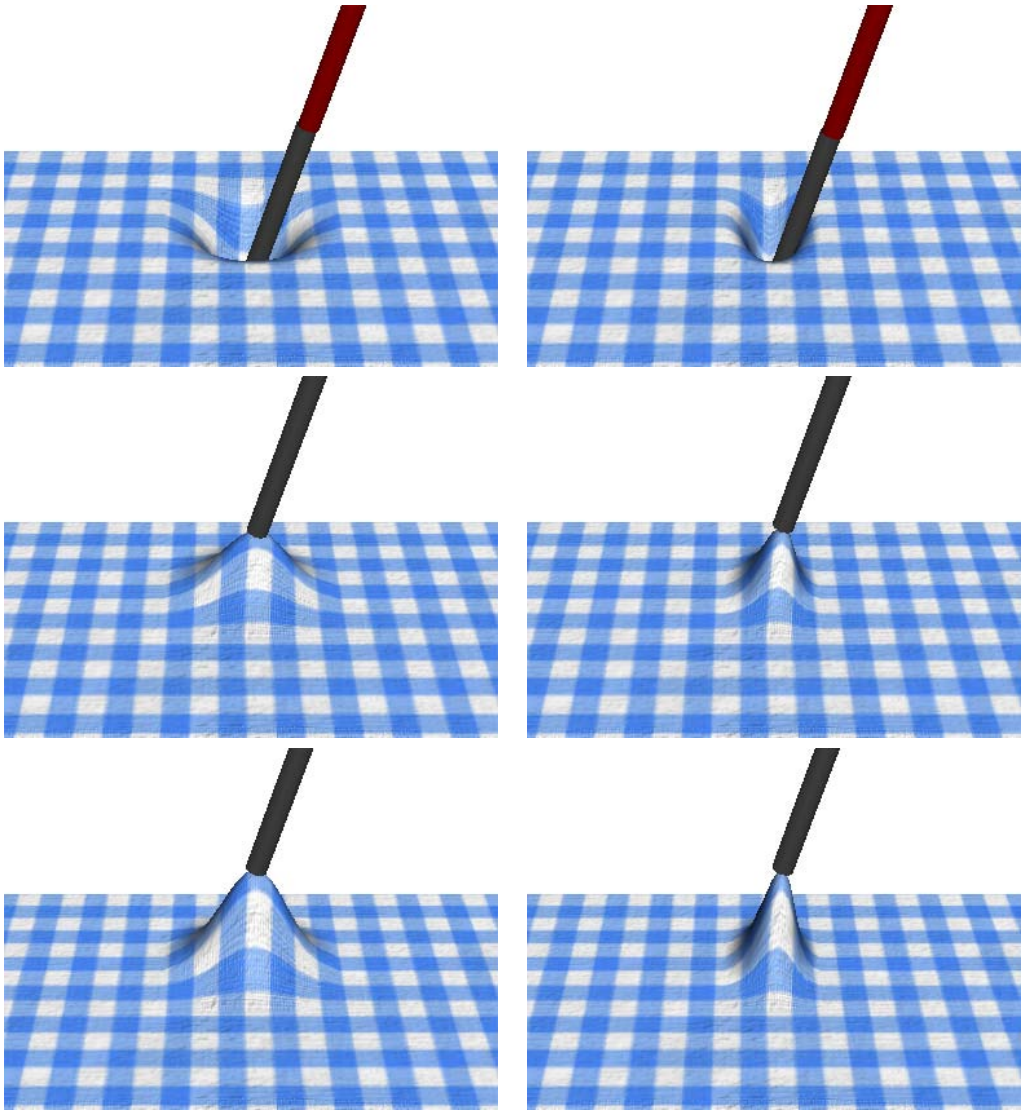


Figure 7.13: Variable kernel for deformation of plaid textile on a flat surface.

7.7.3 An Upholstery Example - Textile Draping on Chair

Figure 7.14 shows a textile wrapped onto a deformable surface. The users are able to push and pull the fabric on the surface of the chair model interactively. This can be useful for online shopping, where buyers can interactively touch a chair or a piece of fabric.

Figure 7.14: The deformation of textile wrapped on the deformable seat.

7.8 Summary

In this chapter two different case studies have been presented which successfully utilize our proposed DGM representation. In the case of organ deformation below were demonstrated:

1. realistic organ visualization,
2. real-time organ visualization,
3. deformation in real-time with DGM,
4. analysis of the deformation, and
5. comparison of deformation with related work by De et al. [31],

In the second case study these were demonstrated

1. realistic and real-time deformation of textile,
2. analysis of the shape of deformation, and
3. several textile examples including the draping of textile on a chair.

The main emphasis in this thesis is on visualization, realistic rendering and real-time interactive deformation. However, it is straightforward to integrate our real-time deformation to any haptic device for haptic feedback. To add haptical realism, a haptic device, such as the PHANTOM device developed by SensAble Technologies, can be used to control the movement of the virtual tool, and to convey the haptic feedback to the user. Our deformation method can support an update rate that is sufficient to provide haptic feedback (1kHZ for PHANTOM).

Chapter 8

Conclusion and Future Work

We have introduced a parameterized representation of 3D organ meshes for the simulation of laparoscopic surgery. Arbitrary input virtual organ meshes are parameterized and resampled into regular high resolution models. The high resolution models increase the visual quality of the virtual organ. The parameterized mesh is helpful for the simulation in many aspects. Firstly, it is useful for collision detection. Secondly, we introduced a free-form deformation approach for the parameterized mesh to simulate the interaction between the tool and the organ to achieve an update rate fast enough to provide haptic feedback. The shape of the deformation is easily controlled by two parameters. Lastly, we have also developed a procedure to handle the contact between the organ and another object.

8.1 Contributions

The main contributions of this thesis are:

1. **Deformable Geometry Map Representation:** We have proposed a new Deformable Geometry Map (DGM) representation. Arbitrary input surface meshes are parameterized and resampled into regular high resolution models. The high

resolution models increase the visual quality. The parameterized mesh can be used as a deformable model. The DGM representation also helps to handle collision detection and contacts between objects.

2. **Deformation of Geometry Maps:** We introduced a free-form deformation approach for the parameterized mesh to simulate the interactions between the tool and the organ to achieve an update rate fast enough to provide haptic feedback. The parametric deformable shape representation described in this work helps to approximate the local deformation. The advantage of the local deformation now includes the ability to interact with the 3D shape model using the 2D parameter space instead of the 3D mesh. As a result, the deformation computation need not be performed on the entire mesh.

3. **Interaction with the Deformable Mesh:** We also proposed a procedure to handle the collision between tool and surface as well as the contact between the organ and another object.

4. **Texture Deformation:** The appearance of the 3D mesh is accomplished using appropriate textures. Because of the parameterization, the texture mapping is implicit. In our examples, the texture mapping works well for the deformable surfaces.

5. **DGM for Surgery Simulation:** We have demonstrated the parameterized representation of 3D organ meshes for the simulation of laparoscopic surgery. The high resolution models increase the visual quality of the virtual organ. The parameterized mesh is helpful for the simulation in a few aspects: collision detection, free-form deformation, simulation of the interaction between the tool and organ, and an update rate fast enough to provide haptic feedback. The DGM representation also helps to handle the contact between the organ and another object.

6. **DGM for Textile Deformation:** We have demonstrated the parameterized representation of 3D meshes for the simulation of deformation of textiles. The high resolution models increase the visual quality of the virtual textiles. The parameterized mesh is helpful for the simulation as shown in our examples.

8.2 Discussions

In this section commonly raised questions and how this thesis addresses these concerns are discussed. The solutions and contributions presented in this thesis are also highlighted.

8.2.1 Imaging vs Deformable Object Modeling

How are the imaging modalities such as MRI, CT, US related to deformable modeling?

Imaging modalities such as MRI, CT, US have enabled geometry, appearance and texture of deformable tissues to be visualized.

- Organ modeling from imaging (CT Data): Deformable tissue also referred to as soft tissue can be visualized from data captured using techniques described in Delingette and Ayache [36]. The surface of the an organ such as the liver is extracted, and surface visualization techniques such as isosurface rendering are used. The paper also described interactions with such surfaces. Thus the geometry is represented as triangles in an isosurface. The surface appearance is visualized using surface rendering and texture mapping. For deformation, the triangle mesh is transformed to a new shape to visualize the behavior. This has been described in detail in Section 2.2.1.

- Organ Modeling from preoperative and intra-operative images: As explained in

Section 2.2.3, preoperative and intra-operative images have also been modeled to help surgeons in surgical decision making. The main emphasis of Warfield et al. [89] is on capturing intra-operative deformations of anatomical structures so as to help in image-guided therapy.

- Modeling of deformation of organs from imaging data: Recently, such imaging techniques have also been used to capture the deformation of soft tissue over a period of time. Deformable tissue can be visualized with four-dimensional computed tomography (4DCT) technology [86, 49, 72]. 4DCT dataset is acquired from CT data that is oversampled at each slice during several CT tube rotations. Multiple images are then reconstructed for each slice and distributed over the acquisition time. Each of these images represents a different anatomical state. The tissue motion and deformation during the acquisition time is thus captured.

- Deformable modeling in this thesis: Our work does not deal with the imaging modalities. It also does not deal with reconstructing the initial 3D shapes. Instead our approach assumes a well defined organ is available in 3D shape and it is used as an input into our system. The output of our system is an instrumented 3D model that is capable of deformation. We also introduced the approach of deformation for such instrumented models. Since our approach is based on free-form deformation, our system is capable of simulating a wide range of object properties as shown in Table 7.1 and Table 7.2. Hence the exact validation to conform to a specific object can be defined by the end user whether it is for a surgery tool or for an interactive textile tool.

8.2.2 Medical Visualization and Virtual Surgery

How is traditional medical visualization different from virtual surgery?

Traditional visualization applications such as virtual colonoscopy visualizes human anatomy through image acquisition, surfaces reconstruction, and display of structures. There is no prediction of surgical outcome as the information provided to the user is used mainly for diagnosis. As such, it falls under category 1 (first generation) as defined in Figure 2.8. However, it can be used as a preoperative procedure for surgery planning. It is utilized to visualize and locate the position of polyps by navigating in the colon, which helps to plan the surgery beforehand. Examples of simulators that can predict the outcome of an operation would be Computer-Aided Plastic Surgery used by Pieper et al. [68], and the cleft clip surgery simulation by Sheppard [76].

8.2.3 Physical Behavior of Tissues

How are the material (i.e. soft tissue) properties reflected faithfully in the computation? How accurate, and therefore how useful, can the results be for haptic feedback?

There is no optimum deformable model that combines all the requirements for surgery simulation. In computer graphics we tend to use computational models to mimic the real objects while achieving a certain level of realism and interactivity. One approach is to have faithful simulation of the physics. Our approach is to use heuristic models to mimic the physical behavior of tissues. The behavior currently shown in this thesis is an approximate deformation that creates the appearance of the real tissue deformation (see Figure 7.4 and Figure 7.5). Like the spring constants in mass-spring models, the parameters β and l do not have direct quantitative relation to the material property.

- However, if the end user carefully tunes the parameters, a wide range of material properties can be approximated, as shown in Table 7.1 and Table 7.2.
- Depending on the tissue type, one can constrain which parameter values to be

used and the region affected by the deformation.

- If these parameters can be pre-computed in a computationally expensive and physically based simulation, the values can be used as parameters to control the deformation and provide better realism.

8.2.4 Novelty of DGM

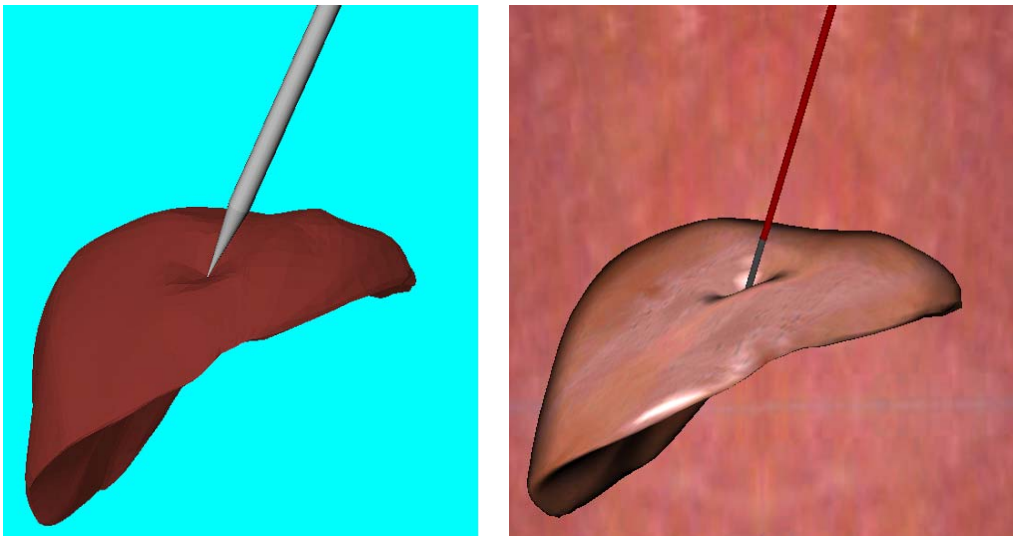
The parameterization of surface mesh has traditionally been used by researchers for texture mapping and remeshing. We have not seen any research reported in the literature that performs *deformation* on parameterized mesh. This thesis demonstrates how a parameterized mesh can support deformation.

8.2.5 Qualitative Comparison of DGM with Related Work

The organ deformation result from our experiments is compared with the result of previous work by De et al. [31] in Figure 8.1. The example in [31] used finite spheres for real-time simulation and display of deformation. In our case, we use a parametric free form deformation, and a bump mapped specular shading for visualization. The comparison shows that, although our method is simple, realistic result can be achieved. Moreover, our visual rendering effect is more realistic.

8.2.6 Quantitative Comparison of DGM with Existing Techniques

Quantitative comparisons are normally done using the number of triangles in the organ, with the time taken for deformation, visual rendering, and haptic rendering presented. However, it is not an easy task to compare DGM with other existing techniques quantitatively since only limited data has been published on the performance of each tech-



(a) (b)

Figure 8.1: The comparison of the organ deformation results. (a) Results from De et al. [31]. (b) Our method.

nique. Moreover, it is difficult to have comparison between different techniques with different hardware configurations. Table 8.1 lists the time results of various deformable models. The comparison shows that DGM has considerably high speed in relation to existing approaches.

8.2.7 Limitation of DGM

The models presented in Chapter 7 each have 12,800 triangles. Based on the time results listed in Table 7.3, the limit for visual rendering with DGM is about 80,000 triangles. The limit for haptic rendering is about 1,800 triangles for stable and consistent haptic feedback. Force extrapolation techniques are usually employed to increase the haptic rate [31, 36] and hence can increase the limit for haptic rendering. Moreover, with higher end hardware configurations, this limit can be higher.

DGM is most suitable to model objects with big area of deformable surface which is relatively flat. For surgical simulation, organs that are suitable for deformable geometry

Deformation Techniques	Total Size	Deformed Size	Visual Frame Rate	Haptic Frame Rate
DGM	12800 triangles	200 triangles	30 Hz	4728 Hz
		800 triangles	30 Hz	1155 Hz
		1800 triangles	30 Hz	932 Hz
The Method of Finite Spheres (De et al. [8][31])	N.A.	34 Sphere nodes	Real-time	200 Hz
Hybrid Model (Delingette and Ayache [36])	1537 vertices and 7039 tetrahedra	1537 vertices and 7039 tetrahedra	Real-time	—
Mass-spring Model (Brown et al. [13] [14][15])	8000 nodes	8000 nodes	30 Hz	—
Sphere-Filled Organ Model (Suzuki et al. [81])	512 spheres	N.A.	25-30 Hz	—
	1212 spheres	N.A.	12-15 Hz	—
FEM with Parallel Computing (Székely et al. [83])	21 elements	21 elements	Real-time	500 Hz
Fast FE Model (Berkley et al. [10])	863 nodes	724 nodes	30 Hz	Above 1000 Hz

Table 8.1: Quantitative comparison of DGM to existing techniques.

map include liver, stomach, kidney, heart, etc. Tube and duct shaped organs, such as intestine and vessel are better modeled with mass spring models [13, 14, 15] or the approaches introduced by Raghupathi et al. [71].

8.3 Future Work

The work reported in this thesis can be extended in three distinct directions.

8.3.1 Improvements in Geometry performance

- **Automatic Parameterization of Geometry:** We have proposed a Deformable Geometry Map for effective and efficient interactive deformation. One limita-

tion of current method is that the parameterization process is conducted semi-automatically. Even though the complexity of the input mesh is sufficient for the examples used in this thesis, such as the stomach, liver, and textile meshes, it is a limitation for more complex shapes. In addition, the parameterization quality in terms of least area distortion cannot be guaranteed. In future work, some automatic processes are planned to be adopted for parameterization, such as the geodesics-based approach proposed by Lee et al. [52].

- **Multi-layered Map on GPU:** In DGM, an arbitrary 3D input mesh is parameterized and resampled into a regular 2D parameterized model. The parameterization consists of seven layers of flat maps: geometry map, normal map, texture map, bump map, collision map, deformation map, and contact map as shown in Figure 8.2. These layers of parameterization have the features of compact and implicit data representation. Future work will focus on the realization of this multi-layered geometry map in hardware using GPUs to accelerate the computational and rendering performance.

8.3.2 Improvements in Physically Based Realism

- **Physically based Deformation of Geometry Maps:** More realistic deformation can be produced with the physically based deformation methods explained in Chapter 3 and Chapter 4. A possible direction of future work is to improve the deformation with more realistic deformation methods, such as mass-spring model and finite element method (FEM). The deformation can be calculated similarly in the 2D parameter space, then transferred back to the 3D mesh. Since the vertices are regularly spaced in the parameter space, no additional remeshing is required for mass-spring model or FEM.

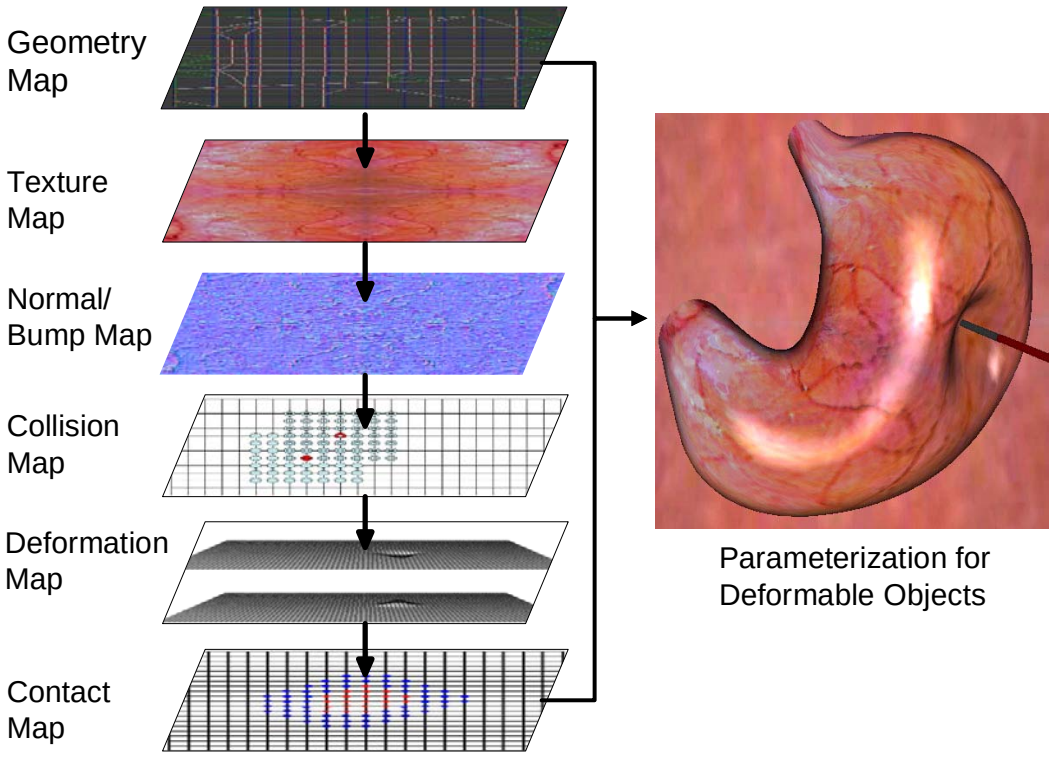


Figure 8.2: Multi-layer Deformable Geometry Maps.

- **Cloth Simulation with Parameterized Geometry Maps:** The parameterized mesh can be view as a model of cloth. Using this model, one can also solve for the deformation of the cloth using an implicit method of integration. This approach can also take into account deformation which can occur at different locations of the mesh (as highlighted for example in Figure 6.6). Parameterization leads to lots of new possibilities. With parameterized mesh, now it is possible to incorporate any algorithm designed for cloth simulation to be used in surgery simulation. The benefit will be interactive surgery simulation with physically based models. One possible extension will be to adapt the cloth simulation using implicit method of integration [6] for surgery simulation.

- **Validation of Material Property:** The parameters used to mimic the material property of human tissues and textiles should be compared with real soft tissues

or textiles samples and validated.

8.3.3 Improvements in Haptics and Interaction

- **Integration of Haptic Interaction:** Because of the efficient DGM representation and the interactive real-time deformation, combined with the realistic cinematic quality visualization, the technique of DGM has the potential for haptic applications. To add haptical realism, a haptic device, such as the PHANTOM device developed by SensAble Technologies, can be used to control the movement of the virtual surgery tool, and to convey the haptic feedback to the user. Our deformation method can support an update rate high enough to provide haptic feedback (1kHZ for PHANTOM).

- **Cutting and Suturing in Surgery Simulation:** More types of tool-tissue interactions can be explored using DGM in the future work. For example, cutting of an organ can be implemented with a procedure similar to the incision procedure proposed by Suzuki et al. [81]. Since the geometry map is resampled at a high density, the incision can be assumed to be always along the edges of the reconstructed triangle mesh. Incision planes are constructed by separating the vertex along the incision and adding incision point based on the incision depth. Another type of interaction in surgery simulation is suturing. A suturing process with DGM can be implemented based on the process introduced by Berkley et al. [10].

Acronyms

AABB	Axis Aligned Bounding Box
BFS	Breadth-First Search
CT	Computed Tomography
DGM	Deformable Geometry Map
DOF	Degree of Freedom
FE	Finite Element
FEM	Finite Element Method(s)
FFD	Free Form Deformation
FTM	Force Transmittal Mechanism
GPU	Graphics Processing Unit
HS	Height Spans
LBIE	Local Boundary Integral Equation
LOD	Level of Detail
MFEM	Meshless Finite Element Method
MLPG	Meshless Local Petrov-Galerkin
MRI	Magnetic Resonance Imaging
NURBS	Non-Uniform Rational B-Spline
OBBTree	Oriented Bounding Box Tree
PAFF	Point-Associated Finite Field
PCMFS	Point Collocation-Based Method of Finite Spheres
PDE	Partial Differential Equation
PET	Positron Emission Tomography
SA	Simulated Annealing
VR	Virtual Reality

Author's Publications

Journal Publications:

1. Q. Liu, E.C. Prakash, and M.A. Srinivasan. Interactive Deformable Geometry Maps: Efficient Modeling for Interactive Deformation of Non-Rigid 3D Objects. *The Visual Computer*, 23(2):119-131, 2007.

2. Q. Liu and E.C. Prakash. Cyber Surgery: Parameterized Mesh for Multi-modal Surgery Simulation. *Journal of Visualization*, 9(4):373-380 2006.

3. Q. Liu, E.C. Prakash and A. Anwar. Flat Maps: A Multi-Layer Parameterization for Surgery Simulation. *International Journal of Shape Modeling*, submitted, 2006.

4. Q. Liu and E. C. Prakash. Enhanced Graphical Modeling of Deformable Organs for Surgery Simulation. *Machine Graphic & Vision*, under revision, 2006.

Conference Publications:

1. Q. Liu and E. C. Prakash. Flat Maps: A Multi-Layer Parameterization for Surgery Simulation. In *Medicine Meets Virtual Reality (MMVR 14)*, pages 340-342, Long Beach, CA, USA, January 2006.

2. E. C. Prakash, Q. Liu, A. Anwar, K. C. Wee, N. Ravikumar, and K. C. Teh.

A Practical Surgery Simulator. In *Proc. of Int'l Conf. on Information and Automation (ICIA 2005)*, Colombo, Sri Lanka, December 2005.

3. Q. Liu and E. C. Prakash. Deformable Geometry Maps. In *Proc. of HAPTEX'05 - VR Workshop on Haptic and Tactile Perception of Deformable Objects*, pages 64-71, Hannover, Germany, December 2005.

4. Q. Liu and E. C. Prakash. Cyber Surgery: Parameterized Mesh for Multi-modal Surgery Simulation. In *Proc. of 6th Pacific Rim Conference on Multimedia (PCM 2005)*, pages 934-945, Jeju Island, Korea, November 2005.

5. Q. Liu and E. C. Prakash. Surgery Simulation with Surface Parameterization. In *Proc. of 18th Int'l Workshop on Computer Animation and Social Agents (CASA 2005)*, pages 185-186, Hong Kong, October 2005.

6. A. Anwar and Q. Liu. Simple Realistic Simulation for a Surgery Game. In *Proc. of Cyber Games: Int'l Conf. on Games Research and Development*, pages 52-56, Singapore, March 2005.

7. Q. Liu and E. C. Prakash. An Investigation of the Meshless Finite Element Method (MFEM) for Models of Deformable Objects. In *Proc. of 7th Int'l Workshop on Advanced Image Technology (IWAIT 2004)*, pages 335-339, Singapore, January 2004.

8. Q. Liu and E. C. Prakash. The Parameterization of Joint Rotation with the Unit Quaternion. In *Proc. of 7th Int'l Conf. on Digital Image Computing: Techniques and Applications (DICTA 2003)*, pages 409-417, Sydney, Australia, December 2003.

Bibliography

[1] *LapSim Demo Video*. Surgical Science, www.surgical-science.com.

[2] Pierre Alliez, Mark Meyer, and Mathieu Desbrun. Interactive geometry remeshing. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 347–354, New York, NY, USA, 2002. ACM Press.

[3] S. N. Atluri, H. G. Kim, and J. Y. Cho. A critical assessment of the truly Meshless Local Petrov-Galerkin (MLPG), and Local Boundary Integral Equation (LBIE) methods. *Computational Mechanics*, 24(5):348–372, 1999.

[4] S. N. Atluri and S. Shen. The Meshless Local Petrov-Galerkin (MLPG) method: a simple & less-costly alternaltive to the finite elment and boundary element methods. *CMES: Computer Modeling in Engineering & Sciences*, 3(1):11–51, 2002.

[5] S. N. Atluri and T. Zhu. A new Meshless Local Petrov-Galerkin (MLPG) approach in computational mechanics. *Computational Mechanics*, 22(2):117–127, 1998.

[6] D. Baraff and A. Witkin. Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1998. ACM Press.

[7] D. Bartz. Virtual endoscopy in research and clinical practice. *Computer Graphics Forum*, 24(1):111–126, 2005.

[8] C. Basdogan, S. De, J. Kim, M. Manivannan, H. Kim, and M. A. Srinivasan. Haptics in minimally invasive surgical simulation and training. *IEEE Computer Graphics & Applications*, 24(2):56–64, 2004.

[9] C. Basdogan and M. A. Srinivasan. Haptic rendering in virtual environments. In *Virtual Environment Handbook, Ed: K. M. Stanney, Lawrence Erlbaum Associates, Ch. 6*, pages 171–134, 2002.

[10] J. Berkley, G. Turkiyyah, D. Berg, M. A. Ganter, and S. Weghorst. Real-time finite element modeling for surgery simulation: An application to virtual suturing. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):314–325, May/June 2004.

[11] S. Blackett, D. Bullivant, and P. Hunter. Free form deformation for biomedical applications. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 88, Washington, DC, USA, 2003. IEEE Computer Society.

[12] Joel Brown, Jean-Claude Latombe, and Kevin Montgomery. Real-time knot-tying simulation. *The Visual Computer*, 20(2-3):165–179, 2004.

[13] Joel Brown, Kevin Montgomery, Jean-Claude Latombe, and Michael Stephanides. A microsurgery simulation system. In *Proceedings of Medical Image Computing and Computer-Assisted Intervention - MICCAI 2001*, pages 137–144, October 2001.

[14] Joel Brown, Stephen Sorkin, Cynthia Bruyns, Jean-Claude Latombe, Kevin Montgomery, and Michael Stephanides. Real-time simulation of deformable objects: Tools and application. In *CA '01: Proceedings of the Computer Animation*, pages 228–236, November 2001.

[15] Joel Brown, Stephen Sorkin, Jean-Claude Latombe, Kevin Montgomery, and Michael Stephanides. Algorithmic tool for real-time microsurgery simulation. *Medical Image Analysis*, 6:289–300, 2002.

[16] GC Burdea. Haptics issues in virtual environments. *Computer Graphics International, 2000. Proceedings*, pages 295–302, 2000.

[17] H. Çakmak, H. Maass, and U. Kühnapfel. VSOne, a virtual reality simulator for laparoscopic surgery. *Minimally Invasive Therapy and Allied Technologies*, 14(3):134–144, June 2005.

[18] B.-Y. Chen, Y. Ono, H. Johan, M. Ishii, T. Nishita, and J. Feng. 3D model deformation along a parametric surface. In *Proceedings of IASTED Visualization, Imaging and Image Processing (VIIP) 2002*, pages 282–287, September 2002.

[19] M. Chen, C. Correa, S. Islam, MW Jones, PY Shen, D. Silver, SJ Walton, and PJ Willis. Deforming and animating discretely sampled object representations. *State of the Art Report, Proceedings of Eurographics 2005*, pages 113–140.

[20] Kup-Sze Choi, Hanqiu Sun, and Pheng-Ann Heng. Interactive deformation of soft tissues with haptic feedback for medical learning. *IEEE Transactions on Information Technology in Biomedicine*, 7(4):358–363, 2003.

[21] Kup-Sze Choi, Hanqiu Sun, and Pheng-Ann Heng. An efficient and scalable deformable model for virtual reality-based medical applications. *Artificial Intelligence in Medicine*, 32(1):51–69, September 2004.

[22] Adrian James Chung, Fani Deligianni, Pallav Shah, Athol Wells, and Guang-Zhong Yang. Enhancement of visual realism with BRDF for patient specific bronchoscopy simulation. In *MICCAI (2)*, volume 3217 of *Lecture Notes in Computer Science*, pages 486–493, 2004.

[23] J. D. Cohen, M. C. Lin, D. Manocha, and M. K. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pages 189–196, 1995.

[24] S. Coquillart. Extended free-form deformation: a sculpturing tool for 3D geometric modeling. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 187–196, New York, NY, USA, 1990. ACM Press.

[25] S. Coquillart and P. Jancène. Animated free-form deformation: an interactive animation technique. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 23–26, New York, NY, USA, 1991. ACM Press.

[26] Steven A. Cover, Norberto F. Ezquerra, James F. O'Brien, Richard Rowe, Thomas Gadacz, and Ellen Palm. Interactively deformable models for surgery simulation. *IEEE Computer Graphics & Applications*, 13(6):68–75, 1993.

[27] D. Zeltzer D. T. Chen. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 89–98, New York, NY, USA, 1992. ACM Press.

[28] A. Barr D. Terzopoulos, J. Platt and K. Fleischer. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, New York, NY, USA, 1987. ACM Press.

[29] S. De and J. Bathe. The method of finite spheres. *Computational Mechanics*, 25:329–345, 2000.

[30] S. De and K. J. Bathe. Towards an efficient meshless computational technique: the method of finite spheres. *Engineering Computations*, 18(1/2):170–192, 2001.

[31] S. De, J. Kim, and M. A. Srinivasan. A meshless numerical technique for physically based real time medical simulations. In *Medicine Meets Virtual Reality 2001 (Proceedings of MMVR 2001)*, pages 113–118, 2001.

[32] S. De, M. Manivannan, J. Kim, M. A. Srinivasan, and D. Rattner. Multimodal simulation of laparoscopic heller myotomy using a meshless technique. In *Medicine Meets Virtual Reality 02/10 (Proceedings of MMVR 2002)*, pages 127–132, 2002.

[33] Suvranu De, Jung Kim, Yi-Je Lim, and Mandayam A. Srinivasan. The point collocation-based method of finite spheres (PCMFS) for real time surgery simulation. *Computers and Structures*, 83:1515–1525, 2005.

[34] Suvranu De, Yi-Je Lim, Muniyandi Manivannan, and Mandayam A. Srinivasan. Physically realistic virtual surgery using the Point-Associated Finite Field (PAFF) approach. *Presence: Teleoperators & Virtual Environments*, 15(3):294–308, June 2006.

[35] Fani Deligianni, Adrian James Chung, and Guang-Zhong Yang. Patient-specific bronchoscope simulation with PQ based 2D/3D registration. *Computer Aided Surgery*, 9(5):215–226, 2004.

[36] Herve Delingette and Nicholas Ayache. Hepatic surgery simulation. *Communications ACM*, 48(2):31–36, 2005.

[37] F. Dong, G. J. Clapworthy, H. Lin, and M. Krokos. Volumetric hatching: Illustration of medical volume data using line strokes. *IEEE Computer Graphics & Applications*, 23(4):44–52, 2003.

[38] H. Du and H. Qin. Dynamic PDE-based surface design using geometric and physical constraints. *Graphical Models*, 67(1):43–71, 2005.

[39] M.S. Floater and K. Hormann. Surface Parameterization: a Tutorial and Survey. *Advances In Multiresolution For Geometric Modelling*, pages 157–186, 2005.

[40] S. F. Gibson and B. Mirtich. *A Survey of Deformable Modeling in Computer Graphics*. Technical Report TR-97-19, Mitsubishi Electric Research Laboratories, Ambridge, MA, USA, 1997.

[41] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180, New York, NY, USA, 1996. ACM Press.

[42] C. M. Grimm. Parameterization using manifolds. *International Journal of Shape Modeling*, 10(1):51–81, 2004.

[43] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 62–67, 2003.

[44] X. Gu, S. J. Gortler, and H. Hoppe. Gometry images. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 355–361, New York, NY, USA, 2002. ACM Press.

[45] P.S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, 1986.

[46] G. Hirota, S. Fisher, and A. State. An improved finite-element contact model for anatomical simulations. *The Visual Computer*, 19(5):291–309, 2003.

[47] Richard Holbrey, Andrew Bulpitt, Ken Brodlie, Mark Walkley, and Julian Scott. A model for virtual suturing in vascular surgery. In *TPCG '04: Proceedings of the Theory and Practice of Computer Graphics 2004 (TPCG'04)*, pages 50–58, Washington, DC, USA, 2004. IEEE Computer Society.

[48] A.E. Kaufman, S. Lakare, K. Kreeger, and I. Bitter. Virtual colonoscopy. *Communications of ACM*, 48(2):37–41, 2005.

[49] P. Keall. 4-dimensional computed tomography imaging and treatment planning. *Semin Radiat Oncol*, 14(1):81–90, 2004.

[50] I. Kim, S. De, and M.A. Srinivasan. Physically Based Hybrid Approach in Real Time Surgical Simulation With Force Feedback. *Medicine Meets Virtual Reality 11: NextMed: Health Horizon*, pages 158–164, 2003.

[51] M. LeDuc, S. Payandeh, and J. Dill. Toward modeling of a suturing task. In *Graphics Interface'03 Conference*, pages 273–279, June 2003.

[52] Haeyoung Lee, Yiying Tong, and Mathieu Desbrun. Geodesics-based one-to-one parameterization of 3D triangle meshes. *IEEE Multimedia*, 12(1):27–33, 2005.

[53] Yi-Je Lim and Suvranu De. On the use of meshfree methods and a geometry based surgical cutting algorithm in multimodal medical simulations. In *Proceedings of IEEE internatioal Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'04)*, pages 295–301, March 2004.

[54] Jean-Christophe Lombardo, Marie-Paule Cani, and Fabrice Neyret. Real-time collision detection for virtual surgery. In *CA '99: Proceedings of the Computer Animation*, pages 82–91, Washington, DC, USA, 1999. IEEE Computer Society.

[55] Q. Luo and J. Xiao. Modeling complex contacts involving deformable objects for haptic and graphic rendering. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[56] M. Mahvash and V. Hayward. High-fidelity haptic synthesis of contact with deformable bodies. *IEEE Computer Graphics and Applications*, 24(2):48–55, March-April 2004.

[57] M. Mahvash, V. Hayward, and J. E. Lloyd. Haptic rendering of tool contact. In *Proceedings Eurohaptics 2002*, pages 110–115, 2002.

[58] R. Mayoral, N. G. Tsagarakis, M. Petrone, G. J. Clapworthy, D. G. Caldwell, and C. Zannoni. Integration of haptic and visual modalities for a total hip replacement planning system. In *MediVis '05: Proceedings of IEEE Medical Information Visualisation*, pages 30–35. IEEE Computer Society Press, 2005.

[59] U. Meier, O. López, C. Monserrat, M.C. Juan, and M. Alcaniz. Real-time deformable models for surgery simulation: a survey. *Computer Methods and Programs in Biomedicine*, 77(3):183–197, March 2005.

[60] Wolfgang Müller-Wittig, Ulrich Bockholt, Jose Luis Los Arcos, and Gerrit Voss. Enhanced training environment for minimally invasive surgery. In *WETICE 2001: Proceedings of 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 269–272. IEEE Computer Society, june 2001.

[61] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[62] Fabrice Neyret, Raphael Heiss, and Franck Sénégas. Realistic rendering of an organ surface in real-time for laparoscopic surgery simulation. *The Visual Computer*, 18(3):135–149, may 2002.

[63] G. M. Nielson, P. Brunet, M. Gross, H. Hagen, and S. V. Klimenko. Research issues in data modeling for scientific visualization. *IEEE Computer Graphics & Applications*, 14(2):70–73, 1994.

[64] K. Onoue and T. Nishita. An interactive deformation system for granular material. *Computer Graphics Forum*, 24(1):51–60, 2005.

[65] Shahram Payandeh, John Dill, and Jian Zhang. A study of level-of-detail in haptic rendering. *ACM Transactions on Applied Perception*, 2(1):15–34, January 2005.

[66] A. Petersik, B. Pflesser, U. Tiede, K. H. Höhne, and R. Leuwer. Realistic haptic interaction in volume sculpting for surgery simulation. In *Surgery Simulation and Soft Tissue Modeling, Proceedings of IS4TM 2003*, pages 194–202, 2003.

[67] A. Petersik, B. Pflesser, U. Tiede, K.H. Höhne, and R. Leuwer. Realistic haptic volume interaction for petrous bone surgery simulation. In *CARS 2002: Proceedings of Computer Assisted Radiology and Surgery*, pages 252–257, 2002.

[68] SD Pieper, DR Laub Jr, and JM Rosen. A finite-element facial model for simulating plastic surgery. *Plast Reconstr Surg*, 96(5):1100–1105, 1995.

[69] E. C. Prakash, Q. Liu, A. Anwar, K. C. Wee, N. Ravikumar, and K. C. Teh. A practical surgery simulator. In *ICIA 2005: Proceedings of Int'l Conf. on Information and Automation*, December 2005.

[70] E. C. Prakash, M. Manivannan, and M. A. Srinivasan. A new approach for the synthesis of glistening effect in deformable anatomical objects displayed with haptic

feedback. In *Medicine Meets Virtual Reality 02/10 (Proceedings of MMVR 2002)*, pages 369–375, 2002.

[71] Laks Raghupathi, Laurent Grisoni, Francois Faure, Damien Marchal, Marie-Paule Cani, and Christophe Chaillou. An intestinal surgery simulator: Real-time collision processing and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):708–718, 2004.

[72] E. Rietzel, T. Pan, and GT Chen. Four-dimensional computed tomography: image formation and clinical protocol. *Med Phys*, 32(4):874–889, 2005.

[73] A. P. Santhanam, S. N. Pattanaik, J. P. Rolland, C. Imielinska, and J. Norfleet. Physiologically-based modeling and visualization of deformable lungs. In *11th Pacific Conference on Computer Graphics and Applications (PG'03)*, pages 507–511, 2003.

[74] S. Schein and G. Elber. Placement of deformable objects. *Computer Graphics Forum*, 23(4):727–739, 2004.

[75] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160, New York, NY, USA, 1986. ACM Press.

[76] L.M. Sheppard. Virtual surgery brings back smiles. *IEEE Computer Graphics & Applications*, 25(1):6–11, 2005.

[77] D. Sorid and S. Moore. The virtual surgeon. *IEEE Spectrum*, pages 26–31, July 2000.

[78] A. Sourin, O. Sourina, and T.S. Howe. Virtual orthopedic surgery training. *IEEE Computer Graphics & Applications*, 20(3):6–9, 2000.

[79] O. Sourina, A. Sourin, and T.S. Howe. Orthopedic surgery training on personal computer. *International Journal of Information Technology*, 6(1):16–29, May 2000.

[80] M. A. Srinivasan and C. Basdogan. Haptics in virtual environments: taxonomy, research status, and challenges. *Computers and Graphics*, 21(4):393–404, 1997.

[81] S. Suzuki, N. Suzuki, A. Hattori, A. Uchiyama, and S. Kobayashi. Sphere-filled organ model for virtual surgery system. *IEEE Transactions on Medical Imaging*, 23(6):714–722, 2004.

[82] Nikolai Svakhine, David S. Ebert, and Don Stredney. Illustration motifs for effective medical volume illustration. *IEEE Computer Graphics & Applications*, 25(3):31–39, 2005.

[83] G. Székely, Ch. Brechbuhler, R. Hutter, A. Rhomberg, N. Ironmonger, and P. Schmid. Modelling of soft tissue deformation for laparoscopic surgery simulation. *Medical Image Analysis*, 4(1):57–66, 2000.

[84] N. M. Thalmann and F.E. Wolter. Workshop on haptic and tactile perception of deformable objects. In *Proceedings of HAPTEX 2005*, pages 1–86, 2005.

[85] Gino van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, 2(4):1–13, 1997.

[86] SS Vedam, PJ Keall, VR Kini, H. Mostafavi, HP Shukla, and R. Mohan. Acquiring a four-dimensional computed tomography dataset using an external respiratory signal. *Physics in Medicine and Biology*, 48(1):45–62, 2003.

[87] Anna Vilanova and Eduard Gröller. Geometric modeling for virtual colon unfolding. In H. M G. Brunnett, B. Hamann, editor, *Geometric Modeling for Scientific Visualization*, pages 453–468. 2004.

[88] Vasily Volkov. *Adaptive local refinement and simplification for cloth simulation*. Master's Thesis, Nanyang Technological University, Singapore, 2003.

[89] S.K. Warfield, S.J. Haker, I.F. Talos, C.A. Kemper, N. Weisenfeld, A.U.J. Mewes, D. Goldberg-Zimring, K.H. Zou, C.F. Westin, W.M. Wells, C.M.C. Tempany, A. Golby, P.M. Black, F.A. Jolesz, and R. Kikinis. Capturing intraoperative deformations: research experience at brigham and women's hospital. *Medical Image Analysis*, 9(2):145–162, April 2005.

[90] Yongmin Zhong, Bijan Shirinzadeh, and Weiyin Ma. Solid modelling in a virtual reality environment. *The Visual Computer*, 21(1-2):17–40, 2005.